UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

Memoria Trabajo Fin de Grado

Escuela de Ingeniería Informática

Universidad de Las Palmas de Gran Canaria

Aplicación web para la gestión de tareas siguiendo la metodología GTD

Autor: José Aythami Ramírez Medina

Tutor: Javier Sánchez Pérez

Las Palmas de Gran Canaria

Diciembre 2013

Agradecimientos

Quisiera dar las gracias a todas aquellas personas que ha contribuido y propiciado que este proyecto haya salido adelante.

En primer lugar a mi familia por estar siempre apoyándome y confiar en mí en todo momento.

A mi tutor Javier Sánchez Pérez, por su inestimable ayuda, sin el cual este proyecto no hubiera sido posible .

A mis amigos que han brindado su apoyo y ánimos y sin los que no estaría en donde estoy. Me gustaría dar un agradecimiento especial a mi amigo John Wu Wu, el cual me ha apoyado en todo momento a lo largo de este proyecto.

¡A todos, gracias!

Resumen

El presente trabajo final de grado tiene por finalidad ofrecer una solución que ayude a las personas a gestionar sus tareas tanto personales como empresariales de una manera más productiva.

Actualmente este tipo de aplicaciones tienen mucho éxito. Se decidió que el desarrollo de esta aplicación fuera con la metodología Getting Things Done (GTD), ya que es una metodología que aumenta la productividad y reduce el estrés laboral. A día de hoy, no hay muchas aplicaciones que utilicen esta metodología, y las que las utilizan lo hace de una forma muy básica.

Junto a esta metodología y guiándonos de la experiencia del tutor se intentó combinar esta metodología con controles de tiempo para mejorar aún más la productividad de las personas que utiliza dicho software.

El resultado obtenido de este trabajo final de grado fue la base de una aplicación web para la gestión de tareas. El software creado es totalmente funcional, muy fácil de usar, muy intuitivo, y usa la filosofía Getting Things Done. Básicamente los objetivos principales conseguidos en este proyecto fueron:

- La gestión de usuarios
- La gestión de tareas y proyectos
- Aplicación de la metodología GTD
- Control del tiempo productivo, e improductivo, interrupciones, temporizadores.

La aplicación ha sido realizada como trabajo final de grado en ingeniería informática, cumpliendo con todas las fases del desarrollo del software; para obtener un producto funcional que fuera aprobado por el tutor que haría el rol de potencial cliente.

En el presente proyecto se ha seguido la metodología RUP, dirigida por casos de uso, iterativa e incremental. Para completar el proceso se ha realizado la elaboración de una lista de características, la especificación de los casos de uso, una fase de análisis, una de diseño, implementación y prueba. Las tecnologías utilizadas han sido, principalmente, Ruby On Rails, HTML5, CSS , AJAX y JAVASCRIPT.

El objetivo a largo plazo es que esta solución pueda ser tomada como base de implementación, donde haciendo las mejoras necesarias se pueda poner en el mercado un gran software de gestión de tareas siguiendo la metodología GTD.

Abstract

This final work degree is intended to provide a solution that helps people manage their personal and business tasks in a more productive way.

Currently these applications are very successful. It was decided to develope this application using the Getting Things Done (GTD) methodology that increases productivity and reduces stress.

Today, there are not many applications that use this methodology and the ones which use it, only use it at very basic levels. Along with this methodology and the Project Tutor's experience and guidance, we aim to combine this methodology with time controls to enhance the productivity of people who use this software even further.

As a result of this final work degree we obtained the basis for a web-based application for managing tasks. The software created is fully functional, easy to use, very intuitive and uses the Getting Things Done philosophy.

Basically the main objectives achieved in this project are:

- User management
- The management of tasks and projects
- Application of the GTD methodology
- Control of productive and unproductive time, interruptions and timers.

The application has been made as a final work degree in computer engineering, accomplishing with all the phases of software development, in order to get a functional product approved by the tutor who would act as a potential customer.

In this project, the methodology followed was RUP, a case driven, iterative and incremental methodology. We have elaborated a list of features, a specification of use cases, an analysis phase, a design, implementation and testing in order to fully complete the process. The technologies used are, Ruby On Rails, HTML5, CSS, AJAX and Javascript.

The long term objective is that this solution can be taken as a basis for future implementations, in which after providing the necessary improvements, can on the market a great task management software following the GTD methodology.

"El viaje más largo comienza con un primer paso"

Proverbio chino

"*Yo antes pesaba que los grandes proyectos los hacían las grandes personas, y ahora me doy cuenta que los grandes proyectos hacen a las personas grandes*

Anónimo

Contenido

Capítulo 1: Introducción

Actualmente vivimos en una sociedad cada vez más estresada donde tenemos muy poco tiempo, y por ello, es fundamental aprender a utilizarlo. La sociedad actualmente vive con mucho estrés. Las prisas nos acompañan siempre desde que nos levantamos y en muchas ocasiones no sabemos priorizar la importancia de las cosas. El estrés se origina por el ritmo frenético en que se vive. La gente se estresa simplemente porque no conocen una manera más adecuada o más efectiva de hacer las cosas. Tenemos que buscar formas de mejorar nuestra productividad tanto a nivel personal como a nivel empresarial.

Muchas personas para planificar su día necesitan dos herramientas básicas: un calendario o agenda y un gestor de tareas. Casi todo el mundo ya utiliza una agenda, pero se han encontrado a muchas personas que no tienen un buen gestor de listas. Es una filosofía poco usada por la mayoría, pero de gran utilidad, muchas de las personas todavía no tienen ninguna herramientas para gestionar sus tareas. Simplemente porque no conocen los ***"principio de la productividad personal".***

David Allen en su libro ***"Organízate con eficacia"*** nos intenta demostrar que existe un sistema de organización del trabajo que nos permite liberar la mente de las tensiones que inhiben nuestra creatividad, y que nos hacen más eficaces en todos los aspectos de la vida. El sistema propuesto por Allen soluciona ansiedades y desconciertos, y nos permiten transformar nuestro modo de trabajar y la manera de percibir nuestros retos cotidianos. Esta filosofía de vida tanto social como empresarial puede resultar en un principio caótica pero se ha demostrado en muchos estudios anteriores que realmente es una metodología que aumenta la productividad y reduce el estrés laboral.

Para conseguir funcionar estos principios necesitamos herramientas que nos ayuden a conseguir estos resultados mostrados anteriormente. Para ello nos ayudaremos de herramientas para gestionar nuestras tareas y ser más eficaces en el día a día.

Simplemente una gestión de tareas se puede llevar a cabo con un bolígrafo y un papel en donde se pondrían las tareas que tenemos que hacer, digamos que este método es el más simple.

Este método es tan efectivo como cualquier otro, pero intentaremos conseguir esta metodología usando un software adaptado para ello, pues consideramos que de esta forma es un método quizás más efectivo, limpio y más rápido.

Una aplicación para gestionar tareas sólo debe cumplir dos requerimientos principales:

1. Facilitar la creación y finalización de tareas
2. La posibilidad de crear varias listas para los contextos

Se ha decidido que en este trabajo final de grado se creará nuestro propio software de gestión de tareas, nuestro software de gestión de tareas seguirá la metodología presentada anteriormente cuando citamos David Allen, dicha metodología se llama ***Getting Things Done (GTD).***

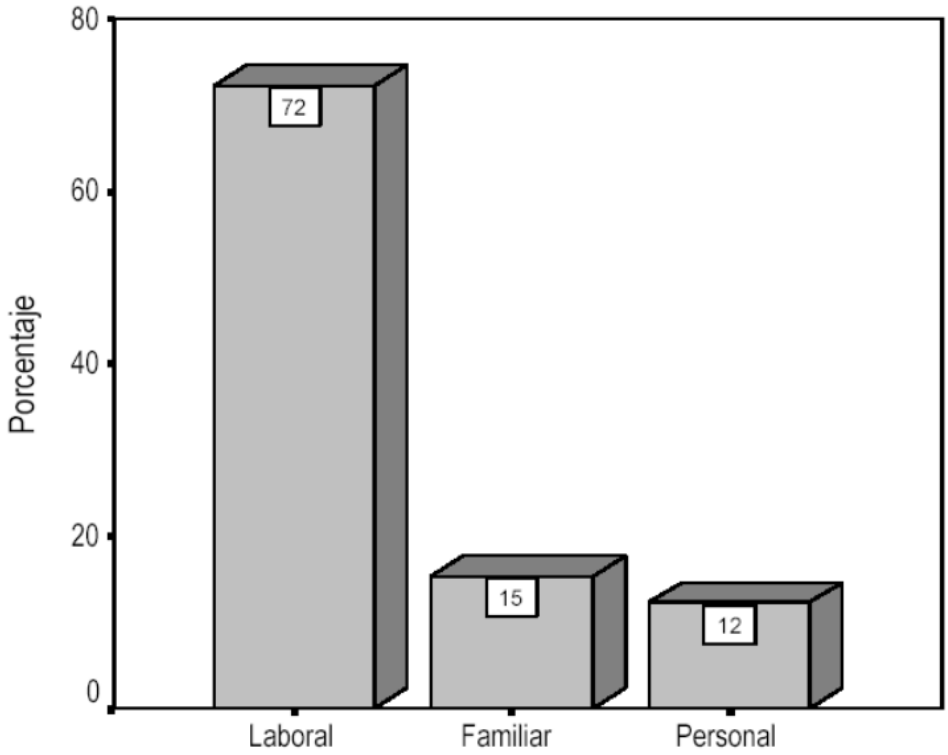
1.1 Aportaciones

Como se había comentado brevemente en la introducción de este trabajo final de grado, actualmente vivimos en una sociedad en donde cada vez las personas están más estresadas tanto a nivel personal como profesional.

Desde la entrada en vigor de la Ley de Prevención de Riesgos Laborales, en 1995, se ha dado un impulso a los aspectos relacionados con la Salud Laboral, entre los factores desencadenantes de distintos problemas de salud, deterioro de las relaciones interpersonales, absentismo y disminución de la productividad, se encuentra el **estrés**.

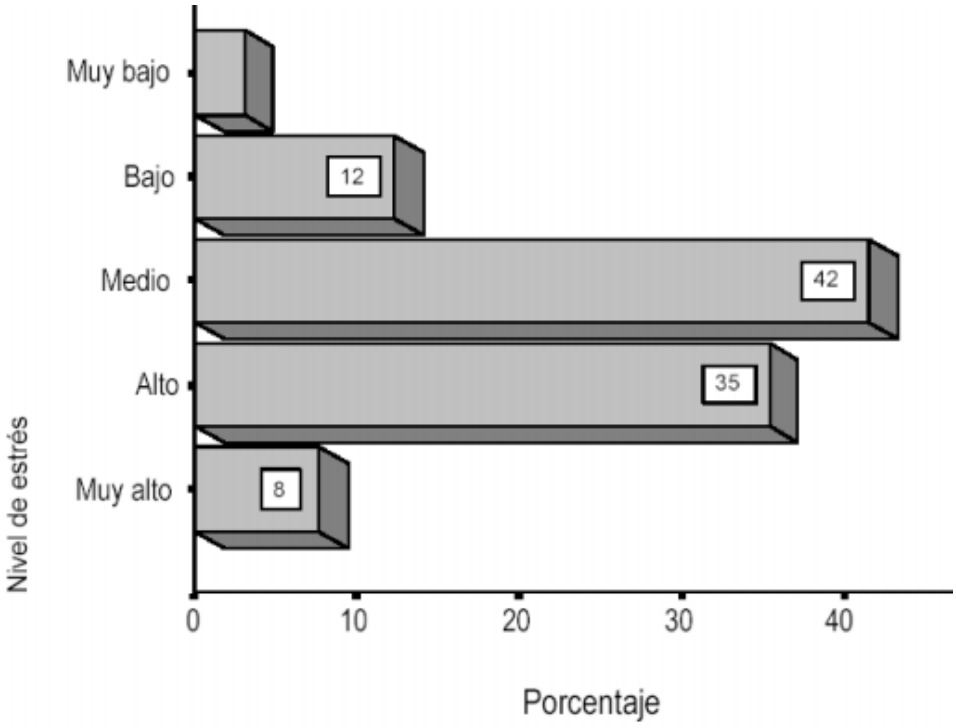
La Comisión Europea, a través de la Fundación Europea para la Mejora de las Condiciones de Vida y Trabajo ha realizado un estudio sobre el estrés laboral en el que concluye que el 28% de los trabajadores europeos padece estrés y el 20% se sienten "quemados" en su trabajo,siendo los sectores más afectados los trabajos manuales especializados, el transporte, la restauración y la metalurgia.

Los altos costes personales y sociales generados por el estrés laboral, han dado lugar a que organizaciones internacionales como la Unión Europea y la OMS insistan cada vez más en la importancia que tienen la prevención y el control del estrés en el ámbito laboral.



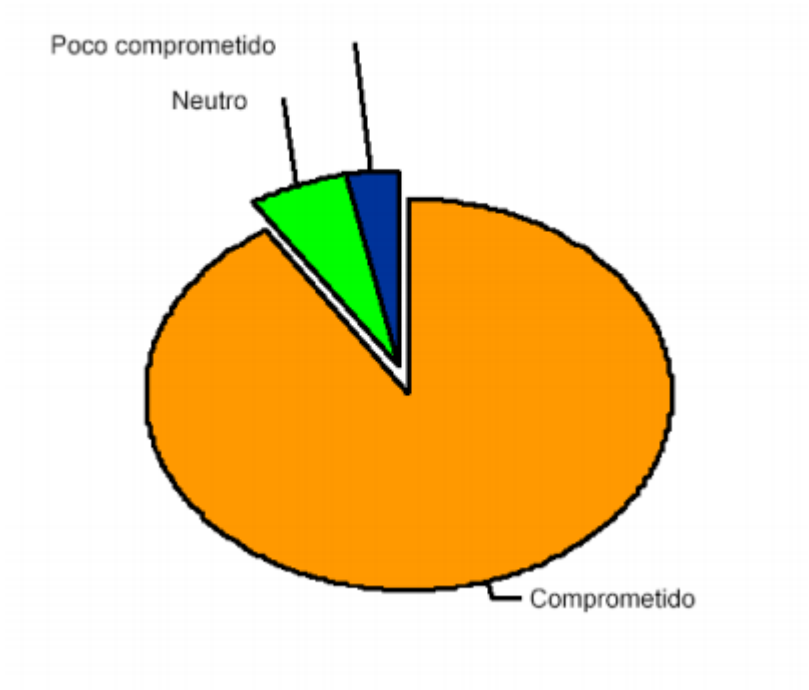
Distribución de porcentajes del área que más contribuye al estrés

Sin nos enfocamos en el sector laboral, podemos ver en la siguiente gráfica la distribución del porcentaje del estrés laboral.



Distribución en función de grado de compromiso

El 91,4 % está comprometido con los objetivos y finalidades de la orientación.



El estrés, además de producir ansiedad, puede producir enfado o ira, irritabilidad, tristeza, depresión, y otras reacciones emocionales, que también podemos reconocer.

Pero además de estas reacciones emocionales podemos identificar claramente otros síntomas producidos por el estrés, como son el agotamiento físico, la falta de rendimiento,

etc. Si el estrés es muy intenso y se prolonga en el tiempo, puede llegar a producir enfermedades físicas y desórdenes mentales, en definitiva problemas de salud.

Como se ha observado en las gráficas anteriores, actualmente el estrés supone un problema bastante serio, hay que intentar buscar medidas que ayuden o minimicen los efectos.

Este estudio fue el punto de partida para buscar herramientas que ayuden a las personas a quitarse de encima algo de estrés y para ello se ha pensado en la filosofía GTD, que es una filosofía que ayuda a reducir el estrés laboral y personal.

Se pensó que si se realiza una herramienta de gestión de tareas personales y laborales usando esta filosofía, se puede reducir en gran medida algo de estrés existente en la actualidad, y de esta forma ayudar positivamente a la sociedad.

Por otro lado supondrá un impacto económico bastante grande, ya que este tipo de aplicaciones de gestión de tareas actualmente son de gran interés por las personas que conocen las ventajas de la productividad personal.

Para finalizar existiría un impacto tecnológico, ya que actualmente no existe mucho software de este tipo que utilice la metodología GTD.

1.2 Motivación

La motivación que determinó la realización de este proyecto final de grado tiene su origen en mi tutor Javier Sánchez Pérez, que me propuso realizar un trabajo final de grado en la creación de un software de "***gestión de tareas siguiendo la metodología GTD"***, ya que actualmente este tipo de aplicaciones tiene mucho éxito.

Junto a esta metodología y guiándonos de la experiencia que el tutor de este proyecto tiene en relación con software de este tipo, se intentará combinar esta metodología con controles de tiempos, para mejorar aún más la productividad de las personas que utilizan dicho software.

Por otro lado, la evolución de las nuevas tecnologías y la aparición de nuevas plataformas de desarrollo ha proporcionado la motivación necesaria para indagar en el conocimiento de lenguajes de programación distintos a los usados durante la carrera. Las tendencias actuales del mercado, me llevó a querer conocer el entorno Ruby On Rails y a estudiar la posibilidad de aplicarlo en futuros proyectos.

Así pues, la realización del proyecto final de grado se presentó como una oportunidad inmejorable de ampliar y profundizar en el proceso de especificación y diseño de un sistema. En dicho proceso se reflejará los artefactos usados en las fases que componen el ciclo de vida del software, implantando Ruby On Rails como tecnología de desarrollo.

1.3 Objetivos

El presente trabajo final de grado cuyo título es "***Aplicación web para la gestión de tareas siguiendo la metodología GTD***" pretende establecer una guía y base sólida que sustente la implementación de un sistema software.

Este proyecto se inicia partiendo de una idea concebida tanto de mi tutor de proyecto como mía. Tras un estudio previo viendo software similares al que se desea realizar, y documentando nuestras ideas preconcebidas, decidimos que éste fuera mi trabajo final de grado.

Dada la naturaleza de un trabajo final de grado, se pretende que el proyecto supere la frontera del análisis y diseño establecida por la metodología y se adentre en la implementación y construcción del sistema.

Para poder realizar esta serie de propósitos, se establecen a modo de control ciertos objetivos de diferentes índoles. Los objetivos generales que se desea conseguir son los que aparecen en la siguiente lista:

- Mostrar las competencias adquiridas durante el proceso de formación del alumno. El proyecto final de grado sirve en cierta medida como validador de la capacidad del alumno para enfrentarse a un trabajo real del mercado laboral. Esta afirmación anterior conlleva la rápida asimilación de una tecnología y una constante evolución de las técnicas de diseño y programación.
- En el proyecto se han tenido que agregar conocimientos adquiridos durante la titulación como:
    - Base de datos
    - Programación
    - Ingeniería del software
    - Diseño de interfaces
    - Otras....

- El desarrollo del software es un proceso de aprendizaje continuo en donde debemos ser capaces de adaptarnos con facilidad.
- Aprender adaptarse ante los cambios que surgen de manera imprevistas durante el desarrollo del software.
- Potenciar el trabajo del equipo, aunque este proyecto es individual hay que hablar con el cliente para saber que se desea, en esta ocasión el cliente es mi tutor de proyecto, así que este proyecto es un trabajo de ambos.
- Aprender adaptarnos a metodologías ágil de desarrollo, documentando cada fase del proceso en cada momento.

Los objetivos específicos que se desea conseguir son los que aparecen en la siguiente lista:

- Gestión de acceso de usuarios al sistema. Se controla la fase de registro y autenticación al sistema.
- Gestión de tareas. Se dará posibilidad al usuario de que pueda crear y gestionar sus tareas.
- Gestión de proyectos. Los usuarios del sistema podrán crear proyectos asociados a tareas o a otros proyectos.

- Gestión de interrupciones. Los usuarios subscritos al sistema podrán interrumpir y coordinar las interrupciones producidas cuando tienen una tarea en marcha.
- Gestión GTD. Las tareas serán gestionadas siguiendo la filosofía GTD.
- Construcción de una aplicación web mediante el uso de tecnologías como ruby and rails
- Utilización de sistema de control de versiones

1.4 Gestión de Tareas

Como se ha mencionado anteriormente el presente proyecto es un software de gestión de tareas en donde se intentará organizar el trabajo tal y como describe David Allen el creador de Getting Things Done (GTD).

David Allen en su libro *"**Organízate con eficacia**"* nos intenta demostrar que existe un sistema de organización del trabajo que nos permite liberar la mente de las tensiones que inhiben nuestra creatividad, y que nos hace más eficaces en todos los aspectos de la vida.

David Allen sostiene que nuestra mente tiene una capacidad limitada para almacenar información y propone una serie de fórmulas prácticas para eliminar las tensiones e incrementar nuestra capacidad de trabajo y nuestro rendimiento.

Organízate con eficacia se fundamenta en unas sencillas normas básicas de organización del tiempo, como por ejemplo la necesidad de determinar cuál es el siguiente paso a dar en cada uno de nuestros proyectos, o la regla de los dos minutos (si surge una tarea pendiente y se puede hacer en menos de dos minutos, debe hacerse inmediatamente).

El sistema propuesto por Allen soluciona ansiedades y desconciertos, y nos permite transformar nuestro modo de trabajar y la manera de percibir nuestros retos cotidianos. Esta filosofía de vida tanto social como empresarial puede resultar en un principio caótica pero se ha demostrado en muchos estudios anteriores que realmente es una metodología que aumenta la productividad y reduce el estrés laboral.

Por eso, y teniendo presente estas afirmaciones anteriores el presente trabajo final de grado usará esta increíble metodología de organización.

La explicación de la metodología en sí se explicará en futuros apartados, el apartado concreto será el **5.1.1**

1.5 Estructura del Documento

El presente proyecto está bastante orientado a la rama de ingeniería del software en donde en cada capítulo incluye un marco teórico que respalda las diferentes secciones del documento.

Se recomienda al lector que siga el flujo normal del documento. Antes de comenzar a leer el capítulo de implementación es requisito imprescindible haber leído los capítulos de tecnologías y herramientas usadas.

La presente documentación está estructura de la siguiente manera:

- **Estado del arte.** En este capítulo se hará un estudio sobre los diferentes software encontrados similares a los de este trabajo final de grado. Intentaremos ver las características principales que ofrecen estos software y se buscarán las diferentes características que aportamos frente a los software actuales.

- **Planificación de trabajo:** En este capítulo se realizarán diferentes funcionalidades. En primer lugar, hablaremos sobre metodología seleccionada. En esta sección también incluirá el plan de trabajo usado, y analizaremos el presupuesto necesario para crear el trabajo final de grado.

- **Herramientas y Tecnologías:** En este capítulo se detallará las herramientas físicas y las tecnologías usadas para proceso de construcción de la aplicación propuesta.
- **Desarrollo de la aplicación:** En éste capítulo se describirá el proceso para la realización del trabajo final de grado. Empezaremos por conocer el dominio del problema y modelo de negocio. Una vez entendido el contexto de la aplicación se definirá cuál será la arquitectura de los requisitos más importantes. Tras este punto se hará el análisis de los casos de uso, así como de las operaciones que se realizaran.

  Continuaremos con el diseño creando los diferentes modelos de la arquitectura del sistema:

  - diagrama de la arquitectura
  - diagrama de clases
  - diagrama de paquetes
  - modelo de despliegue
  - diseño de la base de datos

  Tras el diseño de la arquitectura lo implementaremos usando las tecnologías mencionadas, y realizaremos diferentes técnicas usadas para evaluar el software implementado.

- **Anexos:** En este Capítulo se detallará diferentes apartados para completar la memoria. Algunos de los apartados que nos encontraremos van a ser un manual de usuario para aprender cómo funciona el software. Analizaremos las diferentes competencias cubiertas en este trabajo final de grado.

  También veremos más en profundidad un apartado sobre metodologías usadas a día de hoy y aprenderemos la información que afecta a la legislación vigente sobre proyectos informáticos similares al TFG.

Capítulo 2: Estado Actual del Arte

El presente capítulo intentará introducir al lector en un breve estudio sobre aplicaciones similares encontradas actualmente en el mercado referente a este trabajo final de grado.

Esta sección está organizada de la siguiente manera:

1. Listado de las aplicaciones encontradas en el mercado a día de hoy divididas en tres secciones diferentes:

    - Aplicaciones generales
    - Aplicaciones usando la filosofía GTD
    - Aplicaciones en dispositivos móviles

2. Introducción de diferentes características que intentaremos estudiar en cada una de las aplicaciones que vamos a exponer.
3. Estudio de algunas de las aplicaciones encontradas para las tres categorías mencionadas anteriormente.
4. Comparación de nuestra aplicación con las anteriores estudiadas, veremos que características diferentes aportamos en comparación a las demás aplicaciones del mismo estilo.

Para este estudio se han seleccionado una serie de 43 aplicaciones diferentes encontradas actualmente en el mercado. De las 43 listadas aquí analizaremos las tres primeras de cada bloque.

Nombre del software		
General	**GTD**	**Móvil**
1. **Pivotal tracker**	1. **Nirvana**	1. **Google Keep**
2. **Trello**	2. **Toodledo**	2. **Evernote**
3. **teambox**	3. **Things**	3. **Astrid**
4. Achievo	4. Famundo	4. Colornote
5. Clocking IT	5. Remember the milk	5. Ak notepad
6. Todoyu	6. GTDagenda	6. Catch notes
7. WebCollab	7. Mytodos	7. Wunderlist
8. EGroupWare	8. Nexty	8. Any.do
9. Redmine	9. Vitalist	9. Remember the milk
10. Trac	10. Todo.ly	10. Gtasks
11. Google Tasks	11. Teux deux	
12. Mantis	12. Todoist	
13. Dotproject	13. Toodledo	
14. Basecamp	14. Things	
15. ActiveCollab	15. Omnifocus	
16. Wunderlist		
17. Groupcamp projects		
18. Celoxis		
19. Harvest		
20. Mavanlink		

Las características que se intentará estudiar en cada una de las aplicaciones son las siguientes:

Planificación

Una de las tareas más comunes en el manejo de proyecto es la planificación de una serie de eventos. Algunas de las dificultades para la planificación de proyecto pueden ser:

- Eventos que dependen de la creación de otros eventos.
- Planear que las personas trabajen en las tareas requeridas.
- Asignar los recursos necesarios a las tareas.
- Manejo de las incertidumbres con las estimaciones de duración de ciertas tareas.
- Acomodar las tareas para cumplir con ciertos hitos.
- Manejar varios proyectos simultáneamente para cubrir los requerimientos.

Provisión de la información

Para poder justificar todo el tiempo que se emplea en utilizar el software de manejo de proyectos.

Los requerimientos típicos entre los programas más comunes son:

- Listas de tareas por persona.
- Listas de planificación de recursos.
- Información del tiempo que las tareas requerirán para su terminación.
- Advertencia temprana de posibles riesgos para el proyecto.
- Información de la carga de trabajo y los días feriados o vacaciones para los empleados.
- Información histórica de cómo han progresado proyectos similares anteriormente desarrollados.

Basado en el Web

El software de la administración de proyectos se puede poner en ejecución con una Aplicación Web. De esta forma se accede a través de una Intranet o de una Extranet usando un web browser.

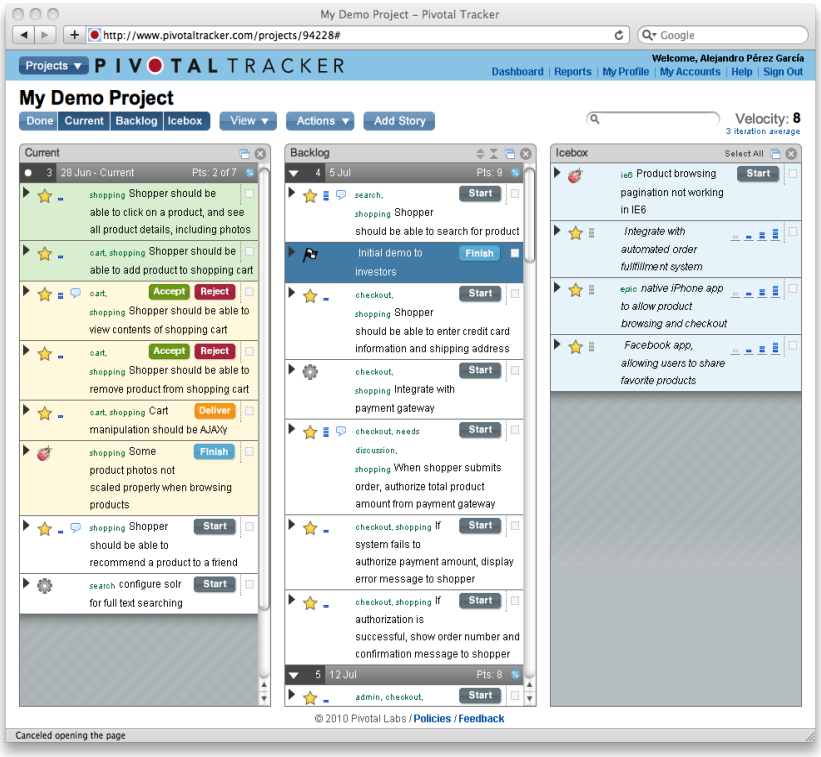
Esto tiene varias ventajas y desventajas de aplicaciones web:

- Se puede acceder desde cualquier tipo de computadora sin la instalación de software.
- Facilidad del control de acceso.
- Naturalmente multiusuario.
- Solamente una instalación/versión de software para mantener.
- Originalmente es más lento para responder que las aplicaciones de escritorio.
- Capacidad gráfica más limitada que las aplicaciones de escritorios.

A continuación, una vez estudiado los puntos que se intentará analizar en cada software. Se empezará a presentar cada software observando las características más interesantes de cada uno de ellos.

2.1 General

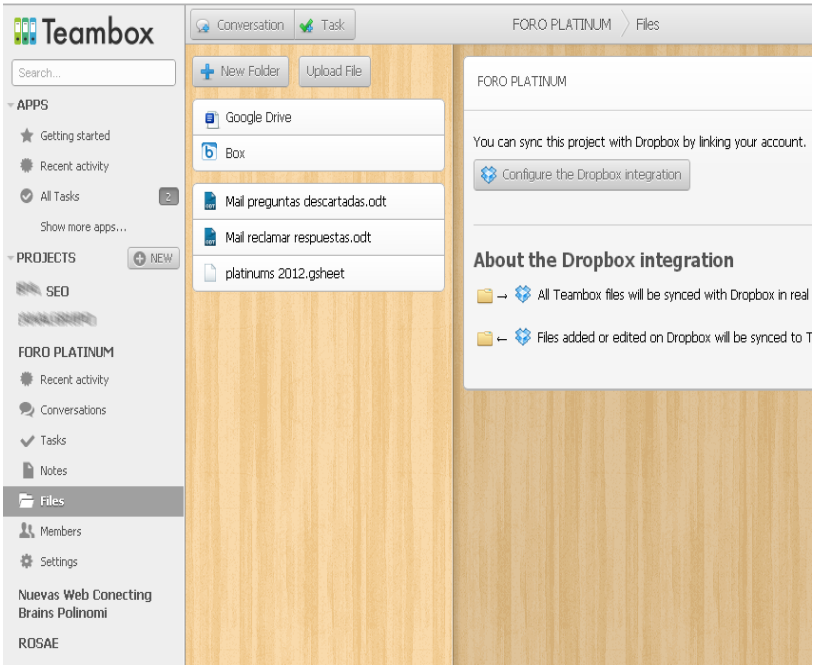
2.1.1 Pivotal Tracker



Es un gestor de proyectos con una fuerte orientación a metodologías ágiles. Este programa se ejecuta en el navegador. Las tareas no se estiman en horas, sino en puntos que determinan tras un tiempo la velocidad del equipo.

Se trata de un servicio gratuito, también en la nube, con posibilidad de crear múltiples proyectos y de asignarles personas con diversos roles. Las tareas se crean en la nevera (*the Icebox*) y se van organizando dinámicamente en función de parámetros variables, como los hitos definidos, el ritmo del equipo y las tareas pendientes.

2.1.2 Teambox

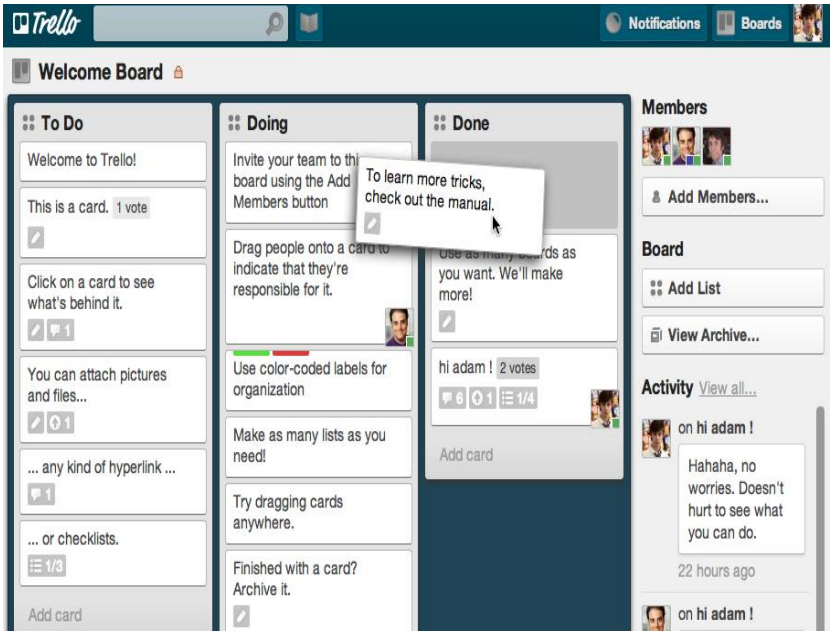


Es una plataforma segura de gestión de proyectos, que permite la colaboración entre equipos simplificando la comunicación en un entorno orientado a tareas. La comunicación dentro de la herramienta es totalmente virtual y social.

Teambox posee un chat potente y amigable, y listas de tareas para estructurar todos los proyectos, así como conversaciones que puedes convertir en acciones.Las notas te permiten compartir documentación con tu equipo como si se tratara de un Wiki, generando una mayor inmersión por parte de todo el equipo en el proyecto.

Este software dispone de un sistema de time tracking, permitiendo la gestión del tiempo de cada proyecto y tarea. Además permite la integración de archivos con Google Docs, Dropbox.., como si se tratara de una carpeta compartida.

2.1.3 Trello



Trello es un nuevo sistema creado para facilitar el trabajo y la colaboración entre miembros de un mismo grupo.

La idea es permitir que cualquier líder vea lo que está haciendo su equipo en un momento determinado, ofreciendo para ello un panel de información bastante completo e intuitivo.

Podemos asignar tareas a cualquier persona y usar el concepto de “tarjeta” para cada proyecto, que incluye las conversaciones, actividades, archivos adjuntos, actualizaciones, etc. Para incluir una persona en una tarjeta específica, simplemente tenemos que arrastrarla y soltarla en la deseada.

2.2 Getting Thing Done(GTD)

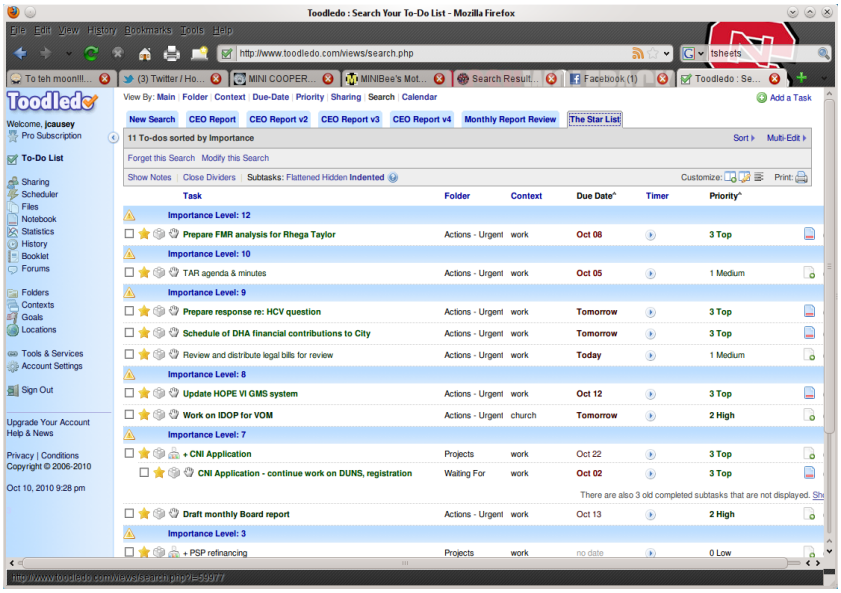
2.2.1 Nirvana

Nirvana es una sólida opción para la gestión de tareas gracias a que permite una manipulación súper sencilla de su interfaz (con un espléndido uso del drag and drop a la cabeza) y sus funciones completísimas.

Además de un diseño altamente intuitivo, Nirvana ofrece adición de tareas desde el correo electrónico, búsquedas rápidas, notificaciones externas, exportación, etc.

Nirvana es un software que se accede a través de un navegador web, una característica importante es su facilidad del control de acceso.

2.2.2 Toodledo



Toodledo es una de las herramientas de GTD, que he encontrado con más potencia, flexibilidad y compatibilidad. Es una herramienta vía Web, por tanto se usa desde tu propio navegador, no hay nada más portable que eso. Pero si queremos usar otras aplicaciones instaladas, su potente API permite sincronizar tus tareas con otras aplicaciones. Además su versión de móvil permite trabajar offline incluso desde el móvil.

Es un software muy flexibilidad podemos organizar las tareas de tantas maneras diferentes que Toodledo se adapta al usuario de forma que no tienes que adaptarte tu al programa, sino al revés.

2.2.3 Things

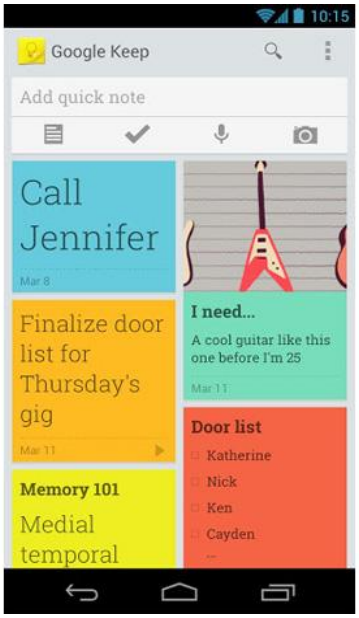
Things es una de las aplicaciones GTD más conocidas para el sistema operativo Mac y está enfocada a la gestión de tareas y proyectos. Dispone de diferentes carpetas donde guardar y alojar los asuntos pendientes en función de la organización prevista por el usuario. El buzón de entrada permite almacenar información que será necesaria utilizar más adelante, es una forma de "descargar información de la mente".

Para la gestión de tareas dispone de una pestaña de "proyectos activos" donde cada proyecto no es más que la agrupación de determinados asuntos relacionados.

Respecto a la gestión del tiempo**,** propone diferentes pestañas con las labores previstas para ese día, "hoy", así como las "siguientes" y las "programadas", es decir, trabajo concretos que no están asignadas a un proyecto determinado o bien que deben repetirse de forma cíclica.

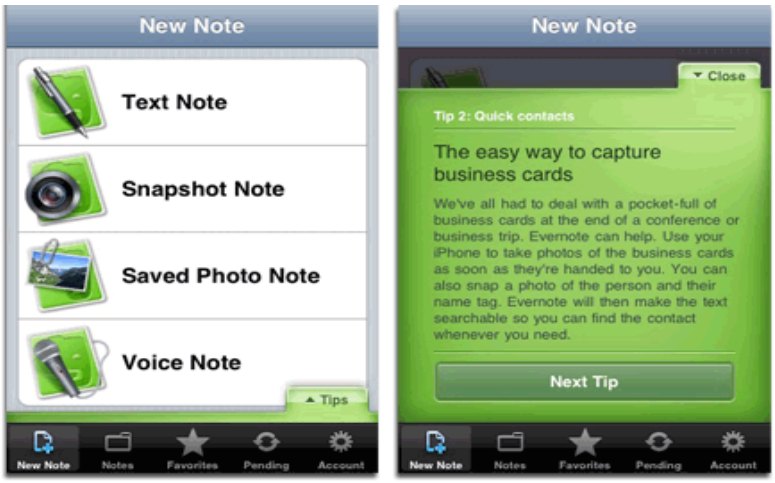
2.3 Móvil

2.3.1 Google Keep



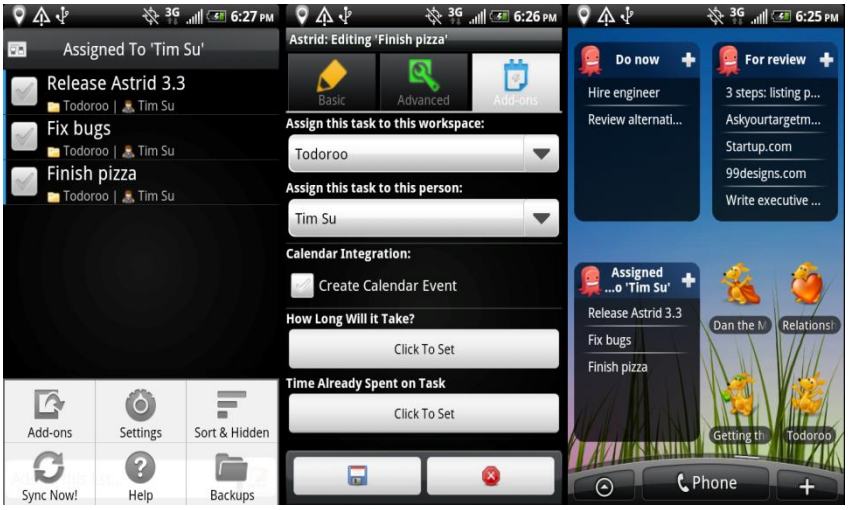
Google Keep destaca por la sencillez de su interfaz, a la vez que resulta agradable e intuitiva gracias a la utilización de colores. Google Keep te permite escribir y organizar tus notas con diferentes colores y tamaños de letra, y puedes crear tanto notas escritas, como de voz o con imágenes**.** Dispone de widgets de escritorio para facilitar el acceso directo a los archivos desde la pantalla de inicio que estarán además disponibles desde cualquier dispositivo u ordenador gracias al almacenamiento en la nube.

2.3.2 Evernote



Evernote es posiblemente el gestor de notas más conocido, descargado y completo de la red**.** Te ayuda a mejorar tu productividad manteniendo ordenadas tus notas, ya sean escritas, de imagen o de voz, a las que podrás acceder a través de tu cuenta en la nube. Como opciones avanzadas, con Evernote tienes la posibilidad de compartir tus archivos con terceros mediante Facebook, Twitter o correo electrónico, o por ejemplo dispone de una forma de búsqueda avanzada a través de textos que aparezcan en fotografías**.**

2.3.3 Astrid



Astrid, es de gran ayuda si necesitas un poco de organización en tu agenda**.** Podrás crear tareas independientes y listas de tareas, creando recordatorios para que no se escapen las fechas importantes. Dispone de un paquete avanzado que te puedes descargar de la web, con el que podrás añadir tareas por voz, o conseguir más widgets de escritorio.

2.4 Comparativa

Una vez leído los 9 ejemplos de software diferentes que hay en el mercado y habiendo estudiado sus características, es la hora de reflexionar y ver que aporta mi aplicación diferente a las demás.

Para comenzar, en la mayoría de estos ejemplos no usan la metodología GTD y los que sí la utilizan a excepciones de un par, el uso de GTD es muy básico. En mi aplicación se intentará que el uso de la metodología GTD no sea básica, se intentará que éste lo más integrada a esta filosofía. Lo que se busca es que sea el software más integrado a GTD de los software que se usa hoy en el mercado.

Una de las características que se intentará tratar es lo que llamamos “planificación”. Estas características la hemos visto anteriormente en el principio de este documento, al igual que las dificultades para la planificación. Todas estas dificultades se resuelven siguiendo la metodología GTD. Por ejemplo la dificultad **“**Eventos que dependen de la creación de otros eventos” GTD la trata en una carpeta especial de espera de eventos.

La siguiente característica era la provisión de información, aquí mi aplicación dispondrá de muchas de las funcionalidades que tiene muchas de las aplicaciones que hemos visto anteriormente.

Algunas de estas funcionalidades son:

- Listas de tareas.
- Lista de proyectos
- Información del tiempo que las tareas requerirán para su terminación.
- Advertencias tempranas de posibles riesgos para el proyecto.
- Información histórica de proyectos anteriores
- Etc…

Otra funcionalidad propia que se le quiere dar a este software que quizás difiera de los otros programas sea control de tiempo productivo e improductivo. Con lo del tiempo improductivo me refiero a cuánto tiempo a estado interrumpido para una tarea comparándolo con su tiempo real en producción. Todos estos controles de tiempo intentaremos realizarlos en gráficas que muestre al usuario lo productivo que ha sido ante una tarea o proyecto y viceversa.

Finalmente, como todo software visto anteriormente, mi proyecto estará basado en una aplicación web. Este hecho anterior implica una serie de ventajas e inconveniente de realizar una aplicación para el navegador. Estas ventajas e inconveniente las habíamos visto al comienzo de este documento en la sección “Basado en el web”.

Capítulo 3: Planificación del Trabajo

3.1. Metodología de Desarrollo

En este capítulo es donde se justificará la metodología seguida en la definición e implementación del proyecto.

En las dos últimas décadas las notaciones de modelado y posteriormente las herramientas pretendieron ser las *"balas de plata"* para el éxito en el desarrollo de software, sin embargo, las expectativas no fueron satisfechas. Esto se debe en gran parte a que la metodología de desarrollo, había sido postergada. De nada sirven buenas notaciones y herramientas si no se proveen directivas para su aplicación.

Así pues , ha comenzado un creciente interés en las metodologías de desarrollo. Hasta hace poco el proceso de desarrollo llevaba asociada un marcado énfasis en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada. Este esquema "**tradicional**" para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de ceremonia en el proceso.

Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Ante las dificultades para utilizar metodologías tradicionales con estas restricciones de tiempo y flexibilidad, muchos equipos de desarrollo se resignan a prescindir del *"buen hacer"* de la ingeniería del software, asumiendo el riesgo que ello conlleva.

En este escenario, las **metodologías ágiles** emergen como una posible respuesta para llenar ese vacío metodológico. Por estar especialmente orientadas para proyectos pequeños, las metodologías ágiles constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto.

La necesidad de un proceso promete hacerse más crítica, especialmente en empresa u organizaciones en las cuales los sistemas software son esenciales, tal como las financieras, las de control de tráfico, etc.

Con esto queremos decir que la dirección con éxito del negocio o la ejecución de la misión pública depende del software que la soporta. Estos sistemas software se hace más complejos, su tiempo de salida al mercado necesita reducirse, y su desarrollo por tanto, se hace más difícil.

Por razones como éstas, la industria del software necesita un proceso para guiar a los desarrolladores, al igual que una orquesta necesita una partitura de un compositor para dirigir el concierto.

3.1.1 ¿Qué es un Proceso de Desarrollo de Software?

Un proceso define quien está haciendo qué, cuándo y cómo alcanzar un determinado objetivo. En la ingeniería del software el objetivo es construir un producto software o mejorar uno existente .Un proceso efectivo proporciona normas para el desarrollo eficiente de software de calidad. Captura y presenta las mejores prácticas que el estado actual de la tecnología permite. En consecuencia. reduce el riesgo y hace el proyecto más predecible. El efecto global es el fomento de una visión y una cultura comunes.

Es necesario un proceso que sirva como guía para todos los participantes *(clientes, usuarios, desarrolladores y directores ejecutivos)*. No nos sirve ningún proceso antiguo; necesitamos uno que sea el mejor proceso que la industria pueda reunir en este punto de su historia.

Por último necesitamos un proceso que esté ampliamente disponible de forma que todos los interesados puedan comprender su papel en el desarrollo en que se encuentra implicados.

Un proceso de desarrollo del software debería también ser capaz de evolucionar durante muchos años. Durante esta evolución debería limitar su alcance, en un momento del tiempo dado a las realidades que permite las tecnologías, herramientas, personas y patrones de organización.

- **Tecnologías**. El proceso debe construirse sobre las tecnologías disponibles en el momento en que se va a emplear el proceso.
- **Herramientas**. Los procesos y las herramientas deben desarrollarse en paralelos. Las herramientas son esenciales en el proceso. Dicho de otra forma, un proceso ampliamente utilizado puede soportar la inversión necesaria para crear las herramientas que lo soporten.
- **Personas**. Un creador del proceso debe limitar el conjunto de habilidades necesarias para trabajar en el proceso a las habilidades que los desarrolladores actuales poseen, o apuntar aquellas que los desarrolladores puedan obtener rápidamente. Hoy es posible empotrar en herramientas software técnicas que antes requerían amplios conocimientos, como la comprobación de la consistencia en los diagrama del modelo.
- **Patrones de organización**. Aunque los desarrolladores de software no pueden ser expertos tan independientes como los músicos de una orquesta, están muy lejos de los trabajadores autómatas. El creador del proceso debe adaptar el proceso a las realidades del momento.

Los ingenieros del proceso deben equilibrar estos cuatros conjuntos de circunstancias. Además el equilibrio debe está presente no sólo ahora, sino también en el futuro. El creador del proceso debe diseñar el proceso de forma que pueda evolucionar, de igual forma que el desarrollador de software intenta desarrollar un sistema que no sólo funciona este año, sino que evoluciona con éxito en los años venideros.

Un proceso debe madurar durante varios años antes de alcanzar el nivel de estabilidad y madurez que le permitirá resistir a los rigores del desarrollo de productos comerciales, manteniendo a la vez un nivel razonable de riesgo en su utilización. El desarrollo de un producto nuevo es bastante arriesgado en sí mismo como para añadirle el riesgo de un proceso que esté poco validado por la experiencia de uso.

En estas circunstancias, un proceso puede ser estable. Sin este equilibrio de tecnologías, herramientas, personas y organización, el uso del proceso sería bastante arriesgado.

Las secciones siguientes se abordarán los grupos metodológicos, se hará un balance de lo que nos puede aportar, y se decidirá por la que será la metodología elegida para nuestro trabajo final de grado.

3.1.2 Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo de Software (PUD) es una metodología propuesta por Ivar Jacobson, Grady Booch y James Rumbaugh para realizar desarrollo de software.

Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software.

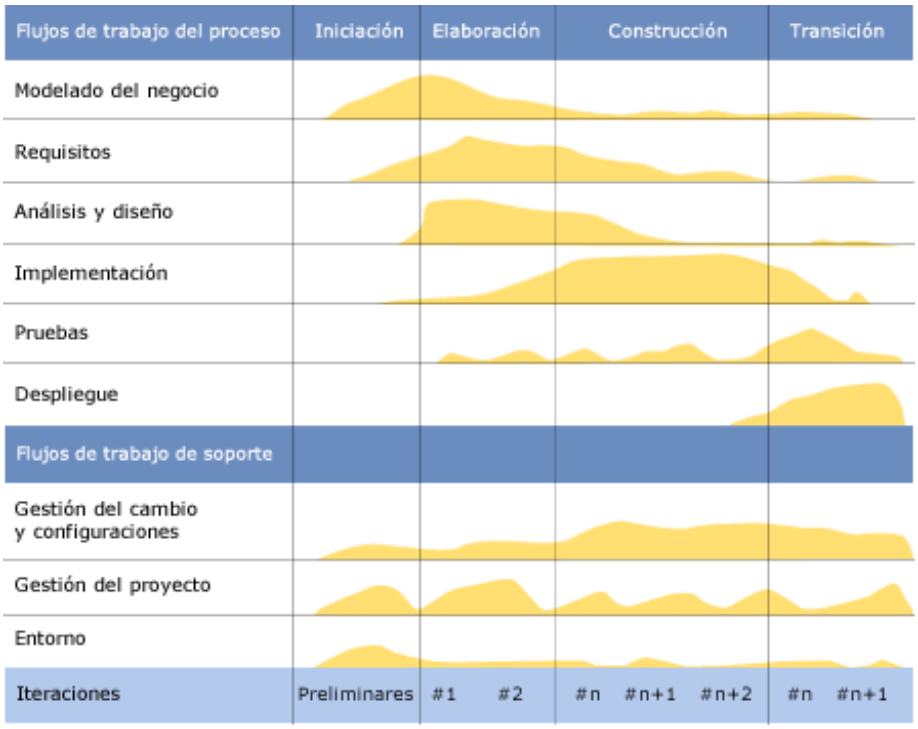
El Proceso Unificado de Desarrollo Software es un marco de desarrollo de software que se caracteriza por:

- ***Estar dirigido por casos de uso:*** Está dirigido por casos de uso, porque con éstos se especifican las funcionalidades que el sistema proporciona al usuario, es decir, se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc.
- ***Centrado en la arquitectura:*** Asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La analogía con la construcción es clara, cuando construyes un edificio existen diversos planos que incluyen los distintos servicios del mismo: electricidad, fontanería, etc.
- ***Es iterativo e incremental:*** Es un marco de desarrollo compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio puede incluir varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un *incremento* del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo. Cada una de estas iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: Análisis de requisitos, Diseño, Implementación y Prueba. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

3.1.2.1 Fases del Proceso

Por lo tanto, como se ha mencionado anteriormente el marco de desarrollo del proceso está compuesto de 4 fases que son: **inicio, elaboración, construcción y transición.**

- La etapa de **inicio** tiene como objetivo determinar la visión del proyecto y definir lo que se desea realizar.
- La etapa de **elaboración** es dónde se determina la arquitectura óptima del proyecto.
- Las etapas de **construcción y transición** tratan de obtener la capacidad operacional inicial y obtener el producto acabado.



La figura 3.1 muestra las fases en el proceso de desarrollo unificado. A la izquierda de la misma aparecen las disciplinas o etapas a realizar durante el proceso de creación del software.

Los objetivos de cada una de las disciplinas son:

- **Modelado de negocio.** Analizar y entender las necesidades del negocio para el cuál se está desarrollando el software.
- **Requisitos.** Proveer una base para estimar los costos y el tiempo de desarrollo del sistema.
- **Análisis y diseño.** Trasladar los requisitos analizados anteriormente a un sistema automatizado y desarrollar una arquitectura para el sistema.
- **Implementación.** Crear software que se ajuste a la arquitectura diseñada y que tenga el comportamiento deseado.
- **Pruebas o test.** Asegurarse de que el comportamiento requerido es correcto y que todo lo solicitado está presente.
- **Instalación o despliegue.** Producir distribuciones del producto y distribuirlo a los usuarios.

- **Disciplina de soporte.** Determinan las fases de control que son necesarias realizar durante el proyecto**.**
- **Configuración y administración del cambio.** Analizar todas las versiones del proyecto.
- **Gestión del proyecto.** Planifica los recursos que se deben emplear.
- **Entorno o ambiente.** Controlar todo el entorno que rodea a la producción de software

3.1.2.2 Ventajas y Desventajas del Rup

A continuación se listan las ventajas y desventajas más relevantes en el Proceso Unificado de Desarrollo Software.

Ventajas	Desventajas
Evaluación en cada fase que permite cambios de objetivos	La evaluación de riesgos es compleja
Iterativo e incremental	Solo existen problemas de comunicación entre el ingeniero del software y el usuario
Implementa las mejores prácticas de la ingeriería del software	Excesiva flexibilidad para algunos proyectos
Dirigido por caso de uso	Se pone al cliente en una situación que puede ser incómoda para él
Se reduce el riesgo y se tiene versiones operativas desde etapas tempranas.	Cliente debe ser capaz de describir y entender a un gran nivel de detalle.
Basado en la arquitectura	

3.1.2.3. Elección de la Metodología

En la actualidad, no hay una metodología universal para el desarrollo de software, sino un conjunto de metodologías en donde tienes que escoger la que mejor se adapte al sistema que quieres realizar.

Antes de comenzar hay que aclarar que PUD aunque pueda aparentar ser una metodología ágil y muchas personas lo consideran ágil, ya que es una metodología de carácter adaptativo, iterativo, incremental y unificada no lo es.

PUD no es una metodología ágil porque el trabajo que define implica liberaciones muy tardías en entornos de producción. Aunque PUD puede proporcionar feedback en todas las etapas, la respuesta al cambio no es ágil, ya que PUD sigue un marco de trabajo donde se le da gran importancia al proceso de desarrollo. Si los cambios se piden en la fase de transición, estos cambios se harían sobre la versión del producto en la que se ha trabajado durante el ciclo de vida, por lo que el esfuerzo necesario para hacer las modificaciones es mayor, y en consecuencia, el tiempo que tarda el usuario en poder disfrutar de ellos en el entorno de producción también es superior.

Una vez aclarado este punto, la elección de la metodología planteada está en si elegimos una metodología tradicional o una metodología ágil.

Como hemos observado en capítulos anteriores la metodología tradicional presenta cierta dificultades a la hora de realizar un proyecto, como por ejemplo:

- ✓ Fases previas pueden ser muy costosas dada la especificación de requisitos, análisis y diseño.
- ✓ Perdida de flexibilidad ante cambios
- ✓ Gran importancia en la documentación
- ✓ Desarrollo más lento

Por otra lado la metodologías ágiles:

- ✓ Es especialmente definidos para proyectos con requisitos poco definidos o cambiantes
- ✓ usado para equipos pequeños

Dada la naturaleza de este proyecto final de grado y sopesando ambas ramas metodológicas se decidió que la metodologías escogida sea ***PUD (proceso unificado de desarrollo)***.

Los principales motivos de la elección han sido:

- ✓ Que sea un proceso guiado por la arquitectura (especificación de casos de uso) y trabajo junto al usuario final del software.
- ✓ Necesidad de formalizar el proceso con una buena documentación, puesto que proporciona los artefactos y modelos para ello.
- ✓ La metodología se complementará con UML para la descripción de los distintos diagramas del modelado.
- ✓ A ser un proyecto de una sola persona se considera que un proceso guidado iterativo, incrementa, y unificado es una buena opción para desarrollar el software dada la poca experiencia que en este caso posee el desarrollador.

Para más información , exíste un anexo (D) como referencia a metodologías ágiles.

3.2. Planificación Temporal

3.2.1 Plan de Trabajo

En relación al Plan de Trabajo a llevar a cabo, las etapas a cubrir durante el desarrollo del TFG propuesto fueron las siguientes:

- ✓ **Propuesta y requisitos previos**: Nos reunimos con nuestros tutor de proyecto para recopilar los requisitos que deberán tener nuestro trabajo de fin de grado para que sea lo suficientemente completo para pasar la aprobación del tribunal y en paralelo ir aprendiendo de forma exhausta el framework Ruby On Rails.
- ✓ **Análisis de Requisitos:** Una vez recopilada toda la información en las entrevistas con nuestros tutor, extraemos la información importante y la esquematizamos. Definimos, organizamos, cada uno de los requisitos que hayamos extraído y los desarrollamos de forma exhausta. En esta fase también describiremos los casos de

uso, evitando en la medida de lo posible la redundancia de los mismos y maximizando la posibilidad de reutilizarlos.

- ✓ **Diseño:** En esta fase estudiaremos la mejor forma para incorporar la funcionalidad del sistema teniendo en cuenta que usaremos el framework de desarrollo de aplicaciones Ruby On Rails, con especial atención a que haremos uso del patrón Modelo-Vista-Controlador.
- ✓ **Implementación:** En esta fase codificaremos la aplicación para que sea funcional según hemos especificado en la fase de diseño; además, será necesario aprender la forma de utilizar de forma correcta el framework Ruby On Rails). En esta fase va incluida la creación de la estructuras necesarias para la interacción con el Sistema de Gestión de Bases de Datos (SGBD) .
- ✓ **Pruebas**: En este paso comprobaremos que la corrección del sistema que hemos desarrollado. En caso de encontrar algún fallo es el momento de solucionarlo y repetir las pruebas.
- ✓ **Memoria:** Toda la documentación que conforma esta memoria es el resultado de detallar cada uno de los documentos que hemos ido generando a través de la realización de la misma.
- ✓ **Presentación:** Creación y preparación de la presentación que mostraremos en la etapa de defensa ante el tribunal.

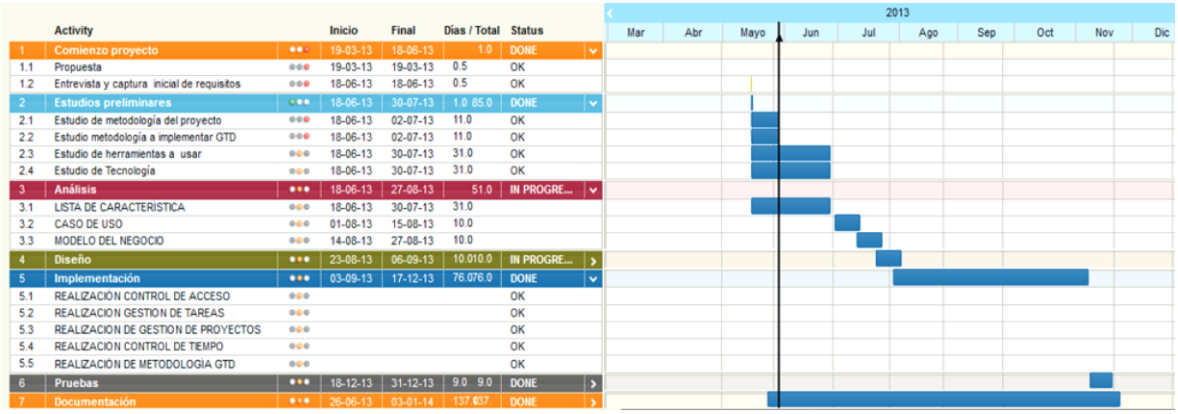
3.2.2 Diagrama de Gantt

Una vez explicados los aspectos generales de la gestión del proyecto, procedemos a detallar la planificación del mismo. Para realizar la planificación se ha utilizado una herramienta básica en la rama de ingeniería del software; el diagrama de Gantt.

El diagrama de Gantt es una popular herramienta gráfica utilizada para mostrar el tiempo previsto de esfuerzo para realizar un trabajo. A pesar de que, en principio, el diagrama de Gantt no indica las relaciones existentes entre actividades, la posición de cada tarea a lo largo del tiempo hace que se puedan identificar dichas relaciones e independencias. Fue Henry Laurence Gantt quien, entre 1910 y 1915, desarrolló y popularizó este tipo de diagrama en Occidente. En gestión de proyectos, el diagrama de Gantt se ha convertido en una herramienta básica con la finalidad de representar las diferentes unidades mínimas de trabajo y las fases, tareas y actividades programadas como parte de un proyecto.

A groso modo, el diagrama está compuesto por un eje vertical donde se establecen las actividades que constituyen el trabajo que se va a ejecutar, y un eje horizontal que muestra en un calendario, la duración de cada una de ellas.

En las siguiente página se muestra la figura que contiene la planificación del proyecto completo. La unidad de tiempo básica está expresada en días laborables teniendo en cuenta que cada día de trabajo corresponden a tres horas reales.



Según el diagrama de Gantt , mostrado en la figura anterior, podemos decir que el proyecto realizado ha tenido una duración de **170** días, **555** horas, donde se han realizado las fases de diseño, análisis, implementación y documentación, además, de algunas reuniones presenciales con el tutor del proyecto.

La tarea que ha durado más tiempo ha sido la de documentación puesto que se comenzó con el hito de 'inicio de proyecto' y finaliza al terminar el proyecto.

También en la etapa de implementación nos ha llevado bastante tiempo, aunque si sumamos las etapas de análisis, diseño, implementación y prueba vemos que eso nos lleva un total de 250 horas dedicadas al prototipo.

3.3. Presupuesto

En este apartado se va a mostrar de manera detallada el presupuesto desglosado del proyecto, especificando los diferentes gastos que han sido necesarios para su realización.

3.3.1 Resumen de Horas Dedicadas

Basándonos en el diagrama de Gantt expuesto en apartados anteriores es posible obtener número de horas totales dedicadas al proyecto.

Etapas	Dedicación (horas)
Etapa 1. Propuesta y requisitos previos	20
Etapa 2.Analisis de requisitos	45
Etapa 3.Diseño	50
Etapa 4.Implementación	250
Etapa 5. Prueba	30
Etapa6.Documentación y presentación	160
Dedicación Total	**555**

Tabla resumen.- de las etapas con sus respectivas horas

3.3.2 Resumen de Personal

En la siguiente tabla se muestran los cargos correspondientes al personal informático cualificado para realizar las distintas tareas o actividades. Los salarios por hora están en consonancia con los salarios de empleados en empresas similares del sector. Todos los costes son calculados sin I.V.A.

Cargo	Número de horas	Coste hora	Total(€)
Diseñador	50	25€	1250
Ingenieros (Programadores)	250	20€	5000
Responsable de documentación	160	17€	2.720
			8970

3.3.3 Resumen de Hardware

En la siguiente tabla se muestran los equipos informáticos adquiridos con su coste de amortización durante el periodo que dura el proyecto. Todos los costes son calculados sin I.V.A

Descripción	Unidades	Coste (€)	Coste total (€)
Ordenador intel core i5	1	800	800
Teclados logitech g15	1	30	30
Ratón Logitech Mx518	1	25	25
Monitor TFT LG L190TQ	1	100	100
Impresora hp	1	100	100
			1055€

3.3.4 Resumen de Software y Licencias

En la siguiente tabla se muestran las herramientas software necesarias para el proyecto. Todos los costes son calculados sin I.V.A.

Descripción	Unidades	Costes (€)	Costes total (€)
Microsoft office Profesional 2007	1	0	0
Editor Latex	1	0	0
Rails	1	0	0
Mozilla Firefox	1	0	0
Google Chrome	1	0	0

Total	0

3.3.5 Resumen de Material Fungible

En la siguiente tabla se muestra el material fungible que se estima necesario para la realización del proyecto, así como sus costes. En material de escritorio englobamos folios, bolígrafos, gomas, lápices, clips, carpetas, archivadores, recambios, grapadoras y demás material de oficina.

Todos los costes son calculados sin I.V.A.

Descripción	Costes totales(€)
Material de escritorio variado	150
Recambios de impresora	100
Total	**250€**

3.3.6 Resumen del Presupuesto Total

En la siguiente tabla se muestra el sumatorio de los totales anteriormente calculados pero sin el I.V.A. incluido. A la suma de los costes le vamos a añadir un veinte por ciento en concepto de costes indirectos, lo cual equilibrará los riesgos del proyecto y aquellos otros valores que no se han tenido en cuenta al realizar el presupuesto.

Descripción	Costes totales (€)
Equipo de trabajo	10.275€
Amortización	120€
Subcontratación de tareas	0
Coste de funcionamiento	900€
Costes indirectos (20 %)	2.526€
Total	13821€

Capítulo 4: Herramientas y Tecnologías

4.1 Tecnologías

En este capítulo y el siguiente mencionaremos las herramientas y tecnologías que entran en juego durante todo el proceso de desarrollo del software. Se incluye estos capítulos antes de la fase de implementación debido a que partes de las tecnologías que se usan ya estaban planificadas antes de las fases de diseño e implementación.

Las tecnologías que se expondrán a continuación son las siguientes:

- ✓ Ruby como lenguaje para la lógica de negocio
- ✓ HTML5 y CSS3 para la maquetación de vistas
- ✓ Javascript como lenguaje en lado del cliente
- ✓ JSON para el intercambio de datos
- ✓ MySQL como sistema de gestión de bases de datos

4.1.1 Ruby como Lenguaje para la Lógica

Ruby es un lenguaje , interpretado, reflexivo y orientado a objetos. Fue creado por el programador japonés Yukihiro Matsumoto, quien comenzó a trabajar en Ruby en 1993, y lo presentó públicamente en 1995. Este lenguaje combina una sintaxis inspirada en Python y Perl y su implementación oficial es distribuida bajo una licencia de software libre.

Como se ha mencionado Ruby es orientado a objetos: todos los tipos de datos son un objeto, incluidas las clases y tipos que otros lenguajes definen como primitivas (como enteros, booleanos y "nil"). Toda función es un método. Las variables siempre son referencias a objetos, no los objetos mismos. Ruby soporta herencia con enlace dinámico, A pesar de que Ruby no soporta herencia múltiple, se puede simular incluyendo todo los módulos que desees a una clase padre.

Ruby ha sido descrito como un lenguaje de programacion multiparadigma: permite programación procedural (definiendo funciones y variables fuera de las clases haciéndolas parte del objeto raíz Object). Soporta reflexión y meta programación, además de soporte para hilos de ejecución gestionados por el intérprete. Ruby tiene tipado dinámico, y soporta polimorfismo de tipos (permite tratar a subclases utilizando la interfaz de la clase padre).

Las características más importantes del lenguaje son :

- ✓ **Rápido y sencillo**: son innecesarias las declaraciones de variables, las variables no tienen tipo y la sintaxis es simple y consistente.
- ✓ **Orientado a objetos**.
- ✓ **Cuatro niveles de ámbito de variable:** global, clase, instancia y local.
- ✓ **Manejo de excepciones**, como Java y Python, para facilitar el manejo de errores.
- ✓ Iteradores y closures (pasando bloques de código).
- ✓ **Expresiones regulares nativas**
- ✓ Posibilidad de redefinir los operadores (sobrecarga de operadores).
- ✓ **Recolección de basura automática**.
- ✓ **Ruby es fácilmente portable**

- ✓ Tiene **manejo de hilos** independiente del sistema operativo.
- ✓ etc ...

4.1.2 Html5 y Css3 para la Maquetación de Vistas

4.1.2.1 Estructura con Html5

HyperText Markup Language (HTML) es el lenguaje con el que se escribe la estructura y la semántica del contenido de un documento Web. Permite describir la estructura y el contenido en forma de texto, además de complementar el texto con objetos tales como imágenes. Este lenguaje se escribe mediante etiquetas, que aparecen especificadas por corchetes angulares (<y >). Algunos ejemplos de elementos HTML son: <img>, <title>, <p> o . La extensión utilizada para los archivos de formato HTML es.htm ó .html.

HTML es un estándar internacional con especificaciones que son mantenidas por el **World WideWeb Consortium**. Es considerado un "estándar viviente" y está siempre técnicamente bajo construcción. La versión más actual de la especificación HTML se conoce como HTML5. HTML5 es la última versión de HTML y XHTML. El estándar HTML define un solo lenguaje que puede ser escrito usando la sintaxis de HTML, pero también usando la sintaxis más estricta de XML, y también se ocupa de las necesidades de las aplicaciones Web.

El marcado estructural es el que describe el propósito del texto, aunque no define cómo severa el elemento. HTML5 no describe el estilo y formato del contenido, sólo el propio contenido y su significado. Para la capa de presentación se usa Cascading Style Sheets(CSS) del que se hablará más adelante. Cabe destacar que HTML permite incluir scripts, hojas de estilos o ficheros en javascript.

HTML5 introduce muchas características actuales que permiten a los desarrolladores crear aplicaciones y sitios web con la funcionalidad, velocidad, rendimiento y experiencia de aplicaciones de escritorio.

Las características que se introducen en HTML5:

- ✓ **Multimedia y gráficos**. El navegador se ha convertido en una plataforma de pleno derecho para los juegos, animaciones, películas, cualquier cosa gráfica de verdad. Características como:
    - gráficos vectoriales (SVG y Canvas).
    - API de audio rico y baja latencia de red de WebSockets
    - API de gráficos y tecnologías que permiten crear una experiencia atractiva para los usuarios.
- ✓ **Trabajo desconectado de la red**. La API permite el uso de la web offline, es decir, aunque no exista conectividad, se podrán usar las aplicaciones en el navegador. Es posible gracias al caché de aplicaciones, su almacenamiento local, etc .
- ✓ **Rendimiento**. Permite que las aplicaciones web puedan rivalizar con las de escritorio debido a que los motores gráficos han mejorado sustancialmente.
- ✓ **Fácil desarrollo**. Permite orientar el desarrollo para más tipos de dispositivos con un menor esfuerzo.
- ✓ **Seguridad**. Las aplicaciones pasan a ser más seguras.

- ✓ **Bajo coste y fácil mantenimiento**. Las aplicaciones web listas para entornos de producción, comparadas con las de escritorio, corren a través de múltiples plataformas, son más fáciles de mantener y más barata de producir.
- ✓ **Audio y video.**
- ✓ **Web semántica**. Se añaden etiquetas para manejar la que se conoce como web 3.0 o web semántica: header, footer, article, nav... Estas etiquetas permiten describir cuál es el significado del contenido.
- ✓ **DOM**.En esta nueva versión aumenta la importancia del scripting DOM para el comportamiento web.

Por lo tanto, podemos decir que todas estas razones es lo que ayuda a afirmar que HTML5 acelera el ritmo de la innovación.

4.1.2.1.1 Abstracción Lenguaje de Marcado Html Usando Haml

Haml se usa para abstraerse del lenguaje de marcado HTML. HAML es, según sus propios creadores, "la belleza de los haiku traída al marcado html". Es otra forma de crear marcado html, usando la indentación del código no como la buena práctica que suele ser, sino como una obligación que tiene significado propio, logrando así una manera más condensada y gráfica de crear plantillas html. Es una alternativa a las plantillas ERB para RoR, que encaja mejor con la filosofía rails.

Veámoslo con un ejemplo:

```
1  <div id="perfil" class="ficha">
2  <div id= "avatar" class="borded" >
3     <img class= "centrada" src="avatar.jpg" alt="" /></div>
4  <div id= "nombre" class="campodetexto" >
5     <label>Nombre</label>
6
7  <%= @usuario.nombre %></div>
8  <div id= "ap1" class="campodetexto" >
9     <label>Primer Apellido</label>
10
11 <%= @usuario.apellido1 %></div>
12 <div id= "about" class="areadetexto" >
13    <label>Acerca de mi</label>
14
15 <%= @usuario.aboutme %></div>
16 </div>
```

Típica plantilla ERB. Lo más sencillo que puede hacerse. Con correcta indentación, que delimita que bloque está dentro de que tag.

Ahora pensemos a lo minimalista. El tag de cierre nos indica dónde acaba un bloque, y saber así lo que está dentro y lo que está fuera.

¿Existe alguna manera de saber que está dentro y que está fuera?

Pues sí, la indentación. Al fin y al cabo la usamos para ver las inclusiones de forma más visual. Hagámosla obligatoria y quitemos el tag de cierre.

¿Qué más se podría quitar que se repite mucho?

Miremos el código. Div, div, div, div. Los documentos html casi siempre se dividen con este tag. Podríamos hacer que si no digo lo contrario, todas los tags sean un -div-. Y aun más: id="", class="", id="", class="". Son atributos para los tags que usamos constantemente. Podríamos simplificar los id="" con # y los class="" con . olvidarnos de las comillas y el igual. Y además, cuando metemos un bloque ruby, hay que usar un tag con <%= . Usemos solo el igual.

Esto es lo que queda:

```
1  #perfil.ficha
2    #avatar.borded
3      %img.centrada{:src => "avatar.jpg" }
4    #nombre.campodetexto
5      %label Nombre
6      %p= @usuario.nombre
7    #ap1.campodetexto
8      %label Primer Apellido
9      %p= @usuario.apellido1
10   #about.areadetexto
11     %label Acerca de mi
12     %p.textogris= @usuario.aboutme
```

Bastante más legible. Y más conciso. Y por encima de todo, hemos escrito menos y cuando tengamos que releer esto porque nos hemos equivocado, y siempre nos equivocamos, enseguida vemos donde va cada cosa.

Por lo tanto, Haml acelera y simplifica la creación de plantilla html, siendo sus principales características: la limpieza, la legibilidad y la velocidad de producción.

4.1.2.2 Presentación con Css3

Las hojas de estilo en cascada (Cascading Style Sheets, o sus siglas CSS) hacen referencia a un lenguaje de hojas de estilos usado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas como es el caso de HTML5.

La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento HTML. Cualquier cambio de estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento, permitiendo así a los desarrolladores controlar el estilo y el formato de múltiples páginas web al mismo tiempo.

Los trabajos en el CSS3 comenzaron a la vez que se publicó la recomendación oficial de CSS2, y los primeros borradores de CSS3 fueron liberados en junio de 1999. Debido a la modularización del CSS3, diferentes módulos pueden encontrarse en diferentes estadios de su desarrollo.

Entre sus principales ventajas se destacan:

- ✓ Control centralizado de la presentación de un sitio web completo, con lo que se agiliza de forma considerable la actualización del mismo.
- ✓ Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web. Esto aumenta considerablemente la accesibilidad.
- ✓ Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario.

4.1.3 Javascript como Lenguaje Lado Cliente

JavaScript es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS).

JavaScript se diseño con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java, aunque no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). JavaScript se interpreta al mismo tiempo que las sentencias van descargándose junto con el código HTML.

4.1.4 Json para Intercambio de Datos

JSON, (JavaScript Object Notation), es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. Su simplicidad ha generalizado su uso, usándose especialmente como alternativa a XML en AJAX.

En JavaScript, un texto JSON se puede analizar fácilmente usando el procedimiento eval(), lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.

En la práctica, JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia (de aquí su uso por Yahoo, Google, etc, que atienden a millones de usuarios).

4.1.5 Mysql como Sistema de Gestión de Bases de Datos

4.1.5.1¿Qué es Mysql?

Es un sistema de gestión de bases de datos relacional, fue creada por la empresa sueca MySQL AB, la cual tiene el copyright del código fuente del servidor SQL, así como también de la marca.

MySQL es un software de código abierto, licenciado bajo la GPL de la GNU, aunque MySQL AB distribuye una versión comercial, en lo único que se diferencia de la versión libre, es en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de otra manera, se vulneraría la licencia GPL.

El lenguaje de programación que utiliza MySQL es Structured Query Language (SQL) que fue desarrollado por IBM en 1981 y desde entonces es utilizado de forma generalizada en las bases de datos relacionales.

4.1.5.2 Historia de Mysql

MySQL surgió alrededor de la década del 90, Michael Windenis comenzó a usar mSQL para conectar tablas usando sus propias rutinas de bajo nivel (ISAM). Tras unas primeras pruebas, llegó a la conclusión de que mSQL no era lo bastante flexible ni rápido para lo que necesitaba, por lo que tuvo que desarrollar nuevas funciones.
Esto resulto en una interfaz SQL a su base de datos, totalmente compatible a mSQL.

El origen del nombre MySQL no se sabe con certeza de donde proviene, por una lado se dice que en sus librerías han llevado el prefijo “my” durante los diez últimos años, por otra parte, la hija de uno de los desarrolladores se llama My. Así que no está claramente definido cuál de estas dos causas han dado lugar al nombre de este conocido gestor de bases de datos.

4.1.5.3 Características Principales

Inicialmente, MySQL carecía de algunos elementos esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de esto, atrajo a los desarrolladores de páginas web con contenido dinámico, debido a su simplicidad, de tal manera que los elementos faltantes fueron complementados por la vía de las aplicaciones que la utilizan. Poco a poco estos elementos faltantes, están siendo incorporados tanto por desarrolladores internos, como por desarrolladores de software libre.

En las últimas versiones se pueden de destacar las siguientes características principales:

- ✓ **El principal objetivo de MySQL es velocidad y robustez.**
- ✓ **Interioridades y portabilidad**
  - Escrito en C y en C++
  - Probado con un amplio rango de compiladores diferentes
  - Funciona en diferentes plataformas.
  - APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby.
  - Usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice.
  - Tablas hash en memoria, que son usadas como tablas temporales.

- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.
- Uso completo de multi-threaded mediante threads del kernel. Pueden usarse fácilmente multiple CPUs si están disponibles
- El código MySQL se prueba con Purify (un detector de memoria perdida comercial) así como con Valgrind, una herramienta GPL

✓ **Tipos de columnas**

- Diversos tipos de columnas: enteros con/sin signo de 1, 2, 3, 4, y 8 bytes de longitud, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM.
- Registros de longitud fija y longitud variable.

✓ **Sentencias y funciones**

- Soporte completo para operadores y funciones en las cláusulas de consultas SELECT y WHERE.
- Por ejemplo:

```
SELECT CONCAT(first_name, ' ', last_name)
   -> FROM citizen
   -> WHERE income/dependents > 10000 AND age > 30;
```

- Soporte completo para las cláusulas SQL GROUP BY y ORDER BY. Soporte de funciones de agrupación (COUNT(), COUNT(DISTINCT ...), AVG(), STD(), SUM(), MAX(), MIN(), y GROUP_CONCAT()).
- Soporte para LEFT OUTER JOIN y RIGHT OUTER JOIN cumpliendo estándares de sintaxis SQL y ODBC.
- Soporte para alias en tablas y columnas como lo requiere el estándar SQL.
- DELETE, INSERT, REPLACE, y UPDATE devuelven el número de filas que han cambiado (han sido afectadas). Es posible devolver el número de filas que serían afectadas usando un flag al conectar con el servidor.
- El comando específico de MySQL SHOW puede usarse para obtener información acerca de la base de datos, el motor de base de datos, tablas e índices.
- El comando EXPLAIN puede usarse para determinar cómo el optimizador resuelve una consulta

✓ **Seguridad**

- Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está cifrado cuando se conecta con un servidor.

✓ **Escalabilidad y límites**

- Soporte a grandes bases de datos. Usamos MySQL Server con bases de datos que contienen 50 millones de registros. También conocemos a usuarios que usan MySQL Server con 60.000 tablas y cerca de 5.000.000.000.000 de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).Un índice puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB, o TEXT.

✓ **Conectividad**

- Los clientes pueden conectar con el servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows de la familia NT (NT,2000,XP, o 2003), los clientes pueden usar named pipes para la conexión. En sistemas Unix, los clientes pueden conectar usando ficheros socket Unix.

✓ **Localización**

- El servidor puede proporcionar mensajes de error a los clientes en muchos idomas.
- Soporte completo para distintos conjuntos de caracteres, incluyendo latin1 (ISO-8859-1), german, big5, ujis, y más. Por ejemplo, los caracteres escandinavos 'â', 'ä' y 'ö' están permitidos en nombres de tablas y columnas. El soporte para Unicode está disponible
- Todos los datos se guardan en el conjunto de caracteres elegido. Todas las comparaciones para columnas normales de cadenas de caracteres son case-insensitive.

✓ **Clientes y herramientas**

- MySQL server tiene soporte para comandos SQL para chequear, optimizar, y reparar tablas.

4.1.5.4 Ventajas y Desventajas de Mysql

Ventajas	Desventajas
Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.	Un gran porcentaje de las utilidades de MySQL no están documentadas.
Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.	No es tan intuitivo, como otros programas (ACCESS).
Facilidad de configuración e instalación.	
Soporta gran variedad de Sistemas Operativos Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.	
Conectividad y seguridad	

4.1.5.5 Por qué Utilizar Mysql

MYSQL es un sistema de base de datos fácil de aprender. Aquellos administradores de bases de datos que han tenido experiencia con otros manejadores de bases de datos podrán adaptarse rápidamente a su uso.

Incorpora características modernas (orientado a objetos, manejo de bloqueos predefinido). Es altamente configurable, con lo cual puede personalizarse de acuerdo al escenario donde será utilizado.

Ha sido diseñado para mantenerse estable con grandes volúmenes de datos y transacciones. Es altamente escalable, tanto en la enorme cantidad de datos que puede manejar y en el número de usuarios concurrentes que puede administrar.

Es multiplataforma (cuenta con versiones para sistemas operativos Unix y Windows). MYSQL no es sólo un sistema de base de datos de gran alcance capaz de usarse en las empresas, es todo una plataforma de desarrollo sobre la cual puedes desarrollar todo tipo de software que requieren un SGBDR de grandes capacidades.

MySQL es un software es Open Source, esto significa que es posible para cualquiera usar y modificar el software. Cualquiera puede bajar el software MySQL desde internet y usarlo sin pagar nada. Si lo desea, puede estudiar el código fuente y cambiarlo para adaptarlo a sus necesidades.

El software MySQL usa la licencia GPL (GNU General Public License), para definir lo que puede y no puede hacer con el software en diferentes situaciones.

4.1.6 Otras Tecnologías Relacionadas

4.1.6.1 Ajax

El término AJAX fue publicado por Jesse James Garrett el 18 de Febrero de 2005. El término AJAX se puede traducir como (JavaScript asíncrono + XML).

Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen.

Las tecnologías que forman AJAX son:

- ✓ **XHTML y CSS.** Para crear una presentación basada en estándares.
- ✓ **DOM.** Para la interacción y manipulación dinámica de la presentación.
- ✓ **XML, XSLT y JSON.** Para el intercambio y la manipulación de información.
- ✓ **XMLHttpRequest.** Para el intercambio asíncrono de información.
- ✓ **JavaScript.** Para unir todas las demás tecnologías.

Normalmente, el modelo clásico de las aplicaciones web funciona siguiendo el siguiente esquema:

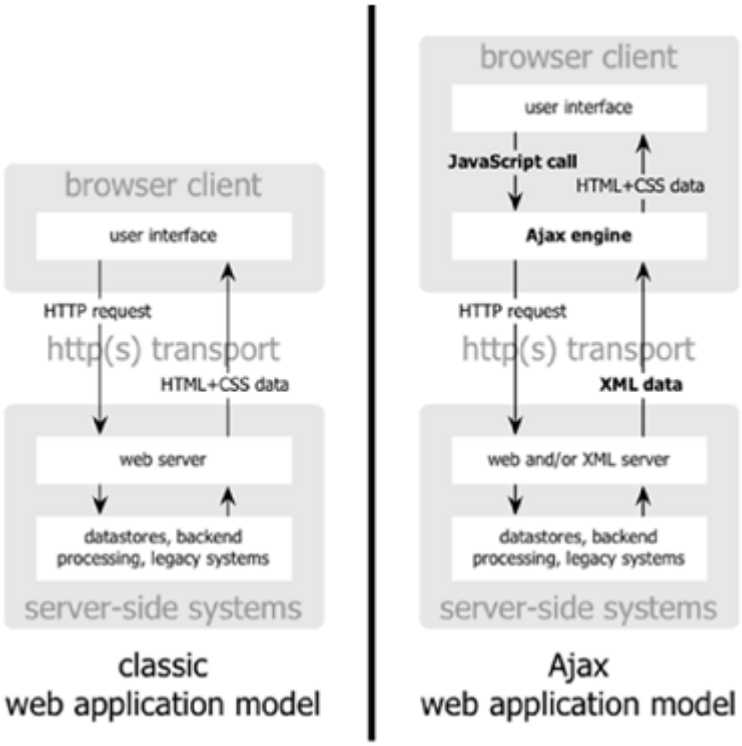
1. Las acciones del usuario en la interfaz disparan un requerimiento HTTP al servidor web.
2. El servidor efectúa un proceso (*recopila información, procesa números, hablando con varios sistemas propietarios*), y le devuelve una página HTLM al cliente.

Al realizar peticiones continuas al servidor, el usuario debe esperar a que se recargue la página con los cambios solicitados. Si la aplicación debe realizar peticiones continuas, la interacción con el usuario se convierte en algo molesto.

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX llamada "Ajax Engine" mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

A continuación se muestra una imagen comparando ambos modelos el tradicional y usando AJAX:



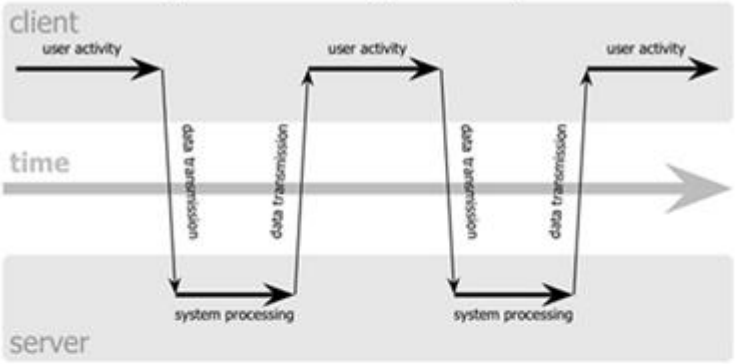
¿Cómo se consigue intercambio de información?

Como hemos mencionado anteriormente intercambio de información con el servidor se produce en un segundo plano. Las peticiones HTTP al servidor se sustituyen por peticiones JavaScript que se realizan al elemento encargado de AJAX. Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata. Si la interacción requiere una respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX.

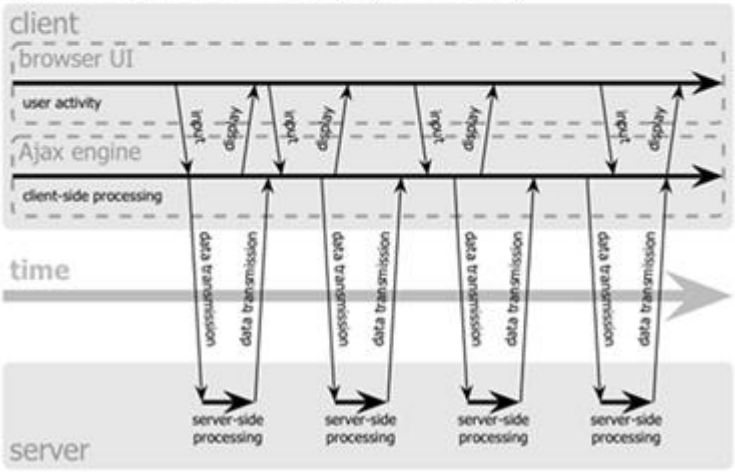
En este caso, la interacción del usuario no se ve interrumpida por recargas de página o largas esperas por la respuesta del servidor.

A continuación se mostrará una imagen comparativa mostrando las aplicaciones web tradicionales con su interacción síncrona y la comunicación asíncrona de las aplicaciones creadas con AJAX :

classic web application model (synchronous)



Ajax web application model (asynchronous)



4.2 Herramientas

Las distintas fases de trabajo del proyecto están apoyadas por un uso de software específico para su desarrollo. Para el desarrollo de la aplicación se ha contado con un equipo formado por el tutor de proyecto y el propio desarrollador.

Durante el proceso se diferencian dos fases que están estrechamente ligadas:

1. documentación y gestión.
2. desarrollo e implementación del proyecto.

Cada una de estas fases ha requerido el uso de distintas herramientas para conseguir su objetivo.

A continuación se enumeran detallando la sección en la que se encuentran en este documento y la versión utilizada durante su uso.

1. Para la documentación y gestión.

- ✓ Microsoft Office Word como procesador de texto—versión estable 2007
- ✓ Sublime text 2 como editor de código—Versión 0.0.4 (1003)
- ✓ StarUML para realización de diagramas—Versión 5.0.2 Windows
- ✓ Balsamiq Mockups para diseño de prototipos de interfaces—Versión 2.1.1.9
- ✓ Photoshop como editor de imágenes—Versión 12.0.4 (CS5)

2. Para el desarrollo e implementación del software se ha utilizado.

- ✓ Ruby on Rails como framework Ruby— Última versión estable
- ✓ jQuery como framework Javascript— Última versión estable
- ✓ Bootstrap como framework HTML5/CSS3—Versión 2.3
- ✓ Git como sistema de control de versiones—Última versión estable

4.2.1 Recursos Físicos Usados

Para dar cabida al conjunto de herramientas y tecnologías señaladas, se ha utilizado un soporte físico como equipo de trabajo con las siguientes características:

- ✓ **Modelo**. lenovo Z500
- ✓ **Procesador.** Intel Core i5 . 2,60 GHz
- ✓ **Memoria**. 6 Gb DDR3
- ✓ **Gráficos**. GeForce GT 740M
- ✓ **Disco Duro**. 438 Gb SATA
- ✓ **Software**. Windows 8
- ✓ **Conexión internet**. Banda ancha 10Mb

4.2.2 Microsoft Office Word

Microsoft Office es una suite de oficina que abarca e interrelaciona aplicaciones de escritorio, servidores y servicios para los sistemas operativos Microsoft Windows y Mac OS. Microsoft Word es un software destinado al procesamiento de textos. Fue creado por la empresa Microsoft, y actualmente viene integrado en la *suite* ofimática Microsoft Office.

Originalmente fue desarrollado por Richard Brodie para el computador de IBM bajo sistema operativo DOS en 1983. Las versiones actuales son Microsoft Office Word 2013 para Windows y Microsoft Office Word 2011 para Mac. Ha llegado a ser el procesador de texto más popular del mundo.

Microsoft Office word usa la filosofía conocida como procesadores WYSIWYG, es decir, "*lo que ves es lo que obtienes*".

4.2.3 Sublime Text 2 como Editor de Código

Sin lugar a duda la herramienta del software que más usamos los programadores es sin duda el editor de código, es por ello que es muy importante elegir un buen editor si queremos que nuestra productividad no sufra. Sublime Text es un editor de texto y editor de código fuente creado en Python. Se distribuye de forma gratuita, sin embargo no es software libre se puede obtener una licencia para su uso ilimitado, pero él no disponer de esta no genera ninguna limitación más allá de una alerta cada cierto tiempo.

Algunas de las características más destacadas de este editor son:

- ✓ **Minimapa:** Consiste en una previsualización de la estructura del código, es muy útil para desplazarse por el archivo cuando se conoce bien la estructura de este.
- ✓ **Multi Selección:** Hace una selección múltiple de un término por diferentes partes del archivo.
- ✓ **Multi Cursor:** Crea cursores con los que podemos escribir texto de forma arbitraria en diferentes posiciones del archivo.
- ✓ **Multi Layout:** Trae siete configuraciones de plantilla podemos elegir editar en una sola ventana o hacer una división de hasta cuatro ventanas verticales o cuatro ventanas en cuadricula.
- ✓ **Soporte nativo para infinidad de lenguajes:** Soporta de forma nativa 43 lenguajes de programación y texto plano.
- ✓ **Syntax Highlight configurable:** El remarcado de sintaxis es completamente configurable a través de archivos de configuración del usuario.
- ✓ **Búsqueda Dinámica:** Se puede hacer búsqueda de expresiones regulares o por archivos, proyectos, directorios, una conjunción de ellos o todo a la vez.
- ✓ **Auto completado y marcado de llaves:** Se puede ir a la llave que cierra o abre un bloque de una forma sencilla.
- ✓ **Soporte de Snippets y Plugins:** Los snippets son similares a las macros o los bundles además de la existencia de multitud de plugins.
- ✓ **Configuración total de Keybindings:** Todas las teclas pueden ser sobrescritas a nuestro gusto.
- ✓ **Acceso rápido a linea o archivo:** Se puede abrir un archivo utilizando el conjunto de teclas Cmd+P en Mac OS X o Ctrl+P en Windows y Linux y escribiendo el nombre del mismo o navegando por una lista. También se puede ir a una línea utilizando los dos puntos ":" y el número de línea.

- ✓ **Paleta de Comandos:** Un intérprete de Python diseñado solo para el programa con el cual se puede realizar infinidad de tareas.
- ✓ **Coloreado y envoltura de sintaxis:** Si se escribe en un lenguaje de programación o marcado, resalta las expresiones propias de la sintaxis de ese lenguaje para facilitar su lectura.
- ✓ **Pestañas:** Se pueden abrir varios documentos y organizarlos en pestañas.
- ✓ **Resaltado de paréntesis e indentación:** Cuando el usuario coloca el cursor en un paréntesis, corchete o llave, resalta esta y el paréntesis, corchete o llave de cierre o apertura correspondiente.

4.2.4 Staruml para Realizar Diagramas

StarUML es una herramienta para el modelamiento de software basado en los estándares UML (Unified Modeling Language) y MDA (Model Driven Arquitecture).

StarUML en un principio era un producto comercial y actualmente es de licencia abierta. El software heredó todas las características de la versión comercial y poco a poco ha ido mejorando sus características para dar soporte completo al diseño UML, entre las cuales se encuentran:

- ✓ Diagrama de casos de uso
- ✓ Diagrama de clase
- ✓ Diagrama de secuencia
- ✓ Diagrama de colaboración.
- ✓ Diagrama de estado.
- ✓ Diagrama de actividad.
- ✓ Diagrama de componentes
- ✓ Diagrama de despliegue.
- ✓ Diagrama de composición estructural

Otras funcionalidades de las que dispone el software son:

- ✓ Definir elementos propios para los diagramas, que no necesariamente pertenezcan al estándar de UML,
- ✓ La capacidad de generar código a partir de los diagramas y viceversa, actualmente funcionando para los lenguajes c++, c# y java.
- ✓ Generar documentación en formatos Word, Excel y PowerPoint sobre los diagramas.
- ✓ Patrones GoF (Gang of Four) , EJB (Enterprise JavaBeans) y personalizados.
- ✓ Plantillas de proyectos.
- ✓ Posibilidad de crear plugins para el programa.

4.2.5 Balsamig Mockups para Diseño de Prototipo de Interfaces

Balsamiq Mockups es una herramienta que nos permite realizar fácilmente una representación esquemática o prototipo de la solución que llevaremos adelante, sin entrar en etapas posteriores como el diseño gráfico o la programación web.

Gracias a esta herramienta, podemos comunicar rápidamente las propuestas de solución, sin invertir demasiada cantidad de tiempo en esta primera etapa. Podemos acordar con el cliente aspectos claves de la solución a desarrollar, como la distribución general de los elementos, sus jerarquías y la navegación de los mismos.

Balsamiq Mockups nos provee de representaciones de todos los elementos utilizados para la construcción de una web: pantallas de navegadores, títulos, imágenes, etc. Dispone de más de 75 modelos ya definidos. Simplemente lo organizarlos en un documento y ya podemos tener una primera aproximación de la solución a desarrollar.

4.2.6 Photoshop como Editor de Imágenes

Adobe Photoshop es una aplicación para la creación, edición y retoque de imágenes. Es desarrollado por la compañía Adobe Systems. Se lanzó originalmente para computadoras Apple, pero luego saltó a la plataforma Windows.

Este programa se ha hecho muy popular, se ha convertido en un estándar en retoque fotográfico. Reconoce múltiples formatos de imágenes: PSD, PDD, PostScript, EPS, DCS, TIFF, BMP, GIF, JPEG, TIFF, PNG, PDF, ICO, RAW, entre otros.

Photoshop en sus versiones iníciales trabajaba en un espacio formado por una sola capa, donde se podían aplicar toda una serie de efectos, textos, marcas y tratamientos.

En la actualidad el software a mejorado bastante incluyendo un espacio de trabajo multicapa, inclusión de elementos vectoriales, gestión avanzada de color (ICM / ICC), tratamiento extensivo de tipografías, control y retoque de color, efectos creativos, posibilidad de incorporar plugins de terceras compañías, exportación para sitios web entre otros.

Aunque el propósito principal de Photoshop es la edición fotográfica, éste también puede ser usado para crear imágenes, efectos, gráficos y más en muy buena calidad.

4.2.7 Ruby and Rails como Framework de Ruby

Ruby on Rails, es una estructura de soporte para la programación web desarrollada en el lenguaje Ruby por David Heinemeier Hansson, como resultado del desarrollo de Basecamp. La primera versión del framework salió a la luz pública en julio de 2004, así que Ruby on Rails, es un framework muy reciente en comparaciones con otros soportes más consolidados. Para entender qué es Ruby on Rails, primero tenemos que conocer un poco en qué se apoya.

Los dos principios fundamentales sobre los cuáles se apoya Ruby on Rails, son:

- **No te repitas.** No te repitas significa que las definiciones deberían hacerse una sola vez. Dado que Ruby on Rails es un framework de pila completa, los componentes están integrados de manera que no hace falta establecer puentes entre ellos.

  Por ejemplo, en ActiveRecord, las definiciones de las clases no necesitan especificar los nombres de las columnas; Ruby puede averiguarlos a partir de la propia base de datos, de forma que definirlos tanto en el código como en el programa sería redundante.

- **Convenciones antes que configuración.** Significa que el programador sólo necesita definir aquella configuración que no es convencional.

  Por ejemplo, si hay una clase Historia en el modelo, la tabla correspondiente de la base de datos es historias, pero si la tabla no sigue la convención (por ejemplo blogposts) debe ser especificada manualmente (set_table_name "blogposts"). Así, cuando se diseña una aplicación partiendo de cero sin una base de datos preexistente, el seguir las convenciones de Rails significa usar menos código, ya que el propio framework, por defecto, genera ciertos elementos.

  Las convenciones existentes en Rails han sido definidas por el grupo de desarrolladores con la intención de facilitar el desarrollo, pero eso no implica que dichas convenciones no se puedan redefinir y cada cual configurarlas de otra manera y a su medida.

4.2.7.1 Servicios Restful

REST define un set de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes.REST emergió en los últimos años como el modelo predominante para el diseño de servicios. De hecho, REST logró un impacto tan grande en la web que prácticamente logró desplazar a SOAP y las interfaces basadas en WSDL por tener un estilo bastante más simple de usar.

Principios de REST

Una implementación concreta de un servicio web REST sigue cuatro principios de diseño fundamentales:

1. **Utiliza los métodos HTTP de manera explícita.** Una de las características claves de los servicios web REST es el uso explícito de los métodos HTTP. REST hace que los desarrolladores usen los métodos HTTP explícitamente de manera que resulte consistente con la definición del protocolo. Este principio de diseño básico establece una asociación uno-a-uno entre las operaciones de crear, leer, actualizar y borrar y los métodos HTTP.

   De acuerdo a esta asociación:

   - ✓ Se usa **POST** para crear un recurso en el servidor
   - ✓ Se usa **GET** para obtener un recurso
   - ✓ Se usa **PUT** para cambiar el estado de un recurso o actualizarlo
   - ✓ Se usa **DELETE** para eliminar un recurso

2. **No mantiene estado.** El no mantener estado mejora el rendimiento de los servicios web y simplifica el diseño e implementación de los componentes del servidor, ya que la ausencia de estado en el servidor elimina la necesidad de sincronizar los datos de la sesión con una aplicación externa, es decir, se deben enviar peticiones que incluyan todos los datos necesarios para cumplir el pedido, de manera que los componentes en los servidores intermedios puedan redireccionar y gestionar la carga sin mantener el estado localmente entre las peticiones.

3. **Expone URIs con forma de directorios.** Las URIs representa una jerarquía,es decir, las URIs contiene una única ruta raíz, y va abriendo ramas a través de las subrutas para exponer las áreas principales del servicio. De acuerdo a esta definición, una URI no es solamente una cadena de caracteres delimitada por barras, sino más bien un árbol con subordinados y padres organizados como nodos.

4**. Transfiere XML, JavaScript Object Notation (JSON), o ambos.** La representación de un recurso en general refleja el estado actual del mismo y sus atributos al momento en que el cliente de la aplicación realiza la petición. La representación del recurso son simples. Esto podría ser una representación de un registro de la base de datos que consiste en la asociación entre columnas y tags XML, donde los valores de los elementos en el XML contienen los valores de las filas. O la representación de un recurso es una fotografía en el modelo de datos del sistema.

4.2.8 Jquery como Framework Javascripts

jQuery es un framework de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Es un software libre y de código abierto. Ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, podemos obtener resultados en menos tiempo y espacio.

Sus principales características:

- ✓ Selección de elementos DOM.
- ✓ Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1-3.
- ✓ Eventos.
- ✓ Manipulación de la hoja de estilos CSS.
- ✓ Efectos y animaciones.
- ✓ Animaciones personalizadas.
- ✓ AJAX.
- ✓ Soporta extensiones.
- ✓ Utilidades varias como obtener información del navegador, operar con objetos y vectores, funciones como trim() (elimina los espacios en blanco del principio y final de una cadena de caracteres), etc.
- ✓ Compatible con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+,
- ✓ Liviano. El archivo del framework ocupa unos 56 KB (19KB en caso de enviarse de manera comprimida
- ✓ Estabilidad y buena documentación. Cuenta con un gran equipo de desarrolladores a cargo de mejoras y actualizaciones del framework.

Su uso:

jQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX.La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello utiliza las funciones **$() o jQuery().**

Función $()

La forma de interactuar con la página es mediante la función $(), un alias de jQuery(), que recibe como parámetro una expresión CSS o el nombre de una etiqueta HTML y devuelve todos los nodos (elementos) que concuerden con la expresión. Esta expresión es denominada **selector** en la terminología de jQuery.

```
$("#tablaAlumnos"); // Devolverá el elemento con id="tablaAlumnos"
$(".activo");       // Devolverá una matriz de elementos con activo
```

Una vez obtenidos los nodos, se les puede aplicar cualquiera de las funciones que facilita la biblioteca.

```
// Se elimina el estilo (con removeClass()) y se aplica uno nuevo (con
addClass()) a todos los nodos con class="activo"
$(".activo").removeClass("activo").addClass("inactivo");
```

O por ejemplo, efectos gráficos:

```
// Anima todos los componentes con class="activo"
$(".activo").slideToggle("slow");
```

Función jquery()

Comúnmente antes de realizar cualquier acción en el documento con jQuery(), debemos percatarnos de que el documento esté listo. Para ello usamos $(document).ready();, de esta forma:

```
$(document).ready(function() {

  //Aquí van todas las acciones del documento.

});
```

4.2.9 Bootstrap como Framework Html5 y Css3

Es un framework para el desarrollo de aplicaciones web desarrollo de Twitter. El framework presenta aspecto bastante limpio y construido sobre las tecnologías más modernas (HTML5, CSS3,Responsive Web Design). Gracias a la tecnología Responsive, permite que los sitios construidos con la ayuda de Bootstrap funcionen bien en ordenadores de escritorio, tabletas o teléfonos móviles.

Entre sus principales características se encuentran:

- ✓ Ofrece grids fijados y líquidos – 724px, 940px, 1170px
- ✓ Ofrece 12 columnas
- ✓ Sus herramientas de la interfaz de usuario son ideales para la creación rápida de un prototipo
- ✓ Se gestiona a través det Github
- ✓ Compatible con Chrome, Firefox, Safari, IE+ y Opera con tabletas y teléfonos inteligentes.
- ✓

4.2.10 Git como Sistema de Control de Versiones

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo, de modo que se puedan recuperar versiones específicas más adelante.

Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas.

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número archivos de código fuente. Hoy en día Git se ha convertido en un sistema de control de versiones con funcionalidad plena.

El flujo de trabajo básico en Git es:

1. Modificar una serie de archivos en directorio de trabajo.
2. Preparar los archivos, añadiendo instantáneas de ellos al área de preparación.
3. Confirmar los cambios, lo que toma los archivos tal y como están en el área de preparación, y almacena esa instantánea de manera permanente en el directorio de Git remoto.

Sus principales características son:

- ✓ Fuerte apoyo al desarrollo no-lineal, rapidez en la gestión de ramificaciones y mezclado de diferentes versiones. Git incluye herramientas específicas para navegar y visualizar un historial de desarrollo no-lineal.
- ✓ Una presunción medular en Git es que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente, conforme se pasa entre varios programadores que lo revisan.
- ✓ Gestión distribuida. Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramificaciones adicionales y pueden ser fusionados en la misma manera que se hace con la ramificación local.
- ✓ Los almacenes de información pueden publicarse por HTTP, FTP, o mediante un protocolo nativo, ya sea a través de una conexión TCP/IP simple o a través de cifrado SSH.
- ✓ Los repositorios Subversion y SVK se pueden usar directamente con git-svn.
- ✓ Todas las versiones previas a un cambio determinado, implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio (denominado autenticación criptográfica de historial).

Capítulo 5: Desarrollo de la Aplicación

5.1 Requisitos del Sistema

El primero de los pasos a seguir para comenzar a desarrollar el producto se ha llevado a cabo con la elección de la metodología, quien nos marcará las pautas a seguir durante la creación del proyecto.

El fin de este capítulo es el de entender y analizar las necesidades del negocio para el cual se está desarrollando el software.

En este apartado se describe de forma detallada cada una de las funcionalidades que pueden componer el proyecto, con el fin de guiar el desarrollo hacia el sistema correcto.

Antes de entrar en materia, se va a plantear cuáles serán los objetivos a conseguir .

- **Captura de requisitos**. El fin último es el de llegar a conocer el dominio del problema y entender el modelo de negocio. Se desarrolla junto al tutor que hará rol de cliente un modelo de dominio y se plantearán las posibles características que pueda tener el software (enumeración de requisitos candidatos).

  Dada la insuficiencia del tiempo para la creación de un software final, y dada la inmensa lista de característica encontrada, se priorizará sobre aquellas características que aporte más valor al negocio.
- **Especificación de requisitos**. Una vez que se ha entendido el contexto en el que se sitúa la aplicación, se define cuál será la arquitectura de los requisitos más importantes para detallarlos de forma exhaustiva. Tanto los requisitos funcionales como los no funcionales tienen que ser especificados.

La captura de requisitos es un acto de descubrimiento, es el proceso de averiguar normalmente en circunstancias difíciles lo que se debe construir. Para comenzar este descubrimiento de requisitos se ha seguido el siguiente flujo de trabajo:

- **Enumerar los requisitos candidatos.** El propósito fundamental del flujo de trabajo de los requisitos es guiar el desarrollo hacia el sistema correcto. Esto se consigue mediante una descripción de requisitos del sistema sobre qué debe y qué no debe hacer el sistema.
- **Comprender el contexto del sistema.** Hay por lo menos dos aproximaciones para expresar el contexto de un sistema software: modelado del dominio y modelado del negocio.

  1. Un **modelo del dominio** describe los conceptos importantes del contexto como objetos del dominio y enlaza estos objetos unos con otros. La identificación y la asignación de un nombre para estos objetos nos ayuda a desarrollar un glosario de términos que permitirán comunicarse mejor a todos los que están trabajando en el sistema. Más adelante, los objetos del dominio nos ayudarán a identificar algunas de las clases a medida que analizamos y diseñamos el sistema.
  2. Un **modelo del negocio** es un conjunto a un nivel superior del modelo del dominio e incluye algo más que sólo los objetos del dominio. El objetivo del modelado del negocio es describir los procesos con el objetivo de comprenderlos. Este modelo también establece las competencias

requeridas en cada proceso: sus trabajadores, sus responsabilidades y las operaciones que llevan a cabo .

- **Capturar requisitos funcionales/ y no funcionales.** Para hacer capturar requisitos funcionales usamos la técnica basada en los casos de uso. Estos casos de uso capturan tanto los requisitos funcionales como los no funcionales que son específicos de cada caso de uso. Cada usuario quiere que el sistema haga algo para él, es decir, que lleve a cabo ciertos casos de uso. Por tanto, podemos concluir que, para el usuario, un caso de uso es un modo de utilizar el sistema.

5.1.1 Getting Things Done -(GTD)

Como hemos comentado al comienzo de esta memoria, *Getting Things Done* (GTD) es un método de gestión de las actividades y el título de un libro de David Allen que en español se ha titulado 'Organízate con eficacia'.

GTD se basa en el principio de que una persona necesita liberar su mente de las tareas pendientes guardándolas en un lugar específico. De este modo, no es necesario recordar lo que hay que hacer y se puede concentrar en realizar las tareas.

A diferencia de otros expertos en gestión del tiempo, Allen no se centra en el establecimiento de prioridades. En su lugar, él insta a la creación de listas de tareas específicas para cada contexto, por ejemplo, una lista de llamadas telefónicas pendientes o recados que hacer en la ciudad. También sugiere que cualquier nueva tarea que pueda ser completada en menos de dos minutos debería ser hecha inmediatamente.

La psicología de GTD se basa en hacer fácil el almacenamiento, seguimiento y revisión de toda la información relacionada con las cosas que necesitas hacer. Allen sugiere que muchos de los bloqueos mentales en los que nos encontramos a la hora de completar ciertas tareas, vienen dados por una planificación insuficiente (por ejemplo, para cualquier trabajo debemos aclarar lo que se debe conseguir y qué acciones se deben llevar a cabo para completarlo). Según Allen, es más práctico hacerlo reflexionando previamente sobre ello, generando una serie de acciones que hacer más tarde, sin necesidad de volver a planificarlo durante su realización.

Allen también sostiene que nuestro "sistema de recordatorios interno" (nuestra memoria), es considerablemente ineficiente y rara vez nos acordamos de lo que necesitamos hacer en el momento y en el lugar en el que podemos hacerlo. Por tanto, las "acciones próximas" almacenadas según el contexto en un "sistema confiablemente externo" actúan como soporte que nos asegura que lo recordaremos en el momento y lugar adecuados para su realización.

Allen propone una estructurar de nuestras tareas en un sistema de listas. Hay algunas estandarizadas Entrada de tareas, Siguiente, Algún día, Proyectos, En espera, pero podemos potenciar el sistema a través de una estructura auxiliar de listas que nos permita clasificar las acciones a realizar por otros criterios necesario para finalizarla:

- **Por ubicación**: Si estamos en casa o en la oficina habrá tareas que no podremos realizar por no disponer del entorno o las herramientas de trabajo adecuado. Creamos listas como Oficina, Casa o más específicas como Reunión, Ordenador, Encargos para recolectar tareas a realizar ante nuestro equipo informático o cuando salimos de compras. La clave radica en agrupar tareas que

podamos realizar de golpe sin cambiar de entorno, preservando nuestra concentración y economizando tiempo.

- **Por tiempo**: Disponemos siempre de una lista de tareas para realizar en un intervalo corto de tiempo. Aquellos diez minutos que tenemos entre reunión y reunión, esa media hora que nos sobra a la hora de comer o al final de la jornada. No siempre dispondremos de la energía y del tiempo suficiente para completar una gran tarea, la lista de pequeñas acciones potenciará nuestro día aprovechando los momentos muertos.

La idea es descargar todo de nuestra memoria ram y traspasarlas a nuestro sistema de listas. La primera vez podemos quedarnos horas recopilando todos los elementos dispersos de nuestra mente, y de nuestro despacho. Una vez finalizado procesaremos cada una de las acciones enviándolas a una de las listas del sistema.

Es importante definir el concepto **ACCIÓN**. Para GTD una tarea es una acción física. Por ejemplo, una cosa es hacer la declaración de la renta (incorrecto) y otra es ir a Solicitar el borrador de la declaración a hacienda (correcto). Delimitamos la primera acción a realizar con más precisión, evitando perdernos en tareas vagamente definidas y que pueden estar compuestas de varios pasos, para estas crearemos un proyecto. Gtd considera un proyecto todo aquel objetivo que necesita más de una acción física para ser completado. En el ejemplo anterior la tarea hacer declaración de la renta podría componerse de las siguientes acciones:

1. *Buscar los impresos para la declaración de la renta*
2. *Cumplimentar la documentación necesaria*
3. *Confirmar el borrador*
4. *Realizar / Esperar transferencia del pago.*

Un proyecto GTD no tiene porqué coincidir con lo que nosotros concebimos como proyecto. En el caso de hacer obras en casa, nuestro proyecto Hacer obras – en GTD – se descompondría en varios proyectos del orden de: Construir pared, Enyesar, Pintar habitación.., cada uno de ellos con sus acciones físicas pertinentes

Si no entendemos los conceptos acción y proyecto podemos llegar a sobrecargar nuestro sistema de control de tareas con acciones demasiado complejas y poco desglosadas, que lo convertirán en algo demasiado confuso, o dicho de otra manera, en un conjunto de listas de tareas convencional.

GTD funciona por su simplicidad a la hora de indicarnos que debemos hacer, y como debemos organizar nuestro trabajo, sin estas cualidades perderemos todo el dinamismo que nos aporta.

5.1.1.1 Principios de GTD

Los principios esenciales de GTD son los siguientes:

5.1.1.1.1 Recopilar

La acumulación de frentes abiertos provoca la carga de nuestra memoria inmediata con detalles y 'COSAS' a recordar. La necesidad continua de tener presentes los detalles de nuestros proyectos nos lleva a utilizar nuestra mente como sistema para recordar, mantiene nuestra memoria permanentemente ocupada, generando un nivel constante de preocupación que nos roba energía. Sumado a otras dificultades diarias, puede favorecer la aparición del estrés. Nuestro objetivo pasa por disponer de toda la energía evaporada en el proceso para derivarla a procesos creativos que nos permitan generar nuevas ideas y un estado de estabilidad para hacer frente al día a día de forma menos estresante.

GTD propone la utilización de un sistema para almacenar todos los detalles de nuestros proyectos y tareas pendientes sobre un soporte material. En lugar de utilizar nuestra mente para guardar la información necesaria, utilizamos un sistema de listas**,** con el objetivo de desocupar nuestra memoria inmediata y ganar agilidad y concentración.

El primer paso para iniciar GTD es el proceso de recopilación.Antes de comenzar con el proceso de recopilación debo hacer referencia a la **Input box / Bandeja de entrada**. Es un concepto similar a la bandeja de entrada de nuestras aplicaciones de correo electrónico, donde se descargan todos los correos recibidos y posteriormente son leídos y procesados por el usuario. La Bandeja de entrada representará el espacio físico – o sobre soporte electrónico – donde depositamos todos los elementos pendientes, todas las ideas para un proyecto o cualquier cosa que ocupe nuestra mente. Puede tratarse de una bandeja para depositar documentos o una carpeta en nuestro ordenador, la clave es su ubicación: Un lugar de fácil acceso y presente durante nuestro tiempo de trabajo. Si no está al alcance de la mano puede que desistamos de colocar algún documento por pereza o por olvido, lo que nos hará perder la confianza en el sistema, provocando que volvamos a cargar nuestra mente con información sobre nuestros asuntos.

La recopilación es el proceso con el que traspasaremos de nuestra mente al sistema de listas de GTD todas las tareas pendientes y temas abiertos. Para empezar es indispensable disponer de una Entrada física, donde dejaremos todos aquellos papeles, correos, documentos e ideas-descritas sobre papel, para ser posteriormente procesados. La primera vez que realizamos la recopilación es probable que tardemos un tiempo considerable, hablamos de horas. Tengamos en cuenta que tendremos que revisar todos los rincones de nuestra memoria, de nuestros lugares habituales de trabajo – la oficina y en casa – y todos los emplazamientos donde guardamos documentación o material de consulta. Si no hemos realizado nunca una revisión similar, o hace tiempo que no verificamos nuestros archivos, nos sorprenderemos de todo lo que puede salir a la superficie. La revisión debe abarcar nuestros asuntos profesional y personales indistintamente ya que la mente no realiza distinción alguna a la hora de gestionar la memoria dedicada a nuestros compromisos.

Habrá temas que dispongan de una representación física, como una carta o un documento, tal vez un montón de facturas pendientes de clasificación o una invitación a un evento … Muchos otros, los que representan ideas para proyectos actuales / futuros, necesitarán ser transcritos de forma breve y clara sobre papel – o sobre el soporte electrónico que

utilizamos – para ser introducidos también en la Bandeja de Entrada para conservar una referencia física del asunto**.**

Una vez realizada la recopilación inicial, se realizará una cada semana. Será más ligera, pero igual de importante. Su objetivo principal sigue siendo el mismo: Descargar nuestra mente de cualquier asunto gestionable con el sistema de listas. En un segundo plano -no menos importante- tiene la función de mantener 100% operativo el sistema, evitando que volvamos a cargar en memoria detalles del día a día provocando una degradación de nuestra creatividad y nuestros recursos.

La función de la recopilación es vaciar nuestra RAM para poder dedicarse a otros fines**.** Una vez finalizada la recopilación daremos el segundo paso, el procesado de cada uno de los elementos de la entrada enviando a una de las lista del sistema GTD o en nuestro sistema de archivo.

Uno de los procesos más importantes es el de recopilación. Tenemos que crear un Input box / Bandeja de Entrada de ideas físico donde verteremos todos los temas pendientes, ideas, proyectos en curso, frentes abiertos, o cuestiones que tengamos pendientes. La idea es descargar todo de nuestra memoria RAM y traspasarlas a nuestro sistema de listas. La primera vez podemos quedarnos horas recopilando todos los elementos dispersos de nuestra mente, y de nuestro despacho.

5.1.1.1.2 Procesar

Una vez finalizado la recopilación, procesaremos cada una de las acciones siguiendo el esquema mostrado a continuación.

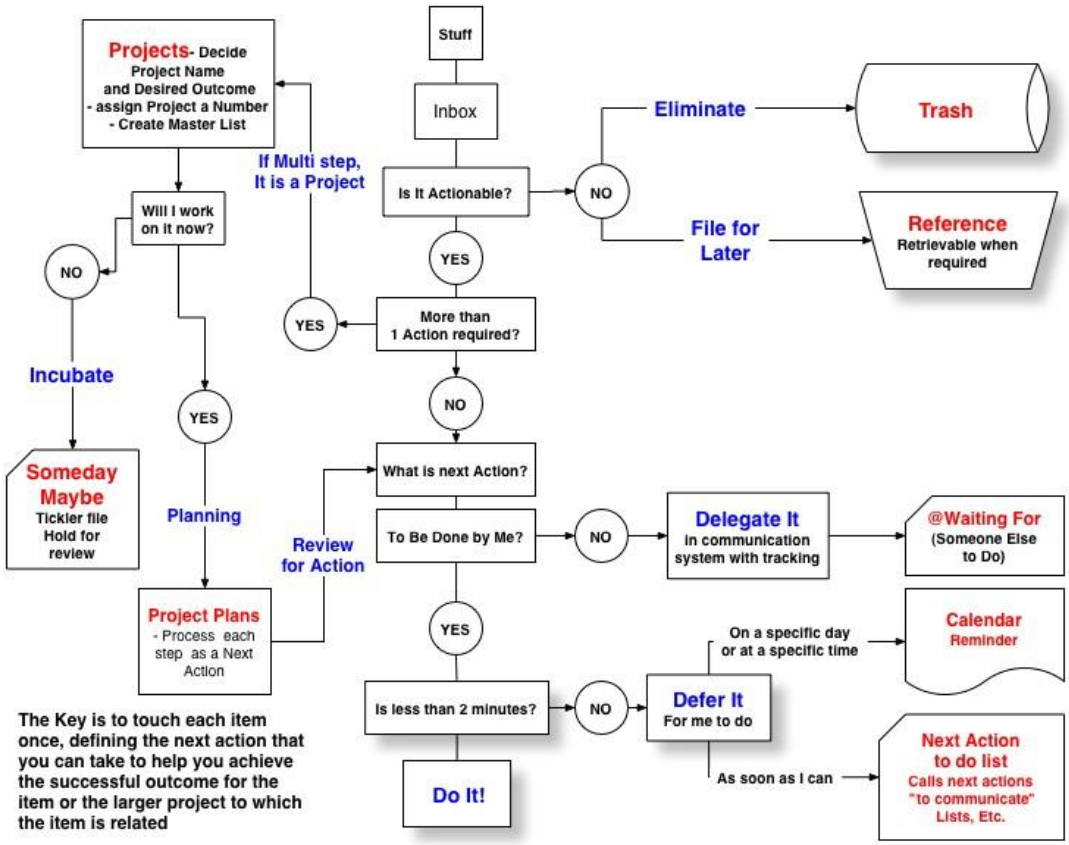


Figura 5.1.1.1.2.*1* Algoritmo Getting Things Done imagen tomada de la web freshpathconsulting cuya URL de la imagen es http://freshpathconsulting.files.wordpress.com/2009/11/gtd.jpg

Ejemplo: "Recoger un pedido importante en un almacén"

¿Necesitamos realizar una actividad para realizarla? Sí
¿Tenemos que hacerla nosotros? Sí
¿Nos lleva menos de dos minutos? No, no podemos hacerla en menos de 2 min
Asignarle una carpeta (fase 3: Organizar)

5.1.1.1.3 Organizar

Allen describe el conjunto de listas sugeridas que puedes usar para mantener un seguimiento de los elementos que esperan ser atendidos:

- **Bandeja de entrada**. Depositaremos todas las acciones o tareas que se nos presenten. En una segunda fase éstas se procesan, distribuyéndolas en el resto de carpetas del sistema, eliminándolas o archivándolas. Personalmente registro todas las ideas, problemas pendientes de resolución que se presenten, o potenciales proyectos a realizar. A pesar de ser una idea remota, la depósito en la bandeja de entrada, al procesarla ya lo enviaré a la lista Algún día, o abriré un nuevo proyecto definiendo las siguientes acciones a realizar. La cuestión es descargar nuestra mente para no perder ni un ápice de nuestra creatividad.

- **Siguiente**. Reservada para las acciones a realizar a corto plazo, durante el día o la semana. Se nutre tanto de tareas de la Entrada, como de las listas restantes. La única condición es que la acción se realice en breve.

  Para aumentar la eficiencia de la lista siguiente no hay que restringir a las listas base del sistema. Tenemos que pensar que si depositamos las acciones que pensamos realizar en la lista Siguiente es posible que no podamos realizarlas debido a limitaciones de materiales. No es una ubicación idónea o no disponer de las herramientas adecuadas puede obligarnos a aplazar las tareas. Para evitar encontrarnos en una situación de bloqueo como las mencionadas, crearemos un sistema paralelo de listas con nombres como Encargos, Oficina, Casa, Ordenador… Cada uno de ellos describe un emplazamiento, una herramienta o una actividad a realizar, en las que nos será útil tener agrupadas las acciones para poder realizar sin revisar todo el apartado de próximas acciones cada vez, sobre todo si administramos listas con una gran cantidad de puntos.

- **En espera**. Se enumeran los asuntos que necesitan la participación de un tercero para ser finalizados. Recibir una respuesta por correo, una llamada, una verificación de un trabajo realizado. Según nuestro trabajo puede quedar reducida en una anécdota, o si dependemos de terceras personas con frecuencia, convertirse en un punto de referencia. En cualquier caso tenemos que revisarla periódicamente según el uso: Diaria o superior si tiene un gran movimiento, o semanalmente si su uso es moderado.

- **Proyectos**. Abrimos una lista para cada uno de los proyectos en marcha. Consideremos un proyectos aquella tarea que consta de más de una acción física para ser completado, por lo tanto será fácil que acaben siendo numerosos. Es aconsejable eliminar las listas de proyectos al terminar la última acción. No debemos temer abrir listas de esta tipología, su gran número nos hará darse cuenta de la multitud de asuntos que tenemos entre manos.

- **Algún día**. es la lista donde se acumulan todas aquellas tareas y potenciales proyectos que no tienen fecha de entrega, que por falta de tiempo por el momento no se pueden realizar. Engloba un conjunto de acciones muy heterogéneo, desde migrar el blog hasta cambiar de coche, pasando por aprender francés. Para conservar su integridad dos consejos:

  No confundir Algún día con un cajón de sastre, si lo hacemos se acabará convirtiendo en un auténtico caos. No guardar tareas parcialmente realizadas o material de consulta, es mejor crear una carpeta para cada uno de los conceptos mencionados.

Un calendario también es algo importante para llevar un seguimiento de tus citas y compromisos; sin embargo, Allen recomienda específicamente que el calendario se reserve para lo que él denomina "paisaje yermo" : cosas que se deben hacer obligatoriamente en un plazo específico, o reuniones y citas que han sido fijados en un momento en particular. La clave definitiva de la organización según GTD es el sistema recordatorio. La sugerencia de Allen es que mantengas un único sistema recordatorio ordenado alfabéticamente para hacer tan rápido y fácil como sea posible el hecho de almacenar y buscar la información que necesitas.

5.1.1.1.4 Revisar

Las listas de acciones y recordatorios serán completamente inútiles si no las revisamos al menos diariamente o siempre que tengamos un momento libre. La disciplina GTD requiere que se revisen todas las acciones destacadas, proyectos e ítems “en espera”, asegurándose de que todas las tareas nuevas o eventos venideros están incluidos en el sistema y que estarán actualizado.

La revisión consiste en:

- ✓ Revisar todo lo que tienes pendiente por ejemplo: correo electrónico que ha quedado pendiente de contestar o que se ha separado para releer y contestar con más calma. Hay que mirar el calendario y añadir a la lista para procesar todas las citas de la próxima semana.
- ✓ Procesar tu bandeja de entrada física , convirtiendo todos los documentos, citas, correspondencias en nuevas entradas en tu bandeja de entrada virtual del software seleccionado.
- ✓ Revisión de las lista base de GTD
  1. Eliminar tareas que ya hemos realizado
  2. Reorganizar las tareas cuando haya cambiado nuestra prioridades
- ✓ El cuarto paso es la clave para la mejora de nuestros hábitos productivos. En la parte final de la revisión evaluaremos las decisiones tomadas durante la semana en el ámbito productivo. Repasamos cada uno

de los problemas y valoramos la solución aplicada, si ésta no ha sido del todo óptima proponemos una alternativa de cara a ir mejorando con el sistema.

5.1.2 Control de Tiempo

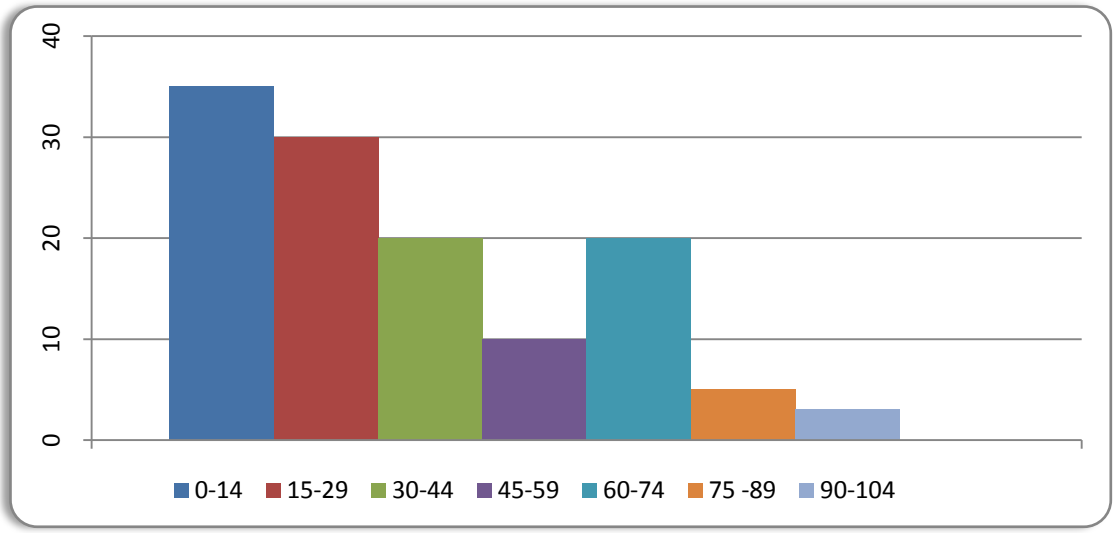
Este capítulo describe un procedimiento y una forma de controlar y registrar la manera de gastar tu tiempo. También se dan ejemplos de los tipos de registros de tiempos a guardar.

La formas de mejorar la calidad del trabajo comienza por entender lo que haces realmente. Esto significa que tiene que conocer las tareas que haces, cómo las haces y los resultados que obtienes.

El primer paso en este proceso es definir las tareas y averiguar cuánto tiempo gastas en cada una de ellas. Para hacer esto, debes medir tu tiempo real. Este capítulo describe cómo medir el tiempo y anotarlo en una tabla. Cuando registre el tiempo recuerda que el objetivo es obtener datos de cómo trabajas realmente. La forma y el procedimiento utilizado para reunir los datos no es tan importante mientras los datos sean exactos y completos.

5.1.2.1 Control del Tiempo

Cuando las personas hablan generalmente sobre lo que hacen, a menudo utiliza las horas como medida. Pero esto no es muy útil. La razón es porque tú, realmente, no dedicas una hora completa a algo. Por ejemplo en la figura de abajo mostramos un resumen de tiempos de 7 estudiantes tomando de un curso reciente en donde había que describir algún ejercicio de codificación relativamente pequeños. Estos ejercicios requiere una media de 4 a 6 horas cada uno, y los estudiantes tuvieron que controlar cuándo comenzaba y paraban su trabajo. Como puedes observar, el 90% de los períodos de trabajo de los 7 estudiantes fueron inferiores a una hora.



Nota: El eje de las Y representa los periodos (%) y el eje de la X representa las logitudes de los períodos de trabajo en minutos

La cantidad típica de tiempo no interrumpido que los ingenieros dedican a sus tareas es generalmente inferior a una hora. Medir el trabajo en unidades de horas no proporcionará

el detalle necesario para posteriormente planificar y gestionar tu trabajo. Es mucho más fácil controlar el tiempo en minutos que en fracciones de una hora. Consideran, por ejemplo, los registros de tiempo que obtendrías si utilizases fracciones de hora. Las entradas serian números como 0,38 o 1,25 horas. Aunque las unidades de horas pueden ser más útiles en resúmenes mensuales o semanales, estas cantidades fraccionadas son difíciles de calcular y complicado de interpretar. Por ejemplo, en vez de poner 0,38 horas, es más fácil entender y de registrar en 23 minutos.

Una vez que has decidido controlar el tiempo, es más fácil controlarlo en minutos que en horas.

5.1.2.2 Uso de Tabla de Registro de Tiempo Normalizada

La siguiente tabla de ejemplo es utilizada para registra el tiempo.

Estudiante:_____ Fecha_____

Profesor:_____ Clases_____

Fecha	Comienzo	Fin	Tiempo de interrupción	Tiempo Real	Actividades	Comentarios	C	U

Cuaderno de registro de tiempo

La parte superior de la tabla, llamada cabecera tiene unos espacios para poner tu nombre, fecha de comienzo, el nombre del profesor y el nombre o numero de clase. Las columnas del cuerpo de la tabla son para registrar los datos de tiempos. cada periodo de tiempo se introduce en una línea de la siguiente forma:

- Fecha. La fecha de realización de alguna actividad, como asistir a clase o escribir un programa.
- Comienzo. La hora de comienzo de la actividad.
- Fin. La hora que termina de hacer la actividad.
- Interrupción. Cualquier pérdida de tiempo debida a interrupciones.
- Tiempo Real. El tiempo a cada actividad en minutos entre los tiempo de comienzo y fin menos el tiempo de interrupción
- Actividad. Nombre descriptivo para la unidad.
- Comentarios. Una descripción más completa de lo que están haciendo, el tipo de interrupción o cualquier cosa que podrían ser útil cuando posteriormente analice los datos de tiempo.
- C(completado).Rellenar esta columna cuando termines una tarea como leer un capítulo del libro de texto o escribir un programa
- U(unidades).El número de unidades de una tarea acabada.

Un ejemplo de cómo rellanar la tabla de registro completo lo podemos ver a continuación.

Estudiante: **Estudiante Y** Fecha: **05/10/2013**

Profesor: **Señor Z** Clases: **CS1**

Fecha	Comienzo	Fin	Tiempo de interrupción	Tiempo Real	Actividades	Comentarios	C	U
05/10	9:00	9:50		50	Clase	Clase		
	12:40	1:18		38	Cod.	Ejercicio 1		
	2:45	3:53	10	58	Cod.	Ejercicio 1		
	6:25	7:45		80	Texto	Leer texto capítulo 1 y 2	x	2
06/10	11:06	12:19	6+5	62	Cod	Ejercicio 1, descanso, charla	x	1
07/10	9:00	9:50		50	Clase	clase		
	1:15	2:35	3+8	69	Cod	Ejercicio 2, descanso, charla	x	1
	4:18	5:11	25	28	Texto	Texto capítulo 3,charla con maría	x	1
08/10	6:42	9:04	10+6+12	114	Cod	Ejercicio 3	x	1
09/10	9:00	9:50		50	Clase	Clase		
	12:38	1:16		38	Texto	Texto capítulo 4		
10/10	9:15	11:59	5+3+22	134	Revisión	Prepara examen, descanso, teléfono, charla		

Cuaderno de registro de tiempo

Aquí el estudiante Y escribe su nombre y la fecha en que comienza a rellenar la tabla. También escribe el nombre del profesor y el nombre o número del curso.

Cuando comenzó a guardar registro tiempo el día 9 de septiembre, registro en la parte superior izquierda. Su primera tarea fue asistir a clase de ip a las 9:00 AM. La clase duró hasta las 9:50, o sea un total de 50 minutos. La estudiante puso 9:00 en la columna de comienzo, 9,50 en la columna de fin y los 50 minutos de duración en la columna de tiempo real. A la derecha puso de la actividad en la columna denominada actividad.

Finalmente, en la columna comentario, anotó lo que estaba haciendo. Como la estudiante Y continuo haciendo actividades durante varios días, anotó sus otras actividades para el curso de ip en la tabla de registro de tiempo.

5.1.2.3 Gestión de las Interrupciones

Un problema común con el control de tiempo son las interrupciones. Es sorprendente la frecuencia con que somos interrumpidos por llamadas telefónicas, personas que quieren charlar , molestias ocasionales o la necesidad de descansar. La forma de gestionar las interrupciones en el cuaderno de registro de tiempo, consiste en anotarlas en la columna

tiempo de interrupciones como se observa en la tabla siguiente la estudiante Y comenzó su trabajo de casa a las 11:06 del día 10 y paró a las 12:19.Durante dicho período, tuvo dos interrupciones, una de 6 minutos y otra de 5 minutos. El tiempo neto que dedicó a su trabajo de casa fue de 73 minutos menos 11 minutos de interrupción(tiempo improductivo),es decir un total de 62 minutos(tiempo productivo).

La estudiante no solamente anotó estos tiempo en el cuaderno de registro de tiempos sino que también describió brevemente las interrupciones en la columna de comentarios.

Una cosa que he encontrado útil es utilizar el cronómetro para controlar las interrupciones. Cuando me interrumpen, pongo en marcha el cronómetro y cuando estoy preparado para reanudar el trabajo, lo paro. Antes de comenzar a trabajar de nuevo, anoto el tiempo de interrupción en el cuaderno y pongo a cero el cronómetro. Encuentro esto más adecuado que apuntar el tiempo de inicio de cada interrupción.

Puesto que el tiempo de las interrupciones no es tiempo de trabajo productivo, se debe controlar las interrupciones. Los datos de tiempos registrado pueden utilizarse para comprender con qué frecuencia se interrumpe tu trabajo. Las interrupciones no son solamente un despilfarro del tiempo, sino que rompe tu ritmo de pensamiento, llevándote a la ineficiencia y al error. Comprender como eres interrumpido te ayudará a mejorar la calidad y eficiencia de tu trabajo.

5.1.2.4 Control de Tareas Finalizadas

Para controlar cómo gastar tu tiempo, necesitamos controlar los resultados producidos. Para la asistencia a clase o reuniones, por ejemplo, un registro de tiempo sería adecuado. Cuando desarrollas programas necesitas saber cuánto trabajo has realizado. Podrías calcular la productividad de la tarea. Un ejemplo podría ser cuanto tiempo necesitas para leer un capítulo de un libro o escribir un programa. Con este conocimiento podrás mejorar la planificación de tus futuros trabajos.

Las columnas C y U de la derecha del cuaderno de registro de tiempo, te ayuda a identificar rápidamente el tiempo dedicado a las distintas tareas y lo que has hecho.

Para rellenar la columna C comprueba cuando has terminado una tarea, por ejemplo el estudiante Y lee el libro desde 6:25 hasta las 7:45 el 5 octubre. Durante este tiempo completó la lectura de dos capítulos. Por ello puso una X en la columna C (complementado).Para anotar las unidades de trabajo acabadas, puso un 2 en la columna de U(unidades).

Posteriormente cuando resumas tu trabajo de la semana rápidamente podrías ver estas columnas para encontrar los elementos acabados y cuánto tiempo ha dedicado. Para tener unos registro de tiempo exactos, es importante completar las columnas C y U cada vez que acabes una tarea que tengas resultados medibles. Si olvidas hacerlo, puedes fácilmente encontrar la información, pero es mucho más fácil rellenarlo en el momento que acabas la tarea.

5.1.2.5 Resumen Semanal de Actividades

Para hacer un plan de periodo, es importante entender cómo gastas tu tiempo. El primer paso es registrar tu tiempo utilizando el cuaderno de registro de tiempos. Después de reunir los datos de tiempos una semana o dos , empezarás a ver cómo empleas el tiempo.

Puesto que los registro de tiempos son muy detallados para el propósito de la planificación, necesitas resumir los datos de una forma más útil. A continuación se muestra el formato de tiempo que son más adecuados para la planificación del período.

Nombre: Fecha:

1	Tarea									Total
2	Fecha									
3	D									
4	L									
5	M									
6	MI									
7	J									
8	V									
9	S									
10	totales									
11	Tiempos y Medidas del Período Numero de Semanas(número anterior +1)__									

12	Resumen de las semanas anteriores									
13	totales									
14	Med.									
15	Máx.									
16	Mín									

17	Resumen incluyendo la última semana									
18	Total									
19	Avg									
20	Máx									
21	Mín									

En la línea de 1-10 del resumen semanal de actividades, registras el tiempo que dedicas a cada actividad principal durante cada día de la última semana. en la parte inferior derecha desde la línea 13 a la 16 está la media el máximo y el mínimo que le dedicas a cada tarea durante la semana anterior al semestre. Desde la línea de la 18 a la 21 muestra el total la media, el máx., el min, de tiempo que le dedicas a cada trabajo definido para todo el semestre , incluyendo la última semana.

5.1.2.6 Resumen de Tiempos Semanales

Un ejemplo de resumen semanal de actividades se mostrará a continuación:

La forma de rellenar la tabla es de la siguiente manera:

1. Anota tu nombre y fecha
2. Anota las categorías de actividades como cabecera de las columnas.
3. Escribe fecha en la columna fecha, anotando el domingo como primer día de la semana
4. Para cada día suma todos los minutos para cada actividad en los registro de tiempo y anota el resultado en la columna correspondiente.
5. Después de completar todas las casillas para cada día, se obtiene el total para cada día en la columna total.
6. Repite este proceso para cada día de la semana (filas 5 -9)
7. Calcular los totales semanales para cada tarea clasificada en la fila 10.
8. A continuación se obtiene el total de los tiempos de las actividades que aparece en la fila 10.
9. Por último suma los totales para comprobar que obtiene el mismo resultado. Si no fuera así , se debería volver a calcular el total de todos los días y de todas las tareas para localizar el error.

Nombre: Estudiante Y Fecha: 05/10/2013

1	Tarea	Clase	Codificar	Preparar	leer					Total
2	Fecha		Programación	Exámenes	textos					
3	D		96		80					176
4	L	50	69		28					147
5	M		114							114
6	MI	50			38					88
7	J			134						134
8	V									0
9	S	50	62							112
10	totales	150	341	134	146					771
11	Tiempos y Medidas del Período -Numero de Semanas(número anterior +1) 2									

5.1.2.7 Calculo de los Tiempos y Media del Periodo

Un resumen sencillo de tus tiempos semanales sería suficiente si estuvieras únicamente interesado en una semana , pero realmente está interesado en los tiempos medios, máximos y mínimos dedicados a esta tareas durante un semestre o año completo. Para mostrar cómo obtener estos datos, complementamos a continuación el resumen semana de actividades mostradas en la tabla siguiente.

1. Anota el total de semanas transcurridas. Comprueba la fila 11 del resumen semanal de actividades de semanas anteriores para ver cuantas semanas se tuvieron en cuenta.

2. En las filas 13-16 de la tabla de la semana actual copia todas las entradas de la fila 18-21 de la tabla de la semana anterior
3. Suma los tiempos dedicados a cada tarea en la semana actual. En cada columna de la fila 18 anota el resultado de sumar las filas 13 y 10.
4. Calcula el tiempo medio dedicado semanalmente a cada tarea durante el semestre. En la fila 19 divide el valor de cada columna de la fila 18 por el número de semanas.
5. Para encontrar el mayor tiempo dedicado a una tarea durante una semana, calcula el valor máximo en la fila 20. comparando cada columna de la fila 15 con misma columna de la fila 10.Anota el valor mayor en la fila 20.
6. Para saber el tiempo que has dedicado a una tarea durante una semana, calcular el tiempo mínimo. Para calcular el valor mínimo de la columna 21, compara la columna de la fila 16 con la misma columna de la fila 10.Anota el valor más pequeño en la fila 21.
7. Observa que excepto para la primera semana, el tiempo total, máximo y mínimo serán diferentes.

12	**Resumen de las semanas anteriores**					
13	**totales**	150	341	134	146	771
14	**Med.**	150	341	134	146	771
15	**Máx.**	150	341	134	146	771
16	**Mín**	150	341	134	146	771
17	**Resumen incluyendo la última semana**					
18	**Total**	300	682	268	292	1542
19	**Med**	150	341	134	146	771
20	**Máx**	150	341	134	146	771
21	**Mín**	150	341	134	146	771

5.1.3 Lista de Características

En este Apartado se pretende redactar una lista de todas aquellas características que tiene la aplicación web que vamos a realizar.

5.1.3.1 Roles de Usuarios

En este paso realizaremos una clasificación de todos los posibles usuarios del sistema , es sólo una aproximación para identificar y asociar cada característica con cada uno de los usuarios.

Los diferentes usuarios identificados inicialmente son:

- ✓ **Usuarios registrados**:Es el usuario que ya consta en las bases de datos del servicio y posee pleno control sobre el software que se le asigna desde el momento de su registro.
- ✓ **Usuario no registrado (anónimo)**:Es el usuario que no se encuentra en el sistema, su única funcionalidad es ver la información de la página principal del sitio y poder registrarse en dicho sistema.
- ✓ **Administrador del sitio**: Es el rol de los usuarios que controlan el sitio web.

5.1.3.2 Valores de Planificación

Cada característica tiene también un conjunto de valores de planificación que son los incluidos a continuación:

- ✓ **Código**: Es el identificador de la característica. Se especifica como: LC + Categoría + Número.
- ✓ **Nombre de las características**.
- ✓ **Descripción**: Breve descripción de lo que comprende la característica.
- ✓ **Prioridad**: Se asigna una prioridad a cada una con el fin de determinar el orden en que se van a ir desarrollando. Las prioridades pueden ser:
  - Muy Alta (81-100)
  - Alta (61-80)
  - Media (41-60)
  - Baja (21-40)
  - Muy Baja (1-20)
- ✓ **Estado**: Cada característica tiene un estado asociado que irá variando a medida que progrese el proyecto. Los estados pueden ser:
  - ***Aceptado*:** La característica se desarrollará en esta versión del producto.
  - ***Planificada*:** La característica ya ha sido planificada y se empezará a desarrollar en un plazo de tiempo corto.
  - ***En desarrollo*:** Ya se está desarrollando.
  - ***Finalizada*:** Se ha terminado de desarrollar.
  - ***Postergada*:** No se desarrollará hasta una versión futura.
  - ***Rechazada*:** Probablemente no se desarrollará en ninguna versión.
- ✓ **Rol**: El rol de usuario que puede usar la característica.
- ✓ **Coste Estimado**: el coste en una medida (tiempo, dinero, recursos) que conlleva realizar la característica.
- ✓ **Riesgo**: Cada característica puede tener asociado un riesgo que representa la dificultad para conseguir implementarla correctamente. Se utilizarán tres niveles:
  - ***Crítico*:** Riego que se debe corregir lo antes posible.
  - ***Significativo*:** Es un riesgo con cierta importancia que tendrá que ser mirando
  - ***Rutinario*:** Riesgo con el nivel más bajo que contiene cada característica por defecto.

5.1.3.3 Catalogación de la Lista de Características

Para una mayor facilidad en la catalogación, la lista de características está dividida por categorías que representan una aproximación a los módulos que compondrán el sistema.

Debido a que la metodología que se usa es iterativa incremental, durante la primera iteración se desarrollarán aquellos requisitos con mayor prioridad y de estado aceptado.

Al final del proceso,éstos requisitos pasarán a estar en estado finalizado para dar comienzo a la planificación de la siguiente iteración.

5.1.3.3 .1-Navegación-Web Info (A)

LC-A1: Página de Inicio de la aplicación
Se debe mostrar una página web como interfaz de inicio. Aquí es donde el usuario llega al acceder a la aplicación por primera vez.Se hace referencia a las principales funciones de la aplicación y se enlaza a los formularios de registro y acceso.
Prioridad-Muy alta (99)
Estado- Aceptado
Rol- Usuarios no registrado (anónimo),Usuario registrado,Usuario administrador
Coste
Riesgo- Rutinario

LC-A2: Tour de la aplicación
Se debe mostrar una página web como interfaz de entrada con un tour de las características o funcionalidades de la aplicación, mostrando todas las cualidades destacadas de la misma. Se incluyen páginas acerca de cada uno de los módulos que se gestionan y capturas de pantalla de las mismas
Prioridad- Muy alta (98)
Estado-Aceptado
Rol- Usuarios no registrado (anónimo),Usuario registrado,Usuario administrador
Coste
Riesgo-Rutinario

LC-A3: Blog de la aplicación
Se debe mostrar una página web como interfaz de entrada con un blog que será regentado por los administradores de la aplicación en donde se mostrará las noticias o características o funcionalidades nueva de la aplicación.
Prioridad- Muy alta (97)
Estado-Aceptado
Rol- Usuarios no registrado (anónimo),Usuario registrado,Usuario administrador
Coste
Riesgo-Rutinario

LC-A4: FAQ de la aplicación
Se debe mostrar una página web como interfaz de entrada en donde se muestre las preguntas más frecuante que suelen preguntar los usuarios sobre esta web y sobre la aplicación.
Prioridad- Muy alta (96)
Estado-Aceptado
Rol- Usuarios no registrado (anónimo),Usuario registrado,Usuario administrador
Coste
Riesgo-Rutinario

LC-A5: Contact us de la aplicación
Se debe mostrar una página web como interfaz de entrada con un apartado donde pueda ponerse en contacto los usuarios con nuestra compañía, en dicha página habrá dos secciones un formulario de contacto y información sobre teléfonos y direcciones de nuestra compañía.
Prioridad- Muy alta (95)
Estado- Aceptado
Rol- Usuarios no registrado (anónimo),Usuario registrado,Usuario administrador
Coste
Riesgo-Rutinario

LC-A6: Work desk de la aplicación
Se debe mostrar una página web como interfaz interna de la aplicación , está página será la página central de trabajo de la aplicación,aquí el usuario podrá usar todas las funcionalidades de gestión de task, project,GTD, interruption etc.
Prioridad-Muy alta (95)
Estado- Aceptado
Rol- Usuarios no registrado (anónimo),Usuario registrado,Usuario administrador
Coste
Riesgo-Rutinario

5.1.3.3 .2-Registro, Logueo y Usuarios (B)

LC-B1: Registrarse
Un cliente podrá Registrarse para convertirse en un usuario del sistema.(Usuario registrado) para ello tendrá que rellenar un formulario de inscripción.El formulario consta de varios campos como el e-mail, la contraseña...
Prioridad-Muy alta (94)
Estado-Aceptado
Rol-Usuarios no registrado (anónimo)
Coste
Riesgo- Crítico

LC-B2: Loguearse
Una vez registrado el usuario podrá entrar en el sistema a través de esta sección y utilizar todas las funcionalidades del software.Para poder acceder a la parte interna de la aplicación el usuario deberar introducir el email y la contraseña.
Prioridad-Muy alta (93)
Estado-Aceptado
Rol- Usuarios registrado, Usuarios administradores
Coste
Riesgo- Crítico

LC-B3: Recuperación de contraseña
Se le da la posibilidad al usuario que una vez registrado el usuario podrá recuperar la contraseña del sistema olvidada introduciendo su email.
Prioridad-Muy alta (93)
Estado-Aceptado
Rol- Usuarios registrado
Coste
Riesgo- Significativo

LC-B4: Cerrar sesión
Los usuarios podrán cerrar la sesión iniciada cuando deseen apretando el botón de sign out.
Prioridad-Muy alta (92)
Estado-Aceptado
Rol- Usuarios registrado
Coste
Riesgo- Significativo

LC-B5: Acceso a work desk
El sistema debe permitir a los usuario ir a la página work desk(escritorio de trabajo). En ella se muestran las opciones todas las opciones principales de la aplicación donde el usuario podra gestionar, task, project, interruption..
Prioridad-Muy alta (93)
Estado-Aceptado
Rol- Usuarios registrado
Coste
Riesgo- Crítico

LC-B6: Configuración de cuenta
Los usuarios que accedan al sistema pueden modificar su cuenta. Se trata de modificar datos insertados en el registro de la cuenta como email,contraseña...
Prioridad-Muy alta (93)
Estado-Aceptado
Rol- Usuarios registrado
Coste
Riesgo- Significativo

LC-B7: Cancelar cuenta
Los usuarios que accedan al sistema pueden cancelar su cuenta,apartir de este momento la cuenta asociada al usuario quedará inhabilita.
Prioridad-Muy alta (93)
Estado-Aceptado
Rol- Usuarios registrado
Coste
Riesgo- Significativo

5.1.3.3 .3-Gestión de Tareas (C)

LC-C1: Gestión de tareas
El usuario podrá gestionar sus tareas realizando acciones como añadir, eliminar o modificar tareas,ver tareas y buscar tarea Las búsquedas pueden ser realizarán, por el nombre de la tarea o el estado de las tareas si esta en estado de pausa, empezada...
Prioridad- Muy alta (98)
Estado-Aceptado
Rol- Usuarios registrado
Coste
Riesgo-Crítico

LC-C2: Enviar tareas por correo
El sistema debe permitir que los usuarios puedan enviar tareas por correo a otras personas.
Prioridad- Media (44)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-C3: Clonar tareas
El sistema permitirá una funciones de clonar una tarea existente, en donde se le aplicará a la nueva tarea todos los campos de la tarea orígen.
Prioridad- Media (43)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-C4: Ordenar tareas
El sistema permitirá que el usuario pueda ordenar tareas por fecha de creación , por fecha de vencimiento, por estado del ciclo de vida de la tarea,por nombre , por nota, etc.
Prioridad- Media (42)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-C5: Ejecucción de la tarea
El sistema permitirá que el usuario pueda gestionar el ciclo de vida de una tarea, empezando la tarea, interrumpiendo la tarea o finalizando dicha tarea.
Prioridad- Muy alta (91)
Estado- Aceptada
Rol- Usuarios registrado
Coste
Riesgo-Significativo

5.1.3.3 .4-Gestión de Proyectos (D)

LC-D1: Gestión de Proyectos
El usuario podrá gestionar sus proyectos realizando acciones como añadir, eliminar ,modificar proyectosk, ver poryectos o bucar proyecto, las realizaciones de las busqueda de proyecto serán por el nombre.
Prioridad- Muy alta (98)
Estado-Aceptado
Rol- Usuarios registrado
Coste
Riesgo-Crítico

LC-D2: Gestión y manipulación de subproyectos (subcarpetas)
El sistema debe permitir al usuario crear subproyectos dentro de un proyecto padre.Además el sistema debe permitir El sistema debe permitir mover proyectos dentro de un proyecto, así como mover tareas dentro de un proyecto.
Prioridad- Muy Alta (96)
Estado- Aceptada
Rol- Usuarios registrado
Coste
Riesgo-Crítico

LC-C3: Ordenar Proyectos (Carpetas)
El sistema permitirá que el usuario pueda ordenar proyectos por fecha de creación , por título, fecha de creación,nivel.
Prioridad- Muy Alta (96)
Estado- Aceptada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-C4: Crear proyecto privados con contraseñas o públicos
El sistema dará la posibilidad de que el usuario pueda crear proyectos públicos o proyecto privados donde requiera de una contraseña.
Prioridad- Media (43)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-C5: Crear diagrama de gantt asociado al trabajo de un proyecto
El sistema dará la posibilidad al usuario de que pueda crear diagramas de gantt para asociarlo a los proyectos y poder ver un seguimiento del mismo.
Prioridad- Media (45)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-C6: Gestión diagrama de gantt de los Proyectos
El sistema dará la posibilidad al usuario de que pueda gestionar los diagramas de gantt que son asociados a los proyectos con operaciones básicas de crud
Prioridad- Media (45)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-C7: Gestión de informes de proyecto
El sistema dará la posibilidad al usuario de que pueda gestionar los informe que son asociados a los proyectos y a los diagramas de gantt generados usando para ello operaciones de control de crud.
Prioridad- Media (46)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-C8: Guardar los diagramas o informes en diferentes formatos
El sistema dará la posibilidad al usuario de que pueda guardar en diferentes formatos tanto los informes como los diagrama de gantt asociados a los proyectos, los formatos que se permitiran será pdf, word, jpg.
Prioridad- Media (42)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-C9: Exportación de los diagramas o informes
El sistema dará la posibilidad al usuario de que pueda exportar tanto los diagramas como los informes, la exportación podrá hacerse de forma directa a su ordenador o a traves de un medio de almacenamiento en la nube como puede ser dropbox,drive,box.
Prioridad- Media (47)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-C10: Importación de los diagramas o informes
El sistema dará la posibilidad al usuario de que pueda importar tanto los diagramas como los informes, la importación podrá hacerse de forma directa desde su ordenador o a traves de un medio de almacenamiento en la nube como puede ser dropbox, drive, box.
Prioridad- Media (46)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-C11: Roles de usuarios por proyecto
El sistema dará la posibilidad de asociar diferentes roles dentro de un proyecto a diferentes usuarios del sistema que esten asociado a un grupo de trabajo.
Prioridad- Media (48)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

5.1.3.3 .5-Gestión de Tiempo(E)

LC-E1: Gestión contador de tiempo de interrupciones
El usuario podrá gestionar sus interrupciones asociadas a las tareas realizando acciones como añadir, eliminar, buscar, o modificar dichas interrupciones.
Prioridad- Muy alta (97)
Estado-Aceptado
Rol- Usuarios registrado
Coste
Riesgo- Crítico

LC-E2: Ejecucción de interrupciones
El sistema debe permitir al usuario gestionar el ciclo de vida de las interrupciones,usando para ellos un control de parada o retorno de la tarea previamente en marcha.
Prioridad- Muy Alta (96)
Estado- Aceptada
Rol- Usuarios registrado
Coste
Riesgo-Crítico

LC-·E3: Gestión de gráficos o informes de seguimiento de interrupciones
Se le dará la posibilidad al usuario de que pueda gestionar informes o gráficos para el seguimiento de las interrupciones.La gestión estará compuesta por acciones de creación,edición,eliminación y listado de informes o gráficos.
Prioridad- Muy Alta (96)
Estado- Aceptada
Rol- Usuarios registrado
Coste
Riesgo-Significativo

LC-E4: Exportación de los diagramas o informes de interrupciones
El sistema dará la posibilidad al usuario de que pueda exportar tanto los diagramas como los informes, la exportación podrá hacerse de forma directa a su ordenador o a traves de un medio de almacenamiento en la nube como puede ser dropbox,drive,box.
Prioridad- Media (47)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-E5: Importación de los diagramas o informes de interrupciones
El sistema dará la posibilidad al usuario de que pueda importar tanto los diagramas como los informes, la importación podrá hacerse de forma directa desde su ordenador o a traves de un medio de almacenamiento en la nube como puede ser dropbox, drive, box.
Prioridad- Media (46)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-E6: Gestión de hoja de asistencias
El sistema dará al usuario la posibilidad de que pueda gestionar hojas de asistencias de diferentes usuarios que estén relacionado con un grupo de trabajo.Las acciones de control serán las básicas:creación,modificación,eliminación,visualización
Prioridad- Media (45)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo-Rutinario

LC-E7: Gestión de eventos del día a través de de un calendario
Se le dará la posibilidad al usuario de que pueda gestionar un calendario para las citas del día.Las gestión será una gestión básica de creación,modificación,eliminación,visualización
Prioridad- Muy Alta (96)
Estado- Aceptada
Rol- Usuarios registrado
Coste
Riesgo-Significativo

LC-E8: Alarma de fecha de vencimiento de tareas, proyectos o citas.
El sistema dará al usuario aviso de vencimiento de tareas, proyectos o citas.
Prioridad- Alta (81)
Estado- Planificada
Rol- Usuarios registrado
Coste
Riesgo-Significativo

5.1.3.3 .6- Getting Things Done (GTD) (F)

LC-F1: Colocación de tareas en carpeta inicial INBOX
El usuario colocará toda tarea nueva, o proyecto creado en un proyecto padre inicial llamado INBOX cuyo simnolo será "/" representando la raiz de directorio de archivo de los sistemas operativos.
Prioridad- Muy alta (99)
Estado- Aceptada
Rol- Usuarios registrado
Coste
Riesgo-Crítico

LC-F2: Gestión del algoritmo GTD
El usuario gestionará los proyectos y las tareas siguiendo la filosofía gtd para ello se dispondrá de funcionalidad como: resolución de tareas inferioes a 2 minutos, organización de tareas superiores a 2 minutos, aviso cada 24 horas de revisión de prioridades sobre las tareas.
Prioridad- Muy alta (98)
Estado- Aceptada
Rol- Usuarios registrado
Coste
Riesgo-Crítico

LC-F3:Filtrados de carpetas siguiendo la filosofía GTD
El sistema dará la posibilidad al usuario de filtrar las tareas siguiendo la filosofía GTD de si una tarea esta en espera, no es importante, se tiene que hacer de inmediato etc.
Prioridad- Muy alta (98)
Estado- Aceptada
Rol- Usuarios registrado
Coste
Riesgo-Crítico

LC-F4:Asistente de gestión de la filosofía GTD
El sistema dará la posibilidad al usuario de usar un asitente para aprender la filosofía GTD.
Prioridad- Alta (81)
Estado- Planificada
Rol- Usuarios registrado
Coste
Riesgo-Significativo

5.1.3.3 .7- Gestión de Colaboración de Equipos.(G)

LC-G1: Construcción de equipo de trabajos
Una de las funcionalidad de este software será que los usuarios puedan crear equipos de trabajos y autogestionarse .
Prioridad- Baja(39)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-G2: Gestión de usuario en los equipos de trabajo
El sistema dará la posibilidad de una gestión básica de usuario dentro de los equipos de trabajo. Las funcionalidades serán: edición de equipo,eliminación de equipos,Listado de personas que forman el equipo,permisos a los miembros del equipo,asi como roles entre los el personal del equipo.
Prioridad-Baja(38)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-G3: Gestión de archivos compartido en grupos de trabajos
El sistema dará la posibilidad de una gestión básica de compartimiento de archivos Las funcionalidades serán: subir fichero,compartir fichero,descargar fichero
Prioridad-Baja(37)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-G4: Un administrador de versiones de archivos
El sistema dará la posibilidad de un administrador de versiones de archivos básico, donde se almacene, datos como fecha de modificación, autor de la modificación, recuperación de una versión anterior
Prioridad-Baja(36)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-G5: Gestión de notas en grupo
El sistema dará la posibilidad de una gestión básica de notas en grupo, las funcionalidades será postear notas en un tablón de anuncio, crearlas, editarlas, eliminarlas, responder una nota en particular etc.
Prioridad-Baja(35)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-G6: chat
El sistema dará la posibilidad al usuario de que pueda chatear con las personas de un grupo de trabajo
Prioridad-Baja(37)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-G7: Correo electrónico
Se le dará la posibilidad al usuario de que pueda envía correo electrónicos a los componentes del grupo o a cualquier persona simplemente usando nuestro entorno de correo electrónico.
Prioridad-Baja(36)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-G8: Notificaciones
El software le dará al usuario una característica de notificaciones tanto para las notificaciones de grupo como para las notificaciones del software
Prioridad-Baja(35)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-G9: Video conferencia
El usuario podrá realizar video conferencias con otros usuario que esté en su grupo de trabajo o esté almacenado en su agenda personal.
Prioridad-Baja(34)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-G10: foro
Posibilidad que se le dará al usuario de poder crear un foro en donde pueda crear temas, subtemas, etc.
Prioridad-Baja(36)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-G11: wikis
Posibilidad para crear wikis, páginas dentro de la wiki, creación de enlaces, etc.
Prioridad-Baja(35)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

5.1.3.3 .8- Configuración(H)

LC-H1: Cambiar el color de la página interna(work desk)
Se le dará la posibilidad en esta sección de configurar varias cosas del entorno. En esta opción en particular será cambiar el color del escritorio de trabajo (página interna principal)
Prioridad-Baja(39)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-H2: Cambiar los datos de cuenta
Aquí el usuario tendrá la posibilidad de cambiar los datos personales de su cuenta.
Prioridad-Baja(38)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-H3: Cambiar posición de menús
Con esta opción el usuario podrá cambiar las posiciones de los menús a un distinto orden del preestablecido de base.
Prioridad-Baja(37)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-H4: Cambiar idiomas
El usuario podrá cambiar a diferentes idiomas la página entera desde su contenido hasta sus herramientas. El software original vendrá con el idioma inglés de base.
Prioridad-Baja(36)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-H5: Activar notificaciones por correo
El usuario podrá si lo desea activar las notificaciones por correo, en cuyo caso cuando la herramienta notificaciones tenga nuevas entradas se le mandará un correo al usuario con las notificaciones producidas.
Prioridad-Baja(35)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-H6: Ocultar cuenta de correo
El usuario podrá ocultar su cuenta de correo electrónico para que otros usuario no la vean
Prioridad-Muy baja(15)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-H7: Cambiar zona horaria
Al usuario se le dará la posibilidad de cambiar la zona horaria según le convenga.
Prioridad-Muy baja(17)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

5.1.3.3 .9- Gestión de Pagos (I)

LC-I1: Comprar un nuevo paquete de características para la aplicación
El usuario podrá comprar nuevas características para el software de base. Al comprarlas se activará esta nuevas características con los cual ya podrá disfrutar de ellas.
Prioridad-Muy baja(17)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-I2: Comprar más espacio
Se le dará la opción al usuario de comprar más espacio del preestablecido de base. El espacio preestablecido de base es de 1GB.(Espacio de cuenta gratuita)
Prioridad-Muy baja(16)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-I3: Pagar cuota mensual al software de pago(software con más opciones y espacio de base)
Aquí el usuario tendrá la posibilidad de pagar por el espacio comprado. Además el usuario también podrá pagar en esta sección, la cuota mensual por la versión de pago(la versión de pago contiene todas las funcionalidades, y además se le concederá un espacio de 50GB)
Prioridad-Muy baja(16)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-I4: Gestión de forma de pago
Se le dará la opción al usuario de poder pagar con diferentes opciones entre las que se encuentra, paypal, cuenta bancaria,tarjeta de crédito, o a través de tarjetas de dinero canjeables,etc.
Prioridad-Muy baja(14)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

5.1.3.3 .10- Gestor para Creación de Documentos en Línea (J)

LC-J1: Gestión de documentos en línea
El sistema implementará un gestor de documentos en línea,podrás crear diferentes documentos en línea, modificarlos, eliminarlos, vizualizarlos
Prioridad-Muy baja(14)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-J2: Gestión de herramientas de formatos
El sistema implementará un gestor de herramientas de formatos como por ejemplo: operaciones básicas cursiva, negrita, subrayado,color de texto, justificación de texto, insertar tablas, fotos, pie de página, márgenes, orientación, tamaño, columnas, temas predefinidos de fábrica.
Prioridad-Muy baja(13)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

5.1.3.3 .11- Rastreos y Corrección de Bugs (K)

LC-K1: Log de errores producidos en el programa
Se creará un fichero de log de los errores producidos en el programa en el trascurso de su uso, también aparecerá un registros de los cambios o acciones realizadas por los usuarios que utilice el software
Prioridad-Muy baja(10)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-K2: Rastreo y corrección de bugs
Se creará una característica para rastrear los posibles bugs que se han producidos en el software y no han aparecido en el fichero de log de errores generales.Además otra característica de corrección de errores para poder solucionar los errores producidos a lo largo del funcionamiento del software.
Prioridad-Muy baja(9)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-K3: Reportar bug a los administradores del software
Se dará una característica de reportar a los creadores del software los bug encontrados, permitiendo así poder ayudar en futuras versiones del software.
Prioridad-Muy baja(8)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

5.1.3.3 .12- Móvil (L)

LC-L1: Creación de software para el sistema operativo android, IOS.
Se creará este mismo software para el sistema operativo android y para el sistema operativo IOS
Prioridad-Muy baja(7)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-L2: Sincronización con el móvil
Se permitirá que el usuario pueda sincronizar sus datos con la versión de este software para móvil
Prioridad-Muy baja(6)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

LC-L3: Envío de notificaciones al móvil
Se le permitirá al usuario que pueda recibir notificaciones en la versión de móvil
Prioridad-Muy baja(5)
Estado- Postergada
Rol- Usuarios registrado
Coste
Riesgo- Rutinario

5.1.4 Modelo del Dominio

En este subcapítulo se realiza el modelo de dominio. El modelo del dominio se utiliza con frecuencia como fuente de inspiración para el diseño de objetos software.

En UML, un modelo de dominio se representa con un conjunto de diagramas de clase en los que no se define ninguna operación.

Pueden mostrarse:

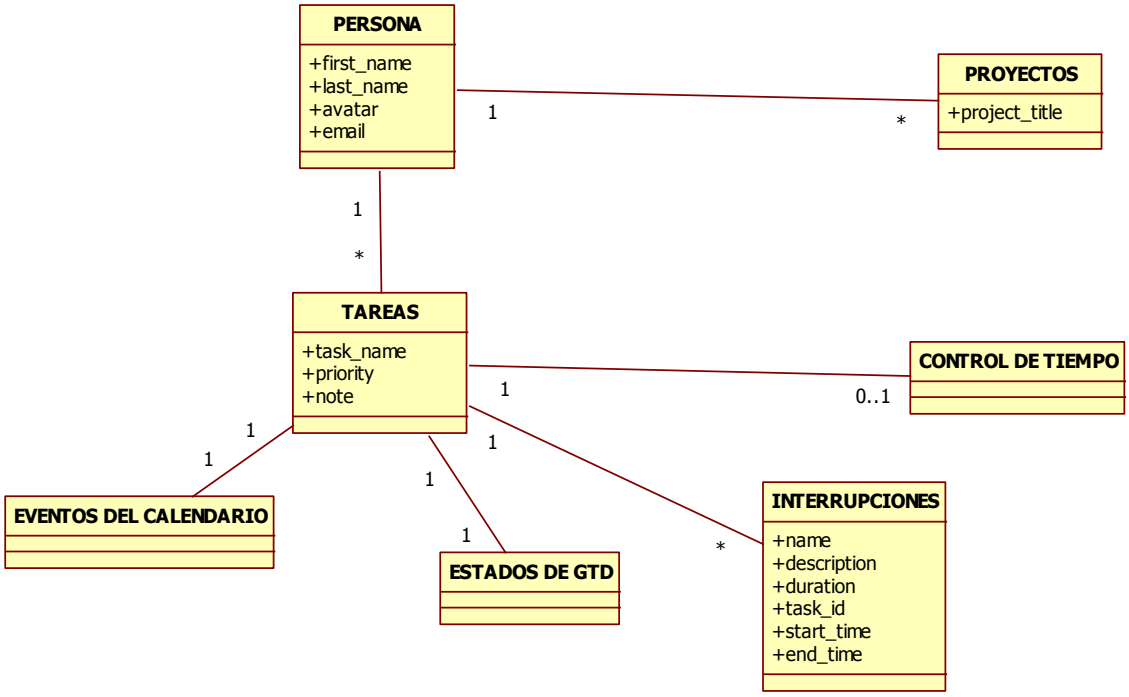
- Objetos de dominio o clases conceptuales
- Asociaciones entre clases conceptuales
- Atributos de las clases conceptuales

El modelo del dominio muestra a los modeladores las clases conceptuales o vocabulario del dominio significativas en un dominio del problema; es el artefacto más importante que se crea durante el análisis orientado a objetos. Informalmente, una clase conceptual es una idea, cosa u objeto. Más formalmente, una clase conceptual podría considerarse en términos de su símbolo, intensión, y extensión:

- **Símbolo:** Palabras o imágenes que representan una clase conceptual.
- **Intensión:** La definición de una clase conceptual.
- **Extensión:** El conjunto de ejemplos a los que se aplica la clase conceptual

Por tanto, una tarea primordial de la fase de análisis consiste en identificar varios clases conceptuales del dominio y documentar los resultados en un modelo conceptual. Una cualidad que debe ofrecer un modelo conceptual es que representa cosas del mundo real, no componentes del software.

A continuación se mostrará el modelo de dominio creado para este trabajo final de grado.



5.2 Requisitos del Software

Hace dos capítulos se obtuvo la lista de características, ésta lista nos ayuda a tener un esbozo de cuáles pueden ser las necesidades a cubrir con la aplicación. A medida que se realizaba, se fue analizando y viendo la importancia ligada al valor que aportan en base al modelo de negocio.

El presente capítulo presenta estas características en forma de casos de uso. Para obtener una descripción de la arquitectura, hay que priorizar los casos de uso y así determinar cuáles son necesarios para el desarrollo ,es decir, aquellos que se llevarán primero a las fases de análisis, diseño, implementación. Los demás casos de uso se dejarán para iteraciones posteriores.

Los casos de uso muestran el camino a través de los requisitos, análisis, diseño, implementación y pruebas para producir un sistema. Con el modelo de casos de uso y diagramas de casos de uso asociados como punto de comienzo, se puede describir cada caso de uso en detalle.

Para ello se hace una especificación precisa de la secuencia de acciones necesarias para realizar el caso de uso. Se obtiene una descripción detallada de las diferentes funcionalidades del sistema, así como la definición de la interacción de cada uno de los actores que utilizan el sistema con las diferentes secuencias de acciones que pueden realizar.

El proceso seguido para alcanzar los objetivos reseñados en los párrafos anteriores es el siguiente:

1. Definición de actores del sistema
2. Creación de modelo de casos de uso para cada paquete que forma el sistema
3. Especificación detallada de casos de uso

Todo éste proceso utilizado en éste capítulo se llevará a cabo ya que RUP se basa fundamentalmente en los casos de uso como base para establecer la arquitectura del proyecto, proporcionándonos así una clara perspectiva del sistema completo.

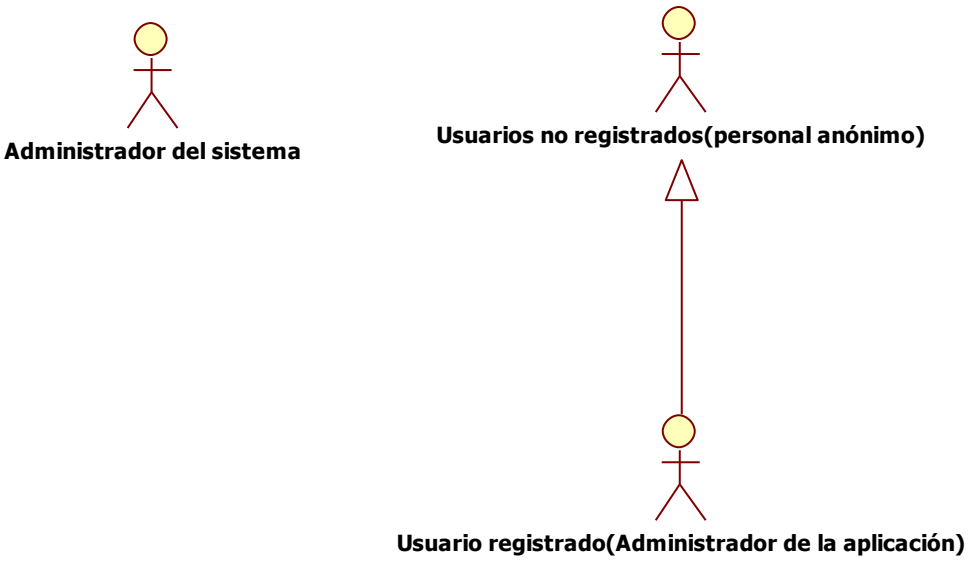
5.2.1 Actores del Sistema

El modelo de caso de uso describe lo que hace el sistema para cada tipo de usuario. Cada uno de éstos usuarios se representa mediante uno o más actores. Los actores suelen corresponder con trabajadores de un negocio. Cada rol de un trabajador define lo que hace el trabajador en un proceso de negocio concreto, es decir, los roles que desempeña un trabajador pueden emplearse para obtener los roles que cumple el actor del sistema correspondiente. Se dota a cada trabajador con un caso de uso del sistema para cada uno de sus roles.

A continuación se va a definir los diferentes actores que participan en los casos de uso del sistema.

Este proyecto tiene tres tipos de usuarios diferenciados:

- **Usuario registrado(Administrado de la aplicación).** Este tipo de actor acaparan prácticamente la totalidad de funcionalidades y acciones que puede desarrollar el sistema. Recordar, simplemente, que la diferencia entre estos roles radica en las limitaciones de acceso a ciertas funcionalidades del sistema definidas.
- **Usuario no registrado(Personal anónimo).** Es el actor para quien va dirigida la aplicación. Nótese que este tipo de actor únicamente puede acceder a la interfaz pública de la aplicación
- **Administrador del sistema.** Es quien actúa como administrador de la aplicación desarrollada. La función es la de atender a los problemas, se encargará del blog.



5.2.2 Modelo de casos de uso

Un modelo de casos de uso es un modelo del sistema que contiene actores, casos de uso, sus relaciones y las especificaciones. El modelo de casos de uso permite que los desarrolladores de software y los clientes lleguen a un acuerdo sobre las condiciones y posibilidades que debe cumplir el sistema.

El modelo de caso de uso proporciona la entrada fundamental para el análisis, el diseño y las pruebas.UML nos permite presentar el modelo en diagramas que muestran los actores y los casos de uso desde diferentes puntos de vista y con diferentes propósitos.

Para abordar este trabajo de una forma más ordenada se ha decidido crear paquetes que contendrán los diferentes diagramas de casos de usos.

5.2.2.1 Lista de Paquetes de los casos de uso

El sistema se divide en diversos paquetes que serán identificados mediante una clave o nombre, para posteriormente describir los casos de usos que forman dichos paquetes.

Los paquetes que se han definidos son los siguientes.

Nombre	Descripción
DCU-PGAG	Paquete de gestión de actividades generales
DCU-PGCP	Paquete de gestión de configuración del perfil
DCU-PGT	Paquetes de gestión de tareas
DCU-PGP	Paquete de gestión de proyectos
DCU-PGI	Paquete de gestión de interrupciones
DCU-PGGTD	Paquete de gestión metodología GTD

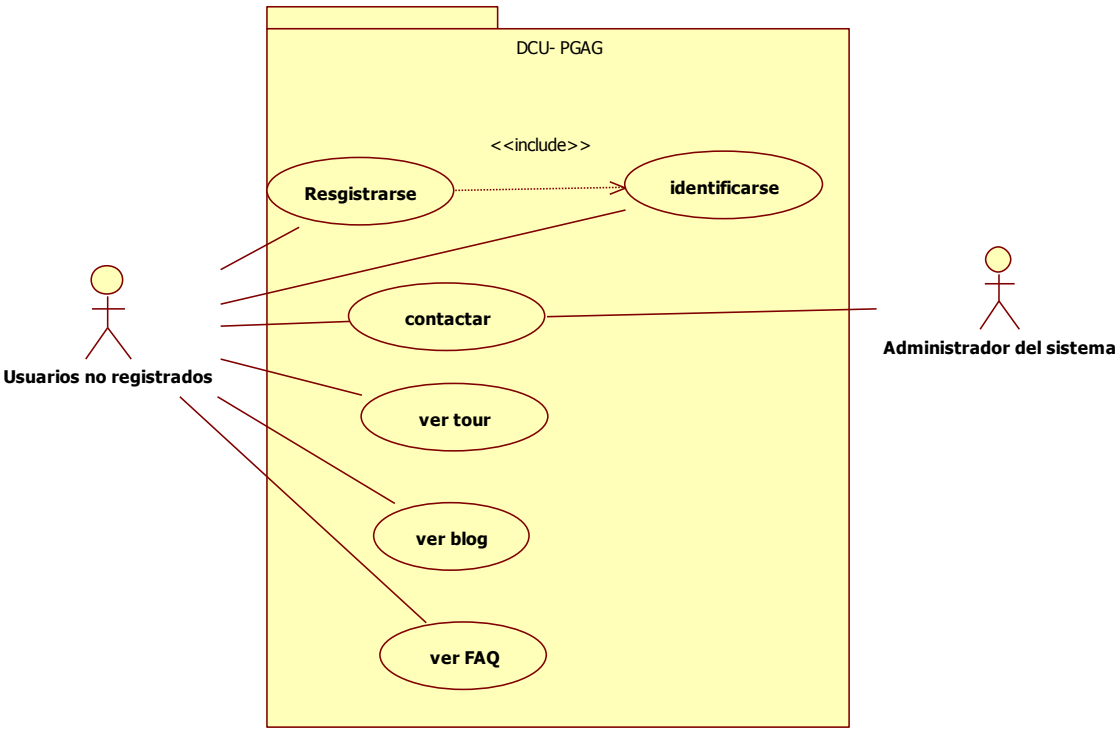
Lista de paquetes de casos de uso del sistema

A continuación se procederá a definir las relaciones que se establecen entre los actores del sistema y los diferentes casos de uso. Estos diagramas nos dará de una manera clara a que funcionalidades del sistema puede acceder cada tipo de usuario.

5.2.2.1.1 Paquete de Gestión de Actividades Generales

El paquete de gestión de actividades contiene los casos de uso que hacen referencia a la interfaz pública y a la identificación en el sistema de los usuarios registrado y no registrados. Específicamente, las acciones que puede hacer el usuario no registrado son:

- **Registrarse e identificarse.** Estas realizaciones permiten al usuario no registrado inscribirse en el sistema, accediendo de este modo a la totalidad de funcionalidades del mismo. A partir de estos casos de uso, el usuario no registrado pasará a ser un usuario registrado.
- **Contactar** Ofrece la posibilidad al usuario de ponerse en contacto con otro los actor del sistema: el administrador del sistema.
- **Tour.** Estas realizaciones permitirán al entrenador informarse sobre los servicios y prestaciones que ofrece la aplicación.
- **Blog.** El usuario podrá informarse con las funcionalidades o la nuevas noticias de la aplicación, esta sección es meramente informativa, es controlada por el administrador del sistema.
- **FAQ.** Otro apartado meramente formativo en dónde simplemente se tendrá las respuestas a las preguntas más frecuente de los usuarios.

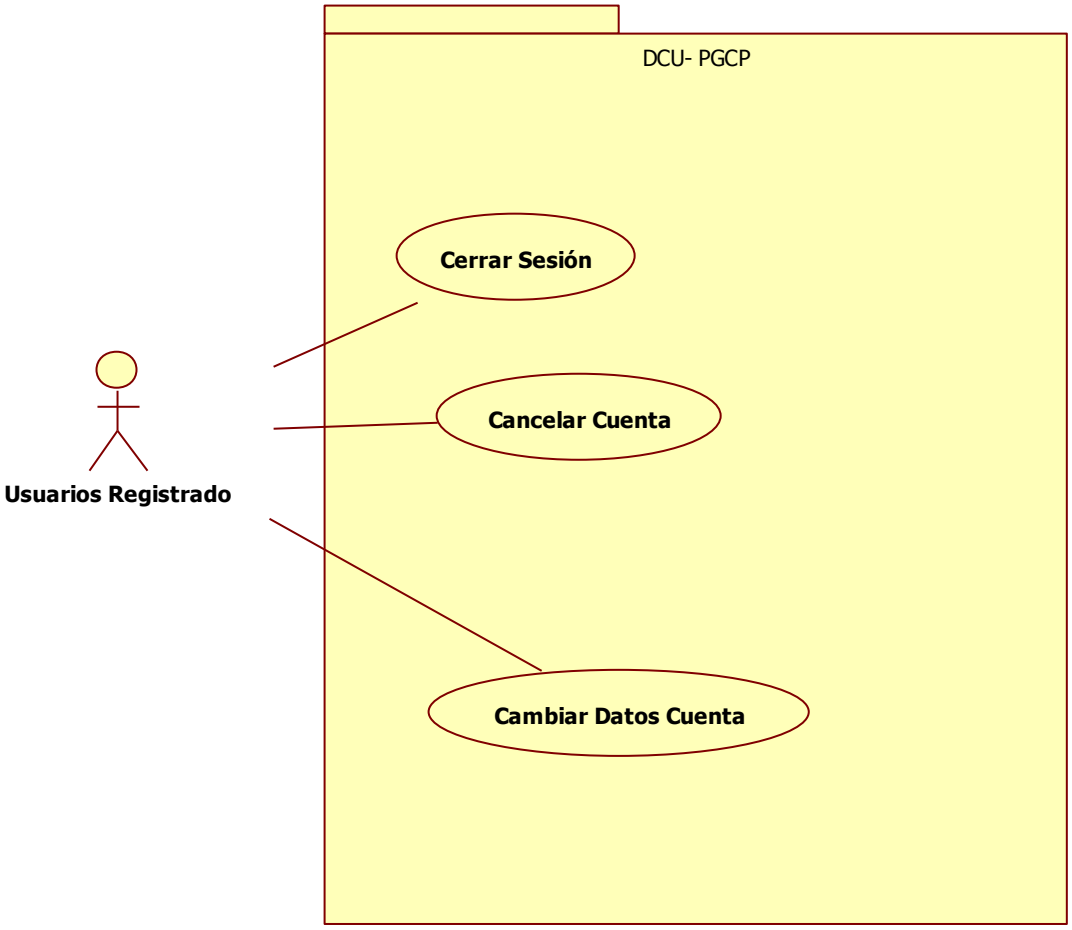


Paquete de gestión de actividades generales

5.2.2.1.2 Paquete de Gestión de Configuración del Perfil

El paquete de gestión de perfil contiene los casos de uso relacionados con la sesión de un usuario registrado. El actor que participa en las realizaciones es el usuario registrado.

- **Cambiar datos cuenta**. Esta realización se ha incluido en la gestión de cuenta porque se da por válido el considerar que la contraseña como el email es parte de los datos configurables por un usuario registrado.
- **Cerrar sesión.** Cuando un usuario identificado termina de usar la aplicación, se le ofrece la posibilidad de cerrar su sesión para que otros usuarios no puedan acceder a su información.
- **Cancelar cuenta.** Cuando un usuario identificado decide que ya no desea seguir teniendo una cuenta en el sistema, se le ofrece la posibilidad de eliminarla de la aplicación. Esto conlleva que no podrá volver acceder con la misma cuenta, puesto que ésta se inhabilita.



Paquete de gestión de configuración de perfil

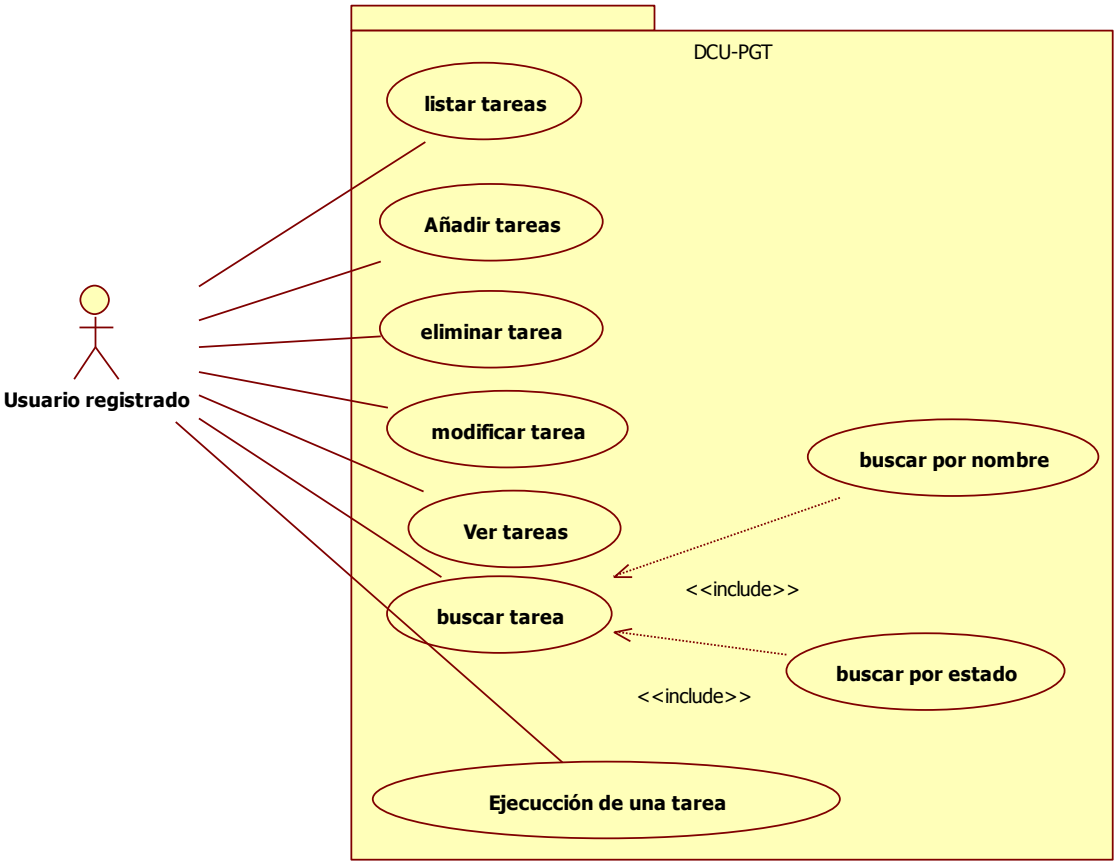
5.2.2.1.3 Paquete de Gestión de Tareas

El paquete de gestión de tareas es la base de la aplicación. El usuarios registrado participa en una serie de realizaciones que conforma este paquete, representando las acciones que se pueden realizar para el control de las tareas. El acceso a estas funcionalidades se ha producido gracias a su previo ingreso en el sistema mediante el caso de uso de identificarse.

Como en el resto de paquetes que se detallarán en las próximas secciones, la estructura básica está compuesta por casos de uso denominados CRUD(create, read, update and delete).

Los casos de uso que contiene este paquete son:

- **Añadir tarea.** Esta realización permite al usuario registrado crear una tarea en el sistema, creándose así tareas pendientes que tendrá que realizar.
- **Lista de tareas y ver tareas.** El listado de tareas contiene todos las tareas añadidas previamente por el usuario registrado. Accediendo a cada uno de ellas se permite ver la datos más específicos de cada una de las tareas .
- **Modificar tarea.** Esta realización permite al usuario registrado modificar los datos almacenados en el sistema de un tarea específico.
- **Eliminar tarea**. Este caso de uso está destinado a eliminar del sistema todos los datos asociados a una tarea .
- **Buscar tarea.** La realización permite que un usuario obtenga consultas sobre el listado total de tareas en el sistema. Se pueden usar las funciones de búsquedas específicas por nombre o filtrando por categoría de estados.
- **Ejecucción de la tarea**. El usuario podrá controlar en todo momento el estado en que se encuentra una tarea que puede ser:
    - **Empezar tarea.** El usuario podrá empezar una tarea que deseé en ese momento empezará a contar el tiempo.
    - **Pausar tarea.** El usuario podrá pausar una tarea especificando la interrupción asociada a dicha pausa.
    - **Retomar una tarea.** El usuario podrá seguir con una tarea que había sido pausada
    - **Finalizar una tarea.** El usuario podrá finalizar una tarea cuando haya terminado en ese momento dejará de contar el contador de tiempo.

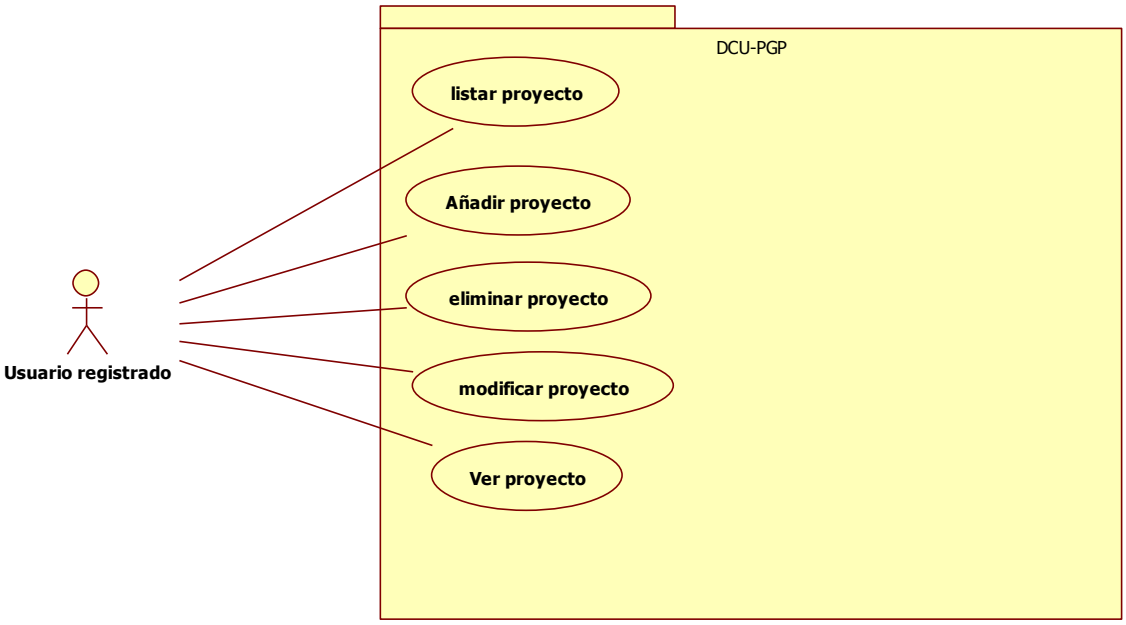


Paquete de gestión de tareas

5.2.2.1.4 Paquete de Gestión de Proyecto

El paquete de gestión de proyectos junto con paquete de gestión de tareas son la base de esta aplicación. Dicho paquete de gestión de proyectos contiene los casos de uso relacionados con los proyectos diarios realizados por un usuario registrado.

- **Añadir proyecto.** Esta realización permite al usuario registrado añadir al sistema un proyecto, también al momento de crearlo puedes elegir el nivel donde colgará el proyecto dentro de un árbol de directorio.
- **Lista de proyectos y ver proyectos.** El listado de proyectos contiene todos los proyectos que se han realizado o se está realizando. Ver proyectos concede la posibilidad al usuario de ver los datos sobre el proyecto así como el listado de tareas que lo componen.
- **Modificar proyecto**. Desde esta realización permite al usuario registrado modificar los datos almacenados en el sistema de un proyecto específico.
- **Eliminar proyecto.** Dicha realización está destinada a eliminar del sistema todos los datos asociados a un proyecto —las tareas que lo componen también tendrán que ser eliminados haciendo uso de las realizaciones correspondientes—.

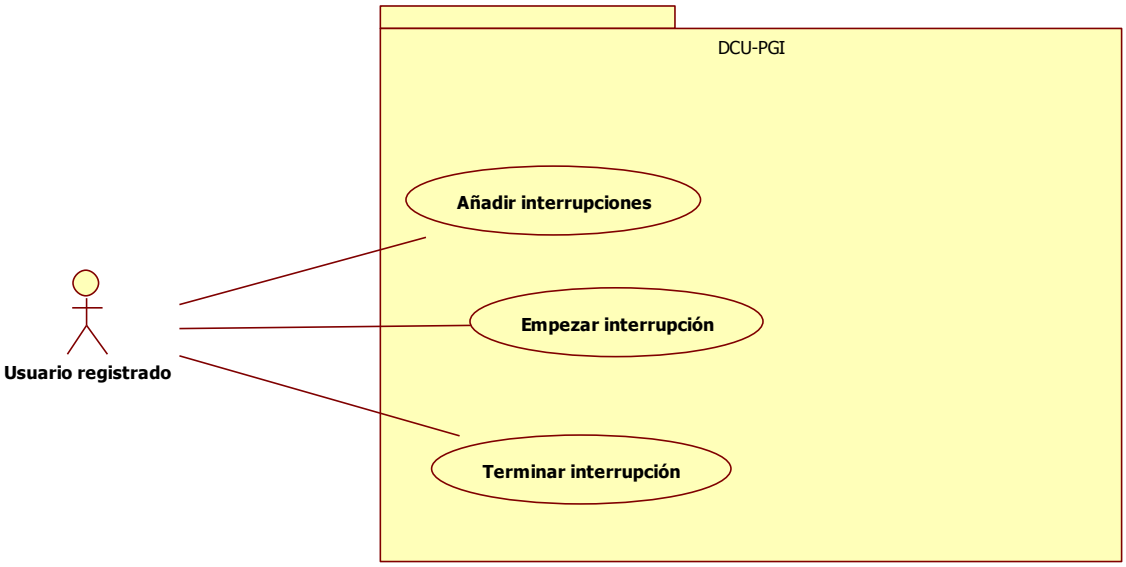


Paquete de gestión de proyectos

5.2.2.1.5 Paquete de Gestión de Interrupción

El paquete de gestión de interrupciones contiene los casos de uso relacionados con los interrupciones diarios realizados por un usuario registrado sobre las tareas.

- **Añadir interrupción.** Esta realización permite al usuario registrado añadir al sistema una interrupción, también al momento de crearlo tienes que especificar el nombre y descripción de la interrupción, así como poner en marcha dicha interrupción.
- **Terminar interrupción**. Desde esta realización permite al usuario registrado terminar una interrupción generada a partir de una tarea.
- **Empezar interrupción.** Desde esta realización permite al usuario registrado empezar una interrupción generada a partir de una tarea.

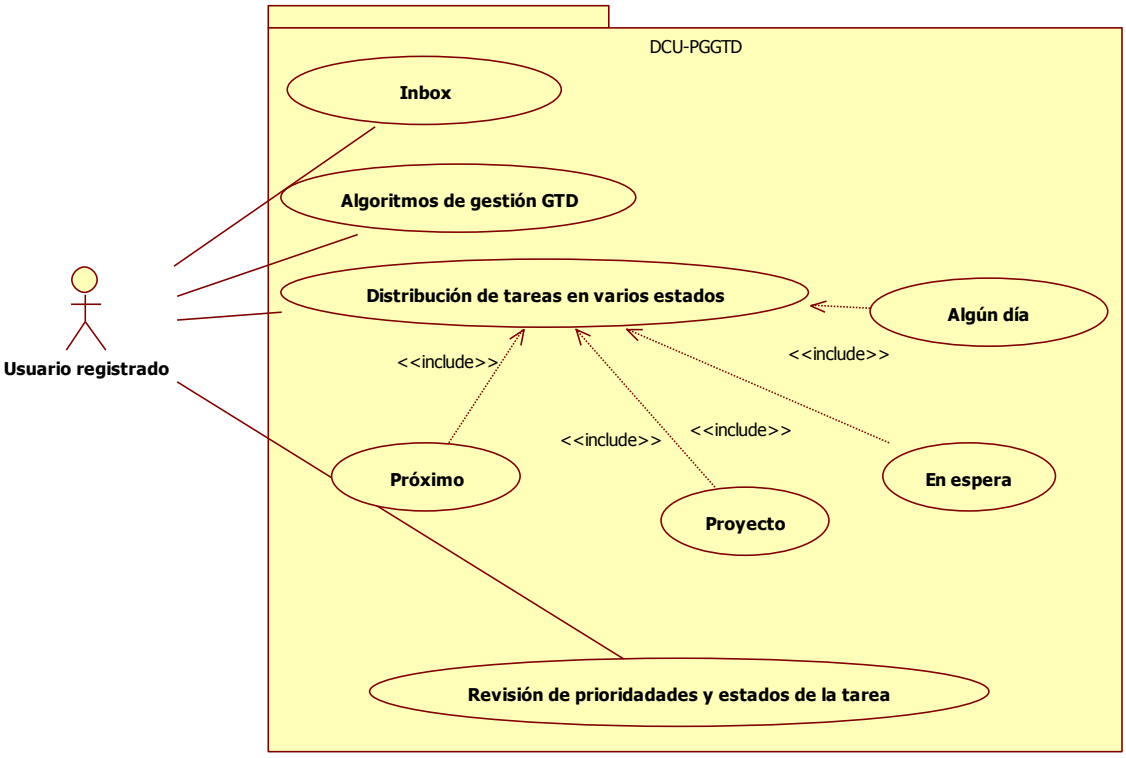


Paquete de gestión de interrupciones

5.2.2.1.6 Paquete de Gestión de Gtd

El paquete de gestión de GTD contiene los casos de uso relacionados con los metodología GTD.

- **Inbox.** Todas las tareas y proyectos del sistema una vez creado será enviados a inbox la carpeta raíz del sistema.
- **Algoritmo de gestión GTD.** Con esta realiza el usuario podrá gestionar siguiendo el algoritmo gtd las tareas creadas.
- **Distribución de las tareas en varios Estados**. Con esta realización GTD distribuirá las tareas en diferentes estados de realización:
  - **Próximo**. Incluimos las tareas que debes realizar cuanto antes
  - **Proyecto**. Incluimos las tareas que implica realizar más de una acción
  - **En espera**. Son las tareas que necesita la intervención de otra persona para ser realizada.
  - **Algún día**: Son tareas que debemos hacer pero no son prioritarias.
- **Revisión de prioridad y estados de las tareas .** Con esta realización se permite al usuario que pueda revisar cada 24 horas las tareas, viendo si han cambiado la prioridad o el estado de dicha tarea.



Paquete de gestión de interrupciones

5.2.3 Especificación de los Casos de Uso

La especificación, puede verse como un proceso de representación. Los requisitos se representan de manera que lleven al éxito de la implementación del software. Para cada realización se realizará una descripción del escenario principal como de cada uno de los escenarios asociados a una realización.

Además, se indicará:

- ID como localizador del caso de uso.
- Actores que participan en la realización.
- Precondiciones que deben cumplirse para que se inicie la realización.
- Postcondiciones que deben cumplirse tras completar la realización, indicando de este modo en qué estado queda el sistema tras finalizar dicha realización.

Cabe destacar que a la hora de especificar los casos de uso, se han tenido presentes los requisitos no funcionales obtenidos a partir de la lista de características inicial. Por otra parte, es importante decir que en una serie de realizaciones se definen relaciones de inclusión y extensión con otras realizaciones. Cuando se describa los flujos principales y alternativos de la realización, se especificarán los puntos de extensión de dicha realización indicando para cada uno de ellos la condición de activación.

A continuación se muestran las especificaciones de los casos de uso detectados en el modelo:

Especificación del caso de uso: Registrarse	
ID	DCU-PGAG.01
Nombre	Registrarse
Descripción	Realiza el registro en la aplicación de un usuario aún no registrado
Actores	Usuario no registrado
Precondiciones	---
Postcondiciones	Se registra en el sistema usuario no registrado
Flujo Normal de Eventos	
1. El usuario no registrado selecciona la opción de registrarse 2. El sistema muestra un formulario de registro con los campos de: email y contraseña 3. El usuario no registrado rellena cada campo del formulario y pulsa el botón de enviar 4. El sistema valida los datos del formulario y registra al usuario. Las validaciones son: formato de email debe ser correcto; longitud mínima de contraseña; confirmar contraseña debe ser igual a contraseña.	
Flujo Alternativo	
3.1. Si el usuario no rellena todos los campos del formulario, el sistema muestra un error y vuelve al paso 2 3.2. Si la contraseña insertada no es igual a la confirmación de contraseña, el sistema muestra un error y vuelve al paso 2 4.1. Si el sistema encuentra error en la validación de datos del formulario, muestra mensaje de error y vuelve al paso 2	

Especificación del caso de uso registrarse

Especificación del caso de uso: Identificarse	
ID	DCU-PGAG.02
Nombre	Identificarse
Descripción	Permite el acceso a un usuario a la aplicación
Actores	Usuario registrados
Precondiciones	Usuario tiene que estar registrado en el sistema
Postcondiciones	usuario accede al sistema
Flujo Normal de Eventos	
1.El usuario selecciona la opción de identificarse 2. El sistema muestra un formulario con los campos de identificación: email y contraseña 3. El usuario rellena los campos del formulario de identificación y pulsa el botón de enviar 4. El sistema valida los datos del formulario. Las validaciones son: formato de email correcto; par email-contraseña debe pertenecer a email registrado en la aplicación. 5. El sistema inicia una sesión en la aplicación para el usuario identificado 6. El sistema muestra la página de "work desk" del usuario identificado	
Flujo Alternativo	
3.1. Si el usuario no rellena todos los campos del formulario, el sistema envía un mensaje de error y vuelve al paso 2 4.1. Si los datos de identificación no coinciden con el email y la contraseña de un usuario registrado, el sistema muestra un mensaje de error y vuelve al paso 2	

Especificación del caso de uso Identificarse

Especificación del caso de uso: Contactar	
ID	DCU-PGAG.03
Nombre	Contact us
Descripción	Permite realizar la puesta en contacto de un usuario con un administrador
Actores	usuario no registrado
Precondiciones	---
Postcondiciones	Mensaje de usuario entregado al administrador
Flujo Normal de Eventos	
1. El usuario selecciona la opción de contactar con administrador 2. El sistema muestra una página estática con la información de contacto y con un formulario con los campos de: nombre, email, mensaje 3. El entrenador rellena los campos del formulario y pulsa el botón de enviar 4. El sistema comprueba que se hayan rellenado todos los datos y envía la información al email de contacto del administrador	
Flujo Alternativo	
4.1. Si el sistema detecta que no se han rellenado todos los campos del formulario, muestra un mensaje de error y vuelve al paso 2 4.2. Si el sistema detecta que el email no tiene una estructura correcta, muestra un mensaje de error y vuelve al paso 2	

Especificación del caso de uso contactar

Especificación del caso de uso: tour	
ID	DCU-PGAG.04
Nombre	tour
Descripción	Muestra una página estática como interfaz de entrada, con un tour de las características de la aplicación
Actores	Usuario no registrado
Precondiciones	---
Postcondiciones	Se le muestra al usuario el tour de la aplicación
Flujo Normal de Eventos	
1. El usuario selecciona la opción de ver tour de la aplicación 2. El sistema muestra una página estática con toda la información acerca del tour de la aplicación	
Flujo Alternativo	

Especificación del caso de uso tour

Especificación del caso de uso: FAQ	
ID	DCU-PGAG.05
Nombre	faq
Descripción	Muestra una página estática como interfaz de entrada, con las preguntas más típicas realizadas por los usuario de la aplicación.
Actores	usuario no registrado.
Precondiciones	---
Postcondiciones	Se le muestra al usuario el faq de la aplicación
Flujo Normal de Eventos	
1. El usuario selecciona la opción de ver faq de la aplicación 2. El sistema muestra una página estática con toda la información acerca del faq de la aplicación	
Flujo Alternativo	

Especificación del caso de uso faq

Especificación del caso de uso: blog	
ID	DCU-PGAG.06
Nombre	Blog
Descripción	Muestra una página estática como interfaz de entrada, con las noticias más importantes de la aplicación o eventos próximos.
Actores	Usuario no registrado, administrador
Precondiciones	---
Postcondiciones	Se le muestra el blog de la aplicación
Flujo Normal de Eventos	
1. El usuario selecciona la opción de ver blog de la aplicación 2. El sistema muestra una página estática con toda la información acerca del blog de la aplicación	
Flujo Alternativo	

Especificación del caso de uso blog

Especificación del caso de uso: Cerrar Sesión	
ID	DCU-PGCP.01
Nombre	Cerrar Sesión
Descripción	Finaliza la sesión iniciada tras identificarse un usuario registrado en el sistema.
Actores	Usuario registrado
Precondiciones	Usuario regsitrado identificado
Postcondiciones	Sistema finaliza la sesión del usuario
Flujo Normal de Eventos	
1. El usuario selecciona la opción de cerrar sesión 2. El sistema elimina la sesión iniciada por el usuario 3. El sistema redirecciona al entrenador a la página principal pública de la aplicación	
Flujo Alternativo	

Especificación del caso de uso cerrar sesión

Especificación del caso de uso: Cancelar cuenta	
ID	DCU-PGCP.02
Nombre	Cancelar cuenta
Descripción	Permite que un usuario registrado que cancele su cuenta activa en la aplicación
Actores	Usuario registrado
Precondiciones	Usuario identificado
Postcondiciones	Eliminada del sistema la cuenta del usuario registrado
Flujo Normal de Eventos	
1. El usuario selecciona la opción cancelar cuenta 2. El sistema muestra un formulario para verificar que se quiere cancelar la cuenta 3. El sistema elimina la cuenta de ese usuario registrado	
Flujo Alternativo	

Especificación del caso de uso cancelar cuenta

Especificación del caso de uso: Cambiar datos cuenta	
ID	DCU-PGCP.03
Nombre	Cambiar datos cuenta
Descripción	Permite que un usuario registrado que cambie la información de su cuenta asociada al sistema.Dicha información son email, contraseña.
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	Información de cuenta usuario modificada
Flujo Normal de Eventos	
1. El usuario selecciona la opción de modificar cuenta 2. El sistema muestra un formulario con campos de su cuenta en modo de edición. 3. El usuario modifica los datos del formulario que desee y pulsa el botón para enviar el formulario 4. El sistema valida los campos y comprueba que los campos obligatorios estén cumplimentados 5. El sistema almacena los datos modificados en el perfil asignado al usuario	
Flujo Alternativo	
1. Si el sistema detecta que no se han rellenado los campos obligatorios o hay un fallo de validación de datos, el sistema muestra un mensaje de error y vuelve al paso 2	

Especificación del caso de uso cambiar datos cuenta

Especificación del caso de uso: Lista de tareas	
ID	DCU-PGT.01
Nombre	Ver listar tareas
Descripción	Muestra el listado de los tareas insertados por un usuario en la aplicación
Actores	Usuario registrado.
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario selecciona la opción de ver lista de tareas 2. El sistema muestra el listado de todos las tareas pertenecientes al usuario registrado. Los datos que se deben mostrar son los de: nombre.	
Flujo Alternativo	
2.1 Si el usuario no ha insertado ningún tarea en el sistema, el sistema muestra la carpeta raíz "INBOX" vacía	

Especificación del caso de uso ver lista de tareas

Especificación del caso de uso: ver tarea	
ID	DCU-PGT.02
Nombre	Ver tarea
Descripción	Muestra una tarea específica perteneciente al usuario registrado
Actores	Usuario registrado
Precondiciones	Usuario está identificado y viendo el listado de tareas
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario selecciona la tarea que quiere ver de la lista de tareas 2. El sistema muestra los datos pertenecientes a la tarea especificada	
Flujo Alternativo	

Especificación del caso de uso ver tarea

Especificación del caso de uso: Añadir tarea	
ID	DCU-PGT.03
Nombre	Añadir tarea
Descripción	Permite a un usuario registrado insertar una tarea en el sistema
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	Actualizada la lista de tarea asociados al usuario registrado
Flujo Normal de Eventos	
1. El usuario selecciona la opción de añadir tarea 2. El sistema muestra un formulario con los campos: nombre, descripción, fecha de vencimiento, prioridad. 3. El usuario inserta los valores que se le piden en el formulario 4. El sistema valida los campos y comprueba que los campos obligatorios estén cumplimentados. 5. El sistema almacena el registro de la nueva tarea y muestra mensaje de usuario añadido correctamente	
Flujo Alternativo	
3.1. Si el usuario pulsa el botón de cancelar, ir a paso 3.2 3.2. El sistema no almacena el registro de la tarea se cancela todo el proceso 4.1. Si el sistema detecta que no se han rellenado los campos obligatorios o hay fallo de validación de datos, el sistema muestra un mensaje de error y vuelve al paso 2	

Especificación del caso de uso añadir tarea

Especificación del caso de uso: Modificar tarea	
ID	DCU-PGT.04
Nombre	Modificar tarea
Descripción	Modifica una tarea específica perteneciente a un usuario registrado
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y viendo la tarea específica a modificar
Postcondiciones	Actualizados los datos de la tarea modificada
Flujo Normal de Eventos	
1. El usuario selecciona la opción de modificar la tarea 2. El sistema muestra un formulario con los campos de nombre, descripción, fecha de vencimiento, prioridad. 3. El usuario modifica alguno de los campos del formulario y pulsa el botón de modificar 4. El sistema valida los campos modificados. 5. El sistema almacena los datos de la tarea modificada y muestra el mensaje de tarea modificada correctamente	
Flujo Alternativo	
3.1. Si el usuario pulsa el botón de cancelar, ir a paso 3.2 3.2. El sistema no almacena los datos de la tarea se cancela todo el proceso 4.1. Si el sistema detecta que hay un fallo en la validación de los datos, muestra un mensaje de error y vuelve al paso 2	

Especificación del caso de uso modificar tarea

Especificación del caso de uso: Eliminar tarea	
ID	DCU-PGT.05
Nombre	Eliminar tarea
Descripción	Elimina del sistema una tarea perteneciente a un usuario registrado.
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y viendo la tarea específica a eliminar
Postcondiciones	Actualizado el listado de tarea asociados al usuario registrado
Flujo Normal de Eventos	
1. El usuario selecciona la opción de eliminar tarea 2. El sistema muestra un mensaje para confirmar la eliminación de la tarea 3. El usuario pulsa el botón de eliminar 4. El sistema elimina de la aplicación la tarea especificada	
Flujo Alternativo	
3.1. Si el usuario pulsa el botón de cancelar, ir a paso 3.2 3.2. El sistema no elimina la tarea de la aplicación, se cancela el proceso	

Especificación del caso de uso eliminar tarea

Especificación del caso de uso: buscar tarea	
ID	DCU-PGT.06
Nombre	Buscar tarea
Descripción	Busca un tarea perteneciente a un usuario registrado
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario selecciona la opción de buscar tarea 2. El sistema muestra un formulario donde el usuario insertará los parámetros de búsqueda 3. El usuario inserta el parámetro de búsqueda (buscar por nombre o estados) 4. El sistema muestra un listado con las tareas que cumplen los requisitos de búsqueda	
Flujo Alternativo	
4.1. Si no se encuentran tareas que cumplan con los requisitos de búsqueda, el sistema muestra un mensaje de error y vuelve al paso 3	
Puntos de Extensión	
3.1. Si el usuario desea realizar únicamente una búsqueda por nombre. Ver Caso de DCU-PGP.07 3.2. Si el usuario desea realizar únicamente una búsqueda por estados .Ver caso de Uso DCU-PGP.08	

Especificación del caso de uso buscar tarea

Especificación del caso de uso: buscar tarea por nombre	
ID	DCU-PGT.07
Nombre	Buscar tarea por nombre
Descripción	Realiza una tarea perteneciente a un usuario registrado por el nombre
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario selecciona como parámetro en la búsqueda: nombre 2. El sistema muestra las tareas que coincidan con el nombre de la búsqueda	
Flujo Alternativo	
2.1. Si no se encuentra ninguna tarea que cumpla las condiciones de búsqueda, mostrar mensaje de error y volver al paso 1	

Especificación del caso de uso buscar tarea por nombre

Especificación del caso de uso: buscar tarea por estado	
ID	DCU-PGT.08
Nombre	Buscar tarea por estado
Descripción	Realiza una tarea perteneciente a un usuario registrado por el estado
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario selecciona como parámetro en la búsqueda: el estado que desea 2. El sistema muestra las tareas que coincidan con el estado de la búsqueda	
Flujo Alternativo	
2.1. Si no se encuentra ninguna tarea que cumpla las condiciones de búsqueda, mostrar mensaje de error y volver al paso 1	

Especificación del caso de uso buscar tarea por estado

Especificación del caso de uso: Gestión de estado de una tarea	
ID	DCU-PGT.09
Nombre	Gestión de estado de una tarea
Descripción	Control de estado de una tarea , si ha empezado, si ha finalizado, si está pausada, o está en macha
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario aprieta un botón para que empiece o pause una tarea. 2.El sistema identifica que la tarea acaba cuando acaba la fecha de vencimiento o le das acabar en el botón finalizar.	
Flujo Alternativo	
1.1 el usuario no aprieta el botón empezar con lo cual no empieza la tarea hasta que haga el paso 1	
Puntos de Extensión	
1.1. Si el usuario desea empezar tarea. Ver Caso de DCU-PGT.10 1.2. Si el usuario desea pausar tarea. Ver Caso de DCU-PGT.11 1.3. Si el usuario desea retornar tarea. Ver Caso de DCU-PGT.12 1.4. Si el usuario desea finalizar tarea. Ver Caso de DCU-PGT.13	

Especificación del caso de uso Gestión de estado de una tarea

Especificación del caso de uso: Empezar tarea	
ID	DCU-PGT.10
Nombre	Empezar tarea
Descripción	Empieza a contar el contador de la tarea
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y haber creado una tarea
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario selecciona empezar una tarea. 2. El contador a funcionar hasta la fecha de vencimiento	
Flujo Alternativo	
1.1. Si no presionas empezar tarea la tarea no empieza.	

Especificación del caso de uso buscar empezar tarea

Especificación del caso de uso: pausar tarea	
ID	DCU-PGT.11
Nombre	pausar tarea
Descripción	Contador de la tarea se detiene de contar
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y haber creado una tarea y haber comenzado dicha tarea
Postcondiciones	Contador se detiene
Flujo Normal de Eventos	
1. El usuario selecciona pausar una tarea. 2. El contador se detiene hasta la que vuelva a retomar la tarea	
Flujo Alternativo	
1.1. Si no presionas pausar tarea la tarea no se pausa	

Especificación del caso de uso buscar pausar tarea

Especificación del caso de uso: retomar una tarea	
ID	DCU-PGT.12
Nombre	Retomar una tarea
Descripción	Empieza a volver a funcionar el contador de la tarea desde el punto donde fue pausada
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y haber creado una tarea, y haber presionado pausar tarea
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario selecciona retomar una tarea. 2. El contador vuelve a funcionar hasta la fecha de vencimiento	
Flujo Alternativo	
1.1. Si no presionas retomar tarea la tarea no vuelve a continuar.	

Especificación del caso de uso buscar retomar tarea

Especificación del caso de uso: finalizar tarea	
ID	DCU-PGT.13
Nombre	Finalizar tarea
Descripción	Una tarea termina su ciclo de vida
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y haber creado una tarea, y haber empezado la tarea
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario selecciona finalizar una tarea. 2. El contador termina porque ha llegado a la fecha de vencimiento o porque ha terminado la tarea antes de la fecha de vencimiento	
Flujo Alternativo	
1.1. Si no presionas finalizar la tarea la tarea no finaliza hasta que llegue la fecha de vencimiento.	

Especificación del caso de uso buscar finalizar tarea

Especificación del caso de uso: Lista de proyectos	
ID	DCU-PGP.01
Nombre	Ver listar proyectos
Descripción	Muestra el listado de los proyectos insertados por un usuario en la aplicación
Actores	Usuario registrado.
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario selecciona la opción de ver lista de proyectos 2. El sistema muestra el listado de todos los proyectos pertenecientes al usuario registrado. Los datos que se deben mostrar son los de: nombre, nivel.	
Flujo Alternativo	
2.1 Si el usuario no ha insertado ningún proyecto en el sistema, el sistema muestra la carpeta raíz "INBOX"	

Especificación del caso de uso ver lista de proyectos

Especificación del caso de uso: ver proyectos	
ID	DCU-PGP.02
Nombre	Ver proyecto
Descripción	Muestra un proyecto específico perteneciente al usuario registrado
Actores	Usuario registrado
Precondiciones	Usuario está identificado y viendo el listado de proyectos
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario selecciona el proyecto que quiere ver de la lista de proyectos 2. El sistema muestra los datos pertenecientes al proyecto especificado	
Flujo Alternativo	

Especificación del caso de uso ver proyectos

Especificación del caso de uso: Añadir proyecto	
ID	DCU-PGP.03
Nombre	Añadir proyecto
Descripción	Permite a un usuario registrado insertar un proyecto en el sistema
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	Actualizada la lista de proyecto asociados al usuario registrado
Flujo Normal de Eventos	
1. El usuario selecciona la opción de añadir proyecto 2. El sistema muestra un formulario con los campos: nombre 3. El usuario inserta los valores para el nombre y el nivel en donde estará colgado 4. El sistema valida los campos y comprueba que los campos obligatorios estén cumplimentados. 5. El sistema almacena el registro del nuevo proyecto y muestra mensaje de usuario añadido correctamente	
Flujo Alternativo	
3.1. Si el usuario pulsa el botón de cancelar, ir a paso 3.2 3.2. El sistema no almacena el registro del proyecto y cancela todo el proceso 4.1. Si el sistema detecta que no se han rellenado los campos obligatorios o hay fallo de validación de datos, el sistema muestra un mensaje de error y vuelve al paso 2 4.3. Si existe un proyecto perteneciente al usuario con el mismo nombre en el mismo nivel, el sistema muestra un error y vuelve al paso 2	

Especificación del caso de uso añadir proyecto

Especificación del caso de uso: Modificar proyecto	
ID	DCU-PGP.04
Nombre	Modificar proyecto
Descripción	Modifica un proyecto específico perteneciente a un usuario registrado
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y viendo el proyecto específico a modificar
Postcondiciones	Actualizados los datos del proyecto modificado

Flujo Normal de Eventos
1. El usuario selecciona la opción de modificar del proyecto 2. El sistema muestra un formulario con los campos de nombre. 3. El usuario modifica alguno de los campos del formulario y pulsa el botón de modificar 4. El sistema valida los campos modificados. 5. El sistema almacena los datos del proyecto modificado y muestra el mensaje de proyecto modificado correctamente
Flujo Alternativo
3.1. Si el usuario pulsa el botón de cancelar, ir a paso 3.2 3.2. El sistema no almacena los datos del proyecto y cancela todo el proceso 4.1. Si el sistema detecta que hay un fallo en la validación de los datos, muestra un mensaje de error y vuelve al paso 2

Especificación del caso de uso modificar proyecto

Especificación del caso de uso: Eliminar proyecto	
ID	DCU-PGP.05
Nombre	Eliminar proyecto
Descripción	Elimina del sistema un proyecto perteneciente a un usuario registrado.
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y viendo el proyecto específico a eliminar
Postcondiciones	Actualizado el listado de proyectos asociados al usuario registrado
Flujo Normal de Eventos	
1. El usuario selecciona la opción de eliminar proyecto 2. El sistema muestra un mensaje para confirmar la eliminación del proyecto 3. El usuario pulsa el botón de eliminar 4. El sistema elimina de la aplicación el proyecto especificado junto con las tareas pertenecientes al mismo	
Flujo Alternativo	
3.1. Si el usuario pulsa el botón de cancelar, ir a paso 3.2 3.2. El sistema no elimina el proyecto de la aplicación, se cancela el proceso	

Especificación del caso de uso eliminar proyecto

Especificación del caso de uso: Añadir interrupción	
ID	DCU-PGI.03
Nombre	Añadir interrupción
Descripción	Permite a un usuario registrado insertar una interrupción en una tarea especificada del sistema
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	Actualizada la lista de interrupciones asociados a la tarea y al usuario registrado
Flujo Normal de Eventos	
1. El usuario selecciona la opción de añadir interrupción 2. El sistema muestra un formulario con los campos: nombre, descripción 3. El usuario inserta los valores para el nombre y la descripción 4. El sistema valida los campos y comprueba que los campos obligatorios estén cumplimentados. 5. El sistema almacena el registro del nueva interrupción y muestra un mensaje al usuario de " interrupción añadida correctamente"	
Flujo Alternativo	
3.1. Si el usuario pulsa el botón de cancelar, ir a paso 3.2 3.2. El sistema no almacena el registro del interrupción y cancela todo el proceso 4.1. Si el sistema detecta que no se han rellenado los campos obligatorios o hay fallo de validación de datos, el sistema muestra un mensaje de error y vuelve al paso 2	

Especificación del caso de uso añadir interrupción

Especificación del caso de uso: Empezar interrupción	
ID	DCU-PGI.06
Nombre	Empezar interrupción
Descripción	Creación y comienzo contador del tiempo de la interrupción
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario selecciona crea la interrupción en una tarea específica 2. Se pone en marcha un contador de tiempo para contar el tiempo de interrupción	
Flujo Alternativo	

Especificación del caso de uso Empezar interrupción

Especificación del caso de uso: Finalizar interrupción	
ID	DCU-PGI.07
Nombre	Finalizar interrupción
Descripción	Finaliza el contador de tiempo interrumpido
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	---
Flujo Normal de Eventos	
1. El usuario selecciona retomar una tarea específica 2. El sistema finaliza contador de tiempo asociado a la interrupción que estaba en marcha	
Flujo Alternativo	

Especificación del caso de uso finalizar interrupción

Especificación del caso de uso: inbox	
ID	DCU-PGGTD.01
Nombre	inbox
Descripción	Carpeta raíz del sistema a donde va inicialmente todo proyecto o tarea creado por la aplicación.
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación
Postcondiciones	---
Flujo Normal de Eventos	
1.El usuario se identifica en el sistema 2.El usuario entra en su "work desk" donde se le mostrará su carpeta raíz inbox.	
Flujo Alternativo	
1.1 El usuario no se identificar correctamente y se queda en el punto 1 hasta que lo haga.	

Especificación del caso de uso de inbox

Especificación del caso de uso: Algoritmo GTD	
ID	DCU-PGGTD.02
Nombre	Algoritmo GTD
Descripción	El sistema en toda tarea pone en marcha el algoritmo GTD para llevar la gestión de las tareas.
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación, y haber creado tareas almacenadas en inbox.
Postcondiciones	---
Flujo Normal de Eventos	
1.Sacar de inbox tareas 2.Poner en marcha el algoritmo GTD para la gestión de las tareas	
Flujo Alternativo	
1.1 No sacar tareas de inbox porque no hay ninguna en la carpeta.	

Especificación del caso de uso de Algoritmo GTD

Especificación del caso de uso: Distribución de la tarea en varios estados	
ID	DCU-PGGTD.03
Nombre	Distribución de la tarea en varios estados
Descripción	Para tareas que tenemos que hacerlas nosotros que nos lleva más de 2 minutos hacerlas el sistema gestiona estas tareas asignándolas en una carpetas específicas.
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y comienzo de algoritmo GTD
Postcondiciones	---
Flujo Normal de Eventos	
1.El Algoritmo GTD identifica que la analiza la tarea 2.El algoritmo GTD observa que es una tarea que tenemos que hacer nosotros 3. El algoritmo GTD observa que la tarea nos lleva más de 2 minutos hacerla. 4.El algoritmo GTD decide distribuir la tarea a una carpeta en particular de un conjunto de carpetas con objetivos claros cada una de ellas.	
Flujo Alternativo	
2.1 El algoritmo GTD observa que es una tarea que no tenemos que hacerla nosotros sino tenemos que delegarla en otra persona 3.1 El algoritmo GTD observa que la tarea se puede realizar en menos de 2 minutos.	
Puntos de Extensión	
4.1. Si el usuario es distribuido a la carpeta próximo. Ver Caso de DCU-PGGTD.04 4.2. Si el usuario es distribuido a la carpeta proyecto. Ver Caso de DCU-PGGTD.05 4.3. Si el usuario es distribuido a la carpeta En espera. Ver Caso de DCU-PGGTD.06 4.4. Si el usuario es distribuido a la carpeta Algún día. Ver Caso de DCU-PGGTD.07	

Especificación del caso de uso de distribución de la tarea en varios estados

Especificación del caso de uso: Estado próximo	
ID	DCU-PGGTD.04
Nombre	Estado próximo
Descripción	Se incluye tareas que se debe realizar cuanto antes
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y haber empezado analizar una tarea con el algoritmo GTD
Postcondiciones	---
Flujo Normal de Eventos	
1.El algoritmo GTD decide después de analizar el la tarea con el algoritmo GTD ponerlo en la carpeta próximo debido a su prioridad.	
Flujo Alternativo	
1.1 El algoritmo decide ponerlo en otra carpeta excluyendo la carpeta próximo.	

Especificación del caso de uso de próximo

Especificación del caso de uso: Estado proyecto	
ID	DCU-PGGTD.05
Nombre	Estado proyecto
Descripción	Se incluye tareas que implican realizar más de una acción
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y haber empezado analizar una tarea con el algoritmo GTD
Postcondiciones	---
Flujo Normal de Eventos	
1.El algoritmo GTD decide después de analizar el la tarea con el algoritmo GTD ponerlo en la carpeta proyecto debido a que la tarea implica realizar más de una acción	
Flujo Alternativo	
1.1 El algoritmo decide ponerlo en otra carpeta excluyendo la carpeta proyecto.	

Especificación del caso de uso de proyecto

Especificación del caso de uso: Estado En espera	
ID	DCU-PGGTD.06
Nombre	En espera
Descripción	Son tareas que necesita la intervención de otra persona para ser realizada.
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y haber empezado analizar una tarea con el algoritmo GTD
Postcondiciones	---
Flujo Normal de Eventos	
1.El algoritmo GTD decide después de analizar el la tarea con el algoritmo GTD ponerlo en la carpeta En espera debido necesita la intervención de otra persona para ser realizada.	
Flujo Alternativo	
1.1 El algoritmo decide ponerlo en otra carpeta excluyendo la carpeta en espera.	

Especificación del caso de uso de en espera

Especificación del caso de uso: Algún día	
ID	DCU-PGGTD.07
Nombre	Algún día
Descripción	Son tareas que debemos hacer pero no son prioritarias
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y haber empezado analizar una tarea con el algoritmo GTD
Postcondiciones	---
Flujo Normal de Eventos	
1.El algoritmo GTD decide después de analizar el la tarea con el algoritmo GTD ponerlo en la carpeta Algún día debido a que son tareas que no son prioritarias	
Flujo Alternativo	
1.1 El algoritmo decide ponerlo en otra carpeta excluyendo la carpeta Algún día.	

Especificación del caso de uso de Algún día

Especificación del caso de uso: Revisión de prioridades y estado de las tareas	
ID	DCU-PGGTD.08
Nombre	Revisión de prioridades y estado de las tareas
Descripción	Cada 24 hora el sistema avisará al usuario para que revise sus tareas a ve si han cambiado las prioridades o los estados de las tareas y las sitúen en lugar correcto
Actores	Usuario registrado
Precondiciones	Usuario identificado en la aplicación y haber usado el algoritmo GTD para analizar tareas
Postcondiciones	---
Flujo Normal de Eventos	
1.El sistema avisa con una alerta al usuario tras 24 hora 2.El usuario deberá para cada tarea revisar las prioridades y el estado	
Flujo Alternativo	

Especificación del caso de uso de Revisión de prioridades y estado de las tareas

5.2.4 Requisitos no Funcionales

El sistema debe cumplir una serie de requisitos que son independiente de los objetivos principales del proyecto.

Los requisitos no funcionales generales estarán enmarcados en los siguientes aspectos:

- La solución debe estar basada en web.
- La aplicación debe estar diseñada y desarrollada sobre la plataforma Ruby On Rails.
- Orientada a objetos.
- De fácil mantenimiento, uso de guías y patrones con especial énfasis en el patrón MVC, documentación y de fácil ubicación de componentes.
- Que permita y utilice reutilización de código.

Además intentaremos que el software cumpla con un par de *requisitos de apariencia* como:

- Que las pantallas usen un estilo visual característico y que no moleste a la vista, siendo lo más agradable a la vista que se pueda
- .Se debe diseña la aplicación atendiendo a las reglas establecidas por HTML 5.x y CSS 3.x. Su simplicidad inicial debe ser lo más fácil posible.

En cuanto a requisitos de *usabilidad y accesibilidad* se intentará que las interfaces del sistema sea lo más fáciles de utilizar que se pueda para ellos:

- Las interfaces estará formadas por pantallas muy ligeras con poca información simultánea.
- El sistema tendrá que poderse usar de forma intuitiva sin la necesidad de leer ningún manual de usuario.
- Se utilizará imágenes y iconos representativos para la información mostrada en las pantallas
- Evitaremos que en los formularios se pidan datos repetidos en otros formularios

- El sistema mostrará mensajes y textos descriptivos para que el usuario sepa en todo momento que requiere el sistema
- El sistema informará siempre del resultado de las operaciones realizadas. De éste modo el usuario tendrá certeza de que la operación realizada se ha producido del modo esperado.

En cuanto a los requisitos de *rendimiento y mantenibilidad* se debe cumplir:

- El tiempo de respuesta del sistema para cualquier petición del usuario tiene que ser lo más pequeños posible que se pueda
- Debe contemplar requerimientos de confiabilidad y consistencia de los componentes de negocio ante recuperaciones. En caso de fallas de algún componente, no debe haber pérdida de información.
- El sistema debe ser mantenible. Su existencia está ligada a Ruby On Rails y, por tanto, debe ser fácil su actualización para poder seguir siendo funcional en las siguientes versiones del framework.

Finalmente en cuanto a requisitos de *seguridad y fiabilidad del sistema*, se deben tener en consideración los siguientes criterios:

- Los usuarios deberán estar registrados y autentificados en el mismo. La información privada de cada usuario sólo podrá ser modificada por el propio usuario.
- El sistema deberá estar protegido contra el software malintencionado así como de usuarios malintencionados.

5.3 Análisis

El objetivo del análisis es comprender el problema y comenzar a desarrollar un modelo visual de lo que se está tratando de construir, independiente de la tecnología o lenguaje que utilicemos .

El análisis se centra en identificar los objetos que forma el sistema, describiendo la realización de casos de uso y sirve como una abstracción del modelo de diseño. Básicamente el análisis consiste en traducir los requisitos funcionales en conceptos de software. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero.

El lenguaje que se utiliza en el análisis se basa en un modelo de análisis que nos ayudará a refinar los requisitos y a razonar sobre los aspectos internos del sistema. Durante el análisis, se identifican, de manera continua, nuevos paquetes del análisis, clases y requisitos comunes a medida que el modelo de análisis evoluciona, y los paquetes de análisis concretos continuamente se refinan y mantienen.

A diferencia del modelo de casos de uso que captura la funcionalidad del sistema, el modelo de análisis da forma a la arquitectura para soportar las funcionalidades que en el modelo anterior se expresan.

Sus características principales son:

- Proporciona un diseño preliminar, pues contiene paquetes que se usan para organizar el modelo de análisis en piezas más manejables, que representan abstracciones o subsistemas y una primera vista del diseño.
- Puede ayudar a descubrir una necesidad de clases adicionales.
- Proporciona una prueba de completitud a los casos de uso, antes de pasar al diseño.
- Proporciona un diseño preliminar de la arquitectura del sistema, denotando los paquetes de análisis de alto nivel.

Por lo tanto para conseguir todo estas características este capítulo se organizará de la siguiente forma:

1. **Realización de los casos usos**. En esta apartado se aportará información de las interacciones que se producen internamente en el sistema entre los diferentes componentes que conforman su arquitectura usando para ello diagrama de colaboración.
2. **Realización de los diagramas de clases.** Se mostrarán las clases que interviene en los diagramas de colaboración, mostrando la relación que existe entre cada una de ellas.
3. **Realización de los diagramas de paquetes.** En esta sección se esbozará el modelo de análisis y la arquitectura mediante la identificación de paquetes, proporcionando así un medio para organizar el modelo de análisis en piezas más pequeñas y más manejables.

5.3.1 Realización de los Casos de usos

En esta sección se pretende aportar una descripción de las interacciones que se producen internamente en el sistema entre los diferentes componentes que conforman su arquitectura. Para ellos nos ayudaremos de los diagramas de interacción los cuales son diagramas que describen cómo grupos de objetos colaboran para conseguir algún fin. El objetivo de estos diagramas es mostrar objetos, así como los mensajes que se pasan entre ellos dentro del caso de uso, es decir, capturan el comportamiento de los casos de uso.

Actualmente existe dos tipo de diagramas de interacción:

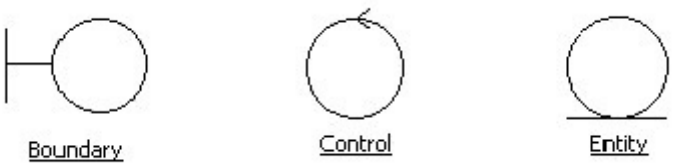
- diagramas de secuencia
- diagramas de colaboración.

Se puede decir que un diagrama de colaboración es una forma alternativa al diagrama de secuencias a la hora de mostrar un escenario. Para este análisis del proyecto final de grado se ha decidido usar diagramas de colaboración.

En cada diagrama de colaboración se identificarán 3 tipos de clases de análisis, ya que trabajamos en una arquitectura 3 capas.

- **Boundary**. Se utiliza para modelar la interacción entre el sistema y sus actores, es decir, una clase boundary sirve como medio de comunicación entre un actor y el correspondiente caso de uso. Representan ventanas, formularios, paneles, interfaces de comunicación, etc..
- **Entity**. Se utiliza para modelar información que posee una vida larga y que es a menudo persistente. Están asociadas a algún fenómeno o concepto, como una persona, un objetivo del mundo real o un suceso del mundo real.
- **Control**. Representan coordinación, secuencia, transacciones y control de los objetos y se usan para encapsular el control de un caso de uso en concreto.

UML utiliza como mecanismo para extender estas notaciones de clase los siguientes iconos:

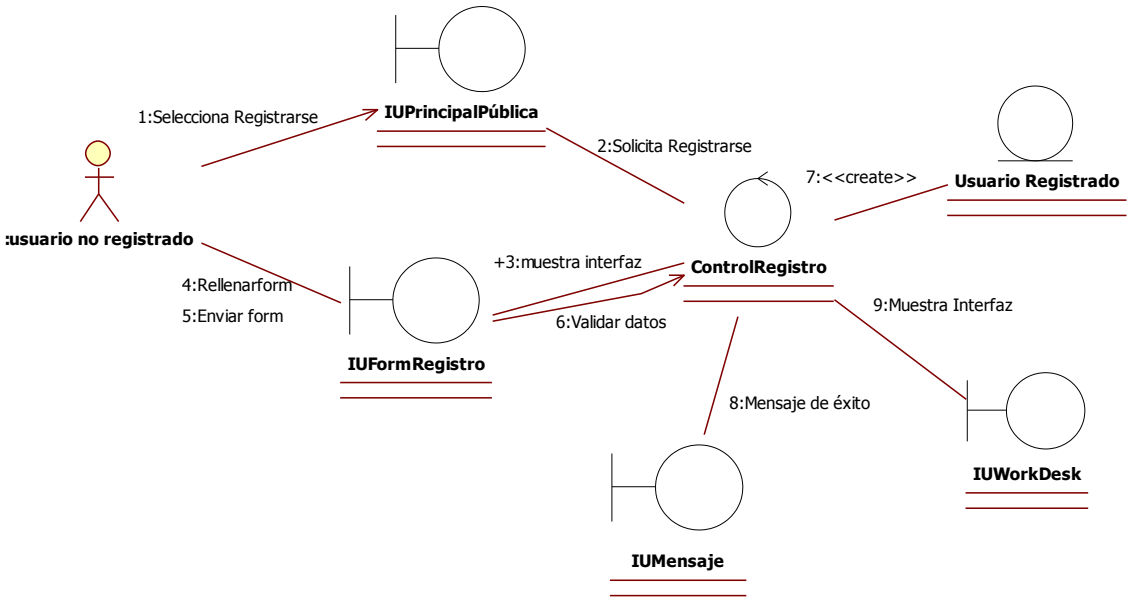


En este documento se detallan las principales colaboraciones relacionadas con los principales casos de uso descritos en el proyecto.

Por tanto, se detallarán los diagramas de colaboración que hacen referencia a la *gestión de actividades generales, gestión de tareas, gestión de proyectos, gestión de interrupciones, gestión de la filosofía GTD.*

5.3.1.1 Colaboraciones para la Gestión del Acceso

La primera de las colaboraciones es la referente al registro de usuarios en el sistema. Una vez que se finaliza el proceso de registro, el usuario pasa a formar parte de los actores "usuario Registrado".



Seguido del proceso de registro, continúa el de identificación en la aplicación. El usuario registrado debe acceder al sistema con los datos que insertó en el formulario de registro. En caso de no ser así, fallará la validación. Si no hay ningún error, el controlador de sesión permitirá el inicio de sesión correctamente.

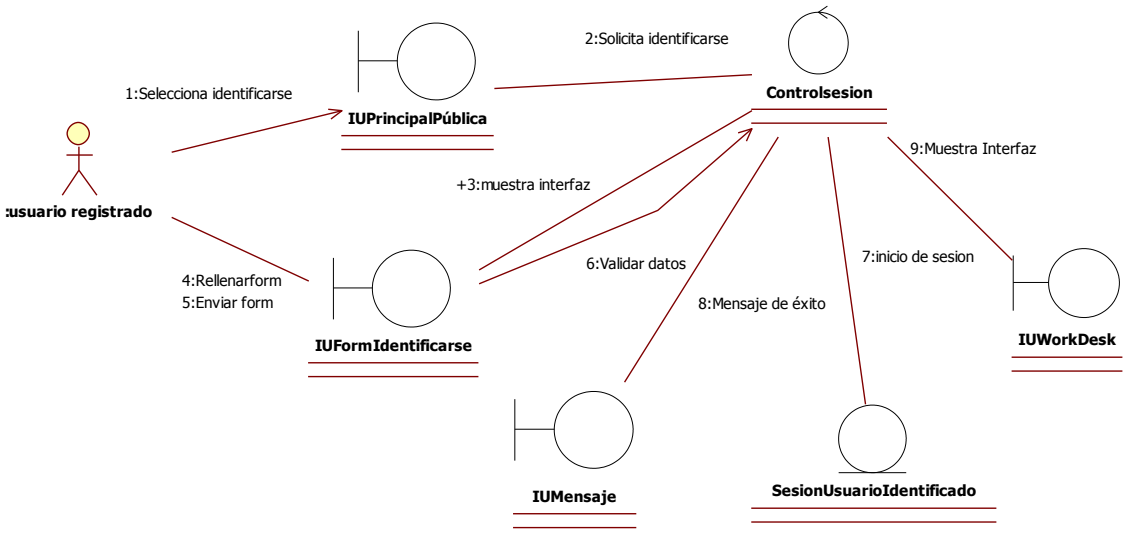


Diagrama de colaboración para identificarse en el sistema

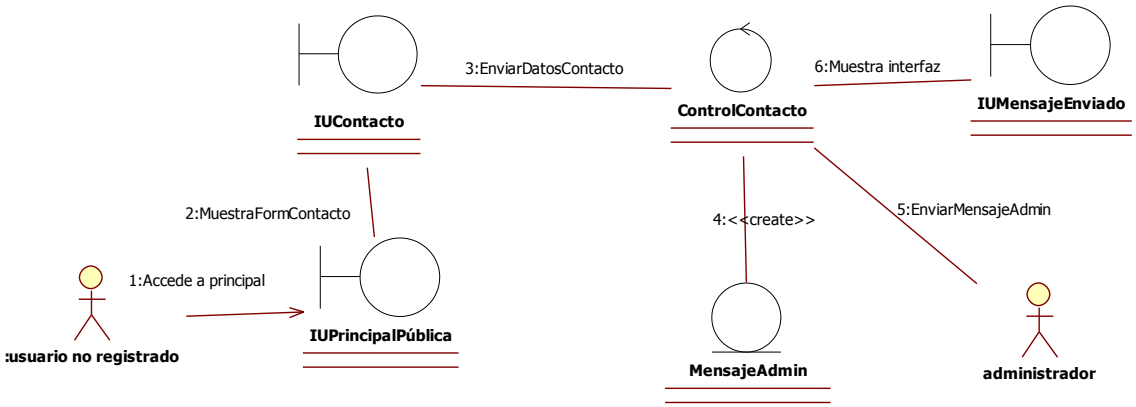


Diagrama de colaboración para Contactar con el administrador

5.3.1.2 Colaboraciones para la Gestión del Perfil

A continuación se muestra el proceso para la finalización de una sesión previamente iniciada.

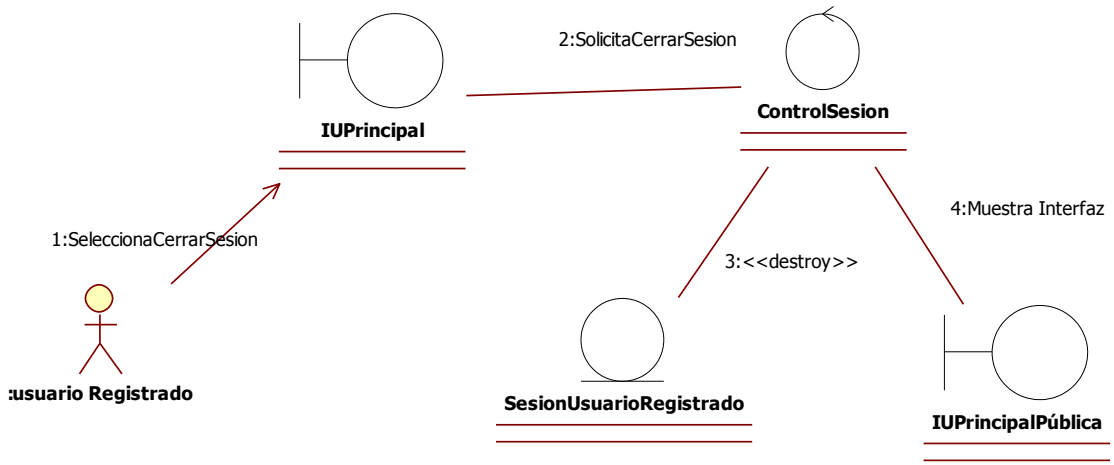


Diagrama de colaboración para cerrar sesión

Tras el registro e identificación, cada usuario es poseedor de un perfil cuenta. En él sealmacenan los datos personales de la cuenta y de configuración de la sesión. Dada la naturaleza cambiante de la información, el usuario puede modificar su perfil de cuenta para actualizar sus datos.

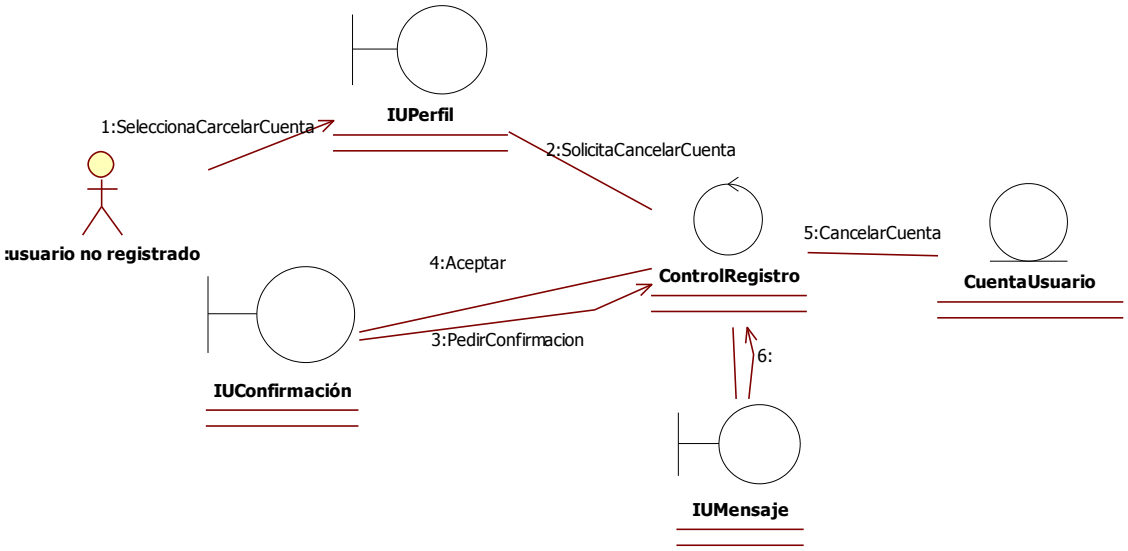


Diagrama de colaboración para cancelar cuenta

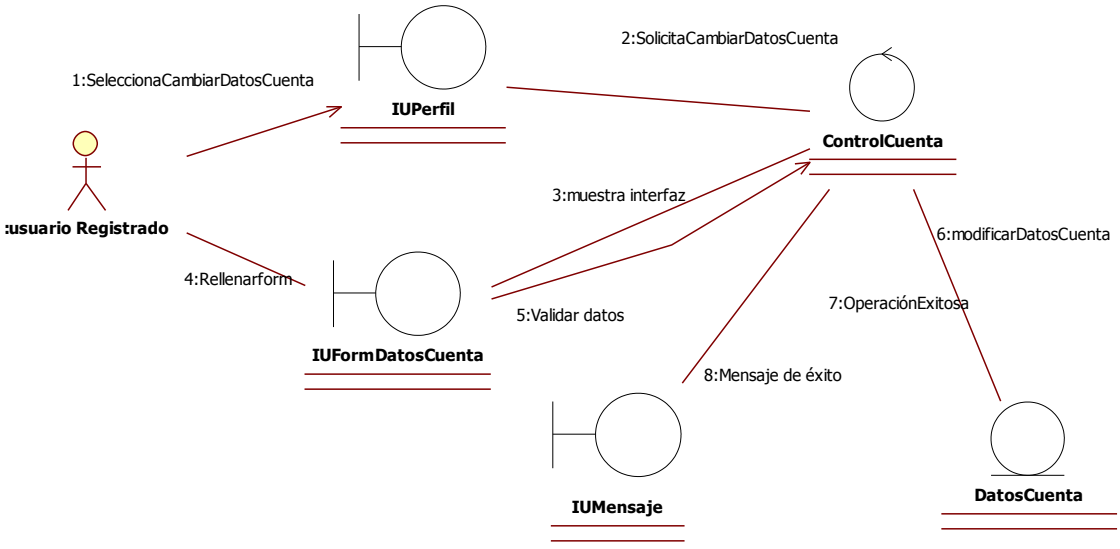


Diagrama de colaboración para modificar datos cuenta

5.3.1.3 Colaboraciones para la Gestión de Tareas

La primera de las funcionalidades que se analizará será la de gestión de tareas pertenecientes a un usuario registrado. Se selecciona la opción de añadir una nueva tarea y se rellenan los datos del formulario mostrado. En caso de éxito, se creará la entidad tarea.

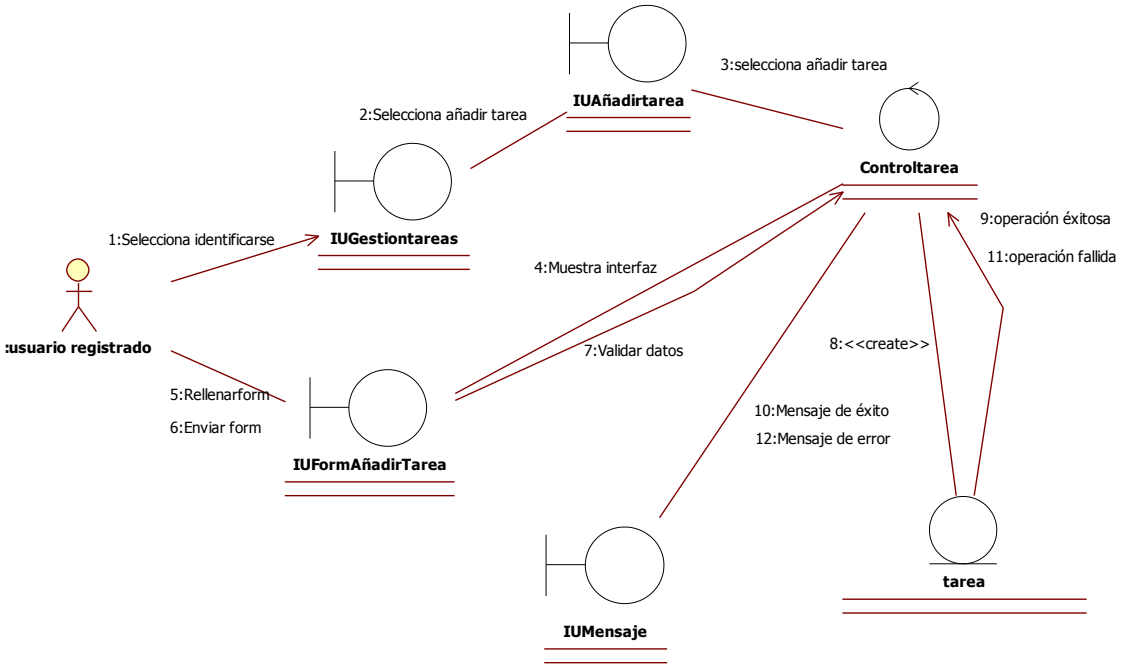


Diagrama de colaboración para añadir tarea

A medida que se van añadiendo tareas al sistema, un usuarios está en disposición de ver un listado con los mismos .

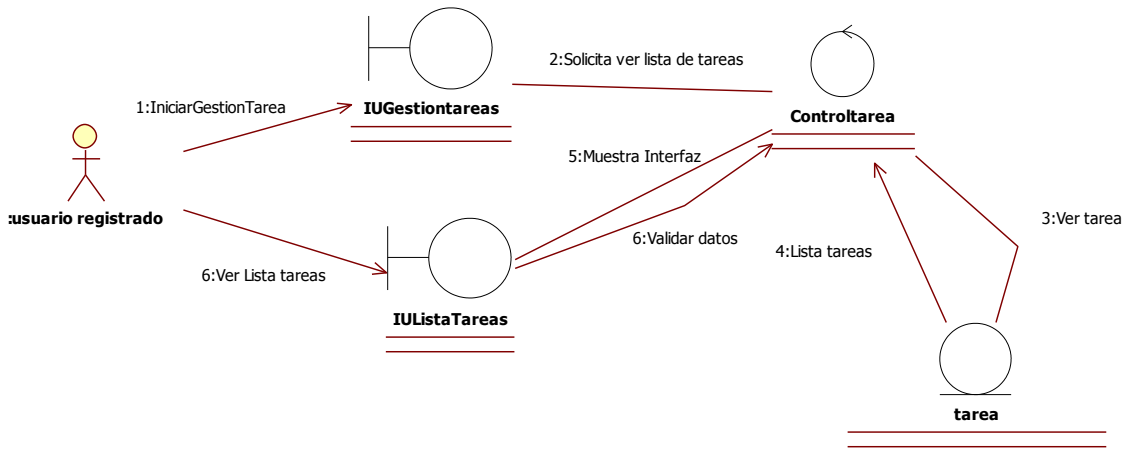


Diagrama de colaboración para ver listado de tareas

A continuación se muestra la colaboración para ver una tarea. El usuario registrado solicita ver la datos de una tarea determinado de la lista de tareas totales. Esto es gestionado por el controlador, mostrando una interfaz con los datos pertinentes.

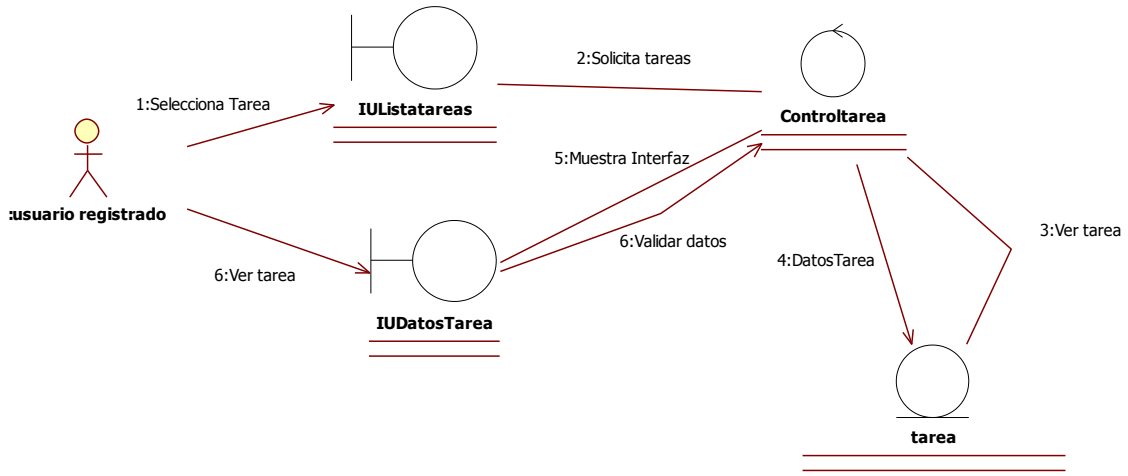


Diagrama de colaboración para ver tarea

Tanto para la modificación como eliminación de una tarea el proceso es muy similar. La salvedad es que uno muestra un formulario editable para poder actualizar la información y el otro elimina un registro concreto perteneciente al usuario registrado.

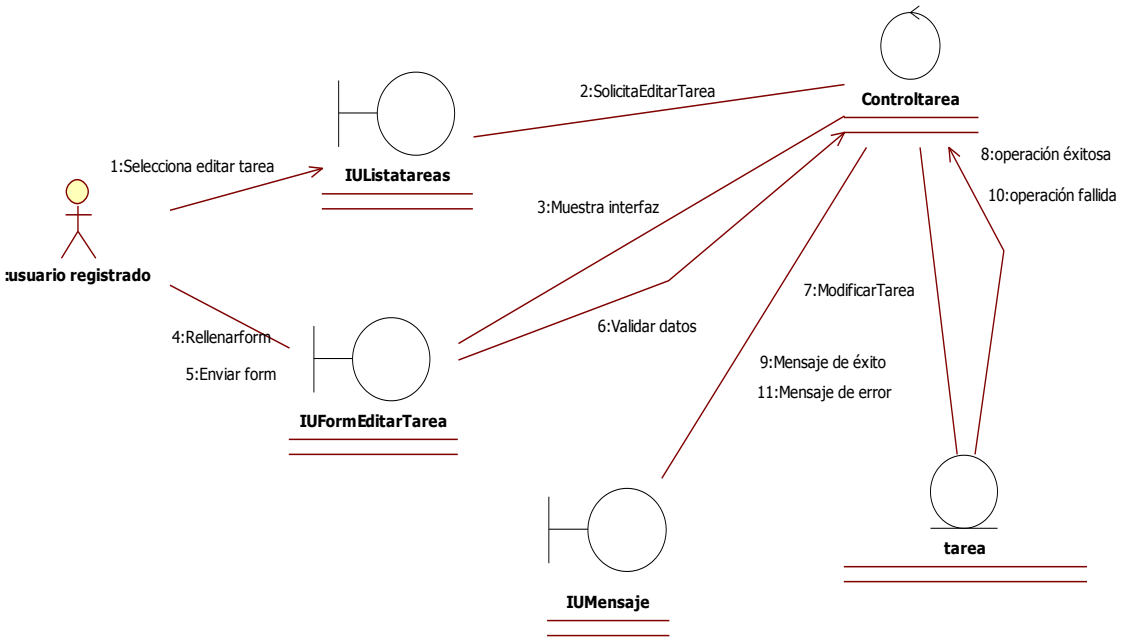


Diagrama de colaboración para modificar tarea

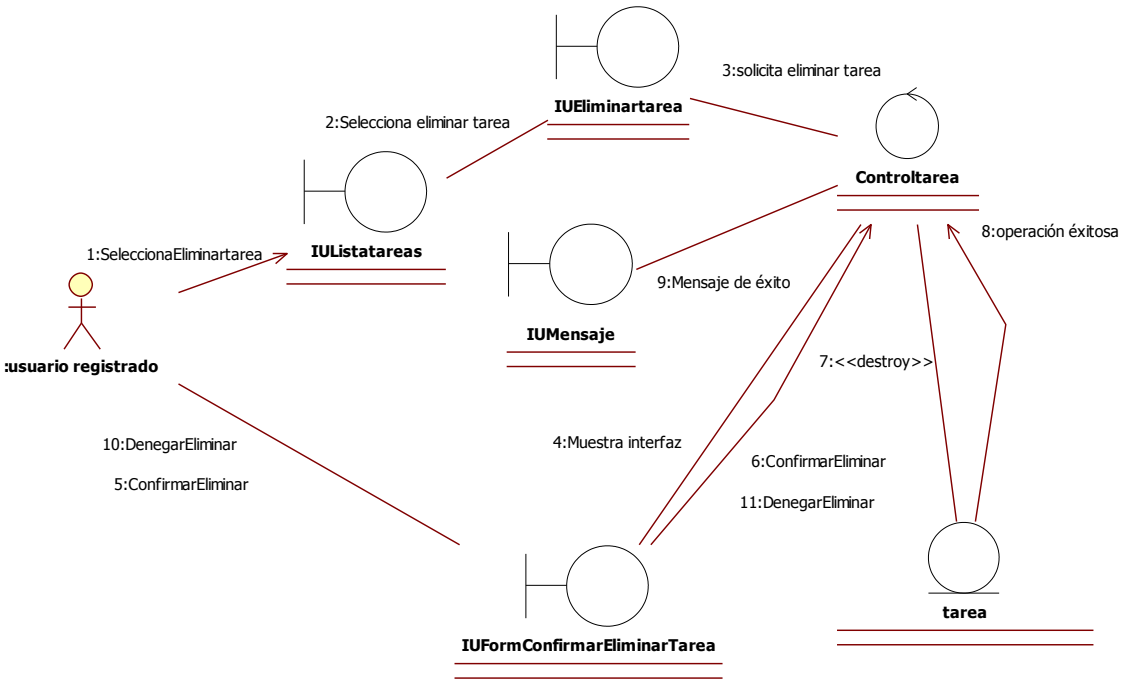


Diagrama de colaboración para eliminar tarea

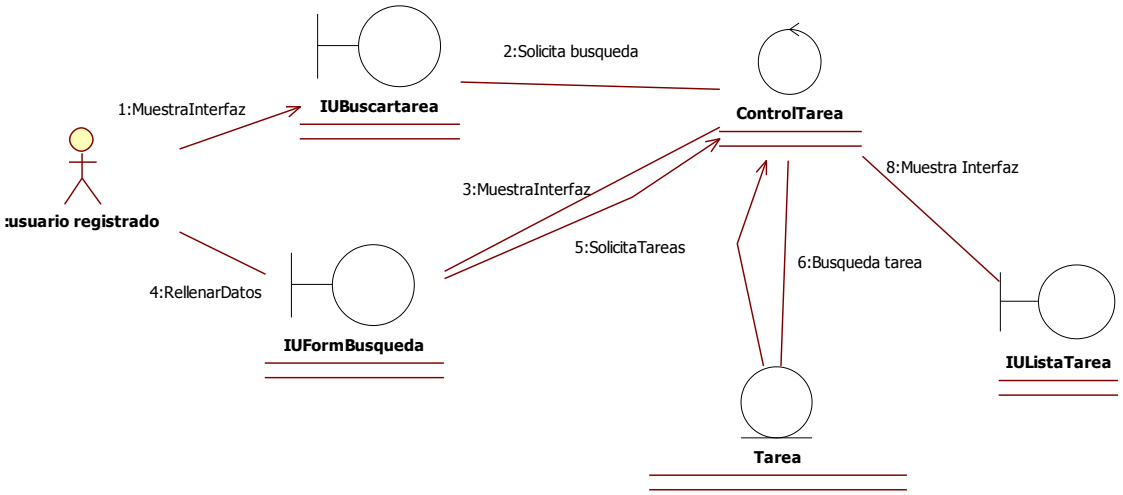


Diagrama de colaboración para búsqueda de tarea

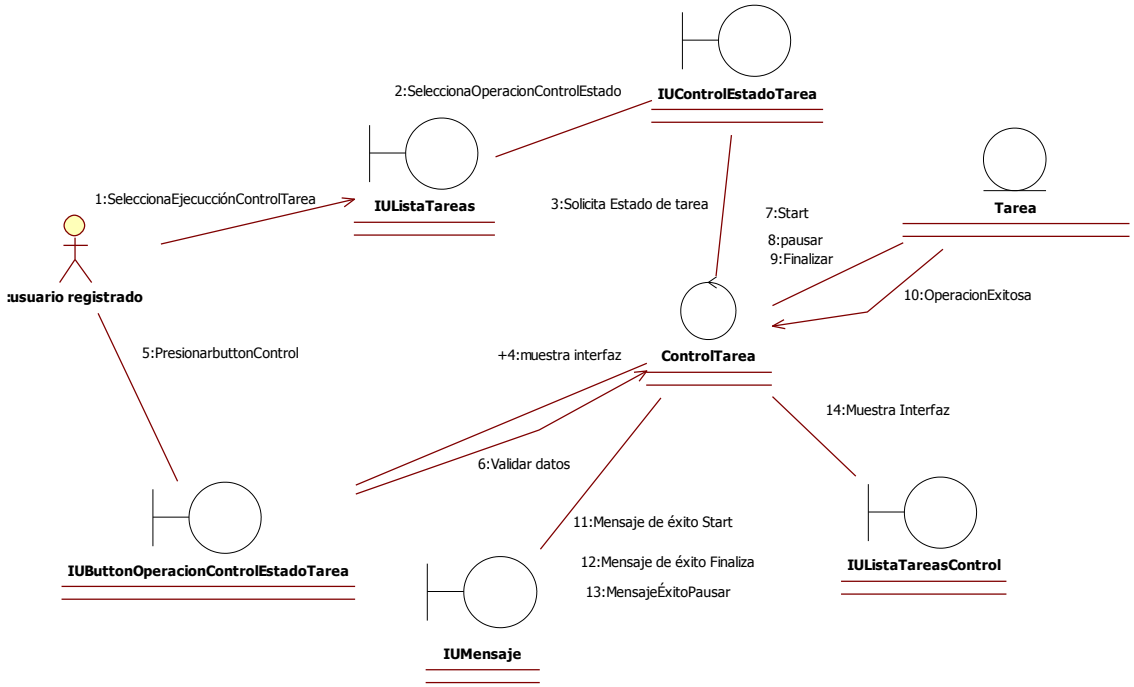


Diagrama de colaboración Ejecucción de una tarea

5.3.1.4 Colaboraciones para la Gestión de Proyectos

En este apartado de proyectos, se selecciona la opción de añadir una nuevo proyecto y se rellenan los datos del formulario mostrado. En caso de éxito, se creará la entidad proyecto.

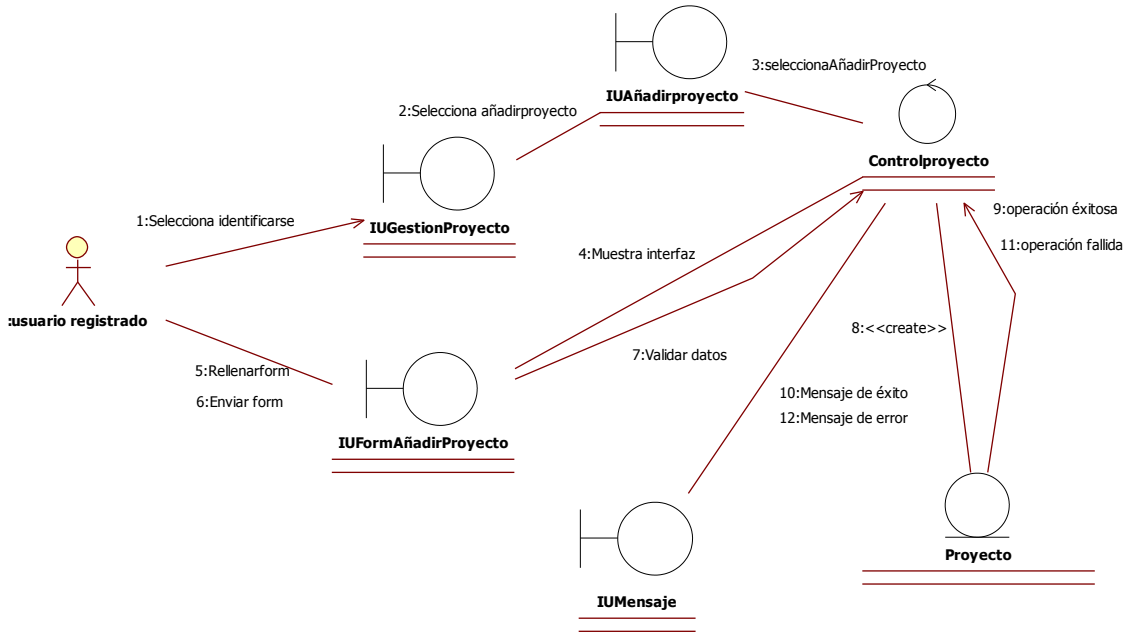


Diagrama de colaboración para añadir proyecto

A medida que se van añadiendo proyectos al sistema, un usuarios está en disposición de ver un listado con los mismos .

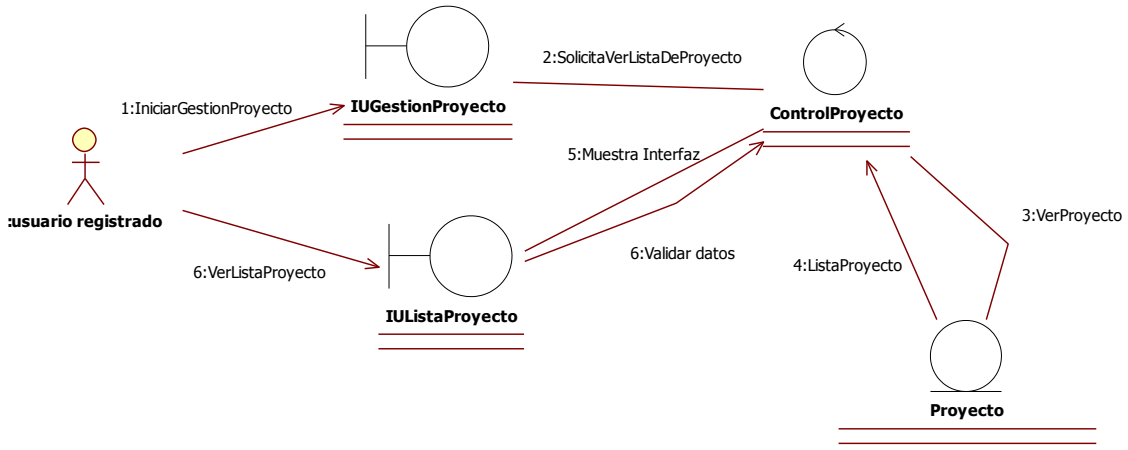


Diagrama de colaboración para ver listado de proyectos

A continuación se muestra la colaboración para ver un proyecto. El usuario registrado solicita ver la datos de un proyecto determinado de la lista de proyecto. Esto es gestionado por el controlador, mostrando una interfaz con los datos pertinentes.

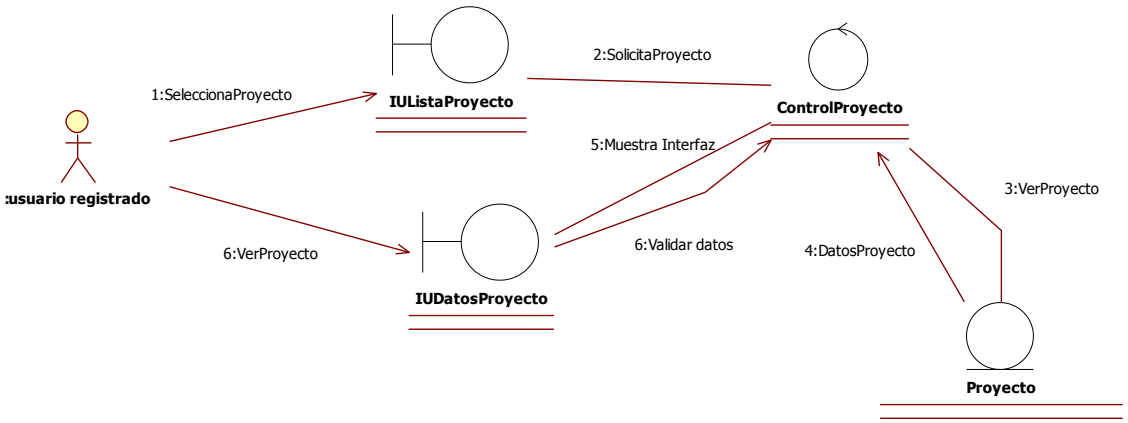


Diagrama de colaboración para ver proyecto

Tanto para la modificación como eliminación de un proyecto el proceso es muy similar. La salvedad es que uno muestra un formulario editable para poder actualizar la información y el otro elimina un registro concreto perteneciente al usuario registrado.

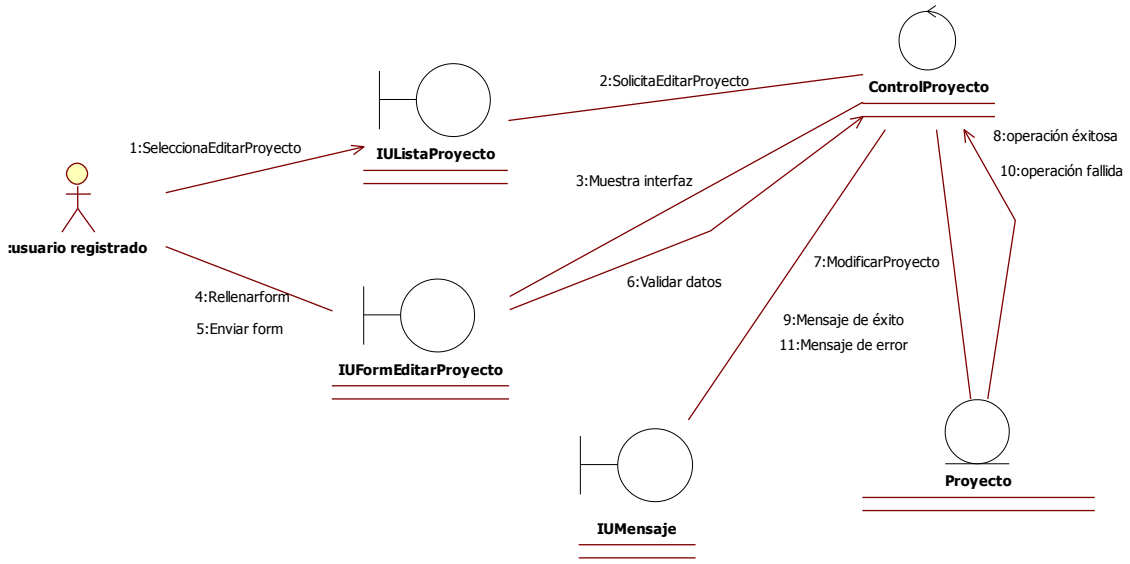


Diagrama de colaboración para modificar proyecto

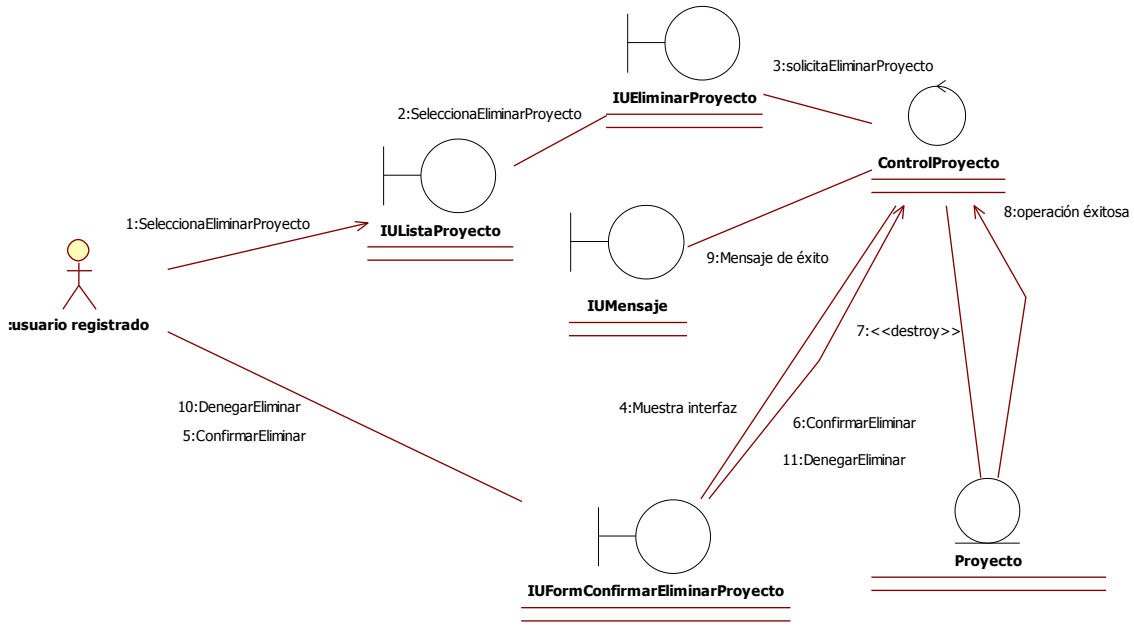


Diagrama de colaboración para eliminar proyecto

5.3.1.5 Colaboraciones para la Gestión de Interrupciones

Empezaremos con el diagrama de añadir una nueva interrupción y se rellenan los datos del formulario mostrado, creando a entidad interrupciones

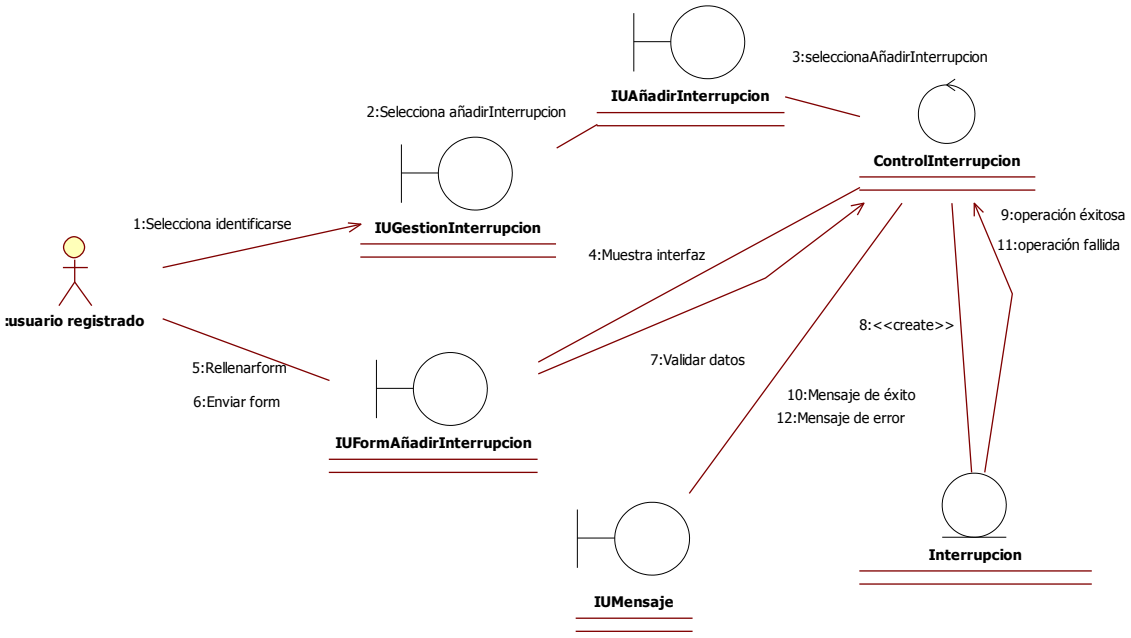


Diagrama de colaboración para añadir interrupciones

Los siguientes diagramas son para el control de estado de las interrupciones cuando quieres que empiece a contar el tiempo y cuando finaliza.

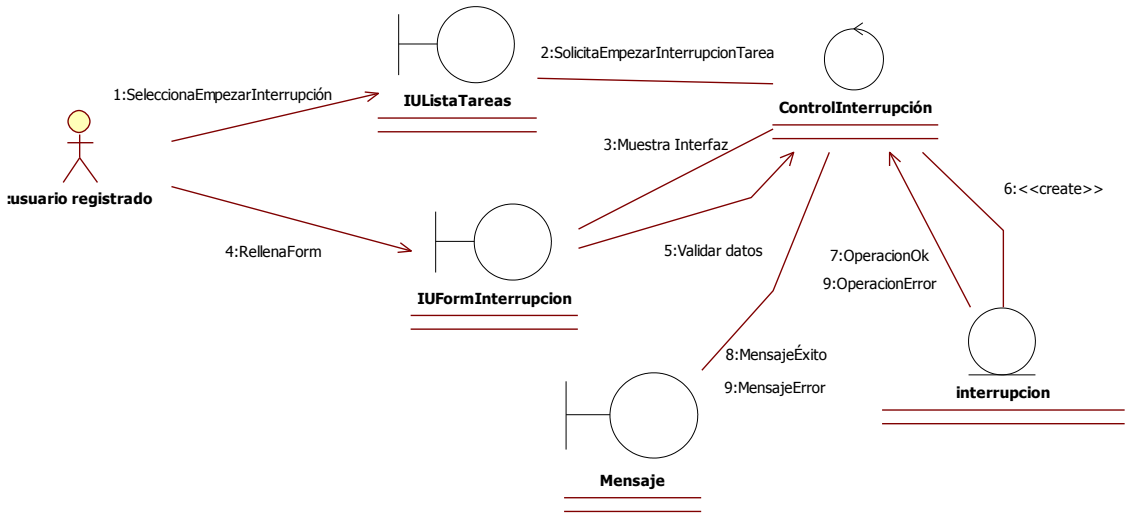


Diagrama de colaboración empezar interrupción

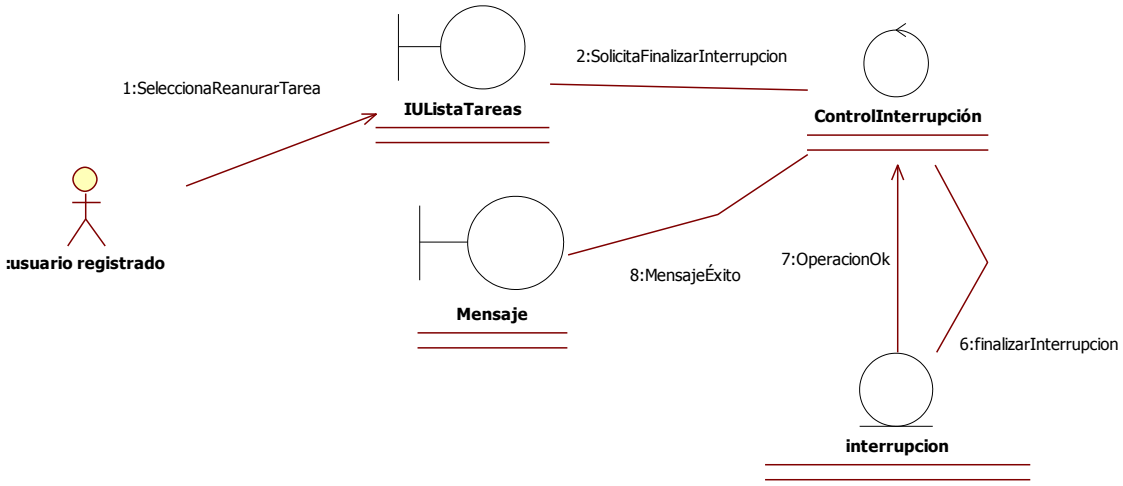


Diagrama de colaboración finalizar interrupción

5.3.1.6 Colaboraciones para la Gestión de Gtd

Los diagramas asociados a la gestión GTD se exponen a continuación:

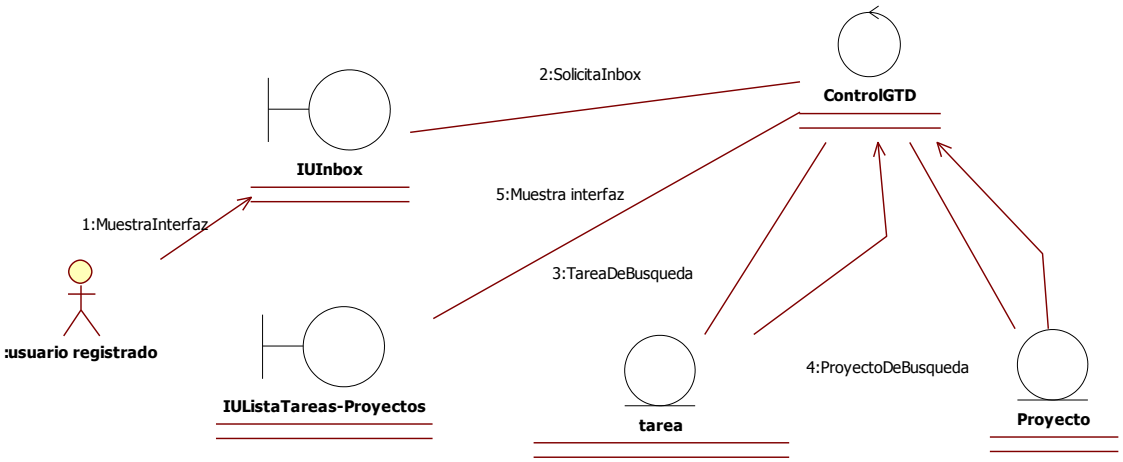


Diagrama de colaboración inbox

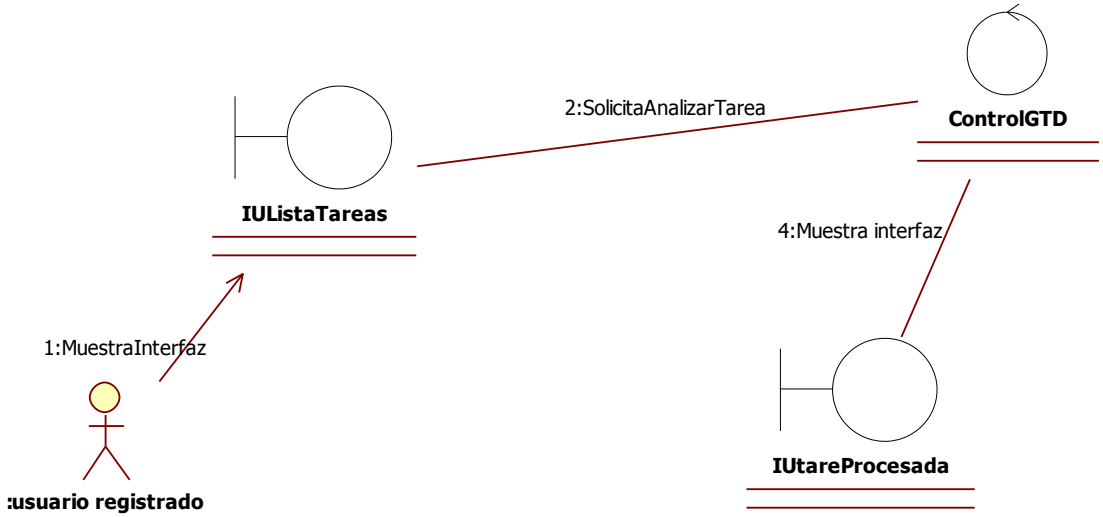


Diagrama de colaboración de analisis del algortimo GTD

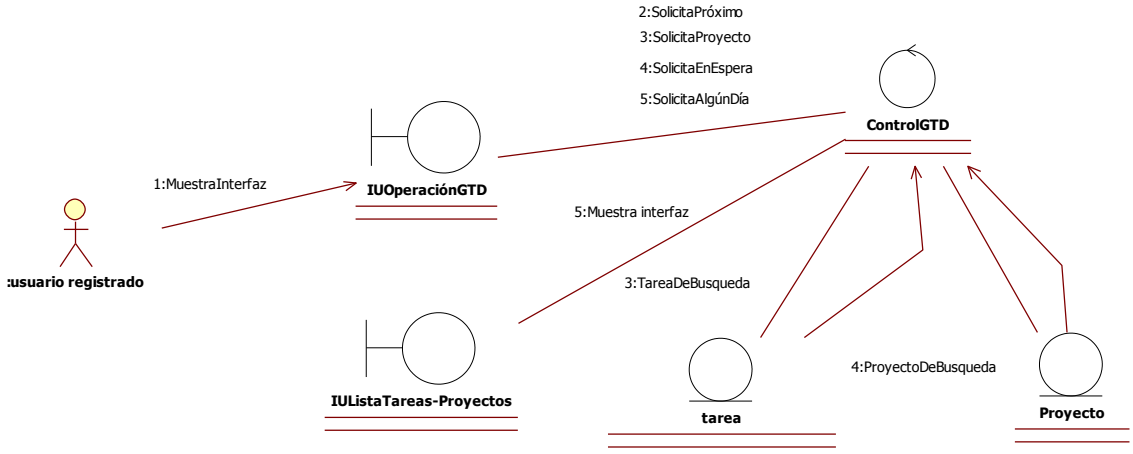


Diagrama de colaboración distribucción de acciones de GTD

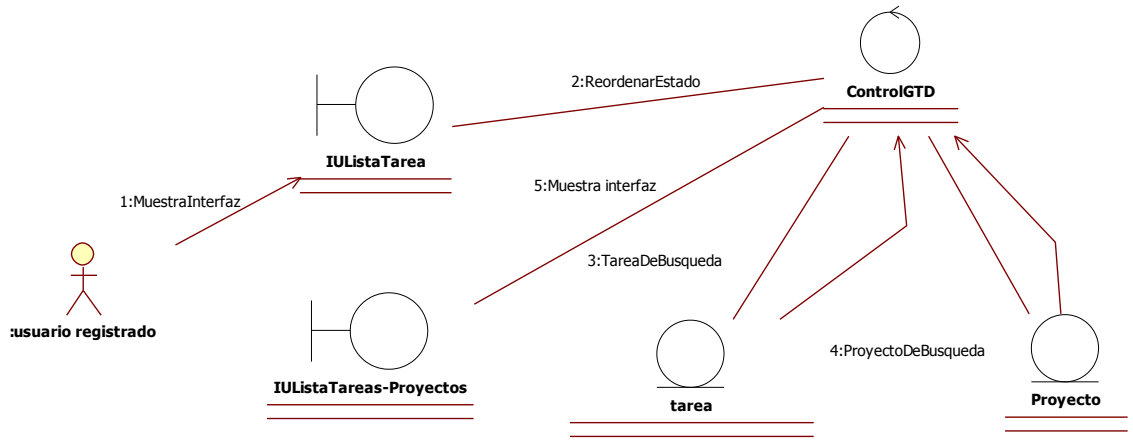
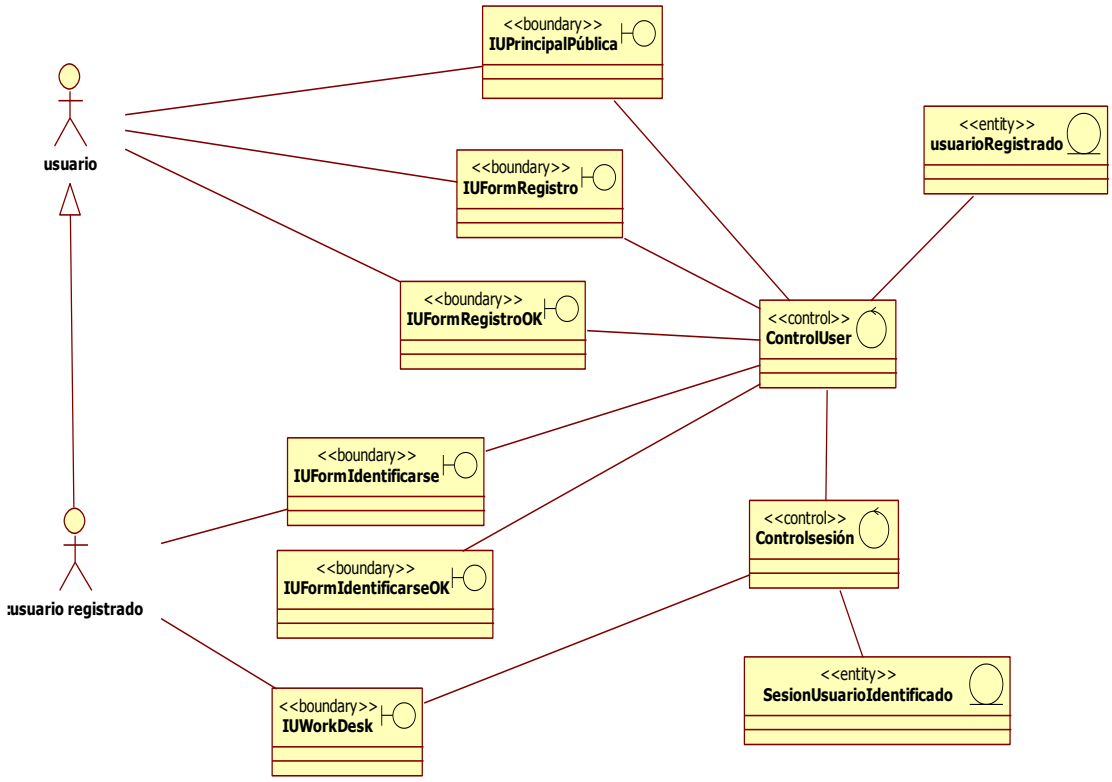


Diagrama de colaboración de Revisión de tareas GTD

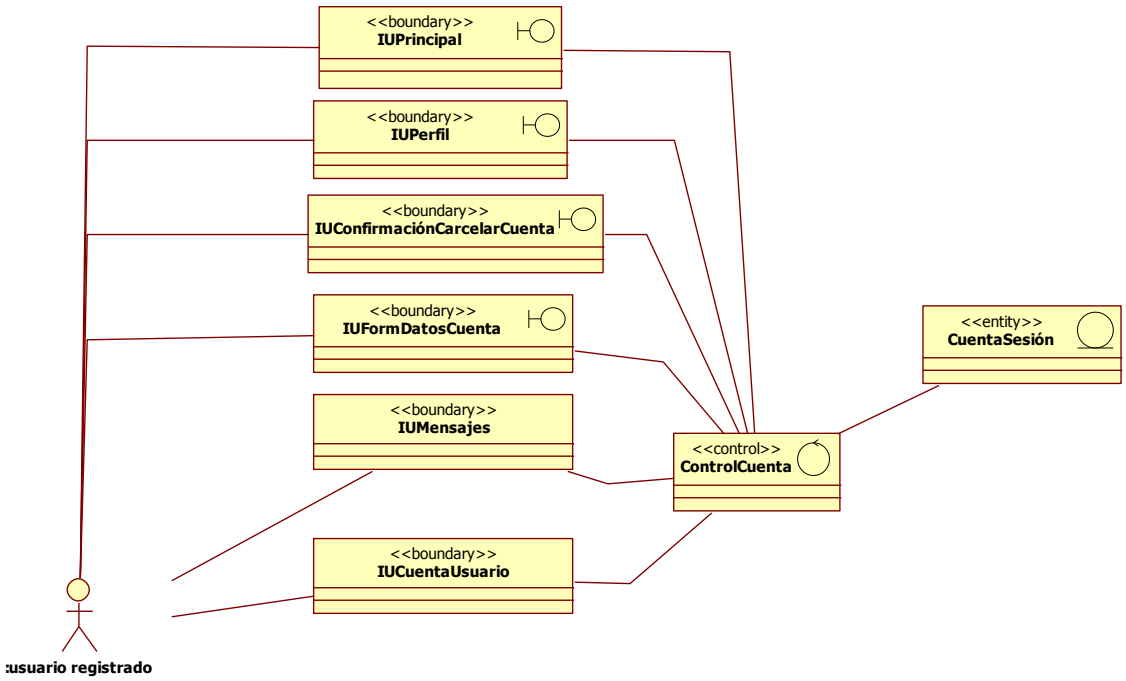
5.3.2 Realización de los Diagramas de Clases

En este apartado se mostrarán las clases que han intervenido en los diagramas de colaboración, mostrando la relación que existe entre cada una de ellas a nivel de conceptos. Dichas clases Se han estructurado en base a los paquetes principales de la aplicación, y por sencillez de algunos requisitos, no se representarán la totalidad de las características de la aplicación.

5.3.2.1 Diagrama de Clase de Actividades Generales

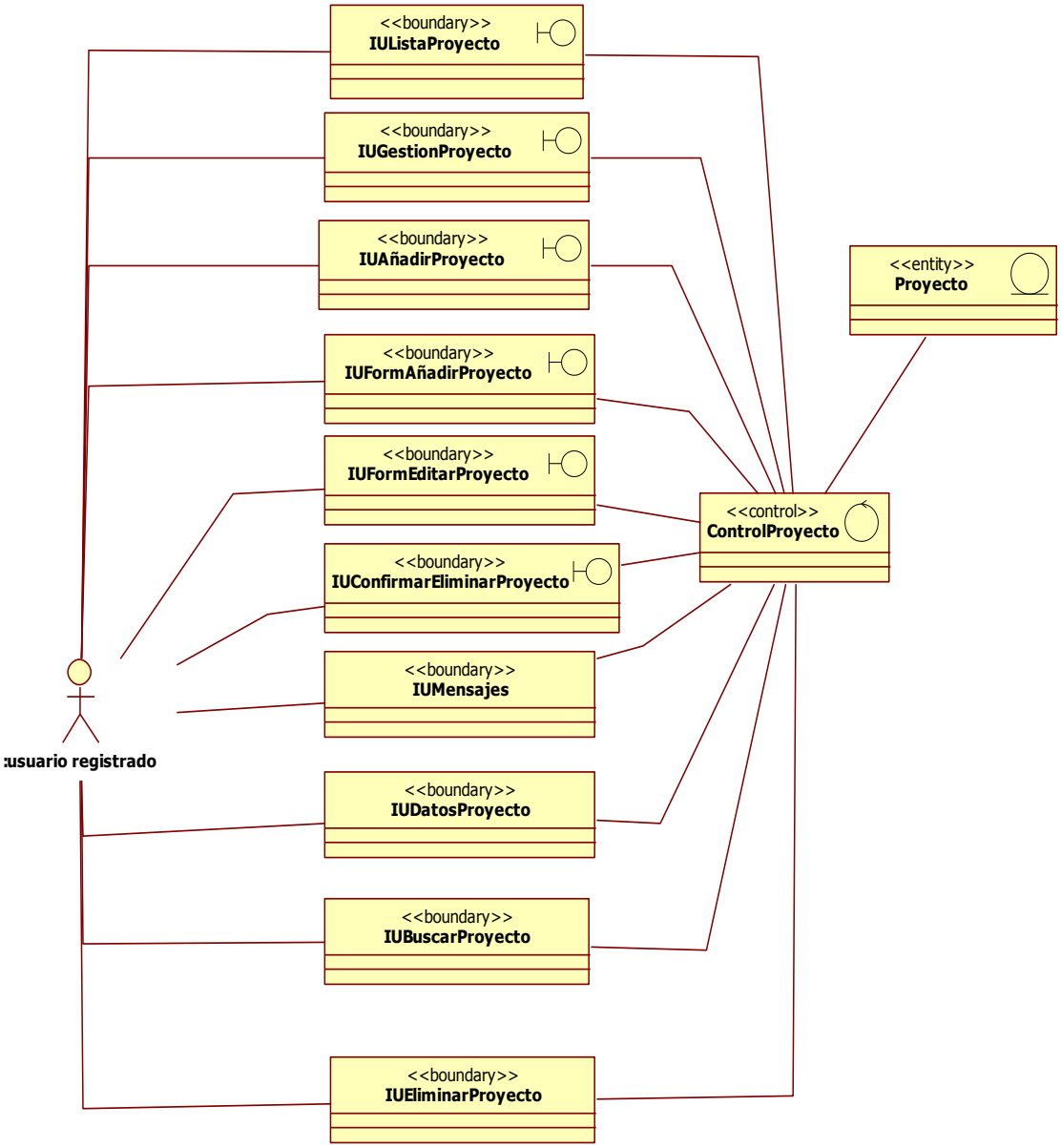


5.3.2.2 Diagrama de Clase de Gestión de perfil

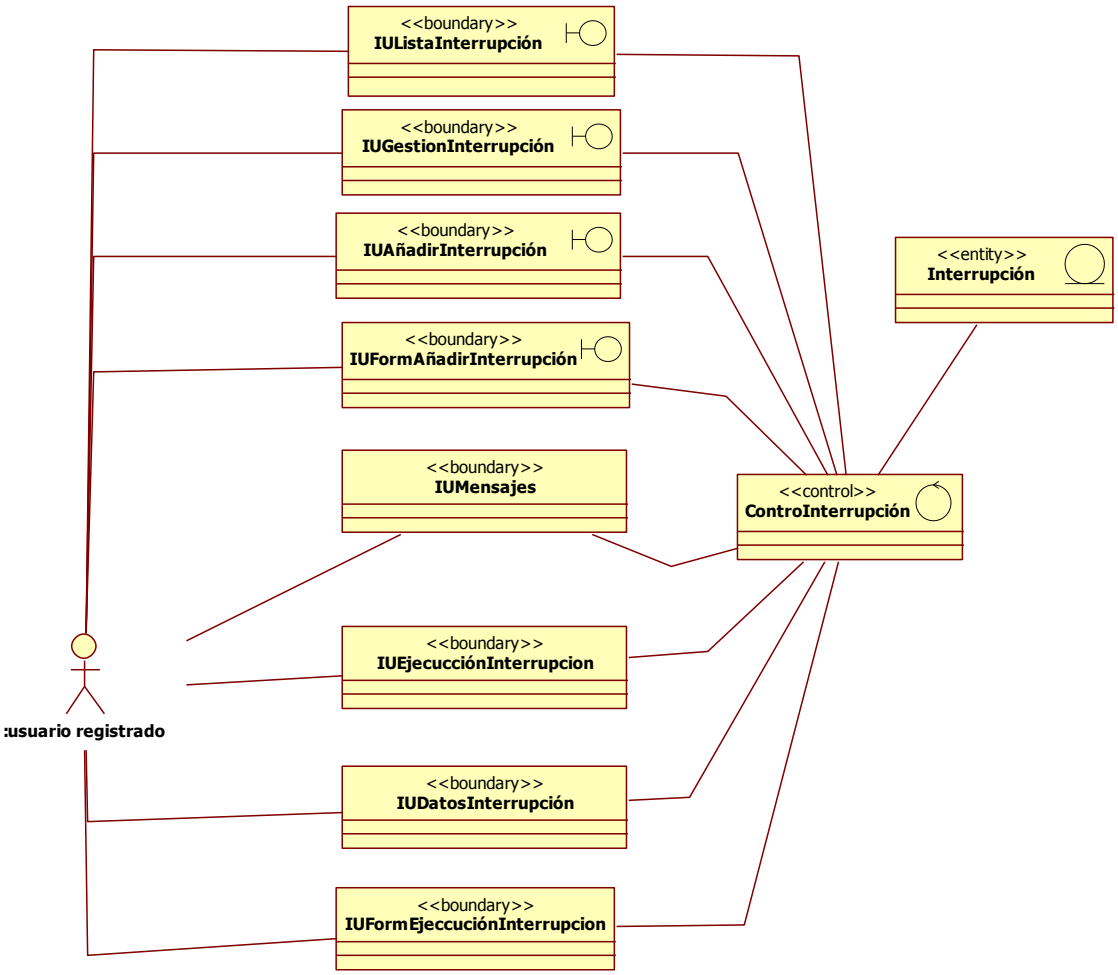


5.3.2.3 Diagrama de Clase de Gestión de Tareas

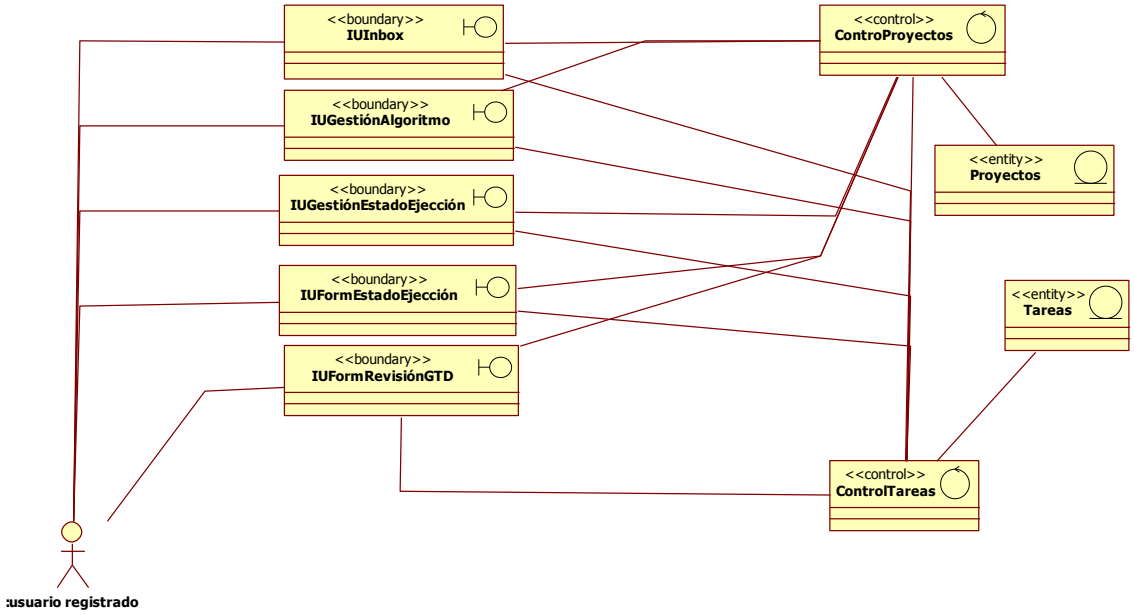
5.3.2.4 Diagrama de Clase de Gestión de Proyecto



5.3.2.5 Diagrama de Clase de Gestión de Interrupción



5.3.2.6 Diagrama de Clase de Gestión de Gtd

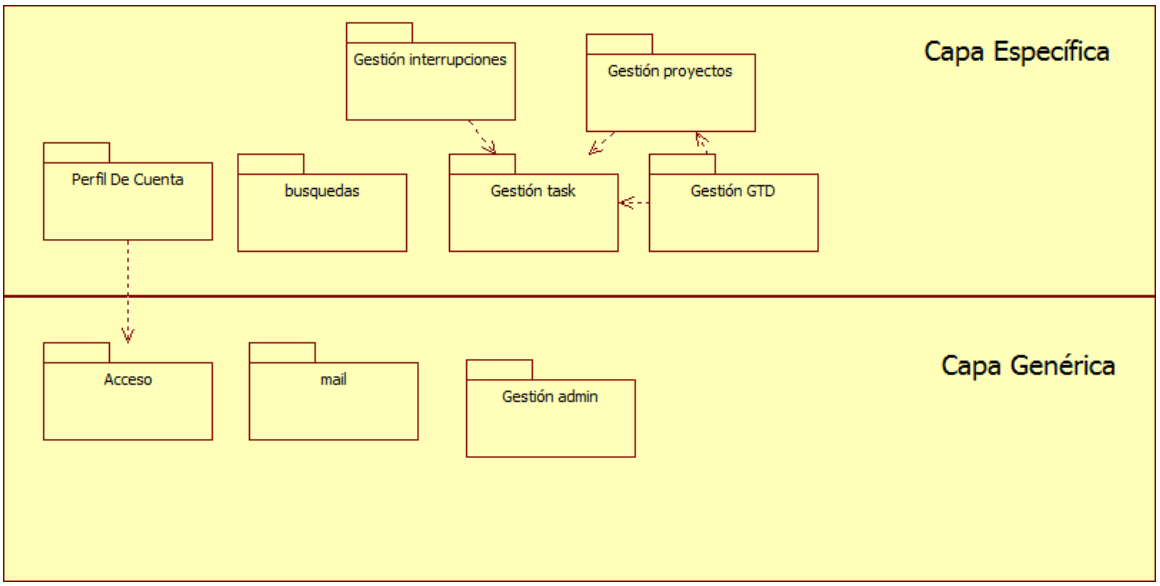


5.3.3 Diagramas de Paquetes

En esta sección lo que se desarrollará los diagramas de paquetes. El propósito es esbozar el modelo de análisis y la arquitectura mediante la identificación de paquetes del análisis, clases del análisis más evidentes y requisitos comunes. Los paquetes de análisis proporcionan un medio para organizar el modelo de análisis en piezas más pequeñas y más manejables. Normalmente la arquitectura del sistema estará dividida en diferentes capas:

- capa específica
- capa genérica
- capa middleware o entorno de trabajo
- capa del sistema operativo

En el análisis se dará forma a las dos primeras, lo que ayudará a tener una base para la fase de diseño.


Capa Específica
Gestión interrupciones
Gestión proyectos
Perfil De Cuenta
busquedas
Gestión task
Gestión GTD
Capa Genérica
Acceso
mail
Gestión admin

5.4 Diseño

En este capítulo se establecerá la actividad de diseño, para ello se partirá del modelo de análisis ya que nos proporciona una compresión detallada de los requisitos. El diseño constituye una representación coherente y bien planificada de los programas, concentrándose en las interrelaciones de los componentes y en las operaciones lógicas implicadas .

El diseño nos proporciona adquirir una compresión profunda sobre aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, tecnologías de interfaz de usuario, etc. Debemos ser capaces de descomponer los trabajos de implementación en partes más manejables. El diseño nos proporciona una entrada apropiada para futuras actividades de implementación capturando los requisitos o subsistemas individuales , interfaces y clases.

Para abordar este trabajo de diseño el proceso se dividirá en varias fases:

- **Diagrama de la arquitectura(véase apartado 5.4.2 diseño arquitectónico)**. Donde se han detallado los módulos que componen las vistas estática y dinámica de la aplicación, junto a los patrones que componen la arquitectura.
- **Diseño de clases con las relaciones entre modelos**. Detallando las asociaciones entre clases ActiveRecord que mapearán la información con las tablas de la base de datos, ya que Rails soporta la lógica de negocio en los modelos.
- **Diagrama de clases por paquetes con las relaciones entre MVC**. Este diagrama será con las relaciones entre los paquetes de diseño.
- **Diagrama de paquetes**. Termina de completar cómo interactúa cada componente entre sí.
- **Modelo de bases de datos**. Para la persistencia se diseña cada una de las tablas que luego se implementaran usando ActiveRecord.
- **Diagrama de despliegue**. Aprovechando el servidor local que incluye Rails, se modela el comportamiento de cada componente en tiempo de ejecución.
- **Prototipo de la interfaz**. Realización del diseño de las diferentes pantallas que contendrá la aplicación de cliente final.

5.4.1. Diseño Arquitectónico

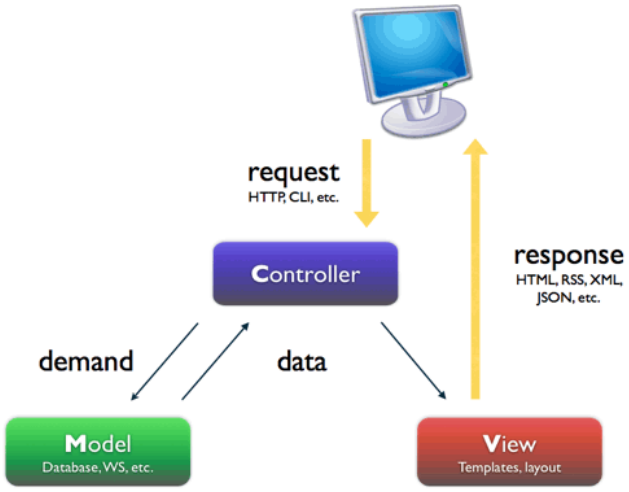
Diseño arquitectónico representa la estructura de datos y los componentes del programa que se requieren para construir un sistema, constituye el estilo arquitectónico que tendrá el sistema, la estructura y las propiedades de los componentes que ese sistema comprende, y las interrelaciones que tienen lugar entre todos los componentes arquitectónicos del sistema. El diseño arquitectónico por lo tanto es la primera fase del proceso de diseño y representa un enlace crítico entre los procesos de ingeniería de diseño y de requerimientos.

Desde un inicio se impuso el uso del framework Rails para el desarrollo del proyecto, decisión que conlleva evitar el tener que tomar algunas decisiones en cuanto a la arquitectura ya que rails nos proporciona una arquitectura base donde desarrollaremos los servicios de nuestra aplicación.

Antes de empezar con los modelos arquitectónicos es imprescindible hacer una breve introducción sobre los patrones de rails.

5.4.1.1 Patrón Mvc

El patrón MVC es un patrón de arquitectura de software en el cual se separan los componentes de una aplicación en tres capas, la capa de datos, la capa de interfaz y la capa lógica. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.



A continuación explicaremos cada una de éstas etapas más concretamente:

- **Modelo:** Es el encargado de administrar la lógica de tu aplicación y tiene como finalidad servir de abstracción de algún proceso del mundo real, tiene acceso a la Base de Datos. En Rails se utilizan principalmente para la gestión de las reglas de interacción con una tabla de base datos correspondiente usando ActiveRecord.
- **Vista:** Es simplemente la representación visual del modelo, es decir en otras palabras la interfaces usuario. Al tener esta capa desacoplada del modelo, no hace falta pensar todas las lógicas internas del sistema nuevamente, si no simplemente cambiar la parte visual del mismo. En Rails son en la mayoría de los casos ficheros HTML con código Ruby embebido donde se representa la información.
- **Controlador:** Es el escuchador de los eventos que genere el usuario, es decir es el que permite que interactúen el usuario con el sistema. Interpreta los eventos (la entradas) a través del teclado y/o ratón. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo de control generalmente es el siguiente:

1. El usuario realiza una acción en la interfaz

2. El controlador trata el evento de entrada.
3. El controlador notifica al modelo la acción del usuario, lo que puede implicar un cambio del estado del modelo (si no es una sólo una consulta)
    1. Se genera una nueva vista. La vista toma los datos del modelo. El modelo no tiene conocimiento directo de la vista.
    2. La interfaz de usuario espera otra interacción del usuario, que comenzará otro nuevo ciclo

5.4.1.2 Patrón Active Record

Active Record es el patrón de diseño que utiliza RoR para el acceso a datos. Es una clase para la administración y funcionamiento de los modelos. Esta clase proporciona la capa objeto-relacional que sigue rigurosamente el estándar ORM (Tablas en Clases, Registros en Objetos, y Campos en Atributos), facilita el entendimiento del código asociado a base de datos y encapsula la lógica específica haciéndola más fácil de usar para nosotros los programadores.

Las relaciones entre tablas también siguen unas convenciones predefinidas:

- "tiene uno" es una clave ajena
- "tiene muchos" es una clave en otra tabla
- "muchos a muchos" es una tabla intermedia.

Como he dicho anteriormente Active Record soluciona el problema que surge cuando desde una programación orientada a objetos se quiere acceder a los datos de bases de datos relacionales.

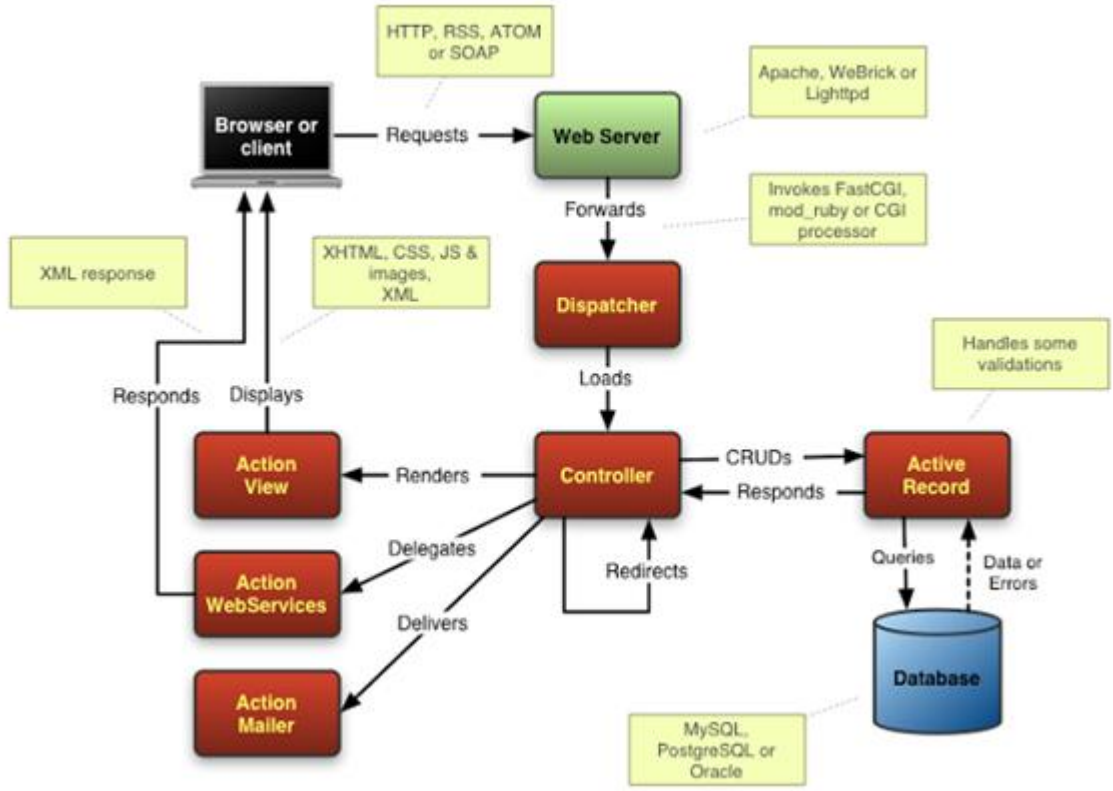
Además Active Record proporcionan otras ventajas como :.

- Las acciones del CRUD (Insertar, leer, modificar y eliminar) están encapsuladas así que se reduce el código y se hace más fácil de comprender.
- Código fácil de entender y mantener
- Se reduce el uso de código SQL considerablemente lo que implica cierta independencia del manejador de base de datos que usamos.

A continuación explicaremos cómo funciona el flujo de control de acciones de rails:

1. Cliente realiza una petición a través de una vista en un navegador, usando alguno de los protocolos soportados
2. El servidor tramita la petición y la envía al enrutador de Rails
3. El enrutador mapea la petición y ejecuta el par controlador/acción que se encarga de ella
4. El controlador puede:
    - redireccionar a otros controladores si es necesario
    - ejecutar acciones sobre el modelo, que por lo general coincide con peticiones CRUD
    - renderizar una vista
    - renderizar la información en otro tipo de datos, tal como XML;
    - pasar a un servicio de correo
5. En el caso de que el controlador necesite de la lógica de modelo, Active Record se encarga del mapeo objeto-relacional con la base de datos

6. En el caso de que se requiera renderizar una vista, Action View entra en juego mostrando la respuesta a la petición inicial en forma de vista



A continuación se detallan brevemente cada uno de los módulos que componen la arquitectura.

1. **Action Mailer**. Este módulo es el encargado de proveer el servicio para el envío de emails. Es capaz tanto de procesar emails entrantes como de crear nuevos que posteriormente serán enviados.

2. **Action Pack**. El módulo de Action Pack es quién gestiona las capas de controladores y vistas del patrón MVC. Los módulos que lo componen capturan las peticiones de los usuarios a través del navegador y las mapean en acciones. Estas acciones están incluidas en la capa de controladores, siendo las encargadas de hacer las llamadas para renderizar la vista que corresponda. Action Pack está dividido en tres módulos Action Dispatch, Action Controller y Action View.

- **Action Dispatch**. Maneja el enrutamiento de las peticiones a través del navegador. Parsea peticiones web y realiza procesamiento avanzado HTTP, tales como gestión de cookies ´o sesiones.
- **Action Controller**. Después de que el enrutador (Action Dispatch)procese una petición, se ejecuta el controlador encargado de manejarla. De este módulo es de donde heredarán el resto de controladores de la aplicación web. Action Controller contiene acciones que controlan los modelos y las vistas: accede a la información de los modelos cuando sea necesaria, renderiza una vista o re direccionada hacia

otro controlador. Así mismo, también es el encargado de manejar las sesiones y el flujo general de la aplicación.

- **Action View.** Es llamado por Action Controller. Renderiza una vista pedida por el usuario a través del navegador. Este módulo provee de mecanismos para el uso de: plantillas para generación de vistas; helpers que ayuden con la lógica y generación de HTML; métodos para representar la información en otros formatos.

3. **Active Model**. Define la interfaz entre los módulos de Action Pack y Active Record.

4. **Active Record**. Módulo usado para crear clases del modelo, los cuáles contienen la lógica de negocio, manejan las validaciones y relaciones, mapean objetos a tablas, y soporte para el uso de diferentes SGBD.

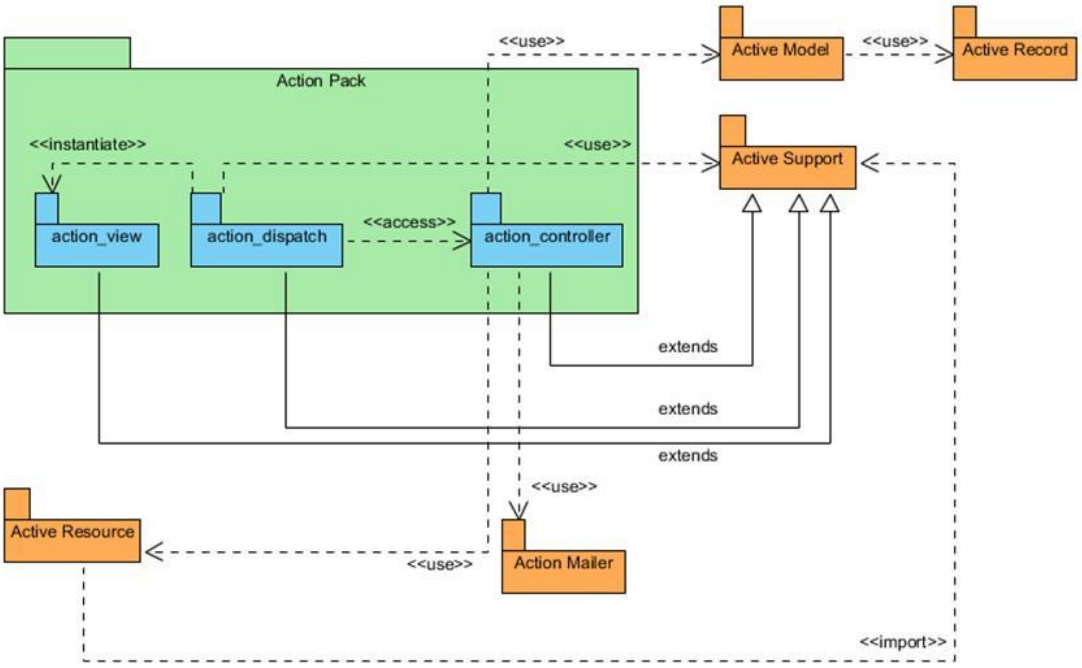
5. **Active Resource**. Se usa para manejar las conexiones entre los servicios RESTful y los objetos que controlan la lógica de negocio. Sigue el mismo principio que Active Record: el reducir la cantidad de código necesario para gestionar recursos. Active Resources mapea clases del modelo con recursos REST de la misma manera que Active Record lo hace entre clases de modelo y tablas de la base de datos.

6. **Active Support**. Es una colección de clases de soporte y librerías de extensión de Ruby que son útiles en el desarrollo sobre Rails. Incluyen un completo soporte para tratamiento de ristras, internacionalización, tratamiento de fechas/horas y testing.

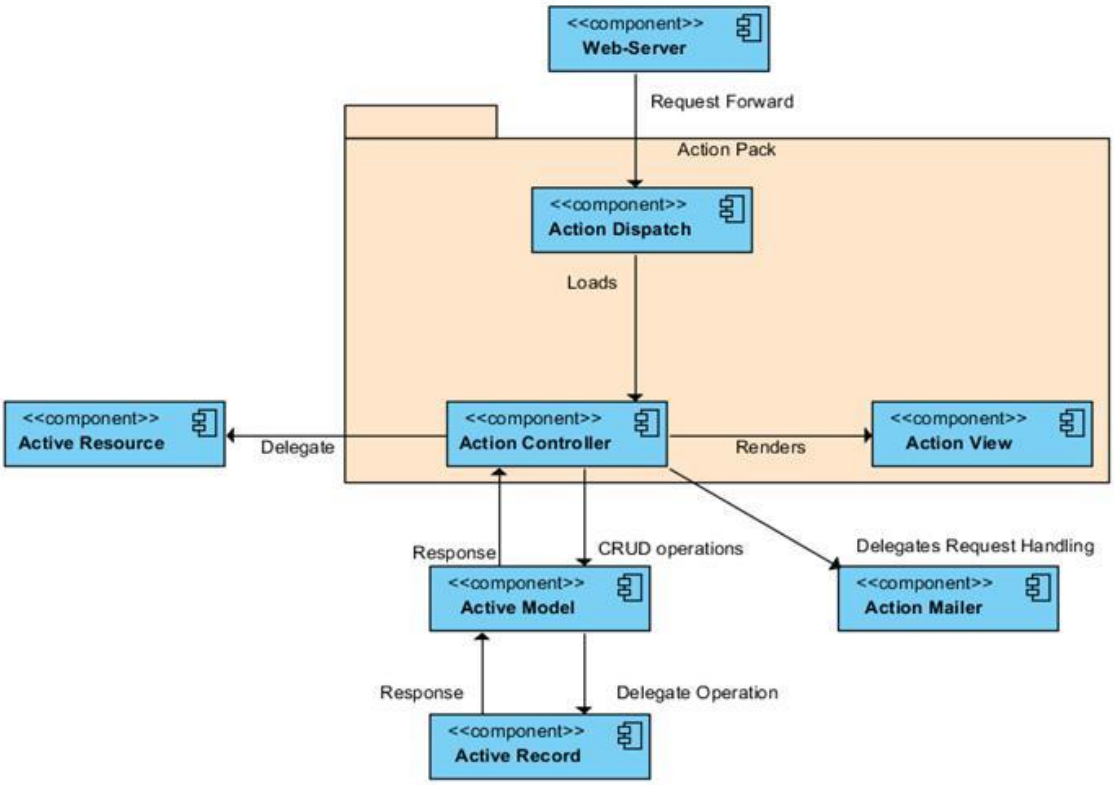
7. **Railties.** Es el código del núcleo de rails que se encarga de construir nuevas aplicaciones. Es el responsable de la unión de todos los módulos anteriores. Además, se encarga de todo el proceso de bootstrapping, la interfaz de línea de comandos y proporciona generadores de código.

Las vistas seleccionadas para analizar la arquitectura son: vista de módulos y vista de componentes.

1. La vista de módulos contiene los diagramas UML que representan la vista estática de todos los componentes.



2. Vista de componentes muestra los diagramas con la arquitectura del sistema de forma dinámica, qué componentes existen en el momento de ejecución y cómo se comunican entre sí.



5.4.2 Diagramas de Clases

En la introducción se detalla brevemente el proceso seguido para definir la etapa de diseño de la aplicación.

En este apartado en concreto, haremos referencia a los diagramas de clases. El propósito de este diagrama es el de representar los objetos fundamentales del sistema, es decir un diagrama de clase describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

Para realizar un diagrama de clases, como primer paso se necesita esbozar una o varias clases del diseño, dada la entrada en términos de clases de análisis o interfaces. Si tomamos una clase de diseño para que proporcione una interfaz los métodos utilizados es dependiente de la tecnología de interfaz específica que se utilice. En este caso en particular, al ser un dispositivo web y móvil hacia los que está orientado la aplicación y al usar HTML5/CSS3, los estereotipos deberían ser elementos relacionados con la clase, sin embargo para simplificar los diagramas de clase usaremos ((ActionView)) como estereotipo.

Normalmente las aplicaciones necesita que se diseñe clases de entidades que represente información persistente, para ello existe el uso de tecnologías de base de datos. En Rails estas clases están relacionadas con((ActiveRecord)) y, en general, son denominadas clases del modelo.

Cuando diseñamos las clase de control , estas clases son una tarea dedica ya que encapsulan secuencias o coordinación entre otros objetos del sistema. Lo que se hará será que todos los controladores que creemos hereden de Application Controller convirtiéndose este en el controlador padre de todos los demás controladores creados en la aplicación.

Durante la construcción del diseño se ha tenido que asumir varias imposiciones de diseño marcadas por el entorno de desarrollo que hemos utilizado:

- Como se detalló en la arquitectura, la aplicación se apoya en el patrón MVC donde el controlador interactúa con el modelo y con las vistas.
- Todos los controladores heredan de Application Controller.
- Todos los modelos heredan de ActiveRecord::Base, encargada de mapear objetos con las tablas de la base de datos. ActiveRecord::Base contiene métodos para las validaciones de datos de los modelos. En los modelos se establece la lógica de negocio. Es decir, en los modelos es donde se realiza las asociaciones entre entidades.
- Todas las vistas serán renderizadas por el módulo Action View, incluyendo soporte para el uso de helpers de Rails.
- Existe correspondencia directa entre los nombres de las acciones (métodos de clase) de los controladores y las vistas asociadas.

A continuación se mostrarán las principales clases que intervienen en la fase y que están apoyadas por la arquitectura del software:

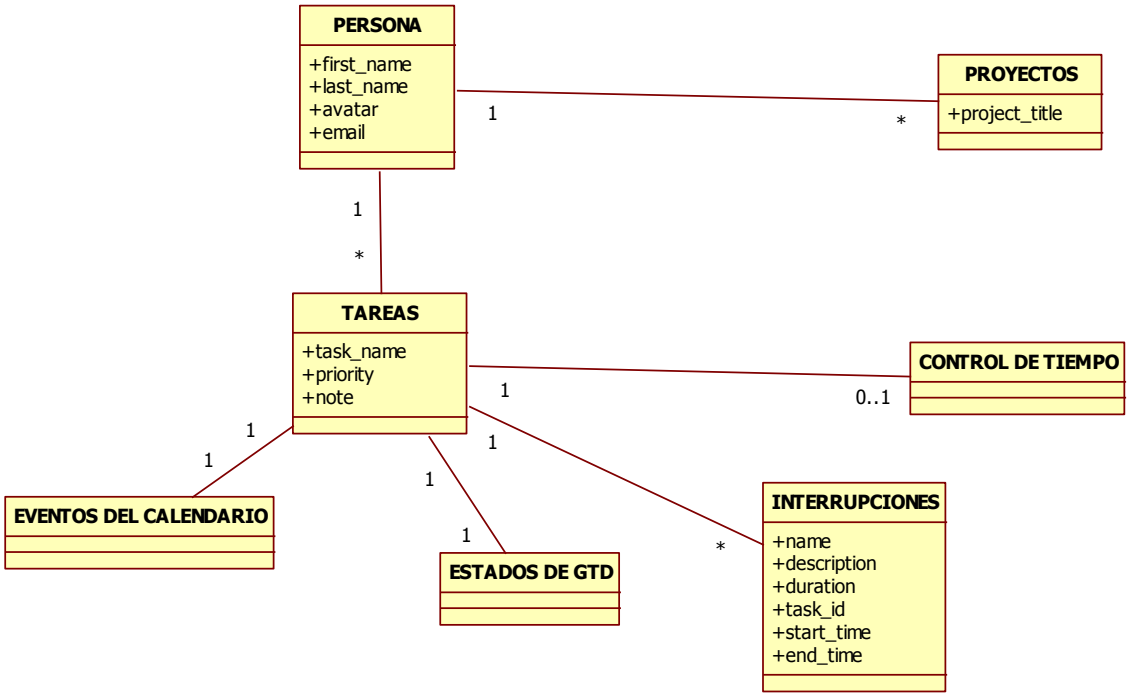


Diagrama de clase entre modelos

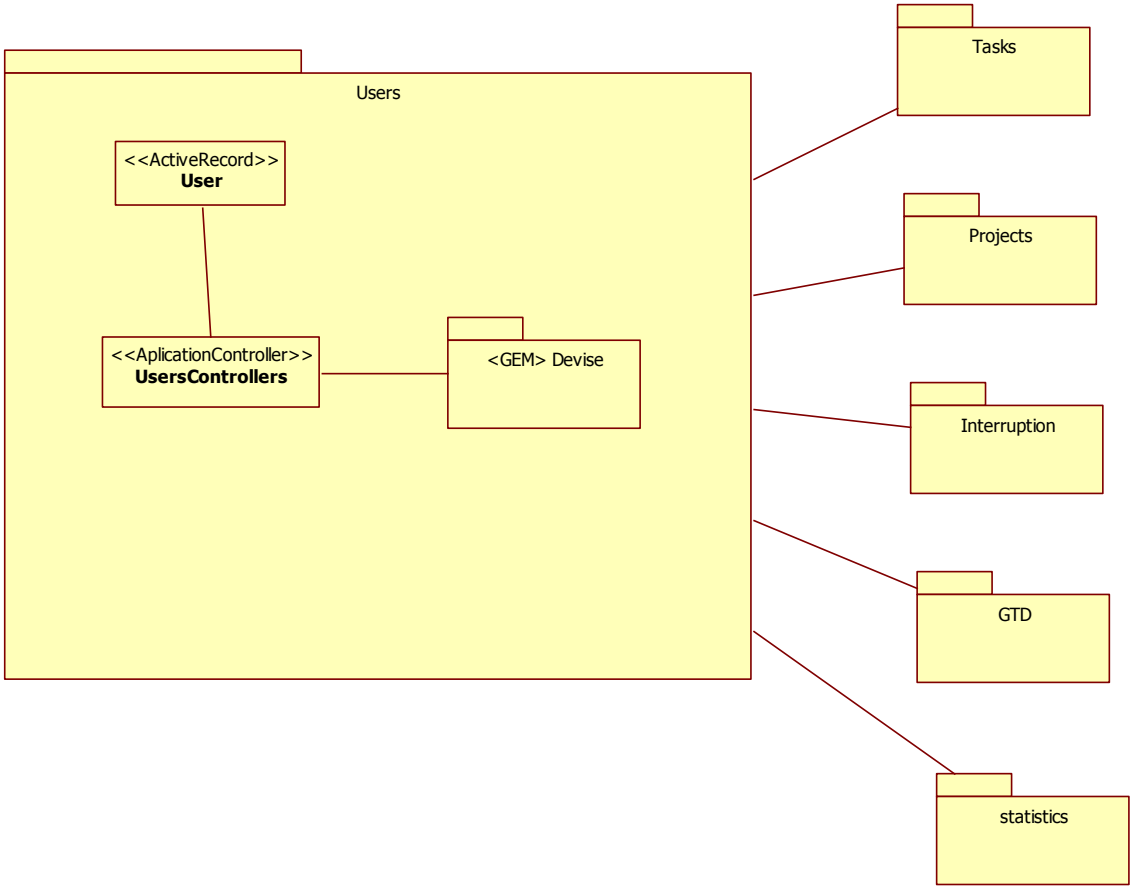


Diagrama de clase MVC-User

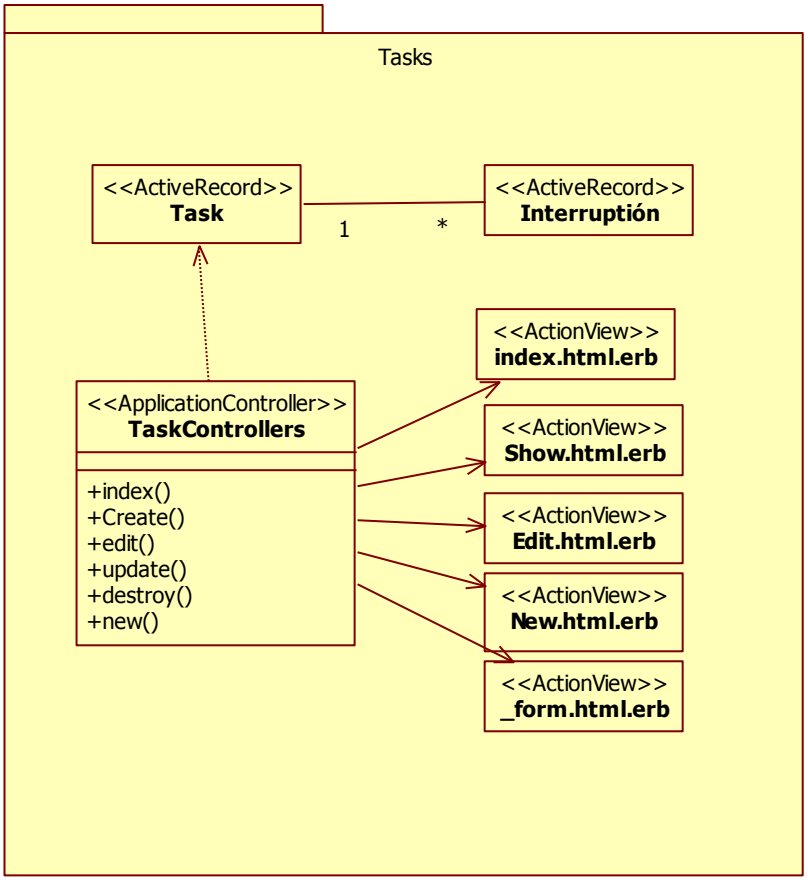


Diagrama de clase MVC-Task

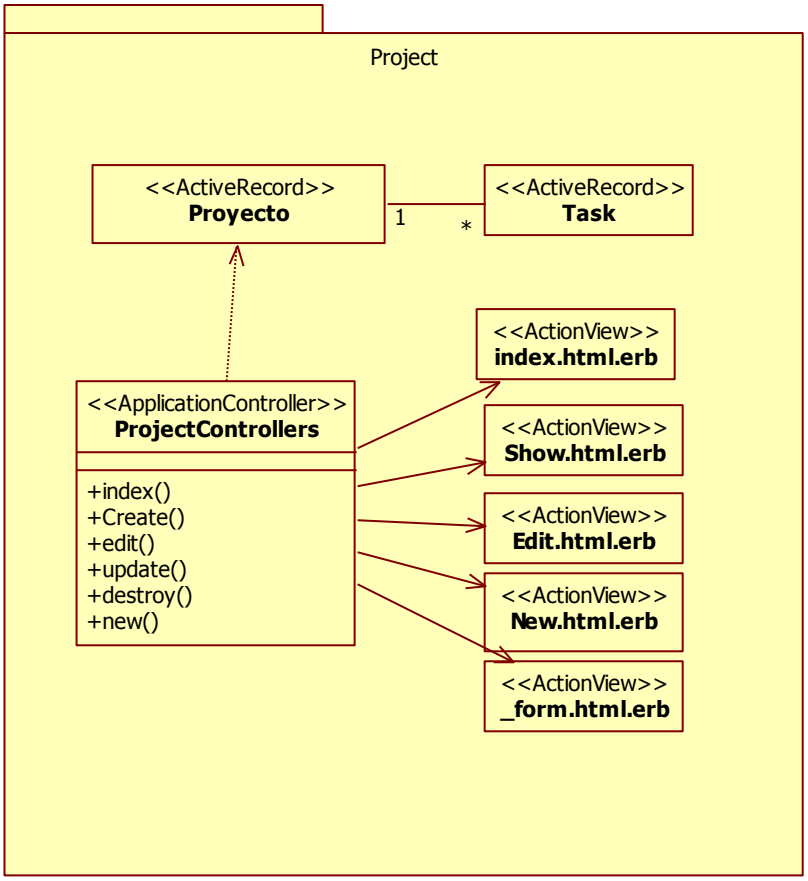


Diagrama de clase MVC-Project

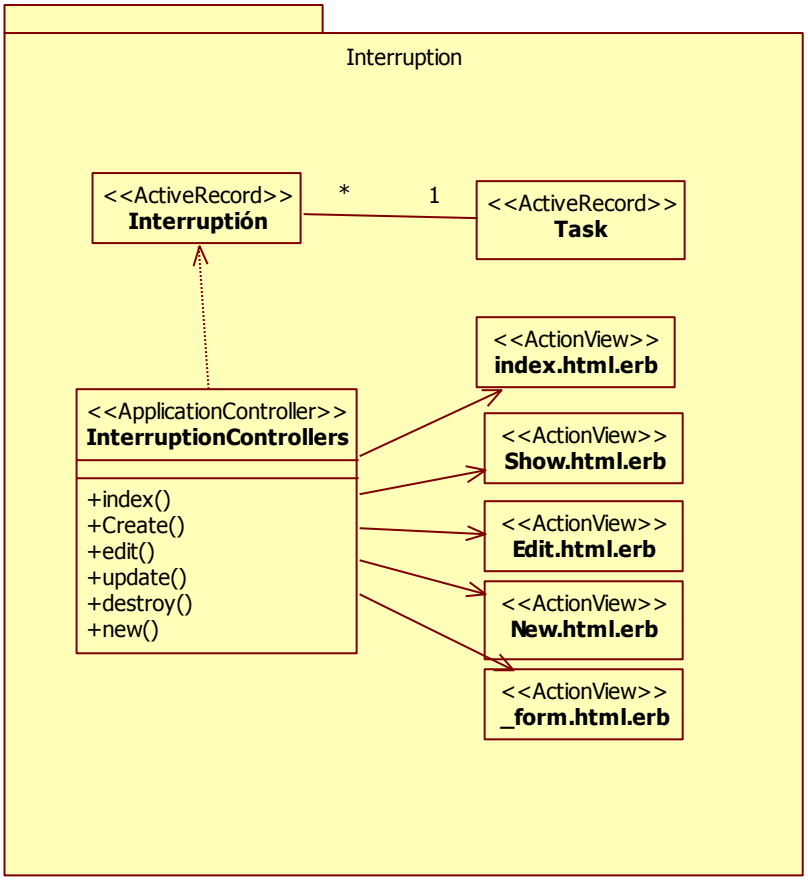


Diagrama de clase MVC-Interruption

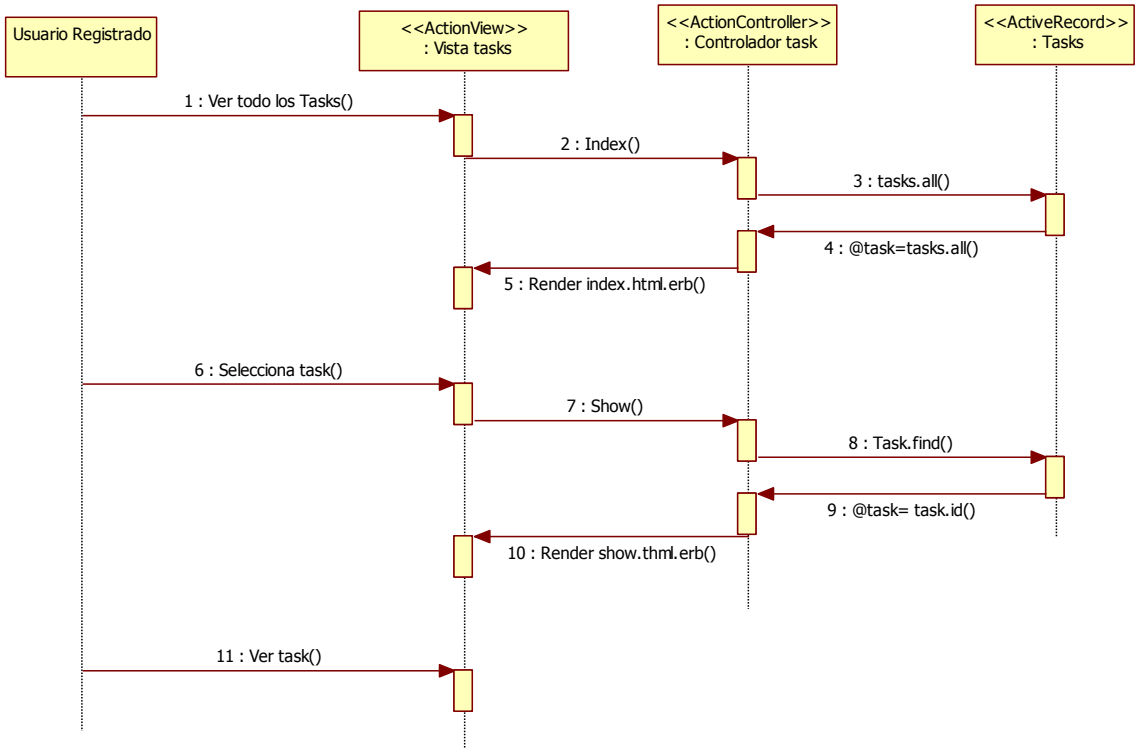
5.4.3 Diagramas de Secuencias

Un **diagrama de secuencia** muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. El diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario y mensajes intercambiados entre los objetos, es decir, capturan el comportamiento de los casos de uso.

La secuencia de acciones en un caso de uso comienza:

1. Cuando un actor invoca el caso de uso mediante el envío de algún tipo de mensaje al sistema.
2. Se obtiene el mensaje del actor con algún objeto de diseño.
3. Después el objeto de diseño llama a algún otro objeto, y de esa manera los objetos implicados interactúan para realizar y llevar a cabo un caso de uso. En el diseño, es preferible representar esto con diagramas de secuencia, ya que el centro de atención principal es el encontrar secuencias de interacciones detalladas y ordenadas en el tiempo.

A continuación observaremos un ejemplo de iteración para la gestión de tarea. Con el fin de que el lector se sitúe en cómo interactúan los objetos entre sí, en este ejemplo se muestra algunos diagramas de secuencia para la gestión de tarea. Queda mencionar, que no se incluyen todos los diagrama de secuencia que cubren los casos de uso porque al usar el patrón MVC los diagramas de interacción que se generan son muy parecidos.



La figura anterior muestra el diagrama de secuencia para ver tarea. El flujo de acción que se sigue es el siguiente:

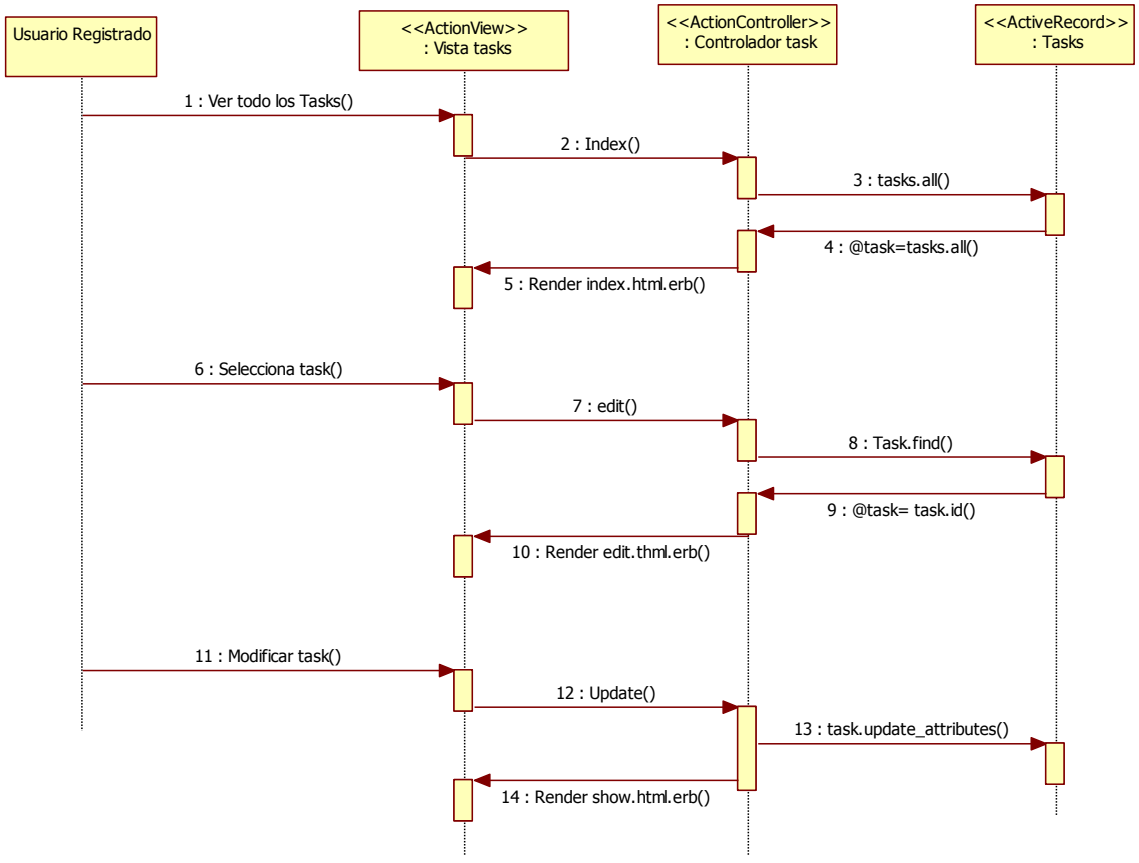
1. Usuario registrado hace una petición para ver la lista de tareas. El actor interactúa con la vista enviando el mensaje de que quiere que se le muestre un listado con todos las tareas.
2. La vista de tarea hace una petición (ejecutar index) al controlador de tarea. La vista envía un mensaje al controlador para que éste ejecute el método index correspondiente al encargado de mostrar el listado de todos los entrenamientos del sistema.
3. Controlador envía mensaje a modelo de tarea pidiéndole que le devuelva un array con todos los entrenamientos que hay almacenados en la base de datos.
4. El modelo devuelve en una variable todos las tareas que pertenecen al usuario.
5. Controlador renderiza la vista correspondiente a ver tareas (index.html.erb).
6. Mostrada la vista al usuario registrado, éste selecciona la opción de ver una tarea en particular.

7. Al pulsar sobre el elemento de interfaz, la vista envía el mensaje al controlador con la petición de que se ejecute la acción show. Usando el método POST, se envía el identificador del tarea que se quiere ver.
8. Capturado el identificador de la tarea que se desea ver, el controlador hace una búsqueda de este entrenamiento sobre el modelo.
9. El modelo devuelve en una variable la tarea correspondiente.
10. El controlador renderiza la vista para mostrar la tarea. Esta vista tiene acceso a las variables instanciadas en el controlador, por lo que dispone de los datos de la tarea que lo compone.

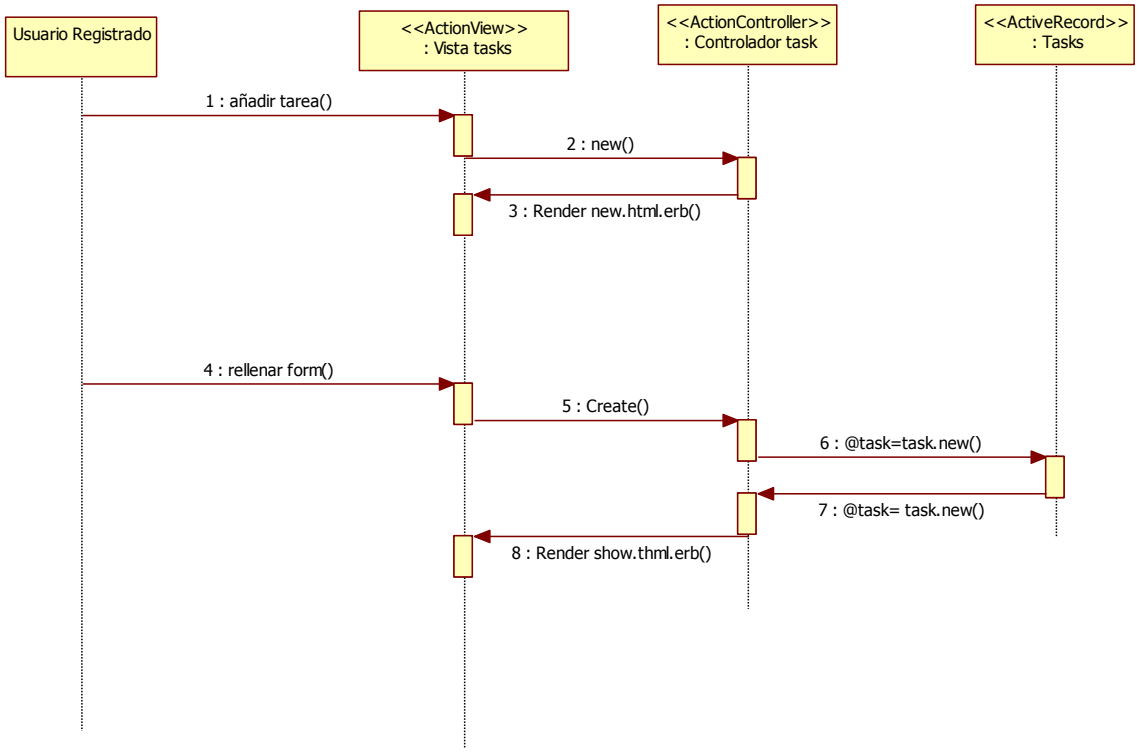
A continuación se muestra el diagrama de interacción para modificar tarea. El flujo de interacciones es muy similar al explicado anteriormente. Sin embargo, una vez que se ha obtenido la tarea que se desea editar y se ha renderizando la vista edit.html.erb empiezan las variaciones.

El usuario registrado modifica la tarea a través del formulario que aparece en la vista de editar. Sobre él se realizan las modificaciones necesarias, cambiando el contenido de cualquiera de los campos que lo conforman. Al pulsar enviar en la vista, ésta envía un mensaje de modificación "update" al controlador de tarea.

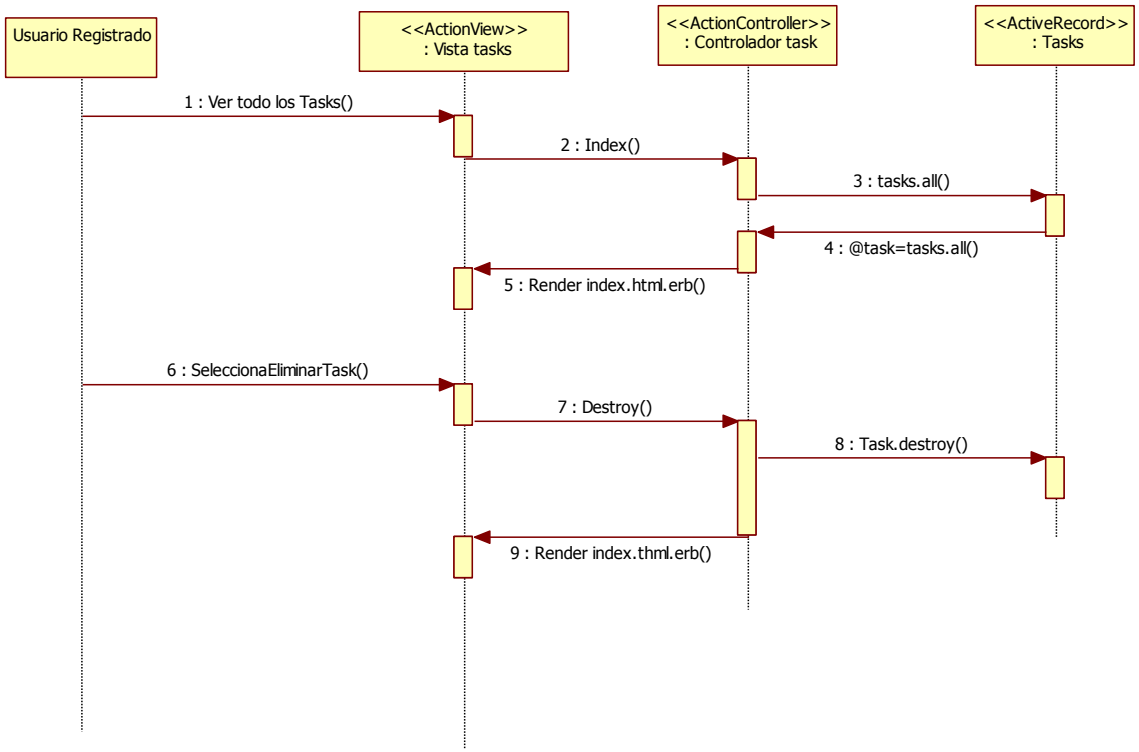
El controlador tarea envía el mensaje a los modelos de tarea . Realizadas las modificaciones, se renderiza la vista de ver tarea (show.html.erb) con el mensaje de que la tarea ha sido modificada.



Las próximas dos figuras muestran las interacciones para añadir y eliminar una tarea.



Añadir una tarea nueva



Eliminar una tarea

Tal y como se refleja en ellas, el flujo de interacciones entre objetos es muy similar a las mostradas anteriormente.

Ahora el lector puede comprender por qué se ha reducido esta sección únicamente a estos cuatro diagramas de secuencia. A parte hay que reseñar que los diagramas de secuencia son muy similares a los diagramas de colaboraciones (estos últimos en un nivel de abstracción más alto). Con las colaboraciones, estos diagramas de secuencia y los diagramas de clases, hay suficiente documentación para comprender el diseño del sistema.

5.4.4 Diagramas de Paquetes

El diagrama de paquetes que se muestra a continuación completa a los diagramas de clases de la sección anterior. Es uno de los últimos pasos para formalizar y entender la fase de diseño.

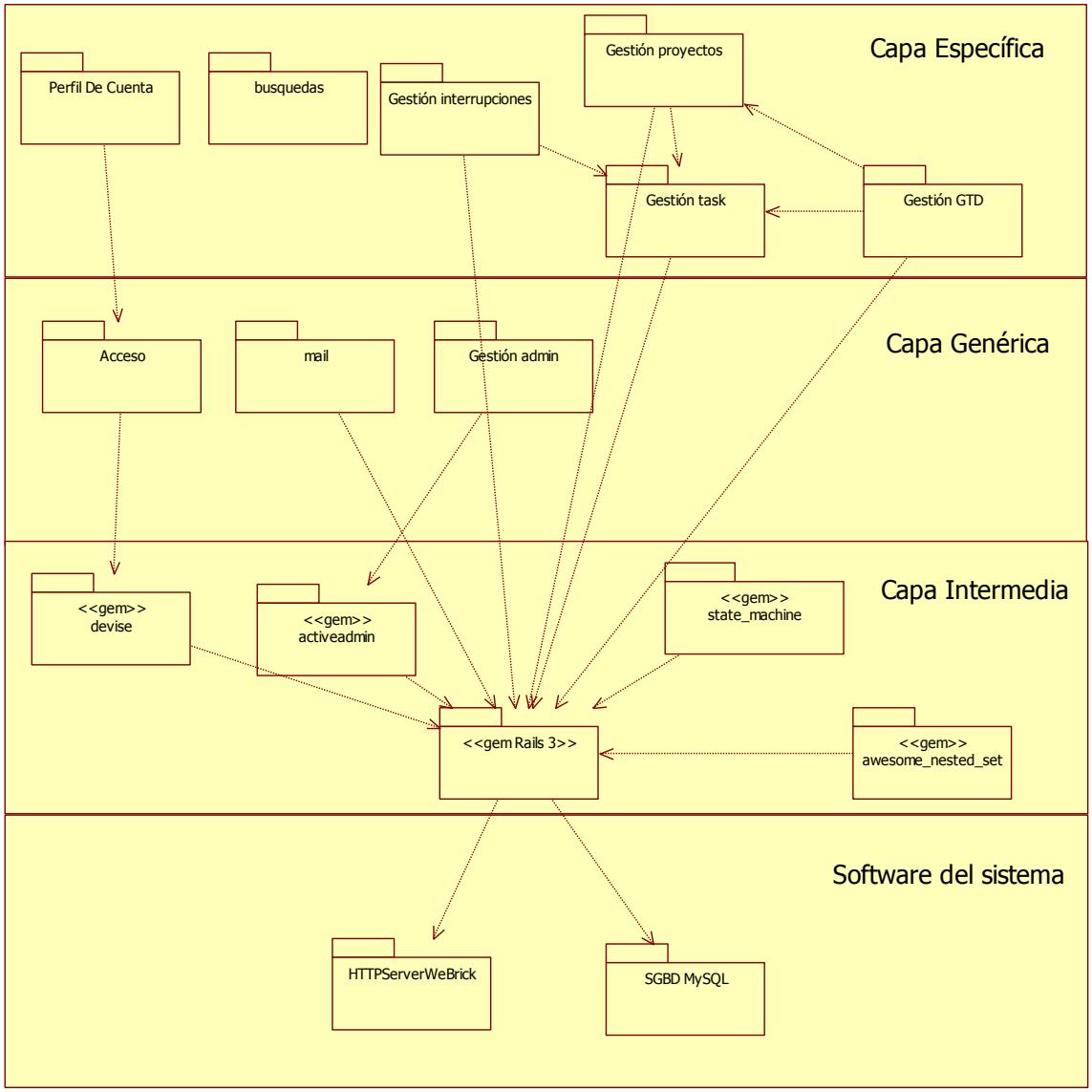


Diagrama de paquete con subsistemas capa intermedia y capa software del sistema

Previamente, en la fase de análisis se hace una primera aproximación de las capas necesarias de la aplicación, donde se detalló la composición de las capas genérica y específica. El siguiente paso es identificar los subsistemas intermedios y de software del sistema. El software del sistema y la capa intermedia constituyen los cimientos de un sistema, ya que toda la funcionalidad descansa sobre software como sistemas operativos, sistemas de gestión de bases de datos, software de comunicaciones, tecnologías de distribución de objetos, kits de diseño de IGU y tecnologías de gestión de transacciones.

Debido a que se usará el framework Rails, la capa intermedia se compondrá básicamente por las gemas (equivalente a componentes Ruby que se necesitarán). Tanto las capas específicas como genéricas dependerán de ellas para una integración total del sistema.

Para el soporte del middleware(capa intermedia) se incluye en la cuarta —software del sistema— tanto el servidor web (que en entornos de desarrollo será una versión de WEBrick preparada para ejecutar Ruby On Rails) como el sistema de gestión de bases de datos (MySQL , aunque podría cambiarse fácilmente por PostgreSQL o SQLite3).

5.4.5 Modelos de Bases de datos

Normalmente la mayoría de los sistemas de información que se desarrolla en la actualidad suelen estar apoyadas en una base de datos. El uso de bases de datos, a diferencia de otros sistemas de persistencia de datos, conlleva una serie de ventajas que se enumeran a continuación:

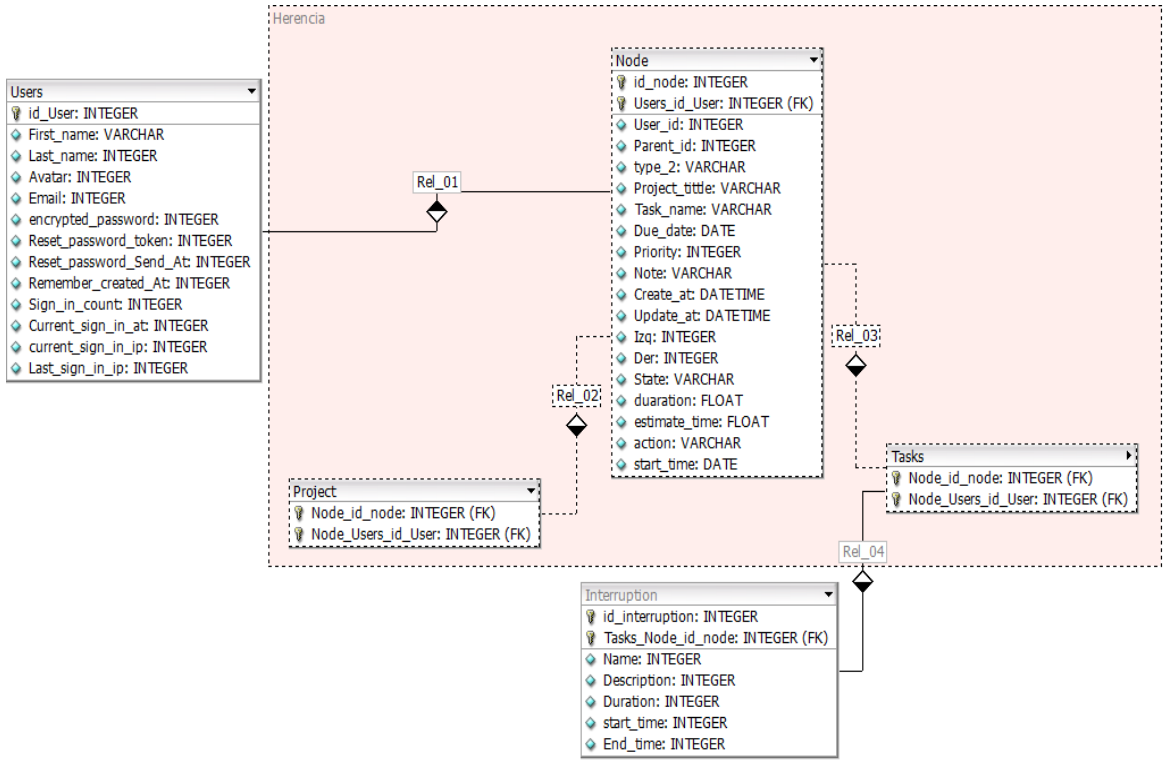
1. Independencia de los datos, los programas y procesos. Permite modificar los datos sin modificar el código de las aplicaciones.
2. Menor redundancia. No hace falta tanta repetición de datos. Sólo los buenos diseños de datos tienen poca redundancia.
3. Mayor seguridad en los datos. Al limitar el acceso a ciertos usuarios.
4. Datos más documentados. Gracias a los metadatos que permiten describir la información de la base de datos.
5. Acceso a los datos más eficiente. La organización de los datos produce un resultado más óptimo en rendimiento.
6. Menor espacio de almacenamiento. Gracias a una mejor estructuración de los datos.

Un **modelo de base de dato** es un tipo de modelo de dato que determina la estructura lógica de una base de dato y de manera fundamental determina el modo de almacenar, organizar y manipular los datos.

En este proyecto el diseño físico de la base de datos es gestionada por el framework Rails usando migraciones. Las migraciones son clases de Ruby que generan Código SQL para la implementación de las tablas. Esta acción provoca que el sistema sea casi independiente de la base de datos a excepción de la configuración inicial SGBD, en donde le dices al framework que SGBD se usará para que todas las migraciones sean traducidas al lenguaje soportado.

En esta sección por tanto, se mostrará el diseño físico de la base de datos del sistema. Para realizar este diseño físico se ha seguido la premisa fundamental de evitar la redundancia de datos, porque malgastan el espacio y aumentan la probabilidad de que se produzcan errores e incoherencias. También que la información almacenada en la base de datos sea correcta y completa para su manipulación.

A continuación se muestra un listado de cada una de las tablas que conforman el modelo de datos del sistema. Para ello se ha usando el diagrama E/R .Este diagrama E/R nos ayudará a representar el diseño físico de la base de datos, siendo reflejo de las tablas mencionadas en apartados anteriores.



5.4.6 Modelo de Despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo.

El modelo de despliegue se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución tiene una influencia principal en su diseño.

El Diagrama siguiente, es el diagrama de despliegue de la aplicación. El actor principal actúa en el nodo cliente a través del navegador. Desde él se conecta usando la red de internet mediante protocolo HTTP con el nodo servidor.

En el nodo servidor se encuentran los subsistemas definidos desde la fase de diseño. En el diagrama, por simplificación, se muestran cada uno de los componentes que representan el patrón MVC, obviando que cada uno de ellos incluye los paquetes de la capa específica

y genérica del sistema. A través del patrón ActiveRecord heredados en los modelos se actúa con la del SGBD.

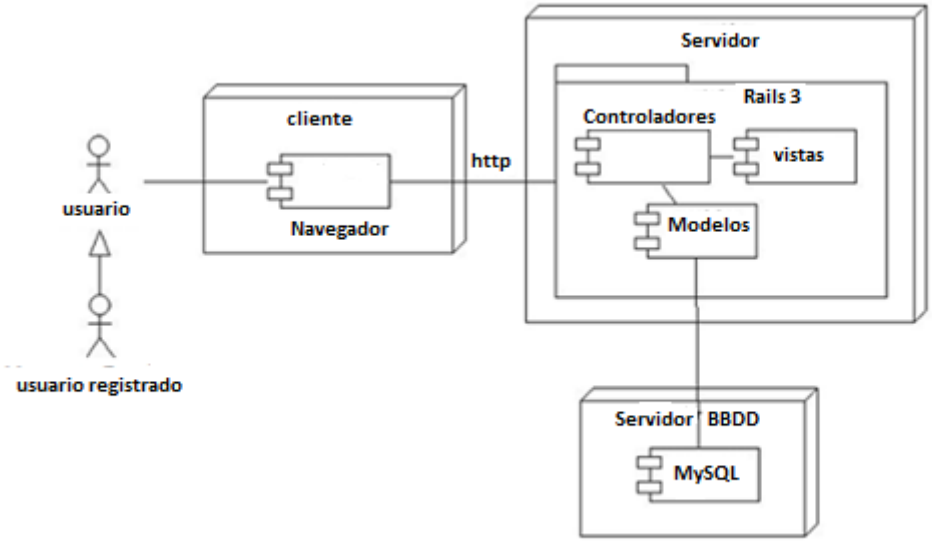
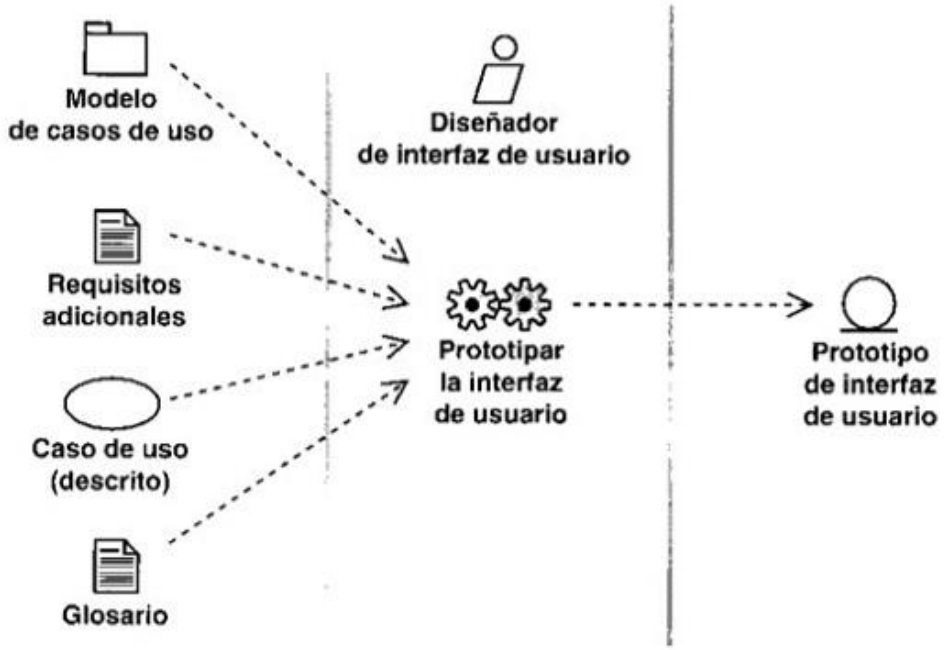


diagrama de despliegue

5.4.7 Prototipo de Interfaz de Usuario

Hasta ahora, se ha desarrollado un modelo de casos de uso que especifica qué usuarios hay y para qué necesitan utilizar el sistema. Esto se ha presentado en los diagramas de casos de uso, en las descripciones generales del modelo de casos de uso y en las descripciones detalladas para cada caso de uso.



En este capítulo intentaremos prototipar la interfaz de usuario. Las interfaces cobran relevancia en los sistemas software actuales, pudiendo incluso marcar el éxito de un producto. Para que un prototipo sea éxito tiene que contener una buena usabilidad.

¿Qué es usabilidad?

Usabilidad se define en el estándar ISO 9241 como el grado en el que un producto puede ser utilizado por usuarios específicos para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un determinado contexto de uso.

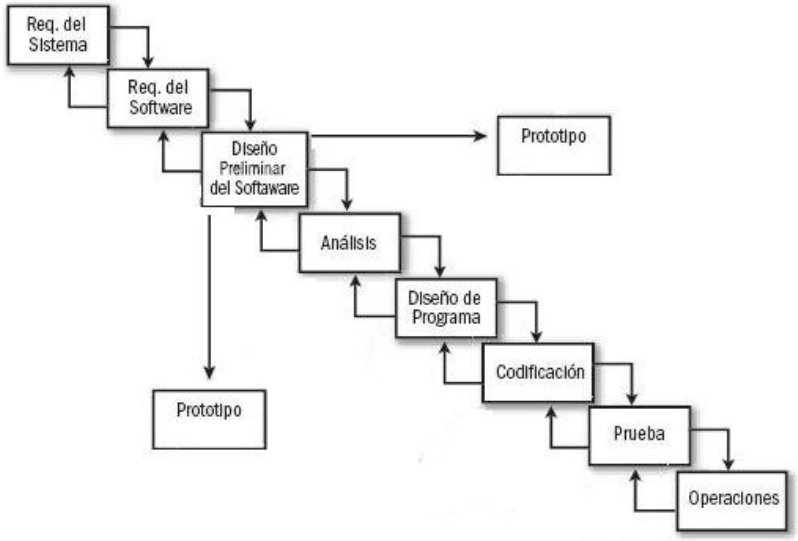
La usabilidad no puede definirse como un atributo simple de un sistema, pues implicará aspectos distintos dependiendo del tipo de sistema a construir. Para poder estudiar la usabilidad se descompone habitualmente en los siguientes cinco atributos básicos:

- **Facilidad de aprendizaje**. Cuán fácil es aprender la funcionalidad básica del sistema, como para ser capaz de realizar correctamente la tarea que desea realizar el usuario.
- **Eficiencia**. El número de transacciones por unidad de tiempo que el usuario puede realizar usando el sistema. Lo que se busca es la máxima velocidad de realización de tareas del usuario.

- **Recuerdo en el tiempo**. Este atributo refleja el recuerdo acerca de cómo funciona el sistema que mantiene el usuario, cuando vuelve a utilizarlo tras un periodo de no utilización.
- **Tasa de errores**. Este atributo contribuye de forma negativa a la usabilidad de un sistema. Se refiere al número de errores cometidos por el usuario mientras realiza una determinada tarea. Un buen nivel de usabilidad implica una tasa de errores baja.
- **Satisfacción**. Muestra la impresión subjetiva que el usuario obtiene del sistema. En muchos casos, existen aplicaciones que son tremendamente eficientes, pero muy difíciles de aprender a usar.

Por tanto, la usabilidad de un sistema está ligada principalmente a la interacción del mismo. Esta interacción no está definida en la interfaz gráfica, sino que va ligada con el código que implementa la funcionalidad del sistema. Por esta razón se quiere realizar el prototipo de la interfaz de usuario.

La siguiente imagen nos muestra en que paso del proyecto vamos y donde encaja el prototipo de la interfaz de usuario:



Para la aplicación que nos atañe se ha elegido realiza un boceto que se convertirá en código para una posterior evaluación en la fases de diseño e implementación. A medida que van surgiendo las iteraciones, la interfaz va cambiando hacia una evolución constante. La construcción de la interfaz permitirá al usuario llevar a cabo los casos de uso de manera eficiente. El resultado final de esta actividad será un conjunto de esquemas de interfaces de usuario y prototipos de interfaces de usuario que especifican la apariencia de esas interfaces para cada uno de los actores más importantes.

Para la construcción del prototipo se han seguido los siguientes pasos:

1. **Diseño lógico de la interfaz de usuario**. Comenzar con los casos de uso e intentar discernir qué se necesita de las interfaces de usuario para habilitar los casos de uso para cada actor.

   Se realizo siguiendo el siguiente estudio:

- Información que necesita el usuario para realizar la tarea
- Terminología y símbolos del dominio del problema
- Descripción de cómo se realizan actualmente esas tareas

2. **Diseño físico de la interfaz de usuario**. Crear el diseño físico de la interfaz de usuario y desarrollar prototipos que ilustren cómo pueden utilizar el sistema los usuarios para ejecutar los casos de uso. Mediante la especificación de qué se necesita antes de decidir cómo realizar la interfaz de usuario, llegamos a comprender las necesidades antes de intentar realizarlas.
3. **Evaluación**. Medir el alcance de la aplicación y los niveles de eficiencia de los prototipos junto al cliente.

Por lo tanto, partiendo de los puntos anteriores intentaremos construir prototipos ejecutables de las configuraciones más importantes de elementos de interfaz de usuario. El uso de estas técnicas puede prevenir muchos errores que serán más caros de corregir en fases posteriores. A la hora de la revisión de prototipos para validarlos es conveniente verificar lo siguiente:

- Permite que el actor navegue de forma adecuada.
- Proporciona una apariencia agradable y una forma consistente de trabajo con la interfaz de usuario.
- Cumple con estándares relevantes como el color, tamaño de los botones y situación de las barras de herramientas.

El prototipo de interfaz de usuario que desarrollamos en las secciones siguientes deberá servir como especificación de la interfaz de usuario.

5.4.7.1 Prototipo para Interfaz Pública

Como se ha mencionado, en las siguientes secciones se realizará la interfaz de usuario que servirá como prototipo para la aplicación. La primera de las interfaces que se plantea es la interfaz pública. las pantallas que componen esta interfaz son informativas y muestra las siguientes áreas:

- **Página de inicio**. Primera página que se muestra al acceder a la aplicación web.
- **Página de tour**. Hace referencia al apartado que da a conocer todas las funcionalidades propuestas en el sistema.
- **Página de Contacto** . Ofrece la posibilidad al usuario de ponerse en contacto con otro los actor del sistema: el administrador del sistema.
- **Página de Blog.** El usuario podrá informarse con las funcionalidades o la nuevas noticias de la aplicación, esta sección es meramente informativa, es controlada por el administrador del sistema.
- **Página de FAQ.** Otro apartado meramente formativo en dónde simplemente se tendrá las respuestas a las preguntas más frecuente de los usuarios.

La primera de las pantallas que se muestra es la de la página de inicio. Está compuesta de tres zonas que formarán la base para el resto de interfaces de esta sección:

- **Encabezado**. Bloque formado por el logo de la aplicación junto a dos menús. El menú principal es el que contiene los apartados accesibles de la interfaz (inicio, tour y contacto, acceso y registro del sistema).
- **Cuerpo**. Bloque formado por el contenido de la sección que se muestra al usuario. Los elementos que aparecen en esta zona son los que se explicarán con más detalle.
- **Pie de página**. Únicamente se compone de los enlaces a datos de carácter informativo de las demás páginas de la aplicación.

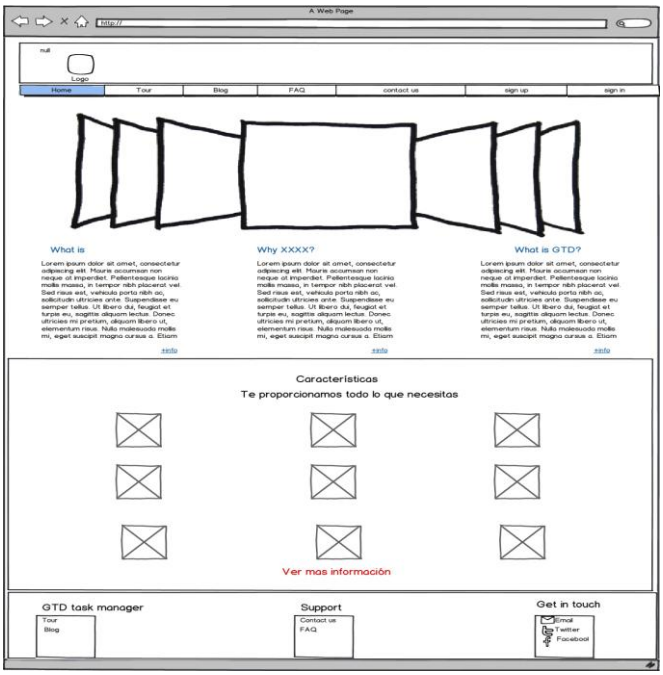
Página de Inicio

En el encabezado siempre habrá lo que ya mencionamos justamente en el párrafo anterior.

El cuerpo de la página de inicio tendrá un slideshow mostrando noticias de última hora. También en esta sección nos encontraremos con 3 columnas de información sobre la aplicación en sí, que es GTD, o porque usar esta aplicación, etc.

Finalmente el final de esta sección mostraremos información sobre las características más relevantes del sistema de esta forma obtendremos la atención del usuario.

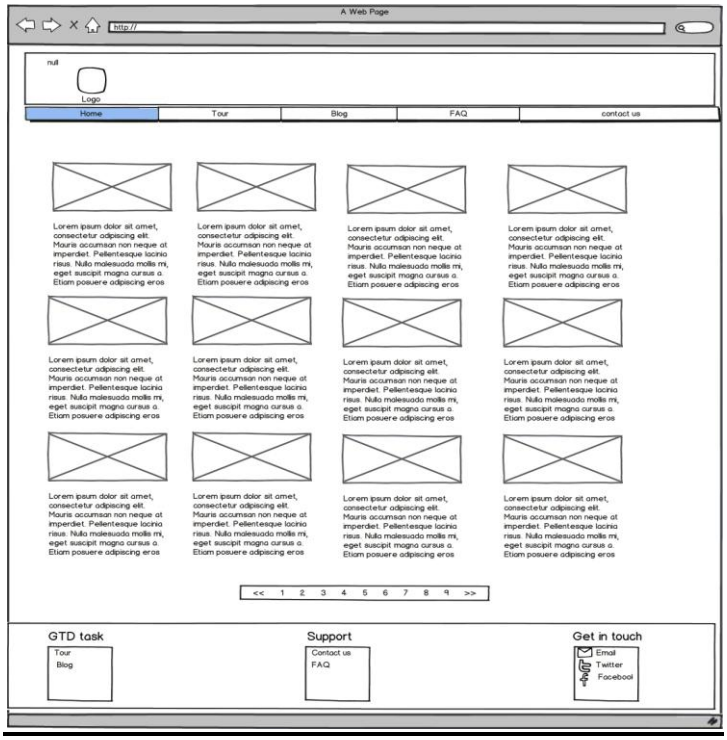
En cuanto, al pie de la página de inicio como en todas las demás se mostrará lo que habíamos dicho anteriormente, enlaces sobre diferentes sitios de la aplicación, y links hacia diferentes redes sociales, etc.



Página de tour

En el encabezado y el cuerpo siempre será el mismo en todas estas pantallas de la interfaz pública.

El cuerpo de la página contendrá imágenes con su respectivos textos explicando cada uno de las características del software.

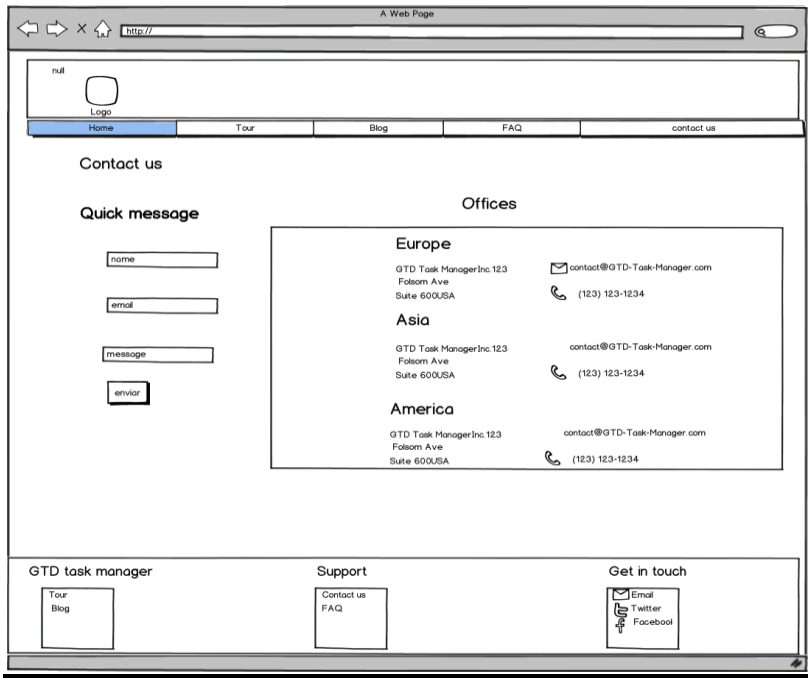


Página de Contacto

En el encabezado y el cuerpo siempre será el mismo en todas estas pantallas de la interfaz pública.

El cuerpo de la página contendrá dos secciones diferenciadas la primera un formulario para que los usuarios se pongan en contacto con los administradores.

La segunda de la secciones es meramente informativa en dónde se expondrán teléfonos correos y direcciones de las diferentes secciones de la empresa distribuida por todo el mundo.

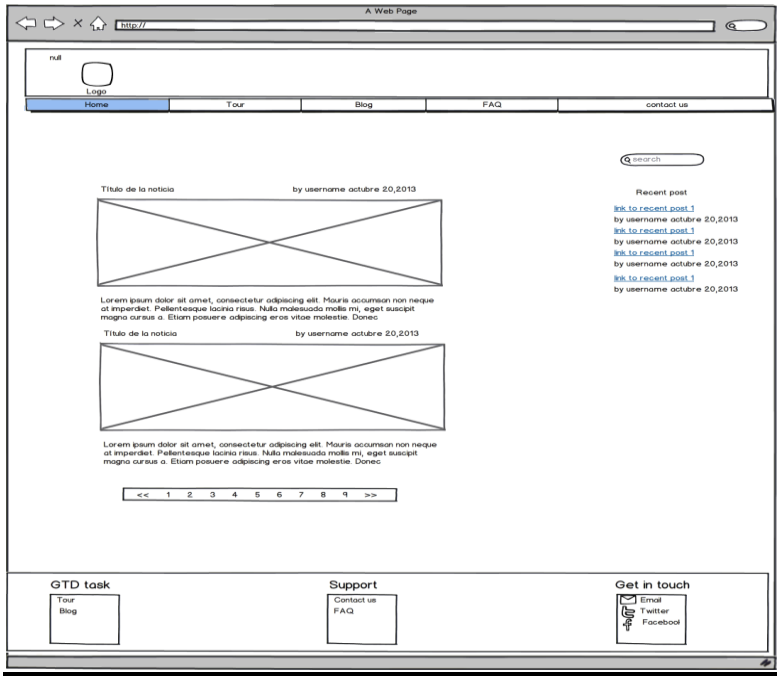


Página de Blog

En el encabezado y el cuerpo idem de los apartados anteriores.

El cuerpo de la página de blog contendrá un simple blog con noticias, imagenes y cuerpo de la noticias. Este blog se encargará el administrador del software.

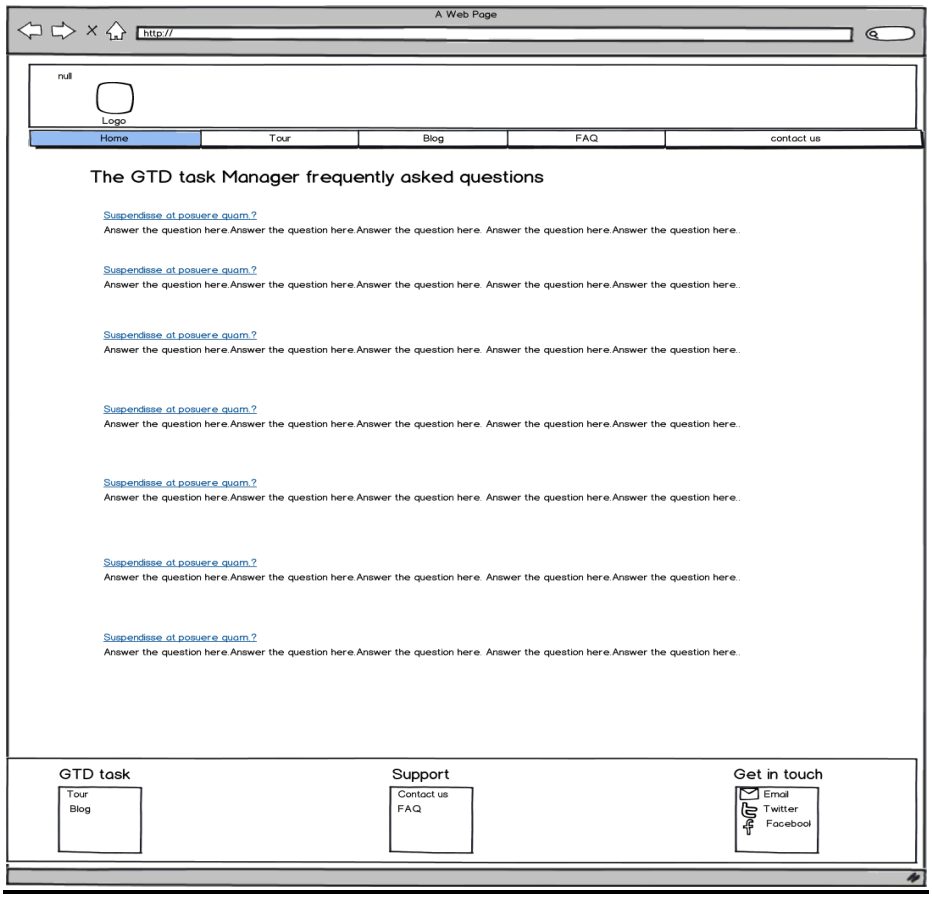
También en esta página contendrá un buscador y un listado con los links de las noticias más recientes.



Página de FAQ

En el encabezado y el cuerpo idem de los apartados anteriores.

El cuerpo de la página contendrá preguntas y respuestas más preguntadas por los usuarios que utilizan la aplicación.



The GTD task Manager frequently asked questions

Suspendisse at posuere quam.?
Answer the question here.Answer the question here.Answer the question here. Answer the question here.Answer the question here..

Suspendisse at posuere quam.?
Answer the question here.Answer the question here.Answer the question here. Answer the question here.Answer the question here..

Suspendisse at posuere quam.?
Answer the question here.Answer the question here.Answer the question here. Answer the question here.Answer the question here..

Suspendisse at posuere quam.?
Answer the question here.Answer the question here.Answer the question here. Answer the question here.Answer the question here..

Suspendisse at posuere quam.?
Answer the question here.Answer the question here.Answer the question here. Answer the question here.Answer the question here..

Suspendisse at posuere quam.?
Answer the question here.Answer the question here.Answer the question here. Answer the question here.Answer the question here..

Suspendisse at posuere quam.?
Answer the question here.Answer the question here.Answer the question here. Answer the question here.Answer the question here..

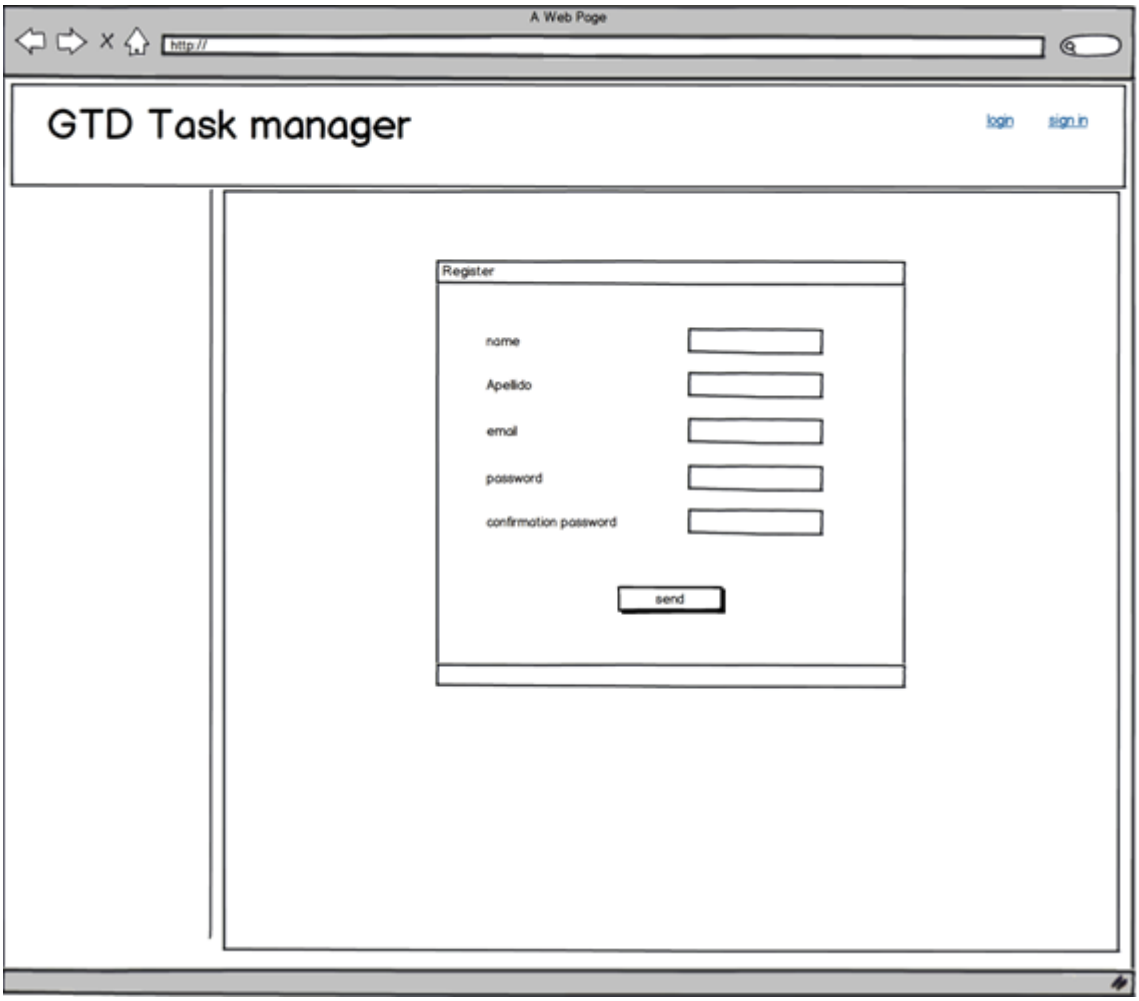
5.4.7.2 Prototipo para el Registro y el Acceso

Las interfaces para el registro y el acceso de un usuario al sistema son muy similares. Su estructura es algo diferente al resto de páginas de la aplicación, eliminando elementos que puedan interferir en el objetivo último —registrarse o acceder—.

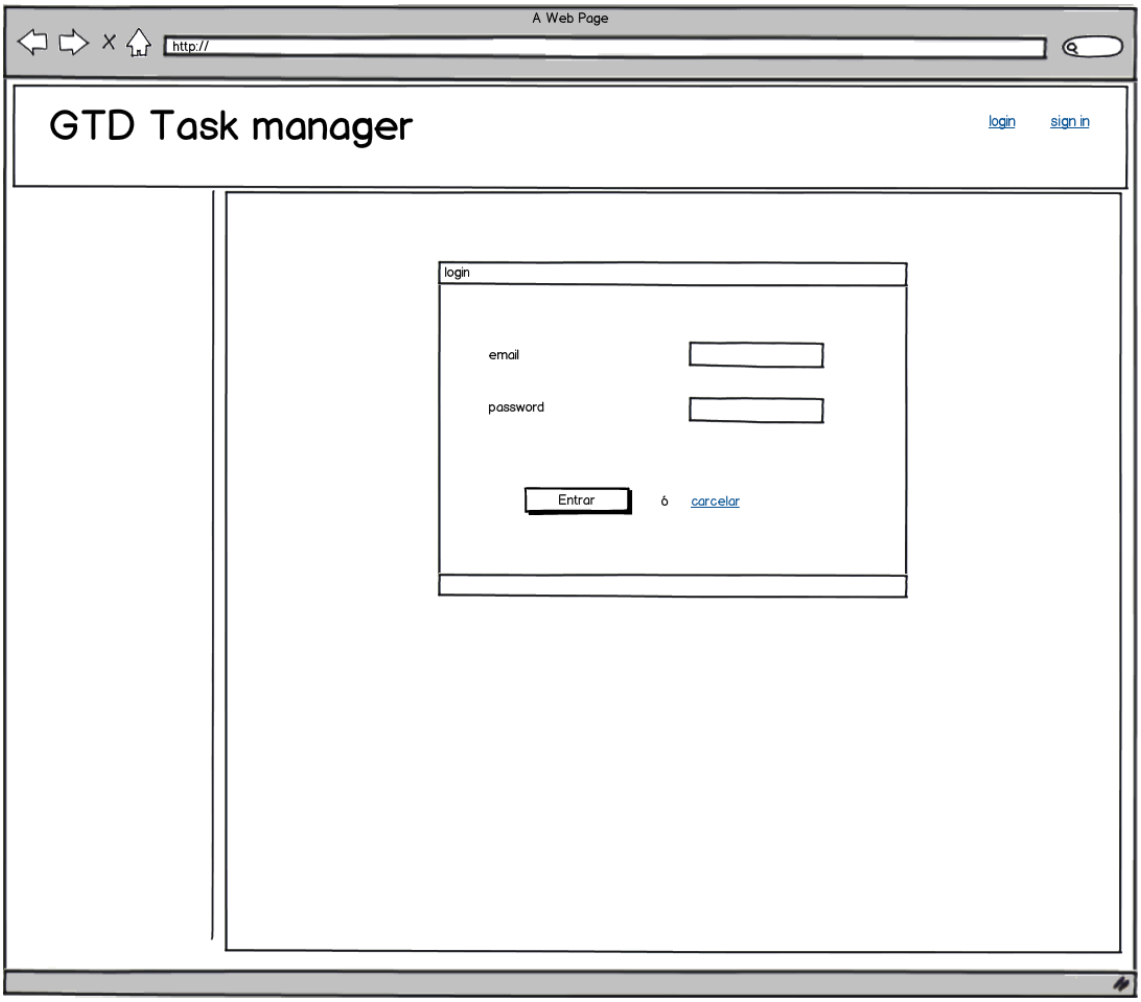
Tanto el registro como el acceso aparece en ventanas emergentes. Si se desea cancelar el proceso de registro o acceso, siempre se podrá volver a la interfaz pública principal haciendo clic en el aspa de cierre.

Para el registro se ha estudiado la complejidad del formulario, llegando a la conclusión de que para el usuario final se debe minimizar el número de campos necesarios a rellenar. Quedaría de la siguiente forma: nombre, apellidos, email, contraseña confirmar contraseña.

Por otro lado, en la interfaz para el acceso al sistema se propone el uso del correo electrónico para poder realizar la identificación correctamente.



Interfaz para registrarse.



Interfaz para acceder a la aplicación

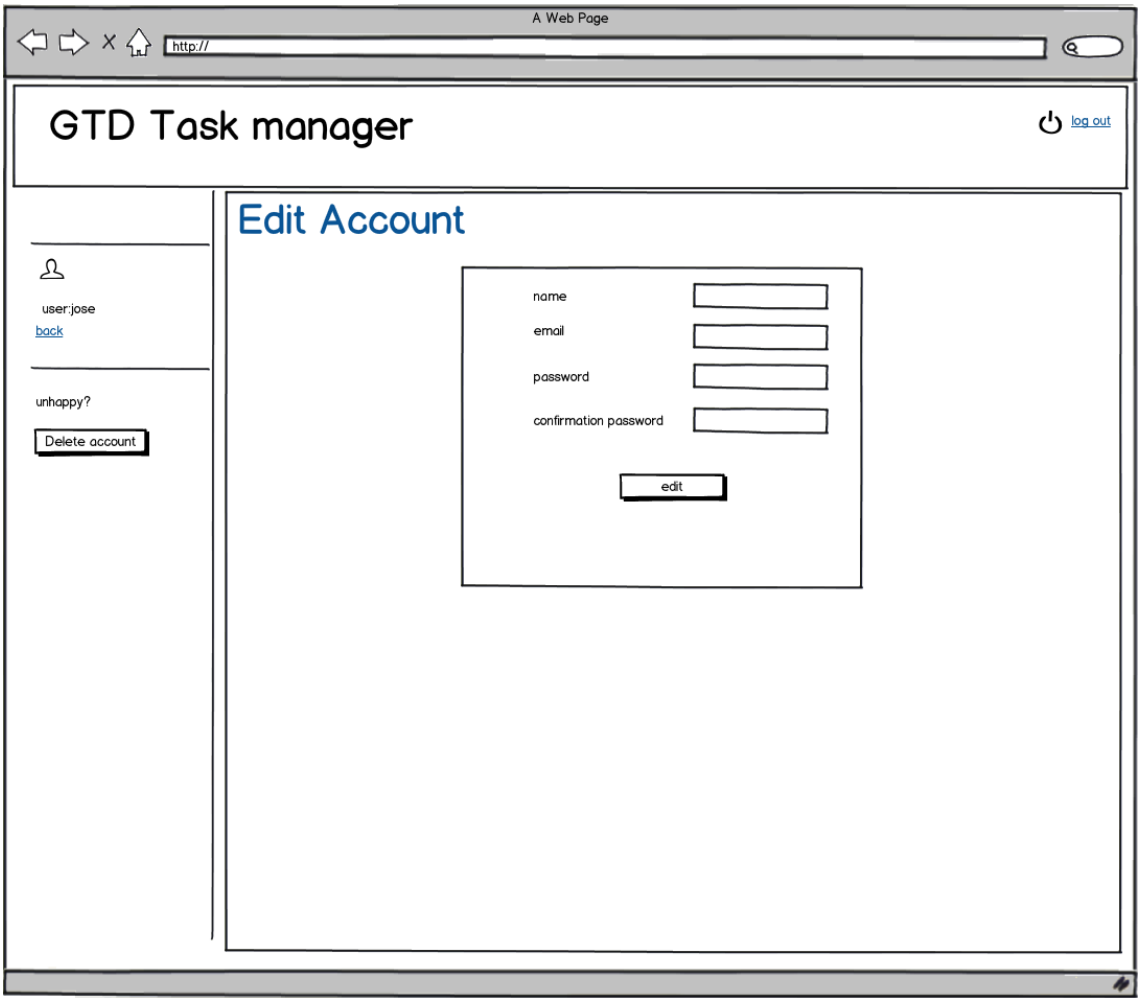
5.4.7.3 Prototipo para Interfaz Work-Desk

Esta es la interfaz principal de la aplicación donde estará la mayoría de las funcionalidades de la aplicación.

Como podemos observar habrá un menú en la parte de arriba de la aplicación en donde se observará tres cosas el título de la aplicación ,dos botones que nos permitirá crear proyecto y tareas. Se ha decidido ponerlo aquí pues son las dos funcionalidades principales de la aplicación y tiene que está en un sitio donde llame bastante la atención. Y finalmente la funcionalidad para cerrar sesión de la aplicación.

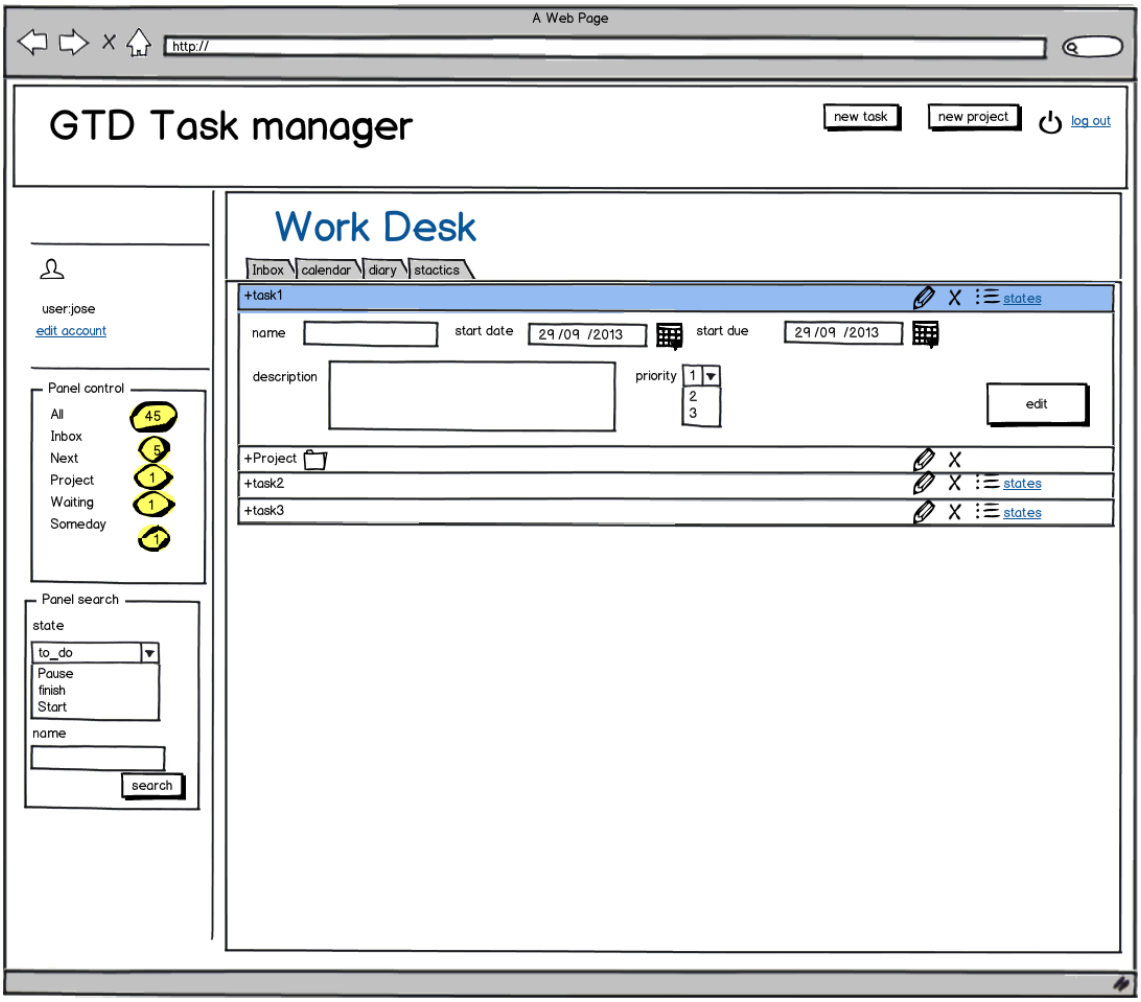
También en la parte derecha de la interfaz habrá una columna donde se podrá distinguir 3 partes diferenciadas:

- La primera es una parte de información del usuario donde podrá configurar su cuenta de acceso. En esta sección no sólo podrás editar los campos puesto en el momento de registro sino que también se le permitirá la funcionalidad de poder borrar la cuenta asociada al sistema.

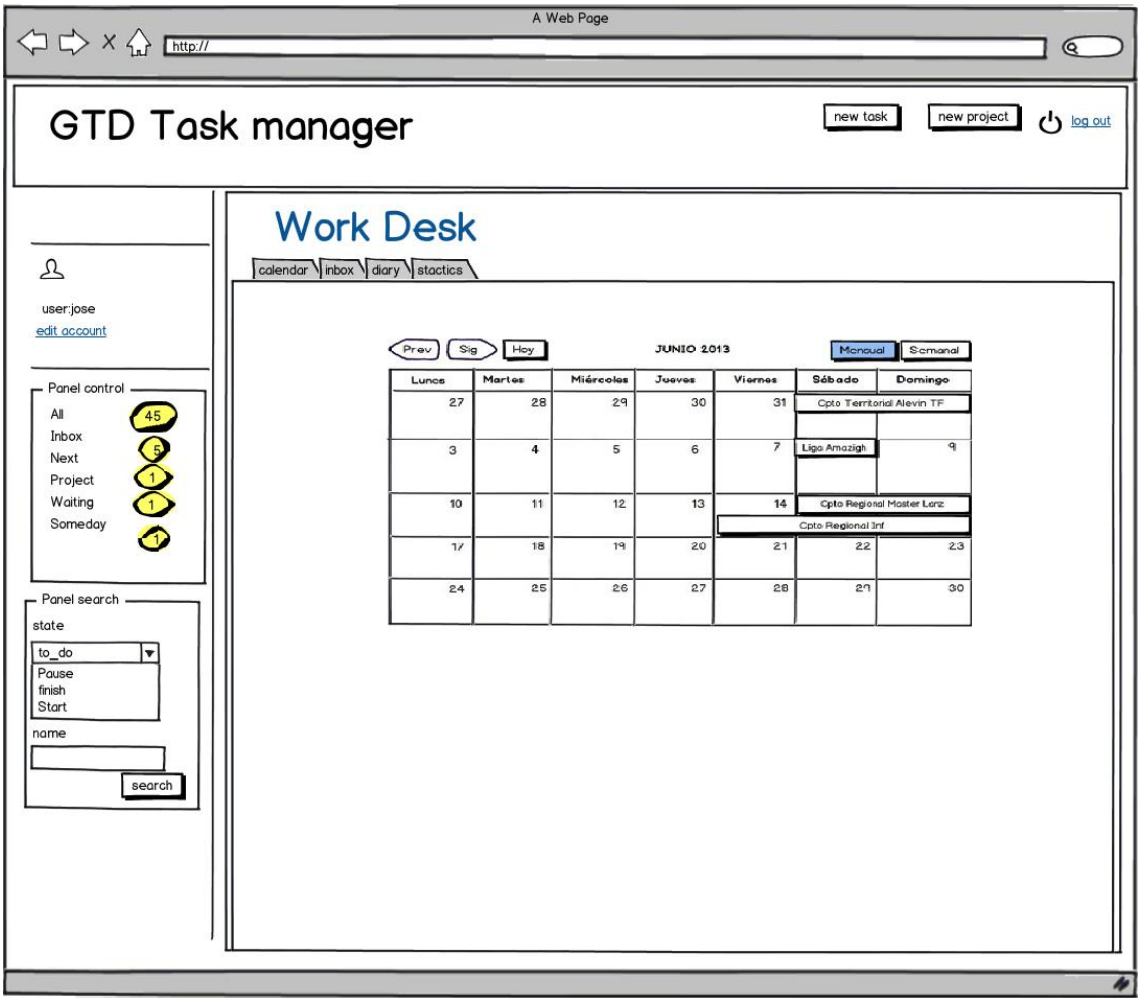


- La segunda se trata del panel de control de los estados de GTD.
- la tercera de las parte es un panel de búsqueda en donde podrá buscar por el nombre de la tarea o por el estado en que se encuentra cada tarea, tarea empezada, pausada, finalizada, etc ....

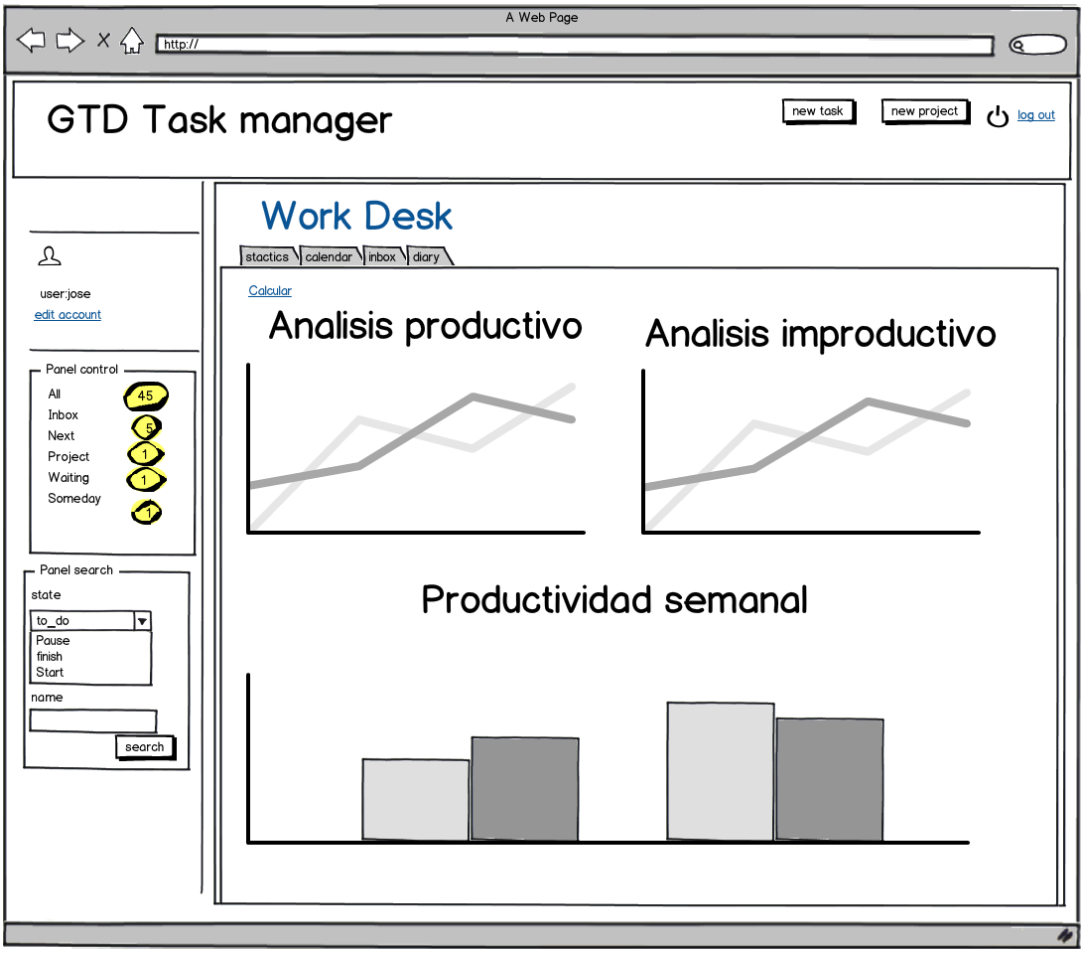
En la columna derecha de la aplicación es donde se verá los resultado de lo que vayas haciendo en esta sección se encontrará un "drag and drop" con las tareas y los proyectos. Cada tarea tendrá una series de funcionalidades tales como editar la información de la tarea eliminarla cambiar el estado de gtd(inbox, next, proyect....) ,empezar tarea, pausar , terminar una tarea. Todos estas acciones se controlará con cronómetro de tiempos internos que contabilice el tiempo productivo e improductivo.



A continuación también se mostrará que en esta sección podemos encontrar un calendario en donde podremos ver las tareas de un forma clara a lo largo de los días.


A Web Page
http://
GTD Task manager
new task
new project
log out
Work Desk
calendar
inbox
diary
stactics
user:jose
edit account
Panel control
All 45
Inbox 5
Next 1
Project 1
Waiting 1
Someday 1
Panel search
state
to_do
Pause
finish
Start
name
search
Prev
Sig
Hoy
JUNIO 2013
Mensual
Semanal
Lunes
Martes
Miércoles
Jueves
Viernes
Sábado
Domingo
Cpto Territorial Alevin TF
Liga Amazigh
Cpto Regional Master Lanz
Cpto Regional Inf

Otra de la funcionalidades que tendremos, será ver los tiempo productivos e improductivos de una forma más gráficas .



5.5 Implementación

En este apartado no centraremos en la implementación del producto. La implementación es la actividad en la que se construye la aplicación utilizando un lenguaje de programación concreto siguiendo las directrices marcadas por los documentos del diseño.

La implementación se encarga de concretar el diseño teniendo en cuenta todos estos factores, para ello hay que definir los propósitos que se deben alcanzar en esta fase:

- **Planificación de los requisitos que se desarrollarán en cada iteración**. Es un proceso incremental, lo que da lugar a que se implemente en una sucesión de pasos pequeños y manejables.
- **Diagrama de de despliegue**. Se situará el sistema en fases de despliegues pudiendo lograr una correcta puesta en marcha si se lleva a fases de producción.
- **Implementar las clases y subsistemas encontrados durante el diseño**. En particular, las clases se implementan como componentes de fichero que contienen código fuente.
- En este capítulo acabará presentándose cómo se lleva a cabo la implementación de la aplicación.

Para cumplir con éstos propósitos se ha seguido un proceso compuesto por varias fases:

1. **Instalación y configuración del entorno de trabajo**. Son varias las herramientas que intervienen en la creación del software, por lo que es importante dejar la base del sistema preparado para poder soportar los framework de desarrollo. También en este punto se detalla las pautas que hay que seguir para poder dejar el sistema listo para comenzar a desarrollar cada componente en Ruby On Rails.
2. **Desarrollo de los componentes pertenecientes a cada requisito**. Este documento no reflejará el código de cada componente llevado a cabo, pero sí que mostrará los detalles de la arquitectura planteada y de los elementos básicos para comprender el código implementado.

El objetivo no es acabar mostrando cada uno de los ficheros que contienen las clases, sino mostrar el código base que se necesita conocer para comprender el entorno de trabajo y los componentes que conforman el software.

5.5.1 Instalación y Configuración del Framework

5.5.1.1 Instalación de Rvm (Ruby Version Manager)

En esta sección instalaremos RVM (Ruby Version Manager) que permite instalar y administrar diferentes instancias de Ruby, esto se hace para evitar el inconveniente de trabajar con versiones diferentes a las que se maneja en entorno de producción. Por ejemplo, podrías querer instalar la versión 1.8.7 y la 1.9.2 y decidir cuándo usar cual.

Para instalar RVM , se ha tenido que insertar en la consola las siguientes instrucciones:

```
# Necesitamos tener instalado curl para descargar fichero
$ sudo apt-get install curl
# instalamos RVM
\curl -L https://get.rvm.io | bash -s stable
```

```
# Ahora simplemente tenemos que cargar RVM en nuestro nuevo shells. Para ello modificamos el fichero ~/.bash_profile, añadiendo al final del fichero la siguiente línea:
[[ -s "$HOME/.rvm/scripts/rvm" ]] && . "$HOME/.rvm/scripts/rvm" >>~/.bash_profile
# Para comprobar que todo está correcto, abrimos una nuevo shell y ejecutamos:
$ type rvm | head -1
# Deberíamos ver: rvm is a function, entonces quedo instalado
```

Desde aquí el sistema estará preparado para poder instalar cualquier versión de Ruby o de las distintas gemas que se usen en las posteriores fases.

5.5.1.2 Instalación de RoR (Ruby on Rails)

Ruby On Rails es una gema en sí, por lo que primero hay que instalar una versión de Ruby sobre la que usar la gema de rails en la versión deseada. El script necesario para realizar la tarea es el siguiente:

```
# Instalando la versión de Ruby 1.9.3
rvm install ruby 1.9.3

#usamos la versión que hemos intalado

rvm use 1.9.3

#Para comprobar, escribimos en una Terminal:
ruby -v

Y nos debería devolver algo como esto:
ruby 1.9.3p392 (2013-06-22 revision 39386) [i686-linux]
```

A continuación instalamos Ruby and Rails siguiendo los pasos siguientes:

```
# Instalando la versión de Rails 3.0.9
gem install rails --version=3.0.9

Cuando termina, verificamos que tengamos la versión que queremos:
rails -v

# nos devolvería algo así Rails 3.0.9
```

El entorno de desarrollo está listo para la creación del esqueleto de la aplicación. Rails dispone de un comando que ayuda a ello.

```
Rails new prototype # creamos una nueva aplicación en Rails
cd prototype # Accedemos a la carpeta de nuevo proyecto
rm public/index.html # borramos  la página index de bienvenida
bundle install # Se instalan o se actualizan las gemas que incluye el framework por defecto.
```

A continuación se muestra un script donde se ejecuta una instancia de un servidor incluido en Rails, pudiendo así ejecutar la aplicación a través del navegador (accesible a través de http://localhost:3000).

```
Rails server #Ejecuta el servidor de rails
Rails console #Muestra la consola de rails
```

5.5.1.3 Estructura de los Ficheros en RoR (Ruby on Rails)

El comando "rails new prototype" usado anteriormente crea una estructura de ficheros automáticamente con el esqueleto de la aplicación. La carpeta que contendrá el proyecto toma el nombre pasado como parámetro al comando, en este caso (**prototype**).

A continuación se mostrará la estructura de los ficheros más importantes generados en este proyecto:

Fichero/Carpeta	Propósito
App/	En este directorio se organizan los componentes de la aplicación. Tiene subdirectorios que contiene los controladores, vistas, modelos y helpers.
App/controller/	Aquí se encuentra las clase de los controladores. Los controladores maneja los requerimientos web del usuario.
App/helpers/	Aquí se encuentra todas las clases helpers que ayuda a las clases: modelos, vista, controlador. Los helpers ayuda a mantener el código lo más reducido y limpio posible.
App/models/	Contiene los datos que modelan y empaqueta los datos almacenados en la base de datos de la aplicación.
App/views/	Contiene las plantillas que pertenece a la capa de vista de la aplicación el formato del fichero es html.erb o haml.erb
Config/	Contienen los ficheros de configuración que necesitarán en la aplicación incluyendo la configuración de la base de datos(database.yml), la estructura del entorno rails(environment.rb) y el direccionamiento de las peticiones de recursos entrantes(routes.rb)
Db/	Normalmente, las aplicaciones rails tendrán un modelo de objetos que acceden a tablas de una base de datos relacionar. En ella se encuentra un fichero para ver la estructura de la tabla que se creará en el SGBD seleccionado (schema.rb), y una carpeta que contiene migraciones de la base de datos.
Doc/	Es en donde se almacena la documentación de la aplicación, se genera usando rake doc:app.
Gemfile	Fichero que permite especificar las gemas que se utilizaran en la aplicación.
Lib/	Aquí se almacena biblioteca o módulos específicos para la aplicación. Básicamente cualquier tipo de código específico que no pertenece a los controladores modelo o helpers
Log/	Aquí van los log de errores, se puede encontrar logs de errores para el servidor(server.log) y también para cada ambiente de rails develoment.log, test.log, production.log
Public/	Igual que el directorio público de un servidor web, ésta carpeta contiene todos los archivos que no cambian como html,js,css...
Readme	Fichero a editar para insertar el resumen o manual de la aplicación

Script/	Este directorio contiene los script para lanzar y administrar varias de las herramientas que se usan con rails como por ejemplo para generar código (generate), para lanzar el servidor web (server),etc.
Test/	Las pruebas que se escriban y las que cree rails van en éste directorio.
Temp/	Directorio para almacenar archivos temporales para procesamiento inmediato.

5.5.1.4 Configuración Inicial de RoR (Ruby on Rails)

Cuando Rails genera la estructura de ficheros, deja el sistema listo para ejecutarse, usando parámetros por defecto para su configuración. Aún así podemos si lo deseamos cambiar el SGBD a utilizar.

El tipo de SGBD puede ser especificado en un archivo de configuración (config/database.yml). Si editamos este fichero, se encontrará que por defecto se usa SQLite3 en vez de MySQL. La razón por la cual viene preparado para SQLite3 es porque es un SGBD muy ligero, aunque para entornos de producción no es muy recomendable ya que puede desbordarse si hay muchas peticiones simultáneas.

Se ha mencionado el término de entornos, por lo que a continuación se detallan los propósitos de cada uno:

- **Entorno de desarrollo (development).** Especificado como el entorno que usa el desarrollador en su máquina local.
- **Entorno de pruebas (test).** Utilizado para ejecutar los test que se generen.
- **Entorno de producción (production).** Cuando la aplicación está lista para poner en marcha y llevarla a un entorno de despliegue, es aquí donde se ejecuta.

La configuración por defecto para el entorno de desarrollo (que es el que se utilizará durante toda la implementación) es el siguiente (fichero config/database.yml):

```
development:
  adapter: sqlite3
  user: root
  password:
  database: prototype_development.sqlite3
  pool: 5
  timeout: 5000
```

Aunque podía ser válido y el proceso podría haber arrancado usando este SGBD, se ha cambia a MYSQL por ser más robusto. Este proceso habría que hacerlo con los tres entornos predeterminados. Quedaría de la siguiente forma:

```
development:
  adapter: mysql2
  user: root
  password:
  database: prototype_development
  pool: 5
```

```
  timeout: 5000

test:
  adapter: mysql2
  user: root
  password:
  database: prototype_test
  pool: 5
  timeout: 5000

production:
  adapter: mysql2
  user: root
  password:
  database: prototype_production
  pool: 5
  timeout: 5000
```

Ahora que se ha configurado la base de datos, es momento para que Rails cree una base de datos vacía para poder empezar a trabajar. Para ello hay que ejecutar el siguiente comando:

```
#crear base de datos
rake db:migrate
# repuesta de la orden en caso ok
migrations successfully created
```

5.5.1.5 Gemas Utilizadas en RoR (Ruby on Rails)

La manera más frecuente de reutilizar código en Rails es con la inclusión en el proyecto de gemas. Recordemos que las gemas son plugins y/o códigos añadidos que permiten el uso de nuevas funcionalidades y herramientas.

El archivo Gemfile es quien contiene la especificación de cada gema usada. A continuación se muestra una lista con las que se incluyen en el proyecto:

Gema	Propósito
Rails	Incluye todas las funcionalidades del framework web. Aquí se tiene que especificar la versión con la que se está trabajando.
Jquery-rails	Provee el controlador para poder usar la API de Jquery y JqueryUI
devise	Incluye un módulo para realizar la identificación y el registro de usuarios en el sistema.
debugger	Este módulo nos proporciona poder realizar debugger en el sistema de desarrollo.
awesome_nested_set	Esta gema nos permite poder crear una estructura de árbol.
bootstrap-sass	Esta módulo nos da la oportunidad de usar la api de bootstrap.
haml	Nos proporciona poder crear marcado html, usando la indentación del código. Es otra forma de crear código html de una forma más fácil.
mysql2	Nos permite usar SGBD MYSQL
simple_form	Esta gema nos permite hacer formulario.
state_machine	Este módulo nos permite realizar una máquina de estado
the_sortable_tree	Esta funcionalidad nos permite que el árbol se vea en forma de drag and drop.

5.5.2 Controladores

En esta sección veremos que el controlador es una clase Ruby que hereda de ApplicationController::Base y contiene métodos como cualquier otra. Los controladores son los encargados de recibir las peticiones de los usuarios y disparar las acciones para poder devolver la respuesta a los usuarios. Se implementan mediante clases en la carpeta app/controllers y necesitan componerse de métodos que se denominarán "acciones".

Para crear un controlador se usa el siguiente script:

```
rails g controller task index
=>
create app/controllers/tasks_controller.rb
router get "tasks/index"
invoke erb
create app/views/tasks
create app/views/tasks/index.html.erb
```

De esta manera se ha creado un controlador para la gestión de tareas (TasksController) con un método index, que se usará para listar el total de tareas que pertenece a un usuario. Al ejecutar el comando se puede comprobar que se crea la clase correspondiente al controlador, se añade al fichero de rutas el mapeador de la URL tasks/index, y se crean las vistas asociadas. Las rutas sirven para conocer qué controlador y qué método tiene que mapear cuando se reciba una petición. Rails "por defecto" divide las URL "rutas" de la siguiente forma:

- http://direccionservidor/controlador/acción

Por tanto, si se recibe una petición que proviene dehttp://localhost:3000/tasks/index, el ControladorTasksController será el encargado de recibirla y ejecutar el método index,

renderizando posteriormente la vista index.html.erb (que contiene el código HTML que se mostrará por pantalla al usuario). El siguiente código muestra un ejemplo de clase para controlado:

```
class TasksController < ApplicationController
def  index
  @tasks= current_user.tasks.all
  # [...]
  respond_to do |format|
    format.html # render index.html.erb
    format.json do
    # [...]
    end
  end
end




def show
   @tasks= current_user.tasks.find(params[:id])
  # [...]
  respond_to do |format|
    format.html # render index.html.erb
    format.json do
   # [...]
    end
 end
end
```

Observando el método show, se puede ver cuando se ejecuta esta acción se crea una instancia de la variable @tasks con el contenido de una petición al modelo task, buscando uno que concuerde con el parámetro pasado. El parámetro puede ser de dos tipos HTTP GET o POST. Para usar la variable instanciada, únicamente hay que irse a la vista correspondiente y hacer una llamada.

```
<div id="tasks">
  <ul>
      <% @tasks.each do |S| %>
        <li>
          <p><%= s.name %></p>
          <p><%= s.description %></p>
        </li>
      <%end%>
  </ul>
</div>
```

5.5.2.1 Fichero de rutas

Sobre las rutas y el fichero que mapea se ha hablado brevemente en la sección anterior. Está estrechamente con los controladores, por lo que se pasa a ver un poco más en detalle su funcionamiento.

A modo de recordatorio se presenta la definición de ruta: "todo aquello que definen el nombre y dirección de las URL dentro de una aplicación web, y que es mapeada a través de un controlador y una acción."

Para entender el uso de rutas, hay que hablar un poco de la arquitectura REST. REST es una técnica de arquitectura software para sistemas distribuidos. En la actualidad se usa para describir cualquier interfaz web simple que utiliza XML y HTTP para intercambiar mensajes. REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

- **Protocolo cliente/servidor sin estado**. Cada mensaje HTTP contiene toda la información necesaria para comprender la petición.
- **Conjunto de operaciones bien definidas**. Que se aplican a todos los recursos de información. HTTP en sí define un conjunto de operaciones: GET, POST, PUT y DELETE.
- **Sintaxis universal para identificar recursos**. En un sistema REST cada recurso es direccionadle únicamente a través de su URL.

Un concepto importante en REST es la existencia de recursos (elementos de información), que puedan ser accedidos utilizado un identificador global (mapeada sobre la URL). Para manipular estos recursos, los clientes y servidores se comunican a través de un interfaz estándar (HTTP) e intercambian representaciones de estos recursos (los ficheros HTML que se descargan y envían al navegador).

El fichero config/routes.rb es el encargado de realizar el enrutamiento en Rails. La siguiente tabla muestra la forma en que los verbos HTTP son usados junto con las rutas para ejecutar una determinada acción en un controlador:

Verbo	Ruta	Acción	Propósito
GET	/tasks	index	Muestra una lista con todos las tereas de usuario
GET	/tasks/new	New	Devuelve un formulario HTML para crea una tarea
POST	/tasks	create	Crea una nueva tarea
GET	/tasks/:id	show	Muestra un nadador específico con el parámetro id
GET	/tasks/:id/edit	edit	Devuelve un formulario HTML para modificar un nadador
PUT	/tasks/:id	update	Modifica un nadador específico

DELETE	/taks/:id	destroy	Elimina un nadador específico

La siguiente imagen representa un pequeño ejemplo de la forma en que se utilizan los verbos en los ficheros de rutas (config/routes.rb).

Se pueden definir recursos (resources :tasks), que incluye automáticamente el mapeo de todos los verbos y rutas señalados en la tabla anterior. Sobre el controlador se tendrán que crear las acciones necesarias para soportarlo.

Existe la posibilidad de crear tantas rutas como sean necesarias.

```
SWN::Aplication.routes.draw do
  # [...]
  resources :tasks, :path=> "tareas" # [...]
  # [...]
  root :to =>"pages#index"
end
```

Rails ofrece la utilización de un comando "rake routes" para ver cuáles son las rutas accesibles en la aplicación desarrollada.

A continuación se muestra las rutas generadas automáticamente cuando se inserta un recurso, donde aparece el verbo HTTP que se usa, la ruta mapeada desde el navegador y la combinación controlador/acción que se hará cargo de la petición.

```
rake routes
=>
GET        /tasks(.:format)          tasks#index
POST          /tasks(.:format)       tasks#create
GET        /tasks/new(.:format)      tasks#new
GET        /tasks/:id/edit(.:format) tasks#edit
GET        /tasks/:id(.:format)      tasks#show
PUT        /tasks/:id(.:format)      tasks#update
DELETE     /tasks/:id(.:format)      tasks#destroy
```

5.5.2.2. Lista de Controladores Creados en el Proyecto

Nombre de controlador	Propósito
application_controller.rb	Controlador principal de la aplicación. Es el controlador padre de rails del cual heredaran los demás controladores
Task_Controller.rb	Controlador que soportará la lógica de las tareas
Project_controller.rb	Controlador que soportará la lógica de los proyectos
Interrupction_controller.rb	Controlador que soportará la lógica de las interrupciones
User_controller.rb	Soporta la lógica para la gestión de cuentas de usuarios en la aplicación.

5.5.3 Modelos

El modelo es el componente encargado del acceso a datos. La capa de modelo comprende los objetos de negocio almacenados en nuestras fuentes de datos además de definir la lógica de negocio y los accesos a las fuentes de datos. Toda aplicación necesita almacenar la información de la aplicación web en una base de datos. Sin embargo, se hace difícil combinar el uso de bases de datos relacionales con los lenguajes de programación orientados a objetos (OO). A la larga se suelen obtener códigos que son muy difíciles de mantener debido al intercambio de sentencias SQL y del código orientado a objetos que se use.

Rails ofrece una alternativa donde se encapsula el acceso a la base de datos detrás de una capa de clases independiente del SGBD utilizado. La aplicación usa estas clases y sus objetos, que nunca interactúan con la base de datos directamente. Se encapsula toda la funcionalidad en una capa y se desacopla el código de la capa de nivel inferior con detalles de acceso a la base de datos. Esto se realiza con el uso de las tecnologías ORM. Las librerías ORM mapean las tablas de la base de datos en clases. Si la base de datos tiene una tabla llamada tareas, la aplicación tendrá una clase llamada Tarea. Las filas en esta tabla corresponden a objetos de la clase, y las columnas como atributos de dicho objeto. ActiveRecord es la capa ORM proporcionada por Rails.

Los modelos son archivos de Ruby (.rb) que se encuentran en el directorio app/models. Heredan de ActiveRecord::Base, que como se ha explicado, es la interfaz sobre la que se apoya Rails para ORM.

Para crear modelos, Rails ofrece la posibilidad de usar un comando por consola:

```
#comando para crear un modelo y su correspondiente migración
rails model Task name:string description:string # [...]
=>
   create app/models/tasks.rb
   create db/migrate/25092013091453_create_tasks.rb
```

Los parámetros del comando serán los campos de la tabla o los atributos del objeto que representa al modelo. Se crea un fichero task.rb con la clase que contiene al modelo y un archivo con una migración "se explica en la siguiente sección". A continuación se muestra un fragmento del modelo para tareas (app/models/task.rb):

```
class Task < Node
  include ActiveModel::ForbiddenAttributesProtection
 # Relación con los modelos
  has_many :interruptions

  # Máquina de estado para las tareas
  state_machine :state, :initial => :to_do do

    event :activate do
      transition [:to_do, :paused, :expired] => :active
    end

    event :finish do
      transition [:active] => :finished
    end

    event :pause do
      transition [:active] => :paused
    end

    event :expire do
      transition [:active] => :expired
    end

  end
end
```

La primera de las partes está relacionada con el uso de métodos para gestionar las relaciones entre modelos del sistema. La segunda parte es algo específica para este modelo, es la lógica de una máquina de estado necesaria para controlar el estado de las tareas.

A parte es muy común encontrar validaciones en los modelos, aunque en este modelo no se muestre por el momento ninguna validación de un formulario por ejemplo.

5.5.3.1 Relaciones Entre Modelos

Habiendo conocido algo más sobre la forma de trabajar con los modelos, es hora de hablar sobre las relaciones en los modelos de Rails, parte fundamental para poder trabajar con soltura.

Como es normal, el protagonista sigue siendo ActiveRecord. Se está tratando con objetos de modelo y sus relaciones, y no con filas y columnas, ya que parte de la magia de ORM es que puede convertir las relaciones de clave externa hacia otra tabla referenciada Foreign Key en mapeos entre objetos de alto nivel, ayudando así a que ActiveRecord entienda las relaciones que se quieren abordar en la base de datos.

Vamos a los tipos de relaciones.

- **belongs_to**. Declara que la clase tiene una relación de padre con la clase que contiene la declaración.
- **hasone**. Declara que una clase determinada es un hijo de su clase con una clave externa foreign key.
- **hasmany**. Define un atributo que se comporta como una colección de los objetos hijo, accediendo a lo hijos como un array. Usado para crear relación uno-a-muchos.
- **hasone :through**. Define una conexión uno-a-uno a través de un modelo que hace de conexión entre otros dos modelos
- **hasmany :through**. Define una conexión uno-a-muchos a través de un modelo que hace de conexión entre otros dos modelos.
- **hasandbelongstomany**. Crea un atributo que es una colección.

5.5.3.2 Validaciones

Durante las operaciones normales que se realizan en una aplicación, los objetos pueden ser creados, modificados o eliminados. ActiveRecord provee de medios en el ciclo de vida de los objetos para que se puedan controlar estas operaciones.

Las validaciones permiten asegurarnos de que únicamente se insertarán datos válidos en la base de datos. Por ejemplo: puede ser útil comprobar que si un tarea tiene una campos con nombre, descripción y fecha de vencimiento "entre otros", pues que si no se cumplimentan estos campos, el registro no sea insertado en la base de datos al no ser válido.

Existen varias fases en la que estas comprobaciones pueden hacerse:

- Validaciones en el cliente usando Javascript y otras técnicas.
- Validaciones a nivel de controlador, que pueden llegar a ser muy difíciles de mantener, puesto que añaden más lógica de la que se debería.
- Validaciones a nivel de modelo. A éstas últimas son las que nos referimos en esta sección y que ActiveModel facilita mediante diferentes métodos. Son innumerables los tipos de validaciones que ofrece la API.

5.5.3.3 Lista de Modelos Creados en el Proyecto

Nombre de controlador	Propósito
Node.rb	Lógica de negocio para las nodos un nodo puede ser tareas o proyecto.
Task.rb	Lógica de negocio para las tareas
Project.rb	Lógica de negocio para los proyectos
Interrupction.rb	Lógica de negocio para las interrupciones
User.rb	Soporta la lógica negocio usuarios del sistema

5.5.4 Vistas

La última de las secciones para conocer como está estructurada la implementación de la aplicación es la de las vistas. Siguiendo el patrón de MVC y, como ya se ha explicado, una petición que llega desde el navegador a la aplicación, es recibida mediante una URL a través de un "dispatcher" por el controlador.´

Este, compuesto por métodos que realizan diferentes acciones, se encargará de realizar las acciones pertinentes a través del modelo. Por ejemplo, buscar datos, eliminarlos, guardarlos, etc. Una vez realizada la acción, irá en busca de la vista que muestra al usuario la respuesta del sistema. Por tanto, la vista es lo que ve el usuario: la interfaz del sistema a través del la cual realiza peticiones. Las vistas son básicamente ficheros *.erb (código ruby embebido) que contienen su parte estática en HTML y su parte dinámica en código Ruby. En el directorio de trabajo se encuentran en app/views. Los ficheros Javascript, CSS y de imágenes se encuentran en el directorio public/.

Cuando se trata de las vistas, éstas se pueden componer en 3 tipos diferentes:

- **Layouts o plantillas**. Las aplicaciones suelen tener una interfaz en la que la distribución de elementos suele ser similar en todas las pantallas que la forman. Para la reutilización de código se usan los layouts, con los que se proveerá de la estructura común que tendrán todas las vistas que los utilicen.

  A continuación se muestra un ejemplo de código para vista que se usa como plantilla. Las vistas de acción que las usan se incluirán cuando se ejecute el código de bloque (yield).

```
!!!
%html
  %head
    %meta{:charset => "utf-8"}
    %meta{:name => "viewport", :content => "width=device-width, initial-scale=1, maximum-scale=1"}
    %title= content_for?(:title) ? yield(:title) : "GTD Task Manager"
    %meta{:content => "GTD Task Manager | web application using GTD Task manager", :name => "description"}
    %meta{:content => "José Aythami Ramírez medina", :name => "author"}
    = stylesheet_link_tag "application", :media => "all"
    = javascript_include_tag "application"
    = csrf_meta_tags
    = yield(:head)
  %header
    .navbar.navbar-fixed-top
      %nav.navbar-inner
        .container
          = render 'layouts/navigation'
    #main{:role => "main"}
      .container
        = render 'layouts/messages'
        = yield
      %footer
        @ FAQ 2013-3000 GTD task Manager, Inc. All Rights Reserve
```

```
= javascript_include_tag 'application'
= include_gon
```

- **Vistas de acción**. Vistas por defecto donde se muestra las acciones de un controlador. Están separadas en directorios dentro de la carpeta app/views/ en función del controlador que las use, teniendo el mismo nombre que la acción que la renderiza. Por ejemplo, app/views/tasks/index.html.erb contendría la vista que se renderiza para la acción index del controlador de tarea . En ellas se puede acceder a las variables instanciadas por el controlador. Así mismo, pueden contener vistas parciales, código HTML/XML, CSS, Javascript y Ruby embebido.

- **Parciales**. Es un mecanismo para poder reutilizar código en las vistas y manejar de una forma más sencilla pequeños pedazos de código. La convención de nombres que se usa es insertando un guion bajo delante del nombre de una vista. Por ejemplo: app/layouts/_navegation.html.erbes una vista parcial del menú de la aplicación, puesto que en todas las pantallas se necesita mostrar el menú. Un detalle a tener en cuenta es que desde la vista parcial se puede acceder a las mismas variables de instancia del controlador a las que se puede acceder desde la vista que la incluye.

Para incluir una vista parcial únicamente habría que poner el siguiente código:

```
#Llamamos  a la navegación que en todas las páginas son iguales.
= render 'layouts/navigation'
```

La vista parcial se busca en el mismo directorio de la vista que la incluye. Por tanto, habría que buscar el parcial dentro de app/views/layouts

5.5.5 Progresión de las Iteraciones en la Implementación

El proceso de construcción del proyecto ha sido iterativo incremental. A continuación se muestra la progresión a lo largo de las iteraciones realizadas. Decir que cada iteración conlleva la realización de las fases de análisis, diseño y construcción; ciclo de vida que se repite hasta llegar a versiones estables.

Numero de iteraciones	Descripción
Iteración 1	Preparar el sistema Instalación del entorno de trabajo Configuración del entorno de trabajo
Iteración 2	Registro e identificación de usuario Recuperación de contraseña Cierre de sesión Modificar datos de cuenta
Iteración 3	Página de inicio Página de tour Página blog Página de Faq
Iteración 4	Página work desk "interfaz"

	Añadir tarea Ver lista de tarea y tarea específica Eliminar y modificar tarea
Iteración 5	Crear proyectos Ver y modificar proyectos Eliminar proyectos
Iteración 6	Crear gestión de control de tiempo Crear gestión GTD Crear revisión de GTD

5.6 Pruebas

La prueba de software es una actividad dentro del desarrollo del software en la cual un sistema se ejecuta en circunstancias previamente especificadas, registrándose los resultados obtenidos. El objetivo en las pruebas es realizar un proceso de evaluación en el que los resultados obtenidos se comparan con los resultados esperados para localizar fallos en el software. Nuestro objetivo es pues, diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de esfuerzo y tiempo.

La realización de las pruebas no es asegurar la ausencia de defectos en un software, únicamente pueden demostrar que existen defectos en el software. Esto se realiza para obtener una determinada predicción de fiabilidad o un cierto nivel de confianza en el software probado.

A continuación se presentan algunas características de una buena prueba:

1. Una buena prueba ha de tener una alta probabilidad de encontrar un fallo.
2. Una buena prueba debe centrarse en dos objetivos: probar si el software no hace lo que debe hacer, o probar si el software hace lo que no debe hacer.
3. Una buena prueba no debe ser redundante. El tiempo y los recursos son limitados, así que todas las pruebas deberían tener un propósito diferente.
4. Se debería emplear la prueba que tenga la más alta probabilidad de descubrir una clase entera de errores.
5. Una buena prueba no debería ser ni demasiado sencilla ni demasiado compleja. Cada prueba debería realizarse separadamente.

Para probar el software se seguirá el siguiente proceso:

1. **Metodología de Pruebas.** Aprenderemos cual es la metodología correcta para la implementación de las pruebas.
2. **Recopilación de información de niveles de prueba.** Se recopilará información sobre los diferentes niveles de pruebas existentes en desarrollo del software.
3. **Recopilación de información de tipos de pruebas .** Se recopilará información sobre algunos tipos de prueba.
4. **Implementación de las pruebas realizadas.** En éste paso lo que se pretende es representar los casos de pruebas, los datos que se utilizarán como entrada para ejecutar el software a probar. Aplicando los casos de prueba generados previamente e identificando los posibles fallos producidos al comparar los resultados esperados con los obtenidos se corrigieran las faltas asociadas a los fallos identificados.

Podría ser normal que el lector esperase encontrar en esta sección un listado con la especificación de cada uno de los casos de prueba generados.

5.6.1 Metodología de Prueba

Los procesos de aseguramiento de calidad de un producto de software suelen dividirse en lo que respecta a su componente analítico en pruebas estáticas y dinámicas. La diferencia fundamental entre estos tipos de pruebas, radica en que las pruebas estáticas se centran en evaluar la calidad con la que se está generando la documentación del proyecto, por medio de revisiones periódicas, mientras que las pruebas dinámicas, requieren de la ejecución del software con el fin de medir el nivel de calidad con la que este fue codificado y el nivel de cumplimiento en relación con la especificación del sistema.

Realizar pruebas dinámicas a un producto de software, suele en la mayoría de los casos confundirse con una simple actividad de ejecución de pruebas y reporte de incidencias, sin embargo, para productos de complejidad media en adelante, lo recomendable es implementar de manera formal una metodología de pruebas que se ajuste y acople uniformemente con la metodología de desarrollo seleccionada por la firma desarrolladora.

Para procesos de desarrollo basados en la metodología RUP o métodos tradicionales, implementar una metodología de pruebas es totalmente viable, teniendo en cuenta que estas metodologías están orientadas a la documentación y a la formalización de todas las actividades ejecutadas. Si por el contrario, la firma desarrolladora guía su proceso bajo lineamientos basados en metodologías ágiles, será necesario reevaluar la conveniencia de ejecutar todas las actividades que implica un proceso de pruebas formal, lo que en la mayoría de los casos, conlleva a reducir al mínimo las actividades relacionadas con un proceso de pruebas, circunstancia que naturalmente puede desencadenar en la liberación de productos con bajos niveles de calidad.

Un proceso de pruebas formal, está compuesto, cuando menos por las siguientes 5 típicas etapas:

1. Planeación de pruebas.
2. Diseño de pruebas.
3. implementación de pruebas.
4. Evaluación de criterios de salida.
5. Cierre del proceso.

5.6.1.1 Planeación de Pruebas.

Es la etapa en donde se ejecutan las primeras actividades correspondientes al proceso de pruebas y tiene como resultado un entregable denominado plan de pruebas el cual debe estar conformado en cuando menos por aspectos tales como:

- **Alcance de la prueba:** determina que funcionalidades del producto y/o software serán probadas durante el transcurso de la prueba. Este listado de funcionalidades a probar se extrae con base a un análisis de riesgos realizado de manera previa, que tienen en cuenta variables tales como el impacto que podría ocasionar la falla de una funcionalidad y la probabilidad de falla de una funcionalidad. Producto de este análisis, se cuenta con información adicional que permite determinar además del alcance detallado del proceso de pruebas, la prioridad con la que las funcionalidades deben probarse y la profundidad de las pruebas.
- **Tipos de Prueba:** en este punto se debe determinar qué tipos de pruebas requeriría el producto. No todos los productos de software requieren la aplicación de todos los tipos de pruebas que existen, por esta razón, es estrictamente necesario que el líder de pruebas se plantee preguntas que le permitan determinar

qué tipos de prueba son aplicables al proyecto en evaluación. Los posibles tipos de prueba a aplicar son: pruebas de stress, pruebas de rendimiento, pruebas de carga, pruebas funcionales, pruebas de usabilidad, pruebas de regresión, entre otros.

- **Estrategia de Pruebas:** teniendo en cuenta que no es viable probar con base a todas las posibles combinaciones de datos, es necesario determinar a través de un análisis de riesgos sobre que funcionalidades debemos centrar nuestra atención. Adicionalmente, una buena estrategia de pruebas debe indicar los niveles de pruebas (ciclos) que aplicaremos y la intensidad o profundidad a aplicar para cada nivel de pruebas definido. En este punto también es importante definir los criterios de entrada y salida para cada ciclo de pruebas a ejecutar.
- **Criterios de Salida:** entre las partes involucradas en el proceso, se define de manera formal, bajo qué condiciones se puede considerar que una actividad de pruebas fue finalizada. Los criterios de salida se deben definir para cada nivel de pruebas a ejecutar. Algunos ejemplos de criterios de salida que pueden ser utilizados son: porcentaje de funcionalidades de alto riesgo probadas con éxito, número defectos críticos y/o mayores aceptados, etc.
- **Otros aspectos:** tal y como se realiza en cualquier plan de proyecto, se debe incluir una estimación de tiempos, los roles y/o recursos que harán parte del proceso, la preparación del entorno de pruebas, cronograma base, etc.

5.6.1.2 Diseño de Pruebas

Una vez elaborado y aprobado el plan de pruebas, el equipo de trabajo debe iniciar el análisis de toda la documentación existente con respecto al sistema, con el objeto de iniciar el diseño de los casos de prueba. Los entregables claves para iniciar este diseño pueden ser: casos de uso, historias de usuario, arquitectura del sistema, diseños, manuales de usuario (si existen), manuales técnicos (si existen). El diseño de los casos, debe considerar la elaboración de casos positivos y negativos.

Los casos de prueba negativos permiten validar cómo se comporta el sistema ante situaciones atípicas y permite verificar la robustez del sistema, atributo que constituye unos de los requerimientos no funcionales indispensable para cualquier software. Por último, es necesario definir cuáles son los datos de prueba necesarios para la ejecución de los casos de prueba diseñados.

5.6.1.3 Implementación y Ejecución de Pruebas

La ejecución de pruebas debe iniciar con la creación de los datos de prueba necesarios para ejecutar los casos de prueba diseñados. La ejecución de estos casos, puede realizarse de manera manual o automatizada; en cualquiera de los casos, cuando se detecte un fallo en el sistema, este debe ser documentado y registrado en una herramienta que permita gestionar los defectos (Bug Tracker). Una vez el defecto ha sido corregido por la firma desarrolladora en su respectivo proceso de depuración, es necesario realizar un re-test que permita confirmar que el defecto fue solucionado de manera exitosa. Por último, es indispensable ejecutar un ciclo de regresión que nos permita garantizar, que los defectos corregidos en el proceso de depuración de la firma, no hayan desencadenado otros tipos de defectos en el sistema.

5.6.1.4 Evaluación de Criterios de Salida

Los criterios de salida son necesarios para determinar si es posible dar por finalizado un ciclo de pruebas. Para esto, es conveniente definir una serie de métricas que permitirán al finalizar un proceso de pruebas, comparar los resultados obtenidos contra las métricas definidas, sí los resultados obtenidos no superan la métricas definidas, no es posible continuar con el siguiente ciclo de pruebas.

Existen varios tipos de criterios de salida dentro de los cuales se pueden mencionar: cubrimiento de funcionalidades en general, cubrimiento de funcionalidades críticas para el sistema, Número de defectos críticos y mayores detectados, etc. También es importante aclarar que el proceso de pruebas puede ser suspendido y/o paralizado, debido entre otros, a aspectos relacionados con el presupuesto y la calidad mínima del sistema requerida para el inicio formal de pruebas.

5.6.1.5 Cierre del Proceso

Durante este periodo de cierre el cual históricamente se ha comprobado que se le destina muy poco tiempo en la planeación, se deben cerrar las incidencias reportadas, se debe verificar si los entregables planeados han sido entregados y aprobados, se deben finalizar y aprobar los documentos de soporte de prueba, analizar las lecciones aprendidas para aplicar en futuros proyectos, etc.

5.6.2 Niveles de Pruebas del Desarrollo del Software

En un proceso de pruebas formal, suelen confundirse con mucha facilidad, los niveles de pruebas con los tipos de prueba, y a pesar de que se encuentren íntimamente relacionadas, tienen connotaciones diferentes en el proceso. Para entender un poco más, vamos a partir del hecho de que las pruebas pueden ejecutarse en cualquier punto del proceso de desarrollo de software, y es aquí donde los niveles de prueba nos permiten entender con claridad los diferentes puntos o etapas en donde pueden ejecutarse ciertos tipos de prueba. Por lo anterior, es común que algunas personas se refieran a los niveles de pruebas o intenten clasificarlos como: pruebas de desarrollador, pruebas funcionales y pruebas de usuario final. Sin embargo, la terminología apropiada para referirse a los diferentes niveles corresponde a la siguientes cuatro clasificaciones que son:

- pruebas unitarias
- pruebas de integración
- pruebas de sistema
- pruebas de aceptación

A continuación una breve descripción de cada nivel de prueba:

5.6.2.1 Pruebas Unitarias

Son los tests más importantes, los ineludibles. Cada prueba unitaria "unit test" en inglés es un paso que andamos en la implementación del software.

Estas pruebas son ejecutadas normalmente por el equipo de desarrollo, básicamente consisten en la ejecución de actividades que le permitan verificar al desarrollador que los componentes unitarios están codificados bajo condiciones de robustez, esto es, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada. Adicionalmente, Las pruebas sobre componentes unitarios, suelen denominarse pruebas de módulos o pruebas de clases, siendo la convención definida por el lenguaje de programación la que influye en el término a utilizar.

Por último, es importante que toda la funcionalidad de cada componente unitario sea cubierta, por al menos, dos casos de prueba, los cuales deben centrarse en probar al menos una funcionalidad positiva y una negativa.

Todo test unitario debe ser:

1. **Atómico**. significa que el test prueba la mínima cantidad de funcionalidad posible. Esto es, probará un solo comportamiento de un método de una clase. El mismo método puede presentar distintas respuestas ante distintas entradas o distinto contexto. La prueba unitaria se ocupará exclusivamente de uno de esos comportamientos, es decir, de un único camino de ejecución. A veces, la llamada al método provoca que internamente se invoque a otros métodos; cuando esto ocurre, decimos que el test tiene menor granularidad. Lo ideal es que los test unitarios prueban lo que es indivisible, pero sin embargo hay veces que vale la pena ser menos estrictos con la atomicidad del test para evitar abusar de los dobles de prueba. Independiente significa que un test no puede depender de otros para producir un resultado satisfactorio. No puede ser parte de una secuencia de tests que se deba ejecutar en un determinado orden.

2. **Independiente**. La prueba debe funcionar siempre igual independientemente de que se ejecuten otras pruebas o no.
3. **Inocuo**. Significa que no altera el estado del sistema. Al ejecutarlo una vez, produce exactamente el mismo resultado que al ejecutarlo veinte veces. No altera la base de datos, ni envía emails, ni crea ficheros, ni los borra.
4. **Rápido**. Rápido tiene que ser porque ejecutamos un gran número de tests cada pocos minutos y se ha demostrado que tener que esperar unos segundos tras la ejecución de la batería resulta muy improductivo. Un sólo test tendría que ejecutarse en una pequeña fracción de segundo.

5.6.2.2 Pruebas de Integración

Son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Este tipo de pruebas son ejecutas por el equipo de desarrollo y consisten en la comprobación de que elementos del software que interactúan entre sí, funcionan de manera correcta.

Las pruebas de integración (algunas veces llamadas integración y testeo) es la fase de la prueba de software en la cual módulos individuales de software son combinados y probados como un grupo. Son las pruebas posteriores a las pruebas unitarias y preceden a las pruebas del sistema.

5.6.2.3 Pruebas de Sistema

Este tipo de pruebas deben ser ejecutadas idealmente por un equipo de pruebas ajeno al equipo de desarrollo, una buena práctica en este punto corresponde a la tercerización de esta responsabilidad. La obligación de este equipo, consiste en la ejecución de actividades de prueba en donde se debe verificar que la funcionalidad total de un sistema fue implementada de acuerdo a los documentos de especificación definidos en el proyecto.

Los casos de prueba a diseñar en este nivel de pruebas, deben cubrir los aspectos funcionales y no funcionales del sistema. Para el diseño de los casos de prueba en este nivel, el equipo debe utilizar como bases de prueba entregables tales como: requerimientos iníciales, casos de uso, historias de usuario, diseños, manuales técnicos y de usuario final, etc. Por último, es importante que los tipos de pruebas ejecutados en este nivel se desplieguen en un ambiente de pruebas / ambiente de pre-producción cuya infraestructura y arquitectura sea similar al ambiente de producción, evitando en todos los casos utilizar el ambiente real del cliente, debido principalmente, a que pueda ocasionar fallos en los servidores, lo que ocasionaría indisponibilidad en otros servicios alojados en este ambiente.

5.6.2.4 Pruebas de Aceptación

Independientemente de que el proceso de pruebas lo hagan hecho terceros , y que la firma responsable de estas actividades haya emitido un certificado de calidad sobre el sistema objeto de prueba, es indispensable, que el cliente designe a personal que haga parte de los procesos de negocio para la ejecución de pruebas de aceptación, es incluso recomendable, que los usuarios finales que participen en este proceso, sean independientes al personal que apoyó el proceso de desarrollo. Cuando las pruebas de aceptación son ejecutadas en instalaciones o ambientes proporcionados por la firma desarrolladora se les denominan pruebas Alpha, cuando son ejecutadas desde la infraestructura del cliente se les denomina pruebas Beta. En los casos en que las pruebas de aceptación del producto se hayan ejecutado en el ambiente del proveedor, el aplicativo no podrá salir a producción, sin que se hayan ejecutados las respectivas pruebas Beta en el ambiente del cliente, de lo anterior es importante concluir, que las pruebas Alpha son opcionales, pero las pruebas Beta son obligatorias.

5.6.3 Tipos de Pruebas del Desarrollo del Software

En cada uno de estos niveles de prueba citados anteriormente, se podrán ejecutar diferentes tipos de prueba tales como: pruebas funcionales, no funcionales, de arquitectura y asociadas el cambio de los productos.

A continuación mostrará algunos tipos de pruebas realizados concretamente para este proyecto final de grado:

- Pruebas de testing
- Pruebas de estrés
- Pruebas de validación de interfaces
- Pruebas de corrección de contenido

5.6.3.1 Pruebas de Testing

En el mercado existen bastantes empresas que se dedican a buscar testers para probar las aplicaciones de clientes que contratan sus servicios. En nuestro caso, se ha contado con la presencia de especialistas en el ámbito del software. Esto es fundamental en la vida del producto porque se obtiene un feedback rápido de usuarios específicos.

Cuando se habla de pruebas de usuarios, no sólo tenemos que orientarlas a probar el sistema desarrollado. Este tipo de pruebas ayudan a evaluar todo el proceso de construcción del software junto al cliente. Como se menciona a lo largo del documento, la participación del cliente es fundamental para conseguir que el producto tenga éxito.

A lo largo de las distintas iteraciones, junto al cliente se lleva a cabo la evaluación de:

- Necesidades que pueden ser cubiertas por la aplicación, tanto las propuestas por los desarrolladores como las detectadas por los distintos expertos que participan en el desarrollo.
- Entendimiento del dominio que rodea a la aplicación.
- Corrección de los prototipos de interfaces presentados. La manera de interactuar entre elementos del dominio mostrada en las colaboraciones.
- Software finalizado. Tras la implementación los clientes prueban la aplicación para comprobar que efectivamente cubre las expectativas planteadas desde el inicio. A medida que el cliente iba interviniendo en el proceso, se fueron descubriendo nuevas necesidades que se plantean para un trabajo futuro.

5.6.3.2 Pruebas de Estrés

Las pruebas de estrés son una particularidad de las pruebas no funcionales y sirven para verificar el comportamiento de una aplicación bajo una demanda excesiva.

El objetivo es poder generar una gran cantidad de peticiones a la aplicación y verificar su comportamiento, y de esta manera poder garantizar el número máximo de peticiones bajo las cuales la aplicación, servidor, interacción con otros aplicativos, etc.

Para poder realizar este tipo de pruebas hay que contemplar muchas variables y se vuelve un tanto complejo su implementación y análisis.

Por ejemplo:

- Deben realizarse sobre un ambiente lo más parecido al de Producción. De esta manera, los resultados serán congruentes con los experimentados en producción.
- En caso que no se tenga un ambiente similar a Producción, el análisis se vuelve complejo; esto es porque es difícil poder extrapolar los resultados en este ambiente y llevarlos a un análisis productivos, ya que por lo general los sistemas no se comportan de una manera lineal para poder predecir.
- Por otro lado, se debe contemplar varios clientes que realicen las pruebas y los clientes deberán ejecutarse en diferentes computadoras. Esto debido a que tenemos que considerar la carga de CPU que se genera en cada computadora y no sobrecargarlas, sino las peticiones que generen irán disminuyendo.
- Si se realizan estas pruebas distributivamente, se tendrá que considerar la recolección de los resultados de cada una de las computadoras y analizar el conglomerado de datos.

En general, para poder realizar este tipo de pruebas deberá tenerse un plan bien definido de la arquitectura de servidores / computadoras que las realizarán, un método de recolección y análisis centralizado.

5.6.3.3 Pruebas Validación de Interfaces

El propósito de los test de validación de interfaces es el de encontrar errores de diseño de las interfaces de la aplicación. Todos los detalles que componen las distintas pantallas son examinadas exhaustivamente para completar todos los requisitos no funcionales especificados durante las primeras fases de desarrollo del software.

Los elementos a los que hay que prestar especial atención son:

- **Menús de navegación.** Analizar cada uno de los menús que componen la aplicación buscando elementos que no deban estar presentes.
- **Botones y enlaces**. Comprobar que cada uno de los enlaces llevan a los destinos deseados.
- **Redireccionamiento**. Lo mismo que con el punto anterior, es necesario verificar que aquellas acciones que generen un redireccionamiento, lleven al lugar adecuado.
- **Paginación**. En las interfaces que se incluyan controladores para la paginación, hay que probar que cuando no hay ningún dato insertado, éste actée como debe. Lo mismo para cuando se insertan muchos más registros.
- **Formularios**. Verificar que el contenido enviado es el mismo que el que se almacena en el sistema; comprobar que tanto los campos obligatorios como los que dependen de un formato son correctos; y probar que todos los desplegables tengan el contenido adecuado.
- **Presentación de tablas**. Las tablas son los elementos que peor se adaptan a las interfaces de dispositivos móviles, razón por la que hay que hacer hincapié en que por lo menos su visualización en altas resoluciones sea correcta. Ligado a las tablas, también se puede mencionar el orden de presentación de la información.
- **Búsquedas**. Usar los motores de búsquedas para ver si devuelven los resultados que se esperan.

5.6.3.4 Pruebas de Corrección de Contenido

Las interfaces están compuestas por textos que enriquecen la experiencia del usuario que usa la aplicación. En muchos casos, estos texto ayudan a comprender mejor las funcionalidades que se presentan en el software.

La técnica para llevar a cabo este tipo de pruebas es bastante rudimentaria. Hay que realizar la lectura de los textos de cada una de las interfaces que componen la aplicación. El objetivo es descubrir los siguientes tipos de errores:

- Sintácticos
- Tipográficos
- Gramaticales
- Semánticos
- Estructuración del contenido

5.6.4. Implementación de Pruebas Realizadas

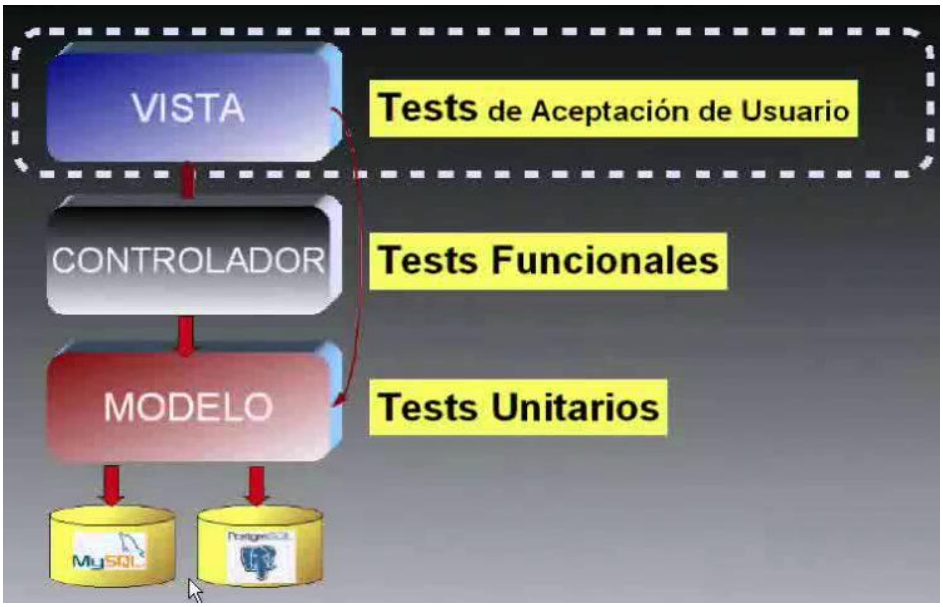
5.6.4.1 Preparación del Entorno de Pruebas

Ruby On Rails y la automatización de pruebas van de la mano. Rails permite la inclusión de entornos para pruebas de forma muy fácil. Además existen también diversas gemas que da soporte para testing. Por ello, antes de comenzar a mostrar el proceso de configuración del entorno de pruebas, se quiere hacer mención a las herramientas utilizadas.

5.6.4.2 Herramientas Utilizadas

- **Navegador Chrome**. Tanto para la ejecución de pruebas automáticas como aquellas realizadas por los clientes y equipo, se usa el navegador web Chrome o firefox. La razón por la que se usa el Navegador Chrome en exclusiva es porque ofrece una gran variedad de funciones para poder examinar códigos (HTML, CSS y Javascript) en el momento de la ejecución.

- **Test**. Es un entorno para pruebas que viene con el framework rails en sí, que permite diseñar escenarios de forma fiable, fáciles de escribir y de entender. A través de esta funcionalidad de la gema de rails, se creará prueba unitarias, funcionales y de aceptación de usuario.



5.6.4.3 Instalación de la Funcionalidad de Testing en Rails

Antes de de realizar los test tenemos que configurar el entorno para que acepten las pruebas.

1. Configure su config / database.yml de la siguiente manera:
   test:
     adapter: mysql2

```
encoding: utf8
database: prototype_test
username: root
password:
host: localhost
pool: 5
timeout:5000
```

2. A continuación debemos Configurar la base de datos:

   Hasta ahora hemos utilizado sólo base de datos de desarrollo de la aplicación, pero ahora es necesario asegurarse de que también se cree la base de datos de pruebas. Vamos a crear bases de datos de pruebas de la siguiente manera:

   rake db:test:prepare

A continuación se muestra un ejemplo de prueba tanto en el controlador como en el modelo. El objetivo no es acabar mostrando cada uno de las pruebas realizadas, sino mostrar el código base que se necesita conocer para comprender como se realiza la metodología de las pruebas.

5.6.4.4 Realización de las Pruebas en el Modelo

Cuando se genera un modelo, Rails también genera un script de prueba unitaria para el modelo en el directorio de prueba. En dicho archivo podemos escribir nuestras pruebas.

A continuación se mostrará algunos pequeños ejemplo de uso de las pruebas de unidad.

El formato de las pruebas básica es así:

```
require 'test_helper'
class TaskTest < ActiveSupport::TestCase
  test " should create a task " do
          # here code  unit test
  end
end
```

A continuación escribimos unos cuantos ejemplos.

```
require 'test_helper'

class TaskTest < ActiveSupport::TestCase
    test " should create a task " do
    task = Task.new
    task.task_name = 'task1'
    task.note = 'this is task 1'
    task.action = 'Inbox'
    assert task.save
  end
```

```
  test "should delete a task" do
    task = Task.new
    task.task_name = 'task1'
    task.note = 'this is task 1'
    task.action = 'Inbox'
    assert task.save
    task_id = Node.find(task.id)
   assert task_id.destroy
  end
end
```

Por último ejecutamos la prueba de la siguiente manera:

Rake test:units

Aquí está la salida de ejecutar el caso de prueba con éxito:

Running tests:

..

Finished tests in 0.132365s, 15.1098 tests/s, 22.6647 assertions/s.

2 tests, 3 assertions, 0 failures, 0 errors, 0 skips

5.6.4.5 Realización de las Pruebas en el Controlador

Las pruebas en el controlador también son conocidas como pruebas funcionales. Las pruebas funciones se realiza para intentar probar los siguientes tipos de funcionalidades de los controladores:

- Si el redireccionamiento es el esperado
- Si presenta la plantilla prevista
- Si el enrutamiento es el adecuado

Rails soporta 5 tipos de solicitudes y cuando se realiza pruebas funcionales es necesario simular cualquiera de los cinco tipos de peticiones HTTP que su controlador procesará:

- get
- post
- put
- head
- delete

Los tipo de solicitud "get " y "post " son los más utilizados en las pruebas de control. Todos estos métodos tienen cuatro argumentos :

- La acción de un controlador
- Una almohadilla opcional de parámetros de la petición
- Un hash de sesión opcional
- Un hash de flash opcional

A continuación vamos a ver algunos ejemplos de cómo podemos probar nuestros controladores.Cuando se genera un controlador, Rails crea un script de prueba funcional para el controlador, este archivo de prueba funcional lo crea en la carpeta de prueba.

```
require 'test_helper'
class TasksControllerTest < ActionController::TestCase
  include Devise::TestHelpers
  setup do
    @user = User.create(first_name: 'Jhon', last_name: 'wu', email: 'prueba@gmail.com',
password: '123456789', password_confirmation: '123456789')
    sign_in @user
  end

  test "should get index" do
    get :index
    assert_response :success
  end

  test "should create task" do
    assert_difference('Task.count') do
      post :create, task: { task_name: 'task1', note: 'task 1 is  the first task', action: 'Inbox' }
    end

    assert_redirected_to tasks_path
  end

  test "should show task" do
    @task = Task.create(task_name: 'task1', note: 'task 1 is  the first task', action: 'Inbox' )
    get :show, id: @task.id
    assert_response :success
  end

  test "should get edit" do
    @task = Task.create(task_name: 'task1', note: 'task 1 is  the first task', action: 'Inbox' )
    get :edit, id: @task.id
    assert_response :success
  end

  test "should update task" do
    @task = Task.create(task_name: 'task1', note: 'task 1 is  the first task', action: 'Inbox' )
    put :update, id: @task.id, task: { task_name: @task.task_name, note: @task.note,
action: @task.action }
    assert_redirected_to task_path
  end

  test "should destroy task" do
```

```
    @task = Task.create(task_name: 'task1', note: 'task 1 is  the first task', action: 'Inbox' )
    assert_difference('Task.count', -1) do
      delete :destroy, id: @task.id
    end
    assert_redirected_to tasks_path
  end
end
```

Ejecutamos los casos de pruebas:

ruby -I'lib:test' test/functional/tasks_controller_test.rb

Esto da el resultado siguiente:

Running tests:

.....

Finished tests in 1.728759s, 3.4707 tests/s, 4.6276 assertions/s.

6 tests, 8 assertions, 0 failures, 0 errors, 0 skips

Para hacer prueba en la vista lo que se conoce como prueba de aceptación de usuario se pueden utilizar gemas como Capybara.Las pruebas de aceptación reflejan cómo los usuarios reales interactúan con la interfaz.

Con Capybara se reproducen una a una todas las acciones del usuario. Esta gema ofrece métodos para reproducir el comportamiento de un usuario en una aplicación web.Sin embargo en este proyecto no se ha realizado prueba de aceptación de usuario hasta el momento ya que nos hemos centrado en las pruebas de unidad y las pruebas funcionales.

Capítulo 6: Conclusiones y Trabajos Futuros

6.1 Conclusiones

Una vez concluido el proyecto de especificación e implementación del sistema software para la gestión de tareas siguiendo la metodología GTD, es el momento de analizar y valorar el proceso de realización de dicho proyecto, así como los objetivos culminados.

En primer lugar, cabe destacar que este proyecto me ha permitido aplicar y profundizar los contenidos explicados a lo largo de la carrera. Los conocimientos adquirido a lo largo de la etapa docente del alumno ha sido crucial para un correcto desarrollo de la aplicación. Asignaturas como Ingeniería del Software, Metodología de Desarrollo Ágil, Bases de Datos, junto a Metodología de Programación I/II y Estructura de Datos I/II han servido para que el alumno pueda llegar a finalizar con éxito el proyecto.

La realización del sistema desde cero me ha permitido, adentrarme en cada una de las etapas del proceso de desarrollo de un sistema software, desde la obtención y definición de requisitos hasta el diseño e implementación del mismo. Gran parte del éxito de un proyecto radica en aplicar un exhaustivo control y realizar una planificación de lo que se pretende llevar a cabo. Para ello, es fundamental el uso de metodologías que marquen las pautas a seguir. En mi caso se optó por usar RUP, lo que nos ayudo a definir los artefactos y las fases que se muestran en el documento.

Algo que he aprendido es que la planificación de horas inicial (300h), se han superado con creces debido a mi corta experiencia realizando estimaciones. De cara a futuros proyectos con clientes, es mejor realizar una estimación en la que exista holgura a otra en la que no se puedan cumplir los tiempos.

Otras cosas que he aprendido con este proyecto final de grado es que es fundamental estar en continuo proceso de aprendizaje. Casi diariamente salen al mercado nuevas tecnologías que pueden ser aplicadas al producto que se ha elaborado. Sin ir más allá, los tiempos de de nuevas gemas del framework Ruby On Rails son muy cortos y, por ello, hay que estar siempre al día para intentar aportar valor con ellas a la aplicación.

En cuanto a conclusiones específica obtenidas tras la realización de la aplicación me gustaría remarcar que este proyecto ha servido como marco para aprender nuevas tecnologías de programación tales como HTML5, CSS3, Ruby y Ajax. Además, se ha obtenido un profundo conocimiento del framework Ruby On Rails del cual se está muy satisfecho. Rails provee de medios para obtener productos software orientados a entornos web con gran agilidad .

Hay que decir que se ha ido aprendiendo ruby and rails a medida que se realizaba el proyecto final de grado, por lo que hay aspecto en la arquitectura que se podría mejorar tras conocer mejor como funciona ruby and rails. Desacoplando más el diseño creando clase que provean de más servicios a la propia aplicación,etc.

Uno de los requisitos de la aplicación era que fuese portable y accesible desde cualquier dispositivo con internet, objetivo que se ha cumplido al usar el framework Bootstrap para el diseño de las vistas. El uso de esta herramienta ha permitido que las interfaces sean

intuitivas, sencillas y de fácil aprendizaje. En cuanto a los dispositivos sobre los que se han probado han sido básicamente en ordenadores personales.

Sobre la aplicación se puede decir que ha supuesto un primer paso para trabajar sobre la que podrá ser una herramienta útil para la gestión personal o empresarial de las personas.

Se han cubierto todos los objetivos específicos de los que se hablaba en la introducción de este documento desde la gestión de tareas, gestión de proyectos, interrupciones, implantación de la filosofía GTD y registro de usuario.

Me gustaría destacar que este proyecto de final de grado ha sido la culminación de un proceso de formación intenso. Este proyecto de final de grado me ha proporcionado un pequeño ejemplo de lo que podré realizar en un futuro próximo.

6.2 Trabajos Futuros

El presente trabajo de final de grado ha desembocado en el desarrollo de una aplicación útil y de fácil uso para gestionar las tareas personales o empresariales de las personas usando la filosofía GTD.

Dadas las limitaciones temporales del proyecto, hay aspectos que podrían trabajarse en un futuro, ya sea por otros alumnos interesados o por mi persona.

Se listan algunas posibilidades para potenciar y mejorar las funcionalidades ya desarrolladas:

- ✓ Capacidad de multitarea al mismo tiempo
- ✓ Desarrollo gráfico de los tiempo de interrupciones
- ✓ Mejorar aún más integración de GTD
- ✓ Mejorar la adaptabilidad en dispositivos móviles de las tablas .

Además de las mejoras reseñadas anteriormente se han seleccionado tres propuestas que son interesantes, planteándolas de forma que puedan servirle a otros alumnos para desarrollar trabajos de grado.

1. Creación de funcionalidades para la gestión de equipo.

- ✓ **Situación actual de la aplicación**. La aplicación actual ofrece soporte para usuarios individuales.
- ✓ **Introducción al problema**. Si se quiere usar esta aplicación para gestionarse para realizar un proyecto , actualmente el software no contempla ninguna capacidad para trabajar en equipo.
- ✓ **Propuesta**. Se propone crear un módulo para la aplicación que permita la gestión de equipo, desde un coordinador que gestione a los demás integrantes del grupo hasta diferentes funcionalidades de comunicación, chat, videochat, etc.
- ✓ **Modelo de negocio**. Esta mejora puede introducir una fuente de ingreso en el modelo de negocio de la aplicación final. Podría suponer nuevos objetivos principales de la aplicación que le daría más valor al producto final.

2. Integración online de editor de texto

- ✓ **Situación actual de la aplicación**. La aplicación actual no ofrece nada parecido a un editor de texto en línea.
- ✓ **Introducción al problema**. Sería muy interesante que una aplicación de gestión pueda gestionar los archivos subidos o nuevo desde una aplicación similar Microsoft office.
- ✓ **Propuesta**. Se propone la posibilidad de crear una aplicación similar a Microsoft office en donde los usuario puedan modificar los archivos subidos o generados en el programa.
- ✓ **Modelo de negocio**. Esto daría más valor al productor final y cuando la aplicación se pusiera en línea generaría posiblemente más entusiasmo entre los usuario que lo use y eso generaría más ingresos.

3. Creación de un software para móviles.

- ✓ **Situación actual de la aplicación**. La aplicación actual ofrece un software online pero no tiene un software específico para telefonía móvil que se pueda sincronizar con la aplicación web.
- ✓ **Introducción al problema**. La aplicación se podría usar en un móvil accediendo a través de un explorador, el problema es que no se vería bien dada las dimensiones de los móviles, por eso muchas aplicaciones crean sus propios software para móviles.
- ✓ **Propuesta**. Se propone generar un software específico para telefonía móvil, por ejemplo android, en donde se pudiera sincronizar la aplicación de teléfono con su cuenta de la aplicación web.
- ✓ **Modelo de negocio**. Esta software supondría otro software en el mercado, más captura de clientes, y posiblemente más ingresos.

A. Manual de Usuario y Software

El presente manual tiene como objetivo el de familiarizar al lector con las distintas funcionalidades que la aplicación ofrece, mostrándole también como llevar a cabo cada una de ellas.

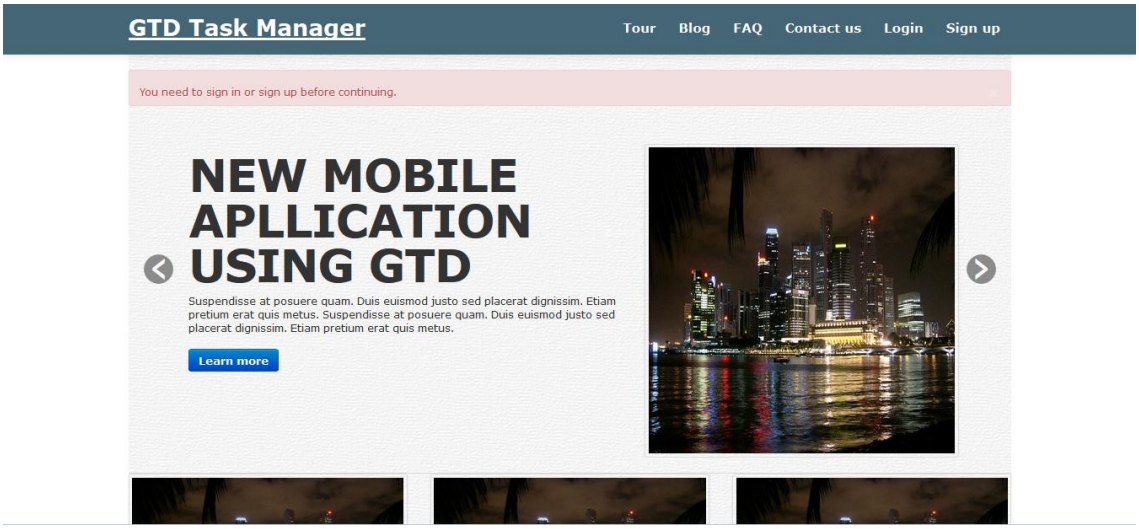
Este documento permite acceder rápidamente a la información sobre la aplicación desarrollada. Para cada uno de los módulos se dan respuesta a las posibles interacciones del usuario con el sistema.

Este manual no contiene información acerca de cómo instalar la aplicación.

A.1. Cuestiones Generales

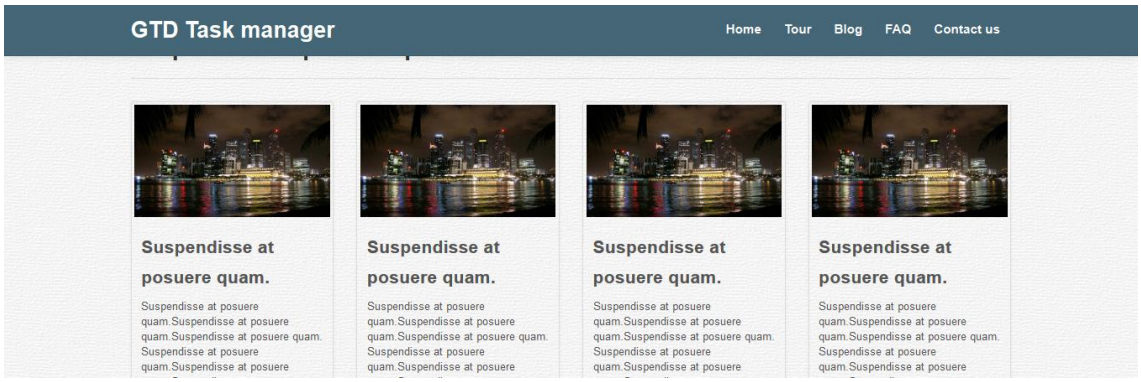
A.1.1 ¿Cómo puedo acceder a GTD Task manager?

Para poder acceder a la aplicación se debe disponer de un dispositivo con navegador web y conexión a internet. Al ser un servicio online, la aplicación dispondrá de una página de bienvenida. Desde ella se puede acceder a la fase de registro e identificación en el sistema. El objetivo es el de mostrar las capacidades del producto de forma sencilla y clara para los usuarios.

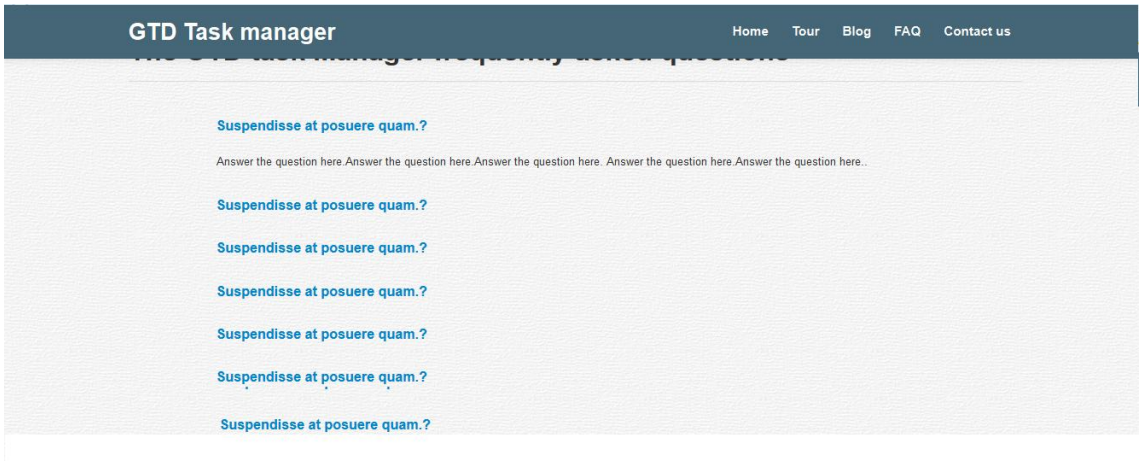


Desde esta página de inicio se pueden acceder a los siguientes enlaces:

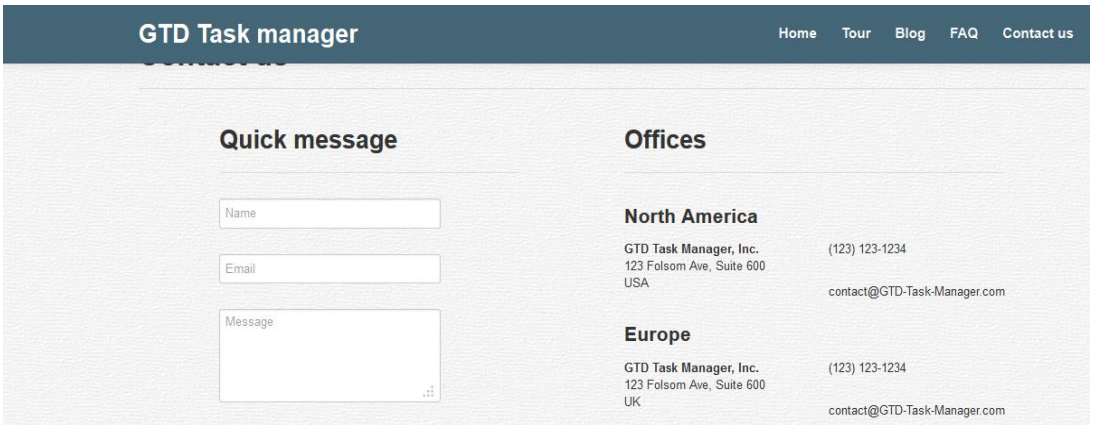
- **Tour de la aplicación**. Esta interfaz muestra las funcionalidades que se incluyen en la aplicación.



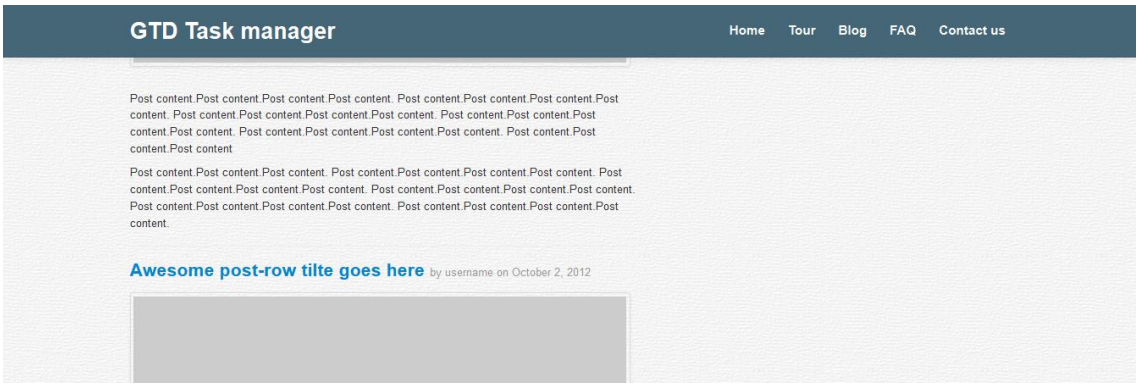
- **FAQ** .Las preguntas más frecuente que no hace los usuarios



- **Contact us**. En esta sección podrás observar en donde nos ubicamos y podrán contactar con el administrador de la aplicación.

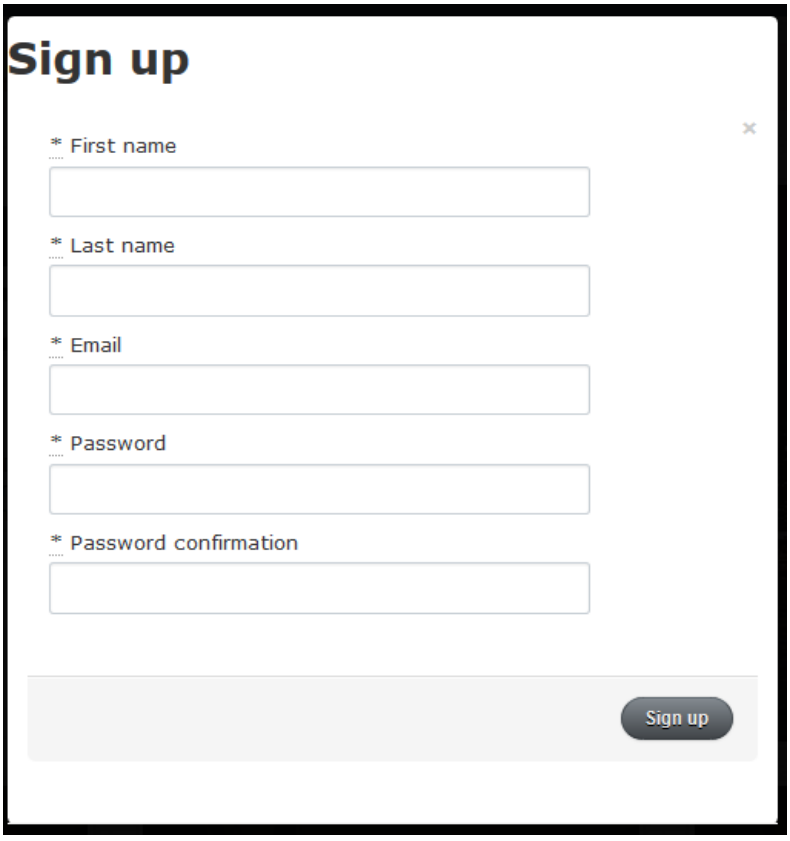


- **Blog**. Es un blog asociado a la aplicación en donde podrán ver las diferentes noticias relacionadas con la empresa y la aplicación.



A.1.2 Soy un usuario interesado en usar la aplicación, pero no estoy registrado en ella aún, ¿puedo solicitar mi registro?

Para poder acceder a la aplicación es preciso estar registrado en ella. En la página principal de la aplicación "inicio", en la barra de menú superior, existe un enlace que permitirá registrarse en el sistema. Para finalizar el proceso se ha de cumplimentar un formulario de registro insertando los campos que se muestra en la imagen siguiente.

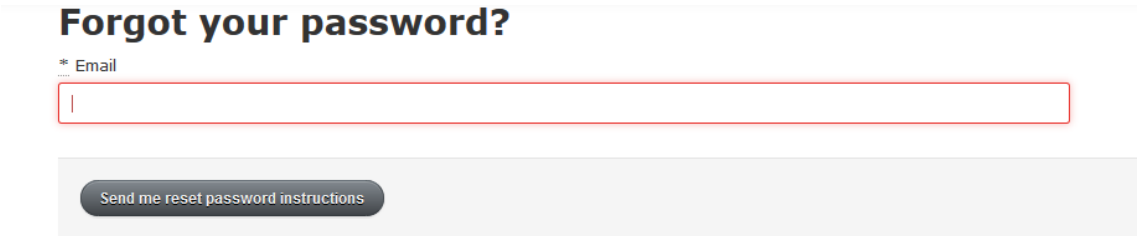


A.1.3 Una vez registrado, ¿cómo puedo identificarme para empezar a usar las funcionalidades?

Desde la pantalla de inicio se muestra un enlace para identificarse. Se redireccionar al usuario a una interfaz donde debe insertar sus credenciales (que deben coincidir con los que se ingresaron en la fase de registro) para poder comenzar a usar el software.

A.1.4 No recuerdo mis datos de acceso a la aplicación, ¿puedo recuperarlos?

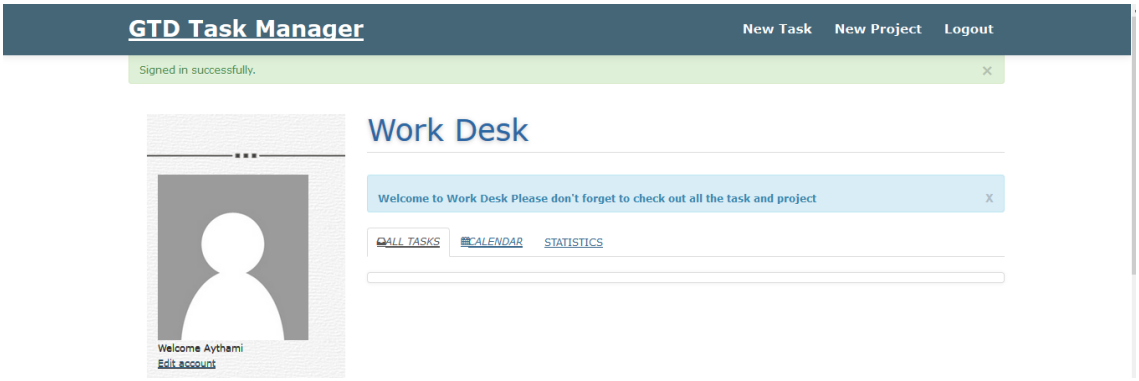
Si el usuario no recuerda su contraseña a la hora de identificarse, puede pedir al sistema que le envíe un email indicándole una nueva contraseña. La forma de acceder a esta funcionalidad es a través de las interfaces de registro e identificación. Existe un enlace de recordar contraseña tras el cual se genera un formulario donde se pide al usuario que inserte la dirección de correo electrónico que usó durante la fase de registro, siendo ahí donde se le envíe la nueva contraseña.



A.2. Sección "Work Desk" y Configuración de Cuenta

A.2.1 Tras acceder a la aplicación, ¿qué me encuentro?

La aplicación cuenta con una interfaz llamada "work desk" en donde el usuario podrá trabajar todas las funcionalidades del software.



Esta interfaz está dividida en diferentes partes:

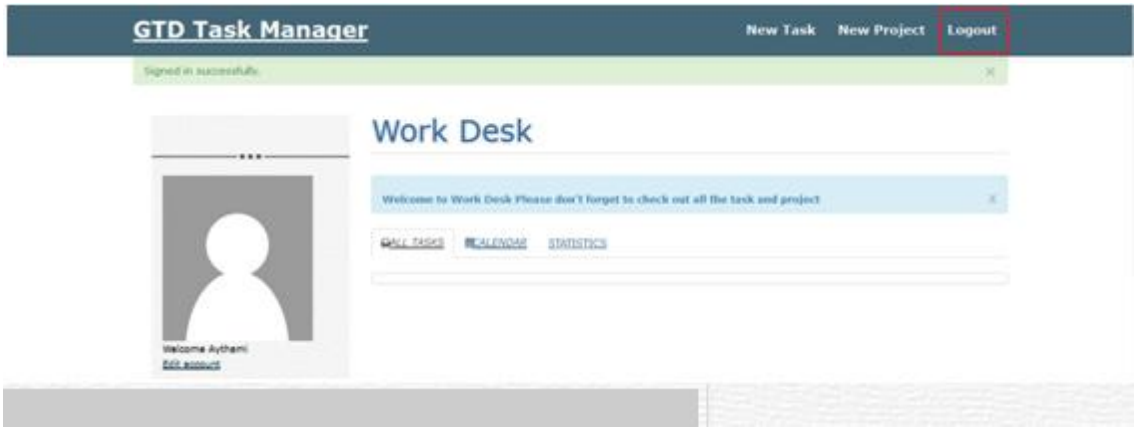
- **drag and drop**. En donde podemos ver los diferentes proyectos y tareas que vamos creando.
- **Calendario**. En donde podemos visualizar las tareas a lo largo del mes.
- **Estadísticas**. En esta sección de los tiempo productivos e improductivos de nuestros trabajos.

A.2.2 He conseguido entrar ya, ¿cómo me muevo por la aplicación?

La estructura visual de la aplicación está dividida en tres zonas. La primera de ellas se compone de una barra superior de acceso rápido la segunda es una barra de navegación lateral, donde aparecen diferentes funcionalidades de la metodología GTD para poder trabajar tanto las tareas como los proyectos que nos creemos.
La tercera es el bloque central de contenido, área destinada para presentar toda la información que los usuarios necesitan.
Todas las interfaces del software disponen de una barra de rutas que indica en qué punto de la aplicación nos encontramos. Por ejemplo, en el caso de encontrarnos en la página de calendario.

A.2.3 He finalizado mi actividad en GTD Task Manager , ¿cómo cierro mi sesión?

En la barra de navegación superior se encuentra siempre disponible un enlace para cerrar sesión. Cuando el usuario pulsa sobre él, finaliza la sesión en el sistema y se le redirecciona a la página principal de inicio.

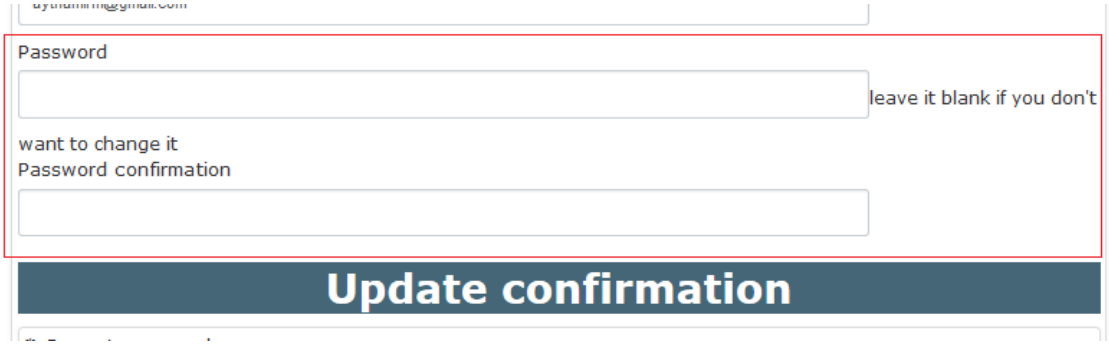


A.2.4 ¿Existe la posibilidad de modificar los datos ingresados durante la fase de registro?

Con el fin de que el usuario disponga de sus datos lo más actualizados posibles, se dispone de una interfaz para la configuración del perfil. Se puede acceder a ella a través de la barra de navegación lateral usando el enlace de configuración de cuenta.

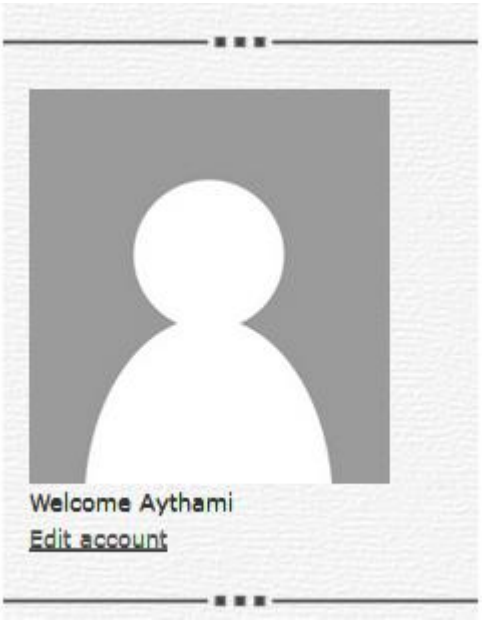
A.2.5 ¿Puedo cambiar la contraseña de acceso al sistema?

Por supuesto que sí. Es más, recomendamos que cada cierto tiempo el usuario la modifique. Puede modificar tantas veces como le sea necesario la contraseña de acceso. Se recomienda que se haga al menos una vez al mes para evitar problemas de seguridad. Para hacerlo, desde la interfaz de configuración de cuenta.



A.2.6 ¿Para qué sirve la fotografía del avatar?

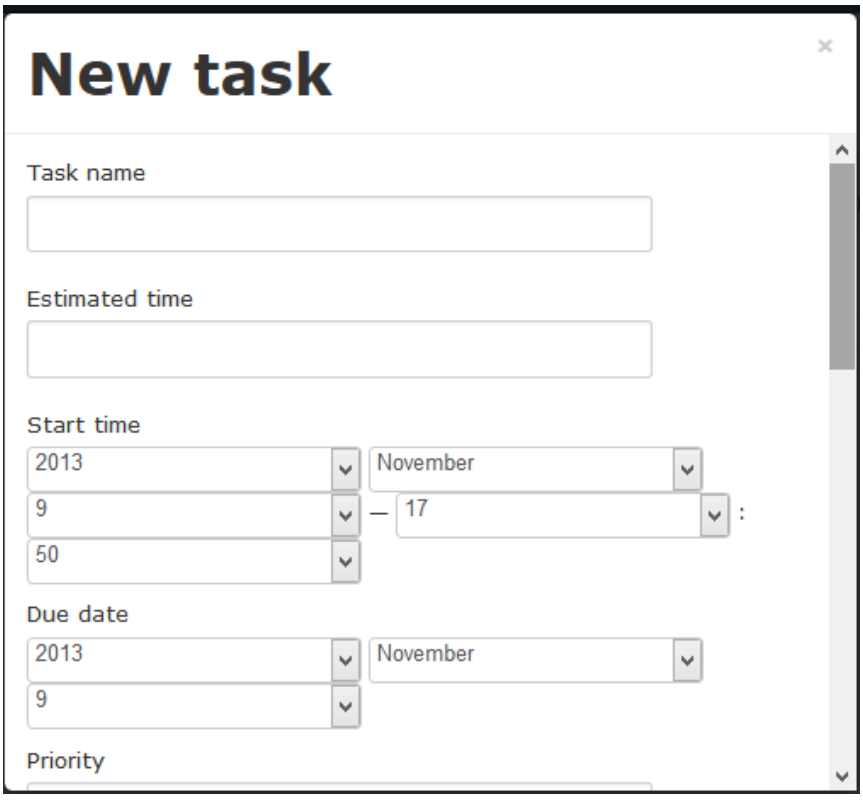
Para personalizar la interfaz, se le ofrece a los usuarios la posibilidad de adjuntar una foto que se muestre en la barra de navegación lateral. Para poder subir la imagen se debe hacer desde configuración del cuenta. Pulsando sobre el botón de buscar, se selecciona la fotografía (no puede ser de tipos que no sean imágenes ni tampoco puede superar los 2mb).



A.3. Gestión de Tareas

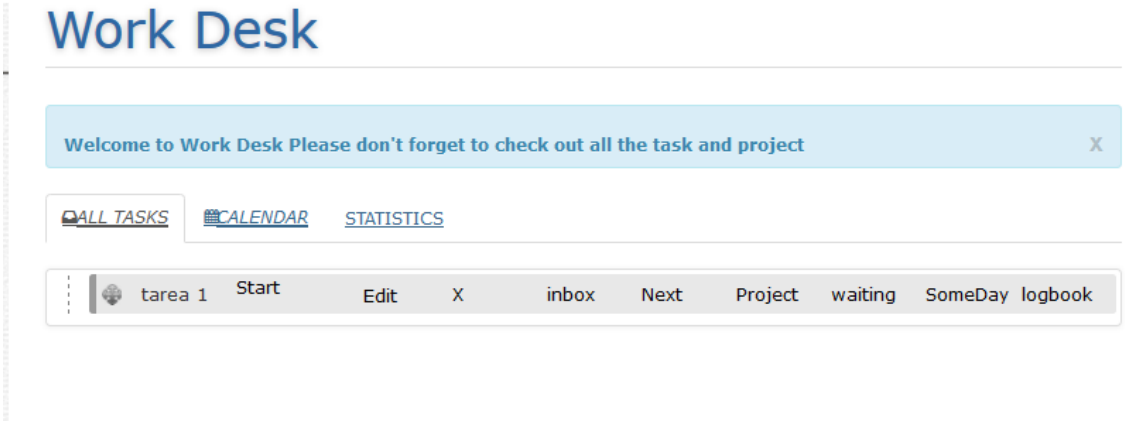
A.3.1 ¿Cómo agrego las tareas en la aplicación?

En la barra de navegación superior se encuentra siempre disponible un enlace para crear una tarea. Cuando el usuario pulsa sobre él, se abre un ventana modal en la cual el usuario puede insertar los datos referentes a la tarea.



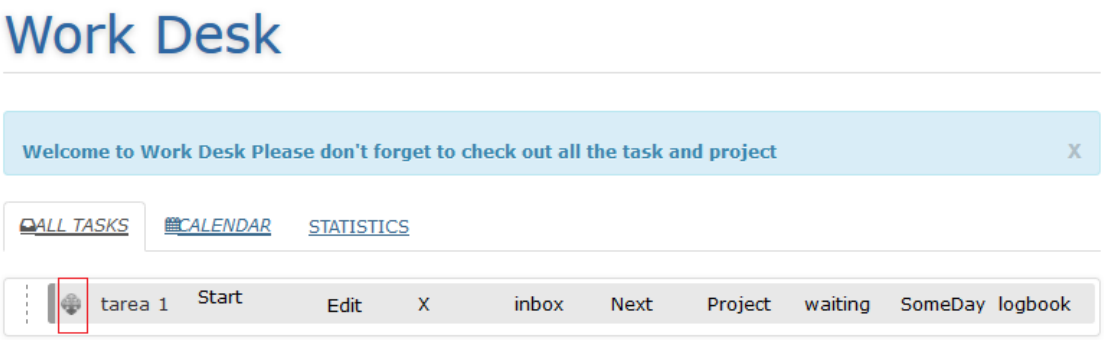
A.3.2 Una vez añadidos, ¿cómo se qué tareas se han insertado?

Saber que tarea se ha insertado es muy fácil, en la primera pestaña hay un "drag and drop" en donde se puede ver las tarea que se van creando. También se puede ver las tareas que se han creado a través de un calendario .



A.3.3 ¿Puede ordenarse y filtrarse la lista de tareas?

Como se utiliza el "drag and drop" podemos mover las tareas y ordenarla como nos gustarían. Al mismo tiempo en la barra de navegación lateral hay mucho filtros distintos de búsquedas.



A.3.4 ¿Y si quiero realizar alguna búsqueda de un tarea en particular?

En la barra de navegación lateral se puede buscar una tarea en particular a través de su nombre.

A.3.5 ¿En qué consiste la Filtrados de GTD?

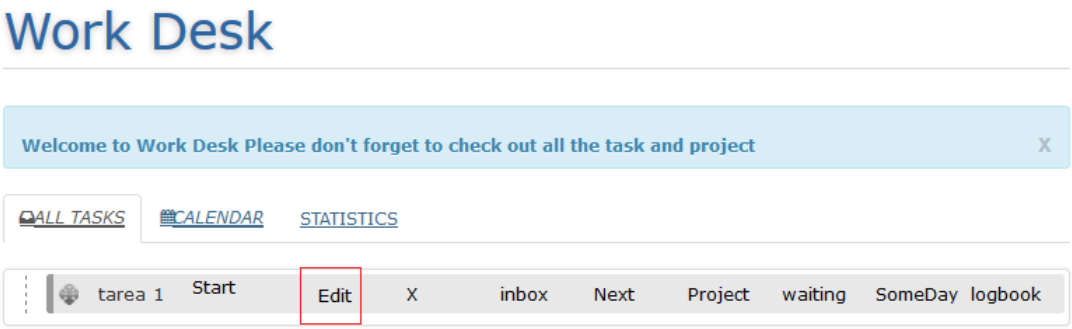
- **Inbox:** Aquí se incluye todas las tareas y proyectos cuando se crean predefinida.
- **Próximo**: Incluimos las tareas que debemos realizar cuanto antes.
- **Proyecto**: Incluimos las tareas que implicar realizar más de una acción.
- **En espera**: Son tareas que necesitan de la interrupción de otra persona para ser realizada.
- **Algún día**: Son tareas que debemos hacer pero no son prioritarias.

A.3.6 ¿Se puede filtrar por el estado en que se encuentra una tarea?

Sí. Una tarea pude estar en diferentes estados de ejecución, pude haber empezado la tarea , estar en ejecución , haber sido interrumpida o haber terminado. Todas estos filtros son posibles gracias a una opción que se encuentra en el menú lateral de la aplicación.

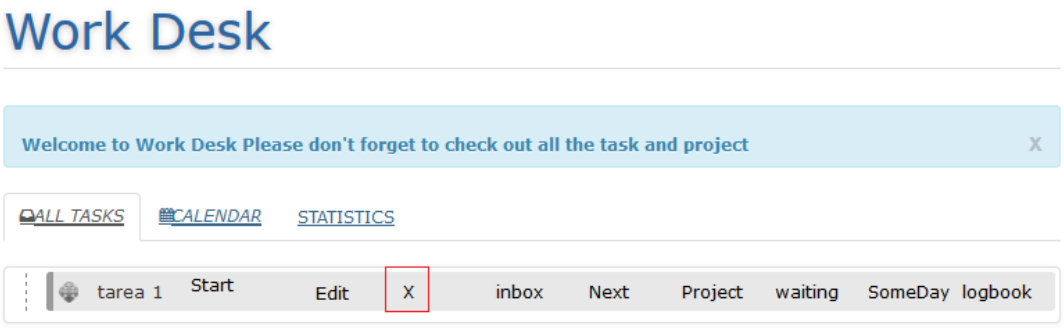
A.3.7 ¿Se puede modificar la información de un tarea?

Cada tarea creada se pude modificar sus datos , en el drag and drop hay una opción para poder editar la tarea.



A.3.8 ¿Y si deseo eliminarlo de la aplicación?

Cada tarea creada se pude eliminar , en el drag and drop hay una opción para poder eliminar la tarea.

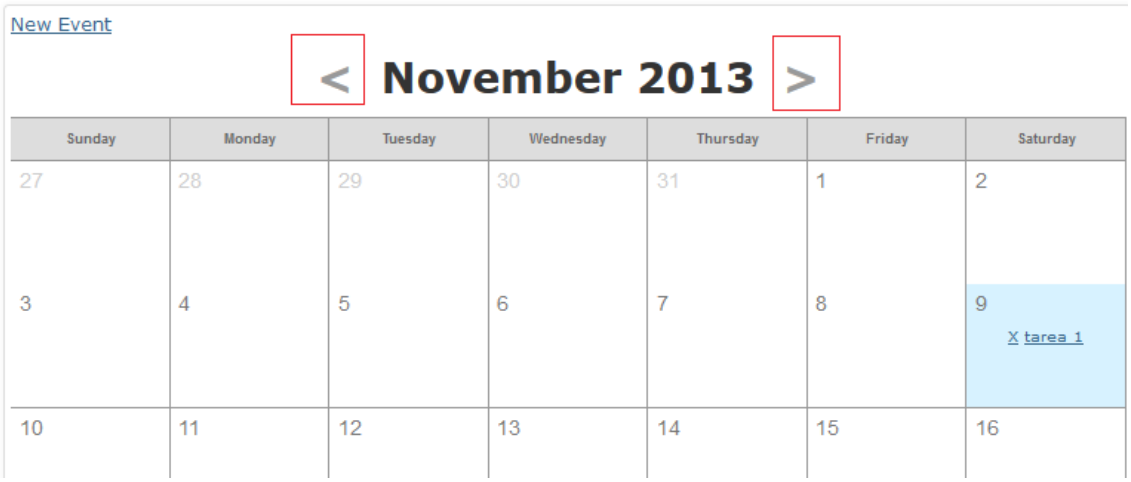


A.3.9 ¿Podemos filtrar las tareas a través de un calendario?

En la segunda pestaña del contenedor de datos central se encuentra un calendario el cual nos da la posibilidad de ver las diferentes tareas que se encuentra para un día en particular.

A.3.10 Una vez que se ha mostrado el calendario, ¿podemos desplazarnos en el tiempo?

Sí. podemos movernos entre los diferentes meses que compone el año con las barras de navegación que se encuentra a ambos lados del mes actual del calendario.



A.3.11¿podemos crear nuevas tareas desde la sección del calendario?

Sí. Existe la posibilidad de crear un nuevo evento para el calendario. Esta opción se encuentra encima del calendario.

A. 4. Gestión de Proyecto

A. 4.1 ¿En qué consiste la gestión de proyectos?

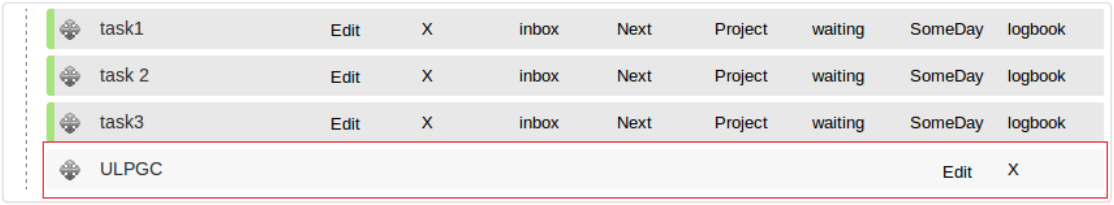
La gestión de proyecto consiste en poder crear proyectos, cuyo objetivo será hacer agrupaciones de tareas que contenga una misma afinidad.

A.4.2 ¿Cómo pueden crear un proyecto?

En la barra de navegación superior se encuentra siempre disponible un enlace para crear un proyecto. Cuando el usuario pulsa sobre él, se abre un ventana modal en la cual el usuario puede insertar los datos referentes al proyecto.

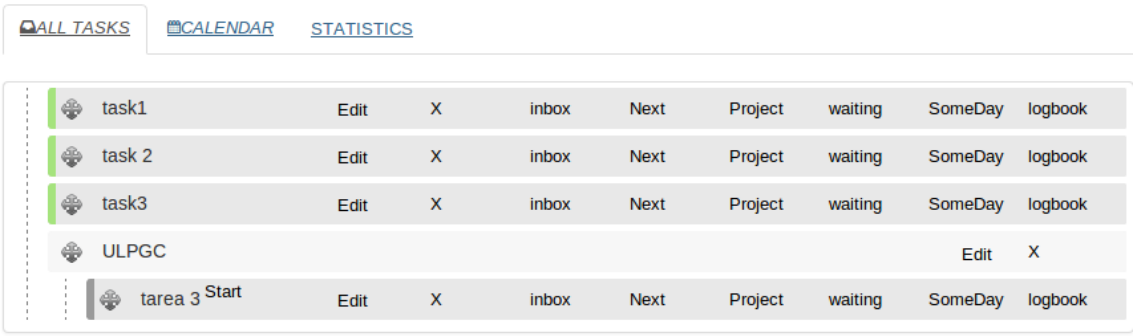
A. 4.3 ¿Cómo pueden verse cada uno de los proyectos creados ?

Saber qué proyecto se ha insertado es muy fácil, en la primera pestaña hay un "drag and drop" en donde se puede ver los proyectos que se han creando.



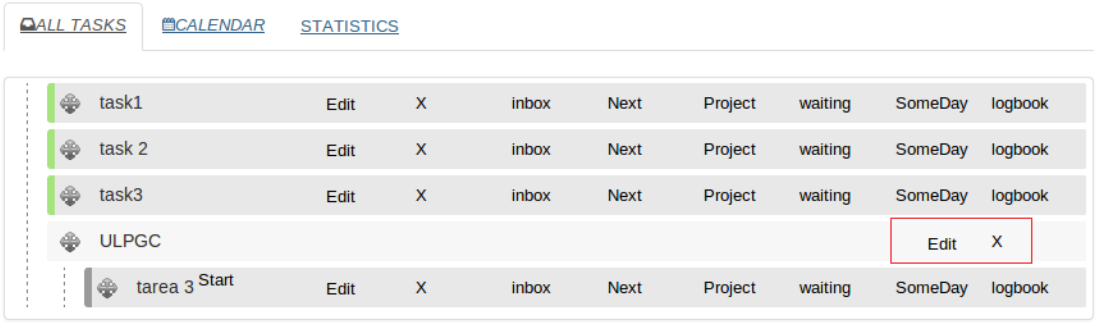
A.4.4 ¿Qué información puede aportar un proyecto?

Un proyecto simplemente es un separador de diferentes tareas que tienen en común alguna afinidad.



A.4.5 Una vez que se ha insertado un proyecto, ¿cómo se podría editar o eliminar?

Cada proyecto creado se pude modificar sus datos o eliminar, para ello simplemente te diriges al drag and drop y en cada proyecto hay una opción para poder editar el proyecto, y otra opción para poder eliminar el proyecto.



A.5 Gestión de GTD

A.5.1 ¿En qué consiste la gestión de GTD?

Es una metodología para gestionar eficazmente las tareas profesionales o personales. Dicha metodología según varios estudios realizados aumenta la productividad y reduce el estrés laboral.

A.5.2. ¿Qué significa inbox ?

Aquí se incluye todas las tareas y proyectos cuando se crean predefinida.

A.5.3 ¿Qué significa Next ?

Incluimos las tareas que debemos realizar cuanto antes.

A.5.4 ¿Qué significa Project ?

Incluimos las tareas que implicar realizar más de una acción.

A.5.5 ¿Qué significa Waiting ?

Son tareas que necesitan de la interrupción de otra persona para ser realizada.

A.5.6 ¿Qué significa Someday ?

Son tareas que debemos hacer pero no son prioritarias.

A.5.7 ¿Existe aviso de sistema para recordatorio de revisión?

Sí, cada 24 horas desde su ingreso en la aplicación, los usuarios serán avisado a través de un correo electrónico para que revise sus tareas, de que elimine las tareas realizadas, reorganice las tareas cuando haya cambiado nuestra prioridad, etc.

A.6 Gestión de Control de Tiempo

A.6.1 ¿En qué consiste la gestión de Control de tiempo?

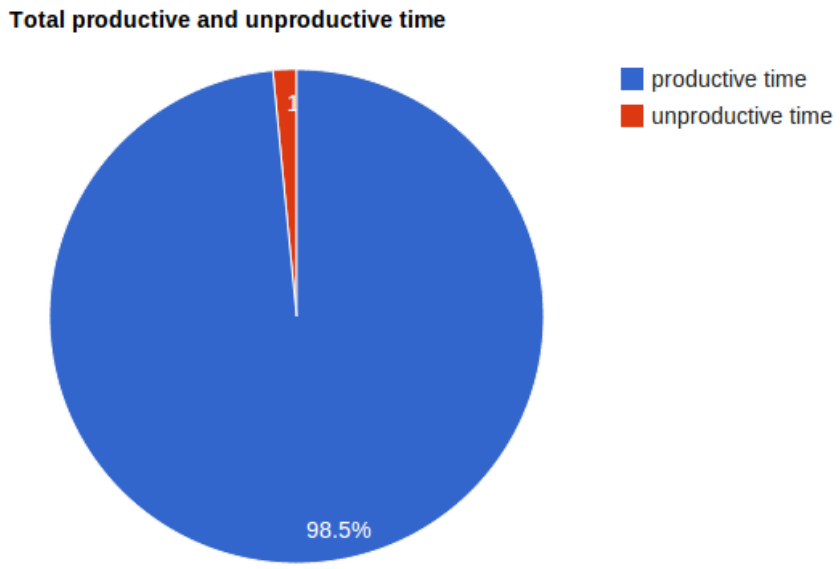
Consiste en gestionar el tiempo productivo e improductivo que se produce al hacer nuestra tareas.

A.6.2 ¿Podemos saber el tiempo productivo ?

Sí, en la tercera pestaña de la parte central de datos encontrará una sección en donde podrás observar estadística de nuestro tiempo productivo.

A.6.3 ¿Podemos saber el tiempo improductivo ?

Sí, en la tercera pestaña de la parte central de datos encontrará una sección en donde podrás observar estadística de nuestro tiempo improductivo.



A.6.4 ¿Cómo obtendremos gráficas sobre los datos de tiempo ?

Sí, en la tercera pestaña de la parte central de datos encontrará una sección en donde podrás observar estadística. En esta sección se creará las gráficas con los datos almacenados. de forma automática cada vez que el tiempo productivo o improductivo cambie.

A.6.5 ¿Podemos controlar una tarea ?

Sí, En el "drag and drop" podemos controlar el estado de ejecución que esta una tarea, puede haber empezado, o está en pausa o ha finalizado.

También habrá una gestión de colores para saber el estado en que está la tarea, cuando el color es gris significa que la tarea no ha empezado, cuando es verde la tarea ha finalizado, cuando es amarillo la tarea está en marcha y cuando es rojo la tarea esta interrumpida.

A.6.6 ¿Podemos controlar una interrupción ?

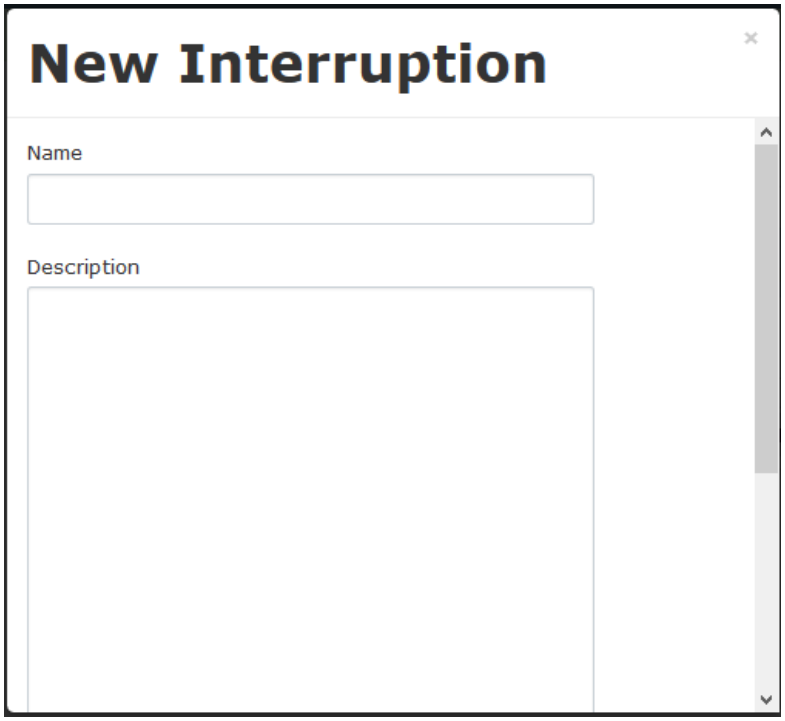
Sí, En el "drag and drop" podemos controlar el estado la creación de una interrupción y la finalización de la misma que supone el retorno del tiempo productivo de una tarea. Con la opción "pause" la cual si la apretamos generará una interrupción y con la opción "resumen" terminamos la interrupción en marcha.

A.6.7 ¿En qué consiste la gestión de interrupciones?

La gestión de interrupciones consiste en controlar las interrupciones que se produce en un tarea, para poder controlar los tiempos improductivos producidos en el trabajo.

A.6.8¿Cómo se añaden interrupciones a la aplicación?

Cada tarea tendrá una funcionalidad para crear interrupciones , para ello simplemente te diriges al drag and drop y encontrará la opción de interrupción de tareas.



B. Competencias

En este capítulo se describirá la manera en la que se ha cubierto las competencias exigidas al Trabajo de Fin de Grado.

B.1. CII01

Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

Esta competencia ha sido desarrollada en los capítulos de¨"**Diseño e implementación**". En dichos capítulos mostramos las causas y los hechos derivados de las decisiones de diseño tomadas.

B.2. CII02

Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.

En el capítulo " **Planificación del trabajo**" se relata el sistema de organización usado para alcanzar los objetivos propuestos.

La valoración del impacto social y económico de este TFG queda reflejada por su parte en el capítulo "**Introducción**" del apartado introductorio de este trabajo final de grado.

B.3. CII04

Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.

El pliego de condiciones se divide en dos apartados perfectamente diferenciados:

- ✓ **Especificaciones de materiales y equipos**: Estas especificaciones se realiza en el capítulo 4 "herramientas y tecnologías".
- ✓ **Especificaciones de ejecución**: en este apartado del Pliego se hace constar cómo será realizado el proyecto, es decir, su proceso de fabricación o construcción a partir de los materiales que serán utilizados. Esto se realiza en el capítulo 5 "Desarrollo de la aplicación" donde se encarga de todas esta especificaciones.

Por lo tanto, con la suma del capítulo 4 y el capítulo 5 queda cubierta esta competencias especificando las condiciones asociadas a este Trabajo de Fin de Grado.

B.4. CII18

Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.

El cumplimiento de esta competencia queda descrita en el capítulo de "**Normativa y Legislación**"

B.5. TFG01

El ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la ingeniería en informática de naturaleza profesional en el que se sinteticen e integre las competencias adquiridas en la enseñanza.

Esta competencia se realizará en la presentación y defensa ante un tribunal universitario. También desde el punto de vista académico, este TFG se orientó al aprovechamiento de los conocimientos adquiridos durante la carrera como son:

- ✓ Algoritmos, programación y estructuras de datos
- ✓ Metodología de desarrollo ágil
- ✓ Seguridad informática
- ✓ Desarrollo programático orientado al navegador

C. Normativa y Legislación

C.1. Ley de Protección de Datos

En este trabajo final de grado hay que tener en cuenta una serie de requisitos legales debido a que existirán datos de carácter personal. El sistema debe cumplir con el Reglamento de Seguridad del Real Decreto994/1999 de la Ley Orgánica de Protección de Datos Personales (LOPD).

La Ley orgánica de Protección de Datos Personales tiene por objetivo el garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar. Dicha ley será aplicada a los datos de carácter personal registrados en soporte físico que los haga susceptibles de tratamiento, y a toda modalidad de uso posterior de estos datos por los sectores público y privado.

Si la LOPD no estuviera en vigor, nuestros datos estarían en manos de la especulación. Un ejemplo claro lo vemos cuando nos registramos a una página web, la cual normalmente suele pedir tu nombre, tus apellidos, dirección, correo electrónico, gustos personales, etc. Sin duda estos datos deberían estar protegidos, sin embargo hoy en día es muy común por ejemplo que tu correo electrónico se proporcione sin tu consentimiento a otra empresa, lo cual le da vía libre para enviar correo basura de forma indiscriminada, es por ello que es preciso denunciar este tipo de acciones que van en contra de la LOPD.

C. 2. Licencias

C.2.1 GNU GPL

Conocida como licencia publica general (GNU Generical Public License), se trata de una licencia software elaborada por la fundación del software libre en el año 1989.

Esta licencia nace de la necesidad de proteger la distribución, modificación y uso de software evitando la apropiación indebida del código que pueda dar lugar a las restricciones y libertades de los usuarios. La licencia GPL permite que el software se distribuya y modifique libremente siempre y cuando no se altere la licencia original.

Por otra parte, cuando surja la necesidad de incluir código sujeto a otras licencias libres, el resultado final deberá estar bajo la licencia GPL.

C.2.2 LGPL

Licencia similar a la GPL que permite que una biblioteca determinada pueda ser usada tanto por programas libres como por programas no libres. Con lo cual, se da la posibilidad de poder distribuir programas bajo cualquier licencia con la salvedad de aquellos trabajos derivados, cuyos términos de modificación los establece la licencia por la cual se rigen.

C.2.3. Creative Commons

Las licencias Creative commons ó CC tienen su origen en la licencia GNU GPL, cuyo objetivo es el de ofrecer al autor de una obra un mecanismo sencillo y eficaz a la hora de

establecer una serie de condiciones asociadas a la obra. Las condiciones que se pueden imponer son las siguientes:

- ✓ **Reconocimiento** (Attribution):En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.

- ✓ **No Comercial** (Non commercial):La explotación de la obra queda limitada usos no comerciales.

- ✓ **Sin obras derivadas** (No Derivate Works):La autorización para explotar la obra no incluye la transformación para crear una obra derivada.

- ✓ **Compartir Igual** (Share alike):La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

D. Medologías Ágiles

Sobre la década de los 90, surgió un enfoque que iba contra toda creencia hasta entonce conocida. Surgió que mediante procesos altamente definidos se iba a logra obtener software en tiempo, costo y con la calidad requerida. Dicho enfoque fue planteado por J. Martin en 1991 dándose a conocer como RAD (Rapid Application Development). La metodología consistía en un entorno de desarrollo altamente productivo, en el que participaban grupos pequeños de programadores utilizando herramientas que generaban código en forma automática tomando como entradas sintaxis de alto nivel. En general, se considera que este fue uno de los primeros hitos en pos de la agilidad en los procesos de desarrollo.

Pero no fue hasta mayo de 1997 cuando Kent Beck tuvo éxito en un proyecto donde había desarrollado una metodología que luego se llamaría XP iniciando el movimiento de metodologías ágiles . Sin embargo, ninguna de estas metodología creadas hasta la fecha contaban con una aprobación, pues se le consideraba por muchos desarrolladores como meramente intuitiva.

No fue hasta la reunión celebrada en febrero de 2001 en Utah-EEUU, cuando nace el término "*ágil*" aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas. Varias de las denominadas metodologías ágiles ya estaban siendo utilizadas con éxito en proyectos reales, pero les faltaba una mayor difusión y reconocimiento.

Tras esta reunión se creó ***The Agile Alliance,*** una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida es fue el Manifiesto Ágil, un documento que resume la filosofía *"ágil".*

D.1. El Manifiesto Ágil

El manifiesto ágil se compone de 4 valores y 12 principios. Puede sintetizarse este concepto, comprendiendo los cuatro valores del Manifiesto:

- **Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas**. La gente es el principal factor de éxito de un proyecto software. Si se sigue un buen proceso de desarrollo, pero el equipo falla, el éxito no está asegurado; sin embargo, si el equipo funciona, es más fácil conseguir el objetivo final, aunque no se tenga un proceso bien definido. No se necesitan desarrolladores brillantes, sino desarrolladores que se adapten bien al trabajo en equipo. Así mismo, las herramientas (compiladores, depuradores, control de versiones, etc.) son importantes para mejorar el rendimiento del equipo, pero el disponer más recursos que los estrictamente necesarios también puede afectar negativamente. En resumen, es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el

equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

- **Desarrollar software que funciona más que conseguir una buena documentación**. Aunque se parte de la base de que el software sin documentación es un desastre, la regla a seguir es “no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante”. Estos documentos deben ser cortos y centrarse en lo fundamental. Si una vez iniciado el proyecto, un nuevo miembro se incorpora al equipo de desarrollo, se considera que los dos elementos que más le van a servir para ponerse al día son: el propio código y la interacción con el equipo.
- **La colaboración con el cliente más que la negociación de un contrato**. Las características particulares del desarrollo de software hace que muchos proyectos hayan fracasado por intentar cumplir unos plazos y unos costes preestablecidos al inicio del mismo, según los requisitos que el cliente manifestaba en ese momento. Por ello, se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.
- **Responder a los cambios más que seguir estrictamente un plan**. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta puesto que hay muchas variables en juego, debe ser flexible para poder adaptarse a los cambios que puedan surgir. Una buena estrategia es hacer planificaciones detalladas para unas pocas semanas y planificaciones mucho más abiertas para unos pocos meses.

Los valores anteriores inspiran los doce principios del manifiesto. Estos principios son las características que diferencian un proceso ágil de uno tradicional. Los dos primeros son generales y resumen gran parte del espíritu ágil. Son:

I*. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor*. Un proceso es ágil si a las pocas semanas de empezar ya entrega software que funcione aunque sea rudimentario. El cliente decide si pone en marcha dicho software con la funcionalidad que ahora le proporciona o simplemente lo revisa e informa de posibles cambios a realizar.

II. *Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva*. Este principio es una actitud que deben adoptar los miembros del equipo de desarrollo. Los cambios en los requisitos deben verse como algo positivo. Les va a permitir aprender más, a la vez que logran una mayor satisfacción del cliente. Este principio implica además que la estructura del software debe ser flexible para poder incorporar los cambios sin demasiado coste añadido. El paradigma orientado a objetos puede ayudar a conseguir esta flexibilidad.

Luego existen una serie de principios que tienen que ver directamente con el proceso de desarrollo de software a seguir.

III*. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas*. Las entregas al cliente se insiste en que sean software, no planificaciones, ni documentación de análisis o de diseño.

IV*. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto*. El proceso de desarrollo necesita ser guiado por el cliente, por lo que la interacción con el equipo es muy frecuente.

V*. Construir el proyecto en torno a individuos motivados*. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo. La gente es el principal factor de éxito, todo los demás (proceso, entorno, gestión, etc.) queda en segundo plano. Si cualquiera de ellos tiene un efecto negativo sobre los individuos debe ser cambiado.

VI*. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo*. Los miembros de equipo deben hablar entre ellos, éste es el principal modo de comunicación. Se pueden crear documentos pero no todo estará en ellos, no es lo que el equipo espera.

VII. *El software que funciona es la medida principal de progreso*. El estado de un proyecto no viene dado por la documentación generada o la fase en la que se encuentre, sino por el código generado y en funcionamiento. Por ejemplo, un proyecto se encuentra al 50% si el 50% de los requisitos ya están en funcionamiento.

VIII*. Los procesos ágiles promueven un desarrollo sostenible*. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante. No se trata de desarrollar lo más rápido posible, sino de mantener el ritmo de desarrollo durante toda la duración del proyecto, asegurando en todo momento que la calidad de lo producido es máxima.

Finalmente los últimos principios están más directamente relacionados con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo.

IX. *La atención continua a la calidad técnica y al buen diseño mejora la agilidad*. Producir código claro y robusto es la clave para avanzar más rápidamente en el proyecto.

X*. La simplicidad es esencial*. Tomar los caminos más simples que sean consistentes con los objetivos perseguidos. Si el código producido es simple y de alta calidad será más sencillo adaptarlo a los cambios que puedan surgir.

XI. *Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos*. Todo el equipo es informado de las responsabilidades y éstas recaen sobre todos sus miembros. Es el propio equipo el que decide la mejor forma de organizarse, de acuerdo a los objetivos que se persigan.

XII*. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento*. Puesto que el entorno está cambiando continuamente, el equipo también debe ajustarse al nuevo escenario de forma continua. Puede cambiar su organización, sus reglas, sus convenciones, sus relaciones, etc., para seguir siendo ágil.

Los firmantes de los valores y principios de este Manifiesto son: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland y Dave Thomas.

D.2 Revisión de Metodologías

Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito el manifiesto ágil y coinciden con los principios enunciados anteriormente, cada metodología tiene características propias y hace hincapié en algunos aspectos más específicos.

A continuación se resumen dichas metodologías ágiles:

- **SCRUM**. Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.
- **Crystal Methodologies**. Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo (de ellas depende el éxito del proyecto) y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).
- **Dynamic Systems Development Method** (**DSDM**) . Define el marco para desarrollar un proceso de producción de software. Nace en 1994 con el objetivo el objetivo de crear una metodología RAD unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases.
- **Adaptive Software Development** (**ASD**) . Su impulsor es Jim Highsmith. Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.
- **Feature-Driven Development** (**FDD**) . Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Sus impulsores son Jeff De Luca y Peter Coad.
- **Lean Development** (**LD**). Definida por Bob Charette's a partir de su experiencia en proyectos con la industria japonesa del automóvil en los años 80 y utilizada en numerosos proyectos de telecomunicaciones en Europa. En LD, los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades que mejoren la productividad del cliente. Su principal característica es introducir un mecanismo para implementar dichos cambios.

A continuación se compara las distintas aproximaciones ágiles en base a tres parámetros: vista del sistema como algo cambiante, tener en cuenta la colaboración entre los miembros del equipo y características más específicas de la propia metodología como son simplicidad, excelencia técnica, resultados, adaptabilidad, etc.

También incorpora como referencia no ágil el Capability Madurity Model (CMM).

	CMM	ASD	Crystal	DSDM	FDD	LD	Scrum	XP
Sistema como algo cambiante	1	5	4	3	3	4	5	5
Colaboración	2	5	5	4	4	4	5	5
Resultados	2	5	5	4	4	4	5	5
Simplicidad	1	4	4	3	5	3	5	5
Adaptabilidad	2	5	5	3	3	4	4	3
Excelencia técnica	4	3	3	4	4	4	3	4
Práctica de colaboración	2	5	5	4	3	3	4	5
Media total	**2**	**4.6**	**4.6**	**3.6**	**3.7**	**3**	**4.4**	**4.6**

Ranking de "agilidad" (Los valores más altos representa mayor agilidad)

D.3 Comparación de Metodologías Ágiles y Tradicionales

Vamos a enumerar las principales diferencias de una Metodología Ágil respecto de las Metodologías Tradicionales (llamadas peyorativamente “no ágiles” o “pesadas”).

La siguiente tabla recoge estas diferencias que no se refieren sólo al proceso en sí, sino también al contexto de equipo y organización que es más favorable a cada uno de estas filosofías de procesos de desarrollo de software.

Metodología Ágil	Metodología tradicional
Pocos artefactos. El modelado es precindible, modelos desechables.	Más artefactos.El modelado es esencial,mantenimiento de modelos
Pocos roles,más genericos y flexibles	Más roles,más específicos
No existe un contrato tradicional,debe ser bastante flexible	Existe un contrato prefijado
Cliente es parte del equipo de desarrollo	El cliente interactua con el equipo de desarrollo mediante reuniones
Orientadas a proyectos pequeños.Corta duración(o entregas frecuentes),equipos pequeños(<10 integrantes) y trabajando en el mismo sitio	Aplicada a proyectos de cualquier tamaño pero suele ser usada y más efectivas en proyectos grandes y con equipos posiblemente dispersos
La arquitectura se va difiniendo y mejorando a lo largo del proyecto	Se promueve que la arquitectura se defina tempranamente en el proyecto
Énfasis en los aspectos humanos: el individuo y el trabajo en equipo	Énfasis en la definición del proceso:roles actividades y artefactos
Basadas en la heurísticas provenientes de prácticas de produción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Se esperan cambios durante el proyecto	Se espera que no ocurran cambios de gran impacto durante el proyecto

Bibliografía

[1] Documentación oficial bootstrap (en línea).**http://twitter.github.io/bootstrap/.**

[2] Documentación oficial jquery (en línea).**http://api.jquery.com.**

[3] Documentación oficial ruby (en línea).**http://www.ruby-doc.org/core/.**

[4] Documentación oficial ruby on rails (en línea).**http://api.rubyonrails.org.**

[5] Documentación oficial rvm (en línea).**https://rvm.io.**

[6] Metodologías Ágiles (en línea).**http://www.informatizate.net.**

[7] Metodologías Ágiles spain (en línea).**http://www.agile-spain.com.**

[8] Documentación startUML(en línea). **http://staruml.waxoo.com**

[9] Documentación Balsamiq (en línea). **http://www.glidea.com/blog/balsamiq-mockups-una-herramienta-para-realizar-wireframes**

[10] Documentación Photoshop (en línea).**http://www.alegsa.com.ar/Dic/photoshop.php / http://es.wikipedia.org/wiki/Adobe_Photoshop**

[11] Documentación para pruebas (en línea).**http://pruebasdelsoftware.wordpress.com/**

[12] Richardson, L. y Ruby, S.RESTful Web Services. O'Reilly Media, 2007.

[13] Jacobson, I.; Booch, G. y Rumbaugh, J.El Proceso Unificado de Desarrollo de Software /Ivar Jacobson, Grady Booch, James Rumbaugh. 84-7829-036-2. Addison Wesley, 2008

[14] David Allen. Getting Things Done: The Art of Stress-Free Productivity .978-0-14-200028-1. Penguin , 2007

[15] Watts S.Humphrey.Introducción al Proceso Software Personal.84-7829-052-4. Addison Wesley, 1997