Proyecto fin de carrera de la Escuela de Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria.

Titulo del proyecto: DESARROLLO DE UN PROTOTIPO DE COMPONENTE WEB PARA LA VISUALIZACIÓN E INTERACCIÓN CON MAPAS DE FORMA ONLINE.

Autor: Samuel Yeray Díaz Cabrera

Tutores: Francisco Mario Hernández Tejera
Blas J. Galván González

Dedicatoria

Dedico este proyecto a mi madre, Ana, por su esfuerzo y constancia durante todos estos años.

Agradecimientos

En primer lugar quiero agradecer a mis tutores, Mario Hernández Tejera y Blas Galván González por su paciencia, apoyo y libertad que me han dado a lo largo de la elaboración de este proyecto.

Agradecer al personal de la División de Computación Evolutiva y Aplicaciones en Ingeniería (CEANI) del Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería (SIANI) por sus consejos y ayudas dadas en la realización de este proyecto, especialmente a Juan Pedro Ramos Ponce, por su supervisión y sus buenas ideas en estos últimos meses de desarrollo.

Mención especial a mi compañera Nayarit Santana Pacheco por su inestimable compañía y optimismo. Sin su apoyo y su ayuda este proyecto no hubiese podido realizarse.

Agradecer también a mis compañeros y amigos, Aaron, Eduardo, Mario y Patricia por su apoyo durante estos años y, especialmente a Orlando y Zenaida por su compañía y su buen hacer durante estos últimos meses.

Un agradecimiento especial a mis amigos José, Ruyman y Joy por acompañarme durante estos años y aguantar mis nervios.

Por último agradecer a mis padres y hermano, sin cuyo apoyo nada de esto hubiese sido posible.

Índice

Capítulo 1.- Introducción

Este proyecto surge de la propuesta de los profesores y tutores de este proyecto, D. Blas J. Galván González y D. Mario Hernández Tejera, que con el fin de continuar con la investigación e innovación que se está desarrollando en la División del CEANI, del Instituto Universitario SIANI de la ULPGC, proponen realizar un entorno software para el análisis de riesgos en entornos industriales de forma online.

1.1. El problema

En la actualidad, el desarrollo tecnológico está fuertemente vinculado a la evolución de las sociedades, mejora sus condiciones de vida. Este desarrollo requiere de una mayor industrialización, lo que permite que exista un mayor riesgo de ocurrencia de accidentes. Por ello, es necesario el control del riesgo de que se produzcan los mismos.

Actualmente los establecimientos industriales, tales como fábricas, almacenes, puertos, etc., están obligados a elaborar un Informe de Seguridad por la legislación existente que tiene cada estado ante las emergencias. Se aplican leyes y reglamentos que garantizan la seguridad propia y del entorno en el que se encuentra un establecimiento industrial.

Todo establecimiento que deba elaborar un Informe de Seguridad debe realizar un Análisis de Riesgo. Es una herramienta de ayuda al control de riesgos en el que se identifican acciones orientadas a la prevención de la ocurrencia de accidentes, así como la planificación de las actuaciones en caso de que se produzca. El objetivo principal es minimizar las consecuencias de un accidente.

En general, el Análisis de Riesgo trata de estudiar, evaluar, medir y prevenir los fallos y averías de los sistemas y de los procedimientos operativos que pueden iniciar y desencadenar sucesos no deseados (accidentes) que afecten a las personas, los bienes y el medio ambiente.

En el ámbito de los establecimientos industriales, tiene como objetivo identificar accidentes graves que pueden ocurrir en el mismo. Así como determinar las consecuencias y los daños producidos. Su contenido es el siguiente:

1. Identificación de peligros de accidentes graves.
2. Cálculo de consecuencias.
3. Cálculo de vulnerabilidad.
4. Relación de accidentes graves identificados.
5. Medidas de prevención, control y mitigación.

La idea general propuesta es el desarrollo de una plataforma online que permita realizar las tareas necesarias para la elaboración de un Análisis de Riesgo.

1.2. Justificación

Debido a la complejidad y envergadura de la idea inicial, se decidió dividir la misma para la realización de 2 proyectos fin de carrera.

El primero, una plataforma multiusuario para el análisis y gestión de los riesgos de accidentes industriales titulado "*Diseño y prototipado de una plataforma web sobre base cartográfica para el análisis de riesgos en entornos industriales*", realizado por Nayarit Santana Pacheco.

Y el segundo, la creación de un visor de mapas para esta plataforma, titulado "*Desarrollo de un prototipo de componente web para la visualización e interacción con mapas de forma online*"; el actual proyecto.

El principal objetivo de este proyecto es el desarrollo de una herramienta de visualización de mapas genéricos, que funcione de forma online y sea capaz de editar sobre ellos, diferentes tipos de entidades. A su vez, su desarrollo está enfocado a la creación de un componente que permita su utilización en futuras aplicaciones. Y además, proporcionar una documentación de las funciones dadas al desarrollador, la interfaz del componente o API, que permita su integración.

A continuación se definen un conjunto de términos utilizados en el resto del documento.

1.3. Terminología

En esta apartado se detallan un conjunto de conceptos relacionados con el proyecto. Los cuales serán utilizados en el transcurso de la memoria por lo que es interesante conocerlos.

- **Accidente**.- Suceso imprevisto, que altera la marcha normal o prevista de las cosas, especialmente el que causa daños a una persona o cosa.

- **Sistema de coordenadas**.- En geometría, un sistema de coordenadas es un sistema que utiliza uno o más números (coordenadas) para determinar unívocamente la posición de un punto o de otro objeto geométrico.

- **Entidad**.- Elemento único representable gráficamente, que puede representar un elemento físico o un elemento virtual, como puede ser un accidente o las zonas de riesgos producidas por este.

- **Mapa**.- Representación geográfica de una parte de la superficie terrestre, en la que se da información relativa a una ciencia determinada.

- **Plano**.- Representación esquemática, en dos dimensiones y a determinada escala, de un terreno, una población, una máquina, una construcción, etc.

En esta memoria se utiliza con las acepciones de representación de un terreno o una instalación industrial.

- **Recurso**.- Elemento identificable que es susceptible de ser afectado por un accidente.

- **Riesgo**.- Es la posibilidad de ocurrencia de un accidente. Está vinculado a un tipo de daño o de repercusión dañina específica en un periodo de tiempo determinado o en circunstancias determinadas.

- **Peligro**.- Situación en la que existe la posibilidad, amenaza u ocasión de que ocurra una desgracia o un contratiempo.

- **Sistema de Información Geográfica (GIS)**.- Es un sistema para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión.

- **Visor GIS**.- Representa un mapa con datos GIS, con objetos del mundo real (carreteras, el uso del suelo, altitudes). Existen 2 tipos de GIS, los que representan los datos de forma vectorial y los que lo hacen de forma raster. Los más populares son aquellos que se centran en el manejo de datos vectoriales. Los de tipo raster son muy utilizados en estudios que requieran la generación de capas continuas, necesarias en fenómenos no discretos; también en estudios medioambientales donde no se requiere una excesiva precisión espacial (contaminación atmosférica, distribución de temperaturas, localización de especies marinas, análisis geológicos, etc.).

- **Vulnerabilidad**.- Es el índice o grado de daño que puede ser producido en caso de accidente.

- **Zona de riesgo**.- Área que representa la zona afectada por la repercusión de la ocurrencia de un accidente.

Capítulo 2.- Estado del Arte

Como se planteó en el capítulo anterior, el proyecto que se aborda consiste en el desarrollo de un visor web que permita su integración en un entorno de simulación de accidentes para permitir la generación de Análisis de Riesgos.

En esta sección se analizan 2 tipos de visores, visores cartográficos de mapas donde se representan información y planos, y visores relacionados con entornos de simulación.

En el primer grupo se analizan los visores online más conocidos. El objetivo del análisis es destacar los aspectos de usabilidad más importantes de los mismos para ofrecer en nuestro desarrollo una experiencia de usuario semejante.

En el grupo de los visores de Análisis de Riesgo se describen las características más destacadas relacionadas con las simulaciones de los visores analizados.

A continuación se listan todas aplicaciones analizadas.

- Google Maps
- Yahoo! Maps
- ArcGis
- Phast

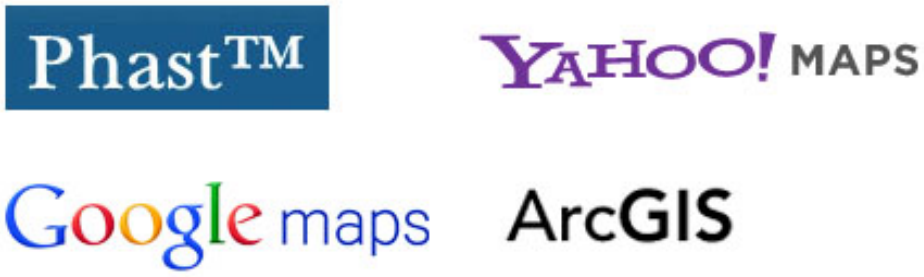

Ilustración 1: Visores analizados

Puesto que las plataformas que se estudian están en constante actualización, las imágenes que acompañan a los análisis pueden no reflejar la interfaz de los visores en la actualidad. Las imágenes reflejan la instantánea en el momento de redacción de este documento.

Se analizan en primer lugar, los visores cartográficos online.

2.1. Visores cartográficos de propósito general

En esta sección se analizan visores online de propósito general, de los que se pretenden obtener características de usabilidad. Se comenzará por los más utilizados y se irá avanzando en el análisis hacia visores con menos repercusión.

2.1.1. Google Maps

Google Maps, es el servicio de mapas proporcionado por Google. Es un servicio online que, dependiendo de la ubicación del usuario, permite ver mapas básicos o personalizados con información sobre negocios locales, su ubicación, datos de contacto e incluso el cómo llegar a ellos. Tiene diferentes vistas, satélite y mapa, donde se muestra de diferente manera la zona actual. Permite además observar la ubicación desde diferentes niveles de zoom.

La plataforma muestra la siguiente interfaz:

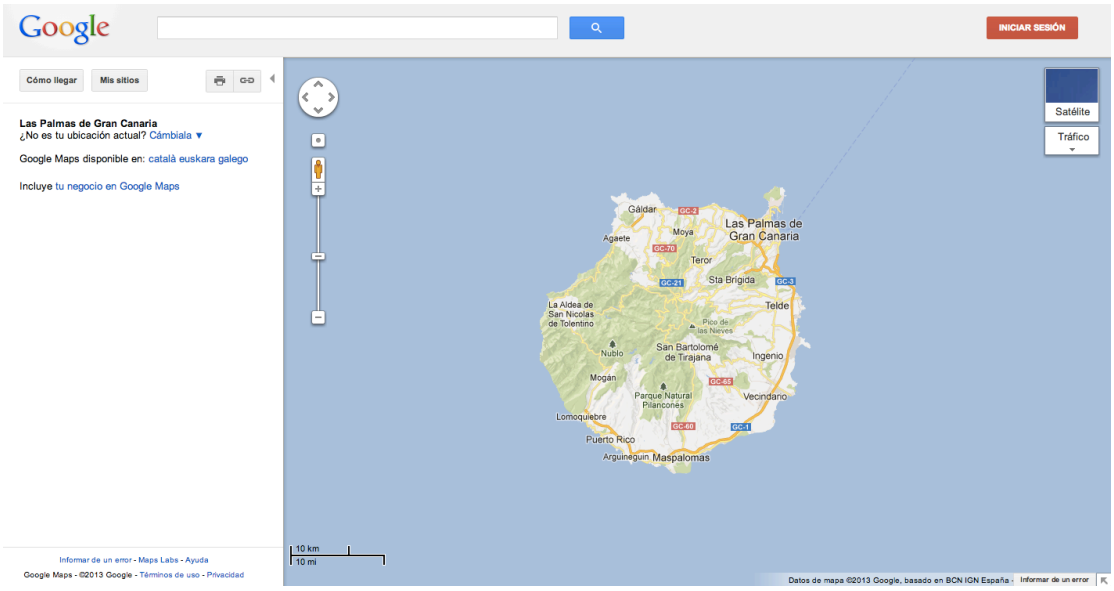

Ilustración 2: Entorno Google Maps

Dispone de una interfaz bastante intuitiva y clara, donde se pueden realizar diferentes acciones, desde buscar una localización, ampliar o reducir la vista, desplazarse por el mapa, intercambiar entre distintos tipos de vistas (mapa o satélite), etc.

Ilustración 3: Visor central de Google Maps

Su visor central contiene diversos controles:

- Un control para realizar desplazamientos del plano.
- Un control para cambiar el nivel del zoom. También permite posicionar un muñeco en el mapa para activar la vista "Street View" (Una vista que permite ver la zona escogida a pie de calle).
- Incluye también una barra horizontal de información sobre la escala actual.
- Incluye información del copyright del mapa que se está mostrando.
- Un botón para desplegar un mini visor del mapa, para mover el plano de forma más cómoda.
- Tiene botones para el intercambio de distintos tipos de vista, mapa, satélite y relieve. En este mismo botón se ofrece la posibilidad de activar distintas capas de información, que incluyen desde vista de tráfico hasta videos y fotos geo posicionados.
- Incluye además botones para imprimir la imagen, enviarla por correo electrónico, o sacar un enlace a dicha posición del mapa.

Esta plataforma ofrece una API para que la utilicen terceros. Podría ser una herramienta a utilizar.

La otra alternativa, que a continuación se analiza, es el servicio Yahoo! Maps, proporcionada por la empresa Yahoo!.

2.1.2. Yahoo! Maps

Ofrece una interfaz similar a la de Google Maps. Tiene diferentes campos de búsqueda, para encontrar lugares, negocios, o establecer una ruta entre 2 puntos, A y B.

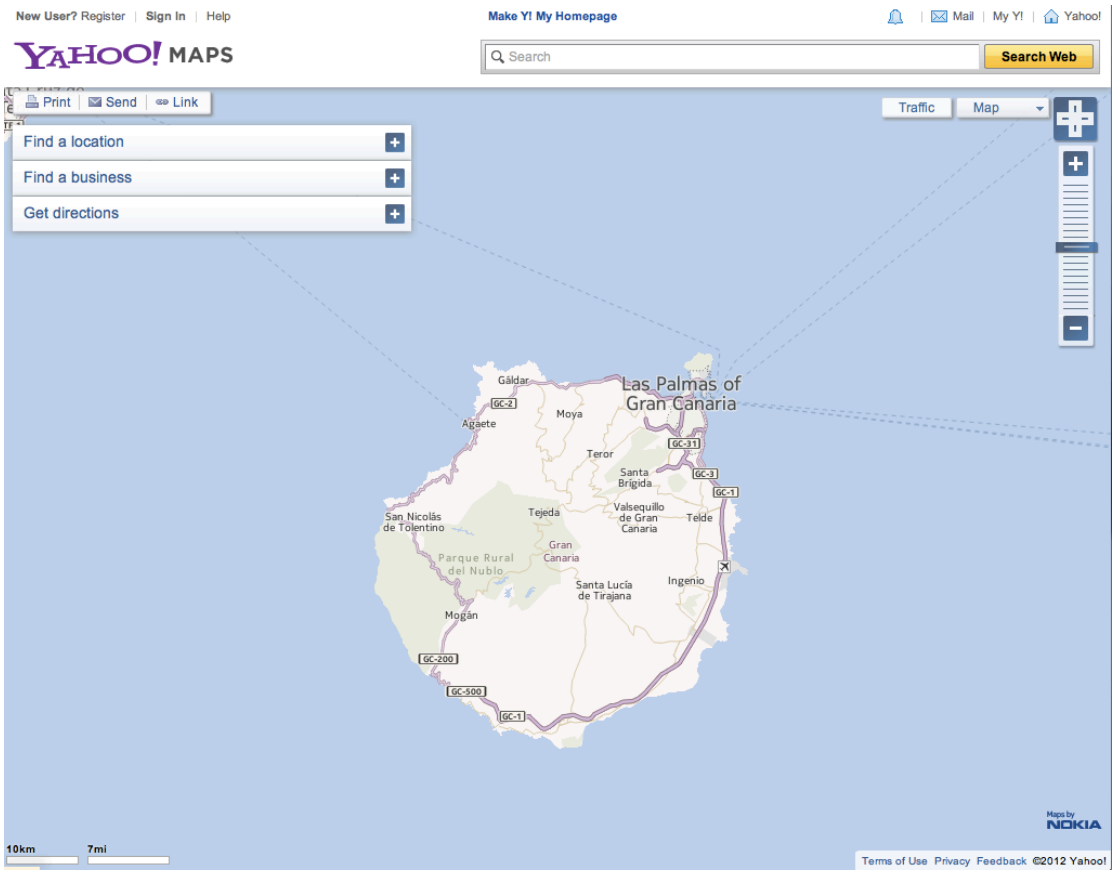

Ilustración 4: Entorno Yahoo! Maps

Su visor central es similar al proporcionado por Google. Tiene controles para:

- Cambiar el tipo de mapa mostrado.
- Cambiar el nivel de zoom sobre el mapa.
- Un gráfico con la escala a la que se encuentra el mapa, en millas y kilómetros.
- Opciones para imprimir, enviar o guardar el mapa mostrado o mostrar el trafico en vivo.
- Un selector de idioma.
- Un desplegable con un mini visor.
- Varias etiquetas con información sobre el copyright del creador de la aplicación. Y del proveedor de los mapas.

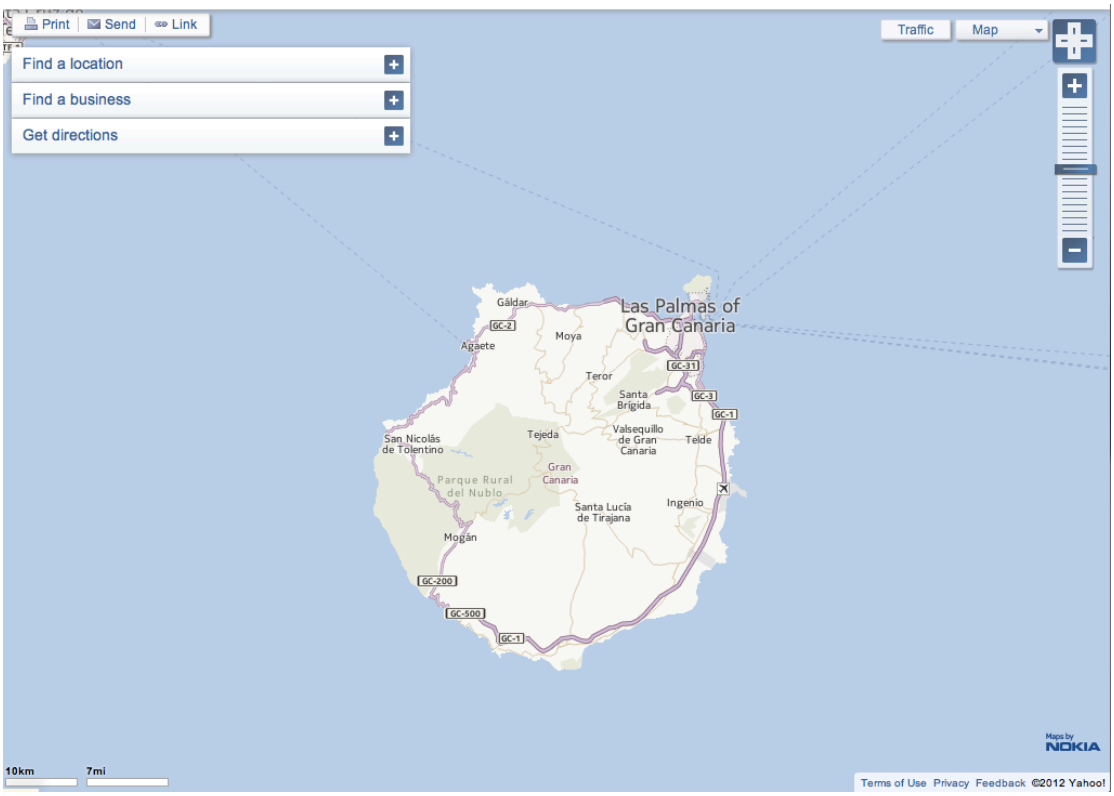

Ilustración 5: Visor central de Yahoo! Maps

Yahoo! Maps al igual que Google Maps ofrece también una API, pero a diferencia de esta última, quién ofrece la API no es la propia Yahoo! Maps, sino NOKIA, ya que Yahoo! Maps está utilizando los mapas creados por NOKIA.

2.1.3. ArcGIS Viewer

Es una plataforma que ofrece distintos tipos de mapas con herramientas muy potentes para la creación de aplicaciones en la plataforma que se quiera. Ofrece APIs para proporcionar mapas o información con las siguientes tecnologías:

Android, ArcObject, Flex, IOs, Java, JavaScript, Python, REST, SharePoint, SilverLight, SOAP, SQL, Windows Mobile, Windows Phone, WPF.

Analizando el visor utilizado para la tecnología FLEX, éste tiene el siguiente aspecto:

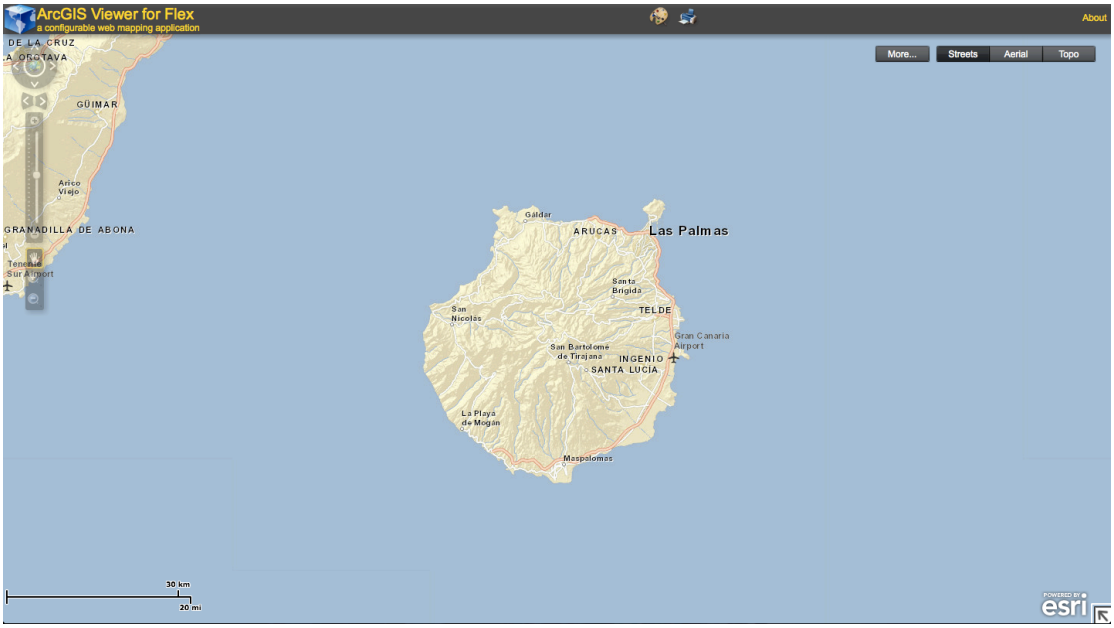

Ilustración 6: Visor de ArcGIS

Su visor tiene las opciones vistas por los visores anteriores y además dispone de un componente que muestra opciones desplegables para realizar acciones como cambiar el tipo de mapa, realizar zooms, imprimir, etc. todo esto en un entorno muy intuitivo y de aspecto agradable.

Explorando las opciones de los distintos paneles, existen opciones para mostrar un listado de visualización de capas de información, así como opciones para mostrar gráficas sobre datos meteorológicos.

2.1.4. Resumen del análisis

Tras analizar estos visores destacan los siguientes aspectos de usabilidad:

- Los visores tienen componentes visuales para el acceso rápido a las distintas funcionalidades.
- Todo el manejo de los mismos, se desarrolla a través del uso del ratón, aunque también es posible utilizar los componentes visuales.
- Las funcionalidades ofrecidas a través de usos del ratón son las siguientes:
    - Mover el mapa, a través de hacer click y arrastrar el ratón.
    - Ampliar o reducir una zona visible donde se sitúa el cursor del ratón. Se gira la rueda del ratón en un sentido u otro para ampliar o reducir la vista.
    - Ampliar una zona visible donde se sitúa el cursor del ratón, realizando un doble click sobre ese punto.
    - Funciones avanzadas dependiendo de alguna opción escogida en algún panel, tal como pintar un punto de inicio y un punto final para obtener una ruta entre los 2 puntos.

- Las principales funciones ofrecidas por los componentes visuales son las siguientes:

    - Conjunto de botones para desplazar el mapa sin tener que realizarlo arrastrando. Contiene los botones de arriba, abajo, izquierda y derecha.

    - Barra de zoom, permite realizar zoom.

    - Barra de búsqueda, busca la ubicación deseada en el visor.

    - Barra para cambiar el tipo de visualización, permite cambiar el tipo de plano mostrado.

    - Barra de información adicional, permite mostrar otro tipo de información diferente.

    - Barras adicionales, permiten utilizar funcionalidades características de cada visor, por ejemplo el Street View de Google Maps.

Algunos de los visores que se han analizado suministran una API para incorporar el visor a distintos tipos de entornos de desarrollo. Esto conlleva ciertas ventajas y desventajas.

En general el utilizar APIs de terceros proporcionan ventajas tales como:

- Rapidez. Integrar en cuestión de segundos los servicios ofrecidos por la API a una aplicación libera al programador del desarrollo de gran parte de la programación, permitiendo centrar sus esfuerzos en el dominio de negocio de la aplicación.

- Robustez. El uso extendido de las APIs permite que estas estén en bastante contacto con toda clase de situaciones, por lo que el producto puede considerarse bastante depurado, por lo que se garantiza una respuesta adecuada en la mayoría de las ocasiones.

- Mantenimiento e Innovación. En general las grandes compañías no dejan que sus productos se deterioren, por lo que es bastante común el mantenimiento y la mejora de los servicios accedidos a través de su API, que además en muchos casos es totalmente transparente al programador que las usa.

Por el contrario, también tienen sus desventajas:

- Condiciones cambiantes. Al ligar un software a una tercera empresa por un contrato tan débil como es la licencia de uso de una API, normalmente se está expuesto a la voluntad de la otra parte, lo que puede suponer recortes en las prestaciones, obsolescencia de métodos implementados o pérdida de la gratuidad del servicio.

- Disponibilidad. El depender de un software de un tercero también está condicionado a su disponibilidad. Las caídas de servicios externos, pueden dejar nuestros servicios inutilizados.

- Privacidad. Las leyes son muy distintas para cada país, por lo que al usar una API de un servicio externo se obliga al software a cumplir tanto las leyes que rigen en el país que proporciona el servicio como las leyes del país origen del software.

Tomando como ejemplo la plataforma Google Maps, esta ofrece algunas limitaciones que hacen cuestionar su utilización.

- Para utilizar esta API, se ha de obtener una "Google Maps Key". Esta clave está limitada para una única URL lo que obliga a tener un sitio Web para el desarrollo de las aplicaciones.

- La "Google Maps Key" es solamente válida para un directorio del sitio Web.

- Su uso está condicionado a unas condiciones y términos que son incompatibles con ciertas aplicaciones. Por ejemplo:

    - La aplicación debe ser gratuita para todo aquel que la use.

    - Debe tener un acceso libre, no puede ejecutarse en una red interna o en una comunidad cerrada al acceso público.

- También está condicionado a un número de accesos diarios, en este caso, 2500 solicitudes diarias.

A continuación se analiza un visor offline centrado en la representación de simulaciones.

2.2. Visor orientado al Análisis de Riesgo

En esta sección se analizará un caso de herramienta útil en el Análisis de Riesgo.

2.2.1. Phast

En la actualidad en el análisis de riesgos existe una aplicación de éxito bastante avanzada. Se trata de Phast de la compañía DNV (Det Norske Veritas). Es una aplicación de escritorio de pago para el análisis de riesgos industriales que abarca todas las fases del proceso de la gestión de un posible riesgo; desde el diseño del accidente concreto hasta el análisis de los resultados obtenidos con la simulación realizada. Permite analizar el estado actual de una zona industrial, y sobre el plano evaluar posibles zonas de riesgos de los accidentes simulados.

El funcionamiento básico de Phast consiste en:

1. Establecer un estado inicial, configurando valores de condiciones climáticas, tales como velocidad y dirección del viento, o la temperatura ambiental.

2. Añadir los posibles accidentes existentes en las instalaciones. Se pueden añadir, uno o varios accidentes, cada uno con un modelo matemático y unas propiedades que establecen el comportamiento del mismo. Estos accidentes están basados en procesos químicos.

3. Realizar la simulación de los modelos matemáticos establecidos, obteniéndose tablas y gráficas correspondientes a los datos obtenidos con las simulaciones de cada accidente. En este proceso, se calcula la concentración de la sustancia, la radiación, la toxicidad, la sobrepresión de explosión, etc.

4. Para cada accidente, la aplicación muestra los resultados de la simulación, desde que ocurre y se libera, hasta que se forma una nube tóxica y ésta se dispersa. Se generan tablas con los resultados y se representan en el visor, una o varias áreas de la repercusión del accidente, estableciendo las zonas de riesgo. Cada área establece una zona correspondiente a un tipo de peligrosidad, dada por el modelo matemático utilizado.

A continuación se describe su entorno gráfico.

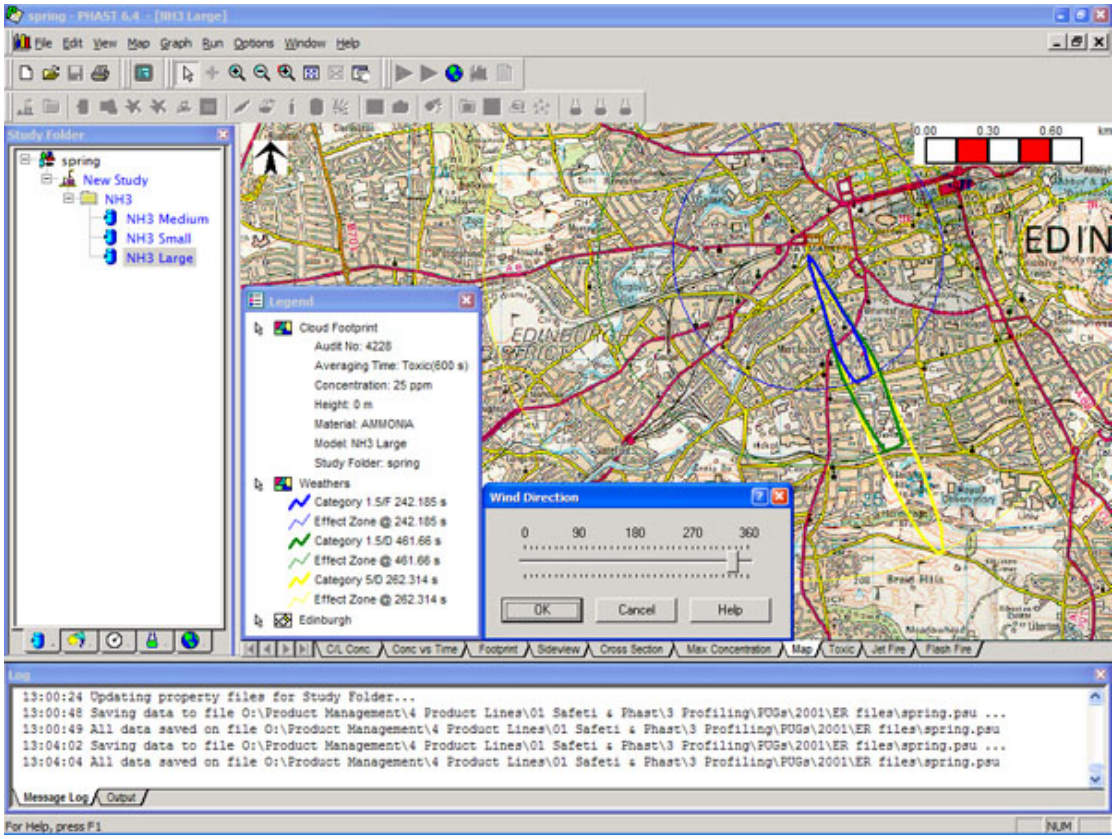

Ilustración 7: Entorno Phast

La ventana principal del Phast, se encuentra dividida en varias secciones. Tiene una barra superior de opciones donde se encuentran las opciones de guardar, imprimir, seleccionar, realizar zoom, y simular. Una barra lateral izquierda donde se establecen las propiedades de las condiciones y accidentes establecidos sobre el escenario. La barra inferior donde se muestra los resultados de los comandos ejecutados y por último el panel central, donde se encuentra el mapa con su leyenda asociada y un panel para establecer la dirección del viento.

El visor utilizado en esta herramienta, es en sí, muy básico, una imagen utilizada de fondo sobre la que es posible realizar ampliaciones o reducciones de la vista, y mostrar encima de ésta, las distintas capas de información gráfica.

Tras realizar una simulación, el visor representa las zonas de riesgo producidas por el accidente. La ilustración 8 muestra un ejemplo de las mismas.

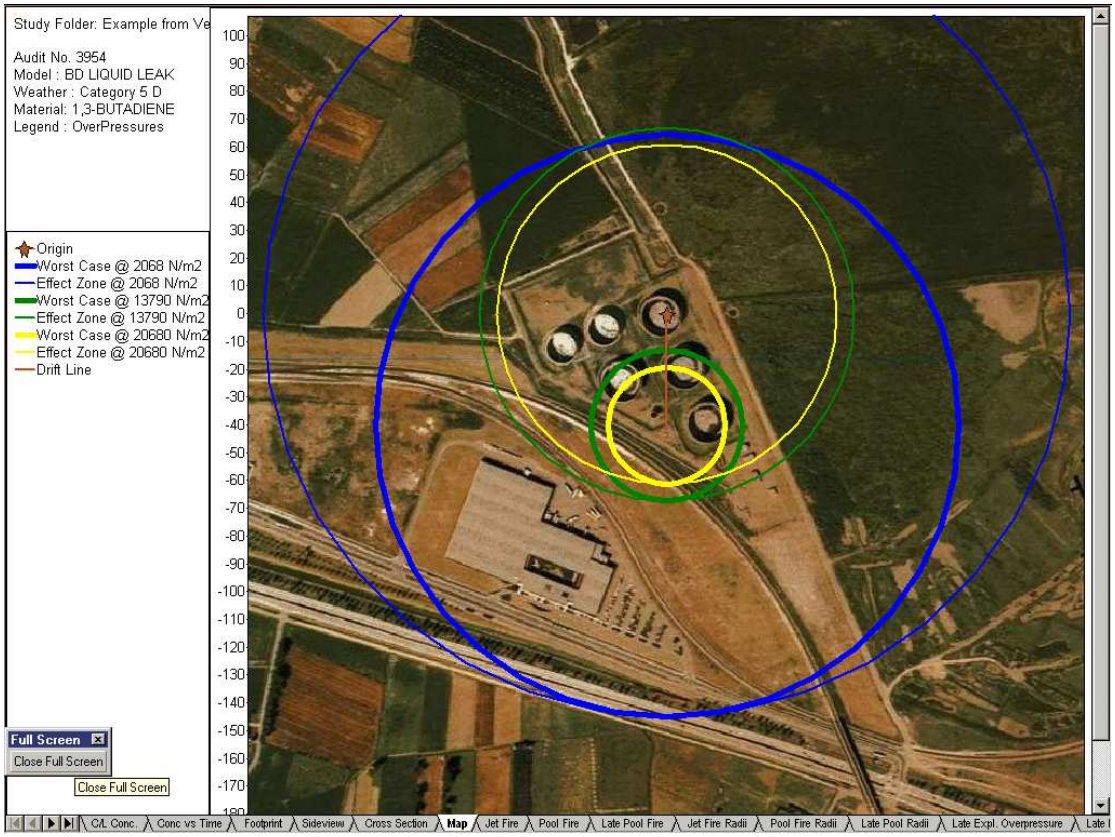

Ilustración 8: Visor Phast

El visor dispone a su izquierda un panel con la leyenda de los colores y valores asociados a las distintas zonas. Éstas las representa con distintos tipos de circunferencias. Algunas están centradas en el accidente, mientras que otras están desplazadas, esto depende del tipo de resultado obtenido, las desplazadas son emisiones de gases(nubes) producidas por el accidente.

El ejemplo escenifica el desplazamiento de las nubes mediante la representación de una recta. Ésta indica la dirección y la distancia que ha tomado la nube desde el punto original(el accidente), hasta el instante actual de representación.

A continuación se realiza un resumen de las características propias de este tipo de visor orientado a las simulaciones.

2.2.2. Resumen del análisis

El principal uso del visor descrito es el de la representación del resultado de una simulación de un modelo matemático. Como tal, se van a listar las características más importantes del mismo.

- Se representa una imagen cargada por el usuario, normalmente un plano de su establecimiento industrial.

- Se establecen los posibles accidentes que pueden causar algún tipo de riesgos.

- Se representa en el visor los resultados obtenidos de la simulación del modelo matemático. Correspondiendo habitualmente a un área afectada y a un posible desplazamiento desde su origen.

- Se observa el comportamiento de las diferentes zonas de riesgo a lo largo del tiempo. Mostrando un comportamiento dinámico de la situación producida por la reproducción del accidente.

2.3. Conclusiones

El objetivo de este proyecto no es el de desarrollar un mejor visor GIS que los vistos en el apartado anterior, puesto que no se cuenta ni con el mismo personal, ni con el mismo presupuesto para llevar a cabo un desarrollo de tanta envergadura.

El objetivo principal consiste en la creación de un visor que permita realizar simulaciones como las observadas en el producto Phast, pero de forma online. Y que además mantenga una usabilidad similar a los visores populares.

Ninguno de los visores analizados se adapta a las necesidades que se abordan en este proyecto, ser un visor online orientado a la simulación de accidentes y al análisis de los riesgos producidos por estos.

Se podría utilizar un visor que suministre una API para implementar nuestro caso de uso, pero teniendo en cuenta las desventajas vistas anteriormente en el uso de APIs de terceros, utilizarlos supondría una limitación, bien por los términos y las condiciones de uso que limitarían la utilización final del visor, o bien por el coste de mantenimiento del software si la API o los mismos términos y condiciones cambian.

Por ello, se ha realizado el análisis de los distintos visores con el fin de obtener características destacables para aplicarlos al desarrollo. Estas son:

1. Representación de un plano.

2. Movimiento del plano.

3. Posicionamiento de figuras dentro de este.

4. Manejo de capas de simulación.

5. Proporcionar una API para su utilización.

Capítulo 3.- Objetivos

El principal objetivo de este proyecto es la creación de un componente software para la visualización e interacción con planos, así como una librería de funciones (API) que permita su integración en el proyecto de gestión de riesgos en entornos industriales "*Diseño y prototipado de una plataforma web sobre base cartográfica para el análisis de riesgos en entornos industriales*".

Además, la API a proporcionar debe mantener una terminología genérica, no relacionada con la representación de accidentes y riesgos, para permitir la integración del componente en otro tipo de entornos.

Los principales objetivos a cubrir por el visor son los siguientes:

- Carga y visualización de planos localizados en un sistema de coordenadas.
- Movimiento del plano.
- Zoom sobre el plano.
- Entidades posicionadas en el mapa con unas coordenadas.
- Inserción, modificación y borrado de entidades sobre el mismo.
- Configuración gráfica de las entidades(Colores, tamaños, bordes, etc.).
- Entidades dinámicas, tienen un comportamiento en el tiempo.
- Adaptabilidad ante diferentes situaciones.
- Permitir añadir o desactivar controles en el visor.
- Disponibilidad de una API clara que permita su configuración y su utilización por otro software.

La API a especificar debe ofrecer una funcionalidad completa. Actuar tanto sobre la configuración visual del visor o de sus subcomponentes, como en la representación de diferentes tipos de información.

3.1. Alcance

Este proyecto abarca el desarrollo de 3 elementos unívocos:

1. La creación de un componente web basado en una tecnología para la construcción de RIAs .

2. La creación de una API junto a su documentación correspondiente que documente la funcionalidad ofrecida para permitir la integración y utilización del mismo por parte de terceros.

3. La creación de una aplicación "*demo*" para la demostración de la funcionalidad del componente.

Capítulo 4.- Metodología

Para el desarrollo de este proyecto se decidió utilizar la metodología del "*Proceso Unificado de Desarrollo de Software*", debido a una de las principales características de este proyecto, la creación de un prototipo. Esta metodología permite un proceso basado en iteraciones en las que se desarrollan prototipos a medida que se cumplen requerimientos.

Cumple con las siguientes características principales.

- Dirigida por casos de uso.
- Centrada en la arquitectura.
- Iterativa e incremental.

Un caso de uso es un fragmento de funcionalidad del sistema. Muestra un resultado de valor para el usuario. Por ello guían el proceso de desarrollo. Durante el diseño, implementación y transición los desarrolladores crean una serie de modelos basándose en los casos de uso. Se dice por tanto que el proceso está dirigido por los casos de uso.

Los casos de uso están profundamente relacionados con la arquitectura del software. Deben encajar con la arquitectura, y a su vez, ésta debe permitir el desarrollo de todos los casos de uso requeridos en la actualidad y en un futuro.

El Proceso Unificado de Desarrollo está centrado en la arquitectura porque el arquitecto desarrolla la arquitectura a partir de la comprensión de un conjunto reducido de los casos de uso, los fundamentales.

Es iterativo e incremental ya que el esfuerzo de desarrollo del proyecto se divide en partes más pequeñas o mini proyectos. Cada parte es una iteración que resulta ser un incremento del proyecto.

Las iteraciones son controladas, esto es, se seleccionan y se planifica una forma de ejecución. La selección de la parte que entra en cada iteración se determina mediante el conjunto de casos de uso que amplían la funcionalidad y los riesgos más importante que deben mitigarse.

El Proceso Unificado de Desarrollo se repite a lo largo de una serie de iteraciones que constituyen la vida del desarrollo del sistema. Cada iteración constituye una versión del sistema.

4.1. Fases

Generalmente cada iteración tiene una duración fija y corta, y en ella se deben ir cumpliendo objetivos para el correcto avance del proyecto. Este enfoque permite que exista una realimentación entre clientes y desarrolladores al mostrar a los mismos un conjunto de artefactos cada cierto tiempo. Posibilita la detección de necesidades no vistas en análisis previos y su implementación en futuros prototipos. Es una metodología bastante moderna al permitir añadir nuevas características al software en cada iteración.

Las iteraciones por las que pasa el desarrollo de un proyecto se agrupan en fases. Estas proporcionan un sentido a las mismas categorizándolas en función de los objetivos que se cumplan.

Las fases de las que consta el Proceso Unificado de Desarrollo son: inicio, elaboración, construcción y transición.

- **Fase de inicio**.- Esta fase tiene como propósito definir y acordar el alcance del proyecto, así como identificar los riesgos potenciales asociados al proyecto, es decir, hacer una obtención de requisitos.

- **Fase de elaboración**.- Se seleccionan los casos de uso que permiten describir el problema, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar.

- **Fase de construcción**.- El propósito de esta fase es la de completar el sistema cumpliendo los distintos requisitos propuestos para su desarrollo en la iteración actual.

- **Fase de transición**.- En esta fase se asegura que el software está libre de errores, que cumple los requisitos y proporciona una documentación de apoyo.

Durante las fases de inicio y elaboración se utilizan herramientas de modelado como las proporcionadas por el lenguaje de modelado UML. En las fases de construcción se utiliza el lenguaje propio de la tecnología escogida. Y finalmente, en la fase de transición se utilizan herramientas de testeo y de generación de documentación.

Tras cada iteración se evalúa el resultado obtenido. Se analizan las características positivas y negativas para conservarlas o cambiarlas. Se eligen nuevas funcionalidades y capacidades para agregar al sistema. En definitiva, se deciden nuevos objetivos a abordar que deben ser completados al final de la siguiente iteración.

Capítulo 5.- Planificación y Temporización

Como se detalla en el capítulo anterior, el Proceso Unificado de Desarrollo fabrica objetos completos o artefactos al final de cada iteración. Este enfoque permite establecer un conjunto de hitos a satisfacer en cada etapa.

La planificación se estableció en función del número de prototipos o de objetos a desarrollar. Los cuales establecieron el número de iteraciones por las que se debía pasar.

A continuación se listan las principales actividades que se realizan en cada fase. Hágase notar que todas las actividades se recorren en cada iteración en mayor o menor medida, dependiendo de la fase en la que se encuentre.

5.1. Actividades

Fase 1: Inicio

Actividad 1.1 Documentación y herramientas

- Realización de encuestas y entrevistas.
- Adquisición de información.
- Evaluación y selección de herramientas necesarias para la generación de la documentación del análisis.
- Generación de documentación sobre herramientas.

Actividad 1.2 Análisis de requisitos de usuario

- Análisis de requerimientos de usuario.
- Generación documentación análisis de requisitos de usuario.

Actividad 1.3 Análisis de requisitos de software

- Análisis de requerimientos de software.
- Generación documentación análisis de requisitos de software.

Fase 2: Elaboración

Actividad 2.1 Estudio previo

- Selección de casos de usos a elaborar.

- Estudio de las herramientas seleccionadas para la elaboración del proyecto.

Actividad 2.2 Diseño del componente

- Diseño del módulo gestor del plano.
- Diseño del módulo controlador del tiempo.
- Diseño del módulo controlador de la escala.
- Diseño del módulo gestor de entidades.
- Diseño de subcomponentes visuales.
- Generación de documentación.

Actividad 2.3 Diseño del encapsulado

- Diseño de las propiedades públicas.
- Diseño de los eventos suministrados.
- Elección de los paquetes suministrados.
- Elección de eventos suministrados.
- Elección de la funcionalidad dada y los parámetros establecidos.
- Elección de la nomenclatura de las interfaces de entrada/salida.

Actividad 2.4 Diseño de la aplicación de Test

- Selección de la funcionalidad ofrecida.
- Generación de bocetos.

Fase 3: Construcción

Actividad 3.1 Implementación del componente

- Implementación del módulo gestor del plano.
- Implementación del módulo controlador del tiempo.
- Implementación del módulo controlador de la escala.
- Implementación del módulo gestor de entidades.
- Implementación de subcomponentes visuales.

Actividad 3.2 Implementación de interfaz de la aplicación de Test

- Implementación de la aplicación.

Fase 4: Transición

Actividad 4.1 Test y prueba del componente en otro entorno

- Entrega e integración del prototipo.
- Corrección de errores.

Actividad 4.2 Validación de usuario final

- Evaluación de sugerencias.

Actividad 4.3 Documentación

- Confección de manuales de integración del componente.
- Confección de la API para desarrollador.

5.2. Temporización

El tiempo utilizado para el desarrollo se basa en el número de iteraciones realizadas, y estas a su vez en el número de objetos elaborados.

Para el desarrollo de este proyecto se han realizado once iteraciones. En ellas, la mayor carga de trabajo ha sido realizada en la fase de construcción, donde se han desarrollado el mayor número de objetos.

A continuación se describen los objetos desarrollados en cada fase.

Fase	Iteración	Objeto
Inicio	1	Requisitos de Usuario y de Software.
Elaboración	2	Diseño del Componente y del Encapsulado.
Elaboración	3	Diseño del Componente y del Encapsulado.
Construcción	4	Prototipo 1.
Construcción	5	Prototipo 2.
Construcción	6	Prototipo 3.
Construcción	7	Prototipo 4.
Construcción	8	Prototipo 5.
Transición	9	Documentación, API y publicación web.
Transición	10	Documentación, API y publicación web.
Transición	11	Documentación, API y publicación web.

Tabla 1: Relación de objetos desarrollados en cada fase

En la fase de construcción se han desarrollado los principales prototipos del componente. Estos han cubierto los siguientes hitos.

- **Prototipo 1**.- Carga y visualización del plano.
- **Prototipo 2**.- Movimiento y escala del plano.
- **Prototipo 3**.- Representación de entidades.
- **Prototipo 4**.- Creación del sistema de tiempo.
- **Prototipo 5**.- Corrección de errores y homogeneización de la API.

Para concluir se muestra la relación de horas utilizadas en cada fase por iteración recorrida.

	Inicio	Elaboración		Construcción					Transición		
Disciplinas											
Actividad 1.1	20										
Actividad 1.2	40	10									
Actividad 1.3	40	20	10								
Actividad 2.1		50	30								
Actividad 2.2		20	30	20	10	10		10			
Actividad 2.3			20		10						
Actividad 2.4			10	10	10	10	10	10			
Actividad 3.1				50	40	40	50	30	20	10	
Actividad 3.2				10	20	30	30	20	10		10
Actividad 4.1				10	10	10	10	30	20	20	20
Actividad 4.2									30	20	20
Actividad 4.3									20	50	50
Iteración	1	2	3	4	5	6	7	8	9	10	11
Horas	100	200		500					300		
				Horas totales					1100		

Tabla 2: Temporización establecida

Capítulo 6.- Recursos necesarios

En esta sección se detallan las herramientas hardware y software utilizadas para el desarrollo del proyecto.

6.1. Hardware

Para el desarrollo de este proyecto se han utilizado las siguientes herramientas hardware:

- MacBook Air 2012.
- Pendrive Verbatim 8 GB.
- Disco duro externo Toshiba 500 GB.

Destacar además, que se ha contado con un puesto de trabajo en la División CEANI del Instituto Universitario SIANI donde se ha dispuesto a uso del alumno de:

- PC Dual Core.
- Impresora Láser.

6.2. Software

Para detallar los elementos software utilizados se realiza una clasificación por categorías de los mismos.

6.2.1. Sistemas Operativos

- Windows 7.
- Mountain Lion OS X 10.8.4.

6.2.2. Editores de texto

- Writer de OpenOffice 3.0.
- Word de Microsoft Office 2011.

6.2.3. Editores de presentaciones

- PowerPoint de Microsoft Office 2011.

6.2.4. Entornos de desarrollo

- Flex Builder 3.0 (Licencia Educativa proporcionada por Adobe).
- Flash Builder 4.6 (Licencia Educativa proporcionada por Adobe).

6.2.5. Herramienta de sincronización entre dispositivos

- Dropbox 2.0.22.

6.2.6. Sistema de versiones

- Git.
- Entorno gráfico SmartGit 3.
- Entorno gráfico SourceTree 1.5.6 .

6.2.7. Herramientas Organizativas

- Evernote 5.2.1.
- Things 2.2.1.

6.2.8. Herramientas de Diagramación

- ArgoUML 0.34.
- OmniGraffle Professional 5.4.
- CACOO (Herramienta online www.cacoo.com).
- LucidChart (Herramienta online www.lucidchart.com).

6.2.9. Navegadores Web

- Google Chrome 28.0.1500.95.
- Safari 6.0.5.
- Microsoft Internet Explorer 10.

Capítulo 7.- Requerimientos

El Proceso Unificado de Desarrollo establece que la captura de requerimientos se realice siguiendo unas pautas, estas son:

- Enumerar los requisitos candidatos.
- Comprender el contexto del sistema.
- Capturar los requisitos funcionales.
- Capturar los requisitos no funcionales.

En cada pauta se generan diferentes formas de documentación. En unos casos se generan modelos, como el modelo de dominio o el modelo de casos de usos, y en otros, tablas descriptivas. Al seguir estas pautas se obtiene una comprensión del problema de forma continuada y progresiva.

Se sigue, en todos ellos, la regla número uno de la obtención de requerimientos, se generan los documentos utilizando un lenguaje comprensible para el cliente. Es la primera aproximación al problema sin entrar en aspectos técnicos.

En los apartados siguientes se detallan los documentos finales obtenidos por la captación de requerimientos, estos son, el listado de objetivos, los requisitos funcionales compuesto por los requisitos de usuario y los requisitos de software, los requisitos no funcionales, y el modelo de dominio del problema.

Antes de entrar en detalle en estos documentos se describen las plantillas utilizadas como formato en las tablas de descripción de requisitos.

7.1. Plantillas

Para realizar la enumeración y descripción de requisitos, se ha utilizado una modificación de las "plantillas de Durán y Bernárdez" dadas por Amador Durán y Beatriz Bernárdez, de la Universidad de Sevilla, que describen en su "Metodología para la Elicitación de Requisitos de Sistemas Software" (Durán, A. & Bernárdez, B. , 2002).

La metodología descrita por Amador Durán y Beatriz Bernárdez realiza una distinción entre los objetivos principales, los requerimientos iniciales, de los requisitos funcionales y los no funcionales del problema. A continuación se muestran las plantillas utilizadas en cada caso.

Plantilla para la descripción de objetivos

OBJ-**<id>**	***<Nombre descriptivo>***
Descripción	<El sistema deberá *<objetivo a cumplir por el sistema>*>
Estabilidad	*<Estabilidad del objetivo>*
Comentarios	*<Comentarios adicionales sobre el objetivo>*

Tabla 3: Plantilla de objetivos

Plantilla para la descripción de requisitos no funcionales

RNF-**<id>**	***<Nombre descriptivo>***
Versión	*<Nº de la versión actual>*
Autores	*<Autor de la versión actual>*
Fuentes	*<Fuente de la versión actual>*
Objetivos asociados	*OBJ-x <nombre del objetivo>*
Requisitos asociados	*Rx-y <nombre del requisito>*
Descripción	*El sistema deberá comportarse tal como se describe en los siguientes casos de uso.*
Importancia	*<Importancia del objetivo>*
Urgencia	*<Urgencia del objetivo>*
Estado	*<Estado del objetivo>*
Estabilidad	*<Estabilidad del objetivo>*
Comentarios	*<comentarios adicionales sobre el objetivo>*

Tabla 4: Plantilla de requisitos no funcionales

Plantilla para la descripción de casos de uso

R<U|S>-CU-**<id>**	***<Nombre descriptivo>***
Versión	*<Nº de la versión actual>*
Autores	*<Autor de la versión actual>*
Objetivos asociados	*OBJ-x <nombre del objetivo>*
Actor Principal	*<Actores>*
Personal involucrado e intereses	<Actores involucrados junto a sus intereses>
Precondiciones	*<Precondición del caso de uso>*
Garantías de éxito (Postcondiciones)	*<Postcondición del caso de uso>*
Escenario principal de éxito (o Flujo Básico)	*<Secuencia de pasos del flujo principal>*
Extensiones (o Flujos alternativos)	*<Secuencia de pasos del flujo secundario>*

Tabla 5: Plantilla de casos de uso

7.1.1. Descripción de los apartados

El significado de los campos que componen las distintas tablas es el siguiente:

- **Identificador y nombre descriptivo**.- Cada objetivo o requisito debe identificarse por un código único y un nombre descriptivo.

- **Versión**.- Para poder gestionar distintas versiones, este campo contiene el número o la fecha de la versión actual del objetivo.

- **Autores, Fuentes**.- Estos campos contienen el nombre y la organización de los autores y de las fuentes, de la versión actual del objetivo, de forma que se pueda llegar hasta las personas que propusieron la necesidad del requisito.

- **Descripción**.- Este campo se debe completar con la descripción del objetivo.

- **Importancia**.- Indica la importancia del cumplimiento del objetivo para los clientes y usuarios. Se puede asignar un valor numérico o alguna expresión enumerada como vital, importante o quedaría bien. En el caso de que no se haya establecido aún la importancia, se puede indicar que está por determinar (PD).

- **Urgencia**.- Indica la urgencia del cumplimiento del objetivo para los clientes y usuarios en el supuesto caso de un desarrollo incremental. Como en el caso anterior, se puede asignar un valor numérico o una expresión enumerada como inmediatamente, hay presión o puede esperar.

- **Estado**.- Indica el estado del objetivo desde el punto de vista de su desarrollo. El objetivo puede estar en construcción si se está elaborando, pendiente de negociación si tiene algún conflicto asociado pendiente de solución, pendiente de validación si no tiene ningún conflicto pendiente y está a la espera de validación o, por último, puede estar validado si ha sido validado por clientes y usuarios.

- **Estabilidad**.- Detalla la estabilidad del objetivo, es decir una estimación de la probabilidad de que pueda sufrir cambios en el futuro. Esta estabilidad puede indicarse mediante un valor numérico o mediante una expresión enumerada como alta, media o baja o PD en el caso de que aún no se haya determinado.

- **Comentarios**: Cualquier otra información sobre el objetivo que no encaje en los campos anteriores puede recogerse en este apartado.

- **Precondición**: Se expresan en lenguaje natural las condiciones necesarias para que se pueda realizar el caso de uso.

- **Secuencia normal**: Contiene la secuencia normal de interacciones del caso de uso. En cada paso, un actor o el sistema realiza una o más acciones, o se realiza (se incluye) otro caso de uso.

- **Postcondición**: Expresa en lenguaje natural las condiciones que se deben cumplir después de la terminación normal del caso de uso.

- **Extensiones**: Especifica el comportamiento del sistema en el caso de que se produzca alguna situación excepcional durante la realización de un paso determinado.

7.2. Descripción del problema

El Análisis de Riesgo se concibe como una herramienta de ayuda al control del riesgo. Permite la identificación de acciones encaminadas a la prevención de la ocurrencia de accidentes, así como la planificación de las actuaciones en caso de que se produzcan, con el fin de minimizar las consecuencias de la ocurrencia del citado accidente.

Las principales actividades que se realizan son las siguientes:

- La identificación de peligros de accidentes.

- El cálculo de consecuencias o de las zonas de riesgo según los valores umbrales.

- Cálculo de vulnerabilidad.

- Relación de accidentes graves identificados.

- Medidas de prevención, control y mitigación.

La persona encargada de realizar estas actividades, es el Analista de Riesgos.

7.2.1. Analista de Riesgos

Elabora el Análisis de Riesgo en base a simulaciones de modelos matemáticos para determinar las consecuencias producidas por la ocurrencia de un accidente.

Entrando en detalle, en primer lugar, el Analista visita el establecimiento industrial con el fin de identificar los elementos susceptibles de producir un accidente, y en el otro lado, los elementos a proteger, los recursos.

Tras la localización e identificación de los elementos, el Analista realiza simulaciones para determinar las consecuencias producidas por la ocurrencia de un accidente. Este procedimiento consiste en la ejecución del modelo matemático correspondiente al accidente posicionado sobre el plano que se simula. El valor resultante representa el área afectada por la consecuencia de producirse el accidente.

Un modelo matemático describe teóricamente un objeto de la realidad, en nuestro caso, el análisis de riesgos, representa una explosión o una fuga de alguna sustancia, o cualquier otra causa que pueda provocar la ocurrencia de un accidente. Cada modelo depende de un tipo de parámetros o de otros; por ejemplo, en el caso de una explosión depende del tipo de sustancia que explosione y de la presión a la que esté sometida, por decir algunos.

Tras la realización de la simulación, representa el resultado para determinar que elementos han sido afectados. Así como determina el grado de daño producido en ellos. Generalmente esta representación es realizada como una zona circular, denominada zona de riesgo.

Comúnmente para cada accidente, el Analista de Riesgos realiza varias simulaciones para obtener varias zonas de riesgo. Cada zona está relacionada con un efecto determinado, por ejemplo, distintos niveles de presión que generan consecuencias diversas en el ser humano, heridas de diferentes grados, aturdimiento, sordera, etc.

Las zonas de riesgo permiten al Analista definir medidas de prevención, control y mitigación de la ocurrencia de un accidente. Determinar actuaciones especificas; por ejemplo, cambiar de ubicación los elementos identificados, además de concretar el equipo necesario para estar dentro de una zona concreta.

Para finalizar, genera un informe donde detalla los elementos de la simulación, indicando las distintas zonas de riesgo de los accidentes, como los elementos que entran en cada una de ellas y las medidas de prevención, control y mitigación.

7.3. Modelo de dominio

Se muestra a continuación el modelo de dominio que se creó en la toma de requerimientos donde se reflejan los elementos conceptuales que utiliza el Analista de Riegos.

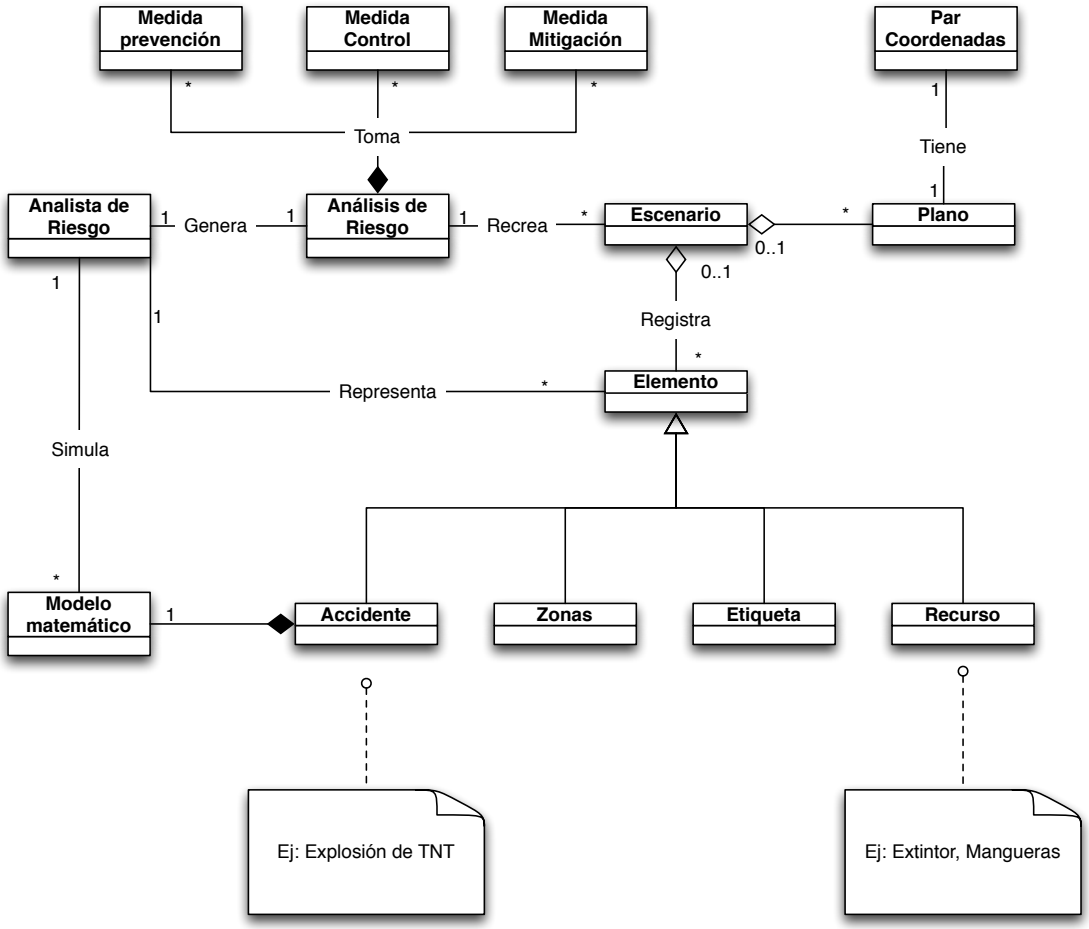

Ilustración 9: Diagrama del modelo del dominio

7.4. Objetivos del sistema

Se realiza una primera aproximación al problema estableciendo los objetivos principales del proyecto. Estos objetivos establecen la base necesaria para la especificación del resto de requisitos.

Objetivo 1.- Cargar planos

OBJ-01	Cargar planos
Descripción	El sistema deberá cargar planos en un sistema de coordenadas, que permita posicionar distintos elementos sobre él, tales como recursos o accidentes.
Estabilidad	Alta
Comentarios	Ninguno

Tabla 6: Objetivo 1.- Cargar planos

Objetivo 2.- Representar riesgos y recursos

OBJ-02	Representar accidentes y recursos
Descripción	El sistema deberá mostrar sobre el plano accidentes y recursos posicionados en coordenadas concretas.
Estabilidad	Alta
Comentarios	Ninguno

Tabla 7: Objetivo 2.- Representar riesgos y recursos

Objetivo 3.- Representar simulaciones

OBJ-03	Representar simulaciones
Descripción	El sistema deberá mostrar sobre el plano el resultado de realizar la simulación de un modelo matemático.
Estabilidad	Alta
Comentarios	Ninguno

Tabla 8: Objetivo 3.- Representar simulaciones

7.5. Requisitos de usuario

La captación de requisitos de usuario se desarrolla mediante la realización de diagramas de casos de uso. En ellos se plasman los distintos actores que participan en el sistema, junto a las principales acciones que realizan. Permiten obtener una visión global de las relaciones entre los actores y sus funciones.

Se especifican a continuación los actores que existen en el ámbito del problema, así como sus principales actividades al realizar un análisis de riesgos.

7.5.1. Actor.- Analista de Riesgos

Como se vio en la descripción del problema, en el análisis de riesgos, la utilización del visor de planos viene dada por un tipo de usuario, el Analista de Riesgos.

A continuación se listan las principales acciones que realiza el Analista de Riesgos.

Actor	Acciones principales.
Analista de Riesgos	Coge un plano. Mueve el plano. Cambia de plano. Posiciona accidentes o recursos. Simula un modelo matemático de un accidente. Genera un informe. Propone soluciones.

Tabla 9: Acciones básicas del Analista de Riesgos

Profundizando en las principales acciones del listado anterior, se obtiene una lista de acciones más atómicas que amplían el conjunto de acciones realizadas por el Analista de Riesgos.

Actor	Acción
Analista de Riesgos	Borra etiquetas.
	Borra accidentes.
	Borra etiquetas.
	Borra recursos.
	Borra simulaciones.
	Coge un plano.
	Coge un plano de nivel más cercano.
	Coge un plano de nivel más lejano.
	Delimita zonas.
	Escoge tipo de accidente.
	Establece las propiedades del recurso.
	Establece las propiedades del accidente.
	Establece tipo de recurso.
	Establece una leyenda de colores.
	Genera un informe de resultados.
	Marca zonas con una etiqueta.
	Mide distancias.
	Mueve el plano actual.
	Posiciona recursos.
	Posiciona accidentes.
	Representa la simulación del modelo matemático.
	Simula un modelo matemático.
	Ve el nivel del plano.
	Ve la leyenda de colores.

Tabla 10: Descripción ampliada de las acciones básicas del Analista de Riesgos

Siguiendo esta línea de ampliación de requisitos, a continuación se detallan los casos de uso que cubren estas acciones.

7.5.2. Modelo de Casos de Uso

Se detallan los casos de uso utilizando diagramas de UML y se especifican los mismos mediante las plantillas para la descripción de casos de uso.

Casos de uso de Requisitos de Usuario.- Manejo del plano

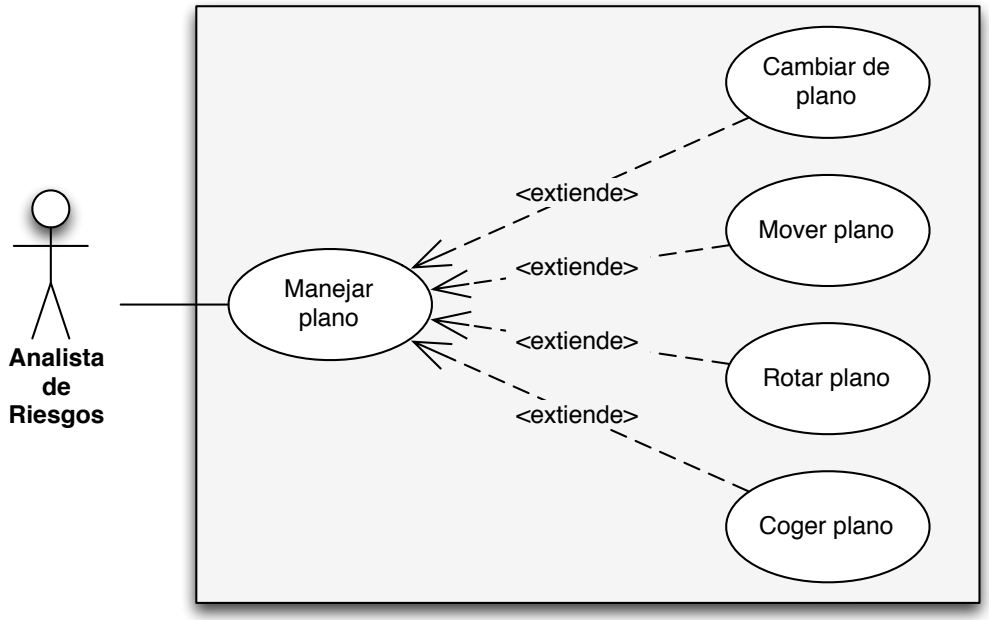

Ilustración 10: Casos de uso "Manejo del plano"

Caso de uso 01.- Manejar plano

RU-CU-01	Manejar plano
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-01.- Cargar planos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Manipula el plano hasta adaptarlo a sus necesidades.
Precondiciones:	El Analista de Riesgos va a realizar un análisis de riesgos de una zona determinada. Necesita su plano.
Garantías de éxito (Postcondiciones):	Selecciona el plano que quiere y lo adecúa a sus necesidades.
Escenario principal de éxito (o Flujo Básico):	1. El Analista selecciona el plano adecuado a una escala determinada, lo mueve, lo rota hasta dejarlo en la posición deseada.
Extensiones (o Flujos Alternativos):	1. Si es necesario el Analista cambia de plano para obtener otro nivel de detalles del mismo.

Tabla 11: RU-CU-01.- Manejar plano

Caso de uso 02.- Cambiar de plano

RU-CU-02	Cambiar de plano
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-01.- Cargar planos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Selecciona un plano a un nivel de escala diferente al actual, porque el actual no abarca todas las detalles necesarios.
Precondiciones:	El Analista necesita un plano diferente al actual con otro nivel de detalles.
Garantías de éxito (Postcondiciones):	Selecciona un plano diferente que se adapta a sus necesidades.
Escenario principal de éxito (o Flujo Básico):	1. El Analista observa que el plano actual no le sirve para sus necesidades actuales. 2. El Analista cambia a un plano con otro nivel de detalle del actual. 3. Guarda el plano anterior.
Extensiones (o Flujos Alternativos):	1. El Analista mantiene presente los 2 planos.

Tabla 12: RU-CU-02.- Cambiar de plano

Caso de uso 03.- Mover plano

RU-CU-03	Mover plano
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-01.- Cargar planos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Desplaza el plano hasta localizar su área de interés.
Precondiciones:	El Analista necesita tener el plano adecuado donde se encuentra el área determinada del plano.
Garantías de éxito (Postcondiciones):	Localiza la zona determinada.
Escenario principal de éxito (o Flujo Básico):	1. El Analista mueve el plano. 2. Localiza el área determinada
Extensiones (o Flujos Alternativos):	

Tabla 13: RU-CU-03.- Mover plano

Caso de uso 04.- Rotar plano

RU-CU-04	Rotar plano
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-01.- Cargar planos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Rota el plano hasta localizar y adecuar su área de interés a sus necesidades.
Precondiciones:	El Analista necesita tener el plano adecuado donde se encuentra el área determinada del plano.
Garantías de éxito (Postcondiciones):	Ubica el plano de forma adecuada a sus intereses.
Escenario principal de éxito (o Flujo Básico):	1. El Analista gira el plano.
Extensiones (o Flujos Alternativos):	

Tabla 14: RU-CU-04.- Rotar plano

Caso de uso 05.- Coger plano

RU-CU-05	Coger plano
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-01.- Cargar planos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Coge de su conjuntos de planos el adecuado para el análisis actual.
Precondiciones:	El Analista necesita un plano para realizar un análisis de riesgos.
Garantías de éxito (Postcondiciones):	Coge un plano de una zona
Escenario principal de éxito (o Flujo Básico):	1. El Analista coge un plano de su archivo de planos.
Extensiones (o Flujos Alternativos):	

Tabla 15: RU-CU-05.- Coger plano

Casos de uso de Requisitos de Usuario.- Posicionamiento y eliminación de elementos

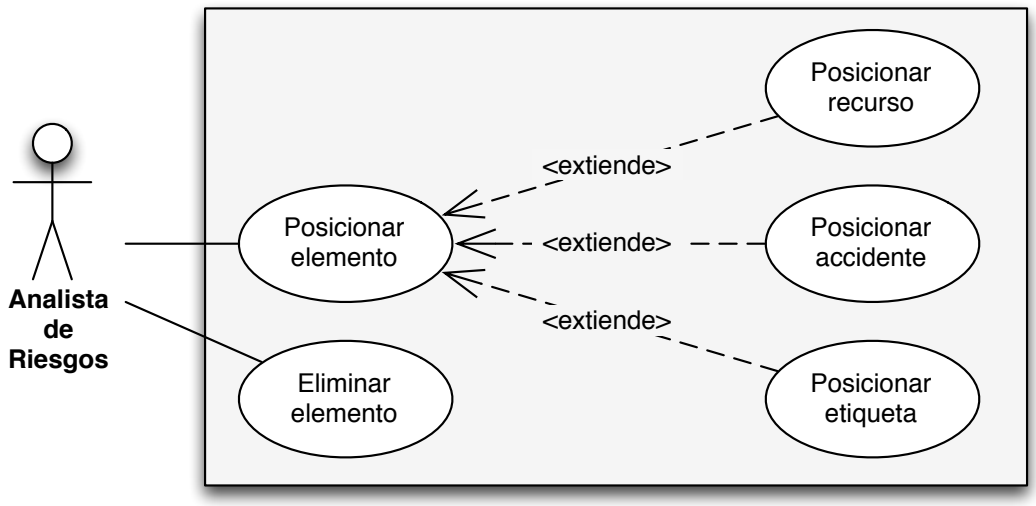

Ilustración 11: Casos de uso "Posicionamiento y eliminación de elementos"

Caso de uso 06.- Posicionar elemento

RU-CU-06	Posicionar elemento
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Representa distintos elementos sobre el plano.
Precondiciones:	El Analista ya ha cogido un plano y se ha ubicado en su zona de interés.
Garantías de éxito (Postcondiciones):	Posiciona sobre el plano distintos elementos.
Escenario principal de éxito (o Flujo Básico):	1. Determina la posición de distintos elementos existentes o nuevos elementos sobre el plano. 2. Representa con distintas formas y colores los distintos elementos.
Extensiones (o Flujos Alternativos):	1. La determinación de ciertos elementos puede estar condicionado a otros elementos ya situados. O a otras condiciones. P.ej. La representación de las simulaciones, está condicionada a la posición de los accidentes.

Tabla 16: RU-CU-06.- Posicionar elemento

Caso de uso 07.- Eliminar elemento

RU-CU-07	Eliminar elemento
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Determina que ciertos elementos no necesitan estar representados en el plano.
Precondiciones:	El Analista ya ha cogido un plano, ha ubicado su zona de interés y hay elementos representados que pueden ser eliminados.
Garantías de éxito (Postcondiciones):	Un elemento representado es eliminado, deja de ser representado del plano.
Escenario principal de éxito (o Flujo Básico):	1. El Analista localiza el elemento representado que quiere borrar. 2. Borra su representación.
Extensiones (o Flujos Alternativos):	1. Elimina un elemento que está vinculado a otros. 2. Es necesario eliminar esos elementos vinculados.

Tabla 17: RU-CU-07.- Eliminar elemento

Caso de uso 08.- Posicionar recurso

RU-CU-08	Posicionar recurso
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Posiciona un tipo de elementos, los recursos.
Precondiciones:	El Analista ya ha cogido un plano, ha ubicado su zona de interés y va a posicionar elementos susceptibles de ser afectados por la simulación de un accidente.
Garantías de éxito (Postcondiciones):	Se representa sobre el plano un recurso, elementos que pueden ser afectados por la simulación de un accidente. Por ejemplo un extintor, un acceso, etc.
Escenario principal de éxito (o Flujo Básico):	1. El Analista localiza la posición donde estará ubicado el recurso. 2. Escoge una representación gráfica , una forma, unos colores. 3. Dibuja el recurso.
Extensiones (o Flujos Alternativos):	

Tabla 18: RU-CU-08.- Posicionar recurso

Caso de uso 09.- Posicionar accidente

RU-CU-09	Posicionar accidente
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Posiciona un tipo de elementos, los accidentes.
Precondiciones:	El Analista ya ha cogido un plano, ha ubicado su zona de interés y va a posicionar elementos que pueden provocar accidentes. Estos elementos tienen asociados algoritmos para realizar simulaciones.
Garantías de éxito (Postcondiciones):	Se representa sobre el plano un accidente. Estos elementos tienen asociados algoritmos para realizar determinar su comportamiento. Por ejemplo, contenedores de explosivos, gases, etc.
Escenario principal de éxito (o Flujo Básico):	1. El Analista localiza la posición donde estará ubicado el accidente. 2. Escoge una representación gráfica , una forma, unos colores. 3. Dibuja el accidente
Extensiones (o Flujos Alternativos):	1. Tras realizar la simulación del riego, el Analista representa el resultado de la misma. Con una posición relativa a la posición inicial del accidente.

Tabla 19: RU-CU-09.- Posicionar accidente

Caso de uso 10.- Posicionar etiqueta

RU-CU-10	Posicionar etiqueta
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Posiciona un tipo de elementos, las etiquetas.
Precondiciones:	El Analista ya ha cogido un plano, ha ubicado su zona de interés y va a etiquetar elementos del plano.
Garantías de éxito (Postcondiciones):	Se representa sobre el plano una etiqueta. El Analista escribe sobre el plano el nombre de una localización, elementos móviles, etc.
Escenario principal de éxito (o Flujo Básico):	1. El Analista localiza la posición donde estará ubicado la etiqueta. 2. Determina que va a etiquetar 3. Escoge una representación gráfica , unos colores. 4. Escribe la etiqueta.
Extensiones (o Flujos Alternativos):	

Tabla 20: RU-CU-10.- Posicionar etiqueta

Casos de uso de Requisitos de Usuario.- Posicionamiento de recursos

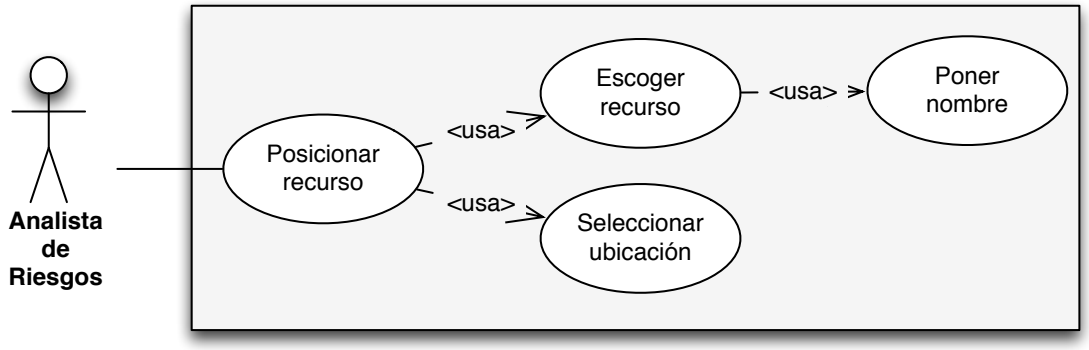

Ilustración 12: Casos de uso "Posicionamiento de recursos"

Caso de uso 11.- Escoger recurso

RU-CU-11	Escoger recurso
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Escoge un recurso del conjunto de recursos que el Analista necesita representar.
Precondiciones:	El Analista ya ha cogido un plano, ha ubicado su zona de interés y va a seleccionar un recurso a posicionar.
Garantías de éxito (Postcondiciones):	Selecciona el recurso que va a posicionar.
Escenario principal de éxito (o Flujo Básico):	1. El Analista elige de su conjunto de recursos a posicionar , el recurso que quiere elegir. 2. Escoge su representación gráfica. 3. Elige un nombre.
Extensiones (o Flujos Alternativos):	

Tabla 21: RU-CU-11.- Escoger recurso

Caso de uso 12.- Poner nombre

RU-CU-12	Poner nombre
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Etiqueta un recurso o un accidente con un nombre identificativo.
Precondiciones:	El Analista ha seleccionado un recurso para posicionarlo en el plano y necesita identificarlo.
Garantías de éxito (Postcondiciones):	El elemento tiene un nombre para poder ser identificado.
Escenario principal de éxito (o Flujo Básico):	1. El Analista etiqueta el elemento con un nombre identificativo.
Extensiones (o Flujos Alternativos):	

Tabla 22: RU-CU-12.- Poner nombre

Caso de uso 13.- Seleccionar ubicación

RU-CU-13	Seleccionar ubicación
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Selecciona un ubicación para un elemento que no tenga ubicación todavía, o al que quiera cambiarle la misma.
Precondiciones:	El Analista ha seleccionado un recurso y lo tiene etiquetado con su nombre.
Garantías de éxito (Postcondiciones):	El elemento está correctamente posicionado sobre el plano.
Escenario principal de éxito (o Flujo Básico):	1. El Analista elige la ubicación correspondiente para el elemento.
Extensiones (o Flujos Alternativos):	En caso de que tenga ya una ubicación: 1. Si es un accidente y está simulado, se desplaza también la representación de su simulación. 2. Sino se le cambia la ubicación a una posición correcta.

Tabla 23: RU-CU-13.- Seleccionar ubicación

Casos de uso de Requisitos de Usuario.- Posicionamiento de accidentes

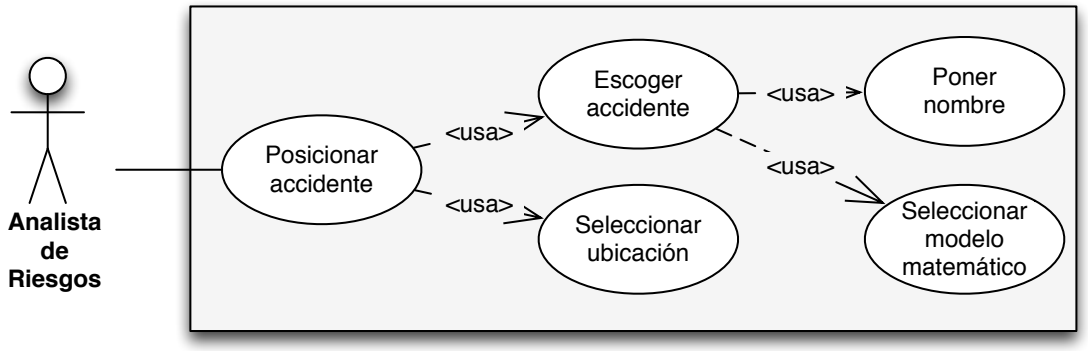

Ilustración 13: Casos de uso "Posicionamiento de accidentes"

Caso de uso 14.- Escoger accidente

RU-CU-14	Escoger accidente
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Escoge un tipo de accidente del conjunto de accidentes que el Analista necesita simular y representar.
Precondiciones:	El Analista ya ha cogido un plano, ha ubicado su zona de interés y va a seleccionar un accidente a posicionar.
Garantías de éxito (Postcondiciones):	Selecciona el tipo de accidente que va a posicionar.
Escenario principal de éxito (o Flujo Básico):	1. El Analista elige de su conjunto de accidentes a posicionar , el accidente que quiere elegir, así como su modelo matemático. 2. Selecciona sus propiedades. Estas dependen del modelo matemático elegido. 3. Escoge su representación gráfica. 4. Elige un nombre identificativo.
Extensiones (o Flujos Alternativos):	

Tabla 24: RU-CU-14.- Escoger accidente

Caso de uso 15.- Seleccionar modelo matemático

RU-CU-15	Seleccionar modelo matemático
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Escoge el modelo matemático aplicable al tipo de accidente que quiere representar.
Precondiciones:	El Analista ya ha cogido un plano, ha ubicado su zona de interés y ha elegido un tipo de accidente a representar, va a escoger un modelo matemático para ese accidente.
Garantías de éxito (Postcondiciones):	El accidente a posicionar tiene su modelo matemático seleccionado.
Escenario principal de éxito (o Flujo Básico):	1. Elige de sus modelos matemáticos el correspondiente al accidente que quiere representar. 2. Anota el modelo matemático asociado al accidente.
Extensiones (o Flujos Alternativos):	

Tabla 25: RU-CU-15.- Seleccionar modelo matemático

Casos de uso de Requisitos de Usuario.- Posicionamiento de etiquetas

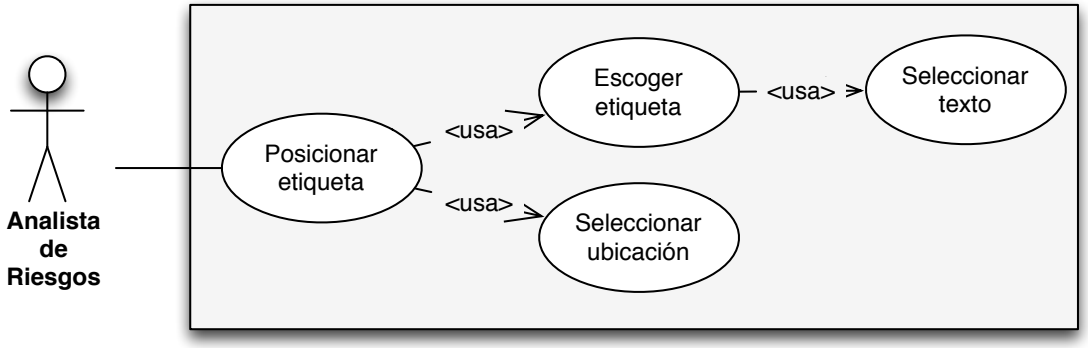

Ilustración 14: Casos de uso "Posicionamiento de etiquetas"

Caso de uso 16.- Escoger etiqueta

RU-CU-16	Escoger etiqueta
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Escoge un tipo de etiqueta para etiquetar sobre el plano una zona o un elemento dibujable.
Precondiciones:	El Analista ya ha cogido un plano, ha ubicado su zona de interés y va a escoger un tipo de etiqueta para escribirla sobre el plano.
Garantías de éxito (Postcondiciones):	Selecciona el tipo de etiqueta que va a posicionar.
Escenario principal de éxito (o Flujo Básico):	1. El Analista escoge un tipo de etiqueta 2. Escoge el texto a dibujar.
Extensiones (o Flujos Alternativos):	

Tabla 26: RU-CU-16.- Escoger etiqueta

Caso de uso 17.- Seleccionar texto

RU-CU-17	Seleccionar texto
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Escoge un tipo de etiqueta para etiquetar sobre el plano una zona o un elemento dibujable.
Precondiciones:	El Analista ya ha cogido un plano, ha ubicado su zona de interés y va a escoger un tipo de etiqueta para escribirla sobre el plano.
Garantías de éxito (Postcondiciones):	Escoge el texto que va a utilizar la etiqueta.
Escenario principal de éxito (o Flujo Básico):	1. El Analista escoge un tipo de etiqueta 2. Escoge el texto a dibujar.
Extensiones (o Flujos Alternativos):	

Tabla 27: RU-CU-17.- Seleccionar texto

Casos de uso de Requisitos de Usuario.- Análisis de simulación

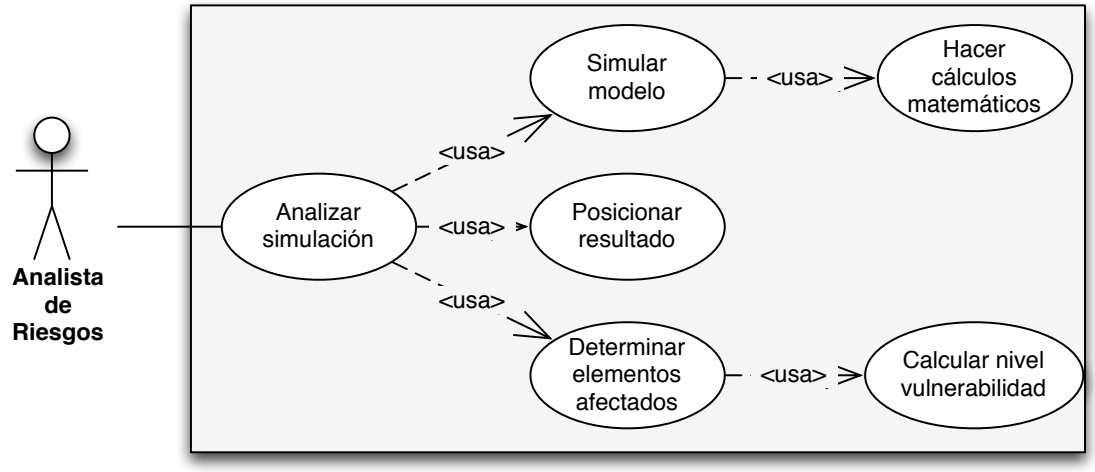

Ilustración 15: Casos de uso "Análisis de simulación"

Caso de uso 18.- Analizar simulación

RU-CU-18	Analizar simulación
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-03.- Representar simulaciones.
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Realiza sus estudios analizando los resultados de las simulaciones de los accidentes. Se determinan que elementos son afectados por la simulación para así, tomar decisiones a favor de la seguridad.
Precondiciones:	El Analista ya ha cogido un plano, se ha ubicado en su zona de interés y al menos tiene un accidente configurado con un modelo matemático.
Garantías de éxito (Postcondiciones):	Se obtiene el análisis de una situación planteada por la simulación de un accidente con su modelo matemático. Y se determinan que elementos son afectados por el accidente.
Escenario principal de éxito (o Flujo Básico):	1. El Analista simula el modelo matemático con los datos del problema. 2. Representa sobre el plano el resultado de la simulación. Habitualmente una zona de riesgo. 3. Determina los elementos afectados. 4. Genera un informe.
Extensiones (o Flujos Alternativos):	

Tabla 28: RU-CU-18.- Analizar simulación

Caso de uso 19.- Simular modelo

RU-CU-19	Simular modelo
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-03.- Representar simulaciones.
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Simula el modelo matemático para obtener un resultado, generalmente este resultado será un área. Este resultado es representado sobre el plano.
Precondiciones:	El Analista tiene accidentes con modelos matemáticos asociados situados sobre el plano.
Garantías de éxito (Postcondiciones):	Obtiene un resultado producido por la simulación de un modelo matemático ajustado a las propiedades del accidente.
Escenario principal de éxito (o Flujo Básico):	1. Realiza cálculos correspondientes al modelo matemático y a los datos correspondientes al accidente. 2. Obtiene el resultado de la simulación.
Extensiones (o Flujos Alternativos):	

Tabla 29: RU-CU-19.- Simular modelo

Caso de uso 20.- Hacer cálculos

RU-CU-20	Hacer cálculos matemáticos
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-03.- Representar simulaciones.
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Para simular el modelo matemático, el Analista necesita realizar cálculos matemáticos con los datos particulares de un accidente.
Precondiciones:	El Analista tiene accidentes con modelos matemáticos asociados situados sobre el plano. Ha procedido a realizar la simulación.
Garantías de éxito (Postcondiciones):	Obtiene un resultado producido por la simulación de un modelo matemático ajustado a las propiedades del accidente.
Escenario principal de éxito (o Flujo Básico):	1. Realiza cálculos matemáticos.
Extensiones (o Flujos Alternativos):	

Tabla 30: RU-CU-20.- Hacer cálculos

Caso de uso 21.- Posicionar resultado

RU-CU-21	Posicionar resultado
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-03.- Representar simulaciones.
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Tras realizar la simulación, el Analista representa sobre el plano el resultado de la misma.
Precondiciones:	El Analista tiene accidentes con modelos matemáticos asociados situados sobre el plano. Ha realizado la simulación
Garantías de éxito (Postcondiciones):	Representa y ubica un resultado producido por la simulación de un modelo matemático.
Escenario principal de éxito (o Flujo Básico):	1. Elige la configuración visual . 2. Posiciona sobre el plano el resultado.
Extensiones (o Flujos Alternativos):	

Tabla 31: RU-CU-21.- Posicionar resultado

Caso de uso 22.- Determinar elementos afectados

RU-CU-22	Determinar elementos afectados
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-03.- Representar simulaciones.
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Tras realizar la simulación y representar la misma, el Analista determina que elementos han sido afectados por el accidente. También determina cuanto ha sido afectado el elemento por el resultado de la simulación.
Precondiciones:	El Analista tiene accidentes con modelos matemáticos asociados situados sobre el plano. Ha realizado la simulación.
Garantías de éxito (Postcondiciones):	Lista el conjunto de elementos afectados y su grado de afección.
Escenario principal de éxito (o Flujo Básico):	1. Observa los elementos afectados por el área producida por la simulación. 2. Determina el grado de vulnerabilidad de cada elemento.
Extensiones (o Flujos Alternativos):	

Tabla 32: RU-CU-22.- Determinar elementos afectados

Caso de uso 23.- Calcular nivel de vulnerabilidad

RU-CU-23	Calcular nivel de vulnerabilidad
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-03.- Representar simulaciones.
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Para cada elemento que se encuentre dentro del área producida por una simulación determina su grado de vulnerabilidad.
Precondiciones:	El Analista tiene accidentes con modelos matemáticos asociados situados sobre el plano. Ha realizado la simulación y está observando un elemento que se encuentra dentro del área del resultado de la simulación.
Garantías de éxito (Postcondiciones):	Determina cuanto ha sido afectado un elemento por una simulación.
Escenario principal de éxito (o Flujo Básico):	1. Determina el grado de afección de cada elemento.
Extensiones (o Flujos Alternativos):	

Tabla 33: RU-CU-23.- Calcular nivel de vulnerabilidad

Casos de uso de Requisitos de Usuario.- Generación de informe

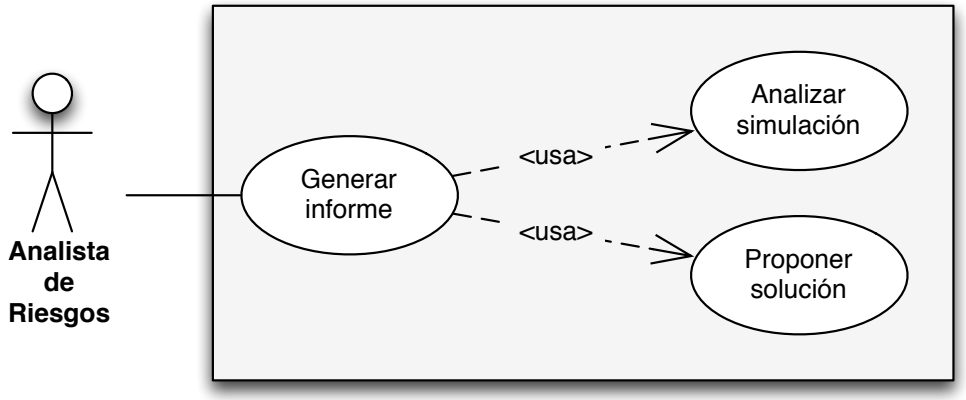

Ilustración 16: Casos de uso "Generación de informe"

Caso de uso 24.- Generar informe

RU-CU-24	Generar informe
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Con la información recabada de la simulación, el Analista genera un informe.
Precondiciones:	El Analista ha realizado una simulación y ha listado que elementos son afectados por la misma.
Garantías de éxito (Postcondiciones):	Genera un informe completo con la información de las correspondientes simulaciones, así como los elementos afectados, además añade posibles soluciones.
Escenario principal de éxito (o Flujo Básico):	1. Recolecta toda la información previa de simulaciones, los distintos elementos. 2. Analiza las consecuencias de la simulación. 3. Propone soluciones.
Extensiones (o Flujos Alternativos):	

Tabla 34: RU-CU-24.- Generar informe

Caso de uso 25.- Proponer solución

RU-CU-25	Proponer solución
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	
Actor Principal:	Analista de Riesgos.
Personal involucrado e intereses:	• Analista de Riesgos: Con la información recabada de una simulación, el Analista piensa una solución.
Precondiciones:	El Analista ha realizado una simulación y ha listado que elementos son afectados por la misma.
Garantías de éxito (Postcondiciones):	Obtiene un conjunto variado de soluciones .
Escenario principal de éxito (o Flujo Básico):	1. Analiza la información de las simulaciones, así como los elementos afectados. 2. Toma decisiones para crear una solución que mejore la situación en la zona tras un accidente como el producido por la simulación.
Extensiones (o Flujos Alternativos):	

Tabla 35: RU-CU-25.- Proponer solución

7.6. Requisitos software

Los requisitos vistos en el apartado anterior están basados en el principal caso de uso del visor, la creación de un Análisis de Riesgo. Se ha de tener presente que el objetivo fundamental del proyecto es el desarrollo de un visor genérico orientado a las simulaciones, que pueda ser utilizado en un conjunto amplio de proyectos de simulación.

Los requerimientos establecidos anteriormente son la base para la especificación que se desarrolla en esta sección. El objetivo es encontrar una especificación de requisitos del producto software que sea eficaz para el Análisis de Riesgo y no limite su utilización en otros desarrollos. En definitiva, una descripción completa de los requisitos que debe cumplir el componente, esta vez sí, orientado al software.

7.6.1. Requisitos de usuario descartados

Para comenzar, se listan los casos de uso vistos anteriormente que no tienen cabida en este proyecto. Son aquellos relacionados con el Análisis de Riesgo, pero que no están ligados con el uso del componente visor.

Caso de uso	Descripción
RU-CU-15	Seleccionar modelo matemático.
RU-CU-19	Simular modelo.
RU-CU-20	Hacer cálculos matemáticos.
RU-CU-23	Calcular nivel de vulnerabilidad.
RU-CU-25	Analizar simulación.
RU-CU-26	Proponer solución.

Tabla 36: Requisitos de usuario descartados para la descripción de requisitos de software

A continuación se realiza una especificación similar a la sección anterior, donde se identifican actores y casos de uso enfocados al uso del software.

7.6.2. Actores del sistema

La utilización del componente visor de planos viene dada por dos tipos de actores. El actor que hace uso del visor integrándolo en su aplicación, *el Diseñador*, y el actor que hace uso de la aplicación final que incorpora el visor y que en definitiva, lo utiliza, *el Usuario*.

A continuación se describe de forma general el uso que le dará cada tipo de actor.

Actor	Uso
Usuario	Desplazar el plano. Escalar el plano. Manejar el tiempo. Posicionar elementos en el plano. Ver el plano.
Diseñador	Configurar el visor a sus necesidades. Configurar componentes visibles del visor. Integrar el visor de planos en su aplicación Representar elementos en el plano.

Tabla 37: Acciones básicas realizadas en el sistema por los actores Usuario y Diseñador

Los roles identificados se solapan en ciertas funcionalidades, tanto el Diseñador como el Usuario pueden representar elementos sobre el plano. La diferencia radica en el modo de realizar y obtener los resultados. Mientras que el Usuario normal actúa mediante las interfaces de entrada de su sistema para indicar la ubicación de la entidad, el Diseñador realiza estas acciones mediante la utilización de la librería de funciones facilitada por el componente.

A continuación, partiendo de los casos de uso generales, se ha procedido a profundizar en los mismos descomponiendo los anteriores en casos de uso más específicos para cada actor.

7.6.3. Actor.- Usuario

El Usuario hace uso de las características accesibles del visor desde la entrada básica del PC, es decir, el ratón y el teclado. Realiza las siguientes acciones:

Actor	Acción
Usuario	Ampliar el nivel de zoom.
	Cambia el nombre de la capa.
	Cambiar el instante de tiempo actual.
	Crear capa.
	Eliminar capa.
	Eliminar etiquetas.
	Eliminar figuras geométricas.
	Eliminar imágenes.
	Eliminar nombre a figura.
	Insertar etiquetas.
	Insertar figuras geométricas.
	Insertar imágenes.
	Insertar nombre a figura.
	Mover el plano.
	Ocultar capas.
	Reducir el nivel de zoom.
	Seleccionar figuras o imágenes.
	Ver el plano.
	Ver las coordenadas actuales.
	Ver la simulación.
	Ver tiempo inicial, actual y final.

Tabla 38: Descripción ampliada de las acciones realizadas por el actor Usuario

7.6.4. Actor.- Diseñador

El Diseñador integra el componente visor adaptándolo a sus necesidades, por ello este actor tiene unos casos de uso diferentes al Usuario. Requiere una funcionalidad diferente, orientada a la adaptación y configuración del visor a las necesidades del problema que trate su aplicación.

Otra manera de ver al Diseñador, es como el sistema que integra al componente visor, hace uso de las funciones que éste da, y es informado de lo que ocurre en el visor.

El diseñador incorpora el visor a su aplicación adaptándolo a las necesidades de su aplicación. Como tal, es quién hace uso del conjunto de funciones suministrada.

Realiza las siguientes acciones:

Actor	Acción
Diseñador	Activar herramientas visibles al visor. Cargar un plano. Configurar el número de niveles de escala. Configurar el tiempo actual. Configurar el tiempo inicial y final. Configurar comportamiento figuras. Configurar estética figuras. Configurar herramientas visibles al visor. Configurar la estética del visor. Crear capas. Crear figuras. Desactivar herramientas visibles al visor. Eliminar capas. Eliminar figuras. Eliminar figuras. Eliminar imágenes. Eliminar texto. Establecer el sistema de coordenadas. Insertar imágenes. Insertar texto. Insertar un instante futuro para una figura. Obtener información de lo que el usuario realiza. Posicionar un plano.

Tabla 39: Descripción ampliada de las acciones realizadas por el actor Diseñador

Siguiendo esta línea de profundización, a continuación se detallan los casos de uso.

7.6.5. Modelo de Casos de Uso

Se describe la interacción de los dos actores anteriores con el componente en los siguientes casos de uso.

Casos de uso de Requisitos del Software.- Integración

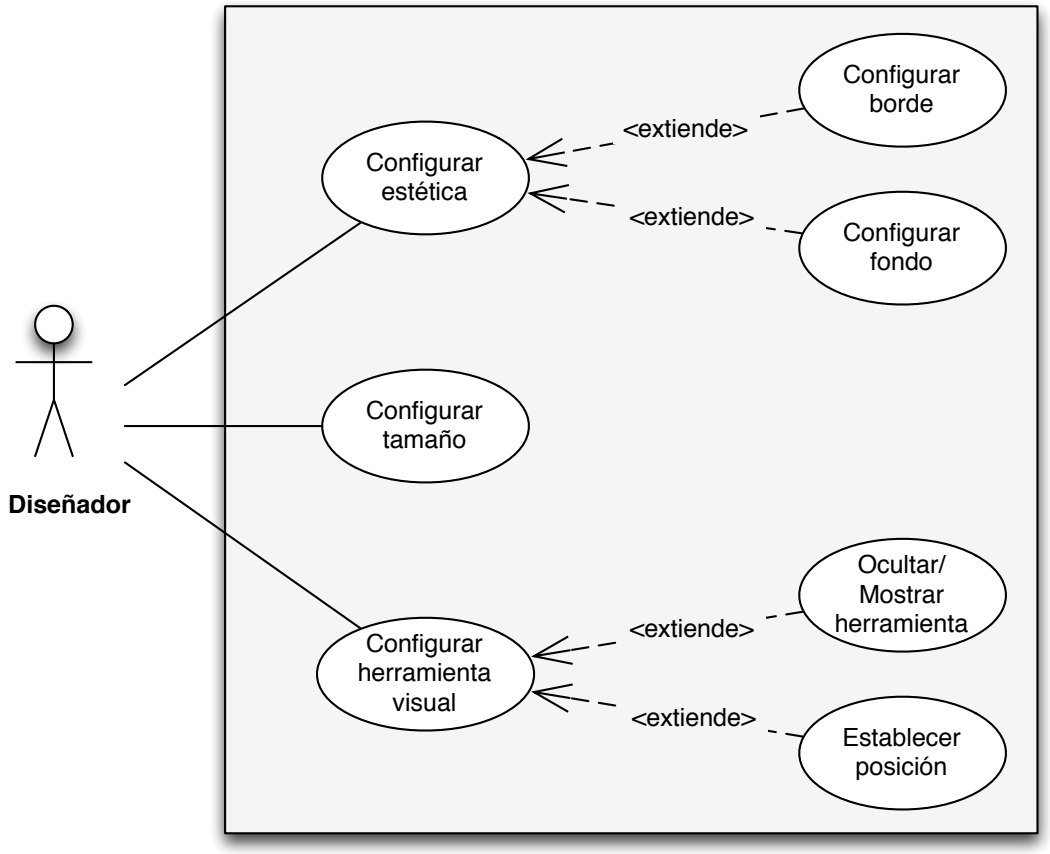

Ilustración 17: Casos de uso "Integración"

Caso de uso 01.- Configurar estética

RS-CU-01	Configurar estética
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Integra el componente visor y le da una apariencia estética semejante al resto de componentes que contiene.
Precondiciones:	El Diseñador ya ha declarado e instanciado el componente.
Garantías de éxito (Postcondiciones):	El componente visor muestra la apariencia que el Diseñador ha establecido.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador configura el borde exterior del componente. 2. El Diseñador configura la apariencia interior del contenido que se muestra en el visor.
Extensiones (o Flujos Alternativos):	1. La configuración puede ser realizada en cualquier momento.

Tabla 40: RS-CU-01.- Configurar estética

Caso de uso 02.- Configurar borde

RS-CU-02	Configurar borde
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Integra el componente visor y le da una apariencia estética semejante al resto de componentes que contiene. En este caso de uso ajusta la apariencia del borde.
Precondiciones:	El Diseñador ya ha declarado e instanciado el componente.
Garantías de éxito (Postcondiciones):	El componente visor muestra la apariencia del borde que el Diseñador ha configurado.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador define el tamaño del borde. 2. El Diseñador define el color del borde.
Extensiones (o Flujos Alternativos):	1. La configuración puede ser realizada en cualquier momento.

Tabla 41: RS-CU-02.- Configurar borde

Caso de uso 03.- Configurar fondo

RS-CU-03	Configurar fondo
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Integra el componente visor y le da una apariencia estética semejante al resto de componentes que contiene. En este caso de uso ajusta la apariencia del fondo del visor.
Precondiciones:	El Diseñador ya ha declarado e instanciado el componente.
Garantías de éxito (Postcondiciones):	El componente visor muestra la apariencia del fondo que el Diseñador le ha configurado.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador configura el color de fondo del visor. 2. El Diseñador define un texto y ajusta sus propiedades (texto, color, tamaño).
Extensiones (o Flujos Alternativos):	1. La configuración puede ser realizada en cualquier momento.

Tabla 42: RS-CU-03.- Configurar fondo

Caso de uso 04.- Configurar tamaño

RS-CU-04	Configurar tamaño
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Integra el componente visor, para ello le da un tamaño al visor, lo suficientemente grande como para incluir las herramientas visuales que necesita.
Precondiciones:	El sistema ya cuenta con la declaración e instanciación del componente.
Garantías de éxito (Postcondiciones):	El componente visor tiene el tamaño adecuado.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador ajusta el ancho y alto del componente.
Extensiones (o Flujos Alternativos):	1. La configuración del tamaño puede ser realizada en cualquier momento.

Tabla 43: RS-CU-04.- Configurar tamaño

Caso de uso 05.- Configurar herramienta visual

RS-CU-05	Configurar herramienta visual
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Integra el componente visor. Establece las herramientas visuales que quiere proporcionar al usuario.
Precondiciones:	El sistema ya cuenta con la declaración e instanciación del componente.
Garantías de éxito (Postcondiciones):	El componente visor dispone de las herramientas visuales que el Diseñador quiere suministrar, estableciendo además su ubicación.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador establece que herramientas necesita el Usuario. 2. Ajusta su posición.
Extensiones (o Flujos Alternativos):	1. La configuración puede ser realizada en cualquier momento, no solo el inicial. Posible conjunto de herramientas (Visor de capas, de tiempo, de escala, de coordenadas, etc.)

Tabla 44: RS-CU-05.- Configurar herramienta visual

Caso de uso 06.- Ocultar/Mostrar herramienta

RS-CU-06	Ocultar/Mostrar herramienta
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Integra el componente visor. Está configurando las herramientas que quiere proporcionar al usuario.
Precondiciones:	El sistema ya cuenta con la declaración e instanciación del componente.
Garantías de éxito (Postcondiciones):	El componente visor dispone de las herramientas visuales que el Diseñador quiere suministrar, ocultando aquellas que no son necesarias en su entorno.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador oculta o muestra la herramienta escogida.
Extensiones (o Flujos Alternativos):	1. Se puede ocultar/mostrar las herramientas en cualquier momento.

Tabla 45: RS-CU-06.- Ocultar/Mostrar herramienta

Caso de uso 07.- Establecer posición

RS-CU-07	Establecer posición
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Integra el componente visor. Está configurando donde quiere situar las herramientas que quiere proporcionar al usuario.
Precondiciones:	El sistema ya cuenta con la declaración e instanciación del componente.
Garantías de éxito (Postcondiciones):	El componente visor dispone de las herramientas visuales en la posición que el Diseñador requiere.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador adecúa la posición de las herramientas a su especificación.
Extensiones (o Flujos Alternativos):	1. El Diseñador puede cambiar la posición de las herramientas en cualquier momento.

Tabla 46: RS-CU-07.- Establecer posición

Casos de uso de Requisitos del Software.- Carga del plano

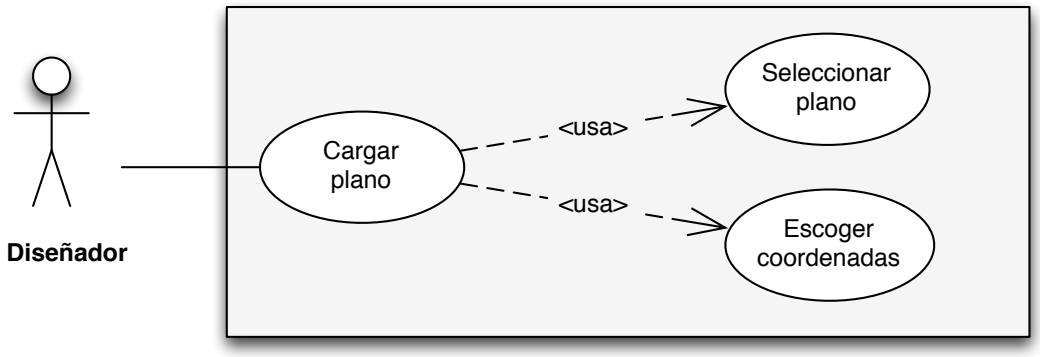

Ilustración 18: Casos de uso "Carga del plano"

Caso de uso 08.- Cargar plano

RS-CU-08	Cargar plano
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-01.- Cargar planos
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Tras integrar y configurar el visor, quiere cargar un plano, una imagen representativa.
Precondiciones:	El visor ya está integrado y configurado, y no hay ningún plano cargado previamente.
Garantías de éxito (Postcondiciones):	El componente visor muestra el plano que el Diseñador quiere cargar.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador selecciona el origen del plano, seleccionando una url o la propia imagen. 2. Para cargar el plano, el Diseñador indica en que coordenadas se encuentra el mismo.
Extensiones (o Flujos Alternativos):	1. Puede cargar imágenes con formatos usuales, jpg y png.

Tabla 47: RS-CU-08.- Cargar plano

Caso de uso 09.- Seleccionar plano

RS-CU-09	Seleccionar plano
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-01.- Cargar planos
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Tras integrar y configurar el visor, quiere cargar un plano.
Precondiciones:	El visor ya está integrado y configurado, y no hay ningún plano cargado previamente.
Garantías de éxito (Postcondiciones):	El componente visor muestra el plano que el Diseñador quiere cargar.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador debe seleccionar que imagen cargar, bien seleccionando su url o bien la propia imagen.
Extensiones (o Flujos Alternativos):	

Tabla 48: RS-CU-09.- Seleccionar plano

Caso de uso 10.- Escoger coordenadas

RS-CU-10	Escoger coordenadas
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-01.- Cargar planos
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Tras integrar y configurar el visor, quiere cargar un plano. Debe introducir las coordenadas asociadas al mismo. Las utilizará como sistema de coordenadas para la representación de entidades.
Precondiciones:	El visor ya está integrado y configurado, y no hay ningún plano cargado previamente. Las coordenadas deben ser positivas.
Garantías de éxito (Postcondiciones):	El componente visor muestra el plano que el Diseñador quiere cargar.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador introduce al visor el conjunto de coordenadas mínimas y máximas donde está posicionado el mapa.
Extensiones (o Flujos Alternativos):	

Tabla 49: RS-CU-10.- Escoger coordenadas

Casos de uso de Requisitos del Software.- Configuración de los sistemas

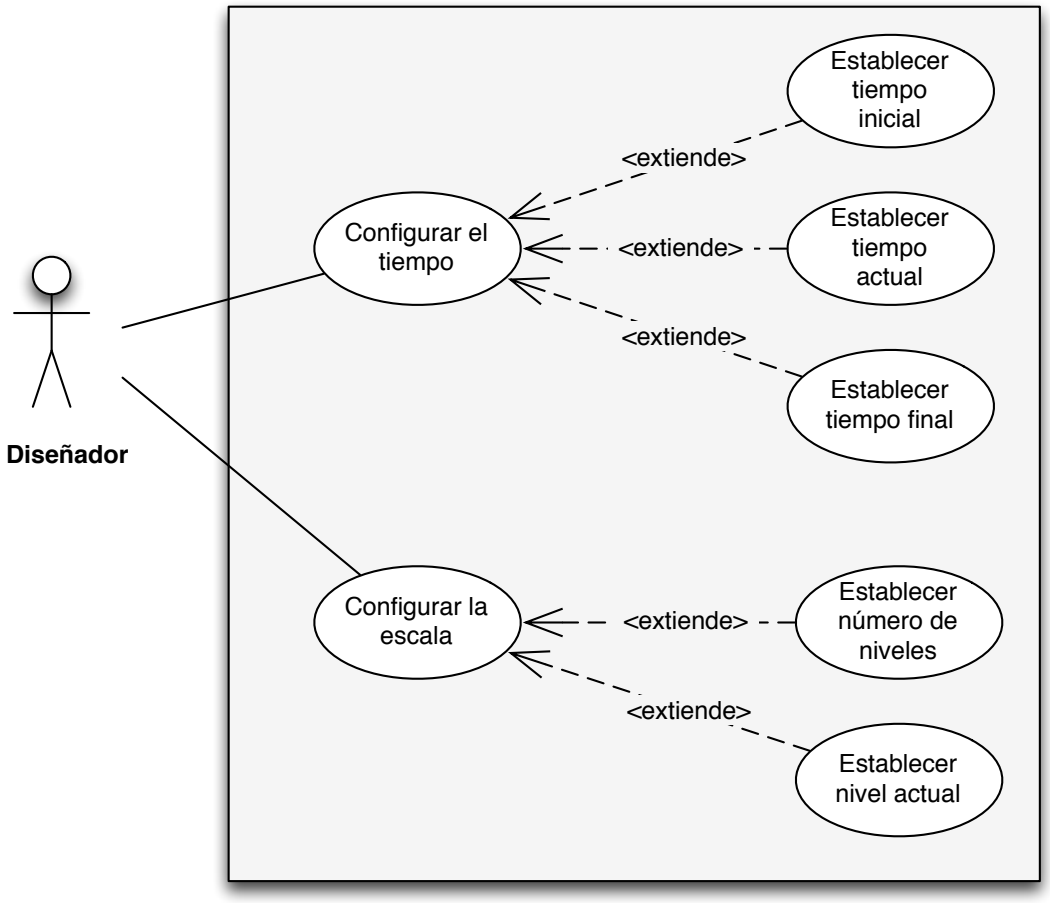

Ilustración 19: Casos de uso "Configuración de los sistemas"

Caso de uso 11.- Configurar el tiempo

RS-CU-11	Configurar el tiempo
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-03.- Representar simulaciones.
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Tras cargar un plano, el Diseñador configura el sistema de tiempo. Se gestiona el tiempo de forma que existe un tiempo inicial y un final, así como un instante actual.
Precondiciones:	Se ha cargado un plano en el visor, el Diseñador quiere configurar el mismo con los valores de tiempo adecuados para su simulación. Los valores de tiempo deben ser positivos
Garantías de éxito (Postcondiciones):	El componente visor establece unos tiempos iniciales y finales para la representación de entidades. Finalmente configura el tiempo actual de representación del componente.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador introduce el valor de tiempo inicial. 2. Introduce el valor de tiempo final. 3. Introduce el valor de tiempo actual.
Extensiones (o Flujos Alternativos):	1. El Diseñador puede cambiar estos valores en cualquier momento, adecuándolos a cualquier cambio posible en su

	simulación.

Tabla 50: RS-CU-11.- Configurar el tiempo

Caso de uso 12.- Establecer tiempo inicial

RS-CU-12	Establecer tiempo inicial
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-03.- Representar simulaciones.
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Una vez se ha cargado un plano, el Diseñador configura el sistema de tiempo. El Diseñador cambia el instante de tiempo inicial para ampliar o reducir su simulación.
Precondiciones:	Se ha cargado un plano en el visor, el Diseñador quiere configurar la escena con los valores de tiempo adecuados para su simulación. El valor de tiempo inicial debe ser positivo y menor o igual que el instante de tiempo final.
Garantías de éxito (Postcondiciones):	El componente visor establece un tiempo inicial válido.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador introduce el valor de tiempo inicial.
Extensiones (o Flujos Alternativos):	1. El Diseñador puede cambiar este valor en cualquier momento, adecuándolo a cualquier cambio posible en su simulación. 2. En caso de que el tiempo actual sea menor que el inicial, se corrige el actual y se establece igual al tiempo inicial.

Tabla 51: RS-CU-12.- Establecer el tiempo inicial

Caso de uso 13.- Establecer tiempo actual

RS-CU-13	Establecer tiempo actual
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-03.- Representar simulaciones.
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Tras la carga de un plano, el Diseñador configura el sistema de tiempo. Cambia el instante de tiempo actual.
Precondiciones:	Se ha cargado un plano en el visor, el Diseñador quiere configurar la escena con los valores de tiempo adecuados para su simulación. El valor de tiempo actual debe ser positivo y mayor o igual que el instante de tiempo inicial. Y menor o igual que el instante de tiempo final.
Garantías de éxito (Postcondiciones):	El componente visor establece un tiempo actual válido
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador introduce el valor de tiempo actual.
Extensiones (o Flujos Alternativos):	1. El Diseñador puede cambiar este valor en cualquier momento, adecuándolo a cualquier cambio posible en su simulación.

Tabla 52: RS-CU-13.- Establecer tiempo actual

Caso de uso 14.- Establecer tiempo final

RS-CU-14	Establecer tiempo final
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-03.- Representar simulaciones.
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Tras la carga de un plano, el Diseñador configura el sistema de tiempo. El Diseñador cambia el instante de tiempo final para ampliar o reducir su simulación.
Precondiciones:	Se ha cargado un plano en el visor, el Diseñador quiere configurar el mismo con los valores de tiempo adecuados para su simulación. El valor de tiempo final debe ser positivo y mayor o igual que el instante de tiempo inicial.
Garantías de éxito (Postcondiciones):	El componente visor establece un tiempo final válido.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador introduce el valor de tiempo final.
Extensiones (o Flujos Alternativos):	1. El Diseñador puede cambiar este valor en cualquier momento, adecuándolo a cualquier cambio posible en su simulación. 2. En caso de que el tiempo actual sea mayor que el final, se corrige el actual y se establece igual al tiempo final.

Tabla 53: RS-CU-14.- Establecer tiempo final

Caso de uso 15.- Configurar la escala

RS-CU-15	Configurar la escala
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Tras la carga de un plano, el Diseñador configura el sistema de escala. Se especifica el número de niveles de escala necesarios en el visor.
Precondiciones:	Se ha cargado un plano en el visor, el Diseñador quiere configurar la vista del mismo.
Garantías de éxito (Postcondiciones):	El componente visor muestra el plano como desea.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador introduce el número de niveles de visión del plano. 2. Especifica el nivel actual de zoom.
Extensiones (o Flujos Alternativos):	1. El Diseñador puede cambiar este valor en cualquier momento.

Tabla 54: RS-CU-15.- Configurar la escala

Caso de uso 16.- Establecer el número de niveles

RS-CU-16	Establecer el número de niveles
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Tras la carga de un plano, el Diseñador configura el sistema de escala. Especifica el número de niveles de escala necesarios en el visor.
Precondiciones:	Se ha cargado un plano en el visor, el Diseñador quiere configurar la vista del mismo.
Garantías de éxito (Postcondiciones):	El componente visor dispone del número de niveles de escala establecido por el visor.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador introduce el número de niveles de visión del plano.
Extensiones (o Flujos Alternativos):	1. El Diseñador puede cambiar este valor en cualquier momento.

Tabla 55: RS-CU-16.- Establecer el número de niveles

Caso de uso 17.- Establecer nivel actual

RS-CU-17	Establecer el nivel actual
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Tras la carga de un plano, el Diseñador configura el sistema de escala. Especifica el nivel actual de zoom del plano.
Precondiciones:	Se ha cargado un plano en el visor, el Diseñador quiere configurar la vista del mismo.
Garantías de éxito (Postcondiciones):	El componente visor muestra el plano como desea.
Escenario principal de éxito (o Flujo Básico):	1. Especifica el nivel actual de zoom.
Extensiones (o Flujos Alternativos):	1. El Diseñador puede cambiar este valor en cualquier momento.

Tabla 56: RS-CU-17.- Establecer nivel actual

Casos de uso de Requisitos del Software.- Identificación de información visual

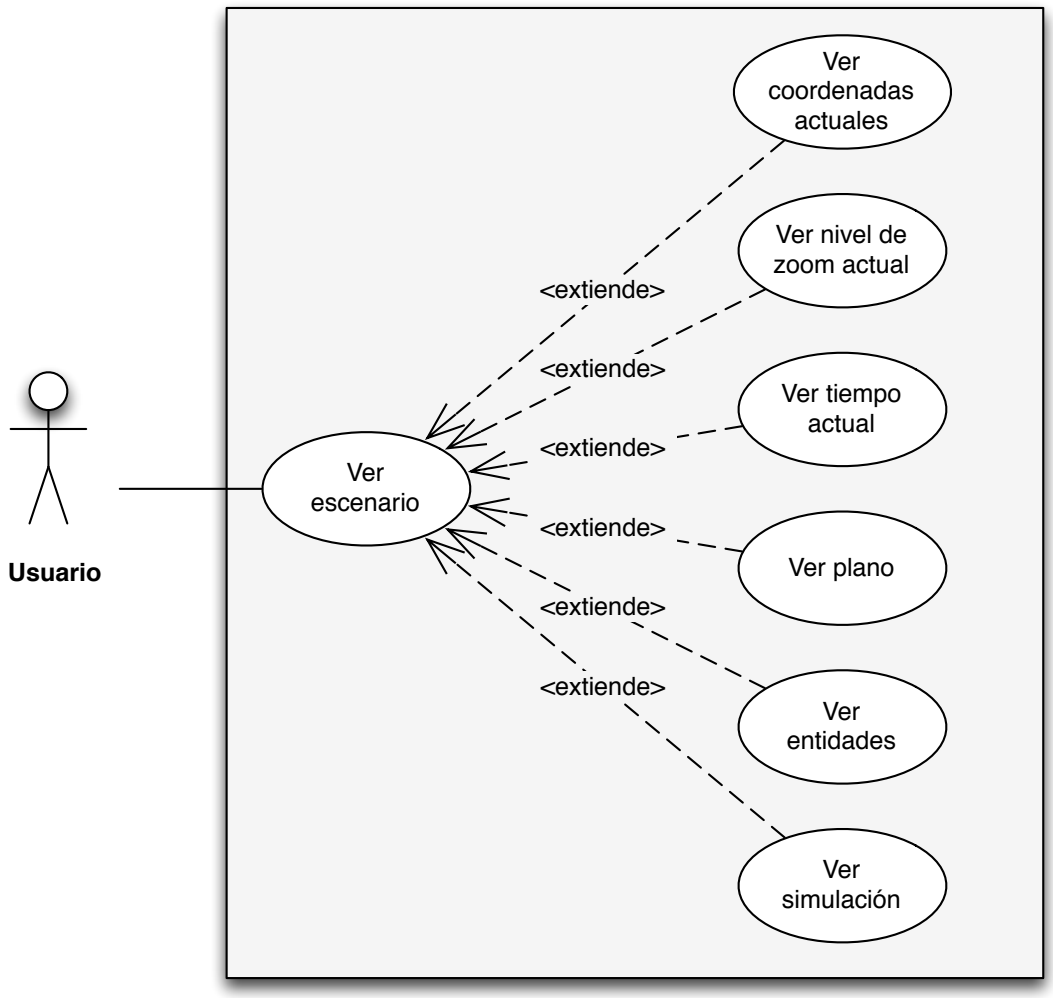

Ilustración 20: Casos de uso "Identificación de información visual"

Caso de uso 18.- Ver escenario

RS-CU-18	Ver escenario
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario
Personal involucrado e intereses:	• Usuario: El componente visor muestra un entorno donde se representan distintos tipos de datos de utilidad para el Usuario.
Precondiciones:	Se ha cargado un plano en el visor.
Garantías de éxito (Postcondiciones):	El Usuario obtiene información visual.
Escenario principal de éxito (o Flujo Básico):	1. El Usuario ve todos los aspectos del escenario planteado.
Extensiones (o Flujos Alternativos):	

Tabla 57: RS-CU-18.- Ver escenario

Caso de uso 19.- Ver coordenadas actuales

RS-CU-19	Ver coordenadas actuales
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario
Personal involucrado e intereses:	• Usuario: Desea ver las coordenadas actuales de lo que observa.
Precondiciones:	Se ha cargado un plano en el visor.
Garantías de éxito (Postcondiciones):	El Usuario obtiene información visual sobre las coordenadas de una zona del mapa.
Escenario principal de éxito (o Flujo Básico):	1. El Usuario a simple vista obtiene las coordenadas de una zona determinada.
Extensiones (o Flujos Alternativos):	

Tabla 58: RS-CU-19.- Ver coordenadas actuales

Caso de uso 20.- Ver nivel de zoom actual

RS-CU-20	Ver nivel de zoom actual
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario
Personal involucrado e intereses:	• Usuario: Se ubica en el nivel actual de escala, obtiene cuantos niveles de escala le quedan por aumentar o por disminuir.
Precondiciones:	Se ha cargado un plano en el visor.
Garantías de éxito (Postcondiciones):	El Usuario obtiene información visual sobre el número de niveles de escala que hay, y del nivel actual
Escenario principal de éxito (o Flujo Básico):	1. El Usuario a simple vista obtiene el nivel actual de escala.
Extensiones (o Flujos Alternativos):	

Tabla 59: RS-CU-20.- Ver nivel de zoom actual

Caso de uso 21.- Ver tiempo actual

RS-CU-21	Ver tiempo actual
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario
Personal involucrado e intereses:	• Usuario: Conoce en que momento de una simulación se encuentra.
Precondiciones:	Se ha cargado un plano en el visor.
Garantías de éxito (Postcondiciones):	El Usuario se ubica en el instante de tiempo actual en el que se encuentra.
Escenario principal de éxito (o Flujo Básico):	1. El Usuario a simple vista obtiene el tiempo actual

Extensiones (o Flujos Alternativos):	

Tabla 60: RS-CU-21.- Ver tiempo actual

Caso de uso 22.- Ver plano

RS-CU-22	Ver plano
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario
Personal involucrado e intereses:	• Usuario: El usuario ve el plano donde se van a producir simulaciones.
Precondiciones:	Se ha cargado un plano en el visor.
Garantías de éxito (Postcondiciones):	El Usuario ve el plano cargado, donde se representan distintas entidades.
Escenario principal de éxito (o Flujo Básico):	1. El Usuario a simple vista ve el plano.
Extensiones (o Flujos Alternativos):	

Tabla 61: RS-CU-22.- Ver plano

Caso de uso 23.- Ver entidades

RS-CU-23	Ver entidades
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario
Personal involucrado e intereses:	• Usuario: Ve la información representada del escenario, distintas entidades posicionadas por él o por el Diseñador.
Precondiciones:	Se ha cargado un plano en el visor.
Garantías de éxito (Postcondiciones):	Las entidades deben mostrar un look lo suficientemente llamativo para destacar sobre el plano.
Escenario principal de éxito (o Flujo Básico):	1. El Usuario a simple vista identifica las entidades representadas en el escenario.
Extensiones (o Flujos Alternativos):	

Tabla 62: RS-CU-23.- Ver entidades

Caso de uso 24.- Ver simulación

RS-CU-24	Ver simulación
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario
Personal involucrado e intereses:	• Usuario: Ve la representación de los resultados producidos por las distintas simulaciones realizadas en el escenario.

Precondiciones:	Se ha cargado un plano en el visor.
Garantías de éxito (Postcondiciones):	Las simulaciones deben mostrar un look lo suficientemente atractivas para destacar sobre el plano.
Escenario principal de éxito (o Flujo Básico):	1. El Usuario a simple vista identifica los resultados de las simulaciones.
Extensiones (o Flujos Alternativos):	

Tabla 63: RS-CU-24.- Ver simulación

Casos de uso de Requisitos del Software.- Manejo del plano

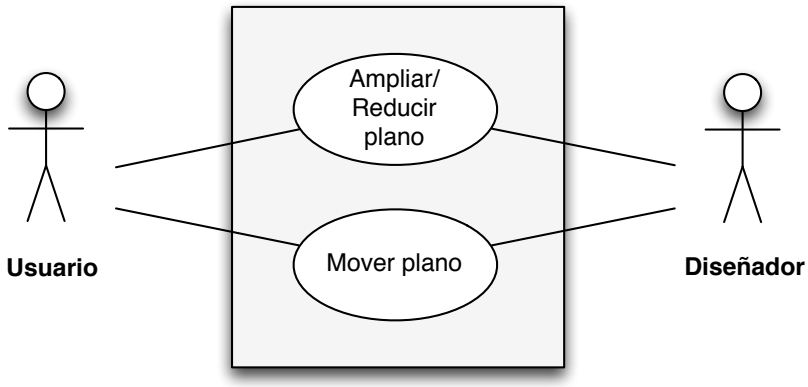

Ilustración 21: Casos de uso "Manejo del plano"

Caso de uso 25.- Ampliar/Reducir plano

RS-CU-25	Ampliar/Reducir plano
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Cambia el nivel de escala para obtener un nivel mayor de detalle sobre el plano. • Diseñador: Cambia el nivel de escala para mostrar detalles de alguna entidad en concreto.
Precondiciones:	Debe de haber más de 1 nivel de escala.
Garantías de éxito (Postcondiciones):	Se realiza muestra una vista del plano y de sus entidades y simulaciones correspondiente a la ampliación o reducción del nivel de escala.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. El Usuario cambia a través de una herramienta visual o mediante la entrada del sistema(ratón), el nivel de escala actual. Diseñador 1. El Diseñador introduce en el componente el nivel de escala que quiere utilizar.
Extensiones (o Flujos Alternativos):	

Tabla 64: RS-CU-25.- Ampliar/Reducir plano

Caso de uso 26.- Mover plano

RS-CU-26	Mover plano
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Cambia el nivel de escala para obtener un nivel mayor de detalle sobre el plano. • Diseñador: Cambia el nivel de escala para mostrar detalles de alguna entidad en concreto.
Precondiciones:	Debe de haber más de 1 nivel de escala.
Garantías de éxito (Postcondiciones):	Se realiza muestra una vista del plano y de sus entidades y simulaciones correspondiente a la ampliación o reducción del nivel de escala.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. El Usuario cambia a través de una herramienta visual o mediante la entrada del sistema(ratón), el nivel de escala actual. Diseñador 1. El Diseñador introduce en el componente el nivel de escala que quiere utilizar.
Extensiones (o Flujos Alternativos):	

Tabla 65: RS-CU-26.- Mover plano

Casos de uso de Requisitos del Software.- Manejo del tiempo

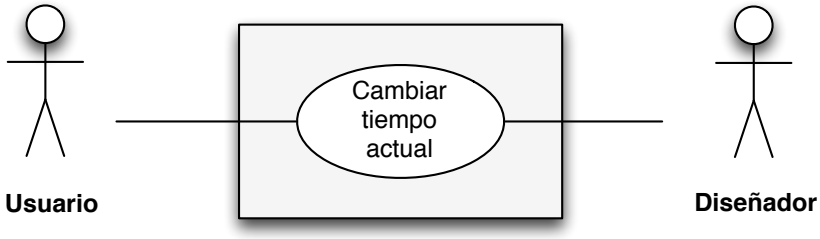

Ilustración 22: Casos de uso "Manejo del tiempo"

Caso de uso 27.- Cambiar tiempo actual

RS-CU-27	Cambiar tiempo actual
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Cambia el instante de tiempo actual para ver la evolución de una simulación o el desplazamiento de una entidad. • Diseñador: Cambia el instante de tiempo actual para realizar una reproducción de una simulación.
Precondiciones:	El número de instantes de tiempo es mayor que 1.
Garantías de éxito (Postcondiciones):	Se representa en el visor el conjunto de entidades y simulaciones correspondientes al instante cambiado.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. El Usuario cambia a través de una herramienta visual, el instante de tiempo actual. Diseñador 1. El Diseñador introduce en el componente el tiempo que quiere mostrar.
Extensiones (o Flujos Alternativos):	

Tabla 66: RS-CU-27.- Cambiar tiempo actual

Casos de uso de Requisitos del Software.- Manejo de las capas

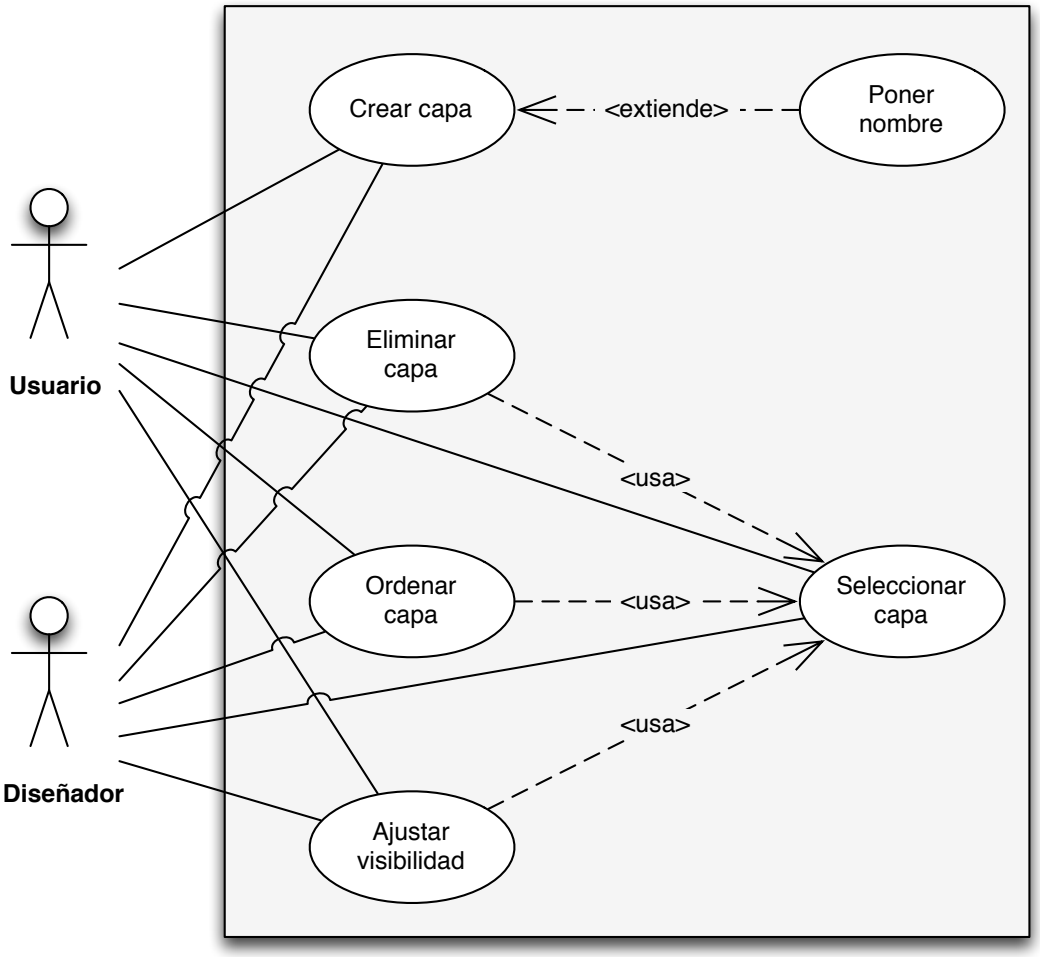

Ilustración 23: Casos de uso "Manejo de las capas"

Caso de uso 28.- Crear capa

RS-CU-28	Crear capa
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Crea una capa para organizar lógicamente las entidades que necesita representar. • Diseñador: Categoriza sus distintos tipos de entidades en capas. De forma que en un capa representa entidades y en otras simulaciones.
Precondiciones:	Debe existir un plano representado.
Garantías de éxito (Postcondiciones):	Existe una capa lógica disponible para la representación de entidades.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. El Usuario crea a través de una herramienta visual una capa. 2. Le especifica a la capa un nombre. Diseñador 1. El Diseñador crea una capa especificándole un nombre, y una visibilidad.
Extensiones (o Flujos Alternativos):	

Tabla 67: RS-CU-28.- Crear capa

Caso de uso 29.- Poner nombre

RS-CU-29	Poner nombre
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: A nivel visual el Usuario requiere tener las capas identificadas por un nombre. • Diseñador: Da un nombre ilustrativo a la capa para facilitar la comprensión del Usuario al manejar las capas.
Precondiciones:	Debe existir un plano representado.
Garantías de éxito (Postcondiciones):	La capa tiene el nuevo nombre.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. El Usuario selecciona a través de la herramienta visual de capas la capa a cambiar el nombre. 2. Le especifica a la capa un nombre. Diseñador 1. El Diseñador indica al componente la capa a la que debe cambiar el nombre.
Extensiones (o Flujos Alternativos):	

Tabla 68: RS-CU-29.- Poner nombre

Caso de uso 30.- Eliminar capa

RS-CU-30	Eliminar capa
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Elimina una capa y las entidades representadas en la mismo. • Diseñador: Elimina una capa y las entidades representadas en la misma.
Precondiciones:	Debe existir la capa con las entidades representadas.
Garantías de éxito (Postcondiciones):	Las entidades pertenecientes a la capa son eliminadas de la representación.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. El Usuario selecciona a través de la herramienta visual de capas la capa a eliminar. 2. Elimina la capa. Diseñador 1. El Diseñador indica al componente que debe borrar la capa indicada con un identificador.
Extensiones (o Flujos Alternativos):	

Tabla 69: RS-CU-30.- Eliminar capa

Caso de uso 31.- Seleccionar capa

RS-CU-31	Seleccionar capa
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Selecciona una capa para su reordenamiento, su cambio de visibilidad o su eliminación. • Diseñador: Selecciona una capa para realizar una operación sobre ella, reordenarla, ajustar su visibilidad, o eliminarla.
Precondiciones:	Debe existir al menos una capa.
Garantías de éxito (Postcondiciones):	La capa es seleccionada.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. El Usuario selecciona a través de la herramienta visual de capas la capa. Diseñador 1. El Diseñador indica al componente que capa hay que seleccionar. Si no existe no se selecciona.
Extensiones (o Flujos Alternativos):	

Tabla 70: RS-CU-31.- Seleccionar capa

Caso de uso 32.- Ordenar capa

RS-CU-32	Ordenar capa
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Ordena una capa. Permite poner entidades encima o debajo de otras dependiendo de la posición de la capa que lo contiene. • Diseñador: Ordena una capa para mostrar entidades encima de otras.
Precondiciones:	Debe existir al menos dos capas.
Garantías de éxito (Postcondiciones):	La representación visual mostrada en el visor obtiene una vista de las capas con la nueva posición.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. El Usuario selecciona a través de la herramienta visual de capas la capa a ordenar. 2. Arrastra la capa a su nueva posición. Diseñador 1. El Diseñador indica al componente que debe cambiar de posición la capa indicada por un identificador.
Extensiones (o Flujos Alternativos):	

Tabla 71: RS-CU-32.- Ordenar capa

Caso de uso 33.- Ajustar visibilidad

RS-CU-33	Ajustar visibilidad
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Muestra u oculta capas • Diseñador: Muestra u oculta capas para mostrar lo que desee.
Precondiciones:	Debe existir la capa.
Garantías de éxito (Postcondiciones):	En la vista aparecen las capas que son visibles.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. El Usuario selecciona a través de la herramienta visual de capas la que quiere ver u ocultar. 2. Toca una opción para mostrar u ocultar la capa. Diseñador 1. El Diseñador indica al componente que debe mostrar u ocultar la capa indicada por un identificador.
Extensiones (o Flujos Alternativos):	

Tabla 72: RS-CU-33.- Ajustar visibilidad

Casos de uso de Requisitos del Software.- Representación de entidades

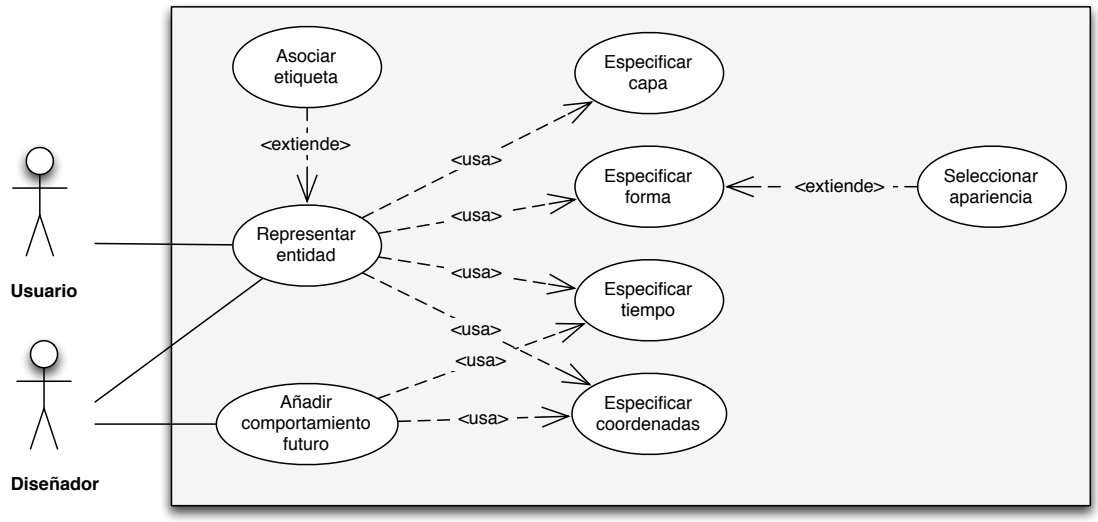

Ilustración 24: Casos de uso "Representación de entidades"

Caso de uso 34.- Representar entidad

RS-CU-34	Representar entidad
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Registra una entidad en el visor. • Diseñador: Registra una entidad en el visor, puede ser el resultado de una simulación.
Precondiciones:	Debe estar seleccionada una capa.
Garantías de éxito (Postcondiciones):	Se registra la entidad y se muestra su representación en el visor.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. El Usuario selecciona una entidad a posicionar, esta se refleja en una forma básica o en una imagen. 2. Indica la coordenada donde se posiciona la nueva entidad. 3. El componente visor registra y muestra la entidad en el instante de tiempo actual y en la capa seleccionada. 4. Puede especificar una etiqueta a la entidad. Diseñador 1. El Diseñador registra en el componente las entidades especificando la capa, las coordenadas, el tipo de forma a utilizar y el instante de tiempo en el que se encuentra. 2. Puede especificar una etiqueta a la entidad.
Extensiones (o Flujos Alternativos):	

Tabla 73: RS-CU-34.- Representar entidad

Caso de uso 35.- Especificar coordenadas

RS-CU-35	Especificar coordenadas
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Se seleccionan unas coordenadas pertenecientes al plano cargado. • Diseñador: Especifica unas coordenadas guardadas o calculadas con algún proceso pertenecientes al plano cargado.
Precondiciones:	Las coordenadas deben estar en el rango de coordenadas en las que se encuentra el plano.
Garantías de éxito (Postcondiciones):	Se obtienen unas coordenadas válidas para la representación de una entidad.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. Indica la coordenada donde se posiciona la nueva entidad. Diseñador 1. Especifica la coordenada donde se ubica la entidad.
Extensiones (o Flujos Alternativos):	

Tabla 74: RS-CU-35.- Especificar coordenadas

Caso de uso 36.- Especificar forma

RS-CU-36	Especificar forma
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Utiliza la forma asociada a la entidad que representa. • Diseñador: Representa sus entidades y resultados con una forma determinada. Esta forma puede configurar su apariencia con una configuración dada por el Diseñador.
Precondiciones:	Se deben utilizar formas suministradas por el componente visor.
Garantías de éxito (Postcondiciones):	La representación de la entidad utiliza la forma escogida.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. Escoge un tipo de entidad a representar. 2. Se representa la entidad con la forma asociada. Diseñador 1. Especifica la forma a utilizar en la representación de la entidad o en el resultado de una simulación. 2. La forma se configura con una serie de características básicas, colores de relleno, de borde, tamaños de borde, etc.
Extensiones (o Flujos Alternativos):	

Tabla 75: RS-CU-36.- Especificar forma

Caso de uso 37.- Seleccionar apariencia

RS-CU-37	Seleccionar apariencia
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Utiliza la apariencia asociada a la entidad que representa. • Diseñador: Representa sus entidades y resultados con una apariencia determinada.
Precondiciones:	
Garantías de éxito (Postcondiciones):	La representación de la forma utiliza la apariencia escogida.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. Se representa la forma con la apariencia asociada. Diseñador 1. Especifica los valores de las propiedades de la apariencia a utilizar en la representación de la entidad o en el resultado de una simulación. Estas propiedades pueden ser colores de relleno, de borde, tamaños de borde, etc.
Extensiones (o Flujos Alternativos):	

Tabla 76: RS-CU-37.- Seleccionar apariencia

Caso de uso 38.- Asociar etiqueta

RS-CU-38	Asociar etiqueta
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Asocia a las entidades representadas en el visor una etiqueta identificativa.
Precondiciones:	La entidad debe existir.
Garantías de éxito (Postcondiciones):	La entidad escogida tiene asociada una etiqueta.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. Se representa la forma con la etiqueta asociada. Diseñador 1. Registra para una entidad registrada un texto con un máximo de 30 caracteres. 2. El componente muestra la representación de la etiqueta.
Extensiones (o Flujos Alternativos):	

Tabla 77: RS-CU-38.- Asociar etiqueta

Caso de uso 39.- Especificar capa

RS-CU-39	Especificar capa
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Para poder representar una entidad especifica la capa en la que quiere registrar la entidad. • Diseñador: Especifica la capa donde se van a registrar las entidades.
Precondiciones:	Debe existir el identificador de la capa.
Garantías de éxito (Postcondiciones):	Si existe el identificador de la capa, se selecciona.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. El usuario selecciona una capa en una herramienta visual en el componente. Diseñador 1. Al registrar la entidad indica la capa donde se registra la entidad.
Extensiones (o Flujos Alternativos):	

Tabla 78: RS-CU-39.- Especificar capa

Caso de uso 40.- Añadir comportamiento futuro

RS-CU-40	Añadir comportamiento futuro
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Tras realizar una simulación, representa el resultado de la misma en un instante de tiempo futuro.
Precondiciones:	Debe existir una figura inicial, a la que se asocie un comportamiento futuro.
Garantías de éxito (Postcondiciones):	Si existe la figura inicial, se le asocia un comportamiento futuro.
Escenario principal de éxito (o Flujo Básico):	1. Selecciona la figura a la que quiere añadir el comportamiento futuro. 2. Inserta un comportamiento futuro indicando: a. Nuevas coordenadas b. Instante de tiempo futuro
Extensiones (o Flujos Alternativos):	1. Si existe ya un comportamiento futuro para una figura dada en el instante de tiempo seleccionado, se elimina el comportamiento previo.

Tabla 79: RS-CU-40.- Añadir comportamiento futuro

Caso de uso 41.- Especificar tiempo

RS-CU-41	Especificar tiempo
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Usuario, Diseñador
Personal involucrado e intereses:	• Usuario: Al posicionar una entidad utiliza el instante de tiempo actual. • Diseñador: Para representar cualquier entidad, indica el instante de tiempo en el que se representa.
Precondiciones:	El tiempo seleccionado debe existir, entre el instante de tiempo inicial y el final
Garantías de éxito (Postcondiciones):	Se selecciona el instante de tiempo indicado.
Escenario principal de éxito (o Flujo Básico):	Usuario 1. Utiliza el instante de tiempo actual en el visor. Diseñador 1. Indica un instante de tiempo perteneciente al intervalo entre tiempo inicial y el final.
Extensiones (o Flujos Alternativos):	

Tabla 80: RS-CU-41.- Especificar tiempo

Casos de uso de Requisitos del Software.- Modificación/Eliminación entidades

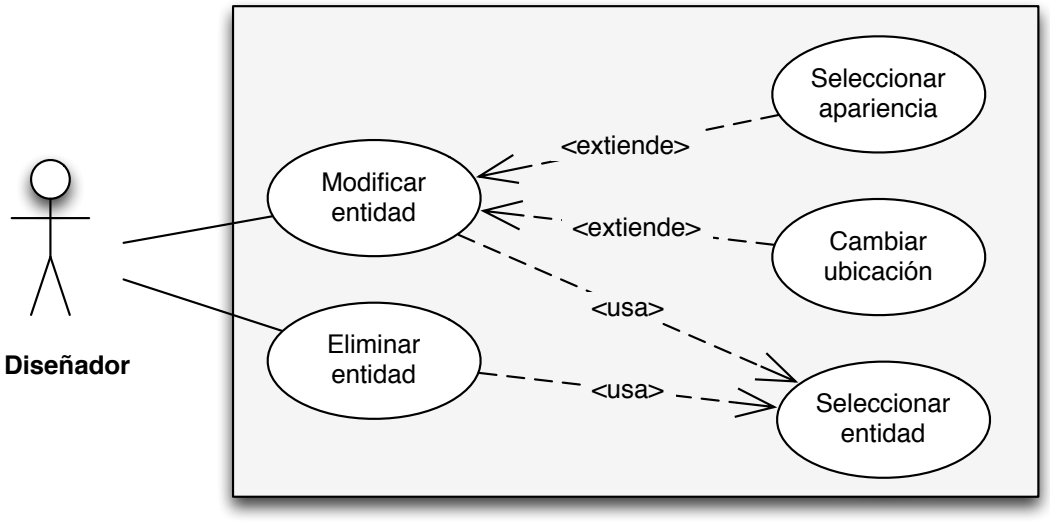

Ilustración 25: Casos de uso "Modificación/Eliminación entidades"

Caso de uso 42.- Modificar entidad

RS-CU-42	Modificar entidad
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Desea modificar la apariencia o la ubicación de una entidad.
Precondiciones:	Debe existir la entidad que se quiere modificar.
Garantías de éxito (Postcondiciones):	La entidad tendrá una nueva apariencia o una nueva posición.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador indica la entidad a modificar. 2. Establece los nuevos valores de apariencia que requiere. 3. Establece las nuevas coordenadas de la entidad.
Extensiones (o Flujos Alternativos):	

Tabla 81: RS-CU-42.- Modificar entidad

Caso de uso 43.- Eliminar entidad

RS-CU-43	Eliminar entidad
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Desea eliminar una entidad del escenario.
Precondiciones:	Debe existir la entidad que se quiere eliminar.
Garantías de éxito (Postcondiciones):	El escenario planteado no tendrá la entidad seleccionada.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador selecciona la entidad a eliminar. 2. El sistema la elimina.
Extensiones (o Flujos Alternativos):	1. Si no se encuentra la entidad no ocurre nada.

Tabla 82: RS-CU-43.- Eliminar entidad

Caso de uso 44.- Seleccionar entidad

RS-CU-44	Seleccionar entidad
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Selecciona una entidad para modificarla o eliminarla.
Precondiciones:	Deben existir entidades.
Garantías de éxito (Postcondiciones):	Se selecciona la entidad requerida.
Escenario principal de éxito (o Flujo Básico):	1. El Diseñador indica el identificador de la entidad. 2. Indica además el instante de tiempo en el que busca la entidad. 3. Si existe en el componente, se selecciona y se devuelve un valor positivo.
Extensiones (o Flujos Alternativos):	1. Si no se encuentra la entidad se indica con un valor negativo.

Tabla 83: RS-CU-44.- Seleccionar entidad

Caso de uso 45.- Cambiar ubicación

RS-CU-45	Cambiar ubicación
Versión:	1.0
Autor	Samuel Díaz Cabrera
Objetivos asociados:	Obj-02.- Representar accidentes y recursos. Obj-03.- Representar simulaciones.
Actor Principal:	Diseñador
Personal involucrado e intereses:	• Diseñador: Modifica una entidad cambiando su ubicación.
Precondiciones:	Deben existir la entidad a cambiar su ubicación.
Garantías de éxito (Postcondiciones):	La entidad seleccionada tiene unas coordenadas nuevas.
Escenario principal de éxito (o Flujo Básico):	1. Se cambian las coordenadas actuales de la entidad por unas nuevas
Extensiones (o Flujos Alternativos):	

Tabla 84: RS-CU-45.- Cambiar ubicación

7.7. Requisitos no funcionales

Son características requeridas del sistema, del proceso de desarrollo, del servicio prestado o de cualquier otro aspecto del desarrollo, que señala una restricción del mismo.

Estos requisitos se establecieron al comienzo del desarrollo del proyecto. Establecieron pautas básicas que debía cumplir el sistema software a desarrollar.

Requisito no funcional 01.- Desarrollo de un Componente integrable en aplicaciones web.

RNF -01	Desarrollo de un Componente integrable en aplicaciones web
Versión	1.0
Autores	Samuel Díaz Cabrera
Fuentes	Samuel Díaz Cabrera
Objetivos asociados	
Requisitos asociados	
Descripción	El sistema deberá ser un componente integrable dentro de una aplicación web.
Importancia	Vital
Urgencia	10
Estado	Realizado
Estabilidad	Alta
Comentarios	El proyecto ha cumplido este requisito no funcional.

Tabla 85: RNF-01.- Desarrollo de un componente integrable en aplicaciones web

Requisito no funcional 02.- Aplicación Rica en Internet (RIA)

RNF -02	Aplicación Rica en Internet
Versión	1.0
Autores	Samuel Díaz Cabrera
Fuentes	Samuel Díaz Cabrera
Objetivos asociados	
Requisitos asociados	
Descripción	Debido al creciente aumento de estas aplicaciones en Internet, la tecnología a usar está condicionada a ser una tecnología potente y con un amplio futuro de utilización en entornos online.
Importancia	Vital
Urgencia	10
Estado	Realizado
Estabilidad	Alta
Comentarios	Se ha establecido que el lenguaje a utilizar sea ActionScript, para desarrollar una aplicación en Flash.

Tabla 86: RNF-02.- Aplicación Rica en Internet (RIA)

Requisito no funcional 03.- Compatibilidad con navegadores

RNF -03	Compatibilidad con navegadores
Versión	1.0
Autores	Samuel Díaz Cabrera
Fuentes	Samuel Díaz Cabrera
Objetivos asociados	
Requisitos asociados	
Descripción	La aplicación debe ser compatible en el mayor número de navegadores para que pueda ser ejecutada sobre cualquier plataforma.
Importancia	Vital
Urgencia	10
Estado	Realizado
Estabilidad	Alta
Comentarios	Las aplicaciones desarrolladas con tecnología de RIAs son altamente compatibles con los navegadores más populares del mercado. Ya se ha establecido la tecnología a utilizar para llevar a cabo el proyecto, utilizándose Adobe Flash.

Tabla 87: RNF-03.- Compatibilidad con navegadores

Requisito no funcional 04.- Portabilidad

RNF -04	Portabilidad
Versión	1.0
Autores	Samuel Díaz Cabrera
Fuentes	Samuel Díaz Cabrera
Objetivos asociados	
Requisitos asociados	
Descripción	Se requiere que el componente sea ejecutable con independencia del sistema operativo.
Importancia	Vital
Urgencia	10
Estado	Realizado
Estabilidad	Alta
Comentarios	Al utilizar tecnología de RIAs y estar basadas estas en la ejecución de programas en navegadores web, ya se ha conseguido cumplir este requisito.

Tabla 88: RNF-04.- Portabilidad

Requisito no funcional 05.- Reusable

RNF -05	Reusable
Versión	1.0
Autores	Samuel Díaz Cabrera
Fuentes	Samuel Díaz Cabrera
Objetivos asociados	
Requisitos asociados	
Descripción	Los componentes son usualmente diseñados para ser utilizados en escenarios diferentes por diferentes aplicaciones, sin embargo, algunos componentes pueden ser diseñados para tareas específicas. Este es nuestro caso, donde se diseñará el componente como un visor de planos para cualquier tipo de ámbito.
Importancia	Vital
Urgencia	10
Estado	Realizado
Estabilidad	Alta
Comentarios	Es una característica imprescindible de los componentes.

Tabla 89: RNF-05.- Reusable

Requisito no funcional 06.- Facilidad de desarrollo

RNF -06	Facilidad de desarrollo
Versión	1.0
Autores	Samuel Díaz Cabrera
Fuentes	Samuel Díaz Cabrera
Objetivos asociados	
Requisitos asociados	
Descripción	Los componentes implementan interfaces bien definidas para proveer la funcionalidad definida permitiendo el desarrollo sin impactar otras partes del sistema. El componente se desarrollará procurando mantener una estructura que facilite el mantenimiento del código y de futuras ampliaciones.
Importancia	Vital
Urgencia	10
Estado	Realizado
Estabilidad	Alta
Comentarios	Sin comentarios.

Tabla 90: RNF-06.- Facilidad de desarrollo

Requisito no funcional 07.- Proporcionar una API funcional

RNF - 07	Proporcionar una API funcional
Versión	1.0
Autores	Samuel Díaz Cabrera
Fuentes	Samuel Díaz Cabrera
Objetivos asociados	
Requisitos asociados	
Descripción	Junto al componente debe suministrarse una API funcional para que el Diseñador utilice las funciones necesarias.
Importancia	Alta
Urgencia	Baja
Estado	Desarrollado
Estabilidad	
Comentarios	La api se suministra en un formato sólido(papel) y en uno lógico(digital)

Tabla 91: RNF-07.- Proporcionar una API funcional

Con los requerimientos establecidos, se transforman los mismos en el análisis, a objetos del análisis, donde se realiza una descripción de los mismos en un lenguaje más técnico.

Capítulo 8.- Análisis

Tras la captura de requerimientos se desarrolla un refinamiento y una estructuración de los mismos. El objetivo es conseguir una comprensión más precisa y una descripción fácil de mantener que ayude a estructurar el sistema completo. En esta ocasión, el refinamiento y estructuración de los requerimientos se realiza utilizando un lenguaje técnico orientado al desarrollo del software.

El Modelo de Análisis que se presenta en este capítulo desarrolla un sistema compuesto por paquetes y clases de análisis que reflejan el resultado del proceso de estructuración realizada sobre los requerimientos capturados en el capítulo anterior.

Se estructuran los requisitos de manera que se facilite su comprensión, su preparación, su modificación, y en general, su mantenimiento. Esta estructura (basada en clases de análisis y paquetes) es independiente de la estructura que se dio a los requerimientos (basada en casos de uso).

La representación de estos paquetes y clases de análisis se realiza siguiendo la notación UML planteada por Ivar Jacobson, Grady Booch y James Rumbaugh en el libro "*El Proceso Unificado de Desarrollo de Software*" (Rumbaugh, J. & Jacobson, I. & Booch, G., 2000b) basado en la representación de paquetes compuestos de clases de interfaz, de control y de datos, tal como se muestra en la ilustración 26. Estos diagramas presentan una primera aproximación para la estructuración que se realiza en la etapa de diseño.

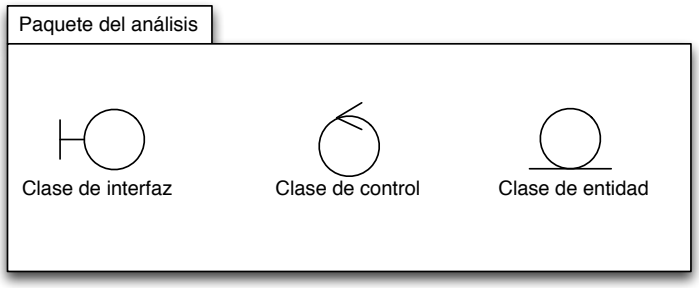

Ilustración 26: Paquetes y clases del análisis

Las clases de interfaz se utilizan para modelar la interacción entre los actores y el sistema. Son las encargadas de presentar y recibir información.

Las clases de control se utilizan para modelar los procesos de coordinación, gestión, control etc. que existen en el sistema.

Las clases de entidad se utilizan para modelar los conceptos de información que se encuentran presentes en el sistema.

8.1. Modelo de Análisis

Al analizar el conjunto de requisitos software se observa una estructuración lógica basada en los siguientes conceptos:

- Sistema de integración del componente.
- Sistema de control del plano.
- Sistema de gestión de capas.
- Sistema de gestión de entidades.
- Sistema de gestión del tiempo.
- Sistema de gestión de la escala.
- Sistema de representación de la vista.

Estos conceptos forman los paquetes de análisis en los que se van a agrupar las distintas clases. A continuación se realiza el desglose y descripción de cada paquete.

8.1.1. Paquete de análisis 01.- Sistema de integración del componente

Este paquete lo forman aquellas clases encargadas de la configuración del componente, que permiten su adaptación visual en el entorno que lo integra.

En él se cubren los siguientes requerimientos software:

Requerimiento	Nombre
RS-CU-01	Configurar estética
RS-CU-02	Configurar borde
RS-CU-03	Configurar fondo
RS-CU-04	Configurar tamaño
RS-CU-05	Configurar herramientas visuales
RS-CU-06	Ocultar/Mostrar herramienta
RS-CU-07	Establecer posición

Tabla 92: Requerimientos cubiertos por el "Sistema de integración"

El diagrama de análisis que representa este sistema es el siguiente:

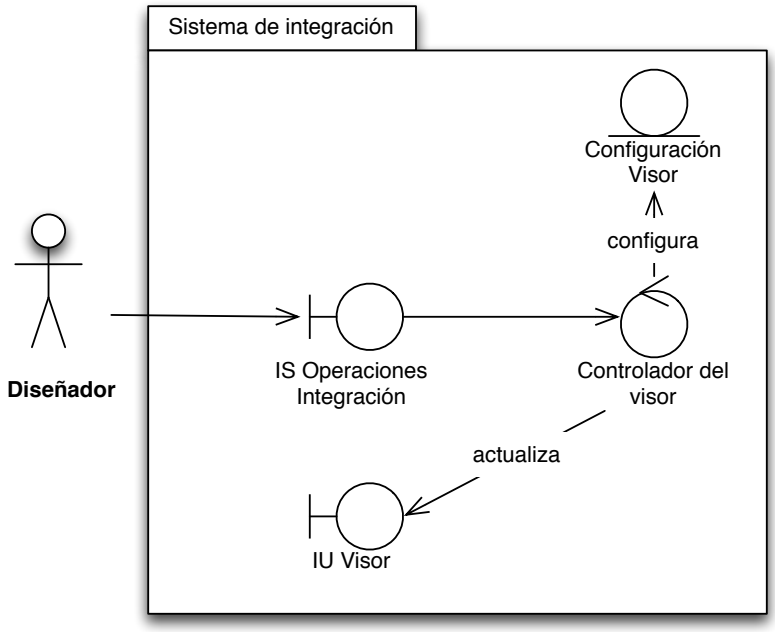

Ilustración 27: Paquete de análisis "Sistema de integración"

El Diseñador a través de la interfaz *IS Operaciones Integración* hace una petición de cambio de configuración visual. Esta petición se lleva al *Controlador del visor*, el cual gestiona la petición actualizando los valores correspondientes en la entidad *Configuración Visor*.

Tras actualizar la entidad indica a la interfaz *IU Visor* que actualice su representación.

8.1.2. Paquete de análisis 02.- Sistema de control del plano

Este paquete lo forman aquellas clases encargadas de la gestión de la carga del plano.

En él se cubren los siguientes requerimientos:

Requerimiento	Nombre
RS-CU-08	Cargar plano
RS-CU-09	Seleccionar plano
RS-CU-10	Escoger coordenadas

Tabla 93: Requerimientos cubiertos por el "Sistema de control del plano"

El diagrama de análisis que representa este sistema es el siguiente:

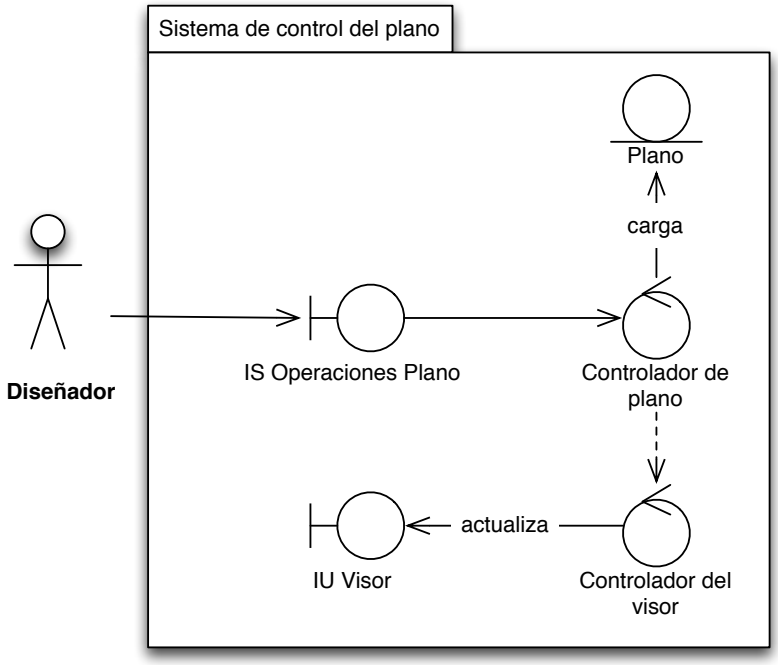

Ilustración 28: Paquete de análisis "Sistema de control del plano"

A través de la interfaz *IS Operaciones Plano*, el Diseñador realiza las operaciones de carga y configuración del plano. Introduciendo la información necesaria para la gestión del plano, esta petición es comunicada al *Controlador de plano*, el cual se encarga de gestionar la clase de entidad *Plano*. Si es necesario actualizar la vista e interfaz *IU Visor*, el *Controlador de plano* informa a la clase *Controlador del visor*.

8.1.3. Paquete de análisis 03.- Sistema de gestión de capas

Este paquete lo forman aquellas clases encargadas de la gestión del sistema de capas.

En él se cubren los siguientes requerimientos:

Requerimiento	Nombre
RS-CU-28	Crear capa
RS-CU-29	Poner nombre
RS-CU-30	Eliminar capa
RS-CU-31	Seleccionar capa
RS-CU-32	Ordenar capa
RS-CU-33	Ajustar visibilidad

Tabla 94: Requerimientos cubiertos por el "Sistema de gestión de capas"

El diagrama de análisis que representa este sistema es el siguiente:

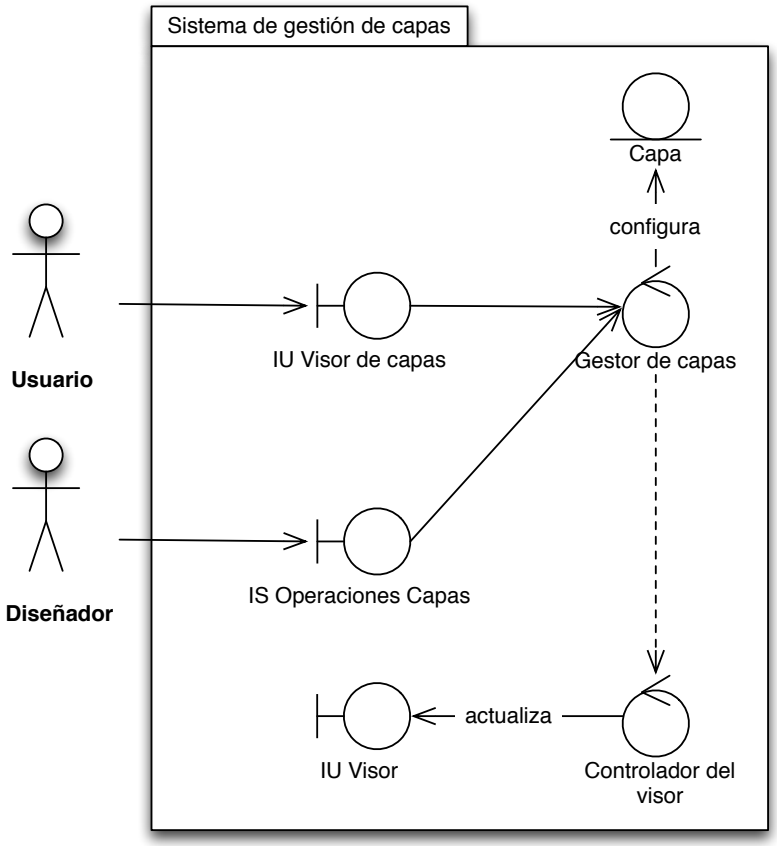

Ilustración 29: Paquete de análisis "Sistema de gestión de capas"

En este diagrama operan los dos actores del sistema, tanto el Usuario, como el Diseñador realizan operaciones sobre el sistema de capas.

El primero, hace uso de la interfaz de usuario *IU Visor de capas*, desde donde tiene acceso a operaciones como la creación, eliminación, ocultación de una capa. A través de esta interfaz se comunica con el *Gestor de capas*, el cual gestiona las entidades *Capa*.

A diferencia del Usuario, el Diseñador opera con el *Gestor de capas*, mediante la interfaz software *IS Operaciones Capas*.

Por último, si es necesario actualizar la vista e interfaz *IU Visor*, el *Gestor de capas* informa a la clase *Controlador del visor*.

8.1.4. Paquete de análisis 04.- Sistema de gestión de entidades

Este paquete lo forman aquellas clases relacionadas con el tratamiento de las entidades representadas en el escenario.

En él se cubren los siguientes requerimientos:

Requerimiento	Nombre
RS-CU-34	Representar entidad
RS-CU-35	Especificar coordenadas
RS-CU-36	Especificar forma
RS-CU-37	Seleccionar apariencia
RS-CU-38	Asociar etiqueta
RS-CU-39	Especificar capa
RS-CU-40	Añadir comportamiento futuro
RS-CU-41	Especificar tiempo
RS-CU-42	Modificar entidad
RS-CU-43	Eliminar entidad
RS-CU-44	Seleccionar entidad
RS-CU-45	Cambiar ubicación

Tabla 95: Requerimientos cubiertos por el "Sistema de gestión de entidades"

El diagrama de análisis que representa este sistema es el siguiente:

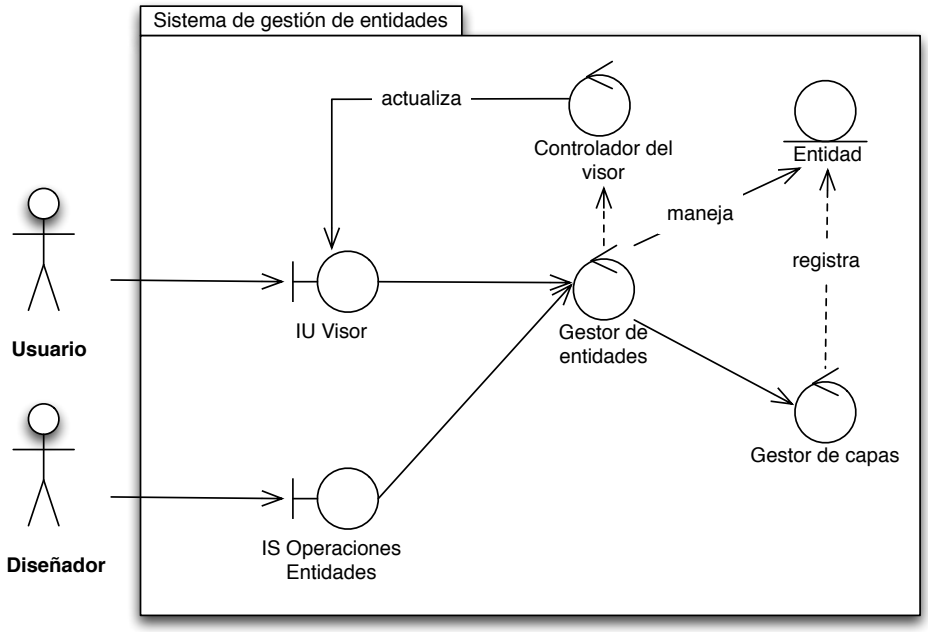

Ilustración 30: Paquete de análisis "Sistema de gestión de entidades"

En el sistema de gestión de entidades, los distintos actores del sistema interactúan con sus respectivas interfaces. Hacen uso de esta para especificar el tipo de figura, su ubicación temporal y su configuración visual. El *Gestor de entidades* se encarga de crear y configurar la entidad, así como de informar al *Gestor de capas* que debe tratar la *Entidad*.

Finalmente, si la entidad es visible o se ha producido algún cambio en la visualización, el *Gestor de entidades* informa a la clase *Controlador del visor* que actualice la interfaz *IU Visor*.

8.1.5. Paquete de análisis 05.- Sistema de gestión de tiempo

Este paquete lo forman aquellas clases encargadas de la gestión del sistema de tiempo. Se configura el tiempo inicial, actual y final del sistema, así como se gestionan los cambios de tiempo actual.

En él se cubren los siguientes requerimientos:

Requerimiento	Nombre
RS-CU-11	Configurar el tiempo
RS-CU-12	Establecer tiempo inicial
RS-CU-13	Establecer tiempo actual
RS-CU-14	Establecer tiempo final
RS-CU-27	Cambiar tiempo actual

Tabla 96: Requerimientos cubiertos por el "Sistema de gestión de tiempo"

El diagrama de análisis que representa este sistema es el siguiente:

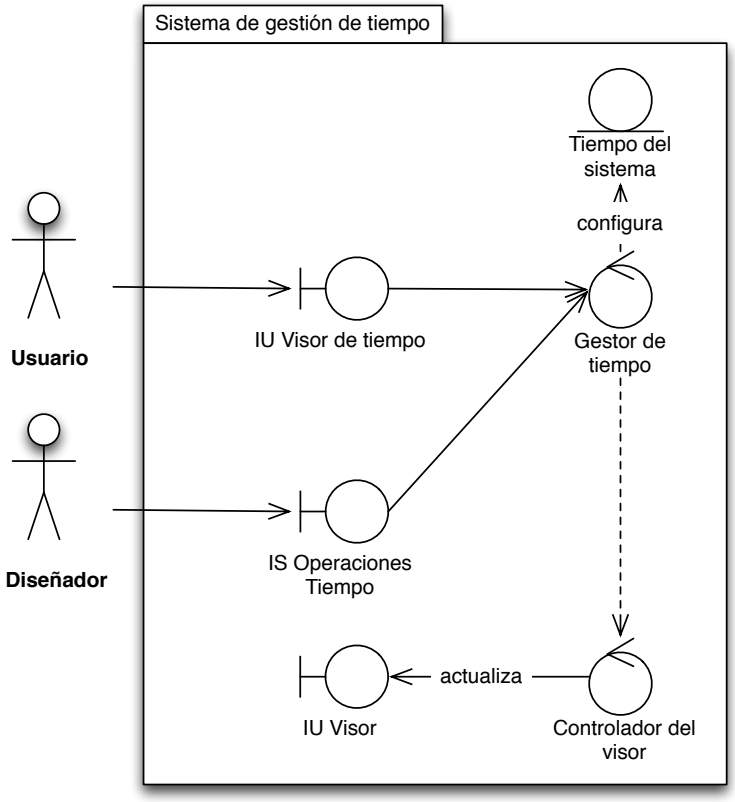

Ilustración 31: Paquete de análisis "Sistema de gestión de tiempo"

A través de las interfaces de usuario y software, *IU Visor de tiempo* y *IS Operaciones Tiempo*, los distintos actores operan con el sistema de gestión de tiempo. Este se encarga de configurar la entidad *Tiempo del sistema*, que es quién contiene los datos relativos a la configuración temporal del sistema.

Si se realiza un cambio en el valor de tiempo actual, es necesario refrescar la interfaz de usuario *IU Visor*, por ello, el *Gestor de tiempo* informa al *Controlador del visor* para ello.

8.1.6. Paquete de análisis 06.- Sistema de gestión de escala

Este paquete lo forman aquellas clases encargadas de la gestión del sistema de escala.

En él se cubren los siguientes requerimientos:

Requerimiento	Nombre
RS-CU-15	Configurar la escala
RS-CU-16	Establecer el número de niveles
RS-CU-17	Establecer el nivel actual

Tabla 97: Requerimientos cubiertos por el "Sistema de gestión de escala"

El diagrama de análisis que representa este sistema es el siguiente:

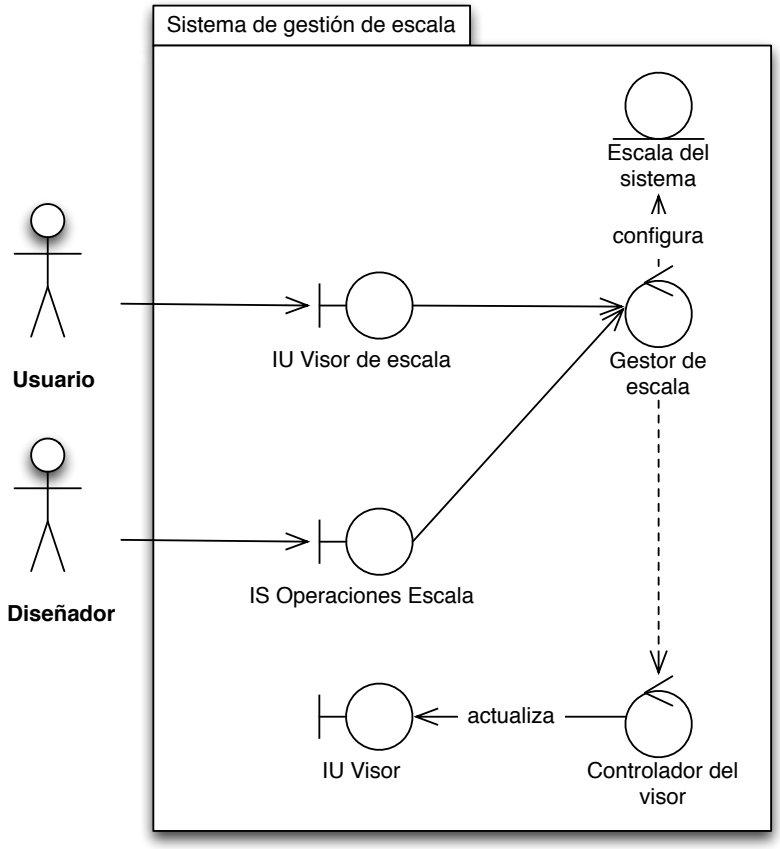

Ilustración 32: Paquete de análisis "Sistema de gestión de escala"

Como en los sistemas previos, los actores disponen de interfaces propias, en este caso, *IU Visor de escala* y *IS Operaciones Escala*, con las que operar con el componente. Estas interfaces comunican la operación con el *Gestor de escala*, el cual se encarga de gestionar la entidad *Escala del sistema*. Esta clase modela los valores internos del sistema relacionados con la configuración de la escala.

Si se realiza un cambio de escala actual, es necesario refrescar la interfaz de usuario *IU Visor*, por ello, el *Gestor de escala* informa al *Controlador del visor* para ello.

8.1.7. Paquete de análisis 07.- Sistema de representación de la vista

Este paquete lo forman aquellas clases encargadas de la representación realizada en la vista. Se ubican en este sistema aquellos casos de uso que actualizan la vista.

En él se cubren los siguientes requerimientos:

Requerimiento	Nombre
RS-CU-18	Ver escenario
RS-CU-19	Ver coordenadas actuales
RS-CU-20	Ver nivel de zoom actual
RS-CU-21	Ver tiempo actual
RS-CU-22	Ver plano
RS-CU-23	Ver entidades
RS-CU-24	Ver simulación
RS-CU-25	Ampliar/Reducir plano
RS-CU-26	Mover Plano

Tabla 98: Requerimientos cubiertos por el "Sistema de representación de la vista"

El diagrama de análisis que representa este sistema es el siguiente:

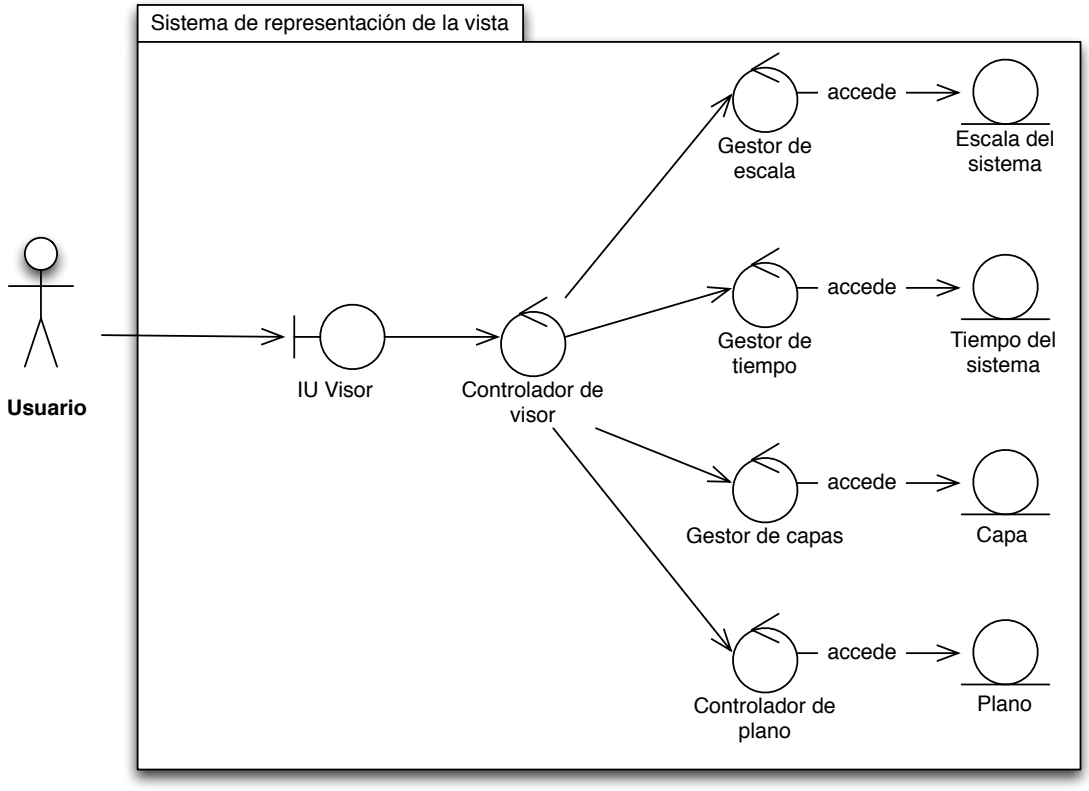

Ilustración 33: Paquete de análisis "Sistema de representación de la vista"

El actor Usuario opera con el visor a través de la interfaz *IU Visor*, realizando cambios de nivel en el plano, o moviéndolo, o viendo algún tipo de información mostrada por el mismo. El sistema de representación se encarga de mostrar en cada momento la información correspondiente a la situación actual del escenario.

Toda operación que realiza el Usuario se comunica con el *Controlador de visor*, el cual se encarga de mostrar la vista actualizada basándose en el estado de los distintos sistemas. Para ello, se comunica con los distintos controladores de donde obtiene la información de las distintas clases entidad.

Capítulo 9.- Diseño

En este capítulo se modela la estructura que soporta el conjunto total de requisitos establecidos, y en definitiva da forma al sistema final. Se contextualiza tanto la estructura externa (elementos hardware y sus relaciones) como la interna (elementos software) del sistema, proporcionando una visión global del entorno que lo rodea y de los elementos que lo constituyen.

El diseño que se establece en este capítulo está centrado en el principal objetivo del proyecto, la creación de un componente software para la visualización y representación entidades y simulaciones sobre un plano.

Por otro lado, para mostrar las distintas funciones que proporciona el componente, se diseña una aplicación que hace un uso directo de los diversos métodos de comunicación que proporciona el componente.

9.1. Arquitectura

Uno de los principales requerimientos no funcionales que se establecieron, fue el desarrollo del componente para un tipo de tecnologías de desarrollo de aplicaciones en concreto, las RIA. Éstas tienen por filosofía el desarrollar aplicaciones web orientadas a los servicios, pero con la particularidad de ofrecer una experiencia de usuario similar a la de una aplicación clásica de escritorio. Para ello, las tecnologías de desarrollo categorizadas como RIAs ofrecen elementos característicos de las aplicaciones de escritorio como pueden ser botones, cajas de texto, paneles organizativos, etc. o aspectos de comportamiento, como la ausencia de recarga en la página.

De forma genérica una aplicación RIA, que haga uso de servicios web, suele utilizar una arquitectura cliente-servidor. Utiliza un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de servicios, llamados servidores y los demandantes, los clientes. En el siguiente diagrama se muestra el esquema básico de esta arquitectura.

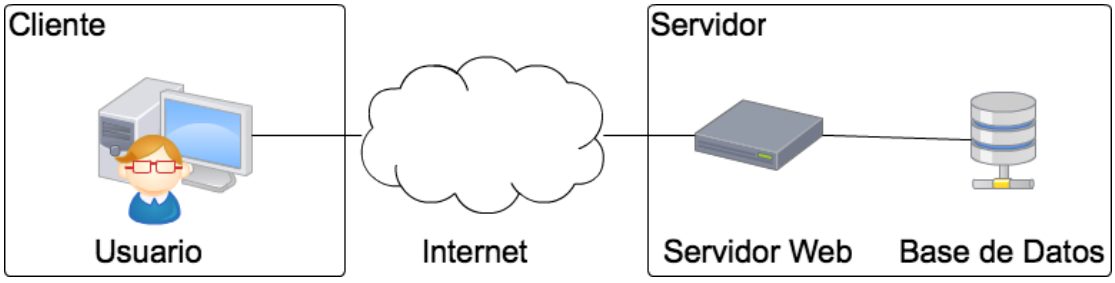

Ilustración 34: Arquitectura tradicional Cliente-Servidor

En la parte del cliente se desarrolla la interacción entre el usuario y la interfaz de la aplicación. Es la encargada de ejecutar comandos que hagan peticiones hacia el servidor, así como manejar y presentar los resultados de las mismas.

El servidor se encarga de procesar las peticiones realizadas por la aplicación cliente. Si se requiere, utiliza bases de datos para estructurar y almacenar la información. Tras determinar el resultado de la petición, devuelve los mismos al cliente.

Una descripción detallada de las principales características que cumplen las RIA, se encuentra en el Anexo I: Arquitectura de una RIA.

En el esquema visto en la ilustración 34, el componente visor de planos pertenece a la aplicación cliente. Concretamente es un elemento de la vista de la misma, esto es, forma parte de la capa de presentación de la aplicación RIA. Al tratarse de un componente, es un elemento distinguible y separable que dispone de diversas interfaces públicas para su comunicación con el resto de elementos.

Un componente es un bloque de código sólido e independiente del resto del sistema que permite realizar una función determinada. Las principales características esenciales que debe cumplir un componente son las siguientes:

- Reusable
- Sin contexto específico
- Extensible
- Encapsulado
- Independiente

Estas características quedan reflejadas en el siguiente diagrama, que muestra la independencia de un componente respecto a la aplicación que lo contiene. Una descripción ampliada se encuentra en el Anexo II: Características de un componente.

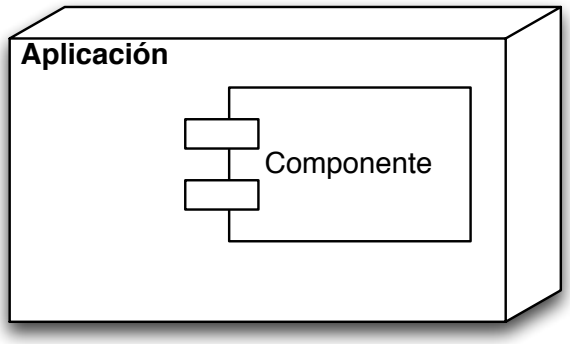

Ilustración 35: Integración de un componente

De forma genérica, un componente ofrece interfaces de comunicación de 2 tipos, de entrada y de salida. Con las primeras, es la aplicación quién inicia la comunicación con el componente, mientras que con las segundas es este último el que inicia la operación.

9.2. Tecnología

Para el desarrollo de este tipo de aplicaciones, se requieren un conjunto amplio de tecnologías. Por un lado, tecnologías aplicadas a la aplicación cliente, y por otro, tecnologías aplicadas a la aplicación servidor. El proyecto que se aborda está centrado en tecnologías aplicadas en la aplicación cliente, puesto que es donde se ubica el desarrollo del componente visor.

Partiendo del carácter gráfico del proyecto y a su utilización en futuros proyectos, se planteó el uso de las siguientes tecnologías.

- Adobe Flash utilizando el framework Adobe Flex.
- Microsoft Silverlight.

Entre estas alternativas se decidió utilizar la primera por la alta popularización de la tecnología utilizada, Flash, y gracias a su plug-in, su alta integración en los distintos navegadores; lo que asegura una ejecución en la mayoría de ordenadores actuales. Según cálculos estadísticos, el plug-in se encuentra instalado en más del 98% de los ordenadores actuales y más del 90% tiene instalada la última versión del plug-in Adobe Flash Player.

Además, se valoró:

- La alta capacidad de personalización en los contenidos desarrollados, dando un aspecto visual destacable a las aplicaciones resultantes.
- La existencia del framework de desarrollo Flex 3.0, un conjunto de elementos prediseñados altamente configurables.
- Su referencia y documentación. Debido a la popularización de la tecnología, existe bastante documentación sobre cualquier aspecto de la misma.

La plataformas de desarrollo de Flash utilizada es Adobe Flex Builder 3.0, donde la programación se realiza mediante los lenguajes ActionScript 3.0 y MXML.

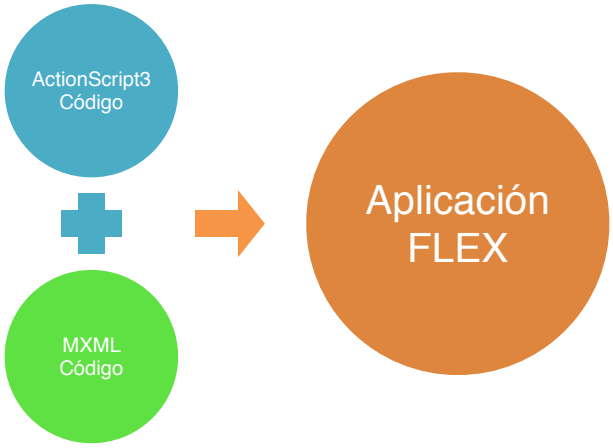

Ilustración 36: Lenguajes utilizados para la construcción de una aplicación Flex

Este entorno integra el framework open-source Flex 3.0 creado por Adobe. Proporciona diversos elementos de diseño tales como botones, paneles, elementos para dibujar gráficas, etc. creados con el fin de permitir el desarrollo ágil de aplicaciones RIA, orientados a la web.

Actualmente el entorno de desarrollo ha cambiado de nombre para denominarse Adobe Flash Builder y va por la versión 4.7. Por su lado, el framework Adobe Flex ha pasado a manos de la fundación Apache llamándose Apache Flex.

9.3. Diseño del componente

En este apartado se detallan los aspectos de diseño más importantes del desarrollo del componente.

9.3.1. Nombre

Para la definición del nombre del componente se ha optado por basar el mismo en el objetivo básico del componente, la visualización de mapas. El nombre escogido ha sido MViewer, que proviene de la contracción de Map Viewer.

9.3.2. Solución

El diseño del sistema que se plantea en este capítulo tiene como base principal el uso del patrón Modelo Vista Controlador, MVC (ver Anexo III: Patrones) y el modelo de análisis presentado en el capítulo anterior. Éste basa sus diagramas en la utilización de clases de interfaz, de control y de entidad. Esta categorización permite que el modelo de análisis sea fácilmente adaptable al modelo de diseño realizando un emparejamiento entre estas clases y la clasificación establecida por el patrón MVC.

Esta solución encaja perfectamente en el diseño del componente, puesto que es un elemento software con una gran carga gráfica basada en la representación de un modelo, en este caso un plano con entidades y simulaciones.

Vistas

La solución establecida se basa en ofrecer 2 tipos de vistas, vistas de Usuario y vistas de Diseñador.

Vistas de Usuario

Son las interfaces gráficas que suministra el componente. Tienen por objetivos mostrar la parte correspondiente del modelo y ofrecer la interacción necesaria al Usuario. En este conjunto, se encuentran el propio visor del plano y el conjunto de herramientas visuales para el manejo de la escala o el tiempo.

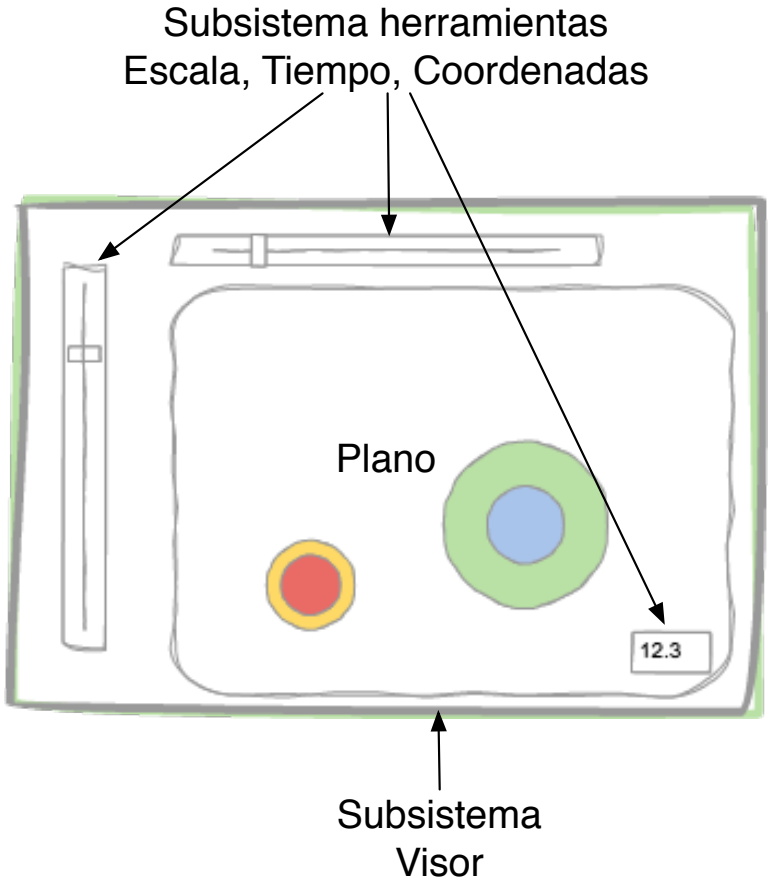

Ilustración 37: Composición de la vista de Usuario

Nótese que el Subsistema Visor es el principal subsistema del componente. Se realiza una distinción del mismo con los otros subsistemas de herramientas porque es éste quien agrega las herramientas visuales.

Vistas del Diseñador

Es la vista que ofrece el componente visor al Diseñador o a la aplicación que lo contiene. Es una vista basada en la comunicación entre elementos software. El componente ofrece una serie de interfaces para su comunicación, tanto de entrada como de salida.

En el caso de las interfaces de entrada, el componente ofrece métodos y propiedades para hacer uso de su funcionalidad, para configurarlo, para registrar o eliminar entidades, para añadir una simulación, etc.

En las interfaces de salida, el visor notifica eventos ocurridos en el mismo, como un cambio de escala, un cambio de tiempo, un registro o eliminación de una entidad, una acción producida por el Usuario, etc.

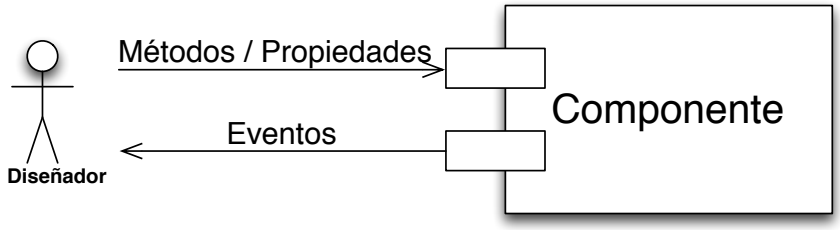

Ilustración 38: Vista del Diseñador

Controladores

Forman las clases que se encargan de gestionar las peticiones realizadas en las vistas, operando sobre ellas o sobre el modelo.

Modelo

Contiene la información asociada con el sistema de representación, en este caso, el plano y las entidades registradas. Así mismo, también se reflejan como modelo las clases de información de los sistemas de tiempo, de capas, o de escala.

Vista del modelo de análisis como MVC

La ilustración 39 escenifica la adaptación del modelo de análisis en el modelo de diseño utilizando el patrón MVC. Refleja la participación en el sistema de los actores Usuario y Diseñador actuando sobre sus interfaces (IU e IS respectivamente). Existe una distinción en el tipo de interfaz utilizada por los actores del sistema, ya que, aunque ambas forman parte de la Vista del componente, son de distinta naturaleza.

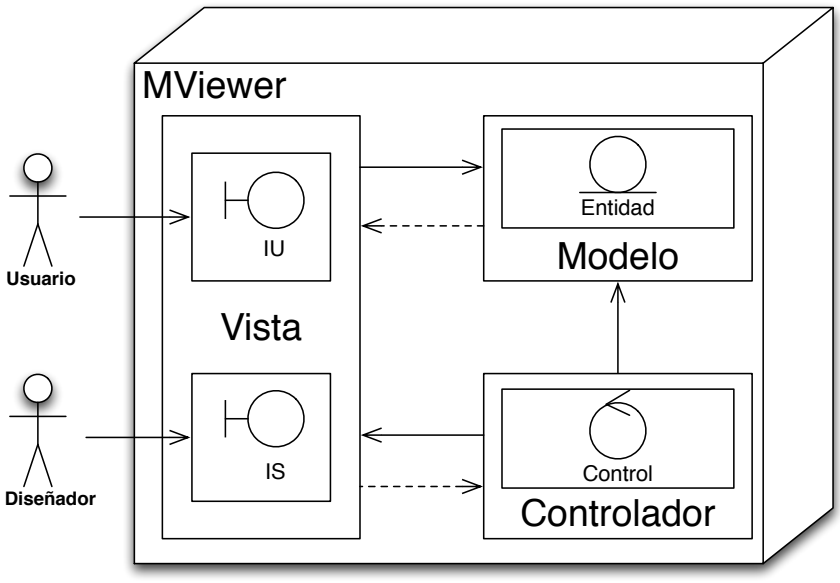

Ilustración 39: Adaptación del modelo de análisis al modelo de diseño basado en el patrón MVC

De igual manera, ilustra el esquema general que siguen los subsistemas que forman el sistema visor. En las secciones siguientes se describen y detallan las clases de diseño que contienen.

9.3.3. Subsistemas

A partir de los paquetes de análisis, se realiza una traducción directa de estos para encontrar los subsistemas de diseño que forman el componente.

Cada subsistema está compuesto por clases de diseño, que a diferencia de las clases de análisis, representan ya una estructura lógica del sistema cercana a la estructura final.

El conjunto de subsistemas de diseño establecidos son los siguientes:

- Subsistema de integración del componente.
- Subsistema de control del plano.
- Subsistema de gestión de la escala.
- Subsistema de gestión del tiempo.
- Subsistema de entidades.
- Subsistema de gestión de capas.
- Subsistema principal de vista.

La siguiente ilustración presenta un diagrama representativo del contenido del componente.

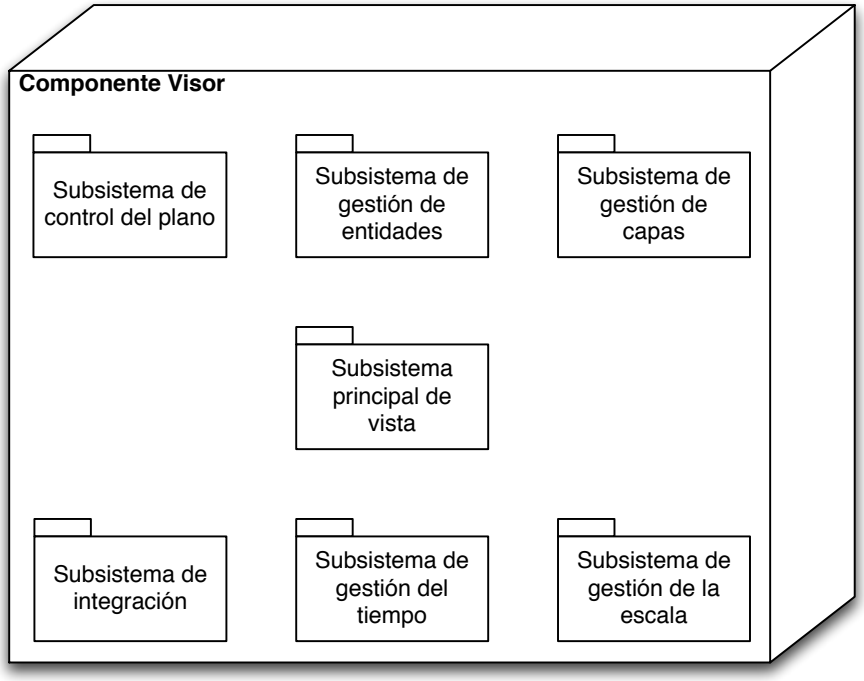

Ilustración 40: Composición del componente visor en subsistemas

A continuación se procede a desglosar cada subsistema, describiendo e identificando las principales clases de diseño, los patrones aplicados, la descripción de las interfaces dadas, y las interacciones con otros subsistemas.

Subsistema 01.- Subsistema de integración

Lo forman aquellas clases que permiten la integración y adaptación del componente a la aplicación que lo contiene. Principalmente se manejan en este subsistema tareas de configuración estética. Se adapta la presentación del componente a las necesidades de la aplicación que lo agrega.

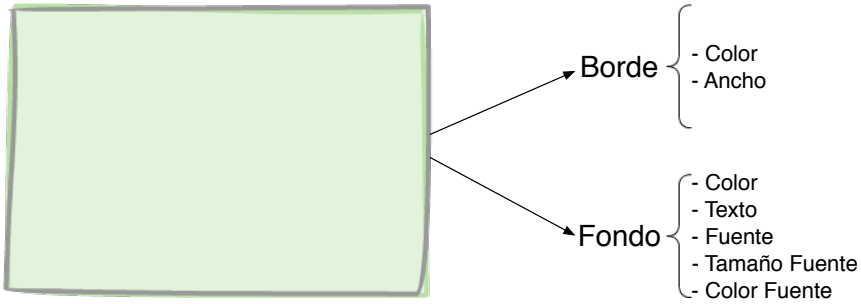

Ilustración 41: Aspectos estéticos configurables

El siguiente diagrama identifica las clases de diseño que conforman este subsistema.

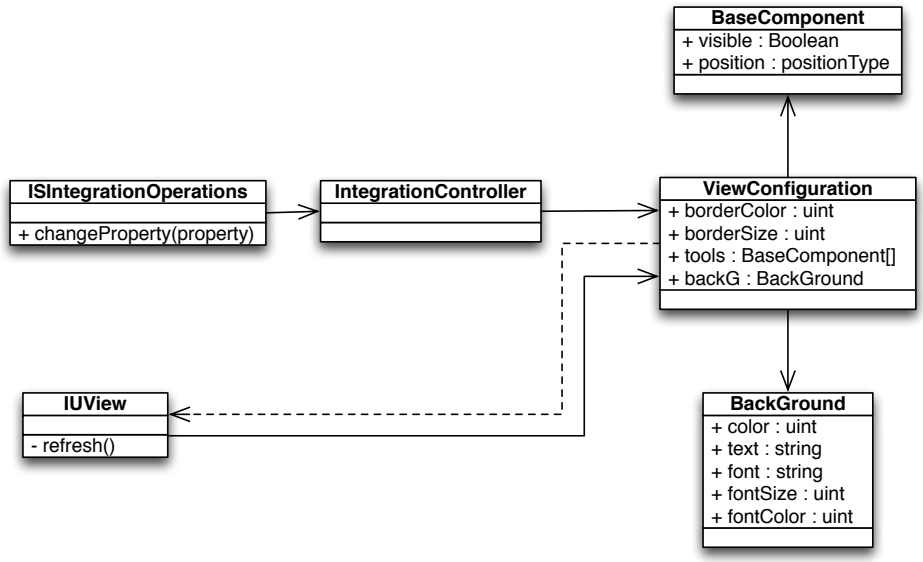

Ilustración 42: Diagrama de clases del Subsistema de Integración

A través de la interfaz software *ISIntegrationOperations* el Diseñador es capaz de configurar la estética del visor y de los elementos que se muestran sobre el mismo. Accede a las propiedades de la clase *ViewConfiguration*, que estructura las propiedades estéticas del visor.

Del mismo modo accede a la visibilidad y posición de las distintas herramientas que suministra el componente.

Subsistema 02.- Subsistema de control del plano

Este subsistema se encarga de realizar la carga del plano y de modelar la representación interna que mantiene el sistema.

La carga consiste en la adquisición del plano desde fuera del visor. En este caso se plantean 2 opciones, o bien que la aplicación que agrega el componente proporcione la imagen, o bien, sea tarea del componente cargar la imagen desde una URL. Además es necesario posicionar el plano en unas coordenadas, para ello se indican las mínimas y las máximas.

La representación lógica que se plantea, está estrechamente ligada al manejo de la vista de Usuario que realiza el subsistema principal de vista. De momento sólo es necesario tener el concepto de que la imagen del plano se trocea en trozos o celdas cuadradas de un tamaño fijo.

La siguiente ilustración refleja la tarea de carga y representación interna realizada por este subsistema.

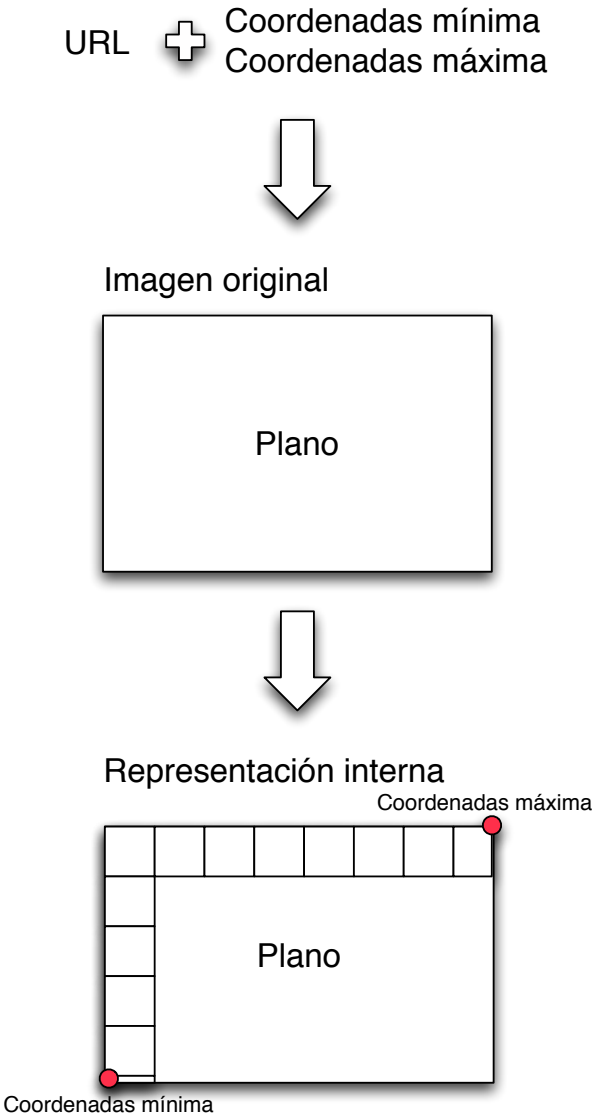

Ilustración 43: Carga y representación lógica del plano

El diagrama de clases para este subsistema es el siguiente:

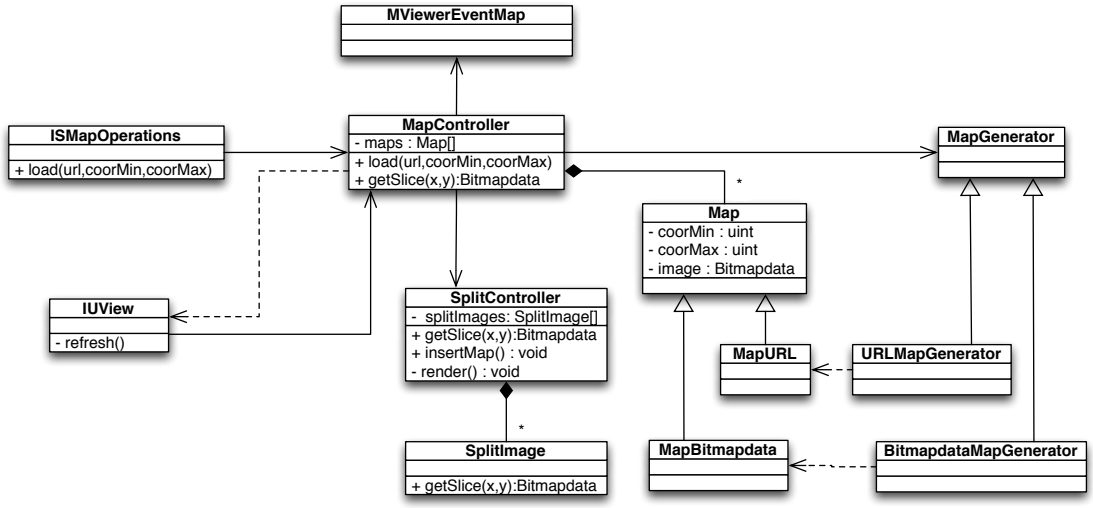

Ilustración 44: Diagrama de clases del Subsistema de Control del Plano

El sistema planteado tiene 3 elementos principales, la interfaz para el diseñador, *ISMapOperations*, el controlador principal, *MapController*, y el conjunto de clases que cargan y modelan el plano.

Para la carga se diseña un sistema basado en el patrón Factory (ver Anexo III: Patrones) que permite la obtención del plano desde diversas fuentes, en el diagrama, desde una url o suministrado con el formato de imágenes utilizado en AS3, la clase Bitmapdata.

Para su representación lógica se establecen las clases *SplitController* y *SplitImage*, esta última es la clase modelo que almacena las celdas del plano. Ambas clases proporcionan el método *getSlice(x,y)* para acceder a éstas.

Por último especificar que tras la carga y la estructuración interna del plano, el controlador *MapController* notifica a la vista del visor *IUView* que actualice la representación y emite un evento de tipo *MviewerMapEvent* hacia el exterior del componente, para la aplicación que lo agregue.

Subsistema 03.- Subsistema de gestión de la escala

Este subsistema gestiona la escala de la vista presentada. Principalmente se realizan tareas de configuración sobre el mismo y operaciones de cambio de nivel de escala.

El concepto de escala que se trata, está vinculado a las coordenadas del plano dado y a la representación del mismo. Se establece una proporción pixel/coordenada, que permite definir un valor base como nivel de zoom base. Con este nivel de zoom base, se establecen un número de niveles de zoom, en los que cada nivel es una ampliación o reducción del valor base en función de un porcentaje.

La configuración vendrá dada por la especificación del número de niveles y el porcentaje de incremento o reducción del valor base de zoom.

El diagrama de clases que modela este subsistema es el siguiente:

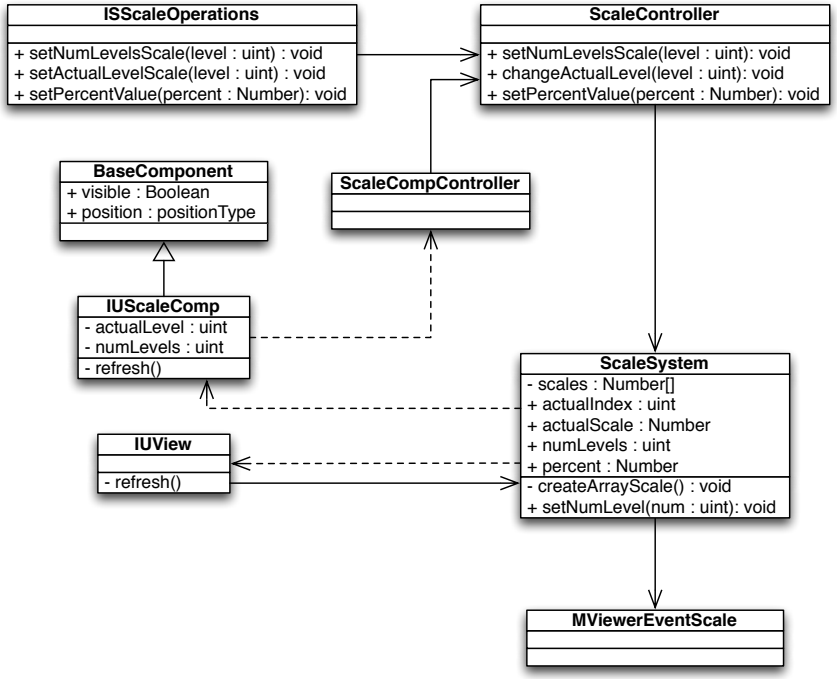

Ilustración 45: Diagrama de clases del Subsistema de Gestión de la Escala

El sistema planteado está formado por las interfaces de operación *ISScaleOperations* y *IUScaleComp*, la primera es la interfaz software dada al Diseñador, mientras que la segunda, conforma la herramienta visual que utiliza el Usuario.

La interfaz gráfica cuenta con un controlador propio para gestionar su representación visual y operar sobre el subsistema. Este accede al controlador principal *ScaleController*, el cual opera con la clase *ScaleSystem*. Esta representa el sistema de escala del sistema, y es la clase principal de este subsistema. Contiene el número de niveles de zoom del sistema, así como los valores de escala (pixel/coordenada) para cada uno. Cualquier evento ocurrido sobre el modelo es notificado hacia el exterior con un evento de tipo *MViewerEventScale*.

Subsistema 04.- Subsistema de gestión del tiempo

El visor que se presenta se centra en la representación de entidades y simulaciones. Un elemento importante para la representación de estas últimas es el concepto de tiempo.

Cuando se simula un modelo matemático se obtiene un resultado. Este puede depender o no de la variable del tiempo. Un resultado que depende del tiempo muestra un comportamiento dinámico a medida que avanza este. Es necesario por tanto representar esta evolución.

La solución que se plantea consiste en modelar el sistema de tiempo como un conjunto de instantes. El componente visor muestra para cada instante una presentación diferente del estado del escenario.

Para ello el sistema de tiempo cuenta con un instante de tiempo inicial, uno final y uno actual. Se establece con esto un período en el que se pueden representar entidades y simulaciones.

El diagrama de clases establecido es el siguiente:

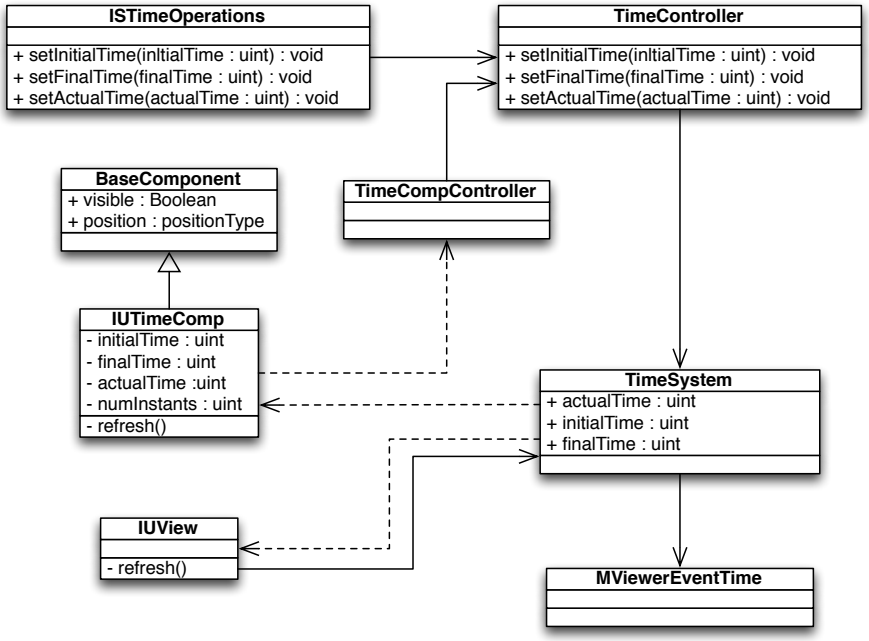

Ilustración 46: Diagrama de clases del Subsistema de Gestión del Tiempo

Sigue un esquema similar visto al del subsistema 03. En este caso la clase del modelo principal es *TimeSystem*. La cual modela el instante de tiempo inicial, final y actual del sistema.

Se proporcionan las interfaces *ISTimeOperations* y *IUTimeComp. Esta última* es la herramienta visual que utiliza el Usuario para desplazarse por el conjunto de instantes del sistema.

Un cambio del instante de tiempo inicial, final o actual provoca una actualización de las interfaces *IUTimeComp* y de *IUView*, y el lanzamiento del evento *MViewerEventTime*.

Subsistema 05.- Subsistema de entidades

Representa al conjunto de clases establecidas para realizar el registro, representación y, eliminación de entidades y el resultado de sus simulaciones.

A nivel lógico, las entidades y los resultados de sus simulaciones se representan de la misma manera, como figuras. Una figura es toda representación que se realice sobre el plano.

El diseño de las figuras se establece teniendo en cuenta las siguientes características básicas:

- Tiene un identificador.- Tras el registro, toda figura adquiere un identificador que permite que sea referenciada para modificarla o eliminarla.

- Tiene un tiempo de vida.- En un sistema de representación de figuras con capacidad temporal es necesario determinar cuando son visibles estas. Para ello, las figuras disponen de una configuración temporal. En este caso la configuración temporal determina si existen o no en el instante seleccionado, y son por ello, representadas o no.

- Está localizada.- Toda figura está localizada tanto gráficamente, por unas coordenadas, como temporalmente, por el tiempo de vida de la misma.

- Tiene un tipo de comportamiento.- En un sistema temporal se establece la posibilidad de que una figura cambie a medida que avanza el tiempo. Este cambio viene dado por un desplazamiento o por un cambio en su forma.

  Para permitir esto, se definen los instantes temporales dentro de una figura, permiten especificar nuevas coordenadas en un instante de tiempo determinado.

  Teniendo 2 instantes de una figura en instantes de tiempo no sucesivos es necesario especificar que sucede en los instantes intermedios. Por ello se han diseñado unos comportamientos básicos para representar el mayor número de comportamiento.

  - **Comportamiento discreto**.- Entre un instante de tiempo y otro, la figura no es representada. Solo mantiene su representación en aquellos instantes temporales registrados en la figura.

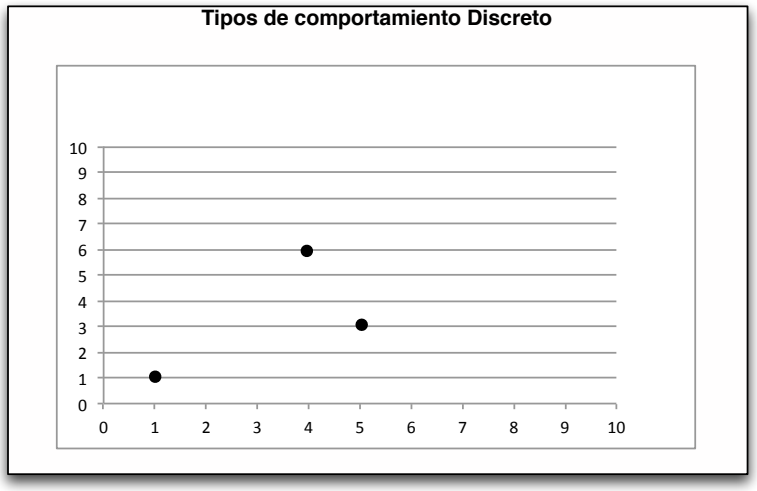

Ilustración 47: Comportamiento Discreto

- **Comportamiento continuo.-** En los instantes temporales intermedios entre 2 instantes registrados en la figura, se realiza una representación constante del primer valor registrado.

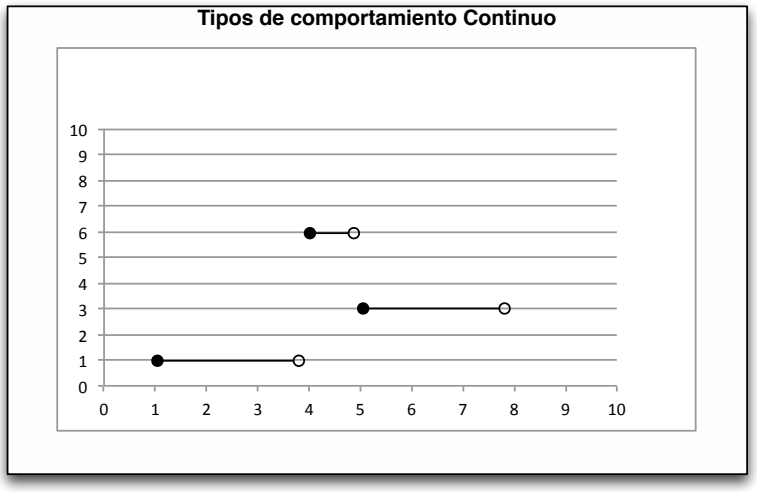

Ilustración 48: Comportamiento Continuo

- **Comportamiento interpolado lineal.-** Entre 2 instantes registrados se realiza una interpolación lineal para determinar el nuevo valor.

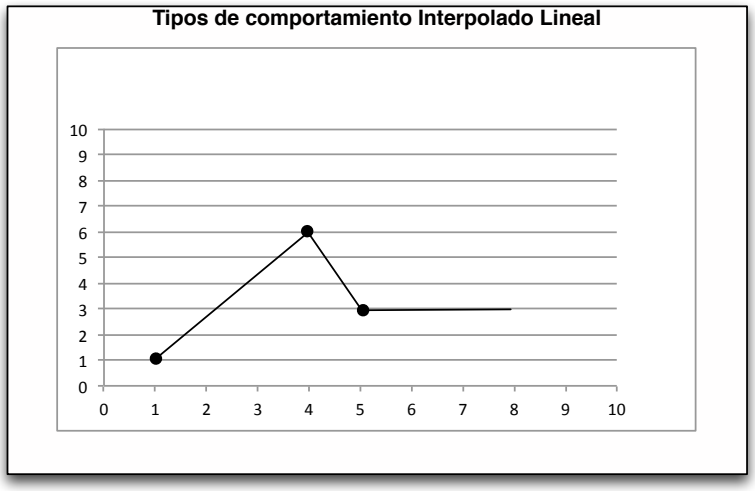

Ilustración 49: Comportamiento Interpolado Lineal

- Tienen una representación.- Las figuras tienen una forma asociada para su representación. Esta incluye además una configuración estética que permite personalizar su representación gráfica.

A continuación se muestra el diagrama de clases de diseño establecido para este subsistema.

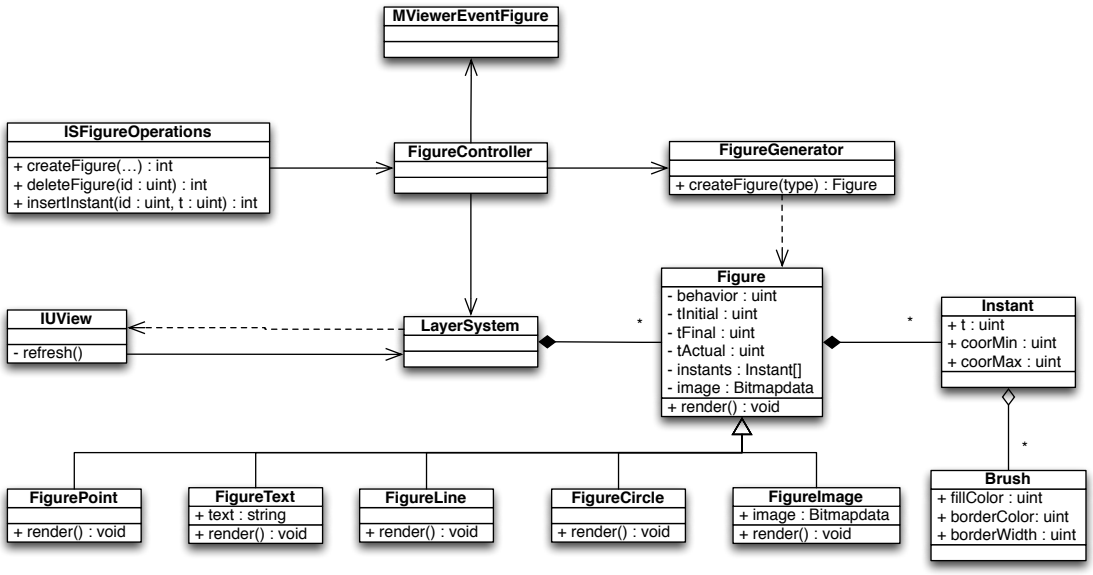

Ilustración 50: Diagrama de clases del Subsistema de Entidades

Destacar de este diagrama que no existe un *FigureSystem* propiamente dicho, el *FigureController* se encarga de gestionar las operaciones dadas por la interfaz *ISFigureOperations*, que principalmente delegará estas operaciones en el constructor de figuras *FigureGenerator;* y en proporcionar éstas al *LayerSystem*. Este último sistema se plantea a continuación, estructura de forma lógica las figuras en capas de información.

Subsistema 06.- Subsistema de gestión de capas

Se diseña este subsistema con el propósito de establecer una estructura interna que sirva para clasificar las formas registradas. La estructura clasifica las figuras en capas, de forma que se pueden aplicar ciertas operaciones a un grupo concreto de ellas.

La siguiente ilustración muestra el agrupamiento en capas.

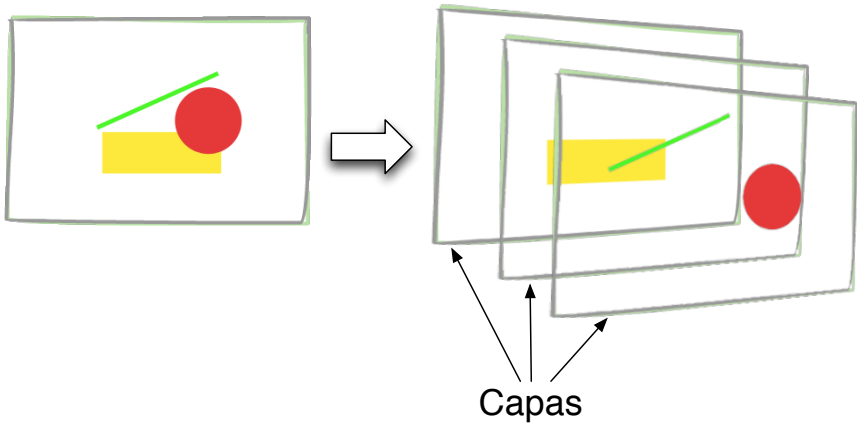

Ilustración 51: Descomposición de una imagen en un conjunto de Capas

El sistema que se propone es el siguiente:

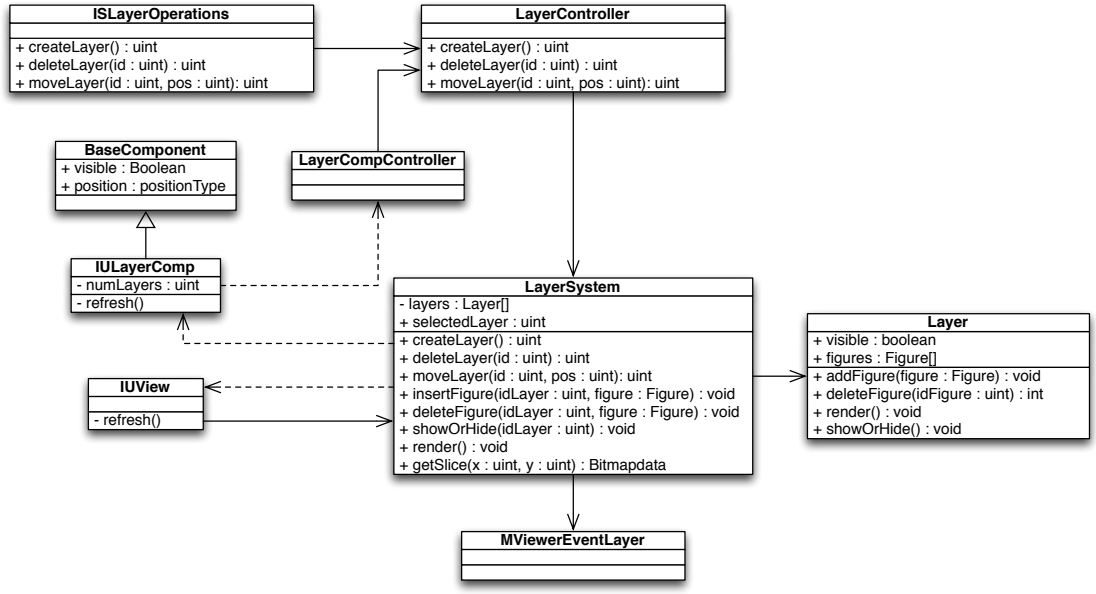

Ilustración 52: Diagrama de clases del Subsistema de Gestión de Capas

El *LayerSystem* está centrado en la organización de las figuras en capas *Layer*. Realiza operaciones para su inserción, eliminación y ocultamiento principalmente. Además dispone de funciones para obtener el renderizado final de las capas.

El siguiente subsistema se encarga de gestionar la vista presentada al Usuario.

Subsistema 07.- Subsistema principal de vista

Este subsistema es el encargado de presentar el escenario al Usuario, mostrando el plano y las entidades establecidas. También gestiona la interacción que realiza el Usuario con el componente.

La presentación que se realiza consiste en mostrar el plano junto con las figuras. Para ello la solución adoptada se basa en utilizar un conjunto limitado de imágenes cuadradas formadas por la composición de un trozo cuadrado de plano y un trozo cuadrado de la representación obtenida del sistema de capas, de donde se obtiene la imagen de las figuras. La utilización de un conjunto limitado de cuadrados permite un manejo ágil del visor, al tener que operar con un número reducido de objetos representables. La ilustración 53 muestra la composición de la vista, obtenida de los cuadrados del plano y del sistema de capas.

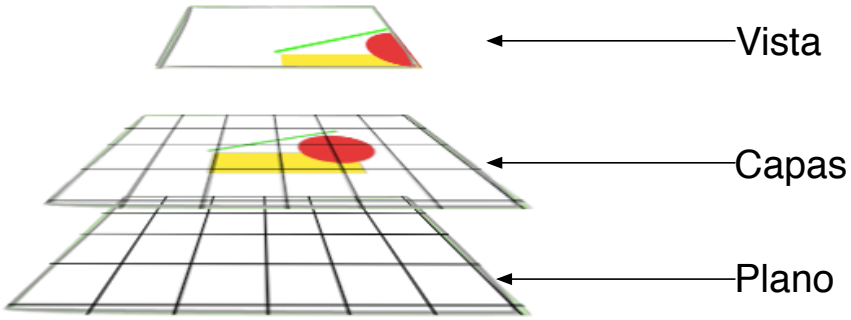

Ilustración 53: Composición final de la vista del visor

La imagen presentada está condicionada por el nivel de zoom en el que se esté y el instante de tiempo actual, de donde se obtiene una presentación actual de las figuras.

Este subsistema se encarga de gestionar la interacción producida por el Usuario, básicamente a través del uso del ratón. Atiende los eventos producidos por el ratón y actúa en consecuencia.

A continuación se muestra su diagrama de clases:

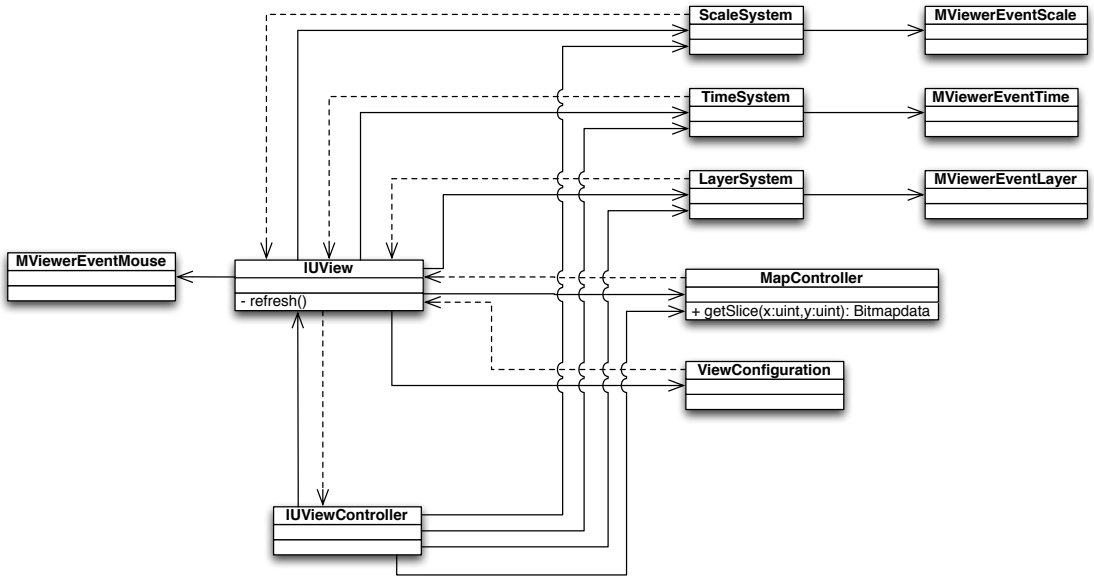

Ilustración 54: Diagrama de clases del Subsistema Principal de Vista

La interfaz *IUView* genera su interfaz obteniendo datos del resto de subsistemas. Escucha los eventos producidos por estos y refresca la vista en consecuencia.

El controlador *IUViewCotroller*, gestiona las peticiones realizadas por el Usuario sobre la interfaz, y las transmite al correspondiente subsistema.

9.3.4. Aspectos variados del diseño

En esta sección se abordan aspectos de diseño que, si bien no han sido visto en apartados anteriores, son de interés para la comprensión de la solución dada.

Espacio de trabajo

Todo elemento registrado en el modelo debe tener unas coordenadas. La solución planteada requiere que se establezca a nivel lógico un espacio de trabajo. Una vez establecido es posible registrar y representar el plano y las figuras.

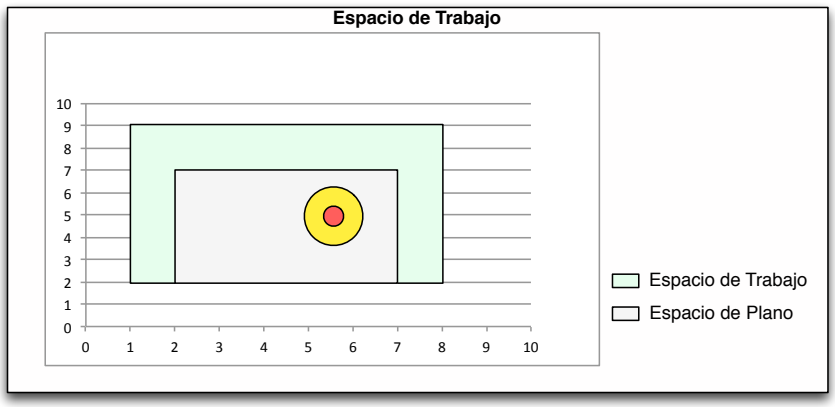

Ilustración 55: Espacio de Trabajo

Como se muestra en la ilustración 55, no todo el espacio de trabajo tiene por qué coincidir con el espacio del plano.

Modos de funcionamiento para el usuario

La interacción del usuario se clasifica en 2 grandes grupos, operaciones de manejo de la vista y operaciones de manejo del modelo. Operaciones que a priori son incompatibles, puesto que, por ejemplo, no se permite realizar un movimiento del plano e insertar una figura a la vez.

Se establecen los modos para diferenciar la forma de actuar tras recibir un evento producido por el usuario.

Los modos básicos establecidos son los siguientes:

- Manejo del visor.
    - Mover y cambiar de nivel de zoom la vista.
- Operar con el modelo.
    - Registrar una figura.
    - Seleccionar figura.
    - Obtener coordenadas.

Listado de formas

El conjunto de formas que se establece para la representación de entidades y simulaciones es el siguiente:

- Punto.
- Línea.
- Círculo.
- Texto.
- Imagen.

Configuración gráfica de las formas

La configuración de la presentación gráfica de las formas se establece mediante objetos de configuración estética. Estos objetos modelan aspectos gráficos tales como los colores, grosores, valores de transparencia, etc. de las figuras registradas.

La solución dada, tiene una estructura jerárquica para determinar que configuración estética corresponde a una figura.

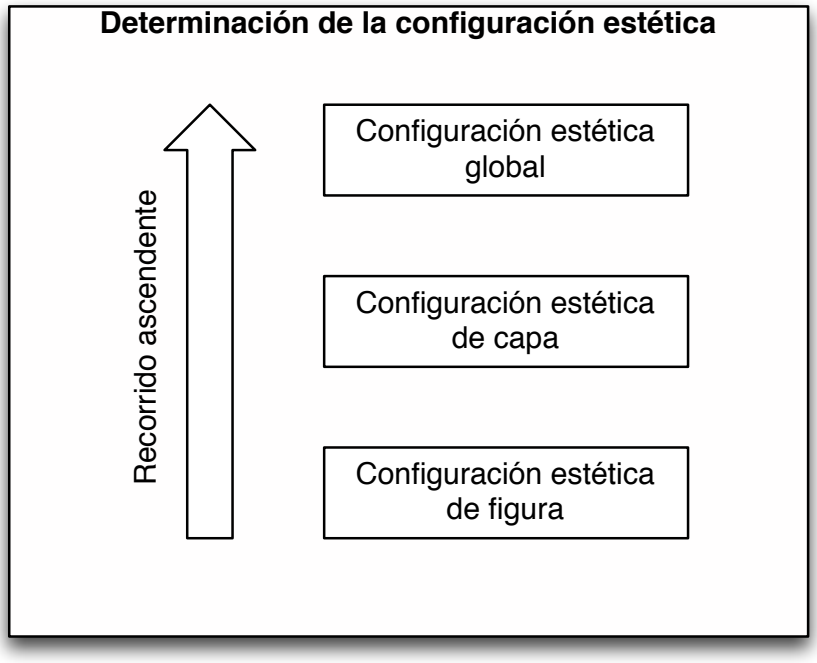

Ilustración 56: Procedimiento de obtención de la configuración estética de una figura

Si una figura no tiene una configuración estética asociada, se obtiene ésta de la capa que la contiene. Si la capa no tiene esa información se accede a la configuración estética global del sistema.

9.4. Diseño de la aplicación de demostración

La aplicación de demostración tiene por objetivo mostrar y hacer uso de toda la funcionalidad ofrecida por el componente visor. Pretende ser un punto de acceso directo a las propiedades, funciones y eventos que suministra el componente.

El diseño que se realiza en esta sección se centra en los aspectos estructurales y organizativos de la aplicación.

9.4.1. Estructura

La siguiente ilustración muestra la guía principal establecida para el desarrollo de la aplicación de demostración.

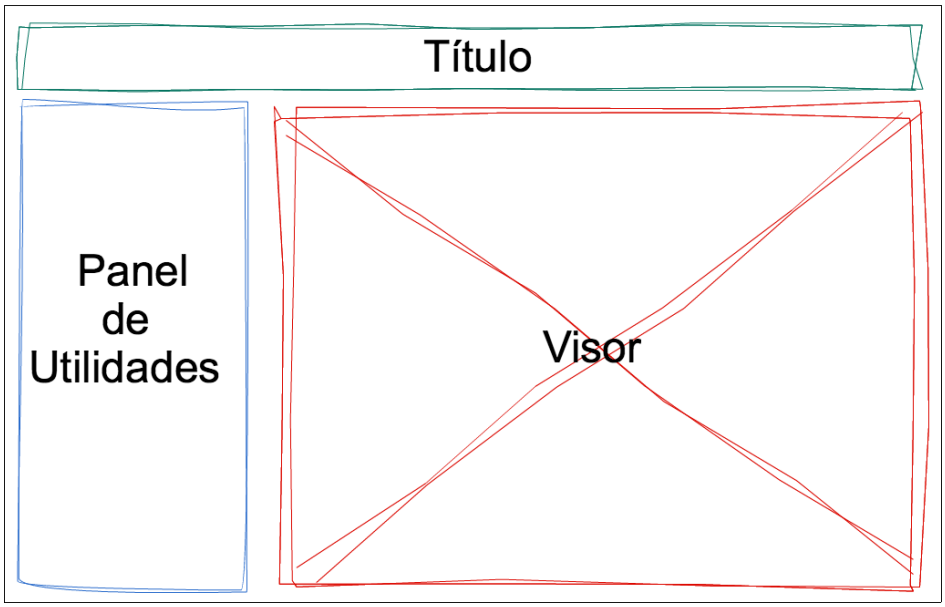

Ilustración 57: Boceto principal

El boceto realizado está compuesto por una parte principal donde se muestra el visor, junto a un panel de configuración y manejo donde se acceden a las distintas funciones del componente, con la adición de un logo o título representativo de la aplicación.

Se pretende con este esquema presentar y focalizar la atención del usuario en el principal elemento de la aplicación, el visor, sin dejar de lado su cometido, ofreciendo un panel de utilidades para la demostración de las funciones del componente.

9.4.2. Panel de utilidades

Su principal cometido es estructurar y organizar el acceso a la funcionalidad dada por el componente. A continuación se presentan los 2 diseños realizados para su especificación.

Opción A

Se presenta un diseño por pestañas para la organización de las distintas opciones, tal como muestra el siguiente boceto.

Ilustración 58: Configuración en pestañas tradicionales

Opción B

Como alternativa a la opción A, se establece una opción organizada en pestañas apiladas o en acordeón.

Ilustración 59: Configuración en pestañas apiladas

9.4.3. Título

Dado que el enfoque de esta aplicación es meramente demostrativo, el título elegido para la aplicación es MViewer Demo.

Capítulo 10.- Implementación

Esta fase del desarrollo se caracteriza por la construcción del producto, llevándose a cabo las tareas de codificación de la aplicación. Ya que es una tarea meramente constructiva, en este capítulo se mostrarán los resultados de este proceso.

10.1. Componente visor

Se ha construido un componente visor de planos que permite una alta adaptación a diversos entornos. El producto tiene 2 vistas principales, sin cargar el plano y con él cargado.

La primera vista permite mantener el visor en un estado sin plano. Esta vista es configurable por el Diseñador. Permite modificar los aspectos gráficos de los elementos señalados en la ilustración 60, el fondo y el borde.

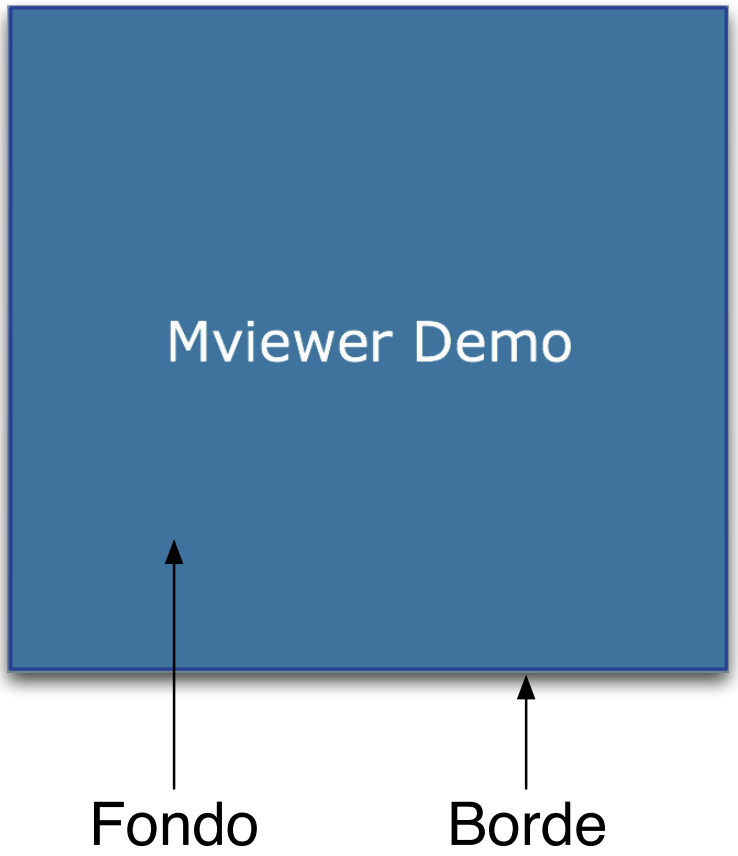

Ilustración 60: Vista sin el plano cargado

La segunda vista, y la más importante es la vista de presentación del plano y las figuras registradas sobre el mismo. Esta vista esta formada por el visor y el conjunto de herramientas dadas al Usuario para interactuar con el componente.

Son configurables a nivel gráfico en visibilidad, posición y en transparencia. La siguiente ilustración muestra esta descomposición.

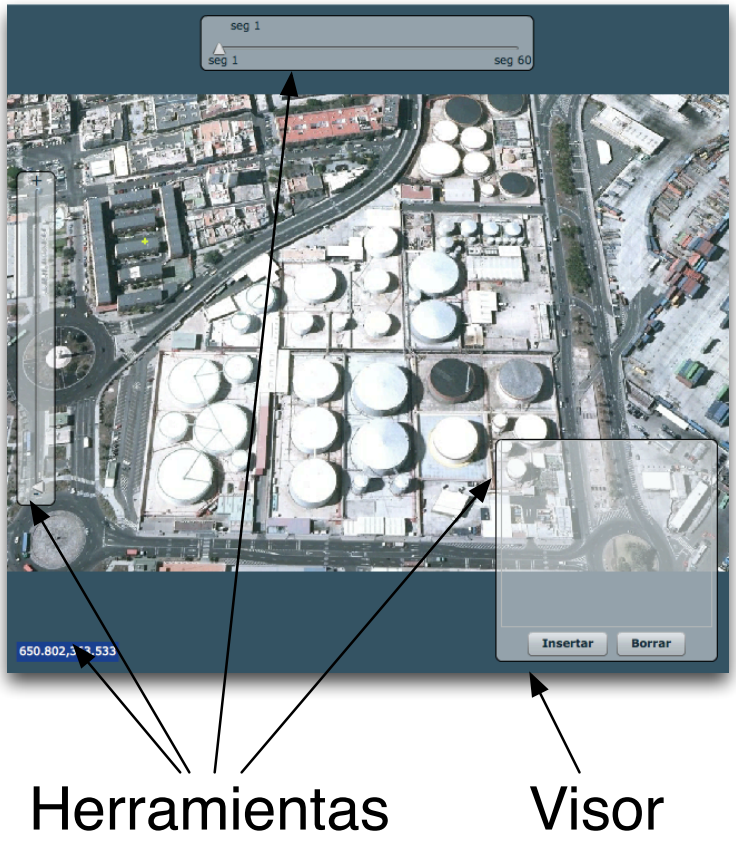

Ilustración 61: Elementos de la vista principal

El visor se encarga de presentar el escenario planteado, esto es, mostrar una imagen localizada en unas coordenadas con unas figuras posicionadas sobre el mismo.

Para ilustrar la respuesta del visor se plantea una situación ficticia como ejemplo.

Ejemplo

En el puerto se quiere simular la explosión de un tanque de combustible y determinar que recursos son afectados por los gases liberados. Para ello, se posicionan el accidente *Accidente 1* y los extintores *Ext 0..5* en el instante inicial, t=1s.

Ilustración 62: Escenario planteado

Se realiza un proceso de simulación que determina la zona afectada por los gases emitidos por la explosión del combustible. Se obtienen los valores principales del radio de la zona afectada en los instantes de tiempo 2, 30, y 60 segundos. La siguiente tabla y diagrama muestran este comportamiento dinámico.

Instante de tiempo	Radio
1	0
2	75
30	50
60	10

Tabla 99: Relación instante de tiempo-radio del ejemplo

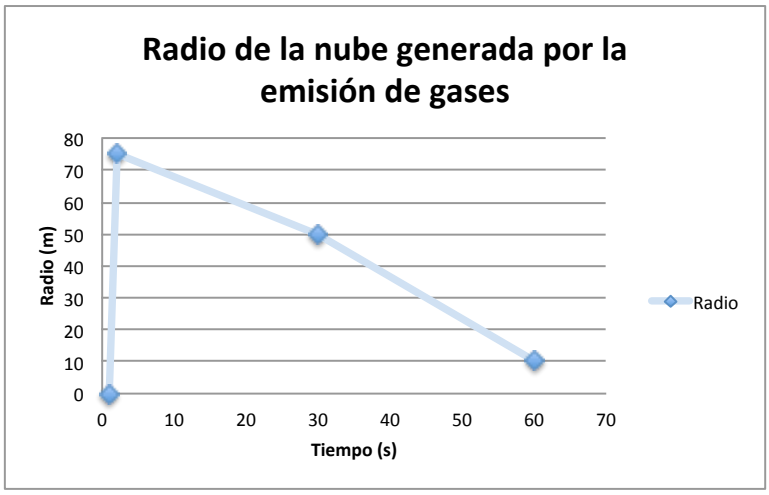

Ilustración 63: Evolución del radio de la nube de gases emitida

El visor se encarga de representar estos valores en los instantes correspondientes, interpolando cualquier valor intermedio entre 2 instantes registrados.

Las siguientes ilustraciones muestran la evolución de la nube generada.

La primera ilustración muestra el escenario original con el accidente y los extintores posicionados.

Ilustración 64: Escenario planteado antes de producirse la explosión en t=1s

Justo en el momento después de producirse la explosión (ilustración 65), la nube de gases obtiene el máximo radio. La zona afectada alcanza a los extintores Ext 3, Ext 4 y Ext 5.

t=2 s

Ilustración 65: Escenario planteado en los primeros instantes de la explosión, t= 2s

Transcurridos 30 segundos, el área afectada por los gases disminuye respecto al instante inicial. En este escenario, los extintores Ext 3 y Ext 5 ya no se encuentran bajo la nube de gases, por lo que quedan libres para su acceso. Sin embargo, el extintor Ext 4 sigue estando afectado.

t=30 s

Ilustración 66: Escenario planteado en el instante t=30s

Pasados 60 segundos la zona afectada por los gases emitidos es casi nula.

Ilustración 67: Escenario planteado en el instante t=60s

Tomando las consideraciones de este ejemplo simbólico, es sencillo tomar medidas en caso de la ocurrencia de un accidente. Se aprecia como el visor se comporta de forma correcta para representar las entidades y los resultados de las simulaciones realizadas sobre estas.

Este ejemplo es una pequeña muestra de las capacidades de que dispone el visor, pudiendo representar, además de puntos, círculos e imágenes, figuras en forma de líneas, o texto.

10.2. Documentación web

La utilización del componente viene dada por el uso de su API; esto es, el conjunto de propiedades, funciones, clases y eventos a las que el Diseñador tiene acceso.

Esta API ha sido documentada en 2 versiones, una en formato físico, papel (ver Anexo VI: Application Programming Interface(API)), y otra en formato digital, una página web. Se puede acceder a esta última versión en el cd que se adjunta, en la siguiente ruta "***./Documentación Web/index.html***". Esta documentación proporciona una información detallada que contiene de forma estructurada los paquetes, clases, métodos y parámetros de entrada y salida necesarios para el uso del componente.

La página desarrollada tiene 2 secciones, una barra izquierda donde navegar por los distintos paquetes y clases, y la zona central donde se expone de forma ordenada la distinta información.

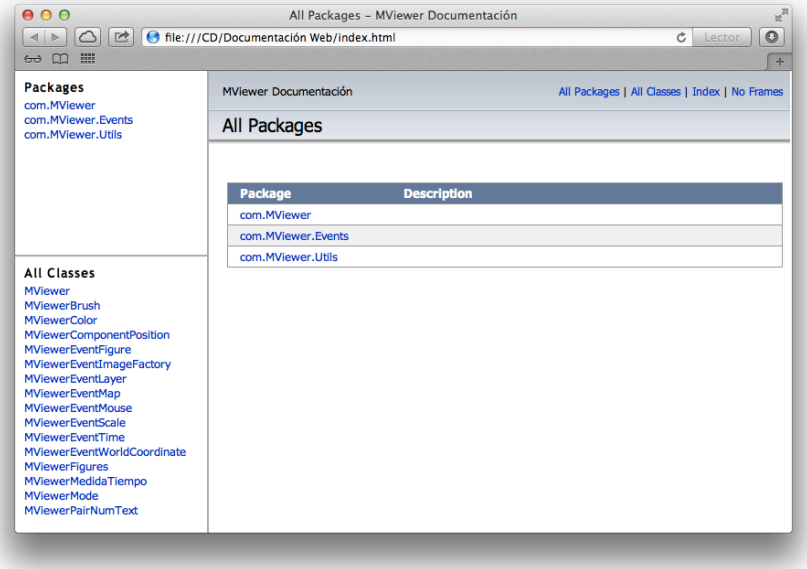

Ilustración 68: Página principal de la documentación web

Al acceder a una clase se obtiene su descripción y, se listan y describen las propiedades, funciones y eventos que tiene.

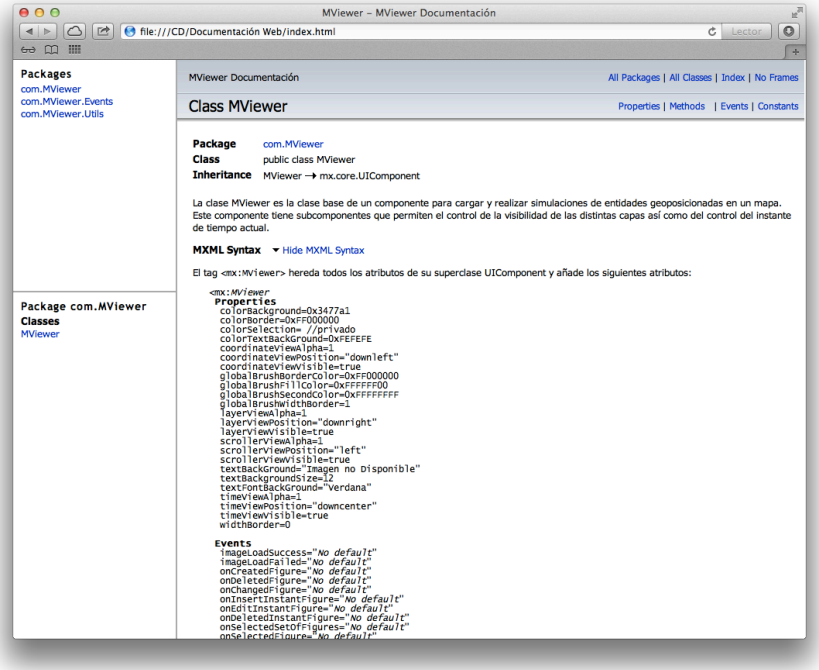

Ilustración 69: Vista interna de la clase MViewer

Dentro de cada clase se listan e indexan de forma tabulada las propiedades, funciones y eventos emitidos.

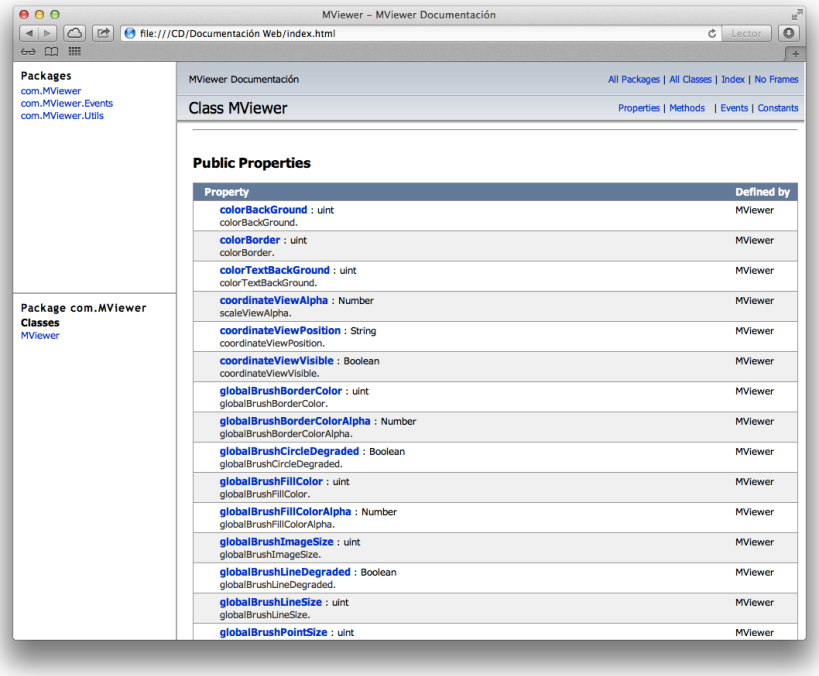

Ilustración 70: Listado de propiedades de la clase MViewer

El detalle de una función viene por una descripción de la misma y de los parámetros de entrada y salida de la misma.

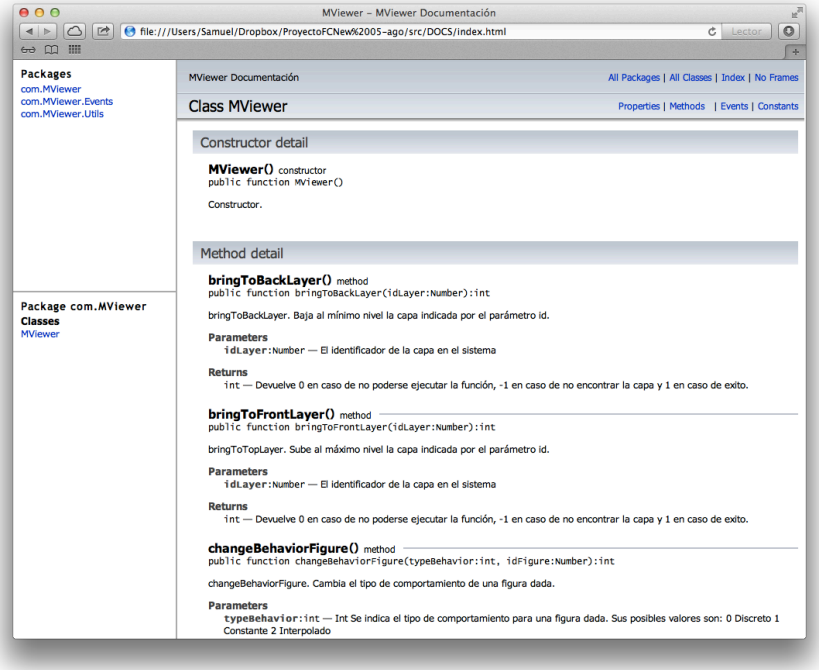

Ilustración 71: Descripción de funciones de la clase MViewer

10.3. Aplicación de Demostración

La aplicación creada sigue las pautas directas indicadas en el diseño. Esta formada por un área principal donde se encuentra el visor y un panel lateral donde se accede a las herramientas del mismo.

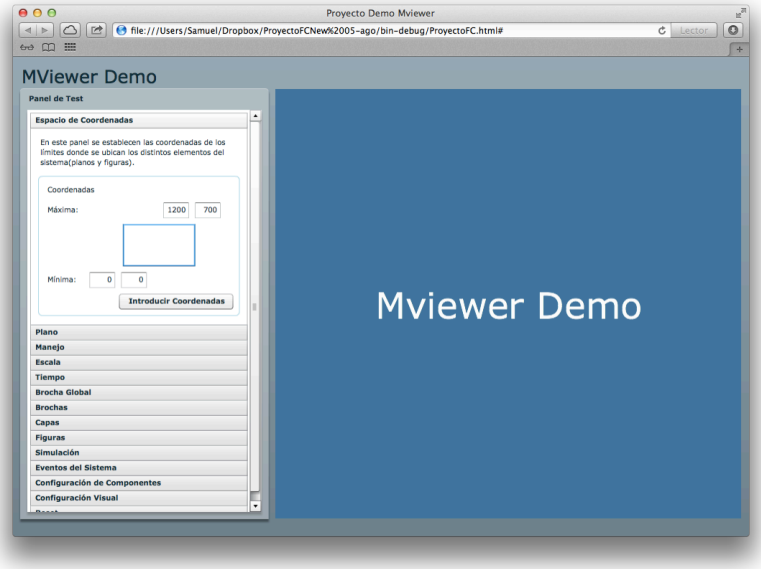

Ilustración 72: Aplicación de demostración

La solución final comenzó utilizando la opción A planteada en el diseño, una estructura basada en pestañas. Esta solución no llegó a ser la final porque a medida que iban aumentando el número de opciones, la estructura iba degenerando al colapsar los títulos de las pestañas, dificultando su visibilidad.

Finalmente se optó por utilizar la opción B del diseño, un panel estructurado en pestañas apiladas o en acordeón, para clasificar la funcionalidad del componente.

Las secciones desarrolladas en el panel lateral intentan plasmar toda la funciones dada por el visor. La clasificación es la siguiente:

- **Espacio de coordenadas**.- Establece las dimensiones del espacio de coordenadas, donde se registran el resto de elementos presentables por el visor.
- **Plano**.- Carga el plano indicando las coordenadas de su posición.
- **Manejo**.- Selecciona el modo de funcionamiento del visor.
- **Escala**.- Cambia los parámetros de la escala del sistema.
- **Tiempo**.- Establece los tiempos del visor.
- **Brocha Global**.- Configuración gráfica global.
- **Brochas**.- Crea, elimina y modifica el conjunto de configuraciones creadas.

- **Capas**.- Crea, edita y elimina capas del sistema.
- **Figuras**.- Crea, edita y elimina figuras en el sistema. Además permite establecer comportamientos avanzados a las figuras.
- **Simulación**.- Reproduce el escenario actual, esto es, recorre los instantes de tiempo configurados del sistema, para ir viendo cambios en la escena presentada.
- **Eventos del sistema**.- Muestra los eventos lanzados por el visor.
- **Configuración de componentes**.- Configura la posición, visibilidad y transparencia de las herramientas integradas en el visor.
- **Configuración visual**.- Cambia la configuración visual del componente.
- **Reset**.- Reinicia el escenario establecido.

La construcción de los distintos apartados se ha realizado de forma que ofrezcan un estilo lo más homogéneo posible. Todos ofrecen un estilo genérico compuesto por una descripción y secciones en forma de recuadro donde se clasifican las distintas opciones de manejo del visor. La siguiente ilustración muestra este esquema.

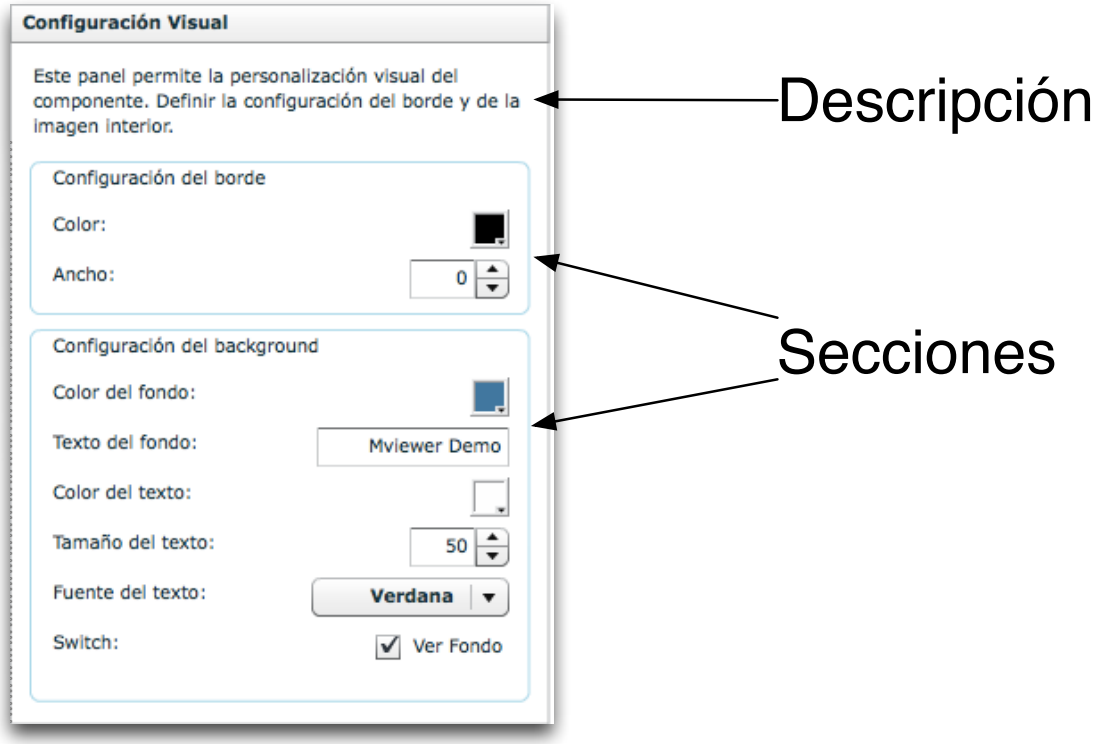

Ilustración 73: Estructura principal de una pestaña del panel de herramientas

Las pestañas son apoyadas por ventanas flotantes para la realización de operaciones que requieren un conjunto de datos más específico, como pueden ser las operaciones de creación, modificación y aspectos avanzados.

Ilustración 74:Vista de las ventanas flotantes

Capítulo 11.- Transición

Una vez que el sistema ha alcanzado cierta capacidad operativa, el proyecto entra en la fase de transición. Se considera que el sistema ofrece la confianza suficiente como para operar en el entorno del cliente, aunque no sea necesariamente perfecto y sea susceptible de modificaciones. Esto permite que el cliente pueda descubrir problemas, riesgos y defectos, así como descubrir la necesidad de nuevas características.

Principalmente en esta fase se han ido realizando entregas de los prototipos al cliente principal del componente visor. Esto ha permitido descubrir errores y nuevas necesidades no descubiertas en las fases de requisitos y de implementación.

11.1. Productos entregables

Esta entrega la forman los siguientes productos:

- Componente visor.
- Documentación web de la API suministrada.
- Aplicación de demostración.

Además en esta fase, se ha creado una página web donde ofrecer información relacionada con el proyecto y proporcionar un enlace directo a estos productos desarrollados. Actualmente se encuentra alojada en un servidor proporcionado por la División del CEANI del Instituto Universitario SIANI; se puede acceder a ella a través de la siguiente dirección http://193.145.154.10:8082/WebMViewer.

11.2. Página web

Las siguientes ilustraciones muestran la interfaz actual de la web.

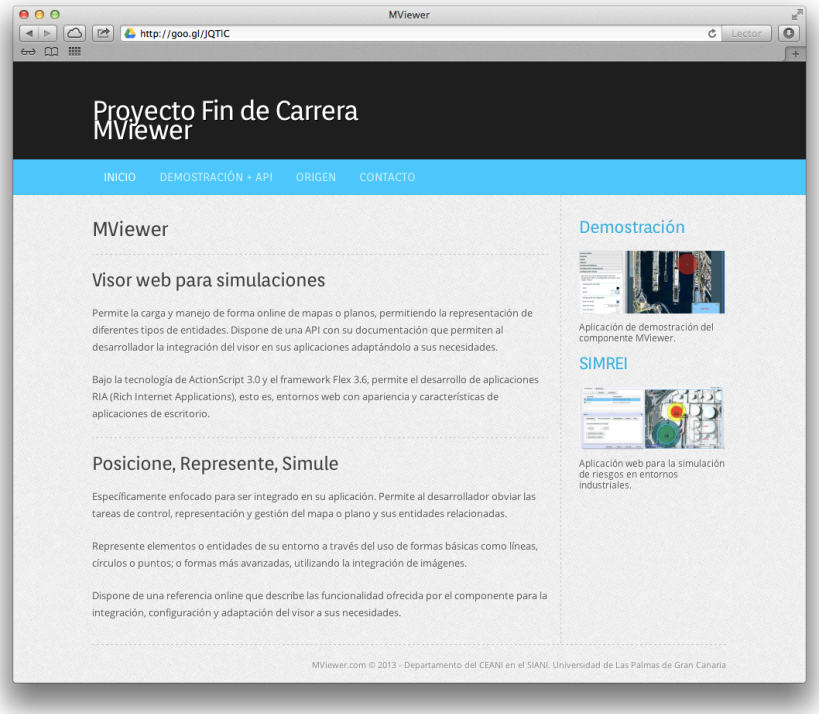

Ilustración 75: Portada de Inicio de la página web

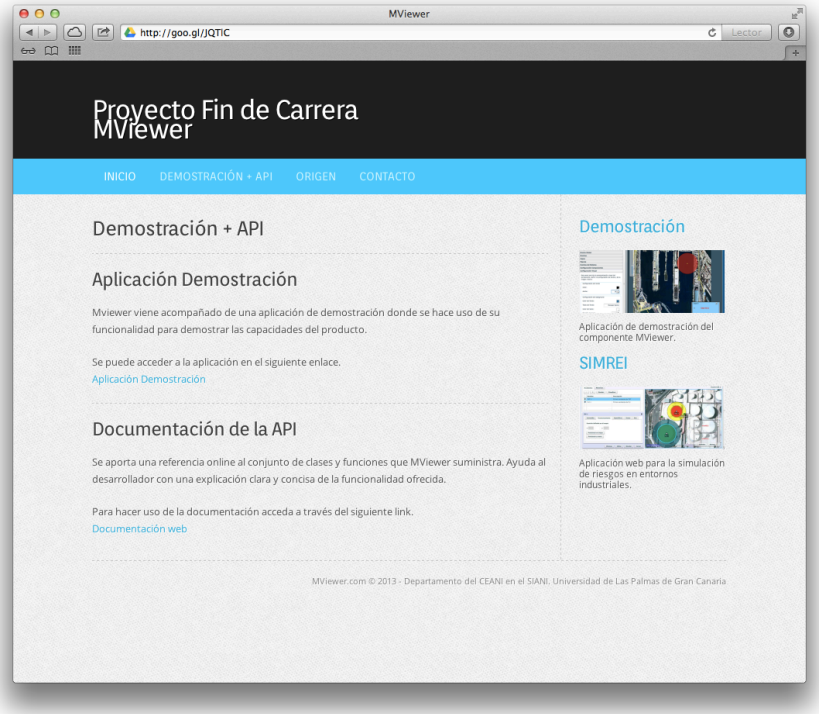

Ilustración 76: Sección Demostración + API

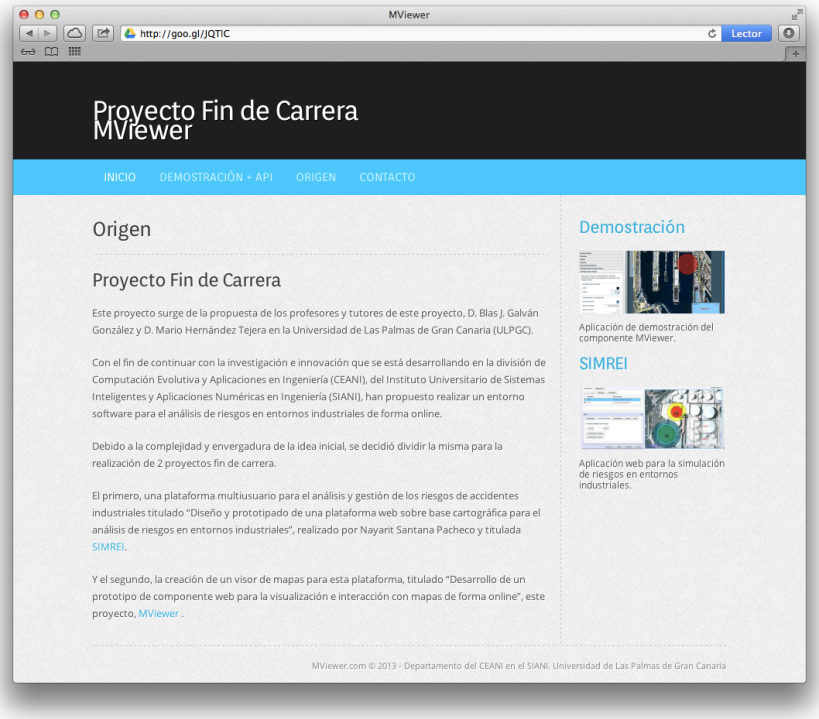

Ilustración 77: Sección Origen

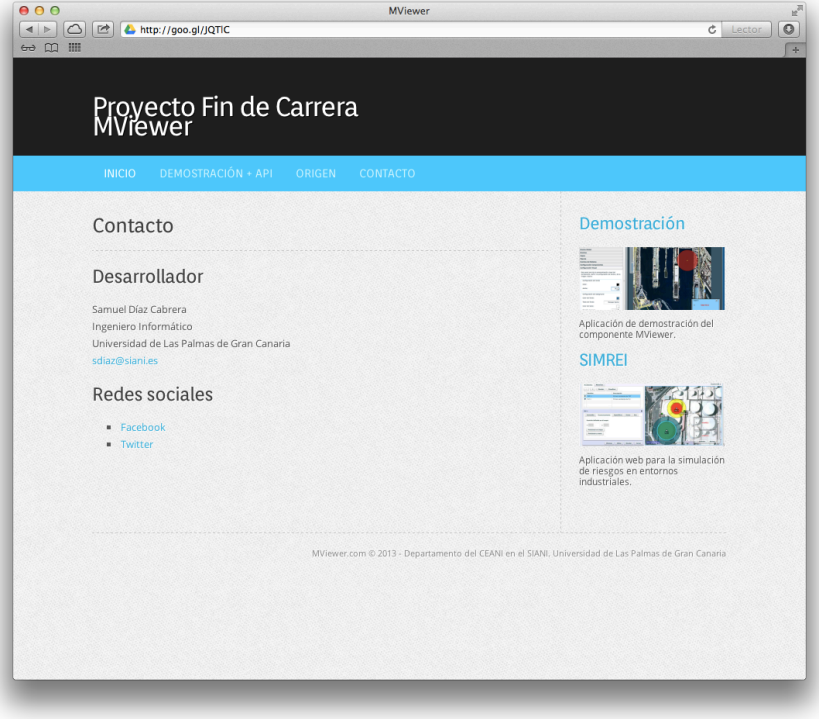

Ilustración 78: Sección Contacto

Cuenta con las siguientes secciones:

- **Inicio**.- Explicación de las características principales del componente.
- **Demostración + API**.- Explicación y acceso a estos elementos.

- **Origen**.- Explicación del origen del desarrollo del componente.
- **Contacto**.- Referencia del desarrollador.

11.3. Aplicación del Cliente

La fase de transición se realizó teniendo contacto permanente con el cliente. En este caso se trata de otro proyecto de fin de carrera titulado "*Diseño y prototipado de una plataforma web sobre base cartográfica para el análisis de riesgos en entornos industriales*" desarrollado por Nayarit Santana Pacheco.

Este caso de uso ha adaptado el funcionamiento del componente visor de planos para la representación de accidentes y la gestión de los riesgos producidos por estos en un entorno multiusuario.

En esta fase donde se ha llevado a cabo la integración del componente en esta aplicación, el cliente ha contado con toda la ayuda posible. Principalmente se han realizado las siguientes actividades:

- Reuniones para llevar a cabo la integración del visor.
- Encuentro de nuevas necesidades.
- Localización y corrección de errores.

Las siguiente ilustración muestran el visor integrado en la aplicación del cliente.

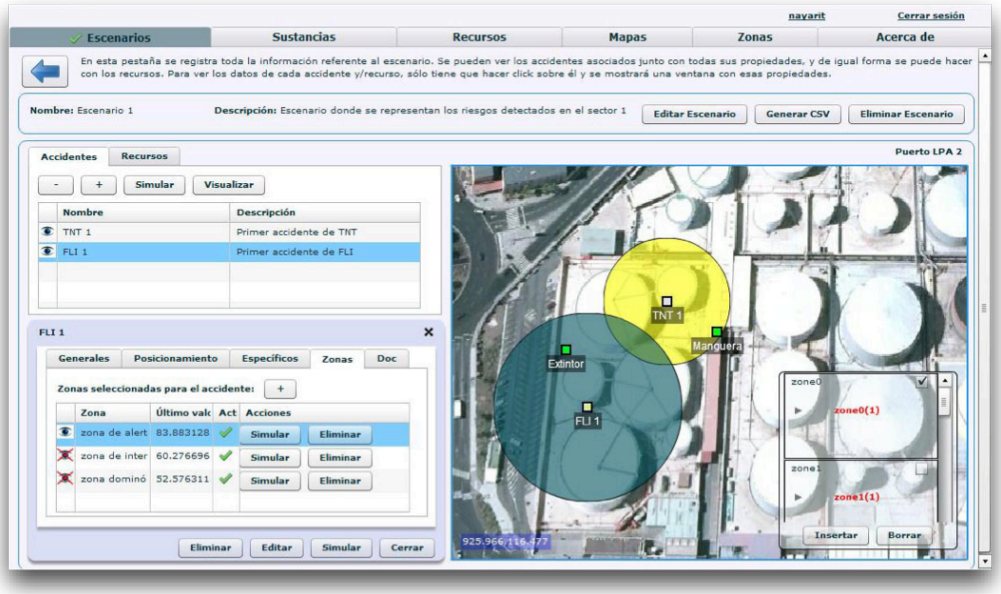

Ilustración 79:Vista de simulación de la aplicación SIMREI

Capítulo 12.-
Conclusiones y Principales Aportaciones

Para comenzar, me gustaría dejar patente que el proyecto me ha brindado la oportunidad de enfrentarme a un problema complejo y completo. Elaborando la solución desde la captación de requisitos hasta la entrega del mismo, algo que nunca había realizado.

Un proyecto complejo por multitud de factores; el tener que afrontar la creación de un producto con unas dimensiones considerables; el aprender un lenguaje nuevo, o el desarrollar un producto que iba a ser integrado en otro proyecto; han sido algunas de ellas. Pero, la mayor de las dificultades ha estado en la carga gráfica del mismo; ya que, durante el desarrollo de la carrera sólo he tenido una asignatura orientada a la programación gráfica, por lo que mi dominio de las interfaces gráficas era mínima.

Un proyecto completo por la cantidad de detalles a desarrollar y a tener en cuenta. No sólo se trataba de crear un visor; al tratarse de un componente integrable se requería un encapsulamiento del producto, ofreciendo métodos de entrada y de salida. Por ello, el componente debía documentar estas interfaces y ofrecer un acceso sencillo y cómodo a la misma dando lugar a la creación de la referencia web de la documentación. Por otro lado, se requería mostrar la funcionalidad dada y también ofrecer junto con el componente un ejemplo del uso de la API proporcionada originando la creación de la aplicación de demostración.

Durante el tiempo de desarrollo se ha realizado un proceso ingenieril completo. Ha cubierto todas las etapas ingenieriles del desarrollo de software; la captación y el análisis de requisitos, el diseño de la solución, su implementación y la transición o entrega de los productos desarrollados.

Produce especial satisfacción observar los resultados obtenidos, por la gran cantidad de actividades diferentes realizadas, la construcción, la documentación o la entrega, pero sobre todo por el hecho de que el visor ya está siendo utilizado en otro desarrollo.

Por otra parte, se ha creado un producto que mejorará las labores de investigación realizadas en la División del CEANI del Instituto Universitario SIANI. Permite desarrollar nuevas aplicaciones de simulación online que incorporen características de geo posicionamiento, lo que facilita la operativa del personal al dar una mayor flexibilidad al poder acceder a las mismas desde cualquier navegador.

Ya que el proyecto se ha realizado utilizando la tecnología Flash y a que ésta se encuentra en entredicho en los últimos años, comparándose con HTML 5, me gustaría terminar reflejando mi opinión personal sobre este debate tecnológico.

12.1. Flash vs HTML 5, Opinión personal

Quiero comenzar esta exposición afirmando que HTML 5 es una tecnología con un futuro prometedor, con vistas de ser ampliada, y seguro que será la tecnología que predominará en los desarrollos web en los próximos años.

También quiero recalcar que al comienzo de este proyecto no existía este debate. Todavía no habían aparecido las primeras declaraciones a favor de HTML 5 y en contra del uso de Flash. Los principales argumentos en contra del uso de Flash aparecieron por:

- Sus problemas de seguridad.
- El consumo energético que conlleva su utilización.

Todo esto aparece en un momento de despegue tecnológico de los dispositivos móviles; terminales con una batería limitada y en los que se pretende prologar al máximo la duración de esta.

En el momento actual, Adobe ha solucionado bastantes problemas de seguridad y utiliza la tecnología Adobe Air basada en flash para los terminales móviles. La marca Flash como tal, ha quedado marcada por este debate, lo que les ha llevado a perder la utilización del plug-in Adobe Flash Player en gran parte de los terminales móviles.

A pesar del declive producido, expondré el por qué todavía es útil utilizar una tecnología como Flash para un desarrollo como el que se ha realizado, en lugar de comenzar a desarrollar en HTML5.

- HTML 5 todavía no es un estándar, la previsión actual indica que se estandarizará a finales del año 2014. A partir de este año, se ampliará esta versión con las versiones HTML 5.1 y HTML 5.2.
- En el momento de redacción de esta memoria, no todos los navegadores soportan de igual manera todas las características del borrador HTML 5. En cambio, existen plug-ins de Flash para prácticamente todos los navegadores que se pueden encontrar en un entorno de escritorio.
- Al ser un borrador, sus características no son definitivas, por lo que todavía es susceptible a futuros cambios.
- HTML 5 todavía no cuenta con un conjunto de herramientas comparables a las que proporcionaba Adobe con su framework Flex 3.0 (versión utilizada en este proyecto)
    - No existe ninguna librería de componentes gratuita en HTML 5, lo suficientemente integrada y probada para la realización de un

proyecto complejo. Esta carencia complicaría la realización del proyecto en tareas que no pertenecen a la temática del problema.

- En cambio, Adobe con el framework Flex 3.0, proporciona todo un conjunto de librerías con herramientas útiles testeadas y válidas, así como la documentación necesaria para centrar las tareas de ingeniera solo en el objetivo del proyecto.

- En el marco de la realización de un proyecto de carrera, donde el tiempo es limitado, la investigación de la tecnología HTML 5, y el desarrollo de componentes secundarios, supondría un coste adicional e innecesario.

Para finalizar, recalcar que Flash todavía no ha dejado de estar soportado por los principales actores de la industria del software, p.ej. Google sigue utilizándolo en varios de sus productos más populares, el navegador Google Chrome y la plataforma de reproducción de videos Youtube.

Capítulo 13.- Trabajos Futuros

Si bien se ha creado un producto completo, siempre es susceptible de ser mejorado y ampliado. En este capítulo se hace hincapié en aquellos aspectos que pueden ser mejorados. Se lista además un conjunto de temáticas donde puede ser utilizado el componente visor para la representación de entidades.

13.1. Aspectos a mejorar

- Composición de un plano conforme a varias imágenes.
- Mostrar diferentes imágenes a diferentes niveles de escala.
- Utilización de servidores de planos.
- Ampliación de las características configurables de la apariencia de las formas.
- Representación de formas complejas como polígonos.
- Creación de un subcomponente visor de las propiedades de la entidad seleccionada.
- Creación de una herramienta regla, para medir distancias.
- Posibilidad de establecer un modo de bloqueo, que solo visione, sin mover, ni escalar el plano.
- Adición de modos de interpolación de posición más complejos.
- Adición de interpolación en el color de representación de las formas.

13.2. Temáticas

Se propone además la utilización del visor en los siguientes campos:

- **Posicionamiento online de dispositivos geo localizados**.- Cualquier tipo de dispositivo geo localizado puede ser objeto de un proyecto utilizando el componente visor. Esto permite disponer en la web de un acceso inmediato a su ubicación. Se podría además, plantear el mostrar la trayectoria que ha seguido el dispositivo en un periodo de tiempo, o mostrar información variada de algún tipo de evolución que haya seguido el dispositivo, o mostrar las condiciones medioambientales a las que está sometido el dispositivo.

- **Generación de planes integrales de seguridad**.- En el ámbito de la seguridad de instalaciones, se realizan informes de seguridad donde se determina el grado de seguridad del recinto y se toman decisiones sobre que elementos son necesarios y donde deben estar posicionados. Una herramienta enfocada a la elección y posicionamiento de elementos de seguridad, y a la determinación del grado de seguridad que se establece en la instalación, puede minimizar el coste de la implantación del plan integral.

- **Organizador de eventos lúdicos**.- Aplicación para la organización de recursos humanos y técnicos en un recinto cerrado. El visor cubriría la faceta de posicionamiento y selección de recursos. Esta aplicación podría tener dos tipos de usuarios, el organizador y el trabajador que recibe directrices. El organizador ve la ubicación de sus recursos y en tiempo real podría ordenar un cambio de posición a los mismos, o el ejecutar algún tipo de tarea.

Bibliografía

Cole, A.
2008 *Learning Flex 3: Getting up to Speed with Rich Internet Application*. United States of América, O´Reilly Media, Inc. 2008. 304p

Donatis, A. de
2006 *Advanced ActionScript components: Mastering the Flash components*. United States of América, FriendSof. 2006. 558p

Durán, A. & Bernárdez, B.
2002 *Metodología para la Elicitación de Requisitos de Sistemas Software*. Sevilla 2002, Edición 2.3 en http://www.lsi.us.es/~beat/. 82p.

Gamma, E. & Helm, R. & Johnson, R. & Vlissides, J.
2003 *Patrones de diseño: Elementos de software orientado a objetos*. Traducción en Madrid. Pearson Educación, S.A .2003. 384p.

Kazoun, C. & Lott, J.
2008 *Adobe Flex 3*. United States of America, O´Reilly Media, Inc. 2008. 636p.

Larman, Craig
2003 *UML y patrones : una introducción al análisis y diseño orientado a objetos.* Editorial Alhambra S. A. 2003. 590p.

Lott, J. & Patterson, D.
2007 *ActionScript 3 with Design Patterns*. United States of America, Peachpit Press.2007. 286p.

Mook, C.
2008 *ActionScript 3.0*. Traducción en Madrid. Anaya Multimedia, S.A. 2008. 1088p.

Nielsen, J.
2000 *Usabilidad. Diseño de sitios web*. Traducción en Madrid. Pearson Educación S.A. 2000. 432p.

Peters, K.
2007 *ActionScript 3.0 Animation Making Things Move!*. United States of America, FriendSof. 2007. 542p.

Rumbaugh, J. & Jacobson, I. & Booch, G.
2000a *El lenguaje unificado de modelado: Manual de referencia*. Traducción en Madrid. Pearson Educación, S.A. 2000. 552p.
2000b *El Proceso Unificado de Desarrollo de Software.* Traducción en Madrid. Pearson Educación, S.A. 2000. 434p.

Shupe, R. & Rosser, Z.
2008 *Learning ActionScript 3.0 : a beginner's guide*. Canadá. 2008. 363p.

Anexo I.- Arquitectura de una RIA

Las aplicaciones de tipo RIA, de forma general tienen un cliente separado de su capa de servicios. Las RIA se basan en una arquitectura de tipo cliente-servidor. Esta compuesto por los siguientes elementos:

- **Cliente**.- Maneja la interacción entre el usuario y la interfaz del usuario. El usuario invoca comandos, actualiza vistas y carga datos. Se mantiene un estado de la aplicación, se gestionan las peticiones que se realizan al servidor y se controla la presentación de los datos.

- **Servidor**.- Se gestionan las peticiones recibidas del cliente produciendo una acción en el servidor, un proceso que trata datos del cliente, o del propio servidor produciendo una respuesta en el cliente. Generalmente dispone de una base de datos para almacenamiento de datos del cliente.

De forma esquemática, la arquitectura de una RIA tiene el siguiente diagrama:

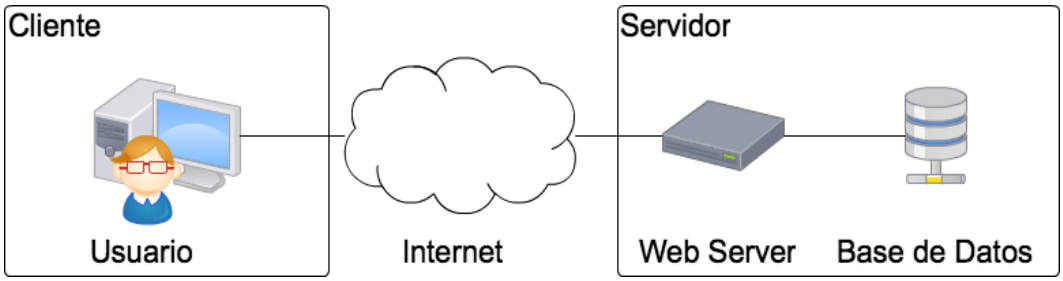

Ilustración 80: Arquitectura tradicional Cliente-Servidor

1.1 Características claves

- **Comunicaciones avanzadas**.- Permiten nuevas tecnologías de comunicación, así como ofrecen un soporte para entradas y salidas asíncronas.

- **Complejidad**.- Las aplicaciones RIA suelen ofrecer soluciones avanzadas que son más complejas y difíciles de diseñar, desarrollar, implementar y depurar que las aplicaciones web tradicionales.

- **Instalación y mantenimiento**.- Solamente se requiere de la instalación de un plug-in o una máquina virtual o sandbox, que generalmente es más rápida que la instalación de una aplicación tradicional. Así mismo, las actualizaciones se realizan de forma automática.

- **Offline**.- Puede ser usada sin conexión a Internet, reteniendo el estado en la máquina cliente.

- **Seguridad**.- Se mejora la seguridad por medio de actualizaciones automáticas y el uso de un sandbox.

- **Rendimiento**.- Al poder realizar operaciones en el cliente, se evita el envío de peticiones al servidor, incrementando su rendimiento. Por el contrario, esta característica requieren un hardware avanzado en la parte del cliente.

- **Riqueza**.- Añaden características avanzadas que no son nativas del navegador.

1.2 Beneficios

A pesar del esfuerzo para coordinar distintas tecnologías en un desarrollo de este tipo, las RIA ofrecen un conjunto de beneficios:

- No necesitan instalación (solo se requiere mantener actualizado el navegador).

- Las actualizaciones del software son automáticas.

- Se puede acceder a ellas desde cualquier ordenador con una conexión a Internet sin depender del sistema operativo que este utilice.

- La utilización de un sandbox añade una capa de seguridad extra que hace que sea menos probable una infección por virus.

- La ausencia de recarga de página dota de un mayor dinamismo a estas aplicaciones.

- Funcionalidad avanzada que no se puede obtener utilizando solo HTML, como por ejemplo, arrastrar y pegar o realizar cálculos en el lado del cliente sin la necesidad de enviar la información al servidor.

Anexo II.- Características de un componente

Un componente es un objeto de software específicamente diseñado para cumplir con cierto propósito. Su principal característica es la reusabilidad. Para conseguir esta característica el componente debe ser:

- Robusto, comprobando la validez de las entradas.
- Capaz de dar mensajes de error apropiados.
- Diseñado pensando en que será utilizado de maneras imprevistos.
- Completamente documentado.

2.1 Principios fundamentales

Los principios fundamentales que debe cumplir un componente son los siguientes:

- **Reusable**.- Los componentes son usualmente diseñados para ser utilizados en escenarios diferentes por diferentes aplicaciones.
- **Sin contexto especifico**.- Los componentes son diseñados para operar en diferentes ambientes y contextos.
- **Extensible**.- Un componente puede ser extendido desde un componente existente para crear un nuevo comportamiento.
- **Encapsulado**.- Los componentes exponen interfaces que permiten al programa usar su funcionalidad. Sin revelar detalles internos, detalles del proceso o del estado.
- **Independiente**.- Los componentes están diseñados para tener una dependencia mínima de otros componentes.

2.2 Beneficios

Los principales beneficios de la utilización de componentes son:

- **Facilidad de Instalación**.- Tras un cambio de versión, los componentes suelen mantener las mismas interfaces, permitiendo el cambio directo de componente sin realizar modificaciones en el código.
- **Coste reducido**.- El uso de componentes de terceros permite disminuir el tiempo de desarrollo, así como el costo.

- **Facilidad de desarrollo**.- Los componentes implementan un interface bien definido para proveer la funcionalidad ofrecida permitiendo el desarrollo sin impactar en otras partes del sistema.

Anexo III.- Patrones

En este anexo se describen los principales patrones utilizados durante el desarrollo de este proyecto.

3.1 Patrón MVC

Este patrón estructura el sistema en tres elementos claramente diferenciados, el modelo, la vista y el controlador. Establece también el sentido de comunicación que existe entre ellos. A continuación se describen brevemente.

- **Modelo**.- Son aquellas clases encargadas de almacenar y utilizar la información del dominio de datos.

- **Vista**.- Pertenecen las clases encargadas de presentar los datos representados por el modelo, además de realizar la interacción con el usuario.

- **Controlador**.- Se encargan de dar sentido a las acciones del usuario. Actúa sobre los datos representados por el modelo.

De forma gráfica el patrón MVC se representa con el siguiente diagrama.

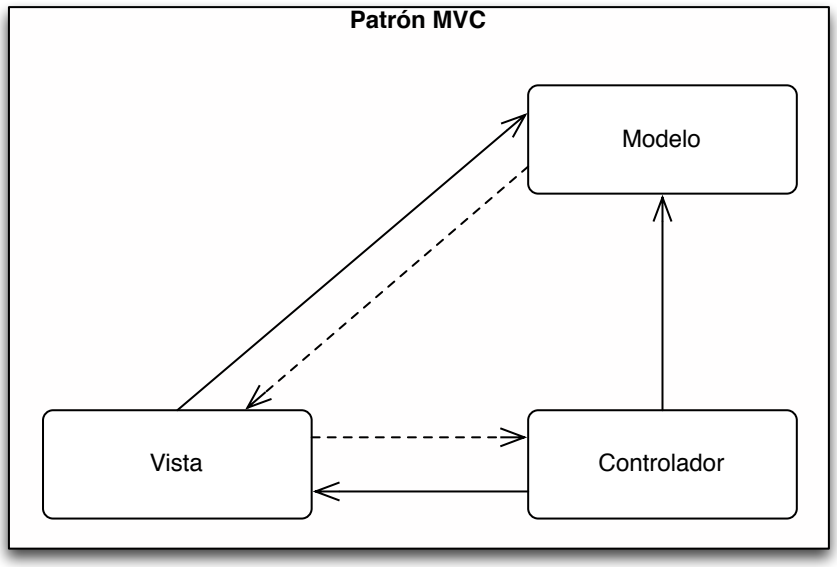

Ilustración 81: Estructura tradicional del patrón MVC

La comunicación que se realiza entre estos elementos es la siguiente:

- El usuario opera con los elementos que forman la Vista.

- La vista notifica que ha ocurrido algo en la vista, produciendo una interpretación de lo ocurrido por el controlador, el cual actúa sobre el modelo.

- El controlador actúa sobre el modelo o sobre la propia vista, alterándola.

- Si el modelo cambia, informa de este hecho, permitiendo que la vista se actualice con la información actualizada.

3.2 Patrón Composite

Se basa en conjuntar objetos simples en estructuras complejas para formar jerarquías permitiendo tratar a la estructura como una de ellos.

El componente realizado es el ejemplo más claro de utilización de este patrón, permitiendo actuar sobre un todo en lugar de cada elemento agregado. El visor hereda de la clase UIComponent y a su vez agrega subcomponentes que también heredan de esta clase.

3.3 Patrón Factory

Se utiliza para crear objetos sin especificar un tipo concreto. Permite disponer de un conjunto de clases fábricas o creadoras que se encargan de construir objetos Producto concretos. Se basa en definir una clase *Creador* de la que heredan *Creador Concreto* que son los que en última instancia crean las clases *Producto*.

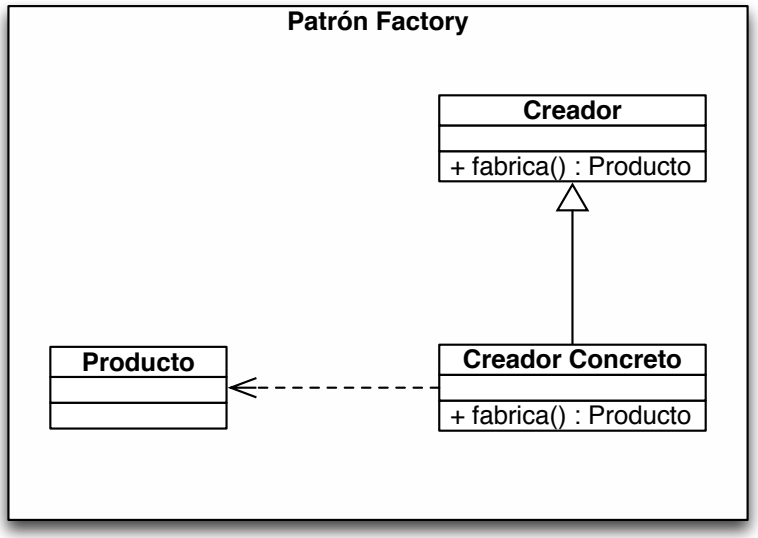

Ilustración 82: Estructura principal del patrón Factory

3.4 Patrón Singleton

Se describe este patrón para escenificar que todos los patrones tienen su lugar y que, a veces, aunque pueda parecer que se pueden utilizar, no son soluciones válidas. Sólo la experiencia en el uso de patrones, da al ingeniero la capacidad de decidir cuales son los adecuados.

El patrón Singleton se utiliza para restringir la creación de objetos pertenecientes a una clase. Se utiliza para garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. La figura 83 ilustra la composición de esta clase.

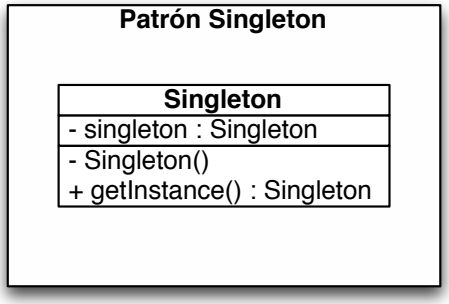

Ilustración 83: Patrón Singleton

La propia clase contiene una instancia a sí misma, y bloquea, al tener un constructor privado, la creación de objetos de la misma. Para su creación proporciona el método *getInstance()*, el cual comprueba si ya existe una instancia creada o no, para crearla.

Este patrón se utilizó para modelar las clase de modelos del sistema, representar clases que a priori deben ser únicas como por ejemplo los subsistemas de tiempo, de escala o de capas.

Este esquema en el que los subsistemas eran únicos, no dio problema mientras se utilizaba en el entorno controlado del desarrollo de este proyecto. Pero al ser suministrado en la fase de despliegue al proyecto SIMREI, se descubrió que no era una solución válida si se requiere la utilización de 2 o más visores, puesto que comparten el mismo modelo, y los mismos subsistemas. Finalmente se optó por la eliminación del uso de este patrón.

Anexo IV.- Detalles sobre la implementación del proyecto

4.1 Implementación de componentes en AS 3.0

Todos los componentes en Flex heredan de la clase *UIComponent*. Este es el componente más básico de la arquitectura introducida por Flex, y de la cual deben heredar el resto de componentes que vayan a ser mostrados en la interfaz de usuario.

Cuando se crea un componente personalizado, se tienen que sobrescribir los métodos de la clase *UIComponent*. Se implementa la estructura básica del componente implementando el constructor y los siguientes métodos heredados de *UIComponent*:

- **Constructor**: debe invocar al método *super()* para llamar al constructor de la clase padre. Se debe utilizar para inicializar las propiedades de la clase. No se deben crear objetos visibles (display objects) en el constructor, para ello esta la siguiente función.

- **CreateChildren**: Se utiliza para crear objetos hijos o otros componentes asociados al que se esta definiendo. No se puede llamar directamente al método. Flex lo ejecuta cuando se realiza una llamada a la función addChild para añadir el componente a su padre.

- **CommitProperties**: Este método se invoca para definir o actualizar las propiedades del componente. Flex realiza una llamada a esta función cuando se realiza una llamada a *invalidateProperties()*. Después de esta llamada, se vuelve a ejecutar *commitProperties()*.

- **Measure**: este método se utiliza para definir el tamaño por defecto del componente. Flex llama a esta función cuando se ha realizado una llamada a *invalidateSize()*. Que ocurre cuando se añade un componente a un contenedor.

- **LayoutChrome**: se utiliza en las clases contenedoras y sus subclases para definir un borde alrededor del contenedor.

- **UpdateDisplayList**: este método se encarga de dibujar el componente, en él se incluirán todas las sentencias de código que tengan que ver con la visualización del componente.

  Sus principales usos son los siguientes:

- Configurar el tamaño y la posición de los elementos del componente a mostrar.
- Para dibujar cualquier elemento visual necesario para el componente.

A continuación se muestra la secuencia de llamadas a estos métodos una vez se ha creado el componente.

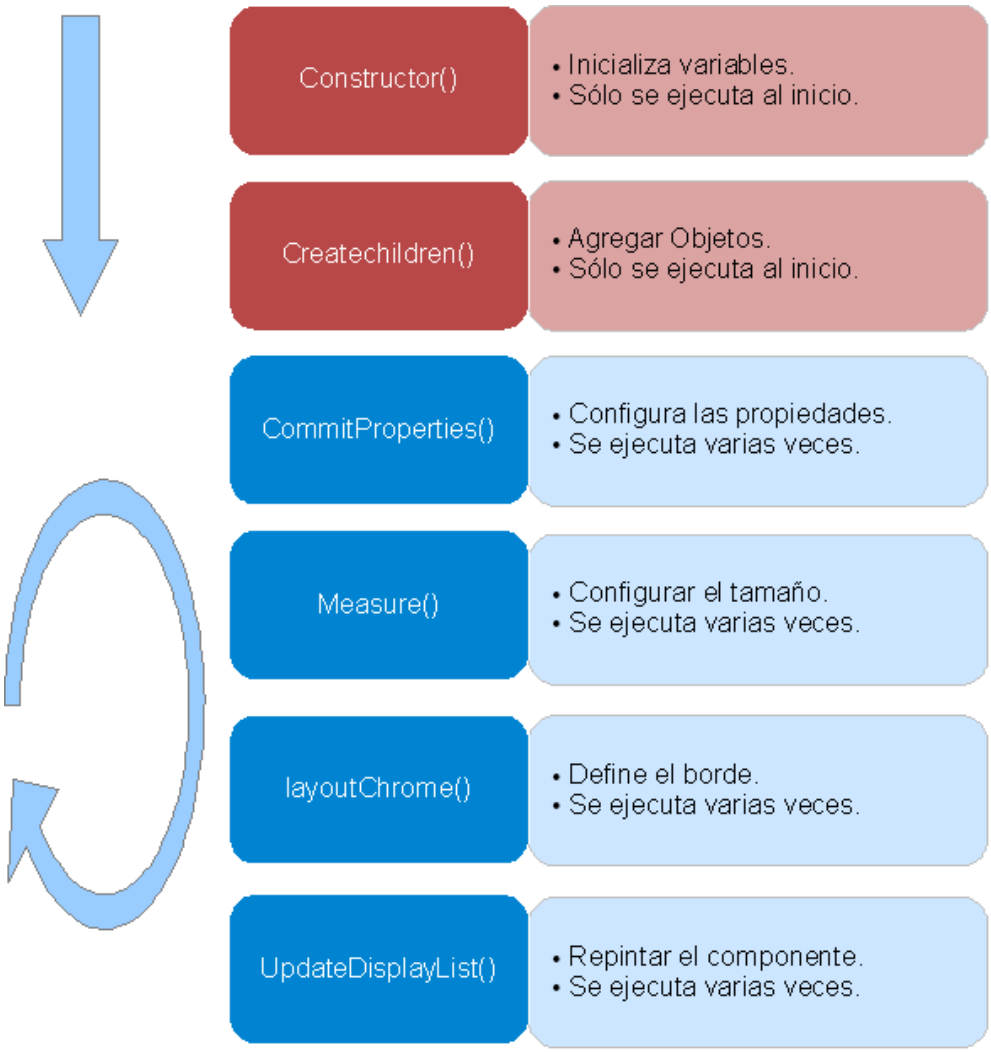

Ilustración 84: Proceso de ejecución de las funciones de la clase UIComponent

El constructor y la función *createChildren()* se ejecutan sólo una vez al inicio de la creación del componente. Los restantes métodos se ejecutan durante la vida del componente.

Es importante conocer, que estos últimos métodos (en azul) no se pueden invocar directamente, si se desean ejecutar se deben utilizar unas funciones pensadas para eso:

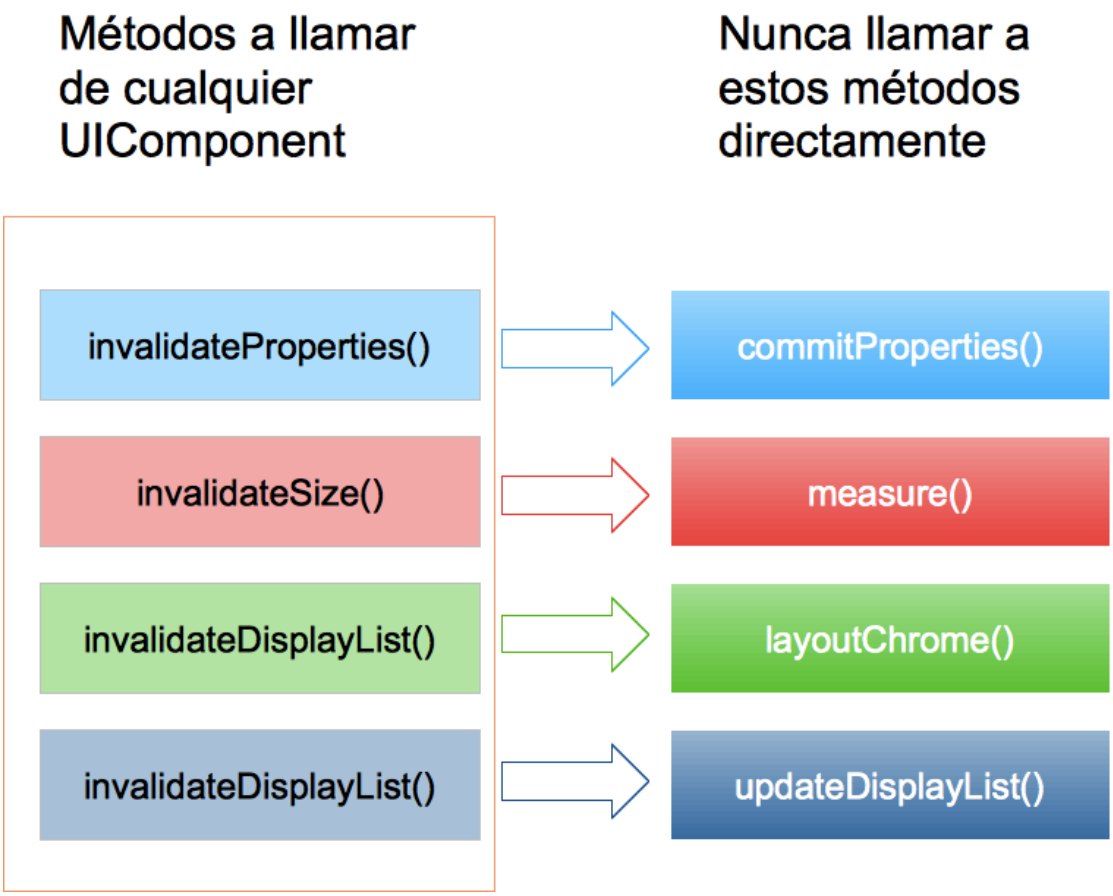

Ilustración 85: Métodos que invocan los métodos privados de UIComponent

Cuando se ejecutan los métodos *invalidateProperties()*, *invalidateSize()* e *invalidateDisplayList()*, se indica al framework de Flex que debe realizar una actualización. Él se encarga de realizar esta acción cuando considere más oportuno llevarla a cabo.

4.2 Estados del visor

Para controlar la ejecución de operaciones cuando no correspondan, se han definido un conjunto de estados para el visor. Estos son los siguientes:

- Creado.- Tras ser instanciado, se encuentra en este estado.
- Configurado.- Pasa a este estado al introducirle las coordenadas del espacio de trabajo.
- Uso.- Pasa a este estado al cargar un plano.

De forma genérica una función dada por la API tiene esta estructura.

```
public function name():int{
        if(isAvailable("name")){
                return internalFunction();
        }
        return 0;
}
```

Se ejecuta la función *isAvailable()* para comprobar si la función puede ejecutarse o no.

4.3 Formato y tamaño de la imagen

La carga del plano está limitada a imágenes de tipo JPG, GIF o PNG y con las siguientes limitaciones en tamaño:

Versión de Flash Player	Limitación en tamaño
V. 9	2880 px en alto y ancho.
V. 10	8191 px en alto o ancho y el número total de pixeles no puede exceder 16,777,215 px.
V. 11	La limitación ha sido eliminada, ahora depende del sistema operativo.

Tabla 100: Relación de tamaños permitidos en las distintas versiones de Flash Player

4.4 Implementación de la documentación web

Esta documentación se ha desarrollado utilizando la herramienta AsDoc que viene incluida en la versión 3.6A del SDK de Flex.

Esta aplicación permite utilizar un tipo de comentarios de ActionScript para documentar las clases desarrolladas. Genera una página web con la descripción de las clases, funciones y propiedades que componen el código.

Para utilizar esta utilidad hay que hacer una llamada a la misma por la línea de comandos. En nuestro caso se ha hecho uso de una herramienta proporcionada por el entorno de desarrollo para la ejecución de aplicaciones externas donde se ha utilizado su comando.

El comando a ejecutar es el siguiente:

```
aasdoc
-source-path .
-doc-sources "./com/MViewer/Mviewer.as"
-output DOCS
-main-title "MViewer Documentación"
-window-title "MViewer Documentación"
-exclude-classes
com.MViewer.Componentes.LayerComp.LayerCompView
com.MViewer.Componentes.LayerComp.LayerView
com.MViewer.Componentes.TimeComp.TimeCompView
com.MViewer.Componentes.ScrollerComp.ScrollerCompView
--
```

Se usan los siguientes parámetros.

Parámetro	Descripción
Source-path	Indica la carpeta que contiene el código a documentar.
Doc-sources	Listado de archivos que debería ser documentado.
Output	Indica el directorio de destino del resultado generado por la aplicación.
Main-title	Especifica el título principal que aparece al comienzo de la página web.
Window-title	Especifica el título que aparece en el navegador.
Exclude-classes	Se listan un conjunto de clases a excluir del proceso.

Tabla 101: Descripción de los parámetros utilizados en la generación de la documentación

Anexo V.- Manual de instalación

Este manual da las claves para comenzar a utilizar el componente desarrollado. Describe los requisitos necesarios para su uso en otras aplicaciones, y muestra los primeros pasos para integrarlo en el entorno de desarrollo y en la propia aplicación.

5.1 Requisitos

El componente desarrollado tiene los siguientes requisitos para la aplicación que lo integre:

- SDK Flex 3.6A
- Navegador compatible con el Plug-in Flash Player 10

5.2 Primeros pasos

La integración del componente visor en un proyecto consiste en la copia de los archivos del mismo en la carpeta del mismo. A continuación es necesario realizar una actualización del proyecto para reconocer los archivos agregados.

Detalladamente los pasos son los siguientes

1. Copia de la carpeta "**./Componente/src/*com***" del cd que se suministra.
2. En la carpeta del nuevo proyecto, pegar la carpeta anterior en el directorio "***src***" del mismo.
3. En el entorno de desarrollo Flex Builder es necesario refrescar el proyecto.
4. !Ya está! El componente se encuentra listo para utilizarse.

A continuación se muestra la creación de un proyecto nuevo en el entorno Flex Builder con la incorporación del componente.

1. Creación del proyecto Flex "*install-Example*" a través de New->Flex Project. Se rellenan los campos con los datos del nuevo proyecto (Nombre, directorio, etc) y por último se finaliza en el botón Finish.

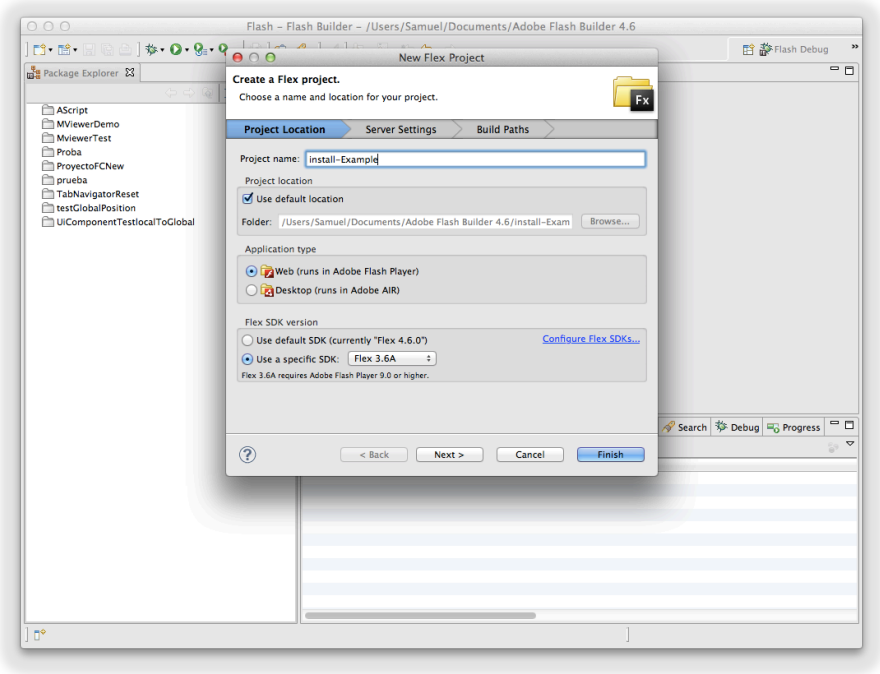

Ilustración 86: Paso 1.- Creación del nuevo proyecto

2. A continuación se copian los ficheros del componente de la carpeta "**./Componente/src/*com***" al directorio "***src***" del nuevo proyecto.

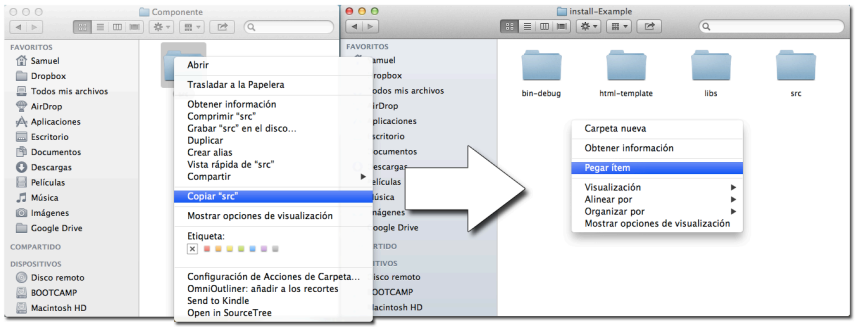

Ilustración 87: Paso 2.- Copiar archivos del componente en el nuevo proyecto

3. Si la agregación del componente no se refleja en el entorno de desarrollo se debe realizar un refresco del proyecto para actualizar la estructura del mismo.

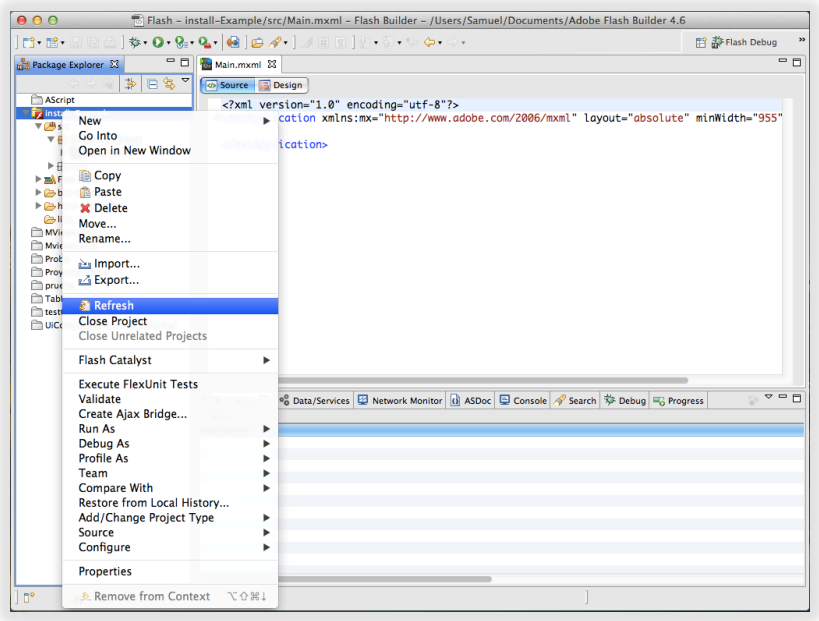

Ilustración 88: Paso 3.- Refrescar el proyecto en el entorno de desarrollo

4. El componente ya forma parte del proyecto "install-Example". Para comprobar que se encuentra agregado se puede comprobar su integración en el panel de componentes de la vista de Diseño del entorno Flex Builder.

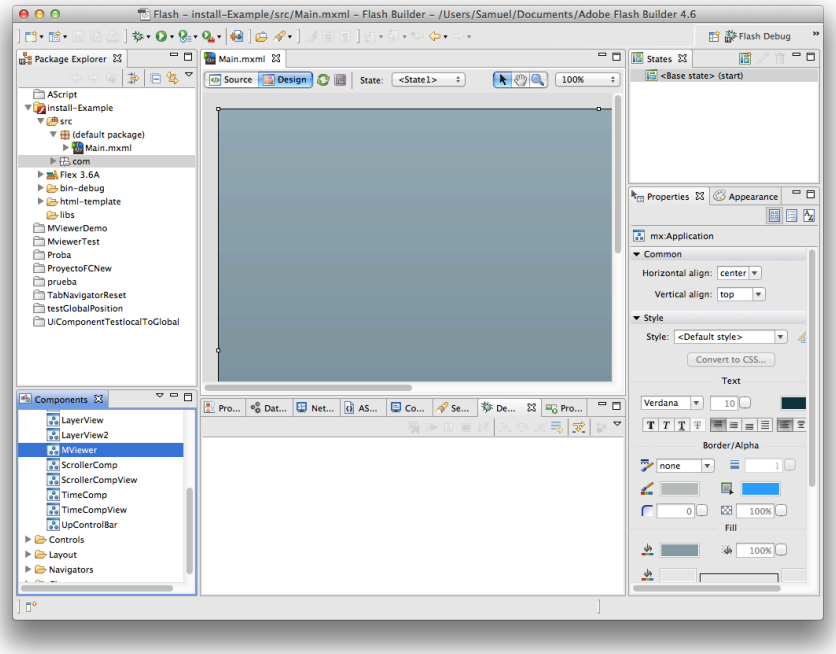

Ilustración 89: Paso 4.- Comprobación visual de la integración de MViewer

5. Ya está listo el componente. Existen dos maneras de llevar a cabo la instanciación del mismos.

a. Desde la vista de diseño, arrastrando el mismo a su posición y dándole un tamaño.

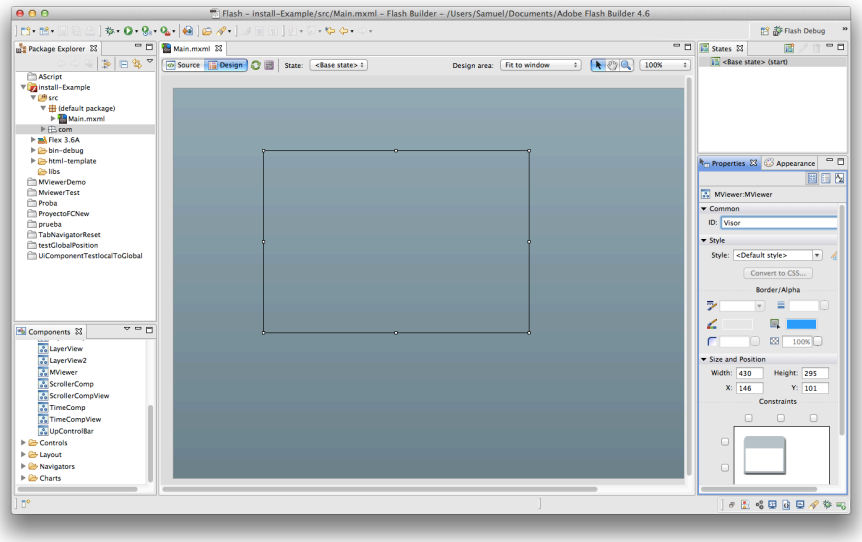

Ilustración 90: Paso 5.a.- Inserción de MViewer desde la vista de diseño

b. De forma alternativa se puede instanciarlo con código ActionScript.

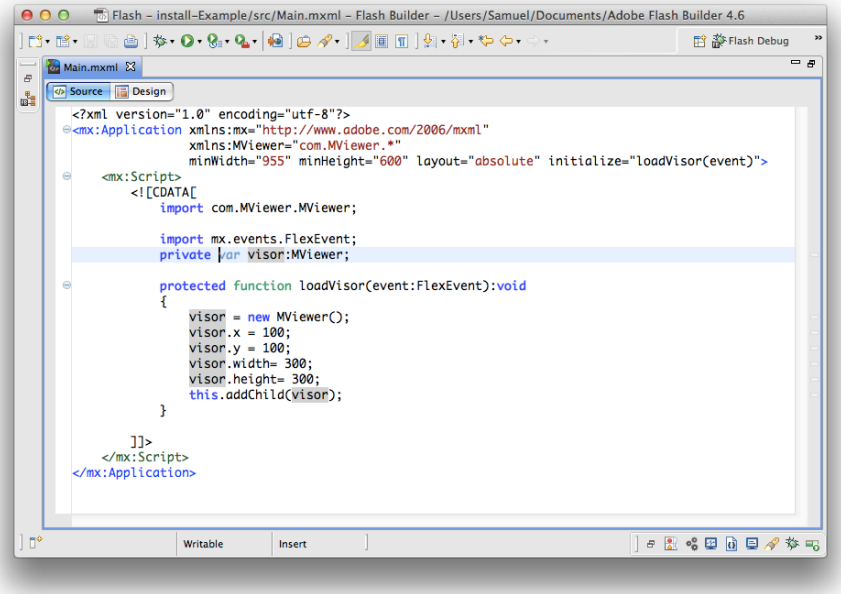

Ilustración 91: Paso 5.b.- Inserción de MViewer desde la vista de código.

6. Por último se comprueba su funcionamiento ejecutando el proyecto.

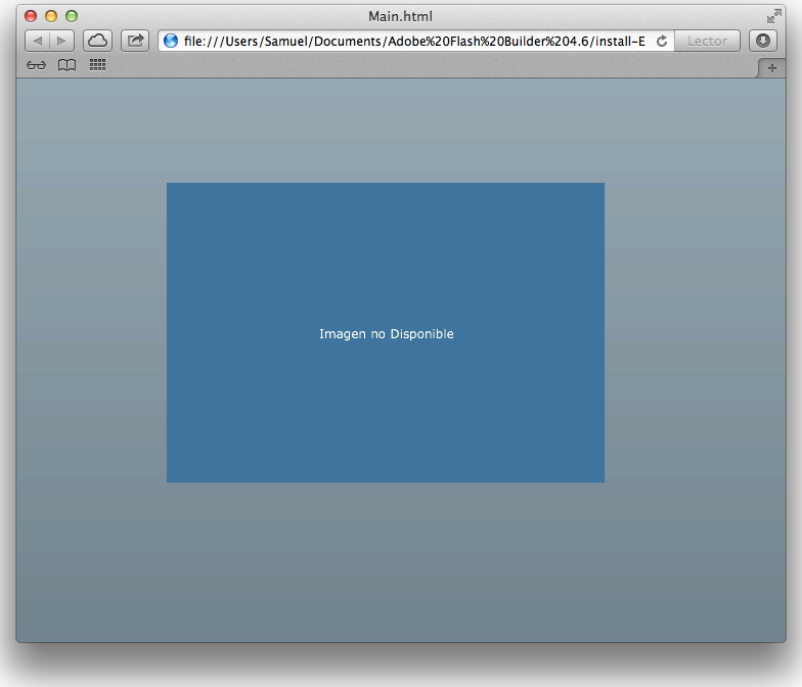

Ilustración 92: Paso 6.- Comprobación de la ejecución del proyecto

El componente está listo para poder utilizarse y poder hacer uso de sus propiedades y funciones. Ya se puede hacer uso de la API para adaptar e integrar el componente a la temática del nuevo proyecto.

Anexo VI.- Application Programming Interface (API)

La interface para utilizar el componente se describe en las siguientes páginas. En éstas se detallan el nombre y la funcionalidad de los paquetes, clases, eventos, funciones y propiedades que se suministran junto al componente. Además existe una versión digital, que se suministra en formato de página web. Proporciona un acceso directo e inmediato a esta documentación. Además contiene información avanzada añadiendo información de los parámetros de entrada y salida utilizados en cada función. Se puede acceder a la misma accediendo en la dirección "***./Documentación Web/index.html***" del disco que se adjunta con este libro.

6.1 Índice de paquetes y clases

En la siguiente tabla se listan los distintos paquetes y clases dados con el componente. Además se indica la página de este libro en la que se encuentra de forma detallada la descripción de las distintas clases y funciones.

Tabla 102: Índice de paquetes y clases de la API

6.2 Paquetes

6.2.1 Paquete com.MViewer

Incluye la clase principal del componente, que agrega toda la funcionalidad del mismo.

Clase Mviewer

La clase MViewer es la clase base de un componente para cargar y realizar simulaciones de entidades geoposicionadas en un mapa. Este componente tiene subcomponentes que permiten el control de la visibilidad de las distintas capas así como del control del instante de tiempo actual, el nivel de escala, etc.

Como elemento mxml el tag <mx:MViewer> hereda todos los atributos de su superclase UIComponent y añade los siguientes atributos:

```
<mx:MViewer
 Properties
  colorBackground=0x3477a1
  colorBorder=0xFF000000
  colorTextBackGround=0xFEFEFE
  coordinateViewAlpha=1
  coordinateViewPosition="downleft"
  coordinateViewVisible=true
  globalBrushBorderColor=0xFF000000
  globalBrushFillColor=0xFFFFFF00
  globalBrushSecondColor=0xFFFFFFFF
  globalBrushWidthBorder=1
  layerViewAlpha=1
  layerViewPosition="downright"
  layerViewVisible=true
  scrollerViewAlpha=1
  scrollerViewPosition="left"
  scrollerViewVisible=true
  textBackGround="Imagen no Disponible"
  textBackgroundSize=12
  textFontBackGround="Verdana"
  timeViewAlpha=1
  timeViewPosition="downcenter"
  timeViewVisible=true
  widthBorder=0

 Events
  imageLoadSuccess="No default"
  imageLoadFailed="No default"
  onCreatedFigure="No default"
  onDeletedFigure="No default"
  onChangedFigure="No default"
  onInsertInstantFigure="No default"
  onEditInstantFigure="No default"
```

```
  onDeletedInstantFigure="No default"
  onSelectedSetOfFigures="No default"
  onSelectedFigure="No default"
  onDeSelectedFigure="No default"
  onInsertTagFigure="No default"
  onInsertLayer="No default"
  onRemoveLayer="No default"
  onModifyLayer="No default"
  onSelectLayer="No default"
  onUpLayer="No default"
  onBringToFrontLayer="No default"
  onBringToBackLayer="No default"
  onDownLayer="No default"
  mouse_down="No default"
  mouse_up="No default"
  onActualScaleChanged="No default"
  onPercentValueChanged="No default"
  onNumZoomsScaleChanged="No default"
  onInitialTimeChanged="No default"
  onFinalTimeChanged="No default"
  onActualTimeChanged="No default"
  onIncrementalTimeChanged="No default"
  setCoordinatesSuccess="No default"
  setCoordinatesFailed="No default"
 >
```

A través de código ActionScript se puede acceder a las siguientes propiedades, funciones y eventos:

Propiedades

Propiedad	Descripción
colorBackGround	Define el color de background del fondo inicial
colorBorder	Define el color del borde del componente
colorTextBackGround	Define el color del texto insertado en el fondo inicial
coordinateViewAlpha	Se define el alfa del componente visual de la "escala".
coordinateViewPosition	Define la posición del componente visual de la escala.
coordinateViewVisible	Define la visibilidad del componente visual de la escala.
globalBrushBorderColor	Define el color del borde de la brocha global.
globalBrushBorderColorAlpha	Define el alfa del color del borde de la brocha.
globalBrushCircleDegraded	Define si existe un degradado de color en el relleno de un circulo con la configuración de la brocha global. Si está activada, se realizará un degradado radial en el relleno de los círculos.
globalBrushFillColor	Define el color principal de relleno de la brocha.
globalBrushFillColorAlpha	Define el alfa del color principal de relleno de la brocha.
globalBrushImageSize	Define el número de pixeles utilizados por la brocha global al pintar imágenes, definiendo su dimensión. Por defecto se utiliza 20x20px.
globalBrushLineDegraded	Define si existe un degradado de color en el trazo de la línea en la configuración de la brocha global. Si está activada, se realizará un degradado lineal en el trazo de una línea.
globalBrushLineSize	Define el número de pixeles utilizados en el ancho de las líneas usado por la brocha global. No se cuenta en esta cantidad el borde añadido a la línea.
globalBrushPointSize	Define el número de pixeles utilizados al pintar un punto de la brocha global. No se cuenta en esta cantidad el borde del punto.
globalBrushSecondColor	Define el color secundario de la brocha.
globalBrushSecondColorAlpha	Define el alfa del color secundario de relleno de la brocha.
globalBrushTextColor	Define el color del texto utilizado al pintar los Tags de las figuras.
globalBrushTextSize	Define el tamaño utilizado por la brocha global al dibujar texto. Por defecto se utiliza el tamaño 12.
globalBrushWidthBorder	Define el ancho del borde de la brocha global
layerViewAlpha	Se define el alfa del componente visual manejador de las capas.
layerViewPosition	Define la posición del componente visual manejador de las capas.
layerViewVisible	Define la visibilidad del componente visual manejador de las capas.
scrollerViewAlpha	Se define el alfa del componente visual manejador de la escala.
scrollerViewPosition	Define la posición del componente visual manejador de la escala.
scrollerViewVisible	Define la visibilidad del componente visual manejador de la escala.
textBackGround	Define el contenido del texto utilizado en el fondo inicial.
textBackGroundSize	Define el tamaño del texto utilizado en el fondo inicial.
textFontBackGround	Define la fuente del texto utilizado en el fondo inicial.
timeViewAlpha	Se define el alfa del componente visual manejador del tiempo.
timeViewPosition	Define la posición del componente visual manejador del tiempo.
timeViewVisible	Define la visibilidad del componente visual manejador del tiempo.
widthBorder	Define el ancho del borde del componente

Tabla 103: Propiedades suministradas por la clase MViewer

Eventos

Evento	Descripción
imageLoadFailed	Lanzado cuando falla la carga de un mapa.
imageLoadSuccess	Lanzado cuando la carga de un mapa ocurre exitosamente.
mouse_down	Lanzado cuando se recibe el evento mouse down sobre el componente.
mouse_up	Lanzado cuando se recibe el evento mouse up sobre el componente.
onActualScaleChanged	Lanzado cuando se cambia el nivel de escala actual.
onActualTimeChanged	Lanzado cuando se cambia el instante de tiempo actual del sistema.
onBringToBackLayer	Lanzado cuando una capa baja al Bottom en el listado de capas del sistema.
onBringToFrontLayer	Lanzado cuando una capa sube al Top en el listado de capas del sistema.
onChangedFigure	Lanzado cuando cambian las propiedades de una figura.
onCreatedFigure	Lanzado cuando se registra una figura.
onDeletedFigure	Lanzado cuando se borra una figura.
onDeletedInstantFigure	Lanzado cuando se borra un instante en una figura.
onDeSelectedFigure	Lanzado cuando se deseleccionan figuras.
onDownLayer	Lanzado cuando una capa baja de nivel en el listado de capas del sistema.
onEditInstantFigure	Lanzado cuando se edita un instante de una figura.
onFinalTimeChanged	Lanzado cuando se cambia el instante de tiempo final del sistema.
onIncrementalTimeChanged	Lanzado cuando se cambia el tipo de incremento del tiempo del sistema.
onInitialTimeChanged	Lanzado cuando se cambia el instante de tiempo inicial del sistema.
onInsertInstantFigure	Lanzado cuando se inserta un instante en una figura.
onInsertLayer	Lanzado cuando se inserta una capa en el sistema.
onInsertTagFigure	Lanzado cuando se inserta un tag a una figura.
onModifyLayer	Lanzado cuando se modifica una capa del sistema.
onNumZoomsScaleChanged	Lanzado cuando cambian el numero de niveles en el sistema de escala.
onPercentValueChanged	Lanzado cuando se cambia el valor de porcentaje entre niveles en el sistema de escala.
onRemoveLayer	Lanzado cuando se elimina una capa del sistema.
onSelectedFigure	Lanzado cuando se selecciona una figura.
onSelectedSetOfFigures	Lanzado cuando se seleccionan un conjunto de instantes.
onSelectLayer	Lanzado cuando se selecciona una capa del sistema.
onUpLayer	Lanzado cuando una capa sube de nivel en el listado de capas del sistema.
setCoordinatesFailed	Lanzado cuando falla la carga del espacio de coordenadas.
setCoordinatesSuccess	Lanzado cuando la carga del espacio de coordenadas ocurre exitosamente.

Tabla 104: Eventos lanzados por la clase MViewer

Métodos

Método	Descripción
MViewer	Constructor
bringToBackLayer	Baja al mínimo nivel la capa indicada por el parámetro id.
bringToFrontLayer	Sube al máximo nivel la capa indicada por el parámetro id.
changeBehaviorFigure	Cambia el tipo de comportamiento de una figura dada.
changeBrushLayer	Cambia la brocha asignada a una capa.
changeComponentMode	Define el modo de funcionamiento actual del visor.
changeInitialFinalFigureTime	Se configura el tiempo de vida de una figura, indicando su tiempo de vida inicial y final.
changeNameFigure	Se configura el nombre de una figura.
changeNameLayer	Cambia el nombre de la capa con identificador id.
changeToBackGround	Estando el visor con el mapa cargado, permite mostrar el fondo inicial.
changeToMap	Estando cargado un mapa, y habiendo ya mostrado el fondo o background inicial, permite volver a mostrar el mapa.
changeVisibilityLayer	Cambia la visibilidad de la capa con identificador id.
createLayer	Crea una capa nueva.
deleteFigureInLayer	Borra la figura de identificador id
deleteImageFigureBitmapdata	Borra un tipo de imagen usado en el modo de dibujado de imágenes.
deleteInstantOfFigure	Se elimina un instante de una figura dada.
deleteLayer	Elimina la capa con identificador idLayer.
downLayer	Baja de nivel la capa indicada por el parámetro id.
editInstantOfFigure	Edita una posición creada de una figura ya creada.
getActualLevelScale	Se obtiene el nivel actual de escala.
getActualTime	Obtiene la medida de tiempo actual del sistema de tiempo.
getGlobalInfo	Se obtiene la información global de las capas y figuras creadas en el sistema.
getIncrementalScale	Se obtiene el valor de la escala incremental. Es un porcentaje.
getIncrementalTime	Obtiene la medida de tiempo incremental asignada al sistema de tiempo.
getInfoFigure	Obtiene la información de una figura.
getLayerBrush	Se obtiene una entidad de la clase Brocha de la capa indicada por idLayer.
getLayerName	Devuelve el nombre de la capa con identificador id.
getLayersId	Devuelve un array con los identificadores de las capas registradas en el sistema.
getLayerVisibility	Visibilidad de la capa con identificador id.
getMaxTime	Obtiene la medida de tiempo máxima asignada al sistema de tiempo.
getMinTime	Obtiene la medida de tiempo mínima asignada al sistema de tiempo.
getNumLevelsScale	Número de niveles del sistema.
getSelectedLayerID	Se obtiene el id de la capa seleccionada actualmente.
insertFigureInLayer	Inserta una figura en una capa determinada o en la seleccionada.
insertImageFigureBitmapdata	Inserta un tipo de imagen para usar en el modo de dibujado de imágenes.
insertInstantInFigure	Inserta una posición nueva, un comportamiento a una figura ya insertada.
insertTagFigure	Se añade información en forma de Tag a una figura dada, útil para añadir información útil para el programador.
insertWorldCoordinates	Inserta las coordenadas del espacio de trabajo. Las

	coordenadas deben ser valores positivos >= 0.
loadMap	LoadMap carga un mapa dado por la url URL, posicionándolo en las coordenadas dadas dentro del sistema de coordenadas.
resetSystem	Resetea el sistema del componente Restaura a sus valores originales el sistema.
selectFigure	Selecciona una figura con identificador id.
selectLayer	Selecciona la capa activa que recibirá eventos de pintado, selección, etc...
setActualScale	Establece un nivel concreto para la visualización del mapa.
setActualTime	Permite establecer de forma programática el tiempo actual del visor con una medida de tipo medida tiempo.
setIncrementalScale	Establece el valor en porcentaje entre los incrementos entre niveles de la escala.
setIncrementalTime	Establece de forma programática el tiempo incremental entre el tiempo mínimo y máximo del sistema de tiempo con una medida de Tiempo.
setMaxTime	Establece el tiempo Máximo en el visor con una medida de Tiempo.
setMinTime	Establece de forma programática el tiempo Mínimo en el visor con una medida de Tiempo.
setNumLevelsScale	Establece el número de incrementos entre la escala mínima y máxima del visor.
upLayer	Sube de nivel la capa indicada por el parámetro id.
zoomIn	Permite hacer zoomIn de forma programática al visor.
zoomOut	Permite hacer zoomOut de forma programática al visor.

Tabla 105: Métodos suministrados por la clase MViewer

6.2.2 Paquete com.MViewer.Events

Paquete que contiene las clases de los eventos lanzados por MViewer.

Clase MViewerEventFigure

La clase MViewerEventFigure representa a los eventos producidos al realizar alguna acción de las figuras.

Permite lanzar los eventos:

- ON_CREATED_FIGURE
- ON_DELETED_FIGURE
- ON_CHANGED_FIGURE
- ON_INSERT_INSTANT_FIGURE
- ON_EDIT_INSTANT_FIGURE
- ON_DELETED_INSTANT_FIGURE
- ON_SELECTED_SET_OF_FIGURES
- ON_SELECTED_FIGURE
- ON_DESELECTED_FIGURE
- ON_INSERT_TAG_FIGURE

Proporciona además un conjunto de variables con información asociada a las figuras

Estas variables son las siguientes:

- idFigure
- idLayerFigure
- coordinate1
- coordinate2
- selectedFigures
- xmlInfo

Propiedades

Propiedad	Descripción
coordinate1	Indica la coordenada inicial de la figura.
coordinate2	Indica la coordenada final de la figura.
idFigure	Es la propiedad pública que contiene el identificador de la figura que ha lanzado el evento.
idLayerFigure	Es la propiedad pública que indica el identificador id de la capa donde ha sido insertada la figura.
selectedFigures	Array que contiene los identificadores de las figuras seleccionadas.
xmlInfo	Xml de la figura.

Tabla 106: Propiedades suministradas por la clase MViewerEventFigure

Constantes

Constante	Descripción
ON_CHANGED_FIGURE	Lanzado cuando cambian las propiedades de una figura.
ON_CREATED_FIGURE	Lanzado cuando se registra una figura.
ON_DELETED_FIGURE	Lanzado cuando se borra una figura.
ON_DELETED_INSTANT_FIGURE	Lanzado cuando se borra un instante en una figura.
ON_DESELECTED_FIGURE	Lanzado cuando se deseleccionan figuras.
ON_EDIT_INSTANT_FIGURE	Lanzado cuando se edita un instante de una figura.
ON_INSERT_INSTANT_FIGURE	Lanzado cuando se inserta un instante en una figura.
ON_INSERT_TAG_FIGURE	Lanzado cuando se inserta un Tag a una figura.
ON_RENDER_MAX_SIZE_PROBLEM	Lanzado cuando ocurre un problema de renderizado de las figuras debido a la limitación de flash de un tamaño máximo de pixeles.
ON_SELECTED_FIGURE	Lanzado cuando se selecciona una figura.
ON_SELECTED_SET_OF_FIGURES	Lanzado cuando se seleccionan un conjunto de instantes.

Tabla 107: Eventos dados por la clase MViewerEventFigure

Clase MViewerEventImageFactory

La clase MViewerEventImageFactory informa de la inserción y eliminación de bitmapdatas añadidas a la factoría de imágenes. Estas se utilizan para el dibujado de figuras con una representación dada.

Permite lanzar los eventos:

- ON_IMAGE_FACTORY_ELEMENT_INSERTED

Proporciona además un conjunto de variables con información asociada al evento

Estas variables son las siguientes:

- idImageFactoryElement

Propiedades

Propiedad	Descripción
idImageFactoryElement	Contiene el identificador del elemento de la factoría que produce el evento.

Tabla 108: Propiedades suministradas por la clase MViewerEventImageFactory

Constantes

Constante	Descripción
ON_IMAGE_FACTORY_ELEMENT_INSERTED	Lanzado cuando se registra un bitmapdata en la factoría de imágenes. Se asocia un identificador para hacer referencia a dicho elemento.

Tabla 109: Eventos dados por la clase MViewerEventImageFactory

Clase MViewerEventLayer

La clase MViewerEventLayer representa al evento producido por el sistema de capas.

Permite lanzar los eventos:

- ON_INSERT_LAYER
- ON_REMOVE_LAYER
- ON_MODIFY_LAYER
- ON_SELECT_LAYER
- ON_UP_LAYER
- ON_DOWN_LAYER
- ON_BRING_TO_FRONT_LAYER
- ON_BRING_TO_BACK_LAYER

Proporciona además un conjunto de variables con información asociada a las capas

Estas variables son las siguientes:

- idLayer
- nameLayer
- hasBrush
- visibility

Propiedades

Propiedad	Descripción
hasBrush	Indica si la capa tiene una brocha asociada.
idLayer	Identificador de la capa que ha lanzado el evento.
nameLayer	Nombre de la capa.
visibility	Indica la visibilidad de la capa.

Tabla 110: Propiedades suministradas por la clase MViewerEventLayer

Constantes

Constante	Descripción
ON_BRING_TO_BACK_LAYER	Lanzado cuando se lleva una capa al nivel mínimo.
ON_BRING_TO_FRONT_LAYER	Lanzado cuando se trae al frente una capa.
ON_DOWN_LAYER	Lanzado cuando se baja una capa de nivel.
ON_INSERT_LAYER	Lanzado cuando se inserta una capa en el sistema.
ON_MODIFY_LAYER	Lanzado cuando se modifica una capa del sistema.
ON_REMOVE_LAYER	Lanzado cuando se elimina una capa del sistema.
ON_SELECT_LAYER	Lanzado cuando se selecciona una capa del sistema.
ON_UP_LAYER	Lanzado cuando se sube una capa de nivel.

Tabla 111: Eventos dados por la clase MViewerEventLayer

Clase MViewerEventMap

La clase MViewerEventMap representa al evento producido por el sistema de gestión de los mapas.

Permite lanzar los eventos:

- IMAGE_LOAD_FAILED
- IMAGE_LOAD_SUCCESS

Constantes

Constante	Descripción
IMAGE_LOAD_FAILED	Lanzado cuando falla la carga de un mapa.
IMAGE_LOAD_SUCCESS	Lanzado cuando la carga de un mapa ocurre exitosamente.

Tabla 112: Eventos dados por la clase MViewerEventMap

Clase MViewerEventMouse

La clase MViewerEventMouse representa al evento producido por ratón sobre el visor, así como tiene las contantes de los mismos.

Permite lanzar los eventos:

- MOUSE_DOWN
- MOUSE_UP

Proporciona además un conjunto de variables con información asociado al evento.

Estas variables son las siguientes:

- Xpx
- Ypx
- coordinate1

Propiedades

Propiedad	Descripción
Coordinate1	Coordenada en el espacio de trabajo del punto donde se ha lanzado el evento.
Xpx	Pixel X del visor donde se ha producido el evento.
Ypx	Pixel Y del visor donde se ha producido el evento.

Tabla 113: Propiedades suministradas por la clase MViewerEventMouse

Constantes

Constante	Descripción
MOUSE_DOWN	Lanzado cuando se recibe el evento mouse down sobre el componente.
MOUSE_UP	Lanzado cuando se recibe el evento mouse up sobre el componente.

Tabla 114: Eventos dados por la clase MViewerEventMouse

Clase MViewerEventScale

La clase MViewerEventScale representa al evento producido por el sistema de la escala cuando se realiza algún cambio de configuración en el sistema o se realiza un cambio de escala.

Permite lanzar los eventos:

- ON_ACTUAL_SCALE_CHANGED
- ON_NUMZOOMS_SCALE_CHANGED
- ON_PERCENT_VALUE_CHANGED

Constantes

Constante	Descripción
ON_ACTUAL_SCALE_CHANGED	Lanzado cuando se cambia el nivel de escala actual.
ON_NUMZOOMS_SCALE_CHANGED	Lanzado cuando cambian el número de niveles en el sistema de escala.
ON_PERCENT_VALUE_CHANGED	Lanzado cuando se cambia el valor de porcentaje entre niveles en el sistema de escala.

Tabla 115: Eventos dados por la clase MViewerEventScale

Clase MViewerEventTime

La clase MViewerEventTime representa al evento producido por el sistema de la tiempo cuando se realiza algún cambio en el sistema o se realiza un cambio en los tiempos iniciales, finales o el incremento.

Permite lanzar los eventos:

- ON_INITIAL_TIME_CHANGED
- ON_FINAL_TIME_CHANGED
- ON_ACTUAL_TIME_CHANGED
- ON_INCREMENTAL_TIME_CHANGED

Constantes

Constante	Descripción
ON_ACTUAL_TIME_CHANGED	Lanzado cuando se cambia el instante de tiempo actual del sistema.
ON_FINAL_TIME_CHANGED	Lanzado cuando se cambia el instante de tiempo final del sistema.
ON_INCREMENTAL_TIME_CHANGED	Lanzado cuando se cambia el incremento del tiempo del sistema.
ON_INITIAL_TIME_CHANGED	Lanzado cuando se cambia el instante de tiempo inicial del sistema.

Tabla 116: Eventos dados por la clase MViewerEventTime

Clase MViewerEventWorldCoordinate

La clase MViewerEventWorldCoordinate representa a los eventos producidos por el sistema de coordenadas, así como tiene las contantes de los mismos.

Permite lanzar los eventos:

- SET_COORDINATES_SUCCESS
- SET_COORDINATES_FAILED

Proporciona además un conjunto de variables con información asociada a las coordenadas configuradas.

Estas variables son las siguientes:

- coordinate1
- coordinate2

Propiedades

Propiedad	Descripción
coordinate1	Indica la coordenada mínima del espacio de coordenadas.
coordinate2	Indica la coordenada máxima del espacio de coordenadas.

Tabla 117: Propiedades suministradas por la clase MViewerEventWorldCoordinate

Constantes

Constante	Descripción
SET_COORDINATES_FAILED	Lanzado cuando falla la carga del espacio de coordenadas.
SET_COORDINATES_SUCCESS	Lanzado cuando la carga del espacio de coordenadas es exitoso.

Tabla 118: Eventos dados por la clase MViewerEventWorldCoordinate

6.2.3 Paquete com.MViewer.Utils

Define clases útiles para el correcto uso del componente MViewer.

Clase MViewerBrush

La clase MViewerBrush proporciona la configuración necesaria para aplicar colores a las figuras pintadas.

Las variables creadas con este tipo pueden ser aplicadas tanto a figuras, como a una capa en concreto o a la brocha global.

Si una figura no tiene una configuración de MViewerBrush establecida se cogerá la configuración asociada a la capa. Si esta no existe, se usará la configuración establecida en la brocha global del sistema.

Permite configurar los siguientes aspectos:

- WidthBorder. El ancho del borde de las figuras.
- SizePoint. El tamaño en número de pixeles para el dibujado de puntos.
- SizeLine. El tamaño en número de pixeles para el dibujado de líneas.
- SizeImage. El tamaño en número de pixeles para especificar la dimensión de las imágenes en dim x dim.
- BorderColor. El color del borde de las figuras.
- FillColor. El color de relleno de las figuras.
- SecondColor. Un color secundario para hacer degradados en los colores de relleno, utilizando el color primario y el secundario.
- TextColor. El color utilizado para el dibujado de Tags de las figuras.
- AlphaBorderColor. Especifica el valor de alfa utilizado en el color del borde de las figuras.
- AlphaFillColor. Especifica el valor de alfa utilizado en el color de relleno de las figuras.
- AlphaSecondColor. Especifica el valor de alfa utilizado en el color secundario de las figuras.
- CircleDegraded. Activa el degradado en los círculos.
- LineDegraded. Activa el degradado en las líneas.

Propiedades

Propiedad	Descripción
alphaBorderColor	Define el alpha del color del borde de la brocha.
alphaFillColor	Define el alpha del color de relleno usado en la brocha.
alphaSecondColor	Define el alpha del color secundario para hacer degradados usado en la brocha.
borderColor	Define el color del borde de la brocha.
circleDegraded	Activa el degradado de color en el relleno de un círculo.
fillColor	Define el color principal de relleno de la brocha.
lineDegraded	Define si existe un degradado de color en el trazo de la línea.
secondColor	Define el color secundario de la brocha.
sizeImage	Define el tamaño en pixeles de las imágenes que se pintan.
sizeLine	Define el tamaño del grosor de la línea a dibujar.
sizePoint	Define el tamaño del grosor del punto a dibujar.

sizeText	Define el tamaño utilizado al insertar figuras de tipo Text.
textColor	Define el color utilizado al pintar figuras de tipo Text y los textos del resto de figuras.
widthBorder	Define el ancho del borde de la brocha.

Tabla 119: Propiedades suministradas por la clase MViewerBrush

Métodos

Método	Descripción
Constructor	Permite crear instancias de esta clase.
clone	Crea una copia de una brocha.

Tabla 120: Métodos suministrados por la clase MViewerBrush

Clase MViewerColor

La clase MViewerColor define un listado de constantes de colores para utilizar con el componente MViewer.

Define las distintas constantes para utilizar de la siguiente forma:

- NEGRO
- BLANCO
- ROJO
- AMARILLO
- VERDE
- AZUL

Propiedades

Propiedad	Descripción
color	Devuelve el color en uint

Tabla 121: Propiedades suministradas por la clase MViewerColor

Constantes

Constante	Descripción
AMARILLO	Color amarillo.
AZUL	Color azul.
BLANCO	Color blanco.
NEGRO	Color negro.
ROJO	Color rojo.
VERDE	Color verde.

Tabla 122: Constantes dadas por la clase MViewerColor

Métodos

Método	Descripción
Constructor	Crea una instancia de la clase MViewerColor.

Tabla 123: Métodos dados por la clase MViewerColor

Clase MViewerComponentPosition

La clase MViewerComponentPosition define un listado de constantes de posiciones para utilizar con los subcomponentes como el visor de capas, el visor de tiempo, o el visor de posición.

Define las distintas constantes de la siguiente forma:

- UPLEFT
- UPCENTER
- UPRIGHT
- DOWNLEFT
- DOWNCENTER
- DOWNRIGHT
- LEFT
- RIGHT
- OUTUPLEFT
- OUTUPRIGHT
- OUTDOWNLEFT
- OUTDOWNRIGHT

Constantes

Constante	Descripción
DOWNCENTER	Posición abajo-centro dentro del componente.
DOWNLEFT	Posición abajo-izquierda dentro del componente.
DOWNRIGHT	Posición abajo-derecha dentro del componente.
LEFT	Posición izquierda dentro del componente.
OUTDOWNLEFT	Posición abajo-izquierda fuera del componente.
OUTDOWNRIGHT	Posición abajo-derecha fuera del componente.
OUTUPLEFT	Posición arriba-izquierda fuera del componente.
OUTUPRIGHT	Posición arriba-derecha fuera del componente.
RIGHT	Posición derecha fuera del componente.
UPCENTER	Posición arriba-centro dentro del componente.
UPLEFT	Posición arriba-izquierda dentro del componente.
UPRIGHT	Posición arriba-derecha dentro del componente.

Tabla 124: Constantes dadas por la clase MViewerComponentPosition

Métodos

Método	Descripción
Constructor	Crea una instancia de esta clase.
changePosition	Cambia el valor establecido en una variable de tipo MViewerComponentPosition.

Tabla 125: Métodos dados por la clase MViewerComponentPosition

Clase MViewerFigures

La clase MViewerFigures define un listado de constantes de tipos de Figuras que se pueden utilizar con el componente MViewer.

Define las distintas constantes de la siguiente forma:

- POINT
- CIRCLE
- LINE
- IMAGE
- TEXT

Constantes

Constante	Descripción
CIRCLE	Corresponde al circulo.
IMAGE	Corresponde al tipo imagen.
LINE	Corresponde a la línea.
POINT	Corresponde al punto.
TEXT	Corresponde al texto

Tabla 126: Constantes dadas por la clase MViewerFigures

Clase MViewerTimeMeasure

La clase MViewerTimeMeasure define un listado de constantes de tiempos para utilizar con el componente MViewer.

Define las distintas constantes para utilizar de la siguiente forma:

- DIAS
- HORAS
- MINUTOS
- SEGUNDOS
- DECIMAS
- CENTESIMAS
- MILESIMAS

Constantes

Constante	Descripción
CENTESIMAS	Define la constante asociado al número de segundos que tiene una centésima
DECIMAS	Define la constante asociado al número de segundos que tiene una décima.
DIAS	Define la constante asociado al número de segundos que tiene un día.
HORAS	Define la constante asociado al número de segundos que tiene una hora.
MILESIMAS	Define la constante asociado al número de segundos que tiene una milésima.
MINUTOS	Define la constante asociado al número de segundos que tiene un minuto.
SEGUNDOS	Define la constante utilizada como base de la medida.

Tabla 127: Constantes dadas por la clase MViewerTimeMeasure

Métodos

Método	Descripción
Constructor	Constructor de medidas de tiempo.
clone	Crea una copia de una medida.
toText	Transforma una medida en una ristra de texto.

Tabla 128: Métodos dados por la clase MViewerTimeMeasure

Clase MViewerMode

Lista los modos de funcionamiento del componente MViewer.

Define las distintas constantes de la siguiente forma:

- MOVE
- SELECT
- PAINTPOINT
- PAINTLINE
- PAINTCIRCLE
- PAINTIMAGE
- GETCOORDINATES

Constantes

Constante	Descripción
GETCOORDINATES	En este modo el diseñador puede obtener coordenadas cuando el usuario hace clic en el mapa, sin realizar ningún tipo de dibujado ni movimiento.
MOVE	En este modo el componente permite mover el mapa y escalar.
PAINTCIRCLE	En este modo el usuario puede trazar círculos para marcar áreas.
PAINTIMAGE	Permite que el usuario pinte la imagen seleccionada por el sistema de MViewerImageFactory.
PAINTLINE	En este modo el usuario puede trazar líneas para marcar zonas.
PAINTPOINT	En este modo el usuario puede registrar una entidad en un punto concreto.
SELECT	Permite seleccionar las distintas entidades registradas en el componente en la capa seleccionada.

Tabla 129: Constantes dadas por la clase MViewerMode

6.3 Documentación web

Con el componente se da una versión en formato de página web de la API. Como se ha visto anteriormente se puede acceder a la misma en el contenido del cd que se adjunta, en la siguiente ruta "***./Documentación Web/index.html***". Esta documentación proporciona una información detallada que contiene de forma estructurada los paquetes, clases, métodos y parámetros de entrada y salida necesarios para el uso del componente.

La página desarrollada tiene 2 apartados, una barra lateral izquierda donde navegar por los distintos paquetes y clases, y la zona central donde se expone de forma ordenada la distinta información.

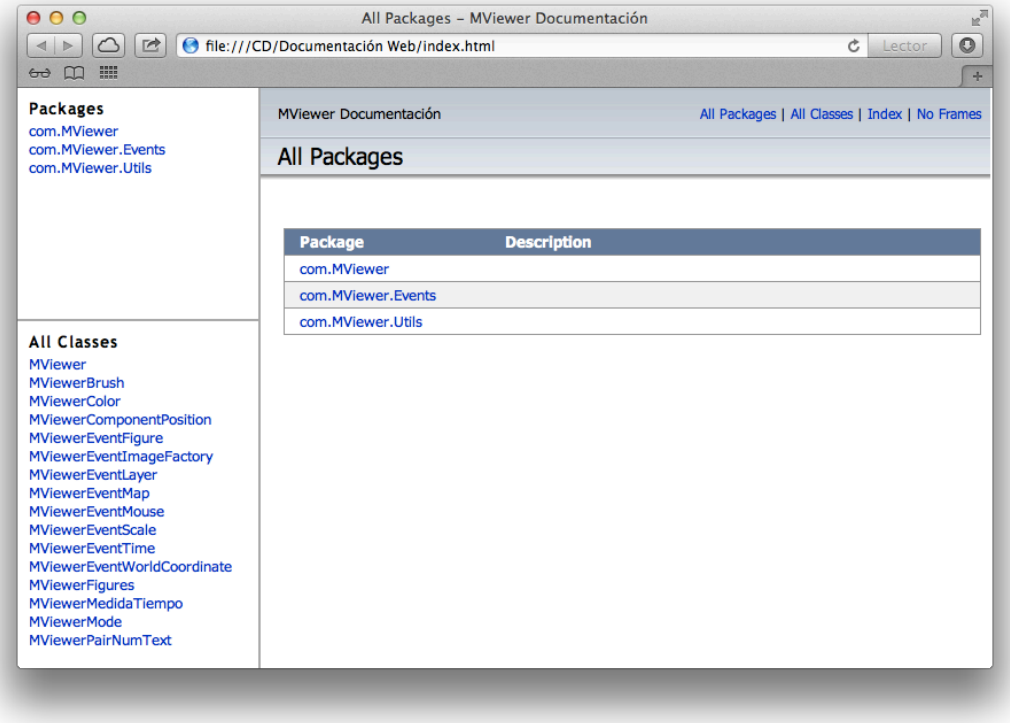

Ilustración 93: Página principal de la documentación web

Anexo VII.- Contenido del CD

Esta memoria viene acompañada con un cd que contiene el producto desarrollado. La siguiente imagen ilustra la estructura del mismo.

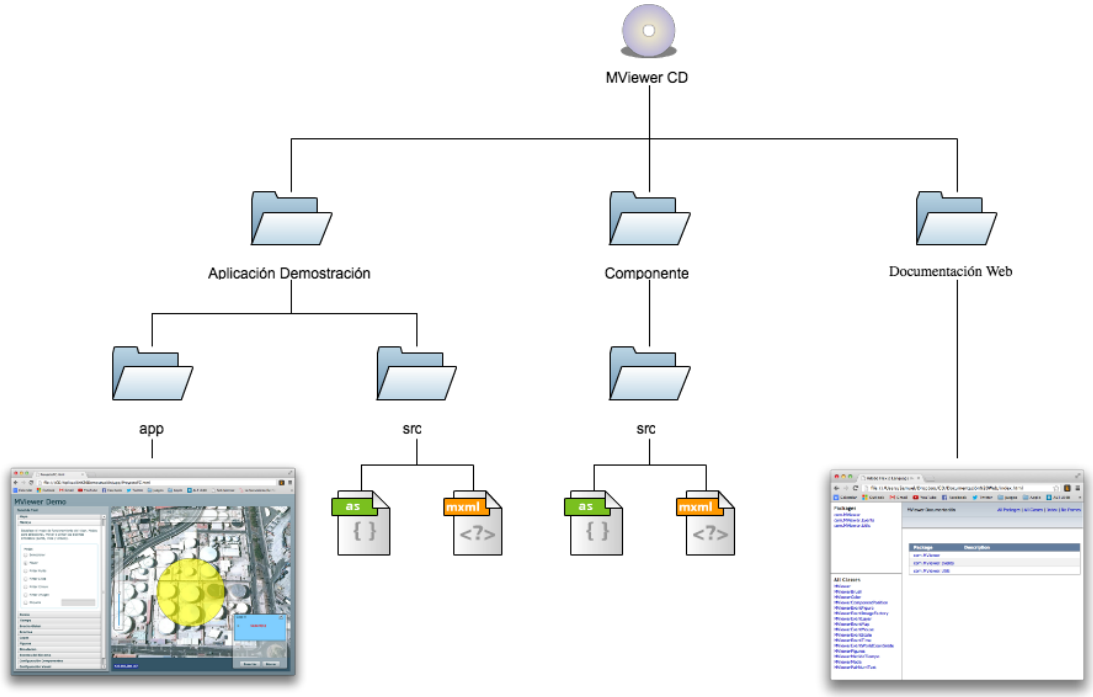

Ilustración 94: Estructura del contenido del CD

En el CD se encuentran los 3 elementos desarrollados, la aplicación de demostración, el componente visor de planos y la documentación web de la API.

La aplicación de demostración contiene el producto final y el código del mismo. El primero se encuentra en la carpeta "**./Aplicación Demostración/app/ ProyectoFC.html**". Su código se localiza en "**./Aplicación Demostración/app/ src/**".

Del componente desarrollado se entrega su código en la ubicación "**./Componente/src/**".

Por último la documentación web de la API se localiza en "**./Documentación Web/index.html**".