
# Aplicación iOS para el seguimiento de cursos no presenciales

---

Proyecto Fin de Carrera
Junio de 2017

Rubén Domínguez Falcón

**Tutores:**
Francisco José Santana Pérez
Javier Toledo Mediavilla

Muchas han sido las personas que me han ayudado a cumplir el objetivo de este proyecto.

Pero quiero hacer especial mención a todo el equipo de TheAgileMonkeys por ser una segunda familia.

A Fran por convertirse en mi nexo entre el mundo profesional y el académico.

Y por último a Eliezer, Yeray y Luis por siempre tener una palabra de aliento en los momentos difíciles.

# Índice general

# Capítulo 1

# Introducción

A lo largo del presente documento se describe el proceso seguido para el desarrollo de una aplicación, para dispositivos iOS, que sirva como plataforma para el apoyo a la enseñanza.

Dos han sido los principales motivos que han impulsado el desarrollo de este proyecto.

- Por un lado, la necesidad del propio autor de adentrarse en el desarrollo de aplicaciones móviles y en especial en el universo iOS.
- Por otro, gracias a la visión del equipo de TheAgileMonkeys, la idea de usar los dispositivos móviles como herramienta para la autoformación.

Haremos hincapié en cada una de las etapas del desarrollo del producto. Desde la captura de requisitos de usuario hasta el diseño arquitectónico y su posterior implementación. Al finalizar este documento el lector tendrá una visión general del proceso de desarrollo, así como de los resultados obtenidos tras la finalización del mismo.

# Capítulo 2

# Objetivos

## 2.1. Aplicación iOS de aprendizaje

El objetivo principal de este proyecto es la creación de un aplicación que aproveche la versatilidad de los dispositivos móviles para propósitos educativos. Ésta servirá de soporte para proveer a sus usuarios de contenido audiovisual que les permita recibir un curso a realizar de forma autodidacta y evaluar su nivel de aprovechamiento del mismo.

La aplicación estará diseñada de forma que sus usuarios puedan recibir diariamente un vídeo con el contenido de una lección del curso. De esta manera, se permite que el usuario tenga la flexibilidad de ver el vídeo en el momento que le resulte más cómodo, pero a su vez se fomenta el seguimiento del curso día a día. Finalmente, una vez completado el curso se activa un cuestionario de autoevaluación con el que el estudiante puede valorar los conocimientos y competencias adquiridos a lo largo del mismo.

Al finalizar el proyecto se habrá construido una aplicación para dispositivos iOS con los casos de uso principales que se enumeran a continuación :

- Registro de usuarios al inicio de curso.
- Visualización vídeos con el contenido de cada lección.
- Envío de una notificación push cuando exista un nuevo vídeo disponible.
- Mostrar el historial de las clases que se han recibido hasta este momento permitiendo al usuario volver a visionar cualquiera de las lecciones ya completadas.
- Mostrar un cuestionario autoevaluable tras acabar el curso.

## 2.2. Servicio de gestión de cursos

Además del desarrollo de la aplicación móvil, se implementará una parte de servidor que permita a un administrador gestionar los contenidos que van a ser mostrados en la aplicación, así como recoger estadísticas del uso de ésta.

Así, los casos de uso que se pretenden alcanzar para el lado del servidor son:

- Gestión de información relativa a los vídeos: Título, texto de la notificación, orden de reproducción.
- Almacenamiento de la información relativa a los usuarios.
- Estadísticas relativas al uso de la aplicación.

# Capítulo 3

# Metodología y Planificación

## 3.1. Metodología

Para la realización de este proyecto se ha empleado un modelo inspirado el proceso de desarrollo iterativo incremental[Mills, 1980].

En este tipo de metodología, el proyecto se planifica en diversos bloques temporales llamados iteraciones. En cada iteración el proyecto evoluciona a partir de los resultados completados en las iteraciones anteriores, añadiendo nuevos objetivos/requisitos o mejorando los que ya fueron completados.

Figura 3.1: Proceso de desarrollo incremental

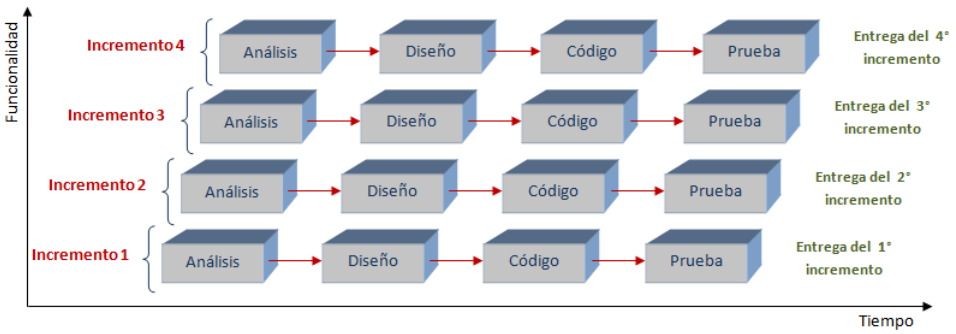



**Figura 1:** El Modelo Incremental

## 3.2. Planificación

Las etapas fundamentales del desarrollo del proyecto y su duración estimada se listan a continuación:

| Actividad | Estimación |
|---|---|
| Estudio Previo:<br>▪ Aprendizaje de Objective-C e introducción al desarrollo de aplicaciones para iOS.<br>▪ Core Data.<br>▪ Controladores básicos en Cocoa. | 90H |
| Diseño de la aplicación:<br>▪ Diseño de Interfaz de Usuario.<br>▪ Diseño de navegación entre pantallas.<br>▪ Diseño arquitectónico. | 110H |
| Implementación de casos de uso y pruebas. | 410H |
| Análisis e integración de herramientas de Backend. | 205H |
| Documentación. | 100H |
| **TOTAL** | **915H** |

# Capítulo 4

# Recursos necesarios

La elección de iOS como plataforma objetivo para el desarrollo de este proyecto ha condicionado, tanto la elección del entorno de desarrollo a utilizar, como la clase de equipo necesario.

## 4.1. Recursos Hardware

Para poder utilizar todas las herramientas de desarrollo que Apple pone a nuestro alcance es necesario un equipo capaz de soportar Mac OS. Concretamente, para este proyecto se ha utilizado un Macbook Pro de 13" de finales de 2012.

Figura 4.1: Macbook Pro

Especificaciones:

| |
|---|
| Macbook Pro 13” (finales 2012) |
| Procesador Intel Core i5 a 2,5 GHz |
| Memoria RAM 8 GB 1600 MHz DDR3 |
| Gráficos Intel HD Graphics 4000 1536 MB |
| Pantalla de 13,3 pulgadas |
| Almacenamiento SSD 128 GB |

Además de hacer uso de los simuladores provistos por Apple, se ha empleado un iPhone 5S con objeto de probar la aplicación directamente sobre un dispositivo real.

Figura 4.2: iPhone5S

Especificaciones:

| iPhone 5S |
|---|
| Capacidad 16 GB |
| Chip A7 64 bits |
| Conectividad Wifi, Bluetooth 4.0, 4G |
| ID táctil huella digitales |
| Pantalla Retina Multi-Touch de 4 pulgadas |
| Camara iSight 8 MP |
| Grabación de video HD 1080p |

## 4.2. Recursos Software

### Entorno de desarrollo: Xcode


Figura 4.3: Icono Xcode

Actualmente, la única opción disponible para la creación de aplicaciones iOS es el entorno de desarrollo que Apple proporciona gratuitamente: Xcode[iTunes S.a.r.l.]. Este entorno ha sido desarrollado por Apple Inc. y sólo se encuentra disponible para sistemas operativos OS X.

### Kit de desarrollo: iOS SDK

Mediante este kit, Apple proporciona las utilidades necesarias para llevar a cabo el desarrollo de aplicaciones para dispositivos iOS. Se integra en el entorno de desarrollo Xcode, e incluye las APIs propias de los sistemas operativos iOS así como simuladores de los diferentes dispositivos de Apple.

### Servicio Backend: Parse

Figura 4.4: Logo Parse

Como ya se ha comentado, la aplicación a desarrollar requiere el almacenamiento de datos relativos a los usuarios, así como poder gestionar información sobre las lecciones y obtener estadísticas relativas al uso de la aplicación. Como un objetivo del proyecto se estableció encontrar una solución que pudiese llevar a cabo estas tareas, siendo Parse[Parse]la alternativa utilizada para tal fin.

Parse es un servicio, actualmente propiedad de Facebook, que proporciona un sistema de almacenamiento en la nube siguiendo una arquitectura basada en entidad-relación. Ofrece además un SDK escrito en Objective-C para la comunicación entre los dispositivos móviles y el backend así como herramientas para la recogida de analíticas.

### Edición de documentación: LyX


Figura 4.5: Logo LyX

La redacción de la documentación se lleva a cabo mediante el uso de LyX[Team], un programa gráfico multiplataforma que permite edición de texto usando LATEX, de manera que hereda todas sus capacidades y a su vez, resulta más sencillo de manejar.

Las ventajas más destacables del uso de LyX en la redacción de este documento son:

- La capacidad de modularización del documento a través del uso de diferentes ficheros, uno por cada capítulo
- La gestión de la bibliografía, dado que Lyx permite su automatización
- La gestión de las figuras y su índice, ya que LyX provee de mecanismos para automatizar la generación del índice de figuras mientras se redacta el texto.

### Wireframes: Balsamiq Mockups 3

Figura 4.6: Logo Balsamiq Mockups 3

Antes de implementar cualquier tipo de proyecto, conviene hacerse una idea de cómo quedará y qué se planea realizar a través de bocetos. Por este motivo, para crear una aplicación móvil viene muy bien plasmar las pantallas que se mostrarán al usuario mediante wireframes (representación esquemática de una web o aplicación), y Balsamiq Mockups 3[Studios] es una buena herramienta que facilita y agiliza la creación de estos bocetos. Con él, la mayoría de elementos de las pantallas se pueden añadir fácilmente arrastrando el elemento a la pantalla deseada. Además, no sólo cuenta con aplicación de escritorio, también dispone de versión web y plugins para diferentes servicios (Google Drive, Jira, ...).

### Modelado UML: StarUML

Figura 4.7: Logo StarUML

Esta aplicación permite generar diferentes diagramas de tipo UML (lenguaje unificado de modelado, cuyo nombre proviene de sus siglas en inglés: Unified Modeling Language), el lenguaje de modelado de mayor aceptación en la actualidad. A través de estos diagramas se puede describir el sistema y su comportamiento. StarUML[MKLab] proporciona las herramientas gráficas para realizar diagramas de casos de uso, de actividad, de clase, entre muchos otros.

### Testing: TestFlight

Figura 4.8: Icono TestFlight

TestFlight[Inc.] es un servicio de distribución en la nube de Apps en beta, el cual ha sido adquirido por Apple Inc. y cuyo uso es muy común en entornos empresariales, ya que permite instalar y testear aplicaciones móviles para un número concreto de usuarios. Cabe destacar que únicamente pueden acceder a esta herramienta aquellos desarrolladores que se encuentran en el iOS Developer Program. A través del servicio, el desarrollador puede distribuir aplicaciones a distintos tipos de testers y recibir logs, reportes de errores y feedback.

## Lenguaje de programación: Objective-C

Tratándose de un desarrollo para dispositivos iOS, es necesario hacer uso de alguno de los lenguajes que Apple provee para este fin. Para este proyecto, el lenguaje utilizado es Objective-C[Brad Cox] puesto que, aunque a día de hoy se puede desarrollar aplicaciones para dispositivos iOS utilizando el lenguaje de programación Swift, en el momento que se comenzó a desarrollar este proyecto Swift se encontraba en fase beta y no contaba con la estabilidad de la que goza en la actualidad.

Objective-C es un lenguaje de programación orientado a objetos creado como un superconjunto de C, y por ello es posible incluir código en C dentro de un programa escrito en Objective-C.

## Control de versiones

### Git

Figura 4.9: Logo Git

Git [Torvalds] es un aclamado sistema de control de versiones, el cual está muy extendido y en consecuencia, existen múltiples repositorios y herramientas para su uso. Cuenta con la ventaja de que Xcode soporta la integración de la gestión de repositorios usando este sistema de control de versiones.

### Bitbucket

Figura 4.10: Logo Bitbucket

Existen múltiples servicios de repositorios para alojar proyectos mediante Git, siendo GitHub y Bitbucket[Atlassian] dos de los más conocidos en el mundo

software. La principal ventaja de Bitbucket frente a Github es que en Bitbucket se pueden crear repositorios privado de forma gratuita y ha sido esta la principal razón para decantarnos por esta solución.

#### Tower 2


Figura 4.11: Logo Tower 2

A pesar de que Xcode permite la integración de control de versiones Git, pudiendo hacer uso de él rápidamente, se ha usado para este proyecto Tower 2 como herramienta para el manejo del sistema de control de versiones. La razón principal es que el alumno está altamente familiarizado con él debido a su uso en otros proyectos.

### Librerías utilizadas

Durante el desarrollo de aplicaciones suele ser necesario añadir librerías y frameworks externos. En este proyecto se han utilizado las siguientes:

#### ActionSheetPicker3.0

ActionSheetPicker3.0 [Cinel] permite presentar fácilmente un ActionSheet con un PickerView, dando oportunidad al usuario de seleccionar una opción dentro de un número inmutable de opciones.

Puede entenderse que un componente ActionSheetPicker es una unión de los siguientes elementos:

ActionSheetPicker = UIPickerView + UIActionSheet

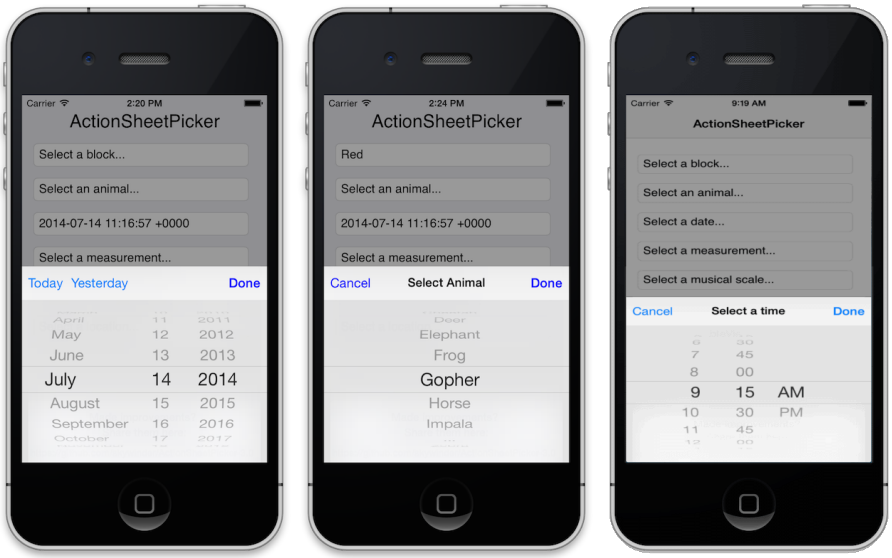



Figura 4.12: Ejemplos ActionSheetPicker3.0

Por otro lado, posee la ventaja de adaptarse según el dispositivo de manera que un iPad muestra una forma distinta a como lo hace en iPhone. De esta manera se puede concretar según el dispositivo los elementos que componen el ActionSheetPicker:

- iPhone/iPod ActionSheetPicker = ActionSheetPicker = Picker + UIActionSheet
- iPad ActionSheetPicker = Picker + UIPopoverController

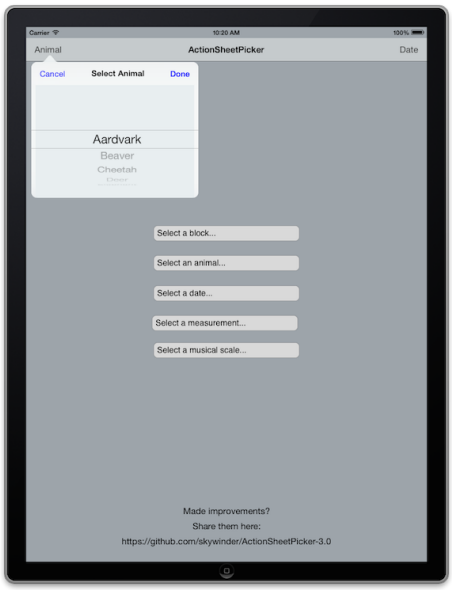



Figura 4.13: Ejemplo ActionSheetPicker3.0 en iPad

### Crittercism


Figura 4.14: Logo Crittercism

Este framework gratuito resulta útil para comprobar el rendimiento, así como para detectar y corregir errores, ya que manda información del uso de la aplicación por parte del usuario con diagnósticos e informe de errores. De esta forma al desarrollador le llega la información necesaria acerca del del comportamiento de la aplicación y de cómo poder solucionar aquellos problemas detectados durante el uso de la misma. Cabe destacar que ha cambiado de nombre, y actualmente se llama Apteligent [Apteligent].

# Capítulo 5

# Estado del Arte

## 5.1. e-learning

Podemos definir al E-Learning como un sistema de formación interactivo, que hace uso de los medios electrónicos para llegar a un alumnado generalmente remoto.

Este sistema ha transformado la educación, abriendo puertas al aprendizaje individual y organizacional. Es por ello que hoy en día está ocupando un lugar cada vez más destacado y reconocido dentro de las organizaciones empresariales y educativas.

Algunos beneficios de la utilización del e-learning son:

- **Aprendizaje Activo**: se trata de una formación que requiere de mucho esfuerzo por parte del alumno, donde el mismo aprende a responsabilizarse de su formación por iniciativa propia.
- **Flexibilidad**: esta es una de las principales ventajas del e-learning: la flexiblidad de horarios. El alumno es quien decide cuándo estudiar, cuándo descargar el programa...
- **Accesso en cualquier momento**: El alumno sólo necesita tener conexión a internet para poder acceder al contenido del curso, realizar tareas o hacer los exámenes.
- **Rapidez y agilidad:** A través de internet el alumno puede obtener recursos rápidamente, sin necesidad de traslados, colas ni esperas.

- **Movilidad y localización**: El alumno puede trasladarse o viajar, y aun así poder realizar el curso.
- **Manejo de tiempo**: El alumno puede realizar el curso a un ritmo que se adapte a sus necesidades, y modificarlo según la situación.
- **Material de aprendizaje y apoyo:** Permite otorgar una gran cantidad de recursos de aprendizaje de una manera fácil y sencilla.

## 5.2. m-learning

El m-learning consiste en el uso de dispositivos móviles como plataforma soporte para el e-learning. Esta tendencia ha ido en aumento debido crecimiento de las capacidades de los móviles en la actualidad, así como la creciente demanda de esta tecnología, la cual cuenta con la gran ventaja de permitir el acceso a la información en cualquier momento y lugar, pues el móvil ha pasado a ser prácticamente una necesidad hoy en día , y ofrece la comodidad y facilidad de tener al alcance de la mano cualquier tipo de contenido.

## 5.3. Diferentes aplicaciones m-learning

### Udemy


Figura 5.1: Logo Udemy

Udemy una plataforma de e-learning que cuenta con muchos cursos de diversa temática. Cuenta con aplicación para dispositivos móviles tanto en Android como en iOS. Esta se encuentra adaptada para poder seguir los cursos a través del smartphone. La aplicación esta disponible en múltiples idiomas y cuenta con cursos en varios idiomas. Se trata de un servicio gratuito con cursos de pago a diferentes precios, gratis algunos de ellos, donde el contenido es ofrecido por otro tipo de usuario, que actúan de profesores. Al comprar un curso se obtiene acceso a todo el curso de manera permanente y el usuario puede seguirlo a su ritmo. Permite descargar las lecciones de los cursos de manera que se pueda seguir el curso de manera offline.

### Coursera

Figura 5.2: Logo Coursera

Se trata de otra plataforma de e-learning que cuenta con presencia en dispositivos móviles. En este caso los cursos están realizados por diferentes instituciones académicas y los cursos suelen ser llevados a cabo mediante unos plazos en los que los profesores pueden realizar el seguimiento de los alumnos y solicitarle la realización de diferentes actividades para completar el curso. Cuenta con cursos en varios idiomas, principalmente inglés, además incluye subtítulos en idioma nativo y en otros cuando esta disponible. Permite pagar los cursos por capítulos, en lugar de pagar el curso entero, y de forma adicional comprar un certificado de acreditación con el curso aprobado. Posee versiones para iOS y Android.

### Memrise

Figura 5.3: Logo Memrise

Existen multitud de aplicaciones móviles centradas en el aprendizaje de idiomas. Memrise lo hace a través de un sistema de aprendizaje mediante tarjetas donde el usuario asocia un conocimiento a una imagen, de forma que aprenda progresivamente mediante unos objetivos de aprender cada día una serie de palabras y repasarla los días siguientes de manera que arraiguen en la memoria. Aunque esta centrada en el aprendizaje de idiomas, también dispone de cursos en otras categorías de aprendizaje. Intercala el aprendizaje de nuevas palabras con ejercicios que comprueban si el alumno ha retenido el conocimiento. ademas de repasos diarios. Aplicaciones y cursos resultan gratuitos, su monetización esta basada en un servicio PRO de pago. Esta disponible tanto para iOS como para Android.

# Capítulo 6

# Análisis

En este capítulo se realizará un estudio exhaustivo del tema a tratar, ahondando en cada uno de los conceptos que deben dominarse e identificando y detallando cada uno de los requisitos de la aplicación.

## 6.1. Dominio técnico

A continuación, se explicarán todos aquellos conceptos relativos al dominio técnico del problema, como por ejemplo el sistema operativo objetivo y los diversos dispositivos en los que se podrá ejecutar la aplicación.

### iOS

Es un sistema operativo móvil de Apple Inc. desarrollado originalmente para el iPhone. Posteriormente pasó también a usarse en otros dispositivos de hardware similar: como el iPod Touch y iPad. Es un derivado de Mac OS X, que a su vez está basado en Darwin BSD. El iOS tiene cuatro capas de abstracción:

- Cocoa Touch
- Media
- Core Services
- Core OS

En el proyecto se trabaja con la versión iOS 7.

### iPhone

Teléfono inteligente multimedia creado por la compañía Apple Inc. Ejecuta el sistema operativo iOS, explicado en el concepto anterior. Existen diferentes generaciones y modelos de iPhone. Debido a la elección del sistema operativo, la aplicación servirá en los iPhone desde el modelo iPhone 4 en adelante.

### iPad

Dispositivo electrónico de tipo tableta desarrollado por Apple Inc. Se sitúa en una categoría entre teléfono inteligente y ordenador portátil, aunque más enfocado al acceso que a la creación de contenido. Existen diferentes generaciones y modelos de iPad. Una gran variedad de estos son capaces de ejecutar iOS 7 o superior.

### iPod Touch

Reproductor multimedia portátil fabricado por Apple Inc. Ejecuta el sistema operativo iOS, al igual que el iPhone y el iPad. Podría considerarse una especia de iPhone sin las funcionalidades de telefonía. A partir de la quinta generación de iPod Touch se encuentra disponible iOS 7 y por tanto la aplicación de este proyecto.

## 6.2. Especificación de requisitos

La especificación de requisitos es un proceso fundamental dentro del desarrollo software. Consiste en una descripción completa del comportamiento del sistema a desarrollar que sirve para establecer con detalle las funciones, servicios y restricciones operativas del mismo. Debe definir exactamente qué es lo que se va a implementar. Esto se consigue mediante:

- Requisitos funcionales. Contienen un conjunto de casos de uso que describen todas las interacciones previstas de cara a los usuarios del software.
- Requisitos no funcionales. Son los requisitos que imponen restricciones al funcionamiento del sistema o a su diseño, como pueden ser los requisitos de funcionamiento, de calidad, de diseño, etc.

## Identificación de Actores

Resulta imprescindible tener bien identificadas los actores que intervendrán en el sistema a analizar de manera que se definan correctamente los casos de uso y requisitos.

| Actor | Tipo | Definición |
|---|---|---|
| Usuario | Principal | Cualquier persona que haga uso de la aplicación es considerada usuario. |

## Diagramas de casos de uso

A continuación se muestran los diagramas de casos de uso realizados y de donde se obtienen los requisitos funcionales.

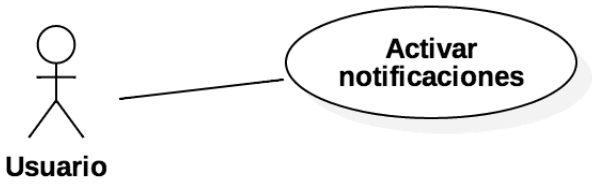



Figura 6.1: Diagrama de Casos de Uso 1

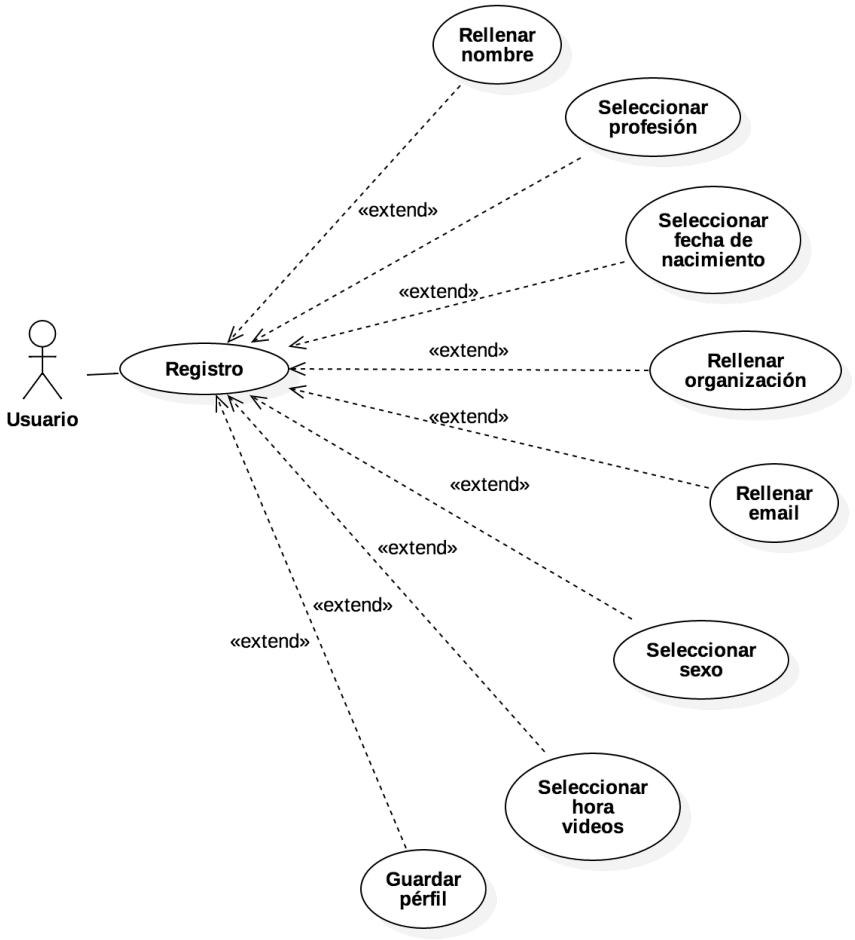



Figura 6.2: Diagrama de Casos de Uso 2


Figura 6.3: Diagrama de Casos de Uso 3

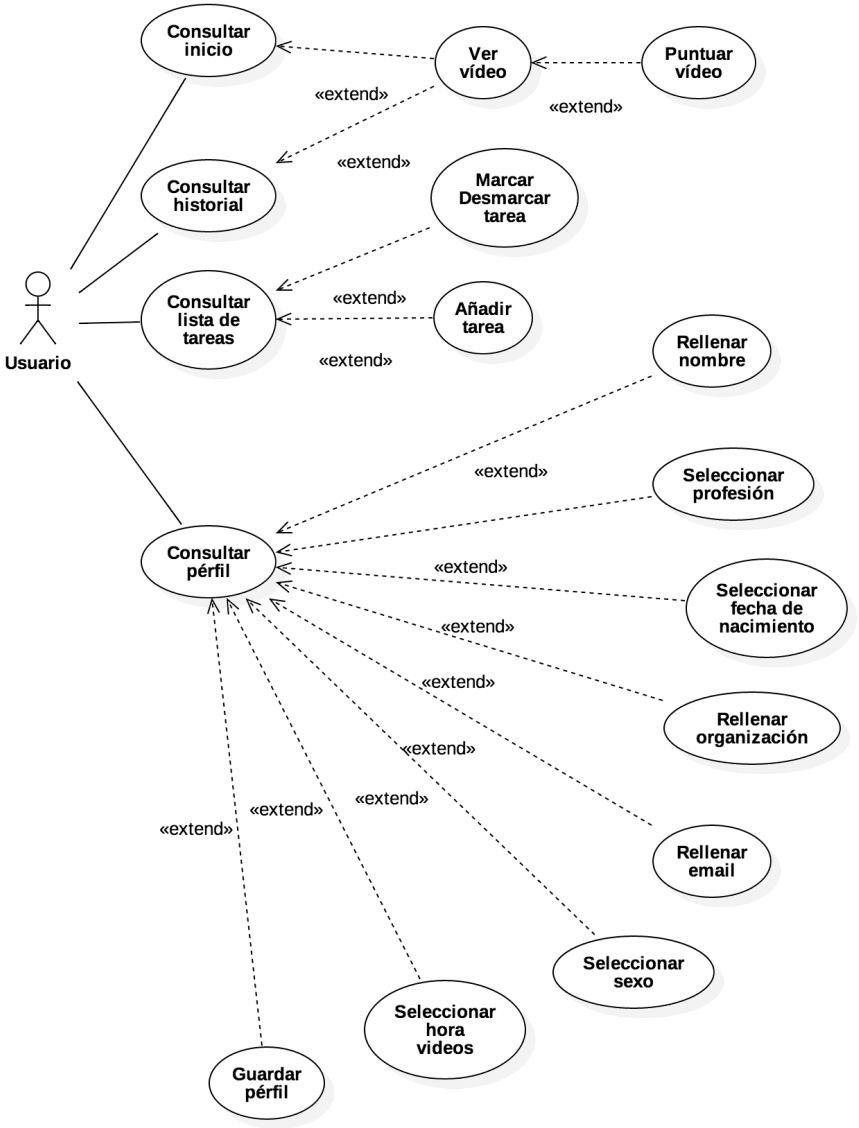



Figura 6.4: Diagrama de Casos de Uso 4

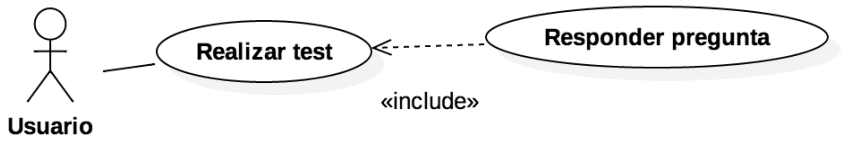



Figura 6.5: Diagrama de Casos de Uso 5

## Requisitos funcionales

Teniendo los diagramas de caso de uso se puede pasar a realizar una especificación detallada de los requisitos funcionales en los casos de uso.

### UC 01 - Activar notificaciones

| | | |
|---|---|---|
| Nombre: | UC 01 - Activar notificaciones | |
| Actor principal: | Usuario | |
| Objetivo: | Decidir si el usuario recibe notificaciones de la aplicación cuando hay un nuevo vídeo disponible | |
| Precondición: | Encontrarse al inicio de la aplicación, no haber completado esta caso de uso previamente. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Selecciona la opción deseada. | 2: Configura los ajustes de notificación de la aplicación en el sistema con la opción seleccionada por el usuario. |
| Postcondición | Ajustes de notificaciones configurado. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 02 - Registro**

| | | |
|---|---|---|
| Nombre: | UC 02 - Registro | |
| Actor principal: | Usuario | |
| Objetivo: | Registrar los datos y preferencias del usuario. | |
| Precondición: | No haber completado el registro previamente. Encontrarse en la pantalla inicial. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Pulsa el botón siguiente. | 2: Visualiza la pantalla de registro. |
| Postcondición | Pantalla de registro visualizada. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 03 - Rellenar nombre**

| | | |
|---|---|---|
| Nombre: | UC 03 - Rellenar nombre | |
| Actor principal: | Usuario | |
| Objetivo: | Añadir o editar el nombre del perfil del usuario. | |
| Precondición: | Encontrarse en una pantalla que visualice el campo de nombre. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Selecciona el campo de nombre. | 2: Muestra el teclado para que el usuario rellene el campo. |
| | 3: Escribe el nombre del usuario y lo acepta. | 4: Cierra el teclado y establece el nombre en la ficha de perfil. |
| Postcondición | Nombre establecido en el campo de nombre. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 04 - Seleccionar profesión**

| | | |
|---|---|---|
| Nombre: | UC 04 - Seleccionar profesión | |
| Actor principal: | Usuario | |
| Objetivo: | Seleccionar la profesión del usuario. | |
| Precondición: | Encontrarse en una pantalla que visualice el botón de profesión. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Activa el botón de selección de profesión. | 2: Muestra al usuario un selector de profesión. |
| | 3: Selecciona la profesión y la acepta. | 4: Cierra la pantalla de selección de profesión y establece la profesión seleccionada en la ficha de perfil. |
| Postcondición | Profesión en el campo de profesión. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 05 - Seleccionar fecha de nacimiento**

| | | |
|---|---|---|
| Nombre: | UC 05 - Seleccionar fecha de nacimiento | |
| Actor principal: | Usuario | |
| Objetivo: | Seleccionar la fecha de nacimiento del perfil del usuario. | |
| Precondición: | Encontrarse en una pantalla que visualice el botón de fecha de nacimiento. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Activa el botón de selección de fecha de nacimiento. | 2: Muestra al usuario un selector de fecha. |
| | 3: Selecciona la fecha deseada y la acepta. | 4: Cierra la pantalla de selección de fecha de nacimiento y establece la fecha seleccionada en la ficha de perfil. |
| Postcondición | Fecha establecida en el campo de fecha de nacimiento. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 06 - Rellenar organización**

| | | |
|---|---|---|
| Nombre: | UC 06 - Rellenar organización | |
| Actor principal: | Usuario | |
| Objetivo: | Añadir o editar la organización del perfil del usuario. | |
| Precondición: | Encontrarse en una pantalla que visualice el campo de organización. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Selecciona el campo de organización. | 2: Muestra el teclado para que el usuario rellene el campo. |
| | 3: Escribe el nombre de la organización y lo acepta. | 4: Cierra el teclado y establece la organización en la ficha de perfil. |
| Postcondición | Organización establecida en el campo de organización. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 07 - Rellenar email**

| | | |
|---|---|---|
| Nombre: | UC 07 - Rellenar email | |
| Actor principal: | Usuario | |
| Objetivo: | Añadir o editar el email del perfil del usuario. | |
| Precondición: | Encontrarse en una pantalla que visualice el campo de email. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Selecciona el campo de email. | 2: Muestra el teclado para que el usuario rellene el campo. |
| | 3: Escribe el email y lo acepta. | 4: Cierra el teclado y establece el email en la ficha de perfil. |
| Postcondición | Email establecido en el campo de email. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 08 - Seleccionar sexo**

| | | |
|---|---|---|
| Nombre: | UC 08 - Seleccionar sexo | |
| Actor principal: | Usuario | |
| Objetivo: | Seleccionar el sexo del perfil del usuario. | |
| Precondición: | Encontrarse en una pantalla que visualice el selector de sexo. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Activa la opción deseada en el selector de sexo. | 2: Establece el selector en la opción de sexo indicado. |
| Postcondición | Sexo establecido en el campo de selección de sexo. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | Habrá un sexo establecido por defecto. | |

**UC 09 - Seleccionar hora vídeos**

| | | |
|---|---|---|
| Nombre: | UC 09 - Seleccionar hora vídeos | |
| Actor principal: | Usuario | |
| Objetivo: | Seleccionar la hora para la disponibilidad de los nuevos vídeos. | |
| Precondición: | Encontrarse en una pantalla que visualice el botón de selección de horario de vídeos. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Activa el botón de selección de horario de vídeos. | 2: Muestra al usuario un selector de hora. |
| | 3: Selecciona la hora deseada y la acepta. | 4: Cierra la pantalla de selección de horario de vídeos y establece la hora seleccionada en la ficha de perfil. |
| Postcondición | Hora establecida en el campo de horario de vídeos. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 10 - Guardar perfil**

| | | |
|---|---|---|
| Nombre: | UC 10 - Guardar perfil | |
| Actor principal: | Usuario | |
| Objetivo: | Guardar el perfil del usuario. | |
| Precondición: | Ficha de perfil o registro visualizada. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Activa el botón de finalización de perfil. | 2: Guarda los datos del perfil del usuario. |
| Postcondición | Datos del perfil de usuario guardados. | |
| Sec. alternativas: | El usuario puede cambiar de pestaña sin guardar. | |
| Excepciones: | Muestra un mensaje de error si existen datos vacíos o mal rellenados. | |
| Req. especiales: | | |

**UC 11 - Consultar tutoriales**

| | | |
|---|---|---|
| Nombre: | UC 11 - Consultar tutoriales | |
| Actor principal: | Sistema | |
| Objetivo: | Mostrar los diferentes tutoriales del curso que puede visualizar el alumno. | |
| Precondición: | Haber concluido el registro y no haber comenzado el curso. | |
| Secuencia principal: | Usuario | Sistema |
| | | 1: Visualiza la pantalla de consultar tutoriales |
| Postcondición | Pantalla de tutoriales visualizada. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 12 - Ver vídeo**

| | | |
|---|---|---|
| Nombre: | UC 12 - Ver vídeo | |
| Actor principal: | Usuario | |
| Objetivo: | Reproducir el vídeo seleccionado. | |
| Precondición: | Encontrarse en una pantalla que permita iniciar la reproducción de un vídeo. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Pulsa sobre el botón de vídeo seleccionado | 2: Reproduce el vídeo seleccionado. |
| | 3: Pulsa el botón de finalizar. | 4: Regresa a la pantalla que lo invoco. |
| Postcondición | Encontrarse en la misma pantalla que lo invoco, y vídeo marcado como visto a tiempo si corresponde. | |
| Sec. alternativas: | | |
| Excepciones: | Si es la primera vez que se ve el último vídeo y dentro del tiempo, termina iniciando el caso de uso UC 20 - Realizar test. | |
| Req. especiales: | El usuario puede pulsar el botón de vídeo antes de que termina, además de controlar su reproducción mediante los controles del reproductor de vídeo. | |

**UC 13 - Consultar inicio**

| | | |
|---|---|---|
| Nombre: | UC 13 - Consultar inicio | |
| Actor principal: | Usuario | |
| Objetivo: | Mostrar la lección del curso actual y el progreso del alumno en el curso. | |
| Precondición: | Haber pasado la pantalla de tutoriales y no haber concluido el curso. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Activa la pestaña de inicio. | 2: Visualiza la pantalla de inicio. |
| Postcondición | Pantalla de inicio visualizada. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | Tras la pantalla de tutoriales, entra por defecto a la pestaña de inicio. | |

**UC 14 - Puntuar vídeo**

| | | |
|---|---|---|
| Nombre: | UC 14 - Puntuar vídeo | |
| Actor principal: | Usuario | |
| Objetivo: | Puntuar el vídeo. | |
| Precondición: | Concluir la visualización de un vídeo del curso dentro del tiempo establecido por primera vez. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Seleccionar una puntuación activar el botón de ir a tareas. | 2: Guarda la puntuación dada al vídeo y muestra la pantalla de lista de tareas. |
| Postcondición | Puntuación de vídeo guarda y pantalla de lista de tareas visualizada. | |
| Sec. alternativas: | El usuario puede decidir no puntuar el vídeo, dirigiéndose directamente a la lista de tareas. | |
| Excepciones: | | |
| Req. especiales: | Tras la pantalla de tutoriales, entra por defecto a la pestaña de inicio. | |

**UC 15 - Consultar historial**

| | | |
|---|---|---|
| Nombre: | UC 15 - Consultar historial | |
| Actor principal: | Usuario | |
| Objetivo: | Mostrar el historial de los vídeos previos del curso. | |
| Precondición: | Haber pasado la pantalla de tutoriales. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Activa la pestaña de historial. | 2: Visualiza la pantalla de historial con un listado de los vídeos previos del curso. |
| Postcondición | Pantalla de historial visualizada. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 16 - Consultar lista de tareas**

| | | |
|---|---|---|
| Nombre: | UC 16 - Consultar lista de tareas | |
| Actor principal: | Usuario | |
| Objetivo: | Mostrar las tareas añadidas por el alumno. | |
| Precondición: | Haber pasado la pantalla de tutoriales. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Activa la pestaña de tareas. | 2: Visualiza la pantalla de lista de tareas con un listado de todas las tareas previamente añadidas. |
| Postcondición | Pantalla de lista de tareas visualizada. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | El usuario accede a lista de tareas tras puntuar un vídeo o finalizar el test final. | |

**UC 17 - Marcar/Desmarcar tarea**

| | | |
|---|---|---|
| Nombre: | UC 15 - Marcar/Desmarcar tarea | |
| Actor principal: | Usuario | |
| Objetivo: | Marcar una tarea como realizada (o desmarcarla), para que el usuario lleve un control sobre las tareas que ha realizado. | |
| Precondición: | Encontrarse en la pantalla de lista de tareas. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Pulsa sobre la tarea desea. | 2: Marca el checkbox de la tarea y cambia el aspecto del texto de la tarea. |
| Postcondición | Tarea marcada y resaltada como realizada. | |
| Sec. alternativas: | Si la tarea ya se encuentra marcada, al pulsarla realiza el proceso contrario, desmarcando el checkbox y dandole el aspecto inicial a la tarea. | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 18 - Añadir tarea**

| | | |
|---|---|---|
| Nombre: | UC 18 - Añadir tarea | |
| Actor principal: | Usuario | |
| Objetivo: | Añadir una nueva nota a la lista de tareas. | |
| Precondición: | Encontrarse en la pantalla de lista de tareas. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Pulsa el botón para nueva tarea (+). | 2: Visualiza la pantalla de nueva tarea. |
| | 2: Rellena el campo de texto con la nueva tarea y pulsa el botón de aceptar. | 3: Guarda la nueva nota y vuelve a la pantalla de lista de tareas con la nueva tarea añadida a la lista como desmarcada. |
| Postcondición | Nota añadida y pantalla de lista de tareas visualizada. | |
| Sec. alternativas: | El usuario puede volver atrás, a la pantalla de lista de tareas cancelando el añadir la nueva tarea. | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 19 - Consultar perfil**

| | | |
|---|---|---|
| Nombre: | UC 19 - Consultar perfil | |
| Actor principal: | Usuario | |
| Objetivo: | Mostrar los datos del perfil del alumno y poder modificarlos. | |
| Precondición: | Haber pasado la pantalla de tutoriales y no haber concluido el curso. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Activa la pestaña de mi perfil. | 2: Visualiza la pantalla de mi perfil con un los campos rellenos con los datos del usuario. |
| Postcondición | Pantalla de perfil visualizada con los datos del usuario. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 20 - Realizar test**

| | | |
|---|---|---|
| Nombre: | UC 20 - Realizar test | |
| Actor principal: | Usuario | |
| Objetivo: | Realizar una serie de preguntas al alumno para que compruebe y fortalezca los conocimientos adquiridos en durante el curso. | |
| Precondición: | Haber visualizado o agotado el tiempo del ultimo vídeo del curso. | |
| Secuencia principal: | Usuario | Sistema |
| | | 1: Visualiza la pantalla de iniciar test. |
| | 2: Pulsa el botón de examen. | 3: Ejecuta el UC 21 - Responder pregunta. |
| | 4: Pulsa el botón de ir a lista de tareas. | 5: Visualiza la pantalla de lista de tareas. |
| Postcondición | Pantalla lista de tareas visualizada. | |
| Sec. alternativas: | | |
| Excepciones: | | |
| Req. especiales: | | |

**UC 21 - Responder pregunta**

| | | |
|---|---|---|
| Nombre: | UC 21 - Responder pregunta | |
| Actor principal: | Usuario | |
| Objetivo: | Responder una de las preguntas del test final del curso. | |
| Precondición: | Encontrarse realizando el test final. | |
| Secuencia principal: | Usuario | Sistema |
| | 1: Pulsa sobre uno de los botones de respuesta (verdadero-falso). | 2: Si la respuesta es correcta visualiza la pantalla de respuesta correcta, si es incorrecta visualiza la pantalla de respuesta incorrecta |
| | 3: Si la respuesta fue correcta pulsa el botón de siguiente pregunta, si fue incorrecta pulsa el botón de reintentar. | 4: Visualiza la pantalla de responder pregunta. Con la siguiente pregunta o repitiendo la misma en el caso de reintentar. |
| Postcondición | Pantalla de responder pregunta visualizada. | |
| Sec. alternativas: | Cuando se acierta correctamente la última pregunta pasa a visualizar la pantalla de fin de test. | |
| Excepciones: | | |
| Req. especiales: | | |

## Requisitos no funcionales

A continuación, se lista el conjunto de requisitos que, si bien no tienen que ver con la especificación funcional del sistema, describen características necesarias.

**R 01 - Localización**

| | |
|---|---|
| Nombre: | R 01 – Localización |
| Descripción: | El software deberá estar localizado al inglés y el español. Esto es necesario debido a que inglés y español son los lenguajes predominantes en el ámbito que pretende distribuirse la aplicación. |
| Necesidad: | Esencial |
| Fuente: | Cliente |

**R 02 - Orientación de la interfaz**

| | |
|---|---|
| Nombre: | R 02 – Orientación de la interfaz |
| Descripción: | Es conveniente que la orientación de la aplicación sea en vertical, pues resulta más cómoda de utilizar y puede utilizarse con una sola mano, no permitiendo la orientación horizontal pues podría resultar molesto para el usuario el cambio de orientación. |
| Necesidad: | Conveniente |
| Fuente: | Cliente |

**R 03 - Dispositivos**

| | |
|---|---|
| Nombre: | R 03 – Dispositivos |
| Descripción: | La aplicación deberá idearse para su ejecución en los dispositivos que admitan iOS 7 en adelante |
| Necesidad: | Esencial |
| Fuente: | Cliente |

## 6.3. Prototipos de interfaz de usuario - Wireframes

Una vez definidos los requisitos funcionales mediante los Casos de Uso, se puede realizar un prototipo de interfaz que satisfaga de una manera intuitiva todas las intenciones del usuario. Para ello, se realizará el prototipado usando wireframes, que son una herramienta de diagramado que facilita tanto la tarea del diseño preliminar de las pantallas que conforman la aplicación como el flujo entre ellas.

### WF 01 - Avisar/Permitir notificaciones

Alerta en pantalla que se muestra la primera vez que el usuario inicia la aplicación informandole de las notificaciones y permitiendole activarlas o no.

Acciones:

Ok → Activa las notificaciones y pasa a WF 02 - Introducción

No permitir → WF 02 - Introducción

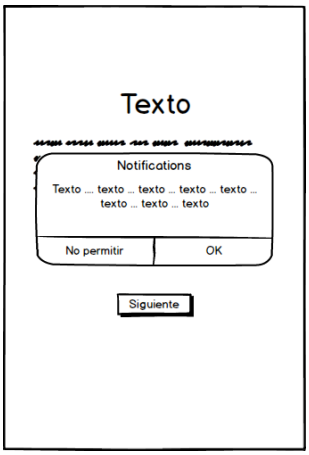



Figura 6.6: Wireframe 01: Avisar/Permitir notificaciones

### WF 02 - Introducción

Pantalla que hace de portada antes de iniciar el registro en el curso, mostrando el nombre del curso y un texto motivador.

Acciones:

Siguiente →WF 03 - Registro

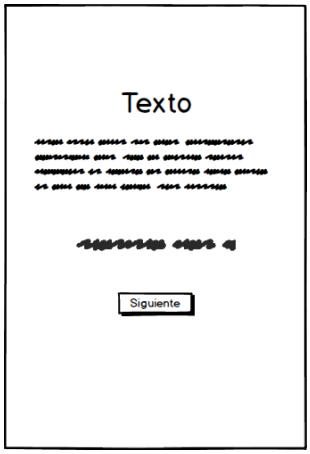



Figura 6.7: Wireframe 02: Introducción

## WF 03 - Registro

Pantalla en la que se rellenan los datos del usuario/alumno del curso a través de diferentes campos mediante texto y selectores que aparecen al seleccionar cada tipo de información a rellenar.

Acciones:

Ok → Si hay errores en la comprobación de información va a WF 07 - Error en registro/perfil, si no va a WF 08 - Tutoriales

Botón Profesión → WF 04 - Selección de profesión

Botón Fecha de nacimiento → WF 05 - Selección de fecha de nacimiento

Botón ¿A que hora verás los vídeos?→ WF 06 - Selección de horario de vídeos

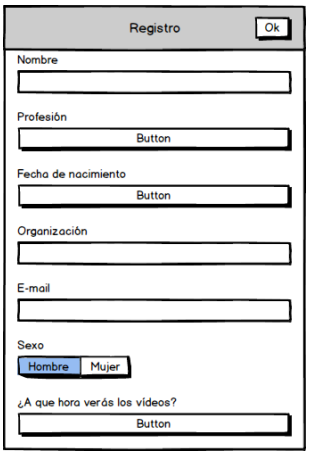



Figura 6.8: Wireframe 03: Registro

## WF 04 - Selección de profesión

Muestra las diferentes posibilidades para la profesión del usuario y permite seleccionar una.

Acciones:

Ok → Vuelve a la pantalla que lo invoco: WF 03 - Registro o WF 13 - Perfil

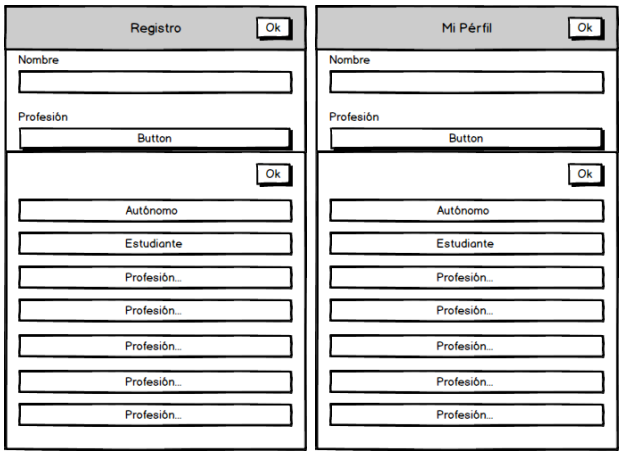



Figura 6.9: Wireframe 04: Selección de profesión

## WF 05 - Selección de fecha de nacimiento

Permite seleccionar la fecha de nacimiento del usuario mediante selectores.

Acciones:

Ok → Vuelve a la pantalla que lo invoco: WF 03 - Registro o WF 13 - Perfil

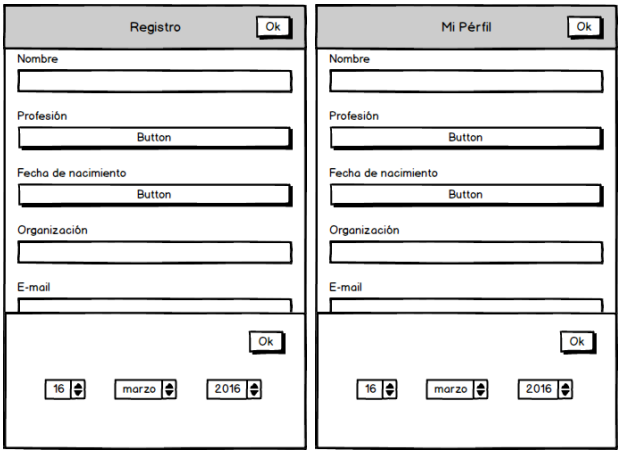



Figura 6.10: Wireframe 05: Selección de fecha de nacimiento

## WF 06 - Selección de horario de vídeos

Permite seleccionar a que hora del día se empieza a tener disponible un nuevo vídeo del curso mediante selectores.

Acciones:

Ok → Vuelve a la pantalla que lo invoco: WF 03 - Registro o WF 13 - Perfil

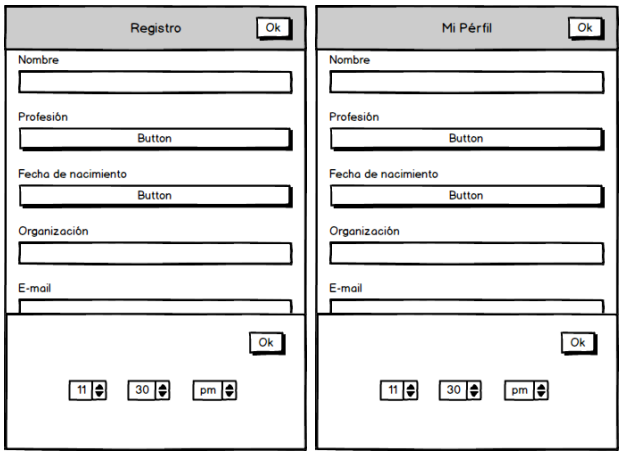



Figura 6.11: Wireframe 06: Selección de horario de vídeos

## WF 07 - Error en registro/perfil

Muestra una notificación indicando la existencia de fallos en los datos rellenados o la ausencia de los datos.

Acciones:

OK →Vuelve a la pantalla que lo invoco: WF 03 - Registro o WF 13 - Perfil

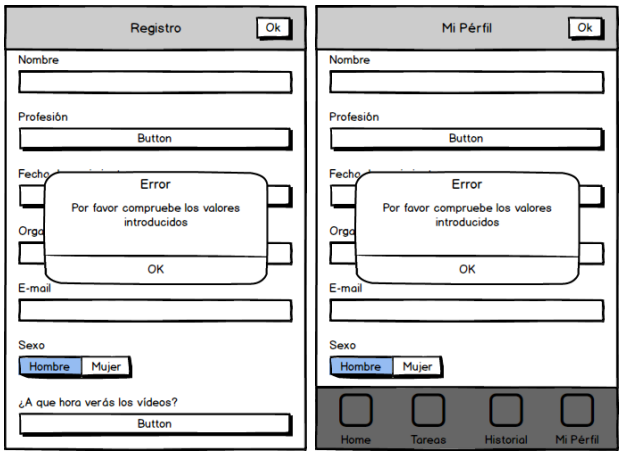



Figura 6.12: Wireframe 07: Error en registro

## WF 08 - Tutoriales

Esta pantalla muestra algunos tutoriales antes de empezar con el curso, que le pueden ser de utilidad al usuario.

Acciones:

Empezar → WF 09 - Inicio

Selección de tutorial → Muestra el vídeo correspondiente y vuelve a esta pantalla.

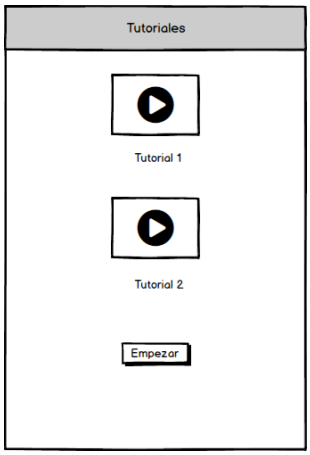



Figura 6.13: Wireframe 08: Tutoriales

## WF 09 - Inicio

Pantalla de inicio durante el transcurso del curso, donde nos indica la lección actual con un enlace a su respectivo vídeo. También muestra información de como va el curso, indicando el progreso actual del curso y el porcentaje que el alumno ha visto dentro del tiempo indicado para cada lección.

Acciones:

Video → Muestra el vídeo correspondiente y si es la primera vez que se reproduce el vídeo luego va a WF 15 - Fin de vídeo, si no después del vídeo vuelve a WF 09 - Inicio

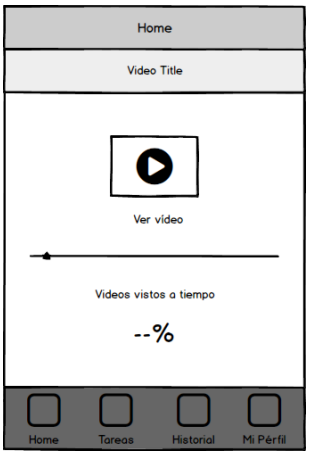



Figura 6.14: Wireframe 09: Inicio

## WF 10 - Lista de tareas

Muestra las notas que el usuario añade a su lista de tareas y permite marcar/desmarcar las tareas.

Acciones:

Selección de tarea → Marca/Desmarca la tarea y reordena la lista por empezando por la tareas desmarcadas.

+ → WF 11 - Nueva tarea

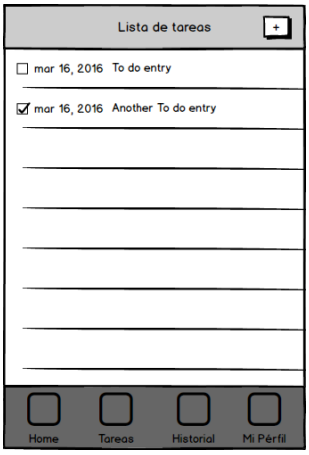



Figura 6.15: Wireframe 10: Lista de tareas

## WF 11 - Nueva tarea

Permite incluir una nueva tarea a la lista de tareas, añadiendo texto.

Acciones:

Lista de tareas → Cancela la introducción de la nueva tarea y va a WF 10 - Lista de tareas

Ok → Añade la nueva tarea desmarcada a la lista de tareas y va a WF 10 - Lista de tareas

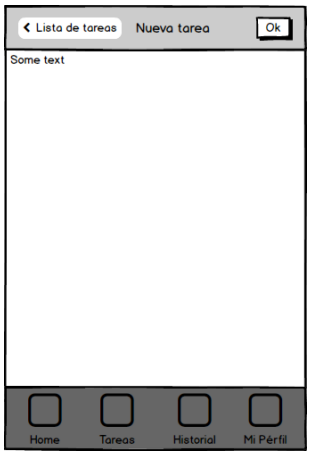



Figura 6.16: Wireframe 11: Nueva tarea

## WF 12 - Historial

Muestra una lista de los videos de las lecciones anteriores y permite volver a verlos.

Acciones:

Selección de video → Muestra el video correspondiente y vuelve a esta pantalla

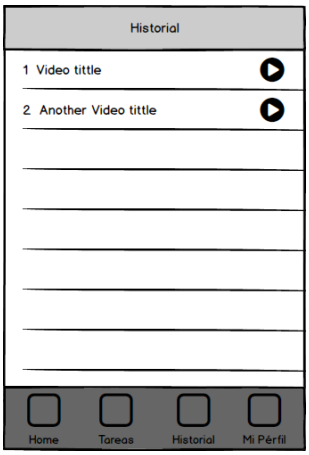



Figura 6.17: Wireframe 12: Historial

### WF 13 - Perfil

Pantalla que permite modificar los datos y preferencias del usuario/alumno.

Acciones:

Ok → Si hay errores en la comprobación de información va a WF 07 - Error en registro/perfil, si no va a WF 14 - Perfil guardado

Botón Profesión → WF 04 - Selección de profesión

Botón Fecha de nacimiento → WF 05 - Selección de fecha de nacimiento

Botón ¿A que hora verás los vídeos?→ WF 06 - Selección de horario de vídeos

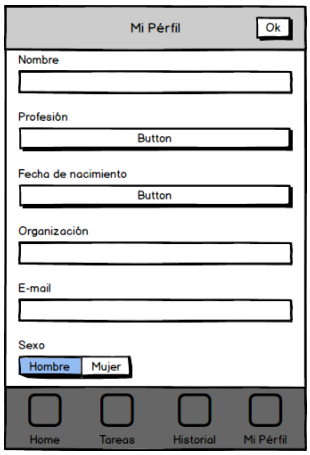



Figura 6.18: Wireframe 13: Perfil

## WF 14 - Perfil guardado

Pantalla que informa que los datos modificados en los datos de perfil del usuario/alumno han sido guardados y actualizados correctamente.

Acciones:

OK → WF 13 - Perfil

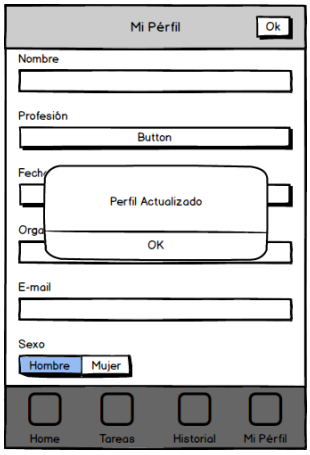



Figura 6.19: Wireframe 14: Perfil guardado

## WF 15 - Fin de vídeo

Esta pantalla se muestra para poder puntuar el vídeo de la lección actual, tras la primera vez que se visualiza, mediante una escala equivalente del 1 al 5, y lleva a la lista de tareas para que el usuario/alumno pueda añadir notas de la lección que acaba de ver.

Acciones:

Mis tareas → Establece la puntuación si se ha votado y va a WF 10 - Lista de tareas

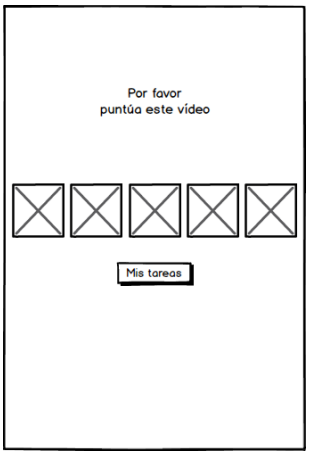



Figura 6.20: Wireframe 15: Fin de vídeo

## WF 16 - Fin de curso

Muestra una felicitación por haber concluido el curso y permite iniciar un test sobre el contenido del curso.

Acciones:

Examen → WF 17 - Pregunta de test

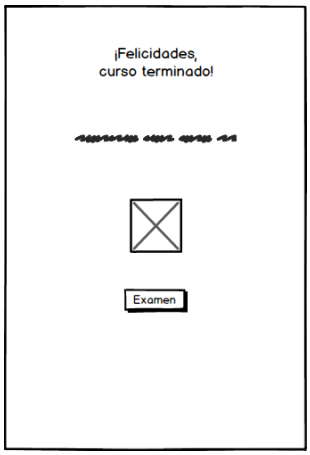



Figura 6.21: Wireframe 16: Fin de curso

### WF 17 - Pregunta de test

Realiza un pregunta sobre el contenido del curso, que el usuario/alumno puede responder mediante verdadero-falso.

Acciones:

Verdadero → Si la respuesta es incorrecta WF 18 - Respuesta de test errónea, si es correcta WF 19 - Respuesta de test correcta, a excepción de la última pregunta que va a WF 20 -Fin de test

Falso → Si la respuesta es incorrecta WF 18 - Respuesta de test errónea, si es correcta WF 19 - Respuesta de test correcta, a excepción de la última pregunta que va a WF 20 -Fin de test

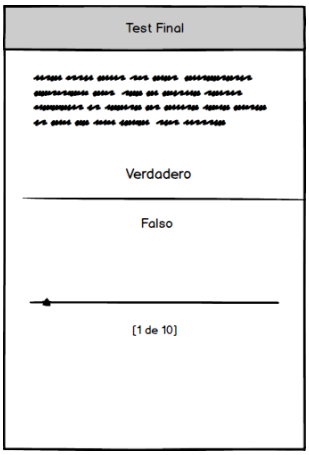



Figura 6.22: Wireframe 17: Pregunta de test

## WF 18 - Respuesta de test errónea

Indica que se ha respondido la pregunta erróneamente y ofrece una explicación a la repuesta correcta de la pregunta de test, para luego permitir volver a la misma pregunta y que el usuario/alumno intente responderla correctamente.

Acciones:

Reintentar → Retorna a la misma pregunta en WF 17 - Pregunta de test


Figura 6.23: Wireframe 18: Respuesta de test errónea

## WF 19 - Respuesta de test correcta

Felicita al usuario/alumno por responder correctamente y permite ir a la siguiente pregunta

Acciones:

Siguiente→ Muestra la siguiente pregunta en WF 17 - Pregunta de test


Figura 6.24: Wireframe 19: Respuesta de test correcta

## WF 20 - Fin de test

Felicita al usuario/alumno por haber terminado el test y le permite ir a su lista de tareas.

Acciones:

Ir a mis tareas → WF 10 - Lista de tareas

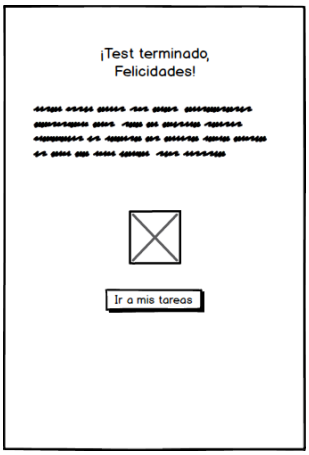



Figura 6.25: Wireframe 20: Fin de test

# Capítulo 7

# Diseño

En este apartado se estudia el desarrollo del diseño del sistema informático a implementar , con especial dedicación a la aplicación de aprendizaje, explicando cada uno de los módulos que la componen.

## 7.1. Diseño arquitectónico del sistema

El diseño arquitectónico del sistema describe cómo se comunica el software consigo mismo, con los sistemas que operan junto a él y con los operadores o usuarios que lo utilizan. Así, en el presente apartado se describirán las diferentes partes que forman la arquitectura del sistema realizando, para ello, una descomposición basada en subsistemas hardware y subsistemas software.

- Subsistemas hardware: elementos que componen la arquitectura hardware del ecosistema de la aplicación, donde cada elemento constituye un nodo dentro de una red, ya que se comunica con otros elementos hardware.
- Subsistemas software: sistemas de funcionalidad independiente. Están divididos en módulos de software, los cuales se comunican entre sí mediante interfaces bien definidas para completar la oferta de servicios disponibles para el usuario final.

### Subsistemas hardware

La arquitectura de este sistema está principalmente basada en el modelo Cliente-Servidor. Se trata de un modelo de aplicación distribuida en el que las tareas

se reparten entre los proveedores de recursos o servicios (servidor) y los demandantes (clientes).

#### Dispositivos iOS

Dispositivos donde será instalada y utilizada la aplicación de aprendizaje por los usuarios finales. Se conectarán a través de internet a un servidor para recibir y mandar los datos necesarios en la realización del curso así como la recogida de estadísticas.

#### Infraestructura de red

El acceso a internet que permite conectar la aplicación con los datos almacenados en el servidor.

#### Servidor Parse

Servicio de backend que permite alojar y manejar los datos de los que hará uso la aplicación, además de manejar usuarios, obtener estadísticas de uso y gestionar los cursos que se lleguen a ofrecer. Los servidores son ofrecidos transparentemente, ya que se trata de un servicio que ofrece Parse.

### Subsistemas software

#### Aplicación de aprendizaje

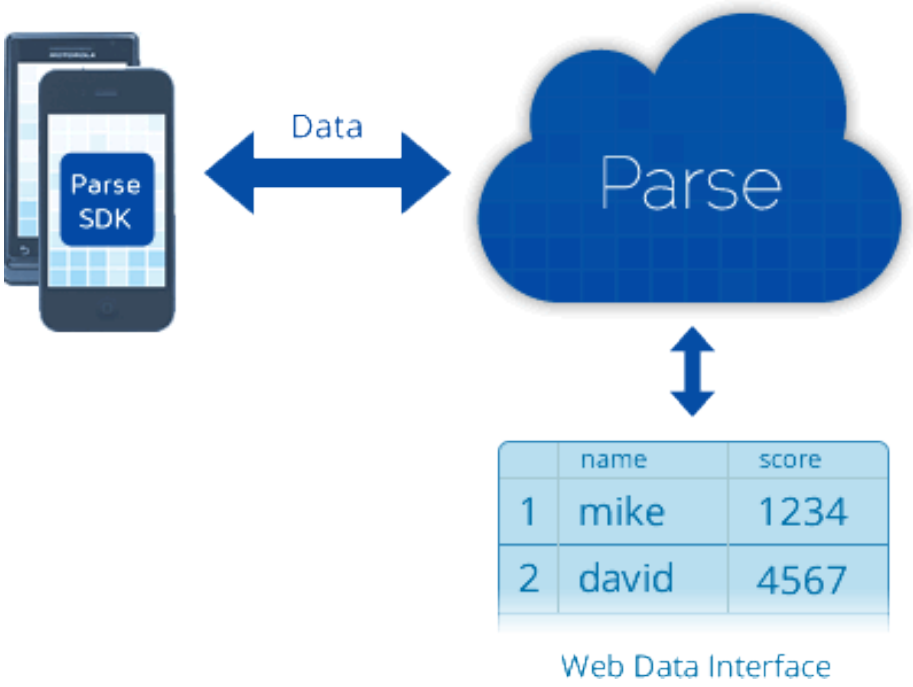
Es la parte más importante del desarrollo. Se trata de una aplicación que los usuarios pueden descargar en sus dispositivos iOS. Sirve exclusivamente para el uso de los alumnos, y no de los desarrolladores de cursos o administradores. Dada sus importancia, se detallará más detalladamente en la sección destinada al análisis del diseño arquitectónico de la aplicación.

#### iOS

Sistema operativo que cuenta con multitud de subsistemas incorporados, ofreciendo una considerable cantidad de servicios y funcionalidades. Es el único sistema operativo que se ejecuta en los dispositivos móviles de Apple(en sus diferentes versiones numeradas).

### Servicios de Parse

Parse constituye el otro gran sistema de la arquitectura: el sistema del lado del servidor.




Cuadro 7.1: Arquitectura Parse

Este sistema se compone de 3 subsistemas:

1. **Parse SDK:** Este subsistema nos proporciona la API y los mecanismos de conexión a la misma para acceder a la información alojada en en servidor desde la aplicación.
2. **Parse Cloud:** Este subsistema nos proporciona las bases de datos, la lógica de acceso a ellas y una interfaz gráfica amigable para poder administrar la información almacenada. Esta interfaz será usada para la gestión de la información relativa a los cursos por parte de un administrador.
3. **Parse Analytics:** Subsistema de analíticas que nos proporciona Parse y que nos permite recoger datos sobre el uso de la aplicación móvil.

## 7.2. Diseño de la Interfaz de usuario

En esta sección, se especifican diversos elementos relacionados con la interacción del usuario con la aplicación.

### Pantalla

La pantalla en una aplicación iOS es importante, pues no solo sirve para mostrar textos, gráficos y material multimedia, que además nos permite interactuar con ella pulsando diferentes elementos o realizando otros gestos. Se trabajará pensando en que todo el contenido quede centrado en la pantalla , y que tenga un relación de aspecto similar en todos los dispositivos, pudiendo tratarse de un iPhone o un iPad.

### Preferencias

Permite al usuario establecer ciertos ajustes para la aplicación desde el panel de preferencias del sistema, aunque pueden modificarse en cualquier momento, son ajustes que no suelen modificarse tras haber sido establecidos por primera vez . En esta ocasión, servirán para que el usuario seleccione si desea o no recibir notificaciones, lo cual se le preguntará al acceder a la aplicación por primera vez.

### Orientación del dispositivo

Se ha establecido que la única orientación permitida en la aplicación sea vertical, a excepción de la reproducción de vídeos . La desactivación de la autorotación se lleva a cabo para no se produzcan las incomodidades derivadas del movimiento que afectan a cómo se presenta el contenido de la aplicación. Asimismo, se ha elegido la orientación vertical debido a que ésta resulta más cómoda y permite que el usuario pueda manipular la aplicación con una sola mano.

## 7.3. Diseño Arquitectónico de la aplicación

El desarrollo de aplicaciones iOS está especialmente unido al patrón Modelo-Vista-Controlador[**?**], siendo éste sobre el cual se ha construido la arquitectura de esta aplicación.

### Patrón Modelo-Vista-Controlador (MVC)

El Modelo Vista Controlador (MVC) es un patrón de diseño software que separa la lógica de la aplicación en tres capas diferenciadas: modelos, vistas y controladores.

- El Modelo representa la parte de la aplicación que implementa la lógica de negocio. Esto significa que es responsable de la recuperación de datos, convirtiéndolos en conceptos significativos para la aplicación, así como de su procesamiento, validación, asociación y cualquier otra tarea relativa a la manipulación de dichos datos.
- La Vista es la capa encargada de mostrar la información al usuario y controla los mecanismos interacción con éste.
- El Controlador actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

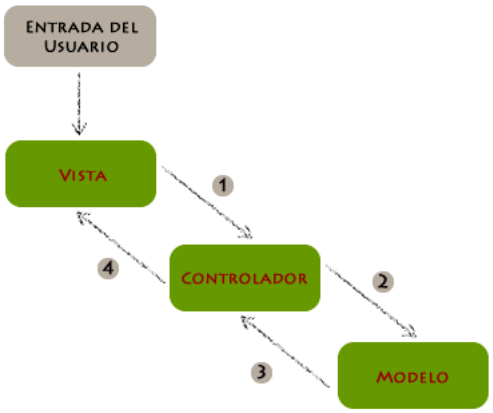



Figura 7.1: Patrón MVC

El flujo de comunicación entre capas queda definido tal como se aprecia en la imagen anterior:

1-) El usuario interactúa con la aplicación y la vista (U objeto de la vista) a través de la interfaz de usuario.

2-) La vista le envía un mensaje al controlador comunicándole, por ejemplo, que hemos presionado un botón y queremos que ello responda a alguna acción.

3-) El controlador recibe el mensaje y contacta con el modelo para realizar la acción y actualizar la información pertinente.

4-) El controlador recoge la información requerida por la vista ya actualizada por el modelo, y por último, actualiza la vista con los cambios que se han hecho en el modelo.

## División de roles en la aplicación

A continuación se describirán todos los elementos que componen cada una de estas capas en la aplicación.

### Modelos

#### User

Este modelo proporciona la lógica para manejar la información de usuarios en la aplicación. Presenta operaciones para la creación, actualización y obtención de información de los usuarios.

| User |
| --- |
| - NSString *modelId<br>- NSDate *birthDate<br>- NSString *email<br>- NSString *gender<br>- NSString *name<br>- NSString *occupation<br>- NSString *organization<br>- NSString *videoAvaliableTime<br>- PFObject *pfObject<br>- NSString * modelId |
| - (void)fetchUserInformationWithCompletion:(void (^)(User *currentUser, NSError *error))completion<br>- (void)updateUserInformationWithCompletion:(void (^)(User *currentUser, NSError *error))completion<br>- (instancetype)init<br>- (void)updatePFObjectInformation<br>- (void) updateWithRemoteInformation |

Figura 7.2: Modelo User

**Lesson**

Este modelo proporciona la lógica para manejar la información de cada lección en la aplicación. Presenta operaciones para la actualización y obtención de información de la lección.

| Lesson |
| --- |
| - NSDate * date<br><br>- NSString * video<br><br>- NSNumber * done |
| + (void)getLessonsForCourseWithCompletionBlock:(void (^)(NSArray *lessons, NSError *error))completion<br><br>+ (void)getVideosForLessonsWithCompletionBlock:(void (^)(NSArray *videos, NSError *error))completion<br><br>+ (BOOL) hasLessonsAvaliable<br><br>+ (BOOL) hasLessonNow<br><br>+ (Lesson *) getActualLesson<br><br>+ (NSUInteger) videosViewed<br><br>- (void) markLikeDone<br><br>- (void)updateWithDate:(NSDate *)date andCompletion:(void (^)(NSError *error))completion |

Figura 7.3: Modelo Lesson

**ToDo**

Este modelo proporciona la lógica necesaria para manejar la información de las tareas del usuario. Presenta operaciones la creación, eliminación, actualización y obtención de la lista de tareas.

| ToDo |
| --- |
| - NSDate * date<br><br>- NSString * body<br><br>- NSNumber * done<br><br>- NSString * parseId |
| + (NSFetchedResultsController *)toDosFetcher<br><br>+ (instancetype)newToDo<br><br>- (void)updateStatusWith:(NSNumber *)status<br><br>- (void)updateWithBody:(NSString *)body status:(NSNumber *)status date:(NSDate *)date andID:(NSString *)serverId<br><br>- (void)destroyToDoWithCompletionBLock:(void (^)(NSError *error))completion |

Figura 7.4: Modelo ToDo

**Test**

Este modelo proporciona la lógica necesaria para manejar la información relativa al examén del estudiante. Presenta la operación para obtener las lista de preguntas del examen para el estudiante.

| Test |
| --- |
| + (void)getTestQuestionsForUser:(User *)user withCompletion:(void (^)(NSArray *questions, NSError *error))completion; |

Figura 7.5: Modelo Test

**Test_Question**

Este modelo proporciona la lógica necesaria para manejar cada pregunta del examen.

| TestQuestion |
| --- |
| - Bool * answer<br><br>- NSString * explanation<br><br>- NSString * statement<br><br>- NSString * modelId |
| - (instancetype)initWithDictionary:(NSDictionary *)dictionary;<br><br>- (void)markAsCorrectForUser:(User *)user withCompletion:(void (^)(NSError *error))completion;<br><br>- (void)markAsWrongForUser:(User *)user withCompletion:(void (^)(NSError *error))completion; |

Figura 7.6: Modelo TestQuestion

### Controladores

#### VideoPlayerViewController

Controla la reproducción de vídeos de la aplicación. Muestra al usuario el reproductor de vídeo nativo al usuario y se encarga de la transición a la siguiente pantalla una vez el usuario ha finalizado la reproducción del mismo.

| VideoPlayerViewController |
| --- |
| + id delegate<br><br>+ NSString *video_url<br><br>+ BOOL need_vote |
| - (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil<br><br>- (void)viewDidLoad<br><br>- (void)didReceiveMemoryWarning<br><br>- (void)playVideo<br><br>- (void) moviePlayBackDidFinish:(NSNotification*)notification<br><br>- (BOOL)shouldAutorotate<br><br>- (UIInterfaceOrientation)preferredInterfaceOrientationForPresentation<br><br>- (NSUInteger)supportedInterfaceOrientations |

Figura 7.7: Clase VideoPlayerViewController

**AccessViewController**

Controla el flujo de navegación inicial de la aplicación, dirigiendo al usuario bien a la lección actual bien al test de autoevaluación en caso de haber terminado el curso.

| AccessViewController |
| --- |
| - (void)viewDidLoad<br>- (void)viewDidAppear:(BOOL)animated<br>- (void)didReceiveMemoryWarning<br>- (void)dismissModals |

Figura 7.8: Clase AccessViewController

**MotivationAdviceViewController**

Controla la pantalla que muestra al usuario el texto inicial de la aplicación y se encarga de la transición a la pantalla que contiene el formulario de registro de usuario.

| MotivationAdviceViewController |
| --- |
| - UIButton *enterButton<br>- UILabel *titleAppLabel<br>- UILabel *adviceLabel |
| - (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil<br>- (void)viewDidLoad<br>- (void)viewWillAppear:(BOOL)animated<br>- (void)didReceiveMemoryWarning |

Figura 7.9: Clase MotivationAdviceViewController

**SignUpViewController**

Controlar la pantalla del proceso de registro de usuario, y es reutilizado a su vez para editar la información del usuario una vez éste haya sido registrado.

SignUpViewController

- (void)viewDidLoad

- (void)viewDidAppear:(BOOL)animated

- (void)signUpButtonTapped

- (BOOL)textFieldShouldReturn:(UITextField *)textField

- (IBAction)selectADate:(UIControl *)sender

- (void)dateWasSelected:(NSDate *)selectedDate
element:(id)element

- (IBAction)selectAnOccupation:(UIControl *)sender

- (void)OccupationWasSelected:(NSNumber *)selectedIndex
element:(id)element

- (void)actionPickerCancelled:(id)sender

- (IBAction)selectATime:(UIControl *)sender

- (void)timeWasSelected:(NSDate *)selectedTime
element:(id)element

- (BOOL)validateForm

- (void)prepareFieldForUpdate

- (BOOL)NSStringIsValidEmail:(NSString *)checkString

- (void)generateNotification:(NSDate *)date
:(NSString*)message

- (PFObject *)getVideoFromLesson: (Lesson *) lesson

Figura 7.10: Clase SignUpViewController

**TutorialViewController**

Controla la pantalla que muestra el tutorial de uso de la aplicación.

| TutorialViewController |
| --- |
| - UILabel *pageTitleLabel<br>- UILabel *statusBarBackgroundLabel<br>- UIButton *beginButton<br>- UILabel *descriptionVideo2Label<br>- UILabel *descriptionVideo1Label<br>- UIView *scrollContentVIew<br>- UIScrollView *scrollView<br>- UIImageView *tutorial1placeholderImageView<br>- UIImageView *tutorial2PlaceholderImageView |
| - (id)initWithNibName: (NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil<br>- (void)viewDidLoad<br>- (void)viewDidLayoutSubviews<br>- (void)viewDidAppear:(BOOL)animated<br>- (void)didReceiveMemoryWarning<br>- (IBAction)Begin:(id)sender<br>- (void) playTutorial1: (UITapGestureRecognizer *)sender<br>- (void) playTutorial2: (UITapGestureRecognizer *)sender<br>- (void)playVideo: (NSString *)path |

Figura 7.11: Clase TutorialViewController

**TabbarViewController**

Se encarga del control de la pantalla principal de la aplicación, en la que el usuario encuentra un tabbar que le permite ir tanto a la lección actual como a su lista de tareas o el historial de los vídeos vistos.

| TabbarViewController |
| --- |
| - (void)setViewControllers:(NSArray<__kindof UIViewController *> *)viewControllers animated:(BOOL)animated;<br><br>- (void)viewDidLoad |

Figura 7.12: Clase TabbarViewController

**ToDoViewController**

Responsable de controlar la lista de tareas, mostrando en ella todas las tareas que ha añadido el usuario, marcándolas o desmarcándolas como realizadas y actualizándolas en función de las acciones del usuario.

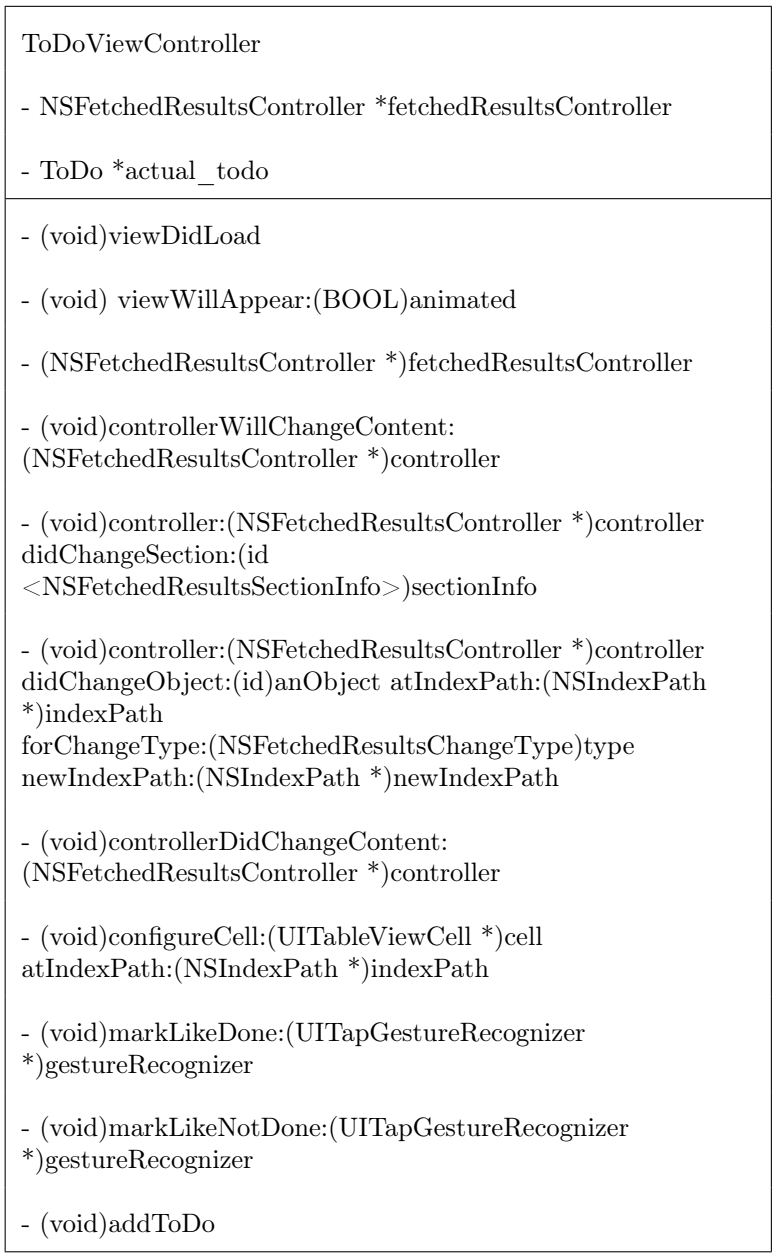



Figura 7.13: Clase ToDoViewController

**EditTodoViewController**

Controla la pantalla de añadir una nueva tarea, y gestionando las operaciones de crearlas o editarlas.

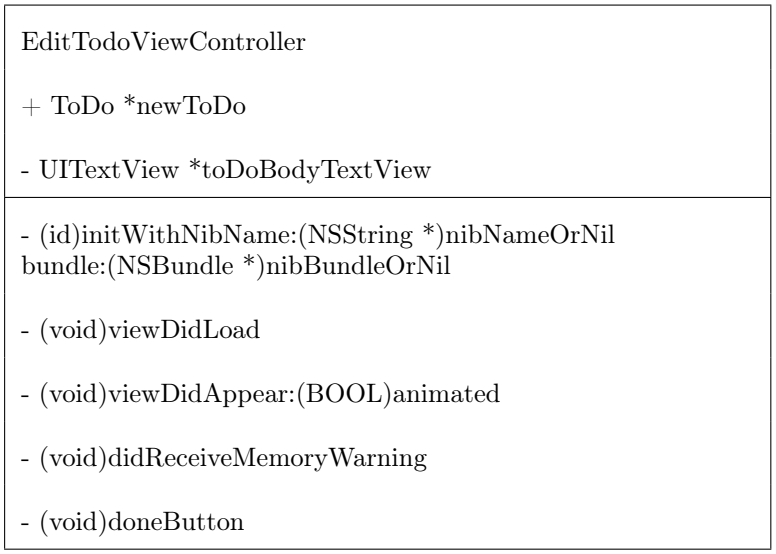



Figura 7.14: Clase EditTodoViewController

**HomeViewController**

Controla la pantalla central del aplicación, mostrando al usuario el progreso realizado y ofreciéndole la opción de ver la lección actual.

| HomeViewController |
| --- |
| - VideosProgressBar *videoProgressBarView<br>- UILabel *HomeTitle<br>- UIImageView *placeholder<br>- UILabel *startVideoLabel<br>- UILabel *videoWachedLabel<br>- UILabel *percentLabel<br>- PFObject *video<br>- Lesson *lesson<br>- NSManagedObjectContext *moc |
| - (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil<br>- (void)viewDidLoad<br>- (void)viewWillAppear: (BOOL)animated<br>- (void)didReceiveMemoryWarning<br>- (void)playVideo:(UITapGestureRecognizer *)sender<br>- (void)loadData<br>- (void)loadPage<br>- (void)videoFinishCallback |

Figura 7.15: Clase HomeViewController

**VoteViewController**

Controla la pantalla en la que el usuario puede valorar la lección.

| voteViewController |
| --- |
| + PFObject *video<br>- UILabel *voteTextLabel<br>- UIButton *goToDoButton<br>- UIButton *addOnePointButton<br>- UIButton *addTwoPointButton<br>- UIButton *addThreePointButton<br>- UIButton *addFourPointButton<br>- UIButton *addFivePointButton<br>- PFObject *current_user |
| - (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil<br>- (void)viewDidLoad<br>- (void)didReceiveMemoryWarning<br>- (IBAction)addPointsTapp:(id)sender<br>- (IBAction)goToDoList:(id)sender |

Figura 7.16: Clase voteViewController

**HistoryViewController**

Controla la pantalla de historial, permitiendo al usuario poder volver a visualizar cualquiera de las lecciones recibidas.

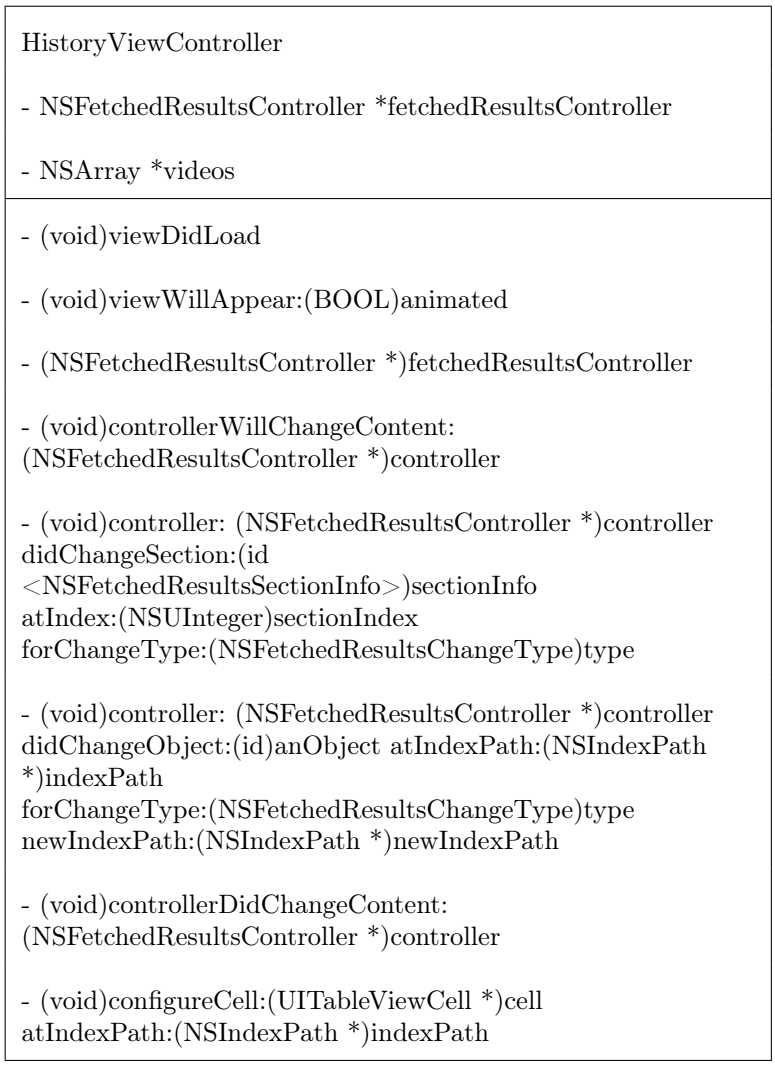



Figura 7.17: Clase HistoryViewController

**TestViewController**

Controla el test final de autoevaluación ofreciendo una pregunta verdadero-falso al usuario y mostrando información del porcentaje de preguntas respondidas.

| TestViewController |
| --- |
| - UITableView *questionsTableVIew<br><br>- NSArray *questions<br><br>- BOOL responseWasCorrect<br><br>- NSUInteger index<br><br>- NSUInteger n_questions<br><br>- (void)viewDidLoad<br><br>- (UIView *) tableView:(UITableView *)tableView viewForFooterInSection:(NSInteger)section<br><br>- (BOOL)tableView: (UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath<br><br>- (void)configureCell:(UITableViewCell *)cell atIndexPath:(NSIndexPath *)indexPath<br><br>- (void)nextQuestion<br><br>- (void)prepareInfo |

Figura 7.18: Clase TestViewController

**QuestionConfirmViewController**

Controla la pantalla de verificación de la respuesta durante el proceso de tests informando al usuario de si ha respondido positivamente.

| QuestionConfirmViewController |
| --- |
| - UILabel *textConfirmLabel |
| - UIImageView *confirmImageView |
| - UIButton *confirmButton |
| - (void)viewDidLoad |
| - (void)didReceiveMemoryWarning |
| - (IBAction)confirmButtonTap:(id)sender |

Figura 7.19: Clase QuestionConfirmViewController

**WrongAnswerViewController**

Controla la pantalla de verificación de la respuesta durante el proceso de tests informando al usuario de si ha respondido negativamente.

| WrongAnswerViewController |
| --- |
| + NSString *message |
| - UILabel *textConfirmLabel |
| - UIImageView *confirmImageView |
| - UIButton *confirmButton |
| - (void)viewDidLoad |
| - (void)didReceiveMemoryWarning |
| - (IBAction)confirmButtonTap:(id)sender |

Figura 7.20: Clase WrongAnswerViewController

**FinishCourseViewController**

Controla la pantalla de finalización del curso, informando al usuario de que el curso ha terminado y redirigiéndole a su historial o lista de tareas.

| FinishCourseViewController |
| --- |
| - UILabel *noticeLabel<br><br>- UIImageView *imageView<br><br>- UIButton *goToButton<br><br>- UILabel *explanationLabel |
| - (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil<br><br>- (void)viewDidLoad<br><br>- (void)didReceiveMemoryWarning |

Figura 7.21: Clase FinishCourseViewController

# Capítulo 8

# Implementación

El proceso de implementación ha sido el más extenso en tiempo de todo el desarrollo del proyecto. En este capítulo destacaremos algunas partes interesantes de este proceso sin entrar en demasiados detalles sobre la codificación.

## 8.1. StoryBoards

A la hora de enfrentarte a la implementación de una interfaz gráfica para una aplicación iOS existen tres grandes herramientas que puedes usar:

1. **Código Personalizado**: Todo el posicionamiento de elementos en la vista así como las animaciones, transiciones etc se maneja mediante código personalizado escrito por el desarrollador. Este es el enfoque más costoso a la hora de crear una interfaz de usuario ya que obliga a programar mucho más, pero es útil cuando tenemos vistas con un alto nivel de personalización que no es fácilmente alcanzable con los generadores de interfaz.
2. **Ficheros XIBs :** Consiste en el uso de "ficheros de interfaz" que tienen extensión **.xib.** Estos ficheros son creados y editados con "Interface Builder", el editor gráfico que proporciona XCode, y cada fichero corresponde a una vista de las mostradas en la Interfaz de Usuario. De esta manera podemos hacer uso del editor gráfico para ayudarnos al posicionamiento y maquetación de los elementos de la vista.
3. **StoryBoards:** Esta herramienta nos permite llevar un paso más allá el concepto de XIB file. StoryBoard es una herramienta visual que te permite construir, mediante Interface Builder, la maquetación de varias vistas de la aplicación así como definir las transiciones entre ellas.

Ninguna de estas opciones es una solución universal y es decisión del desarrollador elegir el enfoque a seguir para la implementación de cada vista pudiendo incluso combinarse las distintas opciones dentro del mismo proyecto.

Durante el desarrollo de esta aplicación, aunque hemos usado código personalizado cuando ha sido necesario, se ha seguido un enfoque basado en StoryBoards.

La aplicación se ha dividido en 3 flujos de navegación que están relacionados entre ellos:

- Registro del Usuario
- Panel de trabajo, que incluye la visualización de la lección actual, el acceso a las tareas pendientes y el acceso al historial
- Evaluación final del curso

Cada uno de estos 3 flujos de navegación ha sido creado mediante un StoryBoard diferente, quedando de la siguiente manera.

## Registro de Usuario

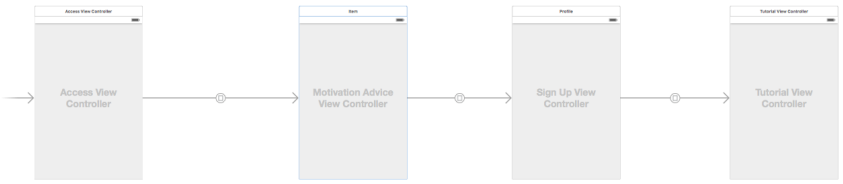



Figura 8.1: UserRegistration StoryBoard

### Panel de trabajo


Figura 8.2: WorkArea StoryBoard

## Evaluación

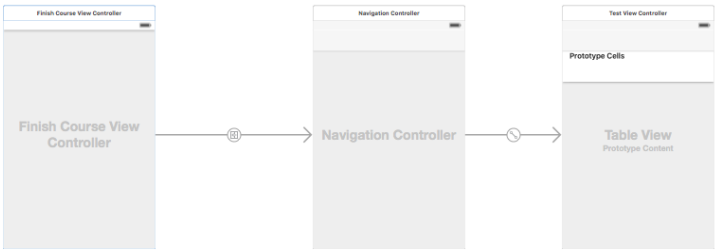



Figura 8.3: Test StoryBoard

Cada uno de los rectángulos representa a un controlador de la interfaz de usuario, y cada una de las flechas representa la transición desde un controlador al siguiente en ese flujo de trabajo.

Veamos más en detalle la transición entre el MotivationAdviceViewController y el SignUpViewController.

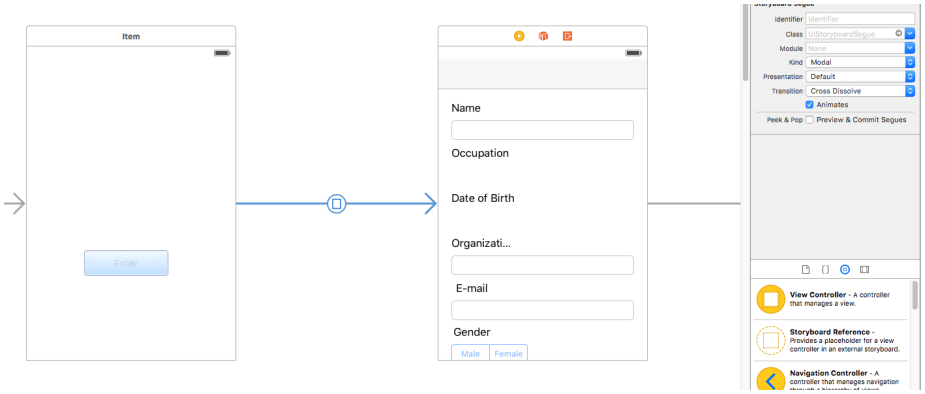



Figura 8.4: storyBoard transition

Durante la fase de registro, al usuario se le presenta una pantalla con una texto de presentación y a continuación el usuario debe presionar un botón para navegar hasta el formulario de registro.

La imagen anterior muestra el detalle de estos 2 controladores. Cuando el usuario presiona sobre el botón del primer controlador, se dispara la transición configurada y el segundo controlador es mostrado en pantalla.

En la parte derecha de la imagen anterior podemos ver las opciones de configuración de la transición elegida para pasar de un controlador al otro.

En este caso se trata de una transición Modal con efecto Cross Disolve.

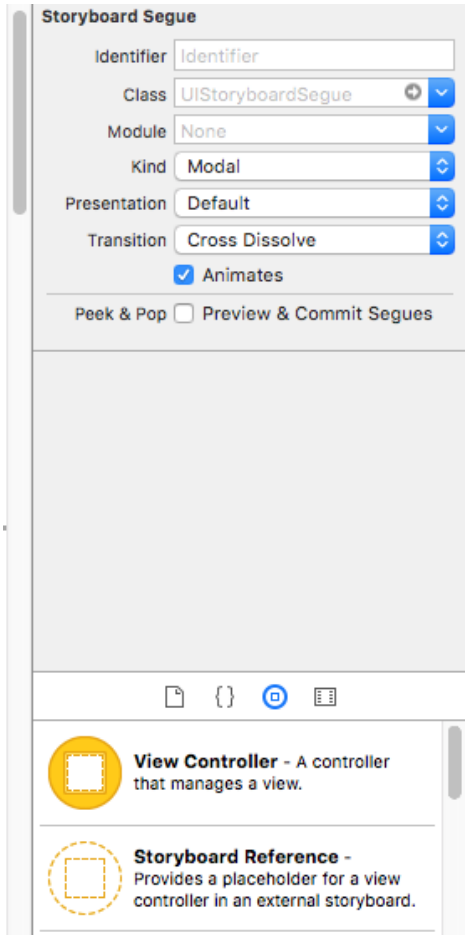



Figura 8.5: transition configuration

## 8.2. Autolayout.

Uno de los problemas a los que se enfrenta un desarrollador cuando implementa una interfaz de usuario para dispositivos móviles es mantener la misma estética en dispositivos con diferentes tamaños de pantalla. Como conseguir que tu aplicación tenga un aspecto similar independientemente del tamaño de la pantalla

del dispositivo Para resolver este problema Apple nos ofrece su herramienta de autolayout.

Autolayout es un sistema que nos permite especificar la posición y tamaño de los elementos en la pantalla en base a una serie de restricciones. Estas restricciones definen la posición de los elementos en la pantalla en relación a otros elementos que se encuentran en la misma.

Veamos un ejemplo básico del uso de Autolayout para entender su funcionamiento.

Supongamos que queremos colocar un botón en el centro de la pantalla de nuestro dispositivo. Para conseguir esto podríamos indicar en código que se dibuje un botón en las coordenadas (X, Y) donde X indicaría en centro en el eje de ordenadas e Y indicaría el centro en el eje de abscisas. O usar el editor gráfico y colocar el botón en el centro de la pantalla quedando de la siguiente manera:

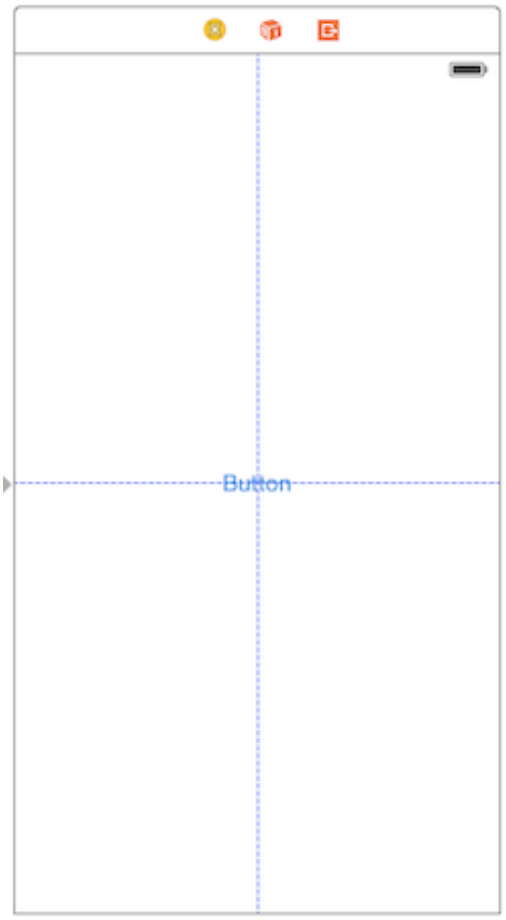



Figura 8.6: Autolayout Ex1

¿Que pasaría ahora si cambiamos las dimensiones de la pantalla?. Si usamos un dispositivo de mayor tamaño o rotamos la pantalla del dispositivo, las posiciones X e Y que habíamos seleccionado para centrar el botón ahora no representarían el centro de la pantalla quedando el botón en una posición incorrecta.

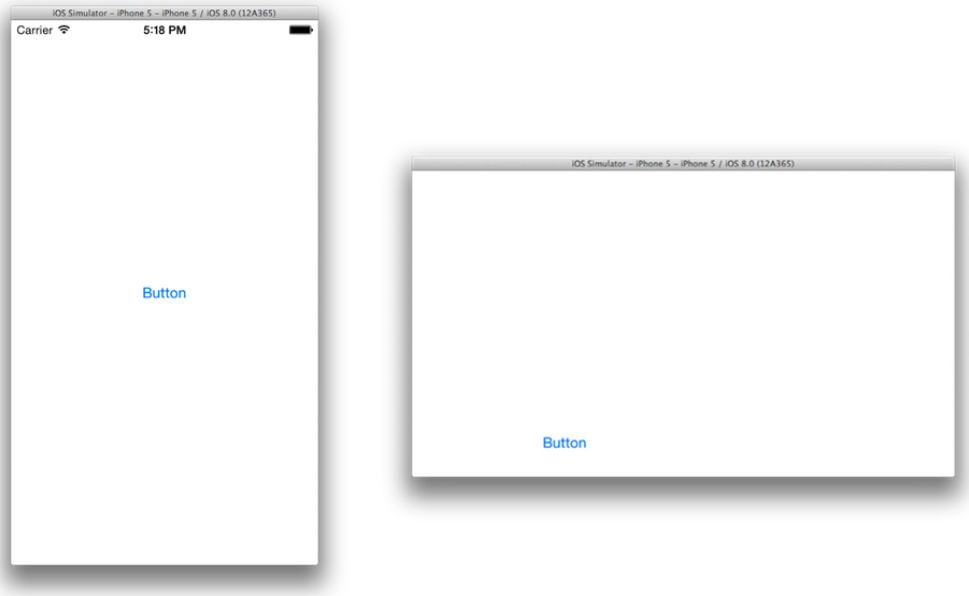



Figura 8.7: Autolayout Ex2

Para resolver esto Autolayout nos permite definir la posición del botón mediante 2 restricciones:

1. El centro del botón en el eje X tiene que ser igual al centro de la vista que lo contiene en el eje X
2. El centro del botón en el eje Y tiene que ser igual al centro de la vista que lo contiene en el eje Y

Definiendo estar 2 reglas el botón siempre estará centrado no importa el tamaño de la pantalla en la que se muestre.

## Constraints

Como hemos dicho Autolayout es un sistema basado en restricciones o constraints como normalmente se conocen. De forma general podemos dividir las restricciones en 2 tipos:

- **Alineación (align)**: Dentro de este tipo encontramos todas las constraints que nos permiten alinear elementos en la pantalla tanto de forma vertical como horizontal. Dentro de este tipo podemos encontrar las siguientes:
  - **Leading Edges:** Alinea el lado izquierdo de ambas vistas
  - **Trailing Edges:** Alinea el lado derecho de ambas vistas
  - **Top Edges:** Alinea el lado superior de ambas vistas
  - **Bottom Edges:** Alinea el lado inferior de ambas vistas
  - **Horizontal Centers:** Alinea las vistas en el eje vertical
  - **Vertical Centers:** Alinea las vistas en el eje horizontal
  - **Baselines:** Alinea las vistas por su línea base, es decir, si la vista contiene un texto alinea la vista para que el texto se encuentre en la misma linea. Si es una vista sin texto, equivale a alinear por el lado inferior.
  - **Horizontal Center in Container**: Alinea una vista en el eje vertical de la vista que la contiene
  - **Vertical Center in Container:** Alinea una vista en el eje horizontal de la vista que la contiene
- **Espaciado (pin)**: Dentro de este tipo encontramos todas las constraints que nos permiten definir tanto el tamaño de los elementos, como el espacio entre ellos y otros elementos en la pantalla.

Todas estas constraints sirven para definir la posición de un elemento en base a otro que ya se encuentra en la pantalla. Internamente estas constraints representan una ecuación (o una inecuación en ocasiones) que permiten calcular la posición exacta del elemento en la pantalla.

```
Item1.atributo1 = multiplicador * Item2.atributo1 + constante
```

En nuestro ejemplo la constante es 0 y el multiplicador 1 porque queremos centrar el componente en el contenedor, es decir

```
contenedor.centerX = componente.centerX
```

Podemos por ejemplo cambiar la constante por 50, con lo que conseguiremos que el componente esté desplazado 50 puntos a la izquierda de la posición de "centrado en X".

## 8.3. Gestión de la Memoria

Aunque cada vez contamos con dispositivos con mayores prestaciones, la gestión de la memoria en aplicaciones móviles sigue siendo particularmente importante. En iOS concretamente cuando una aplicación comienza a consumir demasiada memoria recibe alertas del sistema y si esta no responde liberando memoria corre el riesgo de que el sistema operativo la cierre.

### MRC

Para la gestión de memoria Cocoa hace uso de un contador de referencias. Cada objeto tiene un contador que indica el número de referencias que hay sobre él, de manera que cuando el contador de referencias llega a 0 el objeto es eliminado automáticamente por el sistema operativo.

Antes de iOS 5 el desarrollador debía hacerse cargo de hacer el RELEASE de los objetos que ya no fuese a utilizar y de los RETAIN de aquellos que quería que se conservasen (Las operaciones de release disminuyen en 1 el contador de referencias, las operaciones de retain lo aumentan en 1).

¿Y que ocurría si por cualquier motivo no estuviéramos realizando correctamente los RELEASE de dicho objeto?

Pues al no llegar dicho contador a cero, la memorial consumida por el objeto nunca se liberaba.

Esto en una aplicación pequeña podría no llegar a suponer un problema, pero en aplicaciones donde la carga de memoria sea superior podría acabar derivando en comportamientos inesperados de la app o incluso llegar al punto en que el sistema operativo decida cerrarla.

### ARC

A partir de IOS5 Apple introdujo el conteo automático de referencias (ARC).

ARC realizará por nosotros todas las operativas de RETAIN y RELEASE sin que tengamos que preocuparnos por ello.

La regla es realmente sencilla, ya no tenemos que hacer RETAIN de los objetos, sino que simplemente al establecer un puntero a un objeto, este implicara que el objeto se conserve. En el momento que liberemos dicho puntero, el objeto se liberará automáticamente si no existen otros punteros al mismo.

Un concepto que deberemos tener siempre en mente es el tipo de relación que queremos establecer entre el puntero y los objetos a los que apunta. Ésta podrá ser de dos tipos dependiendo de su naturaleza:

- **Strong**: un puntero de este tipo, implica que nuestra variable será la dueña del objeto al que apunta el puntero, con lo que aumentará el número de referencias al objeto y por tanto no se liberará el mismo mientras el puntero siga activo.
- **Weak**: es un puntero que no implica que nuestra variable sea dueña del objeto apuntado, esto significa que no aumentará el número de referencias del objeto. De este modo, si por cualquier motivo el objeto apuntado fuera liberado, nuestro puntero pasaría ser automáticamente NIL (esto evita que se produzcan errores por apuntar a direcciones de memoria inexistentes).

Para el desarrollo de esta aplicación hemos hecho uso de ARC confiando la gestión de la memoria al compilador.

## 8.4. Mecanismos de persistencia

Durante el desarrollo de la aplicación hemos necesitado asegurar la persistencia de ciertos datos en el dispositivo por ejemplo el identificador del estudiante una vez registrado. iOS nos ofrece una gran variedad de mecanismos de persistencia. A continuación describiremos algunos de ellos.

### User Defaults

El sistema de Users Defaults es una característica heredada de OS X. Inicialmente fue diseñado para almacenar características que el usuario configuraba en

la aplicación y que definían lo que conocemos como “Preferencias de usuario”. Aunque fue diseñado inicialmente para esto puede ser usado para almacenar cualquier tipo de dato siempre y cuando se cumpla el formato clave - valor.

User Defaults está formado por una colección de listas de tipo Clave - Valor existiendo una única lista para cada aplicación instalada en el dispositivo.

La principal razón por la que el sistema User Defaults es usado por los desarrolladores es la facilidad de uso del mismo. A continuación vemos un ejemplo de uso del mismo.

```
01  NSUserDefaults *userDefaults = [NSUserDefaults standardUserDefaults];
02
03  [userDefaults setBool:YES forKey:@"Key1"];
04  [userDefaults setInteger:123 forKey:@"Key2"];
05  [userDefaults setObject:@"Some Object" forKey:@"Key3"];
06
07  [userDefaults boolForKey:@"Key1"];
08  [userDefaults integerForKey:@"Key2"];
09  [userDefaults objectForKey:@"Key3"];
10
11  [userDefaults synchronize];
```

Figura 8.8: UserDefaults

Llamando a

```
[NSUserDefaults standardUserDefaults];
```

tenemos acceso al diccionario y podemos leer cualquier información que haya sido almacenada previamente si conocemos si clave o salvar cualquier tipo de valor asociándole una clave.

Si hacemos operaciones de escritura debemos sincronizar al final de las operaciones el diccionario para que la información quede guardada de forma permanente en el dispositivo.

```
[userDefaults synchronize];
```

## SQLite

SQLite es una base de datos SQL como MySQL o PostgreSQL. La principal diferencia de esta base de datos es que es portable, muy ligera y que puede funcionar sin un servidor de bases de datos, en otras palabras está embebida junto a la aplicación lo que hace que los accesos a la misma sean muy rápidos. Aplicaciones como iPhoto confían en SQLite para el almacenamiento de algunos de sus datos.

## CoreData

CoreData es la capa de ORM que nos proporciona Apple para que podamos trabajar con objetos relacionales a alto nivel. CoreData proporciona al desarrollador Modelos que posteriormente pueden ser serializados a XML, binario o incluso almacenados en una base de datos SQLite, permitiendo al desarrollador trabajar con objetos a alto nivel sin tener que preocuparse del formateo de los mismos para que sean guardados en el mecanismo de persistencia elegido.

Una gran característica que nos proporciona coredata, es que podemos definir los modelos de la aplicación y sus relaciones a través de una interfaz gráfica

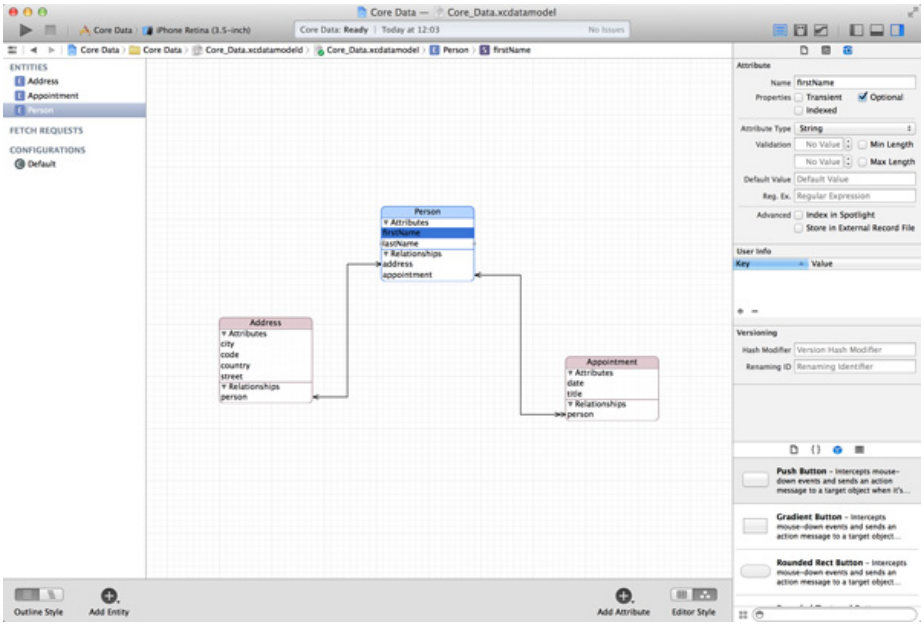



Figura 8.9: CoreData

CoreData es la opción recomendada si tu aplicación depende de una gran cantidad de objetos complejos que necesitan ser almacenados en el dispositivo.

Durante el desarrollo de esta aplicación con el uso de UsersDefaults ha sido suficiente para el almacenamiento de los datos que necesitábamos persistir dado que la mayoría de los datos quedaban salvados en el lado del servidor.

# Capítulo 9

# Pruebas

En todo proyecto software es necesario la realización de pruebas para asegurar la calidad del producto y verificar que se han cubierto todos los requisitos planteados en el análisis. En el presente capítulo se exponen, a modo de resumen, los tipos de prueba a los que se ha sometido la aplicación desarrollada a lo largo de este proyecto.

## 9.1. Pruebas de Caja Blanca

Las pruebas de caja blanca consisten en comprobar las funciones internas de un módulo con el fin de encontrar errores en su implementación. Estas pruebas se han ido realizando a lo largo del proceso de desarrollo, es decir, a medida que se ha ido finalizando la implementación de cada uno de los módulos de la aplicación.

A continuación se enumeran los distintos tipos de pruebas de caja blanca realizadas:

- Recorrido de todos los posibles caminos de ejecución de cada algoritmo para verificar que se cumplen todas las especificaciones.
- Comprobación de bucles mediante la verificación de los casos extremos (ningún elemento y máximo número de elementos) así como pruebas con iteraciones con un número aleatorio de repeticiones.
- Comprobación del resultado de operaciones aritméticas.

## 9.2. Pruebas de Caja Negra

En este tipo de pruebas no se tiene en cuenta el funcionamiento interno de cada módulo, sino que se somete el módulo a una serie de entradas y se evalúa su salida. Mediante este tipo de pruebas hemos testado todas los casos de uso que podría realizar un usuario, siendo las entradas al módulo las acciones realizadas por el usuario y la salida evaluada la respuesta de la aplicación.

## 9.3. Informes de Error

Para ayudar a la detección de errores pasados por alto, se cuenta también con el servicio Crittercism. Cada vez que se produce una excepción inesperada que provoca un cierre de la aplicación, ésta es capturada por Crittercism. La próxima vez que la app es iniciada, se envía automáticamente el informe de error al servidor, desde cuyo panel de control pueden consultarse la traza de la excepción.

## 9.4. Pruebas de Validación

Las pruebas de validación consisten en garantizar que la aplicación construida cumple con las especificaciones y requisitos definidos durante la fase de análisis, para ello:

- Se se ha revisado de nuevo cada requisito y se verificó que cumplía con todo lo establecido.
- Se ha enviado la aplicación finalizada a varias personas, quienes actuaron como testers.

# Capítulo 10

# Análisis de Costos

El coste de la aplicación en esta fase inicial es calculado teniendo en cuenta exclusivamente el coste del desarrollo de la misma, dado que aún no existen costes por mantenimiento o servidores.

Teniendo en cuenta el número de horas necesarias para la realización del proyecto y un coste fijo por horas se tiene:

- Coste total del proyecto en horas: 815h,
- Coste de la hora de desarrollo: 15€/h.

El coste total del proyecto asciende a doce mil doscientos veinticinco euros (12.225€).

# Capítulo 11

# Modelo de negocio

Esta aplicación se ha construido como un prototipo de herramienta para el autoaprendizaje y no está pensada para ser monetizada, aunque podría iterarse sobre ella para convertirla en un producto que genere algún tipo de beneficio en el futuro.

A continuación se describen algunos modelos de negocio que podrían desarrollarse alrededor de esta aplicación:

- **Venta de la aplicación**: Se podría generar contenido para un curso específico, por ejemplo, un curso sobre cómo hablar en público. Consecuentemente, se podría configurar la aplicación con los vídeos adecuados para el objetivo que se persiga , subirla al Apple Store y venderla al precio que se considere oportuno.
- **Ventas inApp:** Existe la posibilidad de integrar en la aplicación la compra de ciertos recursos que acompañen a las lecciones, como pueden ser explicaciones más extensas que las del vídeo, contenido avanzado una vez terminado el curso, etc. de manera que el usuario pudiese comprar este contenido a través de la propia aplicación consiguiendo con ello la monetización.
- **Plataforma de marketing:** También sería posible usar este producto como plataforma para impartir la introducción a contenidos y una vez finalizado el curso, ofrecer al usuario que asista a un taller presencial impartido por el autor de los vídeos. Estos talleres no serían gratuitos, de tal forma que el usuario se haría cargo del coste de los mismos en caso de estar interesado.

Esta aplicación no ha sido creada con un fin monetario pero ofrece toda la capacidad de rentabilización de una plataforma e-learning con el valor añadido de que se encuentra disponible en plataformas móviles.

# Capítulo 12

# Conclusiones

## 12.1. Conclusiones sobre el producto

Tras finalizar con la implementación del producto, contamos con una aplicación que cumple por completo con los objetivos marcados inicialmente:

- Se ha construido una plataforma que permite a un usuario, de forma autodidacta, recibir formación sobre un tema en particular y llevar a cabo un proceso de autoevaluación.
- En el lado del backend, se ha optado por una solución basada en Parse (herramienta provista por Facebook) que cumple con las expectativas iniciales, permitiendo la administración de las variables del curso así como la recolección de estadísticas para futuras iteraciones de la aplicación. Aunque esta solución cumple con las necesidades actuales del producto, cabe la opción de que en el futuro sea reemplazado por alguna tecnología con mayor potencialidad si se continúa trabajando sobre este producto .

## 12.2. Conclusiones personales

A nivel personal, inicié este proyecto con la expectativa de formarme como desarrollador iOS . Al final de este desarrollo, soy consciente que hay todo un universo de conocimiento por adquirir, pero este proyecto ha asentado la base sobre la que seguir edificando mi trayectoria profesional. De hecho, ya me encuentro embarcado en el siguiente proyecto iOS de manera profesional, continuando con él la senda que empezó este proyecto de fin de carrera..

## 12.3. Conclusiones finales

Podemos concluir que el resultado del proyecto ha sido satisfactorio: Se ha cumplido tanto el objetivo de la formación del estudiante en el desarrollo de plataformas iOS como la implementación del producto con los requisitos deseados.

# Capítulo 13

# Trabajos Futuros

La aplicación desarrollada es actualmente un versión inicial, con lo que existen múltiples características que podríamos añadir en el futuro para completarla cada vez más. A continuación se añaden algunos ejemplos de líneas futuras:

- **Reproducción de contenido en streaming**: Actualmente la aplicación gestiona su contenido audiovisual de forma local. Añadiendo la funcionalidad de reproducir vídeo en streaming los cursos podrían ser aún más flexibles (el contenido no está limitado al añadido inicialmente en la aplicación). Esto permitiría añadir nuevo contenido a los cursos, editarlo, o eliminarlo si se considera que alguna lección es innecesaria.
- **Implementación de un servicio de backend personalizado:** El backend de la aplicación funciona mediante Parse en esta primera fase. Parse es una herramienta potente pero limitada, de manera que contar con un servidor propio proporcionaría la flexibilidad necesaria para soportar nuevos casos de uso teniendo el control completo del sistema.
- **Implementación de la aplicación para dispositivos Android:** Esta es una aplicación nativa para dispositivos iOS, no obstante, en el futuro se podría generar la versión Android de la misma, cubriendo con ello la mayor parte de los dispositivos móviles disponibles en el mercado.
- **Hablar con el profesor:** a aplicación está pensada para que el curso sea impartido de forma autodidacta y no existe dentro de ella la posibilidad de recibir feedback o indicaciones del profesor del curso. En el futuro, podría añadirse una nueva funcionalidad que permita al alumno comunicarse con el profesor.

- **Añadir niveles a los cursos:** La aplicación ha sido creada para ser autocontenida y los cursos tienen la cantidad de lecciones incluidas dentro de la misma. Mediante el uso de streaming como sugeríamos en el primer punto o bien mediante la compra de contenido "inApp", podríamos dar al usuario la posibilidad de que ampliase el contenido del curso mediante la adquisición del nivel avanzado del curso que ha superado.

Éstos son sólo algunos de los ejemplos de nuevas características que se podrían añadir para aportar valor a los estudiantes. Además, al tratarse de una aplicación pensada para dispositivos móviles, tiene la particularidad de poder adaptarse a éstos para aprovechar todas las ventajas que nos ofrecen y aplicarlas al ámbito de la enseñanza.

# Capítulo 14

# Anexo: Manual de instalación y uso

En el CD adjunto a la memoria podrá encontrarse el código del proyecto, contenido en la carpeta «source code». Dentro de la carpeta platform se encuentra el fichero Xworkspace del proyecto.

## 14.1. Preparar el entorno

Este proyecto hace uso de cocoapods[Cocoapods] como gestor de dependencias, por lo que es necesario tener configurado el entorno para el uso del mismo. Los pasos para instalar cocoapods son los siguientes:

1. Desde un terminal o consola de comandos, navegar a la carpeta «source code»
2. Una vez situados en la raiz del proyecto ejecutar las siguientes instrucciones

```
$ cd platform/
$ gem install cocoapods
$ pod install
```

## 14.2. Compilación y uso de la aplicación

Una vez instalada cocoapods y todas las dependencias del proyecto, podemos abrir XCode para compilar y ejecutar el proyecto.

Para lanzar el proyecto en XCode tenemos 2 posibles alternativas:

1. Desde el propio terminal, situados en la carpeta «platform» tal como hicimos anteriormente, ejecutar el siguiente comando

```
$ open LPlatform.xcworkspace/
```

2. Navegar a traves de «finder» hasta la carpeta «platform» y abrir el archivo «LPlatform.xcworkspace»

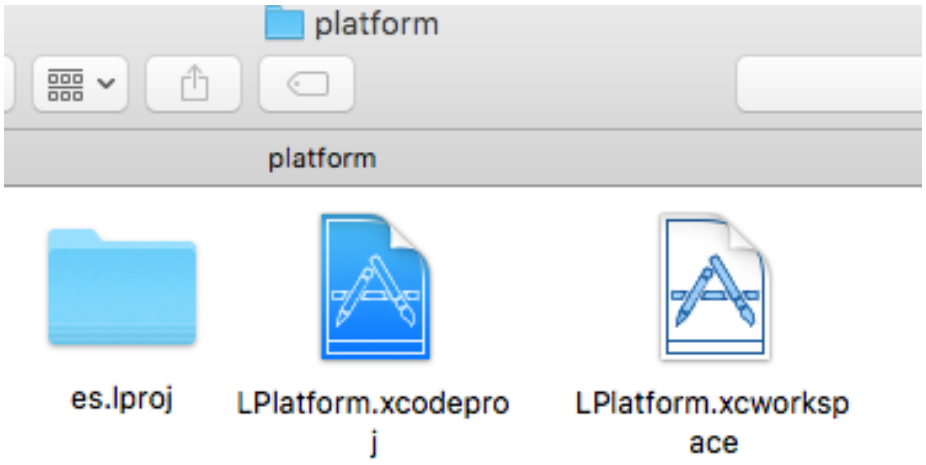



Figura 14.1: ejecutable

Una vez lanzado el proyecto se nos abrirá una ventana de XCode desde la que podremos ver toda la estructura de ficheros y la configuración del mismo.

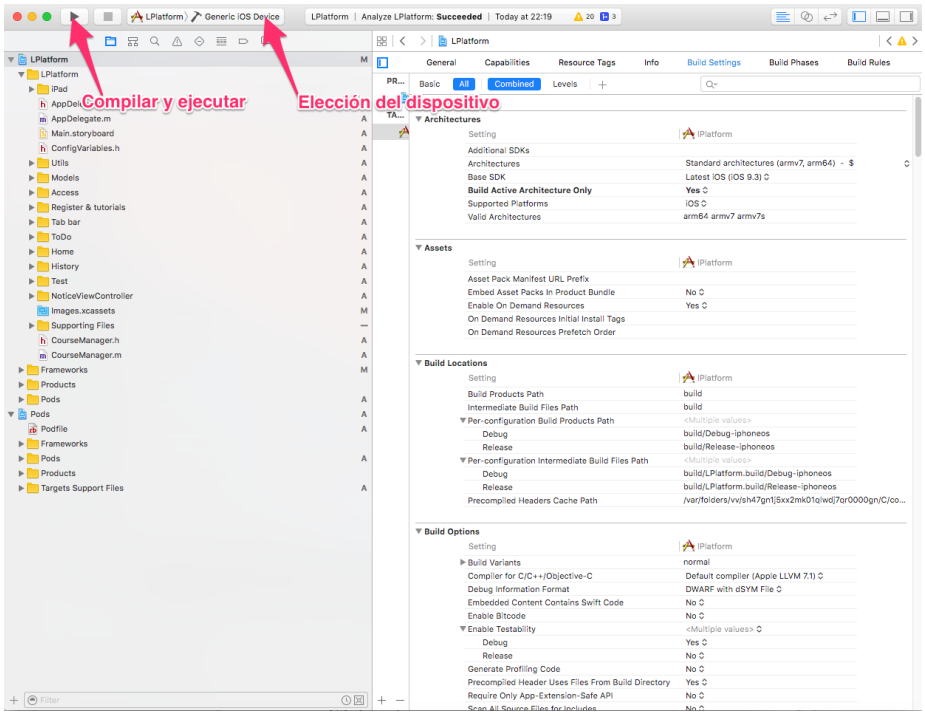



Figura 14.2: Configuración XCode

Para compilar y ejecutar el proyecto sólo es necesario activar el botón Run de la esquina superior izquierda que se observa en la figura 14.2.

## 14.3. Otras consideraciones

Si no se han configurado certificados de desarrollador de Apple con los que firmar la aplicación, ésta sólo podrá ejecutarse en el simulador. Si se desea ejecutar la aplicación en un dispositivo físico, es necesario seguir los siguientes pasos:

- Configurar los certificados de desarrollador de Apple a través de developers.apple.com/membercenter, descargarlos e instalarlos en el equipo. Una vez hecho esto, será necesario configurar XCode para usar los nuevos certificados.
- Conectar el dispositivo iOS en el que se desee al ordenador y esperar a que XCode lo detecte.
- Cambiar el target que se observa en la figura 114.2 para que apunte al dispositivo conectado. Ejecutar la aplicación (Cmd + R o botón Run).

# Bibliografía


Inc Apteligent. Apteligent: Get mobile app intelligence in under 15 minutes. URL `https://www.apteligent.com/`.

Atlassian. Bitbucket - repositorio gratuito de código. URL `https://bitbucket.org/`.

Tom Love Brad Cox. Objective-c.

Tim Cinel. Actionsheetpicker-3.0: Quickly reproduce the dropdown uipickerview / actionsheet functionality on ios. URL `https://github.com/skywinder/ActionSheetPicker-3.0`.

Cocoapods. Cocoapods. URL `https://cocoapods.org/`.

fournova. Tower - the most powerful git client for mac. URL `http://www.git-tower.com/`.

Apple Inc. Testflight beta testing. URL `https://developer.apple.com/testflight/`.

iTunes S.a.r.l. Xcode. URL `https://developer.apple.com/xcode/`.

Luis Jiménez de Asúa. *TRATADO DE DERECHO PENAL TOMO 7*. 1950.

Craig Larman. *Agile and iterative development: a manager's guide*. Addison-Wesley Professional, 2004.

Craig Larman. *Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development*. Pearson Education India, 2005.

Harlan D Mills. Incremental software development. *IBM Systems Journal*, 19 (4):415–420, 1980.

MKLab. Staruml - a sophisticated software modeler. URL `http://staruml.io/`.

Inc. Parse. The open source parse backend. setup your self-hosted parse api today. URL https://parse.com/.

K Schwaber and Mike Beedle. Agile software development with scrum. 2002.

Balsamiq Studios. Balsamiq mockups 3. URL https://balsamiq.com/products/mockups/.

Jeff Sutherland, Ken Schwaber, Co-creators Of Scrum, and Contact Jeff Sutherl. The scrum papers: Nuts, bolts, and origins of an agile process. 2007.

The LyX Team. Lyx – the document processor. URL https://www.lyx.org/.

Linus Torvalds. Git –local-branching-on-the-cheap. URL https://git-scm.com/.

# Índice de figuras