MEMORIA DEL PROYECTO

APLICACIÓN IOS PARA EL ESTUDIO STUDY ASSISTANT

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Septiembre de 2022

Autor:

Carmen Ramajo Alonso

Tutor:

José Rafael García-Bermejo Giner

Autorización para la presentación de este proyecto.

D. José Rafael García-Bermejo Giner, Profesor Titular del Departamento de Informática y Automática de la Universidad de Salamanca en el área de Lenguajes y Sistemas Informáticos

Certifica:

Que el documento “Aplicación iOS para el estudio” ha sido realizado por Dña. Carmen Ramajo Alonso, con DNI 21742077Q y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado en Ingeniería Informática de esta Universidad.

Y para que así conste, a todos los efectos oportunos, firma el presente documento.

En Salamanca, a 26 de agosto de 2022.

José Rafael García-Bermejo Giner
Dpto. de Informática y Automática
Universidad de Salamanca

Resumen

Study Assistant tiene como objetivo servir de asistente a los estudiantes para facilitar su organización y, en consecuencia, comprensión de contenidos, simplificando las tareas académicas. Para alcanzar este objetivo, a través de un mecanismo de metadatos y una base de datos con capacidades documentales organiza los contenidos académicos.

La aplicación se desarrolla de forma nativa para el sistema operativo iOS y dispositivos iPhone en el lenguaje de desarrollo Swift, siguiendo los principios de diseño establecidos por Apple.

Para funcionalidades de autenticación y gestión de bases de datos se integra en la aplicación la plataforma Firebase.

Por último, en cuanto a metodología de trabajo, el proyecto sigue la metodología ágil Scrum, que divide el desarrollo en un conjunto de etapas o “sprints”. Esta metodología se aplica de manera adaptada a las características concretas de este proyecto.

Palabras clave

Aplicación, asistente, estudio, metadatos, iOS, iPhone, Swift, Firebase, Scrum.

Abstract

Study Assistant has the main goal of serving as an assistant to students to ease their organization and, therefore, the understanding of contents and to simplify those tasks related to academic activity. To ensure this goal, it organizes the content through a metadata system and a database with documentary capacities.

The application its developed as a native application for the iOS operative system and iPhone devices in the programming language Swift, following Apple's design guidelines.

For authentication and database functionalities, the application integrates the Firebase platform.

Lastly, as for work methodology, this project uses the agile methodology Scrum, which splits the development process in stages or "sprints". This methodology its applied in an adapted way because of the project's particular features.

Keywords

Application, assistant, study, metadata, iOS, iPhone, Swift, Firebase, Scrum.

Tabla de contenidos

1. Introducción

En el año 1992 la multinacional estadounidense IBM presentó al mundo el primer dispositivo que podría clasificarse como “teléfono inteligente” o “smartphone”, que fue bautizado con el nombre de IBM Simon. Si bien es cierto que este aparato no seguía los estándares con los que distinguimos al Smartphone actual, contaba ya con numerosas cualidades presentes hoy en día en este tipo de dispositivos como sistema de mensajería SMS, agenda, correo electrónico, texto predictivo y por supuesto una pantalla táctil.

Tras la comercialización de IBM Simon, un par de años tras su presentación, otras compañías del sector tecnológico como Hawlett-Packard y Nokia lanzaron al mercado sus propias versiones de este nuevo modelo de aparato.

Más de una década después, Apple lanza en 2007 el iPhone. Nombrado por la revista Time como el “Invento del año”, el iPhone revolucionaría la telefonía móvil y sentaría las bases de diseño del Smartphone moderno, popularizando además estos dispositivos entre el público.

Veintiocho años más tarde y con cientos de modelos de Smartphone disponibles en el mercado nos encontramos en una actualidad en la que este dispositivo supone una herramienta indispensable en nuestro día a día. Desde soporte para tareas cotidianas, hasta ocio e incluso trabajo, el Smartphone está siempre presente la vida de la mitad de la población global. Según un reciente estudio de Strategy Analytics, corporación dedicada al estudio de mercado, casi cuatro billones de personas son propietarias de un Smartphone.

Estos dispositivos suponen un apoyo indiscutible en nuestro día a día; aprovechar sus capacidades e incorporarlos en aquellas tareas más costosas con el fin de facilitarlas o ahorrar tiempo resulta lo más sensato. De este modo, uno de los campos donde más útil puede resultar es en la formación académica.

En términos generales, el objetivo del proyecto que se expone en el presente documento es diseñar y desarrollar una aplicación móvil que sirva de apoyo al estudiante. La herramienta busca facilitar tareas relacionadas con el estudio de una manera sencilla, intuitiva y amena, aprovechando el extendido uso del Smartphone.

1.1. Estructura del documento

La presente memoria detalla el ciclo de vida de desarrollo de la aplicación móvil Study Assistant. La estructura de la memoria es como sigue:

- Capítulo 1: Introducción. Contiene una breve presentación en términos generales de la evolución de la situación tecnológica hasta llegar a la situación actual, así como el principal objetivo de este proyecto.

- Capítulo 2: Objetivos. Expone los diferentes objetivos funcionales, no funcionales y personales que se pretenden conseguir con el desarrollo del proyecto.

- Capítulo 3: Estado del arte. Ilustra la situación actual respecto a las aplicaciones existentes en el mercado relacionadas con el campo académico, comparando sus principales funcionalidades con las de la aplicación de este proyecto.

- Capítulo 4: Marco tecnológico. Introduce las herramientas software y hardware empleadas a lo largo del desarrollo del proyecto.

- Capítulo 5: Metodología de trabajo. Detalla el tipo de metodología que se seguirá para el diseño y desarrollo de la aplicación.

- Capítulo 6: Análisis. Contiene la captura inicial de los distintos requisitos de la aplicación y las tareas a realizar derivadas de estos, la estimación de los requisitos y la planificación temporal para completar su desarrollo. También define la arquitectura del sistema final.

- Capítulo 7: Diseño. Ilustra las decisiones tomadas en el diseño de las interfaces de usuario de la aplicación, así como el diseño de los esquemas de las bases de datos utilizadas.

- Capítulo 8: Desarrollo. Contiene una descripción detallada de las partes más destacables de la codificación de la aplicación.

- Capítulo 9: Pruebas. Expone el conjunto de pruebas realizadas para la verificación del correcto funcionamiento de la aplicación.

- Capítulo 10: Conclusiones y trabajo futuro. Menciona deducciones finales y posibles opciones para continuar trabajando en la aplicación en un futuro a modo de cierre.

2. Objetivos

El objetivo de este proyecto es el desarrollo de una aplicación iOS para dispositivos iPhone que sirva de apoyo al usuario en sus tareas como estudiante. El trabajo abarcará todas las fases presentes en el desarrollo de software con la intención de obtener en última instancia un producto final viable y de calidad.

Se distinguen tres tipos distintos de objetivos en este proyecto: objetivos funcionales, objetivos no funcionales y objetivos personales.

2.1. Objetivos funcionales

Sintetizan los servicios que la aplicación garantizará a la hora de interactuar con el usuario final, otros servicios o productos. Hacen referencia a la funcionalidad del sistema como tal. Se recogieron en primera instancia en el documento de propuesta del proyecto.

1. La aplicación dispondrá de un mecanismo de autenticación con el que gestionar el acceso de los usuarios a la misma.

2. Se articulará un sistema de base de datos remotas que almacenarán la información que se precise manipular.

3. La base de datos será dotada de capacidades documentales.

4. Se implantará un sistema de metadatos sobre la información almacenada en base de datos que permita la agrupación y filtración de esta.

5. Se implementará un mecanismo de almacenamiento de eventos, así como el de lanzamiento de notificaciones en base a los mismos.

2.2. Objetivos no funcionales

Hacen referencia a las propiedades intrínsecas de la aplicación. Contribuyen de manera esencial a la obtención de un producto final con calidad.

1. La aplicación deberá disponer de una interfaz sencilla e intuitiva que siga las guías de diseño de interfaz y experiencia de usuario marcadas para los dispositivos Apple. Deberá hacer uso de los elementos y reglas de diseño extendidos entre desarrolladores y usuarios iOS para garantizar un periodo de habituación mínimo a la aplicación por parte del usuario final.

2. La interfaz desarrollada será adaptativa. Con esta particularidad, la aplicación se podrá utilizar en cualquier dispositivo, siempre y cuando sea compatible con la versión iOS marcada para el despliegue, sin importar el tamaño de pantalla o resolución.

3. La aplicación será desarrollada en inglés y localizada posteriormente en español. Se dispondrá de dos versiones distintas en función del idioma usado en el dispositivo en el que sea ejecutada.

4. La codificación de la aplicación seguirá las convenciones establecidas para el desarrollo de aplicaciones iOS.

2.3. Objetivos personales

Recogen aquellas metas individuales y subjetivas que el alumno espera conseguir con la realización del proyecto y que sirven para el crecimiento y enriquecimiento profesional.

1. El alumno se formará en el desarrollo de aplicaciones utilizando el lenguaje de programación Swift.

2. El alumno se formará en el diseño y utilización de bases de datos no relacionales.

3. El alumno profundizará en las metodologías de trabajo ágiles.

4. El alumno aplicará, entre otros, conocimientos de fundamentos de ingeniería del software y fundamentos de interacción persona-ordenador adquiridos durante el desarrollo del grado, reforzando y profundizando en los mismos.

5. El alumno se adentrará en el campo de diseño y desarrollo de aplicaciones móviles.

3. Estado del arte

3.1. Situación actual

En España el total de hogares que cuentan con un dispositivo de telefonía móvil ascendía en el año 2020 a más del 99% [1], exponiendo así a estos tipos de dispositivos tecnológicos como los más usados por la población.

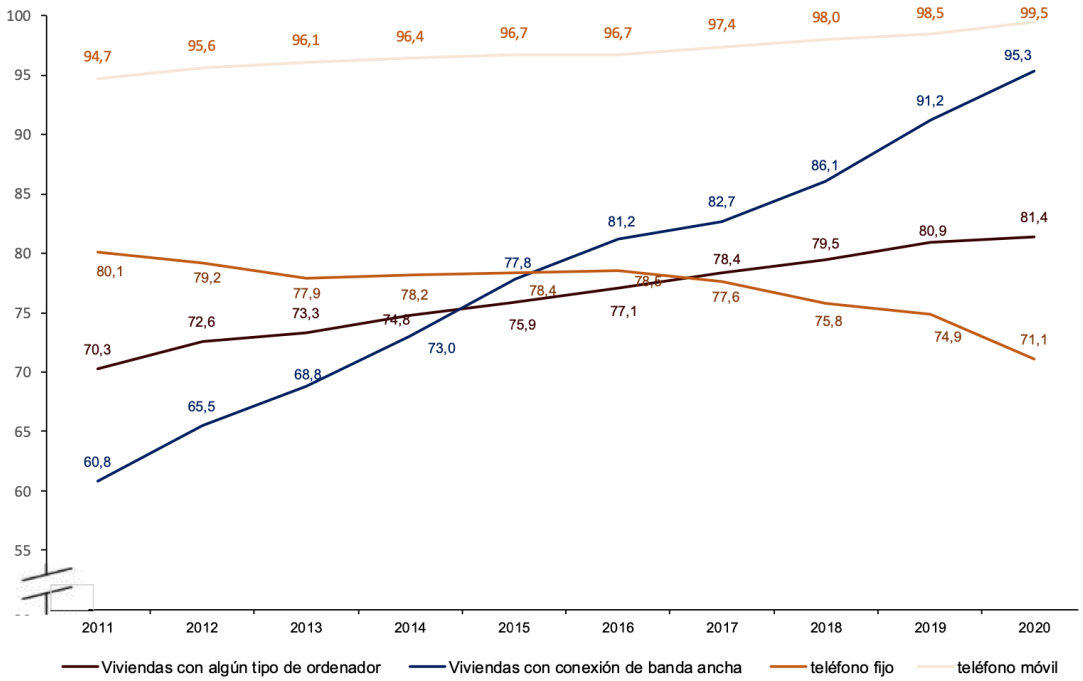


Ilustración 1 - Comparativa de los dispositivos tecnológicos y conexión en hogares españoles

Esta popularización del teléfono móvil se extiende de forma mundial (aunque no homogénea si tenemos en cuenta las regiones del planeta afectadas por la brecha digital [2]). En el año 2021 los usuarios de estos dispositivos le dieron un uso diario de casi 5 horas de media [3], lo que supone aproximadamente un tercio del tiempo total de las horas de vigilia de una persona promedio.

Cabe destacar además el creciente interés de los usuarios por los asistentes. En España en particular, la mitad de los usuarios utilizan un asistente de voz y más del 70% de esta cifra lo hace a través de su teléfono móvil [4].

Ilustración 2 - Porcentajes del uso de asistentes de voz en España

Las estadísticas avalan el acelerado proceso de digitalización y automatización que experimenta el mundo moderno. Los dispositivos tecnológicos, y en particular, el teléfono móvil, así como el uso de internet, juegan un papel fundamental en nuestro día a día. Su uso se extiende ya a prácticamente todos los ámbitos de nuestra vida como las comunicaciones, el ocio, la salud, o la educación.

En el ámbito de la educación en concreto esta digitalización se vio impulsada hace dos años para afrontar los desafíos que trajo consigo la pandemia provocada por la enfermedad por coronavirus (COVID-19). Entre las tendencias tecnológicas en educación y tecnología [5] destacan:

- E-learning. La educación ha distancia ha supuesto un especial reto por la urgencia con la que se ha necesitado implantar y la falta de planificación. Su futuro tiende a una modalidad híbrida en la que se combinen las clases a distancia con las presenciales.

- Aplicación de tecnologías inmersivas. Las técnicas de realidad virtual, aumentada y mixta permiten la creación de entornos virtuales controlados y enriquecidos con los que el alumno puede interactuar.

- Gamificación. Consiste en hacer llegar al ámbito educativo mecánicas utilizadas comúnmente en los juegos con la intención de mejorar los resultados del alumno. Un claro ejemplo de esta técnica es la plataforma Kahoot (https://kahoot.com), notablemente extendida en las aulas para la evaluación de los alumnos a través cuestionarios presentados como concursos o competiciones que fomentan la participación e implicación de los examinados.

- Redes sociales. Muchos docentes se inclinan a usar sistemas que se sirvan de las funcionalidades básicas de las redes sociales para permitir la interacción entre usuarios y que al mismo tiempo se enfoquen en el aprendizaje, de una manera más agradable que las ya establecidas aulas virtuales.

Con todo esto se hace evidente la importancia que llegan a tomar los dispositivos móviles en el ámbito pedagógico y como estos pueden ser considerados ya como una herramienta más tanto dentro como fuera de las aulas.

3.2. Aplicaciones para el estudio

A continuación, se presenta una serie de aplicaciones disponibles en la plataforma oficial de aplicaciones de Apple, App Store, enfocadas al apoyo en el estudio y sus tareas.

Se realizará además una comparativa con la aplicación desarrollada en este proyecto para localizar puntos en común y diferencias con las que poner en valor el objetivo de esta aplicación.

3.2.1. iStudiez Pro (iStudiez Team)

Aplicación para la organización de tareas académicas teniendo en cuenta horarios, exámenes y entregas. Permite almacenar información sobre periodos semestrales, asignaturas, vacaciones y profesorado.

Ofrece sus funcionalidades de manera gratuita pero también dispone de una versión de pago que ofrece además sincronización de datos en la nube multiplataforma [6].

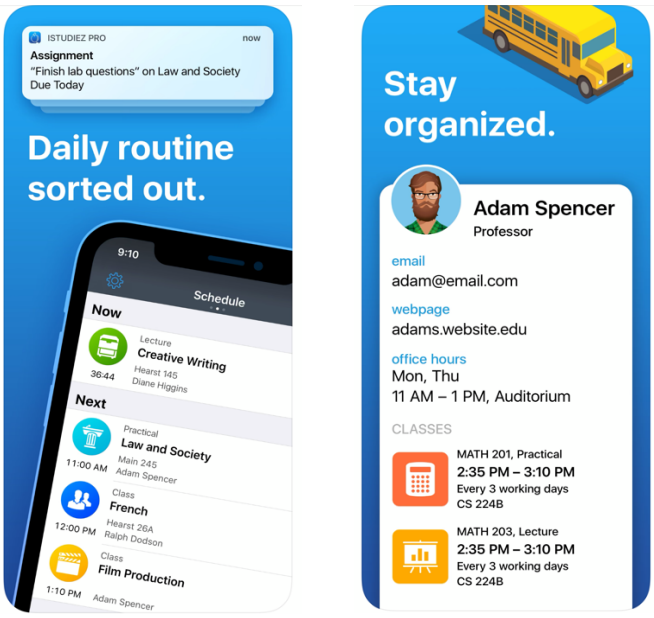


Ilustración 3 - Capturas de pantalla de la aplicación iStudiez Pro

3.2.2. My Study Life (My Study Life, Ltd.)

Herramienta planificadora para la organización y seguimiento de clases, tareas y exámenes. Proporciona también una serie de detallados calendarios que recogen las anteriores tareas y un sistema de recordatorios que alertan al usuario de eventos pendientes de completar [7].

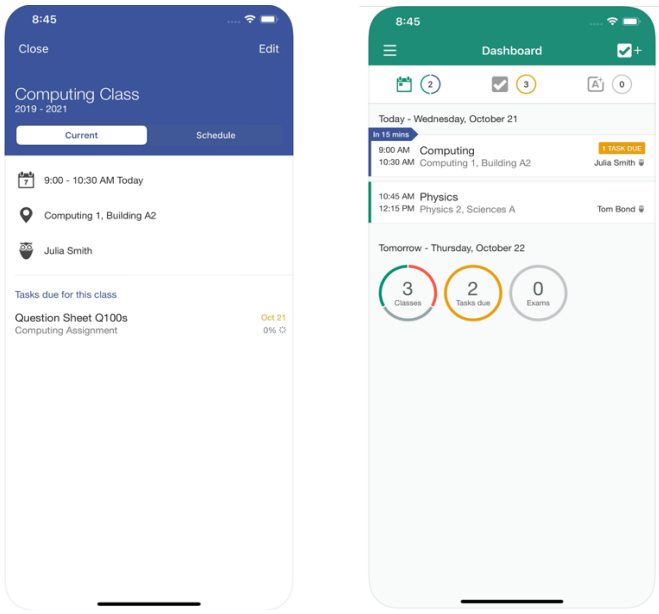


Ilustración 4 - Capturas de pantalla de la aplicación My Study Life

3.2.3. Easy Study (Aplicativos Legais)

Esta aplicación funciona principalmente como un planificador de rutina de estudio agregando detalles como asignaturas estudiadas, tiempo dedicado, colores para una mayor personalización y objetivos que el usuario quiera marcarse [8].

Ilustración 5 - Capturas de pantalla de la aplicación Easy Study

3.2.4. Study Smarter (StudySmarter UG)

Aplicación que se centra en la creación y gestión de material de estudio desde apuntes propios de universidades hasta tarjetas o notas generadas por los usuarios. Introduce además un sistema colaborativo para compartir el material [9].

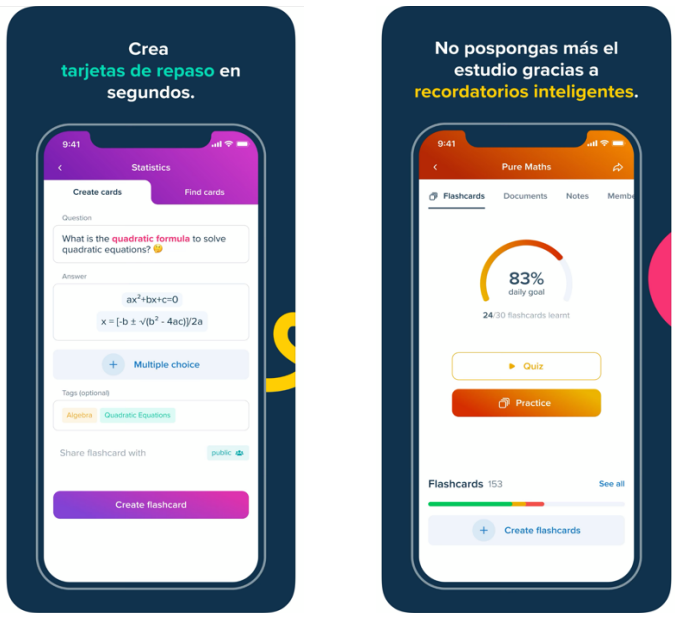


Ilustración 6 - Capturas de pantalla de la aplicación Study Smarter

3.2.5. The Homework App (The Homework App Ltd)

Aplicación que ofrece una serie de herramientas para la gestión de las tareas académicas. La versión gratuita presenta unas funcionalidades muy básicas. Dispone de un calendario de eventos y una agenda de profesores [10].

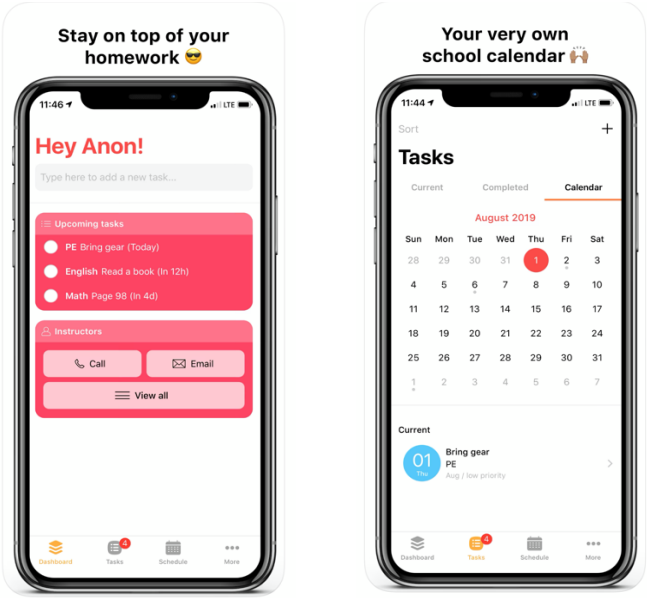


Ilustración 7 - The Homework App

3.2.6. Conclusiones

El análisis de estas aplicaciones permite comprobar como, aunque todas ellas tienen un objetivo similar, el apoyo al estudio del usuario, sus características básicas varían entre todas ellas. Mientras que unas se centran en la elaboración de calendarios otras tienen como principal herramienta la gestión de contenidos y materias, actuando de forma similar a una base de datos.

El objetivo de este proyecto es aunar las características básicas que se pueden encontrar en estas aplicaciones y similares: gestión de un calendario de eventos con notificaciones asociadas, sistema de organización de asignaturas y temas, creación y almacenamiento de documentación y apuntes y agenda de profesorado.

Además de lo anterior, se pretende enriquecer el contenido manejado por la aplicación con el uso de metadatos de manera que se relacione entre sí y permita ser filtrado por el usuario, funcionalidad que no ha sido encontrada en ninguna otra aplicación analizada en esta etapa de análisis de la situación de mercado.

A continuación, se detallarán y compararán aquellas características que las aplicaciones ofrecen en sus versiones totalmente gratuitas.

Funcionalidad	iStudiez Pro	My Study Life	Easy Study	Study Smarter	The Homework App	Study Assistant
Sistema de organización de asignaturas/temas	Solo de asignaturas	Solo de asignaturas	Sí	Sí	No	Sí
Gestión de eventos	Sí	Sí	Sí	Sí	Sí	Sí
Gestión de apuntes	No	No	No	Sí	No	Sí
Enriquecimiento de apuntes	No	No	No	No	No	Sí
Agenda de profesorado	Sí	No	No	No	Sí	Sí

4. Marco tecnológico

4.1. Sistemas operativos

Actualmente existen en el mercado varios sistemas operativos móviles disponibles, sin embargo, son solo dos los que lideran el mercado.

Según los resultados del informe aportado por el especialista en análisis web StatCounter [11], los sistemas operativos móviles más utilizados en el periodo comprendido desde julio de 2021 hasta julio de 2022 son Android e iOS.

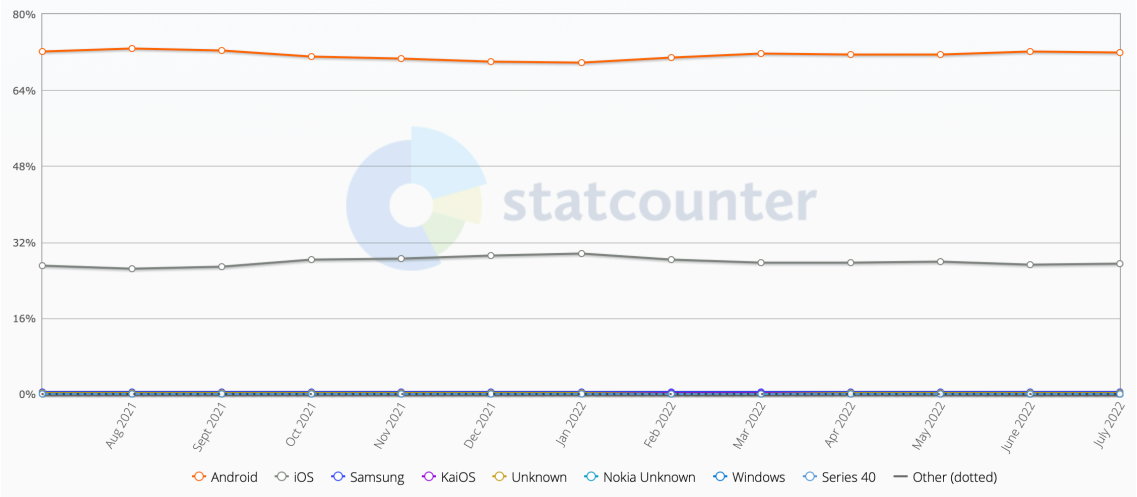


Ilustración 8 - Gráfico de los SSOO móviles más usados a nivel mundial

Estas estadísticas de mercado son el resultado de un estudio a nivel global, no obstante, si profundizamos en los datos podemos comprobar que este porcentaje de uso no es homogéneo si no que varía de una región a otra. Podemos cotejar este detalle a través del estudio de mercado realizado por Kantar [12], compañía dedicada al análisis de datos y mercado.

A continuación, se muestra la diferencia entre el mercado de España, donde la diferencia entre Android e iOS es de más de 60 puntos, y el mercado de los Estados Unidos de América, donde ambos sistemas operativos se encuentran prácticamente alrededor del mismo porcentaje.

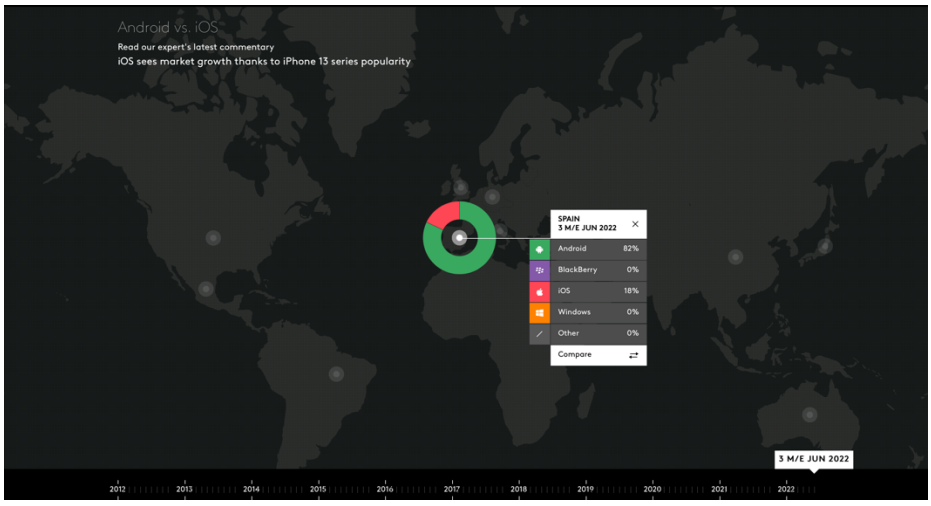


Ilustración 9 - Comparativa de los SSOO móviles más usados en España

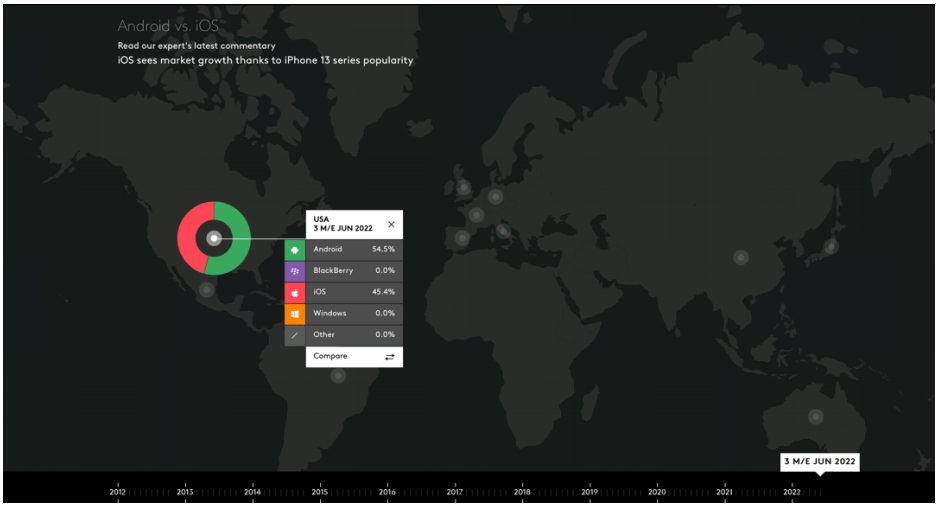


Ilustración 10 - Comparativa de los SSOO móviles más usados en EEUU

4.1.1. Android

En 2003 se funda la compañía Andoid Inc. En Palo Alto, California, donde se desarrolla el sistema operativo orientado a dispositivos móviles Android. En 2005, Google adquiere la compañía y un par de años más tarde presenta al público el sistema operativo al mismo tiempo que anuncia la constitución de la Open Handset Alliance, una alianza comercial para el progreso de estándares abiertos en dispositivos móviles [13].

El sistema operativo Android (https://www.android.com) está basado en el núcleo de Linux, un sistema de código abierto, libre y gratuito. Para dar lugar al sistema operativo móvil, a esta base se le agregan una serie de capas para otorgar unas funcionalidades específicas. Las capas añadidas son: las librerías, el entorno de ejecución que contenía entre otros la máquina virtual Dalvik (susutituida por el ART o Android Runtime a partir de la versión 4.4), el framework de aplicaciones y las aplicaciones propias del sistema. Esto es lo que se conoce como la arquitectura de Android [14].

Las aplicaciones dirigidas al sistema operativo Android se desarrollan mediante el kit de desarrollo Android, Android SDK y el uso del lenguaje Java. Estas se distribuyen a través de la plataforma Google Play Store (https://play.google.com/store/).

4.1.2. iOS

En 1976 es fundada Apple, la primera compañía de la historia en valer tres trillones de dólares [15], con el objetivo de llevar al mercado la computadora “Apple 1” desarrollada por uno de los fundadores originales, Steve Wozniak.

En las sucesivas décadas la compañía se dedica al desarrollo de ordenadores personales, hasta que, en el año 2007, Steve Jobs, otro de los fundadores originales, presenta en la Macworld Expo celebrada en San Francisco el primer teléfono móvil de la compañía, el iPhone [16].

De la mano del primer iPhone llega también el sistema operativo móvil de Apple iOS (https://www.apple.com/es/ios/) con su primera versión, iOS 1.0 [17]. Este sistema incluía aplicaciones originales, pero no soportaba aplicaciones de terceros de desarrolladores externos puesto que no ofrecía plataforma para su desarrollo ni distribución. La compañía estadounidense intentó promover el uso de aplicaciones web para su dispositivo móvil. Estas eran desarrolladas en el lenguaje de marcado HTML y ejecutadas desde el navegador del sistema Safari.

Con la llegada de iOS 2.0 aparece la tienda de aplicaciones oficial del sistema, App Store, y el kit de desarrollo para iOS, iOS SDK, que permitían finalmente el desarrollo y publicación de aplicaciones por parte de terceros.

4.1.3. Comparativa

Dados los datos recopilados es posible realizar una comparativa entre ambos sistemas operativos teniendo en cuenta dos grandes aspectos.

En primer lugar, cabe destacar la diferencia en cuanto al número de usuarios activos del que dispone cada sistema y como se reparten por el mundo. Como se ha expuesto anteriormente, el porcentaje de usuarios Android sobrepasa al de usuarios iOS y la diferencia en estas cifras varía con respecto a la región.

En segundo lugar, es necesario comparar los medios ofrecidos por ambos sistemas operativos para el desarrollo y en especial la publicación de aplicaciones de terceros para sus respectivos usuarios.

Una de las principales características de iOS y principal diferencia con los sistemas Android es que dispone de un ecosistema cerrado. En el caso del primero, la única forma de instalar una aplicación en el dispositivo es a través de la plataforma oficial App Store. Esto conlleva un aumento en el nivel de seguridad para los dispositivos iOS y mayor protección sobretodo ante software malicioso.

A esta diferencia de seguridad entre ambos sistemas se le suma una diferencia en cuanto a la calidad de sus aplicaciones. Mientras que Google Play requiere tan solo unas dos horas para la revisión de las aplicaciones que van a ser publicadas, la App Store necesita unos dos días completos para llevar a cabo estos controles de calidad. Este tiempo de revisión antes de entrar en la tienda para su distribución impacta directamente en la calidad de las aplicaciones ofertadas, así las disponibles en la App Store resultan de mayor calidad.

4.2. Tipos de aplicación

Las aplicaciones móviles se pueden agrupar en tres grandes tipos según las tecnologías utilizadas en su desarrollo y los usuarios a los que va dirigida: nativas, híbridas y web [18].

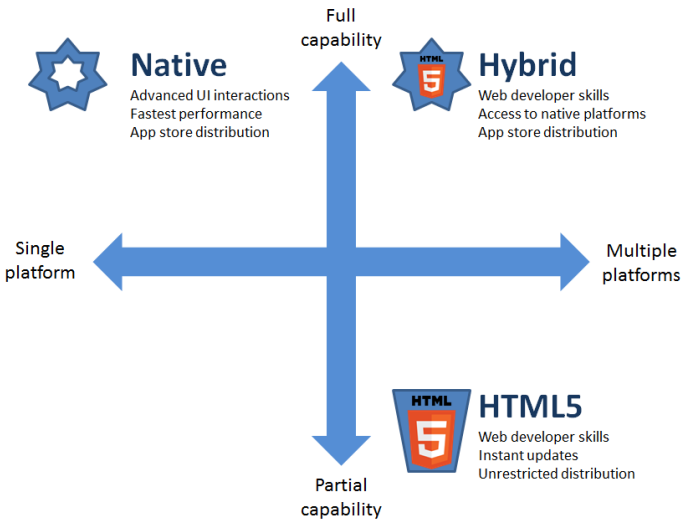


Ilustración 11 - Esquema comparativo de las aplicaciones web, nativas e híbridas

4.2.1. Aplicaciones nativas

Las aplicaciones nativas [19] son aquellas que se diseñan y desarrollan para funcionar en un sistema operativo específico utilizando marcos de trabajo, lenguajes y herramientas en particular para dicho sistema. De este modo, una aplicación nativa Android, por ejemplo, no podrá ser ejecutada en un dispositivo que utilice el sistema operativo iOS ni viceversa. En este caso, para lograr que esa aplicación funcione en ambos sistemas sería necesario desarrollar dos versiones, una para cada sistema, lo que se traduce en código no reutilizable y la elevación de los costes de desarrollo.

Este tipo de aplicaciones actúan de manera más rápida y flexible que los tipos alternativos garantizando un rendimiento máximo.

4.2.2. Aplicaciones híbridas

Por su parte, las aplicaciones híbridas [20] mezclan elementos de las aplicaciones nativas y web. Son diseñadas para funcionar en varios sistemas operativos, para lo cual utilizan lenguajes, librerías y componentes compatibles con los sistemas objetivo.

Estas aplicaciones proporcionan una gran flexibilidad a la hora de reutilizar código. De esta forma el diseño de una aplicación será el mismo para dos o más sistemas operativos, lo que reduce los costes de desarrollo y el tiempo de trabajo.

Al no seguir el lenguaje de diseño del sistema en el que se ejecutan su rendimiento se ve afectado y son más lentas que las aplicaciones nativas.

4.2.3. Aplicaciones web

Por último, las aplicaciones web [21] son sitios web a los que se puede acceder con un dispositivo de mesa como un ordenador personal o con un dispositivo móvil como un teléfono móvil.

El acceso a estas aplicaciones se hace a través del navegador instalado en el dispositivo desde el que se quiera ejecutar y no necesitan ser descargadas ni instaladas.

Este método resulta el más barato y sencillo a la hora de desarrollar una aplicación móvil, sin embargo, es el que peor rendimiento ofrecerá.

4.2.4. Comparativa

A continuación, se presenta una tabla comparativa que presenta las ventajas y desventajas más importantes de los tres tipos de aplicaciones.

	Aplicaciones nativas	Aplicaciones híbridas	Aplicaciones web
Rendimiento	Alto	Medio	Bajo
Coste (en términos económicos y de tiempo)	Alto	Medio	Bajo
Experiencia de usuario	Buena	Buena	Baja
Compatibilidad entre múltiples plataformas	No	Sí	Sí

4.2.5. Conclusiones

Teniendo en cuenta la información recopilada en las secciones anteriores (Sistemas operativos y Tipos de aplicación), así como las propias motivaciones personales del autor, se toma como primera decisión realizar el desarrollo de una aplicación móvil nativa para sistemas iOS.

Esta resolución combina, por un lado, la elección de uno de los sistemas operativos más presentes en el mercado de aplicaciones móviles, que ofrece en su tienda de aplicaciones los mayores estándares de seguridad y calidad; y por otro lado el tipo de aplicación con el mayor rendimiento posible y mejor experiencia de usuario.

4.3. Herramientas hardware

Para el desarrollo de este proyecto se han utilizado dos dispositivos hardware. En primer lugar, un ordenador portátil MacBook Pro con el sistema operativo macOS Big Sur. En segundo lugar, un teléfono móvil iPhone 12 con el sistema operativo iOS 14.8.

Al tratarse de una aplicación nativa dirigida a dispositivos iOS es necesario emplear hardware compatible en el desarrollo y al tratarse de un ecosistema cerrado como es este, dicho hardware debe pertenecer a Apple.

4.4. Herramientas software

4.4.1. Xcode

Xcode (https://developer.apple.com/xcode/) es el entorno de desarrollo integrado o IDE ofrecido para sistemas macOS. Incluye un conjunto de herramientas cuyo propósito es el desarrollo de aplicaciones software para los sistemas macOS, iOS, watchOS y tvOS.

Está disponible de manera gratuita para su descarga desde la App Store o desde la página para desarrolladores de Apple.

Incluye la herramienta Interface Builder, que permite a los desarrolladores el diseño de interfaces de usuario para aplicaciones. Utiliza un mecanismo visual, donde los elementos de la interfaz se disponen en las pantallas que utilizará la aplicación según las necesidades y requerimientos, sin ningún tipo de codificación.

Este entorno ofrece además una serie de simuladores para los diferentes dispositivos móviles en activo para poder realizar pruebas y tareas de depuración.

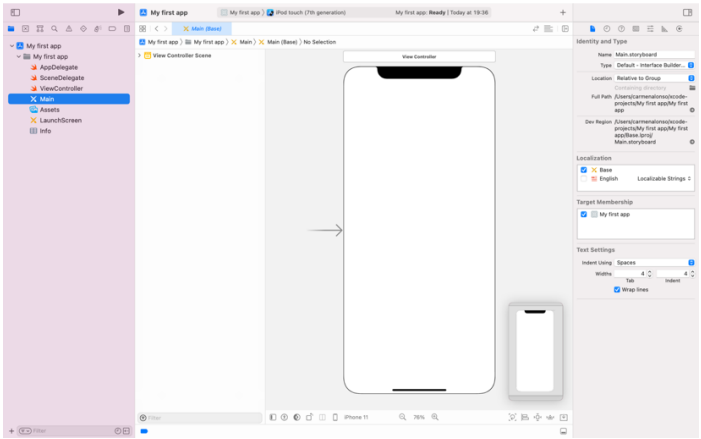


Ilustración 12 - Interfaz gráfica del software XCode

4.4.2. Swift

Swift (https://www.apple.com/es/swift/) es el lenguaje de programación creado por Apple que está dirigido al desarrollo de aplicaciones macOS, iOS, watchOS y tvOS.

Actualmente existen dos frameworks para el diseño de interfaces de usuario bajo este lenguaje: UIKit (https://developer.apple.com/documentation/uikit) y SwiftUI (https://developer.apple.com/documentation/swiftui/). El primero llegó con la presentación de Apple de su propio kit de desarrollo en 2008, mientras que el segundo aparece una década más tarde.

Ambos frameworks ponen a disposición de los desarrolladores de aplicaciones las clases necesarias para la creación de interfaces de usuario. Una de las mayores diferencias entre ambos es el tipo de programación que suponen, mientras que UIKit es un framework imperativo, SwiftUI es un framework declarativo.

Cada marco de trabajo presenta unas características particulares y con ellas una serie de ventajas y desventajas. Mientras que UIKit está plenamente establecido entre desarrolladores, el más reciente SwiftUI se perfila como el futuro del desarrollo de aplicaciones iOS.

En este proyecto se ha decidido hacer uso de UIKit por las siguientes razones:

- Documentación y soporte. Debido al tiempo que ha transcurrido desde su lanzamiento hace ya más de una década, UIKit cuenta con una documentación mucho más extensa y detallada.

- Establecimiento en el mercado. Este framework, como comentado anteriormente, está completamente asentado en el desarrollo iOS y a pesar de la aparición de SwiftUI seguirá utilizándose en el futuro cercano.

- Madurez. SwiftUI es un producto aun inmaduro, susceptible a cambios y que incluso necesita de UIKit para abordar determinadas tareas. Hay soluciones, triviales para UIKit, que actualmente no están disponibles en SwiftUI.

- Simplicidad. UIKit resulta intuitivo y sencillo gracias al uso de storyboards, vistas en las que se construye la interfaz de usuario con elementos visuales a los que se les otorga después funcionalidad a través de código enlazado a los mismos.

4.4.3. Cocoa Pods

CocoaPods (https://cocoapods.org) es un gestor de dependencias para proyectos Swift y Objective-C y uno de los más usados junto a Xcode. Esta herramienta nos ayuda a gestionar librerías de terceros, pudiendo así incluirlas y utilizarlas en nuestros propios proyectos.

4.4.4. Firebase

Firebase (https://firebase.google.com) es una plataforma ubicada en la nube que aparece en 2011 y es posteriormente adquirida por Google. Sobre ella se han ido añadiendo nuevas funcionalidades a lo largo del tiempo hasta contar con una completa y potente suite de utilidades.

Firebase pone a disposición de los desarrolladores un conjunto de herramientas para facilitar las tareas de creación y gestión de aplicaciones móviles, con el objetivo de simplificar el proceso de desarrollo y obtener un producto final de calidad. La herramienta es multiplataforma,

soportando así tanto Android como iOS entre otros y ofrece un plan gratuito lleno de posibilidades. Dispone además de una extensa documentación online y totalmente gratuita para facilitar su integración y uso.

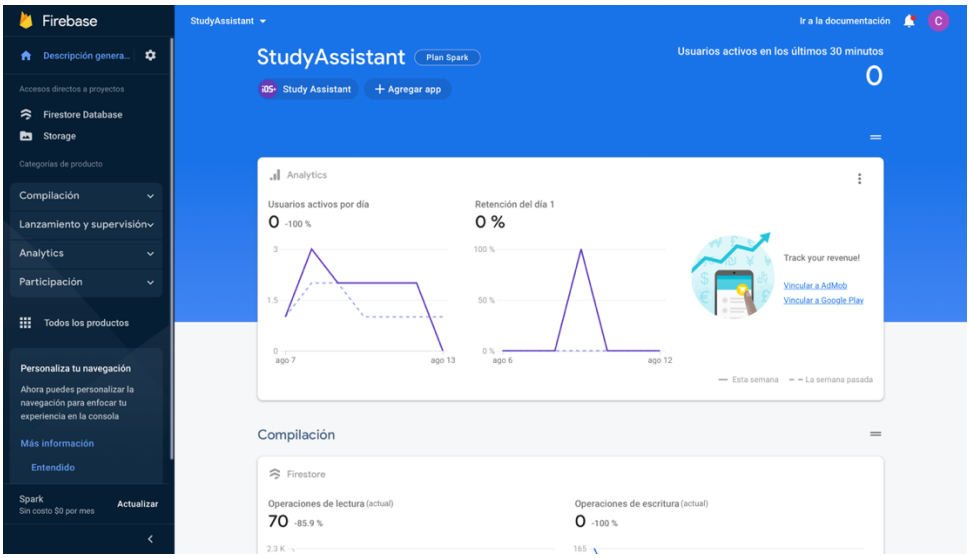


Ilustración 13 - Consola de Firebase

Entre los servicios que ofrece caben destacar aquellos que serán utilizados en el proyecto:

- Firebase Authentication. El servicio de autenticación de usuarios permite crear de manera sencilla un mecanismo de autenticación seguro y fácilmente integrable en la aplicación final. Permite la autenticación de usuarios por medio de proveedores de identidad de terceros como Google, Twitter y Facebook. El módulo se encarga tanto de la lógica del proceso de autenticación como de la gestión de la interfaz de usuario.

- Cloud Firestore. Este servicio consiste en una base de datos en tiempo real. Permite almacenar, sincronizar y consultar información a través de una base de datos no relacional organizada en forma de coleccione y documentos dispuestos en jerarquía. Permite también establecer un conjunto de reglas para determinar el acceso a la información que maneja en función de la identidad de los usuarios.

- Cloud Storage. Este servicio consiste en otra base de datos con la particularidad de que la información que manipula es contenido multimedia como fotografías y documentos. Como la anterior, Cloud Storage permite restringir el acceso según la identidad de los usuarios o las propiedades del objeto almacenado como su nombre o tamaño.

Las ventajas que aporta esta plataforma y que conducen a la decisión de integrarla en el actual proyecto son las siguientes:

- Compatibilidad. Como ya se ha expuesto anteriormente, Firebase ofrece soporte multiplataforma con lo que es apto para trabajar juntamente con iOS.

- Precio. Además de su versión completa de pago, la plataforma ofrece el Plan Spark, un plan gratuito con numerosas funcionalidades y dotado de considerable espacio de almacenamiento.

- Sencillez. Firebase ofrece potentes herramientas capaces de desempeñar funcionalidades complejas en tan solo unas líneas de código, disminuyendo el tiempo de desarrollo.

- Ausencia de servidor. Con la integración de Firebase no es necesaria la implementación de un servidor. Las funcionalidades backend que ofrece se ejecuta sin disponer de un servidor de por medio.

- Soporte y documentación. La plataforma ofrece de manera gratuita documentación completa en línea y además cuenta con una gran comunidad de usuarios desarrolladores, lo que facilita su uso.

4.4.5. Canva

Canva es una aplicación web que ofrece de manera online y gratuita un conjunto de herramientas de diseño gráfico. Dispone de numerosas plantillas para generar diseños y recursos como iconos e imágenes para incluir en nuestros diseños.

La herramienta se ha utilizado para el diseño del logotipo de la aplicación y esquemas complementarios a la información que se expone en algunas secciones.

4.4.6. draw.io

Esta herramienta online, recientemente renombrada como Diagrams.net, permite realizar y exportar diagramas de diversos tipos de manera sencilla.

4.4.7. Visual Paradigm

Esta herramienta UML CASE se ha empleado para la realización de los diagramas desarrollados en la etapa de análisis.

4.4.8. Microsoft Word

Este editor de texto se ha utilizado para el desarrollo y edición del presente documento.

5. Metodología de trabajo

5.1. Metodologías más extendidas

Las metodologías para el desarrollo de software se introducen para organizar, estructurar y llevar a cabo un control del proceso de desarrollo de un sistema o producto software de manera eficaz y productiva. Estas ofrecen un marco de trabajo con el que ponen a disposición de los desarrolladores un conjunto de técnicas y herramientas para el diseño de soluciones software.

En la actualidad, las metodologías de desarrollo software se pueden clasificar en dos grandes grupos. En primer lugar, se encuentran las llamadas metodologías tradicionales, que, como su nombre evoca, han sido utilizadas desde prácticamente los inicios del desarrollo software. Por otro lado, tenemos las metodologías ágiles, fuertemente consolidadas durante la última década.

Si bien es cierto que aun se siguen utilizando metodologías pertenecientes al grupo de las tradicionales, las metodologías ágiles se han popularizado e impuesto sobre el resto. Así lo refleja el estudio de Project Manager Institute (PMI), en el que se indica que hasta el 71% de empresas hacen uso de este tipo de metodologías [22].

Puesto que cada metodología tiene unas características particulares, será el proyecto sobre el que se vaya a aplicar e incluso la empresa los que decidan la metodología en concreto a emplear en cada caso.

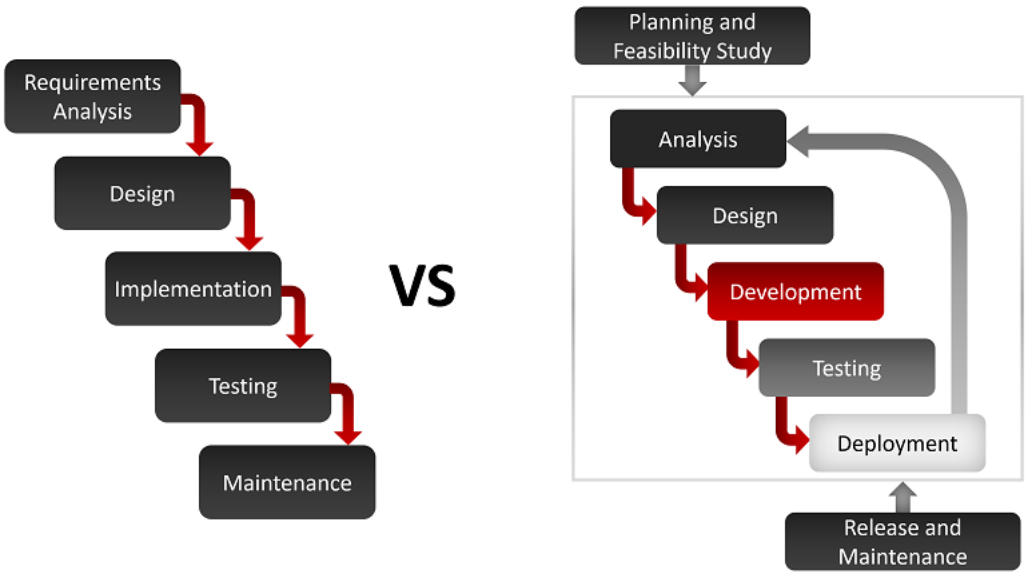


Ilustración 14 - Comparativa de los pasos marcados por metodologías tradicionales y ágiles

5.1.1. Metodologías tradicionales

Este tipo de metodologías siguen un proceso secuencial y en una sola dirección en el que se emplea un enfoque predictivo. En ellas la estimación realizada al inicio y solo al inicio del proyecto tendrá suma importancia pues de ella dependerán los recursos destinados al proyecto. Es también en el inicio donde se realiza, una única vez, la captura de requisitos, lo cual influirá en el curso del desarrollo también.

Esta estimación rígida y total realizada al principio del proyecto convierten a los ciclos de desarrollo de este tipo de metodologías en poco flexibles y con una mala adaptación a los cambios.

Algunas de las metodologías tradicionales más utilizadas son [23]:

- Cascada. Divide el desarrollo en etapas que se organizan en un esquema jerárquico de arriba hacia abajo, lo que le da el nombre. En cada una de estas etapas se llevan a cabo los desarrollos de determinadas funcionalidades del sistema software siguiendo un concreto orden establecido al inicio.

- Prototipado. Esta basado en la construcción de un prototipo de manera rápida basado en los requisitos del cliente para permitir que los usuarios puedan probarlo y, a través del feedback obtenido en dichas pruebas, solucionar errores e incluir nuevos requerimientos. Utiliza un modelo de prueba y error.

- Espiral. Combina las dos metodologías anteriores y añade además la gestión de riesgos. Las etapas que lo conforman son: planificación, análisis de riesgos, desarrollo del prototipo y pruebas.

- Incremental. Supone una de las metodologías más flexibles dentro de las tradicionales ya que permite utilizar el software antes de que sea completado pudiendo ver así resultados de manera más temprana. El producto final se construye de manera progresiva, completando iteraciones.

5.1.2. Metodologías ágiles

Las metodologías ágiles [24] pueden describirse como un conjunto de recursos enfocados a la gestión de proyectos que aportan flexibilidad y eficiencia. Con esto logran adaptar el sistema de trabajo a las condiciones variables del proyecto, lo que tiene como resultado no solo un aumento de la calidad del producto final sino también una disminución de los costes.

Se caracterizan por contribuir a un desarrollo flexible y evolutivo, aportar autonomía al equipo implicado y contribuir a disponer de una buena planificación, comunicación y capacidad de adaptación.

Debido a estas estas características están enfocadas a proyectos en los cuales las necesidades cambian a lo largo del tiempo y con ellas las soluciones que han de adoptarse para satisfacerlas.

Los tipos de metodologías ágiles más extendidos son [25]:

- Scrum. Fracciona el desarrollo en etapas de tiempo breve y fijo que reciben el nombre de sprints. El objetivo es obtener un producto software completo, usable, al final de cada una de estas etapas.

- Kanban. Esta metodología divide las tareas a realizar en el desarrollo en porciones mínimas de trabajo y las clasifica mediante un tablero en tres posibles tipos: pendientes, en curso y finalizadas. Con esto, se establece un flujo visual del trabajo.

- Programación extrema. Considera como clave del éxito las relaciones interpersonales de los participantes en el proyecto. El principal objetivo es crear un ambiente de trabajo óptimo y obtener una comunicación constante con el cliente. Esta metodología se basa en una serie de conceptos: diseño simple, refactorización, pruebas, estándares aplicados a la codificación, propiedad colectiva, programación por parejas, integración continua, entregas semanales, cliente in situ y entregas pequeñas.

5.2. Scrum

Scrum [26] es una metodología para la organización del desarrollo de productos complejos incluida dentro del grupo de las llamadas metodologías ágiles. Constituye un marco de proceso iterativo e incremental.

El framework Scrum tiene como base la realidad de que las demandas del cliente pueden variar a lo largo del desarrollo del proyecto y por tanto también lo hacen los requisitos de este, pasando de ser algo fijo acordado al inicio a algo completamente volátil. A partir de esta idea Scrum pretende dar un enfoque de trabajo centrado en la velocidad de producción, la capacidad de adaptación del equipo, el aprendizaje y las modificaciones imprevistas en los requisitos.

El término Scrum fue utilizado por primera vez en el año 1986 en un artículo titulado The New New Product Development Game [27], donde los autores, Hirotaka Takeuchi e Ikujiro Nonaka, profesores de la Universidad de Hitotsubashi, definieron este método dirigido a incrementar la velocidad y productividad de desarrollo. Para esto se utilizaría un equipo interdisciplinar autoorganizado y un proyecto desgranado en distintas fases que se solaparían entre sí.

5.2.1. Equipo

Scrum está orientado a equipos pequeños y busca una mejora continua en la eficiencia de trabajo y motivación de estos. Estos equipos están formados por tres roles con responsabilidades diferentes: el Product Owner, el Scrum Master y el Development Team.

El Product Owner representa al cliente y define las características que ha de reunir el producto y sus prioridades. El Scrum Master se encarga de verificar que el framework se utiliza de manera correcta, supervisando el adecuado cumplimiento de las normas y métodos que este ofrece. Y por último el Development Team, el equipo de desarrollo es el encargado de llevar a cabo las tareas necesarias para desarrollar el producto final.

5.2.2. Sprints

En este marco de trabajo el proyecto se divide en bloques de tiempo donde se establecen una serie de objetivos que han de ser alcanzados al final de estos. Cada uno de estos intervalos de tiempo fijo o iteraciones recibe el nombre de Sprint. La duración de un sprint puede estar comprendida entre dos y cuatro semanas como máximo.

El correcto desarrollo de estos intervalos se consigue realizando una serie de reuniones de forma periódica con las que se organiza el trabajo durante las semanas que comprenden cada sprint.

5.2.3. Elementos

Scrum utiliza los siguientes artefactos:

- Product Backlog: Se trata de una lista ordenada por prioridades con los requisitos o funcionalidades que solicita el cliente. Es creada por el cliente con ayuda del Scrum Master. Para cada elemento se incluye una descripción, el coste que se le estima, el valor que el cliente le da y la prioridad que tendrá en el proceso de desarrollo.

- Historias de usuario: Se corresponden con información que describe funcionalidades que estarán presentes en el producto. Han de ser cortas y claras. Resultan de la comunicación entre cliente y equipo.

- Sprint Backlog: Es una lista donde aparecerán aquellos ítems del Product Backlog que el equipo de desarrollo se compromete a llevar a cabo en el sprint actual. En ella se reflejan las personas responsables de cada tarea y el tiempo que queda para darlas por finalizadas. La lista se actualiza diariamente según los avances del equipo.

- Incremento: Constituye el resultado de un sprint. Es un artefacto totalmente completado y operativo.

5.2.4. Reuniones

Al comienzo de cada sprint tiene lugar una reunión de planificación donde a partir del Product Backlog se eligen las tareas a completar en ese sprint, así como el procedimiento a seguir para lograrlo, reflejándolo en el Sprint Backlog. Requiere un análisis detallado de las necesidades de cada tarea para poder realizar una estimación acertada de tiempos y costes.

Cada día se realiza una reunión de unos quince minutos donde los miembros del equipo de desarrollo exponen el trabajo realizado el día anterior, el trabajo que realizarán el día actual y los problemas que han encontrado o pueden encontrarse próximamente, todo esto con el objetivo de actualizar las tareas y evitar bloqueos. Esta reunión diaria recibe el nombre de Daily Sprint.

Cuando el sprint acaba se realiza una reunión llamada Review donde se analiza el trabajo llevado a cabo por el equipo a lo largo de esta iteración, es decir, el sprint completado se revisa. Ofrece un punto de partida para la próxima iteración.

En el momento en el que el sprint finaliza se realiza además otra reunión de retrospectiva donde lo que se analiza es la forma de trabajar del equipo. En base a la información extraída de esta actividad se busca mejorar la productividad de manera continua. Se detectan una serie de mejoras que se pretenderán cumplir en el siguiente sprint que tenga lugar.

5.2.5. Desarrollo

El proyecto comienza con una fase inicial de preparación que recibe el nombre de Sprint 0. Esta iteración tiene como objetivo comprender el proyecto en su totalidad de manera que se puedan tomar decisiones en su desarrollo.

Durante esta fase se han de definir los siguientes elementos:

- Proyecto: Es necesario entender el proyecto y sus objetivos, así como las necesidades que requiere el cliente.

- “Terminado”: Ha de definirse lo que conformaría el proyecto completamente terminado.

- Backlog inicial: Al principio del desarrollo será necesaria una lista inicial de requisitos con los que empezar a trabajar.

- Entregables: Objetos pequeños y funcionales que se generan como producto de la ejecución de los sprints y que permiten recibir un feedback por parte del cliente a lo largo del desarrollo.

En esta fase tienen lugar las primeras estimaciones, aunque por lo general se trata de estimaciones inexactas. El marco de trabajo permite invertir tiempo de estimación en otras tareas y dar así prioridad al desarrollo del producto.

Después de este primer sprint se planifican y ejecutan los siguientes necesarios con las reuniones oportunas para su correcto avance.

5.2.6. Ventajas

Algunas de las ventajas que supone la aplicación de Scrum al desarrollo software son:

- Tolerancia a cambios. El enfoque de este marco de trabajo permite adaptarse a los cambios en los requisitos que puedan tener lugar durante el desarrollo del producto, entendiendo que son necesarios en este proceso y forman parte de él.

- Resultados anticipados. Los entregables generados al final de los sprints permiten disponer de un producto funcional en etapas tempranas del desarrollo. Esto posibilita que el cliente valore el producto durante el desarrollo y en base a esta valoración el equipo de trabajo pueda llevar a cabo las correcciones necesarias si las hubiera a medida que avanza.

- Reducción de riesgos. Se minimiza el riesgo gracias a los entregables obtenidos durante el desarrollo y el feedback que generan estos por parte del cliente.

- Mejora de la relación entre cliente y desarrolladores. Las reuniones Scrum y la figura del Product Owner permiten una comunicación exitosa entre el equipo de desarrollo y el cliente. Esta relación garantiza un mínimo cumplimiento en las expectativas del cliente, así como una minimización respecto a documentación innecesaria y errores.

- Mejora del Retorno de la Inversión (ROI). Se minimiza el trabajo innecesario y se produce solo aquel software que aporta valor al negocio. Como consecuencia se obtiene un aumento en el ROI [28].

- Aumento de la calidad del producto. Se debe fundamentalmente dos aspectos: el trabajo iterativo e incremental que permite obtener feedback para la mejora del producto durante el mismo desarrollo a través de los entregables obtenidos al final de las iteraciones y las reuniones que tienen lugar a lo largo de cada sprint que proporcionan una visión clara del avance del proyecto, ayudando a identificar problemas, solucionarlos y adaptar los cambios de requisitos que surjan.

- Aumento de la productividad. Se aumenta la calidad también del propio trabajo del equipo al tratarse de equipos pequeños y auto organizados, características que aumentan la motivación y por consecuencia la productividad.

5.2.7. Conclusiones

Se ha elegido la metodología Scrum para el desarrollo de este proyecto teniendo en cuenta sus dimensiones, características y requerimientos y como estos son compatibles con las particularidades del marco, así como de las ventajas que este ofrece.

- Participantes en el proyecto. Durante el desarrollo de este proyecto solo estarán disponibles dos figuras de trabajo. Scrum está especialmente indicado para equipos pequeños.

- Prevención de riesgos. Con el objetivo de reducir riesgos y aumentar la calidad del producto final se seguirá la metodología Scrum ya que permite alcanzar estos objetivos mediante la generación de entregables. Estos objetos funcionales aportaran un feedback

a lo largo del desarrollo que posibilitará la mejora del producto final, la corrección de errores y la minimización de riesgos lo que contribuirá al éxito del proyecto.

- Soporte a cambios. Scrum permite adaptarse a cambios en los requisitos de manera rápida y satisfactoria. En el proyecto se otorgará prioridad al desarrollo en lugar de aspectos de planificación y/o documentación, el enfoque de la metodología Scrum beneficiará el proceso al ofrecer esta tolerancia a los cambios.

6. Diseño

6.1. Prototipado

Como punto de partida para el diseño de las interfaces que componen la aplicación se ha empleado la técnica de prototipado en papel.

Este método consiste en la creación de prototipos simples de baja fidelidad, lo que significa que presentan solo algunas de las características que estarán presentes en el producto final. Este tipo de prototipo es perfecto para realizar pruebas de conceptos y validaciones de ideas en una fase muy temprana del desarrollo. Además, permiten una gran flexibilidad y velocidad a la hora de su creación.

El prototipado en papel es la opción más económica que se puede encontrar de prototipado. Las herramientas que requiere son materiales muy básicos como papel, lápiz y tijeras.

A continuación, se presentan los prototipos de interfaz de usuario para cada una de las pantallas de la aplicación.

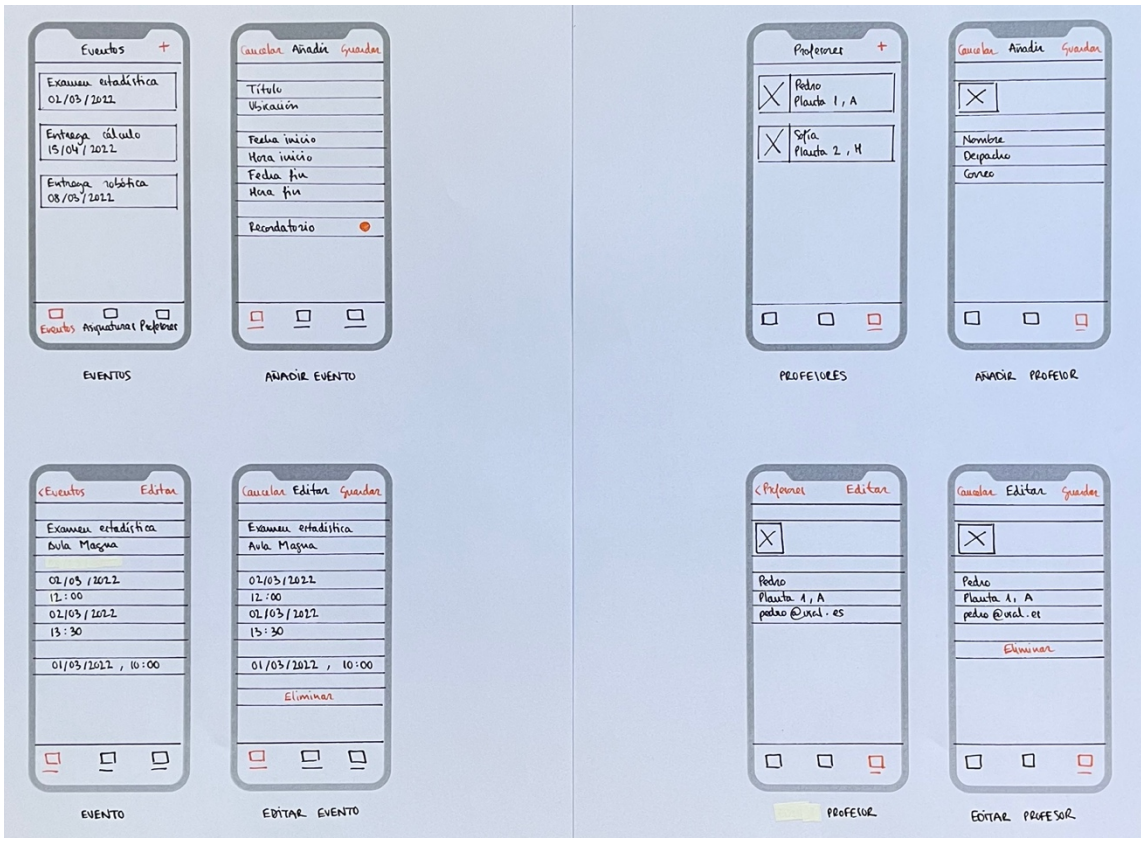


Ilustración 15 - Prototipos en papel de las pantallas de los módulos Evento y Profesores

Ilustración 16 - Prototipos de las pantallas del módulo contenido

El flujo de navegación entre pantallas es el que se muestra en el siguiente esquema:

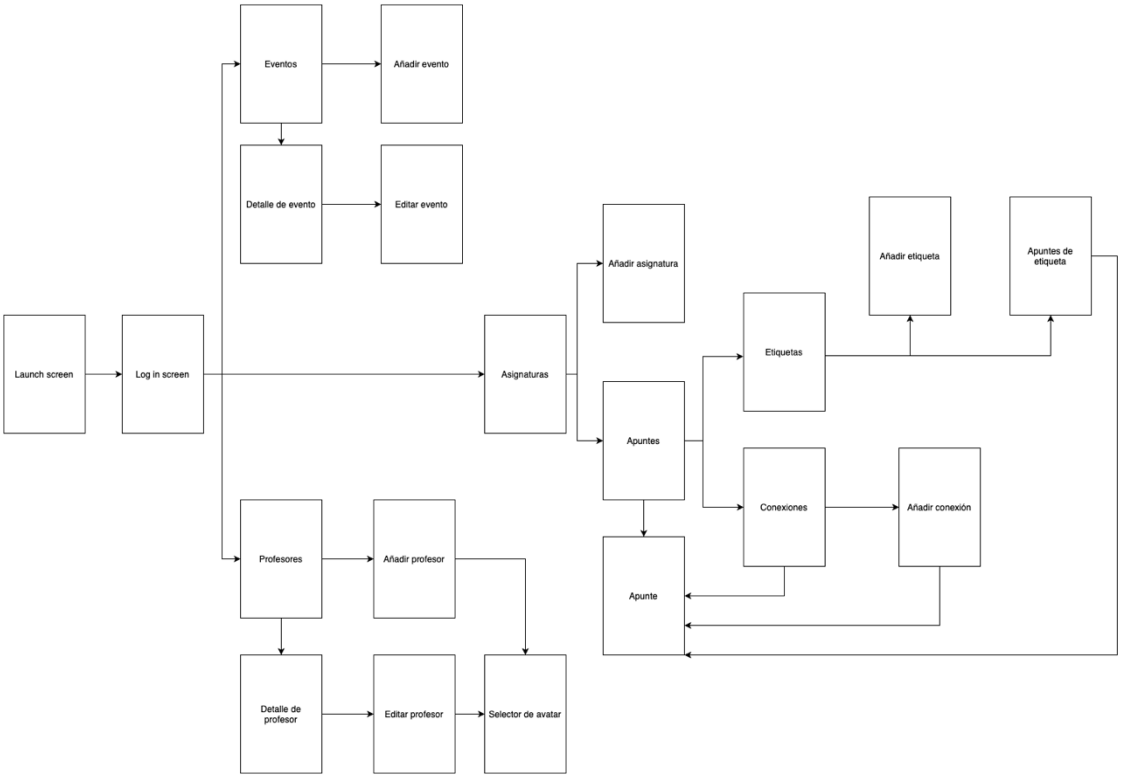


Ilustración 15 - Flujo de navegación de pantallas

6.2. Interfaz de usuario

Para el diseño de la interfaz de la totalidad de esta aplicación se han seguido los principios de diseño que marca Apple según la documentación que pone a disposición pública bajo el nombre de Human Interface Guidelines (HIG) [29].

Esta documentación consiste en una recopilación de recursos para el diseño y desarrollo de aplicaciones destinadas a dispositivos Apple. Ofrece una guía de buenas prácticas para la construcción de interfaces a nivel de diseño, experiencia de usuario y usabilidad contemplando desde las bases de diseño que la compañía adoptó en sus inicios hasta las últimas actualizaciones en tendencias, dispositivos y tecnologías.

Estas prácticas son imprescindibles para conseguir una interfaz que se muestre acorde con la presentación de productos Apple. Estos tienen como pilares de diseño la limpieza, la claridad y la intuición. De este modo, las aplicaciones diseñadas para estos dispositivos han de conservar estas bases en su diseño.

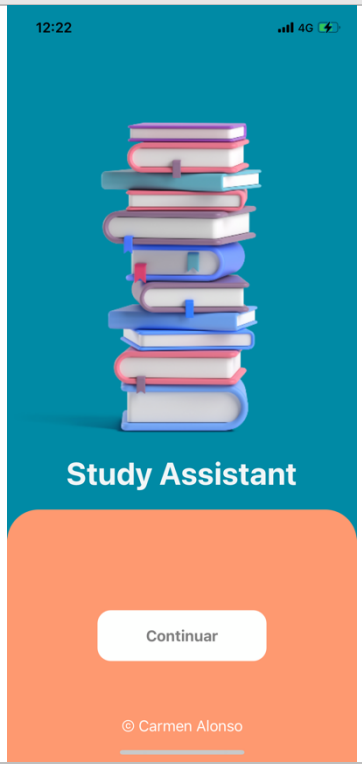
Antes de ilustrar las especificaciones de la interfaz de cada pantalla, se expondrán las decisiones de diseño generales más importantes:

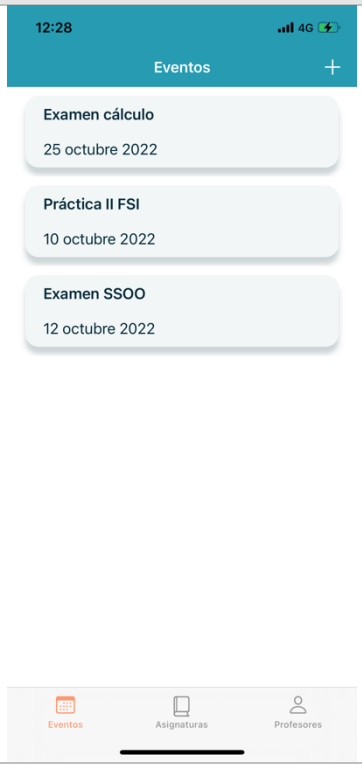
- Pantalla de lanzamiento. Esta pantalla, también nombrada como “launch screen” es la pantalla que aparece en primera instancia al abrir una aplicación, antes incluso de la primera pantalla o pantalla principal. Su función es responder de manera rápida, eficiente y sin costes (tiempo) al usuario para que este perciba con la mayor inmediatez posible que la aplicación ha sido lanzada para ser usada. En dispositivos Apple solo las plataformas iOS, iPadOS y tvOS requieren de manera obligatoria esta pantalla. Teniendo en cuenta lo anterior, la pantalla de lanzamiento ha de tener un diseño similar a la pantalla principal para que el usuario experimente una transición fluida entre ambas, debe evitar mostrar texto, debe contar con un diseño simple que no consuma recursos innecesarios y ha de evitarse su uso para promocionar la marca o compañía que desarrolla la aplicación u otras. Es por todo lo anterior que la pantalla de lanzamiento se ha diseñado con un color sólido igual al que aparecerá de forma predominante en la siguiente pantalla, la principal, sin mensajes, iconos o contenido multimedia de ningún tipo.

- Tablas. Se recomienda presentar la información en forma de listas o tablas ya que facilitan la lectura de sus contenidos. Por este motivo el uso de tablas se hace recurrente a lo largo de la aplicación. En el caso de que la información a presentar sea contenido multimedia como imágenes en lugar de texto lo recomendable es hacer uso de colecciones. Por este motivo se utiliza este tipo de objeto para mostrar el conjunto de avatares posibles para asociar al contacto de un profesor.

- Barra de navegación. La barra de navegación que aparece en la parte superior de la pantalla es un elemento fundamental para facilitar al usuario el uso de la aplicación. De tener un título debe representar claramente la pantalla a la que hace referencia. Debe conservar el botón de retroceso estándar iOS y el texto que aparezca en ella debe tener el suficiente espacio.

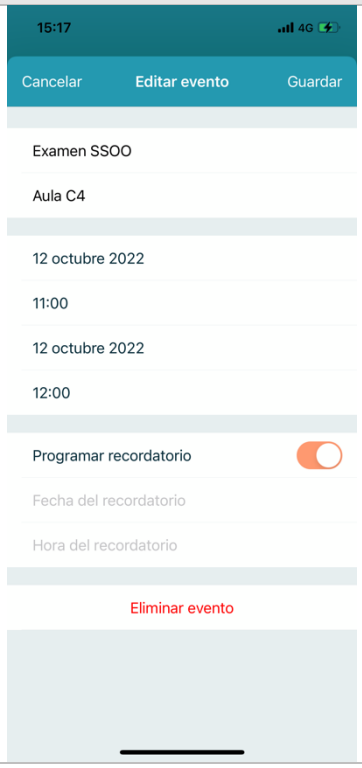
- Tab bar. El tab bar es la barra de navegación de la parte inferior de la pantalla que contiene una serie de ítems para permitir la navegación entre distintas partes o secciones de una misma aplicación. Este elemento debe ser siempre visible, a diferencia de la barra de navegación superior que puede desaparecer en momentos puntuales para conseguir una experiencia más inmersiva si es necesario. Debe utilizar el menor número de ítems posible y sustantivos o verbos concisos para los títulos de estos.

Sign In	
Interfaz	
Descripción	Es la primera pantalla tras la pantalla de lanzamiento o “launch screen”. Permite al usuario identificarse a través del servicio de autenticación proporcionado por Google utilizando Firebase.
Se presenta cuando	El usuario abre la aplicación.
Acciones	○ Pulsación sobre botón “Sign in”/“Continuar”

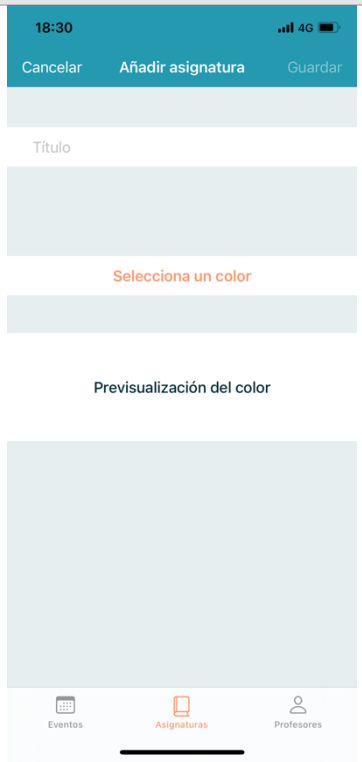
Eventos	
Interfaz	
Descripción	Muestra la lista de eventos registrados por el usuario en forma de tabla.
Se presenta cuando	El usuario se identifica correctamente o pulsa el icono “Eventos” de la barra de navegación inferior.
Acciones	○ Scroll sobre la tabla de eventos. ○ Pulsación sobre un evento. ○ Pulsación sobre botón “+”. ○ Gesto “swipe” para eliminar un evento.

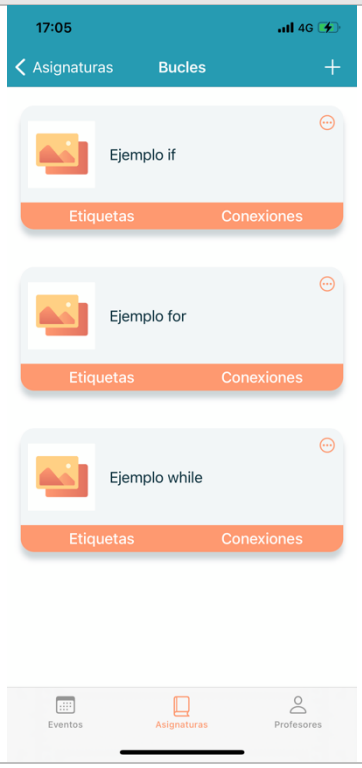
Añadir evento	
Interfaz	
Descripción	Proporciona un formulario con los campos necesarios para crear un nuevo evento.
Se presenta cuando	El usuario pulsa sobre el botón "+" de la barra de navegación superior en la pantalla Eventos.
Acciones	○ Pulsación sobre botón "Cancelar". ○ Pulsación sobre botón "Guardar". ○ Gesto "dismiss" para descartar la pantalla. ○ Pulsación sobre campos "Título" y "Ubicación" para introducir texto. ○ Pulsación sobre campos "Fecha de inicio", "Hora de inicio", "Fecha de fin", "Hora de fin", "Fecha de recordatorio" y "Hora de recordatorio" para mostrar sus respectivos Date picker. ○ Pulsación en swich "programar recordatorio" para activar/desactivar los campos correspondientes al recordatorio. ○ Validación del campo "Título".

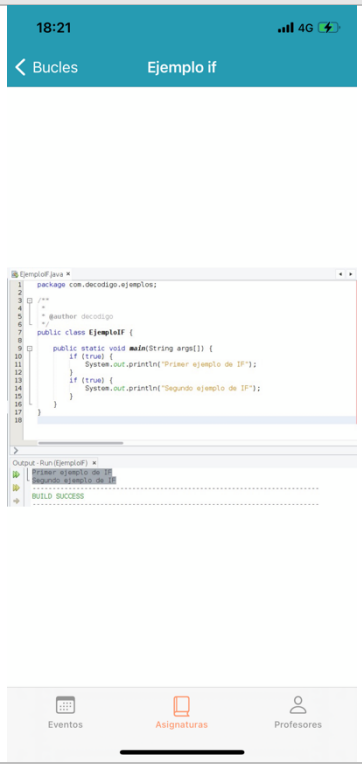
Detalle de evento	
Interfaz	
Descripción	Muestra la información asociada a un evento registrado por el usuario en forma de tabla.
Se presenta cuando	El usuario pulsa sobre un evento en particular en la pantalla Eventos.
Acciones	○ Pulsación sobre botón de retroceso “< Eventos”. ○ Pulsación sobre botón “Editar”.

Editar evento	
Interfaz	
Descripción	Proporciona un formulario con los campos necesarios para editar un evento, así como la opción de eliminación de evento a través del botón “Eliminar evento”.
Se presenta cuando	El usuario pulsa sobre el botón “Editar” de la barra de navegación superior en la pantalla Detalle de evento.
Acciones	○ Pulsación sobre botón “Cancelar”. ○ Pulsación sobre botón “Guardar”. ○ Gesto “dismiss” para descartar la pantalla. ○ Pulsación sobre campos “Título” y “Ubicación” para introducir texto. ○ Pulsación sobre campos “Fecha de inicio”, “Hora de inicio”, “Fecha de fin”, “Hora de fin”, “Fecha de recordatorio” y “Hora de recordatorio” para mostrar sus respectivos Date picker. ○ Pulsación en swich “programar recordatorio” para activar/desactivar los campos correspondientes al recordatorio. ○ Validación del campo “Título”. ○ Pulsación sobre botón “Eliminar evento”.

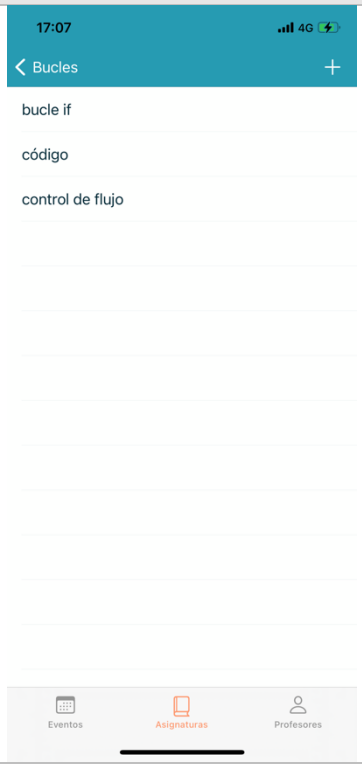
Asignaturas	
Interfaz	
Descripción	Muestra la lista de asignaturas y temas contenidos en estas registradas por el usuario en forma de tabla.
Se presenta cuando	El usuario pulsa sobre el icono “Asignaturas” de la barra de navegación inferior.
Acciones	○ Scroll sobre la tabla de asignaturas. ○ Pulsación sobre una asignatura para mostrar/ocultar sus temas. ○ Pulsación sobre un tema. ○ Pulsación sobre botón “opciones” de una asignatura. ○ Gesto “swipe” para eliminar un tema. ○ Gesto “swipe” para editar un tema. ○ Pulsación sobre botón “+”. Gesto “swipe” para eliminar un evento.

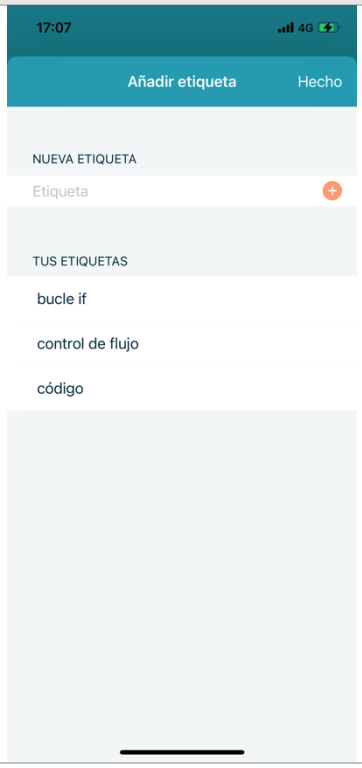
Añadir asignatura	
Interfaz	
Descripción	Proporciona un formulario con los campos necesarios para crear una nueva asignatura.
Se presenta cuando	El usuario pulsa sobre el botón “+” de la barra de navegación superior en la pantalla Asignaturas.
Acciones	○ Pulsación sobre botón “Cancelar”. ○ Pulsación sobre botón “Guardar”. ○ Pulsación sobre campo “Título” para introducir texto. ○ Pulsación sobre botón “Selecciona un color”. ○ Validación del campo “Título”.

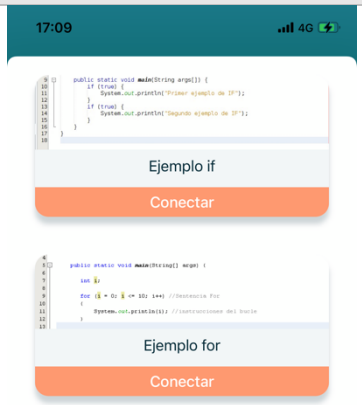
Apuntes	
Interfaz	
Descripción	Muestra la lista de apuntes de una asignatura registrados por el usuario en forma de tabla.
Se presenta cuando	El usuario pulsa sobre una asignatura en particular en la pantalla Asignaturas.
Acciones	○ Scroll sobre la tabla de apuntes. ○ Pulsación sobre un apunte. ○ Pulsación sobre botón “opciones” de un apunte. ○ Pulsación sobre botón “Etiqueta” de un apunte. ○ Pulsación sobre botón “Conexiones” de un apunte. ○ Gesto “swipe” para eliminar un apunte. ○ Pulsación sobre botón “+”. ○ Pulsación sobre botón de retroceso “< Asignaturas”.

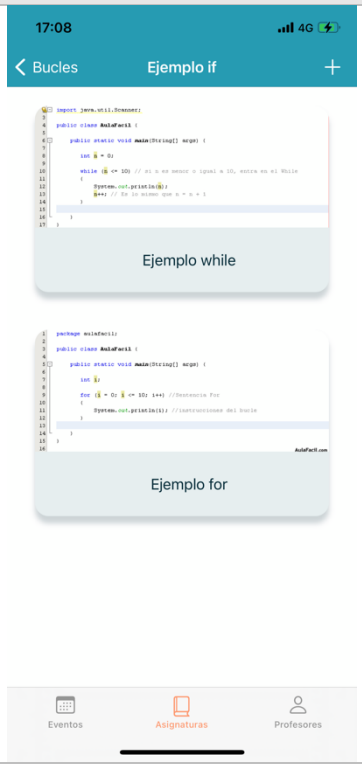
Visor de apuntes (tipo imagen)	
Interfaz	
Descripción	Muestra la imagen asociada a un apunte.
Se presenta cuando	El usuario pulsa sobre un apunte en particular en la pantalla Apuntes, Conexiones, Apuntes de etiqueta o Añadir conexión y este resulta ser de tipo imagen.
Acciones	○ Zoom sobre imagen. ○ Pulsación sobre botón de retroceso “< Tema”.

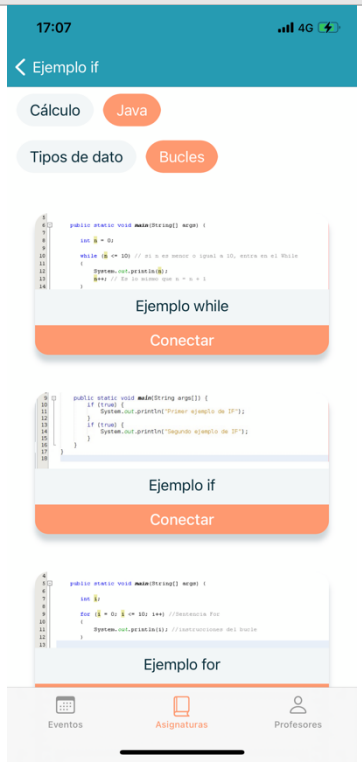
Visor de apuntes (tipo documento PDF)	
Interfaz	
Descripción	Muestra el documento PDF asociado a un apunte.
Se presenta cuando	El usuario pulsa sobre un apunte en particular en la pantalla Apuntes, Conexiones, Apuntes de etiqueta o Añadir conexión y este resulta ser de tipo PDF.
Acciones	○ Scroll sobre documento. ○ Pulsación sobre botón de retroceso “< Tema”.

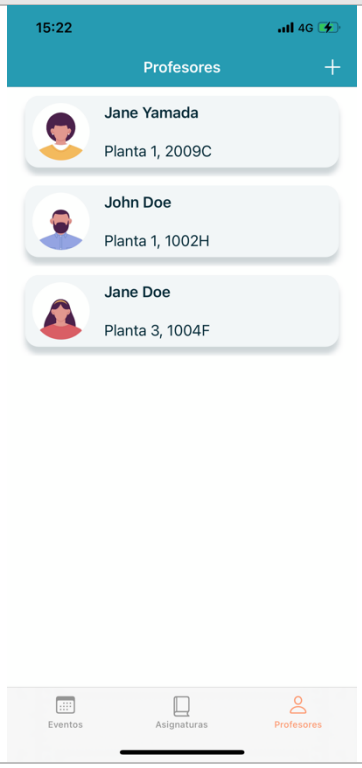
Etiquetas	
Interfaz	
Descripción	Muestra la lista de etiquetas de un apunte registrados por el usuario en forma de tabla.
Se presenta cuando	El usuario pulsa sobre el botón “Etiquetas” de un apunte particular en la pantalla Apuntes.
Acciones	○ Scroll sobre la tabla de etiquetas. ○ Pulsación sobre una etiqueta. ○ Pulsación sobre botón “+”. ○ Gesto “swipe” para eliminar una etiqueta.

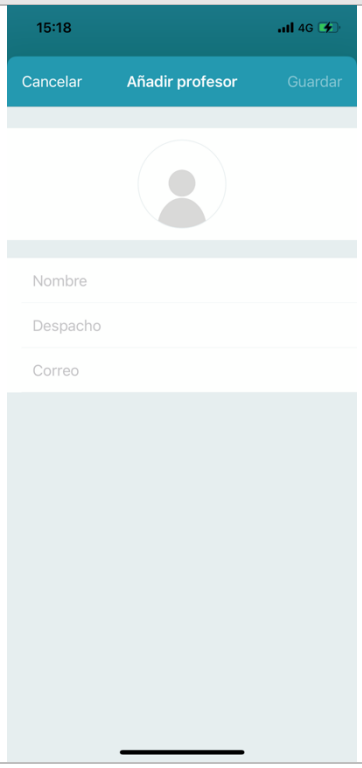
Añadir etiqueta	
Interfaz	
Descripción	Proporciona un formulario para introducir una nueva etiqueta, así como una tabla con las etiquetas registradas anteriormente por el usuario.
Se presenta cuando	El usuario pulsa sobre el botón “+” de la barra de navegación superior en la pantalla Etiquetas.
Acciones	○ Pulsación sobre botón “Hecho”. ○ Gesto “dismiss” para descartar la pantalla. ○ Pulsación sobre campo “Etiqueta”. ○ Pulsación sobre botón “añadir”. ○ Scroll sobre tabla “Tus etiquetas”. ○ Pulsación sobre una etiqueta

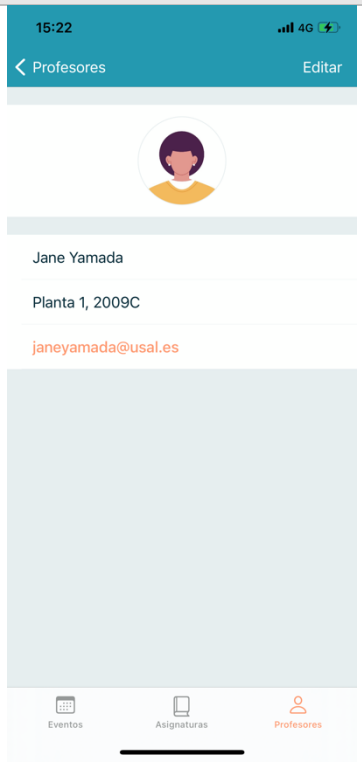
Apuntes de etiqueta	
Interfaz	
Descripción	Muestra la lista de apuntes en los que aparece una etiqueta en forma de tabla.
Se presenta cuando	El usuario pulsa sobre una etiqueta en particular en la pantalla Etiquetas.
Acciones	○ Pulsación sobre un apunte. ○ Pulsación sobre botón “Conectar” de un apunte. ○ Gesto “dismiss” para descartar la pantalla.

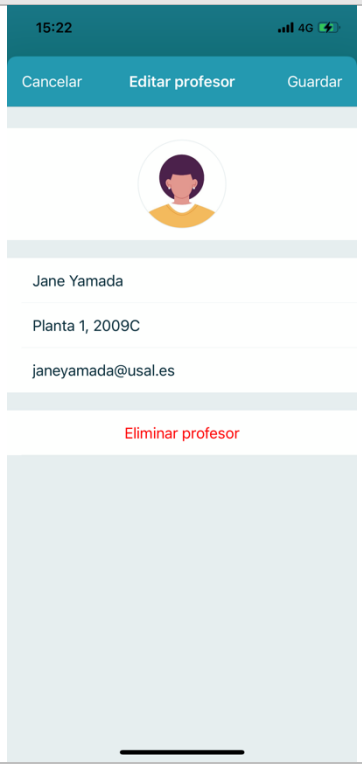
Conexiones	
Interfaz	
Descripción	Muestra la lista de apuntes conectados a otro apunte por el usuario en forma de tabla.
Se presenta cuando	El usuario pulsa sobre el botón “Conexiones” de un apunte particular en la pantalla Apuntes.
Acciones	○ Scroll sobre tabla de apuntes. ○ Pulsación sobre botón “+”. ○ Pulsación sobre botón de retroceso “< Tema”. ○ Gesto “swipe” para eliminar un apunte.

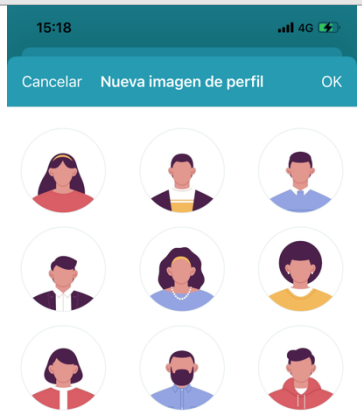
Añadir conexión	
Interfaz	
Descripción	Muestra una lista de apuntes en función de la asignatura y tema al que pertenecen en forma de tabla.
Se presenta cuando	El usuario pulsa sobre el botón “+” de la barra de navegación superior en la pantalla Conexiones.
Acciones	○ Scroll sobre tabla de apuntes. ○ Scroll horizontal sobre colección de asignaturas. ○ Scroll horizontal sobre colección de temas. ○ Pulsación sobre asignatura. ○ Pulsación sobre tema. ○ Pulsación sobre botón “Conectar” de un apunte. ○ Pulsación sobre botón de retroceso “< Apunte”.

Profesores	
Interfaz	
Descripción	Muestra la lista de profesores registrados por el usuario en forma de tabla.
Se presenta cuando	El usuario pulsa sobre el icono “Profesores” de la barra de navegación inferior.
Acciones	○ Scroll sobre la tabla de profesores. ○ Pulsación sobre un profesor. ○ Pulsación sobre botón “+”. ○ Gesto “swipe” para eliminar un profesor.

Añadir profesor	
Interfaz	
Descripción	Proporciona un formulario con los campos necesarios para crear un nuevo profesor.
Se presenta cuando	El usuario pulsa sobre el botón “+” de la barra de navegación superior en la pantalla Profesores.
Acciones	○ Pulsación sobre botón “Cancelar”. ○ Pulsación sobre botón “Guardar”. ○ Gesto “dismiss” para descartar la pantalla. ○ Pulsación sobre campos “Nombre”, “Despacho” y “Correo” para introducir texto. ○ Pulsación sobre campo “avatar”. ○ Validación del campo “Nombre”.

Detalle de profesor	
Interfaz	
Descripción	Muestra la información asociada a un profesor registrado por el usuario en forma de tabla.
Se presenta cuando	El usuario pulsa sobre un profesor en particular en la pantalla Profesores.
Acciones	○ Pulsación sobre botón de retroceso “< Profesores”. ○ Pulsación sobre botón “Editar”.

Editar profesor	
Interfaz	
Descripción	Proporciona un formulario con los campos necesarios para editar un profesor, así como la opción de eliminación de profesor a través del botón “Eliminar profesor”.
Se presenta cuando	El usuario pulsa sobre el botón “Editar” de la barra de navegación superior en la pantalla Detalle de profesor.
Acciones	○ Pulsación sobre botón “Cancelar”. ○ Pulsación sobre botón “Guardar”. ○ Gesto “dismiss” para descartar la pantalla. ○ Pulsación sobre campos “Nombre”, “Despacho” y “Correo” para introducir texto. ○ Pulsación sobre campo “avatar”. ○ Validación del campo “Nombre”. ○ Pulsación sobre botón “Eliminar profesor”.

Selección de avatar	
Interfaz	
Descripción	Proporciona un conjunto de avatares para asociar a un profesor.
Se presenta cuando	El usuario pulsa sobre el campo “Avatar” en el formulario de la pantalla Editar profesor.
Acciones	○ Pulsación sobre botón “Cancelar”. ○ Pulsación sobre botón “OK”. ○ Gesto “dismiss” para descartar la pantalla. ○ Pulsación sobre imagen.

6.3. Bases de datos

Como ya se ha indicado en apartados anteriores, para el almacenamiento de datos se utilizan las bases de datos Firestore y Storage que proporciona la plataforma Firebase.

Desde la consola de Firebase es posible gestionar los proyectos vinculados a la cuanta de Google con la que se está trabajando. Cada proyecto puede contener a su ver múltiples aplicaciones. Una vez seleccionada la aplicación se puede acceder a su vez a las distintas funcionalidades que Firebase ofrece, en este caso las bases de datos.

6.3.1. Firestore

Cloud Firestore es una base de datos no relacional o también clasificada como NoSQL (del inglés, “not only SQL”). La principal característica de este tipo de bases de datos, y lo que las diferencia de las bases de datos relacionales, es que no emplean tablas y filas para el almacenamiento de la información.

Para almacenar datos las bases de datos no relacionales pueden servirse de pares clave-valor, documentos y gráficos. En el caso de Firestore, se utilizan documentos que a su vez se organizan en colecciones.

Los documentos se identifican a través de un sistema de nombres y contienen una serie de atributos declarados como pares clave-valor. Estos atributos pueden identificar diferentes tipos de datos, desde datos sencillos como String hasta datos más complejos como mapas.

Las colecciones actúan de contenedores para los anteriores. Los documentos no pueden existir fuera de las colecciones. Además, las colecciones pueden existir dentro de los documentos, es decir, los documentos contenidos en una colección pueden contener a su vez subcolecciones y objetos anidados.

Para acceder a la información en Firestore se necesita la referencia a la ubicación de esta en la base de datos. Este mecanismo funciona dado que cada documento tiene un identificador para su ubicación de forma única.

Entre las ventajas que supone una base de datos no relacional se encuentran las siguientes:

- Mayor flexibilidad. Principalmente a la hora de diseñar el esquema de la estructura de la base de datos.

- Mayor escalabilidad. Soportan grandes volúmenes de datos.

- Información no estructurada. Permiten la gestión de datos sin necesidad de tener una noción clara y exacta de la información que se almacenará.

El siguiente esquema representa los objetos que conforman la estructura de la base de datos para la aplicación Study Assistant.

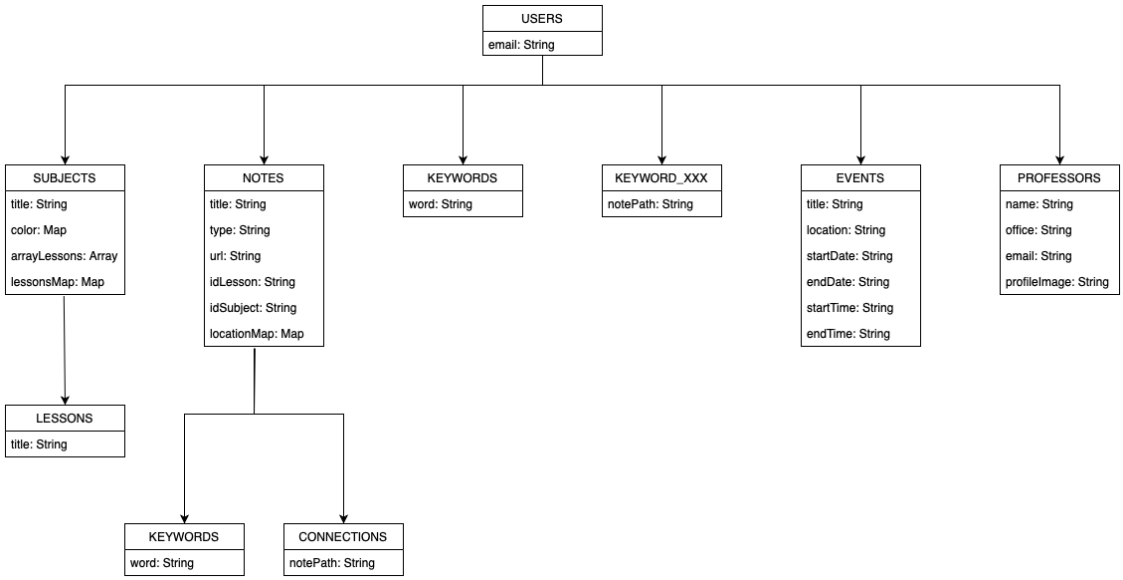


Ilustración 17 - Esquema de tablas en base de datos

Cada tabla representa a su vez una colección y un objeto o documento de esa misma colección. Las listas de cada tabla definen los atributos de los objetos o documentos de la base de datos. Por último, las flechas que dibujan conexiones entre tablas representan la jerarquía que siguen las colecciones, así, por ejemplo, la colección “USERS” contendrá documentos que representarán a los usuarios y estos a su vez contendrán la subcolección “SUBJECTS”.

6.3.2. Storage

Para el almacenamiento de contenido multimedia relacionado con los apuntes se utiliza, como ya mencionado en apartados anteriores de este documento, la base de datos Cloud Storage.
A diferencia de Cloud Firestore, lo que se almacenan en Cloud Storage no son colecciones y sus documentos contenidos sino objetos. En este caso en concreto los objetos serán imágenes con extensión .jpeg y documentos con extensión .pdf.

La información almacenada sigue un simple esquema: para cada usuario se crea una carpeta raíz utilizando como identificador único su dirección de correo electrónico y dentro de este directorio se almacenan todos los objetos que haya registrado en forma de apuntes a través de la aplicación.

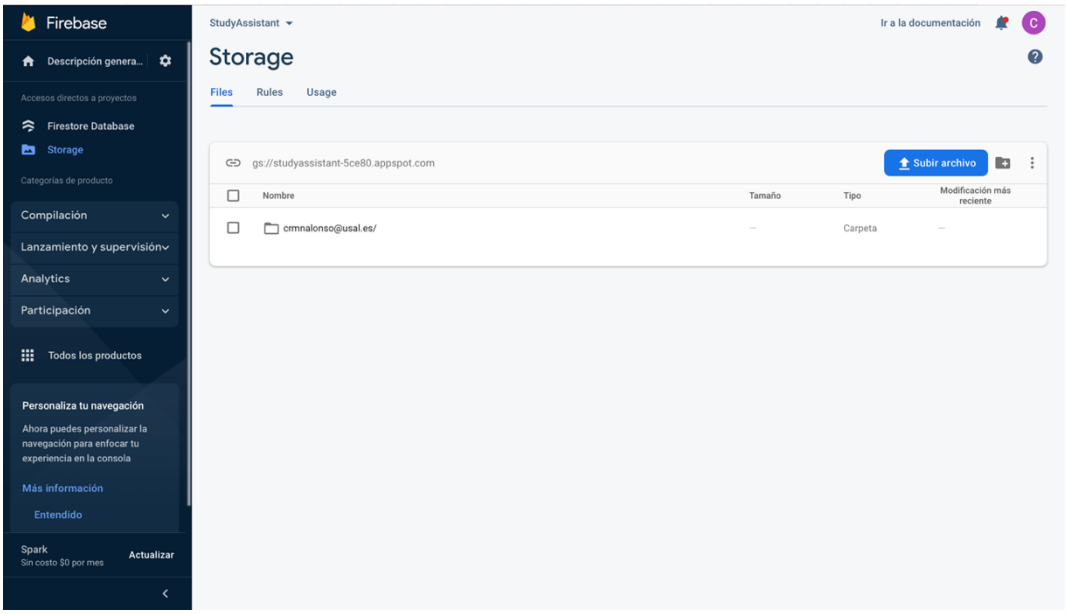


Ilustración 18 - Ejemplo de usuario en Storage

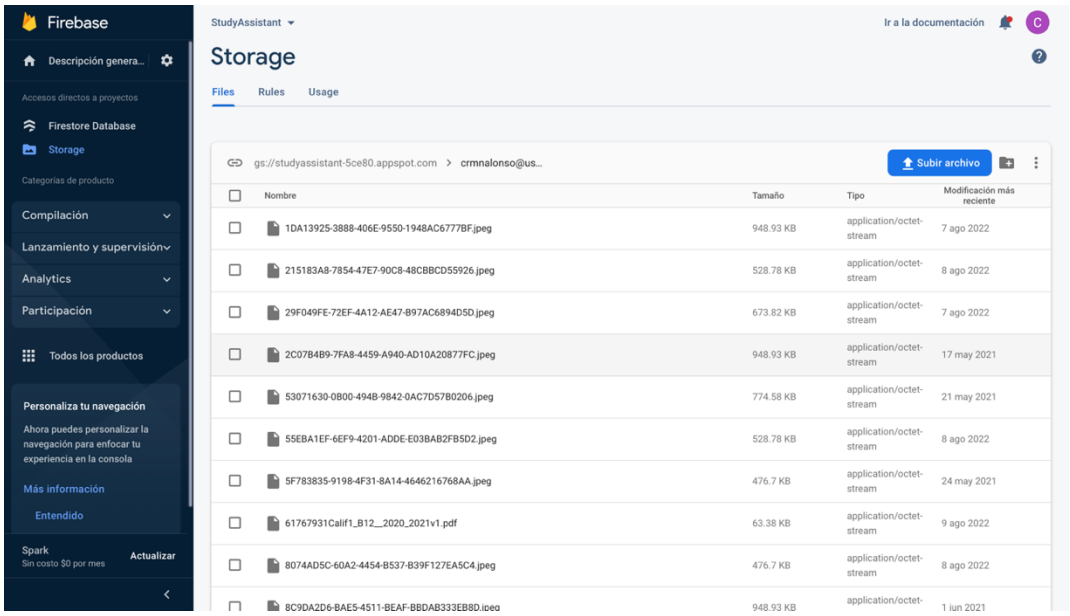


Ilustración 19 - Ejemplo del contenido de un usuario en Storage

6.4. Autenticación de usuarios

Para la autenticación de usuarios de utiliza la opción que ofrece la herramienta Firebase a través de proveedores de identidad federada, en concreto, a través de la plataforma de Google.

El sistema se integra en la aplicación de manera que los usuarios llevan a cabo un proceso de autenticación a través de sus credenciales de Google. De este modo, en ningún momento se gestionan contraseñas ni se pone en riesgo la privacidad de los usuarios.

6.5. Arquitectura

El sistema final resultante de este proyecto consta de una aplicación móvil que actúa a modo de cliente y un servidor externo proporcionado por la plataforma Firebase y sus funcionalidades. Estas dos partes se comunican a través de una conexión a internet.

Cabe puntualizar que la aplicación dispone de un solo tipo de usuario, "usuario estándar" o "estudiante".

A continuación, se muestra un esquema sintetizando la arquitectura de la aplicación.

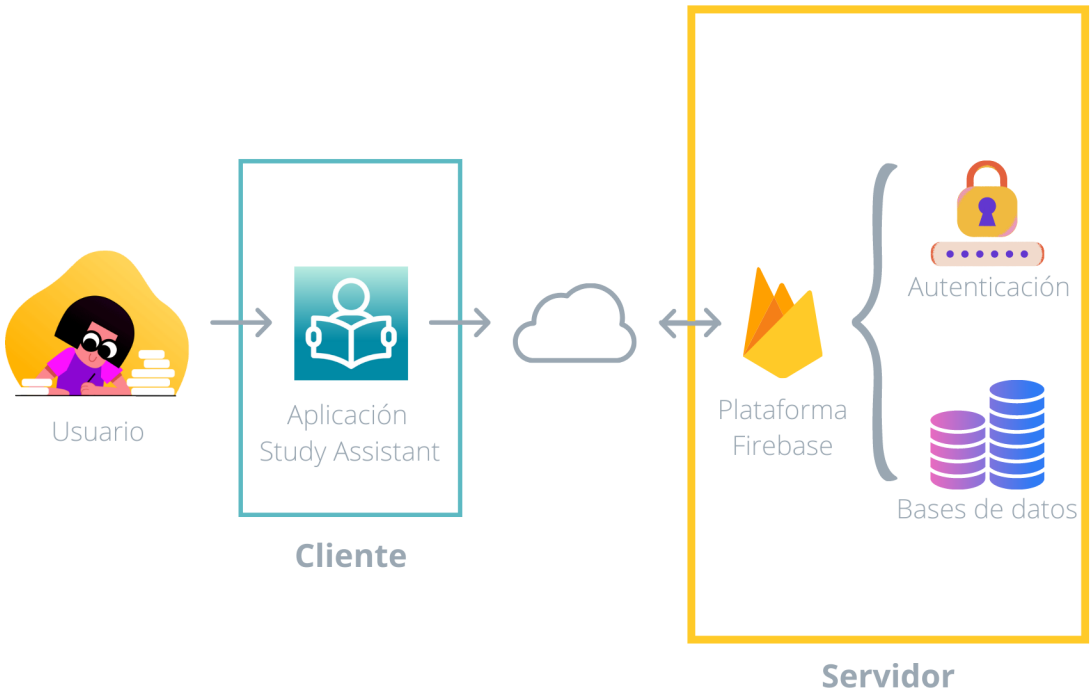


Ilustración 20 - Esquema de la arquitectura de la aplicación

6.6. Metadatos

La etimología de la palabra metadatos proviene del griego "meta", prefijo que significa "después, detrás de, junto a" y "data" que significa "datos" [30]. De forma literal metadatos significa "más allá de los datos". El primer uso de la palabra metadatos, en este ámbito de la información, data de finales de los años sesenta. La acepción más extendida en la actualidad es la de "dato sobre dato".

Esta noción es anterior a la popularización de los sistemas informáticos e Internet, sin embargo, es por las capacidades de manejo de datos digitales que ofrece en especial este último por lo que el concepto comienza a extenderse en estos campos.

Los metadatos consisten en información ligada a unos determinados datos y que sirve para la descripción de estos actuando como un identificador de los objetos.

Estos datos se generan de manera automática y sin control inicial del usuario. Por ejemplo, en el caso de un documento de texto, el usuario introduce una serie de datos en el objeto, en este escenario el propio documento y una vez creado dicho objeto la aplicación en la que fue concebido generará una serie de datos que asociará al documento tales como: autor, fecha de creación, última fecha de modificación o tamaño del documento.

Los metadatos entendidos como ha sido descrito anteriormente, una información complementaria sobre unos determinados datos que describen e identifican a los mismos, reciben el nombre de metadatos descriptivos. La Organización Nacional de Estándares de Información de los Estados Unidos (NISO) distingue además otros tres tipos [31]:

- Metadatos administrativos: Precisan información administrativa con la que se facilita el procesamiento y la gestión de los datos asociados. Se incluye en este grupo información relativa a aspectos como el tipo de documento o los derechos de autor.

- Metadatos estructurales: Aportan información acerca de las relaciones entre los recursos facilitando su procesamiento, navegación y visualización.

- Lenguajes de marcado (Markup languages): Estos lenguajes se caracterizan por combinar contenido y metadatos. El ejemplo más conocido es HTML.

Entendiendo la definición de esta herramienta se hacen evidentes las ventajas que supone su utilización:

- Visualización de datos de manera sencilla y eficiente.

- Posibilidad de creación de relaciones entre objetos.

- Recuperación de información.

- Incremento del valor de los propios datos.

En este proyecto los metadatos se utilizarán con fines estructurales. Así, la integración de metadatos en la información generada por el usuario, los apuntes, permitirán llevar a cabo procesos de búsqueda, organización y gestión de manera enriquecida.

Dadas las características particulares de las bases de datos utilizadas por la aplicación, los metadatos se incluirán en los documentos alojados en las colecciones pertenecientes a la base Cloud Firestore, en concreto, a través de sus campos. Por una parte, existirán metadatos generados automáticamente por el sistema para relacionar los apuntes con las respectivas asignaturas y temas a los que pertenecen. Por otro lado, se dará la posibilidad al usuario de introducir metadatos específicos asociados a cada apunte en forma de “etiquetas”. Estas etiquetas servirán para clasificar los apuntes por contenido o campo, por ejemplo, según el usuario vea conveniente.

7. Implementación

7.1. Estructura del proyecto

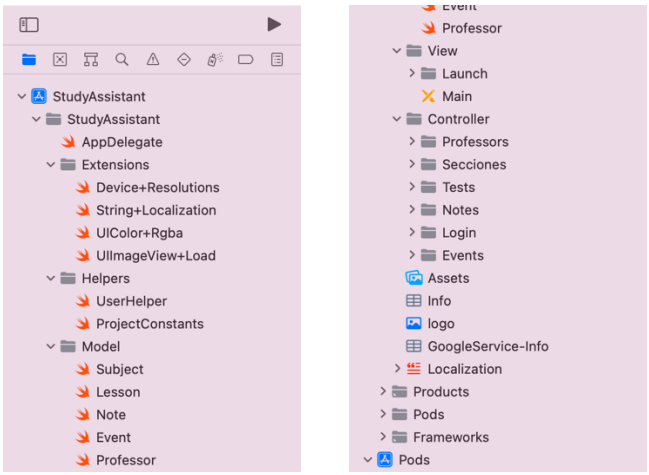


Ilustración 21 - Estructura de carpetas de Study Assistant en XCode

7.1.1. Modelo

El modelo de la aplicación está compuesto por cinco tipos de objeto: Subject (Asignatura), Lesson (Tema), Note (Apunte), Event (Evento) y Professor (Profesor).

Cada uno de ellos dispone de una serie de atributos necesarios para gestionar la información relativa al tipo de objeto, así como un constructor con aquellos atributos obligatorios para inicializar un objeto del tipo.

7.1.2. Vistas

Para el desarrollo de las vistas que componen la interfaz de la aplicación se emplea el framework UIKit, como ya se ha indicado en anteriores capítulos.

Este enfoque se basa en el uso de los llamados “storyboards”. Estos sirven de representación para las pantallas a nivel de diseño de interfaz de usuario, permitiendo el diseño de estas mediante componentes gráficos y controles.

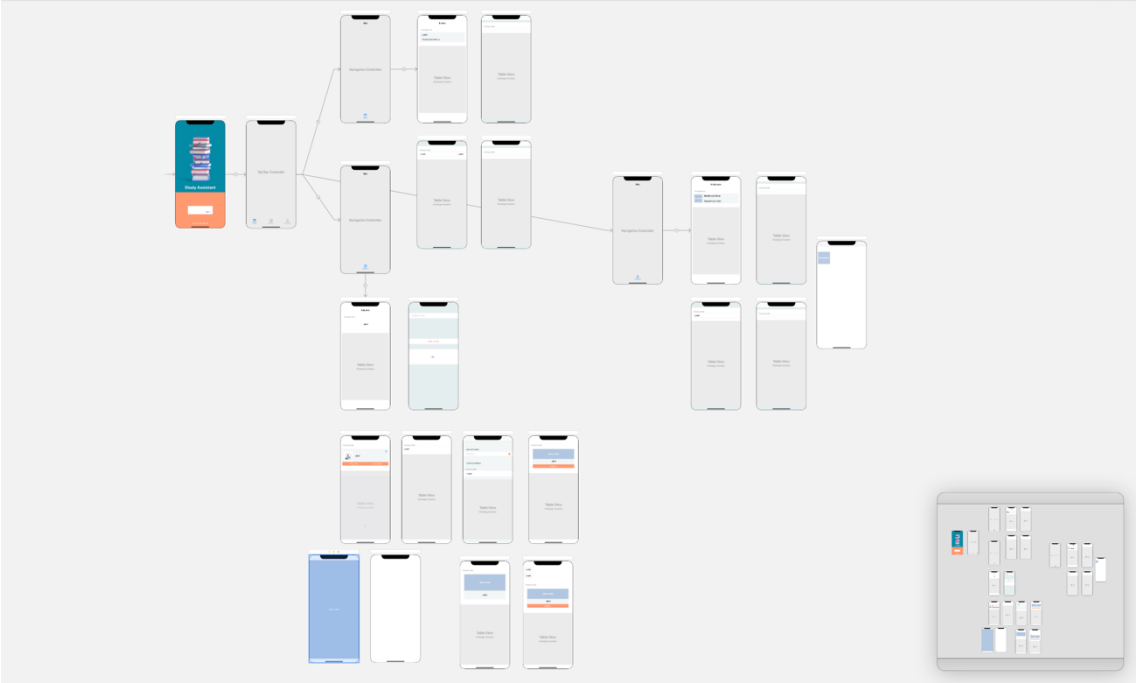


Ilustración 22 - Vista del storyboard Main en Xcode

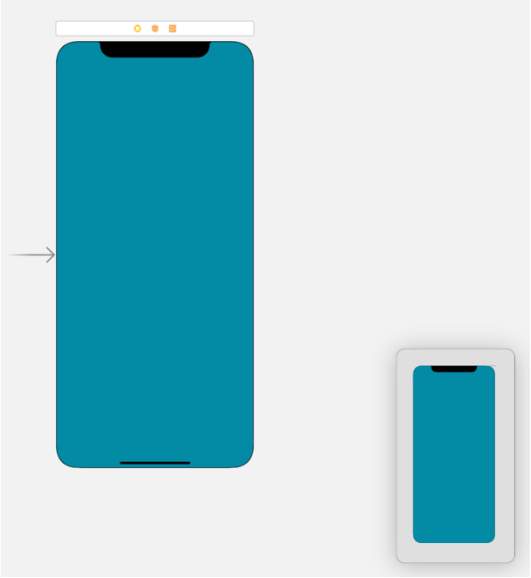


Ilustración 23 - Vista del storyboard LaunchScreen en Xcode

Estas vistas han sido diseñadas específicamente para el modelo iPhone 11. Para conseguir que sus diseños se adapten a la resolución de cualquier dispositivo se ha hecho uso de la herramienta AutoLayout.

A través de AutoLayout se han aplicado las restricciones necesarias de posición, tamaño y márgenes a los distintos elementos gráficos que aparecen en cada pantalla. El tamaño y posición de estos elementos es calculado dinámicamente dada la resolución del dispositivo en el que la aplicación esté ejecutándose, de este modo, el diseño se mantiene intacto "traduciéndose" para las dimensiones de la pantalla de cada dispositivo.

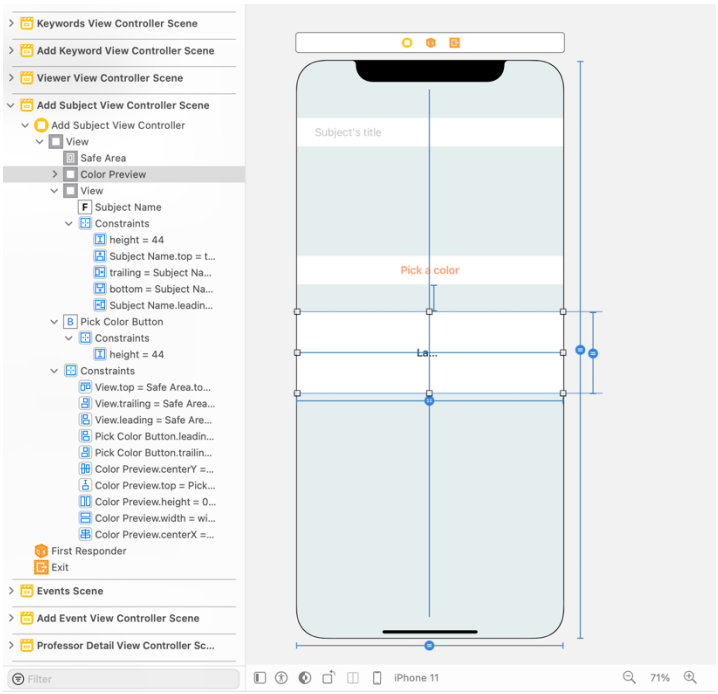


Ilustración 24 - Ejemplo de restricciones aplicadas a un elemento visual en XCode

7.1.3. Controladores

Cada una de las vistas o storyboards tiene asociado un controlador que determina la lógica con la que actuarán los elementos dispuestos en ellas.

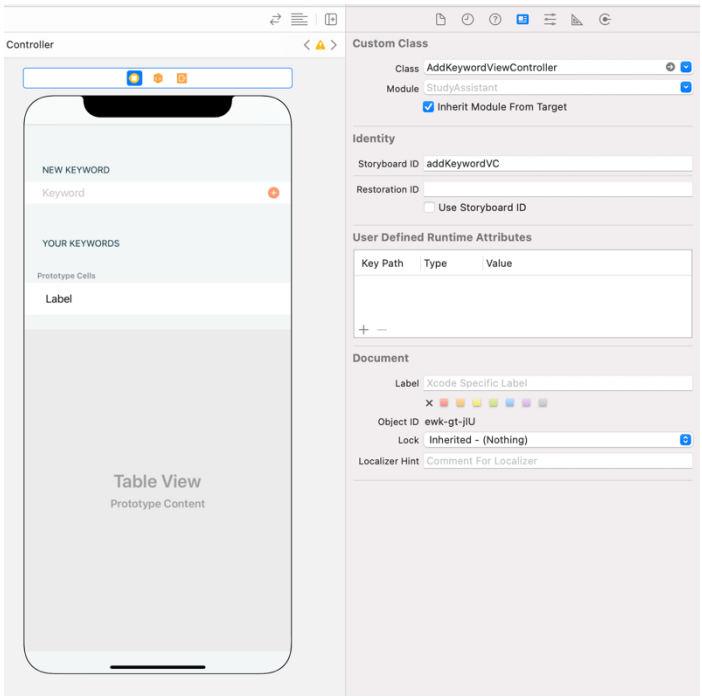


Ilustración 25 - Ejemplo de una vista con controlador asociado en XCode

7.1.4. Comunicación entre controladores

Una de las principales características de los controladores es su capacidad para interactuar con otros controladores.

En primer lugar, desde un primer controlador se puede solicitar la presentación de un segundo controlador. Esto ocurre cuando el usuario realiza la acción de navegar entre diferentes pantallas de la aplicación. Los controladores pueden presentarse de manera que ocupen la totalidad de la pantalla o bien con un estilo de hoja modal superpuesta sobre el anterior controlador. En el primer caso, los controladores se organizan a modo de pila. A la hora de presentar un controlador se realiza una operación "push", básica cuando hablamos de manejo de pilas; mientras que cuando se descarta o cierra dicho controlador, la operación que tiene lugar es la de "pop".

En segundo lugar, los controladores pueden comunicarse con otros para enviar o recibir información. En estos casos es fundamental determinar el tipo de relación entre los controladores puesto que este decidirá la forma de realizar dicha comunicación.

Existen dos métodos para realizar la comunicación:

- Delegados. Se utiliza esta opción cuando la comunicación se da entre solo dos controladores. En este caso, el controlador que enviará la información dispone de un protocolo, algo similar a una plantilla que define ciertos métodos, propiedades u otros requisitos. El controlador que recibirá la información tiene que adecuarse, o, dicho de otro modo, obedecer ese protocolo, ejecutando la lógica que este defina.

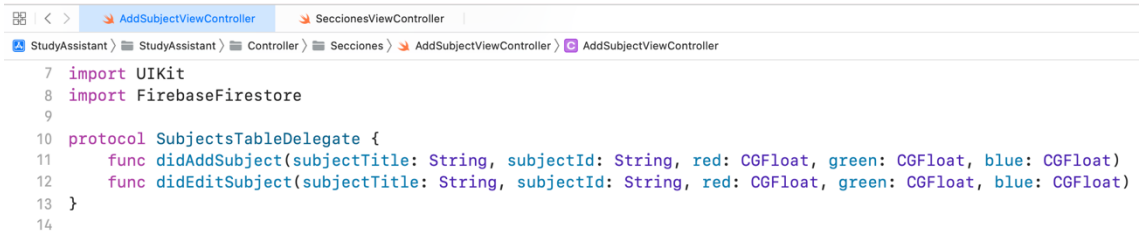


Ilustración 26 - Ejemplo de protocolo

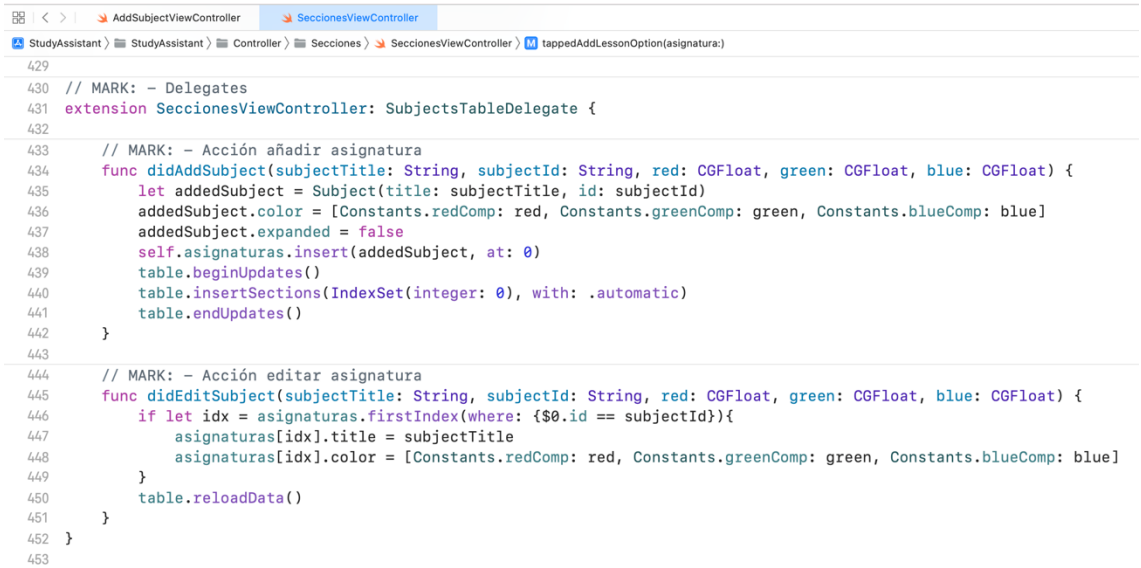


Ilustración 27 - Ejemplo de delegación

- Notificaciones. Cuando la relación entre los controladores que se van a comunicar es del tipo “uno a muchos” se utilizan las notificaciones. En este método, el controlador que enviará la información publica en el centro de notificaciones una notificación con esta información. Los controladores que quieran recibir dicha información, por su parte, se declararán como observadores de esa notificación, siendo así notificados.

```
93
94          NotificationCenter.default.post(name: Notification.Name(Constants.editProfessorNotification), object: self)
95      }
```

Ilustración 28 - Ejemplo de publicación de notificación en el centro de notificaciones

```
EditProfessorViewController | ProfessorDetailViewController | ProfessorsViewController
StudyAssistant 〉 StudyAssistant 〉 Controller 〉 Professors 〉 ProfessorDetailViewController 〉 M viewDidLoad()
33
34          //MARK: - Añadir observador para notificaciones de edición de un profe
35          NotificationCenter.default.addObserver(self, selector: #selector(didEditProfessor), name:
                Notification.Name(Constants.editProfessorNotification), object: nil)
```

Ilustración 29 - Ejemplo de observador en un controlador A

```
EditProfessorViewController | ProfessorDetailViewController | ProfessorsViewController
StudyAssistant 〉 StudyAssistant 〉 Controller 〉 Professors 〉 ProfessorsViewController 〉 M tableView(_:didSelectRowAt:)
42
43          //MARK: - Añadir observador para notificaciones de edición de un profe
44          NotificationCenter.default.addObserver(self, selector: #selector(didEditProfessor), name:
                Notification.Name(Constants.editProfessorNotification), object: nil)
```

Ilustración 30 - Ejemplo de observador en un controlador B

7.2. Comunicación con la base de datos

Para interactuar con la base de datos Firebase se utilizan tres operaciones distintas:

- Carga de datos. A través del método “setData” se cargan los datos incluidos en los parámetros en la dirección especificada en la referencia sobre la que se opera.

- Eliminación de datos. A través del método “delete” se eliminan los datos asociados a un documento especifico. En Firebase no se puede realizar operaciones de borrado recursivas, esto significa que al borrar un documento, si este contiene subcolecciones y documentos a su vez en su interior estos deberán ser eliminados específicamente. Por lo anterior, realizar una operación de borrado sobre una colección no tiene sentido. Cuando el contenido alojado en una colección sea eliminado y esta quede vacía, se eliminará automáticamente del esquema de base de datos.

- Recuperación de datos. La descarga de información se realiza a través del método “getDocuments”, que devuelve todos los documentos alojados en la ruta de la base de datos indicada; o “getDocument”, que devuelve un documento específico.

```
96          ref.setData([Constants.fieldProfessorName:name ?? "",
97                       Constants.fieldProfessorOffice:office ?? "",
98                       Constants.fieldProfessorEmail:email ?? "",
99                       Constants.fieldProfessorImage: imageName ?? Constants.profileImageDefault])
```

Ilustración 31 - Ejemplo de setData

```
50      func loadData() {
51          db.collection(Constants.collectionUsers).document(UserHelper.getUserEmail()).collection(Constants.collectionProfessors)
                .getDocuments() { (querySnapshot, err) in
52              if let err = err {
53                  print("Error getting professors documents: \(err)")
54              } else {
55                  for document in querySnapshot!.documents {
56                      let professor = Professor(name: document.get(Constants.fieldProfessorName) as! String,
57                                                id: document.documentID)
58                      professor.email = document.get(Constants.fieldProfessorEmail) as? String
59                      professor.office = document.get(Constants.fieldProfessorOffice) as? String
60                      professor.profilePic = document.get(Constants.fieldProfessorImage) as? String
61                      self.professors.append(professor)
62                      self.table.reloadData()
63                  }
64              }
65          }
66      }
```

Ilustración 32 - Ejemplo de getDocuments

```
self.db.collection(Constants.collectionUsers).document(UserHelper.getUserEmail()).collection(Constants.collectionProfessors)
    .document(id).delete(){ err in
    if let err = err {
        print("Error removing professor document: \(err)")
    }
}
```

Ilustración 33 - Ejemplo de delete

Para la carga y descarga de información en la base Storage se hace uso de los datos del objeto que se gestiona dado que son objetos multimedia y no colecciones y documentos como en el caso de Firebase.

```
676             let fileRef = storageRef.child(UserHelper.getUserEmail() + "/" + fileName)
677             if let pdfData = NSData(contentsOf: urls[0]) as Data? {
678                 let uploadTask = fileRef.putData(pdfData, metadata: nil) { metadata, error in
679                   guard let metadata = metadata else {
680                       print(error!)
681                     return
682                   }
```

Ilustración 34 - Ejemplo de carga de datos en Storage

7.3. Notificaciones Push

El módulo de eventos implementa un sistema de notificaciones Push para notificar al usuario un aviso a modo de recordatorio de los eventos.

Una notificación se compone de tres partes:

- Petición. Con esta ella se especifica un identificador para la notificación en el sistema, su contenido y su detonante.

- Contenido. Está formado por el texto que aparecerá tanto en el título de la notificación como en el cuerpo. A través de este se puede especificar el sonido que acompañará a la notificación.

- Detonante. El detonante o trigger es la fecha (día y hora) a la que la notificación se presentará en el dispositivo al usuario.

```
180             let content = UNMutableNotificationContent()
181             content.title = Constants.eventNotificationTitle.localize()
182             content.sound = .default
183             content.body = title ?? ""
184
185
186             let trigger = UNCalendarNotificationTrigger(dateMatching: Calendar.current.dateComponents([.year, .month, .day, .hour,
                    .minute, .second], from: triggerDate), repeats: false)
187
188             let request = UNNotificationRequest(identifier: id, content: content, trigger: trigger)
189
190             UNUserNotificationCenter.current().add(request, withCompletionHandler: {error in
191                 if error != nil {
192                     print("error")
193                 }
194             })
```

Ilustración 35 - Ejemplo de creación de una notificación Push

Para poder utilizar las notificaciones Push es necesario solicitar autorización al usuario. Esto ocurre siempre que se haga uso de funcionalidades propias del sistema operativo.

```
98          UNUserNotificationCenter.current().requestAuthorization(
99              options: [.alert, .badge, .sound],
100             completionHandler: {success, error in
101                 if let error = error{
102                     print("error \(error)")
103                 }
104             })
```

Ilustración 36 - Ejemplo de solicitud de autorización para acceder al centro de notificaciones

7.4. Helpers

Este tipo de archivos, como su nombre indica, son “ayudantes”. Proporcionan soporte para una acción determinada que se repite en diferentes puntos a lo largo de toda la aplicación. Con el uso de helpers se evita el uso de código duplicado.

Este proyecto incluye el helper UserHelper.swift, que obtiene el correo electrónico del usuario, elemento con el que se le identifica en el sistema.

7.5. Extensions

Las extensiones o extensions se utilizan para añadir nuevas funcionalidades a clases, estructuras, enumeraciones o protocolos ya existentes.

En este proyecto se implementan cuatro extensiones:

- Device+Resolutions.swift. Utilizada para detectar el dispositivo en el que se está ejecutando la aplicación en base a la medida de la altura de la pantalla detectada.
- String+Localization.swift. Usada para realizar la traducción de los literales del inglés al español.
- UIColor+Rgba.swift. Es empleada para obtener los componentes RGBA de un color del tipo UIColor.
- UIImageView+Load.swift. Usada para la descarga de imágenes almacenadas en la base de datos.

7.6. Constantes

Las constantes utilizadas a lo largo del proyecto se engloban en el archivo ProjectConstants, lo que permite la edición de estos literales de manera rápida y sencilla.

Las constantes son declaradas dentro del struct Constants. Es posible, y una práctica común, declarar structs anidados en el primero como método de clasificación de las constantes. En este caso se ha decidido evitar este enfoque para evitar los accesos adicionales que ello supondría.

Para el nombramiento de las constantes se sigue la convención de nombres que establece Swift. Según esta norma los tipos y protocolos son nombrados con el estilo UpperCamelCase (el primer carácter de cada palabra, así como el primero del nombre completo como mayúscula y el resto como minúscula). El resto es nombrado con el estilo lowerCamelCase (muy parecido a UpperCamelCase, la única diferencia es que el primer carácter del nombre se escribe en minúscula).

7.7. Localización

Como ya ha sido mencionado anteriormente en este documento, la aplicación Study Assistant ha sido desarrollada inicialmente en inglés con el objetivo de traducirla posteriormente al español. Este proceso recibe el nombre de localización.

Para llevar a cabo la localización es necesario indicar los idiomas que serán gestionados en el apartado Localization del proyecto.

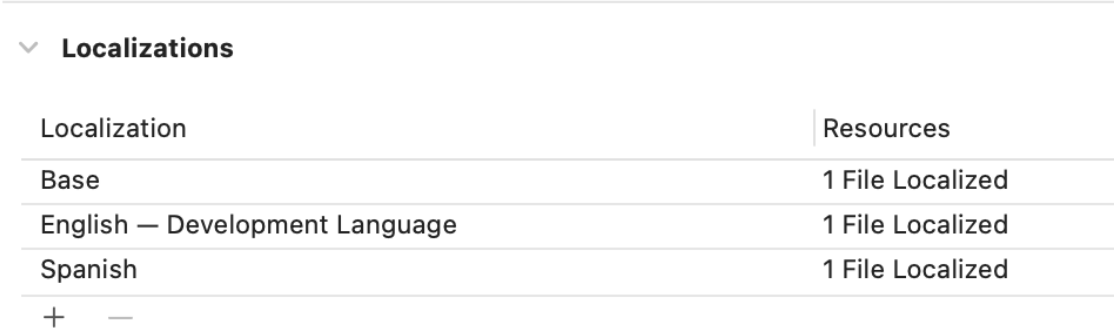


Ilustración 37 - Sección Localizations de la configuración de un proyecto en XCode

Para llevar a cabo las traducciones se utiliza un archivo de cadenas llamado en este caso Localization (Spanish). En él se indican las cadenas que se desean traducir y su correspondiente traducción. Estos dos elementos actúan a modo de par clave-valor, así, dada la clave (cadena original), se presentará su valor (cadena traducida). Esta traducción se lleva a cabo gracias a la extensión mencionada anteriormente String+Localization.swift.

Ilustración 38 - Ejemplo de tabla de localización

8. Pruebas

Para la realización de pruebas de un desarrollo software se pueden llevar a cabo dos enfoques:

- Pruebas de caja blanca. Estas pruebas, unitarias, de integración y sobre las funcionalidades del sistema don llevadas a cabo principalmente durante la etapa de implementación. El responsable de este tipo de pruebas conoce la aplicación, su estructura y el código, por esto precisamente, se llevan a cabo por el equipo de desarrolladores.

- Pruebas de caja negra. En este caso, al contrario que en las pruebas de caja blanca, los responsables de llevar a cabo las pruebas no conocen el sistema internamente ni es necesario que dispongan de conocimientos de programación o similares. Estas pruebas son realizadas por usuarios finales.

Para este proyecto se han realizado de manera paralela al desarrollo pruebas de caja blanca. El objetivo de estas ha sido la comprobación de los siguientes elementos:

- Validación de datos de entrada en formularios.

- Correcta ejecución de operaciones CRUD.

- Persistencia de la información en base de datos.

- Correcta interacción con los elementos de la interfaz gráfica.

- Adaptación del diseño a diferentes dispositivos.

9. Conclusiones y trabajo futuro

Con la exposición de este capítulo se completa finalmente el desarrollo del TFG Aplicación iOS para el estudio: Study Assistant.

A lo largo del desarrollo de este proyecto se ha llevado a cabo una formación constante en áreas de conocimiento totalmente nuevas como el lenguaje de programación Swift o la gestión de bases de datos de tipo no relacional ofrecidas por Firebase.

Además de lo anterior, de forma paralela, se han aplicado y afianzado conocimientos adquiridos gracias al transcurso del Grado en Ingeniería Informática de la Universidad de Salamanca.

Las grandes dificultades encontradas en las distintas fases del desarrollo han conseguido ser solventadas a través de intensos periodos de investigación y pruebas. Entre ellas, cabe destacar el diseño de una base de datos no relacional que se adaptara a las necesidades del proyecto.

En términos de gestión se ha conseguido profundizar en la metodología ágil Scrum, obteniendo una valiosa base de conocimiento para el entorno laboral, en el que, como ya se ha comentado en capítulos anteriores, se ha popularizado enormemente este marco de trabajo.

La realización de un proyecto de estas dimensiones ha permitido además comprender como todos esos campos de la ingeniería informática, aparentemente aislados, se enlazan entre si dando lugar a un proceso de fases coordinadas y coherentes enfocadas a la obtención de un producto de calidad.

Desde un punto de vista más personal, este proyecto me ha concedido la oportunidad de descubrir un nuevo campo de la informática que me apasiona y sobre el que quiero seguir desarrollándome como profesional. Con el desarrollo de una aplicación desde el inicio he logrado participar tanto en tareas de desarrollo e implementación como en tareas de diseño. Esta combinación de la parte técnica del desarrollo de software con una parte más artística, en la que se toman decisiones de diseño, me ha permitido desempeñarme al completo y de la manera que más me gusta.

La realización de este TFG me ha supuesto un punto de partida desde el que introducirme de lleno en el sector del desarrollo de aplicaciones móviles, especialmente iOS.

En definitiva, la realización de este trabajo ha permitido un acercamiento real al campo del diseño y desarrollo de aplicaciones móviles iOS, descubriendo así un camino en esta nueva etapa laboral que comienza.

9.1. Líneas de trabajo futuro

Una vez alcanzado este punto en el proyecto, es posible proponer futuras características en las que trabajar:

- Sistema de usuarios propio. Implantación de un sistema de usuarios que gestione nombres y contraseñas.

- Autenticación de usuarios a través de terceros. Aumento de vías de autenticación por medio de terceros para ofrecer alternativas a aquellos usuarios que no dispongan de una cuenta de Google.

- Actualización del sistema operativo objetivo. Adaptación del sistema a nuevas versiones iOS.

- Nuevas funcionalidades. Ampliación de las funcionalidades ofertadas en la aplicación.
- Pruebas de caja negra. Ejecución de pruebas con usuarios finales.
- Publicación en la App Store. Distribución de la aplicación a través de la tienda de aplicaciones oficial de Apple.

10. Referencias

[1] Instituto Nacional de Estadística. (2020, 16 de noviembre). Encuesta sobre Equipamiento y Uso de Tecnologías de Información y Comunicación en los Hogares [Comunicado de prensa]. https://www.ine.es/prensa/tich_2020.pdf

[2] Iberdrola, S.A. (s.f.). La brecha digital en el mundo y por qué provoca desigualdad. Iberdrola Compromiso social. https://www.iberdrola.com/compromiso-social/que-es-brecha-digital

[3] data.ai. (2022, 12 de enero). The State of Mobile in 2022: How to Succeed in a Mobile-First World As Consumers Spend 3.8 Trillion Hours on Mobile Devices. Data.ai Insights. https://www.data.ai/en/insights/market-data/state-of-mobile-2022/

[4] DITRENDIA. (2021, 22 de octubre). Informe Mobile 2021 – España y Mundo. https://ditrendia.es/informe-mobile-2021-espana-y-mundo/

[5] Fierro, L. (2021, 12 de noviembre). El futuro es hoy: las tendencias de tecnología educativa. Crehana. https://www.crehana.com/blog/empresas/tecnologia-educativa-tendencia/

[6] iStudiez Team (2019). iStudiez Pro (2.0.1) [Aplicación móvil]. App Store. https://apps.apple.com/es/app/istudiez-pro-student-planner/id310636441

[7] My Study Life, Ltd. (2022). My Study Life (3.2.2) [Aplicación móvil]. App Store. https://apps.apple.com/es/app/my-study-life-school-planner/id910639339

[8] Aplicativos Legais. (2022). Easy Study (4.9.7) [Aplicación móvil]. App Store. https://apps.apple.com/es/app/easy-study-gu%C3%ADa-estudiante/id993247888

[9] StudySmarter UG. (2022) Study Smarter (7.6.1) [Aplicación móvil]. App Store. https://apps.apple.com/us/app/studysmarter-school-uni/id1439949520

[10] The Homework App Ltd. (2022). The Homework App (9.9.11) [Aplicación móvil]. App Store. https://apps.apple.com/es/app/the-homework-app/id561371952

[11] StatCounter. (s.f.). Mobile Operating System Market Share Worldwide. StatCounter. https://gs.statcounter.com/os-market-share/mobile/worldwide

[12] Kantar Group and Affiliates (s.f.). Smartphone OS Market Share. Kantar. https://www.kantar.com/campaigns/smartphone-os-market-share

[13] Encyclopaedia Britannica. (s.f.). Android operating system. Encyclopaedia Britannica. https://www.britannica.com/technology/Android-operating-system

[14] Google. (2022, 11 de agosto). Android Architecture. Android Source Docs. https://source.android.com/docs/core/architecture?hl=en

[14] Heath, T. (2018, 2 de agosto). Apple is the first $1 trillion company in history. The Washington Post. https://www.washingtonpost.com/business/economy/apple-is-the-first-1-trillion-company-in-history/2018/08/02/ea3e7a02-9599-11e8-a679-b09212fb69c2_story.html?noredirect=on

[15] Apple Inc. (2007, 9 de junio). Apple Reinvents the Phone with iPhone [Comunicado de prensa]. https://www.apple.com/newsroom/2007/01/09Apple-Reinvents-the-Phone-with-iPhone/

[16] Verge Staff. (2013, 16 de spetiembre). iOS: A visual history. The Verge. https://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad

[17] Digital Dividend. (s.f.). What is the difference between native vs hybrid app? Digital Dividend Blog. https://www.digital-dividend.com/en/native-or-hybrid-the-difference/

[18] S. Gillis, Alexander. (s.f.). Native app definition. TechTarget. https://www.techtarget.com/searchsoftwarequality/definition/native-application-native-app

[19] Anónimo. (s.f.). Hybrid application definition. TechTarget. https://www.techtarget.com/searchsoftwarequality/definition/hybrid-application-hybrid-app

[20] Anónimo. (s.f.). Web application definition. TechTarget. https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app

[21] Project Management Institute. (2017). Success Rates Rise: Transforming the high cost of low performance (BRA-002-2017, 2-16). Project Management Institute. https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf

[22] Indusree. (2018, 22 de junio). Traditional vs. Agile Software Development Methodologies. KPI Partners Blog. https://www.kpipartners.com/blog/traditional-vs-agile-software-development-methodologies

[23] García-Peñalvo, F.J. y García A. (2017). Proceso. 101118: Ingeniería del software I. Universidad de Salamanca.

[24] Navarro, A. (2013). Revisión de metodologías ágiles para el desarrollo de software. Prospectiva. Doi: 11. 30. 10.15665/rp.v11i2.36. https://www.researchgate.net/publication/273302003_Revision_de_metodologias_agiles_para_el_desarrollo_de_software

[25] Zepeda, F.L. (2018). Metodologías Ágiles de Desarrollo de Sistemas [Manual de prácticas de la materia, Instituto tecnológico de Hermosillo]. RI TecNM. https://rinacional.tecnm.mx/bitstream/TecNM/3770/1/MANUAL%20METODOLOGIAS%20AGILES%20FLZM2019.pdf

[26] Sutherland, J. (2010). Scrum Handbook. The Scrum Training Institute. http://www.scrummaster.dk/lib/AgileLeanLibrary/People/JeffSutherland/scrumhandbook.pdf

[27] Takeuchi H. y Nonaka I. (1986). The New New Product Development Game. Harvard Business Review. Enero. https://hbr.org/1986/01/the-new-new-product-development-game

[28] F. Rico, D. (2015). What is the ROI of Agile vs. Traditional Methods? An analysis of XP, TDD, Pair Programming, and Scrum (Using Real Options) https://davidfrico.com/rico08g.pdf

[29] Apple. (s.f.). Human Interface Guidelines. Apple Developer. https://developer.apple.com/design/human-interface-guidelines/guidelines/overview/

[30] Greenberg, J. (2009). Encyclopedia of Library and Information Sciences, Third Edition. Taylor & Francis. https://www.taylorfrancis.com/chapters/edit/10.1081/E-ELIS3-120044415/metadata-digital-information-jane-greenberg

[31] Riley, J. (2017, 1 de enero). Understanding metadata what is metadata, and what is it for? A Primer Publication of the National Information Standards Organization. National Information Standards Organization. https://www.niso.org/publications/understanding-metadata-2017

11. Listado de ilustraciones