UNIVERSIDAD DE VALLADOLID
E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Diseño y desarrollo de una aplicación móvil para el aprendizaje del lenguaje JavaScript

Autor: **Luis Eduardo Casado Vecino**

Tutoras: **Míriam Antón Rodríguez**
María Ángeles Pérez Juárez

Valladolid, julio de 2020

TÍTULO:	**Diseño y desarrollo de una aplicación móvil para el aprendizaje del lenguaje JavaScript**
AUTOR:	**Luis Eduardo Casado Vecino**
TUTORAS:	**Míriam Antón Rodríguez y María Ángeles Pérez Juárez**
DEPARTAMENTO:	**Dpto. de Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**María Ángeles Pérez Juárez**
VOCAL:	**David González Ortega**
SECRETARIO	**Míriam Antón Rodríguez**
SUPLENTE	**Javier Aguiar Pérez**
SUPLENTE	**Mario Martínez Zarzuela**

FECHA:	**julio de 2020**
CALIFICACIÓN:	

Resumen de TFG

En este trabajo se ha llevado a cabo el desarrollo de una aplicación móvil multiplataforma, pues se puede ejecutar tanto en Android como en iOS, de ayuda al aprendizaje del lenguaje de programación utilizado en el desarrollo web JavaScript. La aplicación servirá de ayuda a estudiantes de dicho lenguaje de programación por lo que partirá de la base de que estos estudiantes tienen nociones básicas de programación y sobre todo del trabajo con objetos software.

La aplicación se compone de dos partes bien diferenciadas. Una para que el profesor pueda administrar sus clases y sus alumnos y observar las puntuaciones de estos en los distintos juegos que ofrece la otra parte de la aplicación. Además de los juegos que se mencionan, esta parte de la aplicación, diseñada para el alumno, tiene un apartado donde estos pueden ver la teoría de JavaScript y otro para ver las puntuaciones que llevan en los distintos juegos.

Esta aplicación se ha realizado sobre Flutter, que es un kit de herramientas de Google que utiliza el lenguaje de programación Dart para realizar aplicaciones multiplataforma sobre una única base de código. Esta herramienta se usa en el entorno Android Studio, y posteriormente sirve para ambos sistemas operativos. Además, se ha utilizado Firebase como servicio de base de datos para almacenar la información que genera la aplicación.

Palabras clave: Flutter, Dart, Firebase, Flame, Box2d, JavaScript, desarrollo web, aplicaciones web, e-learning, m-learning

Abstract

In this work, the development of a Multi-platform mobile application has been carried out, since it can be run on both Android and iOS, to help you learn the programming language used in JavaScript web development. The application will help students learn of said programming language, so it will be based on the fact that these students have basic notions of programming and especially of working with software objects.

The application is made up of two distinct parts. One so that the teacher can manage their classes and their students and observe their scores in the different games offered by the other part of the application. In addition to the games mentioned, this part of the application, designed for the student, has a section where they can see the theory of JavaScript and another to see the scores they have in the different games.

This application has been made on Flutter, which is a Google toolkit that uses the Dart programming language to make hybrid applications on a single code base. This tool is used on Android Studio, and it later works for both operating systems. Also, Firebase has been used as a database service to store the information generated by the application.

Keywords: Flutter, Dart, Firebase, Flame, Box2d, JavaScript, web development, web applications, e-learning, m-learning

Agradecimientos

A mis padres, a mi hermano Rodrigo y a mi familia que siempre están ahí para apoyarme con las decisiones que tomo y que me han llevado hasta aquí.

A mis compañeros y amigos por su confianza.

Y a mis tutoras, María Ángeles Pérez Juárez y Míriam Antón Rodríguez por la oportunidad de realizar este proyecto y su gran ayuda para llevarlo a cabo.

Lista de acrónimos

A continuación, se mostrarán una serie de términos que pueden aparecer abreviados como se muestra:

- AOT: Ahead-Of-Time
- API: Application Programming Interface
- ARM: Advanced RISC Machine
- CRUD: Create, Read, Update and Delete
- CSS: Cascading Style Sheets
- FPS: Frames Per Second
- HTML: HyperText Markup Language
- IDE: Entorno de Desarrollo Integrado
- JIT: Just-In-Time
- RAM: Random Access Memory
- SDK: Software Development Kit
- SO: Sistema Operativo
- XAML: eXtensible Application Markup Language
- XML: eXtensible Markup Language

Índice

Índice de ilustraciones y tablas

- Ilustraciones

- Tablas

1. Introducción

1.1. Estructura del documento

El documento está formado por diez capítulos principales.

El presente capítulo aborda los conceptos introductorios del documento. En él se explica cómo está organizado. En la sección de la contextualización se relaciona el proyecto con el marco actual. La sección "Objetivos" explica la razón principal del proyecto. Después se procede a explicar las fases en las que se ha dividido el proyecto y por último se describen las herramientas utilizadas, tanto hardware como software.

En el segundo capítulo se explican las diferentes tecnologías existentes para llevar a cabo el proyecto. En este capítulo se presentarán los diferentes tipos de sistemas operativos móviles, los tipos de aplicaciones, los tipos de desarrollos de aplicaciones y las diferentes herramientas para su desarrollo. Como sección final de este capítulo se realizará una comparativa entre estas últimas.

Los siguientes cuatro capítulos (del capítulo 3 al 6) exponen las diferentes tecnologías que se han utilizado en este desarrollo y de las que es necesario explicar sus características básicas. Estas son Flutter y Dart principalmente, porque son la herramienta de desarrollo utilizada y el lenguaje de programación utilizado. Además, también se expondrán las características principales de la plataforma de servicios Firebase utilizada en el proyecto como base de datos y herramienta de ayuda para la autentificación de usuario. Por último, también se verá el motor de videojuegos para Flutter, Flame y una extensión suya como es el motor de físicas Box2d, herramientas utilizadas también durante el desarrollo.

El séptimo capítulo es el núcleo del documento. En él se describe el desarrollo de la aplicación, pues contiene una descripción técnica y un manual de usuario de la aplicación y por lo tanto es donde más claramente queda reflejado el trabajo realizado.

En el octavo capítulo se realiza un estudio presupuestario de la aplicación teniendo en cuenta aspectos como el tiempo invertido o la complejidad de la aplicación.

El noveno capítulo expone las conclusiones del trabajo desarrollado y propone una serie de líneas futuras para añadir nuevas funcionalidades a la aplicación si se siguiera con el proyecto.

El décimo capítulo recoge las referencias utilizadas durante la realización del presente trabajo y la correspondiente memoria.

1.2. Contextualización

La tecnología cada vez influye más en el día a día de las personas. Casi todo el mundo posee dispositivos electrónicos porque optimizan nuestra vida. Los jóvenes de hoy en día llevan usando la tecnología diariamente desde que tienen memoria, ya que la invención de Internet se produjo antes de que ellos nacieran. Es lógico que la tecnología también se use como apoyo para el estudio, ya que al estar tan familiarizados con ella esta ayuda a mejorar la eficiencia, la productividad y el interés de los jóvenes por las actividades académicas.

La tecnología lleva ayudando a docentes y estudiantes desde hace mucho tiempo, pero la llegada de Internet produjo un cambio en la metodología del estudio gracias al e-learning, que significa aprendizaje a través de Internet. Las principales ventajas del e-learning son la eliminación de las barreras físicas y temporales, ahorro de costes (por ejemplo, en comprar los libros escolares), comunicación constante entre alumnos, docentes y padres, y un aprendizaje más autónomo. Además, aporta flexibilidad y personalización en los procesos de aprendizaje (Semana, 2017).

La web 2.0 y las redes sociales animan a los estudiantes a expresarse y relacionarse, lo que permite aprender de forma interactiva. Además de que la facilidad para compartir contenido que ofrece la web ayuda a los estudiantes a realizar sus actividades sin uso del papel, lo que reduce los costes de la producción de libros y permite acceder a contenido sin tener que moverse de su casa (Semana, 2017).

Al igual que Internet, la llegada de los dispositivos móviles inteligentes ha producido otro cambio en la metodología del aprendizaje. La introducción del m-learning, aprendizaje a través de los dispositivos móviles, se ha producido gracias a los avances en las comunicaciones y en las tecnologías inalámbricas que ha permitido que los dispositivos móviles, cuya motivación inicial fue la de medio transmisor y receptor de una comunicación de voz entre dos personas desde cualquier localización lleguen a ser el dispositivo multifuncional que es actualmente, ya que además de integrar su función primigenia mejorada, integran otras muchas como por ejemplo hacer fotos, marcar la hora o almacenar archivos. Esta evolución ha traído consigo la aparición del “smartphone”, ordenadores de bolsillo con funciones de telefonía móvil pero capaces de realizar otras tareas gracias a las aplicaciones móviles y web (Wikipedia, 2020).

Tal y como se puede observar en la Ilustración 1, el teléfono móvil era el dispositivo más usado para acceder a Internet ya en 2015.

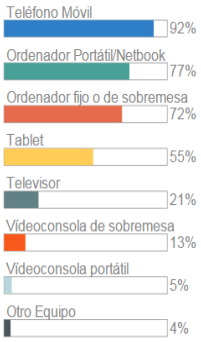


Ilustración 1. Dispositivos más usados para acceder a Internet (López Moreno, 2015)

Las aplicaciones móviles son aquellas que se pueden instalar y ejecutar en los dispositivos móviles. Estas cuentan con la ventaja de que pueden acceder a las características hardware del dispositivo móvil. Las aplicaciones web, en cambio, se ejecutan a través de un tipo de aplicación como es el navegador, que permite el acceso a la web. Este último tipo de aplicación se crea sin tener en cuenta el sistema operativo del smartphone (Lancetalent, 2014).

El uso de estas aplicaciones se ha incrementado tanto que se ha llegado a un punto de eficiencia en la interacción entre usuario y dispositivo que ha convertido al m-learning en una de las tendencias clave dentro de las aplicaciones educativas del e-learning. Las posibilidades en este mercado son muchas gracias a la variedad de dispositivos disponibles y gracias a las ventajas que las aplicaciones conllevan (Ortiz, 2014).

Las aplicaciones en la educación permiten la comunicación en tiempo real entre estudiantes, docentes y padres. También, permiten la distribución de tareas y recursos a través de Internet, ayudando a la superación de barreras geográficas, y además refuerzan el aprendizaje ya que los estudiantes tienen mayor relación con los formatos y medios, como ya se ha comentado previamente.

No existe demasiada documentación científica sobre el uso del m-learning para la enseñanza de la programación, y menos aún para la enseñanza de JavaScript, motivo por el cual resulta interesante desarrollar el presente proyecto. La implementación de una herramienta m-learning presenta muchos retos tecnológicos y pedagógicos. Se trata de ir más lejos del simple acceso a la información, como ya hacen las aplicaciones web, superando las limitaciones propias de los dispositivos móviles como puede ser el reducido tamaño de la pantalla y aprovechando la accesibilidad característica de estos dispositivos (Ortiz, 2014).

1.3. Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación usando como entorno

de desarrollo la herramienta Flutter de Android Studio y como lenguaje de desarrollo Dart. De esta manera la aplicación será multiplataforma y podrá servir, tanto para el sistema operativo Android como para el sistema operativo iOS. Con esta premisa se consigue cubrir la totalidad del mercado actual en cuanto a sistemas operativos móviles.

La aplicación será una herramienta destinada a estudiantes y estará orientada a facilitar el aprendizaje del lenguaje de programación JavaScript. Más concretamente, la aplicación se centrará en reforzar el conocimiento de los alumnos sobre los manejadores de eventos y el trabajo con objetos en JavaScript. La aplicación utiliza las potentes ventajas que proporcionan los dispositivos móviles frente a los métodos más tradicionales del e-learning.

1.4. Fases del Proyecto

La realización del presente proyecto se ha llevado a cabo en varias fases:

- Fase de documentación: Como primer paso, se ha intentado comprender cómo funciona tanto Dart como Flutter antes de realizar un proyecto de tal envergadura. Con respecto a Dart, al tratarse de un lenguaje de programación orientado a objetos y ser relativamente similar a JavaScript no ha resultado muy complejo familiarizarse con él. Por otra parte, el aprendizaje de Flutter sí que ha llevado algo más de tiempo porque para el autor del proyecto era algo totalmente nuevo.
- Fase de investigación: Como paso previo a la realización como tal de la aplicación, se ha hecho una búsqueda de tipos de juegos que pudieran resultar interesantes para luego desarrollarlos en la aplicación.
- Fase de análisis: Antes de la implementación de la aplicación se ha procedido a realizar un análisis de sus funcionalidades, estableciendo los requisitos y los casos de uso de la misma y atendiendo a las sugerencias de las tutoras, puesto que son docentes con amplia experiencia en la enseñanza del desarrollo web.
- Fase de implementación: Desarrollo de la aplicación mediante el uso de diferentes herramientas como Flutter, Dart o Firebase.
- Fase de instalación: Instalación en diferentes dispositivos móviles Android para comprobar su correcto funcionamiento, así como la responsividad de la interfaz de usuario de la aplicación.

1.5. Medios utilizados para el desarrollo

1.5.1. Medios software

Este proyecto se ha realizado en el IDE (Entorno de Desarrollo Integrado) Android Studio en su última versión ya que así lo recomienda web oficial de Flutter. Esta herramienta se ha descargado de su web oficial. También se han utilizado los plugins de Android Studio,

Flutter y Dart, que se han descargado desde el programa como tal, ambos en sus últimas versiones.

Además de esto, se han utilizado varios plugins de Flutter a mayores, descargables desde el terminal de Flutter y que se pueden ver en la web pub.dev:

- cloud_firestore: Plugin necesario para acceder a la base de datos.
- flutter_pdfview: Plugin necesario para visualizar un archivo PDF.
- path_provider: Plugin necesario para acceder al sistema de ficheros dentro del SO (Sistema Operativo).
- firebase_auth: Plugin necesario para realizar la autentificación de usuario.
- firebase_database: Plugin auxiliar a cloud_firestore.
- connectivity: Plugin necesario para comprobar la conexión a Internet.
- flame: Plugin necesario para realizar uno de los juegos.
- dartjsengine: Plugin necesario para poder interpretar código JavaScript en la aplicación.
- jsparser: Plugin auxilar a dartjsengine.
- firebase_storage: Plugin necesario para acceder al PDF guardado en la nube.
- firebase_core: Plugin auxiliar para Firebase.
- firebase_analytics: Plugin auxiliar para Firebase.

1.5.2. Medios hardware

Se han utilizado dos portátiles para la realización de este proyecto:

- ASUS X555LDB de 4GB de RAM (Random Access Memory), x64, i5, con Windows 10.
- MEDION P6645 MD61340 de 8GB de RAM, x64, i5, con Windows 10.

Además, el testeo de la aplicación se ha hecho sobre los siguientes dispositivos Android:

- XIAOMI REDMI NOTE 8 PRO de 6GB de RAM, con Android 10.
- HUAWEI Y5 II de 1GB de RAM, con Android 5.1.
- SAMSUNG S9+ de 6GB de RAM, con Android 10.
- SAMSUNG GALAXY TAB A6 de 2GB de RAM, con Android 8.1.0.
- SAMSUNG GALAXY TAB S6 de 8GB de RAM, con Android 10.

2. Tecnologías existentes

2.1. Sistemas operativos móviles

En primer lugar, se va a realizar una descripción breve de los principales sistemas operativos móviles en la actualidad que son Android, iOS y Windows 10 Mobile.

Android

Este sistema operativo móvil fue desarrollado por Google y se lanzó en 2008. Como características principales cabe destacar que es un sistema operativo de código abierto ya que su núcleo se basa en el kernel de Linux y otros softwares de código abierto. Además, está basado en la arquitectura hardware ARM (Advanced RISC Machine). Usa SQLite para el almacenamiento de datos, además soporta Java, HTML5, Adobe Flash Player… También, permite usar varias aplicaciones a la vez (AndroidOS, 2012).

Su arquitectura se compone de 4 capas que, en orden ascendente, son las siguientes (Android, 2019; Universidad Politécnica de Valencia, 2017):

- Kernel de Linux: Es la base de la plataforma. Permite aprovechar las funciones de seguridad, el manejo de memoria, el multiproceso, el soporte de drivers. Actúa como capa de abstracción entre el software y el hardware subyacente.

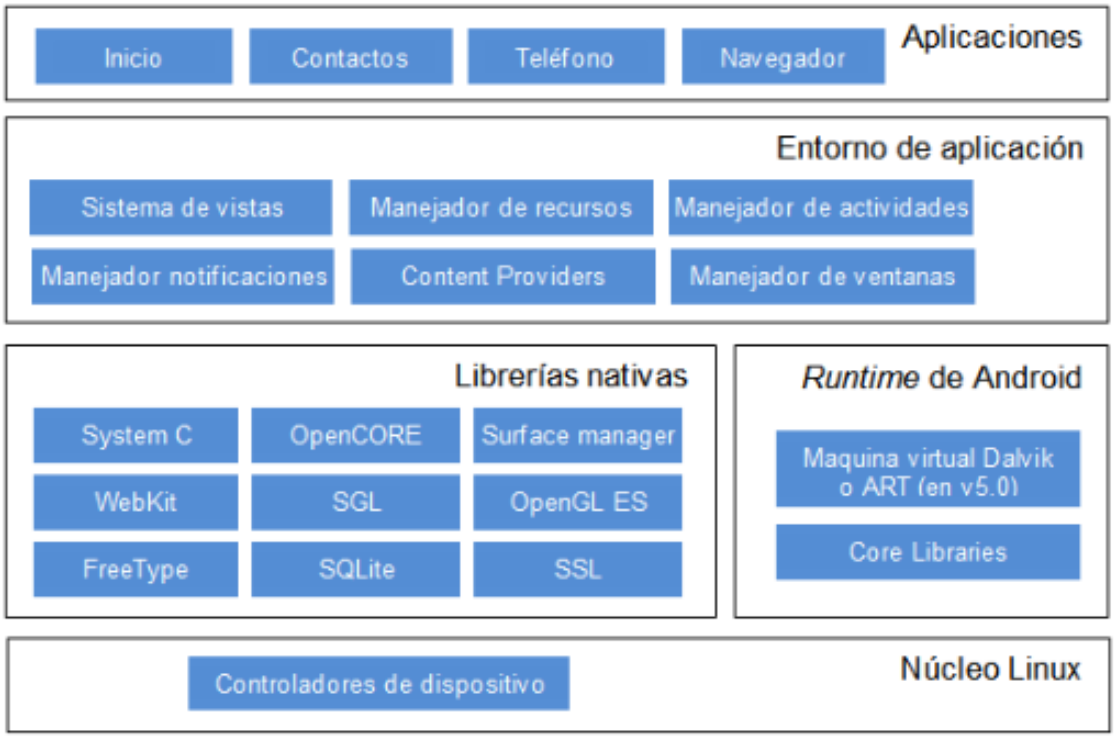


Ilustración 2. Arquitectura de Android (Universidad Politécnica de Valencia, 2017)

- Runtime de Android: Está basada en el concepto de máquina virtual de Java. Facilita la optimización de recursos. Cada aplicación corre en su propio proceso Linux con su propia instancia en este Runtime.
- Librerías nativas: En un mismo nivel de abstracción que el Runtime, son un conjunto de librerías en C/C++ usadas en varios componentes de Android. Están compiladas en el código nativo del procesador. Algunas librerías son: System C library, Media Framework, SQLite, etc.
- Marco de trabajo de la API (Application Programming Interface) de Java: Todo el conjunto de funciones del sistema operativo de Android está disponible mediante APIs escritas en Java.
- Aplicación: La capa más externa formada por los distintos programas que permiten realizar tareas al usuario como el correo electrónico, el navegador, etc.

iOS

Este sistema operativo móvil fue desarrollado por Apple y se lanzó en 2007. iOS deriva del sistema operativo macOS, también de Apple, por lo que es un sistema operativo tipo Unix. A diferencia de Android, no se permite la instalación de iOS en hardware de terceros, por lo que solo está disponible en los propios dispositivos de Apple. (Wikipedia, 2020).

La arquitectura de este sistema operativo está formada por 4 capas diferenciadas. Se presentan a continuación siguiendo la misma lógica que se ha seguido en el caso de Android (Carltics, 2015):

- Núcleo del SO: Basado, como ya se ha expuesto antes, en Unix. En esta capa se encuentran elementos de seguridad, procesos y ficheros.
- Servicios del Núcleo: Capa donde se encuentran los principales servicios del dispositivo, los cuales son usados por las aplicaciones.
- Media: Capa basada en C/ Objective C que contiene las tecnologías que dan acceso a los ficheros multimedia como audio, vídeo, etc.
- Cocoa Touch: Es la capa más exterior, la capa donde los usuarios interaccionan con las aplicaciones.

Windows 10 Mobile

Este sistema operativo móvil desarrollado por Microsoft se lanzó en el año 2015 para suceder a Windows Phone. Se ideó para competir contra Android e iOS, pero al no poder conseguirlo, se dejaron de sacar actualizaciones a partir de 2016 y Microsoft abandonó el desarrollo de negocio móvil (Wikipedia, 2020).

A la hora de tomar la decisión sobre cómo desarrollar la aplicación se ha analizado la situación actual de los sistemas operativos móviles. En la Ilustración 3 se puede apreciar

cómo ha evolucionado la cuota de mercado de los sistemas operativos móviles a nivel mundial en este último lustro.

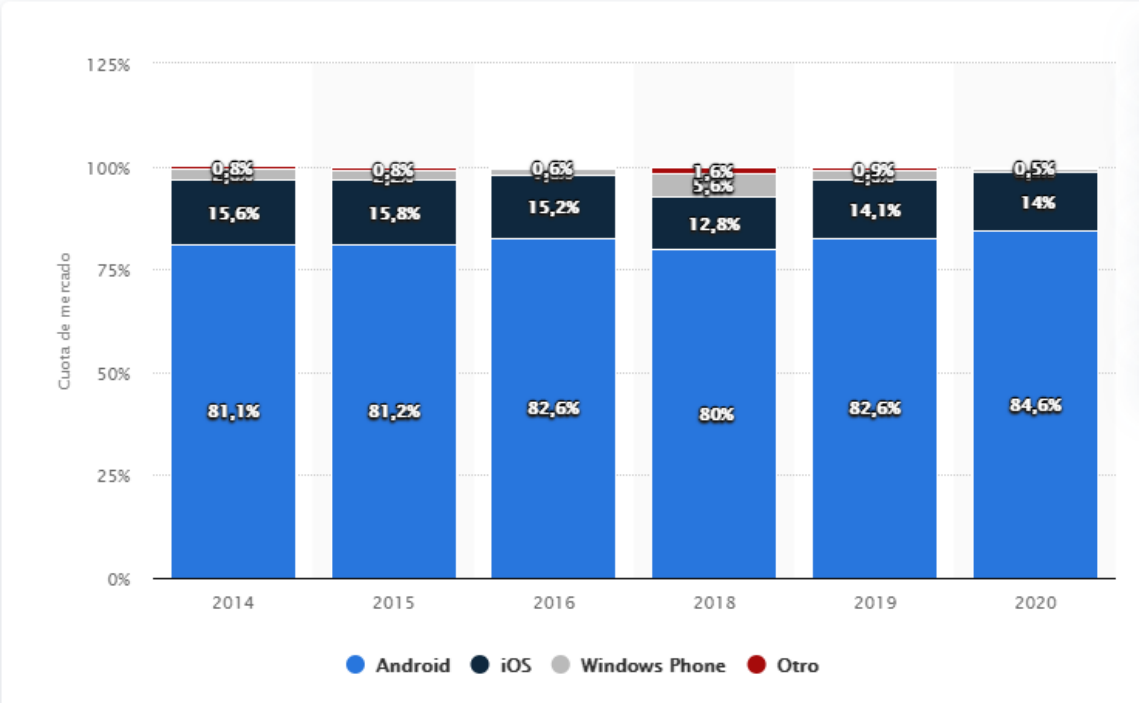


Ilustración 3. Evolución del mercado de dispositivos móviles en los últimas 6 años (Statista, 2019)

Es posible apreciar la gran hegemonía que tiene Android sobre sus competidores, con más de un 80% y aumentado actualmente. Esto es debido principalmente a que es un sistema operativo de código abierto como ya se ha indicado previamente.

También destaca iOS en segunda posición con un buen nicho de mercado, en Norteamérica, donde ocupa entorno a un 40% del mercado, lo que le hace tener un 15% aproximadamente de la cuota de mercado mundial.

Atendiendo a los datos anteriormente expuestos, se ha procedido a realizar la aplicación de manera que sirva tanto para Android como para iOS, ya que ocupan casi la totalidad del mercado. A parte de por esa razón, otra razón de peso para esta elección, derivada de la anterior, ha sido que la documentación sobre cómo realizar aplicaciones es mucho más extensa para estos dos sistemas operativos que para otros como Windows Phone, por ejemplo.

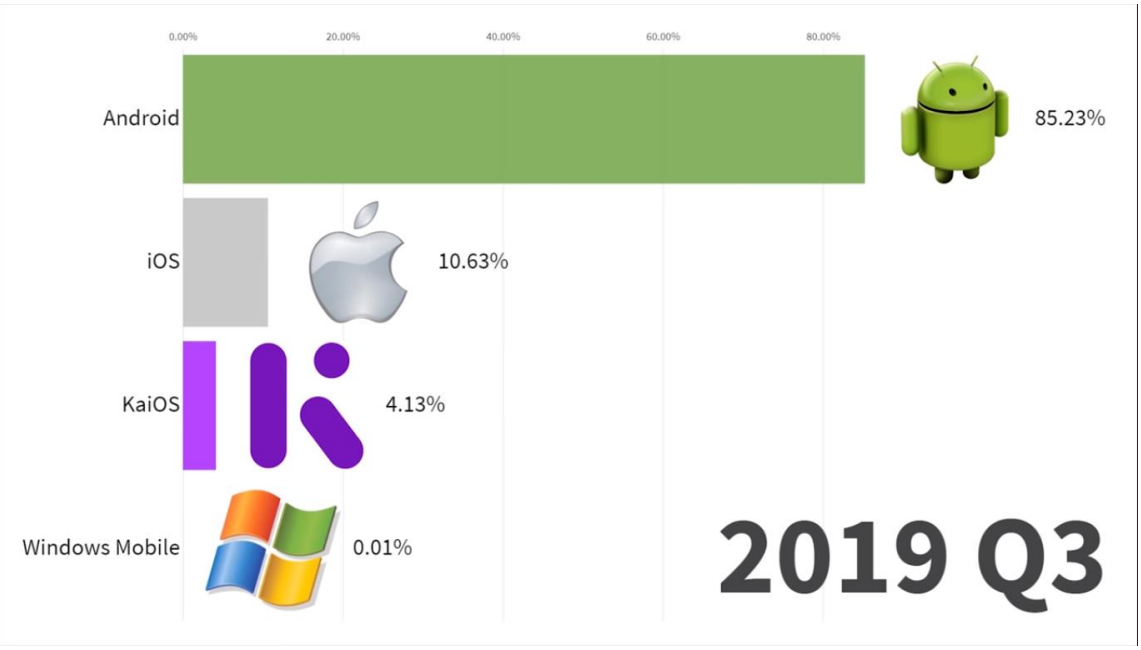


Ilustración 4. Cuota de mercado de los sistemas operativos móviles en el último cuatrimestre de 2019 (Data is Beautiful, 2019)

2.2. Tipos de aplicaciones

Una vez vistos y elegidos los sistemas operativos destino de la aplicación, se procederá ahora a hacer un estudio de las aplicaciones que pueden ejecutarse en los sistemas operativos móviles. Se definirá el término aplicación y se contextualizará para los dispositivos móviles. También se definirán los tipos de aplicaciones existentes en plataformas móviles y las ventajas y desventajas que estos tienen para poder escoger la mejor opción para la aplicación. Como ya se ha visto antes, se pretende hacer un desarrollo tanto para Android como para iOS.

Una aplicación es un programa creado para llevar a cabo o facilitar la realización de una tarea en un dispositivo electrónico. No todos los programas son aplicaciones, como por ejemplo los sistemas operativos, que son programas con un propósito más general y no tan concreto como una aplicación (Sistemas, 2016).

Las aplicaciones nacen de alguna necesidad concreta, y se usan para facilitar o permitir la ejecución de ciertas tareas en las que se ha detectado una cierta necesidad. Las aplicaciones también pueden responder a necesidades lúdicas o laborales. Se suele decir que para cada problema hay una solución, y en informática, para cada problema hay una aplicación (Sistemas, 2016).

Este documento se centrará en el estudio de las aplicaciones que se pueden acceder a través de los dispositivos móviles. Dentro de ellas se pueden diferenciar dos tipos atendiendo a dónde se ejecutan:

- Aplicaciones que se ejecutan desde el propio dispositivo móvil.
- Aplicaciones que se ejecutan a través del navegador.

Si el criterio es la forma de desarrollarlas, la clasificación estaría formada por cuatro tipos diferentes de desarrollo:

- Desarrollo nativo.
- Desarrollo generado.
- Desarrollo web.
- Desarrollo híbrido.

Cada uno de estos cuatro tipos es interesante para unas condiciones determinadas, dependiendo de los recursos disponibles, el tiempo disponible o las necesidades mismas de la aplicación.

2.2.1. Aplicaciones nativas

Son las aplicaciones que solo han sido desarrolladas para un sistema operativo. Por ello el lenguaje utilizado es el nativo de cada sistema operativo. En el caso de Android, sus lenguajes son Java o Kotlin, mientras que, para iOS, sus lenguajes son Objetive-C y Switf.

Ventajas (Peña, 2018):

- Mejor rendimiento posible: Al realizar el desarrollo en el lenguaje nativo, se aprovecha al máximo el acceso a los recursos del dispositivo.
- Se mejora la experiencia de usuario: Al aprovechar al máximo los recursos, la fluidez de la aplicación es la más óptima posible de cara al usuario en lo que se refiere a retardos en cargas o accesos por ejemplo.
- Permite la ejecución en segundo plano.
- Permite usar Notificaciones Push.

Desventajas (Peña, 2018):

- Se debe aprender todo un lenguaje de programación que solo te sirve para esa plataforma, por lo que es algo ineficiente.
- Son más costosos que otros desarrollos: Los demás desarrollos permite desarrollar la aplicación de una vez y esta sirve para varias plataformas. Si se desarrolla la aplicación de forma nativa y se quiere realizar para varias plataformas, esto significa que se tiene que realizar el desarrollo una vez por cada plataforma.
- Código no reutilizable para otras plataformas.
- Necesidad de una App Store para su publicación.

2.2.2. Aplicaciones generadas

Son el tipo de desarrollo más joven. Nacen de la necesidad de realizar aplicaciones nativas con un desarrollo común. Se realizan con herramientas como Xamarin, Flutter o React Native, por ejemplo. Las aplicaciones se generan realizando el desarrollo en el lenguaje determinado de la herramienta y después la herramienta genera la aplicación nativa a partir de ese desarrollo.

Ventajas:

- Código multiplataforma: Aprender el lenguaje de la herramienta sí permite poder desarrollar aplicaciones en varios sistemas operativos.
- Se pueden distribuir por las Apps Stores de todos los sistemas operativos en los que se generan.
- Al generar aplicaciones nativas, el rendimiento es bastante parecido a estas.
- Desarrollo menos costoso, ya que se generan aplicaciones para varias plataformas a la vez.

Desventajas:

- Necesidad de aprender otro lenguaje de programación a mayores: Aunque el lenguaje sirva para varias plataformas, hay que invertir tiempo en su aprendizaje igualmente.

2.2.3. Aplicaciones que se ejecutan a través del navegador (web)

Son aplicaciones a las que se accede a través del navegador, por lo que es necesario tener acceso a Internet para poder ejecutarlas. Se pueden acceder a ellas a través de cualquier dispositivo independientemente del sistema operativo móvil. El lenguaje utilizado para el desarrollo también es independiente, siendo HTML (HyperText Markup Language), CSS (Cascading Style Sheets) y JavaScript los más utilizados.

Ventajas:

- Independientes del sistema operativo móvil del dispositivo.
- Necesitan menos recursos del dispositivo, como, por ejemplo, menos memoria.
- No se tienen que publicar en ninguna App Store.
- Fácilmente escalables, y siempre actualizadas.
- Menos tiempo de desarrollo que en el caso de una aplicación nativa.

Desventajas:

- Es necesario disponer de un navegador móvil y tener acceso a Internet, por lo que no son siempre accesibles.
- No pueden operar en segundo plano, aunque el navegador sí pueda.
- Aunque necesitan menos recursos, la velocidad de la conexión a Internet puede no ser la adecuada y provocar lentitud en la interacción con el usuario.

2.2.4. Aplicaciones híbridas

Se trata de aplicaciones a medio camino entre las nativas y las web. Se desarrollan con los lenguajes web (HTML, CSS y JavaScript) y usan un contenedor nativo como PhoneGrap o Cordova que permite acceder a los recursos del dispositivo de una forma neutral respecto al dispositivo, por lo que es independiente del sistema operativo también (InnovaAge, 2011).

Ventajas:

- Independencia del sistema operativo utilizado.
- El lenguaje de programación es también independiente del sistema operativo.
- Se pueden publicar en las App Stores.
- Pueden acceder a los recursos hardware del dispositivo, siendo esta su principal novedad con respecto a las aplicaciones web.

Desventajas:

- No se tiene acceso a todos los recursos del hardware.
- Experiencia de usuario más cercana a las aplicaciones web.
- Se necesita conexión a Internet para poder acceder a ellas.
- Rendimiento menor que en el caso de una aplicación nativa.

Una vez vistas todas las posibles aplicaciones y su forma de desarrollo, se procede a decidir cuál es la más conveniente para el desarrollo de la aplicación objeto del presente Trabajo Fin de Grado. Como se quiere desarrollar una aplicación tanto para Android como para iOS, la opción de desarrollo nativo resulta muy costosa porque habría que realizar dos aplicaciones en dos lenguajes diferentes.

La opción de aplicación web, tampoco es interesante, pues se pretende sacar el máximo rendimiento al dispositivo, para aprovechar sus funcionalidades. La opción escogida, por tanto, será la de desarrollo generado, pues permite realizar aplicaciones nativas aprovechando así los recursos hardware tanto para Android como para iOS con un solo desarrollo.

2.3. Herramientas de Desarrollo de aplicaciones nativas multiplataforma

El siguiente paso será realizar un estudio de las herramientas que generan aplicaciones generadas más importantes. En concreto se presentarán las dos opciones más conocidas mostrando sus características y funcionalidades principales y se compararán en función de su ventajas y desventajas con el objetivo final de poder escoger la mejor opción para el desarrollo. Las herramientas que se presentarán serán Flutter y Xamarin.

2.3.1. Flutter

Es un SDK (Software Development Kit) de aplicaciones móviles para Android e iOS sobre Android Studio con una única base de código. Está desarrollado por Google, es de código abierto y su primera versión estable fue lanzada en 2018. El lenguaje que utiliza Flutter para desarrollar las aplicaciones se denomina Dart, y es un lenguaje orientado a objetos también desarrollado por Google (Flutter, 2020).

Ilustración 5. Logo de Flutter

2.3.2. Xamarin

Ilustración 6. Logo de Xamarin

Lanzado en 2011 como Mono, fue la primera herramienta disponible para realizar aplicaciones nativas para varias plataformas con una única base de código. En 2016 fue adquirida por Microsoft que la adaptó como una herramienta de desarrollo descargable dentro de Visual Studio. El lenguaje de programación común utilizado es C#. Pertenece a Microsoft y puede utilizarse tanto en Windows, como en Linux como en macOS. También es un framework de código abierto, por lo que su comunidad, también es muy amplia.

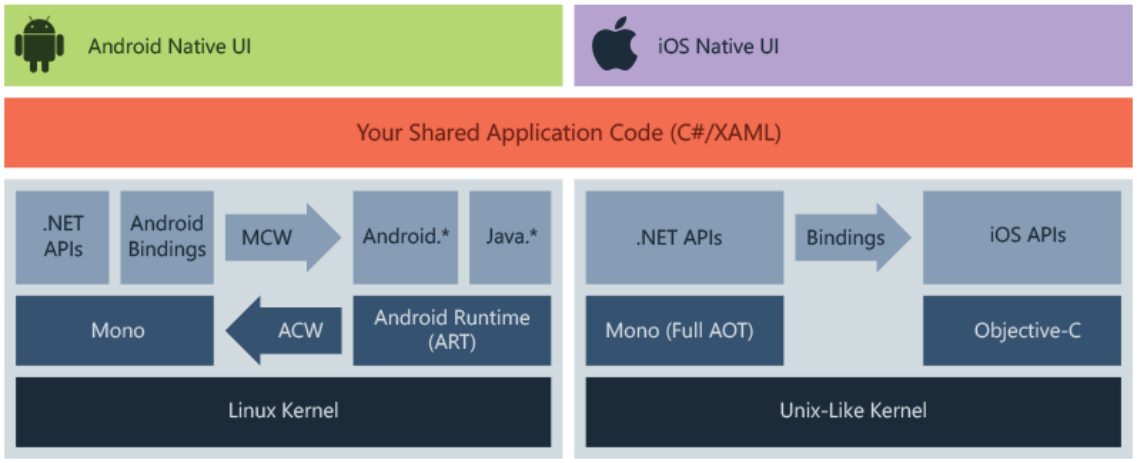


Ilustración 7. Arquitectura de Xamarin (Xamarin Docs, 2019)

2.4. Comparativa Xamarin vs Flutter

Una vez definidas ambas herramientas se realizará una pequeña comparativa entre ellas(Codemagic, 2020; Imagina Formación, 2020; Matos, 2019).:

- Xamarin se lanzó en 2011, mientras que como ya se ha comentado en la contextualización de Flutter, esta herramienta nació mucho más tarde, en el 2018, por este motivo Xamarin es una herramienta con una mayor comunidad en Internet.
- Xamarin utiliza C# como lenguaje de programación, el cual está mucho más globalizado que el utilizado por Flutter (Dart), nuevamente porque C# es un lenguaje de programación mucho más antiguo que Dart. Además de que C# se ha utilizado mucho en el desarrollo web y es la base del .NET Framework de Microsoft.
- En materia de arquitectura, en Xamarin se puede hacer uso de la API nativa de cada plataforma mediante C#, mientras que en Flutter para realizar esta misma operación hay que operar sobre el código nativo de cada plataforma. No obstante, esta operación no debería ser necesaria en Flutter, ya que contiene todos los requerimientos de la plataforma en sí mismo.
- A la hora de realizar la instalación, Xamarin requiere que se instale Visual Studio primero y después el propio Xamarin como una herramienta dentro del mismo IDE. Mientras que la instalación de Flutter no necesita de ningún IDE específico, aunque desde su página oficial se recomienda trabajar con Android Studio.
- En cuanto a la realización de la aplicación, como ya se ha explicado antes, al existir una mayor comunidad en Xamarin, esto facilita la resolución de errores de código típicos gracias a la colaboración de dicha comunidad. Pero a la hora de realizar la depuración y el testeo mediante la herramienta “Hot Reload” de Flutter, la cual se explicará más adelante, es mucho más fácil para el desarrollador realizar estas tareas en el caso de Flutter.
- En lo relacionado al tamaño de la aplicación, ambas herramientas producen aplicaciones más pesadas en relación a las aplicaciones hechas con código nativo, pero Flutter es aún más pesado que Xamarin porque contiene más código binario.

De cara al futuro, Flutter tiene un gran camino por recorrer, por sus características y su origen, ya que Google es el precursor de Android y Android junto con iOS, dominan el mercado. Mientras que Xamarin sin embargo, aunque sigue creciendo, es una opción para los desarrolladores centrados en las tecnologías de Microsoft, las cuales no tienen un nicho de mercado dentro de los sistemas operativos móviles. Para realizar la aplicación de este trabajo, por tanto, se ha optado por Flutter. A continuación, se entrará más en detalle en esta tecnología.

3. Flutter

En esta sección se detallarán los aspectos más importantes de Flutter, como son su arquitectura, su funcionamiento o sus características principales. Es importante realizar esta exposición para que el lector se documente correctamente sobre la herramienta que se ha usado para realizar la aplicación.

3.1. Definición e Historia

Como ya se ha visto antes, Flutter es un kit de herramientas desarrollado por Google en 2018 para realizar aplicaciones nativas tanto en Android como en iOS a través de una única base de código, usando el lenguaje de programación orientado a objetos de Google, denominado Dart.

La primera versión de esta herramienta fue presentada en el Dart Developer Summit de 2015 y se denominó "Sky". En febrero de 2018, En el Mobile World Congress de Barcelona se anunció la primera versión beta de Flutter. En diciembre de 2018, Google anunció la primera versión estable. En septiembre de 2019 se disponía ya de la versión 1.9 (Stolar, 2019).

Al ser una tecnología nueva todavía está en crecimiento, aunque ya se encuentra en el top 20 de las tecnologías más usadas en Github a fecha de diciembre de 2019. Además, esto ha provocado que Dart sea el lenguaje de programación que más ha crecido en Github (Stolar, 2019).

3.2. Arquitectura

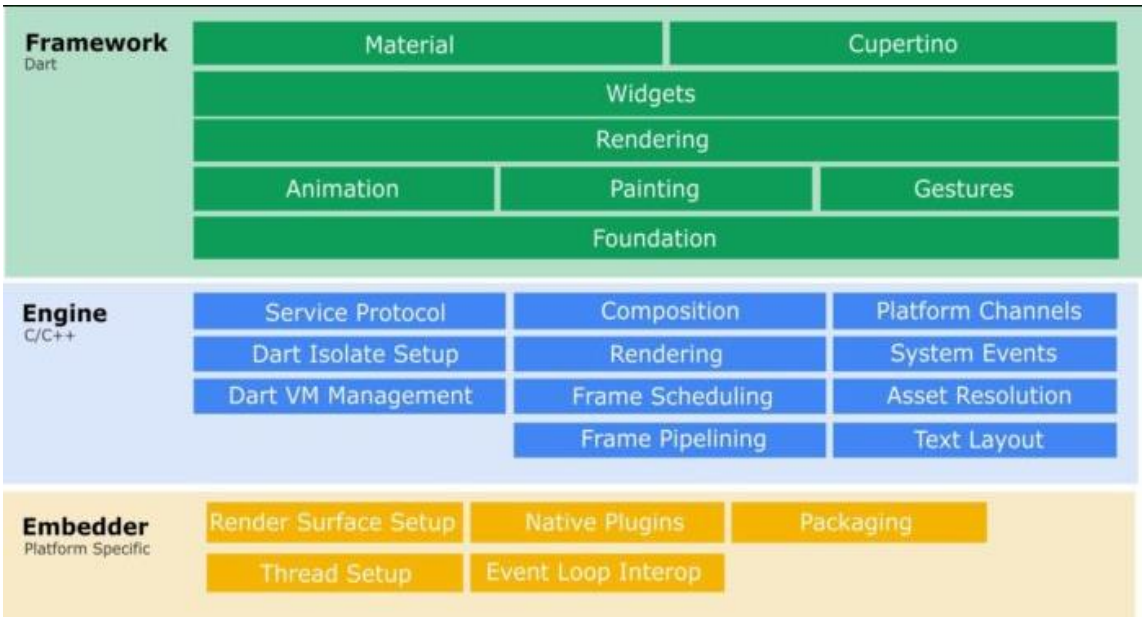


Ilustración 8. Arquitectura de Flutter(Alibaba Tech, 2018)

En la Ilustración 8 se puede apreciar cómo está constituida la herramienta. Está formada por 3 capas bien diferenciadas. Las capas superiores se utilizan más que las capas inferiores.

Framework

Es la capa más superficial, está escrita en lenguaje Dart y se compone de una serie de librerías que actúan como subcapas dentro de esta capa.

La capa más superficial es la interfaz de usuario. Para usar el estilo de Android, se usa la librería Material, mientras que para iOS se usa la librería Cupertino.

La siguiente capa está compuesta por todos los widgets nativos de Flutter. Los widgets son la unidad fundamental de Flutter. Simplificando se puede decir que son todas las clases que existen en Flutter. Representan desde la clase de la aplicación misma, hasta elementos simples de la interfaz, como botones, texto, etc (JavaTPoint, 2019).

Después se ubica la librería Rendering, que se encarga de la representación y el diseño de los distintos widgets de la aplicación en las distintas pantallas (JavaTPoint, 2019).

Inmediatamente debajo se sitúan las librerías Animation, Painting y Gestures encargadas de implementar las animaciones, los colores y la interacción de los widgets con el usuario (JavaTPoint, 2019).

Por último, en esta primera capa, se encuentra la librería Foundation. Esta librería está formada por las clases y funciones de nivel más bajo que son necesarias para desarrollar la aplicación (JavaTPoint, 2019).

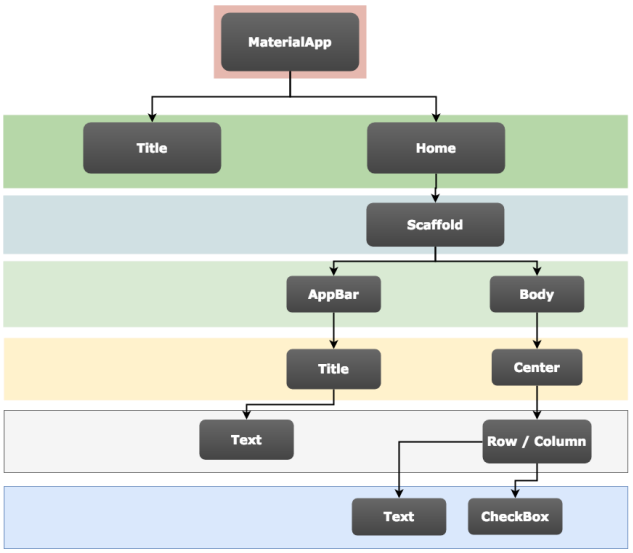


Ilustración 9. Ejemplo de esquema de Widgets (Deepak, 2018)

Engine

Es la capa intermedia de la herramienta. Implementa librerías escritas en C/C++ que controlan los gráficos, las animaciones, la lecturas y escritura de archivos, etc. También contiene una máquina virtual de Dart para poder ejecutar las aplicaciones y la librería de Google Skia, la cual se utiliza para el controlar el diseño y los gráficos a bajo nivel (JavaTPoint, 2019).

Embedder

Es la capa más inferior y está compuesta por código específico de cada plataforma (Flutter, 2019).

3.3. Características principales

Como ya se ha comentado antes en la arquitectura, todo en Flutter es un widget.

Un widget puede ser un elemento de la interfaz (un botón, por ejemplo), pero, además, también puede ser un elemento de estilo (un tipo de fuente, por ejemplo), un aspecto de diseño, (el padding, por ejemplo) y muchas más cosas.

Cada widget pertenece a un widget padre y hereda sus propiedades. En la parte superior está el widget raíz. Cada widget, por lo tanto, puede contener otros widgets de un solo propósito que juntos pueden realizar efectos poderosos (Flutter, 2019).

Las características únicas de un widget se definen mediante la implementación de la función build dentro de la implementación del propio widget. Esta función devuelve un árbol de widgets que representa la parte del widget de la interfaz de usuario (Flutter, 2019).

Ilustración 10. Comparativa entre Widgets Stateless y Stateful (Amateur Coder, 2020)

La función build debe estar libre de efectos secundarios y solo encargarse de la declaración de los elementos del widget. Cuando se ejecute esta función esta devuelve un nuevo árbol de widgets independientemente de lo que haya devuelto antes. Flutter se encarga de comparar el nuevo árbol con el anterior y determinar si ha habido modificaciones. Esto es interesante, por ejemplo, para añadir o quitar algún botón de la interfaz (Flutter, 2019).

Para controlar la interacción con el usuario se utilizan widgets que puedan cambiar sus características. Estos widgets se denominan Stateful. Un widget es Stateful si hereda la clase StatefulWidget y almacena su estado mutable en una subclase de la clase State. Los widgets que heredan la clase StatelessWidget no cambian su estado (Flutter, 2019).

Cada vez que se deba cambiar el estado de un widget, es necesario llamar a la función setState(). De este manara, se vuelve a ejecutar el método build de State (Flutter, 2019).

Esta manera de operar, teniendo objetos State y Widget separados permite que otros widgets traten de igual manera tanto a los StatefulWidgets como a los StatelessWidgets. El padre puede crear una nueva instancia del hijo sin perder el estado de este (Flutter, 2019).

Otra característica importante de Flutter es el “Hot Reload”. Esta herramienta permite realizar cambios en el código de la aplicación mientras esta se ejecuta. Una vez hechos los cambios pulsamos el botón pertinente, y los cambios hechos se aplican en muy poco tiempo (Amateur Coder, 2020).

3.4. Editores de texto compatibles

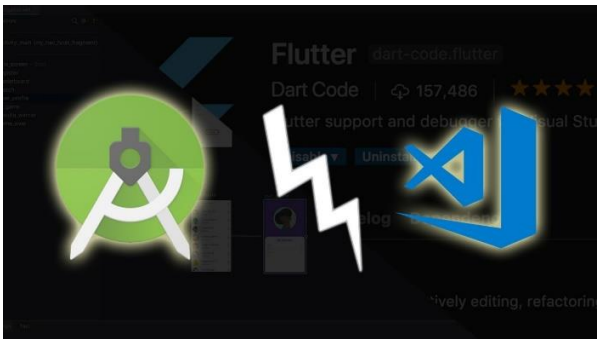


Ilustración 11. Logos de Android Studio y Visual Studio Code

Para desarrollar aplicaciones con Flutter no es obligatorio utilizar ningún IDE en concreto. Se pueden realizar aplicaciones con cualquier editor de texto y la herramienta Flutter en línea de comandos. Aunque es recomendable usar editores como Android Studio/IntelliJ o Visual Studio Code porque contienen una serie de plugins descargables (un compilador para Dart, otro para un análisis de código Dart y el plugin de desarrollo de Flutter) que mejoran notablemente la experiencia de cara al desarrollador (Flutter, 2019).

Para acometer este proyecto se ha utilizado como IDE Android Studio que es el oficial para desarrollar aplicaciones en Android. La web oficial de Flutter recomienda la utilización de este IDE por encima de cualquier otro y esta es la razón por la que se ha escogido este entorno de trabajo.

Este programa se lanzó en 2013 y su función principal, como ya se ha comentado, es desarrollar aplicaciones en lenguaje Android. Está disponible para las últimas versiones de Windows, GNU/Linux o macOS. (Android Studio, 2020).

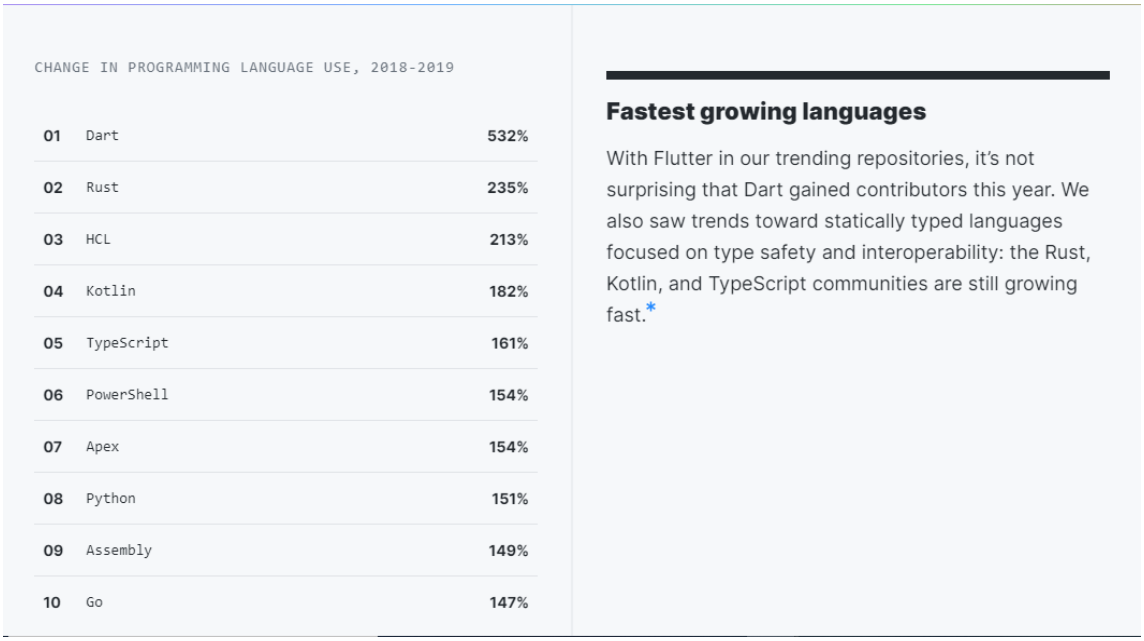
4. Dart

Después de contextualizar Flutter, se procederá a contextualizar el lenguaje de programación que esta herramienta utiliza, denominado Dart.

4.1. Definición e Historia

Es el lenguaje de programación orientado a objetos de código abierto anunciado por Google en octubre de 2011. En noviembre de 2013, dos años más tarde de anunciarse se lanzó su primera versión estable, la versión 1.0. En agosto de 2018 salió la versión 2.0 que incluye un fuerte sistema de tipos (Dart, 2020).

En la actualidad la versión más reciente es la 2.7 lanzada en diciembre de 2019. En ese mismo año ha sido, como ya se dijo previamente, el lenguaje que más ha crecido en GitHub en número de contribuciones. Esto es debido a su uso en Flutter. Flutter es una tecnología creciente en uso, lo que provoca que Dart crezca con ella. No obstante, está lejos de ser unos de los lenguajes más utilizados todavía (Github,2019).



CHANGE IN PROGRAMMING LANGUAGE USE, 2018-2019

01	Dart	532%
02	Rust	235%
03	HCL	213%
04	Kotlin	182%
05	TypeScript	161%
06	PowerShell	154%
07	Apex	154%
08	Python	151%
09	Assembly	149%
10	Go	147%

Fastest growing languages

With Flutter in our trending repositories, it's not surprising that Dart gained contributors this year. We also saw trends toward statically typed languages focused on type safety and interoperability: the Rust, Kotlin, and TypeScript communities are still growing fast.*

Ilustración 12. Lenguajes de programación que más han crecido en 2019 (Github, 2019)

4.2. Objetivos de su lanzamiento

El principal objetivo de Dart es ofrecer mejor soporte en algunos aspectos donde se ha visto que los lenguajes de programación orientada a objetos más usados como Java pueden no ser tan útiles (Dart, 2020).

Además, está optimizado en relación las necesidades de la creación de interfaces de usuario de las aplicaciones en múltiples plataformas. Por otra parte, es fácil de aprender porque tiene una sintaxis muy parecida a los principales lenguajes de programación orientada a objetos (Dart, 2020).

4.3. Características principales y ventajosas para Flutter

4.3.1. Tipos de compilación en Dart.

Hay que tener en cuenta que, históricamente, los lenguajes de programación se pueden dividir en dos tipos, compilados e interpretados, atendiendo a cómo y cuándo se realiza la traducción de código de alto nivel al código máquina. Los del primer tipo son compilados para producir programas de código máquina, que puede ser ejecutados directamente por el hardware. Por su parte, los lenguajes interpretados son ejecutados por un intérprete sin necesidad de producir código máquina previamente (Leler, 2017).

Una máquina virtual es un avanzado intérprete que hace las veces de máquina hardware en software. Una máquina virtual hace más fácil llevar un lenguaje de programación a las nuevas plataformas hardware. En ese caso, frecuentemente, después de la compilación se produce un programa en un lenguaje de programación intermedio antes de ejecutarse en la máquina virtual. Por ejemplo, Java es compilado y se produce un programa en el lenguaje bytecode y después se ejecuta en la máquina virtual de Java (Leler, 2017).

Además, en la actualidad se puede hablar de dos tipos, compiladores JIT (Just-In-Time) y compiladores AOT (Ahead-Of-Time). Los compiladores JIT se ejecutan mientras lo hace el programa también, compilando el programa dinámicamente, mientras que los compiladores AOT son los compiladores originales que se ejecutan antes de ejecutar el programa. Típicamente se relaciona a los lenguajes de programación compilados con los compiladores AOT, mientras que los interpretados usan los compiladores JIT (Leler, 2017).

La compilación AOT se realiza en tiempo de desarrollo y retarda mucho los ciclos entre realizar un cambio en el código y ejecutar el programa, pero la ejecución del programa es más rápida. En contraposición, la compilación JIT provoca un retardo en los ciclos mucho menor, pero la ejecución del programa es mucho más lenta porque mientras se ejecuta el programa, también se compila (Leler, 2017).

Dart soporta ambos tipos de compilación, tanto AOT como JIT, por lo que se aprovecha de sus ventajas. La compilación JIT es usada por Dart en fase de desarrollo, por lo que el compilador es muy rápido. Cuando se desea ejecutar el programa, sin embargo, se debe realizar la compilación AOT. Por lo tanto, Dart tiene unos ciclos de desarrollo rápido y una ejecución también rápida (Leler, 2017).

4.3.2. Tipos de ejecución de un programa de Dart

Además, la ejecución de un programa realizado con Dart se puede realizar de varias maneras. Dart puede ser compilado por JavaScript gracias al compilador dart2js, por lo que se puede ejecutar en los navegadores convencionales. Esto quiere decir que, de alguna forma, se puede reusar el mismo código para realizar aplicaciones web y aplicaciones nativas (Leler, 2017).

El SDK de Dart viene además con una máquina virtual independiente, por lo que Dart se puede ejecutar en el entorno de línea de comandos. El lenguaje intermedio utilizado es el mismo lenguaje Dart (Dart, 2020; Leler, 2017).

La última manera de ejecutar un programa escrito en Dart llegó con la versión 2.6 de Dart y permite ejecutar Dart nativamente con el compilador dart2native (Dart, 2020).

La compilación AOT evita el llamado “puente JavaScript”. Cuando los lenguajes dinámicos, como JavaScript, necesitan operar con el código nativo de la plataforma, se comunican a través de un puente que provoca cambios de estado que se tienen que almacenar, por lo que puede provocar retardos en la fluidez del programa (Leler, 2017).

4.3.3. Funcionamiento de la ejecución multitarea

La mayoría de los lenguajes de programación que soportan varios hilos de ejecución a la vez usan la “preferencia” para cambiar entre hilos. Cada hilo se coloca en una división del tiempo de ejecución y si el hilo excede este tiempo se para la ejecución de ese hilo para que otro hilo se ejecute (Leler, 2017).

Pero si esto ocurre cuando el hilo está utilizando un recurso que se puede compartir entre hilos, como la memoria, puede provocar que la aplicación se cuelgue o que se pierdan datos cuando se produzca un cambio de hilos (Leler, 2017).

Para solucionar este problema se utilizan “bloqueos”, que hacen que los recursos compartidos solo puedan ser utilizados por un solo hilo. Este puede provocar otros problemas incluso peores, como que el recurso quede inutilizado (Leler, 2017).

Dart aborda este problema desde otro punto de vista. Los hilos en Dart se llaman “isolates”. Estos “isolates” no comparten memoria, lo cual evita la mayoría de los bloqueos. Además, se comunican entre ellos mediante mensajes sobre canales (Dart, 2020; Leler, 2017).

Además, Dart utiliza un solo hilo, por lo que no permite la preferencia entre hilos, aunque los hilos se pueden crear explícitamente desde código (usando la programación asíncrona). Esto provoca que el desarrollador tenga más control sobre la ejecución de programa, lo cual es beneficioso para las interfaces de usuario (Leler, 2017).

4.3.4. Utilización de la memoria

La memoria es un recurso compartido, como se ha visto antes, por lo que en muchos lenguajes requieren el uso de "bloqueos". Estos bloqueos pueden provocar que el programa deje de funcionar mientras encuentra memoria libre (Leler, 2017).

Dart usa un avanzado "recolector de recursos" y un esquema de asignación muy rápido porque no tiene "bloqueos", lo que hace que no se produzcan retardos (Leler, 2017).

4.3.5. Diseño gráfico

Flutter no utiliza otro lenguaje a mayores para implementar el "layout" como XML (eXtensible Markup Language) o XAML (eXtensible Application Markup Language) porque se puede realizar todo en Dart. A la hora de crear aplicaciones nativas de Android e iOS con un desarrollo nativo, estos lenguajes se hacen necesarios, pero con Flutter y Dart no. Esta característica hace el programa más rápido a la hora de la depuración (Dart, 2019; Leler, 2017).

5. Firebase

Una vez vistos Flutter y Dart, es conveniente realizar la exposición de la plataforma utilizada como backend de la aplicación. Se trata de Firebase, una herramienta de Google que ofrece multitud de servicios en la nube.

5.1. Definición e Historia

Ilustración 13. Logo de Firebase

Firebase es una plataforma de ayuda para el desarrollo de aplicaciones tanto web como móviles (Android e iOS) a través de la nube. Su función principal es por tanto facilitar la creación de aplicaciones gracias a la gran variedad de servicios que ofrece como autentificación, base de datos, etc. (Peña, 2018).

Se lanzó en 2011 como un servicio que permitía la integración de la funcionalidad de un chat online en las páginas web. Fue comprada por Google en 2014 y en 2016 se convirtió en la plataforma de servicios que es hoy en día. Por último, en 2017, Firebase integró Cloud Firestore, que es la base de datos en tiempo real que ofrece actualmente (Firebase,2020).

5.2. Servicios

Se pueden dividir en cuatro tipos diferentes. Dependiendo si se trata de servicios que ayudan en el desarrollo de la aplicación, servicios que hacen crecer nuestras aplicaciones o servicios que analizan el uso que realizan los usuarios de la aplicación.

5.2.1. Servicios que ayudan al desarrollo de la aplicación:

Servicios que consiguen que las aplicaciones sean más seguras, potentes y escalables (Firebase, 2020):

- Cloud Firestore: Almacena y sincroniza datos entre usuarios y dispositivos a escala global mediante una base de datos NoSQL alojada en la nube.
- Authentication: Administración de usuarios. Firebase Auth ofrece varios métodos de autenticación y proveedores externos como Google o Facebook.
- Cloud Storage: Almacena y comparte contenido generado por usuarios, como imágenes, audio y vídeo, con el almacenamiento de objetos creado para la escala de Google.
- Hosting: Simplifica el hosting web con herramientas creadas específicamente para aplicaciones web modernas.

- Functions: Extiende la aplicación con código de back-end personalizado sin necesidad de administrar ni escalar los servidores.
- Machine Learning: Ofrece herramientas de machine learning para la aplicación.

5.2.2. Servicios que hacer crecer nuestras aplicaciones:

Firebase tiene servicios que ofrecen estadísticas sobre el rendimiento y la estabilidad de las aplicaciones para que se puedan aprovechar los recursos de manera más eficaz (Firebase, 2020):

- Crashlytics: Ayuda a disminuir el tiempo de solución de problemas de la aplicación.
- Supervisión del rendimiento: Diagnostica los problemas de rendimiento de la aplicación en los dispositivos de los usuarios.
- Test Lab: Ejecuta pruebas automáticas y personalizadas para la aplicación en dispositivos virtuales y físicos alojados en Google.
- App Distribution: Permite enviar versiones preliminares de la aplicación a sus testers de confianza.

5.2.3. Servicios que analizan el uso de los usuarios:

Firebase ofrece servicios para que el propietario de la aplicación vea el uso que hacen de ella los usuarios (Firebase, 2020):

- Analytics: Analiza la atribución y el comportamiento de los usuarios en un solo panel para tomar decisiones fundamentadas sobre la planificación del desarrollo del producto.

6. Flame y Box2D

En este apartado se realizará una descripción del motor de juego Flame y el motor de físicas Box2D, que se ha utilizado para desarrollar uno de los juegos de los que se compone la aplicación. Ambos se han utilizado como plugin de Flutter añadidos a la aplicación. Flame proporciona una serie de herramientas que aprovechan las características de Flutter para realizar juegos muy dinámicos de forma muy sencilla, mientras que Box2D permite crear un mundo donde poder aplicar conceptos físicos como la gravedad, el rozamiento o la interacción entre objetos (flame-engine,2020).

6.1. Características principales de Flame

Este motor está basado en la filosofía “game loop” o bucle de juego. Esta filosofía consiste en una serie de instrucciones que se ejecutan cada cierto tiempo de manera indefinida. El tiempo entre repeticiones está regulado por el parámetro FPS (Frames Per Second), que hace referencia a las veces que estas instrucciones se ejecutan en un segundo (JapAlekhin,2019).

Estas instrucciones están formadas por dos funciones o métodos principales. El método “render” prepara el área de la pantalla donde se muestra el juego para después dibujar su estado actual. El método “update”, por su parte, recibe el intervalo de tiempo desde su anterior ejecución y permite actualizar todos los elementos del juego, ya sean personajes, obstáculos o variables internas. Estos dos métodos están sincronizados, como se muestra en la Ilustración 14, Por cada interacción del “game loop” se ejecutan primero el método “update” y luego el método “render” (JapAlekhin,2019).

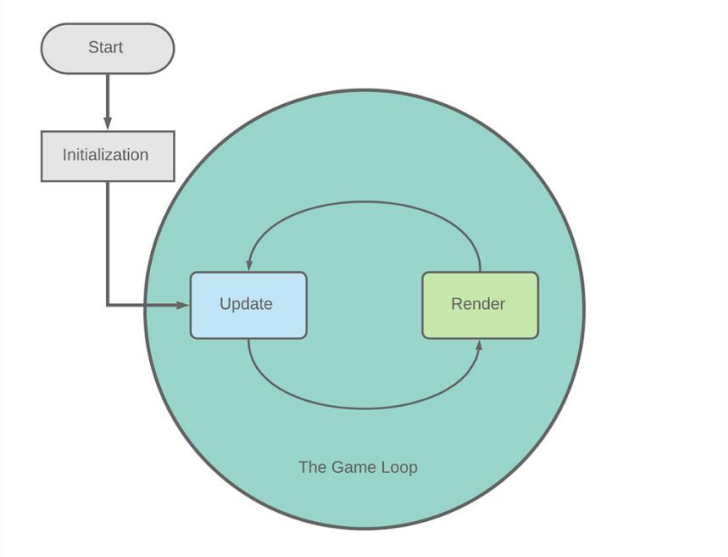


Ilustración 14. Esquema del "game loop" (JapAlekhin,2019)

Cada elemento del juego en la pantalla es un componente. Un componente puede ser una forma geométrica, una imagen, una animación o un cuerpo del motor de físicas Box2D.

6.2. Características principales de Box2D

Box2D es una biblioteca de simulación 2D de cuerpo rígido para juegos. Los desarrolladores lo usan en sus juegos para hacer que los objetos se muevan de manera realista y hacer que el mundo del juego sea más interactivo. Desde el punto de vista del motor del juego, un motor de física es solo un sistema para la animación de procedimientos (Box2D, 2020).

Ilustración 15. Logo de Box2d

Conceptos principales (Box2D, 2020):

- Shape (forma): Objeto geométrico en dos dimensiones.
- Rigid Body (Cuerpo rígido): Es un trozo de materia que es tan fuerte que la distancia entre dos pedazos de materia en el trozo es constante.
- Fixture: Fija una forma a un cuerpo y agrega propiedades de material como densidad, fricción y restitución. También fija una forma en el sistema de colisión para que pueda colisionar con otras formas.
- Constraint (Restricción): Es una conexión física que elimina los grados de libertad de los cuerpos. Un cuerpo 2D tiene 3 grados de libertad (dos coordenadas de traslación y una coordenada de rotación).
- Joint (Articulación): Esta es una restricción utilizada para mantener dos o más cuerpos juntos. Box2D admite varios tipos de articulación: revoluta, prismática, distancia y otras.
- World (Mundo): Es una colección de cuerpos, fixture y restricciones que interactúan entre sí. Box2D admite la creación de mundos múltiples, pero esto generalmente no es necesario ni deseable.

Box2D se compone de tres módulos: Común, Colisión y Dinámico. El módulo Común controla la asignación de memoria, las matemáticas y la configuración del motor de físicas. El módulo Colisión define formas y funciones/consultas de colisión. Este módulo contiene características para que el tratamiento de las colisiones sea más efectivo. Finalmente, el módulo Dinámico es el más complejo y contiene el mundo de simulación, los cuerpos, las fixtures y las articulaciones (Box2D, 2020).

7. Análisis y desarrollo de la aplicación

En este apartado se procede a explicar cómo se ha ido realizando el trabajo en la aplicación. En primer lugar, se presenta una descripción técnica de la aplicación, exponiendo las funcionalidades que debe cumplir, así como describiendo el diseño de la base de datos. Después de la descripción técnica, se presenta el manual de usuario de la app explicando el funcionamiento de la aplicación desde el punto de vista del usuario.

7.1. Descripción técnica

En este apartado se muestran detalladamente las funcionalidades que tiene la aplicación bajo estudio para que se comprenda mejor su objetivo y función. También se describe la base de datos que se ha utilizado, realizando un análisis de las tablas y los campos que componen dicha base de datos.

7.1.1. Captura de requisitos y Análisis

7.1.1.1. Catálogo de requisitos

7.1.1.1.1. Requisitos funcionales

En esta sección se presentan los requisitos funcionales que tiene la aplicación a realizar. Un Requisito Funcional es una característica requerida del sistema que expresa una capacidad de acción de este, es decir, una funcionalidad (Pérez, 2016).

Identificador	**FRQ01**
Fecha última revisión	21/06/2020
Descripción	El sistema deberá permitir al usuario registrarse como profesor o como alumno.

Tabla 1.Requisito Funcional FRQ01

Identificador	**FRQ02**
Fecha última revisión	21/06/2020
Descripción	El sistema deberá permitir al usuario cerrar sesión cuando se encuentre en la pantalla principal.

Tabla 2.Requisito Funcional FRQ01

Identificador	**FRQ03**
Fecha última revisión	21/06/2020
Descripción	El sistema deberá permitir al profesor realizar la gestión (CRUD: Create, Read, Update and Delete) de las clases del sistema.

Tabla 3. Requisito Funcional FRQ03

Identificador	FRQ04
Fecha última revisión	21/06/2020
Descripción	El sistema deberá permitir al profesor realizar la gestión (CRUD) de los alumnos del sistema.

Tabla 4. Requisito Funcional FRQ04

Identificador	FRQ05
Fecha última revisión	21/06/2020
Descripción	El sistema deberá permitir al profesor realizar la gestión (CRUD) de sus credenciales en el sistema.

Tabla 5. Requisito Funcional FRQ05

Identificador	FRQ06
Fecha última revisión	21/06/2020
Descripción	El sistema deberá permitir al alumno poder ver un PDF con la teoría de JS.

Tabla 6. Requisito Funcional FRQ06

Identificador	FRQ07
Fecha última revisión	21/06/2020
Descripción	El sistema deberá permitir al alumno reforzar sus conocimientos de JS con diferentes juegos donde sea necesario estos conocimientos para poder superarlos.

Tabla 7. Requisito Funcional FRQ07

Identificador	FRQ08
Fecha última revisión	21/06/2020
Descripción	El sistema deberá permitir al alumno poder ver sus estadísticas en los diferentes juegos.

Tabla 8. Requisito Funcional FRQ08

7.1.1.1.2. Requisitos no funcionales

Por contraposición, en esta sección se verán los Requisitos No Funcionales de la aplicación. Estos Requisitos son características requeridas del sistema, del proceso de desarrollo o del servicio prestado, que señala una restricción del mismo (Pérez, 2016).

Identificador	NFRQ01
Fecha última revisión	21/06/2020
Descripción	El sistema deberá comunicarse con la base de datos para solicitar los datos del usuario cada vez que este acceda a la app y se haya autenticado.

Tabla 9. Requisito No Funcional NFRQ01

Identificador	**NFRQ02**
Fecha última revisión	21/06/2020
Descripción	El sistema deberá garantizar que el sistema y sus datos solo serán accedidos por usuarios autorizados recordándose además las credenciales de los usuarios para facilitar su acceso al sistema.

Tabla 10. Requisito No Funcional NFRQ02

Identificador	**NFRQ03**
Fecha última revisión	21/06/2020
Descripción	El sistema deberá guardar en la base de datos externa a las clases y los usuarios si el registro se hizo correctamente.

Tabla 11. Requisito No Funcional NFRQ03

Identificador	**NFRQ04**
Fecha última revisión	13/02/2020
Descripción	La interfaz de usuario será eficaz y sencilla para que el usuario comprenda toda la información.

Tabla 12. Requisito No Funcional NFRQ04

Identificador	**NFRQ05**
Fecha última revisión	21/06/2020
Descripción	El sistema deberá mostrar una explicación del funcionamiento de cada juego que tiene la aplicación.

Tabla 13. Requisito No Funcional NFRQ05

Identificador	**NFRQ06**
Fecha última revisión	13/02/2020
Descripción	El sistema deberá informar de los errores que se produzcan cuando el usuario se autentifica o se registra.

Tabla 14. Requisito No Funcional NFRQ06

Identificador	**NFRQ07**
Fecha última revisión	21/06/2020
Descripción	El sistema deberá informar cuando el profesor complete algún cambio del éxito o fracaso del mismo con una alerta de información.

Tabla 15. Requisito No Funcional NFRQ07

Identificador	NFRQ08
Fecha última revisión	21/06/2020
Descripción	El sistema deberá informar cuando el alumno ha superado un nivel o juego con una alerta de información.

Tabla 16. Requisito No Funcional NFRQ08

7.1.1.2. Modelo de Casos de Uso

Los Casos de Uso nos ayudan a descomponer las funcionalidades de la aplicación permitiendo observar las acciones que pueden realizar cada uno de los actores que interactúan con el sistema. En la Ilustración 16 se muestra el Diagrama de Casos de Uso y a continuación del mismo se presenta la descripción textual detallada de cada uno de los Casos de Uso.

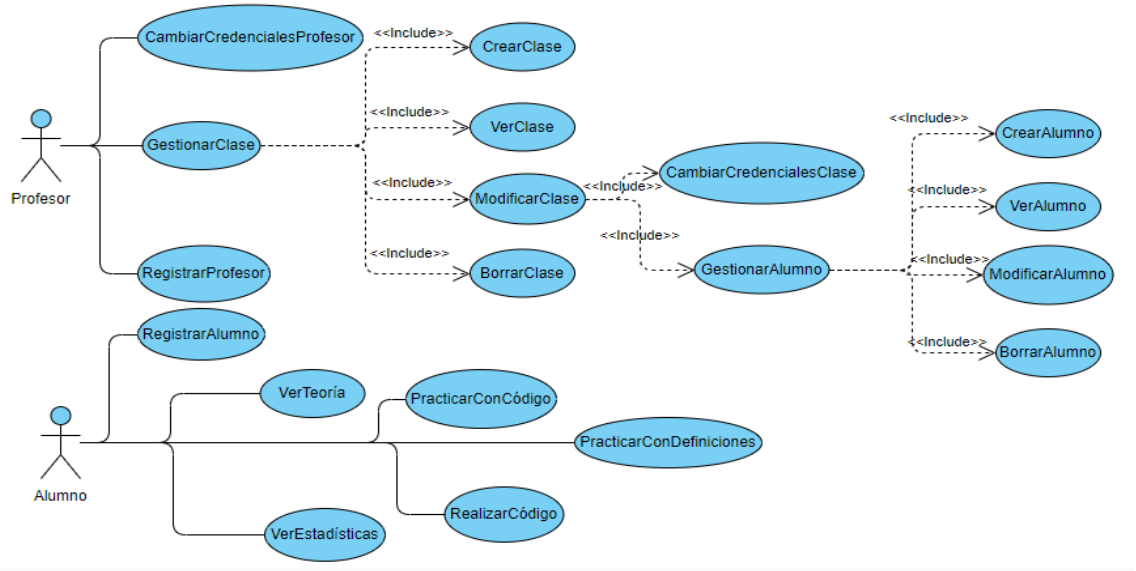


Ilustración 16. Diagrama de Casos de Uso de la Aplicación

7.1.1.2.1. RegistrarProfesor

Identificador	RegistrarProfesor	
Descripción	El usuario de la aplicación podrá registrarse como profesor.	
Actor	Profesor	
Precondición	Ninguna	
	Paso	Acción
	1	El profesor pulsa en “Registrar Profesor”.
	2	El sistema muestra un formulario para que el usuario pueda introducir sus datos (email, contraseña y nombre) para crear una cuenta

Escenario Principal de Éxito		como profesor.
	3	El profesor introduce sus datos y pulsa en “Crear cuenta”.
	4	El sistema da de alta al profesor en el sistema y accede a la página principal del profesor.
PostCondición	El profesor ha sido dado de alto en el sistema.	
Alternativas	**Paso**	**Acción**
	1-3	Si el profesor pulsa el botón “Atrás” o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
	4.1	Si ocurre algo inesperado en el registro y no se completa, se informará al usuario de la razón del fallo con un mensaje en rojo y el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 17.Caso de Uso RegistrarProfesor

7.1.1.2.2. RegistrarAlumno

Identificador	**RegistrarAlumno**	
Descripción	El usuario de la aplicación podrá registrarse como alumno si conoce los datos de su clase y de su profesor.	
Actor	Alumno	
Precondición	Existe al menos un profesor y una clase.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El alumno pulsa en “Registrar Alumno”.
	2	El sistema muestra un formulario para que el usuario pueda introducir sus datos (email, contraseña, nombre, nombre del profesor, clase y contraseña de clase) para crear una cuenta como alumno.
	3	El alumno introduce sus datos y pulsa en “Crear cuenta”.
	4	El sistema da de alta al alumno en el sistema y accede a la página principal del alumno.
PostCondición	El alumno ha sido dado de alta en el sistema.	
Alternativas	**Paso**	**Acción**
	1-3	Si el alumno pulsa el botón “Atrás” o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
	4.1	Si ocurre algo inesperado en el registro y no se completa, se informará al usuario de la razón

		del fallo con un mensaje en rojo y el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 18. Caso de Uso RegistrarAlumno

7.1.1.2.3. CambiarCredencialesProfesor

Identificador	**CambiarCredencialesProfesor**	
Descripción	El profesor podrá cambiar sus credenciales.	
Actor	Profesor	
Precondición	El profesor ha accedido a la aplicación y se ha identificado como tal.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El profesor pulsa “Cambiar Credenciales”.
	2	El sistema muestra un formulario para que el profesor pueda cambiar sus credenciales con las cajas de texto inicializadas con las credenciales actuales.
	3	El profesor cambia sus datos (email, contraseña o nombre) y pulsa en “Cambiar”.
	4	El sistema lanza una ventana para confirmar la acción.
	5	El profesor pulsa la opción afirmativa.
	6	Las credenciales del profesor son cambiadas en el sistema.
PostCondición	El profesor ha modificado sus credenciales.	
Alternativas	**Paso**	**Acción**
	1-5	Si el profesor pulsa el botón “Atrás” o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
	4.1	Si el formato de los datos no es el correcto se muestra un mensaje para indicarlo y se vuelve al paso 2.
	4.2	Si algún campo obligatorio no se ha introducido se muestra un mensaje para indicarlo y se vuelve al paso 2.
	6.1	Si ocurre algún problema en el proceso se muestra un mensaje para indicarlo y el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 19. Caso de Uso CambiarCredencialesProfesor

7.1.1.2.4. GestionarClase

Identificador	**GestionarClase**	
Descripción	El profesor puede crear, ver, modificar y borrar sus clases.	
Actor	Profesor	
Precondición	El profesor ha accedido a la aplicación y se ha identificado como tal.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1.1	El profesor pulsa “Crear Clase”.
	2.1	<<include>>CrearClase
	1.2	El profesor pulsa “Ver Clase”.
	2.2	<<include>>VerClase
	1.3	El profesor pulsa “Modificar Clase”.
	2.3	<<include>>ModificarClase
	1.4	El profesor pulsa “Borrar Clase”.
	2.4	<<include>>BorrarClase
PostCondición	Las de los casos de uso incluidos.	
Alternativas	**Paso**	**Acción**
	1.5	Si el profesor pulsa el botón “Atrás” o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 20. Caso de Uso GestionarClase

7.1.1.2.5. CrearClase

Identificador	**CrearClase**	
Descripción	El profesor podrá crear una clase.	
Actor	Profesor	
Precondición	El profesor ha accedido a la aplicación y se ha identificado como tal.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El profesor pulsa “Crear Clase”.
	2	El sistema muestra un formulario para que el profesor introduzca los datos de la clase (nombre y contraseña).
	3	El profesor introduce los datos de la clase y pulsa en “Crear”.
	4	El sistema lanza una ventana para confirmar la acción.
	5	El profesor pulsa la opción afirmativa.

	6	La nueva clase es dada de alta en el sistema.
PostCondición	El administrador ha creado una clase.	
Alternativas	**Paso**	**Acción**
	1-5	Si el profesor pulsa el botón “Atrás” o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
	4.1	Si el formato de los datos no es el correcto, se muestra un mensaje para indicarlo y se vuelve al paso 2.
	4.2	Si algún campo obligatorio no se ha introducido, se muestra un mensaje para indicarlo y se vuelve al paso 2.
	6.1	Si ocurre algún problema en el proceso, se muestra un mensaje para indicarlo y el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 21. Caso de Uso CrearClase

7.1.1.2.6. VerClase

Identificador	**VerClase**	
Descripción	El profesor de la aplicación podrá ver los datos de una clase.	
Actor	Profesor	
Precondición	El profesor ha accedido a la aplicación y se ha identificado como tal y ha creado al menos una clase.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El profesor pulsa “Ver Clase”.
	2	El sistema lanza una ventana para que se introduzca la contraseña de la clase.
	3	El profesor introduce la contraseña y pulsa en “Confirmar”.
	4	El sistema muestra la lista de alumnos de la clase y añade al menú contextual la opción de modificar las credenciales de la clase.
PostCondición	El administrador ve los datos de la clase y tiene la opción de modificar las credenciales de la misma.	
Alternativas	**Paso**	**Acción**
	1-3	Si el profesor pulsa el botón “Atrás” o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.

	3.2	Si el profesor no introduce una contraseña o la introduce de forma errónea, se vuelve al paso 2.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 22.Caso de Uso VerClase

7.1.1.2.7. BorrarClase

Identificador	**BorrarClase**	
Descripción	El profesor de la aplicación podrá borrar una clase y sus alumnos.	
Actor	Profesor	
Precondición	El profesor ha accedido a la aplicación y se ha identificado como tal y ha creado al menos una clase.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El profesor pulsa “Borrar Clase”.
	2	El sistema lanza una ventana para que se introduzca la contraseña de la clase.
	3	El profesor introduce la contraseña y pulsa en “Confirmar”.
	4	El sistema muestra una ventana de información informando de que la operación se realizó correctamente.
PostCondición	El administrador da de baja la clase y sus alumnos en el sistema.	
Alternativas	**Paso**	**Acción**
	1-3	Si el profesor pulsa el botón “Atrás” o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
	3.2	Si el profesor no introduce una contraseña o la introduce de forma errónea, se vuelve al paso 2.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 23. Caso de Uso BorrarClase

7.1.1.2.8. ModificarClase

Identificador	**ModificarClase**
Descripción	El profesor podrá modificar los datos de una clase.
Actor	Profesor
Precondición	El profesor ha accedido a la aplicación y se ha identificado como tal y ha creado al menos una clase.

Escenario Principal de Éxito	**Paso**	**Acción**
	1.1	El profesor gestiona un alumno.
	2.1	<<include>> GestionarAlumno
	1.2	El profesor pulsa “Cambiar Credenciales de Clase”.
	2.2	<<include>>CambiarCredencialesClase
PostCondición	Las de los casos de uso incluidos.	
Alternativas	**Paso**	**Acción**
	1.3	Si el profesor pulsa el botón “Atrás” o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 24. Caso de Uso ModificarClase

7.1.1.2.9. CambiarCredencialesClase

Identificador	**CambiarCredencialesClase**	
Descripción	El profesor de la aplicación podrá cambiar las credenciales de una clase.	
Actor	Profesor	
Precondición	El profesor ha accedido a la aplicación y se ha identificado como tal y ha creado al menos una clase.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El profesor pulsa “Cambiar Credenciales Clase”.
	2	El sistema muestra un formulario para que el profesor pueda cambiar las credenciales de la clase.
	3	El profesor introduce los datos de la clase y pulsa en “Cambiar”.
	4	El sistema lanza una ventana para confirmar la acción.
	5	El profesor pulsa la opción afirmativa.
	6	Las credenciales de la clase son cambiadas en el sistema.
PostCondición	El administrador ha modificado las credenciales de una clase.	
	Paso	**Acción**
	1-5	Si el profesor pulsa el botón “Atrás” o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
	4.1	Si el formato de los datos no es el correcto, se

Alternativas		muestra un mensaje para indicarlo y se vuelve al paso 2.
	4.2	Si algún campo obligatorio no se ha introducido, se muestra un mensaje para indicarlo y se vuelve al paso 2.
	6.1	Si ocurre algún problema en el proceso, se muestra un mensaje para indicarlo y el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 25. Caso de Uso CambiarCredencialesClase

7.1.1.2.10. GestionarAlumno

Identificador	**GestionarAlumno**	
Descripción	El profesor de la aplicación podrá crear un alumno, borrarlo, ver sus datos y modificarlos.	
Actor	Profesor	
Precondición	El profesor ha accedido a la aplicación y se ha identificado como tal, y ha creado al menos una clase.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1.1	El profesor pulsa “Crear Alumno”.
	2.1	<<include>> CrearAlumno
	1.2	El profesor pulsa “Ver Alumno”.
	2.2	<<include>> VerAlumno
	1.3	El profesor pulsa “Modificar Alumno”.
	2.3	<<include>> ModificarAlumno
	1.4	El profesor pulsa “Borrar Alumno”.
	2.4	<<include>> BorrarAlumno
PostCondición	Las de los casos de uso incluidos.	
Alternativas	**Paso**	**Acción**
	1.5	Si el profesor pulsa el botón “Atrás” o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 26. Caso de Uso GestionarAlumno

7.1.1.2.11. CrearAlumno

Identificador	**CrearAlumno**
Descripción	El profesor de la aplicación podrá crear un alumno.
Actor	Profesor

Precondición	El profesor ha accedido a la aplicación y se ha identificado como tal, y se ha creado al menos una clase.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El profesor pulsa “Crear Alumno”.
	2	El sistema muestra un formulario para que el profesor introduzca los datos (nombre, email y contraseña) del alumno.
	3	El profesor introduce los datos de un alumno y pulsa el botón “Crear”.
	4	El sistema lanza una ventana para confirmar la acción.
	5	El profesor selecciona la opción afirmativa.
	6	El sistema da de alta al alumno.
PostCondición	El profesor ha creado un alumno.	
Alternativas	**Paso**	**Acción**
	1-5	Si el profesor pulsa el botón “Atrás” o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
	4.1	Si el formato de los datos no es el correcto se muestra un mensaje para indicarlo y se vuelve al paso 2.
	4.2	Si algún campo obligatorio no se ha introducido, se muestra un mensaje para indicarlo y se vuelve al paso 2.
	6.1	Si ocurre algún problema en el proceso, se muestra un mensaje para indicarlo y el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 27. Caso de Uso CrearAlumno

7.1.1.2.12. VerAlumno

Identificador	**VerAlumno**	
Descripción	El profesor podrá ver los datos de un alumno.	
Actor	Profesor	
Precondición	El profesor ha accedido a la aplicación y se ha identificado como tal, y se ha creado al menos una clase y un alumno en dicha clase.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El profesor pulsa “Ver Alumno”.
	2	El sistema muestra todos los datos (nombre, email, contraseña, puntuación del juego

		"Rellena el Hueco", puntuaciones del juego "Adivina la palabra" y puntuación del juego "Programar en JS") del alumno en una ventana emergente.
PostCondición	El profesor ve los datos que el alumno tiene.	
Alternativas	**Paso**	**Acción**
	1.1	Si el profesor pulsa el botón "Atrás" o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 28. Caso de Uso VerAlumno

7.1.1.2.13. ModificarAlumno

Identificador	**ModificarAlumno**	
Descripción	El profesor podrá modificar los datos de un alumno.	
Actor	Profesor	
Precondición	El profesor ha accedido a la aplicación y se ha identificado como tal, y ha creado al menos una clase y un alumno en dicha clase.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El profesor pulsa "Modificar Alumno".
	2	El sistema muestra el mismo formulario que en el caso de uso de "Crear Alumno" solo que ahora los campos están rellenos por los datos introducidos en ese caso de uso y serán editables.
	3	El administrador cambia algún campo y pulsa en "Modificar Alumno".
	4	El sistema lanza una ventana para confirmar la acción.
	5	El profesor selecciona la opción afirmativa.
	6	El sistema realiza la actualización del alumno en el sistema.
PostCondición	El profesor ha modificado los datos de un alumno.	
Alternativas	**Paso**	**Acción**
	1-5	Si el profesor pulsa el botón "Atrás" o se sale de la aplicación el Caso de Uso finaliza sin lograr la postcondición.
	4.1	Si el formato de los datos no es el correcto, se muestra un mensaje para indicarlo y se vuelve al paso 2.

	4.2	Si algún campo obligatorio no se ha introducido, se muestra un mensaje para indicarlo y se vuelve al paso 2.
	6.2	Si ocurre algún problema en el proceso se muestra un mensaje para indicarlo y el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 29. Caso de Uso ModificarAlumno

7.1.1.2.14. BorrarAlumno

Identificador	**BorrarAlumno**	
Descripción	El profesor de la aplicación podrá borrar a un alumno, sus datos y su progreso.	
Actor	Profesor	
Precondición	El profesor ha accedido a la aplicación y se ha identificado como tal, y se ha creado al menos una clase y un alumno en dicha clase.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El profesor pulsa “Borrar Alumno”.
	2	El sistema lanza una ventana para confirmar la acción.
	3	El profesor selecciona la opción afirmativa.
	4	El sistema lanza otra ventana para indicar que la operación se realizó de forma exitosa.
PostCondición	El administrador ha borrado a un alumno.	
Alternativas	**Paso**	**Acción**
	1-3	Si el profesor pulsa el botón “Atrás” o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
	4.1	Si ocurre algún problema en el proceso se muestra un mensaje para indicarlo y el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 30. Caso de Uso BorrarAlumno

7.1.1.2.15. VerTeoría

Identificador	**VerTeoría**
Descripción	El alumno podrá ver la teoría de JavaScript en formato PDF
Actor	Alumno

Precondición	El alumno ha accedido a la aplicación y se ha identificado como tal.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El alumno pulsa sobre el botón “Ver Teoría”.
	2	El sistema muestra el PDF dentro de la aplicación.
PostCondición	El usuario puede ver la teoría en formato PDF.	
Alternativas	**Paso**	**Acción**
	1.1	Si el profesor pulsa el botón “Atrás” o se sale de la aplicación el Caso de Uso finaliza sin lograr la postcondición.
	2.1	So ocurre algún problema al abrir el archivo o al abrir el visor PDF, el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 31.Caso de Uso VerTeoria

7.1.1.2.16. PracticarConCódigo

Identificador	**PracticarConCódigo**	
Descripción	El alumno refuerza sus conocimientos de JavaScript a través de relacionar palabras con los huecos en los que las mismas encajan en scripts de código HTML y JavaScript.	
Actor	Alumno	
Precondición	El alumno ha accedido a la aplicación y se ha identificado como tal.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El alumno selecciona la opción “Rellena el Hueco”.
	2	El sistema muestra un nivel del juego donde apareceré un script de código HTML combinado con código JavaScript. En él aparecen hasta 5 huecos donde debería haber palabras. Estas palabras aparecen en una botonera en la parte inferior.
	3	El alumno pulsa en el hueco que quiere rellenar.
	4	El sistema sombrea el hueco seleccionado de azul.
	5	El alumno selecciona la opción correcta de entre las palabras de la botonera.
	6	El sistema rellena el hueco con la palabra y sombrea el hueco de verde.

	7.1	Si faltan huecos por rellenar, se vuelve al paso 3.	
	7.2	Si se han rellenado todos los huecos, el sistema muestra una ventana informando de que se ha superado el nivel.	
	8.1	Si faltan niveles por superar, se vuelve al paso 2.	
	8.2	Si se han superado todos los niveles, el sistema muestra una ventana informando de que se ha superado el juego.	
PostCondición	El alumno completa el juego.		
Alternativas	**Paso**	**Acción**	
	1-8	Si el alumno pulsa el botón "Atrás" o se sale de la aplicación, se actualiza la puntuación máxima si la actual es superior a la que estaba almacenada y el Caso de Uso finaliza sin lograr la postcondición.	
	3.1	Si el alumno no pulsa sobre ningún hueco, se selecciona por defecto el hueco que aparece antes en el código.	
	5.1	Si el alumno selecciona la opción incorrecta por primera vez en un hueco.	
		6.1	El sistema rellena el hueco con la palabra errónea y sombrea el hueco de amarillo.
		7.1	El alumno pulsa en el hueco.
		8.1	El sistema quita la palabra del hueco y la vuelve a situar en la botonera y se vuelve al paso 3, aunque el hueco queda marcado con el color amarillo.
	5.2	Si el alumno selecciona la opción incorrecta por segunda vez en un hueco.	
		6.2	El sistema rellena el hueco con la palabra errónea y sombrea el hueco de rojo.
		7.2	El alumno pulsa en el hueco.
		8.2	El sistema quita la palabra del hueco y la vuelve a situar en la botonera y se vuelve al paso 3, aunque esta vez el hueco no se puede rellenar y queda marcado como fallido.
Requisitos especiales	Ninguno		
Frecuencia esperada	Baja		

Tabla 32. Caso de Uso PracticarConCódigo

7.1.1.2.17. PracticarConDefiniciones

Identificador	**PracticarConDefiniciones**	
Descripción	El alumno refuerza sus conocimientos sobre los conceptos de JavaScript asociándolos a su definición	
Actor	Alumno	
PreCondición	El alumno ha accedido a la aplicación y se ha identificado como tal.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El alumno selecciona la opción “Adivina la palabra”.
	2	El sistema muestra las 4 secciones en las que se dividen los conceptos a adivinar de este juego. Cada sección tiene 2 niveles de dificultad, uno con pistas u otro sin ellas.
	3	El alumno selecciona la opción que quiere jugar.
	4	El sistema muestra un nivel del juego donde aparecen 3 partes bien diferenciadas: la definición de un concepto de JavaScript, una botonera superior con espacios en blanco, uno por cada letra de la palabra y una botonera inferior con dos filas de letras.
	5	El alumno pulsa en una letra de la botonera de abajo.
	6	El sistema sitúa la letra en el hueco en blanco situado más a la izquierda y la quita de la botonera de abajo.
	7.1	Si siguen quedando huecos en blanco en la botonera superior se vuelve al paso 5.
	7.2	Si se completa la palabra y es la correcta el sistema muestra una ventana informando de que se ha superado el nivel.
	8.1	Si faltan niveles por superar, se vuelve al paso 4.
	8.2	Si se han superado todos los niveles, el sistema muestra una ventana informando de que se ha superado el juego.
PostCondición	El alumno completa el juego	
Alternativas	**Paso**	**Acción**
	1-8	Si el alumno pulsa el botón “Atrás” o se sale de la aplicación, se actualiza la puntuación máxima si la actual es superior a la que estaba almacenada y el Caso de Uso finaliza sin lograr la postcondición.

	5.1	Si el alumno está jugando en el modo “Jugar con pistas” y pulsa en el botón “Recibir pista”, el sistema sitúa una letra aleatoria de la palabra a acertar en su posición.	
	8.1	Si se completa la palabra pero no es la correcta, el sistema muestra una ventana informando de que no se ha superado el nivel.	
		9.1	El alumno pulsa una letra de la botonera superior.
		9.2	El alumno pulsa el botón “Recibir pista”, si está jugando en el modo “Jugar con pistas”.
		10.1	El sistema deja en blanco ese espacio de nuevo y sitúa la letra en la botonera de abajo. Se vuelve al paso 5.
		10.2	El sistema lanza una ventana de información de que no tiene espacio en blanco para poder recibir pista. Se vuelve al paso 9.1.
Requisitos especiales	Ninguno		
Frecuencia esperada	Media		

Tabla 33. Caso de Uso PracticarConDefiniciones

7.1.1.2.18. RealizarCódigo

Identificador	**RealizarCódigo**	
Descripción	El alumno refuerza sus conocimientos sobre los conceptos de JavaScript sobre objetos programando para que un robot pueda conseguir monedas y superar los distintos niveles.	
Actor	Alumno	
PreCondición	El alumno ha accedido a la aplicación y se ha identificado como tal.	
. **Escenario Principal de Éxito**	**Paso**	**Acción**
	1	El alumno selecciona la opción “Programar en JS”.
	2	El sistema muestra un nivel del juego compuesto por dos pantallas. Una con el juego y otra con las instrucciones y el campo de texto para poder introducir el código.
	3	El alumno siguiendo las recomendaciones que se le dan escribe el código y pulsa guardar.
	4	El sistema muestra una ventana emergente

		informando de que el código está bien escrito y lo guarda.
	5	El alumno pulsa el botón “play” en la pantalla del juego.
	6	El sistema ejecuta las acciones que el alumno ha programado, el robot llega a la bandera y supera el nivel y el sistema muestra una ventana indicando el logro.
	7.1	Si no era el último nivel se vuelve al paso 2.
	7.2	Si es el último nivel del juego se vuelve a mostrar una ventana que informa del logro.
PostCondición	El alumno completa el juego.	
Alternativas	**Paso**	**Acción**
	1-7	Si el alumno pulsa el botón “Atrás” o se sale de la aplicación, se actualiza la puntuación máxima si la actual es superior a la que estaba almacenada y el Caso de Uso finaliza sin lograr la postcondición.
	3.1	Si el usuario escribe mal el código y pulsa a guardar, entonces el sistema muestra una ventana emergente informando de que el código está mal escrito y lo guarda.
	3.2	El usuario pulsa el botón “play” sin haber guardado el código.
		4.2 El sistema ejecuta el código por defecto que tiene, por lo que el robot no se mueve y no supera el nivel.
		5.2 El alumno pulsa el botón “reset”.
		6.2 El sistema reinicia el nivel por lo que se vuelve al paso 2.
	3.3	Si el alumno pulsa el botón “reset”, el sistema reinicia el nivel por lo que se vuelve al paso 2.
	3.4	Si el alumno pulsa el botón “Mostrar solución”, el sistema muestra la solución a ese nivel.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 34. Caso de Uso RealizarCódigo

7.1.1.2.19. VerEstadísticas

Identificador	**VerEstadísticas**
Descripción	El alumno podrá ver sus estadísticas en todos los juegos que de la aplicación.

Actor	Alumno	
PreCondición	El alumno ha accedido a la aplicación y se ha identificado como tal.	
Escenario Principal de Éxito	**Paso**	**Acción**
	1	El alumno selecciona la opción "Ver Estadísticas" dentro de la pantalla de su perfil.
	2	El sistema muestra las estadísticas del alumno en los diferentes juegos de la aplicación.
PostCondición	El alumno observa sus estadísticas en los diferentes juegos.	
Alternativas	**Paso**	**Acción**
	1.1	Si el alumno pulsa el botón "Atrás" o se sale de la aplicación, el Caso de Uso finaliza sin lograr la postcondición.
Requisitos especiales	Ninguno	
Frecuencia esperada	Media	

Tabla 35. Caso de Uso VerEstadísticas

7.1.2. Base de datos

Como base de datos externa se ha utilizado el servicio de Firebase Cloud Firestore. Esta base de datos de tipo NoSQL, es flexible, escalable y está situada en la nube. Esta base de datos almacena los datos en documentos que contienen campos que se asignan a valores. Estos documentos se almacenan en colecciones, que son contenedores para los documentos y que puedes usar para organizar los datos y compilar consultas. Los documentos admiten varios tipos de datos diferentes, desde strings y números simples, hasta objetos anidados complejos (Firebase, 2020).

Para nuestra aplicación, han hecho falta 5 colecciones o tablas. Cada colección está formada por documentos y cada documento tienen un ID que lo diferencia de los demás en la colección. Cada documento puede tener diferentes campos, aunque para nuestra aplicación cada documento dentro de una colección tendrá los mismos campos.

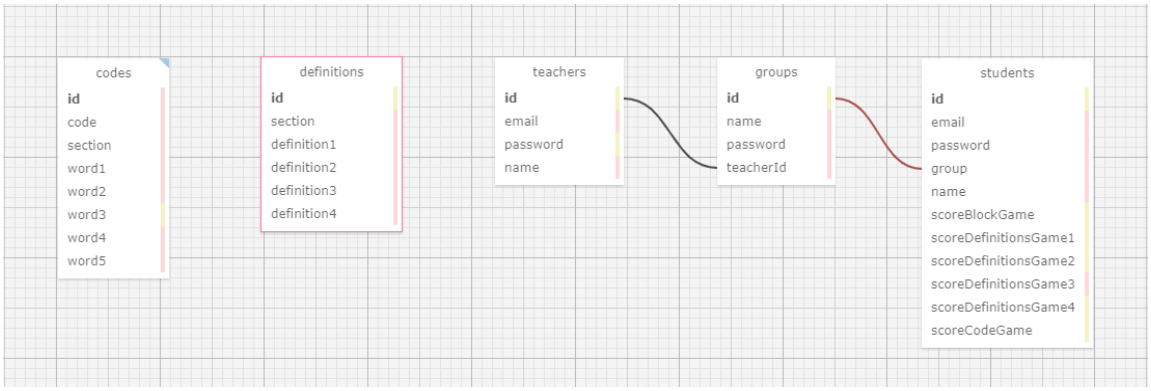


Ilustración 17. Colecciones de la base de datos

7.1.2.1. Colección “codes”

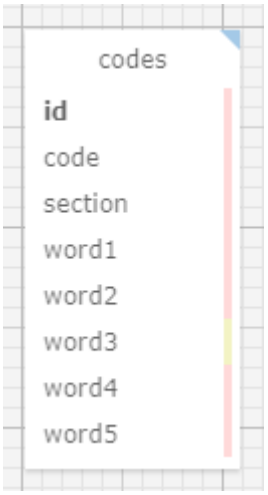


Ilustración 18. Campos de los documentos que forman la colección "codes"

Colección que guarda la información necesaria para crear los diferentes niveles del juego “Rellena el hueco”. Esta información consiste en varios strings.

Campos:

id: Identificador del documento dentro de la colección.

code: Código HTML combinado con código de JavaScript en forma de string.

section: Identificador de la sección de teoría a la que pertenece el script.

word1 … word5: Palabras escogidas dentro del script que luego serán los huecos a rellenar dentro del script y las palabras a escoger para rellenar estos huecos. Puede haber hasta 5 palabras.

7.1.2.2. Colección “definitions”

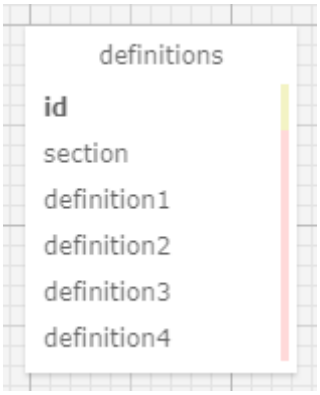


Ilustración 19. Campos de los documentos que forman la colección "definitions"

Colección que guarda la información necesaria para crear los diferentes niveles del juego “Adivina la palabra”. Esta información consiste en varios strings dentro de un documento.

Campos:

id: Identificador del documento dentro de la colección. En este caso el id es la palabra que se tiene que adivinar en el nivel.

section: Identificador de la sección de teoría a la que pertenece el script.

definition1 … definition4: Diferentes definiciones de la palabra a adivinar.

7.1.2.3. Colección “teachers”

Colección que guarda la información necesaria de un profesor.

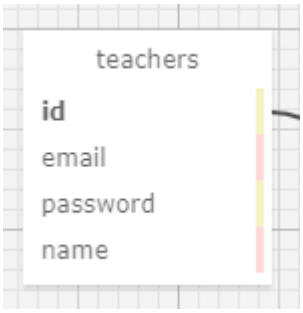


Ilustración 20. Campos de los documentos que forman la colección "teachers"

Campos:

id: Identificador del documento dentro de la colección. Esta vez es una cadena generada automáticamente por la base de datos al crear el profesor.

email: Email con el que se identifica el profesor para acceder a la aplicación.

password: Contraseña con la que se identifica el profesor para acceder a la aplicación.

name: Nombre del profesor.

7.1.2.4. Colección “groups”

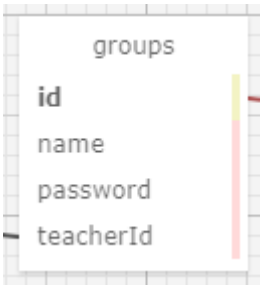


Ilustración 21. Campos de los documentos que forman la colección "groups"

Colección que guarda la información necesaria de una clase. Esta colección tiene una relación 1:n con la colección “teachers”, ya que un profesor puede tener infinitas clases si así lo desea.

Campos:

id: Identificador del documento dentro de la colección. Esta vez es una cadena generada automáticamente por la base de datos al crear al grupo.

name: Nombre de la clase

password: Contraseña de la clase, necesaria por cuestiones de seguridad.

teacherId: Identificador del profesor al que pertenece la clase.

7.1.2.5. Colección “students”

Colección que guarda la información necesaria de un alumno. Esta colección tiene una relación 1:n con la colección “groups”, ya que una clase puede tener infinitos alumnos si así lo desea el profesor.

Campos:

id: Identificador del documento dentro de la colección. Es una cadena generada automáticamente por la base de datos al crear el grupo.

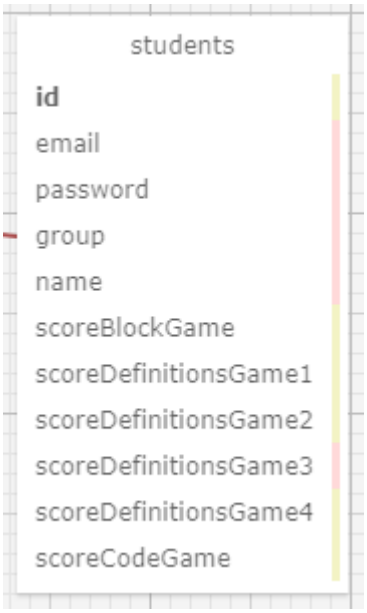


Ilustración 22. Campos de los documentos que forma la colección "students"

email: Email con el que se identifica el alumno para acceder a la aplicación.

password: Contraseña con la que se identifica el alumno para acceder a la aplicación.

Group: Identificador de la clase a la que pertenece el alumno

name: Nombre del alumno.

scoreBlockGame: Puntuación máxima del alumno en el juego "Rellena el hueco".

scoreDefinitionsGame1 … scoreDefinitionsGame4: Puntuación máxima del alumno en cada una de las secciones del juego "Adivina la palabra".
scoreCodeGame: Puntuación máxima del alumno en el juego "Programar en JS".

7.2. Manual de usuario

Una vez vistas las principales funcionalidades de la aplicación, se va a realizar con ayuda de capturas y descripciones un manual para que el usuario sepa desenvolverse por la aplicación. Como son dos tipos de usuarios diferentes los que pueden usar la aplicación (tanto profesores como alumnos), este manual se divide en dos partes. Cada parte, que representa lo que un profesor o un alumno puede hacer está dividida en las diferentes pantallas que la aplicación tiene.

7.2.1. Manual de usuario del Profesor

7.2.1.1. Pantalla de Login y Registro

La primera pantalla a la que se accede según se inicia la aplicación es la pantalla que permite a los usuarios entrar y registrarse en el sistema. Está compuesta por 3 formularios, uno para autenticarse, (común tanto para los profesores como para los alumnos), otro para el registro de los profesores y otro para el registro de los

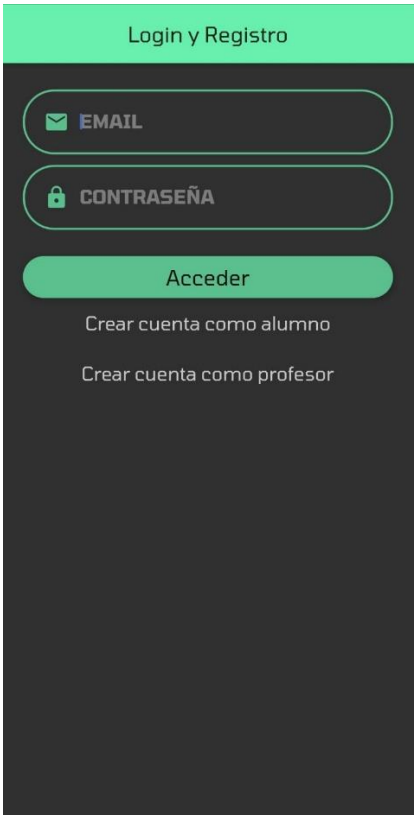


Ilustración 23. Pantalla del login

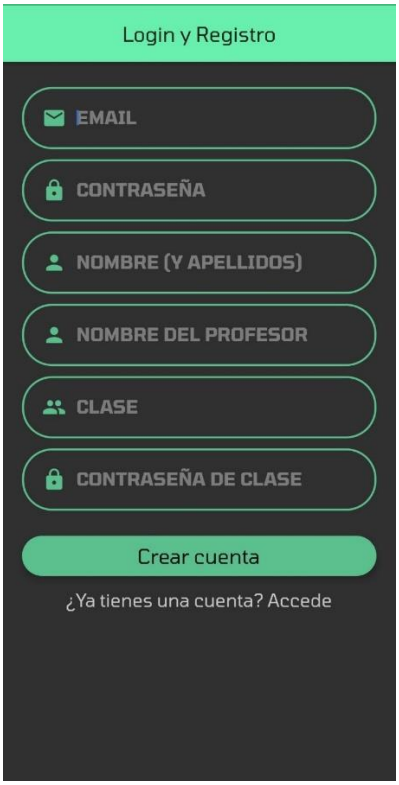


Ilustración 24. Pantalla de registro de un alumno

alumnos. El formulario que aparece por defecto es el que permite autenticarse al usuario, tal y como se muestra en la Ilustración 23.

El formulario para autenticarse está formado por dos campos de texto que permiten introducir respectivamente el email y la contraseña y tres botones.

El botón "Acceder" permite a los usuarios acceder a la aplicación. Los mensajes de error producidos por el sistema en el caso de que algo salga mal al pulsar este botón se presentan de dos formas:

- Si al pulsar este botón dejamos algún campo de texto vacío o escribimos un correo de forma incorrecta, el sistema nos avisará resaltando de rojo el borde del campo de texto y mostrando un mensaje en rojo debajo del campo de texto. Este tipo de aviso puede aparecer en todos los formularios de la aplicación.
- Si se trata de un fallo de conexión a Internet, de conexión a la base de datos, de autentificación o credenciales mal escritas, el sistema avisará mostrando un mensaje en rojo debajo del formulario.

Ilustración 25. Pantalla de registro de un profesor

El botón "Crear cuenta como Alumno" permite acceder al formulario de registro de alumnos. El botón "Crear cuenta como Profesor", por su parte, permite acceder al formulario de registro de profesores.

El formulario de registro de un alumno está formado por 6 campos para introducir texto y 2 botones tal y como se aprecia en la Ilustración 24.

El botón "Crear cuenta" registra al alumno con los datos introducidos y también realiza el login. Este formulario tiene también los dos tipos de mensajes explicados para cuando ocurre un error.

El botón "¿Ya tienes una cuenta? Accede" nos

devuelve al formulario del login.

El formulario de registro de un profesor está formado por 3 campos de texto para introducir el email, la contraseña y el nombre del profesor como se ve en la Ilustración 25.

El botón “Crear cuenta” tiene el mismo funcionamiento que en la anterior pantalla. Registra al profesor en la aplicación y realiza el login. Este formulario también tiene genera los dos tipos de mensajes explicados cuando ocurre un error.

7.2.1.2. Pantalla principal del usuario profesor

A esta pantalla se accede al autenticarse el usuario como profesor o cada vez que abrimos la aplicación. Está formada por una barra superior donde aparecen un menú, el título de la página que es el nombre del profesor y un botón para añadir una clase nueva tal y como se puede observar en la Ilustración 26 y la Ilustración 27.

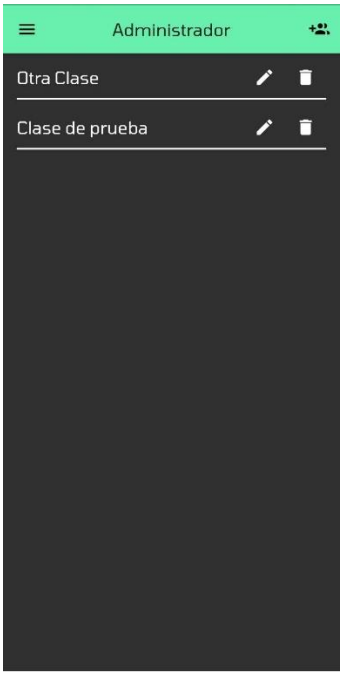


Ilustración 26. Pantalla principal del profesor con clases creadas

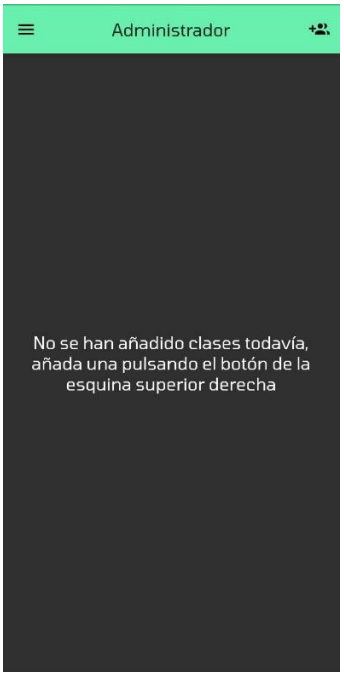


Ilustración 27. Pantalla principal del profesor sin clases

Si se accede por primera vez o no hay ninguna clase creada en el cuerpo de la página aparecerá el siguiente mensaje “No se han añadido clases todavía, añada una pulsando el botón de la esquina superior derecha”.

Si se pulsa el botón del menú contextual (Icono formado por 3 rayas horizontales situado en la parte superior izquierda) aparecerá un menú despegable con los datos del profesor (nombre y email) y dos botones a mayores. El botón formado por el icono de una casa y el texto “Cerrar sesión” sirve como su nombre indica para volver a la pantalla de login y registro. Si se pulsa esta opción aparecerá una ventana emergente para confirmar la operación. El botón más inferior (formado por el icono de una tuerca y el texto “Cambiar Credenciales de Usuario”) sirve para acceder a la página del formulario que permite cambiar las credenciales al profesor.

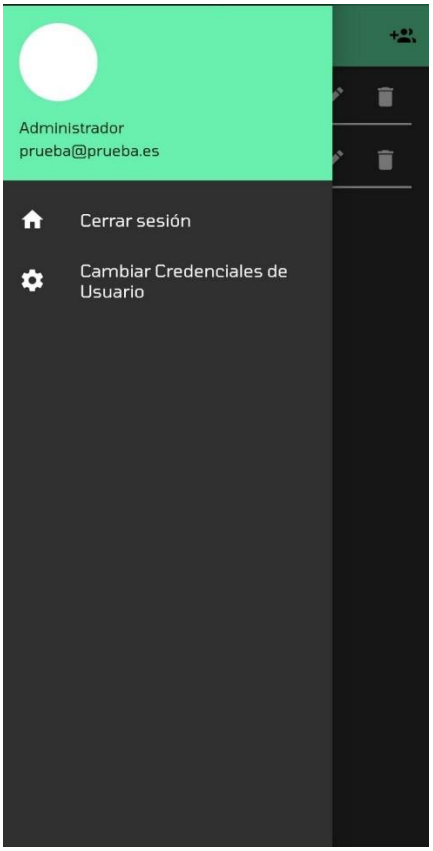


Ilustración 28. Menú contextual desplegado

Si se pulsa el botón de añadir clase (icono del símbolo de una suma y dos personas situado en la esquina superior derecha) se accede al formulario que permite añadir una clase nueva.

Si se pulsa en cualquiera de las dos opciones que tiene cada clase (editar clase o borrar clase) el sistema lanzará una ventana emergente que solicitará la contraseña de la clase tal y como se puede ver en la Ilustración 29. Si el usuario se equivoca escribiendo la contraseña, el sistema lanzará otra alerta para que el usuario prueba a introducir la contraseña de nuevo.

Si se ha pulsado sobre el botón de la izquierda (icono de un lápiz), se accederá a la pantalla de la clase que se explicará a continuación. En cambio, si se ha pulsado sobre el icono de la papelera (botón situado más a la derecha) se borrará la clase y sus alumnos. El sistema lanzará una alerta del resultado de la operación informando sobre el éxito o el fracaso de esta.

Ilustración 29.Ventana emergente que se lanza cuando pulsamos en cualquiera de las opciones de una clase

7.2.1.3. Pantalla "Cambiar Credenciales de Usuario"

Esta pantalla sirve para cambiar las credenciales del profesor. Está formada por un formulario con 3 campos donde poder cambiar el email, el nombre o la contraseña y un campo a mayores para introducir la antigua contraseña por motivos de seguridad. Además de esto tiene el botón "Cambiar" que al pulsarlo lanza la típica alerta de confirmación que se puede ver en la Ilustración 30. Si se acepta esta última se activa todo el procesamiento para cambiar las credenciales del profesor. Al tratarse de un formulario, tiene los dos tipos de avisos explicados en los formularios del login y

registro por si sucede algún error durante el proceso.

Ilustración 30. Pantalla Cambiar Credenciales

Ilustración 31. Alerta de confirmación lanzada al pulsar en "Cambiar"

7.2.1.4. Pantalla "Crear Clase"

Esta pantalla sirve para añadir una nueva clase al sistema. Está formada por un formulario compuesto por dos campos para introducir texto, en concreto el nombre de la clase y su contraseña. La contraseña de la clase se ha añadido como un dato más de identificación de la clase de cara al registro por parte del usuario y por motivos de seguridad cuando el profesor realice una acción sobre una clase

Además de estos dos campos el formulario también tiene el botón "Crear clase". Si se pulsa este botón salta la alerta de confirmación presente en todos los formularios. Si se acepta la confirmación el sistema procede a crear la clase.

Al tratarse de un formulario si algo sucede de forma errónea el sistema avisará con los tipos de mensajes ya

Ilustración 32. Pantalla "Crear Grupo"

explicados.

7.2.1.5. Pantalla principal de la clase

Ilustración 34. Pantalla principal de la clase con alumnos

A esta pantalla se accede cuando se ha pulsado en "editar clase" y se ha introducido la contraseña correcta. Es una pantalla muy similar a la pantalla principal del profesor, ya que se compone de una barra superior con opciones en los extremos y un cuerpo formado por una lista, aunque esta vez la lista es de alumnos y no de clases.

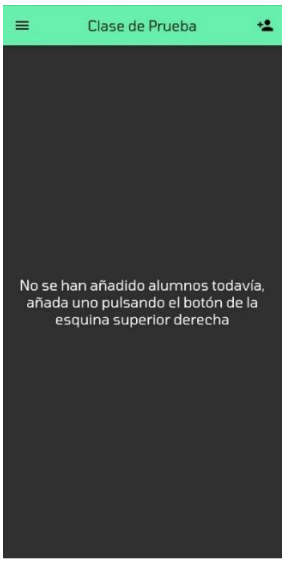


Ilustración 33. Pantalla principal de la clase sin alumnos

Si se accede por primera vez o no hay ningún alumno creado, en lugar de la lista nos aparecerá este mensaje: "No se han añadido alumnos todavía, añada uno pulsando el botón de la esquina superior derecha".

Si se pulsa el botón formado por Icono formado por 3 rayas horizontales situado en la parte superior izquierda se mostrará un menú contextual similar al de la página principal, pero con una opción a mayores. Si se pulsa en esta opción (Botón formado por el icono de una llave inglesa y el texto "Cambiar Credenciales de Grupo" tal y como se ve en la Ilustración 35 se accede al formulario que permite cambiar el nombre y la contraseña de la clase.

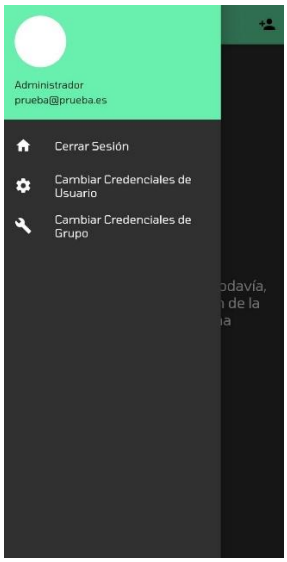


Ilustración 35. Menú contextual de la pantalla clase desplegado

En la esquina superior izquierda de la página se sitúa el botón formado por el símbolo de la suma y una sola persona tal y como se puede ver en todas las ilustraciones de este apartado. Este botón sirve para añadir un alumno a la clase.

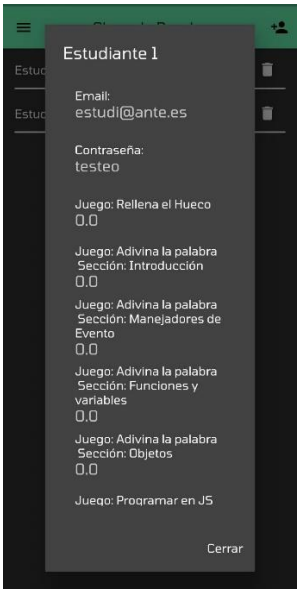


Ilustración 36. Ventana con los datos del alumno desplegada

Cada alumno de la lista tiene 3 iconos que actúan de botones. Al pulsar el icono del ojo se muestran todos los datos del alumno. Desde sus estadísticas en todos los

juegos hasta sus credenciales tal y como se aprecia en la Ilustración 36.Si se pulsa sobre el icono del lápiz se accede al formulario que permite cambiar las credenciales del alumno. Por último, si se pulsa sobre la papelera, el sistema lanzará una ventana de confirmación por si de verdad se quiere borrar al alumno tal y como se ve en la Ilustración 37. En caso afirmativo, el sistema lanzará otro aviso informando del resultado de la operación.

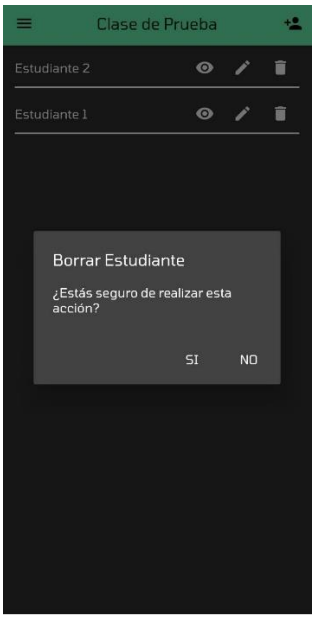


Ilustración 37. Ventana que se despliega cuando queremos borrar a un alumno

7.2.1.6. Pantalla "Cambiar Credenciales de Grupo"

Ilustración 38. Pantalla que contiene el formulario para modificar las credenciales del grupo

Para acceder a esta pantalla hay que pulsar el botón "Cambiar Credenciales de Grupo" del menú desplegable de la pantalla principal de la clase. Está formada por un formulario muy sencillo con dos campos de texto para cambiar tanto el nombre como la contraseña del grupo. Estos campos de textos vienen inicializados con las credenciales actuales del grupo. También aparece el botón "Modificar grupo". Si se pulsa este botón se lanzará el típico aviso que se muestra en la Ilustración 39.

En este formulario también tiene los dos tipos de avisos explicados previamente por si se produce algún error en el proceso.

Ilustración 39. Alerta de confirmación que se lanza cuando se pulsa el botón "Modificar Grupo"

7.2.1.7. Pantalla "Crear Alumno"

A esta pantalla se llega tras pulsar el icono de la esquina superior derecha de la pantalla principal del grupo. Está formada por un formulario con 3 campos para introducir texto que representan el nombre, el email y la contraseña del alumno.

Además, contiene el botón "Crear Estudiante" que, al

pulsarlo, el sistema lanza la alerta de confirmación típica de estos formularios. En caso afirmativo, el sistema procede a dar de alta el alumno en el sistema y lo añade a lista de alumnos de la clase actual.

En este formulario también tiene los dos tipos de avisos explicados previamente por si se produce algún error en el proceso. Si se deja algún campo en blanco o el email no es válido se sombrea de rojo del campo de texto y se escribe el error debajo del campo correspondiente. Si en cambio, el error es por otra cuestión, el mensaje de aviso aparecerá debajo del formulario.

Ilustración 40. Pantalla "Crear Alumno"

7.2.1.8. Pantalla Modificar Alumno

Esta pantalla es similar a la anterior, aunque muestra un par de diferencias. En el título, en el botón principal y en el contenido de la alerta de confirmación en vez de poner “Crear Estudiante”, el mensaje es “Modificar Estudiante”. También tiene los mismos tipos de avisos de error que los demás formularios.
En contraposición, la función que representa es diferente. Este formulario permite cambiar las credenciales de un alumno que ya está creado por parte del profesor.

7.2.2. Manual de usuario del Alumno

7.2.2.1. Pantalla de registro y login

Igual que en el caso del profesor.

7.2.2.2. Pantalla principal del alumno

A esta pantalla se accede tras identificarse el usuario como alumno o tras iniciar la aplicación si el login se hizo previamente. Está formada por la barra típica superior donde se encuentra el icono de 3 rayas horizontales que despliega el menú contextual y el título formado por el nombre del usuario como se puede apreciar en la Ilustración 42. El menú contextual es similar al menú contextual del profesor, con la diferencia de que este menú solo tiene la opción “Cerrar Sesión”. Si se pulsa en ese botón formado por el icono de una casa y el texto

Ilustración 41. Pantalla principal del alumno

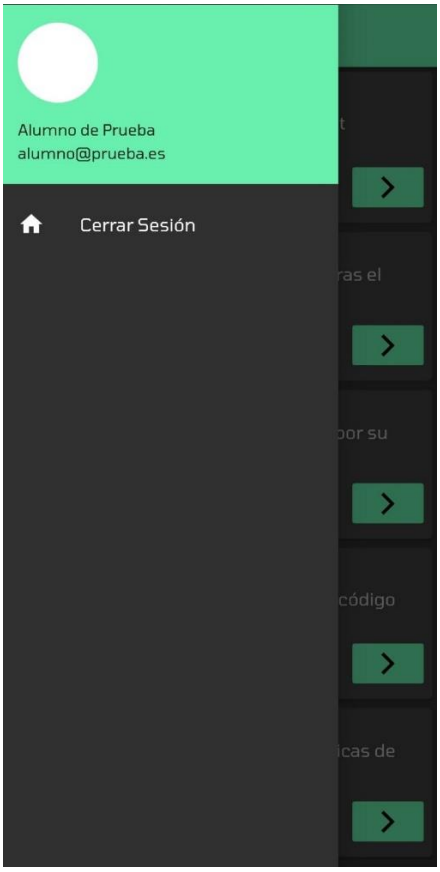


Ilustración 42. Menú contextual desplegado

"Cerrar Sesión" aparecerá una alerta de confirmación. En caso de que se pulse la opción afirmativa "SI" el alumno saldrá a la pantalla de login.
El cuerpo de la página está formado por 5 filas con el mismo formato que representan las 5 diferentes opciones principales a las que el alumno puede acceder. Todas las filas están formadas por un icono, un título, un subtítulo que sirve para exponer una muy breve explicación de lo que se puede hacer en ese apartado si se pulsa el botón de acceso formado por el icono ">".

Si se pulsa en el botón de la primera fila se accede al PDF con la teoría de JavaScript para que el usuario la pueda revisar dentro de la aplicación cuando desee.

Si se pulsa en el botón de la segunda fila se accede al primer juego: "Rellena el Hueco", el cual se explicará más adelante

El botón de la tercera fila sirve para acceder al segundo de los juegos: "Adivina la palabra", que también se explicará después.

Por su parte, el botón de acceso de la cuarta fila permite acceder al tercer y último juego de la aplicación: "Programar en JS".

El botón de la última fila sirve para acceder al apartado donde el alumno puede ver sus estadísticas en los distintos juegos.

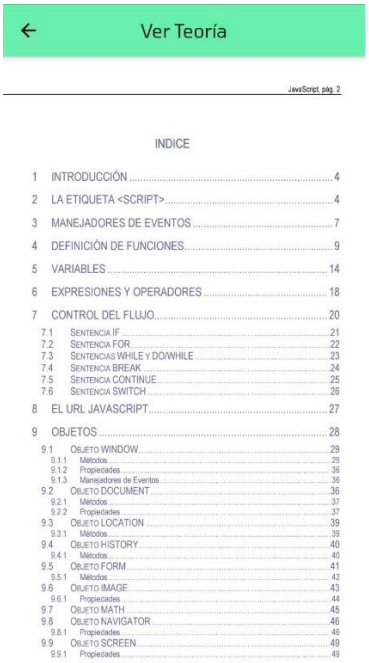


Ilustración 43. Pantalla "Ver Teoría"

7.2.2.3. Pantalla "Ver Teoría"

Esta pantalla está formada por el título "Ver Teoría" en su parte superior y el cuerpo, donde se muestra un archivo PDF que contiene la teoría de JavaScript, por lo que el usuario puede acceder a ella desde la aplicación muy fácilmente. Además, es muy fácil llegar

a cualquier punto del documento ya que, si pulsamos en cualquier apartado del índice que se encuentra al principio del documento, llegamos a ese punto del mismo. Este visor permite también hacer zoom en el documento por si está acción fuera necesaria.

7.2.2.4. *Juego "Rellena el Hueco"*

Este juego está formado por 10 niveles escogidos aleatoriamente entre unos 17 posibles. Cada nivel se compone de un script de código HTML combinado siempre con código JavaScript. Cada script se sitúa en la parte superior del cuerpo de la página ocupando la mayoría del espacio. El script contiene huecos en su interior donde deberían de estar situadas ciertas palabras. Estas palabras se muestran en orden aleatorio en una botonera en la parte inferior del cuerpo tal y como se muestra en la Ilustración 45.

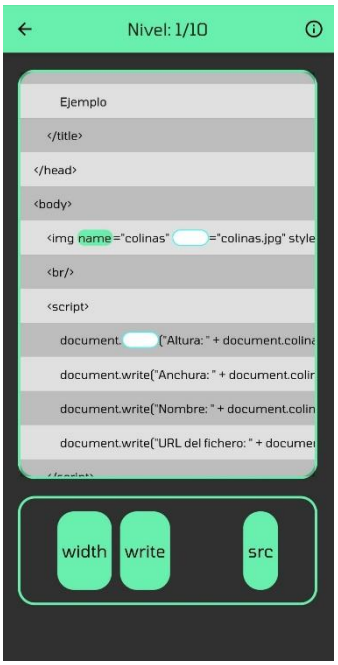


Ilustración 44. Captura "Rellena el Hueco" con un hueco acertado

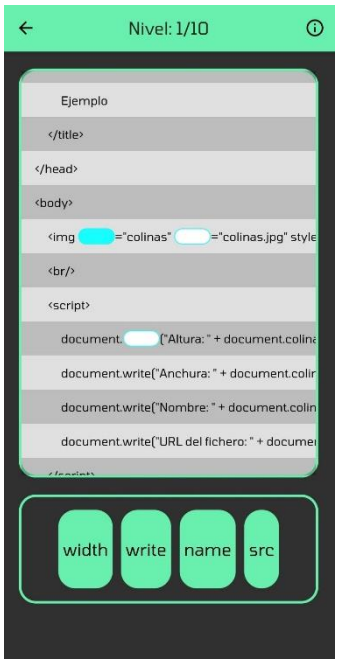


Ilustración 45. Captura del juego "Rellena el Hueco" con un hueco seleccionado

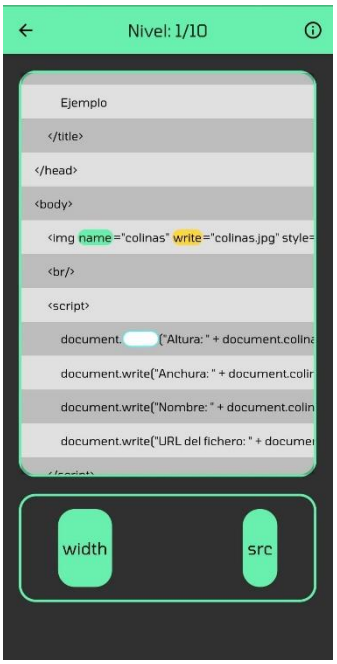


Ilustración 46. Captura del juego "Rellena el Hueco" con un hueco errado en su primer intento y con la palabra errónea aún en su interior

El alumno debe situar cada palabra en su hueco del código correspondiente para poder superar el nivel. Para ello debe seleccionar el hueco que desea rellenar. Esto se consigue pulsando sobre el hueco. Si un hueco está seleccionado su fondo se colorea de azul si no ha sido previamente coloreado de ningún otro color. Por

defecto está seleccionado el primer hueco, aunque este no se colorea. Se puede eliminar la selección si se pulsa el hueco seleccionado y la selección volvería a su opción por defecto.

Una vez seleccionado, o no, el hueco, para poder situar la palabra en su hueco se debe pulsar el botón de la palabra de la parte inferior. Si la palabra es la correcta esta se sitúa en el hueco, se colorea de verde y el hueco y la palabra dejan de ser seleccionables, tal y como se puede ver en la Ilustración 44.

Si la palabra es incorrecta esta se sitúa en el hueco, pero este se colorea de amarillo, tal y como se aprecia en la Ilustración 46. La palabra se puede devolver a la botonera, pero el hueco sigue quedando coloreado de amarillo. Si ahora se selecciona la palabra correcta, la palabra se sitúa en el hueco seleccionado, y sin cambiar el color de fondo el hueco y la palabra dejan de ser seleccionables.

Si se vuelve a escoger la palabra incorrecta, esta se sitúa en el hueco, pero ahora este se colorea de rojo, tal y como se ve en la Ilustración 49. La palabra se puede devolver, como antes a la botonera, pero el hueco deja de ser seleccionable.

Ilustración 47. Captura del juego "Rellena el Hueco" con la alerta de nivel finalizado desplegada

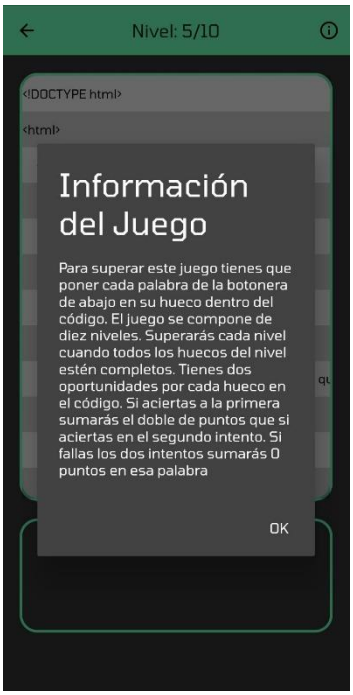


Ilustración 48. Captura del juego "Rellena el Hueco" con la ventana de información desplegada

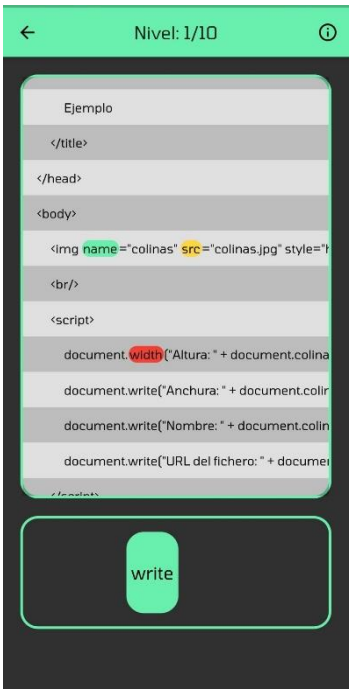


Ilustración 49. Captura del juego "Rellena el Hueco" con un hueco fallado

El nivel se termina cuando todos los huecos dejan de ser seleccionables. Cuando esto ocurre el sistema lanza una alerta de información que informa al usuario de sus progresos en el juego tal y como se ve en la Ilustración 47. El juego termina cuando se superan los 10 niveles.

Por cada palabra acertada a la primera, el usuario consigue una puntuación de 10/nº de palabras. Es decir, que si el nivel tiene 5 palabras consigue 2 puntos (10/5 = 2) y si tiene 4 palabras consigue (10/4 = 2,5). La idea es que el alumno sume 10 puntos si acierta todas las palabras del nivel a la primera. Si falla una vez y acierta a la segunda sumará la mitad de puntos que si acertase a la primera. Es decir que si el nivel tiene 5 palabras conseguirá 1 punto y si tiene 4 conseguirá 1.25 puntos. Si el usuario falla el hueco (no sitúa la palabra correcta en el hueco tras dos intentos) no recibirá puntos por ese hueco.

Además del normal funcionamiento del juego, si un alumno decide abandonar el juego puede hacerlo pulsando el botón "←" situado en la parte superior izquierda. Al realizar esta acción, el sistema lanzará una ventana para confirmar que el usuario realmente quiere abandonar el juego. En caso afirmativo, el sistema guardará la puntuación que el usuario lleve si es superior a la que ya tenía.

Por otra parte, si pulsamos el botón de información, situado en la esquina superior derecha, el sistema lanzará una ventana con un texto explicando el funcionamiento del juego, tal y como se ve en la Ilustración 48.

7.2.2.5. Juego "Adivina la palabra"

Este juego también está formado por 10 niveles escogidos aleatoriamente entre muchos. Cada nivel se compone de un contendor donde se sitúa la definición de un concepto de JavaScript a adivinar, inmediatamente debajo se sitúa otro contenedor donde se sitúan tantos huecos en blanco como letras tiene la palabra a adivinar y debajo de este otro donde se sitúan 24 letras que el usuario puede seleccionar para formar la palabra. Las 24 letras se forman utilizando las letras de la palabra a adivinar y otras escogidas aleatoriamente. Su ordenación también es aleatoria. Se puede observar la interfaz gráfica en la Ilustración 51

La particularidad de este juego es que los conceptos a adivinar están divididos en 4 secciones que son "Introducción", "Manejadores de eventos", "Funciones y Variables" y "Objetos" como se puede ver en la Ilustración 50. Cada sección tiene sus propias definiciones, por lo que son 4 juegos distintos con la misma temática y desarrollo.

Para superar un nivel hay que acertar la palabra a adivinar. Para ello hay que rellenar todos los huecos del contenedor del medio. Si pulsamos en una letra de la botonera de abajo, esta se situará en el hueco más a la izquierda de la botonera del medio y se

quitará de la botonera inferior como se ve en la Ilustración 51. Si se completa la palabra de la botonera del medio aparecerá un mensaje informando de si se ha acertado la palabra o de si no. El mensaje de nivel superado se puede ver en la Ilustración 52. El juego se supera cuando has superado los diez niveles.

Ilustración 50. Pantalla para escoger una de las 8 opciones diferentes de juego

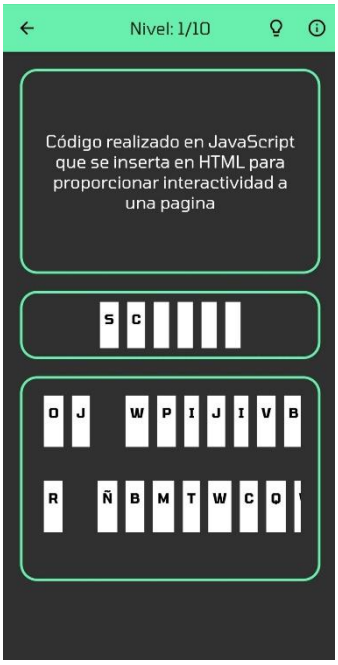


Ilustración 51. Captura del juego "Adivina la palabra" con dos letras situadas

Ilustración 52. Ventana emergente que indica que hemos acertado la palabra

Si se ha fallado la palabra, entonces se deben desalojar las letras mal situadas de la palabra. Para ello pulsando en una letra de la botonera del medio, se devuelve la misma a la botonera inferior.

Por cada palabra acertada a la primera, se sumarán 2 puntos, si la palabra se acierta a la segunda, el alumno sumará 1 punto, si la acierta a la tercera 0.5 puntos y así sucesivamente. La puntuación máxima de este juego por tanto es de 20 puntos.

Cada juego se puede jugar con dos dificultades: con o sin pistas. La dificultad con pistas ofrece al alumno la posibilidad de recibir ayuda de la máquina. Si el alumno pulsa en el icono de la bombilla situado en la parte superior derecha, el sistema situará una letra escogida aleatoriamente de la palabra a adivinar en su sitio. Para ello, quitará la letra de la parte inferior. Esta operación la puede hacer con todas las letras, pero la idea es que esta función sirva para ayudar al alumno en un momento puntual. Para solicitar una pista debe haber huecos sin letras en la botonera de la palabra a adivinar. La dificultad sin pistas no contiene esta funcionalidad por lo que si

no se conoce un concepto no se podrá superar el nivel.

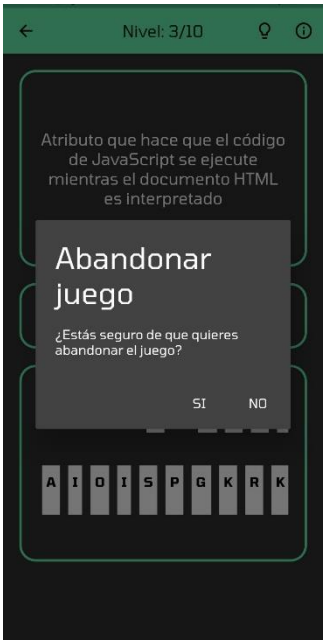


Ilustración 53. Ventana de confirmación de abandono del juego

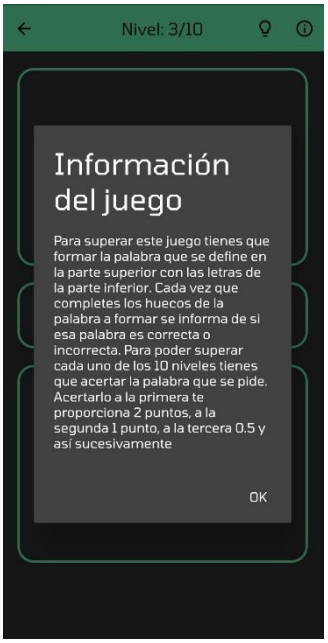


Ilustración 54. Ventana que se muestra cuando se pulsa el icono de información

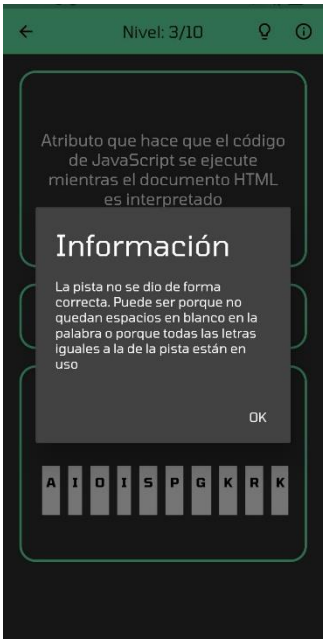


Ilustración 55. Ventana que se muestra cuando no se puede dar la pista al usuario

Este juego tiene la misma operativa que el anterior para poder salirse de él. Si se pulsa en el icono "←" el sistema lanzará una ventana de confirmación, tal y como se ve en la Ilustración 53. En caso de pulsar el botón "SI", el sistema guardará la puntuación si esta es mayor que la anterior que tenía guardada.

Como el anterior juego, este también tiene un icono en la parte superior derecha que al pulsarlo lanza una ventana donde se explica el funcionamiento del juego tal y como se puede observar en la Ilustración 54.

7.2.2.6. Juego "Programar en JS"

Este último juego está formado por 6 niveles fijos, cada nivel se compone de dos pantallas. Ambas pantallas tienen en común el botón con el icono "←" para salir del juego en la esquina superior izquierda que tiene el mismo mecanismo que en los anteriores juegos. Además, comparten el botón para pasar de una pantalla a otra en la parte superior derecha. Este botón está formado por dos flechas apuntando al centro, tal y como se puede observar en todas las capturas de este juego, como por ejemplo en la Ilustración 56.

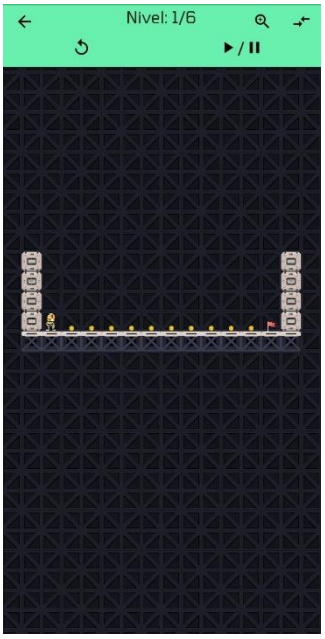


Ilustración 56. Pantalla 2

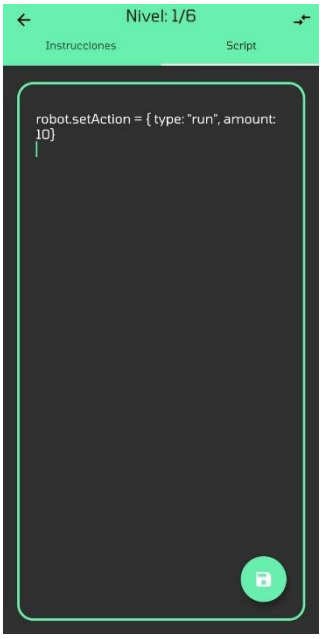


Ilustración 57. Pantalla 1, pestaña "script"

Ilustración 58. Pantalla 1, pestaña "instrucciones"

Cuando se inicia el juego la pantalla que se muestra, a partir de ahora pantalla 1, es la que está formada por dos pestañas (pestaña “instrucciones” y pestaña “script”) a mayores de lo explicado antes.

La pestaña “instrucciones” está formada por un texto común para todos los niveles que explica el funcionamiento del juego y que es lo primero que ve el alumno al entrar al juego. Se puede ver en la Ilustración 58. Esta vez se ha optado por esta opción porque al ser un juego más difícil de entender que los anteriores, de esta manera el alumno tiene la información del funcionamiento del juego según lo inicia. En la parte inferior del texto se muestra una recomendación que cambia con el nivel. Debajo del texto aparece el botón “Mostrar solución” que permite al alumno ver la solución si se atasca.

La pestaña “script” está formada por un campo para introducir texto y un botón con un icono de un disquete que sirve para que el sistema guarde el texto introducido en el campo de texto.

La otra pantalla, a partir de ahora pantalla 2, se compone de dos botones a mayores y el cuerpo que representa al mundo del juego debajo de ello como se ve en la Ilustración 56. El botón de la izquierda representado por una flecha en forma de círculo sirve para reiniciar el nivel y el de la derecha formado por un icono de “play” y otro de “pause” sirve para parar y reanudar el juego.

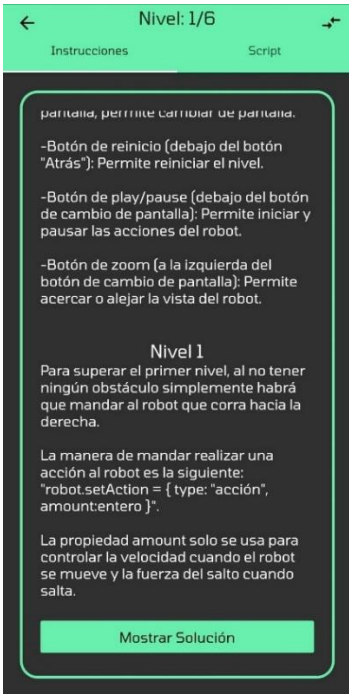


Ilustración 59. Parte inferior de la pestaña "instrucciones"

Ilustración 60. Ventana emergente que avisa de que el código está bien escrito

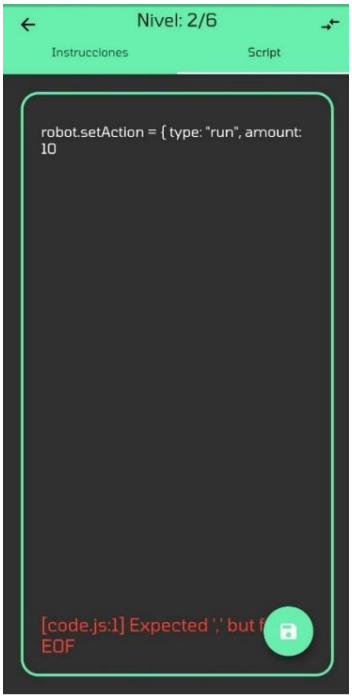


Ilustración 61. El sistema informa de que ha ocurrido un error de sintaxis

Para superar un nivel se debe conseguir que el robot llegue a tocar una bandera evitando una serie de obstáculos que aparecen en el cuerpo de la pantalla 2. Para que el robot actúe se le tienen que dar las instrucciones adecuadas en lenguaje JavaScript. Estas instrucciones deben escribirse en el campo de texto de la pestaña “script” y guardarse en el sistema con el botón que hay preparado para ello en esa misma pestaña. Si las instrucciones no tienen errores de sintaxis aparecerá una ventana que avisará de que el código está bien escrito. Si las instrucciones están mal escritas, el sistema avisará del error escribiéndolo en la parte inferior de la pestaña “script”.

Una vez escrito correctamente el código, el alumno debe ir a la pantalla 2 y pulsar el botón “play/pause”. Si el código hace al robot llegar a la bandera, se superará el nivel y el sistema avisará con otra ventana informativa de ello.

Si el nivel no se supera, el alumno debe pulsar el botón “reset” que reinicia el nivel y por tanto se debe introducir las instrucciones otra vez en el sistema. Si se pulsa el botón de “play/pause” sin haber introducido previamente instrucciones, el robot se actuará porque por defecto el robot tiene programada la acción de esperar que es una de las acciones que puede ejecutar. Las acciones que el robot puede realizar se indican en la pestaña instrucciones.

El sistema guarda las monedas que consigue el robot, las cuales se pueden ver en la Ilustración 56 como estadística de este juego. Se pueden conseguir un máximo de 52 monedas entre los 6 niveles. Estos incrementan su dificultad según se va avanzado, por lo que el nivel 3 es más complicado que el nivel 1.

Si se pulsa en el icono "←" el sistema lanzará una ventana de confirmación. En caso de pulsar el botón "SI", el sistema guardará la puntuación si esta es mayor que la anterior que tenía guardada.

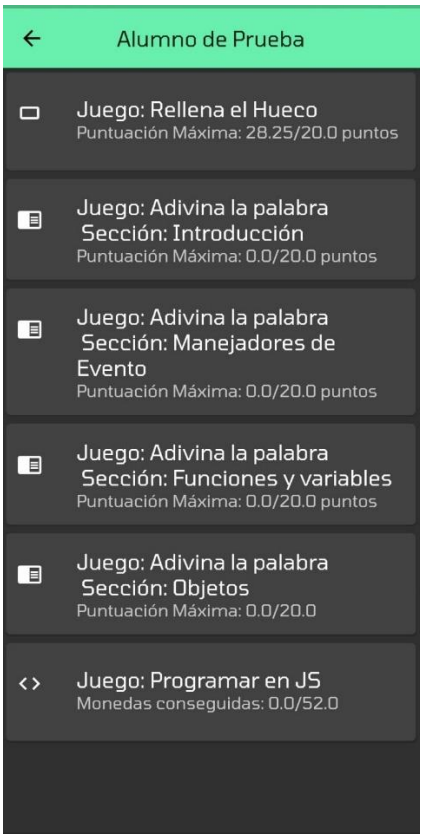


Ilustración 62. Pantalla "Ver Estadísticas"

7.2.2.7. Pantalla "Ver Estadísticas"

Esta pantalla se trata de la última pantalla de la aplicación. Sirve para ver las estadísticas que se han ido comentando de los diferentes juegos. Está compuesta por 6 filas que muestran las estadísticas de todos los juegos. El juego "Adivina la palabra" aporta 4 filas, una por cada sección.

Las filas están formadas por un icono, un título que indica el Juego y la sección en el caso del juego "Adivina la palabra" y un subtítulo que indica la puntuación máxima en ese juego del alumno tal y como se aprecia en la Ilustración 62.

8. Presupuesto Económico

En este apartado se realizará un estudio económico aproximado del desarrollo de la aplicación. Hay que tener en cuenta muchos factores que se irán desarrollando a continuación para llegar a una cifra aproximada.

Existen algunas páginas web que realizan este cálculo en base a unas pocas preguntas. Si se accede, por ejemplo a la página web www.cuantocuestaunaapp.com o www.cuantocuestamiapp.com los factores que se tienen en cuenta son:

- Tipo de plataforma: No es lo mismo desarrollar para Android, o para iOS o para ambas.

- Tipo de interfaz: Una interfaz sencilla como la de nuestra aplicación lleva menos tiempo de desarrollo que una interfaz personalizada.

- Registro: Poder registrarse en la aplicación también es una cuestión a considerar, ya que si los usuarios necesitan validarse para acceder, eso también conlleva tiempo de desarrollo extra.

- Cantidad de idiomas de la aplicación: Si la aplicación se crea para varios idiomas se deben traducir los textos de la aplicación. Además, adaptar la aplicación para que se puede cambiar de idioma suma tiempo al desarrollo.

Teniendo en cuenta estas cuestiones y alguna más, a continuación, se realiza un presupuesto para el desarrollo de la aplicación bajo estudio:

- Salario del programador: En este apartado se tendrá en cuenta que el salario de un programador junior en España está sobre unos 1.269€, el tiempo de desarrollo que ha sido de aproximadamente 2 meses y que el autor de este trabajo está empezando con los desarrollos de alto nivel (Indeed, 2020).

- Gastos del lugar de trabajo: En este apartado se incluye la luz, el agua, la conexión a Internet y el alquiler del mismo.

- Gastos en licencias de software: Aquí se incluyen el gasto de las herramientas software que se han usado para el desarrollo.

- Gastos en la publicación de la aplicación.

No se ha tenido en cuenta el uso de los portátiles ni de los dispositivos móviles utilizados ya que ya están lo suficientemente amortizados.

Razón	*Presupuesto*
Salario (2 meses a 1269€)	2.538€
Luz, Internet, Agua, Alquiler (2 meses a 200€)	400€
Licencias de Software (Android Studio, Flutter, Firebase, Flame)	0€
Publicación en stores	0€
Total	*2.938€*

Tabla 36. Presupuesto ficticio de la aplicación

9. Conclusiones y líneas futuras

En este último apartado se comentarán las conclusiones de la realización de este trabajo, así como las líneas futuras que quedan abiertas a partir de este momento para poder mejorar la aplicación.

9.1. Conclusiones

Desde que empecé a programar siempre tuve la intención de aprender a realizar aplicaciones para móviles que fueran útiles en algún sentido. Como el camino que escogí fue realizar el Grado en Ingeniería de Tecnologías de Telecomunicación, sabía que mi opción más viable para aprender a desarrollar este tipo aplicaciones era realizar un Trabajo de Fin de Grado sobre ello.

Además de esto, mi intención fue siempre realizar un desarrollo que sirviese tanto para Android como para iOS, por lo que cuando me ofrecieron la posibilidad de hacerlo con Flutter acepté de inmediato porque es una herramienta desarrollada por Google que permite tener casi las mismas funcionalidades que un desarrollo nativo.

La temática de la aplicación sí fue más complejo decidirla, la idea era desarrollar juegos para que sirviesen de apoyo al estudio de JavaScript y que estos estuvieran orientados en gran medida a reforzar el aprendizaje de las características más potentes de JavaScript como son los manejadores de eventos y el uso de los objetos de la página web. Creo que con los juegos que se han desarrollado quedan cubiertas las expectativas, ya que tanto con el juego “Rellena el Hueco” como con el juego “Adivina la palabra” se relacionan conceptos y con el juego “Programar en JS” se permite programar al alumno, que es realmente como se aprende a programar.

Este proyecto también ha servido para aprender a utilizar bases de datos como backend para la aplicación. He seguido la opción más recomendaba para Flutter, que no es otra que usar Firebase, otro producto de Google, para implementar la base de datos y también la autentificación de usuarios.

El desarrollo del último juego también me ha permite realizar una exhaustiva búsqueda de información para poder implementar un intérprete de JavaScript dentro de la aplicación. Dart, el lenguaje de programación utilizado no es tan conocido y no existe tanta información como por ejemplo para JavaScript, que es un lenguaje de programación mucho más popular que Dart. El juego “Programar en JS” también me ha permitido iniciarme en el diseño de videojuegos con la herramienta Flame. Si bien es una herramienta sencilla y adaptada a Flutter, me ha permitido aprender conceptos, como el “game loop”, muy interesantes.

Por último, el análisis técnico realizado como paso previo al desarrollo también ha sido de gran utilidad porque te permite centrar las ideas y te proporciona un guion sobre cómo

enfocar el desarrollo de la aplicación, lo cual simplifica mucho el trabajo.

9.2. Líneas futuras

Aunque la aplicación cumple las funcionalidades y especificaciones inicialmente propuestas, estas se pueden ampliar en un futuro para luego desarrollarlas en la aplicación. A continuación, se exponen algunas funcionalidades que se podrían incluir en la aplicación:

- Que el profesor pueda actualizar el PDF de la teoría. De este modo, el profesor subiría el archivo desde la aplicación y los alumnos lo podrían ver desde sus dispositivos.

- Ampliar las estadísticas que recoge la aplicación sobre los juegos, de esta forma tanto el profesor como el alumno reciben más información. Estadísticas como el tiempo gastado, el número de palabras acertadas en los dos primeros juegos, etc.

- Permitir al profesor vetar juegos a los alumnos, por ejemplo, para que en el juego "Adivina la palabra" no puedan usar el modo con pistas. De este modo el profesor monitoriza de forma más eficiente lo que los alumnos pueden hacer con la aplicación.

- Permitir al usuario escoger otros idiomas, además del español, como el inglés o el francés.

- Añadir sonido a los diferentes juegos, para hacer más amena la jugabilidad.

- Añadir más niveles más complicados al juego "Programar en JS". Los niveles creados son suficientes, pero si se quiere complicar la complejidad del juego se pueden crear niveles más difíciles de superar si así se desea.

- También se puede cambiar la base de datos. El tipo de base de datos utilizada es NoSQL. Si se quiere migrar a MySQL se puede hacer, moviendo toda la información guardada en la base de datos actual.

- Se puede añadir más juegos para reforzar los conocimientos de JavaScript.

10. Bibliografía

Alibaba Tech (1 de agosto de 2018). *Making the Most of Flutter: From Basics to Customization*. Medium. Obtenido de https://medium.com/hackernoon/making-the-most-of-flutter-from-basics-to-customization-433171581d01 en junio de 2020.

Amateur Coder (16 de febrero de 2020). *Stateless vs. Stateful*. Medium. Obtenido de https://itnext.io/stateless-vs-stateful-cde9d178084f en junio de 2020.

Android, (7 de mayo de 2020). *Arquitectura de la plataforma*. Obtenido de https://developer.android.com/guide/platform en junio de 2020.

AndroidOS, (2012). *Arquitectura*. AndroidOS. Obtenido de https://androidos.readthedocs.io/en/latest/ en junio de 2020.

Android Studio (2020). *Android Studio*. Obtenido de https://developer.android.com/studio en junio de 2020.

Box2d, (2020). *Overview.* Obtenido de https://box2d.org/documentation/ en junio de 2020.

Carltics, (16 de marzo de 2015). *iOS y su arquitectura interna en 4 capas*. Ticsandroll Blog. Obtenido de http://blog.ticsandroll.es/ios-y-su-arquitectura-interna-en-4-capas/ en junio de 2020.

Codemagic, (18 de marzo de 2019). *Flutter vs Xamarin: A Developer's Perspective*. Codemagic. Obtenido de https://blog.codemagic.io/flutter-vs-xamarin-a-developer-s-perspective/ en junio de 2020.

Dart, (2020). *Dart documentation*. Obtenido de https://dart.dev/guides en junio de 2020.

Data Is Beautiful, (9 de octubre de 2019). *Most Popular Mobile OS 1999 – 2019 [Video]*. Youtube. Obtenido de https://www.youtube.com/watch?v=MMyMB4zm9so en junio de 2020.

Deepak,K (02 de marzo de2018). Flutter: From Zero To Comfortable. Medium. Obtenido de https://proandroiddev.com/flutter-from-zero-to-comfortable-6b1d6b2d20e en junio de 2020.

Firebase, (2020). *Productos*. Obtenido de https://firebase.google.com/products?hl=es en junio de 2020.

Flame-Engine, (2020). *Docs*. Obtenido de https://flame-engine.org/docs en junio de 2020.

Flutter (2020). *Documentación de Flutter*. Obtenido de https://flutter-es.io/docs en junio de

2020.

Github (2020). Th State of the octoverse. Obtenido de https://octoverse.github.com/ en junio de 2020.

Imagina Formación (2020). *Flutter, React Native o Xamarin. ¿Cuál es el mejor framework para desarrollar una aplicación móvil?* Obtenido de https://www.imaginaformacion.com/tutorial/flutter-vs-react-native-vs-xamarin-cual-es-el-mejor-framework/ en junio de 2020.

InnovaAge, (2011). *Apps Hibridas vs Nativas vs Generadas. ¿Qué decisión tomar?* Obtenido de https://www.innovaportal.com/innovaportal/v/696/1/innova.front/apps-hibridas-vs-nativas-vs-generadas-que-decision-tomar en junio de 2020.

JapAlekhin, (24 de febrero de 2019). *Create a Mobile Game with Flutter and Flame – Beginner Tutorial.* Obtenido de https://jap.alekhin.io/create-mobile-game-flutter-flame-beginner-tutorial en junio de 2020.

JavaTPoint, (2018). *Flutter Architecture*. Obtenido de https://www.javatpoint.com/flutter-architecture en junio de 2020.

Lancetalent, (20 de febrero de 2014). *Los 3 Tipos De Aplicaciones Móviles: Ventajas E Inconvenientes*. Obtenido de https://www.lancetalent.com/blog/tipos-de-aplicaciones-moviles-ventajas-inconvenientes/ en junio de 2020.

Leler,W (23 de diciembre de 2017). *Why Flutter Uses Dart*. Hackernoon. Obtenido de https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf en junio de 2020.

López Moreno, M. (15 de junio de 2015). *5 beneficios del m-learning*. Nubemia. Obtenido de https://www.nubemia.com/5-beneficios-del-m-learning/ en junio de 2020.

Matos, L (13 de agosto de 2019). *Flutter vs Xamarin: An opinion of an analisis*. Obtenido de https://luismts.com/flutter-vs-xamarin-an-opinion-of-an-analysis/ en junio de 2020.

Ortiz, O. (2 de junio de 2014). *Herramienta m-learning para el Aprendizaje de Programación Estructurada en los Primeros Cursos de Ingeniería*. Obtenido de http://rita.det.uvigo.es/VAEPRITA/201406/uploads/VAEP-RITA.2014.V2.N2.A2.pdf en junio de 2020.

Peña, J. (8 de febrero de 2018). *Ventajas y Desventajas de la Aplicaciones Nativas*. Arpen Technologies. Obtenido de https://arpentechnologies.com/es/blog/aplicaciones-movil/ventajas-y-desventajas-de-las-aplicaciones-nativas/ en junio de 2020.

Peña, J (1 de febrero de 2018*). ¿Qué es firebase y qué nos aporta?.* Arpen Technologies.

Obtenido de https://arpentechnologies.com/es/blog/aplicaciones-movil/que-es-firebase-y-que-nos-aporta/ en junio de 2020.

Pérez, M. A. (2016). Catálogo de Requisitos.

Semana, (9 de noviembre de 2017). *¿Por qué la educación y la tecnología son aliados inseparables?* Obtenido de https://www.semana.com/educacion/articulo/uso-de-la-tecnologia-en-la-educacion/539903 en junio de 2020.

Sistemas, (2016). *Definición de Aplicación*. Obtenido de https://sistemas.com/aplicacion.php en junio de 2020.

Statista, (2020). Cuota de mercado de pedidos de smartphones a nivel mundial por sistema operativo entre 2014 y 2020. Obtenido de https://es.statista.com/estadisticas/600731/cuota-de-mercado-de-sistemas-operativos-para-smartphones-por-pedidos--2020/ en junio de 2020.

Stolar, A (08 de febrero de 2020). *How to Build an App with Flutter – Part 1. Introduction*. Droids on Roids. Obtenido de https://www.thedroidsonroids.com/blog/how-to-build-an-app-with-flutter-introduction en junio de 2020.

Universidad Politécnica de Valencia, (2017). *Arquitectura de Android*. Universidad Politécnica de Valencia. Obtenido de http://www.androidcurso.com/index.php/recursos/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android en junio de 2020.

Wikipedia, (23 de junio de 2020). *Teléfono Inteligente*. Obtenido de https://es.wikipedia.org/wiki/Tel%C3%A9fono_inteligente en junio de 2020.

Wikipedia, (23 de junio de 2020). *iOS*. Obtenido de https://es.wikipedia.org/wiki/IOS en junio de 2020.

Wikipedia, (17 de junio de 2020). *Windows 10 Mobile*. Obtenido de https://es.wikipedia.org/wiki/Windows_10_Mobile en junio de 2020.