UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática

en Ingeniería del Software

Trabajo Fin de Grado

NamEx – Aplicación móvil de gamificación para el aprendizaje de los nombres de los estudiantes

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería Informática

en Ingeniería del Software

Trabajo Fin de Grado

NamEx – Aplicación móvil de gamificación para el aprendizaje de los nombres de los estudiantes

Autor: Manuel García Martín

Tutor: José Carlos Sancho Núñez

Co-Tutor/es: Andrés Caro Lindo

ÍNDICE GENERAL DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Resumen

Para el profesorado universitario recordar todos y cada uno de los nombres de sus alumnos puede conllevarles un desafío. El alto número de alumnos que pasan cada año por las aulas en la educación superior dificulta el aprendizaje del nombre de todos. Sin embargo, hay expertos que consideran importante para la educación que el docente llame a cada alumno por su nombre, debido a que ayuda al estudiante a motivarse por el aprendizaje e identificarse con la materia impartida.

Ciertos profesionales de la Universidad de Extremadura detectaron esta carencia y buscaron algo que les ayudase a memorizar el nombre de los alumnos para mantener un contacto más cercano con cada uno de ellos. Con esta premisa se desarrolla *NamEx,* una aplicación móvil de gamificación para el aprendizaje de los nombres de los estudiantes.

En consecuencia, el presente proyecto se centra en la documentación del proceso seguido para la creación de la aplicación *NamEx,* que contiene diferentes modos de juego y dificultades, además de un ranking que permite ver las puntuaciones obtenidas. Utilizando la gamificación como la base fundamental en el desarrollo de la misma.

Esta aplicación va a permitir, a todos aquellos interesados en utilizarla, relacionar los nombres de los alumnos con sus caras, de una manera lúdica, divertida y dinámica, posibilitando la visualización de los resultados obtenidos, tanto individuales como de otros compañeros de profesión, dejando abiertas una serie de líneas de futuro en las que se podrá seguir trabajando.

Abstract

The memorization of the students' names can represent a challenge for university professors. The high number of students who are enrolled in higher education each year hinders the learning of their names. Nevertheless, there are experts who consider important for the education the relation between teachers and students and, consequently, the relevance of calling students by their names, since it helps them to be motivated in learning and feel identified with the subject taught.

Some professionals from the University of Extremadura identified this deficiency and looked for a support tool to help them in memorizing the names of the students to create a closer relationship with each of them. With this premise, *NamEx,* a mobile gamification application for learning the names of the students, is developed.

Consequently, this project focuses on the documentation of the process followed for the creation of *NamEx* application, which contains different game modes and difficulties, as well as a ranking that shows the scores obtained, using the gamification as the fundamental basis in the development of the same.

This application will allow the relation of the names of the students with their faces, to all those interested in using it, in a playful, fun and dynamic way, allowing the visualization of the results obtained individually, as well as the score of other colleagues of profession, leaving a series of lines of future in which the application will be able to be developed extensively.

Capítulo 1

1. Introducción y objetivos del proyecto.

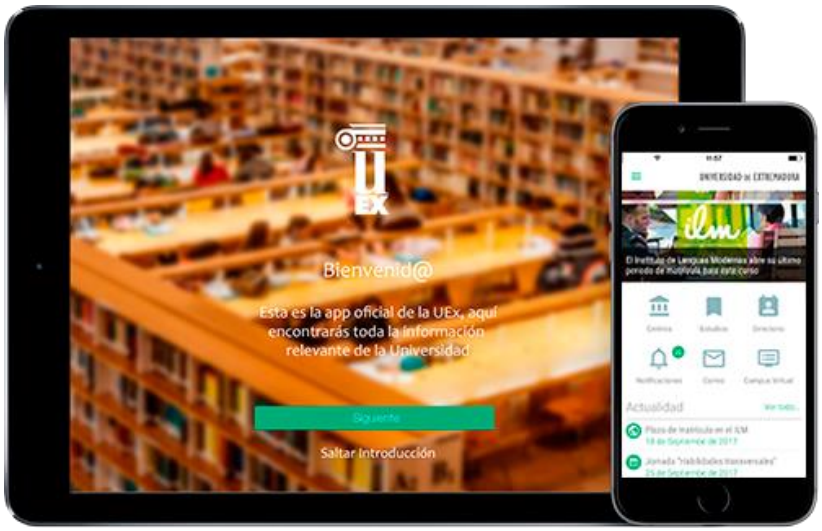
1.1 Introducción.

Inicialmente, la idea que da vida al desarrollo de este proyecto fue transmitida a uno de mis tutores desde la facultad de Ciencias del Deporte situada en este mismo campus universitario de Cáceres, donde algunos profesionales habían detectado un problema existente al tratar con tantos y tan diversos alumnos cada año en las diferentes asignaturas que estos impartían.

El asunto en cuestión trataba de la dificultad con la que se encontraban los docentes para memorizar los nombres de todos los alumnos que tenían en cada una de las asignaturas, ya que no disponían de ninguna herramienta que facilitara o cubriese esta necesidad. Como consecuencia, se ha desarrollado *NamEx*.

Al conocer el problema, se debatió acerca de cuál debía ser la plataforma donde se iba a realizar su desarrollo, pudiendo ser una aplicación de escritorio o una móvil, escogiendo finalmente la segunda opción por los motivos que se van a comentar a continuación.

Desde el año 2015, la Universidad de Extremadura cuenta con una app designada con el mismo nombre, que facilita el acceso a la información de interés para la comunidad universitaria, permitiendo estar al día de los asuntos que conciernen a la UEx, así como la consulta de los planes de grado y postgrado, y del calendario académico/laboral. Asimismo, una vez que el usuario se identifica con su correo y contraseña tiene acceso a sus correos y a los servicios del Campus Virtual, todo centralizado desde una misma herramienta.



1.1. App de la UEx. *Fuente: Universidad de Extremadura.*

Esta aplicación cuenta con miles de usuarios activos, por lo que se puede afirmar que existe una cultura al respecto. Por ello, para intentar expandir su uso, se decide finalmente, realizar una herramienta para dispositivos móviles, intentando en un primer momento el desarrollo de un pequeño módulo de la aplicación existente, realizándose en última instancia una aplicación separada, la cual podrá ser en un futuro integrada a los servicios que ya ofrece la primera.

Una vez decidida la plataforma en la que se iba a realizar el desarrollo, se piensa en cómo lograr el objetivo de un aprendizaje lúdico. Como la misma palabra indica, se deben desarrollar modos de juegos que favoreciesen la absorción del conocimiento mediante su utilización. En consecuencia, se pensó en aplicaciones ya existentes que sirviesen como pasatiempo, basadas en la resolución de un problema o incógnita que involucrase un cierto grado de conocimiento general o de algún tema específico.

Después de realizar una tarea de búsqueda, se descubrieron varias aplicaciones, tales como: *¿Cuánto sabes de primaria?*, *Ahorcado*, *4 Fotos 1 Palabra*, *50x15*, *Personajes Famosos*, etc. No todas servían para el fin deseado, no obstante, sirvieron como base para el desarrollo de ideas que se aplicarían a los modos que finalmente se decidieron desarrollar, incluyendo un ahorcado entre estos.

Finalmente, se introdujo un modo con un test típico, una pregunta con cuatro posibles respuestas, ya que, aunque la aplicación va enfocada a gente implicada y con ganas de realizar un esfuerzo por aprenderse los nombres de los alumnos de una forma más lúdica, no se quiere limitar su uso a otros docentes que prefieren modos de

aprendizaje más íntegros para hacer frente al gran número de estudiantes matriculados. Además, se incluye un ranking que permite la visualización de los resultados obtenidos en las partidas, pudiendo ver de este modo la progresión obtenida y las puntuaciones de los demás participantes en la aplicación, aumentando la competitividad y las ganas de mejorar los resultados obtenidos.

El propósito de este amplio desarrollo es evidente y se fundamenta en los grandes beneficios que puede aportar la presente aplicación con respecto a la inexistencia de herramientas similares, cubriendo la necesidad que fue transmitida y que ha dado lugar a todo el consiguiente proyecto.

1.2 Objetivos del proyecto.

El principal objetivo de este proyecto es proporcionar a la Universidad de Extremadura (*UEx*) una aplicación cuya finalidad es facilitar la enseñanza de los nombres de los alumnos a cada profesor.

Designado con el nombre de *NamEx*, cuyo significado es la suma de las palabras "*Name*", referido a los nombres de los alumnos y "*Ex*", como reseña de la región donde se encuentra la institución universitaria donde se procede al desarrollo.

1.2. Logo de NamEx. *Fuente: Caballero, A. y García, M.*

En base a este objetivo principal, a continuación, se enumeran los objetivos generales y específicos que engloba el proyecto.

Estos objetivos expresan los propósitos genéricos del proyecto:

- Obtener los conocimientos necesarios referentes a las diferentes tecnologías que permitan desarrollar una aplicación móvil.
- Realizar un estudio del arte sobre las aplicaciones móviles basadas en resolución de problemas memorísticos.
- Analizar el conjunto de características y compatibilidades de las plataformas y tipos de desarrollos, así como de los diferentes sistemas de base de datos que se pueden utilizar en el desarrollo de aplicaciones móviles.
- Crear una aplicación móvil que resuelva el problema descrito en capítulos anteriores y permita al profesorado entrenar el aprendizaje del nombre de sus alumnos.

De estos objetivos generales, se desglosan los siguientes objetivos específicos:

- Identificar la existencia de aplicaciones móviles con que tengan el mismo propósito que la planteada en este Trabajo Fin de Grado.
- Analizar diversas aplicaciones de resolución memorística conocidas con el objetivo de evaluar si sus funcionalidades se pueden extrapolar al objetivo de la aplicación que se desea desarrollar.
- Realizar una comparativa entre los tipos y plataformas de desarrollo para conocer las posibilidades que mejor se adaptan a nuestras limitaciones y realizar la elección de las piezas que construirán la aplicación objetivo.
- Crear una aplicación móvil personalizada que facilite al profesorado recordar el nombre de sus alumnos, desarrollada tomando como base las funcionalidades identificadas en el estado del arte de este trabajo.
- Construir una interfaz amigable y sencilla para la aplicación móvil que se propone.

1.3 Estructura del proyecto.

Con el propósito de proporcionar una visión completa y contextualizar la memoria, a continuación, se describe un breve resumen de cada capítulo de este proyecto.

Capítulo 1: En esta sección se detalla el contexto global del proyecto, exponiendo las líneas generales del mismo. Además, se describen los objetivos generales y específicos de este, acompañado de un epílogo general de la estructura del documento.

Capítulo 2: Referido al estado del arte sobre plataformas de desarrollo móvil, tipos de desarrollo y aplicaciones de juegos basados en la resolución de problemas o con un funcionamiento similar al que da vida esta aplicación.

Capítulo 3: Apartado en el que se describe la arquitectura del entorno físico y tecnológico disponible, y en el que se realiza, a partir de la base teórica obtenida en el capítulo anterior, un estudio profundo de los diferentes tipos de desarrollos y *frameworks* consideradas más eficientes y con mayor proyección, que han sido elegidas para el desarrollo de este proyecto. En este capítulo se realizan también varias comparativas de las características, ventajas, desventajas y funcionalidades de las herramientas anteriormente estudiadas, llegando a la justificación de esta investigación.

Capítulo 4: En este capítulo se describe el diseño y la construcción de la aplicación *NamEx*, pudiendo observar los cambios y la evolución de la herramienta desde su etapa inicial hasta el final de su desarrollo.

Capítulo 5: En esta parte se exponen los problemas e inconvenientes superados durante el transcurso del proyecto.

Capítulo 6: En esta sección se explica la planificación seguida para el desarrollo y realización de este proyecto, así como los medios utilizados para la comunicación y coordinación con los tutores del proyecto.

Capítulo 7: Por último, en este capítulo se exponen las principales conclusiones del proyecto y las limitaciones encontradas en el desarrollo de este. Asimismo, se definen posibles líneas de trabajo futuro. Destacar, que al igual que todo el trabajo realizado, las líneas de futuro tienen la finalidad de terminar de dar solución al desafío que da lugar a *NamEx*, para que pueda ser utilizada en un futuro por la comunidad docente de la Universidad de Extremadura.

Bibliografía: Se especifican las citas o menciones utilizadas durante el proyecto. El formato de las citas atiende la norma ISO-690.

Anexo I: En este apartado se recoge el código de la clase utilizada para realizar las preguntas de los diversos modos de juego que contiene *NamEx*, así como el algoritmo de generación de estas.

Anexo II: En este epígrafe se incluye el método que recoge la pulsación de cada botón del modo de juego fácil *Encuentra el orden*, el cual es primordial para su funcionamiento.

Anexo III: Contiene el código del comodín que coloca la primera letra de la palabra buscada en el juego *Encuentra el orden*, el cual tiene una funcionalidad importante del mismo.

Anexo IV: En este apartado se encuentra el código del comodín que coloca una letra aleatoria en el juego *Encuentra el orden*, importante en la funcionalidad del mismo.

Anexo V: En este epígrafe se encuentra el código del método que recoge la pulsación de todos los botones que contienen las letras del abecedario, en el modo de juego *Ahorcado* y que es fundamental para su correcto funcionamiento.

Anexo VI: Este apartado presenta los pasos a seguir para la creación de un proyecto en *Firebase*, la base de datos alojada en la nube elegida en la investigación del proyecto.

Anexo VII: En este epígrafe se detalla el proceso a seguir para la instalación del *framework Android Studio*, utilizado para el desarrollo de *NamEx*.

Anexo VIII: En este epígrafe se detalla el proceso a seguir para la integración del proyecto de *Firebase* creado anteriormente y el proyecto de Android Studio que acaba de ser creado.

Capítulo 2

2. Estado del arte.

Comenzamos analizando el estado del arte, organizando este apartado en tres secciones: un primer nivel en el que se estudian diferentes aplicaciones alternativas que pudieran existir con este funcionamiento u otro similar; un segundo nivel en el que se ve una visión general de diferentes juegos basados en la resolución de problemas, para obtener ideas; y una tercera capa en el que se observa el estado del arte del desarrollo actual de aplicaciones móviles. De esta manera, se crearía la base teórica sobre la que se sustentará y fundamentará el desarrollo de nuestro proyecto.

2.1 Aplicaciones alternativas.

Antes de comenzar con el desarrollo de la aplicación se realizó un estudio para conocer las herramientas que pudieran existir en este ámbito, en las que, de alguna manera, se facilitara el aprendizaje a los docentes del nombre de los alumnos inscritos en las asignaturas que estos enseñaban.

Después de realizar una búsqueda profunda, no se obtuvo resultado alguno de una aplicación con estas características o similares, aunque sí se encontraron diversos estudios y *papers* donde se recogían aplicaciones desarrolladas para estudiantes, entre ellos se destacan dos. Un *paper* acerca del uso de aplicaciones móviles para el mejorar el aprendizaje usando Moodle, y un artículo donde se estudiaban diferentes estrategias utilizadas para la gamificación en el ámbito universitario, midiendo su impacto.

En este primer *paper* de (Aguado, 2011) realizado por profesores de esta misma universidad comentan los beneficios de la inclusión de aplicaciones móviles que generan interés en el alumnado, facilitando el aprendizaje por descubrimiento y basado en problemas, explicando puntos positivos y negativos, junto con detalles más técnicos de acceso e intercambio de datos con la plataforma Moodle.

En el segundo artículo de (Cernuda del Río, 2014), se estudian diversas estrategias de gamificación, algunas de ellas aplicadas en el grado de Biología Sanitaria en la Universidad de Alcalá, con unos resultados bastante favorables, con lo que se demuestra los beneficios reales y tangibles de la incorporación de estas estrategias y del concepto de gamificación al aprendizaje.

Como sumario de todo lo anterior, se puede observar que las herramientas existentes en este ámbito están centradas en el alumnado, siendo nuestra aplicación bastante novedosa debido a que no hay nada que se le parezca, con los beneficios intrínsecos de incluir la gamificación, que es una parte fundamental de esta.

2.2 Herramientas populares en cuestiones memorísticas.

Al no haberse encontrado ninguna herramienta similar de la que poder extraer ideas que aseguren la realización de un desarrollo con garantías de satisfacción general por parte de los usuarios que lo prueben, se decidió buscar otras aplicaciones que fuesen populares cuyo funcionamiento se basara en la realización de ciertas cuestiones memorísticas que el usuario debiera resolver. Finalmente, de ciertas aplicaciones que se encontraron se extrajeron ideas que serían aplicadas en el desarrollo futuro de nuestra aplicación. Entre ellas se encuentran las siguientes:

2.2.1 4 Fotos 1 Palabra.

Fue desarrollada por la empresa *Lotum* y cuenta con más de 10 millones de descargas en el *Play Store* y una puntuación media de 4,4 de 5 en la *Apple Store*, desarrollado en 2012 y cuyo funcionamiento es el siguiente: En la pantalla aparecen cuatro fotos relacionadas con el término que se debe encontrar, como pistas nos aparece el número de letras que contiene dicha palabra, teniendo un espacio en la pantalla donde se colocará cada una de los caracteres que el juego introduce al pulsar el usuario el botón que lo contiene, como se puede observar en su interfaz gráfica.

2.1. Aplicación 4 Fotos 1 Palabra. Fuente: 3djuegos.

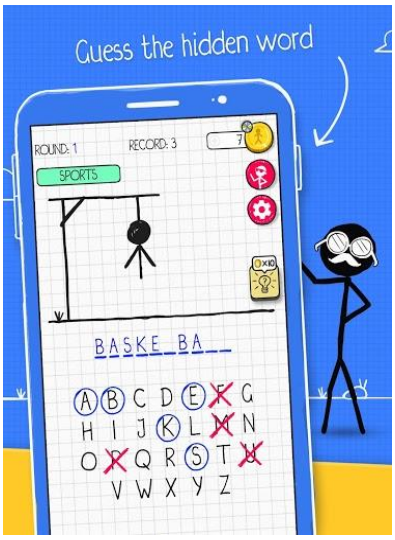
2.2.2 50x15 ¿Quién quiere ser millonario?

50x15, más conocido como ¿Quién quiere ser millonario? es un concurso televisivo emitido en varios países a nivel mundial, entre los que se encuentra España, en el que ha sido difundido en distintas ocasiones y del cual se ha anunciado un regreso de unos pocos capítulos este año 2020. A partir de esta idea se han desarrollado diversos mini juegos *online* con el mismo formato, bastante jugados y aplicaciones móviles, entre las que se encuentra una con más de 100 mil descargas, de donde se obtuvo la idea de añadir comodines a los diversos modos de juego que se iban a diseñar.

2.2. ¿Quién quiere ser millonario? *Fuente: semana*

2.2.3 Ahorcado.

Ahorcado, es un juego muy conocido que consiste en dibujar un hueco por cada letra que contiene la palabra, añadiendo una parte del cuerpo del ahorcado por cada carácter que el jugador va diciendo que no esté incluida en el término buscado, cuenta con una aplicación con el mismo nombre, que fue lanzada por el estudio español de desarrollo de videojuegos *TellmeWow*. Cuenta con más de 10 millones de descargas, resultando entretenido a la par que enriquecedor.



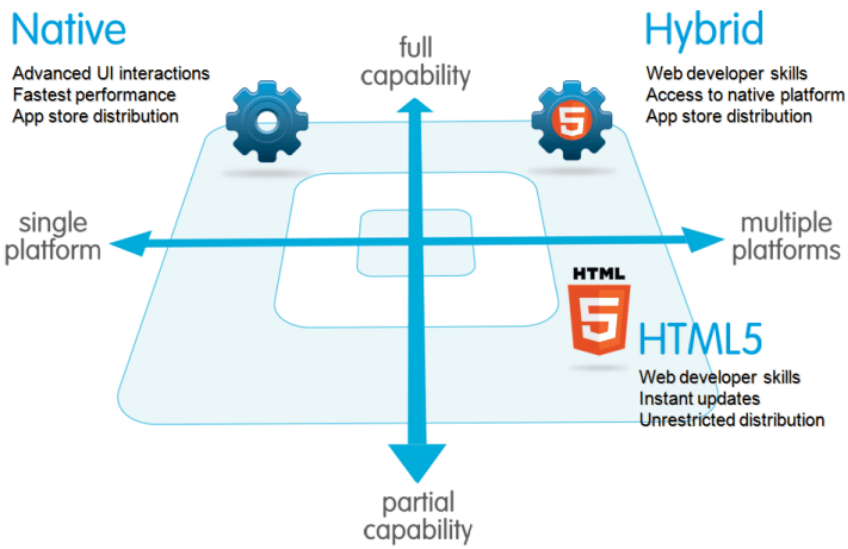
2.3. Ahorcado. *Fuente referenciada (apkamp.com, 2020).*

2.3 Tecnologías.

En esta tercera parte vamos a estudiar diferentes conceptos acerca del desarrollo de aplicaciones móviles, plataformas utilizadas para almacenar y de las que obtener datos, y lenguajes de programación utilizados durante este proceso, que nos ayudará a conocer las tecnologías que están siendo usadas en este ámbito en el presente.

Inicialmente, se van a exponer las distintas plataformas que existen a la hora realizar una aplicación de estas características, que se engloban en diferentes grupos en función de su naturaleza, los principales son: Nativas, desarrolladas de modo específico para cada sistema operativo (*iOS*, *Android* y *Windows Phone*), de los que hablaremos a continuación, siendo su principal ventaja son el rendimiento y el acceso

directo a los servicios nativos del dispositivo, debiendo, por el contrario, realizar aun aplicación diferente para cada plataforma, lo que provoca que los costes aumenten; Web: aplicaciones que se desarrollan con *Javascript*, *HTML* o *CSS*, son compatibles y se adaptan a las diferentes plataformas de los dispositivos móviles, sin embargo, no se consigue aprovechar de forma completa los recursos que estos ofrecen. Híbridas, que combinan aspectos de las apps nativas y webs, en función de las necesidades concretas, limitando costes, debido a que un único desarrollo sirve para las distintas plataformas existentes y complejidad, no logrando, por el contrario, obtener el mayor rendimiento posible, según nos indica (Cadenas, 2019).

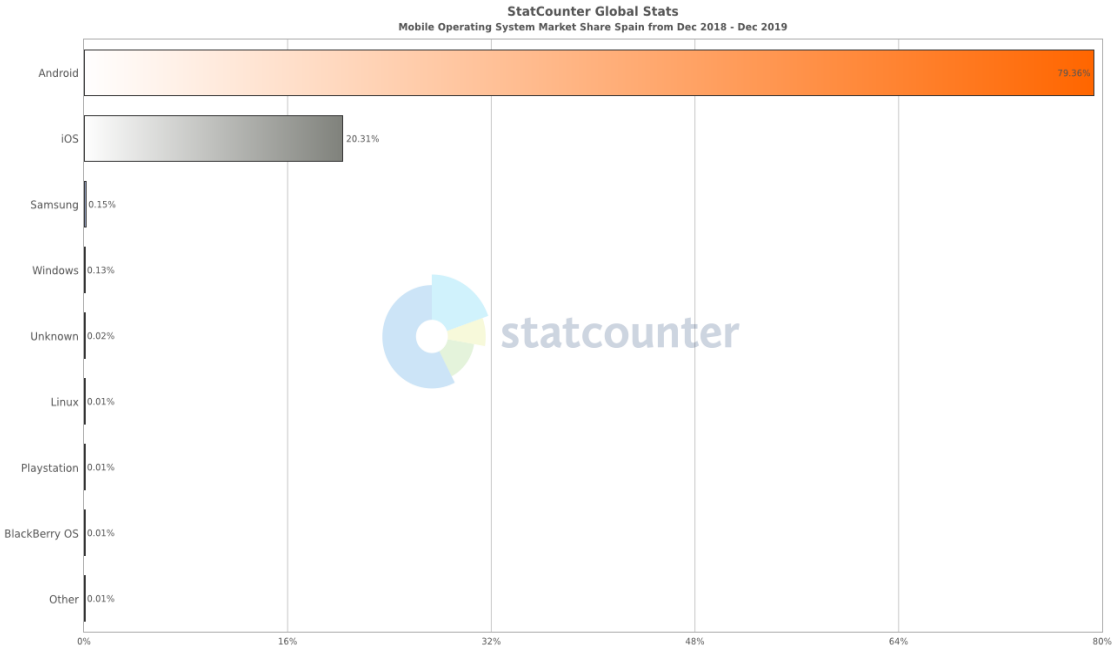


2.4. Tipos de desarrollos móviles. *Fuente referenciada (Developer.salesforce.com, 2019).*

Continuamos con los diferentes sistemas operativos que han sido nombrados en el párrafo superior, de los que describiremos los dos que sobresalen sobre el resto, *iOS* y *Android*. El primero de ellos, es el sistema operativo utilizado en todos los dispositivos fabricados por la multinacional *Apple*, que no permite su instalación en hardware de terceros. Utiliza el lenguaje de programación Swift, que requiere de un ordenador de la compañía junto con el *framework XCode* para realizar su desarrollo, no siendo este *Open Source*, lo que provoca que la comunidad no sea tan extendida. Por último, vamos a hablar de *Android*, el cual otro sistema operativo cuyo propietario y principal promotor es *Google*, basado en el *Kernel* de *Linux*. Cuenta con una gran comunidad de desarrollo, que puede integrar e implementar funciones personalizadas en los dispositivos, debido a que se encuentra en *Open Source*, siendo el código libre, es

decir, que puede ser estudiado, modificado y utilizado con cualquier fin. La plataforma en la que se realiza el desarrollo es *Android Studio*, en los lenguajes de programación *Java* o *Kotlin*.

Como se puede ver en la siguiente gráfica, en este último año han sido los más utilizados, tanto a nivel mundial como a nivel estatal.

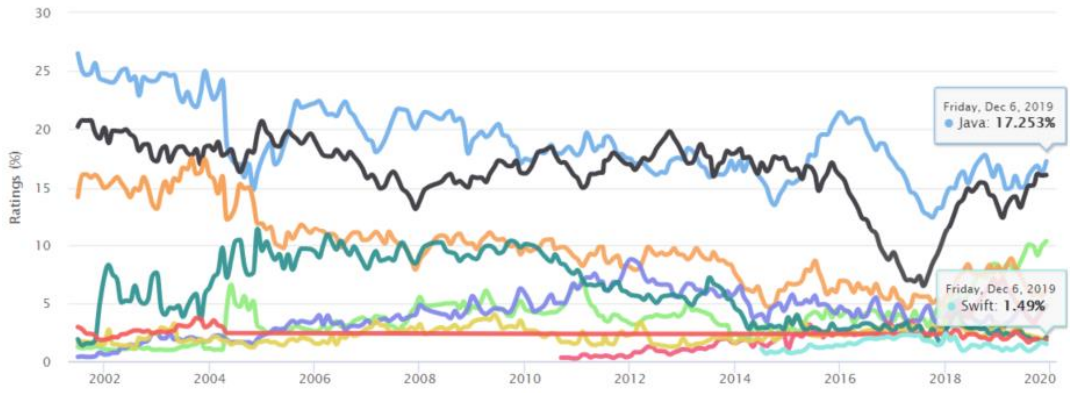


2.5. Cuota de mercado de los sistemas operativos móviles en España.
Fuente referenciada (statcounter.com, 2019).

Este informe realizado mensualmente recoge la cuota de mercado en España, de cada uno los sistemas operativos móviles, siendo sin duda *Android* el más utilizado, con más del 79.36%, seguido por *iOS* con un 20.31%, perseguidos por *Samsung* o *Windows*, con menos de un 0.15%.

Cada uno de estos sistemas operativos utiliza un lenguaje en específico como ya se ha comentado, los tres más utilizados son *Java*, *Kotlin* y *Swift*. El primero de ellos fue desarrollado por *James Gosling* y publicado en 1995, siendo un lenguaje muy versátil que puede ser usado para realizar desarrollos en *Android*, y que a día de hoy según el índice *TIOBE* es el lenguaje más usado del mundo; *Kotlin* también permite desarrollar aplicaciones *Android*, fue creado por la compañía *JetBrains*, publicado en el año 2012, teniendo ambos lenguajes licencia de código libre, hasta la versión 11 del primero de ellos, que deja de ser gratuita. En octubre de 2017, *Google* decidió darle soporte en el set de desarrollo de *Android Studio*, el *framework* utilizado para realizar aplicaciones

para dicho sistema operativo. Que en la actualidad se encuentra en el puesto 30, según el índice *TIOBE* de los lenguajes más usados; Por último, Swift es un lenguaje multiparadigma creado por *Apple* y enfocado al desarrollo de aplicaciones para *iOS* y *macOS*. Es relativamente joven, solo lleva cuatro años en el mercado desde que se lanzase su versión beta, el cual, alcanzó según el índice *TIOBE* el top diez en marzo de 2017, que conserva a día de realización de este trabajo (TIOBE Software BV, 2020).



2.6. Índice TIOBE de los lenguajes de programación más utilizados.
Fuente referenciada (Tiobe Software BV., 2019)

Para terminar, solo hace falta estudiar el entorno de donde se van a obtener los datos de los alumnos y, donde van a ser almacenados estos junto con otros datos generados durante la utilización de la aplicación. La plataforma donde se encuentran almacenados los datos con los que se van a realizar los diferentes modos de juego es el Campus Virtual, que está implementado usando Moodle, una plataforma enfocada a la enseñanza. Moodle significa *Modular Object-Oriented Dynamic Learning Enviroment*, o lo que es lo mismo, Entorno de Aprendizaje Dinámico Orientado a Objetos y Modular, que fue fundado por el pedagogo e informático australiano *Martin Dougiamas* y cuyas ventajas son tantas que a día de hoy cuenta con más de 130 millones de usuarios registrados a nivel mundial, activo en más de 100.000 páginas webs, como nos indica (Tropical Server S.L., 2020).

Por último, para almacenar los datos propios de la aplicación se estudiaron las bases de datos, que pueden ser de dos tipos, *SQL* y *NoSQL*, de las que se escogió una de cada para ser analizadas y en un futuro tomar la decisión de cuál es la más adecuada para el desarrollo. Del primer tipo de base de datos se escogió *SQLite*, base de datos compatible tanto con *Android* como con *iOS*. Es un motor de base de datos de código

abierto, ligero, autónomo, de configuración simple y sin servidor, que se caracteriza por almacenar información persistente de forma sencilla. Del segundo grupo se eligió *Firebase Realtime Database*, también compatible con *Android* y con *iOS.* Es una base de datos de *Google* que permite almacenar y sincronizar datos que se van a alojar en la nube y que van a poder ser sincronizados con todos los clientes en tiempo real, manteniéndose disponibles cuando la app no tiene conexión.

Por conclusión, estudiaremos las plataformas disponibles para realizar desarrollos de aplicaciones móviles, junto con los lenguajes que se utilizan en cada una de ellas, el tipo de desarrollo que se va a seguir, además de la plataforma desde la cual se van a obtener los datos de los alumnos y las bases de datos donde se van a almacenar y recuperar datos que genere el uso de la aplicación.

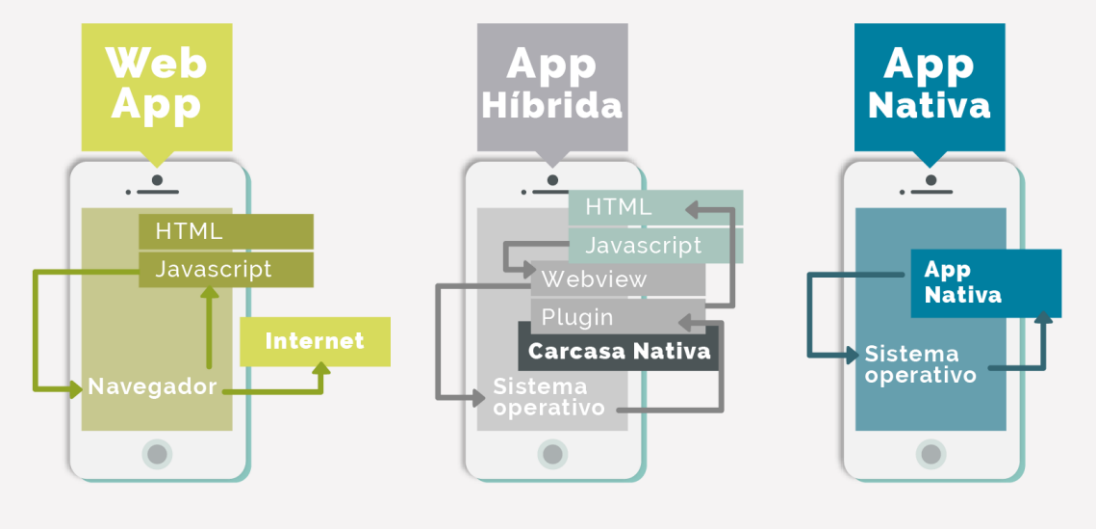
3. Marco conceptual y análisis.

En esta sección, se efectúa un análisis y una revisión general de la tecnología utilizada para construir aplicaciones móviles, entre los que se encuentran el tipo de desarrollo que se va a realizar, analizando las ventajas que cada uno de ellos proporcionan. A continuación, se exponen las diferentes plataformas disponibles para realizar el desarrollo, estudiando brevemente cada una de ellas.

Por último, se examina una base de datos de cada tipo con sus ventajas y se expone brevemente la herramienta Moodle necesaria para la obtención de los datos de los alumnos, finalizando el capítulo con la justificación y elección adecuada de las herramientas que se utilizarán durante el desarrollo de este proyecto.

3.1 Tipos de desarrollo.

Antes de empezar con la construcción de la aplicación, se debe elegir qué tipo de desarrollo es el ideal para este proyecto. En este apartado, se van a exponer cada uno de ellos, analizando las ventajas que cada cual ofrece.



3.1. Tipos de aplicaciones según su desarrollo. *Fuente referenciada (Gsoft.com, 2019).*

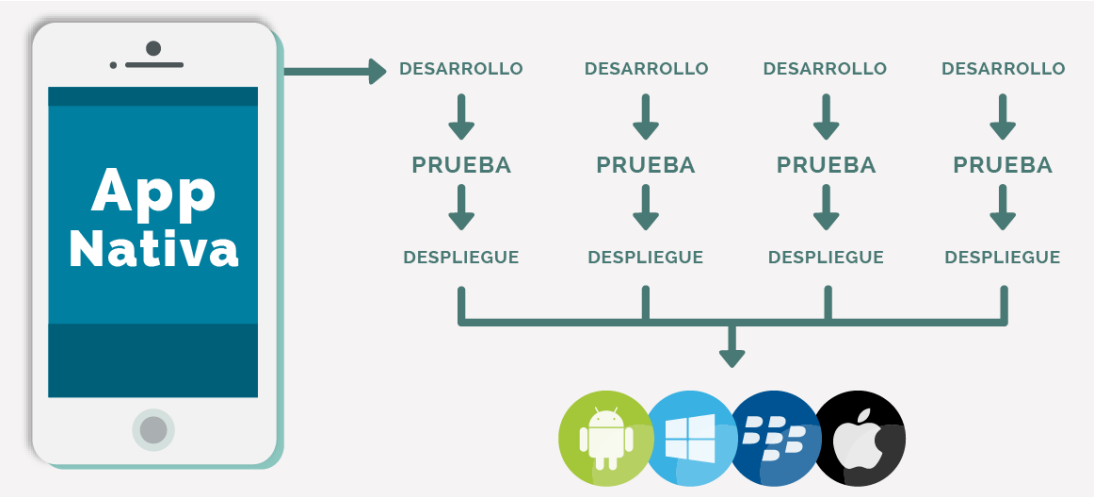
3.1.1 Aplicaciones nativas.

Las aplicaciones nativas son aquellas construidas para una plataforma específica, ya sea *Android*, *iOS*, *Windows Phone*, etc. Se realizan con el lenguaje de programación propio de cada equipo y funcionan en estos sin necesidad de ningún programa externo. Las principales ventajas, como nos indican (Sesma, 2018), son su adaptación completa al dispositivo, pudiendo utilizar todas las funcionalidades que este ofrece (cámara, GPS, acelerómetro, etc.) de una manera más sencilla, debido a que existen menos capas para acceder a esta, junto al código que está optimizado para se lleven a cabo rápidamente. De esta manera, se permite el acceso a las últimas características de la plataforma tan pronto como estén disponibles ya que no hay que esperar a la construcción de *plugins* que permitan su uso y que, generalmente, tardan tiempo en ser desarrollados.

Como consecuencia de lo anterior, se consiguen aplicaciones más rápidas, mejor optimizadas y con un mejor rendimiento, que es otra de las ventajas a destacar. La calidad percibida comienza con el rendimiento. Durante la compilación a código máquina, estas plataformas de desarrollo nativo aprovechan las funcionalidades de los componentes de la *UI*, así *UITableView* o *RecyclerView* pueden mostrar listas con transiciones complejas, animaciones avanzadas y gran suavidad de desplazamiento consumiendo mucha menos energía (Sesma, 2018).

Asimismo, en las aplicaciones nativas, se pueden crear tantos hilos de ejecución como se desee. Se soportan las corrutinas, pudiéndose utilizar servicios para realizar trabajo en *background* (Sesma, 2018).

Como conclusión y sumario, las principales ventajas de este tipo de desarrollo son la capacidad de construir aplicaciones en todo tipo de dispositivos como *Wereables* (*Apple Watch, Android Wear)*, televisiones e incluso coches. Proporcionan una mayor seguridad al reducirse las capas de complejidad previstas de ser atacadas, mejoran la duración de la batería debido a que el código está optimizado para la arquitectura, haciéndose un uso más adecuado de los distintos núcleos de *CPU/GPU*, logrando una mayor velocidad, rendimiento y el aprovechamiento de forma total del hardware del dispositivo. Como punto negativo, se encuentra que el código de la aplicación realizada no es portable, debido a lo cual, hay que realizar un desarrollo diferente para cada sistema en el que se quiera lanzar la aplicación, también para las distintas actualizaciones o versiones que se realicen, por lo que se aumenta mucho tanto los tiempos de desarrollo como los costes del mismo (Sesma, 2018).



10. Flujo del desarrollo de una aplicación nativa. *Fuente referenciada (Gsoft.com, 2019).*

3.1.2 Aplicaciones web.

Las aplicaciones web son aquellas desarrolladas con tecnologías web a las que se accede mediante un navegador a través de Internet o Intranet, lo cual es, precisamente, lo que le confiere la mayoría de ventajas de este tipo de aplicaciones y de inconvenientes, al mismo tiempo (Cadenas, 2019).

Entre las ventajas se encuentran la sencillez de desarrollo y su menor coste, debido a que un único desarrollo que es distribuido permite a todos los usuarios acceder a este a través del navegador, sea cual sea el dispositivo desde el que intente ingresar, ya sea un móvil o un dispositivo de escritorio, es decir, son multiplataforma, como se puede observar en la siguiente imagen (Cadenas, 2019).

11. Flujo del desarrollo de una aplicación web. *Fuente referenciada (Gsoft.com, 2019).*

Sin embargo, para que la aplicación se adapte a los distintos tipos de dispositivos, se debe usar un diseño *responsive*, es decir, un diseño web adaptable que permite la correcta visualización de esta. Se trata de redimensionar y colocar elementos de forma que se adapten al ancho de cada dispositivo, consiguiéndose una mejor experiencia de usuario (Cadenas, 2019).

Las principales ventajas, según (Soluciones I.P. Hispanas S.L., s.f.), de este tipo de diseño adaptativo son la simplicidad en el mantenimiento, debido a que únicamente existe una web para todos los dispositivos, por lo cual, teniendo en cuenta dicho punto, solo existe una única *url* de acceso que favorece el posicionamiento (*SEO*), evitando contenido duplicado en distintas *urls*. Se mejora la usabilidad y la experiencia de usuario, al permitir, por ejemplo, cambiar de orientación su dispositivo y que el sitio web se adapte perfectamente a la nueva visualización.

El principal inconveniente de este tipo de aplicaciones es que ofrecen una experiencia de uso mucho más limitada, ya que no puede, al menos de forma sencilla y directa, acceder a todas las características que ofrece el dispositivo, como

rendimiento gráfico, GPS, acelerómetro, diversos sensores, etc. Además, podría ofrecer un nivel bajo de seguridad, debido a que esta depende directamente del navegador que se utilice. El coste de desarrollo no es muy alto, pero se eleva respecto al desarrollo web estándar, teniendo una mayor dificultad técnica (Cadenas, 2019).

3.1.3 Aplicaciones híbridas.

Las aplicaciones híbridas son una mezcla de las dos anteriores. En ellas, se aprovecha la versatilidad del desarrollo de una aplicación web con *HTML5*, *CSS* y *Javascript*, permitiéndose cierto uso de las funcionalidades *hardware* disponibles en el desarrollo de aplicaciones nativas, al contrario que en los desarrollos web.

Según (Sesma, 2018), en sus inicios, para lograr esta característica, la aplicación web no era accedida a través del propio navegador, sino que era visualizada a partir de un componente llamado *WebView*, que viene a ser un navegador integrado en una aplicación nativa, que puede acceder a estos recursos *hardware* a través de *plugins* específicos, como ya se había comentado en el apartado anterior de las aplicaciones nativas.

Una aplicación híbrida añade un cierto nivel de indirección entre su código fuente y el ejecutable con el que interactúa el usuario, que no existe o es más ligero en los desarrollos nativos (Sesma, 2018).

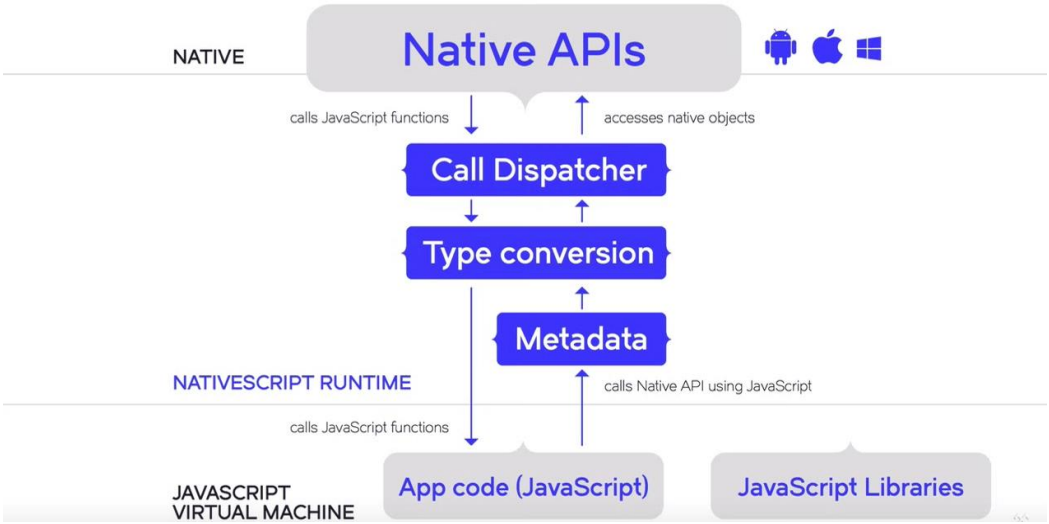
La principal ventaja es la portabilidad del diseño, como se puede ver en la siguiente imagen, en la que se observa que un único desarrollo, con pequeños cambios o ajustes mínimos necesarios, es desplegado en múltiples plataformas diferentes.

12. Flujo del desarrollo de una aplicación híbrida. *Fuente referenciada (Gsoft.com, 2019).*

Este planteamiento está limitado a la calidad de la implementación del navegador que se utilice, siendo el desarrollo de escritorio más factible de realizarse, consiguiendo resultados difícilmente distinguibles de una aplicación nativa. No ocurre lo mismo para entornos móviles (Sesma, 2018).

Hoy en día, el desarrollo no consiste en un *WebView* a toda pantalla, ahora se mantiene un *UI* formado por controles nativos de *iOS* o *Android*, que están configurados y colocados mediante *CSS* y *SGML* (similar a *HTML*). Son controlados desde una máquina virtual JS, que es quien ejecuta el programa, interactúa con el sistema operativo y abstrae las diferencias entre las diversas plataformas (Sesma, 2018).

Como consecuencia, se unifica la experiencia de usuario nativa con la experiencia de desarrollo web, pudiéndose acceder a las funcionalidades que ofrece el dispositivo casi por completo, mediante el uso de diferente *plugins* de conversión de tipos, como se puede observar en la siguiente imagen (Sesma, 2018).



13. Acceso de las aplicaciones híbridas a funcionalidades nativas.
Fuente referenciada (Docs.nativescript.org, 2019).

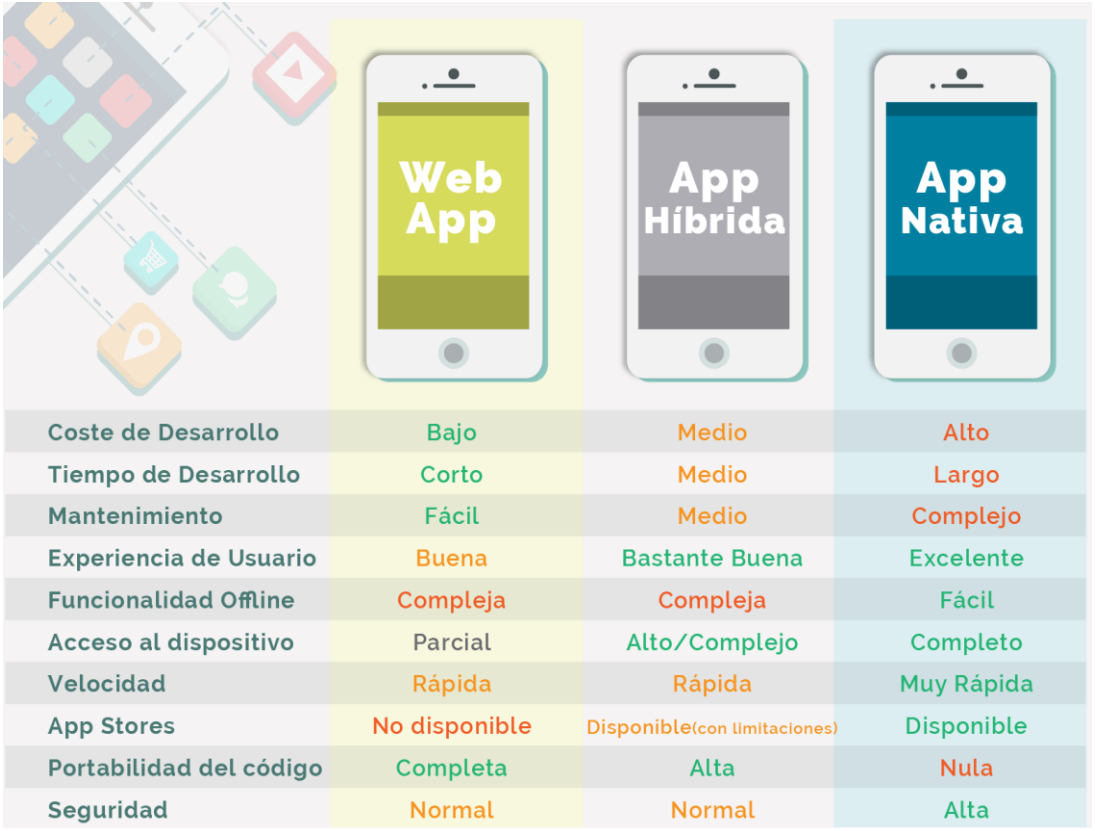
En conclusión, se va a realizar un pequeño resumen con las ventajas de este tipo de desarrollo. Entre ellas se encuentran la portabilidad que viene dada porque son desarrollos multiplataforma; el ahorro de costes al realizar un único diseño, reduciéndose también el *time to market*, tiempo que se consume hasta sacar la aplicación al mercado; la disminución de la complejidad; y el tiempo de realización de actualizaciones. Por el contrario, la velocidad y el rendimiento de estas no son tan altos como en las aplicaciones nativas, además, de que el acceso a las nuevas funcionalidades implementadas en los sistemas operativos no es inmediato, debido a que estos se realizan utilizando *plugins*, que tardan en ser construidos.

3.1.4 Tabla comparativa.

En esta sección, se efectúa la descripción de cada una de las funcionalidades y características que han sido analizadas en los distintos tipos de desarrollos estudiados: Nativo, Web e Híbrido, realizándose la comparativa de las propiedades y operatividades, y utilizándose un color verde para señalar la mejor o mejores opciones, un color anaranjado para indicar una opción aceptable y, por último, un color rojizo que marca la peor opción o la más compleja.

En la parte de justificación de la investigación, se seleccionará el desarrollo que mejor se adapta a nuestras limitaciones. Nos basaremos en la siguiente tabla para

comparar y realizar la elección de una pieza importante para la construcción de *NamEx*.



Coste de Desarrollo	Bajo	Medio	Alto
Tiempo de Desarrollo	Corto	Medio	Largo
Mantenimiento	Fácil	Medio	Complejo
Experiencia de Usuario	Buena	Bastante Buena	Excelente
Funcionalidad Offline	Compleja	Compleja	Fácil
Acceso al dispositivo	Parcial	Alto/Complejo	Completo
Velocidad	Rápida	Rápida	Muy Rápida
App Stores	No disponible	Disponible(con limitaciones)	Disponible
Portabilidad del código	Completa	Alta	Nula
Seguridad	Normal	Normal	Alta

14. Comparación de los diferentes desarrollos móviles. *Fuente referenciada (Gsoft.com, 2019).*

3.2 Plataformas de desarrollo.

Como pieza importante del proyecto está la plataforma de desarrollo utilizada para construir la aplicación. Antes de realizar la selección, se debe estudiar las características de estas para conocer cuál se adapta mejor a las necesidades y limitaciones que se tienen. Con este fin, se van a exponer dos herramientas de desarrollo nativo y otras dos de desarrollo híbrido, habiéndose descartado el desarrollo web, debido a que no se parece al modelo ya existente de la aplicación móvil propia de la Universidad de Extremadura.

3.2.1 Android Studio.

Android Studio es una plataforma de desarrollo nativa. Como bien nos indican en su guía de usuario, es el entorno de desarrollo integrado (IDE) oficial para la construcción de aplicaciones *Android*, basado en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores que ofrece dicho entorno, (Android Inc., 2019), considera que *Android Studio* ofrece más funciones que aumentan su productividad, tales como:

- Un sistema de compilación flexible basado en *Gradle*.
- Un emulador rápido y cargado de funciones.
- Un entorno unificado donde se puede desarrollar para todo tipo de dispositivos *Android*.
- Permite la aplicación de cambios para insertar nuevos aspectos al código y recursos a la aplicación en ejecución, sin necesidad de reiniciar esta.
- Está integrado con *GitHub* y posee plantillas de código que facilitan la compilación de funciones de aplicaciones comunes, pudiéndose también importar códigos de ejemplo.
- Posee una amplia variedad de marcos de trabajo y herramientas de prueba y depuración del código.
- Tiene herramientas de *Lint* para identificar problemas de rendimiento, usabilidad y compatibilidad de la versión, entre otros.

Para terminar, es importante exponer que la utilización y descarga de este *framework* no conlleva un precio acarreado, es gratuita y compatible con la mayoría de sistemas operativos punteros del mercado (Windows, Mac, Linux, Chrome OS). Sin embargo, se debe conocer el lenguaje de Java o Kotlin, de los que existe mucha documentación, además de cursos gratuitos y soluciones a un gran número de problemas gracias a la comunidad que existe en torno a esta herramienta (Android Inc., 2019).

3.2.2 XCode.

XCode es una plataforma de desarrollo nativa. Es un entorno de desarrollo, siendo uno de los *softwares* más complejos que pueden existir en un ordenador, disponible de manera gratuita en la *Apple Store*, para dispositivos de la misma marca. El lenguaje de programación utilizado es *Swift*, que nació en 2014, por lo que es relativamente joven, aunque su uso no para de expandirse (Apple Inc., 2020).

Asimismo, puede utilizarse para crear prototipos rápidos y probar su aplicación en un entorno simulado cuando un dispositivo real no está disponible. Permite el uso de instrumentos para clasificar y analizar la aplicación, mejorar el rendimiento y encontrar problemas de memoria. Estos recolectan datos permitiendo su visualización. Además, permiten crear modelos de aprendizaje automático para las aplicaciones desarrolladas (Apple Inc., 2020).

Por último, es un entorno de desarrollo muy completo, teniendo como limitación la de conocer el lenguaje *Swift*, que permite desarrollos de códigos bastante limpios y sencillos, además de necesitar un equipo de la marca *Apple* para su utilización.

3.2.3 PhoneGap / Apache Cordova.

PhoneGap pertenece a Adobe Systems, que permite desarrollar aplicaciones híbridas mediante tecnología web *HTML5*, *CSS3* y *JavaScript*. Utiliza una API que permite acceder a elementos de hardware del sistema como la cámara, almacenamiento, acelerómetro, etc. y favorece la integración con otros frameworks como por ejemplo *jQuery Mobile*. Además, *Apache Cordova* es la versión de código abierto de *PhoneGap* (Naharro, 2019).

Al contrario que otros *frameworks PhoneGap* /*Apache Cordova* se encarga de la integración dentro del sistema operativo móvil y da acceso a los servicios de éste. Por este motivo, su uso muchas veces viene acompañado de algún otro *framework* disponible, que en muchos casos sólo ofrecen la parte de la interfaz de usuario en *HTML*, *CSS* y *JavaScript*(X), y como ya se ha comentado en el estado del arte, la mayor ventaja es el desarrollo de una aplicación multiplataforma, sin que se deba

conocer el lenguaje propio de las herramientas nativas. Sin embargo, se debe poseer buenos conocimientos en los lenguajes que utiliza dicho *framework* (Naharro, 2019).

3.2.4 Ionic.

Ionic es uno de los *frameworks* más famosos para el desarrollo de aplicaciones híbridas, para crear aplicaciones móviles y de escritorio mediante tecnologías web (*HTML, CSS, JavaScript)*. Es *open source*, pero no es gratuito, la tarifa más barata que ofrecen son 42 euros al mes contratando el servicio durante un año. *Ionic* se centra en la experiencia de interfaz de usuario. Se integra perfectamente con otras bibliotecas y *frameworks*, tales como Angular, aunque también puede utilizarse de forma independiente sin un *framework* de *front-end* mediante la inclusión de un simple *script*. Actualmente, tiene integraciones oficiales con *Angular* y *React*, y el soporte para *Vue* está en desarrollo (Wiegert, 2019).

Ionic integra una capa de diseño con estilos *css* y recursos como iconos. Estos estilos pueden ser adaptados a los estándares de diseño de las plataformas *Android* e *iOS*. Otra de sus muchas ventajas es que tiene su propio IDE de desarrollo que facilita el diseño y programación de aplicaciones (Naharro, 2019).

3.3 Bases de datos.

Como se ha expuesto en el estado del arte, se van a analizar dos bases de datos diferentes, *SQLite* y *Firebase Realtime Database*, siendo ambas compatible con el desarrollo tanto en *Android* como en *iOS*.

3.3.1 SQLite.

La primera de ellas, *SQLite*, es una biblioteca que implementa un motor de base de datos *SQL* transaccional, ligero, autónomo, que no necesita de un servidor

intermediario en la comunicación con esta, siendo los datos accedidos directamente, y no precisa de instalación previa a su uso (SQLite Org, [s.f.]).

El proyecto fue comenzado en el año 2000 por un grupo de desarrolladores que quieren darle soporte hasta el año 2050. El código de *SQLite* es de código abierto y es de uso gratuito para cualquier propósito, ya sea personal o comercial. Por ese mismo motivo, es una de las bases de datos más utilizadas en el mundo a día de hoy (SQLite Org, [s.f.]).

SQLite, es una librería compacta que ocupa poco espacio en disco aun teniendo todas las funcionalidades habilitadas. Existe un equilibrio entre el uso de memoria y la velocidad. Generalmente, se ejecuta más rápido cuanta más memoria tiene disponible; sin embargo, el rendimiento suele ser bueno incluso en entornos que no disponen de demasiada memoria. Dependiendo de cómo se utilice, puede ser más rápido que la entrada-salida directa del sistema de archivos del dispositivo utilizado (SQLite Org, [s.f.]).

Esta base de datos se prueba a conciencia antes del lanzamiento de cada versión y tiene una muy buena reputación por ser muy fiable. Responde correctamente a los errores de asignación de memoria y de entrada-salida de disco, que cumple las características *ACID* en las transacciones incluso si se interrumpe el sistema por fallos eléctricos. Todo ello verificado mediante pruebas automatizadas (SQLite Org, [s.f.]).

Como conclusión, se van a comentar sus ventajas, las cuales, según (Muradas, 2018), son las siguientes:

- Es gratuita, de código abierto y uso libre.
- Es ligera, ocupa poco espacio en memoria.
- Es ideal para la consulta y el almacenado de datos estructurados.
- Permite que las aplicaciones carguen tantos datos como necesite, en lugar de leer todo el disco, ahorrando tiempo y reduciendo el consumo de memoria.
- Facilita que los datos sean accedidos y actualizados mediante consultas *SQL*, reduciéndose en gran medida la complejidad del código.

- Permite que diversos programas que han sido escritos en diferentes lenguajes pueden acceder al mismo archivo de aplicación sin problemas de compatibilidad.

3.3.2 Firebase Realtime Database.

Firebase Realtime Database es una base de datos *NoSQL* propiedad de *Google*, alojada en la nube, que permite almacenar y sincronizar datos en tiempo real con todos los usuarios, manteniéndose disponibles cuando la aplicación no tiene acceso a Internet. Los datos se almacenan en formato *JSON* y es compatible con diferentes desarrollos tanto *Web*, como *iOS* y *Android* (Firebase Inc., 2019).

Desde su página web se informan de cómo funciona y cuáles son las principales funcionalidades que esta ofrece, además de incluir documentación acerca del proceso de configuración de la misma y de adición al proyecto que se esté desarrollando (Firebase Inc., 2019).

Firebase permite la compilación de aplicaciones colaborativas, debido a que proporciona un acceso seguro a la base de datos directamente desde el código del cliente. Se puede activar una opción que autoriza el almacenado de los datos en local, para asegurar su persistencia y acceso a ellos aun cuando la aplicación no tenga conexión. Cuando el usuario vuelve a tener conexión, *Firebase* sincroniza los datos locales que se han producido con las actualizaciones remotas que ocurrieron en el tiempo que se estuvo sin conexión. Los conflictos se solucionan de manera automática (Firebase Inc., 2019).

Realtime Database proporciona un lenguaje flexible de reglas basadas en expresiones para definir cómo se deben estructurar los datos y en qué momento se pueden leer o escribir, además de definir quién tiene acceso a qué datos y cómo se acceden a ellos (Firebase Inc., 2019).

Una de sus principales ventajas en cuanto a sus funcionalidades, al ser una base de datos no relacional, es su *API*, la cual está diseñada para permitir solo operaciones que se puedan ejecutar rápidamente. Esto permite crear una excelente experiencia de usuario de tiempo real que puede servir a millones de clientes sin que se vea afectada su capacidad de respuesta. Además de las funciones que ya han sido expuestas,

Firebase permite el escalamiento en varias bases de datos con uno de los planes que ofrece, el cual es de pago. El primero y más básico es gratuito, como se puede observar en la siguiente imagen obtenida de su web (Firebase Inc., 2019).

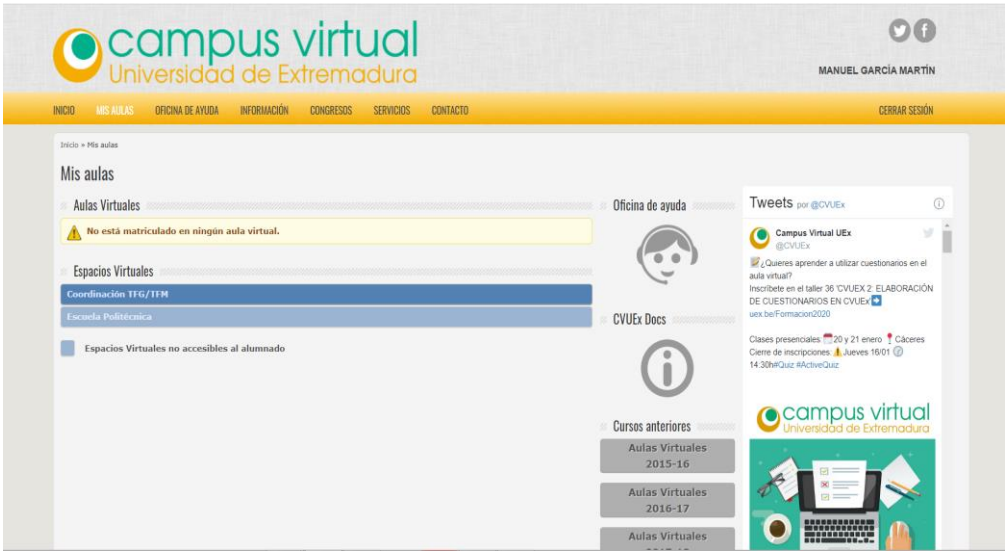
Productos	Plan Spark Límites generosos para aficionados Sin cargo	Plan Flame Precios fijos para apps en expansión USD 25 por mes	Plan Blaze Calcula los precios de las apps a gran escala. Pago por uso ✓ Se incluye el uso gratuito del plan Spark*
Realtime Database			
Conexiones simultáneas	100	200k	200k/database
GB almacenados	1 GB	2.5 GB	USD 5 por GB
GB descargados	10 GB/mes	20 GB/mes	USD 1/GB
Varias bases de dato por proyecto	✗	✗	✓

15. Paquetes de uso y coste ofrecidos para Firebase. *Fuente referenciada (Firebase.com, 2020).*

3.4 Moodle.

Moodle es una plataforma enfocada a la enseñanza, que permite a los docentes trabajar directamente con sus alumnos. Es totalmente gratuito y de código libre, por esa misma razón se convierte en una herramienta ideal, usada por más de 100 mil páginas webs y con más de 130 millones de usuarios registrados, debido principalmente, a su gran funcionamiento. Fundada por el pedagogo e informático australiano *Martin Dougiamas* (Tropical Server S.L., 2020).

Este sistema de gestión de aprendizaje converge en una web que ha sido desarrollada con tecnología *PHP*, bases de datos *MySQL* y con licencia pública *GNU GPL*, lo que permite que pueda ser instalada en casi cualquier servidor web. Como se puede ver en la web del Campus Virtual de la Universidad de Extremadura (*CVUEx*), la cual está desarrollada usando está herramienta, de donde se van a obtener los datos para la versión final de la aplicación (Tropical Server S.L., 2020).



16. Campus Virtual UEx. *Fuente: propia.*

Según afirma (Tropical Server S.L., 2020), Moodle dispone de una interfaz sencilla, que ha sido desarrollada por un equipo de psicólogos y psicopedagogos, y pretende ser bastante accesible para todo el que la usa. Su objetivo pasa por generar una buena experiencia de aprendizaje para todos los usuarios, siendo la comunicación entre estos una base fundamental de esta.

(Tropical Server S.L., 2020) considera que Moodle está orientada a la educación a distancia; sin embargo, es innegable afirmar que es un gran complemento para la educación presencial también, debido a la facilidad de interacción y comunicación que ofrece la plataforma. Toda la comunicación y trabajo se realiza a través de la red. Como consecuencia, puede ser accedida desde cualquier dispositivo, permitiendo la compartición de diferentes documentos o la descarga de apuntes de las asignaturas en las cuales se está dado de alta, entre otras opciones.

Como afirma (Tropical Server S.L., 2020), las ventajas de su uso son muchas y muy variadas:

- Cuenta con una comunidad consolidada, a nivel internacional y bastante activa, con un grupo de desarrolladores dedicados a tiempo completo a continuas mejoras en constante evolución, depurando errores y mejorando las características que ya ofrece.
- Es una herramienta de uso gratuito.

- Permite crear de distintos perfiles basados en roles.
- Ofrece distintos métodos de evaluación.
- Es compatible con cualquier navegador.
- Permite acceder a los datos mediante *web services*, facilitando la integración con otras aplicaciones que hagan uso de ella.

Por último, comentar que la educación con Moodle es bastante similar a la presencial o semipresencial, debido a la gran interacción positiva que se produce entre sus usuarios y la pedagogía que existe detrás de su diseño y desarrollo.

3.5 Justificación de la investigación.

En este apartado, uno de los que cobra mayor importancia en el proyecto, se realiza tanto la elección del tipo de desarrollo que se va a seguir, como la herramienta que permite la construcción de la aplicación, además de las bases de datos que van a ser utilizadas para la continuación del proyecto.

Primeramente, se realiza la selección y justificación del desarrollo a seguir y la herramienta que permite realizar la construcción, de la aplicación siguiendo ese escenario seleccionado.

Es importante destacar que, después de haber analizado los distintos tipos de desarrollos y herramientas de cada uno de ellos, se ha conseguido la función deseada. Se afirma, por tanto, que con cada uno de ellos se podría realizar la construcción de la aplicación, descartando primeramente el escenario web, debido a que se decide la implementación de una aplicación móvil.

Asimismo, se ha tenido en cuenta otras características no evaluadas anteriormente y que facilitan la elección más adecuada para realizar la construcción de *NamEx*. Una de estas características es la curva de aprendizaje que tienen los lenguajes utilizados en algunos *frameworks*, ya que no se dispone de todo el tiempo necesario para formarse en una tecnología y, al mismo tiempo, realizar un desarrollo que tenga unos resultados obtenidos aceptables; si bien esta no ha sido la principal característica para descartar ciertas tecnologías.

En primer lugar, como ya se ha comentado, se desestima el uso de un desarrollo web, debido a la cultura que ya existe en torno a la aplicación propia de la Universidad de Extremadura ya existente.

En segundo lugar, se ha descartado el desarrollo de una aplicación híbrida, debido al alto porcentaje que existe de usuarios que utilizan *Android*. Concretamente en España se encuentra en 79.36%, seguido por iOS con un 20.31%. Aunque el desarrollo es un poco más breve que en el desarrollo nativo, los lenguajes que se utilizan y la inexperiencia con este tipo de *frameworks*, además del precio de algunos de ellos, harían el proceso de construcción de la aplicación más complicado de lo necesario, obteniendo una cuota de mercado poco significativa.

Finalmente, se ha seleccionado el desarrollo de una aplicación nativa. De las opciones restantes, se ha desestimado la utilización del IDE *XCode*. Además de que no existe un gran porcentaje de uso en aparatos tecnológicos en nuestro país, no se dispone de *hardware* de esta marca, lo que se imposibilita el desarrollo.

La opción ganadora ha resultado ser *Android Studio*, debido al gran porcentaje de dispositivos en los que está presente, sumado a que durante el grado ya se ha trabajado con esta tecnología, por lo que durante el desarrollo no se va a trabajar con un *framework* desconocido, ni con el lenguaje utilizado, que va a ser Java. Entre Kotlin y Java, existen en la actualidad pocas diferencias técnicas entre ellos, permitiendo ambos la construcción de aplicaciones con similares resultados. Además, hay que señalar que Java se encuentra situado a día de realización de este trabajo como el lenguaje más utilizado del mundo, según el índice *TIOBE*, habiéndose estudiado durante los últimos 3 años del grado que se ha cursado.

Por otro lado, se ha realizado el estudio de las bases de datos que podían resultar interesantes durante el desarrollo del proyecto, una de cada tipo, relacional y no relacional, conocidas popularmente como *SQL* y *NoSQL*. Después de realizar estudios sobre *SQLite* y *Firebase Realtime Database*, ambas compatibles con el desarrollo en *Android*, se determina que ambas pueden ser incorporadas en el desarrollo. La primera de ellas se encarga de almacenar diferentes datos locales, como caché, generados por los usuarios durante el uso de la aplicación, almacenando datos que deben compartirse entre múltiples clientes en *Firebase*¸ que, al encontrarse en la nube, puede ser accedida por diferentes consumidores.

Finalmente, en cuanto a la utilización de Moodle para la obtención de datos, se ha desestimado, añadiéndose como trabajo futuro, debido a que la conexión con el Campus Virtual tendría que autorizarse por su parte, estando sus encargados con demasiado trabajo como para esperar que esta se produjese antes de la finalización el proyecto.

Por tanto, gracias a los estudios y comparativas realizadas, el software utilizado para la construcción de la aplicación es Android Studio, utilizándose *SQLite* y *Firebase Realtime Database* como *back-end* de la misma.

Capítulo 3

4. Diseño e implementación de la aplicación.

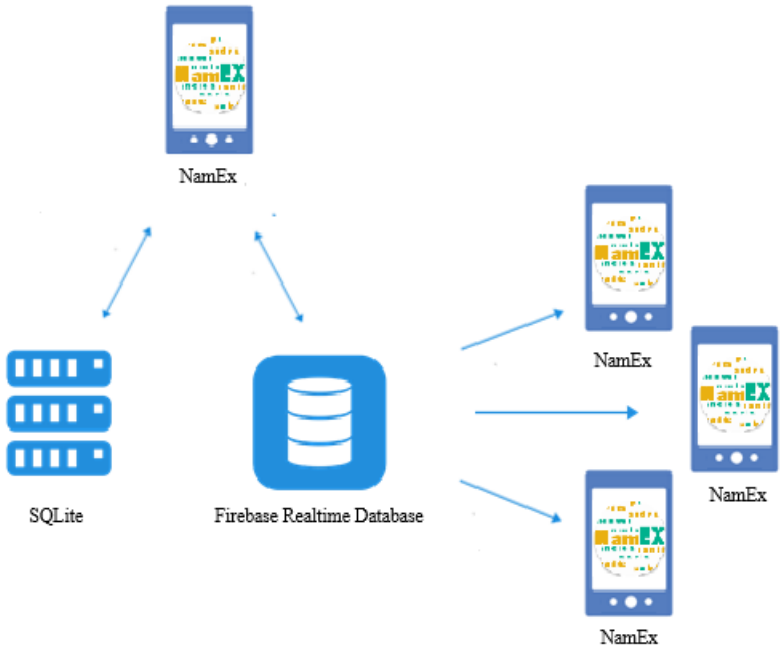
Este apartado describe el proceso seguido durante la construcción de la aplicación Android, permitiendo observar los cambios y la evolución de la herramienta, desde su etapa inicial hasta el final de su desarrollo. Es importante destacar que en este apartado ya se ha realizado tanto la instalación del software con el que se va a trabajar, Android Studio, (Anexo VII), como la configuración de la base de datos Firebase Realtime Database de Google (Anexo VI).

En los siguientes apartados, se va a explicar el funcionamiento de la aplicación en la versión final, indicando en cada uno de ellos los pasos que se han realizado para lograr dicho resultado, comentándose inicialmente, los menús con los que cuenta *NamEx*.

4.1 Arquitectura de la aplicación.

En este primer apartado se va a comentar la arquitectura que tiene la aplicación, donde va a tener cabida el diseño, que se va a comentar más adelante, el cual se va a integrar con las bases de datos que fueron seleccionadas en las primeras etapas del proyecto.

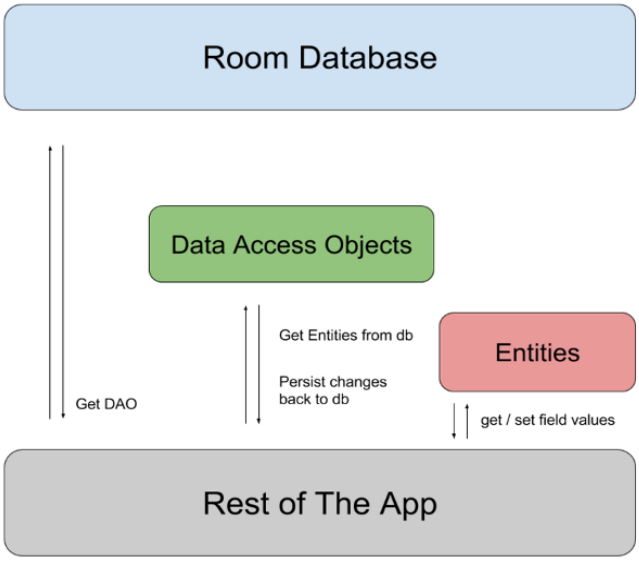
La siguiente imagen muestra la arquitectura resultante de la aplicación.



17. Arquitectura de la aplicación. Fuente: propia.

Como puede observarse, *NamEx* cuenta con una base de datos local, *SQLite*, que permite mantener los datos de los alumnos en caché, entre otros. Asimismo, se comunica con otra base de datos almacenada en la nube, *Firebase Realtime Database*, donde se almacena información que va a ser consultada por los diversos usuarios que utilicen la aplicación, además de los índices que contabilizan las veces que cada usuario ha jugado a cada modo que se encuentra disponible en la misma, que son consultados al iniciar la aplicación, los cuales también se encuentran almacenados en la nube, para que estos no dependan directamente del dispositivo donde se ejecuta *NamEx*.

Para implementar *SQLite* se ha utilizado un componente de arquitectura *Android* conocido como *Room*, que proporciona una capa de abstracción sobre la base de datos y permite acceder a ella sin problema, de la siguiente manera:

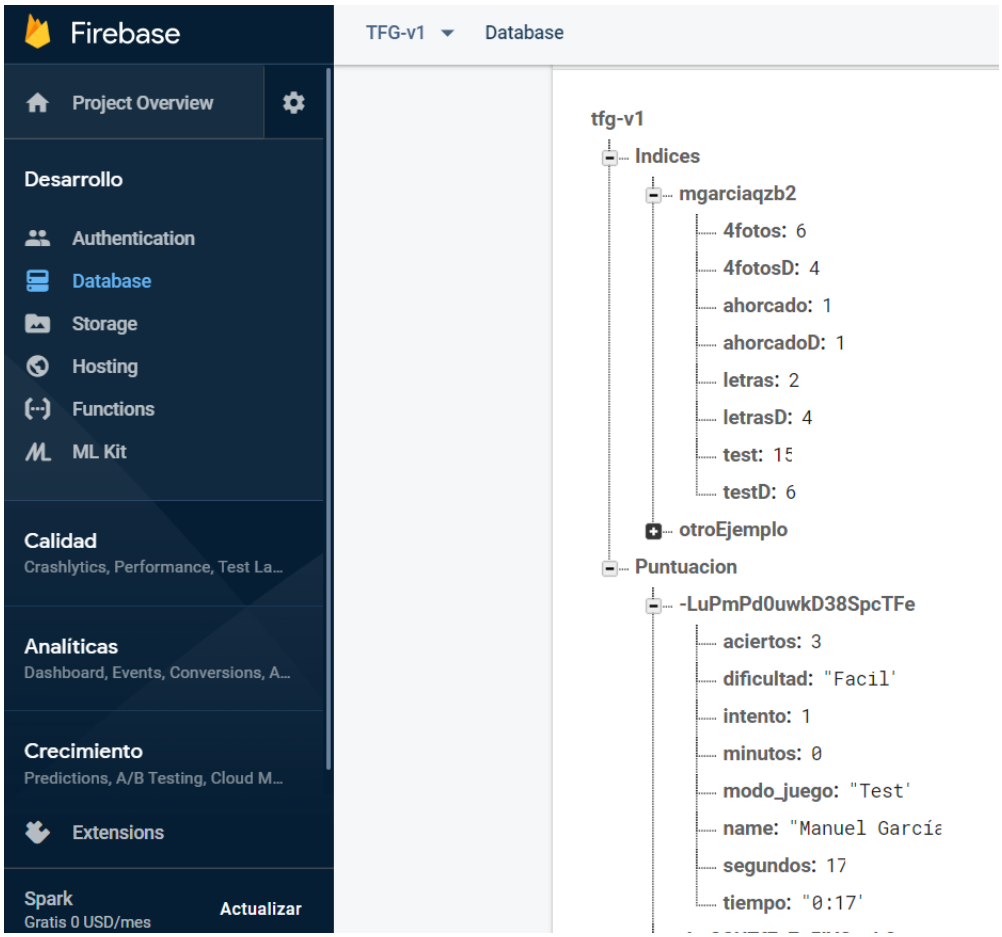


18. Diagrama de arquitectura Room. *Fuente: Android Developers.*

En el desarrollo final se ha necesitado añadir dos tablas a esta base de datos local, una para almacenar los alumnos y otra para guardar las asignaturas que imparte el docente y a la vez usuario, de esta aplicación.

Para finalizar, el uso de *Firebase* requiere primero la creación un proyecto en dicha web. Una vez creado, necesitas enlazarlo con la aplicación que se está desarrollando, lo cual se realiza de una manera sencilla debido a las facilidades que te ofrece el *framework Android Studio*¸ que te muestra los pasos necesarios para completar esta operación, como se puede observar en el anexo VIII.

La estructura final de esta base de datos es la siguiente:



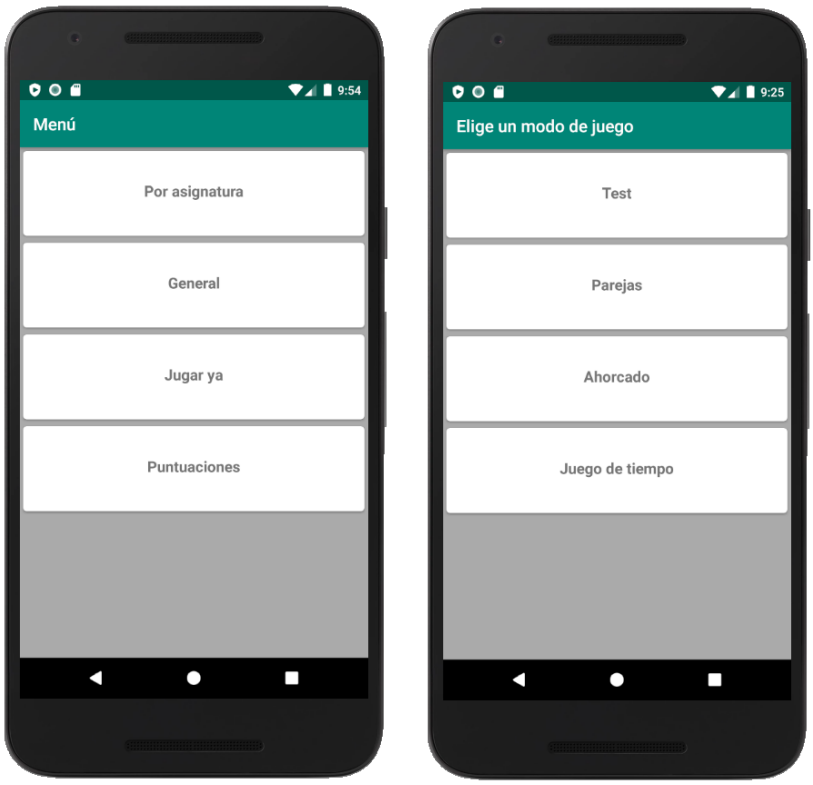
19. Estructura de los datos en Firebase. *Fuente: propia.*

4.2 Estructura de la interfaz desarrollada.

La siguiente etapa del desarrollo fue diseñar la estructura que se quería que tuviera la aplicación. Como en todas las construcciones este es el paso inicial y la idea principal que sigue el diseño es que la aplicación ha de ser sólida y usable, con el objetivo de que el usuario final quiera utilizar la herramienta que se le presenta, para lo cual, que esta sea intuitiva o se parezca a alguna otra a la que ya este acostumbrado es muy importante.

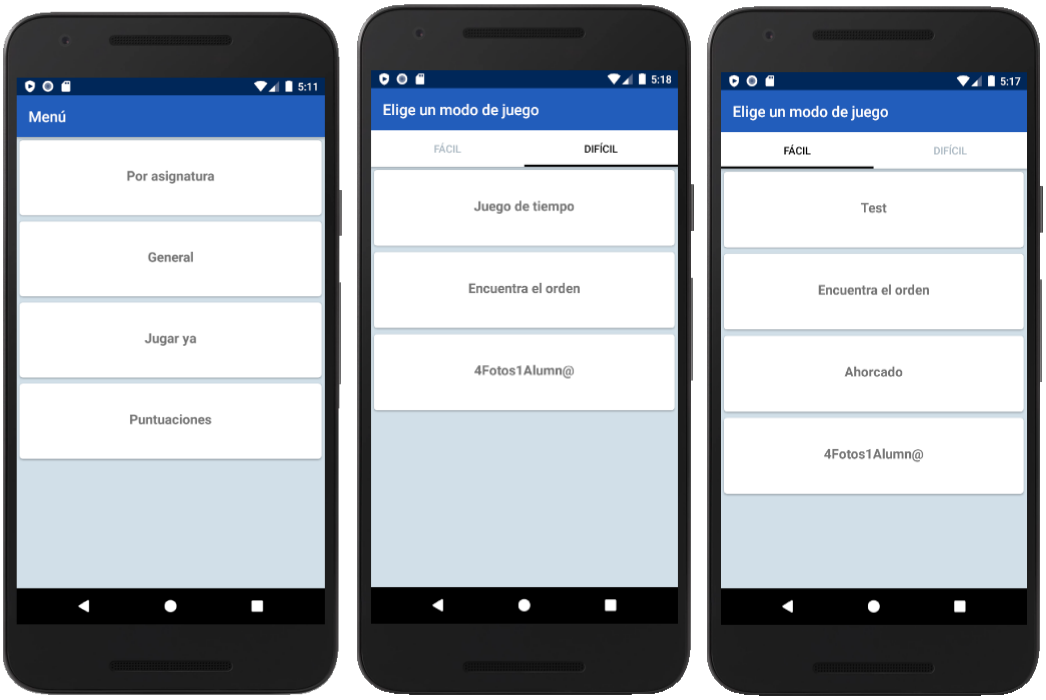
Se decide una distribución en la cual el actor principal fuesen las listas de elementos, asimilando su estructura a la aplicación de telefonía móvil *WhatsApp*, conocida por prácticamente todo el mundo. Que cuenta con más de 5.000 millones de descargas solo en el *Play Store*, lo que hace que el usuario se sienta familiarizado con la herramienta, antes incluso de utilizarla por primera vez. A estas listas se añade cada

una de las categorías y modos que se pensaron desarrollar en un principio, y que como veremos a lo largo de estos párrafos fueron modificados hasta alcanzar el estado final, presentes en la herramienta expuesta en esta investigación.



20. Menús iniciales de NamEx. *Fuente: propia.*

Estos menús iniciales, a lo largo del desarrollo fueron modificándose hasta obtener el resultado que se muestra a continuación.



21. Menús finales de NamEx. *Fuente: propia.*

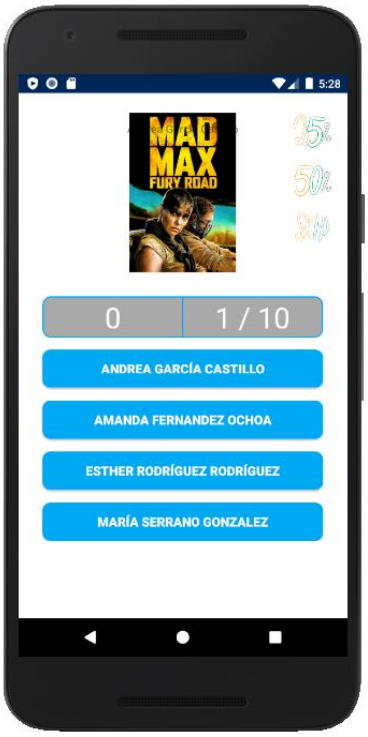
Cada uno de estas funcionalidades se van a comentar en los párrafos inferiores, pudiendo observar las características e interfaces de cada uno de ellos.

4.3 Modos de juego fáciles.

Una vez estructurados los menús con los que cuenta la aplicación, se inicia el desarrollo de cada uno de estos, los cuales van a ser expuestos a continuación.

4.3.1 Test.

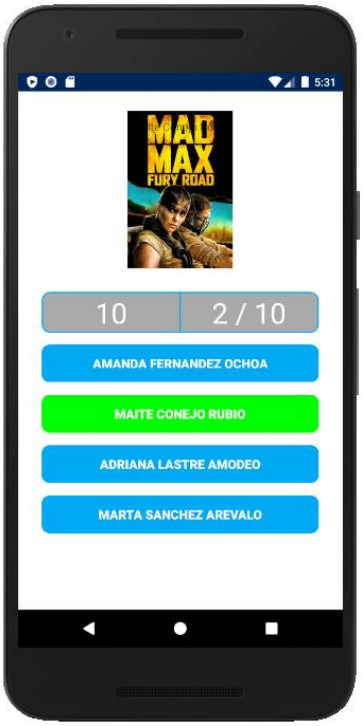
El primer juego desarrollado fue un test que consta de una pregunta y cuatro posibles respuestas, con el funcionamiento que se puede observar en la versión final de la aplicación.



22. Interfaz final modo *Test. Fuente: propia.*

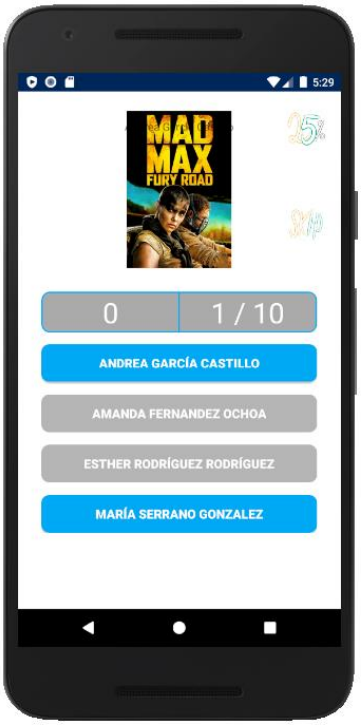
Como se puede ver, la fotografía se corresponde con el alumno por el que se pregunta. En este caso, como no se cuenta con los datos del Campus Virtual, se ha cargado una imagen de Internet, del mismo modo que se haría con la imagen de los estudiantes que van a ser preguntados. Asimismo, cuenta con tres comodines, cada uno realiza la función correspondiente a la imagen que lo representa. En la parte inferior, se encuentra el marcador y los botones con las cuatro respuestas posibles, entre las que aparece el nombre real del alumno en cuestión.

Una vez que se selecciona el nombre de un alumno, se comprueba si es el correcto, en ese caso sucede lo siguiente: el botón seleccionado cambia su color a verde como indicativo de que el usuario ha acertado en su elección. Al mismo tiempo, la pantalla se congela un periodo breve de tiempo para que el docente pueda observar que estaba en lo cierto, bloqueándose los botones para que no puedan ser pulsados y escondiéndose los comodines con el mismo fin, aumentándose la puntuación en el marcador, junto con el número de la pregunta en la que se encuentra el usuario, como se puede observar a continuación.



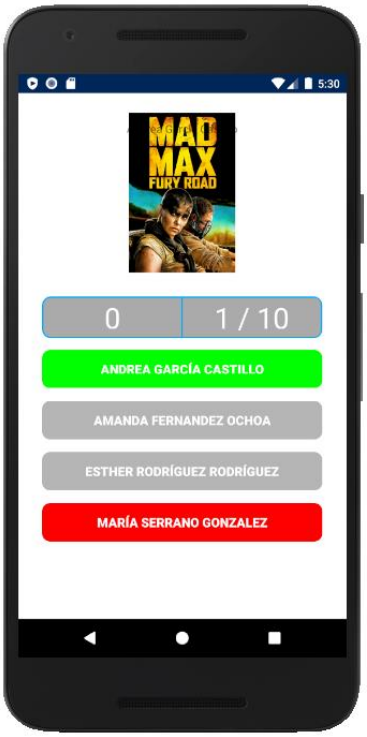
23. Interfaz final modo Test (2) Acierto. *Fuente: propia.*

En la siguiente imagen se puede observar el comportamiento de uno de los comodines, concretamente el del 50%. Como su nombre hace indicar, elimina dos respuestas incorrectas, dejando al usuario tan solo dos posibles soluciones.



24. Interfaz final modo Test (3) Comodín 50%. *Fuente: propia.*

Por último y para que se vea el comportamiento de las respuestas fallidas, se selecciona a un alumno de manera incorrecta, coloreándose de rojo el botón que contiene dicha respuesta, y al igual que pasaba en el caso contrario, tiñéndose de verde el que contiene la respuesta correcta y congelándose la pantalla para que el usuario sea consciente de su error y pueda observar la solución no errónea.



25. Interfaz final modo Test (4) Fallo. *Fuente: propia.*

Por último, una vez se ha contestado a todas las preguntas, se muestra el número de respuestas correctas y erróneas, la puntuación obtenida y el tiempo que se ha tardado en completarlo, variable que se añade posteriormente que puede resultar útil para posibles estudios futuros de los datos obtenidos, que son almacenados en *Firebase* para que puedan ser instanciados por todos los usuarios.

Para llegar a ese aspecto se han realizado diferentes pasos, que van a ser tratados a continuación; al igual que para obtener la funcionalidad descrita.

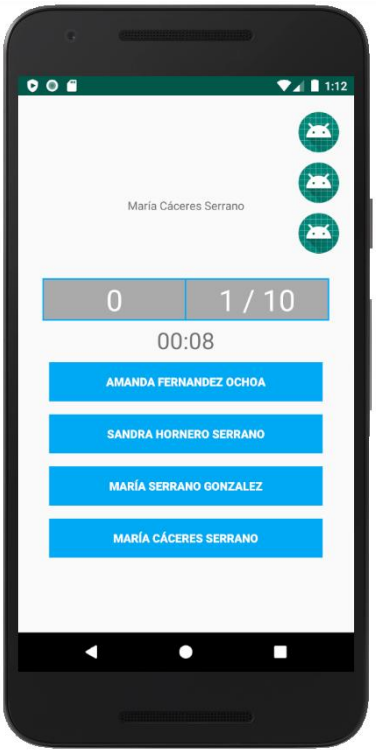
Inicialmente, se construye la interfaz gráfica, que no realiza ninguna actividad, de la siguiente manera.

26. Interfaz inicial modo Test. *Fuente: propia.*

Como se ha comentado en apartado anteriores, la funcionalidad principal de *NamEx* es la facilitación del aprendizaje de los nombres de los alumnos que tienen en sus asignaturas los docentes. Con ese fin se presenta esta modalidad, donde se puede observar el nombre del alumno preguntado en la parte superior, debiendo ser cambiado en alguna de las sucesivas versiones, una vez se obtengan los datos del Campus Virtual y pudiéndose acceder de esa manera a la fotografía de cada uno de los alumnos. Por el momento, se muestra el nombre del alumno preguntado para conocer la respuesta correcta y probar de esa manera que la funcionalidad de este, cuando se desarrolle, se realice correctamente.

En la parte inferior, se encuentra el marcador, que cuenta en la parte izquierda con la puntuación y en la parte derecha con el número de preguntas totales, además de las que ya se han realizado. Asimismo, existe una barra de tiempo que se va rellenando, debiéndose responder la pregunta antes de que esta se complete y se cambie de pregunta. Por último, se encuentran los cuatro botones, uno con cada nombre de un alumno, y que van a ser pulsados comprobándose en ese mismo instante, si la respuesta seleccionada es correcta o no.

La siguiente evolución de la interfaz, se desarrolla al unísono con la adición de la funcionalidad a cada uno de los componentes de esta, cuyos módulos más importantes van a ser comentados al finalizar este apartado, lográndose el siguiente resultado.

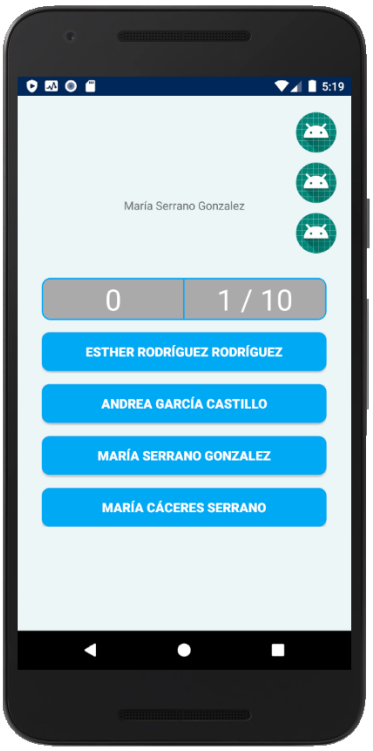


27. Interfaz inicial modo Test (2). *Fuente: propia.*

Como se puede observar, se cambia la tonalidad del marcador y de los botones, siendo estos más estéticos que sus predecesores, además de añadirse tres comodines que van a servir como una pequeña ayuda para facilitar al usuario a adivinar el nombre del alumno preguntado, del que solo conoce su imagen, aunque aún se siga observando el nombre interpelado. Asimismo, se cambia la barra de tiempo, que además de no dar muchas pistas al usuario de cuánto tiempo le queda para responder a la pregunta, tenía un comportamiento poco consistente, lo que podía llevar al usuario a que no respondiera a una pregunta, cambiando a otra sin que la barra hubiera llegado al final, como se comenta en el apartado de problemas encontrados durante el desarrollo del proyecto.

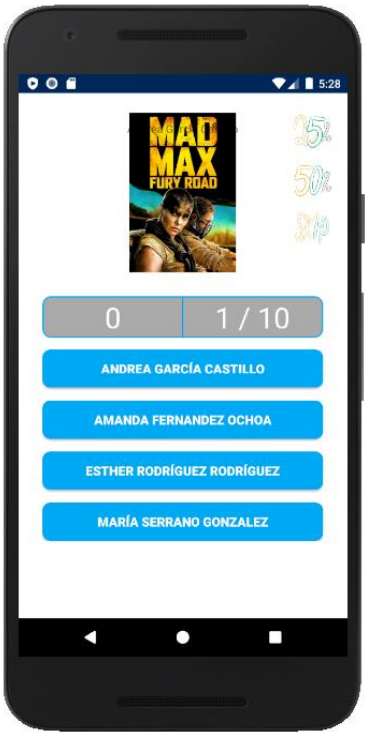
El siguiente cambio implementado no es demasiado significativo, se ha redondeado los bordes de los botones y el marcado, alineándose la posición de estos para que el inicio y fin de todos sea el mismo. Sin embargo, merece la pena ser comentado, debido a que el reloj que contiene la cuenta atrás se elimina, dando lugar

a dos categorías de juegos: los fáciles, que ayudan a aprender los nombres de los alumnos y los difíciles, que ponen más a prueba la adquisición de dichos conocimientos, reduciendo las ayudas y el tiempo o intentos de resolución del problema.



28. Interfaz inicial modo Test (3). *Fuente: propia.*

Por último, se obtiene la imagen que se ha mostrado al inicio de esta sección, siendo esta la versión definitiva desarrollada del modo de juego test.



29. Interfaz final modo Test (2). *Fuente: propia.*

4.3.1.1 Implementación y funciones importantes.

Una de las funciones más importantes es el algoritmo desarrollado para generar las preguntas. Este está contenido dentro de la clase *Pregunta*, cuyo código se encuentra en el anexo I y que se trata a continuación.

La clase *Pregunta.java* contiene una estructura con el mismo nombre, que cuenta con la pregunta, cada una de las respuestas posibles y con la respuesta que es correcta, facilitando su comprobación en el test.

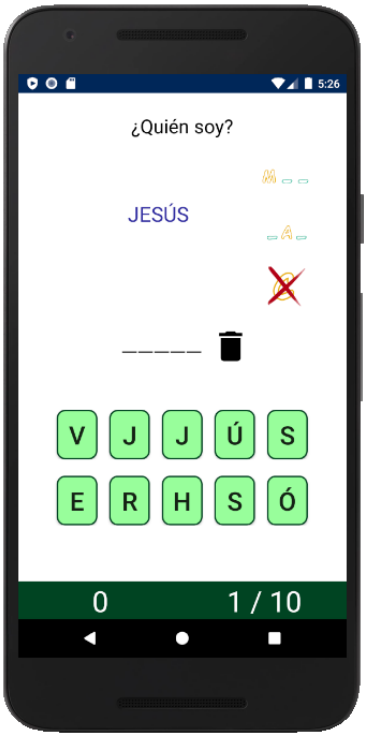
Las preguntas del test se generan partiendo de la lista total de alumnos que tiene el profesor, seleccionando uno al azar. Una vez se obtiene, se comprueba si ya ha sido preguntado en alguna de las cuestiones anteriores. De no ser así, se anota para que no vuelva a utilizarse como pregunta en esa ejecución y se guarda en la lista de respuestas, anotando su imagen como pregunta y su nombre completo como respuesta correcta. Una vez se ha realizado esto último, se repite el proceso anterior tres veces más, para seleccionar las respuestas, que deben ser alumnos con el mismo sexo que el preguntado, porque si no se podría descartar respuestas, lo cual no tendría mucho

sentido, hasta que se obtienen cuatro respuestas en total. Por último, se mezclan las respuestas para cambiar su orden y cada una de ellas se almacena en la pregunta, repitiendo el proceso once veces debido al comodín que nos permite saltar una pregunta, por el que debemos añadir una cuestión extra por si este se utilizase, ya que por su diseño la pregunta se anula y se sustituye por otra, mostrando la respuesta correcta de la que se va a eliminar.

Asimismo, como funciones importantes, se encuentra los comodines, que serán comentados más adelante y el método *onClick*, incluido en la actividad del test, que es el encargado de comprobar qué botón se ha pulsado, ya sea una respuesta o un comodín, comprobando en el primer caso si es correcto o no y coloreando, como se ha mostrado anteriormente, los botones de verde o rojo según si la respuesta seleccionada es correcta o errónea. Sin olvidarse de los comodines, los cuales son una adición que no estaba en el diseño inicial y que se incluyen para facilitar la selección de la respuesta acertada.

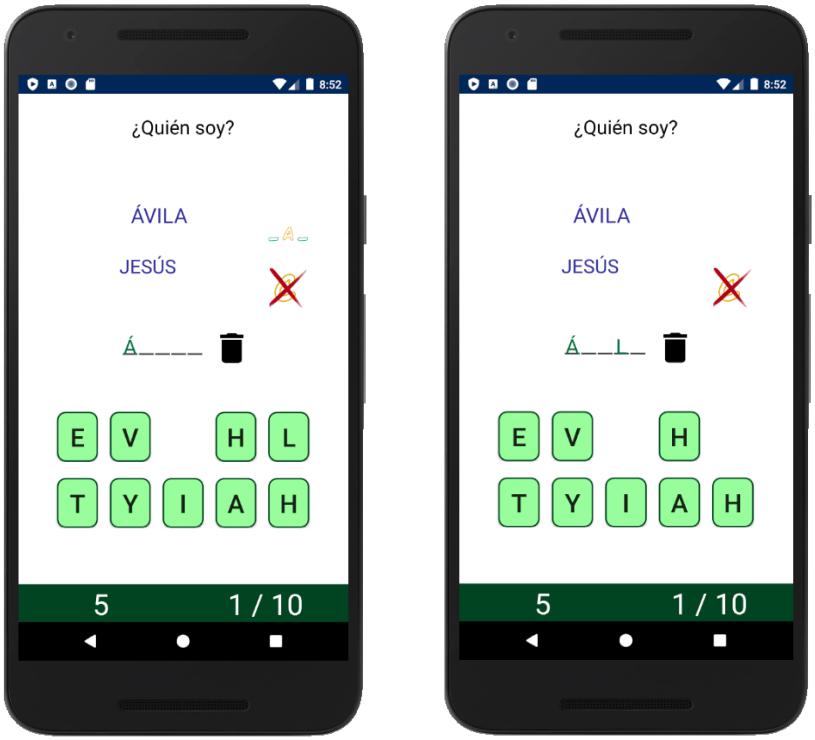
4.3.2 Encuentra el orden.

El siguiente modo de juego que se desarrolla es el denominado como *Encuentra el orden*. Consiste en la formación de una palabra a partir de las letras que están presentes debajo del hueco para adivinar la misma, de manera que el usuario debe rellenar cada uno de ellos pulsando los botones que contienen los caracteres de este, además de otros que no forman parte del nombre para que no sea tan sencillo acertarlo. Una vez se completa, ocurre lo mismo con el apellido, como se puede observar en las sucesivas imágenes.



30. Interfaz final modo Encuentra el orden. *Fuente: propia.*

En este caso, se muestra el nombre en vez de incluir una fotografía, lo cual debe sustituirse cuando la aplicación cuente con datos reales. El nombre que se busca es Jesús, que está oculto entre diez letras de los botones de la parte inferior. Cuando la palabra se ha rellenado, si es errónea, las letras vuelven a su sitio inicial y el usuario debe probar con otra combinación, pudiendo utilizar el botón con forma de papelera para eliminar la última que ha introducido, si cree que es errónea. Si la palabra formada es correcta, se realiza lo mismo con el primer apellido, como se puede observar a continuación.



31. Interfaz final modo Encuentra el orden (2) Uso de comodines. *Fuente: propia.*

En este caso, se utilizan comodines para facilitar la solución al problema. En la primera fotografía, se ha pulsado el que desvela la primera letra del nombre, eliminándose de los botones para que no se pueda volver a utilizar. Asimismo, en la imagen de la derecha, se ha utilizado el que proporciona una letra aleatoria entre la segunda y la última, con el mismo comportamiento que el anterior, mostrándose la letra en el nombre y ocultando el botón que la contiene.

Por último, se puede observar que el marcador indica cinco puntos, pero al mismo tiempo se mantiene en la primera pregunta. Esto está más orientado al modo de juego difícil, que como se verá, tiene intentos limitados para adivinar el nombre o el apellido, como se explicará con posterioridad.

Para lograr la obtención de dicha interfaz se ha realizado un desarrollo incremental, al igual que sucede con el código que da vida a este juego. A continuación, se va a comentar la evolución gráfica del mismo.

Inicialmente se diseña una interfaz sencilla con una única palabra y las mismas letras, estáticas, en los botones para conformarla, donde se comprueba el correcto

funcionamiento del código desarrollado. Como se observa, carece de comodines, de la posibilidad de borrar la última letra añadida y de información alguna sobre la puntuación y las preguntas que se han respondido, debido a que en este momento se trabaja con un ejemplo estático.

32. Interfaz inicial modo Encuentra el orden. *Fuente: propia.*

El próximo paso, cuando se logra el correcto funcionamiento de esta pequeña simulación, es agregarle una funcionalidad mayor, para lo cual, se agrega el nombre y primer apellido de un alumno inventado. Se realizan cambios en la interfaz, se añade la barra de información de los puntos y las preguntas que se han respondido, mostrándose, una vez descubierto, el nombre debajo de la foto del alumno, añadiéndose el apellido junto con sus letras correspondientes, que también debe ser adivinado por el usuario, como podemos observar en las siguientes imágenes.

33. Interfaz inicial modo Encuentra el orden (2). *Fuente: propia.*

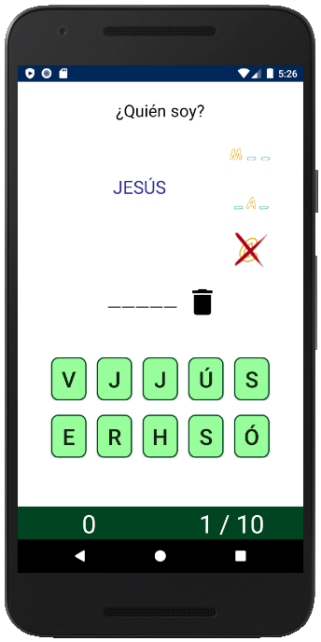
Al igual que en el test, una vez se ha completado el nombre y el apellido del alumno, se produce una pequeña pausa para que el usuario pueda asociar el nombre completo con la cara del alumno o alumna en cuestión.

Por último, se agregan los comodines. Al igual que en el modo anterior, se ha querido introducir una pequeña ayuda que facilite el aprendizaje y acierto de las preguntas, y que va a ser explicado en el siguiente apartado.

Asimismo, se añade un botón que posibilita borrar las letras que se han añadido, teniéndose que cambiar la implementación que daba lugar a la recolección y muestra de las letras pulsadas, pensándose en que con la adición de comodines iban a haber letras que no debían borrarse, teniéndose que guardar y mostrar por separado.

34. Interfaz inicial modo Encuentra el orden (3). *Fuente: propia.*

Por último, se logra la versión que está incluida en la aplicación, donde predomina el color verde, tanto en los botones como en el marcador, y donde se incluye el diseño final de los comodines, ya que como se puede observar en las imágenes anteriores, los comodines no permiten identificar la funcionalidad que realiza cada uno de ello, facilitándose la comprensión de la tarea que realizan y donde predominan los colores principales del logo de la aplicación, dando una mejor forma a todo el conjunto.



35. Interfaz final modo Encuentra el orden (2). *Fuente: propia.*

4.3.2.1 Implementación y funciones importantes.

Una de las funciones a destacar es la que realiza el método *OnClick*, que como ya se ha comentado anteriormente es el encargado de recoger las pulsaciones de los botones que realiza el usuario.

Como se puede observar en el código que se incluye en el anexo II, cuando se pulsa un botón, se recoge, se almacena y se comprueba si el usuario ha seleccionado el botón de borrar, cerciorándose de que existe alguna letra que ya ha sido pulsada que es la que se guarda en la lista denominada *botones_pulsados*. Si así fuere, se hace visible el último botón que contuviera una letra que se ha pulsado, eliminando la letra del nombre que se está formando en la pantalla y reduciéndose el índice, que indica el número de letras que contiene el espacio para ellas, inicialmente vacío, en la pantalla. De no ser así, se añade la letra al nombre en la interfaz, haciéndose invisible el botón seleccionado y aumentándose el tamaño de la palabra "solución". Una vez se ha realizado esta operación, se comprueba si la siguiente posición de la palabra no está vacía, por si el comodín que coloca una letra aleatoria hubiese añadido un carácter en esa posición, aumentándose el índice en una unidad, constatándose en ese momento si se ha alcanzado la longitud total de la palabra. Si nos encontramos en este caso, se verifica si el nombre formado es acertado. De ser así, se prepara el apellido para que se adivine. Si la respuesta no fuera correcta, la palabra "solución" vuelve a quedar vacía, mostrándose de nuevo todos los botones que contienen las letras para que el usuario vuelva a intentar adivinar la solución correcta.

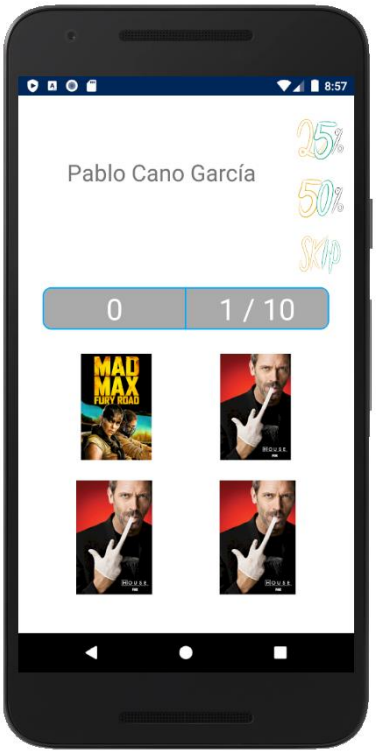
Por último, se encuentran los comodines, de los que se va a describir uno de ellos, concretamente el que se encarga de ocultar un botón que contiene una letra que no forma parte de la palabra buscada, reduciendo la muestra, facilitando de esta manera que el usuario pueda averiguar el término a descubrir. Esto siempre va a suceder debido a que el tamaño máximo permitido del nombre y el apellido es nueve. El código que realiza esta funcionalidad es el siguiente:

```
public void quitarLetra_Comodin(View view){
    nComodin.add(nComodin.size(),1);
    Random RNG = new Random();
    boolean bandera = false;
    int i;
    char letra=' ';
```

```
        while (!bandera) {
            i = RNG.nextInt(alumnoLetters.size());
            letra = alumnoLetters.get(i);
            if(estaEnElNombre(letra)){
                bandera=true;
            }
        }
        añadirLetraLista(letra);
        ocultarBoton(letra);
        comodinQuitaLetra.setEnabled(false);
    }
```

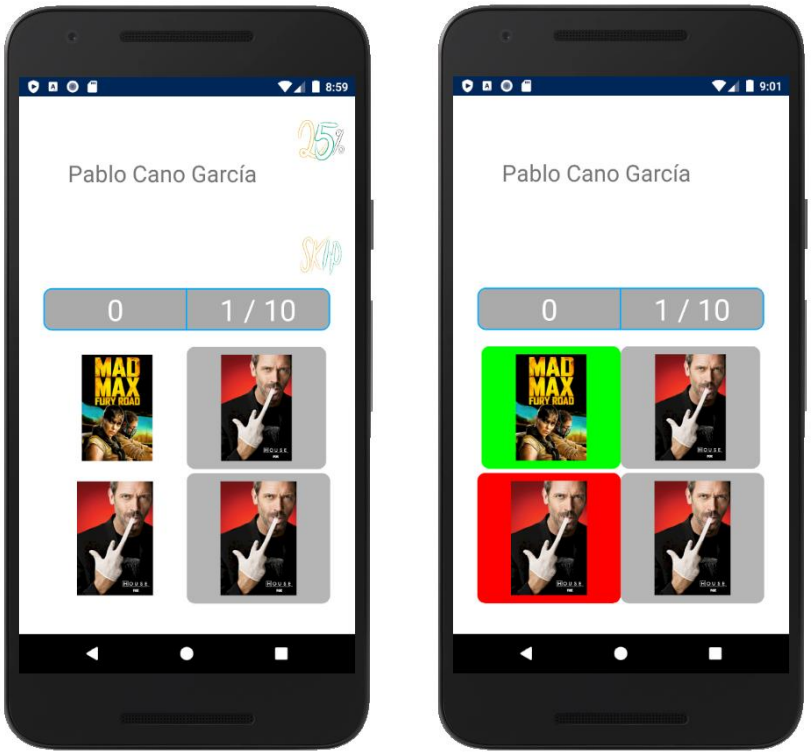
4.3.3 4 Fotos 1 Alumn@.

4 Fotos 1 Alumn@ es el siguiente modo desarrollado, el cual se diseña basándose en el juego conocido como *4 Fotos 1 Palabra*, que había servido de inspiración para la realización del modo de juego anterior, además de tener en cuenta la primera modalidad expuesta. Consiste en la visualización del nombre de un alumno en la parte superior de la pantalla, debiéndose elegir la fotografía que corresponde a dicha persona, entre cuatros estudiantes posibles situados en la parte inferior de esta, como se puede observar a continuación.



36. Interfaz final modo 4 Fotos 1 Palabra. *Fuente: propia.*

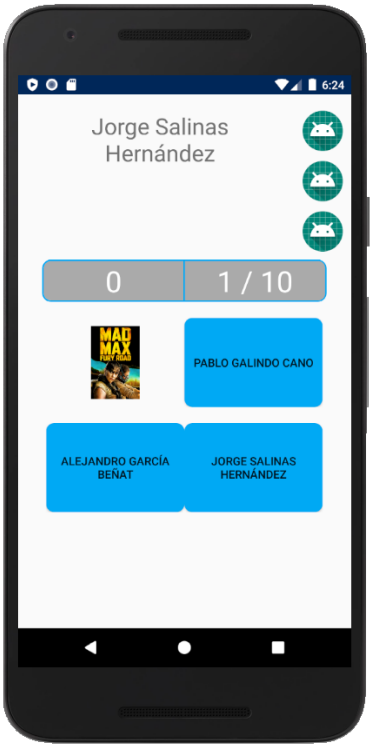
El funcionamiento es sencillo. La persona que está jugando puede seleccionar una de las respuestas o un comodín, que le ayude a seleccionar la fotografía correcta. Las ayudas ofrecidas son las mismas que aparecen en el modo de juego *Test*, con lo que el usuario debe sentirse familiarizado con estas. En la primera imagen, se va a utilizar el comodín del 50% que elimina dos opciones incorrectas, eligiéndose la opción errónea de las restantes para poder apreciar el comportamiento, muy similar al de modos de juego explicados anteriormente, como se puede observar seguidamente.



37. Interfaz final modo 4 Fotos 1 Palabra (2). Comodín 50%. *Fuente: propia.*

Para obtener estos resultados, se han realizado varias implementaciones y cambios en el diseño de la interfaz gráfica, con el fin de obtener un resultado que fuese lo mejor posible, con el comportamiento esperado y usable.

Para comenzar, se parte de un diseño muy similar al que tiene en el modo de juego *Test*, debido a que este juego es su inverso en cuanto a la interfaz se refiere, aunque el comportamiento se mantiene parecido. La primera implementación es la siguiente.



38. Interfaz inicial modo 4 Fotos 1 Palabra. *Fuente: propia*.

Finalmente, una vez se logra el funcionamiento esperado tanto de los botones como de los comodines presentes en la interfaz, se realiza la carga de imágenes, las cuales vienen dadas por una dirección web, del mismo modo que se realizaría si se tuvieran los datos reales de los alumnos obtenidos desde *Moodle*.

En esta versión de la aplicación, debido a que las fotografías que se muestran no se corresponden con ningún alumno, se ha preparado una simulación en la que la respuesta correcta siempre es la primera imagen que aparece, colocada en el botón superior izquierdo. Esto debe modificarse una vez se trabaje con datos reales. Con ese fin, se ha indicado en el código de la aplicación los pequeños cambios que hay que realizar para que el juego pase de ser estático a ser dinámico, obteniéndose la versión final de la aplicación mostrada anteriormente.

4.3.3.1 Implementación y funciones importantes.

Una de las funciones más importantes, además del método *OnClick* que es muy similar al del *Test* que ya ha sido explicado, es la librería que permite la carga de

imágenes a partir de una dirección de Internet, conocida como *Glide*. Su elección se realiza después de consultar con el personal encargado el Campus Virtual para comentar cómo están estructurados los datos con el objetivo de adelantar las posibles modificaciones que se debían incluir una vez prestada la conexión a sus servicios. Debido, a que los datos con los que se ha trabajado hasta el momento son locales, estos cuentan con una estructura ligeramente diferente a la que van a tener los datos finales, una vez se realice la conexión con el Campus Virtual, obteniéndose estos mediante dicho enlace se debían adaptar ciertas estructuras al nuevo formato de los datos. Aunque finalmente esta conexión no se ha realizado, se ha incluido el trabajo de preparación previa a la recepción de los datos reales, añadiéndose esta tarea como línea de futuro del proyecto.

Otra tarea a destacar es el funcionamiento de los comodines que disminuyen la posible frustración generada en algunos casos al no conocer la respuesta acertada. Estos son los mismos que aparecen en el *Test* y que al no haber sido explicados anteriormente, se van a comentar a continuación.

El primero es un comodín del 50%, que elimina dos opciones que no son correctas dejando que el usuario seleccione una de las restantes, con el siguiente código:

```
public void comodin50(View view){
    if(!comodin) { //solo se permite el uso de un comodín por
                   //pregunta
        contador50--;
        int i, contador = 0;
        boolean bandera = true;
        Random RNG = new Random();

        while (contador < 2) {
            i = RNG.nextInt(4);
            Log.d("i","i:"+i);
            bandera = compruebaSiEsLaRespuesta(i);
            if (!bandera) {
                contador++;
            }
        }

        comodin50.setEnabled(false);

        comodin = true;
    }
}
```

Si no se ha usado ningún otro comodín, se resta una vez al contador de veces que se ha usado este, que como máximo van a ser dos. Se obtiene un número aleatorio entre el cero y el tres, correspondiente a cada uno de los botones que aparecen en la

pantalla y se comprueba si este contiene la respuesta correcta, de ser así no se toma como válido. Una vez se encuentren dos botones que no contienen las respuestas, se desactivan y el valor de la *bandera* pasa a ser falso, para que no se pueda usar ningún otro comodín en esa pregunta.

El siguiente comodín es similar al anterior, sin embargo, en vez de eliminar dos opciones únicamente suprime una de las posibles respuestas. Se elimina la variable *contador* y se utiliza en su lugar la bandera, ya que se quiere que se ejecute una sola vez, siempre que no sea la respuesta correcta, en cuyo caso se obtiene otra posición aleatoria entre cero y tres, como se puede ver a continuación.

```
public void comodin25(View view){
    contador25--;
    if(!comodin) { //solo se permite el uso de un comodín por
                   //pregunta
        int i;
        boolean bandera = true;
        Random RNG = new Random();

        while (bandera) {
            i = RNG.nextInt(4);
            bandera = compruebaSiEsLaRespuesta(i);
        }
        comodin25.setEnabled(false);

        comodin = true;
    }
}
```

Por último, el comodín que resta, posibilita saltarse una pregunta, mostrándose la respuesta correcta de la misma forma que hemos visto en el apartado anterior, cuando el usuario elegía una respuesta que resultaba ser errónea. Con este objetivo, se aumenta el número de preguntas que genera el algoritmo a once, anulándose la pregunta al utilizar este comodín e introduciendo una nueva para que el usuario responda a diez preguntas antes de finalizar el test, como se observa en el siguiente código.

```
public void saltarPregunta(View view){
    if(!comodin){
      //desactivamos los botones hasta la siguiente pregunta
        activarDesactivarBotones(false);
        marcarRespuestaCorrectaEnVerde();
        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                //activamos los botones de nuevo
                activarDesactivarBotones(true);
                Common.preguntasList.remove(index);
                comodinSkip.setEnabled(false);
```

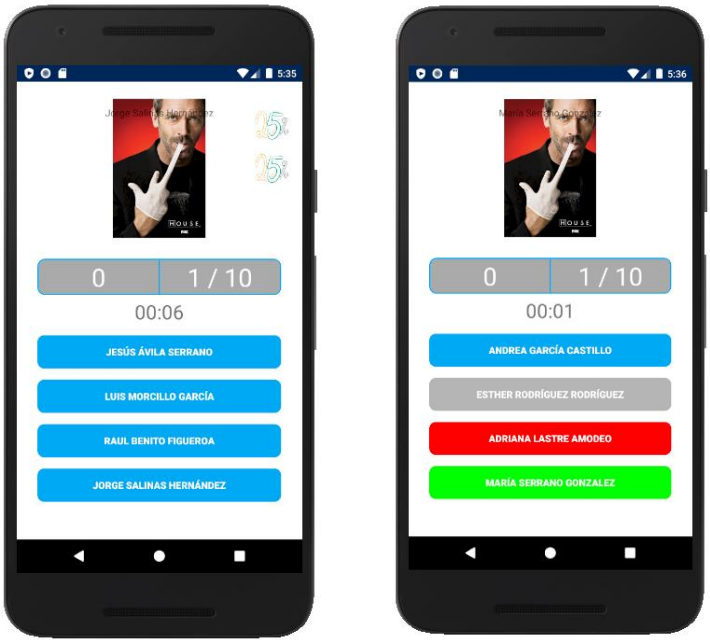
```
                    c3=false;
                    thisQuestion--;
                    comodin = false;
                    showQuestion(index);
                }
            },1500);
        }
    }
```

4.4 Modos de juego difíciles.

Una vez se han estructurado los menús con los que cuenta la aplicación, se inicia el desarrollo de cada uno de estos, los cuales van a ser expuestos a continuación.

4.4.1 Juego de tiempo.

Una vez se había terminado con la implementación del anterior modo de juego, del cual se había suprimido el reloj con la cuenta atrás, se decide aprovechar la funcionalidad y el diseño que se ha conseguido, volviendo a añadir este y reduciendo el número de comodines para que la dificultad se incremente, consiguiéndose de esta forma una nueva categoría de modos de juego. De esta manera, la interfaz definitiva es la siguiente:



39. Interfaz final modo Juego de tiempo. Uso del comodín 50% y respuesta fallida.
Fuente: propia.

Se puede observar una distribución muy similar a la anterior, teniéndose en cuenta que ahora existe una cuenta atrás, que cuando llega a cero muestra la respuesta correcta y pasa a la siguiente pregunta, y que solo existen dos comodines, que pueden ser activados una vez cada uno, del 25%, es decir, solo anula una posible respuesta, como se puede ver en la segunda fotografía. La imagen del doctor *House* se sustituirá en un futuro cuando se realice la conexión con *Moodle* y se obtengan los datos de dicha plataforma por la del alumno preguntado, teniendo que insertar la dirección *url* donde ahora se encuentra la foto que se puede apreciar en pantalla.

4.4.1.1 Implementación y funciones importantes.

Una de las funciones más importantes en este modo de juego por tiempo, además del algoritmo utilizado para generar las preguntas que ha sido comentado con anterioridad, es la función que controla la cuenta atrás, que va reduciéndose de segundo en segundo hasta llegar a cero y es en ese momento cuando al usuario se le muestra la respuesta correcta y se añade a la siguiente pregunta, además de la mencionada anteriormente del método *OnClick*, encargado de comprobar cada botón pulsado y ejecutar una función u otra distinta, de manera que el juego funcione correctamente, tal y como ha sido diseñado.

Por último, los comodines incluidos, en este caso, es el mismo que se repite en dos ocasiones, realizan una tarea importante, la cual elimina una respuesta que es errónea facilitando al usuario acertar el alumno o alumna por el que es preguntado. Estos comodines solo tienen un uso, una vez son utilizados desaparecen. El principal problema que se ha encontrado es que al introducir las pausas donde se muestra si la respuesta es correcta o es errónea estos siguen activos, de manera que pueden ser utilizados por el usuario en ese preciso instante. Para evitar ese comportamiento, ambos comodines se desactivan, almacenando su estado previo y volviendo a ser cargado una vez se ha avanzado a la siguiente pregunta.

4.4.2 Encuentra el orden.

El modo de juego *Encuentra el orden*, anteriormente descrito, se ha adaptado para aumentar la dificultad, proporcionando un reto mayor a los usuarios más avanzados. Este funciona de una manera muy similar a su antecesor, debiéndose formar una palabra a partir de las letras que están presentes en los botones de la parte inferior. En esta ocasión, y para que no sea tan sencillo acertarlos, se coloca un contador debajo de donde se sitúan los comodines, teniendo el usuario dos oportunidades para adivinar la solución correcta. Si no fuera así, se avanza al apellido o a la próxima pregunta, si ya se encontraba el usuario en esa situación. Asimismo, los comodines solo tienen un uso cada uno, como se puede observar en las sucesivas imágenes de la interfaz incluida en la versión final de la aplicación.

40. Interfaz final modo Encuentra el orden difícil. Uso del comodín de la primera letra.
Fuente: propia.

En la primera imagen, se observa la pantalla tal y como se ve al iniciar el juego y después de adivinar o no, cada palabra oculta. En la segunda, el marcador ha

aumentado, lo que indica que el nombre fue resuelto de manera correcta. Asimismo, se utiliza el comodín que dibuja la primera letra, habiéndose intentado responder una vez, de manera errónea. Esto permite observar que el carácter dibujado por el comodín no se elimina al reiniciarse la palabra y el número de intentos se reduce de forma correcta.

Para lograr la obtención de dicha interfaz, se ha partido de una de las últimas versiones del modo de juego sencillo, la más reciente en el momento que se decide el desarrollo de este juego, tal y como se ve en las siguientes imágenes.

41. Interfaz inicial modo Encuentra el orden difícil. Respuesta correcta. *Fuente: propia.*

Como se puede observar, la interfaz es muy parecida a la que se acaba de mostrar anteriormente, donde aparecen comodines sin una imagen adecuada, que no permite observar de manera adecuada si han sido utilizados o no. Por último, se modifica algunos aspectos gráficos, debido a que como se puede ver, la funcionalidad se realiza de forma correcta, mejorando la comprensión de lo que sucede mientras se juega, lográndose conseguir la interfaz tal y como se muestra en la imagen *4.24*.

4.4.2.1 Implementación y funciones importantes.

Una de las funciones a destacar es la que realiza el método *OnClick*, que no se va a comentar debido a que ya ha sido explicado en el apartado del modo de juego que recibe el mismo nombre pero que cuenta con una dificultad menor, no teniendo el código grandes cambios. Asimismo, el cálculo de los intentos restantes es importante pero demasiado sencillo como para comentarlo. Por último, se encuentran los comodines, de los cuales ya se ha explicado uno con anterioridad, los dos restantes son los siguientes: el primero de ellos, permite descubrir la primera letra que conforma la palabra oculta, ya sea esta del nombre o del apellido del alumno o alumna en cuestión, realizado con el código presente en el anexo III.

Primeramente, se obtiene la primera letra del nombre o apellido y se almacena en una lista desde la cuál será añadida de nuevo en la misma posición si el usuario completa los caracteres de la palabra buscada de forma incorrecta.

Posteriormente, se comprueba si el usuario ha empezado a escribir letras de la palabra buscada, si es así, se elimina la letra de los botones pulsados por el usuario para que no pueda ser borrada por este; si por el contrario la letra era incorrecta, se elimina como en el caso anterior de la lista de botones seleccionados y se vuelve a hacer visible, se oculta el botón con la letra correcta y se añade en la primera posición de la palabra, desactivando el comodín.

Si el usuario no ha empezado a escribir, se comprueba que la segunda posición esté vacía. Puede ser que el comodín que da una letra aleatoria haya sido usado y como veremos a continuación, este no aumenta el número de letras que contiene la palabra, por eso debemos hacerlo ahora si es que hubiese ocurrido esta situación, porque si no se sobrescribiría cuando el usuario pulsara cualquier botón.

Si por el contrario no nos encontramos en ninguno de los casos comentados, se aumenta la cantidad de letras en una unidad, añadiendo el carácter en la primera posición y se oculta un botón que contenga tal letra.

Inicialmente, se desarrolla el código sin las comprobaciones incluidas en los *if* y los *else*, pero, a la vista de los errores encontrados y que serán comentados más adelante en el apartado dedicado a estos, se incluyeron como solución a estos casos especiales.

El siguiente comodín tiene la funcionalidad de descubrir una letra en una posición aleatoria de la palabra, sin que esta pueda ser la primera, ya que esa ayuda está cubierta con el primer comodín explicado. Su código se encuentra en el anexo IV.

En primera instancia, se obtiene un número aleatorio entre la segunda letra y la última, que va a servir para seleccionar el carácter que se va a mostrar almacenando la posición para poder restaurarla en el caso de que el usuario falle al adivinar la palabra.

Posteriormente, se comprueba si existe alguna letra escrita por el usuario en esa posición, si es así, se elimina, volviéndose a mostrar el botón y se dibuja la letra correcta de color verde oscuro, para que el usuario conozca que la ejecución se ha realizado de forma correcta, ocultándose al mismo tiempo el botón que incluye dicho carácter, sin permitir que dicha letra pueda ser borrada por el usuario.

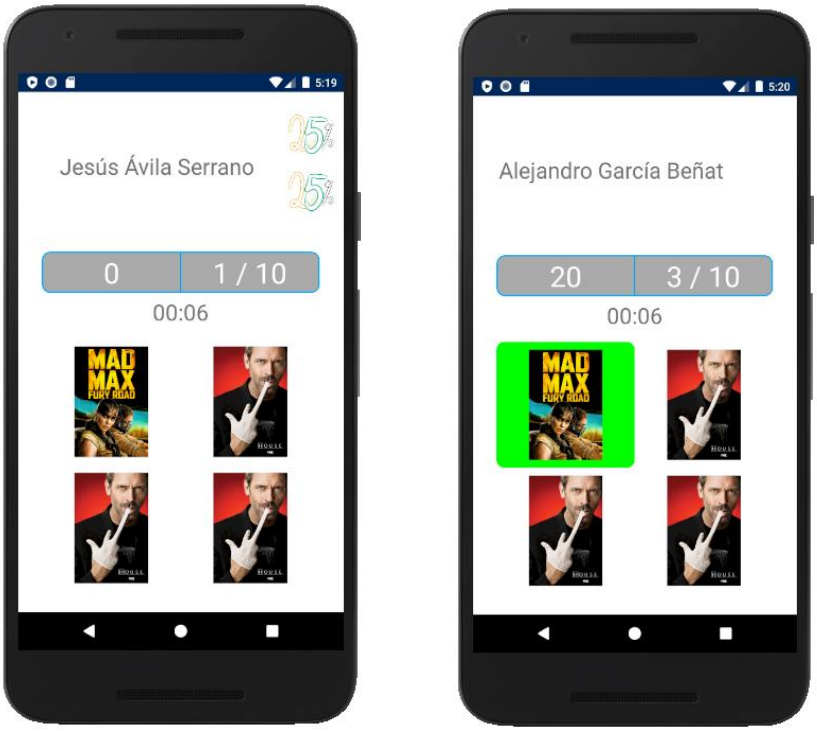
Tras haber realizado diferentes pruebas, se detectan dos casos especiales que hay que tratar. El primero de ellos, cuando la letra aleatoria se coloca en la segunda posición, debiéndose comprobar por cómo está realizada la lógica de este modo, si la primera casilla está ocupada, al mismo tiempo que se comprueba si la tercera está libre, de no estarlo no habría que aumentar la longitud de la palabra que está adivinando el usuario. Por último, si la ejecución de este comodín da como resultado la última letra de la palabra, se comprueba si el resto de la palabra ya ha sido introducida por el usuario. Si se cumple dicho caso, se comprueba que la palabra conformada es correcta o por el contrario no lo es.

4.4.3 4 Fotos 1 Alumn@.

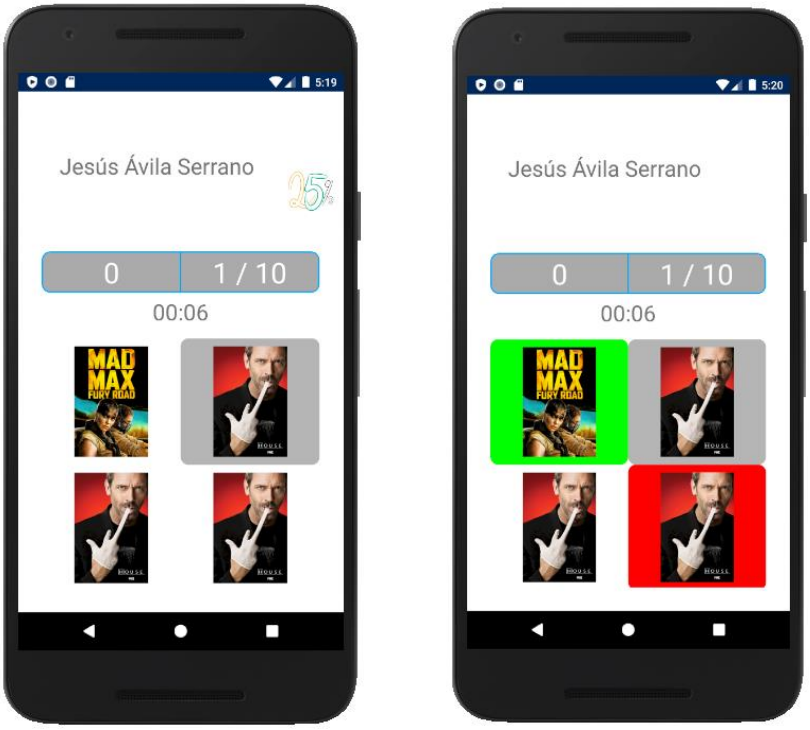
El siguiente modo implementado y último incluido en la aplicación es *4 Fotos 1 Alumn@*, el cual se desarrolla una vez se han completado todos los demás. El funcionamiento es similar al modo de juego que recibe el mismo nombre en la categoría de actividades fáciles y, al mismo tiempo, tiene una interfaz muy parecida al modo de juego denominado *Juego de tiempo*, lo cual hace indicar que el funcionamiento también va a ser similar.

4 Fotos 1 Alumn@ consiste en acertar la fotografía que se corresponde con el nombre del alumno o alumna que aparece en pantalla, entre cuatro posibles opciones. Sin embargo, y como se verá a continuación, al igual que ha pasado en los modos

anteriores, las imágenes de los alumnos son estáticas, debido a la falta de la conexión con el Campus Virtual para recibir los datos reales. Por el momento, se ha preparado una simulación estática donde la respuesta correcta siempre se encuentra en el botón superior izquierdo.



42. Interfaz final modo 4 Fotos 1 Palabra difícil. Respuesta correcta. *Fuente: propia.*



43. 4 Fotos 1 Palabra difícil. Uso del comodín 25%. Respuesta incorrecta. *Fuente: propia*.

Como se puede observar la interfaz, queda de la siguiente manera: con el nombre en la parte superior, dos comodines los mismos que aparecen en *Juego de tiempo* en la parte derecha de la pantalla, con el marcador y las cuatro opciones situados en la parte central e inferior respectivamente.

En la imagen situada en la esquina superior derecha, se puede observar lo que sucede al seleccionar la respuesta correcta. Se produce una pequeña pausa, donde se ocultan los comodines para que no puedan ser utilizados, despejando la interfaz. Al mismo tiempo, la imagen elegida se rodea de verde, indicando que es correcta.

Por último, en las imágenes inferiores, se observa cómo se utiliza un comodín que está disponible, eliminándose una posible respuesta errónea, para terminar con la elección de la respuesta equivocada, mostrando al usuario cuál es la verdadera opción acertada.

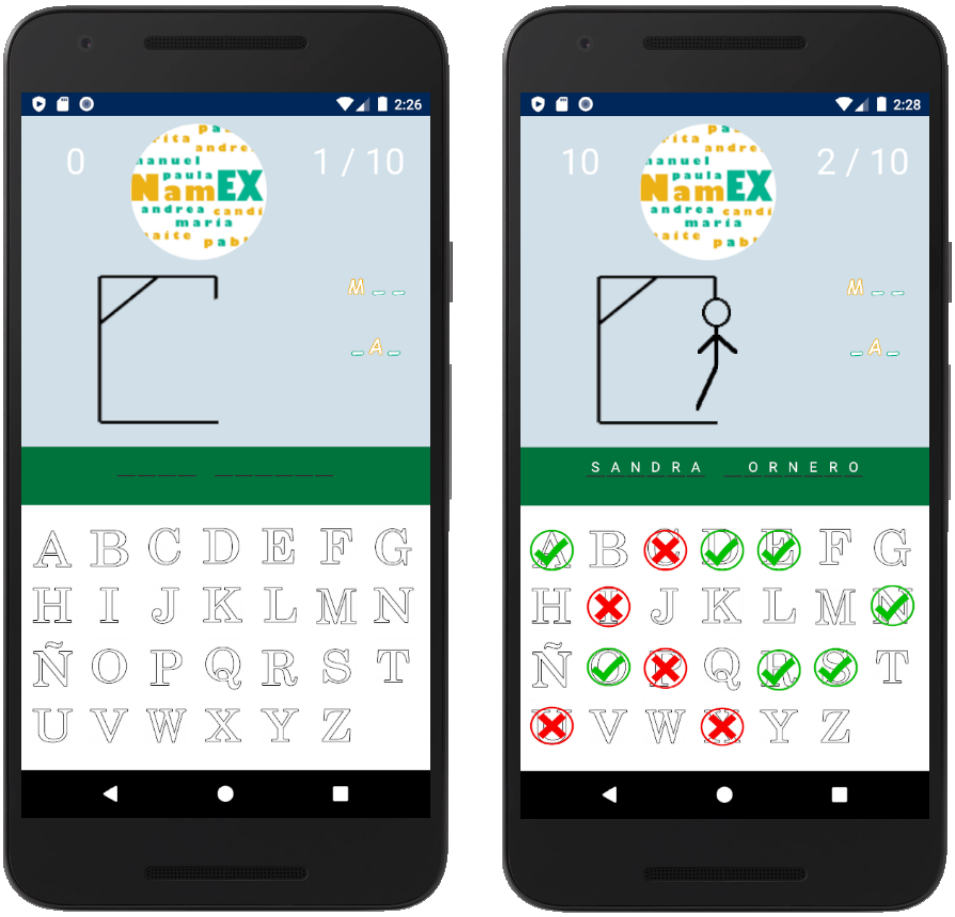
4.4.3.1 Implementación y funciones importantes.

Las funciones más importantes son el método *OnClick*, el algoritmo utilizado para generar las preguntas, la cuenta atrás y los comodines. Todas ellas han sido explicadas a lo largo de los apartados anteriores, por lo tanto, no van a volver a ser comentadas.

4.5 Ahorcado.

El modo de juego *Ahorcado*, como su nombre indica, posee el mismo comportamiento que el famoso juego con el mismo nombre. Sin embargo, esta vez la palabra que hay que descubrir es el nombre y primer apellido de un estudiante. Esta idea nace con la intención de crear una modalidad más casual, aplicando la gamificación como principio básico en su desarrollo.

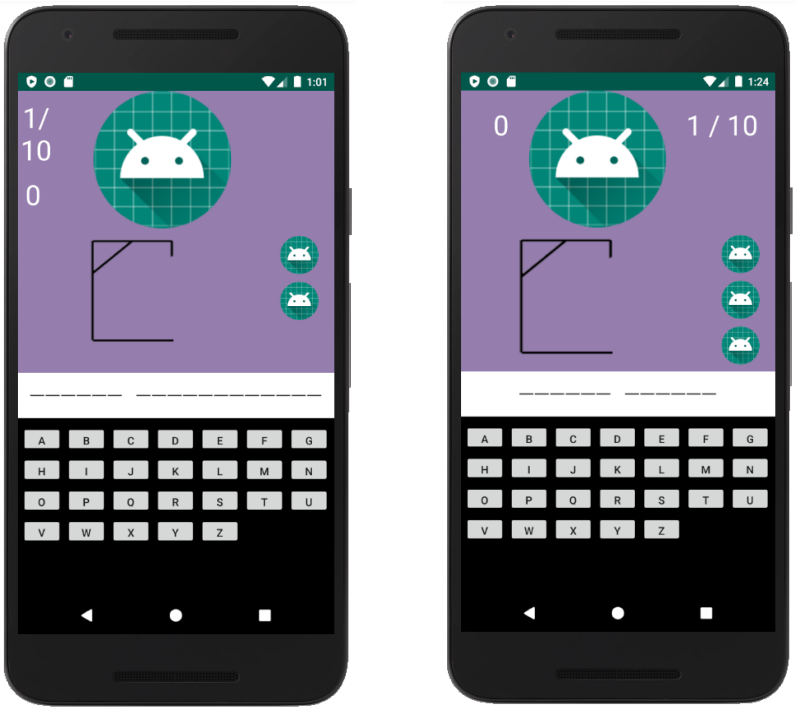
El *ahorcado* no se incluye en la aplicación final, debido a que faltan detalles que integrar en este, comentados con posterioridad, lo que provoca que aún no esté listo para lanzarse. Sin embargo, su funcionalidad principal ha sido implementada y probada, de manera exitosa, como se puede observar a continuación.



44. Interfaz final Ahorcado. *Fuente: propia.*

En estas imágenes se puede comprobar que la operatividad principal funciona de manera correcta, lo que no sucede con los comodines, cuya funcionalidad debe ser añadida, junto con la tarea de mejora gráfica, las cuales han sido incluidas como línea de futuro, para su modificación y finalización en sucesivas implementaciones y mejoras que pueda recibir *NamEx*.

Los desarrollos previos realizados para conseguir la interfaz mostrada van a ser comentados a continuación:



45. Interfaz inicial Ahorcado. *Fuente: propia.*

Inicialmente, la interfaz diseñada es bastante básica, sus colores rompen con el arte del resto de la aplicación y el teclado donde aparecen cada una de las letras del abecedario, sin contar la ñ que se agrega con posterioridad, es muy simple. Sin embargo, el objetivo de estos primeros diseños es permitir realizar pruebas para cerciorar el correcto funcionamiento del código implementado.

Finalmente, aunque el diseño se ha mejorado y la funcionalidad también, es necesario la adición de algunas características, tales como que la imagen se congele, al igual que sucede en otras modalidades, una vez se adivine o se falle el nombre buscado de manera que el usuario pueda observar la solución final en pantalla un instante breve de tiempo, que facilite la memorización del nombre de dicho alumno o alumna. Asimismo, agregar funcionalidad a los comodines, debido a que ahora mismo carecen de ella.

Estas tareas son sencillas, debido a que aparecen de manera muy similar o idéntica en otras partes de la aplicación. Esto permite que quién realice este trabajo tenga que conocer el funcionamiento del resto de modos de juego, familiarizándose así con el código, con lo que no va a tener ningún problema en acabar las tareas que recién han

sido comentadas. Todo ello va a ser comentado en la línea de futuro incluida en el apartado 8.2.4.

4.5.1 Implementación y funciones importantes.

Una de las funciones a destacar es la que realiza la adición de las vistas dinámicas que contienen cada una de las letras que conforman el nombre y el primer apellido del estudiante que está siendo preguntado. Para ello, primeramente, se han eliminado los acentos de estas palabras, debido al problema comentado en el apartado 5.3. Una vez se completa esta operación, se colocan en dos variables distintas el nombre y el apellido y utilizando su tamaño, se ejecuta un bucle donde se crea una *view* por cada carácter que compone la palabra. Una vez se ha completado esta tarea, el usuario puede pulsar los botones que ve en pantalla para intentar adivinar el nombre oculto. El método que recoge esta funcionalidad se llama *letterPressed*. Es el método más importante de esta modalidad, cuyo código se incluye en el anexo V, al que llama cada botón una vez se ha pulsado, cuya función es recorrer cada una de las letras de las dos palabras ocultas en la pantalla, mostrando las letras que coincidan con la que contiene el botón que ha sido pulsado. De manera que, si al acabar dicha comprobación no se ha mostrado ninguna letra, el botón que ha sido pulsado cambia su imagen añadiéndose una cruz roja encima de este y se muestra una parte más del ahorcado, como se ha podido observar en el apartado superior.

Si se ha mostrado alguna letra en la pantalla, el botón pulsado muestra un *tick* verde encima de este, se comprueba si todas han sido ya mostradas, aumentando la puntuación y pasando a la siguiente pregunta. Por el contrario, si se completa el ahorcado antes de que esto suceda, se avanza a la siguiente cuestión, pero no se aumenta la puntuación puesto que no se ha logrado acertar el nombre del estudiante oculto.

4.6 Resultados.

Una vez se completa cada uno de los modos de juego expuestos anteriormente, las puntuaciones obtenidas se muestran en una pantalla de la siguiente manera:

46. Interfaz final Resultados. *Fuente: propia.*

Como se puede observar, cada modo de juego envía los datos, que ha recabado durante la ejecución de la misma, al finalizar este. Estos son recogidos en una pantalla que resume todos, las respuestas correctas y fallidas, la puntuación obtenida y el tiempo que ha tardado en completar el juego. Datos que recopilan la actuación del usuario, lo que le permite a este recordar o ser consciente de lo bien o mal que lo ha hecho.

Dicha información, junto con su nombre y el número de intento realizado en ese modo de juego son enviados a *Firebase* para su almacenamiento, lo que permite una

futura consulta de los mismos que va a dar lugar al ranking con el que cuenta la aplicación.

Para lograr dicha interfaz, se ha realizado una implementación previa, más sencilla, cuya función era comprobar que los datos se enviaban y se recibían bien tanto a su interfaz como a la base de datos nombrada anteriormente, como se puede observar a continuación.

47. Interfaz inicial Resultados. *Fuente: propia.*

Aunque su estructura es similar, el diseño era más básico debido a que su principal utilidad no es esa. Sin embargo, cumple correctamente la razón por la que fue diseñada, comprobándose que el código desarrollado hasta el momento funciona adecuadamente.

4.6.1 Implementación y funciones importantes.

La función más importante es el envío de los datos a *Firebase* para almacenarlos y que puedan ser consultados por el resto de usuarios que poseen la aplicación. Al mismo tiempo, es una tarea sencilla de entender, solo se necesita obtener la referencia de la base de datos y la instrucción para enviar la variable *Puntuación*, que contiene los datos, a una tabla determinada dentro de *Firebase*.

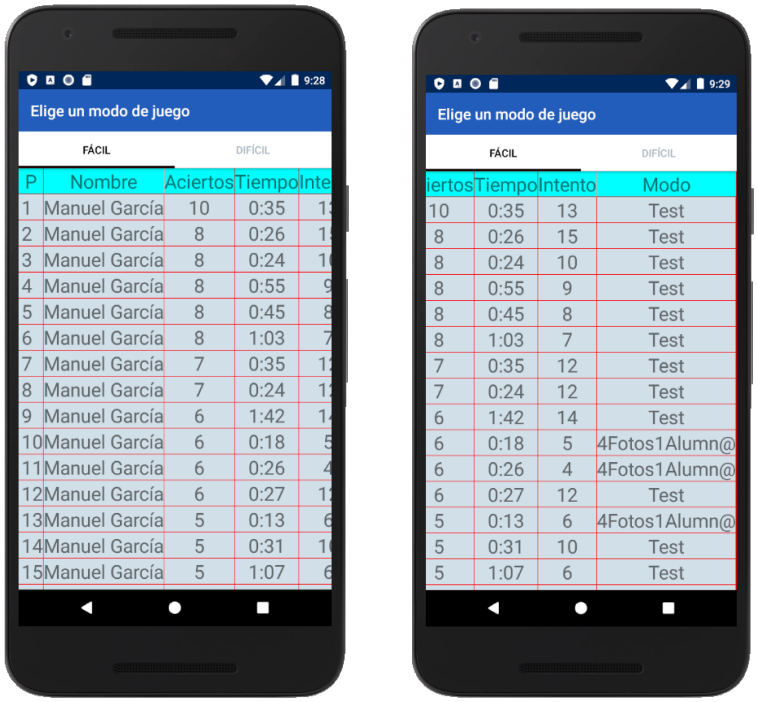
Asimismo, se almacena el índice que indica cuantos intentos se han realizado en cada modo de juego, que va incluido dentro de la variable nombrada anteriormente, de la cuál cabe destacar el cálculo que se realiza para obtener el tiempo que ha empleado el usuario para completar dicha modalidad.

4.7 Ranking.

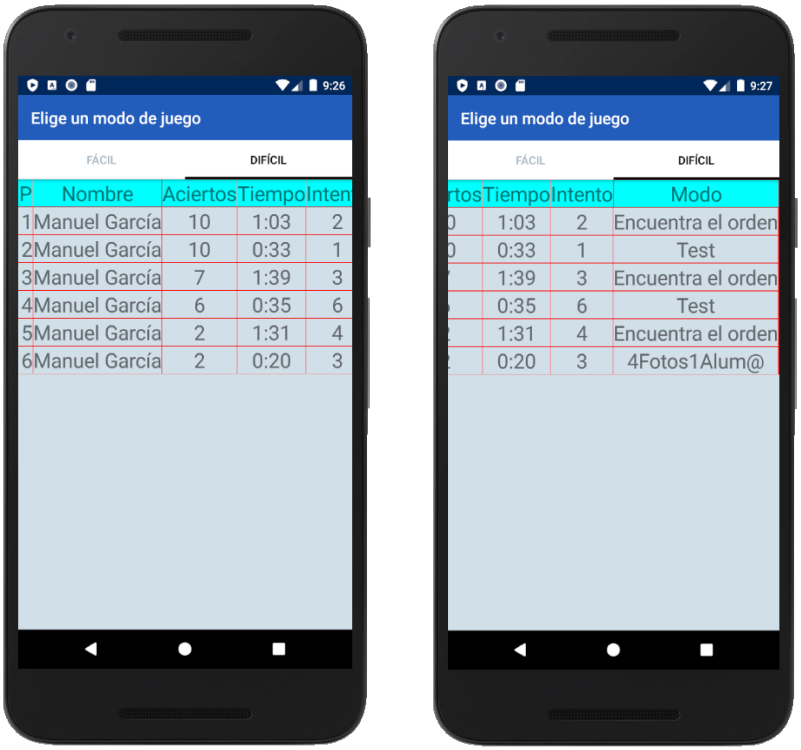
Por último, una vez las puntuaciones que se obtienen al jugar a cada uno de los modos de juego con los que cuenta la aplicación, es necesario añadir una funcionalidad que permita ver reunida dichos datos y esa es la idea que da lugar a este apartado *Ranking*. Sin embargo, esta idea inicial se divide en dos apartados, permitiéndose, en el primero de ellos, la observación de los datos obtenidos por el usuario y en el segundo, todas las puntuaciones que están almacenadas en *Firebase*.

4.7.1 Puntuaciones globales.

Este apartado, como su nombre indica, permite consultar todas las puntuaciones almacenadas en la base de datos. Cuenta con la siguiente interfaz:



48. Ranking global, modo fácil. *Fuente: propia.*



49. Ranking global, modo difícil. *Fuente: propia.*

Como se puede observar, en la pantalla se muestra el nombre del usuario que ha completado una partida, además de las respuestas acertadas, el tiempo que ha

empleado y el modo de juego entre otros, tanto de los modos fáciles, como de los difíciles. Esto permite consultar puntuaciones obtenidas por otros docentes promoviendo la competitividad entre ellos y la mejora de los resultados obtenidos.

4.7.2 Mis resultados.

Por último, esta sección permite la consulta de las puntuaciones obtenidas por el usuario que ha iniciado sesión en la aplicación, como se puede observar en la siguiente imagen:

50. Mis resultados. *Fuente: propia.*

Los datos obtenidos de la base de datos se filtran, mostrándose únicamente por pantalla los que pertenecen al docente que accede a esta funcionalidad de la aplicación. Esto permite que el usuario pueda observar las puntuaciones que ha obtenido a lo largo de la utilización de la misma, pudiendo consultar también el intento en el que lo ha conseguido y el tiempo invertido, percibiendo la mejora obtenida a lo largo los juegos realizados.

Capítulo 4

5. Problemas.

En este apartado se describen algunos de los problemas y dificultades encontradas en el proceso de implementación de la aplicación y, lo más importante, la forma de superarlos.

5.1 Test y Juego de tiempo.

Durante el desarrollo de este modo de juego se encontraron los siguientes conflictos:

5.1.1 Error al usar la clase *CountDownTimer.*

Al realizar la implementación inicial de este modo de juego se incluyó la siguiente barra, que iba rellenándose según avanzaba el tiempo para informar al usuario de cuánto tiempo le quedaba para responder a la pregunta.

Sin embargo, esta no llegaba al final, se quedaba a una pulsación de completarse que equivalía a un segundo. Posteriormente la pregunta se terminaba, sorprendiendo al usuario, ya que la barra no se había completado.

Buscando posibles soluciones en diversos foros de Internet, se encontró que dicha cuestión ya era conocida por otros usuarios. Concretamente, en la página web *stackoverflow*, un usuario (Jeremy, D) había realizado una pregunta relativa al error comentado que estaba presente en su código y otros usuarios habían aportado posibles soluciones.

Posteriormente, tras probar las soluciones comentadas por los usuarios, no se obtuvo un resultado consistente, debido a que, en ocasiones, el comportamiento era correcto, pero en otras ejecuciones volvía a aparecer el error.

Como consecuencia, y teniendo en cuenta que mediante este diseño el usuario no conoce realmente los segundos que le quedan para poder seleccionar una respuesta, se sustituyó dicha barra por una cuenta atrás numérica con los segundos que restan para contestar a la pregunta, como puede observarse en la imagen 4.11.

5.1.2 Respuesta seleccionada, ¿correcta o errónea?

Cuando se tenía una muestra con una única pregunta, se implementó la funcionalidad que coloreaba de verde la respuesta que se había seleccionado si esta era correcta, pintando de rojo en caso contrario, pero manteniendo la adición del color verde en la respuesta que fuese cierta.

Al aumentar la muestra a diez preguntas, una vez se seleccionaba una respuesta, se avanzaba a la siguiente cuestión, dando a conocer al usuario únicamente que la opción seleccionada era la correcta si se había aumentado la puntuación, sin poderse observar de una manera más sencilla, ni conocerse la respuesta acertada en caso de que el usuario hubiese seleccionado una errónea.

La solución encontrada fue la adición de pausas, de alrededor de un segundo entre cuestión y cuestión, de manera que el usuario pudiese observar la información que aparecía en pantalla una vez se había seleccionado una de las posibles respuestas.

5.1.3 Selección múltiple de respuestas en una pregunta.

Este error se descubrió una vez se habían introducido los segundos de pausa en las preguntas que se iniciaban en el momento en que el usuario había respondido a la pregunta en cuestión.

En ese momento, todos los botones de las respuestas seguían activos y al pulsar uno o varios de ellos, el sistema lo almacenaba y utilizaba cada elección como

respuesta de las sucesivas preguntas de forma ordenada. Es por ello que, se tenían que desactivar las botones de manera que el usuario no pudiese seleccionar ninguna opción más hasta que no se avanzase de pregunta, activándose de nuevo en dicho momento, lo cual no resultó complicado.

El desarrolló avanzó y se introdujeron los comodines, donde la solución anterior no era válida dado que se debía comprobar si un comodín ya había sido agotado. De manera que, al activar todos los botones de nuevo, este debía seguir desactivado, para lo cual se implementó un módulo que comprobaba el estado previo de cada comodín, guardándose cada uno de ellos en una variable que se restauraba cuando se volvían a activar las respuestas.

5.2 Encuentra el orden.

Durante el desarrollo este modo de juego se encontraron los siguientes conflictos:

5.2.1 Generar las *views* de cada letra de forma dinámica.

El primer error se presentó al inicio de la implementación del diseño, el cual, debía contener una *view* donde se mostrasen las letras que conforman el nombre del alumno por el que se pregunta.

La idea inicial era añadir una única *view* que contuviese el nombre completo del alumno y que cada vez que el usuario pulsase una letra de la pantalla se fuese añadiendo en la última posición de una cadena de caracteres, *string*, que se iba a mostrar en dicha *view*, sustituyendo el signo que indicaba el espacio reservado para una letra el cual era **‘_’**, por la letra seleccionada.

Este diseño presentaba dos problemas principales. El primero se detectó al realizar pruebas con la interfaz gráfica, ya que, al tener una única vista, el nombre que se mostraba variaba su posición en la pantalla dependiendo de la longitud, algunos se mostraban centrados y otros aparecían desplazados del centro. El segundo apareció en el momento de adición de los comodines, al tener que introducir letras en posiciones aleatorias, la estructura seleccionada no era la correcta.

Como consecuencia se indagó en la generación de vistas, comprobándose si existían maneras de crearlas de forma dinámica, creando una por cada letra que conformase el nombre que se pregunta. Después de ratificar su existencia, se añadieron al proyecto, introduciéndose los cambios necesarios en la estructura existente en dicho momento que permitiera lograr la obtención del comportamiento deseado.

Por último, al intentar implementar el comportamiento de los comodines para que estos funcionasen de forma correcta, se encontraron diferentes errores que van a ser descritos con posterioridad, en los subapartados 5.2.3 y 5.2.4.

5.2.2 Adición del botón de borrado.

Se presentó el caso en el que el usuario se equivoca al pulsar el botón e introduce en la palabra una letra que sabe que es errónea, sin embargo, la manera de solucionar ese error es rellenar todas las letras que conforman dicha palabra que al ser incorrecta se reseteará, permitiendo de nuevo al usuario la subsanación del desliz cometido.

Para facilitar esto, se introduce un pequeño botón que contiene el emoticono de una papelera de reciclaje. Al pulsarse, se elimina la última letra añadida por el usuario, sin embargo, el botón con la letra que acababa de ser retirada no volvía a aparecer. Para subsanar este error, se añadió una variable lista, *array*, donde se iba almacenando las referencias, *id's*, de los botones que se iban pulsando, eliminándose el último de la lista cuando el botón era pulsado, y devolviéndose la visibilidad al botón que contuviese esa referencia.

Debiéndose comprobar el caso en que el usuario pulsara el botón sin haber introducido ninguna letra en la palabra, anulando su función si se encontrara en ese caso, para evitar el cierre inesperado de la aplicación.

5.2.3 Comodín que desvela la primera letra del nombre preguntado.

Al incluir este comodín, que coloca la primera letra del nombre o apellido que esté presente en la pantalla, aparecieron varios problemas, los cuales van a ser explicados a continuación.

El primer error se detectó cuando ya se había terminado el desarrollo del comodín, el cual se comportaba de la manera esperada, salvo cuando el usuario fallaba, en ese momento todas las letras se reseteaban perdiendo la ventaja que ofrecía el comodín que había sido utilizado en esa misma cuestión. La solución implantada fue la comprobación del comodín para constatar si había sido usado, guardando la letra, que era incluida de nuevo en la primera posición una vez se había realizado el reseteo completo de la palabra.

El segundo error se debió al descubrimiento de un caso de uso que no había sido contemplado. Cuando el comodín era usado, no se comprobaba que ya hubiera una letra en la primera posición. De ser así, se comprobaba el *id* del primer botón pulsado almacenado en una lista comentada anteriormente, volviéndose a hacer visible y borrándose tanto el *id* como la letra en cuestión.

El último problema se detectó al realizar pruebas con el comodín que añade una letra en una posición aleatoria entre la segunda y la última, dependiendo de donde se colocara esta, podía provocar un fallo cuando se utilizara este comodín, concretamente cuando la letra se posicionaba en la segunda posición, para lo que se diseñó la siguiente solución.

Debido a como está desarrollado el código se tenía que comprobar ese caso y de estar en él, cuando se usa este comodín se tiene que aumentar el número de letras que contiene la palabra en dos, debido a que la letra añadida anteriormente no aumenta dicha variable, por lo cual, la siguiente letra pulsada por el usuario sustituía a la existente en la segunda posición.

5.2.4 Comodín que añade una letra en una posición aleatoria.

La función de este comodín es añadir una letra en una posición aleatoria, entre la segunda y la última del nombre o apellido que está siendo preguntado en la pantalla. Al finalizar su implementación aparecieron varios conflictos, los cuales van a ser explicados a continuación.

El principal problema existente estaba causado por la forma en la que estaba realizada su implementación. Para comprobar en qué posición debía colocarse la siguiente letra que fuese pulsada por el usuario se utilizaba una variable que iba

aumentándose con cada carácter que se iba añadiendo a la pantalla, después de ser pulsado por la persona que estaba jugando. Al utilizar este comodín, el tamaño no podía aumentarse debido a que, si la palabra estuviese vacía, se empezaría a escribir en la segunda posición.

La solución que se optó por incorporar fue incluida en el método que se encarga de recibir la pulsación de cada botón, comprobando si en esa posición ya hubiese una letra previa, aumentándose la longitud de la palabra en dicho momento y colocándose la que se acaba de pulsar en el siguiente hueco disponible.

Sin embargo, aún existía un problema al integrar el uso con el primer comodín desarrollado. Si se cumple el escenario donde la primera letra ha sido añadida por el comodín que realiza dicha función y al usarse este, la letra se coloca en la segunda casilla hay que comprobar que la tercera casilla este vacía, de ser así, debemos aumentar en dos la variable que contiene el número de letras que están presentes en la palabra que se muestra por pantalla, para evitar que se sobrescriban ambas casillas.

Por el contrario, si la primera casilla no estuviera vacía, pero esta hubiese sido rellenada por el usuario, habría que aumentar en uno la variable donde se almacenan el número de letras, siempre que la tercera casilla estuviese ocupada, de no ser así no se aumenta ninguna variable.

Al igual que en el último caso extremo que se contempló, donde si todas las letras menos la última de la palabra están rellenas, y se utiliza este comodín, ocupando la letra aleatoria resultante la última posición de esta, se debía comprobar si la palabra formada era correcta o no, escenario que no se realizaba correctamente en un principio.

El código finalmente desarrollado puede verse en el anexo IV, que recoge todo lo anteriormente descrito y cuya consulta puede solucionar las posibles dudas que hayan podido surgir después de realizar esta lectura.

Por último, como ya se tenía la experiencia de los problemas que habían aparecido al desarrollar el funcionamiento del comodín anterior, se contempla la comprobación de la ocupación de la casilla donde iba a añadirse la letra aleatoria que proporciona este comodín. Asimismo, se constata si el comodín había sido usado, guardando la posición que ocupaba la letra, junto al carácter de esta, que era de nuevo incluida una vez se había realizado el reseteo de la palabra.

5.3 Acentos en los nombres en el Ahorcado.

Una vez se había concluido el desarrollo del modo de juego que recibe el nombre de Ahorcado, en el cual, se incluía un teclado que contenía todas las letras del abecedario que el usuario debía pulsar para comprobar si el carácter seleccionado formaba parte del nombre o apellido del alumno preguntado. Sin embargo, cuando estos contenían un diacrítico y el carácter correspondiente era pulsado por el usuario la letra no aparecía en la palabra, aunque formara parte de esta.

La solución por la que se optó fue la eliminación de los diacríticos después de ocultar la palabra en la interfaz gráfica, por ende, cuando el usuario seleccionaba una letra, el método que comprobaba si esta formaba parte del nombre o no realizaba una comparación en la variable que ya no contenía ningún tipo de diacríticos, mediante lo cual se logra la ilusión de que, si se pulsa el carácter correcto, este se va a desvelar en la pantalla del dispositivo móvil utilizado, contenga acento o no.

5.4 Obtención del sexo de los alumnos.

Durante la realización de la mayor parte del desarrollo los datos con los que se han trabajado han sido locales, pero a la hora de proporcionar las respuestas a las preguntas aparecían mezclados nombres masculinos y femeninos, independientemente del sexo del alumno que estaba siendo preguntado pudiendo eliminar las opciones que no contuviesen dicho sexo.

Para solucionarlo se utilizó la API llamada Genderize.io que, analizando el nombre del alumno, devolvía su sexo con un porcentaje de acierto más que aceptable, pudiéndose de esta manera dividir las respuestas ofrecidas, obteniéndose una mejor experiencia de usuario al utilizar la aplicación.

Sin embargo, para algunos nombres, principalmente los de mujeres que estaban acentuados, devolvía su sexo como nulo. Por consiguiente, se realizó una segunda petición con todos los nombres que no tenían sexo definido, eliminando los diacríticos, permitiendo de esa manera la obtención del sexo de la mayoría de ellos.

Este proceso se fue realizado debido a que el Campus Virtual de la Universidad de Extremadura (*CVUEx)*, de donde se van a obtener los datos reales que van a ser usados en la aplicación, no almacena este dato en particular, siendo esta solución un recurso que obtenía unos resultados aceptables. Por último, la medida adoptada con los alumnos cuyos sexos seguían siendo nulos al finalizar las dos solicitudes a la API fue eliminar la aparición de estos como las respuestas de las preguntas, subsanando así el problema.

Capítulo 5

6. Puntos fuertes.

En este capítulo, se detallan los puntos fuertes del proyecto, los cuales hacen realmente interesante el uso de esta aplicación con fines educativos por parte de los docentes inscritos en la Universidad de Extremadura. Estos son los siguientes:

1. No existe ninguna aplicación que facilite el aprendizaje de los nombres de los alumnos a los docentes. Como se ha descrito en el estado del arte, después de realizar una profunda búsqueda, no se obtuvo resultado acerca de alguna herramienta parecida a la que ha sido diseñada. Como resultado, una vez detectada esta carencia, el desarrollo de esta aplicación y en el momento que cuente con datos reales, va a ser una aplicación funcional que cubra de una manera muy correcta la necesidad que ha dado vida a su desarrollo.

2. Se aprende jugando. Debido a que la idea principal del desarrollo ha sido la gamificación, se han desarrollado modos de juego amenos que fomentan su uso, logrando al mismo tiempo un aprendizaje más cómodo y divertido que apenas acarrea esfuerzo para el usuario.

3. Se cuenta con un ranking global, cuyo objetivo es fomentar la participación de los docentes y la competitividad entre ellos, debido a que los resultados obtenidos son públicos para el resto de usuarios.

4. Se pueden observar los resultados conseguidos individualmente, logrando ver el progreso que se van obteniendo al hacer uso de la aplicación.

5. Se almacenan los datos que pueden ser utilizados para alguna posible futura investigación como, por ejemplo, la efectividad de la herramienta.

Capítulo 6

En este capítulo se detalla la planificación seguida durante el desarrollo de este proyecto. Asimismo, se expone el medio utilizado para la comunicación y coordinación con los tutores de la investigación.

7. Planificación y coordinación del proyecto.

7.1 Planificación del proyecto.

Para explicar el progreso y cuantificar el tiempo destinado al proyecto, se realiza una planificación del proyecto agrupada en fases. Según las tareas realizadas y su finalidad, se definen las siguientes fases:

Fase inicial: En esta primera fase, se establece una idea general del proyecto, así como de su alcance y objetivos. Se concretan una planificación inicial de las tareas que se intuyen a realizar y se definen los posibles objetivos.

Fase de estudio y análisis del entorno: En esta segunda fase, se realiza un repaso de los conceptos de Android ya conocidos, principalmente conceptos incluidos en la guía de arquitectura de apps, ciclos de vida y características relacionadas con el diseño y la programación en dicho lenguaje. Asimismo, se estudian posibles bases de datos a utilizar, tanto *Firebase* como *SQLite*.

Fase de selección: En este periodo se justifica la elección de las herramientas que se utilizarán para la construcción de la aplicación Android.

Fase de estudio de requisitos: Se establecen los requisitos necesarios para realizar el proyecto. Esta fase se centra en conocer las necesidades de los datos, cuales son, cómo se pueden ser obtenidos y dónde van a ser almacenados.

Fase de diseño: Una vez determinado el punto anterior, se perfila el diseño de la aplicación, definiendo la arquitectura a implementar, bases de datos a utilizar, determinar los datos que se necesitan para hacer funcionar la herramienta, cómo y de dónde obtenerlos.

Fase de implementación: Esta fase es sin duda la más larga del proyecto, ya que en ella se realiza el completo desarrollo de la herramienta desde cero, encontrándose con errores, comentados en el capítulo cinco, obteniendo finalmente el producto deseado.

Fase de pruebas: A lo largo del desarrollo se han ido realizando una serie de pruebas que validan el correcto funcionamiento de cada uno de los aspectos que nos presenta la herramienta, obteniendo errores que deben ser solucionados.

Fase de documentación: Aunque la documentación se realiza de forma paralela a la ejecución de todas las fases, en esta última se necesita una temporalidad adicional para finalizar la documentación de **la investigación**.

Fase	Duración de la fase en horas (aprox.)
Fase inicial	20
Fase de estudio y análisis del entorno	40
Fase de selección	15
Fase de estudio de requisitos	45
Fase de diseño	65
Fase de implementación	200
Fase de pruebas	25
Fase de documentación	155
TOTAL	565

7.1. Fases y termporalidad del proyecto. *Fuente: propia.*

7.2 Coordinación del proyecto.

Los medios utilizados para la comunicación y la coordinación de los tutores del proyecto con el alumno han sido principalmente cuatro. La vía más rápida de comunicación ha sido la aplicación de mensajería instantánea *Whatsapp*, donde intercambiábamos mensajes e impresiones sobre inquietudes y dudas en ciertos aspectos de la implementación o toma de decisiones, siendo una herramienta a la que estamos muy acostumbrados a utilizar en el día a día.

Inicialmente, utilizamos los correos electrónicos para intercambiar información acerca de la investigación, detallando cómo iba a desarrollarse, principalmente para obtener una idea de partida del proyecto. Además, se ha utilizado un espacio virtual facilitado por el Campus Virtual de la Universidad de Extremadura (*CVUEx*) a través de la plataforma de *e-learning* (*Moodle*) que ofrece soporte a la comunidad universitaria.

Asimismo, junto con las reuniones presenciales pertinentes, se han realizado diversas llamadas telefónicas y videoconferencias, donde se ha podido observar el progreso del desarrollo y se han intercambiado impresiones, inquietudes y diversas dudas que han podido surgir a lo largo de este tiempo. Estos tipos de herramientas nos ha permitido trabajar online de forma colaborativa, aportando comunicación instantánea y actualizada.

Capítulo 7

8. Conclusiones y líneas de futuro.

Una vez llegados al final de la investigación, se impone realizar una síntesis conclusiva de los diversos resultados conseguidos. Además, como norma en cualquier trabajo de investigación, se propone la mejora del contexto estudiado.

Se adopta una doble perspectiva; primero, se presentan las conclusiones adquiridas de la investigación, para lo cual, antes se exponen sus limitaciones esenciales. En segundo lugar, se efectúan propuestas de futuro, algunas obtenidas a partir de aspectos consensuados con los tutores, o bien, a raíz de ideas que han ido surgiendo durante el proceso del proyecto.

8.1 Conclusiones.

El diseño y la implementación de la aplicación desarrollada con el fin de ser utilizada en un futuro próximo en la Universidad de Extremadura; denominada NamEx, se ha convertido en el propósito principal de este proyecto.

Venciendo las limitaciones propias de las herramientas utilizadas, la documentación incompleta de algunas funciones que se han utilizado y los conocimientos mínimos iniciales acerca de la creación de una aplicación móvil, se ha conseguido crear y construir una herramienta, que a falta de contar con datos reales posee un gran funcionamiento.

Se han superado los desafíos iniciales que supone el desarrollo de una aplicación tan novedosa de la que no existía otra que pudiera servir de guía. *NamEx* facilitará el aprendizaje de los nombres de los alumnos a todos aquellos docentes de la Universidad de Extremadura que la utilicen.

Para conseguir el resultado presentado, se ha estudiado a fondo las herramientas previamente seleccionadas, que ofrecen diferentes posibilidades a la hora de crear una aplicación móvil, en cuanto a plataformas y tipos de desarrollo.

No obstante, por el *framework* utilizado, la aplicación solo estará disponible para dispositivos *Android*, dejando a un pequeño porcentaje de usuarios que cuentan con dispositivos con otro sistema operativo, principalmente *iOS*, sin poder disfrutar de los beneficios que *NamEx* ofrece, aunque podrá estar disponible en un futuro si se continua con el desarrollo.

Como es bien sabido, la tecnología ha invadido nuestras vidas. Casi todo el mundo posee al menos un dispositivo móvil, lo que hace ideal que la aplicación se haya enfocado a esta plataforma, cubriendo la necesidad detectada y habiéndose centrado en la usabilidad durante todo el proceso de desarrollo de la misma.

8.2 Líneas de futuro.

Las líneas de futuro están enfocadas a las mejoras y a la ampliación de las características de la aplicación, la finalización de la herramienta con la introducción del enlace al Campus Virtual para la obtención de datos reales que permita cumplir el objetivo con el que ha sido desarrollado la aplicación.

8.2.1 Conexión con el Campus Virtual.

La primera línea de futuro y quizás, la más importante, para poder terminar y lanzar la aplicación es la integración con el *Moodle* de la UEx, a través de servicios web que ya están desarrollados por el Campus Virtual de la Universidad de Extremadura para la aplicación móvil propia de esta.

Esta tarea es fundamental para la consecución del objetivo primario con la que nace *NamEx*, el aprendizaje de los nombres de los alumnos por parte de los docentes de la UEx. Sin embargo, por ahora, no se pueden asociar dichos apelativos a su rostro, debido a que se trabajan con datos ficticios.

A pesar de ello, la aplicación se desarrolla y organiza para que una vez se realice dicha integración con los servicios web y se obtengan datos reales, los cambios que haya que realizar sean los mínimos posibles, facilitando, de esta manera, la puesta en

marcha de la aplicación una vez se realicen dichas conexiones, que resultan imprescindibles para finalizar el desarrollo y cubrir la necesidad que da vida a *NamEx*.

Por todo ello, si hay que priorizar un siguiente paso en el desarrollo, sin ninguna duda debe ser este.

8.2.2 Filtrar los alumnos por asignaturas.

En esta línea se pretende dar funcionalidad a la parte de la aplicación que filtra los alumnos por asignaturas, que son utilizados para generar las preguntas de los diferentes modos de juego y a los que se accede a través del menú *asignaturas*, en la pantalla inicial de la aplicación.

Esta opción se ha diseñado con la intención anteriormente expuesta, sin embargo, su funcionalidad no está implementada debido a que no se ha priorizado, al no contar con datos reales de alumnos obtenidos mediante el curso del profesor *logueado*.

Una vez se ha realizado integración comentada en la primera línea de futuro, se puede dar funcionalidad a esta parte del menú, al tener los datos necesarios para ello, lo cual permitiría a cualquier docente el estudio específico de los alumnos de una asignatura concreta.

8.2.3 Realizar estudios con los datos que se han almacenado.

Los datos que son generados durante el uso de *NamEx* son almacenados para ser mostrados a los usuarios de la misma, tanto el ranking global como las puntuaciones individuales. Asimismo, resulta interesante plantear algún uso alternativo de la cantidad ingente de información generada por los usuarios, que va a ser mayor cuantos más docentes la utilicen.

En esta línea de futuro se plantea que estos puedan ser utilizados para realizar estudios de investigación que midan el proceso de aprendizaje seguido por el profesorado y la mejora en los resultados que han obtenido con el uso de esta herramienta.

El análisis de ciertas variables (media, moda, relación de índice y resultado), podrían mostrar correlaciones que permitan afirmar o no la utilidad de *NamEx*, además, de poder plantear cambios en los modos de juego ya instaurados, con el fin de mejorar el proceso de aprendizaje ofrecido por los mismo. Por lo que, igual es interesante, añadir algún campo más a estos datos recogidos, con el fin de conseguir unos resultados más precisos.

8.2.4 Finalizar modo de juego Ahorcado y añadir nuevos.

La última de las líneas busca la finalización del modo de juego *Ahorcado*, el cual necesita cambios en su interfaz y en su funcionalidad, antes de ser publicado en la aplicación final. Estos cambios, que ya han sido referidos con anterioridad, tales como la adición de pausas entre una pregunta y la siguiente o la implementación de la funcionalidad de los comodines, no son nuevos en la aplicación, debido a que aparecen de manera idéntica o muy similar, en otras partes de esta. Esto motiva a quién realice esta línea de futuro a conocer el funcionamiento del resto de modos de juego, familiarizándose así con el código, con lo que no va a tener ningún problema en terminar este *Ahorcado*.

Asimismo, se plantea la implementación de nuevos modos de juego y mejoras en los ya existentes, basados en la opinión de los usuarios que los jueguen y en las puntuaciones que estos obtengan.

Para ello se necesita investigar acerca de nuevos modos de juego basados en la gamificación, que faciliten el aprendizaje y que puedan ser trasladado e implementados para finalmente, ser incluidos en *NamEx*.

Al mismo tiempo que se desarrollan nuevos filtros para mostrar diferentes datos que le puedan resultar interesantes conocer al usuario, se amplían los datos recogidos por la aplicación con diversos objetivos como el uso en estudios. Además, de agregar más categorías de dificultad en los juegos, consiguiéndose que los docentes sigan utilizando la aplicación a lo largo del curso universitario.

Referencias bibliográficas.kn

AGUADO. J. E., RICO, M., SÁNCHEZ, H., Y VALOR, M., 2011. Title-Accessing mobile learning records in Moodle through web services. *IEEE-RITA* [en línea], vol. 6, no. 3, pp. 95-102. [Consulta: 23 enero 2020]. Disponible en: http://wurfl.sourceforge.net/.

ANDROID INC., 2019. Introducción a Android Studio | Desarrolladores de Android. [en línea]. [Consulta: 23 enero 2020]. Disponible en: https://developer.android.com/studio/intro.

APPLE INC., 2020. Xcode - Apple Developer. [en línea]. [Consulta: 23 enero 2020]. Disponible en: https://developer.apple.com/xcode/.

BV, TIOBE Software, 2020. index | TIOBE - The Software Quality Company. [en línea]. [Consulta: 24 enero 2020]. Disponible en: https://www.tiobe.com/tiobe-index/.

CADENAS, R., 2019. ¿Que necesito? ¿Web Apps, App Nativa o App Híbrida? - GSoft. [en línea]. [Consulta: 23 enero 2020]. Disponible en: https://www.gsoft.es/articulos/que-necesito-web-apps-app-nativa-o-app-hibrida/.

CERNUDA DEL RÍO, A., 2014. Replanteándose el entrenamiento memorístico y repetitivo. *ReVisión*, vol. 7, no. 2, pp. 8. ISSN 1989-1199.

FIREBASE INC., 2019. Firebase Realtime Database | Firebase Realtime Database. [en línea]. [Consulta: 23 enero 2020]. Disponible en: https://firebase.google.com/docs/database?hl=es.

MURADAS, Y., 2018. SQLite para Android: La herramienta definitiva | OpenWebinars. [en línea]. [Consulta: 23 enero 2020]. Disponible en: https://openwebinars.net/blog/sqlite-para-android-la-herramienta-definitiva/.

NAHARRO, A., 2019. Frameworks para desarrollo de aplicaciones móviles híbridas | campusMVP.es. [en línea]. [Consulta: 24 enero 2020]. Disponible en: https://www.campusmvp.es/recursos/post/frameworks-para-desarrollo-de-aplicaciones-moviles-hibridas.aspx.

SESMA. M. y BATANERO. A., 2018. VERSUS: Apps híbridas VS Apps Nativas - Paradigma. [en línea]. [Consulta: 23 enero 2020]. Disponible en: https://www.paradigmadigital.com/dev/versus-apps-hibridas-vs-apps-nativas/.

SOLUCIONES I.P. HISPANAS, S.L., [sin fecha]. Desarrollo de aplicaciones móviles, diseño web mobile y responsive. [en línea]. [Consulta: 23 enero 2020]. Disponible en: https://www.solucionesip.com/servicios/aplicaciones-web-movil.

SQL, ORG., [sin fecha]. About SQLite. [en línea]. [Consulta: 23 enero 2020]. Disponible en: https://www.sqlite.org/about.html.

TROPICAL SERVER S.L., 2020. ¿Qué es Moodle? La Guía Definitiva sobre la plataforma Moodle. [en línea]. [Consulta: 23 enero 2020]. Disponible en: https://www.tropicalserver.com/ayuda/que-es-moodle/.

WIEGERT, C. Y LUCAS, E., 2019. What is Ionic Framework? - Ionic Documentation. [en línea]. [Consulta: 24 enero 2020]. Disponible en: https://ionicframework.com/docs/intro.

Anexo I: Clase de pregunta y algoritmo de generación de estas.

El propósito de este anexo es mostrar el código relativo a la clase *Pregunta* que, asimismo, contiene el algoritmo que genera las preguntas utilizadas por cada uno de los modos de juego que incluye la aplicación, de manera que facilite la comprensión del funcionamiento del mismo.

```
public class Preguntas {

    private String pregunta;
    private String respuesta1;
    private String respuesta2;
    private String respuesta3;
    private String respuesta4;
    private String respuesta_correcta;

    public Preguntas(){

    }

    public Preguntas(String url, String respuesta_correcta,
                     int caso) {

        if(caso==0){        //TODO url; Moodle
            this.pregunta = respuesta_correcta;
            this.respuesta_correcta = respuesta_correcta;
        }
        else if(caso==1){
            this.pregunta = respuesta_correcta;
                              //TODO url; Moodle
            this.respuesta_correcta =
"https://image.tmdb.org/t/p/w500/kqjL17yufvn9OVLyXYpvtyrFfak.jpg";
//respuesta_correcta; todo cambiar
        }
        this.respuesta1 = "";
        this.respuesta2 = "";
        this.respuesta3 = "";
        this.respuesta4 = "";
    }

    public Preguntas(String pregunta, String respuesta1, String
      respuesta2, String respuesta3, String respuesta4, String
      respuesta_correcta) {
        this.pregunta = pregunta;
        this.respuesta1 = respuesta1;
        this.respuesta2 = respuesta2;
        this.respuesta3 = respuesta3;
        this.respuesta4 = respuesta4;
        this.respuesta_correcta = respuesta_correcta;
    }

    public String getPregunta() {
        return pregunta;
    }
```

```
    public void setPregunta(String pregunta) {
        this.pregunta = pregunta;
    }

    public String getRespuesta1() {
        return respuesta1;
    }

    public void setRespuesta1(String respuesta1) {
        this.respuesta1 = respuesta1;
    }

    public String getRespuesta2() {
        return respuesta2;
    }

    public void setRespuesta2(String respuesta2) {
        this.respuesta2 = respuesta2;
    }

    public String getRespuesta3() {
        return respuesta3;
    }

    public void setRespuesta3(String respuesta3) {
        this.respuesta3 = respuesta3;
    }

    public String getRespuesta4() {
        return respuesta4;
    }

    public void setRespuesta4(String respuesta4) {
        this.respuesta4 = respuesta4;
    }

    public String getRespuesta_correcta() {
        return respuesta_correcta;
    }

    public void setRespuesta_correcta(String respuesta_correcta) {
        this.respuesta_correcta = respuesta_correcta;
    }

/*

    Generar preguntas()
  */

    private static void seleccionarAlumnoRandom
                              (ArrayList<Alumnos> respuestas) {

        //variables necesarias para seleccionar las respuestas
        Random RNG = new Random();

        Alumnos alumno;
        int alumnoSeleccionado;
        String sexo;
        sexo = respuestas.get(0).getSexo();
        //mientras no tengamos 4 respuestas.
        //Empezamos teniendo la correcta
```

```
        while(respuestas.size()!=4) {
        //todo tiene que haber minimo 4 personas de cada sexo.

            //seleccionamos un numero aleatorio
            alumnoSeleccionado =
                        RNG.nextInt(Common.alumnosList.size());

            //obtenemos el alumno correspondiente
            alumno = Common.alumnosList.get(alumnoSeleccionado);

            //todo posible sexo null
            if(alumno.getSexo() != null &&
                  alumno.getSexo().equals(sexo) &&
                        !respuestas.contains(alumno))
            {
                //si no esta en las respuestas lo añadimos
                respuestas.add(alumno);
            }
        }

    }

    private static void mostrarRespuestas(ArrayList<Alumnos>
                                          respuestas, int correcta) {

        Iterator<Alumnos> it = respuestas.iterator();
        Alumnos alumno = null;
        int i = 0;
        while(it.hasNext()) {
            alumno = it.next();
            System.out.print(alumno.getNombreCompleto());
            if(i == correcta) {
                System.out.println(" Respuesta correcta.");
            }
            else {
                System.out.println();
            }
            i++;
        }
    }

    private static void generarPreguntas(ArrayList<Alumnos>
                                                    respuestas){
        Random RNG = new Random();
        Alumnos alumno;
        //Seleccionamos al alumno que se va a preguntar y lo
imprimimos por pantalla
        int alumnoSeleccionado = RNG.nextInt(
                                    Common.alumnosList.size());

        //Seleccionamos al alumno basandonos en un número aleatorio
        alumno = Common.alumnosList.get(alumnoSeleccionado);

        while(alumno.getPreguntado()) {
            alumnoSeleccionado = RNG.nextInt(
                                    Common.alumnosList.size());
            alumno = Common.alumnosList.get(alumnoSeleccionado);
        }

        //Indicamos que se ha utilizado
        alumno.setPreguntado(true);
```

```
        //Lo modificamos en la lista
        Common.alumnosList.set(alumnoSeleccionado, alumno);

        //añadimos la respuesta a la lista de estas
        respuestas.add(alumno);

    }


    public static void generarPreguntasTest() {

        ArrayList<Alumnos> respuestas = new ArrayList<>();

        generarPreguntas(respuestas);

        seleccionarAlumnoRandom(respuestas);

        guardarPreguntaTestPreparada(respuestas);


    }

    private static void guardarPreguntaTestPreparada
                               (ArrayList<Alumnos> respuestas) {

        //guardamos la respuesta correcta antes de mezclar la lista

            //TODO sustituir datos Moodle    //alumno.getFoto();
        Preguntas pregunta = new
                  Preguntas(respuestas.get(0).getNombreCompleto(),
                        respuestas.get(0).getNombreCompleto(), 0);

        //shuffle a la lista, cambiar su orden de manera aleatoria
        Collections.shuffle(respuestas);

        int i=0;
        //guardamos cada posible respuesta en su apartado
        while(i<4){
            switch (i){
                case 0:
                    pregunta.setRespuesta1(
                        respuestas.get(i).getNombreCompleto());
                    break;

                case 1:
                    pregunta.setRespuesta2(
                        respuestas.get(i).getNombreCompleto());
                    break;

                case 2:
                    pregunta.setRespuesta3(
                        respuestas.get(i).getNombreCompleto());
                    break;

                case 3:
                    pregunta.setRespuesta4(
                        respuestas.get(i).getNombreCompleto());
                    break;
            }
            i++;
```

```
        }

        Common.preguntasList.add(pregunta);
        respuestas.clear();
        pregunta = null;
    }


    public static void generarPreguntasCuatroFotos() {
        Alumnos alumno = null;
        ArrayList<Alumnos> respuestas = new ArrayList<>();

        generarPreguntas(respuestas);

        seleccionarAlumnoRandom(respuestas);

        guardarPreguntaCuatroFotosPreparada(respuestas);


    }

    private static void guardarPreguntaCuatroFotosPreparada(
                                    ArrayList<Alumnos> respuestas) {

        //guardamos la respuesta correcta antes de mezclar la lista
            //Todo cambiar en el constructor para hacerlo dinámico!
        Preguntas pregunta = new
                  Preguntas(respuestas.get(0).getFoto(),
                        respuestas.get(0).getNombreCompleto(), 1);

        //shuffle a la lista, cambiar su orden de manera aleatoria
        Collections.shuffle(respuestas);

        int i=0;
        //guardamos cada posible respuesta en su apartado
        while(i<4){
            switch (i){
                case 0:

pregunta.setRespuesta1("https://image.tmdb.org/t/p/w500/kqjL17yufvn9
OVLyXYpvtyrFfak.jpg");//respuestas.get(i).getFoto()); todo cambiar
                    break;

                case 1:
                    pregunta.setRespuesta2(
                              respuestas.get(i).getFoto());
                    break;

                case 2:
                    pregunta.setRespuesta3(
                              respuestas.get(i).getFoto());
                    break;

                case 3:
                    pregunta.setRespuesta4(
                              respuestas.get(i).getFoto());
                    break;
            }
            i++;
        }
```

```
            Common.preguntasList.add(pregunta);
            respuestas.clear();
            pregunta = null;
        }

}
```

Anexo II: Método *OnClick* del método de juego fácil *Encuentra el orden*.

El propósito de este anexo es mostrar el código relativo al método *OnClick*, el cual es fundamental para el correcto funcionamiento del juego *Encuentra el orden*, de manera que facilite la comprensión de la implementación comentada en el apartado

```
public void onClick(View view) {

    Button clickedButton = (Button) view;
    if(clickedButton.getId() == R.id.borrar){
        if(!botones_pulsados.isEmpty()) {
            //String text = editText.getText().toString();

hacerVisibleBoton(botones_pulsados.get(botones_pulsados.size() -
1));
            botones_pulsados.remove(botones_pulsados.size() - 1);
            letras--;

            if(posLetra==letras){
                letras --;
            }
            charViews[letras].setText("");
        }
    }
    else {
            botones_pulsados.add(clickedButton.getId());
            clickedButton.setVisibility(View.INVISIBLE);

charViews[letras].setText(clickedButton.getText().toString());

            letras++;
            //comprobamos si la siguiente esta vacia, de esta manera
si se pone la ultima letra, pasamos a letras == longitudPalabra
        if (letras!=longitudPalabra &&
!charViews[letras].getText().toString().equals("")) {
            letras++;
        }
    }


    if (letras == longitudPalabra) {

        comprobarPalabraCompleta();

    }
}
```

Anexo III: Comodín que coloca la primera letra de la palabra buscada en el modo *Encuentra el orden*.

El propósito de este anexo es mostrar el código relativo al método del comodín que coloca la primera letra de la palabra buscada en el método *Encuentra el orden*, de manera que facilite la comprensión de la implementación comentada en el apartado

```
public void primeraLetra_Comodin(View view){

    nComodin.add(nComodin.size(),0);
    String nomApe=nombreOapellido();;
    char letra = nomApe.charAt(0);
    añadirLetraLista(letra);

    //por si el usuario ya ha empezado a escribir el nombre
    if(charViews[0].getText()!= letra){

        botones_pulsados.remove(0);
    }
    else{
        //por si el usuario ya ha empezado a escribir el
        //nombre
        if(charViews[0].getText()!= ""){
            hacerVisibleBoton(botones_pulsados.get(0));
            botones_pulsados.remove(0);
        }
     //si la primera posición está ocupada aumentamos en 2 las
    //letras.
        else if(charViews[1].getText()!=""){

            letras+=2;
        }
        else{
            letras++;
        }
        charViews[0].setText(""+letra);
       charViews[0].setTextColor(Color.parseColor("#0A733C"));
        ocultarBoton(letra);
        comodin1.setEnabled(false);
    }
}
```

Anexo IV: Comodín que coloca una letra aleatoria en el modo *Encuentra el orden*.

El propósito de este anexo es mostrar el código relativo al método del comodín que coloca una letra aleatoria en la palabra buscada del método *Encuentra el orden*, de manera que facilite la comprensión de la implementación comentada en el apartado

```
public void letraRandom_Comodin(View view){
    nComodin.add(nComodin.size(),2);
    String nomApe=nombreOapellido();
    Random RNG = new Random();
    int i = RNG.nextInt(nomApe.length()-1);
    //para evitar la primera letra

    posLetra=++i;//de 0 a length-1;
    //aumentamos el valor en uno para no
    //obtener la primera letra

    char letra = nomApe.charAt(i);

    if(charViews[i].getText()!="") {
    //por si el usuario ya ha empezado a escribir el nombre

        if (comparaLetraconLetraBotonxID(letra)) {
            hacerVisibleBoton(botones_pulsados.get(i));
            botones_pulsados.remove(i);
        }
        else{
            hacerVisibleBoton(botones_pulsados.get(i-1));
            botones_pulsados.remove(i-1);
        }
    }
    añadirLetraLista(letra);
    charViews[i].setText(""+letra);
    charViews[i].setTextColor(Color.parseColor("#0A733C"));

    ocultarBoton(letra);
    comodinRandom.setEnabled(false);

    if(i==1 && charViews[0].getText()!=""
             && charViews[2].getText()==""){

        letras++;
    }
    else if(i==longitudPalabra-1 && letras == i){
    //si tenemos todas las letras menos la última
        comprobarPalabraCompleta();
        //y el comodín nos la rellena, comprobamos la palabra
    }
}
```

Anexo V: Código del método letterPressed incluido en el *Ahorcado*.

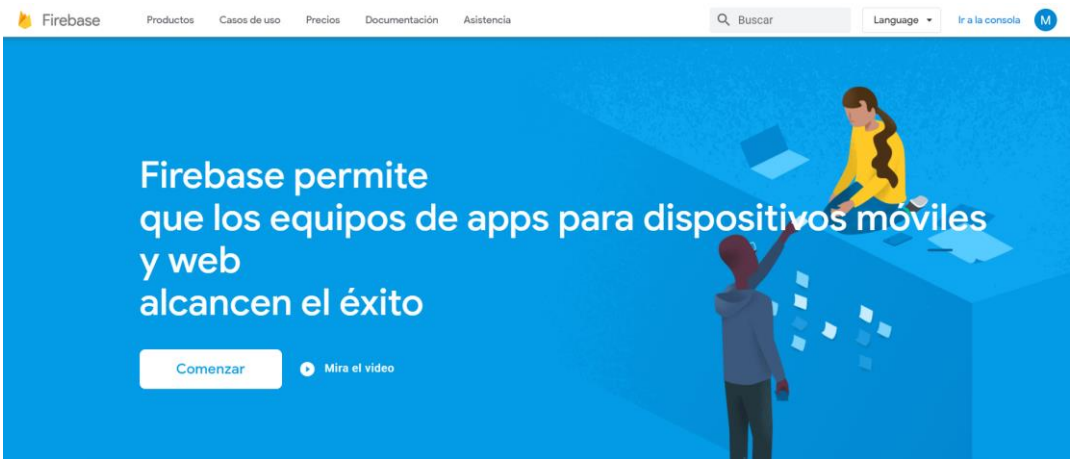
El propósito de este anexo es mostrar el código relativo al método *letterPressed*, de manera que facilite la comprensión de la implementación comentada en el apartado

```
public void letterPressed(View view){
    String letter = ((TextView) view).getText().toString();
    char letterChar = letter.charAt(0);
    Boolean bandera = false;
    view.setEnabled(false);


    for(int i=0; i<nombre.length(); i++){
        if(nombre.charAt(i)==letterChar){
            charViews[i].setTextColor(Color.WHITE);
            bandera = true;
            letras++;
        }
    }
    int pos=nombre.length()+1; //+1 por el espacio en blanco
    for(int i=0; i<apellido.length(); i++){
        if(apellido.charAt(i)==letterChar){
            charViews[pos+i].setTextColor(Color.WHITE);
            bandera = true;
            letras++;
        }
    }
    //cambia el botón por un tick o un cross según sea bandera
    cambiarBackgroundButton(bandera, (Button) view);

    if(bandera) {
        if (letras == tamañoTotal) {
          Toast.makeText(this,"Correcto",Toast.LENGTH_SHORT).show();
           disableButtons();
           preguntas++;
           score+=10;
           showQuestion();
        }
    }
    else if(index < sizeParts){ //va a tener 6 oportunidades.

       parts[index].setVisibility(View.VISIBLE);
       index++;

       if(index==sizeParts) {
          Toast.makeText(this,"Perdiste",Toast.LENGTH_SHORT).show();
           disableButtons();

           preguntas++;

           showQuestion();

       }
    }
}
```

Anexo VI: Creación de un proyecto en Firebase.

El propósito de este anexo es descubrir el proceso de creación de un nuevo proyecto en *Firebase*, lo que nos dará acceso a esta base de datos, que va a ser utilizada durante el desarrollo del proyecto.

Primeramente, introducimos la siguiente dirección en el navegador que tenemos instalado en nuestro equipo: https://firebase.google.com/. En ella, encontramos lo siguiente:



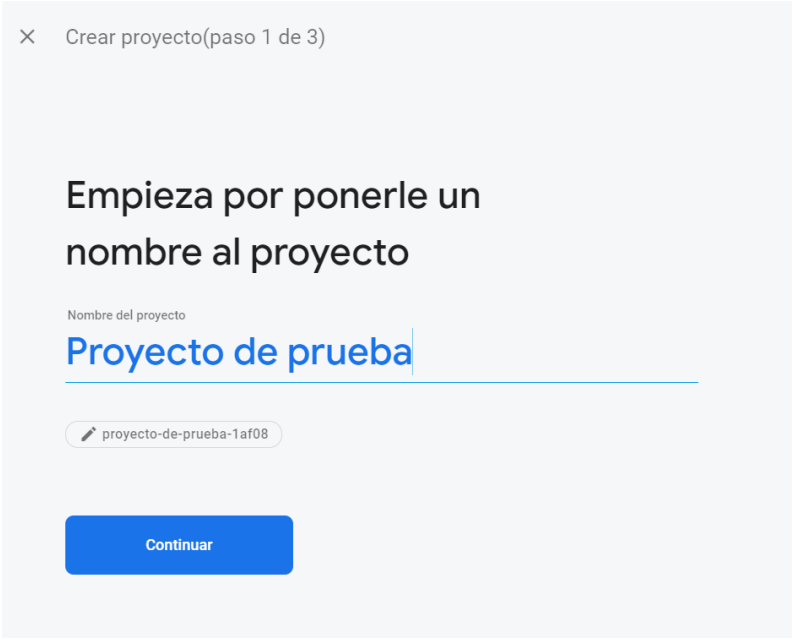
51. Página web Firebase. *Fuente referenciada (Firebase.com, 2020).*

La creación de este proyecto es muy sencilla y está indicada de forma clara en la web, como se puede ver en las respectivas imágenes añadidas en este anexo.

Para empezar con el proceso pulsamos el botón Comenzar. Nos aparecerá la siguiente pantalla, donde podemos ver todos los proyectos que tenemos. Aunque, primeramente, debemos conectarnos con una cuenta de correo de Gmail.

52. Creación de un proyecto en Firebase. *Fuente referenciada (Firebase.com, 2020).*

Se selecciona Añadir proyecto, para comenzar con la creación del mismo.

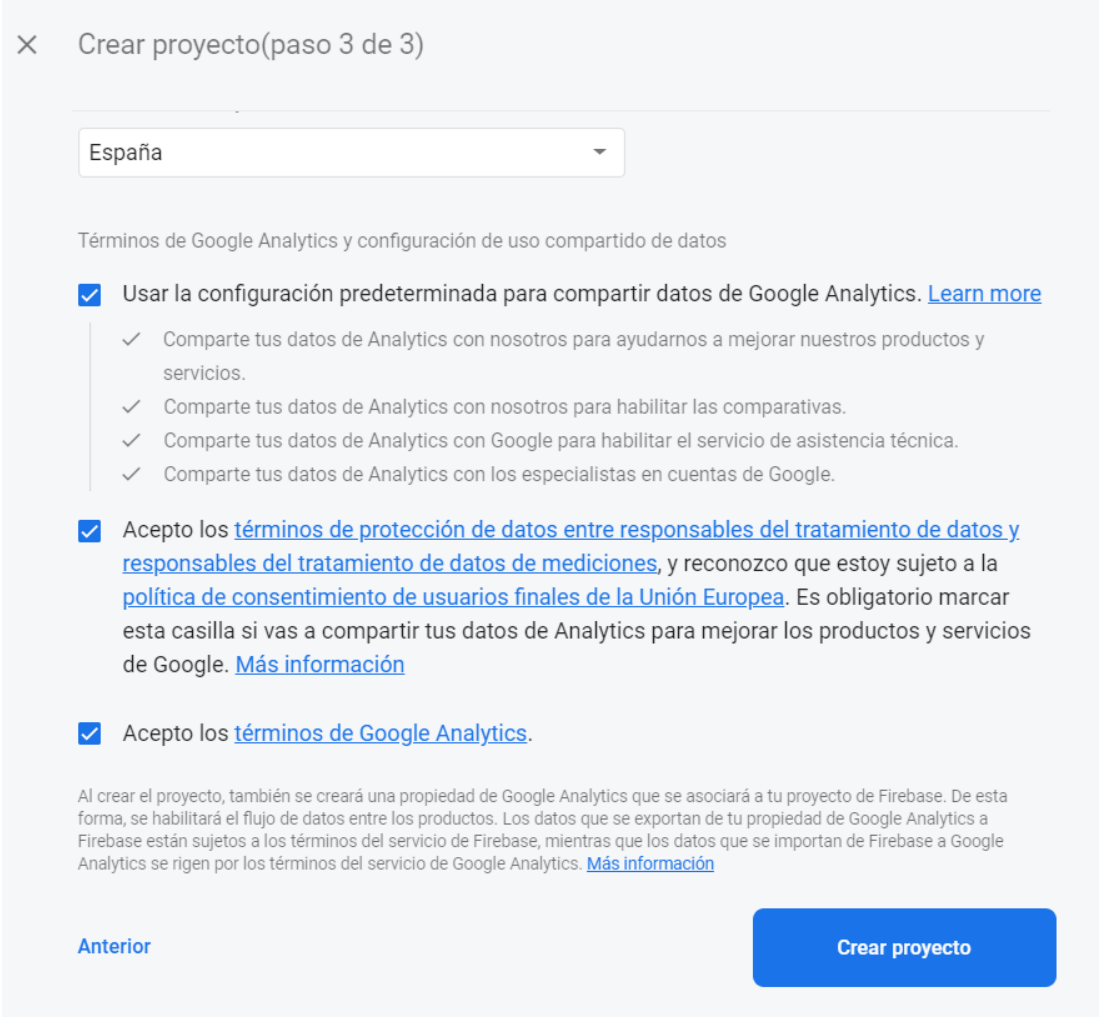


53. Creación proyecto en Firebase paso 1. *Fuente referenciada (Firebase.com, 2020).*

El siguiente paso es seleccionar el nombre del proyecto y pulsando el botón de Continuar.

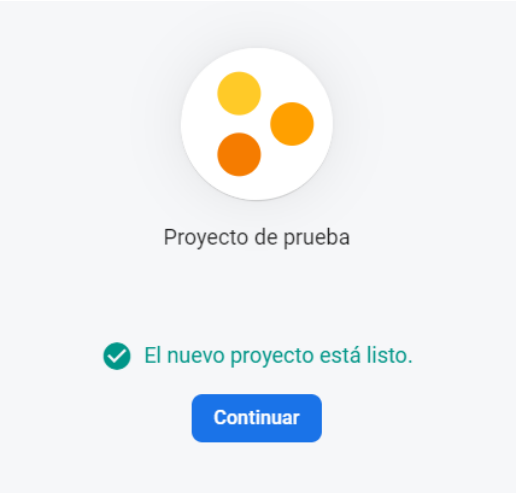
54. Creación proyecto en Firebase paso 2. *Fuente referenciada (Firebase.com, 2020).*

En esta pantalla se indica se quiere habilitar la función de *Google Analytics* que nos ofrece esta misma compañía, en nuestro caso lo vamos a dejar habilitado, y pulsamos el botón azul de Continuar.



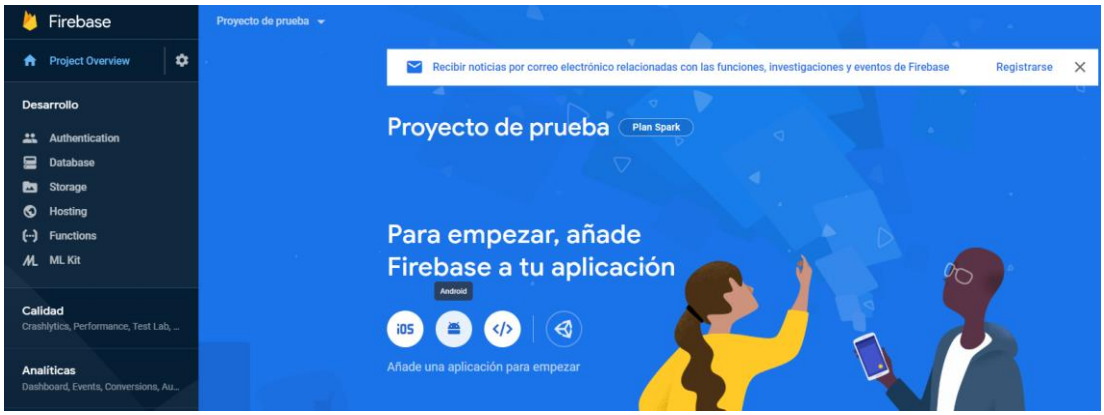
55. Creación proyecto en Firebase paso 3. *Fuente referenciada (Firebase.com, 2020).*

Por último, se selecciona el país donde queremos que este alojado la base de datos y aceptamos los términos y condiciones, una vez han sido leídas, pulsando el botón Crear proyecto para finalizar el proceso. Una vez se ha completado la creación del proyecto aparece la siguiente imagen.



56. Proyecto creado. *Fuente referenciada (Firebase.com, 2020).*

Se pulsa el botón Continuar y accedemos a la consola de administración del proyecto recién creado.



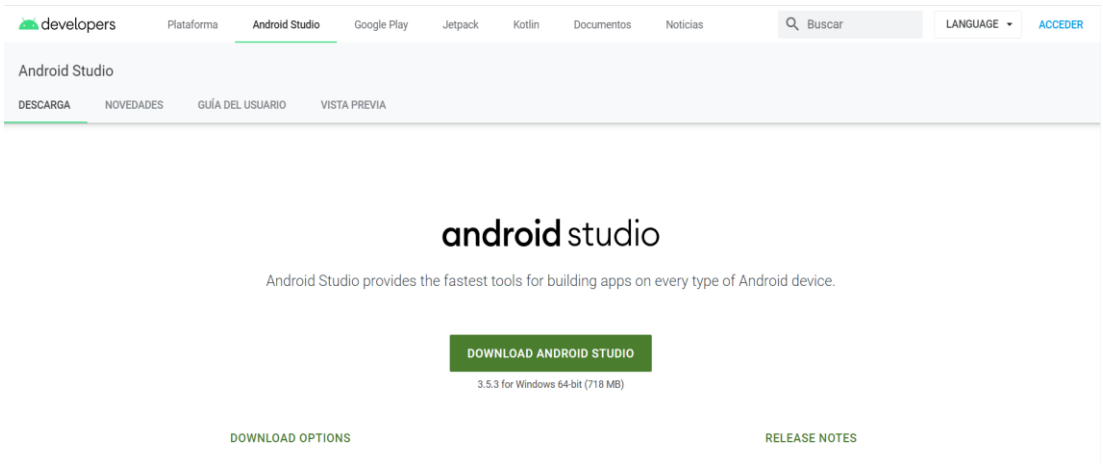
57. Consola de administración de Firebase. *Fuente referenciada (Firebase.com, 2020).*

En este punto, ya se habría creado el proyecto de la base de datos *Firebase* que va a ser incluida en el proyecto. Este proceso se explica en el anexo VIII.

Anexo VII: Instalación de Android Studio.

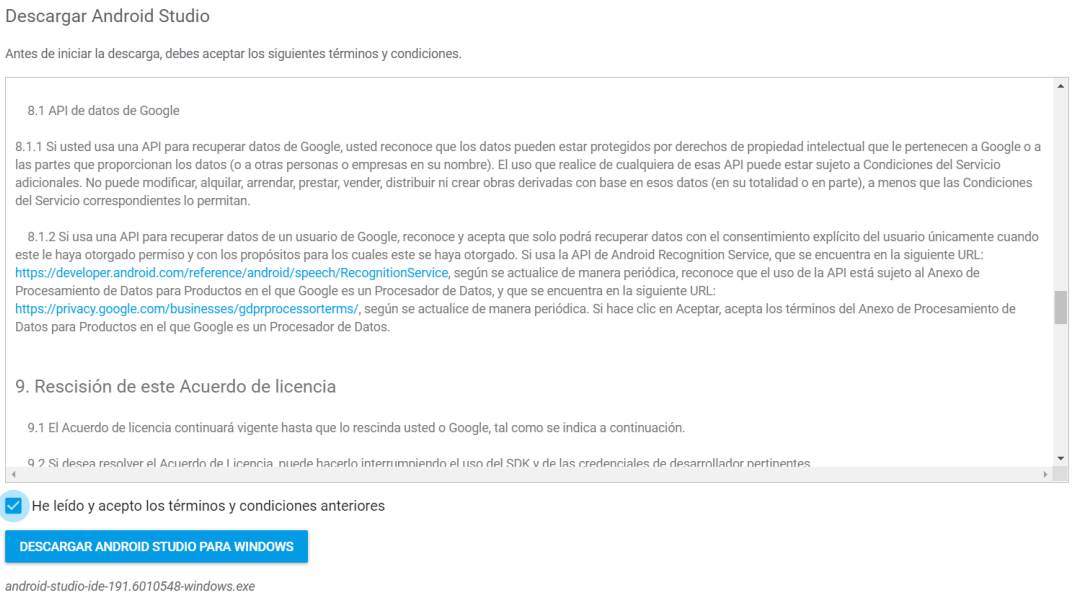
El propósito de este anexo es descubrir de forma minuciosa el proceso de instalación del *framework* seleccionado para realizar el desarrollo de la aplicación *NamEx*, que como se ha indicado con anterioridad es *Android Studio*.

Primeramente, abrimos el navegador que tenemos instalada en nuestro equipo y accedemos a la siguiente dirección url: https://developer.android.com/studio. En ella, nos encontramos lo siguiente:



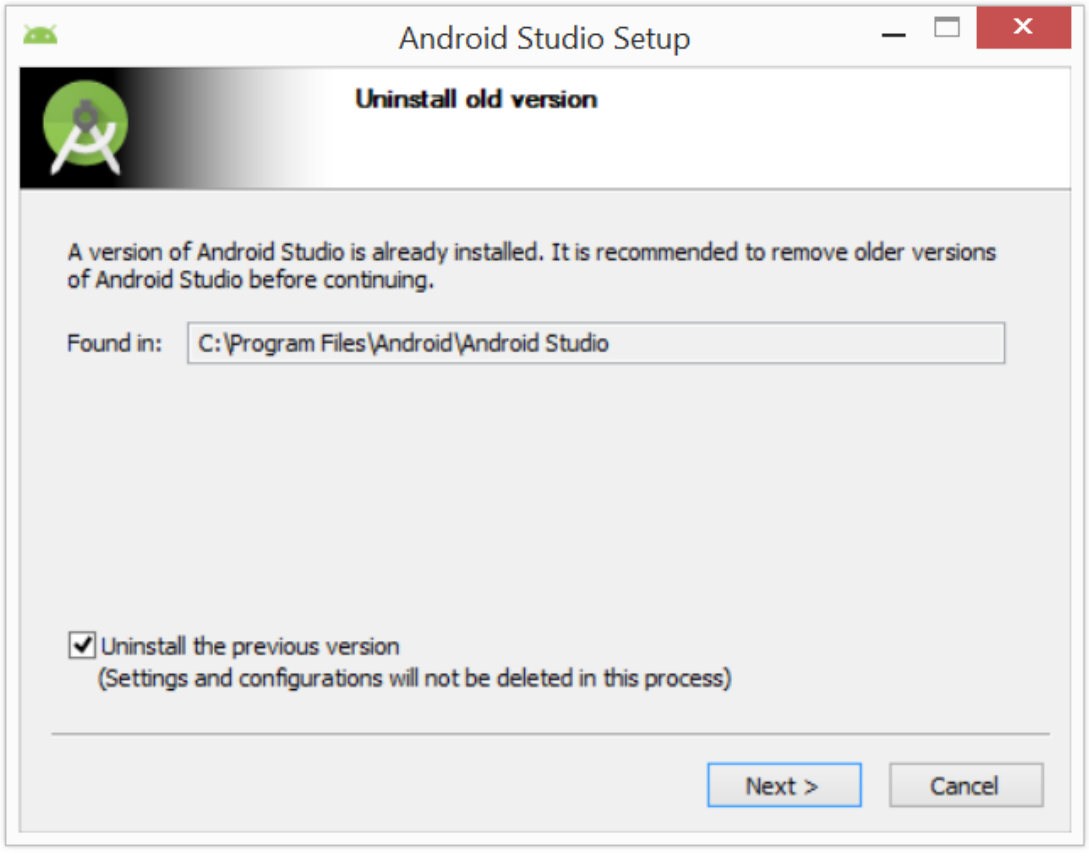
58. Página web de Android. *Fuente referenciada (Developer.android.com, 2020).*

El siguiente paso a realizar es la pulsación del botón verde, que inicia la descarga del archivo de instalación del *framework*. Antes de que esto suceda, hay que aceptar los términos y condiciones, tal y como se puede ver en la siguiente imagen:



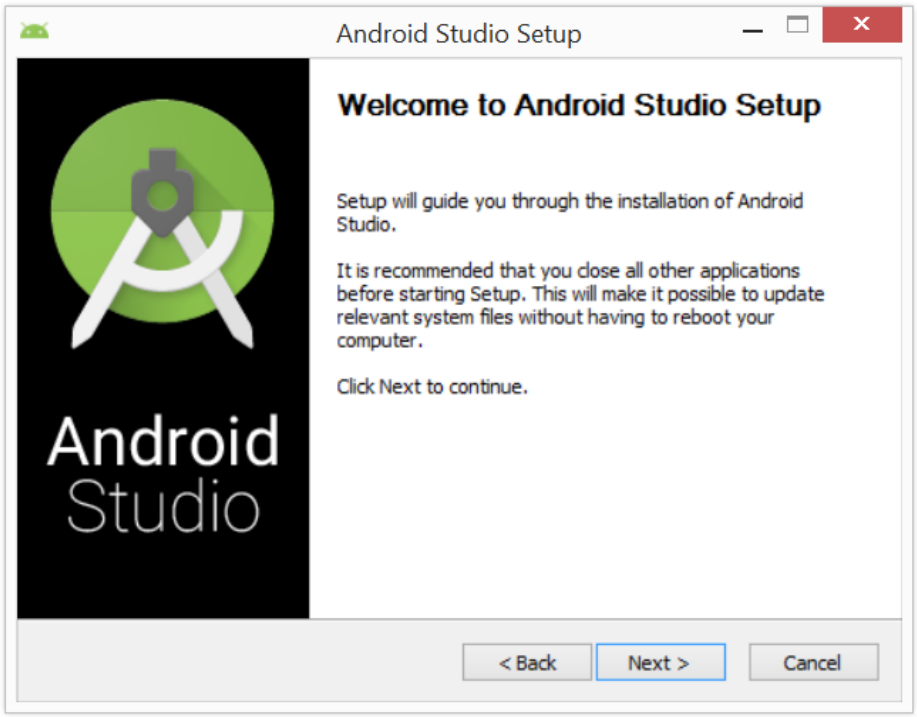
59. Términos y condiciones descarga Android Studio.
Fuente referenciada (Developer.android.com, 2020).

Una vez se ha descargado el ejecutable, pinchamos dos veces encima de él y lo iniciamos.



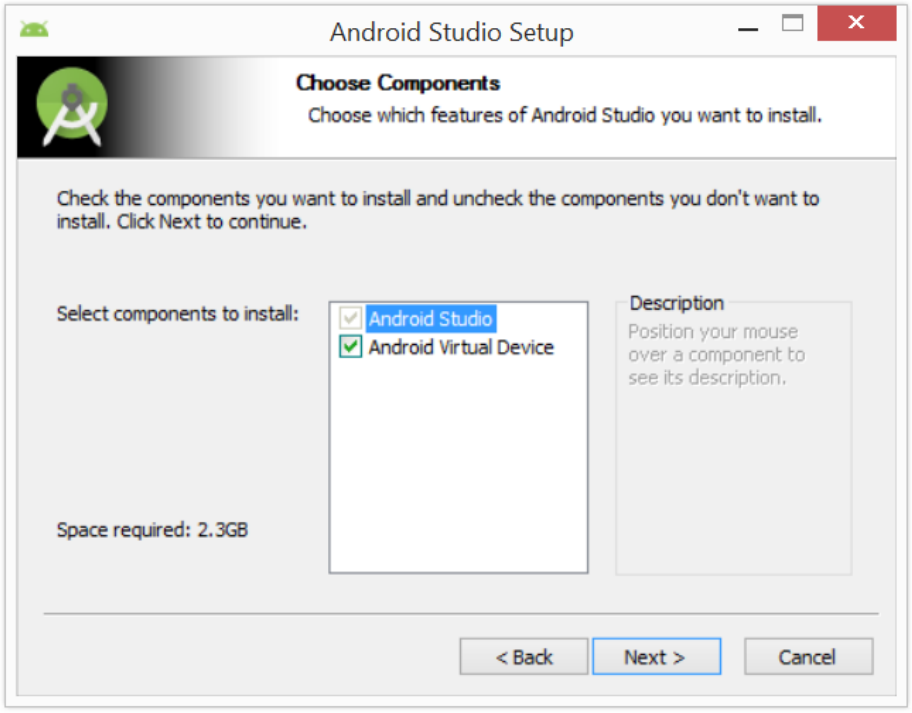
60. Desinstalar versión anterior. *Fuente: propia.*

En este caso, como yo ya tengo instalado una versión de Android Studio, me recomienda que la borre y pulsamos sobre el botón resaltado en azul, Next >.



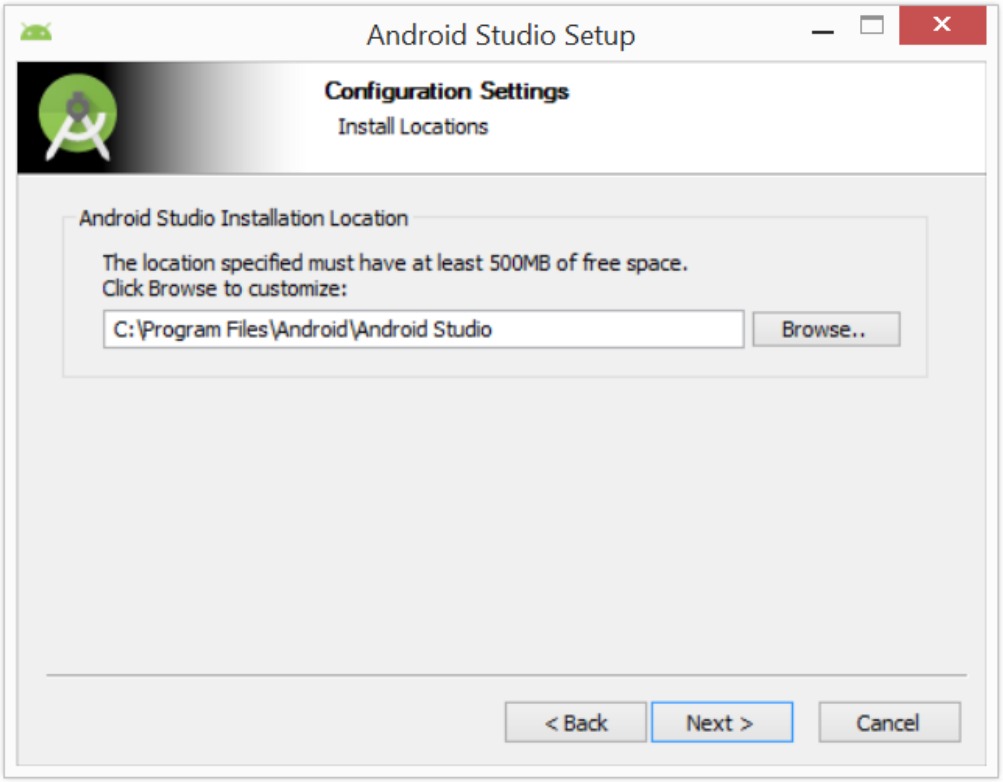
61. Instalación de Android Studio. *Fuente: propia.*

Vamos avanzando en la instalación, siguiendo las recomendaciones que podemos ver en pantalla y pulsamos el botón Next >.



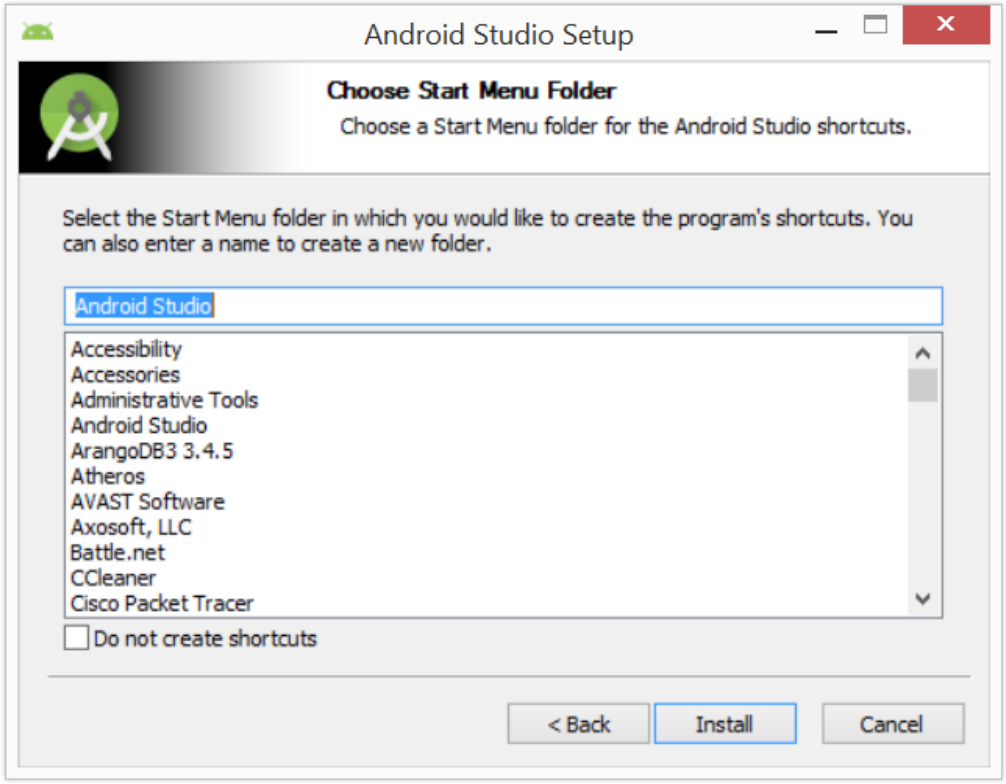
62. Elección de los componentes a instalar. *Fuente: propia*.

El siguiente paso es elegir que componentes instalar, en este caso seleccionamos todos. Android Virtual Device nos permitirá ejecutar un emulador con un dispositivo Android con las características y versión del sistema operativo que seleccionemos, facilitando de esta manera la prueba de la aplicación durante el desarrollo. Pulsamos el botón Next >.



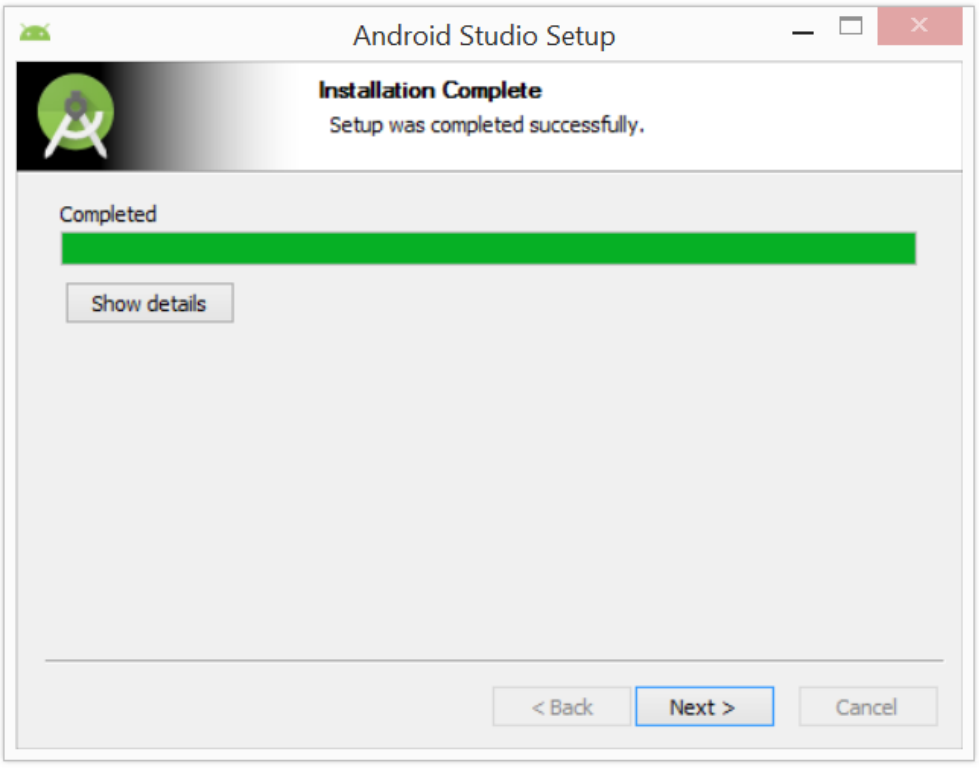
63. Ruta de instalación. *Fuente: propia.*

El siguiente paso es seleccionar la ruta donde va a instalarse Android Studio. Una vez se ha introducido la dirección adecuada se pulsa el botón Next >.



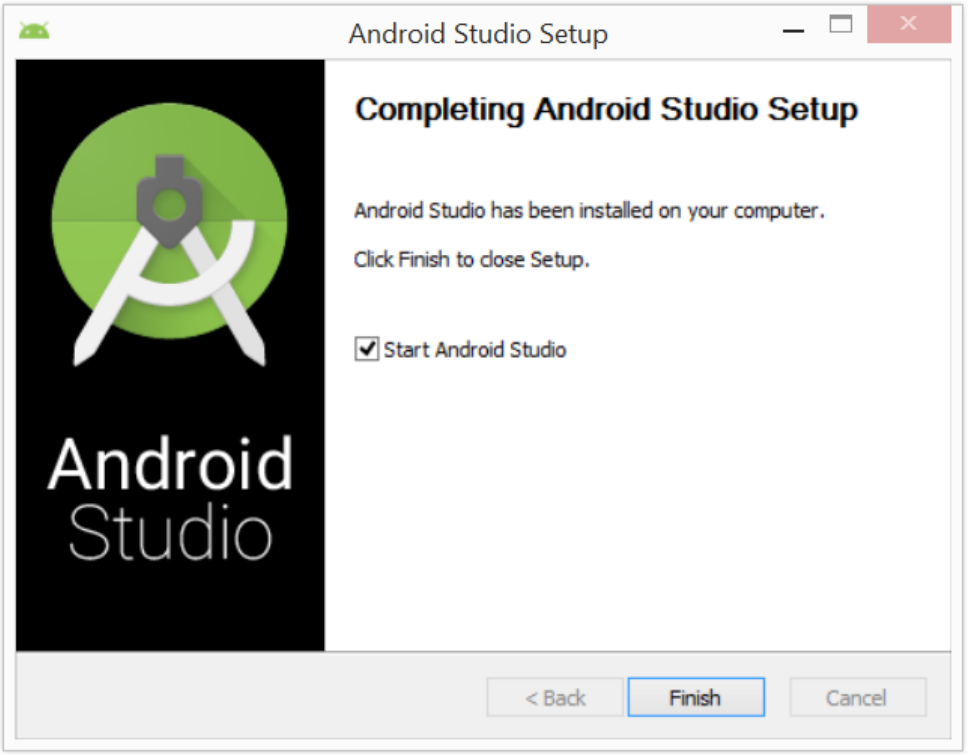
64. Elección de la imagen del atajo en el menú. *Fuente: propia.*

Por último, instalamos la aplicación pulsando el botón resaltado en azul, Install.



65. Instalación completada. *Fuente: propia.*

Una vez se ha completado la instalación, lo cual tardará unos pocos minutos, nos aparece la siguiente pantalla para finalizar la misma, la cual nos permite iniciar Android Studio.



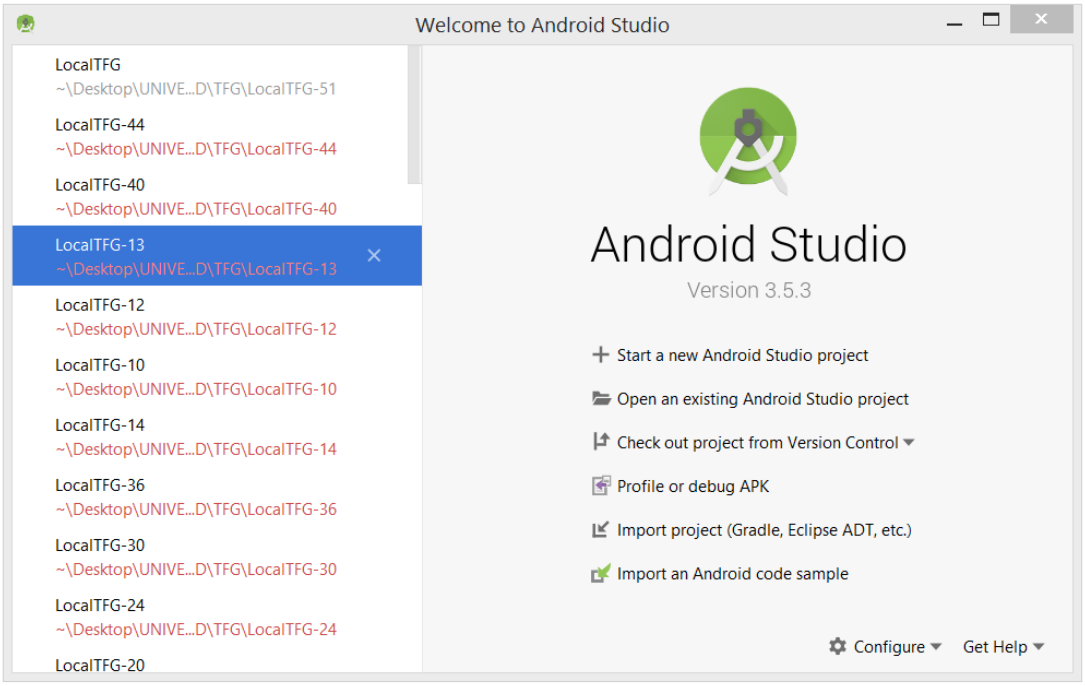
66. Instalación finalizada. *Fuente: propia.*

Una vez tenemos todo listo, falta crear el proyecto sobre el que vamos a empezar el desarrollo y su conexión la base de datos *Firebase*¸ que va a ser explicado en el siguiente anexo.

Anexo VIII: Creación de un proyecto en Android Studio y conexión con el proyecto Firebase anteriormente creado.

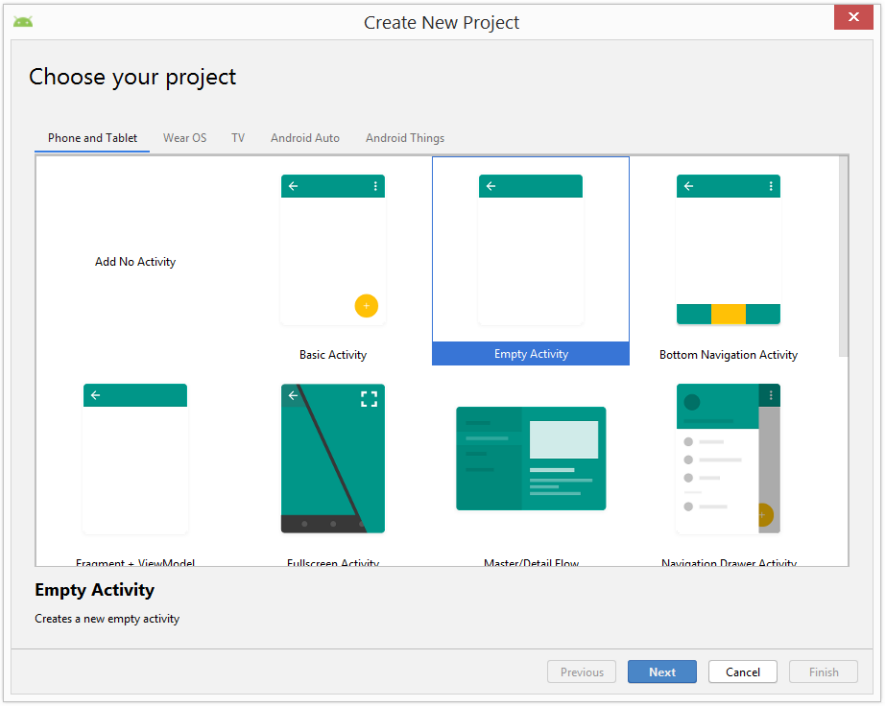
El propósito de este anexo es descubrir el proceso de creación de un proyecto en el *framework* recientemente instalado, *Android Studio*, que permite iniciar el desarrollo, al igual que el seguido para realizar la aplicación *NamEx*.

En primer lugar, se selecciona la opción + Start a new Android Studio project.



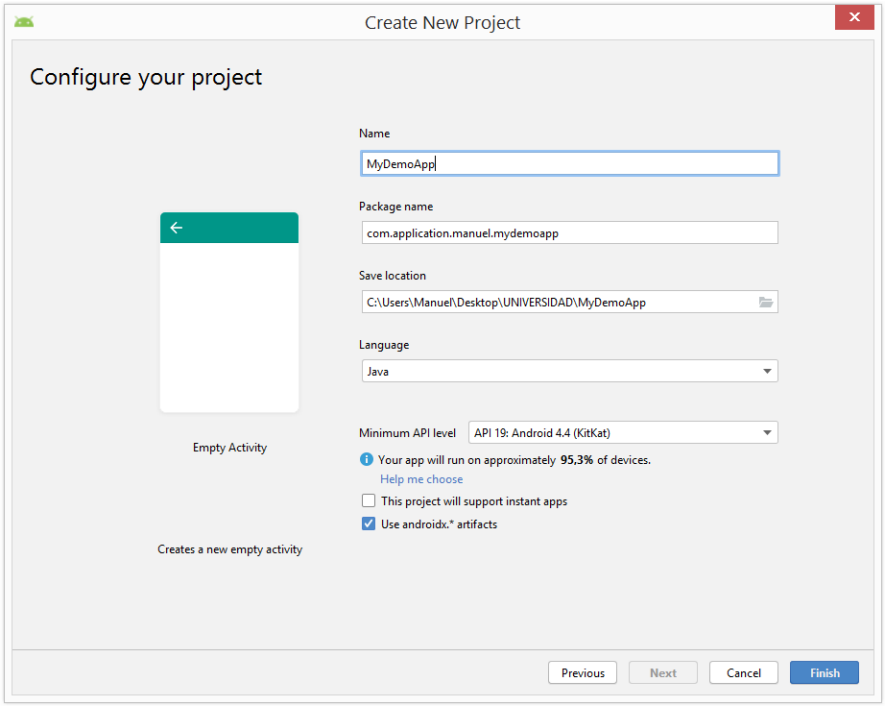
67. Android Studio. *Fuente: propia.*

El siguiente paso es seleccionar el proyecto inicial con el cual se va a empezar. En este caso al ser un proyecto demostración da igual la elección, pero en el caso de que no lo fuera, ha de seleccionarse la opción con la que más interesante resulte comenzar.



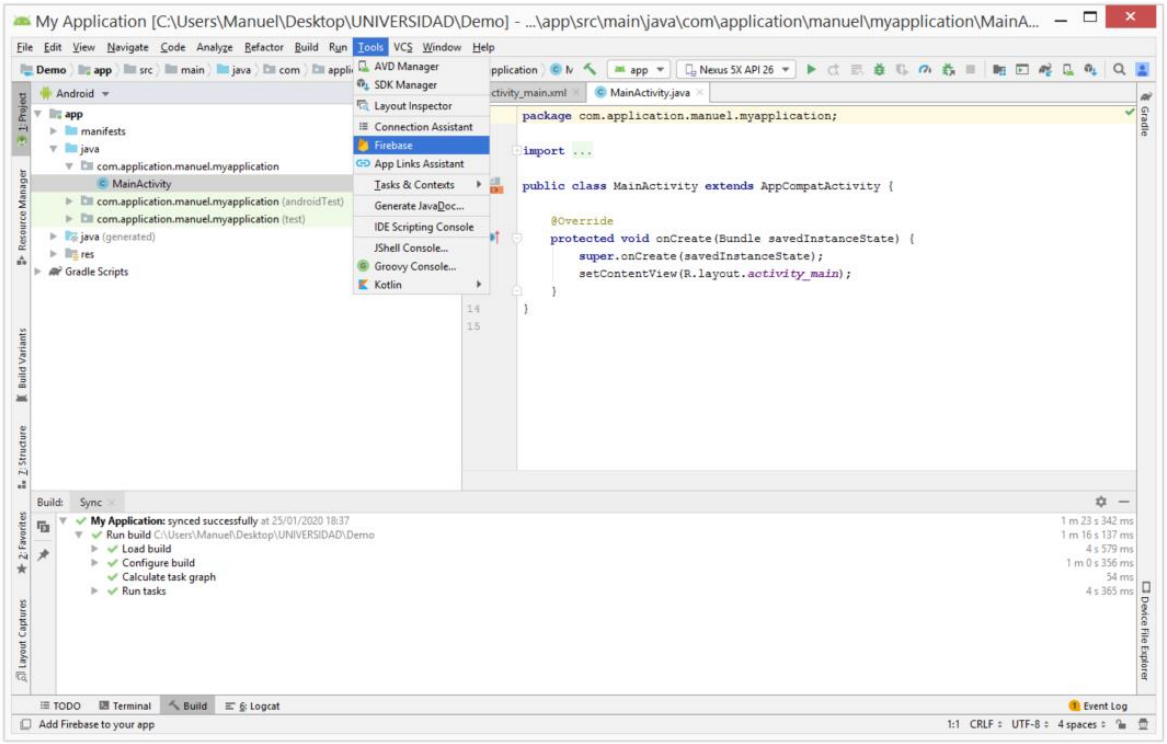
68. Creación nuevo proyecto. *Fuente: propia.*

El siguiente paso es seleccionar el nombre del proyecto, la ruta donde se va a almacenar, el lenguaje que se va a utilizar durante el desarrollo y la versión del sistema operativo mínimo para que funcione la aplicación, como se puede observar en la siguiente imagen:



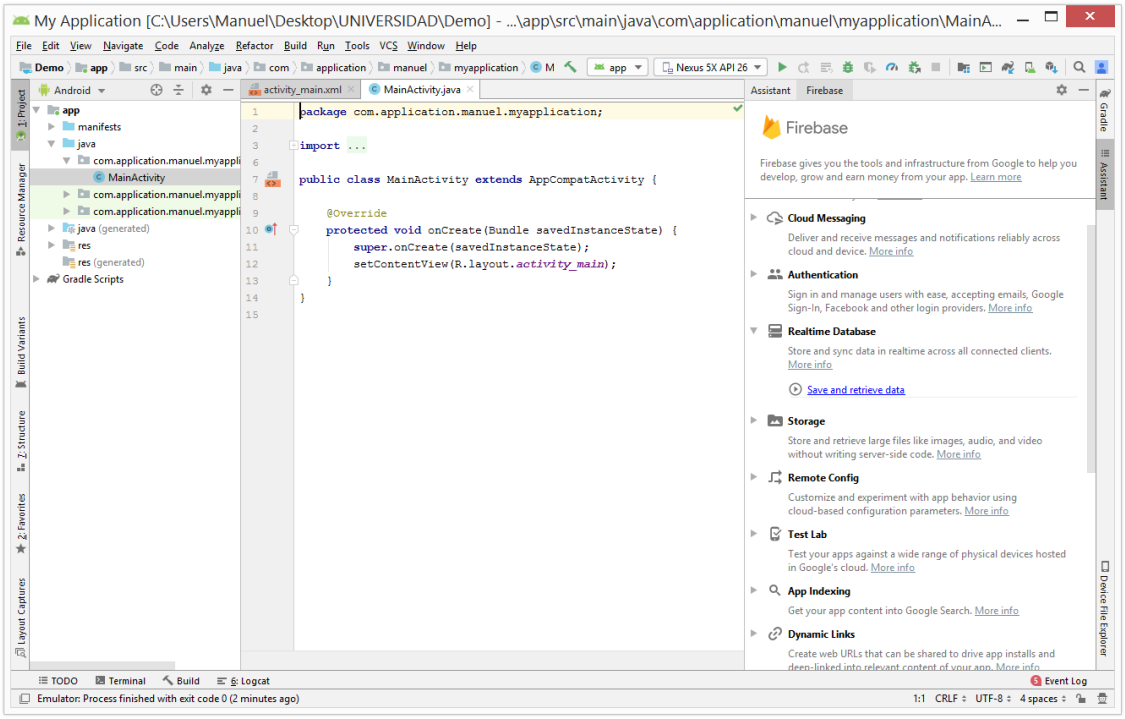
69. Configuración proyecto nuevo. *Fuente: propia.*

Una vez se ha creado el proyecto se va a realizar la conexión con el proyecto de *Firebase* anteriormente creado, de una manera muy sencilla ya que los pasos a realizar nos lo muestra el propio *framework*. Para ello seleccionamos el menú *Tools* y la opción *Firebase*.



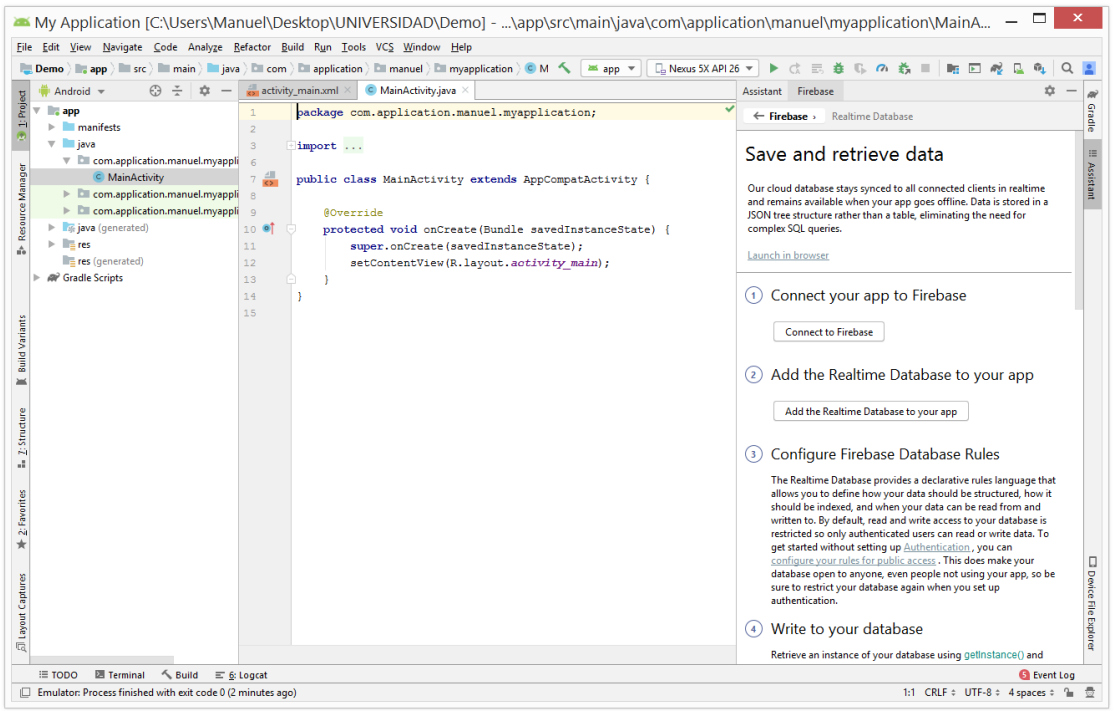
70. Pantalla inicial del proyecto creado. *Fuente: propia.*

Lo siguiente que aparece es un menú lateral de *Firebase* que incluye las características que se pueden incluir en nuestro proyecto. En este caso seleccionamos la que dicta *Realtime Database*, y la opción que aparece *Save and retrive data*, como se puede observar en la siguiente imagen.



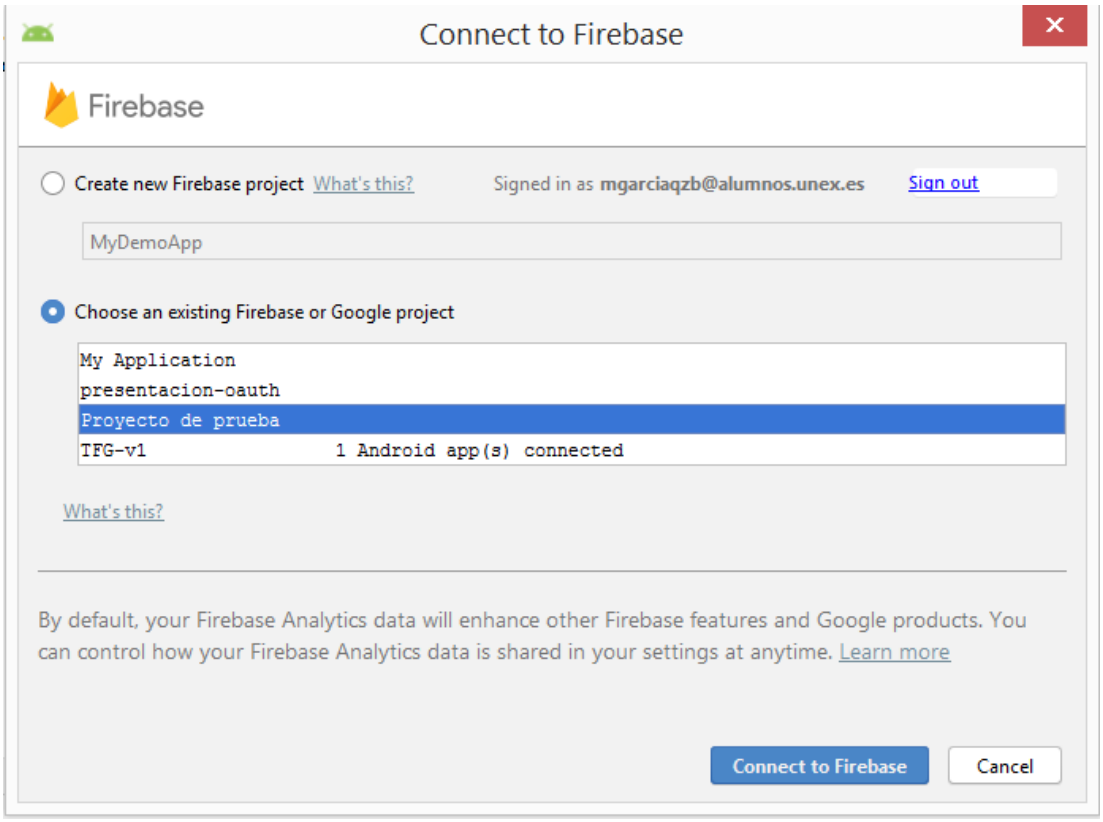
71. Asistente Firebase en Android Studio. *Fuente: propia.*

Una vez se ha seleccionado la opción comentada aparece el siguiente menú:



72. Pasos para guardar y leer datos (1). *Fuente: propia.*

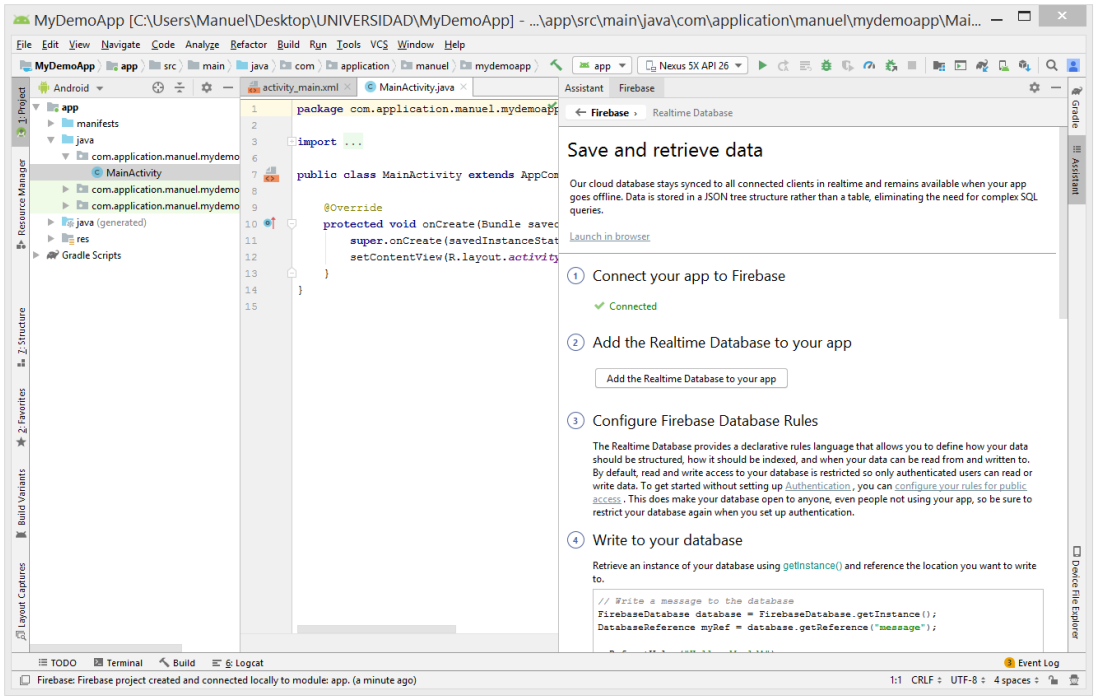
El siguiente paso a realizar es la conexión de la aplicación con la base de datos alojada en la nube *Firebase*, para ello se selecciona el botón *Connect to Firebase*. Una vez seleccionada aparece el siguiente menú, donde se debe seleccionar el nombre del proyecto que se ha creado en el anexo VI.



73. Conexión con Firebase en Android Studio. *Fuente: propia.*

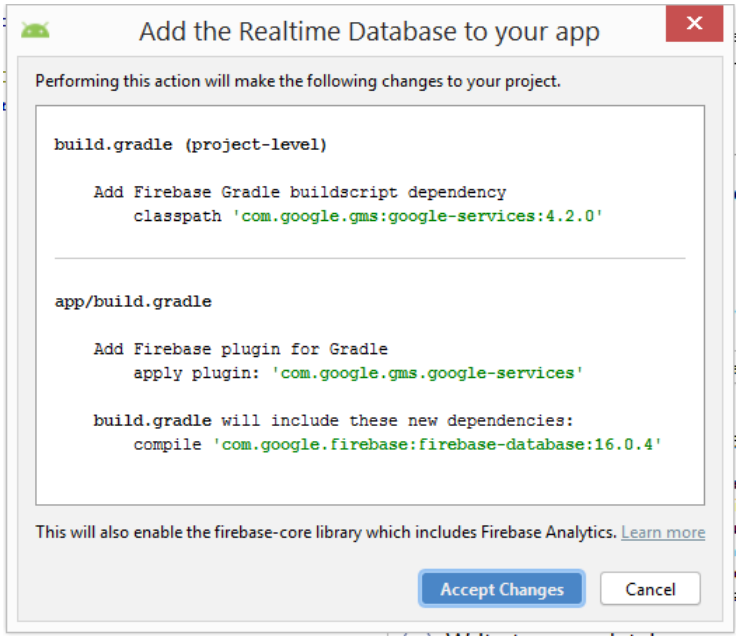
Si no aparece este menú directamente, primero os pedirá conectarse con la cuenta de correo, que debe ser la misma que la que se introdujo en el anexo VI, donde se ha creado el proyecto de *Firebase*.

El siguiente paso a realizar seleccionar el proyecto que se desea utilizar, en nuestro caso recibe el nombre de *Proyecto de prueba*. Una vez seleccionado, se pulsa el botón azul *Connect to Firebase*.



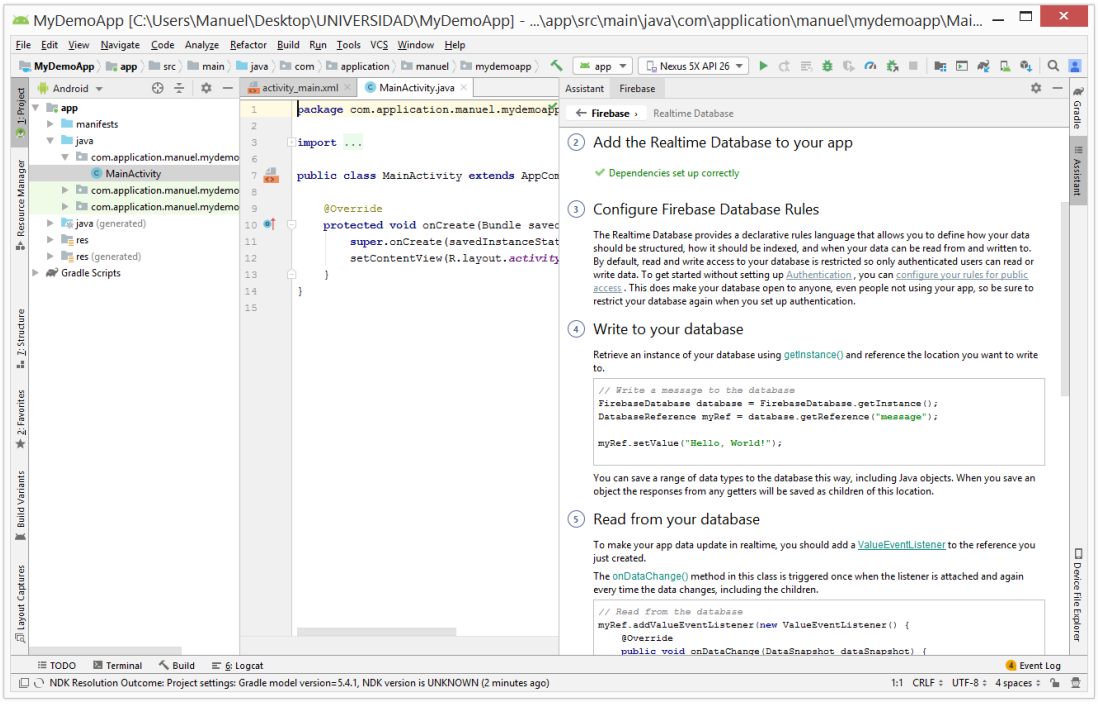
74. Pasos para guardar y leer datos (2). *Fuente: propia.*

Como se puede observar, la conexión se ha realizado correctamente. El siguiente y último paso que se va a realizar se la adición de las dependencias necesarias al proyecto para poder utilizar la base de datos. Para ello se pulsa el botón *Add the Realtime Database to your app*.



75. Dependencais añadidas al Gradle. *Fuente: propia.*

Esta pantalla muestra los cambios que se van a introducir en los archivos *Gradle* del proyecto, al pulsar *Accept Changes*.



76. Pasos para guardar y leer datos (3). *Fuente: propia.*

Por último, se puede observar que las dependencias se han agregado satisfactoriamente y nos muestran los pasos que debemos realizar si queremos escribir datos en la base de datos, leer datos de *Firebase*, además de otras opciones que también permite esta.