UNLPam
Universidad Nacional de La Pampa

FACULTAD DE INGENIERÍA

UNIVERSIDAD NACIONAL DE LA PAMPA

FACULTAD DE INGENIERÍA

PROYECTO FINAL DE GRADO DE INGENIERÍA EN SISTEMAS (PLAN 2017)

Aplicación móvil de Podcast con ejercicios de comprensión para el Área de Inglés de la Facultad de Ingeniería de la UNLPam

Autora: Ivanna F. Jutterpeker
Directora: Dra. María Fernanda Papa
Co-Directora: Mag. Estela Raquel Ramos
Grado Académico: Ingeniería en Sistemas
Fecha de aprobación: 26/06/2023
Jurados:

- Becker, Pablo J. - Facultad de Ingeniería - UNLPam
- Rivera, María B. - Facultad de Ingeniería - UNLPam
- Salto, Carolina - Facultad de Ingeniería - UNLPam

GENERAL PICO - LA PAMPA - 2023

Agradecimientos

En primer lugar, quisiera agradecer a mi familia por su apoyo incondicional a lo largo de estos años de estudio.

Seguidamente, le quiero agradecer a mi tutora la Dra. Fernanda Papa por su predisposición y dedicación, ya que sin sus consejos y correcciones no podría haber llegado a esta instancia tan anhelada. También, le quiero dar las gracias a mi co-tutora Mag. Raquel Ramos y a las profesoras del área de inglés por el apoyo brindado en la producción de este trabajo final.

Por último pero no menos importante, le quiero agradecer a todo el personal, docentes y no docentes y a la Universidad Nacional de La Pampa que han sido parte de mi camino universitario y me formaron como profesional.

Resumen

En este documento de tesis se describe el proceso de desarrollo de la aplicación móvil ***Podcast.ing***, bajo la metodología ágil denominada Scrumban. ***Podcast.ing*** tiene como finalidad fortalecer la enseñanza y el aprendizaje de las macro-habilidades en las asignaturas de inglés de la Facultad de Ingeniería de la UNLPam, adaptando sus características y funcionalidad a las necesidades de las profesoras de dicha área. Las macro-habilidades que se potencian en la aplicación son aquellas que los alumnos identificaron que se deben reforzar a partir de encuestas realizadas.

La idea de ***Podcast.ing*** es, por un lado, brindar a los estudiantes una lista de podcast con su correspondiente ejercitación. Y por el otro, que las docentes tengan visibilidad de los resultados obtenidos por los estudiantes y la posibilidad de incluir, editar y eliminar en la aplicación podcast con ejercicios.

En este trabajo adicionalmente se documenta una revisión de las aplicaciones móviles similares a ***Podcast.ing*** que existen actualmente en el mercado, de algunas de las metodologías ágiles más utilizadas, las herramientas existentes para el desarrollo, y las tecnologías posibles a utilizar. También se detallan todos los procesos por los que se transitó para obtener el producto final según la metodología elegida.

Palabras claves: Aplicación móvil, Flutter, Firebase, Scrumban.

Abstract

This document describes the process development of the mobile application ***Podcast.ing***, based on the methodology called Scrumban. ***Podcast.ing***'s objective is strengthen the teaching and learning of macro-skills in English subjects at the School of Engineering, UNLPam, adapting its characteristics and functionality to the needs of teachers in that area. The macro-skills worked on in the mobile application are those that the students identified as needing to be reinforced based on surveys carried out.

The idea of ***Podcast.ing*** is on the students' side, to provide a list of podcasts with their corresponding exercise. And on the teachers' side, having visibility of the results obtained by the students and also the possibility of including, editing and deleting exercises in the podcast application.

Also, we document a review of mobile applications similar to ***Podcast.ing*** that currently exist on the market, some of the most implemented agile methodologies, existing tools for development, and possible technologies to use. It also details all the processes that went through to obtain the final product, according to the chosen methodology.

Key Words: Mobile app, Flutter, Firebase, Scrumban.

Tabla de contenido

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

En el presente capítulo se brinda información sobre las aplicaciones móviles en la actualidad y su evolución a lo largo de los años. Además, se describe el estado actual de las aplicaciones móviles relacionadas con ejercitación en inglés dentro del mercado, el problema que dio origen a este trabajo y la solución encontrada. Para ello, se definieron las dificultades con las que se enfrenta el equipo docente de inglés, se esbozó una solución, se plantearon los objetivos, para finalizar con un breve resumen de lo que se va a desarrollar dentro de los demás capítulos.

1.1 Aplicaciones móviles

Una aplicación móvil se define como una aplicación de software diseñada para ejecutarse en dispositivos móviles como teléfonos inteligentes y tabletas. A modo de contextualizar el avance de estas aplicaciones, se realiza un breve recorrido por su historia [1]:

- En el año 1994, IBM (International Business Machines) lanza el primer teléfono inteligente con aplicaciones sencillas como el calendario, reloj, calculadora, etc.
- En 1997, los teléfonos Nokia salen a la venta con el juego “Snake” (Viborita) precargado.
- En marzo del 2002, se lanza Blackberry 5810 con el concepto innovador de wireless (inalámbrico) Email. Además de tener precargado las aplicaciones del editor de ringtone, juego Arcade, etc.
- En el año 2007, surge el primer iPhone que incluye aplicaciones referidas al clima y mapas.
- En 2008, se crea el primer teléfono inteligente con Android, y junto a él, el App Store, un repositorio con 500 aplicaciones disponibles para su descarga. Luego de 3 días de su lanzamiento, ya se habían realizado 10 millones de descargas. Además, ese mismo año, se lanza Android Market.
- En el año 2009, App Store sobrepasa un billón de descargas y Facebook es la aplicación más descargada.
- Durante agosto del 2010, Android Market llega a un billón de descargas.
- En el año 2012, se cambia la marca de Android Market a Google Play.

A partir del 2012, la cantidad de descargas de aplicaciones -tanto de App Store como de Google Play- se incrementa a gran velocidad como consecuencia del avance del uso de los teléfonos inteligentes.

A noviembre de 2022 había más de 6 millones de usuarios de teléfonos inteligentes en todo el mundo [2], siendo más del doble de lo que había en el 2016. Y se prevé que para el año 2027 el número se incremente a más de 7.690 millones. Este aumento de usuarios de teléfonos también viene acompañado de un crecimiento de descargas de aplicaciones móviles. En 2021 los usuarios descargaron 230.000 millones de aplicaciones móviles en sus dispositivos. Esto representa más de un 63% de las descargas realizadas en el año del 2016.

Dentro del mercado, solamente en el año 2021, se lanzaron 2 millones de aplicaciones nuevas. Esta ampliación de la cantidad de aplicaciones disponible, también llevó a que los consumidores de las mismas pasen 4,2 horas al día utilizándolas. Siendo las aplicaciones sociales, de fotos/videos y los juegos las más usadas.

En base a todos los datos mencionados se puede concluir que las aplicaciones móviles son un mercado que se encuentra en creciente auge y se espera que continúe con dicha tendencia para los próximos años, debido al incremento de personas que poseen teléfonos inteligentes. Lo cual hace que una aplicación móvil para mejorar las macro-habilidades del lenguaje inglés sea una propuesta motivadora para los estudiantes universitarios de la Facultad de Ingeniería (F.I.) de la Universidad Nacional de La Pampa (UNLPam).

1.2 Planteamiento del problema

Dentro de la F.I. de la UNLPam, se dictan cursos de inglés curriculares y extra-curriculares. Los alumnos pueden presentarse a los exámenes de acreditación sin cursar la materia, en caso de que tengan los conocimientos necesarios para hacerlo. Se deben acreditar 2 o 3 niveles de inglés según el plan de estudios que exige correlatividad con otras materias de las carreras. Los niveles de inglés que se brindan dentro de la F. I. son:

- **Inglés I:** es el primer nivel obligatorio a acreditar. Consiste en el desarrollo de cuatro macro-habilidades (escuchar, hablar, leer y escribir) en el contexto de temas relacionados con la vida académica y laboral. Equivale a un nivel elemental correspondiente al nivel A1/A2 según el Marco Común Europeo de las Lenguas (en adelante CEFR).
- **Inglés II:** es el segundo nivel obligatorio a acreditar y consiste en que los estudiantes puedan desarrollar las cuatro macro-habilidades para comunicarse en situaciones del ámbito académico, laboral y social, en un nivel Pre-intermedio, correspondiente al nivel A2/B1 del CEFR.
- **Inglés III:** es el tercer nivel que se dicta, es opcional para algunas carreras y obligatorio de acreditar en los nuevos planes de estudio de Analista Programador (plan de estudio 2019) e Ingeniería Biomédica (plan de estudio 2019). El objetivo principal es continuar el desarrollo de las cuatro macro-habilidades de la lengua para su desempeño en el mundo académico y laboral. Además, los estudiantes adquieren herramientas para poder realizar presentaciones y construir sus CVs. Se trabaja en el marco de un nivel B1 del CEFR.

Además, se ofrecen cursos extra-curriculares:

- **Inglés Prep:** es un curso que consiste en preparar a los estudiantes que poseen un nivel de inglés que no alcanza para poder integrarse de manera sencilla al curso de inglés I. En Inglés Prep se ofrecen las bases de inglés. Para determinar si un estudiante debería asistir al curso, se realiza una evaluación diagnóstica y de acuerdo a su resultado se le recomienda o no hacerlo.
- **Inglés III:** detallado arriba. Para las carreras de Ingeniería en Sistemas (plan de estudio 2017), Analista Programador (plan de estudio 2015), Ingeniería en Computación (plan de estudio 2015), Ingeniería Industrial (plan de estudio 2017), Ingeniería Electromecánica (plan de estudio 2014) e Ingeniería Electromecánica con Orientación en Automatización Industrial (plan de estudio 2014) es un curso extracurricular.

De esta forma, los cursos brindan a los estudiantes de las diferentes carreras la ejercitación de las cuatro macro-habilidades en situaciones del ámbito académico, laboral y social:

- **Comprensión oral (Listening):** consiste en que el alumno pueda comprender y decodificar el mensaje que le es transmitido de manera oral. Con esta información se encarga de construir significados para responder acertadamente al interlocutor y para evitar la tergiversación de los mensajes. El listening es la primera habilidad que adquiere el ser humano, y es de gran importancia porque gracias a ella se puede desarrollar la habilidad del habla de forma correcta.
- **Expresión oral (Speaking):** se considera que es un proceso que tiene dos direcciones, ya que en ellas están involucrados el hablante y el oyente. Al permitir la interacción, hay un cierto nivel de relación entre habilidad del habla y de la comprensión auditiva. Para desarrollar la expresión oral en inglés se necesita del desarrollo de ciertos indicadores como la pronunciación, la entonación y la fluidez.
- **Comprensión lectora (Reading):** consiste en que el lector pueda interactuar con el texto escrito y que, al mismo tiempo, de forma activa, logre decodificar el mensaje para entenderlo hasta el punto de poder hacer valoraciones críticas sobre este.
- **Expresión escrita (Writing):** consiste en la capacidad de poder realizar producciones propias basándose en alguna temática para la cual se deben dominar conocimientos como la gramática, la ortografía y el vocabulario.

De las cuatro macro-habilidades, la comprensión auditiva y la expresión oral son las que mayor dificultad presentan. A continuación, se muestra la información obtenida a partir de encuestas realizadas por los docentes a los estudiantes de los diferentes niveles de inglés. En la Figura 1 se grafica la respuesta de los estudiantes de inglés II de la cohorte 2021 y 2022 ante la pregunta ¿Cuáles macro-habilidades les gustaría reforzar?. Se debe tener en consideración que los porcentajes que acompañan a las barras de los gráficos están en base a la cantidad de alumnos que han respondido la encuesta. Esto es 53 estudiantes para el año 2021 y 35 para el año 2022.

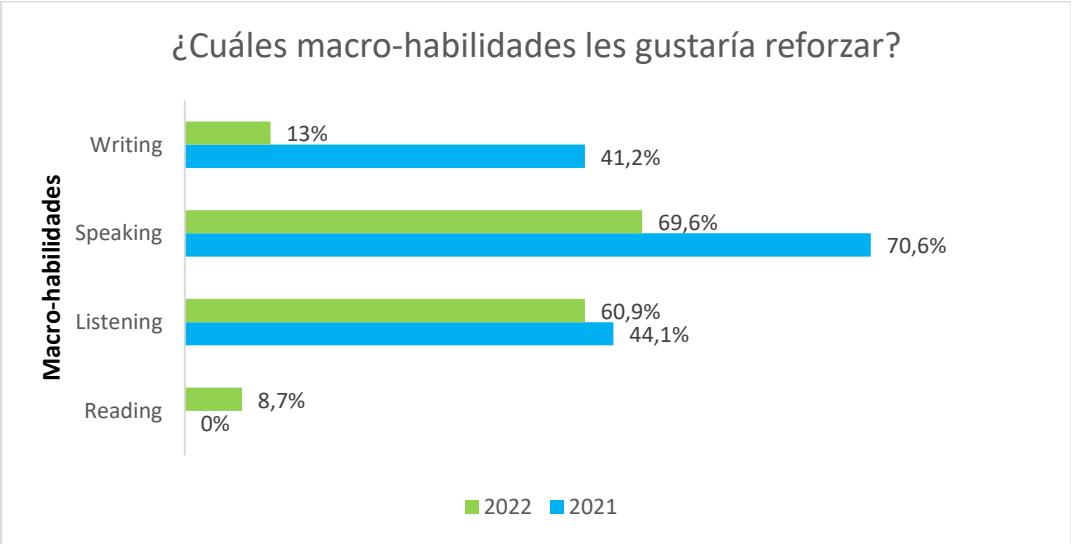


Figura 1: Encuesta de estudiantes de inglés II.

Como se puede observar en la Figura 1 alrededor del 70% de los estudiantes refieren que reforzarían la habilidad de Speaking y un poco menos la habilidad de Listening. A continuación, en la Figura 2 se muestra la respuesta de los estudiantes de inglés I ante la pregunta: ¿Cuáles son tus fortalezas en inglés?. Este gráfico es el resultado

de las encuestas del año 2022 y confirma lo mencionado anteriormente. Los alumnos manifiestan que lo que necesitarían reforzar, tanto en el contexto de que quieren mejorar como también lo que consideran como un punto de debilidad, son las macro-habilidades de Listening y de Speaking.

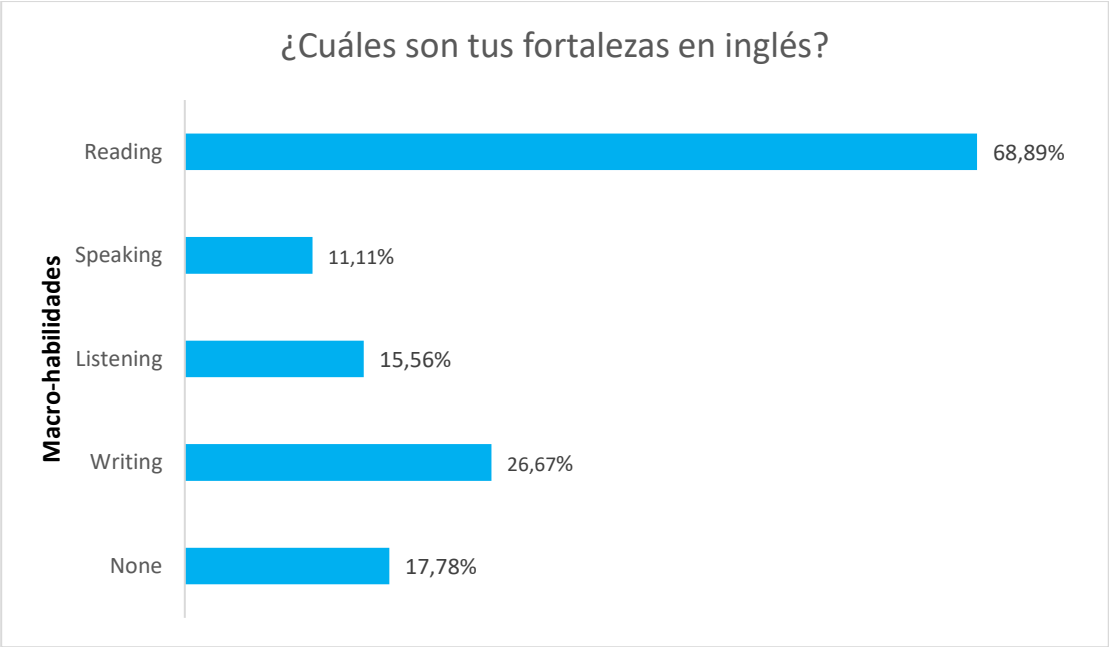


Figura 2: Encuesta de estudiantes de inglés I del año 2022.

En este contexto, las docentes a cargo del área de inglés han implementado diferentes métodos y herramientas para lograr la enseñanza de los contenidos de sus cursos de la manera más didáctica. Habitualmente integran variadas estrategias didácticas y herramientas de Tecnologías de la Información y la Comunicación (TIC) para lograr los objetivos propuestos. Recientemente, con el objetivo de aprovechar el uso de dispositivos móviles para el aprendizaje y ayudar a los estudiantes en el entrenamiento de las dos macro-habilidades mencionadas, se les ocurrió la idea de utilizar una aplicación que sea instalada en los celulares y que los estudiantes puedan manipular todo el tiempo que deseen. En consecuencia, este trabajo de tesis toma la necesidad detectada por el equipo docente del área de inglés, se analizan alternativas de aplicaciones existentes y finalmente se propone desarrollar una aplicación móvil que ayudará a los estudiantes de inglés en la mejora de la comprensión y la expresión oral. Este trabajo busca también fortalecer la enseñanza y el aprendizaje, ofreciendo a los docentes una herramienta propia que se adapte a sus necesidades.

1.3 Estado del arte

La primera actividad que se llevó a cabo una vez planteado el problema a resolver, fue la investigación de herramientas que puedan dar solución a la problemática. El resultado fue que dentro del mercado de aplicaciones móviles existe una gran cantidad de ofertas disponibles para los consumidores. A continuación, se realiza una breve explicación de algunas de las mejores aplicaciones que existen actualmente en el mercado para aprender inglés según Oxford [3]:

1. **Duolingo:** es una aplicación para principiantes donde se puede aprender inglés participando en juegos cortos. En función de diferentes temas, puede aprenderse

hasta siete palabras nuevas por tema y ganar puntos de habilidad después de completar las lecciones.

2. **Babbel:** es una aplicación para adquirir habilidades básicas de conversación. El enfoque de Babbel es el vocabulario, para poder comprender mejor el idioma. Las lecciones se dividen en temas del mundo real.
3. **FluentU:** es una aplicación que utiliza videos atractivos del mundo real; como comerciales y noticias, para convertirlos en experiencias de aprendizaje de inglés. Permite comenzar una conversación de inmediato sobre temas cotidianos o más significativos.
4. **Rosetta Stone:** es una aplicación que no le da ninguna traducción en el idioma propio del estudiante. Este es un método que incentiva a estar inmerso en el idioma y para avanzar hay que ir aprendiendo el contenido. La aplicación incluye ejercicios para aprender las palabras básicas, antes de pasar a formar frases y oraciones.

En la Tabla 1 se aprecia una comparativa de las aplicaciones mencionadas (se puede acceder a las páginas oficiales haciendo clic en los nombres de las aplicaciones presentes en la Tabla 1).

Características	Duolingo	Babbel	FluentU	Rosetta Stone
Macro-habilidades	• Comprensión lectora • Expresión escrita • Comprensión oral	• Comprensión lectora • Expresión escrita • Expresión oral • Comprensión oral	• Comprensión lectora • Expresión escrita • Comprensión oral	• Comprensión lectora • Expresión escrita • Expresión oral • Comprensión oral
Nivel del usuario	Principiante	Principiante	Principiante	Principiante
Características	• Curso completo de inglés con cientos de lecciones. • Actividades de aprendizaje interactivo.	• Práctica todas las habilidades de comunicación en un solo curso. • Las lecciones son interactivas y enseñan temas útiles. • Lingüistas profesionales diseñaron el curso.	• Gran biblioteca de videos en inglés. • Tiene subtítulos y traducciones que ayudan al aprendizaje. • Crea flashcards para practicar nuevo vocabulario.	• Las lecciones inmersivas ayudan a aprender rápidamente. • La tecnología precisa de reconocimiento de voz mejora la pronunciación. • Las lecciones son cortas y fáciles de seguir.
Precio actualizado al 25/11/22	Es gratuito, pero para evitar anuncios se debe abonar 12,99 USD/mes.	7,45 USD/mes.	29,99 USD/mes.	11,99 USD/mes.

Tabla 1: Comparativa entre aplicaciones móviles de enseñanza de lenguajes

Como se observa en la Tabla 1, en la actualidad existen diversas aplicaciones que permiten desarrollar las macro-habilidades del lenguaje inglés. Analizando la información plasmada, se concluye que todas aplican técnicas diferentes de enseñanza. Una desventaja a destacar es que las que cubren tres o cuatro macro-habilidades de aprendizaje son pagas con precios mensuales (fecha consultada 25/11/22) que van desde 7,45 USD a 29.99 USD por mes. Y la aplicación Duolingo, que si bien es gratuita presenta anuncios que genera interrupciones en el proceso de aprendizaje. Muchas veces el corte continuo hace que los usuarios se desanimen o molesten cuando quieren avanzar en el aprendizaje.

En base a la información obtenida, se concluyó que sería de gran utilidad que se cree una aplicación móvil que ayude a potenciar las macro-habilidades que se quiere mejorar en los alumnos, ajustándose a los contenidos curriculares impartidos por las docentes del área de inglés de la F.I. Todo esto tiene como finalidad brindar una herramienta a los estudiantes que les facilite mejorar la habilidad de comprensión auditiva y expresión oral. A su vez, las profesoras se beneficiarán con el conocimiento registrado por la aplicación del avance de sus alumnos en esas macro-habilidades.

1.4 Objetivo

El objetivo de este trabajo es diseñar, desarrollar y poner en producción una aplicación móvil llamada ***Podcast.ing***. Para lograr fortalecer la enseñanza y el aprendizaje de los alumnos, ofreciendo a los docentes una herramienta propia que se adapte a sus necesidades.

La aplicación posee dos vistas diferenciadas de acuerdo al rol del usuario que se loguea. Las funcionalidades para los roles aceptados son:

- Alumno:
  - Puede observar un conjunto de podcast.
  - Puede escuchar un podcast seleccionado y hacer sus ejercicios relacionados.
- Docente:
  - Puede crear, editar y eliminar un podcast y su correspondiente ejercitación.
  - Puede observar las notas de los estudiantes luego de que realizan los ejercicios de un podcast particular.

Para lograr este objetivo, se realizaron las interfaces de usuarios con los respectivos llamados a las funcionalidades del backend. Las cuales, se deben procesar y posteriormente si es requerido hacer una petición al servidor para traer información, editarla o eliminarla.

1.5 Estructura del informe de tesis

Este informe se organizó en siete capítulos donde se introducen los conceptos teóricos y se documentan los pasos llevados a cabo para la implementación de la aplicación móvil ***Podcast.ing***. Seguidamente se da un breve resumen de cada uno de los capítulos del informe, a saber:

- **Capítulo 2 – Metodología:** se ofrece una descripción teórica de las metodologías tradicionales y las metodologías ágiles. De esta última mencionada se explican algunas de sus variantes. Luego se detalla la metodología elegida, enumerando

sus ventajas y desventajas junto con la comparativa de las metodologías que le dieron origen.

- **Capítulo 3 - Planificación y análisis de requisitos del proyecto:** se expone la planificación llevada a cabo, teniendo en cuenta el análisis de las tareas de usuario que dieron origen a la definición de las tareas a realizar y la estimación de tiempo que va a llevar. Además, se detallan las tecnologías y herramientas seleccionadas para el desarrollo.
- **Capítulo 4 - Diseño del proyecto:** se explican los aspectos referidos al diseño de la aplicación móvil, exponiendo el diseño de interfaz de usuario, el diseño del gráfico de propiedades y el diseño de casos de uso.
- **Capítulo 5 - Desarrollo del proyecto:** se explica cómo se realizó la implementación de la aplicación móvil. Pasando desde el proceso de construcción de las vistas, proveedores y servicios dentro del código como también la configuración del almacenamiento en la nube, autenticación de usuario y la base de datos que utiliza procesamiento en tiempo real en una plataforma en la nube.
- **Capítulo 6 - Test del proyecto:** se detallan las pruebas que se realizaron para verificar la funcionalidad de la aplicación móvil desarrollada.
- **Capítulo 7 - Conclusiones y Trabajo Futuro:** se presentan las conclusiones obtenidas del desarrollo del presente trabajo final, teniendo en cuenta el problema y los objetivos planteados. Además, de mencionar las posibles mejoras de la aplicación móvil ***Podcast.ing*** y una perspectiva personal a la que se arribó con el desarrollo del trabajo.

Capítulo 2: Metodología

En el presente capítulo se realiza una definición de las metodologías tradicionales y las metodologías ágiles. Para estas últimas, se da una breve explicación de algunas de las metodologías pertenecientes a este grupo. Posteriormente, se justifica la elección de la metodología utilizada para el desarrollo de la aplicación móvil.

2.1 Definición de metodología ágil

Para comenzar esta sección, se realiza una definición de las metodologías tradicionales y las metodologías ágiles [4].

Las metodologías tradicionales son denominadas como metodologías pesadas (heavyweight) porque se enfocan en:

- Tener una documentación exhaustiva de todo el proyecto.
- Tener una planificación y control del proyecto.
- Tener especificaciones precisas de los requisitos y modelos.
- Cumplir con un plan de trabajo.

Estas metodologías tradicionales se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Esto tiene como objetivo conseguir un software más eficiente y para ello, se hace énfasis en la planificación del trabajo a realizar.

En ambientes donde los requisitos no se pueden predecir o estos pueden variar, las metodologías tradicionales no se adaptan adecuadamente a los cambios. Esto conlleva a que los costes al implementar un cambio sean altos y a tener escasa flexibilidad en proyectos donde el entorno es volátil. Las fases que la componen son:

- Elicitar requisitos.
- Analizar y diseñar los requisitos.
- Construir el producto.
- Probar el producto.
- Desplegar.

Las metodologías ágiles nacen como respuesta a los problemas que puedan ocasionar las metodologías tradicionales y se fundamentan en la adaptabilidad de los procesos de desarrollo. Un modelo de desarrollo ágil, generalmente es un proceso con las siguientes características:

- Incremental: entregas frecuentes con ciclos rápidos.
- Cooperativo: clientes y desarrolladores trabajan constantemente con una comunicación muy fina y constante.
- Sencillo: el método es fácil de aprender y modificar para el equipo.
- Adaptativo: capaz de permitir cambios de último momento.

Las metodologías ágiles proporcionan una serie de pautas y principios junto a técnicas funcionales que hacen que la entrega del proyecto sea menos complicada debido a que hay menos cantidad de documentación y más comunicación entre los clientes y los equipos de trabajo. Por eso son denominadas metodologías livianas (lightweight). Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más

importante que el seguimiento estricto de un plan. Las fases que se repiten en cada iteración son:

- Planificar.
- Priorizar requisitos.
- Analizar y diseñar de requisitos.
- Construir el producto.
- Probar el desarrollo.
- Lanzar el desarrollo.
- Retroalimentar.

En la Tabla 2 se puede observar de manera comparativa los dos tipos de metodologías definidas brevemente.

Características	Metodología Ágil	Metodología Tradicional
Respuesta ante cambios	Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Documentación	Poca documentación.	Documentación exhaustiva.
Artefactos	Pocos Artefactos.	Muchos Artefactos.
Contrato	Contrato bastante flexible. Lo normal es depender de contratos donde se paga por el tiempo necesario de desarrollo en vez de por el desarrollo de un conjunto de requerimientos específicos.	Existe un contrato prefijado.
Participación del cliente	Cliente es parte del equipo de desarrollo y está en el mismo sitio que los trabajadores.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Tamaño del equipo de trabajo	Grupos pequeños (máximo 10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Arquitectura	Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.
Énfasis	Orientado al equipo de trabajo.	Orientado al proceso.
Enfoque	El modelo ágil favorece la adaptación, es decir, referencia a la capacidad de adecuarse si es que surge algún imprevisto o cambio.	El modelo tradicional favorece la anticipación, es decir, referencia a la realización de todas las planificaciones posibles para evitar imprevistos.
Escala de proyectos	Pequeños y medios. A largo plazo puede crecer a proyectos de gran escala.	Grandes.

Tabla 2: Comparativa entre enfoque tradicional y ágil. Adaptado de [4].

A partir de lo expuesto y la comparativa realizada sobre las metodologías, se puede decir que las metodologías tradicionales han demostrado ser efectivas y necesarias en un gran número de proyectos, principalmente para aquellos que son grandes, críticos,

con equipos de desarrollo numerosos y distribuidos geográficamente. Pero también han presentado problemas en otros casos donde los requisitos son muy cambiantes.

Las metodologías ágiles han demostrado ser efectivas para ambientes dinámicos (requisitos volátiles), con equipos de trabajo pequeños que están en un mismo lugar geográfico, produciendo aplicaciones no críticas, dándole mayor valor al individuo, colaborando con el cliente como si fuera parte del equipo y construyendo de forma incremental software funcional en cada iteración.

Debido a lo mencionado anteriormente, se considera que para **Podcast.ing** se puede implementar la metodología ágil debido a que no es un proyecto crítico, las docentes del área de inglés se encuentran en el lugar de la desarrolladora lo que permite que haya una comunicación fluida entre ambas, se va a ir realizando entregables por cada iteración para que las profesoras puedan ver el producto y en caso de que se considere necesario realizar cambios.

Con el objetivo de elegir cuál metodología ágil utilizar, seguidamente se describirán las tres metodologías ágiles más implementadas en la actualidad [5]:

- **Scrum:** es una metodología popular de desarrollo ágil a través de la cual la productividad se vuelve muy alta. Está basada básicamente en el proceso de desarrollo de software incremental. En Scrum todo el ciclo de desarrollo es dividido en una serie de iteraciones donde cada iteración se denomina Sprint. La duración máxima de un Sprint es de 30 días. Comienza con la recopilación de los requisitos del usuario, pero no se espera que todos los requisitos deban cumplirse al principio. El usuario puede cambiar de opinión en cualquier momento durante el desarrollo; agregando nuevas funciones y/o eliminando o actualizando las ya existentes. La siguiente fase es priorizar los requisitos y crear una lista que se conoce como Product Backlog. Se debe hacer una planificación adecuada para cumplir con todos los requisitos, es decir, se debe especificar cuántos Sprints se necesitan para desarrollar el software y cuáles son los requisitos del Product Backlog que deben implementarse en cada Sprint. Después de cada Sprint, se realiza una revisión para determinar si todos los requisitos que se plantearon para ese Sprint en particular ya se han completado o no, y decidir los requisitos que deben implementarse en el próximo Sprint. Al finalizar cada Sprint se obtiene un incremento del software totalmente funcional.

- **Programación extrema (XP):** es la metodología más exitosa para desarrollar software ágil debido a su enfoque en la satisfacción del cliente. XP requiere la máxima interacción con el cliente para desarrollar el software. Divide todo el ciclo de vida de desarrollo de software en varios ciclos cortos. Acepta e incorpora cambios o requerimientos de los clientes en cualquier fase del ciclo de vida del desarrollo. XP comienza con la recopilación de los requisitos del usuario. Dependiendo de estos requisitos, todo el proceso de desarrollo se divide en varias iteraciones. Entonces, la siguiente fase es la planificación de las iteraciones, es decir, decidir el número de iteraciones que son necesarias, priorizar los requisitos y estimar la cantidad de esfuerzo requerido para realizar la implementación. Cada iteración se desarrolla usando programación en pares. Durante la fase de desarrollo pueden surgir requisitos nuevos y el plan de iteración debe ajustarse de acuerdo con eso. El siguiente paso es probar la última versión desarrollada para identificar errores, si se detectan; los errores se eliminarán en la siguiente iteración. Después de cada aceptación se debe realizar un seguimiento del

proyecto de prueba en el que se tome retroalimentación del proyecto sobre cuánto trabajo ya se ha realizado. XP ha introducido muchas cosas nuevas para los desarrolladores, como la programación en pares, la revisión exhaustiva del código, la refactorización de código y el espacio de trabajo abierto.

- **Kanban:** es una metodología que se basa en la teoría de las restricciones y uno de sus principales objetivos es reducir los tiempos de desarrollo, principalmente restringiendo el número de tareas que se trabajan en paralelo. De esta manera, la cantidad de trabajo que hay en un momento puede incrementar o reducir de acuerdo al conjunto de las tareas individuales que se encuentran en desarrollo en esa etapa en curso. Una diferencia importante con la mayoría de las metodologías ágiles es que Kanban no usa iteraciones sino que se enfoca en el flujo de elementos de trabajo individuales. Los principios básicos de Kanban son visualizar el flujo de trabajo y limitar el trabajo en curso. Esto generalmente se hace usando un tablero Kanban que ayuda a visualizar el progreso, como se muestra en la Figura 3.

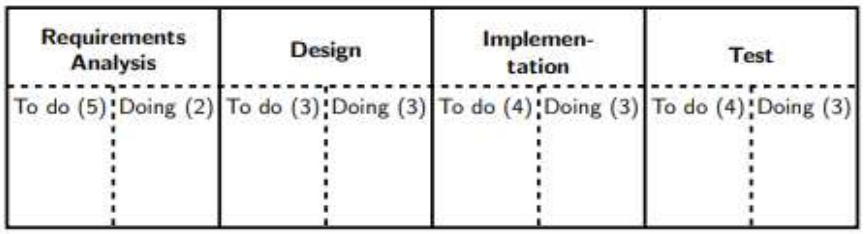


Figura 3: Ejemplo de tablero Kanban. Extraído de [5].

Un híbrido que surge de la mezcla de las metodologías Scrum y Kanban es **Scrumban** [6] que se creó originalmente como una forma de pasar de Scrum a Kanban agregando gradualmente características Kanban a Scrum, pero ahora se usa a menudo como una metodología propiamente dicha. Scrumban básico utiliza los componentes estándar de Scrum, pero además estructura el trabajo realizado dentro de un Sprint utilizando un tablero Kanban con los límites apropiados. De una forma más avanzada, se reduce la planificación de Sprints, moviéndose hacia la planificación on-demand que consiste en identificar un pequeño número de elementos de trabajo con la prioridad más alta para que pueda producirse y entregarse de manera más eficiente y para la satisfacción de los clientes.

2.2 Comparación de Scrum, Kanban y Scrumban

En esta sección se realiza una comparación [6] de las tres metodologías ágiles Scrum, Kanban y Scrumban (metodología híbrida surgida de la combinación de las dos primeras). Para ello, se muestra la Tabla 3 con las diferencias.

Características	Scrum	Kanban	Scrumban
Tablero	Se restablece con cada Sprint, lo que significa que todas las tareas se colocan en la columna de por hacerse (ToDo), lo que indica, que todavía no se han empezado a desarrollar.	No se producen reinicios porque no hay iteraciones: se proporcionan nuevas tareas en un flujo constante.	Generalmente se parece al tablero Kanban, pero se restablece cuando se completan todas las tareas, lo que significa que todas las tareas se colocan en la columna ToDo.
Artefactos	Requiere una acumulación de productos, una acumulación de Sprints y un gráfico de trabajo claramente definidos, lo que requiere más esfuerzo del equipo para mantener los artefactos actualizados.	No exige ningún artefacto específico.	Requiere una acumulación de iteraciones y estimaciones de las tareas.
Iteraciones	Define las iteraciones (llamadas Sprints) como parte del ciclo de vida. Pueden durar de una a cuatro semanas. Al final de cada Sprint, se espera un producto totalmente funcional con nuevas funciones u otras actualizaciones que sean aceptadas por el propietario del producto.	No define iteraciones; ya que las nuevas tareas se definen a pedido.	Tiene iteraciones, las cuales no están estrictamente definidas en términos de tareas y duración, sin embargo, su duración no debe ser mayor a 2 semanas ya que las iteraciones más cortas permiten una adaptación más rápida a los cambios.
Tareas	El tiempo de duración de cada tarea está limitado a la duración del Sprint, en cualquier caso, se trata de dividir tareas largas en otras más pequeñas.	No limita el tiempo de duración de las tareas.	No limita el tiempo de duración de las tareas, aunque tiene iteraciones, permite tareas de ejecución prolongada.
Prioridad	La priorización de tareas se realiza al planificar el Sprint.	Se realiza la priorización de tareas diariamente teniendo en cuenta la planificación justo a tiempo y el principio de “atracción”. La planificación justo a tiempo procura que el producto llegue cuando se necesita (ni antes ni después) y el principio de atracción consiste en que si se introduce una nueva tarea en el flujo de trabajo, debe tener la máxima prioridad para el equipo.	Primero prioriza el trabajo con la planificación del tamaño del depósito, esto consiste en una separación de los objetivos, a largo plazo (poco definidos), que se están preparando, los que están casi para tomarse y los listos para tomar. Luego las tareas se definen y priorizan para cada iteración y, por último, a diario, como con Kanban.

Características	Scrum	Kanban	Scrumban
Estimación del trabajo	Prescribe la estimación de tareas antes de cada Sprint.	No requieren estimación.	
Equipo	Debe ser multifuncional.	Permiten equipos multifuncionales y especializados, según el tipo de producto y lo que funciona mejor para un escenario determinado.	
Roles	• Propietario del producto: es responsable de ToDo. • Equipo de desarrollo. • Scrum master: es responsable de las reuniones diarias y la resolución de problemas no técnicos del equipo.	No define roles especiales, las tareas se dividen entre los miembros del equipo con el objetivo de mantener la metodología ágil.	
Cambios en el plan de trabajo	No permite ningún cambio en el plan de trabajo cuando se está ejecutando el Sprint; es por eso que se hacen planes y estimaciones detallados antes del Sprint.	No proporciona reglas que prohíban cambios en el plan de trabajo en un momento dado. Las tareas en estado ToDo se pueden reemplazar fácilmente por otras nuevas, también las tareas que ya están en proceso se pueden volver a ToDo y se pueden recuperar tareas más importantes.	
Corrección de errores	La corrección de errores está planificada para el próximo Sprint; no sería razonable cambiar el plan actual del Sprint debido a todas las preparaciones y estimaciones que se realizan antes del mismo. En realidad, si surgen errores críticos que deben solucionarse lo antes posible, los equipos de Scrum adoptan diferentes enfoques para abordar este problema. Algunos equipos definen un día de la semana (o parte de un día) como un "día de corrección de errores", otros equipos reducen la cantidad de puntos de historia por Sprint, por lo que aún queda algo de tiempo para cosas inesperadas, como corregir errores.	Permite la corrección de errores no planificados de inmediato: si corregir un error tiene una prioridad más alta que las tareas actuales en ToDo, entonces esta tarea se pone en el tablero.	

Tabla 3: Comparativa entre Scrum, Kanban y Scrumban. Adaptado de [6].

2.3 Metodología ágil aplicada: Scrumban

Para la implementación de este trabajo se consideró la utilización de la metodología híbrida Scrumban. Uno de los motivos fue que la utilización del tablero que proviene de Kanban permite visualizar las tareas que se encuentran en el estado "por hacer", "en desarrollo" y "finalizadas", lo que facilita una mejor organización a lo largo del desarrollo. Además, la separación en Sprint permite establecer objetivos con una fecha de entrega obteniendo un mayor control en los tiempos que se les asigna a las tareas. Finalmente, se resalta la ventaja de la flexibilidad que esta metodología tiene para subir tareas en caso que, las que fueron establecidas en el Sprint ya se encuentren listas. O en caso contrario, si se termina el Sprint y las tareas no fueron finalizadas extender el trabajo en ellas para la siguiente iteración.

Cabe destacar que, debido a que Scrumban es una metodología bastante nueva no se encuentran documentadas las fases de una iteración, por lo tanto, sus etapas [7] fueron desarrolladas teniendo en cuenta que su estructura se asemeja a la de Scrum, a saber:

1. **Planificación:** esta fase tiene como objetivo dividir la idea en piezas de trabajo más pequeñas (funciones) para luego priorizar cada función y asignarla a una iteración.
2. **Análisis de requisitos:** en esta fase se busca identificar los requisitos funcionales del sistema, así como también, quién utilizará el producto y cómo lo utilizará. Estos requisitos deben ser cuantificables, relevantes y detallados.
3. **Diseño:** a partir de los requisitos identificados en la fase anterior se construyen solamente los diagramas que proporcionan valor al proyecto. Cada uno de los diseños debe presentar un aspecto diferente del producto y debe crearse en base a la meta de lo que se quiere desarrollar. Estos varían ya que se deben adaptar a las necesidades del equipo desarrollador del proyecto.
4. **Codificación o desarrollo:** esta fase consiste en crear y probar funciones, y también corregir errores que puedan surgir.
5. **Pruebas:** una vez que se ha desarrollado el código, se prueba según los requisitos para asegurarse que el producto realmente resuelve las necesidades de los clientes y coincide con las historias de usuario.
6. **Despliegue:** el producto es entregado a los clientes para que lo utilicen. Sin embargo, esta acción no significa el final del proyecto. Una vez que los clientes comienzan a usar el producto, pueden enfrentarse a problemas nuevos que el equipo del proyecto tendrá que abordar.

En la Figura 4, se observa una representación de los pasos que se realizan en una iteración.

Figura 4: Proceso de Scrumban en un Sprint. Adaptado de [7].

2.3.1 Ventajas y desventajas

Todas las metodologías de desarrollo tienen sus ventajas y desventajas. A continuación, se realiza una breve clasificación de pros y contras de la metodología Scrumban [8]. Dentro de las ventajas se pueden distinguir:

- **Ayuda a ahorrar tiempo:** si el equipo no está utilizando ninguna forma de gestión de proyectos, usar Scrumban es una buena manera de comenzar con el seguimiento del trabajo que se está realizando. La aplicación de Scrumban evita que el equipo realice trabajo duplicado o dedique tiempo a tareas que no satisfacen el objetivo de un Sprint específico.
- **Ideal para proyectos que crecen a largo plazo:** debido a que Scrumban es una metodología ágil iterativa, permite pequeños cambios en grandes incrementos de tiempo. Esto lo convierte en un excelente marco para usar en proyectos a largo plazo, porque las necesidades del proyecto cambiarán a medida que pasa el tiempo. A medida que cambian las necesidades, Scrumban ayuda a los proyectos a iterar y mejorar sus procesos para mantenerse al día con esos cambios.
- **Los miembros individuales del equipo tienen más independencia:** si el equipo busca tener más autonomía, la metodología Scrumban brinda a sus miembros la oportunidad de tomar decisiones y priorizar el trabajo como mejor les parezca, en lugar de simplemente completar el trabajo asignado por un maestro Scrum o propietario del producto.

En lo que respecta a las desventajas, se pueden considerar:

- **La falta de gestión** puede causar confusión**.** Mientras que la independencia y la autonomía pueden motivar a un equipo, la falta de supervisión puede causar confusión y desorganización en otros equipos.
- **Scrumban es una metodología relativamente nueva** y como consecuencia no tiene tantos procesos establecidos. El proceso de un equipo Scrumban puede verse muy diferente del proceso de otro equipo, y parte de esa razón se debe a que no existe un marco estandarizado como el que existe en Scrum.
- **Los gerentes de proyecto tienen menos control** dado que no hay roles específicos en el equipo de desarrollo. Esto significa que todos tienen la misma facultad para elegir lo que consideran que es la decisión correcta para el Sprint, lo que llevaría a que el gerente de proyecto no posea tanta autoridad sobre el mismo como si lo tiene en la metodología Scrum.

Capítulo 3: Planificación y análisis de requisitos del proyecto

En este capítulo se realiza una explicación de cómo fueron consideradas y llevadas a cabo la primera y segunda etapa dentro de la metodología elegida, las cuales son la planificación y análisis de requisitos. Dichas etapas se encuentran ampliamente relacionadas por lo que se considera a las dos dentro de este capítulo.

En estas etapas se tuvo el mayor contacto con las profesoras de inglés, aunque cabe destacar que a lo largo de toda la implementación hubo una comunicación continua entre la desarrolladora y el equipo docente del área de inglés. Al inicio se obtuvieron los principales requisitos del sistema, los cuales permitieron establecer los ejes base de la aplicación móvil. A partir de lo solicitado, se estimaron fechas de entrega y la prioridad de cada una de las tareas que se establecieron en base a épicas especificadas. Logrando definir aproximadamente la cantidad de Sprint necesarios para cumplir con todas las funcionalidades requeridas. Se utiliza la palabra "aproximadamente" debido a que algunas tareas que fueron planificadas para un Sprint determinado se extendieron al siguiente. En resumen, la planificación consiste en organizar y documentar lo que se realiza para que a medida que pasan las iteraciones poder incrementar el valor de la aplicación. Por lo tanto, en este capítulo se explica cómo se realizó el análisis de las tareas de usuario, a partir de las cuales se definieron las tareas que eran requeridas para desarrollar la aplicación. Dichas tareas en base a su prioridad se establecieron en diferentes Sprint. Además, se explica cuáles fueron las tecnologías y herramientas que se requirieron para la construcción del proyecto.

3.1 Análisis de los Requerimientos

En esta sección se realiza una explicación de cómo fue considerado y llevado a cabo el análisis de los requisitos. Primero, se elicitaron los requerimientos funcionales de la aplicación móvil que las profesoras del área de inglés querían. Para ello, la desarrolladora les pidió escribir en conjunto lo que se denomina historias de usuarios (HU). Una historia de usuario es una explicación general e informal de una función de software escrita desde la perspectiva del usuario final o cliente. Esto significa que son unas pocas frases en lenguaje no técnico que describen el resultado deseado por parte del usuario final. Los tres pasos a seguir para redactar una historia de usuario son los siguientes:

1. Definir el perfil: que indica el rol del usuario final
2. Definir la necesidad: plantea el objetivo que tiene la función de software para el usuario final
3. Definir el propósito: especifica el objetivo de la experiencia del usuario final con la función de software

Se consideró utilizar la creación de las historias de usuario debido a que es una de las prácticas incluidas en las metodologías ágiles y a su vez porque brinda una serie de beneficios, tales como:

- **Las historias mantienen el foco en el usuario.** Una colección de historias mantiene al equipo enfocado en resolver problemas para usuarios reales.
- **Las historias permiten la colaboración.** Con el objetivo final definido, el equipo puede trabajar en conjunto para decidir cómo atender mejor al usuario y cumplir ese objetivo.

- **Las historias estimulan soluciones creativas.** Alientan al equipo a pensar de manera crítica y creativa sobre cómo resolver de la mejor manera un objetivo final.
- **Las historias crean impulso.** Con cada historia que pasa, el equipo de desarrollo disfruta de un pequeño desafío y una pequeña victoria, impulsando a seguir el desarrollo motivado.

Además, son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. También se utilizan en la fase de pruebas, para verificar si el producto cumple con lo que especifica la historia de usuario.

Para la definición de las historias de usuario, se tomó como base lo especificado por las profesoras sobre lo que querían en la aplicación móvil; y en forma conjunta las docentes con la desarrolladora escribieron las historias de usuario. Finalmente, se obtuvo un total de 7 historias de usuario, considerando que el número es adecuado con respecto a la magnitud del proyecto.

A continuación, se muestra en la Tabla 4 la plantilla que se utilizó para definir las historias de usuario, junto con la explicación de cada uno de sus componentes:

<table>
<tr><td colspan="2">Fecha: día de creación de la historia de usuario.</td></tr>
<tr><td>Número: identificador de la historia de usuario.</td><td>Nombre de la Historia de Usuario: descripción general de la historia de usuario.</td></tr>
<tr><td>Usuario: persona que utilizará la funcionalidad del sistema descrita en la historia de usuario.</td><td>Iteración Asignada: número de iteración, en que el cliente desea que se implemente esta historia de usuario.</td></tr>
<tr><td>Prioridad en Negocio: grado de importancia que el cliente asigna a la historia de usuario. Puede ser Alta / Media / Baja.</td><td>Puntos Estimados: tiempo considerado que es requerido para el desarrollo de la historia de usuario.</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: valor de complejidad que la historia de usuario representa al equipo de desarrollo. Puede ser Alta / Media / Baja.</td></tr>
<tr><td colspan="2">Descripción: información detallada de la historia de usuario.</td></tr>
<tr><td colspan="2">Observaciones: consideraciones a tener en cuenta en la historia de usuario.</td></tr>
</table>

Tabla 4: Plantilla de la historia de usuario.

Seguidamente, en las Tablas 5 y 6 se presentan a modo de ejemplo dos historias de usuario que se escribieron en el proyecto. El resto de ellas, se detallan en el Anexo 1.

<table>
<tr><td colspan="2">Fecha: 05/09/22</td></tr>
<tr><td>Número: 1</td><td>Nombre de la Historia de Usuario: login en la aplicación móvil.</td></tr>
<tr><td>Usuario: estudiante/ profesor</td><td>Iteración Asignada: 1</td></tr>
<tr><td>Prioridad en Negocio: Media</td><td>Puntos Estimados: 28 horas</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: Media</td></tr>
</table>

Descripción: el estudiante/profesor debe ingresar a la aplicación móvil colocando email y contraseña.
Observaciones:

Tabla 5: Historia de usuario de login.

Fecha: 05/09/22	
Número: 2	**Nombre de la Historia de Usuario:** registrarse en la aplicación móvil.
Usuario: estudiante	**Iteración Asignada:** 1
Prioridad en Negocio: Media	**Puntos Estimados:** 24 horas
Riesgo en Desarrollo: Media	
Descripción: el estudiante debe poder crear una cuenta para así ingresar a la aplicación móvil.	
Observaciones:	

Tabla 6: Historia de usuario de registrarse.

3.1.1 Análisis de las historias de usuario

En base a lo definido en las historias de usuario creadas, se realizó un análisis de cada una de ellas para definir las funciones de software y los desarrollos de interfaz de usuario a realizar dentro de la implementación del proyecto. A continuación, se muestra un listado con todas las vistas (frontend) y las funcionalidades (backend) obtenidas después del análisis.

- Frontend:
  - Pantalla de inicio de sesión.
  - Pantalla de registración.
  - Pantalla inicio vista de estudiante.
  - Pantalla del reproductor del podcast.
  - Pantallas de los distintos ejercicios.
    - Seleccionar la opción correcta.
    - Verdadero o falso.
    - Escribir la palabra correcta.
  - Pantalla inicio vista de docente.
  - Pantalla de vista de resultados de los ejercicios del estudiante.
  - Pantalla de carga/edición de podcast.
  - Pantalla de carga/edición de ejercicios.
    - Seleccionar la opción correcta.
    - Verdadero o falso.
    - Escribir la palabra correcta.
  - Creación de servicio podcast para conectarse con el backend.
  - Creación de servicio ejercicios para conectarse con el backend.
  - Creación de resultados de ejercicios podcast para conectarse con el backend.

- Backend:
    - Alta, Baja, Modificación, Lectura y Consulta de podcast en la base de datos (BD).
    - Alta, Baja, Modificación, Lectura y Consulta de ejercicios en la BD.
    - Alta, Baja, Modificación, Lectura y Consulta de resultados de ejercicios en la BD.
    - Alta, Baja, Modificación, Lectura y Consulta de estudiantes en la BD.
    - Crear autenticación.
    - Crear carga y descarga de audio en storage. El término storage hace referencia a un almacén de datos.

3.2 Definición de tareas

En base a las historias de usuario se hizo un desglose de tareas, tomando en consideración el análisis realizado en la sección 3.1.1. A continuación, se muestra en base a cada una de las historias de usuarios, qué tareas corresponden para su implementación:

- HU1: login en la aplicación móvil.
    - Tarea 1. Realizar el back de consulta en la base de datos de los usuarios.
    - Tarea 2. Realizar el front de login.
- HU2: registrarse en la aplicación móvil.
    - Tarea 3. Realizar el back de alta en la base de datos de los usuarios.
    - Tarea 4. Realizar el front de registrarse.
- HU3: visualización de los podcast con ejercicios.
    - Tarea 5. Realizar back para lectura en la base de datos de los podcast.
    - Tarea 6. Realizar back para lectura de la base de datos de los ejercicios relacionados a los podcast.
    - Tarea 7. Realizar front de listado de podcast para los estudiantes.
    - Tarea 8. Realizar front para ver el ejercicio:
        - Seleccionar la opción correcta.
        - Verdadero o falso.
        - Escribir la palabra correcta.
- HU4: visualización de corrección del ejercicio realizado por el estudiante.
    - Tarea 9. Realizar back para alta o modificación en la base de datos de los resultados.
    - Tarea 10. Realizar front para ver el resultado obtenido.
- HU5: visualizar los resultados de los estudiantes.
    - Tarea 11. Realizar back para lectura en la base de datos de los resultados.
    - Tarea 12. Realizar front de listado de los resultados de los ejercicios que realizaron los estudiantes.
- HU6: crear, editar y borrar los podcast.
    - Tarea 13. Realizar back para consultar/editar/crear/eliminar en la base de datos de los podcast.

- Tarea 14. Realizar back para consultar/editar/crear/eliminar de la base de datos de los ejercicios relacionados a los podcast.
- Tarea 15. Realizar front de listado de podcast para los docentes.
- Tarea 16. Realizar front para editar/crear un podcast.
- Tarea 17. Realizar front para editar/crear el ejercicio:
  - Seleccionar la opción correcta.
  - Verdadero o falso.
  - Escribir la palabra correcta.

- HU7: reproducir podcast.
  - Tarea 18. Realizar front para ver el podcast.
  - Tarea 19. Realizar back de reproductor de podcast.

3.2.1 Definición de fichas de tareas

Tomando en consideración las tareas en la sección 3.2, se parte con la documentación de las fichas de tarea. Se define como fichas de tarea a la división en actividades más pequeñas de aquello que se encuentra definido dentro de la historia de usuario. Dichas fichas de tarea poseen una estimación del tiempo que se considera que es necesario para completarla (realizada por todo el equipo de desarrolladores). Además, contiene el responsable que va a realizar la tarea, una descripción de lo que se va a realizar y a que historia de usuario se encuentra relacionada. A continuación, se muestra en la Tabla 7 la plantilla de la ficha de tarea que se utilizó, junto con la explicación de cada uno de sus componentes:

Tarea	
Número Tarea: identificador de la tarea.	**Historia de Usuario (Nro. y Nombre):** identificador de la historia de usuario a la que se encuentra relacionada.
Nombre Tarea: descripción general de la tarea.	
Tipo de Tarea: clasificación del tipo de tarea. Puede ser Desarrollo / Corrección / Mejora / Otra (especificar).	**Puntos Estimados:** horas consideradas que va a tomar realizar la tarea.
Fecha Inicio: fecha de comienzo de la resolución de la tarea.	**Fecha Fin:** fecha de finalización de la tarea.
Programador Responsable: persona encargada de desarrollar la tarea.	
Descripción: breve explicación de lo que se debe realizar dentro de la tarea.	

Tabla 7: Plantilla de la ficha de tarea.

En las Tablas 8 y 9 se presentan a modo de ejemplo dos fichas de tarea que se especificaron en el proyecto. El resto de ellas, se detallan en el Anexo 2.

Tarea	
Número Tarea: 2	**Historia de Usuario (Nro. y Nombre):** 1 - Login en la aplicación móvil.
Nombre Tarea: Front de login	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 12hs.
Fecha Inicio: 21/09/22	**Fecha Fin:** 23/09/22
Programador Responsable: Ivanna F. Jutterpeker	

Descripción: Se debe crear la vista del login aplicando el diseño establecido en el prototipo aprobado por las profesoras.

Tabla 8: Ficha de tarea del login.

Tarea	
Número Tarea: 4	**Historia de Usuario (Nro. y Nombre):** 2 - registrarse en la aplicación móvil.
Nombre Tarea: Front de registrarse	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 8hs
Fecha Inicio: 25/09/22	**Fecha Fin:** 25/09/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la vista para registrarse aplicando el diseño establecido en el prototipo aprobado por las profesoras.	

Tabla 9: Ficha de tarea de registrarse.

3.2.2 Especificación Sprint

Adhiriendo a la metodología Scrumban, se sigue el principio de dividir las tareas por iteraciones. Como ya se mencionó cada iteración se denomina Sprint y tiene como objetivo generar entregables al finalizar las semanas establecidas de una iteración. En cada una de las iteraciones se busca mejorar y agregar funcionalidades al proyecto que se va desarrollando.

En cada Sprint se establecen los objetivos que guiarán la iteración. En base a esto se seleccionan las tareas que permitan el cumplimiento del Sprint. Si no se logra desarrollar todas las tareas en una iteración, las que no se pudieron finalizar se pasarán al siguiente Sprint.

Para el desarrollo de la aplicación móvil ***Podcast.ing***, se dividió el proyecto en 4 Sprint, en cada uno de los cuales se entregó una parte del producto completamente funcional. Los distintos Sprint que constarán de 2 semanas son los siguientes:

- Sprint 1- “Login y registro front y back”:
  - Creación de vista del login y registro.
  - Creación de la conexión a authentication de Firebase para el login y el registro.
  - Creación de la conexión (Alta, Baja, Modificación, Lectura, Consulta) a la realtime database de Firebase para el login y el registro.
- Sprint 2- “Creación del front y back de la sección del profesor parte de podcast”:
  - Creación de la vista del listado de podcast con especificaciones para eliminar, editar y crear.
  - Creación de la vista de edición/creación de podcast.
  - Creación de la vista de edición/creación de ejercicios.
  - Creación de la conexión (Alta, Baja, Modificación, Lectura, Consulta) a la realtime database de Firebase para podcast.
  - Creación de la conexión (Alta, Baja, Modificación, Lectura, Consulta) a la realtime database de Firebase para los ejercicios.
  - Crear conexión con storage para almacenar audio.

- Sprint 3- “Creación del front y back de la sección del profesor parte resultados de estudiantes”:
    - Creación de la vista de los resultados obtenidos por los estudiantes.
    - Creación de la conexión (Alta, Baja, Modificación, Lectura, Consulta) a la realtime database de Firebase para resultados de los ejercicios.
    - Creación del widget para hacer cambio de vista de podcast a resultados y viceversa.
- Sprint 4- “Creación del front y back de la sección del estudiante”:
    - Creación de la vista del listado de podcast.
    - Creación de la vista del reproductor de podcast.
    - Creación de la vista de ejercicios.
        - Seleccionar la opción correcta.
        - Verdadero o falso.
        - Completar con la palabra correcta.
    - Creación del almacenamiento del podcast en Firebase Storage.

A continuación, para una mejor comprensión de los Sprints se muestra en la Figura 5 la Estructura de Desglose del Trabajo (EDT) que consiste en una descomposición jerárquica orientada al trabajo que será ejecutado por el equipo del proyecto para lograr los objetivos del mismo y crear los entregables requeridos [9].

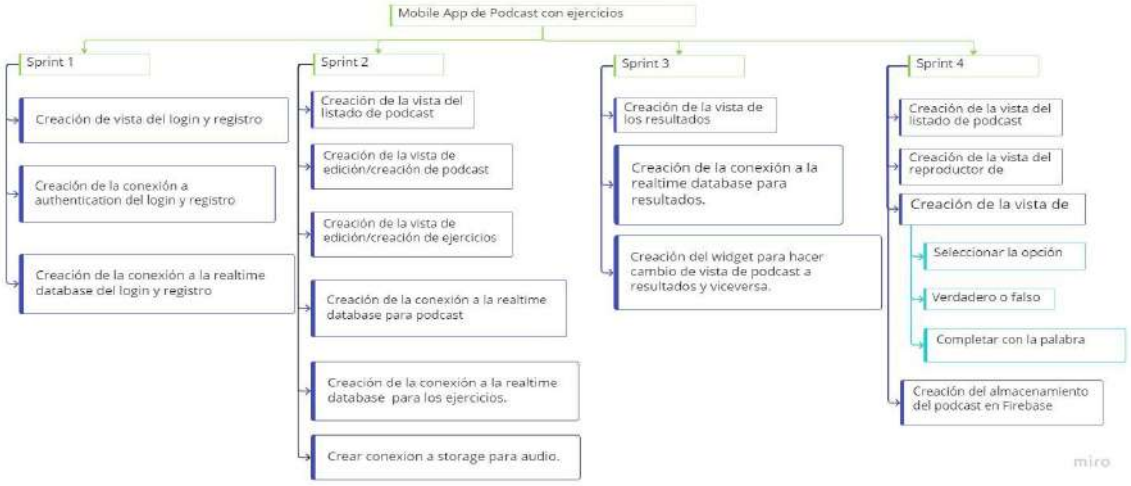


Figura 5: Estructura de Desglose del Trabajo

3.3 Tecnologías y herramientas

La presente sección tiene como objetivo describir las tecnologías y herramientas utilizadas a lo largo del desarrollo del proyecto. Se realiza una comparativa entre las distintas opciones que existen en el mercado y la justificación de lo que se ha seleccionado.

Un framework es una biblioteca de software que proporciona una estructura fundamental para soportar el desarrollo de aplicaciones para un entorno específico. En la Tabla 10 se comparan algunos de los frameworks existentes en el mercado y que fueron considerados como posibles de utilizar al momento de la implementación del proyecto [10].

Características	Flutter	React Native	Ionic	Apache Cordova	Xamarin
Definición	Marco de desarrollo de aplicaciones móviles de Google. Es un extenso kit de desarrollo de software de código abierto.	Marco de desarrollo de aplicaciones móviles de código abierto. Se destaca por su facilidad para crear aplicaciones iOS y Android.	Marco de desarrollo de aplicaciones multiplataforma, aplicaciones web progresivas híbridas y dinámicas.	Marco de desarrollo de aplicaciones móviles de código abierto. Es multiplataforma y las aplicaciones se ejecutan dentro de contenedores destinados a cada plataforma y se basan en enlaces API para acceder a las capacidades de cada dispositivo, como sensores, datos, estado de la red, etc.	Marco multiplataforma lanzado por Microsoft.
Basado	Dart	JavaScript	Angular/Vue/React y Apache Cordova	HTML5, CSS3 y JavaScript	.NET and C#
Aplicaciones desarrolladas	Google Ads, Google Pay, eBay Motors, PostMuse	Facebook, Instagram, Skype, Shopify, FlipKart	MarketWatch, Firstly, ChefSteps, DieselOn	Localeur, Untappd, SparkChess	Volveremos, Aussie Weather, Tube Mate

Tabla 10: Comparativa de Frameworks. Adaptado de [10].

A continuación, por cada framework se ampliará la información brindada en la Tabla 10 agregando una breve explicación de sus principales características (considerar que las URL para acceder a las páginas oficiales de los framework se encuentran en los nombres presentes en la Tabla 10 y disponibles en el Anexo URL), a saber:

- **Flutter**:
  - Es de código abierto de forma gratuita.
  - Compila JavaScript, Intel y ARM. Esta última sigla significa Advanced RISC Machine (máquina avanzada de RISC), siendo RISC un conjunto de instrucciones reducidas.
  - Utiliza arquitectura en capas.
  - Ofrece widgets que se pueden personalizar (iconos, navegación, desplazamiento, entre otros).
  - Tiene API de movimiento ricas en funcionalidades.
  - Proporciona un kit de interfaz de usuario intuitivo.
  - Usa para las imágenes el motor de renderizado 2D Skia. Siendo Skia una biblioteca de gráficos en 2D de código abierto que proporciona API comunes que funcionan en una variedad de plataformas de hardware y software.
  - Ejecuta pruebas de control de calidad automatizadas y depuración simple.
  - Tiene una función de recarga inmediata, lo que permite ver los cambios en el momento sin recargar la aplicación.
- **React Native**:
  - Es de código abierto de forma gratuita.
  - Es de código bajo (proporciona componentes y código que se pueden reutilizar).
  - Tiene numerosos componentes independientes de la plataforma (estos incluyen Texto, Vista e Imagen).
  - Provee migración simple debido al soporte de bibliotecas de terceros.
  - Se adapta a varios tamaños de pantalla.
  - Posee gran compatibilidad con varias extensiones.
  - Es de fácil mantenimiento.
  - Tiene una comunidad extensa.
  - Posee una baja curva de aprendizaje.
- **Ionic**:
  - Es de código abierto de forma gratuita.
  - Utiliza numerosos componentes de JavaScript.
  - Tiene muchos temas y características como carga diferida.
  - Brinda complementos para el uso de dispositivos (por ejemplo, para acceder a la cámara o Bluetooth).
  - Utiliza para la interfaz de usuario vistas web y componentes intuitivos optimizados para dispositivos móviles.
  - Brinda estilo adaptativo.
  - Posee compilación adelantada[1], gestos táctiles optimizados y renderizado previo.
  - Posee una comunidad extensa.

[1] Consiste en la compilación durante la etapa de construcción, antes de que el navegador levante y corra el código.

- **Apache Cordova**:
    - Es de código abierto de forma gratuita.
    - Posee complementos para acceder a hardware como la cámara, GPS, etc.
    - Tiene CLI (interfaz de línea de comandos).
    - Posee acceso a la API del dispositivo nativo.
    - Brinda escenarios para uso fuera de línea.
    - Posee una gran comunidad.
- **Xamarin**:
    - Permite crear aplicaciones para muchos sistemas operativos además de iOS y Android, incluidos Windows, tvOS, macOS, watchOS, entre otros.
    - Se conecta con Microsoft Visual Studio.
    - Utiliza documentos basados en datos.
    - Utiliza archivos de guión gráfico.
    - Posee API nativas y gráficos 2D.
    - Brinda administradores de emuladores de Google y SDK de Android.
    - Posee una infraestructura de backend bastante versátil.
    - Tiene una comunidad activa y es apoyado corporativamente.

La página web de Upsilonit [10] documenta, a Agosto de 2022, el porcentaje de utilización de cada uno de los frameworks mencionados, dicha información se visualiza en la Figura 6. Como se puede observar, Flutter es el framework más utilizado debido a que permite crear aplicaciones multiplataforma donde se le da mucha importancia a la interfaz de usuario facilitando una amplia biblioteca de widgets personalizables de manera sencilla para crear aplicativos atractivos a la vista. Finalmente, y luego de analizar los posibles frameworks, se decide que para este proyecto se va a utilizar Flutter debido a sus grandes ventajas.

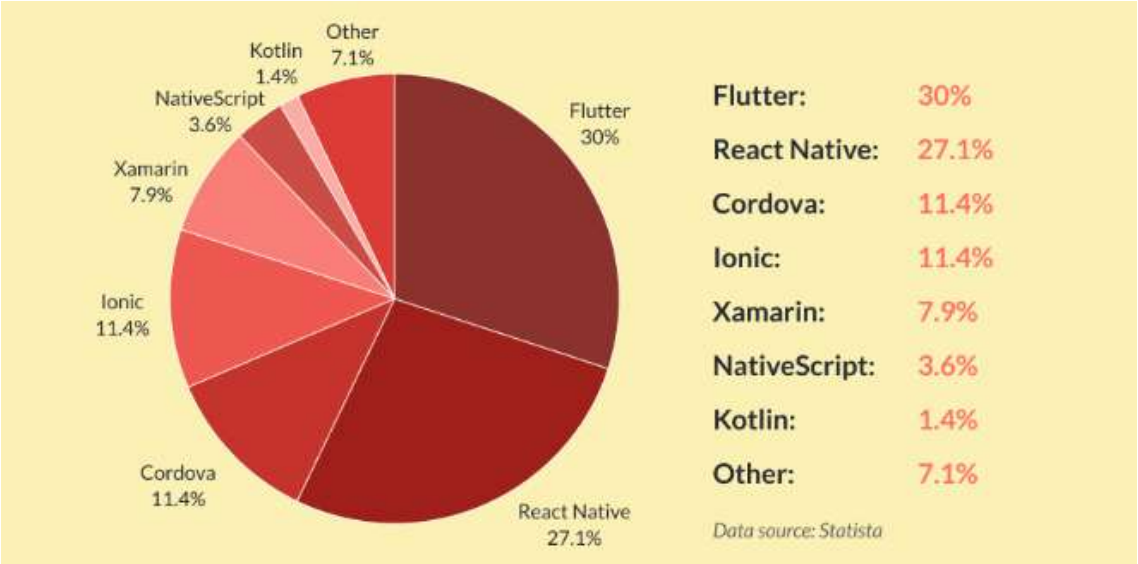


Figura 6: Estadística de utilización de frameworks. Datos recolectados a la fecha 09/06/22. Extraído de [10].

Teniendo en cuenta el framework seleccionado, se analizaron los backend utilizados en la actualidad para las aplicaciones Flutter concluyendo que lo que se implementa en el mercado es: Backend como servicio (BaaS).

BaaS consiste en un servicio en la nube que proporciona herramientas e infraestructura para el desarrollo de aplicaciones. El proveedor de BaaS se encarga de la administración y el mantenimiento del servidor. También proporciona utilidades que ayudan a generar códigos backend y agilizar las tareas de desarrollo.

En la Tabla 11 se muestra una comparación entre los distintos BaaS que se pueden utilizar para Flutter [11].

Características	Back4app	Parse	Firebase	Backendless	AWS Amplify
Descripción	Es una plataforma para crear, alojar y ejecutar aplicaciones escalables en escaso tiempo. Se basa en un marco de código abierto, con varias herramientas de desarrollo avanzadas para aplicaciones web, móviles y de IoT.	Es un marco de código abierto para crear backend de aplicaciones con soporte para muchas tecnologías de desarrollo frontend, incluidas Swift, React Native, Ionic, Java y Xamarin. Se puede utilizar para desarrollar aplicaciones web, móviles, y de IoT.	Es una conocida plataforma backend impulsada por Google. Tiene muchas características que mejoran el proceso de desarrollo de aplicaciones web y móviles. Es bien conocida por su sólida base de datos NoSQL que utiliza el protocolo JSON para las acciones de almacenamiento de datos.	Es un backend móvil como servicio. Ofrece API a todos los usuarios y presenta opciones de alojamiento, como alojamiento compartido en la nube, alojamiento dedicado y servidores administrados.	Es seguro y ágil; gracias a los servidores distribuidos en diferentes partes del mundo. Ofrece muchas características y recursos avanzados para lograr varios objetivos de alojamiento en la nube.
Características	• Base de datos escalable • API • Código en nube • Notificaciones • Almacenamiento	• Bases de datos NoSQL o SQL • API GraphQL o REST • Integración social • Correo electrónico y notificaciones automáticas	• Base de datos en tiempo real • Almacenamiento • Aprendizaje automático • Analítica • API	• Base de datos • Actualizaciones en tiempo real • Desarrollo visual • API • Notificaciones	• Almacén de datos • API (tanto REST como GraphQL) • Predicciones • Almacenamiento • Notificaciones push
Precio actualizado al día 25/11/22	Plan básico gratuito y también hay planes a partir de los 15 USD/mes.	Disponible en Github gratis.	Plan básico gratuito y luego existe una escala por pago según su uso.	Plan básico gratuito y también hay planes de 50 USD/mes.	Plan básico gratuito y luego existe una escala por pago según su uso.

Tabla 11: Comparativa entre BaaS. Extraído de [11].

En base a los BaaS mencionados, la información obtenida de sus respectivas páginas oficiales (URL en los nombres de los BaaS en la Tabla 11 y disponibles en el Anexo URL) y observando cómo son sus funcionalidades, se destaca Firebase; dado que es el que posee una interfaz más amigable, una curva de aprendizaje baja y proporciona en su plan gratuito: autenticación con correo y contraseña, Realtime Database (conexiones simultáneas: 100, GB almacenados: 1GB y varias bases de datos por proyecto:10 GB por mes), etc. Lo cual satisface las condiciones necesarias para ser el backend en el proyecto que se implementó.

Otra decisión importante es la selección de un IDE, para ello se llevó a cabo un análisis de aquellos que son utilizados en la actualidad. Un IDE [12] es un programa de software o una combinación de herramientas que asiste en la escritura y prueba software. En resumen, un IDE es una combinación de herramientas básicas necesarias para el desarrollo de aplicaciones. Consta al menos de un editor de texto, herramientas de automatización de compilación y un depurador. Además, algunos IDE vienen con los beneficios de instalar complementos para extender sus funcionalidades.

En la Tabla 12 se compara los IDE considerados para su utilización (cabe aclarar que las URL para acceder a las páginas oficiales de los IDE se encuentran en los nombres presentes en la Tabla 12 y disponibles en el Anexo URL). Y debajo se detallan algunas características principales de cada uno.

Características	Android Studio	Qt Creator	Xcode	JetBrains Rider	Visual Studio Code
Definición	Es una herramienta popular de programación recomendada por Google. Viene con características que facilitan el trabajo de los desarrolladores en aplicaciones Android. Una de las características esenciales son sus dispositivos virtuales (emuladores).	Es un IDE multiplataforma para desarrolladores móviles experimentados.	Es un IDE de uso común para el sistema operativo macOS. Con este IDE, los desarrolladores implementan software o aplicaciones en Mac que se pueden usar en iOS, iPadOS, macOS, tvOS y watchOS.	Es uno de los más potentes IDE disponibles para el desarrollo de Xamarin. Proporciona funciones adicionales, como inspecciones de código y refactorizaciones, que mejoran su experiencia de C# y permiten a los desarrolladores escribir código sin errores de manera eficiente.	Microsoft introdujo Visual Studio Code que viene con una amplia gama de funciones, como depuración, resaltado de sintaxis, finalización de código inteligente, fragmentos, refactorización de código y Git integrado.
Costo actualizado al día 25/11/22	Gratis.	Es gratis para uso personal pero comienza en 42 USD/mes para uso comercial.	Gratis.	Ofrece una prueba de 30 días y el precio comienza en 149 USD/año.	Es gratis para uso personal pero comienza en 45 USD/mes para uso comercial.
Dispositivos	Windows, Linux y macOS.	Windows, Linux y macOS.	MacOS.	Windows, Linux y macOS.	Windows, Linux y macOS.
Lenguajes	Java, C, C ++, Kotlin, XML.	C y C++.	Swift, AppleScript, C, C++, Objective-C, Objective-C++, Python y Ruby.	Desarrollar aplicaciones para dispositivos Android e iOS.	C, C++, C#, F#, JavaScript, etc.

Tabla 12: Comparativa de IDE. Extraído de [12].

Con respecto a las características principales de los IDE, se puede decir:

- **Android Studio**:
    - Su editor de diseño visual ConstraintLayout permite a los desarrolladores crear diseños rápidamente arrastrando elementos de la interfaz de usuario en lugar de escribir código complejo. Este editor puede verificar los diseños en varios dispositivos y versiones de Android. Puede cambiar el tamaño de los diseños dinámicamente para que se muestren perfectamente según el tamaño de la pantalla.
    - Posee un analizador que verifica la composición del archivo ejecutable de la aplicación (APK) minimizando su tamaño, lo que permite una fácil instalación en dispositivos con poca memoria. También, reduce el tiempo de depuración de los archivos DEX (que contienen el código) y otros recursos.
    - Contiene un emulador rápido de Android que ayuda a los desarrolladores a probar sus aplicaciones en diferentes dispositivos sin tener el dispositivo físico real. Utilizar un emulador es la forma más rápida de visualizar la aplicación sin necesidad de transferir los datos al dispositivo físico.
    - Al tener un sistema de construcción flexible le permite a los desarrolladores personalizar la compilación y generar múltiples variantes de compilación para diferentes dispositivos usando un solo proyecto.
    - Tiene herramientas de creación de perfiles integrados que garantizan estadísticas en tiempo real, como la CPU, la memoria y la actividad de la red. Esto ayuda a los desarrolladores a identificar los cuellos de botella de rendimiento.

- **Qt Creator**:
    - Su editor de código asiste a los desarrolladores a escribir código proporcionando ayudas tales como finalización de código, resaltado de sintaxis, documentación integrada, etc.
    - Permite el control de versiones a partir de varias herramientas externas que incluyen Git, Subversion, Mercurial, etc.
    - A partir de su diseño de interfaz de usuario integrado permite a los desarrolladores crear, con controles listos para usar, aplicaciones basadas en widgets de C++.
    - Ayuda a los desarrolladores en la gestión de proyecto y su construcción proporcionando compatibilidad con Cmake y funciones similares a la compilación cruzada. Siendo Cmake una multiplataforma para poder gestionar la construcción de manera automática del código.
    - Permite a los programadores ejecutar proyectos en varios sistemas integrados, móviles y de escritorio con configuraciones de compilación que permiten cambiar entre múltiples objetivos.
    - Permite a los desarrolladores probar y depurar las aplicaciones en un emulador de dispositivos para proporcionar entornos reales.
    - Compila el código fuente en código nativo más rápido de lo habitual.

- **Xcode**:

    - Viene con un constructor de interfaz que tiene la función de lienzo de diseño de Interface Builder que permite a los desarrolladores crear prototipos de una interfaz de usuario completa sin escribir ningún código. Luego se puede conectar gráficamente la interfaz creada al código fuente.
    - Su control de versiones consiste en Git. Además, junto con el control de fuentes se puede dividir la pantalla en dos y ver las versiones de un archivo donde se resaltan sus diferencias.
    - Su navegador de prueba permite a los desarrolladores saltar rápidamente a cualquier prueba específica y ejecutarla.
    - Permite configurar fácilmente el entorno de Xcode al incluir funciones, como pestañas, comportamientos, etc.
    - Permite a los desarrolladores usar "Command-shift-O" para obtener acceso rápido a cualquier archivo de proyecto. Proporciona la finalización inmediata de la búsqueda, desde la cual se puede elegir el archivo deseado y presionar enter.
- **JetBrains Rider**:
    - Su editor de código permite que los espacios de nombres se completen automáticamente, tiene llaves de inserción automática y resaltado de sintaxis, reorganización de código, refactorizaciones de acceso rápido y acciones de contexto.
    - La navegación y búsqueda permite a los desarrolladores saltar a cualquier archivo, tipo o miembro dentro del código, encontrar configuraciones, con la ayuda de un acceso directo estándar de Buscar en todas partes. Puede encontrar el uso de símbolos, usos entre idiomas y uso de cadenas literales.
    - Los desarrolladores pueden saber cómo funciona el código de terceros, utilizando la opción de descompilador en el archivo ejecutable en C#. Después de eso, el desarrollador puede navegar por el código.
    - Viene con más de 60 acciones de contexto de refactorización de ReSharper (extensión con múltiples funcionalidades). Permite a los desarrolladores cambiar el nombre, extraer métodos, clases, etc.
    - Permite a los desarrolladores ejecutar y depurar pruebas unitarias basadas en NUnit. Marca los métodos de prueba y las clases para que el desarrollador simplemente pueda ejecutar, depurar y administrar estas pruebas.
    - Viene con los controladores de versiones Git, Subversion, Mercurial, Perforce, etc.
- **Visual Studio Code**:
    - Su centro de aplicaciones permite a los desarrolladores automatizar la construcción del código de las aplicaciones de iOS, Android, Windows y macOS. Los desarrolladores pueden probar en miles de dispositivos reales, distribuirlos a probadores beta y tiendas de aplicaciones, y monitorear el uso en el mundo real utilizando datos analíticos.
    - Posee un código inteligente el cual brinda un conjunto de herramientas de finalización automática de código que atraviesan el contexto de su código, como nombres de variables, funciones, etc. Completa una línea a la vez, lo que da como resultado un código más preciso.

- Se puede obtener información sobre el código en lo referido a los cambios realizados, el resultado de esos cambios, referencias, autores, pruebas, etc.

De acuerdo a todos los IDE que se mencionaron anteriormente, el que mejor se adaptaba al desarrollo es Visual Studio Code en su versión gratuita. Además de que se puede usar en Windows que es el tipo de dispositivo con que cuenta la desarrolladora. Una gran ventaja sobre los demás consiste en su posibilidad de amplias gamas de funciones, como depuración, resaltado de sintaxis, finalización de código inteligente, fragmentos, refactorización de código, etc. Sumado a la posibilidad de instalar complementos para extender sus funcionalidades (para este proyecto se instaló el framework Flutter) y su integración con un sistema de control de versiones.

Un sistema de control de versiones es una herramienta utilizada en el desarrollo de software con el objetivo de tener un lugar seguro para el código, permitiendo poseer varias versiones de código en diversas ramas y tener también allí la versión estable. A continuación, se muestra los sistemas de control de versiones de software, así como las diferentes opciones de control de versiones que hay (considerar que las URL para acceder a las páginas oficiales de los sistemas de control de versiones se encuentran en los nombres presentes de los mismos y disponibles en el Anexo URL):

- **Git:** es un sistema de control de versiones distribuido diseñado por Linus Torvalds a partir de las necesidades surgidas en del desarrollo del proyecto del núcleo Linux, donde los requisitos eran un sistema descentralizado, rápido, flexible y robusto. Fue escrito en una combinación de Perl, C y varios scripts de Shell.
- **Apache Subversion** (SVN): es un sistema de control de versiones de código abierto. Es fácil de utilizar con cualquier lenguaje de programación y ofrece un administrador encargado de la seguridad, historial de versiones, control de acceso de usuario, etc. Además, permite el manejo tanto de archivos de texto como binarios.
- **Mercurial:** es un sistema de control de versiones escrito en Python. Que tiene como objetivo ser rápido, portable y fácil de usar. Además, posee una interfaz web integrada.
- **Monotone:** es un sistema de control de versiones distribuido escrito en C ++, que funciona con el protocolo P2P. Los sistemas operativos que permite son: Unix, Linux, BSD, Mac OS X y Windows.

Dado que el IDE elegido (Visual Studio Code) posee incorporado el sistema de versiones Git, se va utilizar este controlador de versiones para poder tener el código almacenado en un repositorio remoto.

Capítulo 4: Diseño del proyecto

En este capítulo se realiza una explicación de cómo fue considerada y llevada a cabo la tercera etapa dentro de la metodología elegida, la cual es diseñar. Se brinda una descripción de los diseños que se realizaron en el proyecto, que son necesarios para la siguiente etapa de la metodología que consiste en la codificación.

El diseño se realiza durante todo el tiempo de vida del proyecto, siendo constantemente revisado y muy probablemente modificado debido a cambios presentados durante el desarrollo. Se parte documentando el diseño de diagrama de Casos de Usos, luego se menciona el diseño de Interfaz de Usuario y por último el diseño de Gráfico de propiedades. Estos diseños, van a ser el soporte para poder crear el código de la aplicación y también la configuración del backend documentados en el siguiente capítulo.

4.1 Diseño de diagrama de Caso de Usos

Cuando se habla de diagrama de caso de uso [13] [14] se especifica solo lo que se supone que debe hacer el sistema, es decir, los requisitos funcionales del sistema sin mencionar los requisitos no funcionales que a menudo incluyen objetivos de rendimiento, lenguajes de programación, etc. La vista de casos de uso captura el comportamiento de un sistema y subsistema, tal y como se muestra a un usuario externo (actor). Divide la funcionalidad del sistema en transacciones que tienen significado para los actores. El lenguaje de modelado que se utiliza es Unified Modelling Language (UML). Y presenta tres componentes principales que se utilizan como estándar para lograr la representación unificada de este tipo de diagramas, a saber:

- **Actor:** es una idealización de un rol que interactúa con el sistema o subsistema. Los actores pueden ser definidos en jerarquías de generalización, en las cuales la descripción de un actor abstracto es compartida y aumentada por la descripción de uno o más actores concretos. Un actor puede ser humano, un sistema informático o algún proceso ejecutable.
- **Caso de uso:** es el encargado de representar el sistema que mediante el intercambio de mensajes se comunica con los actores. El propósito de un caso de uso es definir una pieza de comportamiento coherente sin revelar la estructura interna del sistema. La definición de un caso de uso incluye todo el comportamiento que se le supone las secuencias principales, distintas variaciones del comportamiento normal y todas las condiciones de excepción que pueden darse con dicho comportamiento, junto con la respuesta deseada. Desde el punto de vista del usuario, éstas pueden ser situaciones anormales, pero desde el punto de vista del sistema, son variaciones adicionales que deben ser descritas y manejadas.
- **Relaciones de caso de uso:** son lo que permiten la conexión entre el caso de uso y los actores que interactúan con el mismo o entre los distintos casos de uso. Existen diferentes tipos de relaciones de caso de uso, a continuación, se mencionan en la Figura 7:

Relación	*Función*	*Notación*
asociación	La línea de comunicación entre un actor y un caso de uso en el que participa	———
extensión	La inserción de comportamiento adicional en un caso de uso base que no tiene conocimiento sobre él	«extend» →
generalización de casos de uso	Una relación entre un caso de uso general y un caso de uso más específico que hereda le añade propiedades	——▷
inclusión	La inserción de comportamiento adicional en un caso de uso base que describe explícitamente la inserción	«include» →

Figura 7: Relaciones de caso de uso. Extraído de [13] pág 71.

Seguidamente se podrá ver el diagrama de caso de uso realizado para la aplicación móvil de podcast.

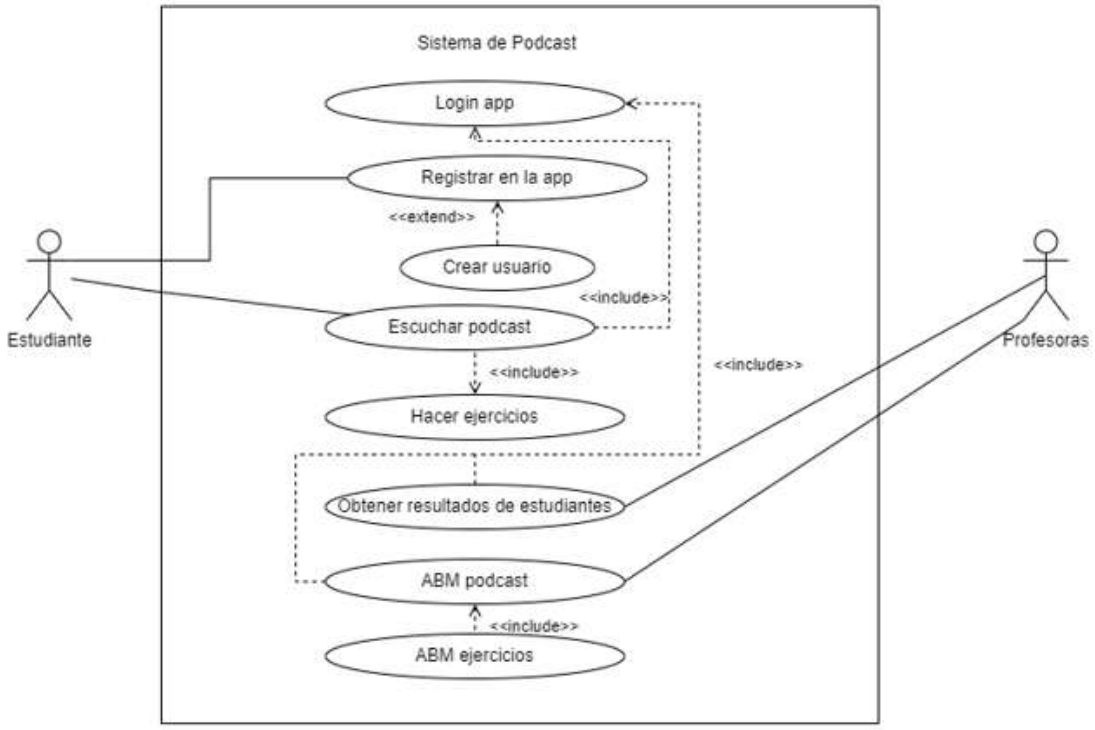


*Figura 8: Diagrama de caso de uso de **Podcast.ing**.*

4.2 Diseño de Interfaz de Usuario

Para elaborar el diseño de Interfaz de Usuario, se realizó una reunión con las profesoras de inglés donde se definieron los colores que debe tener la aplicación móvil y el icono que se va a utilizar para representar a la aplicación móvil junto con su nombre. A partir de eso, se crearon los wireframe [15] o prototipos de las distintas pantallas. Se denomina wireframe al modelo básico que ilustra la forma central y función que se encuentra en una sola pantalla de la página web o aplicación. La fidelidad de estos

wireframes aumentará en detalle a medida que se refinen. La colección de wireframes deben de dar una visión esquelética completa del producto en su totalidad.

Antes de comenzar la implementación se mostraron los wireframe a las profesoras para que los evalúen y en el caso de ser necesario, realizar cambios en los mismos. En las Figuras 9, 10 y 11 se puede observar algunos de los diseños realizados y los demás se encuentran en el Anexo 3.

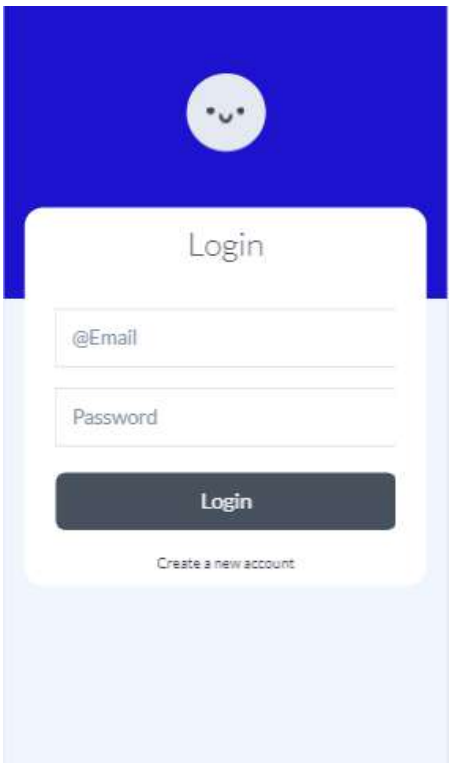


Figura 9: Wireframe de la pantalla de login.

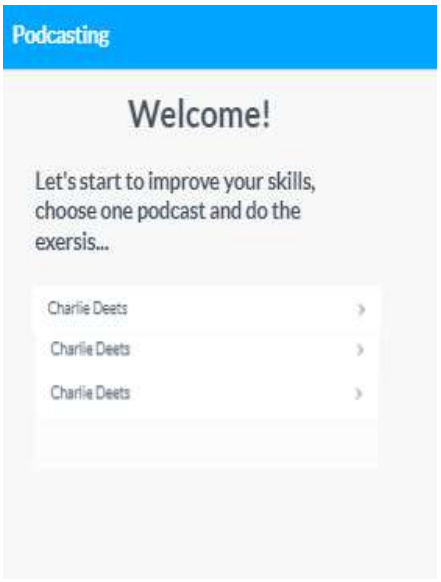


Figura 10: Wireframe de la pantalla principal de los alumnos.

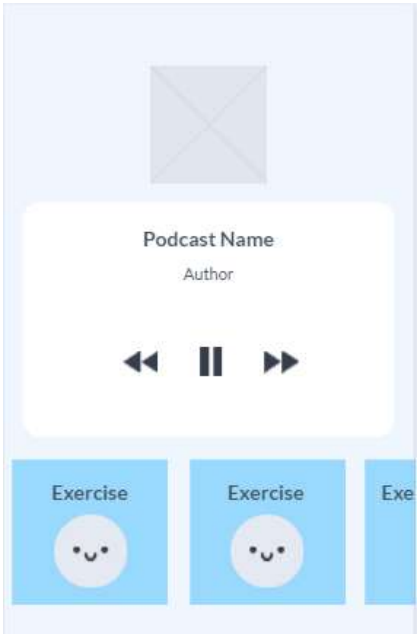


Figura 11: Wireframe de la pantalla del reproductor de podcast seleccionado.

En cuanto a las tecnologías utilizadas para llevar a cabo dichos diseños, se utilizó MarvelApp. Esta herramienta brinda plantillas las cuales permiten la creación de componentes de interfaz de usuario simples y personalizables. MarvelApp cuenta con una versión gratuita la cual posee suficientes componentes y características para hacer cualquier proyecto. Además, es de código abierto, y ofrece todo lo que se necesita para crear aplicaciones modernas y estéticas de manera sencilla.

4.3 Diseño de Gráfico de propiedades

El gráfico de propiedades [16] proporciona un punto de partida visual para el diseño de bases de datos NoSQL. Este tipo de gráfico consta de *nodos* (conceptos, objetos) y *bordes dirigidos* (relaciones) que conectan los nodos. Tanto los nodos como los bordes reciben una etiqueta y pueden tener propiedades. Las propiedades se dan como pares de atributo-valor siguiendo el patrón (*atributo*: valor) con los nombres de los atributos y los respectivos valores.

En la Figura 12 se puede observar el gráfico de propiedades realizado para ***Podcast.ing***, el cual va a ser la estructura de cómo se van a almacenar los datos en la base de datos en la nube NoSQL. La herramienta utilizada para poder realizar dicho gráfico es NoSQL DB Schema Modeling. Esta herramienta para representar una propiedad que es requerida, utiliza un asterisco y para indicar que debe ser única, una estrella.

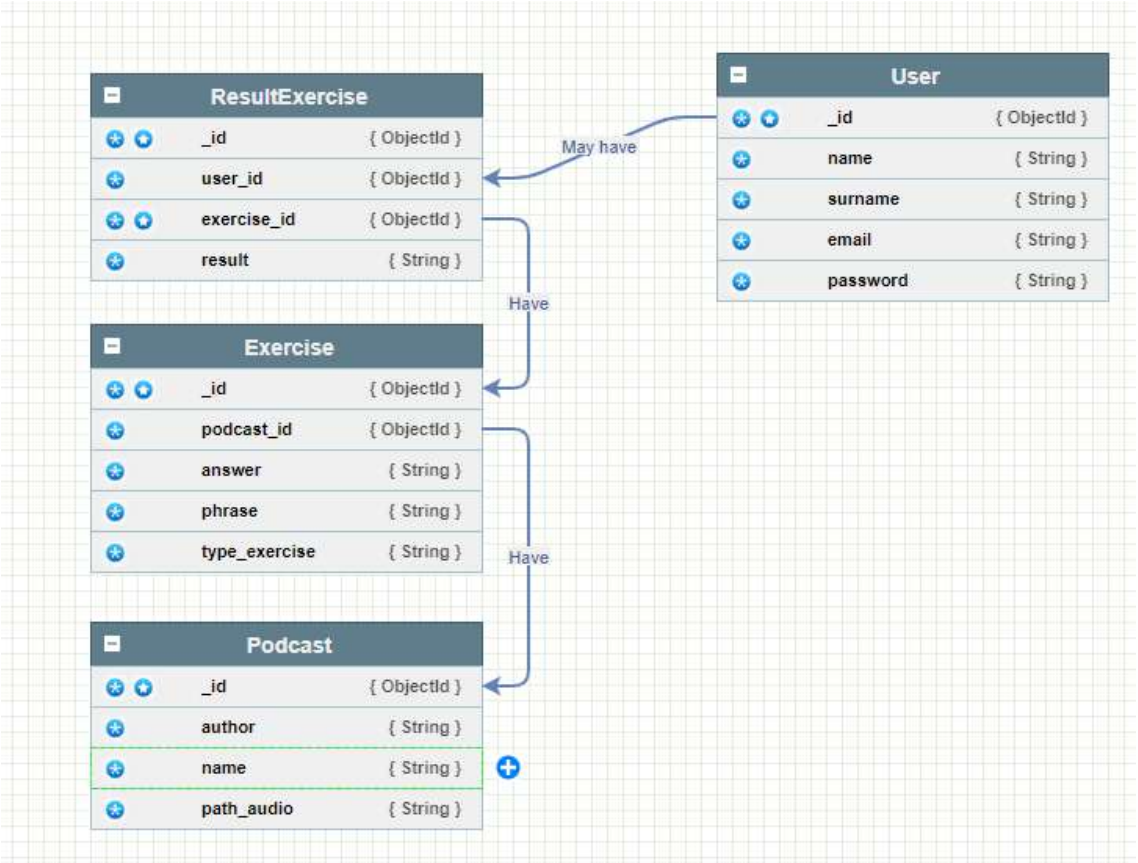


*Figura 12: Gráfico de propiedades de **Podcast.ing**.*

Capítulo 5: Desarrollo del proyecto

En este capítulo se realiza una explicación de cómo fue considerada y llevada a cabo la cuarta etapa dentro de la metodología elegida, la cual es codificar. Se va a describir lo que se realizó para la construcción de ***Podcast.ing***. Esto abarca desde la creación del código hasta la configuración del backend en Firebase. Cabe destacar que la codificación se fue realizando a lo largo de todo el proyecto, en cada una de las iteraciones se fue agregando funcionalidades y mejoras. Respecto a la configuración, se realizó al inicio del proyecto basándose en lo planificado y diseñado.

5.1 Configuración de Firebase

Las configuraciones del backend se comenzaron con la creación del proyecto de Firebase y posteriormente se realizaron las correspondientes con el realtime database, la autenticación y el storages. Estos tres últimos mencionados se pueden realizar en cualquier orden ya que una configuración no condiciona la otra.

Para comenzar a utilizar Firebase, lo que se debe hacer es crear y configurar un proyecto de este BaaS. Para eso se deben seguir los siguientes pasos:

- Crear un proyecto en Firebase
    1. Iniciar sesión en Firebase.
    2. En la consola, hacer clic en Agregar proyecto o Crear un proyecto (Figura 13).

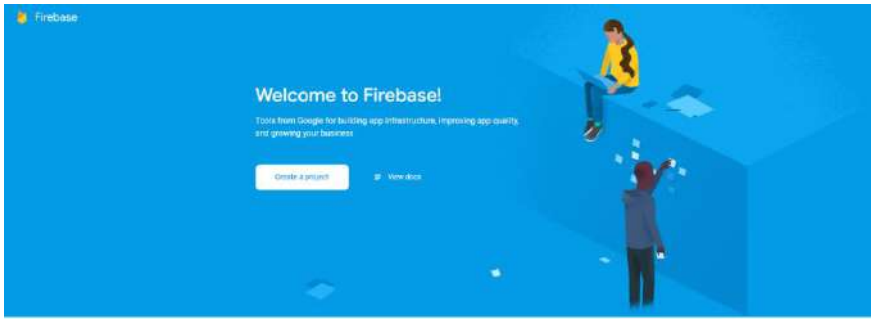


Figura 13: Crear proyecto Firebase.

    3. En el campo Nombre del proyecto, ingresar Firebase-Flutter-Codelab y luego hacer clic en Continuar (Figura 14).

Figura 14: Definir nombre proyecto Firebase.

4. Hacer clic en las opciones de creación de proyectos. Si se solicita, aceptar los términos de Firebase (Figura 15). A continuación, si es necesario configurar Google Analytics.

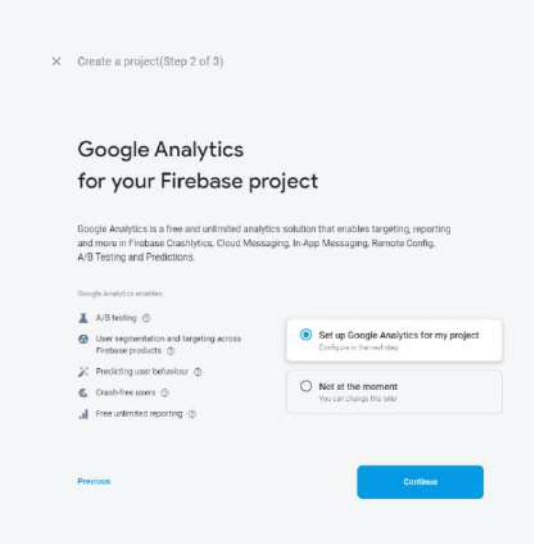


Figura 15: Definir utilización de Google Analytics.

5. Una vez que se ingresa al proyecto recién creado, configurar las apps, tanto Android como iOS, para eso se deben seguir los pasos que se muestran al seleccionar esa parte (Figura 16).

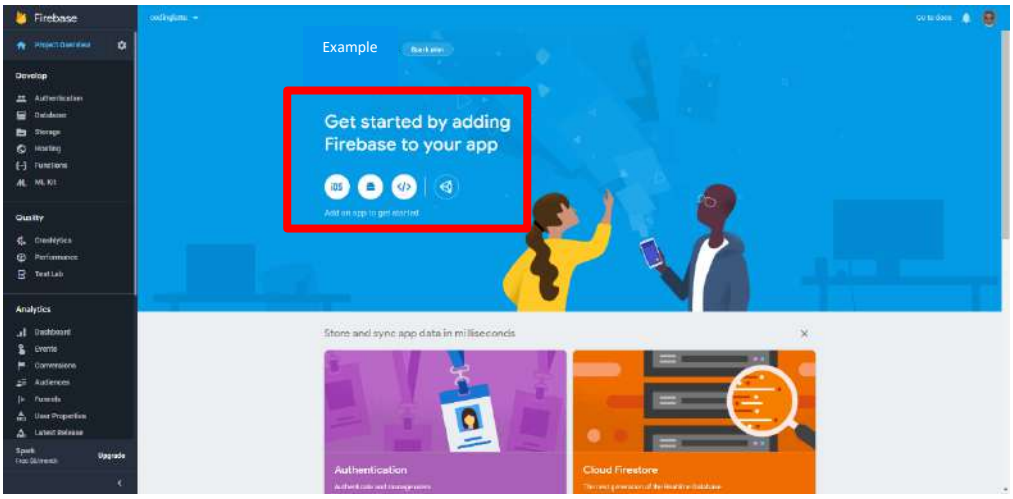


Figura 16: Configuración de apps en Firebase.

6. Instalar la dependencia de firebase_core dentro del proyecto en el archivo pubspec.yaml.
7. En el archivo main.dart, dentro de main() colocar WidgetsFlutterBinding.ensureInitialized() y Firebase.initializeApp().

Seguidamente, se da una explicación de cómo fueron configurados realtime database, el storage y autenticación [17].

5.1.1 Configuración de almacenamiento en la nube

Para configurar el almacenamiento en la nube los pasos a seguir son:

1. En el panel de navegación de Firebase console, seleccionar Almacenamiento y luego hacer clic en Comenzar (Figura 17).

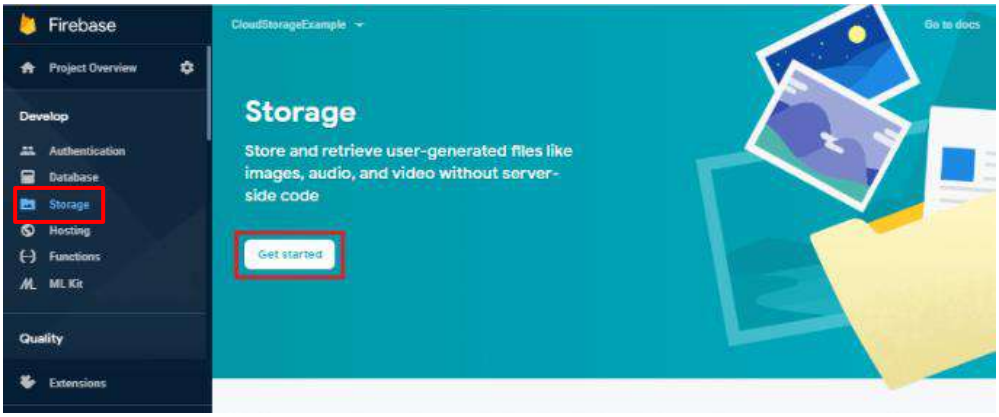


Figura 17: Seleccionar Storage.

2. Revisar los mensajes sobre cómo proteger los datos de Cloud Storage mediante reglas de seguridad (Figura 18). Durante el desarrollo, considerar configurar las reglas para el acceso público.

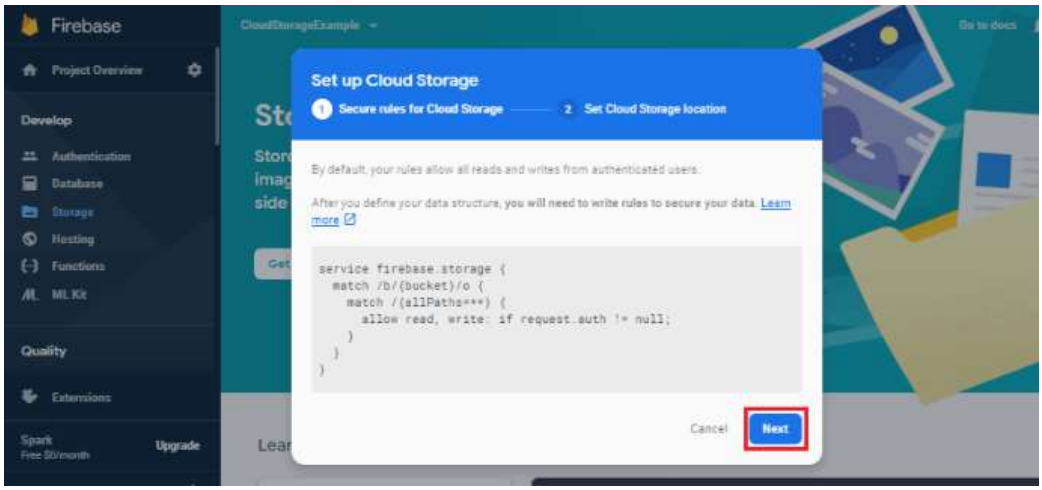


Figura 18: Configurar reglas de seguridad.

3. Seleccionar una ubicación para el depósito de Cloud Storage predeterminado (Figura 19). Esta configuración de ubicación es la ubicación de recursos predeterminada de Google Cloud Platform (GCP) del proyecto.
    a. Tener en cuenta que esta ubicación se usará para los servicios de GCP en el proyecto que requieren una configuración de ubicación, específicamente, su base de datos de Cloud Firestore y su aplicación App Engine (que es necesaria si usa Cloud Scheduler).
    b. Se puede seleccionar una ubicación distinta a la por defecto propuesta que se configuró durante la creación del proyecto o al configurar otro servicio que requiere una configuración de ubicación.

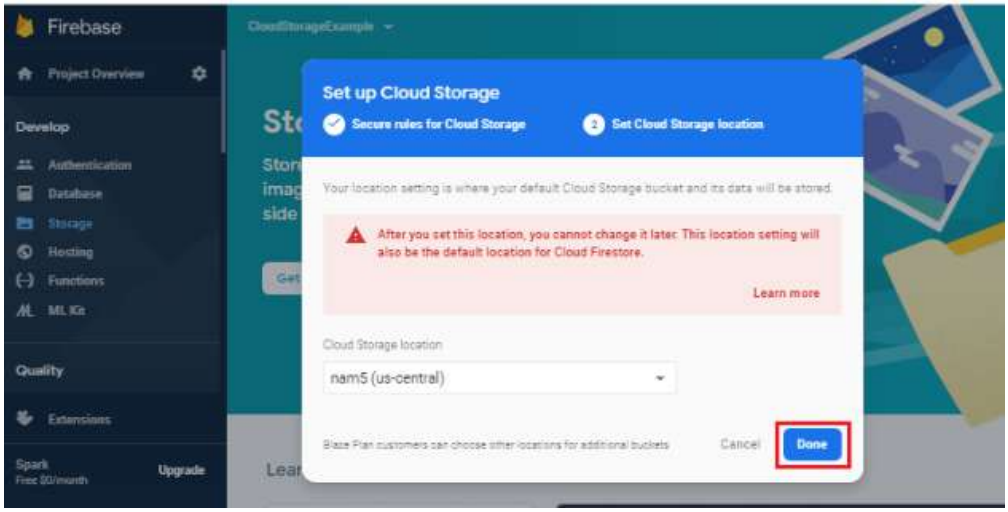


Figura 19: Seleccionar una ubicación del Cloud Storage.

4. Hacer clic en Listo.

A continuación, en el código 1 se visualizan las líneas de código necesarias para la configuración que permiten almacenar los audios de los podcast.

```
      final file = File(platformFile!.path!);

    // Create the file metadata
```

```
final metadata = SettableMetadata(contentType: "video/mp3");

// Create a reference to the Firebase Storage bucket
final storageRef = FirebaseStorage.instance.ref();

// Upload file and metadata
final uploadTask = storageRef
    .child(fileName.path)
    .putFile(file, metadata);

// Listen for state changes, errors, and completion of the upload.
uploadTask.snapshotEvents.listen((TaskSnapshot taskSnapshot) async {
  switch (taskSnapshot.state) {
    case TaskState.running:
      final progress =
          100.0*(taskSnapshot.bytesTransferred / taskSnapshot.totalBytes);
      print("Upload is $progress% complete.");
      break;
    case TaskState.paused:
      print("Upload is paused.");
      break;
    case TaskState.canceled:
      print("Upload was canceled");
      break;
    case TaskState.error:
      // Handle unsuccessful uploads
      break;
    case TaskState.success:
      // Handle successful uploads on complete
       print("Upload successful");
       final snapshot= await uploadTask.whenComplete(() {});
       final urlDownLoad= await snapshot.ref.getDownloadURL();
       print(urlDownLoad);

      break;
  }
});
```

Código 1: Extracto de código donde se ve la configuración para poder almacenar los audios.

5.1.2 Configuración de autenticación de usuario

Para configurar la autenticación se debe habilitar el inicio de sesión con correo electrónico/contraseña (Figura 20):

1. En la sección Autenticación de la consola Firebase (1), abrir la página Método de inicio de sesión (2).
2. En la página Método de inicio de sesión, habilitar el método de inicio de sesión con correo electrónico/contraseña (3 y 4) y hacer clic en Guardar (5).

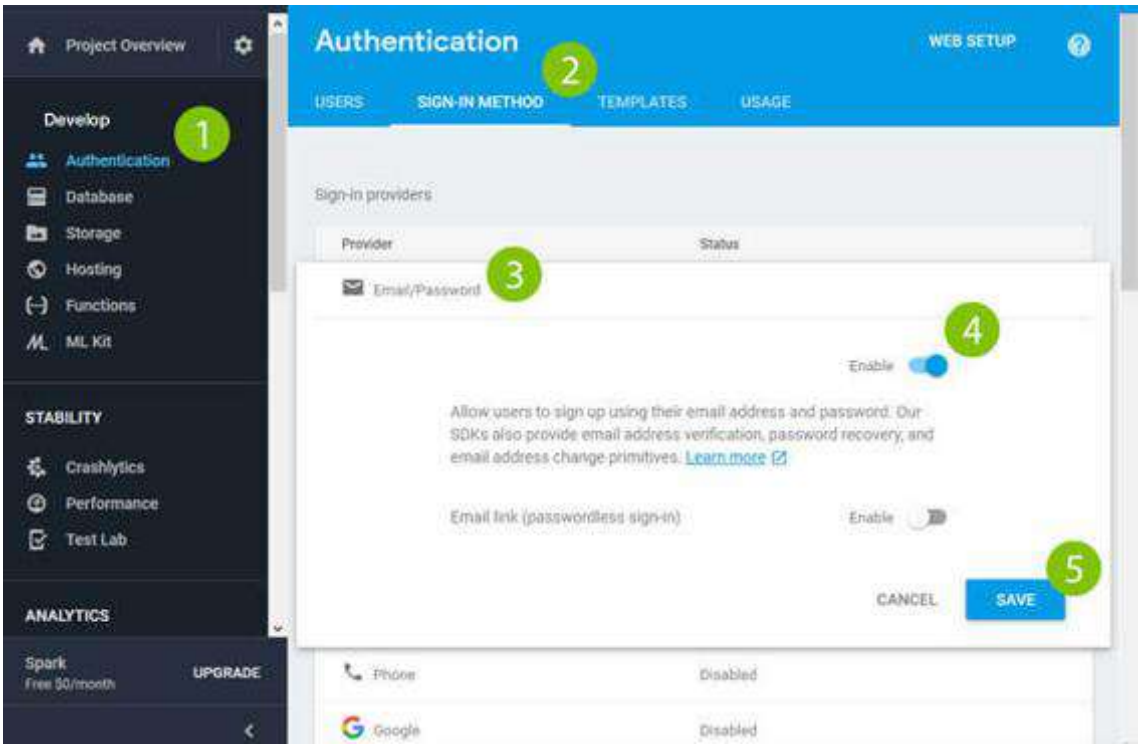


Figura 20: Configuración de autenticación.

A continuación, se muestra en el código 2 como se verifica la autenticación. En el caso que el usuario tenga cuenta se va a crear un token de acceso sino se muestra un error.

```
Future<String?> login(String email, String password)async{
    final Map<String,dynamic> authData ={
      'email':email,
      'password':password,
      'returnSecureToken':true
    };
    final url = Uri.https(_baseUrl,
'/v1/accounts:signInWithPassword',{'key':_firebaseToken});
    final resp = await http.post(url,body:json.encode(authData));
    final Map<String,dynamic> decodeResp = json.decode(resp.body);
    if(decodeResp.containsKey('idToken')){
      await storage.write(key: 'idToken', value:
decodeResp['idToken']);
      print('token ${decodeResp['idToken']}');
      await searchUserBD(email);
      return null;
    }else{
      return decodeResp['error']['message'];
    }
  }
```

Código 2: Extracto de código donde se realiza la verificación de la autenticación.

Como se puede observar en el código se crea un map con los datos para la autenticación y se agrega un parámetro (returnSecureToken) con el objetivo de que devuelva el token cuando sale bien la autenticación. En la siguiente línea se crea la URL con los datos que son requeridos para poder hacer una petición de autenticación. Seguidamente, se realiza la petición y con la respuesta que se brinda, se decodifica y si está correcto se guarda el token y se ingresa a ***Podcast.ing***. En caso contrario, sale un cartel con el error.

5.1.3 Configuración de realtime database

Para configurar la realtime database se deben seguir los siguientes pasos:

1. Ir a la sección realtime database de la consola de Firebase. Se pide seleccionar un proyecto de Firebase existente y seguir el flujo de trabajo de creación de la base de datos (Figura 21).

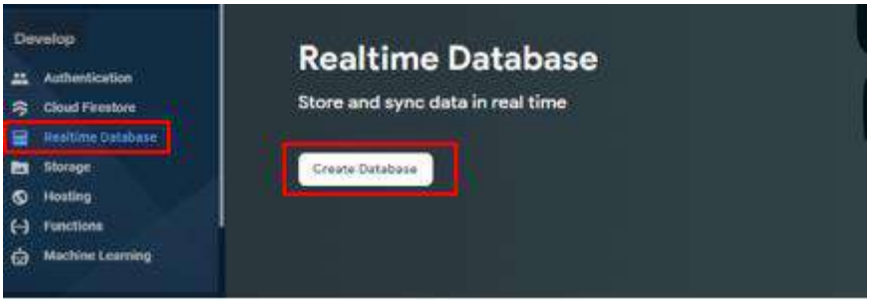


Figura 21: Seleccionar realtime database.

2. Elegir una región para la base de datos. Según la región elegida, el espacio de nombres de la base de datos tendrá el formato <databaseName>.firebaseio.com o <databaseName>.<region>.firebasedatabase .app (Figura 21).

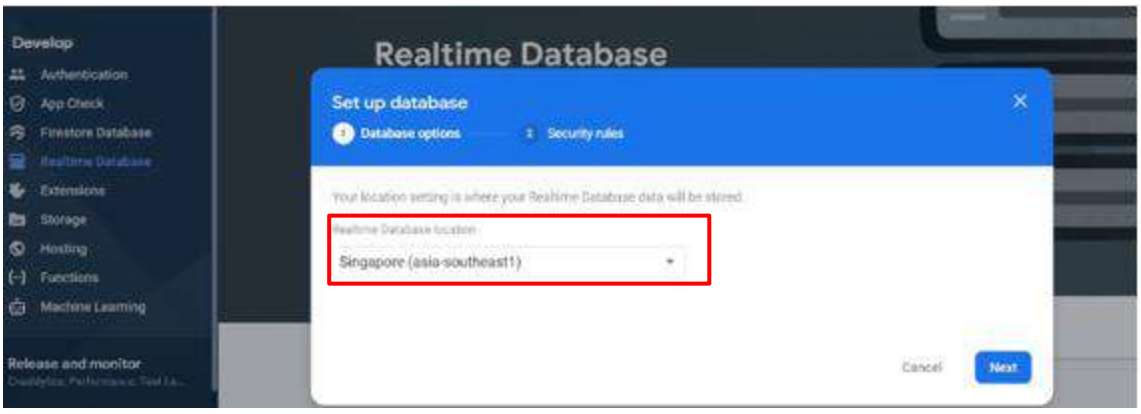


Figura 22: Seleccionar región de ubicación de la B.D.

3. Seleccionar un modo de inicio para sus reglas de seguridad (Figura 23).

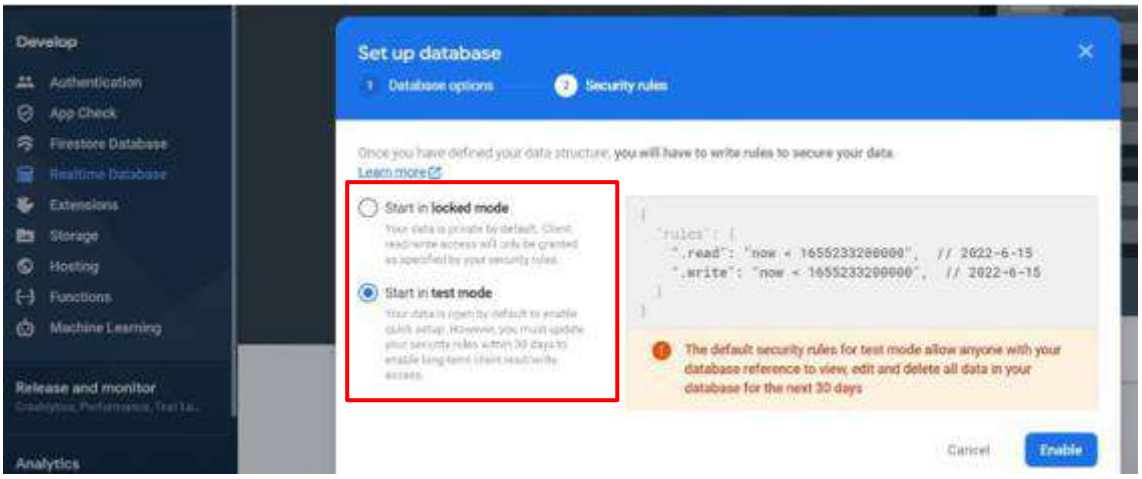


Figura 23: Seleccionar reglas de seguridad.

4. Hacer clic en Listo.

Posteriormente a eso, lo que se realizó fue crear la estructura de los elementos que se pueden ver en la Figura 12.

5.2 Construcción de código

Para esta sección se realiza una explicación de cómo se llevó a cabo la codificación del proyecto. Para esto se tuvo en consideración la documentación oficial de Flutter [18]. Flutter posee una estructura de carpetas y archivos que se crean al momento de construir un proyecto (ver Figura 24), a saber:

- Carpetas:
  - **idea**: usada para guardar la configuración del proyecto en curso. Esta carpeta es ignorada al momento de subir al repositorio Git los archivos.
  - **android**: donde se guarda todo lo necesario que se genera al crear una aplicación Android a partir del proyecto Flutter. Normalmente no será necesario modificar nada en esta carpeta.
  - **ios**: guarda todo lo necesario para crear una aplicación IOS a partir del proyecto Flutter. Al igual que en la carpeta android generalmente no será necesario modificar nada.
  - **build**: donde se guarda el código al compilar una aplicación Flutter, es decir, donde se guarda la aplicación android apk generada. No es necesario modificar nada en esta carpeta.
  - **lib**: donde se encuentran normalmente los archivos que conforman el código fuente de la aplicación Flutter. Se puede estructurar como si fuera un sistema de archivos y permite crear directorios y subdirectorios.
  - **test**: donde se guardan los archivos para realizar pruebas sobre la aplicación de Flutter.

- Archivos:
  - **.metadata**: administrado por Flutter automáticamente y se usa para controlar las propiedades del proyecto Flutter, actualizaciones, etc. Este archivo solo lo cambia Flutter y no debe ser modificado manualmente.
  - **.gitignore**: indica todos los archivos y directorios que deben ser ignorados en caso de subir a un repositorio Git el proyecto. El desarrollador puede

editarlo para agregar los archivos que no desea que sean guardados dentro del repositorio remoto.

- **.packages**: guarda metadatos del proyecto Flutter. Contiene la lista de dependencias del proyecto. Es generado automáticamente por el SDK de Flutter y no debe ser modificado.
- **.pubspec.lock**: indica cómo se construye el archivo pubspec.yaml. No debe editarse.
- **.pubspec.yaml** es un archivo que permite:
  - Establecer la configuración general del proyecto como nombre, descripción y versión del proyecto
  - Indicar las dependencias del proyecto (cuando se hace uso de una nueva funcionalidad, por ejemplo)
  - Indicar archivos de recursos, como imágenes, audios,etc.

  Hay que tener en consideración que este archivo tiene en cuenta la tabulación, es decir, es muy importante mantener el orden y espaciado de los elementos que se escriben ahí.

  Cuando se produce un cambio en este archivo es necesario ejecutar el comando: ***flutter pub get*** (o *flutter packages get*) (en casi todos los IDE´s aparecerá un mensaje indicándolo y ya lo ejecutará el propio IDE). Normalmente, para que se apliquen los cambios realizados en este archivo, se necesita hacer un full restart, parando la aplicación y volviéndola a ejecutar para que la reinstale.
- **nombreproyecto.iml**: este archivo siempre se nombra de acuerdo con el nombre del proyecto y contiene más configuraciones del proyecto Flutter. No debe modificarse.
- **README.md**: archivo en el que aparece la descripción del proyecto. Es el archivo que aparece en un alojamiento de Github en Internet.

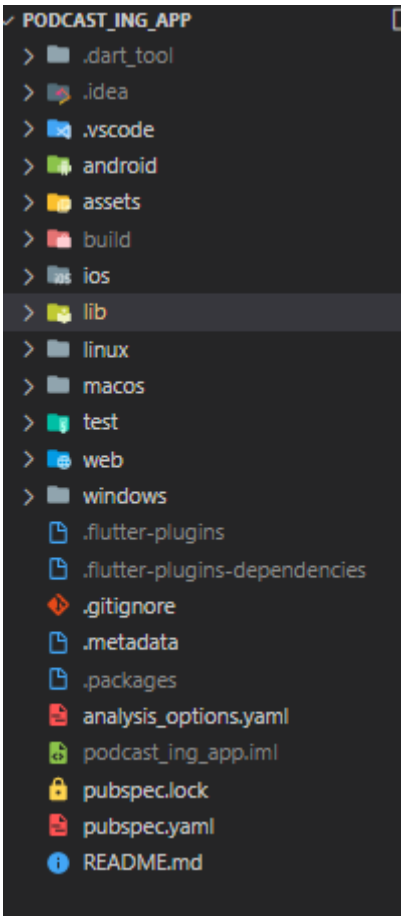


Figura 24: Estructura de carpetas del proyecto.

En la Figura 24 aparece una carpeta adicional a las anteriores mencionadas porque se agregó la carpeta assets, donde se almacenan las imágenes del logo de ***Podcast.ing*** y la utilizada cuando la aplicación móvil está a la espera de la carga de los datos. Dentro de la carpeta lib, el código fuente de la aplicación móvil se dividió en subcarpetas las cuales se podrán visualizar en la Figura 25. El contenido que poseen estas carpetas son:

- **models**: archivos que representan la estructura de los elementos que se usan en la aplicación, por ejemplo: podcast, ejercicios, notas, etc. Cada uno de estos archivos, además, contiene la especificación para transformar estos elementos a JSON y/o a map.
- **providers**: archivos encargados de mantener la persistencia de datos.
- **screens**: archivos que contienen los widgets encargados de cargar las pantallas de la aplicación las cuales son accedidas mediante routers.
- **services**: archivos encargados de permitir la conexión con el backend.
- **ui**: archivos que contienen los estilos personalizados creados para la aplicación, los cuales son utilizados por varios widgets.
- **widgets**: archivos que contienen los widgets que se reutilizan en varias screens.

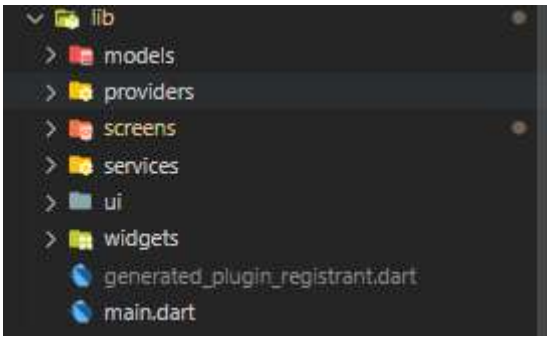


Figura 25: Estructura de carpetas del proyecto.

Antes de continuar a las siguientes subsecciones donde se explica cómo se crearon las vistas, proveedores y servicios dentro del proyecto desarrollado, se debe destacar que cuando se pensó en cómo construir el proyecto se tuvieron en consideración los widgets que provee Flutter.

Este framework tiene widgets statefull y stateless, la diferencia entre ambos consiste en que el primero mantiene dentro de widget la persistencia de los datos, en cambio el otro no. Una desventaja de que el widget sea statefull es el costo que genera debido que ante cualquier cambio este redibuja todo el widget, por lo que para la aplicación se decidió implementar todos los widgets de tipo stateless. Como consecuencia, se debió crear lo que se denominaron proveedores para poder mantener la persistencia de los datos.

Por otra parte, para separar la lógica de backend de la del frontend se utilizaron servicios; logrando de esta manera tener una mayor seguridad al no exponer directamente la lógica de las peticiones que se realizan a Firebase y cargar menos a las vistas dejando un código más simple.

5.2.1 Creación de vistas

El núcleo del mecanismo de diseño de Flutter son los widgets. Tanto los elementos visibles como las imágenes, los íconos y el texto, como también aquellos que no se ven (las filas, columnas,etc.) son widgets. El diseño de una aplicación se alcanza al componer widgets, esto es, construir widgets más complejos a partir de widgets simples. Flutter utiliza su propia biblioteca de widgets y la biblioteca Material que sigue los principios de Material Design. Dichos principios son [19]:

- El Material es la metáfora: se inspira en el mundo real y en sus texturas, incluyendo cómo el material refleja luz y proyecta sombras. Este material es una interpretación de medios como el papel y la tinta en el mundo digital.
- Es atrevido, gráfico e intencional: se guía por los métodos de diseño para impresos: tipografía, rejillas, espacio, escala, color e imágenes, para crear jerarquías, significado y enfoque que meta los usuarios en la experiencia.
- La animación tiene significado: la animación enfoca la atención del usuario y ayuda a mantener la continuidad, por medio de retroalimentación muy sutil y transiciones coherentes. Mientras los elementos de la interfaz aparecen en la pantalla, se transforman y reorganizan a través de la interacción, generando así nuevas transformaciones.

- Una base flexible: el sistema de diseño de Material Design fue creado para permitir que una marca pueda usarla como un medio de expresión. Se integra con una base de código hecho a la medida que permite la integración natural de componentes, plug-ins y elementos de diseño.
- Es multi-plataforma: mantiene el mismo diseño de interfaz en diferentes plataformas, utilizando componentes en Android, iOS, Flutter y para la web.

Por lo tanto, al diseñar la interfaz de usuario el desarrollador puede usar exclusivamente widgets de la biblioteca de Flutter, o de la biblioteca Material. A su vez, puede mezclar widgets de ambas bibliotecas o incluso personalizar widgets existentes o crear un conjunto de widgets propios personalizados. A continuación, se muestran algunos de los widgets más utilizados dentro de una aplicación de Flutter. Los cuales se van a dividir de acuerdo a la biblioteca que los contiene:

- Widgets estándar de Flutter:
  - Container: agrega relleno, márgenes, bordes, color de fondo u otras decoraciones a un widget.
  - GridView: presenta los widgets como una cuadrícula desplazable.
  - ListView: presenta los widgets como una lista desplazable.
  - Stack: superpone un widget encima de otro.
- Widgets de Material:
  - Card: organiza la información relacionada en un cuadro con esquinas redondeadas y una sombra paralela.
  - ListTile: organiza hasta 3 líneas de texto e íconos opcionales iniciales y finales en una fila.

A continuación, el Código 3 muestra la codificación de la pantalla de inicio cuando ingresa un alumno. En el código se podrá visualizar los widgets que se implementaron y posteriormente se explicará dichos widgets junto con un árbol de widgets para poder comprender su anidamiento.

```
class HomeScreen extends StatelessWidget{
  HomeScreen({Key? key}) : super(key: key);
 @override
 Widget build(BuildContext context) {
   final podcastService= Provider.of<PodcastService>(context);
   final authService= Provider.of<AuthService>(context,listen: false);
   if(podcastService.isLoading){
     return const LoadingScreen();
   }
   return Scaffold(
       appBar: AppBar(
       title: const Text('Podcast.ing'),
       automaticallyImplyLeading: false,
       actions:[IconButton(
         icon: const Icon(Icons.login_outlined),
         onPressed:(() {
           authService.logout();
           Navigator.pushReplacementNamed(context, 'login');
         })
       )]
```

```
    ),
    body:SingleChildScrollView(
      child: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 30, vertical: 20),
        child: Column(
          children:  [
            const Text('Welcome!',style: TextStyle(fontSize: 30,fontWeight: FontWeight.bold  )),
            const SizedBox(height:20),
            const Text("Let's start to improve your skills, choose one podcast and do the
                        exercises...",style: TextStyle(fontSize: 20)),
            const SizedBox(height: 30,),
            Container(
              decoration: _listDecoration(),
              width: double.infinity,
              padding: const EdgeInsets.all(15.0),

              child: _podcastList(podcast:podcastService.podcasts),
            ),
          ],),
      ),
    )
  );
 }
 BoxDecoration _listDecoration() {
   return BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.circular(25),
              boxShadow: const [
                BoxShadow(
                  color: Colors.black12,
                  blurRadius: 15,
                  offset: Offset(0,5)
                )
              ]
          );
 }
}
class _podcastList extends StatelessWidget {
 final List<Podcast> podcast;
 const _podcastList({Key? key, required this.podcast}) : super(key: key);
 @override
 Widget build(BuildContext context) {
   final podcastService = Provider.of<PodcastService>(context);
   return SizedBox(
     height: MediaQuery.of(context).size.height*0.1*podcast.length,
     child: ListView.separated(
           itemBuilder: (context, index) =>
             ListTile(
               title: Row(
                 children: [
```

```
71                    Text(podcastService.podcasts[index].name,style:TextStyle(fontSize:20,
72                         fontWeight: FontWeight.bold  )),
73                    const Text(' - ',style: TextStyle(fontSize: 20,fontWeight: FontWeight.w700 )),
74                    Text(podcastService.podcasts[index].author,style: TextStyle(fontSize: 20  )),
75                  ],
76                ),
77              onTap: () {
78                  podcastService.selectedPodcast= podcast[index].copy();
79                  Navigator.pushNamed(context, 'podcastPlayer');
80
81                },
82              ),
83            separatorBuilder: ( _ , __ ) => const Divider() ,
84            itemCount: podcastService.podcasts.length
85            )
86        );
87   }
88 }
```

Codigo 3: Código de la pantalla de inicio cuando ingresa un alumno.

Se comienza con un método build() (línea 4), el cual se encarga de instanciar y devolver un widget. Dentro de este método hay una condición If (línea 7) la cual decide el widget que se va a retornar de acuerdo al valor de la variable *podcastService.isLoading*. Si es verdadero significa que está cargando los datos de los podcast y se va a visualizar el widget que se encuentra en una carpeta de utilización común. Si el valor de esa variable es falso, se podrá ver el widget propio de la pantalla de inicio. Los widgets que se encuentran implementados se muestran a continuación donde se los clasificó en los que son visibles (Figura 26) y los que no (Figura 27).

- Visibles:
  - **Appbar**: consiste en una barra de herramientas y potencialmente otros widgets, la cual se coloca en la parte superior de la pantalla de manera fija.
  - **Text**: Permite ingresar texto y tiene propiedades para darle estilo.
  - **IconButton**: permite ingresar dentro de un botón un icono y definir una acción cuando este se presiona.
  - **ListView**: permite crear una lista desplazable de widgets dispuestos linealmente.

Figura 26: Widgets visibles.

- No visibles:
  - **Scaffold**: implementa la estructura de diseño visual básica.
  - **SingleChildScrollView**: es un box en el que se puede hacer scroll con un solo widget.
  - **SizeBox**: es una caja con un tamaño específico de ancho y alto.
  - **Padding**: le da a los widgets que contiene una separación especificada alrededor.
  - **Column** y **Row**: alinea a los widgets que tiene definidos dentro de su propiedad child, tanto vertical (column) como horizontalmente (row).
  - **BoxDecoration**: permite dar estilo a un widgets
  - **Container**: es un contenedor que define como se muestran, la posición y el tamaño de los widgets definidos dentro de su propiedad child.

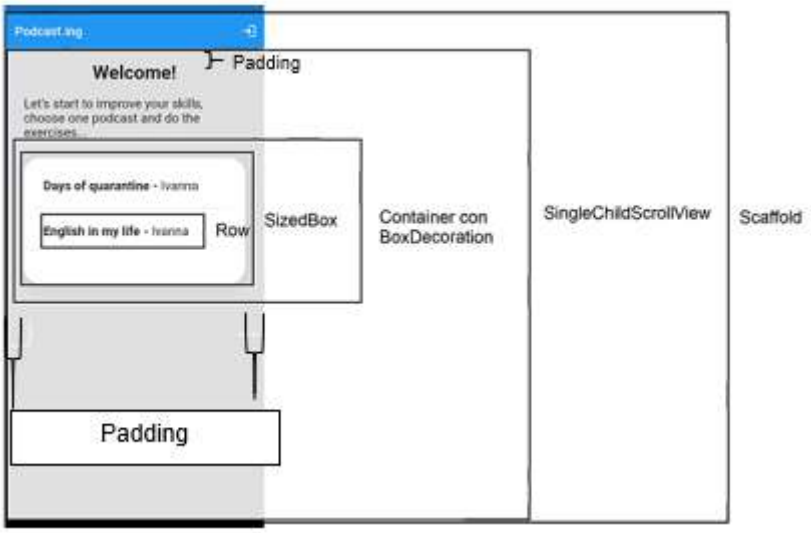


Figura 27: Widgets no visibles.

A continuación en la Figura 28 se puede apreciar el árbol de widgets que compone la pantalla codificada en el código 3.

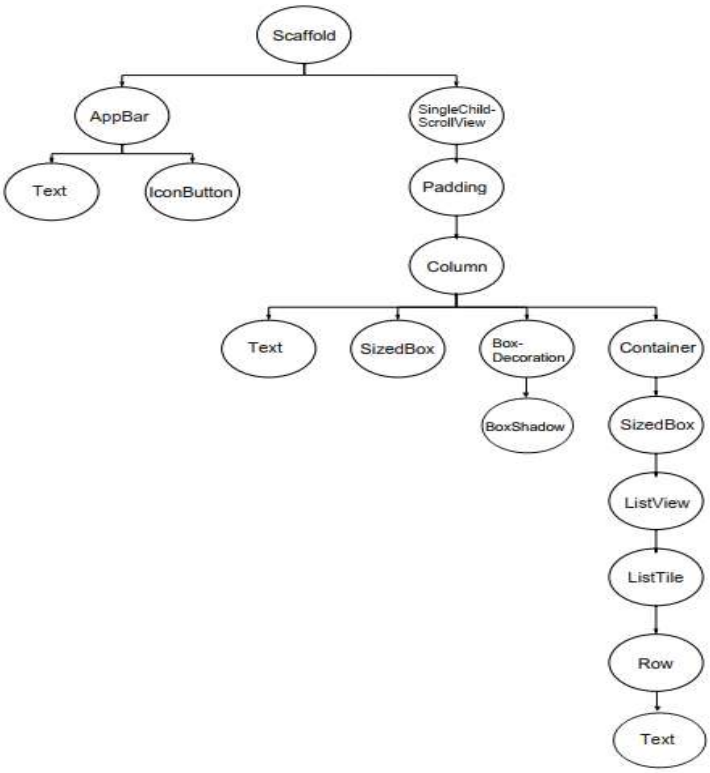


Figura 28: Árbol de widgets.

5.2.2 Creación de proveedores

Para poder implementar proveedores dentro de la aplicación, se tuvo que incorporar una dependencia denominada “provider”. Dicha dependencia se agregó dentro del archivo pubspec.yaml, como se puede ver en el Código 4.

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^1.24.0
  cupertino_icons: ^1.0.2
  provider: ^5.0.0
  audioplayers: ^1.0.1
  http: ^0.13.5
  file_picker: ^5.2.0+1
  firebase_storage: ^10.3.10
  flutter_secure_storage: ^6.0.0
```

Código 4: Extracto del código de dependencia del archivo pubspec.yaml.

Con la utilización de provider y de la clase ChangeNotifier que se encuentra incluida en el SDK de Flutter que proporciona notificaciones de cambios, se puede encapsular el estado de la aplicación.

En el desarrollo de la aplicación se crearon varios proveedores para poder mantener los datos actualizados, algunos de ellos son: podcast_provider, exercise_provider y login_provider. A continuación, en el Código 5 se presenta un ejemplo de uno de los proveedores creados.

```dart
import 'package:flutter/material.dart';


class LoginFormProvider extends ChangeNotifier{
  GlobalKey<FormState> formKey= GlobalKey<FormState>();
  String email    = '';
  String password = '';
  String name='';
  String surname='';
  late bool _isLoading = false;
  bool get isLoading => _isLoading;
  set isLoading(bool value){
    _isLoading = value;
    notifyListeners();
  }
  bool isValidFrom(){
    return formKey.currentState?.validate()??false;
  }
}
```

Código 5: Código del proveedor del login_form_provider.

El objetivo de este proveedor es validar los datos ingresados y almacenar el usuario que se loguea dentro de la aplicación, para permitirle indicar a todos los widgets que se está inicializando.

5.2.3 Creación de servicios

Para permitir la conexión de la aplicación con Firebase, se crearon servicios que por medio de API REST se conectan para poder realizar las peticiones con el objetivo de obtener datos, crear nuevos datos, actualizarlos o eliminarlos. Los tipos de petición pueden ser: GET, POST, PUT, PATCH y DELETE.

A continuación, se muestra en el Código 6 una sección de exercise_service, donde se especifica cómo se realiza la creación en este caso de un ejercicio dentro del realtime database de Firebase.

```dart
Future createExercise(String phrase, String answer, String idPodcast, int type) async{
   final url = Uri.https(_baseUrl,'Exercise.json',{'auth':await storage.read(key: 'idToken')??''});
   Exercise exercise=Exercise(
       idPodcast:idPodcast,
       phrase:phrase,
       typeExercise:type.toString(),
```

```
7         answer:answer,
8       );
9     final resp = await http.post(url,body: exercise.toJson());
10     final decodeData = json.decode(resp.body);
11     exercise.id= decodeData['name'];
12     exercises.add(exercise);
13     return exercise.id;
14   }
```

Código 6: Extracto del código de la creación de ejercicio que se encuentra en exercise_service.

Si se observa en el Código 6, se puede distinguir que en la línea 2 se establece la URL al que se va a efectuar la petición. Debido a que el usuario debe estar autenticado para poder ingresar a la aplicación se le debe enviar dentro del URL el token que se crea cuando el usuario se loguea, esto se debe a que existe dentro de Firebase en la realtime database una regla que permite que solo los usuarios autenticados puedan realizar peticiones. Desde la línea 3 hasta la 8 se construye el elemento que se va a enviar dentro del body de la petición. Analizando ese elemento creado, se puede distinguir que es de tipo Exercise, dicho tipo se encuentra definido dentro del modelo que se creó en la aplicación.

Posteriormente en la línea 9, se realiza la petición POST con la URL y el ejercicio antes definido. Como respuesta a esta petición se devuelve un JSON el cual en la línea 10 se decodifica para obtener lo que viene con el nombre ‘name’. Este dato contiene el identificador del ejercicio que se acaba de crear, el cual se le asigna al ejercicio anteriormente definido para la petición. En la línea 12, se agrega ese ejercicio dentro del array de ejercicios que se encuentran almacenados en el servicio.

Para concluir, si bien se mostró una petición POST, en general, para realizar otro tipo de petición se deben seguir lo mismos pasos:

- Definir el URL con los parámetros que son necesarios.
- Definir el elemento o los elementos que van en el body de la petición.
- Realizar la petición.
- Decodificar el JSON que viene como respuesta.
- Agregar o modificar el array de elementos que se encuentra dentro del servicio.

Como resumen de lo codificado se puede mencionar que, el proyecto completo cuenta con seis archivos de servicios, diez archivos de proveedores y catorce archivos de vistas. Donde las vistas son las que realizan el llamado a los métodos que tienen los servicios con el objetivo de traer información o enviar información para la creación o edición o eliminación de algún elemento como por ejemplo podcast, exercise, etc. Y además, utilizan a los proveedores para poder mantener la persistencia de los datos.

Capítulo 6: Test del proyecto

En este capítulo se realiza una explicación de cómo fue considerada y llevada a cabo la penúltima etapa dentro de la metodología elegida, la cual es probar. En particular se va a hablar del tipo de testing que se realizó en la aplicación móvil, denominado test de caja negra.

6.1 Test de caja negra

El testing [20] abarca el proceso de ejecutar un programa con la intención de encontrar errores. El objetivo final del testing consiste en evaluar la calidad del producto y en caso de hallar errores poder corregirlos. Este proceso le da mayor valor al producto terminado. En particular el tipo de testing denominado test de caja negra, consiste en no tener en consideración el comportamiento interno y la estructura del programa. En su lugar, se concentra en encontrar circunstancias en las que el programa no se comporte de acuerdo con sus especificaciones.

En este enfoque, los datos de prueba se derivan únicamente de las especificaciones (es decir, sin aprovechar el conocimiento de la estructura interna del programa). Este enfoque se sustenta en realizar pruebas de entrada exhaustivas, haciendo uso de todas las posibles condiciones de entrada por caso de prueba. Dado que el programa es una caja negra, la única manera de estar seguro de detectar la presencia de fallas consiste probando cada condición de entrada.

Para llevar a cabo el testeo de ***Podcast.ing***, se realizaron pruebas por cada una de las funcionalidades desarrolladas por Sprint. Se crearon casos de prueba por cada funcionalidad a testear, utilizando la plantilla que se presenta a continuación en la Tabla 13:

Nombre del requerimiento funcional: nombre del requerimiento que se evaluará en el test.	**Número**: identificador del test.
Descripción: definición del requerimiento a ser testeado.	
Prerrequisitos: condiciones previas que se deben cumplir para poder ejecutar el test.	
Pasos: indica cómo se debe proceder para llevar a cabo el test.	
Resultado esperado: es lo que se espera que haga la aplicación móvil.	
Resultado obtenido: este puede coincidir con lo que se espera o puede que surja un error.	

Tabla 13: Plantilla de casos de prueba.

El diseño de los casos de prueba estuvo a cargo de la desarrolladora pero la ejecución fue realizada por el equipo docente de inglés. Para ello, cada caso de prueba fue asignado a una docente, que cumplió el rol de profesora y el rol del alumno. A partir de los resultados obtenidos, se realizaron las modificaciones necesarias para solucionar los problemas identificados. A continuación, se pueden visualizar en las Tablas 14, 15 y 16 tres ejemplos de casos de pruebas junto con los resultados obtenidos, siendo dos para

el rol de estudiante y otra para el rol de profesor. El resto de los casos de prueba, se detallan en el Anexo 4.

Loguearse aplicación móvil (profesor/alumno)	**Número**: 01
Descripción: un usuario quiere acceder a la página principal de la aplicación móvil, pasando por login.	
Prerrequisitos: debe tener una cuenta dentro de la aplicación móvil.	
Pasos: 1. Ingresar a la aplicación con email y contraseña. Los valores utilizados fueron: – Correo: "teacher@gmail.com" – Contraseña: 12345678	
Resultado esperado: pasar a la siguiente pantalla.	
Resultado obtenido: Correcto 05/10/22 Realizado por Raquel Ramos	

Tabla 14: Casos de prueba de logueo.

Registrarse en la aplicación móvil	**Número**: 02
Descripción: un alumno desea ingresar por primera vez a la aplicación móvil por lo que debe crearse una cuenta.	
Prerrequisitos: -	
Pasos: 1. Seleccionar "Create a new account". 2. Ingresar los datos solicitados: nombre, apellido, email y contraseña. Los valores utilizados fueron: – Nombre: "ivanna" – Apellido: "Jutterpeker" – Email: "ivannaflorenciajutterpeker@gmail.com" – Contraseña:123456789 2. Ingresar a la pantalla principal.	
Resultado esperado: ver listado de podcast.	
Resultado obtenido: Correcto 16/11/22 Realizado por Raquel Ramos	

Tabla 15: Casos de prueba de registrarse a la aplicación móvil.

Visualizar notas de los estudiantes	**Número**: 12

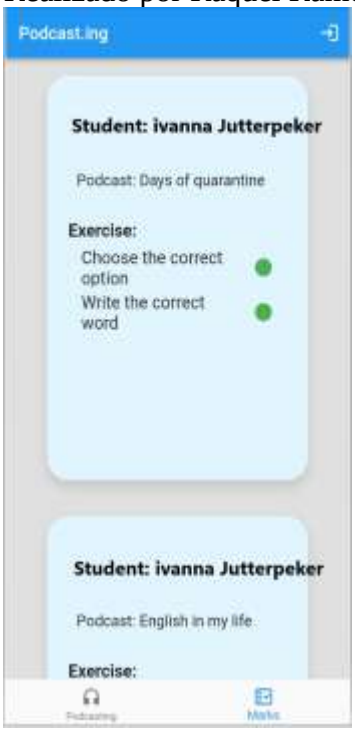
Descripción: una profesora desea ver las notas que tienen los estudiantes que hacen los ejercicios.
Prerrequisitos: debe encontrarse logueado dentro de la aplicación móvil.
Pasos: 1. Seleccionar el botón que representa los resultados de los alumnos. 2. Se puede ver las notas de los alumnos.
Resultado esperado: ver listado de notas de los alumnos.
Resultado obtenido: Correcto Error: el texto del nombre del estudiante se sale de la card. 02/11/22 Realizado por Raquel Ramos 

Tabla 16: Casos de prueba de visualizar notas de los estudiantes.

En base a todos los casos de pruebas que se realizaron, sobre un total de 15 se detectaron tres errores. Luego de la corrección se volvieron a ejecutar los test (re-testing) y no se detectaron nuevos errores. En la Tabla 17 se podrá ver el caso de prueba "visualizar notas de los estudiantes", está vez ya con las correcciones realizadas que permitieron eliminar el error identificado del test.

Visualizar notas de los estudiantes	**Número**: 12
Descripción: una profesora desea ver las notas que tienen los estudiantes que hacen los ejercicios	

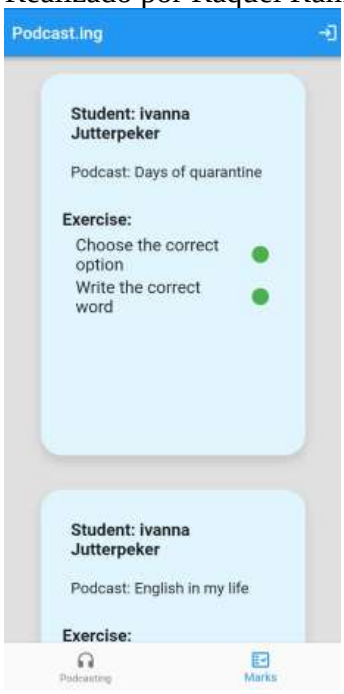
Prerrequisitos: debe encontrarse logueado dentro de la aplicación móvil
Pasos: 1. Seleccionar el botón que representa los resultados de los alumnos 2. Se puede ver las notas de los alumnos
Resultado esperado: ver listado de notas de los alumnos
Resultado obtenido: Correcto 03/11/22 Realizado por Raquel Ramos 

Tabla 17: Casos de prueba de visualizar notas de los estudiantes (corrección).

Para concluir con esta sección, cabe destacar que las pruebas realizadas fueron exitosas, ya que en lo funcional no se detectó ninguna falla pero en lo referente a visualización de las vistas si se detectaron algunos errores. Sin embargo, el hecho de realizar los test permitió identificar errores que no se habían detectado previamente. Dichos errores son de textos dentro de la aplicación que no quedan contenidos dentro del widget superior, esto llevó a que se realicen cambios. Como consecuencia de eso, se debieron crear tareas de corrección de error que no demandaron más de 4 hs cada una y no afectaron en gran medida al Sprint.

Capítulo 7: Despliegue

En este capítulo se hace referencia a la última etapa de la metodología elegida, la cual es el despliegue. Para ello, se muestran las pantallas del producto finalizado. Cabe destacar que esta etapa se fue realizando al finalizar cada uno de los Sprints planteados.

En primer lugar, se van a mostrar las vistas del login y registrarse junto con los errores de los mismos, en las Figuras 29, 30, 31, 32, 33 y 34.

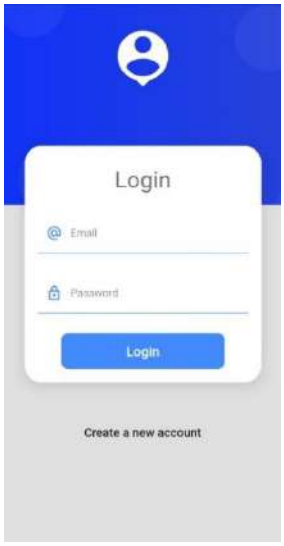


Figura 29: Vista del login.

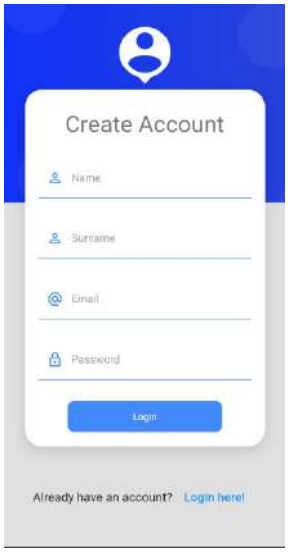


Figura 30: Vista de registrarse.

Figura 31: Vista de login con mensaje de error de dato ingresado.

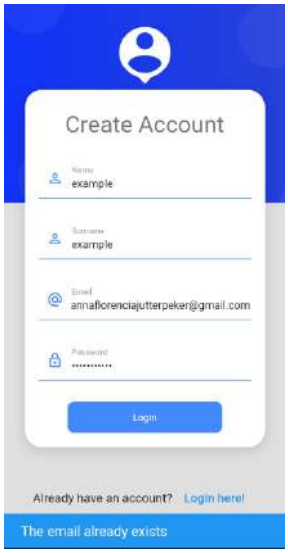


Figura 32: Vista de registrarse con mensaje de error de que ya existe el usuario.

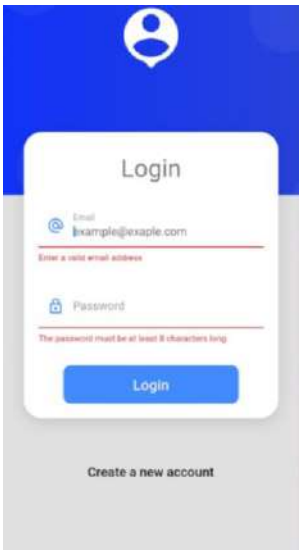


Figura 33: Vista de login con mensaje de error de que se debe completar los campos.

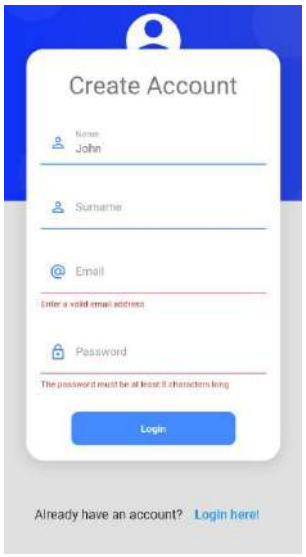


Figura 34: Vista de registrarse con mensaje de error de que se debe completar los campos

Ahora se muestran las pantallas de las vistas de los estudiantes en las Figuras 35, 36, 37, 38, 39, 40 y 41.

Figura 35: Vista de la página principal del estudiante.

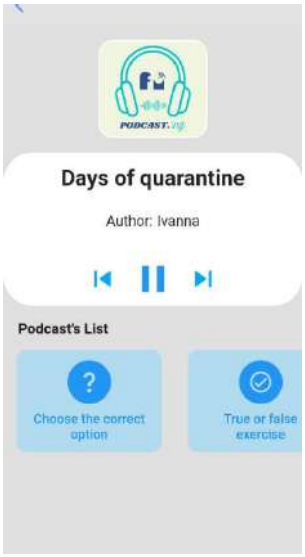


Figura 36: Vista del reproductor del podcast seleccionado.

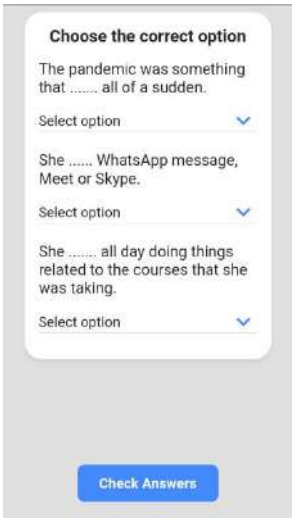


Figura 37: Vista del ejercicio "Choose the correct option".

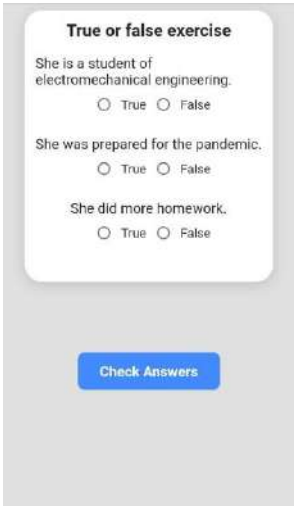


Figura 38: Vista del ejercicio "True or false exercise".

Figura 39: Vista del ejercicio "Write the correct word".

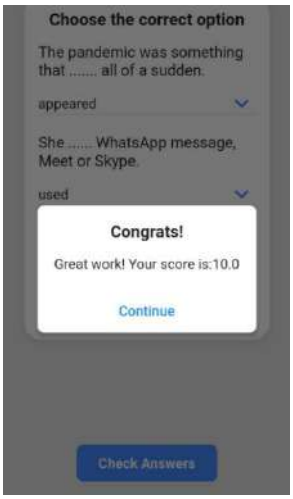


Figura 40: Vista del resultado del ejercicio resuelto (mayor a 6).

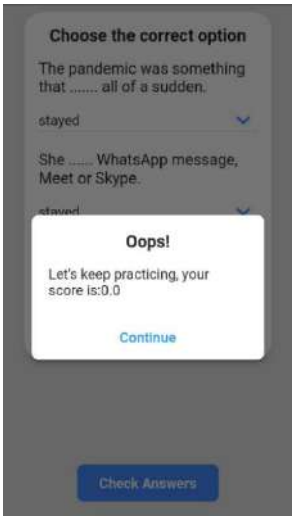


Figura 41: Vista del resultado del ejercicio resuelto (menor a 6).

Seguidamente se muestran las pantallas de las vistas de las profesoras en las Figuras 42, 43, 44, 45, 46, 47 y 48.

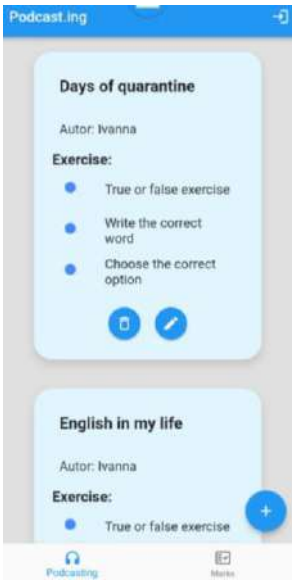


Figura 42: Vista de la página principal de las profesoras (listado de podcast).

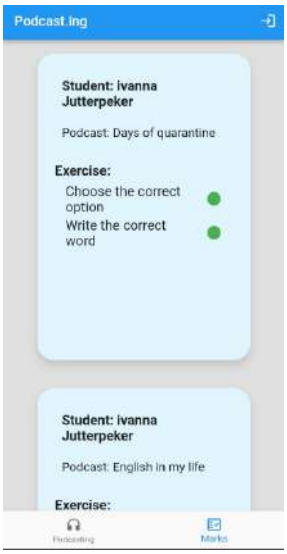


Figura 43: Vista de la página principal de las profesoras (listado de resultados de estudiantes).

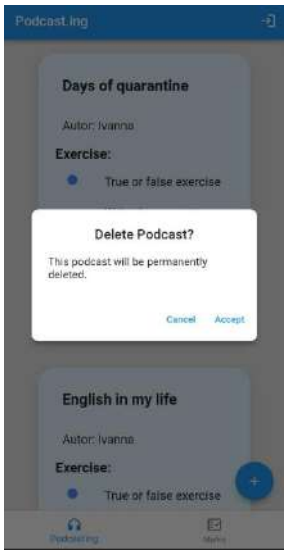


Figura 44: Vista de confirmación para eliminar podcast.

Figura 45: Vista de editar datos de podcast.

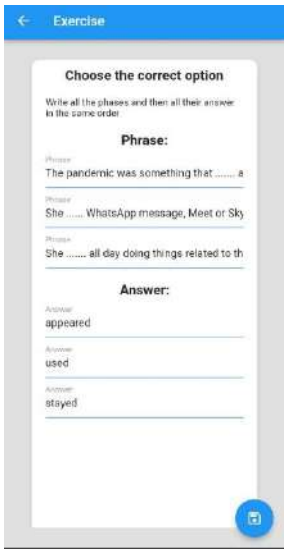


Figura 46: Vista de editar datos de ejercicio.

Figura 47: Vista de crear podcast.

Figura 48: Vista de crear ejercicio.

Se puede destacar que las pantallas que se desarrollaron cumplen con el diseño de interfaz de usuario que se realizó y que fue explicado en la sección 4.2.

Para concluir este capítulo, se debe agregar que esta etapa también consiste en desplegar la aplicación en plataformas como Play Store o App Store. Debido a que la aplicación va a ser de uso únicamente académico para el área de inglés dentro de la Facultad de Ingeniería, no se subió ***Podcast.ing*** a ningún Store. Ya que no sería necesario porque se le puede brindar a los alumnos el archivo ejecutable que contiene todos los datos que se necesitan para instalar y hacer funcionar la aplicación. Además, se debe señalar que para poder tener la aplicación en dichas plataformas se debe pagar 25 USD por la subscripción en Play Store o 99 USD/anual en App Store (fecha actualizada al día 16/12/22).

Capítulo 8: Conclusiones y Trabajo Futuro

En este capítulo se documentan los resultados y las conclusiones obtenidas, seguidamente se brinda una apreciación personal, para finalizar con posibles trabajos a futuro que se pueden realizar para potenciar la aplicación móvil ***Podcast.ing***.

8.1 Resultados y Conclusiones

El objetivo general de este trabajo implicaba diseñar, desarrollar, testear y poner en producción una aplicación móvil para el área de inglés de la Facultad de Ingeniería de la UNLPam. Objetivo que fue cumplido aplicando la metodología ágil Scrumban, logrando iteración a iteración un entregable que a medida que pasaban los Sprint iba adquiriendo más funcionalidades y mejoras.

Al mismo tiempo que se avanzaba se fueron encontrando errores y realizando correcciones, lo que llevó a modificar lo planificado para alcanzar un producto 100% funcional y sin errores.

Por lo tanto, las docentes del área de inglés poseen una aplicación móvil adaptada a los contenidos que ellas requieren para mejorar las macro-habilidades de los alumnos en la exposición oral y comprensión auditiva.

8.2 Apreciaciones personales

Hablando desde una perspectiva personal, el desarrollo de ***Podcast.ing*** era un desafío enorme debido a que se tuvo que aprender un framework y un backend diferente a lo que se conocía, en un plazo de tiempo limitado. Pero se logró realizar y llegar a cumplir con todo lo que se planificó y a su paso también ir resolviendo algunos imprevistos que sucedieron. Dichos imprevistos consistieron en unos errores detectados durante la fase de testing, los cuales los textos no se adaptaban correctamente al estilo de pantalla y se expandían por fuera de los márgenes del widget que los contenía.

Este proyecto fue una gran experiencia, ya que, permitió adquirir nuevas habilidades tanto en lo personal como en lo profesional. Ejemplos de esto son:

- Haber desarrollado una aplicación móvil.
- Interactuar con un cliente real para poder obtener los requisitos y los feedback a lo largo del proyecto. Debido a que no era una relación contractual, la interacción con el cliente fue más fluida y personal.
- Realizar el proyecto completo desde frontend hasta backend, adquiriendo el conocimiento necesario para el desarrollo de ambas.
- Manejar el tiempo al cronograma planificado adecuando el tiempo requerido para el desarrollo de las distintas tareas, con el objetivo de lograr la mayor eficiencia posible.

8.3 Trabajo futuro

Para finalizar, queda como trabajo futuro realizar mejoras a ***Podcast.ing*** para poder ampliar su funcionalidad y potenciar su utilidad. Algunas de las posibles mejoras irían en sentido de favorecer la visualización del resultado obtenido por el alumno, generar ejercicios más interactivos, agregar recompensas para que estimule al alumno a seguir practicando, poner desafíos, agregar una mayor cantidad de consignas, etc. Y en el caso de que aumente el número de estudiantes que consuman la aplicación, será necesario

realizar una migración de lo que se utilizó en el backend para permitir una mayor cantidad de solicitudes de los servicios.

Debido a todo lo mencionado, se puede vislumbrar en proyección al futuro varias mejoras y también la incorporación de nuevas funcionalidades para poder crear una aplicación móvil más completa.

Bibliografía

[1] *Infographic – Evolution of Mobile Apps*. Techahead. https://www.techaheadcorp.com/blog/infographic-evolution-of-mobile-apps/. Accedido 21/04/23

[2] *51 Mobile App Stats That Will Blow Your Mind*. Influencermarketinghub. https://influencermarketinghub.com/mobile-app-stats/. Accedido 21/04/23

[3] *What are the best apps for learning English?*. OxfordInternationalEnglish. https://www.oxfordinternationalenglish.com/what-are-the-best-apps-for-learning-english/ . Accedido 12/11/22.

[4] Maida, EG, Pacienzia, J. (2015). *Metodologías de desarrollo de software*. [Tesis de Licenciatura en Sistemas y Computación, Universidad Católica Argentina]. http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software.pdf Accedido 07/05/23.

[5] Kneuper,R.(2018).*Software Processes and Life Cycle Models*.Springer

[6] Brezočnik, Lucija & Majer, Črtomir. (2016). Comparison of agile methods: Scrum, Kanban, and Scrumban.

[7] *¿Cuál es la diferencia? Agile vs Scrum vs el método de cascada vs Kanban.* Smartsheet. https://es.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-*kanban* Accedido 9/12/22

[8] Laoyan,S.(6 de julio de 2022).Scrumban: The best of two Agile methodologies. Asana .https://asana.com/id/resources/scrumban . Accedido 13/11/22.

[9] Project Management Institute. (2004). *A guide to the Project Management Body of Knowledge (PMBOK guide)*. Project Management Institute. (3° Ed.)

[10] *Top 10 Best Mobile App Development Frameworks in 2022*. Upsilonit. https://www.upsilonit.com/blog/top-10-best-mobile-app-development-frameworks Accedido 19/11/22.

[11] Batschinski,G.(24 de Septiembre de 2020).Best backend for Flutter. https://george-51059.medium.com/best-backend-for-flutter-6948ae5b9367. Accedido 22/11/22

[12] Medewar, S. (2 de febrero de 2022). Los 7 mejores IDE para el desarrollo de aplicaciones móviles. Geekflare. https://geekflare.com/es/best-ide-for-mobile-app-development/ . Accedido 19/11/22.

[13] Booch et al. (2006). *El Lenguaje Unificado de Modelado. Guía del Usuario*. Pearson Educación. (2° Ed.)

[14] Miles y Hamilton.(2006). *Learning UML 2.0*. O'Reilly.

[15] Hamm, M. (2014). *Wireframing Essentials An introduction to user experience design*. Packt Publishing

[16] Meier,A. y Kaufmann, M. (2019).*SQL & NoSQL Databases Models, Languages, Consistency Options and Architectures for Big Data Management*.Springer Vieweg

[17] FlutterFire Overview.Firebase.https://firebase.flutter.dev/docs/overview. Accedido 26/11/22

[18] Flutter documentation.Flutter.https://docs.flutter.dev/ . Accedido 26/11/22.

[19] Angulo, M. (26 de marzo de 2019). Sistema de diseño: Material Design. https://www.tesseractspace.com/blog/sistema-de-diseno-material-design/. Accedido 19/04/22.

[20] Myers, G. (2004). *The Art of Software Testing*. John Wiley & Sons, Inc. (2° Ed.)

Anexos

Anexo 1: historias de usuario

Fecha: 05/09/22	
Número: 6	**Nombre Historia de Usuario:** Poder crear, editar y borrar los podcast.
Usuario: profesor	**Iteración Asignada:**2
Prioridad en Negocio: Alto	**Puntos Estimados:** 40 horas
Riesgo en Desarrollo: Alto	
Descripción: el profesor debe poder crear, editar, borrar y consultar los podcast y sus ejercicios relacionados.	
Observaciones: Tener en consideración los tres tipos de ejercicios: seleccionar la opción correcta, verdadero o falso y escribir la palabra correcta.	

Anexo 1.1: Historia de usuario para poder crear, editar y borrar los podcast.

Fecha: 05/09/22	
Número: 5	**Nombre Historia de Usuario:** visualizar los resultados de los estudiantes.
Usuario: profesor	**Iteración Asignada:** 3
Prioridad en Negocio: Alto	**Puntos Estimados:** 40 horas
Riesgo en Desarrollo: Alto	
Descripción: el profesor debe poder ver los resultados de los ejercicios que los alumnos realizaron.	
Observaciones: -	

Anexo 1.2: Historia de usuario para visualizar los resultados de los estudiantes vista del docente.

Fecha: 05/09/22	
Número: 7	**Nombre Historia de Usuario:** reproducir podcast.
Usuario: estudiante	**Iteración Asignada:** 3
Prioridad en Negocio: Alto	**Puntos Estimados:** 16 horas
Riesgo en Desarrollo: Alto	
Descripción: el estudiante debe poder reproducir el podcast que seleccionó.	
Observaciones: -	

Anexo 1.3: Historia de usuario para reproducir podcast.

Fecha: 05/09/22	
Número: 4	**Nombre Historia de Usuario:** visualización de corrección del ejercicio realizado por el estudiante.
Usuario: estudiante	**Iteración Asignada:** 4
Prioridad en Negocio: medio	**Puntos Estimados:** 16 horas
Riesgo en Desarrollo: bajo	
Descripción: el estudiante una vez que realizó un ejercicio debe poder ver el resultado que se sacó.	
Observaciones: -	

Anexo 1.4: Historia de usuario para la visualización de corrección del ejercicio realizado por el estudiante.

Fecha: 05/09/22

Número: 3	**Nombre Historia de Usuario:** visualización de los podcast con ejercicios.
Usuario: estudiante	**Iteración Asignada:** 4
Prioridad en Negocio: medio	**Puntos Estimados:** 36 horas
Riesgo en Desarrollo: bajo	
Descripción: el estudiante debe poder ver los podcast con los ejercicios que tiene.	
Observaciones: -	

Anexo 1.4: Historia de usuario para la visualización de los podcast con ejercicios.

Anexo 2: fichas de tarea

Tarea	
Número Tarea: 3	**Historia de Usuario (Nro. y Nombre):** 2 - registrarse en la aplicación.
Nombre Tarea: realizar el back de alta en la base de datos de los usuarios.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16hs
Fecha Inicio: 26/09/22	**Fecha Fin:** 30/09/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la conexión a la realtime database al registrarse para crear usuario, además de habilitar la autenticación cuando se registra el usuario.	

Anexo 2.1: Ficha de tarea para realizar el back de alta en la base de datos de los usuarios.

Tarea	
Número Tarea: 1	**Historia de Usuario (Nro. y Nombre):** 1 - login en la aplicación.
Nombre Tarea: realizar el back de consulta en la base de datos de los usuarios.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16hs
Fecha Inicio: 01/10/22	**Fecha Fin:** 04/10/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la conexión a la realtime database al loguearse para verificar usuario, además de validar la autenticación cuando el usuario intenta ingresar.	

Anexo 2.2: Ficha de tarea para realizar el back de consulta en la base de datos de los usuarios.

Tarea	
Número Tarea: 7	**Historia de Usuario (Nro. y Nombre):** 3 - visualización de los podcast con ejercicios.
Nombre Tarea: realizar front de listado de podcast para los estudiantes.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 8hs
Fecha Inicio: 02/11/22	**Fecha Fin:** 04/11/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la vista del listado de podcast para los estudiantes aplicando el diseño establecido en el prototipo aprobado por las profesoras.	

Anexo 2.3: Ficha de tarea para realizar front de listado de podcast para los estudiantes.

Tarea	
Número Tarea: 8	**Historia de Usuario (Nro. y Nombre):** 3 - visualización de los podcast con ejercicios.
Nombre Tarea: realizar front para ver el ejercicio.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 8hs

Fecha Inicio: 05/11/22	**Fecha Fin:** 06/11/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la vista de los ejercicios: seleccionar la opción correcta, verdadero o falso y completar con la palabra correcta, relacionado con los podcast para los estudiantes aplicando el diseño establecido en el prototipo aprobado por las profesoras.	

Anexo 2.4: Ficha de tarea para realizar front para ver el ejercicio.

Tarea	
Número Tarea: 5	**Historia de Usuario (Nro. y Nombre):** 3 - visualización de los podcast con ejercicios.
Nombre Tarea: realizar back para lectura en la base de datos de los podcast.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 4hs
Fecha Inicio: 07/11/22	**Fecha Fin:** 07/11/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la conexión a la realtime database para lectura de los podcast.	

Anexo 2.5: Ficha de tarea para realizar back para lectura en la base de datos de los podcast.

Tarea	
Número Tarea: 6	**Historia de Usuario (Nro. y Nombre):** 3 - visualización de los podcast con ejercicios.
Nombre Tarea: realizar back para lectura de la base de datos de los ejercicios relacionados a los podcast.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16hs
Fecha Inicio: 08/11/22	**Fecha Fin:** 11/11/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la conexión a la realtime database para verificar lectura de los ejercicios y relacionarlos con el podcast para visualizarlos.	

Anexo 2.6: Ficha de tarea para realizar back para lectura de la base de datos de los ejercicios relacionados a los podcast.

Tarea	
Número Tarea: 10	**Historia de Usuario (Nro. y Nombre):** 4 - visualización de corrección del ejercicio realizado por el estudiante.
Nombre Tarea: realizar front para ver el resultado obtenido.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 4hs
Fecha Inicio: 12/11/22	**Fecha Fin:** 12/11/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la vista de la visualización de los resultados aplicando el diseño establecido en el prototipo aprobado por las profesoras.	

Anexo 2.7: Ficha de tarea para realizar front para ver el resultado obtenido.

Tarea	
Número Tarea: 9	**Historia de Usuario (Nro. y Nombre):** 4 - visualización de corrección del ejercicio realizado por el estudiante.

Nombre Tarea: realizar back para alta o modificación en la base de datos de los resultados.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 12hs
Fecha Inicio: 13/11/22	**Fecha Fin:** 15/11/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la conexión a la realtime database para dar el alta y la modificación de los resultados de los ejercicios resuelto por los alumnos.	

Anexo 2.8: Ficha de tarea para realizar back para alta o modificación en la base de datos de los resultados.

Tarea	
Número Tarea: 18	**Historia de Usuario (Nro. y Nombre):** 7 - Reproducir podcast.
Nombre Tarea: Realizar front para ver el podcast.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 8hs
Fecha Inicio: 29/10/22	**Fecha Fin:** 30/10/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la vista del reproductor de podcast aplicando el diseño establecido en el prototipo aprobado por las profesoras.	

Anexo 2.9: ficha de tarea para realizar front para ver el podcast.

Tarea	
Número Tarea: 19	**Historia de Usuario (Nro. y Nombre):** 7 - Reproducir podcast.
Nombre Tarea: Realizar back de reproductor de podcast.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 8hs
Fecha Inicio: 31/10/22	**Fecha Fin:** 01/11/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la carga y descarga al storage del audio del podcast.	

Anexo 2.10: Ficha de tarea para realizar back de reproductor de podcast.

Tarea	
Número Tarea: 11	**Historia de Usuario (Nro. y Nombre):** 5 - visualizar los resultados de los estudiantes vista del docente.
Nombre Tarea: realizar back para lectura en la base de datos de los resultados.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 28hs
Fecha Inicio: 19/10/22	**Fecha Fin:** 25/10/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la conexión a la realtime database para la lectura de los resultados de los ejercicios que resolvieron los alumnos.	

Anexo 2.11: Ficha de tarea para realizar back para lectura en la base de datos de los resultados.

Tarea	
Número Tarea: 12	**Historia de Usuario (Nro. y Nombre):** 5 - visualizar los resultados de los estudiantes vista del docente.
Nombre Tarea: realizar front de listado de los resultados de los ejercicios que realizaron los estudiantes.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 12hs
Fecha Inicio: 26/10/22	**Fecha Fin:** 28/10/22

Programador Responsable: Ivanna F. Jutterpeker
Descripción: Se debe crear la vista del listado de los resultados de los ejercicios resueltos por los alumnos en la sección del docente aplicando el diseño establecido en el prototipo aprobado por las profesoras.

Anexo 2.12: Ficha de tarea para realizar front de listado de los resultados de los ejercicios que realizaron los estudiantes.

Tarea	
Número Tarea: 13	**Historia de Usuario (Nro. y Nombre):** 6 - Poder crear, editar y borrar los podcast.
Nombre Tarea: realizar back para consultar/editar/crear/eliminar en la base de datos de los podcast.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 8hs
Fecha Inicio: 13/10/22	**Fecha Fin:** 14/10/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la conexión a la realtime database para la consultar/editar/crear/eliminar podcast.	

Anexo 2.13: Ficha de tarea para realizar back para consultar/editar/crear/eliminar en la base de datos de los podcast.

Tarea	
Número Tarea: 14	**Historia de Usuario (Nro. y Nombre):** 6 - Poder crear, editar y borrar los podcast.
Nombre Tarea: realizar back para consultar/editar/crear/eliminar en la base de datos de los ejercicios relacionados a los podcast.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16hs
Fecha Inicio: 15/10/22	**Fecha Fin:** 18/10/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la conexión a la realtime database para la consultar/editar/crear/eliminar de los ejercicios relacionados a los podcast.	

Anexo 2.14: Ficha de tarea para realizar back para consultar/editar/crear/eliminar en la base de datos de los ejercicios relacionados a los podcast.

Tarea	
Número Tarea: 15	**Historia de Usuario (Nro. y Nombre):** 6 - Poder crear, editar y borrar los podcast.
Nombre Tarea: realizar front de listado de podcast para los docentes.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 8hs
Fecha Inicio: 05/10/22	**Fecha Fin:** 06/10/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la vista del listado de los podcast con toda la data para los docentes aplicando el diseño establecido en el prototipo aprobado por las profesoras.	

Anexo 2.15: Ficha de tarea para realizar front de listado de podcast para los docentes.

Tarea	
Número Tarea: 16	**Historia de Usuario (Nro. y Nombre):** 6 - Poder crear, editar y borrar los podcast.
Nombre Tarea: realizar front para editar/crear un podcast.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16hs
Fecha Inicio: 07/10/22	**Fecha Fin:** 10/10/22
Programador Responsable: Ivanna F. Jutterpeker	

Descripción: Se debe crear la vista donde se visualiza los campos para adaptar el podcast o para crear uno de cero, aplicando el diseño establecido en el prototipo aprobado por las profesoras.

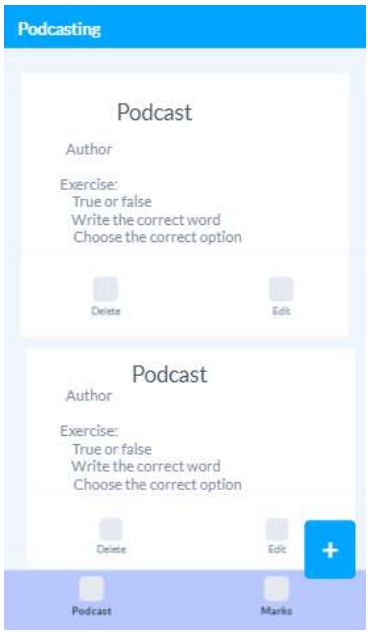
Anexo 2.16: Ficha de tarea para realizar front para editar/crear un podcast.

Tarea	
Número Tarea: 17	**Historia de Usuario (Nro. y Nombre):** 6 - Poder crear, editar y borrar los podcast.
Nombre Tarea: realizar front para editar/crear el ejercicio.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 8hs
Fecha Inicio: 11/10/22	**Fecha Fin:** 12/10/22
Programador Responsable: Ivanna F. Jutterpeker	
Descripción: Se debe crear la vista donde se visualiza los campos para adaptar los ejercicios (seleccionar la opción correcta, verdadero o falso y completar con la palabra correcta) de un podcast o para crear uno de cero, aplicando el diseño establecido en el prototipo aprobado por las profesoras.	

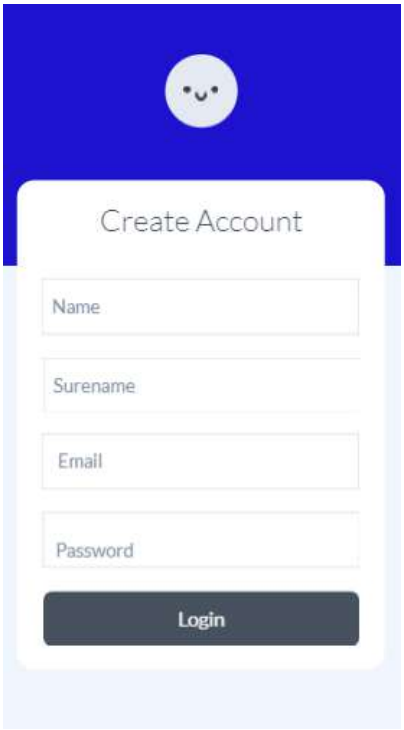
Anexo 2.17: Ficha de tarea para realizar front para editar/crear el ejercicio.

Anexo 3: interfaz de usuario (prototipos)

Anexo 3.1: Interfaz de ejercicio True or False.

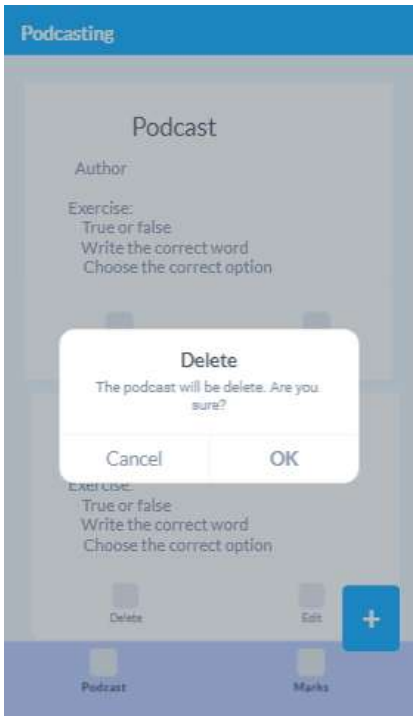


Anexo 3.2: Interfaz de listado de podcast en vista docente.



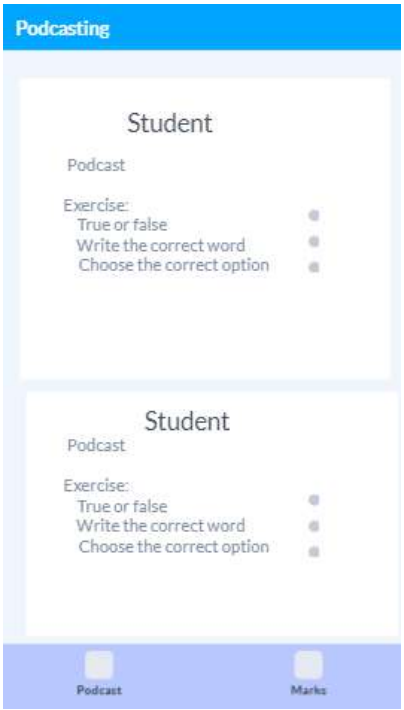
Anexo 3.3: Interfaz de registrarse.

Anexo 3.4: Interfaz del cartel con la nota menor a 6 del ejercicio resuelto.

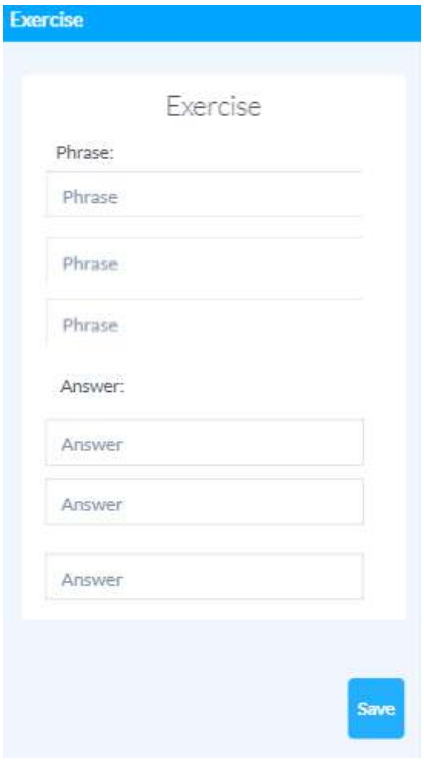


Anexo 3.5: Interfaz del cartel de eliminar podcast.

Anexo 3.6: Interfaz del cartel con la nota mayor a 6 del ejercicio resuelto.



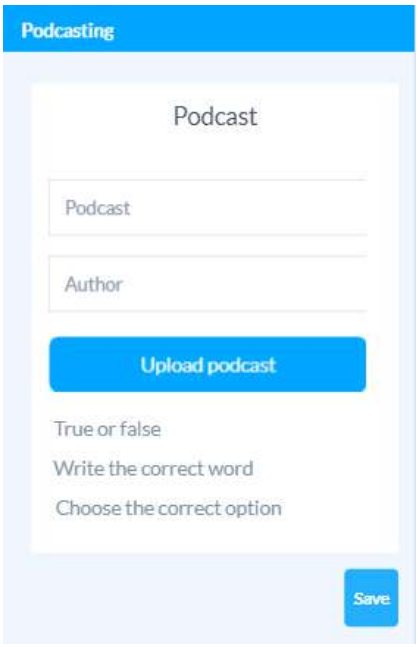
Anexo 3.7: Interfaz del listado de las notas de los alumnos.



Anexo 3.8: Interfaz del formulario para crear un ejercicio.

Anexo 3.9: Interfaz del ejercicio de Write the correct word.

Anexo 3.10: Interfaz del ejercicio Choose the correct option.



Anexo 3.11: Interfaz del formulario para crear un podcast.

Anexo 4: casos de pruebas

Acceso aplicación siendo estudiante	**Número**: 03
Descripción: un estudiante desea ingresar a la pantalla principal de la aplicación.	
Prerrequisitos: debe encontrarse logueado dentro de la aplicación.	
Pasos: 1. Ingresar a la aplicación con email y contraseña. Los valores utilizados fueron:	

– Email: “ivannaflorenciajutterpeker@gmail.com” – Contraseña:123456789
Resultado esperado: ver listado de podcast.
Resultado obtenido: Correcto. Error: el texto del nombre del podcast se sale de la card. 16/11/22. Realizado por Raquel Ramos. 

Anexo 4.1: Casos de prueba de acceso aplicación siendo estudiante.

Acceso aplicación siendo profesora	**Número**: 04
Descripción: una profesora desea ingresar a la pantalla principal de la aplicación.	
Prerrequisitos: debe encontrarse logueado dentro de la aplicación.	
Pasos: 1. Ingresar a la aplicación con email y contraseña. Los valores utilizados fueron: – Email: “teacher@gmail.com” – Contraseña:1234567890	
Resultado esperado: ver listado de podcast.	
Resultado obtenido: Correcto. Error: el texto del nombre del podcast se sale de la card. 19/10/22. Realizado por Raquel Ramos.	

Anexo 4.2: Casos de prueba de acceso aplicación siendo profesora .

Registrarse en la aplicación	**Número**: 05
Descripción: un alumno desea ingresar por primera vez a la aplicación por lo que debe crearse una cuenta pero ya existe ese email ingresado.	
Prerrequisitos: -	
Pasos: 1. Seleccionar “Create a new account”. 2. Ingresar los datos solicitados: nombre, apellido, email y contraseña. Los valores utilizados fueron: – Nombre: “ivanna” – Apellido: “Jutterpeker” – Email: “ivannaflorenciajutterpeker@gmail.com” – Contraseña:123456789 3. Sale un cartel con un mensaje de que ya existe esa cuenta.	
Resultado esperado: Mensaje “The email already exist”.	
Resultado obtenido: Correcto. 05/10/22. Realizado por Raquel Ramos.	

Anexo 4.3: Casos de prueba de registrarse en la aplicación.

Selección de podcast	**Número**: 06
Descripción: un alumno desea escuchar el podcast seleccionado.	
Prerrequisitos: debe encontrarse logueado dentro de la aplicación.	

Pasos: 1. Selecciona el podcast que desea escuchar. El podcast elegido fue: – Podcast: “English in my life”. 2. Le da play cuando se ven todos los datos relacionados al podcast.
Resultado esperado: Escuchar el podcast.
Resultado obtenido: Correcto. 16/11/22. Realizado por Raquel Ramos.

Anexo 4.4: Casos de prueba de selección de podcast.

Selección de ejercicios	**Número**: 07
Descripción: un alumno desea resolver los ejercicios relacionados a un podcast.	
Prerrequisitos: debe encontrarse logueado dentro de la aplicación.	
Pasos: 3. Selecciona el podcast que desea escuchar. El podcast elegido fue: – Podcast: “Days of quarantine”. 2. Le da play cuando se ven todos los datos relacionados al podcast. 3. Selecciona el ejercicio que quiere resolver. El ejercicio elegido fue: – Ejercicio: “True or false exercise”. 4. Puede ver el ejercicio.	
Resultado esperado: Ver el ejercicio para resolver.	
Resultado obtenido: Correcto. 16/11/22. Realizado por Raquel Ramos.	

Anexo 4.5: Casos de prueba de selección de ejercicios.

Resultado del ejercicio	**Número**: 08
Descripción: un alumno después de resolver el ejercicio obtiene el resultado del mismo.	
Prerrequisitos: debe encontrarse logueado dentro de la aplicación.	
Pasos: 4. Selecciona el podcast que desea escuchar. El podcast elegido fue: – Podcast: “Days of quarantine”. 2. Le da play cuando se ven todos los datos relacionados al podcast. 5. Selecciona el ejercicio que quiere resolver. El ejercicio elegido fue:	

- Ejercicio: “True or false exercise”.

6. Puede ver el ejercicio y lo completa. Los valores utilizados fueron:
   - Respuesta: “false”
   - Respuesta: “false”.
   - Respuesta: “true”

7. Visualiza un cartel con su resultado.

Resultado esperado: Cartel con la nota del ejercicio.

Resultado obtenido:
Correcto.
16/11/22.
Realizado por Raquel Ramos.

Anexo 4.6: Casos de prueba de resultado del ejercicio.

Editar un podcast	**Número**: 09

Descripción: una profesora desea modificar un podcast ya creado.

Prerrequisitos: debe encontrarse logueado dentro de la aplicación.

Pasos:

8. Se selecciona el botón del editar relacionado al podcast a modificar. El podcast elegido fue:
   - Podcast: “Days of quarantine”.
2. Se ve los datos del podcast para modificar.
3. Se corrige la información. El valor utilizado fue:
   - Autor: “Ivanna”.
4. Se selecciona el botón de guardar.

Resultado esperado: Aparezcan los datos del podcast actualizados con la nueva información.

Resultado obtenido:
Correcto.
19/10/22.
Realizado por Raquel Ramos.

Anexo 4.7: Casos de prueba de editar un podcast.

Eliminar podcast	**Número**: 10

Descripción: una profesora desea eliminar uno de los podcast.

Prerrequisitos: debe encontrarse logueado dentro de la aplicación.

Pasos:

1. Se selecciona el botón del eliminar relacionado al podcast a modificar.
2. Se ve un cartel de confirmación.
3. Se selecciona el botón de aceptar.

Resultado esperado: se elimina el podcast seleccionado.

Resultado obtenido:
Correcto.
19/10/22.
Realizado por Raquel Ramos.

Anexo 4.8: Casos de prueba de eliminar un podcast.

Editar los ejercicios de un podcast	**Número**: 11

Descripción: una profesora desea modificar un ejercicio relacionado a un podcast ya creado.

Prerrequisitos: debe encontrarse logueado dentro de la aplicación.

Pasos:

5. Se selecciona el botón del editar relacionado al podcast a modificar. El podcast elegido fue:
   - Podcast: “Days of quarantine”.
2. Se ve los datos del podcast para modificar.
3. Se selecciona el ejercicio a modificar. El ejercicio elegido fue:
   - Ejercicio: “Choose the correct option”.
4. Se ve los datos del ejercicio.
9. Se realiza las modificaciones. Los valores utilizados fueron:
   - Consigna 1: “The pandemic was something that … al lof a sudden.”
   - Respuesta: “appeared”
   - Consigna 2: “She … WhatsApp message, Meet or Skype.”
   - Respuesta: “used”.
   - Consigna 3: “She … all day doing things related ti the coures that she was taking.”
   - Respuesta: “stayed”
10. Se selecciona el botón de guardar.

Resultado esperado: Aparezcan los datos del ejercicio actualizados con la nueva información.

Resultado obtenido:
Correcto.
19/10/22.
Realizado por Raquel Ramos.

Anexo 4.9: Casos de prueba de editar los ejercicios de un podcast.

Crear un ejercicio	**Número**: 13

Descripción: una profesora desea crear el ejercicio a un podcast ya creado.
Prerrequisitos: debe encontrarse logueado dentro de la aplicación.
Pasos: 11. Se selecciona el botón del editar relacionado al podcast a modificar. El podcast elegido fue: – Podcast: “Days of quarantine”. 12. Se ve los datos del podcast para modificar. 13. Se selecciona el ejercicio a crear. El ejercicio elegido fue: – Ejercicio: “True or false exercise”. 14. Se escribe las consignas junto con sus respuestas. Los valores utilizados fueron: – Consigna 1: “She is a student of electromechanical engineering.” – Respuesta: “false” – Consigna 2: “She was prepared for the pandemic.” – Respuesta: “false”. – Consigna 3: “She did more homework.” – Respuesta: “true” 15. Se selecciona el botón de guardar.
Resultado esperado: Aparezcan los datos del ejercicio creado dentro del podcast.
Resultado obtenido: Correcto. 19/10/22. Realizado por Raquel Ramos.

Anexo 4.10: Casos de prueba de crear un ejercicio en podcast ya creado.

Crear un ejercicio	**Número**: 14
Descripción: una profesora desea crear el ejercicio a un podcast nuevo.	
Prerrequisitos: debe encontrarse logueado dentro de la aplicación.	
Pasos: 1. Se selecciona el botón del crear podcast. 16. Se selecciona el ejercicio a crear. El ejercicio elegido fue: – Ejercicio: “Choose the correct option” 17. Se escribe las consignas junto con sus respuestas. Los valores utilizados fueron: – Consigna 1: “The pandemic was something that … al lof a sudden.” – Respuesta: “appeared” – Consigna 2: “She … WhatsApp message, Meet or Skype.” – Respuesta: “used”. – Consigna 3: “She … all day doing things related ti the coures that she was taking.”	

– Respuesta: “stayed” 18. Se selecciona el botón de guardar.
Resultado esperado: Aparezcan los datos del ejercicio creado dentro del podcast.
Resultado obtenido: Correcto. 19/10/22. Realizado por Raquel Ramos.

Anexo 4.11: Casos de prueba de crear un ejercicio en podcast nuevo.

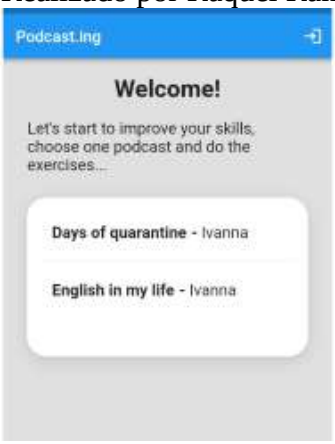
Crear un podcast	**Número**: 15
Descripción: una profesora desea crear un podcast.	
Prerrequisitos: debe encontrarse logueado dentro de la aplicación.	
Pasos: 1. Se selecciona el botón del crear podcast. 2. Se escribe el título del podcast, el autor y se sube el archivo del audio. Los valores utilizados fueron: – Título: “Days of quarantine”. – Autor: “Ivanna”. 3. Se selecciona el botón de guardar.	
Resultado esperado: Aparezcan los datos del podcast creado dentro de la pantalla principal.	
Resultado obtenido: Correcto. 19/10/22. Realizado por Raquel Ramos.	

Anexo 4.12: Casos de prueba de crear un podcast.

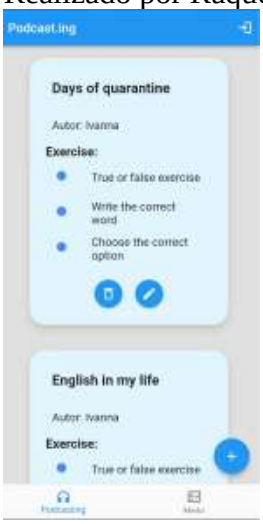
A continuación se observan los test que se volvieron a ejecutar con la corrección del error:

Acceso aplicación siendo estudiante	**Número**: 03
Descripción: un estudiante desea ingresar a la pantalla principal de la aplicación.	
Prerrequisitos: debe encontrarse logueado dentro de la aplicación.	
Pasos: 1. Ingresar a la aplicación con email y contraseña.	
Resultado esperado: ver listado de podcast.	

Resultado obtenido:
Correcto.
16/11/22.
Realizado por Raquel Ramos.



Anexo 4.13: Casos de prueba de acceso aplicación siendo estudiante (corrección).

Acceso aplicación siendo profesora	**Número**: 04
Descripción: una profesora desea ingresar a la pantalla principal de la aplicación.	
Prerrequisitos: debe encontrarse logueado dentro de la aplicación.	
Pasos: 1. Ingresar a la aplicación con email y contraseña.	
Resultado esperado: ver listado de podcast.	
Resultado obtenido: Correcto. 19/10/22. Realizado por Raquel Ramos. 	

Anexo 4.14: Casos de prueba de acceso aplicación siendo profesora (corrección).

Anexo URL

- Flutter: https://flutter.dev/
- React Native: https://reactnative.dev/
- Ionic: https://ionicframework.com/
- Apache Cordova: https://cordova.apache.org/
- Xamarin: https://dotnet.microsoft.com/en-us/apps/xamarin
- Back4app: https://www.back4app.com/
- Parse: https://parseplatform.org/
- Firebase: https://firebase.google.com/
- Backendless: https://backendless.com/
- AWS Amplify: https://aws.amazon.com/es/amplify/
- Android Studio: https://developer.android.com/studio
- Qt Creator: https://www.qt.io/product/development-tools
- Xcode: https://developer.apple.com/xcode/
- JetBrains Rider: https://www.jetbrains.com/es-es/rider/
- Visual Studio Code: https://code.visualstudio.com/
- Git: https://git-scm.com/
- Apache Subversion: https://subversion.apache.org/
- Mercurial: https://www.mercurial-scm.org/
- Monotone: https://www.monotone.ca/