

PROYECTO

NI-HON



CICLO FORMATIVO DE GRADO SUPERIOR:

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

AUTORES:

Alexia Molina, Wei Shan, Daniel Mora

Licencia

Esta obra está bajo una licencia Reconocimiento-Compartir bajo la misma licencia 3.0 España de Creative Commons. Para ver una copia de esta licencia, diríjase:

<http://creativecommons.org/licenses/by-sa/3.0/es/> o envíe una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

RESUMEN

El objetivo de este proyecto es el desarrollo de una aplicación móvil orientada al estudio del idioma **japonés**. Específicamente, se trata una aplicación utilizable en dispositivos **Android**, desarrollada principalmente utilizando el entorno de desarrollo Android Studio.

El punto principal en el que se centra el diseño de esta aplicación es la facilidad de utilización y la creación de interfaces visualmente agradables a la vista y de funcionamiento intuitivo.

La aplicación tiene acceso a una **base de datos** en la que se guardan las credenciales de inicio de sesión y el estado actual de cada cuenta, permitiendo así que los usuarios continúen con el progreso en cualquier momento.

Se incluye un pequeño tutorial que mostrará al usuario el funcionamiento de la aplicación cuando la vaya a utilizar por primera vez.

La actividad principal de la aplicación es deslizable y muestra al usuario las distintas lecciones, por las que le irá guiando **Tanuki**, la mascota de la aplicación. Desde esta actividad se puede acceder a las lecciones, que disponen de una explicación teórica y ejercicios de res- puesta múltiple para que el usuario pruebe los conocimientos que ha adquirido.

ABSTRACT

The goal of this project is developing a mobile phone app aimed at helping students of the **Japanese** language. Specifically, it's an app that can be used on **Android** devices, developed mainly through the integrated development environment Android Studio.

The focus point of the design for this app is the ease of use and creating interfaces that are visually attractive and which usage is intuitive, to avoid making the users go through a learning process just to use the app. This is important for the app to be an effective aid in the learning of the language.

The app has access to a **database** where user credentials are stored, together with the process and current state of each account, thus allowing users to continue their progress at any given moment.

We have also included a tutorial that will show the user how to operate the app the first time they enter it.

The main activity of the app is scrollable and shows the user the different lessons, through which they'll be guided by **Tanuki**, the app's mascot. From this activity they will be able to access each lesson, where they'll find explanations on different subjects and multiple answer tests where they can prove they have acquired the knowledge to advance to the next lesson.

Índice de contenido

INTRODUCCIÓN	9
1 CONTEXTO FUNCIONAL Y TECNOLÓGICO	12
1.1 CONTEXTO FUNCIONAL.....	12
2.2 CONTEXTO TECNOLÓGICO	12
2.2.2 GitHub (Git)	13
2.2.3 Base de Datos	14
2.2.4 Autenticación de usuarios.....	18
2 PLANIFICACIÓN DEL PROYECTO	19
3 DESCRIPCIÓN DE LA SOLUCIÓN: ANÁLISIS Y DISEÑO	23
3.1 ESPECIFICACIÓN DE REQUISITOS.....	23
3.2 SELECCIÓN DE PLATAFORMA TECNOLÓGICA.....	23
3.3 DESCRIPCION DEL DISEÑO DE LA SOLUCION.....	23
3.3.1 DISEÑO GENERAL	23
3.3.2 CARACTERISTICAS DE LA MASCOTA	26
3.3.3 MAIN SCREEN	28
4. DESCRIPCION DE LA SOLUCIÓN: CONSTRUCCIÓN.....	34
4.1 Documentación descriptiva.....	34
4.2 Problemas encontrados y soluciones.....	80
5 VALIDACIÓN DE LA SOLUCIÓN.....	81
5.1 Documentación descriptiva.....	81
5.2 Problemas encontrados y justificación	82
6 FUENTES	85
7 ANEXO	86

Índice de figuras

1. [Duolingo](#)
2. [Renshuu](#)
3. [Babble](#)
4. [Android Studio.](#)
5. [Icono Git y GitHub.](#)
6. [Diagrama de Git Log.](#)
7. [Icono Firebase.](#)
8. [Cloud Firestore \(Tabla “users”\).](#)
9. [Icono SQLite.](#)
10. [SQLite \(Tabla “user”\).](#)
11. [SQLite \(Tabla “questions”\).](#)
12. [SQLite \(Tabla “options”\).](#)
13. [Gráfica usuarios activos \(Firebase\).](#)
14. [Tabla de usuarios autenticados \(Firebase Auth\).](#)
15. [Boceto planificación del proyecto.](#)
16. [Boceto prácticas.](#)
17. [Boceto página principal.](#)
18. [Icono Krita.](#)
19. [Iconos usados en la aplicación.](#)
20. [Fondo de pantalla \(Main v1\).](#)
21. [Fondo de pantalla \(Main v2\).](#)
22. [Mascota \(bebé\).](#)
23. [Mascota \(adulto\).](#)
24. [Mascota \(adulto 2\).](#)
25. [Diseño página principal.](#)
26. [Diseño perfil de usuario.](#)
27. [Iconos del usuario.](#)
28. [Diseño página de lecciones \(1\).](#)
29. [Diseño página de lecciones \(2\).](#)
30. [Diseño página de práctica 1.](#)
31. [Diseño Práctica 1 \(respuesta\).](#)
32. [Captura Splash \(modo día y nocturno\).](#)
33. [Captura Inicio de Sesión.](#)

34. [Método inicio de sesión normal.](#)
35. [Método guardar sesión.](#)
36. [Archivo json de Firebase.](#)
37. [Método inicio sesión con Google.](#)
38. [Mensaje inicio de sesión con Google.](#)
39. [Método añadir usuario.](#)
40. [Mensaje error al iniciar sesión.](#)
41. [Mensaje restablecer contraseña.](#)
42. [Método del mensaje para restablecer contraseña.](#)
43. [Método para restablecer contraseña.](#)
44. [Texto del correo restablecimiento de contraseña.](#)
45. [Captura registro de usuario.](#)
46. [Método de control de error en registro.](#)
47. [Método para guardar usuario.](#)
48. [Método iniciar sesión automática al registrar.](#)
49. [Captura Main.](#)
50. [Método del RecyclerView.](#)
51. [Método de la lista de lecciones.](#)
52. [Método del Adapter del Main.](#)
53. [Método Holder de las lecciones.](#)
54. [Método de asociación de los botones de las lecciones.](#)
55. [Captura Main \(Al pulsar botón de las lecciones\).](#)
56. [Método para vincular las vistas en lecciones.](#)
57. [Método LessonViewHolder.](#)
58. [Método de creación de la actividad de lección.](#)
59. [Método Fragment de lecciones.](#)
60. [Método para crear vistas en lecciones.](#)
61. [Método de funcionalidad de los elementos de las lecciones.](#)
62. [Captura de las prácticas.](#)
63. [Tablas usadas en las prácticas.](#)
64. [Método de consulta en tabla Questions.](#)
65. [Método de consulta en tabla Options.](#)
66. [Captura comportamiento de las prácticas.](#)
67. [Método de control en avance de prácticas.](#)
68. [Método de la variable que controla el avance de las prácticas.](#)
69. [Mensaje de práctica completada.](#)
70. [Método de ejemplo de creación del objeto de practica reto.](#)

71. [Captura del perfil de usuario.](#)
72. [Captura de las opciones de icono del usuario.](#)
73. [Método para guardar icono del usuario.](#)
74. [Captura del progreso del usuario.](#)
75. [Método de sincronización del progreso.](#)
76. [Captura del botón: borrar cuenta.](#)
77. [Mensaje de borrar cuenta.](#)
78. [Mensaje de borrar cuenta \(con éxito\).](#)
79. [Mensaje de borrar cuenta \(sin éxito\).](#)
80. [Método de eliminar cuenta normal.](#)
81. [Método de eliminar cuenta Google.](#)
82. [Método de eliminar datos locales del usuario.](#)
83. [Método de consulta para eliminar datos del usuario.](#)
84. [Captura de configuraciones.](#)
85. [Captura de selección de idioma.](#)
86. [Método de selección de idioma.](#)
87. [Método de cambiar idioma.](#)
88. [Método de obtener idioma del sistema.](#)
89. [Método de modo noche.](#)
90. [Captura del archivo tema noche.](#)
91. [Captura de información.](#)
92. [Ejemplo del archivo JSON para las preguntas.](#)
93. [Método para recoger el JSON.](#)
94. [Método para pasar el JSON a un Objeto.](#)
95. [Capturas sobre las pruebas del registro.](#)
96. [Capturas sobre las pruebas para restablecer contraseña.](#)
97. [Método del mensaje de reinicio.](#)
98. [Mensaje de reinicio.](#)
99. [Método para reiniciar la aplicación.](#)

INTRODUCCIÓN

Este documento responde a la realización del módulo de Proyecto del CFGS en Desarrollo de Aplicaciones Multiplataforma.

El módulo de Proyecto complementa la formación establecida para el resto de los módulos profesionales que integran el título en las funciones de análisis del contexto, diseño del proyecto y organización de la ejecución.

Este proyecto consiste en la creación de una aplicación móvil, NI-HON, que se enfoca en ofrecer una manera de aprender y practicar japones para todos los niveles y de manera gratuita.

Los elementos principales de la aplicación son:

- Lecciones con información de nivel básico.
- Actividades prácticas interactivas para comprobar la adquisición de los conocimientos.
- Menú principal desde el que se puede seguir fácilmente el progreso personal.
- Interfaz agradable.
- Mascota personalizada que cambia conforme el usuario progresá.
- Guardado constante del progreso de los usuarios en la nube.
- Inicio de sesión con Google.

El proyecto está desarrollado con la intención de aportar una opción visualmente agradable, fácil de usar y completamente gratuita al mercado de aplicaciones Android para el estudio del idioma japonés.

DEFENSA DE VIABILIDAD

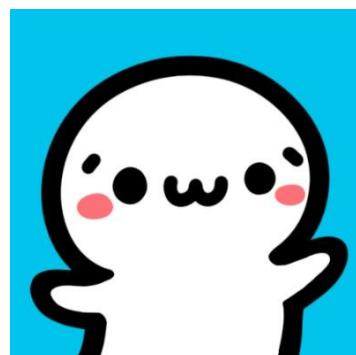
Para comprobar la viabilidad de nuestro producto, hicimos un estudio de mercado para ver nuestras posibilidades en el sector de las aplicaciones para aprender idiomas.

Las distintas aplicaciones más conocidas para aprender idiomas son:

- **Duolingo**



- **Renshuu**



- **Babble**



Nuestro estudio de mercado se centró en ver que es lo que le funciona a la competencia, entender porque les funciona e ideas de cómo mejorarlo.

La primera conclusión que sacamos es que tener una mascota para la aplicación es una buena idea, Duolingo y Renshuu lo hacen. La mascota motiva al usuario a aprender idiomas y hacer que sea mucho más cercano practicarlo.

Queríamos mejorar esa idea, principalmente de dos maneras. Como nuestra app está centrada solo en aprender japones, teníamos que explotar esa idea, por eso en vez de poner algo genérico como mascota, se nos ocurrió la idea del tanuki, de esta manera la mascota estaría inspirada en la cultura japonesa.

Y la segunda manera de potenciar la idea es añadir “evoluciones” al tanuki, de esta manera el progreso se refleja en la aplicación.

Además, el tener una mascota le da una seña de identidad única a la aplicación.

Se hizo una estimación de cuánto tiempo estaría el usuario dentro de la app aprendiendo japones y la conclusión fue muy poco. No es lo mismo usar una aplicación para aprender idiomas que asistir a una academia y retener la atención del usuario para que así pueda aprender de manera más eficiente es todo un reto.

La solución a este problema principalmente fue hacer las lecciones lo más eficientes posible en el tiempo, es decir, que duren muy poco y que sean lo más efectivas posibles. Ese tiempo tiene que adaptarse a lo que dura un viaje en metro, el tiempo que estaría el usuario en un descanso del trabajo, ir al servicio, tomarse un café, etc. Si las lecciones fueran muy largas

1 CONTEXTO FUNCIONAL Y TECNOLÓGICO

1.1 CONTEXTO FUNCIONAL

La aplicación se enfoca en proporcionar a los usuarios una experiencia de aprendizaje del idioma japonés de manera **interactiva, progresiva y animada**. El objetivo es hacer que el proceso sea **agradable y amigable** para el usuario, esto permite **fortalecer sus conocimientos** a través de las **lecciones y prácticas interactivas** ofrecidas.

La aplicación ofrece **funcionalidades** para el **inicio de sesión** y seguimiento del progreso en las **lecciones y prácticas**. Así, permite a los usuarios avanzar de forma estructurada en el aprendizaje. Más parámetros de configuración permite al usuario **personalizar** el App de forma determinada.

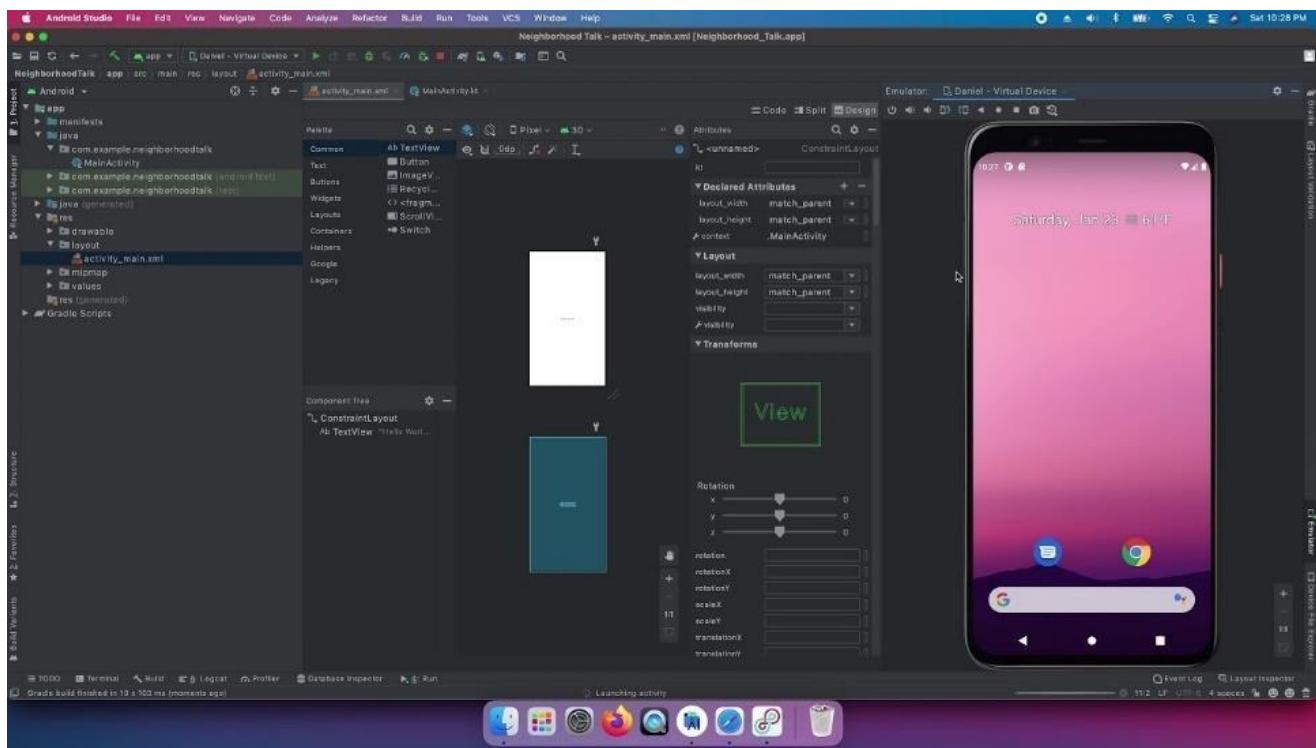
2.2 CONTEXTO TECNOLÓGICO

La aplicación está desarrollada en **Android Studio** utilizando Java (un lenguaje de programación orientado a objetos). Para el diseño se han empleado recursos de Google material, recursos gratuitos de internet y realizadas manualmente por la compañera como diseñadora. La base de datos se utiliza la proporcionada por **Firebase**, una base de datos gratuita que ofrece Google.

2.2.1 ANDROID STUDIO

Se ha utilizado **Android Studio** para el desarrollo de la aplicación, siendo un IDE especializado para el desarrollo de aplicaciones de Android presentando potentes herramientas de emulación y codificación. Además de integrar distintas herramientas de desarrollo como **Git**.

Se ha elegido el IDE de Android Studio por su capacidad para el desarrollo y sus características, como la integración de Git y VSC y los emuladores.



2.2.2 GitHub (Git)

Siendo una plataforma para desarrolladores, se ha gestionado el proyecto "**Ni-Hon**" en esta plataforma y mediante el uso de **Git** permite con mayor facilidad el control del desarrollo, permitiendo la monitorización y cooperación de forma más controlada y sencilla. Además, es posible **integrarlo en Android Studio**.



Enlace del trabajo en GitHub: [ArinoMichi/Ni-Hon: TFG DAM \(github.com\)](https://github.com/ArinoMichi/Ni-Hon)

Se ha utilizado GitHub principalmente por su fácil manejo y para la gestión de diferentes desarrollos.

Principalmente se ha dividido el trabajo en varias ramas:

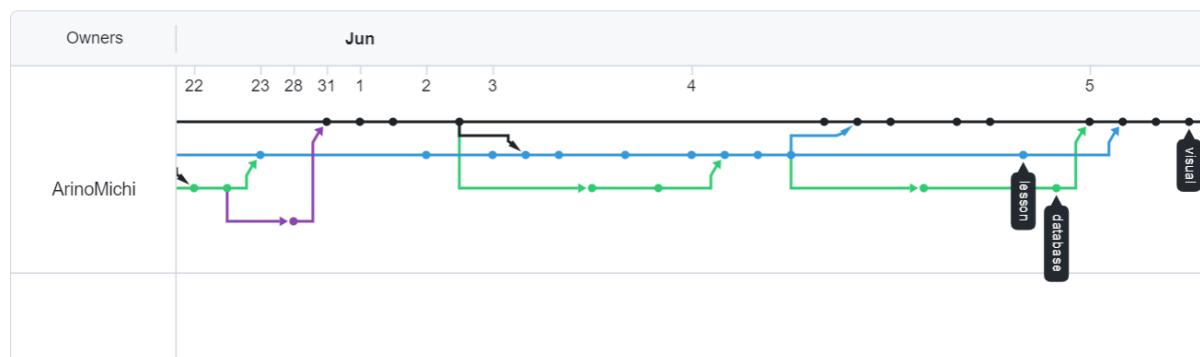
- Diseño y Visual
- Base de datos Firebase
- Funcionalidades como lessons y main

Cada uno ha ido haciendo commits y actualizando el proyecto de manera remota, lo que nos ha permitido incorporar avances ajenos de manera más rápida y ágil.

Ejemplo de diagrama de commits de las ramas:

Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.



2.2.3 Base de Datos

En el proyecto se han empleado **dos bases de datos**:

- Una utilizando la proporcionada por **Firebase** para gestionar los usuarios. Esto proporciona una mejora en la **protección** y un **mayor control** sobre la gestión de los datos del usuario en su cuenta.
- Otra utilizando **SQLite**, una base de datos local ligera, proporcionada por las librerías incorporadas en el sistema de Android. Esto se utiliza para el control de las prácticas mostradas al usuario, es usado por su **rapidez y eficiencia**, además de la **independencia de la conexión a Internet** (que a veces puede ser lenta).

Firebase:

La base de datos proporcionada por Firebase (**Cloud Firestore**) facilita el desarrollo del proyecto de manera considerable, abstrayendo al desarrollador de todas las gestiones de backend que puede tener la aplicación.

Firebase proporciona una base de datos en tiempo real, backend y organizada en forma de árbol JSON. El servicio proporciona a los desarrolladores de aplicaciones una API que permite que la información de las aplicaciones sea sincronizada y almacenada en la nube de Firebase.¹⁵¹⁶ La compañía habilita integración con aplicaciones Android, iOS, JavaScript, Java, Objective-C, Swift y Node.js.

La aplicación gestiona a los usuarios mediante la tabla "**users**", por los campos siguientes:

- "**email**" identifica al usuario (solo existe una cuenta asociada a cada correo).
- "**icon**" permite guardar el ícono seleccionado por el usuario.
- "**level**" identifica las lecciones realizadas por el usuario.
- "**name**" es el nickname del usuario en la app.

The screenshot shows the Google Cloud Firestore interface. At the top, it says "Ni-Hon" and "Cloud Firestore". There are navigation icons and a help button. A message about protecting resources from abuse is displayed, along with a link to "Configurar la Verificación de aplicaciones". Below this, there are two tabs: "Vista del panel" (selected) and "Compilador de consultas". The main area shows a hierarchical view of collections: "ni-hon-98ba1" > "users" > "hUwMmEdsqRRvCKGB2FKvVRmPhKR2". On the right, the details of this specific document are shown:

	hUwMmEdsqRRvCKGB2FKvVRmPhKR2
+ Iniciar colección	+ Agregar documento
users	hUwMmEdsqRRvCKGB2FKvVRmPhKR2 email: "admin2@gmail.com" icon: 0 level: 0 name: "admin" password: "admin2"

Esta base de datos está unida al **Auth de Firebase** que permite la identificación del usuario mediante correo y contraseña o usando una cuenta de Google. Esta gestión del **Firebase Auth** es utilizada para el **login** y para identificar los usuarios registrados en la aplicación.

SQLite:



La base de datos de **SQLite** facilita el acceso a los datos locales, permitiendo realizar las operaciones de **lectura, escritura, actualización y carga** de manera eficaz e instantánea. Es una buena opción para la gestión de las prácticas.

La aplicación tiene tres tablas creadas para poder manejar las prácticas para el usuario: **Usuario, Preguntas y Opciones**.

- La tabla "Usuario" simplemente guarda información básica del usuario (id y correo).

	id	email
	12	prueba@gmail.com
	14	ni.hon.app.user@gmail.com
	15	admin2@gmail.com

- La tabla "Preguntas" guarda todas las preguntas asociadas a las lecciones que se proporcionan. Está estrechamente relacionada con la tabla de opciones mediante un id. Cada pregunta tiene asignado su propio nivel, que corresponde al nivel del usuario y el de las lecciones.

	level	questionES	questionEN	questionCH	idQuestion	complete	retries	email
1	1	Selecciona el hiragan Select the correct hiragana	选择正确的平假名来	0552a51e4-5eeb-43e0	0	0	0	prueba@gmail.com
2	2	¿Cuál es la traducción? What is the translation?	'猫'在日语中的翻译?	1d4f2f2eb-79b3-43e0	0	0	0	prueba@gmail.com
3	3	Selecciona el kanji correcto Select the correct kanji	选择正确的字来表示	27e204708-59c8-4fc0	0	0	0	prueba@gmail.com
4	1	Selecciona el hiragan Select the correct hiragana	选择正确的平假名来	3fd4e83e9-7288-43e0	0	0	0	prueba@gmail.com
5	2	¿Cuál es la traducción? What is the translation?	'狗'在日语中的翻译?	4b982d79a-4549-44e0	0	0	0	prueba@gmail.com
6	3	Selecciona el kanji correcto Select the correct kanji	选择正确的字来表示	5a76999dd-ad5e-44e0	0	0	0	prueba@gmail.com
7	1	Selecciona el hiragan Select the correct hiragana	选择正确的平假名来	679a3bb4c-2a90-4b00	0	0	0	prueba@gmail.com
8	3	Selecciona el kanji correcto Select the correct kanji	选择正确的字来表示	75e0e1a78-a04b-44e0	0	0	0	prueba@gmail.com
9	2	¿Cuál es la traducción? What is the translation?	'家'在日语中的翻译?	8807d9fab-48fe-47e0	0	0	0	prueba@gmail.com
10	1	Selecciona el hiragan Select the correct hiragana	选择正确的平假名来	9db8a2205-d97d-47e0	0	0	0	prueba@gmail.com

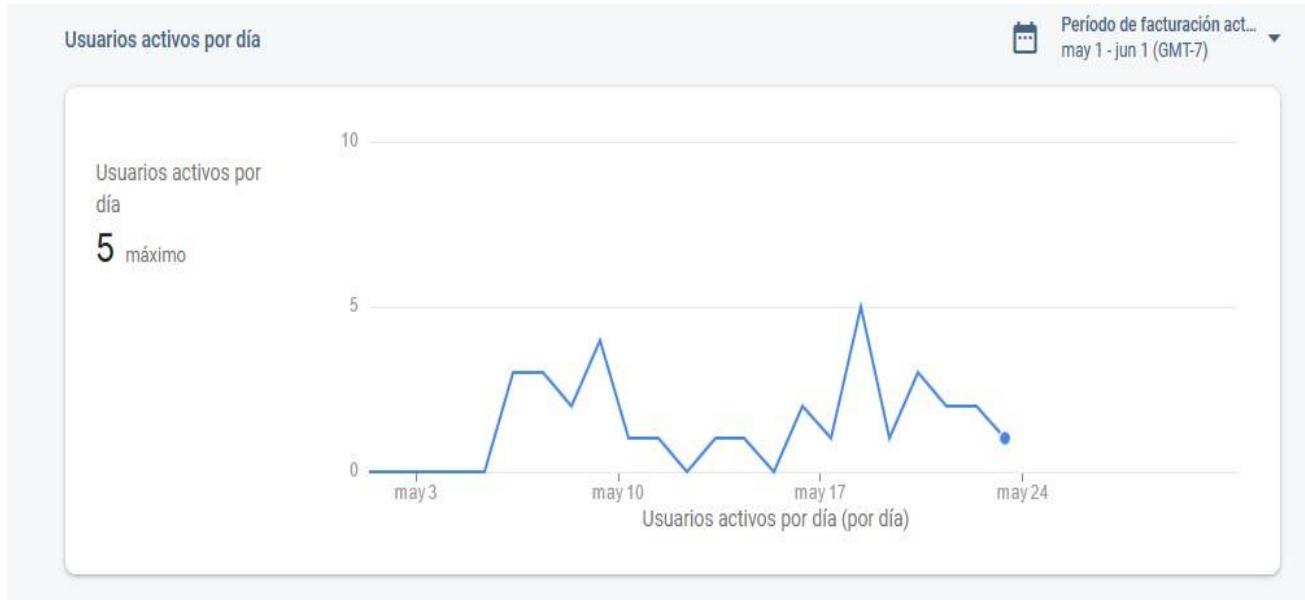
- La tabla "Opciones" muestra las distintas opciones que ofrece a cada pregunta de la tabla anterior y se asocian mediante el id.

	option	rights	idQuestion
1	あ	1	07afcfbe1-ce85-428f-b287-c0e5775f050d
2	い	0	07afcfbe1-ce85-428f-b287-c0e5775f050d
3	う	0	07afcfbe1-ce85-428f-b287-c0e5775f050d
4	え	0	07afcfbe1-ce85-428f-b287-c0e5775f050d
5	いぬ	0	1e63b5160-9ad0-4eed-ad76-b758422b5798
6	ねこ	1	1e63b5160-9ad0-4eed-ad76-b758422b5798
7	さかな	0	1e63b5160-9ad0-4eed-ad76-b758422b5798
8	とり	0	1e63b5160-9ad0-4eed-ad76-b758422b5798

Sobre la gestión y creación de estas tablas se explicarán en más detalle en el apartado de construcción.

2.2.4 Autenticación de usuarios

Para la gestión de la autenticación de los usuarios se ha empleado Firebase Auth. Esta herramienta facilita la gestión de usuarios de la aplicación, proporciona **mayor seguridad y protección**. Además, permite observar los usuarios activos y existentes en una gráfica para los desarrolladores de forma visual.



Como las otras herramientas que ofrece Firebase, al usar el Auth abstrae mucho la complejidad en el desarrollo, ya que permite su uso mediante llamadas al API que proporciona. Además, ofrece y permite la identificación de proveedores de identificación federada como: Google, Facebook, Twitter, etc., para registrarse en el proyecto.

Authentication

Users Sign-in method Templates Usage Settings  Extensiones NUEVA

Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario
admin@gmail.com		9 may 2023	9 may 2023	Op90PMruhgCD3n5xaYETd4YTtat2
alexiamolinamanzanedo@...		8 may 2023	20 may 2023	w2AFcjySP8PCtGBza7t4tQCwFMy2
ni.hon.app.user@gmail.com		6 may 2023	21 may 2023	DQdQJyJuH1ZWCVQmmWMbCTJ...
admin2@gmail.com		6 may 2023	23 may 2023	hUwMmEdsqRRvCKGB2FKvVRmP...

Filas por página: 50 1 - 4 of 4 < >

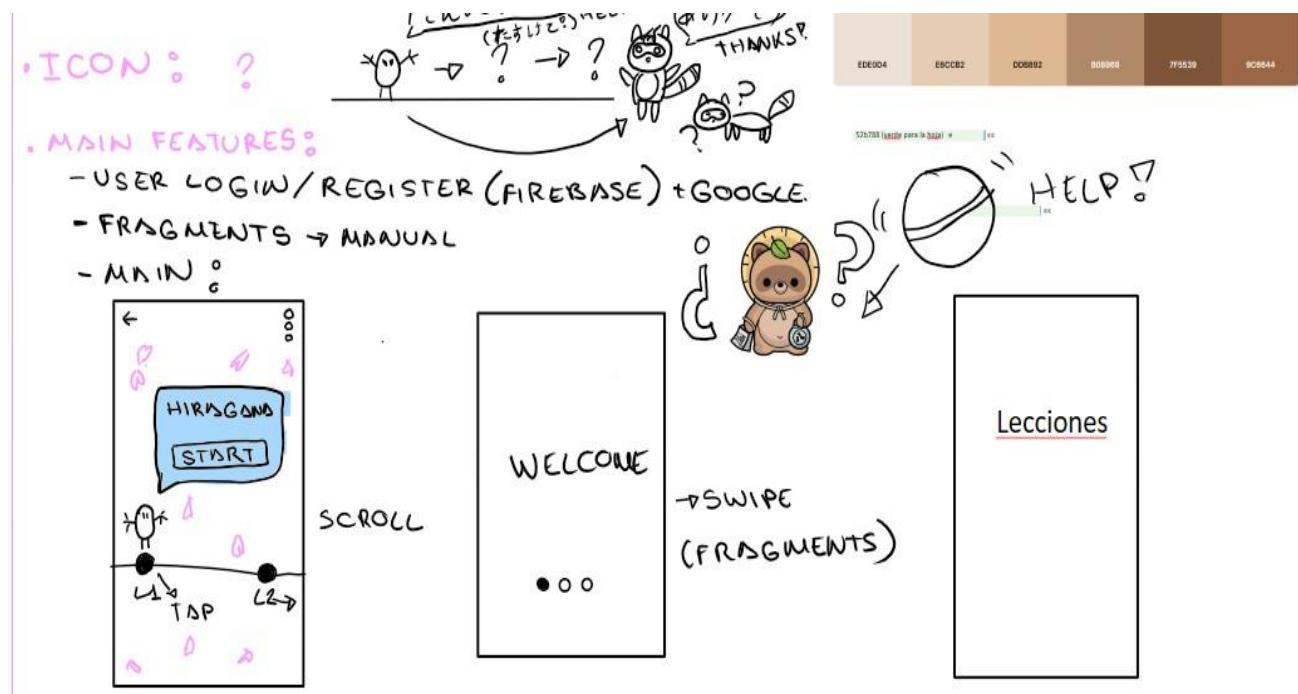
Al usar el Auth que proporciona Firestore abstrae de la complejidad del desarrollo, permitiendo su uso mediante llamadas al api que proporciona. Además, ofrece la identificación de proveedores de identificación federada como: Google, Facebook, Twitter etc.

2 PLANIFICACIÓN DEL PROYECTO

Las tareas para realizar son:

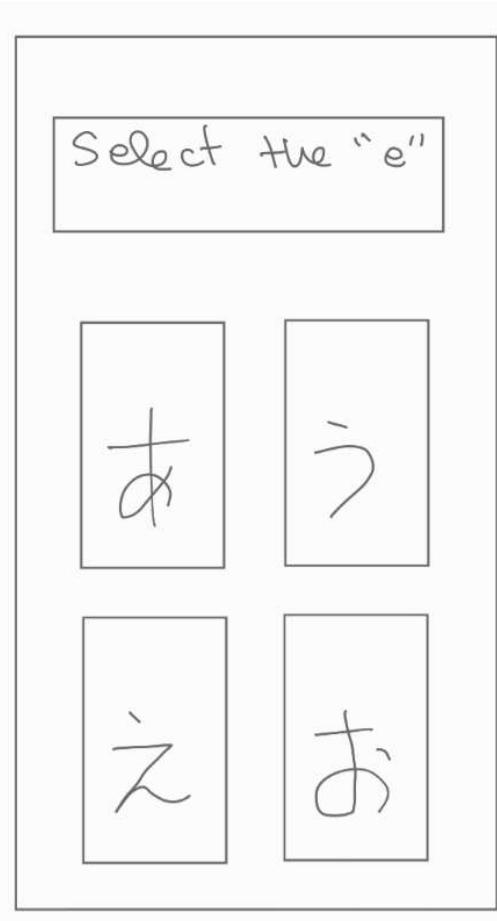
- Elaboración de un prototipo de aplicación a nivel de diseño.

Este fue el primer boceto que se hizo de cara a explicar las funcionalidades principales y la vista general de la aplicación:



- Elaboración de un prototipo de aplicación a nivel funcional
(funcionamiento interno de la aplicación)

Para cubrir este apartado se han realizado bocetos y prototipos a mano de cara a la discusión con el equipo y la tutora del proyecto sobre el diseño de las vistas de la aplicación como por ejemplo de las pantallas principales:



Identificar las tareas o actividades a realizar: **lista de tareas:**

Durante la fase de organización y prototipado de la aplicación, se decidieron unos roles generales de cara a la fragmentación de tareas y al desarrollo:

-Diseño y desarrollo del esqueleto de actividades: Alexia Molina

-Investigación e implantación de BBDD: Wei Shan

-Desarrollo de funcionalidades: Daniel Mora

Aunque los roles no han sido limitantes, ya que al final todos hemos visto y seguido el trabajo del otro, aportado ideas y apoyado.

En cuanto a la fragmentación de tareas y asignación de estas hemos utilizado Slack, tanto como para mantenernos al día de avances como para aumentar el dinamismo de la asignación de tareas.

- Diseño general de la aplicación y especificar requerimientos

Este apartado se ha llevado a cabo mediante la elaboración de prototipos y la búsqueda de una cohesión del diseño a partir de herramientas como Coolors.co para elegir la paleta de colores, flaticon para la elección de paquetes de iconos...

- Diseño de las bases de datos

En cuanto a las bases de datos se ha decidido utilizar una base de datos local para la información usada en las lecciones y ejercicios, y otra basada en firebase de cara al control de puntuación del usuario y la información de log.

- Diseño de las vistas e implementación de estas

Se han diseñado prototipos explicativos tanto estéticos como de funcionalidad para cada pantalla.

- Pruebas de errores y bugs de la aplicación

- Pruebas finales

- Documentación

Determinar los recursos necesarios para cada actividad: lista de recursos.

Asignar los recursos a las tareas.

3 DESCRIPCIÓN DE LA SOLUCIÓN: ANÁLISIS Y DISEÑO

3.1 ESPECIFICACIÓN DE REQUISITOS

Cumpliendo con una de nuestras metas principales, que la aplicación sea accesible para todo el que quiera aprender japonés, los requisitos a nivel de hardware son mínimos, asegurando así que este, prácticamente, al alcance de todos:

- Dispositivos con software Android
- Versión de API de Android 22 o superior

3.2 SELECCIÓN DE PLATAFORMA TECNOLÓGICA

- Desarrollo de la aplicación mediante Android Studio
- App probada en AVD de Android Studio con API 22
- Comprobación de la base de datos y organización mediante una cuenta de Google en la propia página de firebase.

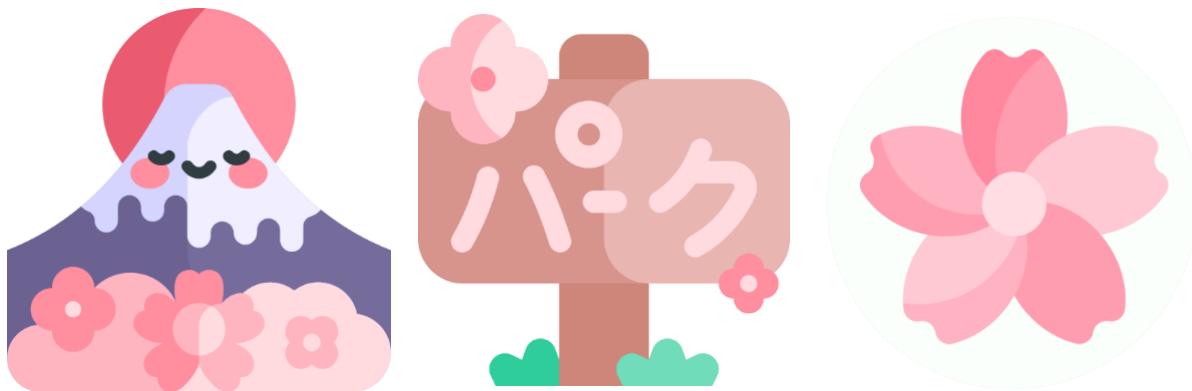
3.3 DESCRIPCION DEL DISEÑO DE LA SOLUCION

3.3.1 DISEÑO GENERAL

Cabe destacar que, para el diseño, retoque y proceso de colores, imágenes e iconos, se ha utilizado la herramienta de dibujo digital “KRITA 5.0”

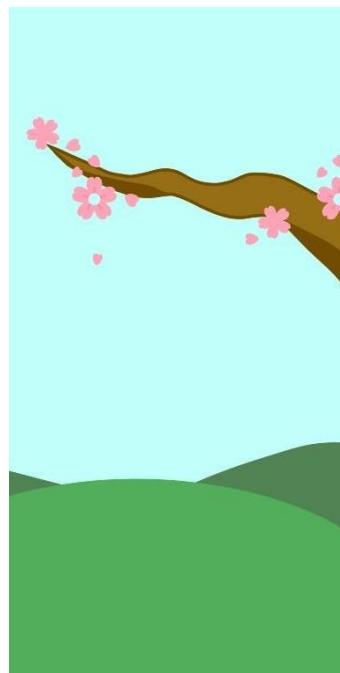


Para el diseño de las pantallas y la estética de la aplicación en general, se ha tenido como principio la búsqueda de un diseño simple, estético y bonito, ya que la principal característica de la aplicación, aparte de su funcionalidad y utilidad principal; aprender japones, es aportar una sensación visual apetecible y encantadora, con un toque adorable que aportan los iconos y la mascota, tanuki, siendo estos iconos reconocibles de Japón como las flores de cerezo, el monte Fuji y demás elementos japoneses de cara a la asociación del usuario de los iconos característicos con el estudio del idioma:



La paleta de colores consta de colores cálidos, en su mayoría rosados, pensados para generar una cohesión general.

En cuanto a la evolución del fondo de la pantalla principal ha habido un claro cambio de colores, se decidió reducir la paleta de colores ya que la primera propuesta contenía colores como el verde y, además, se veía un poco vacía.



Se aplicó la nueva paleta de colores y además se añadieron elementos tan característicos de la cultura japonesa como el cerezo, el torii (puerta sagrada japonesa) y el sol.

Para que en la pantalla principal la cual contiene las lecciones, se entendiese que se puede hacer scroll lateral, decidimos poner una línea discontinua para de esta manera guiar al usuario y que sin palabras ni explicaciones el mismo entienda esa función.

Todos los diseños y dibujos se han hecho a mano para la aplicación, tomando como referencia distintas aplicaciones de estudio de idiomas, de más a menos conocidos, como Duolingo, Renshuu...

El fin último de crear un diseño bonito e intuitivo ha sido hacer más ameno e intuitivo el uso de la aplicación.

3.3.2 CARACTERISTICAS DE LA MASCOTA

La mascota de la aplicación es un Tanuki, un animal característico japonés a juego con la temática de la aplicación.

La principal característica de la mascota es que haciendo avances en las lecciones le ayudas a crecer y evolucionar, desbloqueando nuevos aspectos.

Esta característica está pensada para generar un efecto positivo en el usuario y mantenerle con la curiosidad de saber en qué va a evolucionar, haciendo así, más lecciones.

Estas son las diferentes etapas del Tanuki en esta fase del desarrollo:

Cuando acabas de crear tu cuenta esta así:



(bebé)

Después de completar la primera lección crece y cambia de postura:



(adulto)

Y finalmente al acabar la segunda práctica y de manera provisional, se desbloquea otro cambio de postura:



(adulto 2)

La distribución de cómo evoluciona el Tanuki ahora mismo es completamente provisional, ya que en una fase más madura se desbloquearían cada 3 o 4 lecciones o en base a los aciertos, experiencia y nivel del usuario (valores almacenados en la base de datos) viéndose esta distribución alterada de cara a mostrar el correcto funcionamiento de esta característica.

3.3.3 MAIN SCREEN

La pantalla principal consta de varios elementos destacables:



Al crear una cuenta, esta es la pantalla que aparece.

Como se puede observar hay tres elementos importantes en la pantalla principal.

En el centro está el cuadrado que da acceso a la lección y a las prácticas, mostrando el título de lo que se va a aprender en esa lección y el botón de empezar.

A la izquierda se encuentra la mascota de la aplicación, con la fase en la que este desbloqueado, que se posiciona en la lección (que simboliza el botón con el número) que sea pulsada (siempre que este desbloqueada) con una animación sutil de bajada.

Arriba del todo se encuentra la información del usuario. El username almacenado en la base de datos y el icono elegido.

Al pulsar en tu usuario, se despliega la información del usuario:



Esta pantalla muestra el nombre del usuario, la puntuación, el nivel (con una barra circular que muestra en otro color el progreso) y la opción de borrar la cuenta con un ícono abajo a la derecha.

Al pulsar en el ícono se despliega un menú que permite al usuario elegir entre estos cuatro:



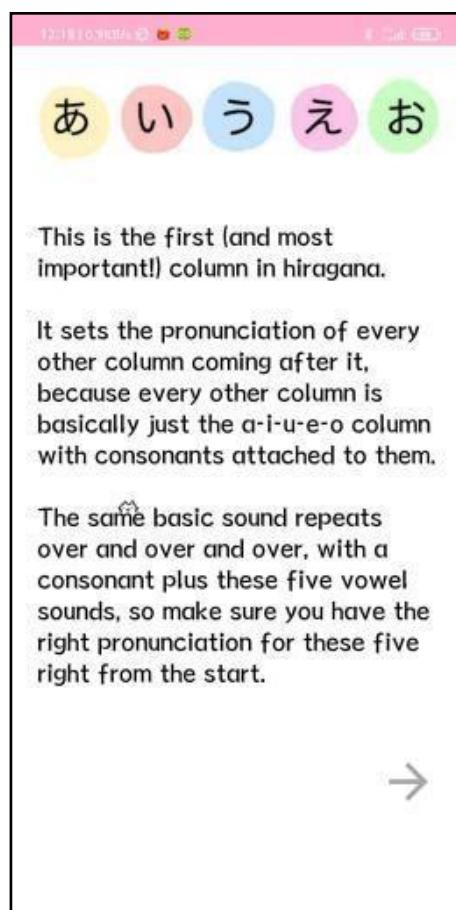
Esta opción se implantó para dar mayor sensación de customización de los perfiles de usuario, permitiendo elegir entre iconos del mismo set, para asociarlo con la aplicación, y guardándolo como campo del usuario en la base de datos.

Como se puede apreciar estamos muy enfocados en que el diseño artístico trabaje en consonancia en la aplicación para poder aportar la mayor armonía posible, por eso, los iconos son seleccionables y con estilos uniformes.

Se probó la idea de que el usuario pueda introducir su propia imagen como foto de perfil, pero en la mayoría de los casos llamaba mucho la atención debido al choque que se produce entre el diseño de la aplicación y la foto de perfil.

Está previsto añadir iconos nuevos desbloqueables para mayor sensación de progreso en la aplicación ya que el componente psicológico es muy importante para tener más posibilidades de que el usuario no se canse de la app.

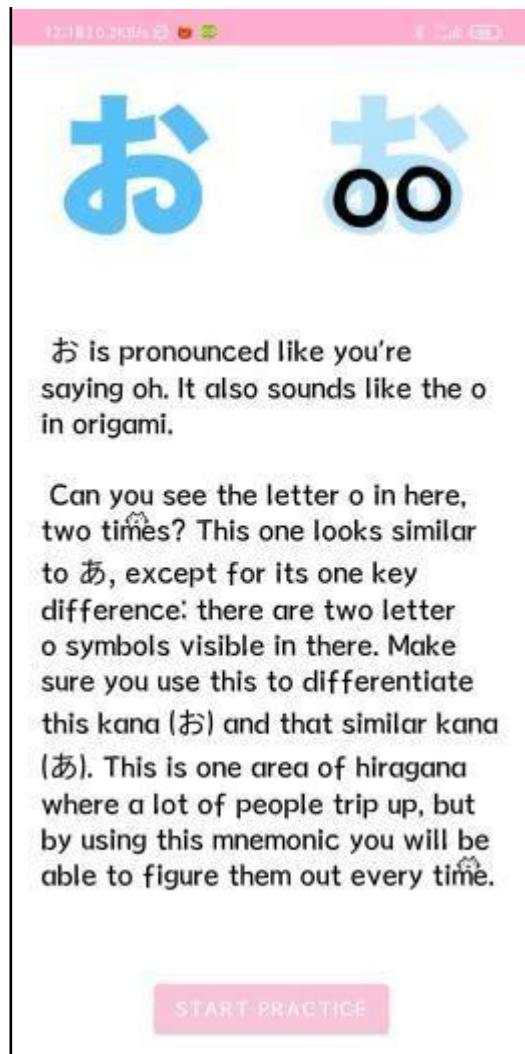
Al pulsar en el botón de start de la pantalla principal, se despliega la lección:



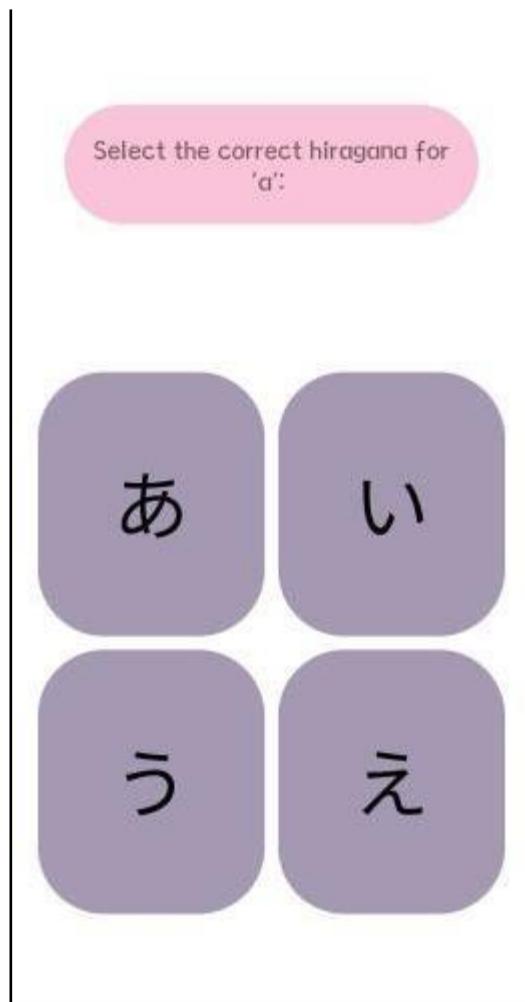
Principalmente consta del texto con la explicación de la lección y una imagen orientativa para facilitar el aprendizaje.

Como indica la flecha de abajo a la derecha, el texto continúa haciendo swipe hacia la derecha, teniendo las siguientes pantallas distinta información sobre la lección.

Se barajaron varias opciones para hacer entender al usuario de que se puede hacer swipe, como por ejemplo enumerar las páginas de las lecciones en la parte inferior derecha para dar la sensación de que la lección es un libro, pero no se entendía, por eso una flecha indicándolo queda bien y es mucho más directo.



Al llegar a la última pantalla de información, se oculta la flecha de swipe y se muestra un botón que te permite empezar la práctica.



La primera pantalla de prácticas es estática, cambia el texto que se muestra, y como se puede ver, es un diseño sencillo, intuitivo e interactivo, ya que te pide que hagas clic en el botón correspondiente.

Al pulsar en el botón hay dos opciones:

-Que la respuesta sea correcta, en cuyo caso el botón de pinta de verde y aparece una animación dinámica de un tick verde que da paso a la siguiente pregunta.

-Que la respuesta sea incorrecta, en cuyo caso el botón se pinta de rojo y no muestra ninguna animación.

No se pasa a la siguiente pregunta hasta que se haya acertado la respuesta.



4. DESCRIPCION DE LA SOLUCIÓN: CONSTRUCCIÓN

4.1 Documentación descriptiva

- La documentación descriptiva de la solución se incluye texto, capturas de pantallas, muestras de código y en general toda la información que ilustre la construcción de los puntos claves de la solución realizada.
- NO se inundará la memoria con el código completo de la solución, el código de la solución está presente en anexo.

4.1.1. Inicio de la Aplicación: Splash Activity.

El **splash Activity** (la primera pantalla que aparece siempre al iniciar la aplicación, a modo de indicar que está cargando y que se ha iniciado).

Es una pantalla **simple y estática**, con el logotipo de la aplicación en el centro y un mensaje de ánimo en japones.

Pasado unos segundos la pantalla pasa automáticamente a la pantalla de inicio de sesión, o a la principal en caso de haberse logado anteriormente el usuario.

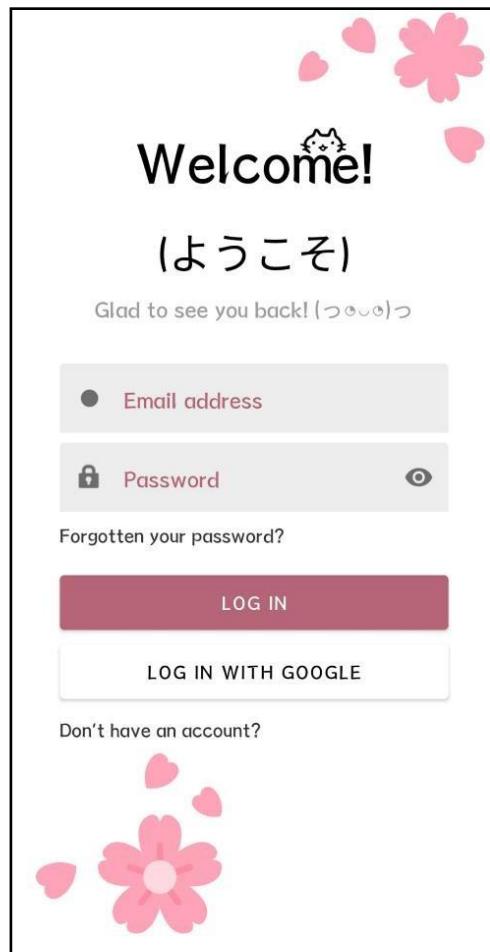


4.1.2. Inicio de Sesión: Login Activity.

El **Inicio de Sesión** consta de una pantalla con dos campos para la introducción del correo y la contraseña, usando el "TextInput Layout" junto al "TextInput EditText" ofrecidos por la librería de **Google Material**.

Dos botones permiten la identificación del usuario para **iniciar sesión** y otro permite el inicio mediante una cuenta de **Google**.

Y dos "TextView" se utilizan como enlaces que permiten dirigir al **registro del usuario** y la **recuperación de la contraseña**. Además, en la pantalla se presentan diversos elementos visuales para ofrecer al usuario una experiencia más agradable.



Métodos permitidos para iniciar sesión:

- Sesión Normal (correo y contraseña).
- Sesión con Google.

Sesión Normal:

Firebase ofrece sus **propios métodos** para la **identificación del usuario**. Solo es necesario hacer llamadas al Firebase Auth para que compruebe si existe el correo y contraseña introducido.

Para un **mayor control de los errores** en la llamada, se **evitará** dejar los **campos en blanco**. Una vez que la identificación sea correcta, se dirigirá al usuario a la pantalla principal y se **guardará** su **sesión** para mantenerla iniciada. En caso contrario, se informará al usuario con un mensaje.

- Para evitar campos en blanco:

```
login.setOnClickListener(v -> {
    if(!email.getText().toString().isEmpty()&&!password.getText().toString().isEmpty()){
        }else{
            Toast.makeText( context: this, text: "Campos vacíos",Toast.LENGTH_LONG).show();
        }
    }
```

- El método que ofrece Firebase para el inicio de sesión usando el Auth usando “**signInWithEmailAndPassword()**”:

```
firebaseAuth.getInstance().signInWithEmailAndPassword(email.getText().toString().trim()
    , password.getText().toString().trim()).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){

    }}
```

- Para el guardado de sesión del usuario que ha iniciado el login se usa “**SharedPreferences**” en ella se guarda la información del usuario, concretamente el correo que se inició la sesión:

```
SharedPreferences sharedpreferences = getSharedPreferences( name: "UserSession", MODE_PRIVATE);
SharedPreferences.Editor editor = sharedpreferences.edit();

editor.putString("token", email.getText().toString().trim());
editor.apply();
```

Sesión con Google:

Firebase también ofrece la posibilidad de iniciar sesión con una cuenta de Google usando la autenticación que presenta Google Service.

Para poder hacer la llamada al servicio de autenticación de Google, se **necesitará una identificación del cliente** que genera automáticamente Firebase en un archivo JSON. Con ello, solo será necesario crear una instancia del cliente con acceso al servicio de autenticación de Google.

El archivo json necesario para poder iniciar sesión con Google se obtiene accediendo al Firebase Console:



Los métodos que ofrece Firebase para iniciar sesión con Google es el “**SignInWithCredential()**”:

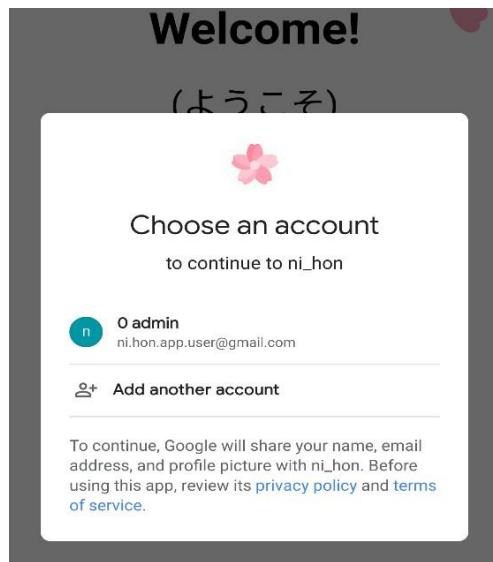
```
INECO(weishan)
  FirebaseAuth.getInstance().signInWithCredential(credential).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @ INECO(weishan
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){
            Log.d(TAG, msg: "signInWithCredential:success");
```

Para que el método anterior funcione se necesita haber **instanciado una identificación del cliente** de acceso a los servicios de autenticación que tiene Google. Esto se realiza mediante el siguiente código:

```
GoogleSignInOptions gso= new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestIdToken("289735160416-nbt9sig9pai0fgofmq503gfnirvljqn1.apps.goog...")
    .requestEmail()
    .build();

mGoogleSignInClient = GoogleSignIn.getClient(activity, gso);
```

Tras realizar lo anterior, solo es necesario esperar la respuesta del servicio ofrecido. Dependiendo de esa respuesta, se iniciará sesión o no.

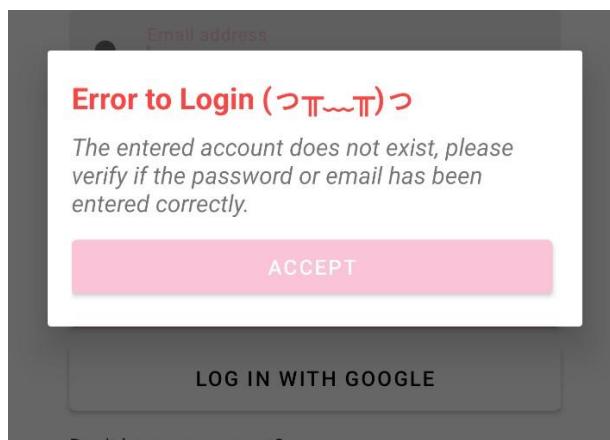


Una vez que el usuario se ha identificado correctamente con su cuenta de Google, se verificará si es la primera vez que inicia sesión.

En caso de ser la **primera vez**, se creará un nuevo campo en la base de datos de **Firebase** y **SQLite** con su información correspondiente que lo identifique. (Para ello se comprobará si existe en la base de datos). La implementación en código es la siguiente:

```
query=userCollRef.whereEqualTo( field: "email",emailGoogle);
query.get().addOnCompleteListener(task1 -> {
    if(task1.isSuccessful()&& task1.getResult().isEmpty()){
        userCollRef.document(GoogleUserId) DocumentReference
            .set(GoogleUser) Task<Void>
                .addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task1) {
                        if(task1.isSuccessful()){
                            Log.d(TAG, msg: "Se ha guardado el usuario Google con id: "+GoogleUserId);
                        }
                    }
                });
}
});
```

En caso contrario, se notificará al usuario con un mensaje indicando el fallo al iniciar sesión.

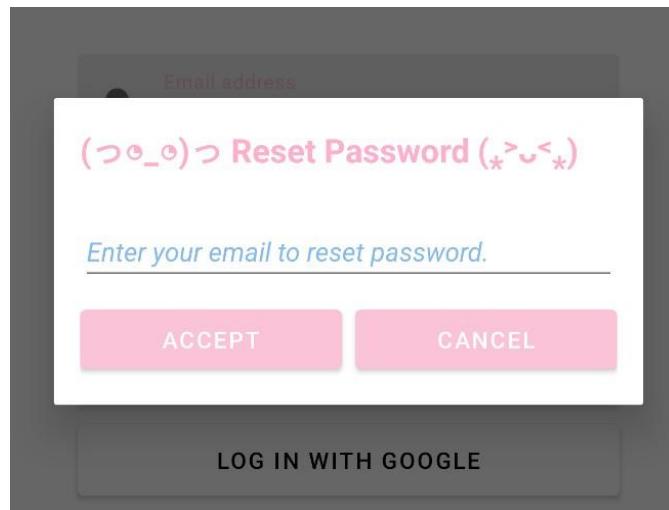


Recuperación de contraseña:

Firebase también ofrece **métodos** que permiten al usuario **restablecer su contraseña** en caso de olvido. Todo el proceso es gestionado por los **servicios de Firebase**.

En la aplicación para restablecer la contraseña del usuario, se ha escogido lo siguiente:

En la pantalla de Login está presente la frase "**Forgotten your password?**" siendo un **enlace** que al accionar muestra un mensaje que pide al usuario el correo para restablecer la contraseña.



El mensaje anterior mostrado es un **AlertDialog personalizado** con un campo editable y dos botones.

```
    + usage = new ECO(wishList);
public void showDialog() {

    View dialogView = LayoutInflater.from( context: this).inflate(R.layout.resetpwd_alertdialog, root: null);
    Button accept = dialogView.findViewById(R.id.button_rrok);
    Button dismiss = dialogView.findViewById(R.id.button_rno);
    EditText remail = dialogView.findViewById(R.id.resetPswd_email);

    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setView(dialogView);
    alertDialog = builder.create();
    alertDialog.setCancelable(false);
    alertDialog.show();
}
```

Los dos botones presentan funciones distintas:

- una simplemente cierra el mensaje.
- La otra envía un mensaje al correo introducido por el usuario para restablecer contraseña.
No se permite la introducción de un campo vacío, si acciona mostrará un mensaje en rojo.

```
accept.setOnClickListener(v -> {
    String sendEmail = remail.getText().toString().trim();
    if (!sendEmail.isEmpty()) {
        SendResetEmail(sendEmail);
    } else {
        remail.setHintTextColor(getColor(android.R.color.holo_red_light));
        remail.setHint(R.string.dialogEmptyEmail);
    }
});
```

Para el envío del correo de restablecimiento de contraseña, se usa las funciones que ofrece Firebase siendo “**sendPasswordResetEmail()**”. Con llamar a este método basta para enviar el correo, sin tener que gestionar nada más, ya que Firebase lo gestiona para el desarrollador. El código es el siguiente:

```
1 usage  • INECO\wei.shan
public void SendResetEmail(String email) {
    mAuth.sendPasswordResetEmail(email).addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            alertDialog.dismiss();
            Toast.makeText(context, text: "Verifique su email", Toast.LENGTH_SHORT).show();
        } else {
            alertDialog.dismiss();
            Toast.makeText(context, text: "Error al enviar el correo", Toast.LENGTH_SHORT).show();
        }
    });
}
```

Cabe destacar que Firebase **permite la personalización** del correo que se envía además del formato estándar. Para ello se modifica dentro de Firebase Console los textos en HTML, el mensaje optado a enviar es el siguiente:

Asunto

Cambia la contraseña de %APP_NAME%

Mensaje

Hola,

¡Te saludamos cordialmente desde el equipo de %APP_NAME%!

Hemos recibido una solicitud para restablecer la contraseña de tu cuenta de correo electrónico %EMAIL%. No te preocupes, estamos aquí para ayudarte.

Por favor, haz clic en el siguiente enlace para restablecer tu contraseña:

[Pinche aquí](#)

Si no has solicitado restablecer tu contraseña, puedes ignorar este correo electrónico.

Gracias por confiar en %APP_NAME%.

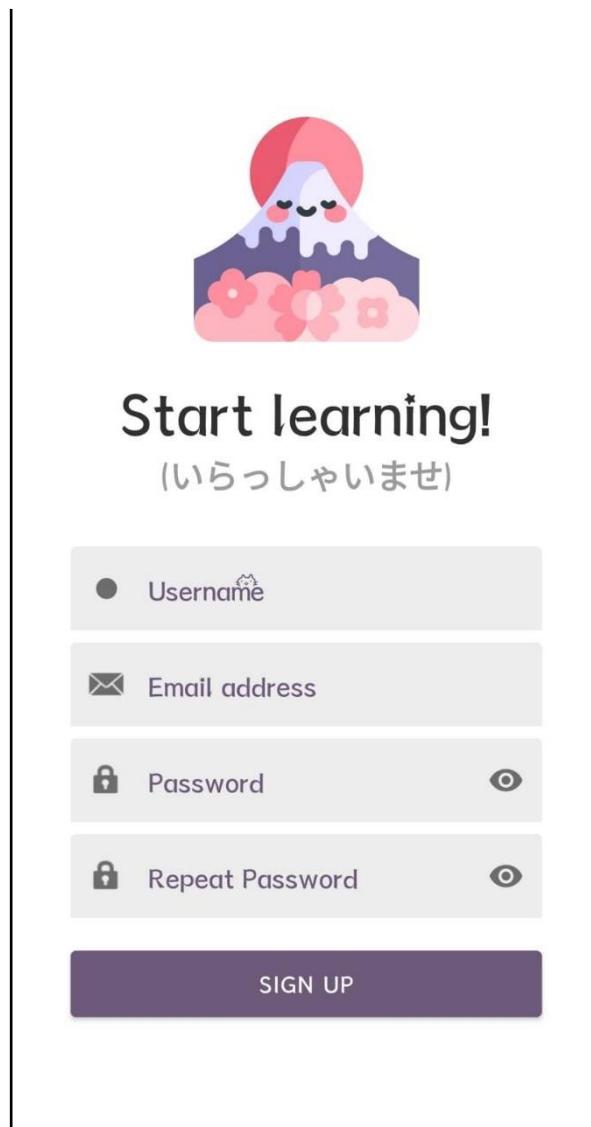
Atentamente,

El equipo de %APP_NAME%

5.1.3. Registrar Cuenta: Sign Up Activity.

La pantalla de registro consta de un formulario con cuatro campos a llenar: **correo, nombre, contraseña** y su repetición.

Más un botón para el guardado del registro, además de diversos estilos proporcionados para una **mayor estética visual**.



En el proceso de gestión de errores al registrar el usuario en la base de datos, se maneja los siguientes casos:

- Haber introducido algún campo vacío.
- Una contraseña inferior a 6 dígitos no será permitida.
- La desigualdad en las contraseñas introducidas por el usuario.

Esto se restringe mediante el siguiente código:

```
if(name.isEmpty()||email.isEmpty()||pswd.isEmpty()||rpswd.isEmpty()){
    Toast.makeText( context: this, text: "Hay campos vacíos",Toast.LENGTH_LONG).show();
} else if(!pswd.equals(rpswd)){
    Toast.makeText( context: this, text: "Las contraseñas introducidas no coinciden",Toast.LENGTH_LONG).show();
} else if (pswd.length()<6){
    Toast.makeText( context: this, text: "La contraseña debe no puede ser inferior a 6 digitos",Toast.LENGTH_LONG).show();
```

Una vez que el usuario complete los campos correctamente, se llevará a cabo el proceso para llamar a Firebase para guardar el usuario en la base de datos usando el método “**createUserWithEmailAndPassword()**”.

Se tiene en cuenta que **solo se guardará el usuario** en la base de datos cuando su correo **no tenga una cuenta asociada** en la aplicación. Esto se gestiona haciendo una consulta en la base de datos para verificar si el correo existe en ella, usando “**Query**” (consultando por el correo).

Código usado para lograr lo anterior:

```
query=userCollRef.whereEqualTo( field: "email", emails);
    ↳ INECO\wei.shan +2
query.get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
    ↳ Wei +2
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
        if(task.isSuccessful()&&task.getResult().isEmpty()){
            myLoginAuth.createUserWithEmailAndPassword(newUser.getEmail(),newUser.getPassword()).addOnCompleteListener(mtask->{
                if(mtask.isSuccessful()){
                    String userId = myLoginAuth.getCurrentUser().getUid();
                    userCollRef.document(userId).DocumentReference
                        .set(UserToAdd) Task<Void>
                    ↳ Wei +2
                    .addOnSuccessListener(new OnSuccessListener<Void>() {
                        ↳ Wei +2
                        @Override
                        public void onSuccess(Void aVoid) {
```

Una vez registrado correctamente, se retornará a la pantalla de login con los valores correo y contraseña.

Tras un retardo de 3 segundos iniciará el login automáticamente sin necesidad de que el usuario vuelva a introducir el correo y la contraseña **agilizando el proceso de inicio de sesión**.

```
intent.putExtra( name: "email",emails);
intent.putExtra( name: "pswd",pswd);
```

En la pantalla de login se recibe el correo y la contraseña del registro e inicia sesión con ella en caso de no recibirla se ignora, el código usado es el siguiente:

```
Intent intent=getIntent();
String emailR=intent.getStringExtra( name: "email");
String passwdR=intent.getStringExtra( name: "pswd");
if(emailR!=null&&passwdR!=null){

    try{
        Thread.sleep( millis: 3000);
        ToMain();
    }catch(InterruptedException e){}
}
```

En caso contrario, se mostrará al usuario un mensaje del motivo de fallo al registrar.

4.1.4. Pantalla Principal: Main Activity.

La pantalla principal de la aplicación es donde se muestran las diferentes lecciones, y está modelada e implementada en MainActivity.java



Los objetivos que se han perseguido a la hora de crear la actividad principal son la escalabilidad del código y el no tener que realizar modificaciones a la hora de añadir nuevas lecciones.

Para cumplir con este objetivo se ha utilizado un **RecyclerView**

```
2 usages
private MainLessonAdapter lessonAdapter;
5 usages
private static RecyclerView recyclerView;

+ daniel-mora-moreno +2
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    setTitle(null);

    recyclerView = findViewById(R.id.recycler);

    lessonAdapter = new MainLessonAdapter(context: this, lessons);
    recyclerView.setHasFixedSize(true);
    recyclerView.setLayoutManager(new MyLinearLayoutManager(context: this, MyLinearLayoutManager.HORIZONTAL, reverseLayout: false));
    recyclerView.setAdapter(lessonAdapter);
}
```

El uso del RecyclerView permite mostrar cada sección como una misma plantilla que se repite horizontalmente y a través de la cual se puede navegar mediante el scroll manual.

Todos los elementos del RecyclcerView se modelan utilizando el mismo layout, modelado como item_component.xml

La información básica de las lecciones se carga directamente en un ArrayList.

Se valoró crear una base de datos adicional independiente de la de usuarios para almacenar esta información, pero no se consideró necesario.

La información incluida en este array es:

- El identificador numérico de la lección.
- El título de la lección.
- Una pequeña descripción del contenido.
- El número de páginas que contiene la lección.

```
public static ArrayList<Lesson> Lessons = new ArrayList<Lesson>(){
{
    add(new Lesson( id: 1, title: "Hiragana", popupText: "Let's learn hiragana!", pages: 3));
    add(new Lesson( id: 2, title: "Katakana", popupText: "Let's learn katakana!", pages: 1));
    add(new Lesson( id: 3, title: "Lesson 3", popupText: "3 is bigger than 2", pages: 1));
    add(new Lesson( id: 4, title: "Lesson 4", popupText: "Sample text", pages: 1));
    add(new Lesson( id: 5, title: "Lesson 5", popupText: "I don't know what to write", pages: 1));
}
};
```

De esta información, el título y la descripción se transfieren a los elementos del RecyclerView y se muestran en un pequeño “popup” cuando el usuario presiona el botón de la lección correspondiente.

El identificador y el número de páginas se transfieren a las propias lecciones, lo que permite obtener la información correspondiente para llenar dichas lecciones.

MainLessonAdapter:

Para hacer uso de la vista RecyclerView que contiene los botones desde los que se accede a las lecciones, se extiende la clase **Adapter**.

Esto permite crear un adaptador personalizado para manejar la funcionalidad de los diferentes elementos.

```
4 usages  ▲ daniel-mora-moreno
public class MainLessonAdapter extends RecyclerView.Adapter<MainLessonAdapter.LessonViewHolder> {

    7 usages
    private final ArrayList<Lesson> lessons;

    4 usages
    private int previousId;
    6 usages
    private int currentId;
    3 usages
    private Context context;

    1 usage  ▲ daniel-mora-moreno
    public MainLessonAdapter(Context context, ArrayList<Lesson> lessons) {
        this.lessons = lessons;
        this.previousId = -1;
        this.currentId = -1;
        this.context = context;
    }
}
```

Una de las peculiaridades de este RecyclerView es que cada elemento consta de dos botones, uno para mostrar la información básica de la lección en forma de pop-up, y otro ya dentro de dicho pop-up que permite al usuario entrar en la propia lección.

Para implementar esto, se ha modificado el método **onCreateViewHolder** del adaptador, añadiendo dos métodos Listener para los dos botones:

onLessonClick es el método que permite mostrar la información de la lección y el botón para acceder a la misma.

También mueve a Tanuki a la lección correspondiente y oculta la información del resto de lecciones si es que ya se estaba mostrando en alguna.

onStartClick inicia la lección en una nueva actividad, y le transfiere los datos de la lección correspondiente en forma de objeto Lesson.

```


    * daniel-mora-moreno
    @NotNull
    @Override
    public LessonViewHolder onCreateViewHolder(@NotNull ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_component, parent, attachToRoot: false);
        * daniel-mora-moreno
        LessonViewHolder lvh = new LessonViewHolder(itemView, new MyClickListener());
        1 usage * daniel-mora-moreno
        public void onLessonClick(int p) {
            previousId = currentId;
            currentId = lessons.get(p).getId() - 1;
            MainActivity.getRecyclerView().getAdapter().notifyItemChanged(previousId);
            MainActivity.getRecyclerView().getAdapter().notifyItemChanged(currentId);
        }
        1 usage * daniel-mora-moreno
        public void onStartClick(int p) {
            Intent intent = new Intent(context, LessonActivity.class);
            intent.putExtra(name: "Lesson", lessons.get(p));
            context.startActivity(intent);
        }
    );
    return lvh;
}


```

Estos métodos se asocian a los botones por medio del identificador, permitiendo de esta forma dar a cada uno la funcionalidad correspondiente:

```


3 usages 1 implementation * daniel-mora-moreno
public interface MyClickListener {
    1 usage 1 implementation * daniel-mora-moreno
    void onLessonClick(int p);
    1 usage 1 implementation * daniel-mora-moreno
    void onStartClick(int p);
}


```

Este código que asigna la funcionalidad a los botones pertenece a la clase LessonViewHolder, que se explica más adelante:

```


    * daniel-mora-moreno
    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.button) {
            listener.onLessonClick(this.getLayoutPosition());
        }
        if (v.getId() == R.id.startLessonButton) {
            listener.onStartClick(this.getLayoutPosition());
        }
    }
}


```



En esta imagen se pueden ver ambos botones y los elementos cargados al presionar el botón que tiene el método onLessonClick:

- La imagen de Tanuki correspondiente al nivel de usuario.
- El popup, que contiene:
 - o El título de la lección.
 - o El texto descriptivo. En este caso las letras que se estudian en la lección.
 - o El botón de inicio de la lección, que tiene asociado el método onStartClick.

Del mismo modo, el método **onBindViewHolder** se ha modificado para que al notificar al elemento tras presionar un botón de lección (ver método `onLessonClick` en la imagen previa), se visibilice en pantalla la información de la lección presionada y se oculte la de la anterior, si es que ya se estaba mostrando alguna.

```
@Override  
public void onBindViewHolder(@NonNull LessonViewHolder holder, int position) {  
    Lesson lesson = lessons.get(position);  
    holder.button.setText(Integer.toString(lesson.getId()));  
    holder.lessonTitleText.setText(lessons.get(position).getTitle());  
    holder.popupText.setText(lessons.get(position).getPopupText());  
    if (position != currentId){  
        holder.popup.setVisibility(View.INVISIBLE);  
    }  
    if (position == currentId){  
        MainActivity.getRecyclerView().smoothScrollToPosition(currentId);  
        holder.popup.setVisibility(View.VISIBLE);  
        holder.popupButton.setVisibility(View.VISIBLE);  
    }  
}
```

Dentro de la clase `MainLessonAdapter` está también la clase **LessonViewHolder**, que extiende **ViewHolder**.

En esta se asignan los identificadores de los layouts a los diferentes componentes del elemento del RecyclerView para así añadirles funcionalidad.

```
3 usages 1 implementation  daniel-mora-moreno  
public interface MyClickListener {  
    1 usage 1 implementation  daniel-mora-moreno  
    void onLessonClick(int p);  
    1 usage 1 implementation  daniel-mora-moreno  
    void onStartClick(int p);  
}
```

```
▲ daniel-mora-moreno
public static class LessonViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {

    MyClickListener listener;
    Button button, popupButton;
    ConstraintLayout popup;
    TextView popupText, lessonTitleText;

    ▲ daniel-mora-moreno
    public LessonViewHolder(View itemView, MyClickListener listener) {
        super(itemView);
        button = itemView.findViewById(R.id.button);
        popupButton = itemView.findViewById(R.id.startLessonButton);
        popup = itemView.findViewById(R.id.popup);
        lessonTitleText = itemView.findViewById(R.id.lessonTitleText);
        popupText = itemView.findViewById(R.id.popupText);

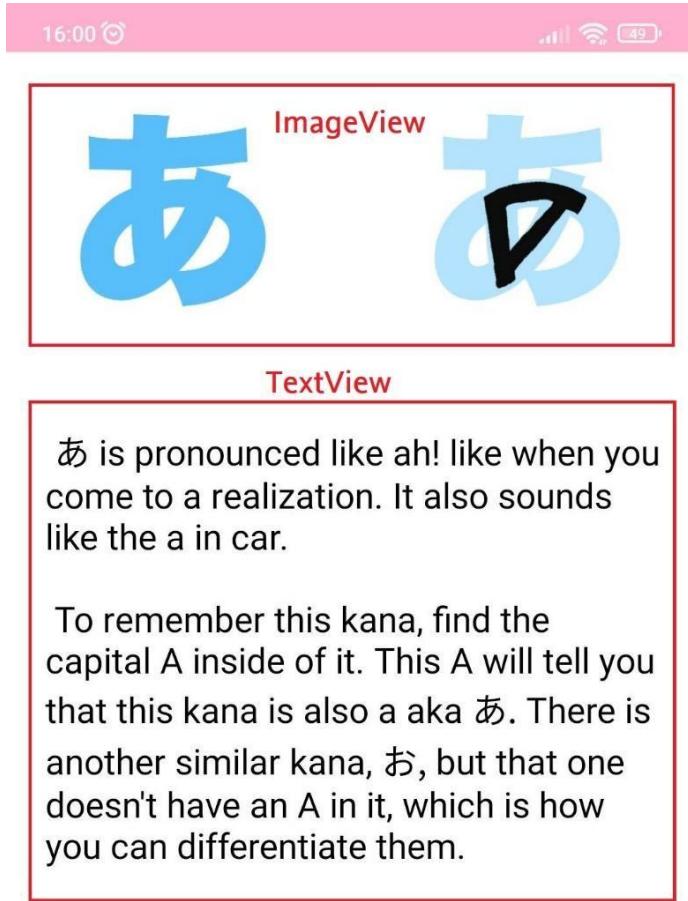
        this.listener = listener;

        button.setOnClickListener(this);
        popupButton.setOnClickListener(this);
    }

    ▲ daniel-mora-moreno
    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.button) {
            listener.onLessonClick(this.getLayoutPosition());
        }
        if (v.getId() == R.id.startLessonButton) {
            listener.onStartClick(this.getLayoutPosition());
        }
    }
}
```

4.1.5. Pantalla de Lecciones: Lesson Activity.

A la pantalla de lección se accede desde el menú principal, y en esta se muestran los diversos contenidos, en forma de texto y acompañados de imágenes.



La lección está modelada en Fragments entre los que el usuario puede alternar deslizándose horizontalmente para leer y estudiar la información.

Nuevamente con el objetivo de crear una aplicación que sea escalable en el futuro, todas las lecciones utilizan el mismo archivo .xml y la misma actividad (LessonFragment.java) como base, cambiandosolamente la información representada en ellas, pero no la estructura.

LessonActivity es la clase principal de las lecciones:

```
▲ daniel-mora-moreno +2
public class LessonActivity extends AppCompatActivity {

    private LessonPagerAdapter pagerAdapter;

    ▲ daniel-mora-moreno +2
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lesson);

        LanguageHelper.setLocale( context: this, LanguageHelper.getLanguage( context: this));

        Intent intent = getIntent();
        Lesson lesson = (Lesson) intent.getSerializableExtra( name: "lesson");

        //fragments
        pagerAdapter = new LessonPagerAdapter( context: this, getSupportFragmentManager(), lesson);
        ViewPager viewPager = findViewById(R.id.view_pager);
        viewPager.setAdapter(pagerAdapter);

    }
}
```

LessonFragment extiende la clase Fragment e implementa los métodos básicos de la misma:

```
▲ daniel-mora-moreno +1 *
public class LessonFragment extends Fragment {

    private static final String LESSON = "lesson";
    private static final String PAGE = "page";
    private int lesson;
    private int page;

    new*
    public LessonFragment() { }

    ▲ daniel-mora-moreno
    public static LessonFragment newInstance(int position) {
        LessonFragment fragment = new LessonFragment();
        Bundle args = new Bundle();
        args.putInt(PAGE, position);
        fragment.setArguments(args);
        return fragment;
    }

    ▲ daniel-mora-moreno
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            page = getArguments().getInt(PAGE);
            lesson = getArguments().getInt(LESSON);
        }
    }
}
```

En esta clase se recibe la información de en qué lección y qué página nos encontramos, de forma que el método OnCreate puede buscar los recursos necesarios (la imagen y el texto correspondiente a la página) y asociarlos a los elementos del layout fragment_lesson.xml.

Tanto las imágenes como los textos están guardados bajo el nombre “lesson_” seguido del id de la lección y número de la página.

Para poder cargar la imagen desde java, se ha hecho uso de la clase Glide.

```
@Override  
public View onCreateView(LayoutInflater inflater, ViewGroup container,  
    Bundle savedInstanceState) {  
  
    View view = inflater.inflate(R.layout.fragment_lesson, container, attachToRoot: false);  
  
    TextView text = view.findViewById(R.id.text);  
    String textId = "lesson_" + lesson + "_" + page;  
    int textResId = getResources().getIdentifier(textId, defType: "string", getActivity().getPackageName());  
    text.setText(getString(textResId));  
  
    ImageView image = view.findViewById(R.id.image);  
    String imageId = "lesson_" + lesson + "_" + page;  
    int imageResId = getResources().getIdentifier(imageId, defType: "drawable", getActivity().getPackageName());  
    Glide.with( fragment: this).load(imageResId).into(image);
```

La lección contiene un botón que sólo aparece en la última página y que permite al usuario acceder al apartado de práctica.

Para que el botón se muestre solamente en la última página de la lección, se han utilizado los siguientes métodos:

```
Button startPractice= view.findViewById(R.id.start_button);  
  
Intent intent = getActivity().getIntent();  
  
if (getArguments() != null && intent != null) {  
    Lesson lesson = (Lesson) intent.getSerializableExtra( name: "lesson");  
    if (getArguments().getInt(PAGE) ==lesson.getPages()) {  
        startPractice.setVisibility(View.VISIBLE);  
    }  
}  
  
startPractice.setOnClickListener(v->{  
    SharedPreferences pref=getActivity().getSharedPreferences( name: "PRACTICE", MODE_PRIVATE);  
    SharedPreferences.Editor editor = pref.edit();  
    editor.putInt("cnt",0);  
    editor.putInt("accessLevel",lesson);  
    editor.apply();  
  
    Intent intent1=new Intent(getActivity(), Practice1.class);  
    startActivity(intent1);  
    getActivity().finish();  
});  
  
return view;
```

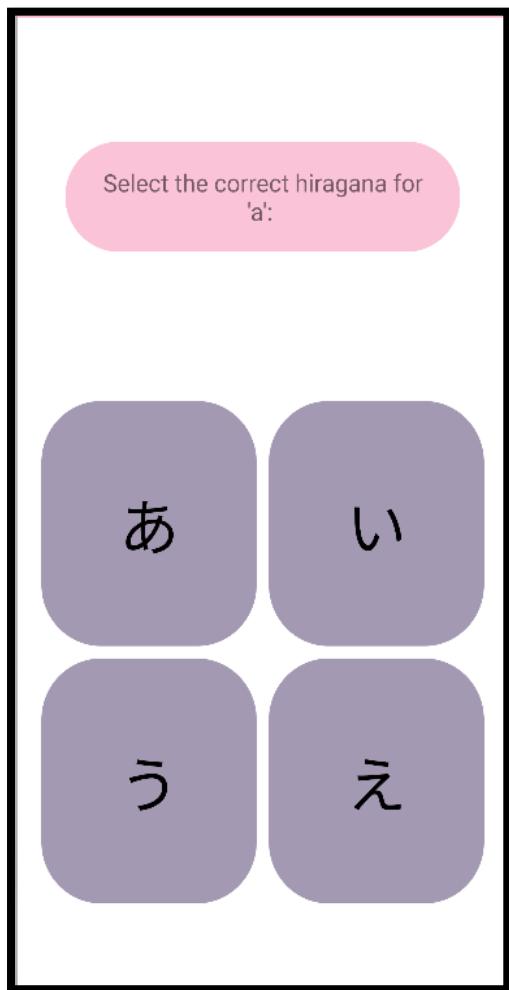
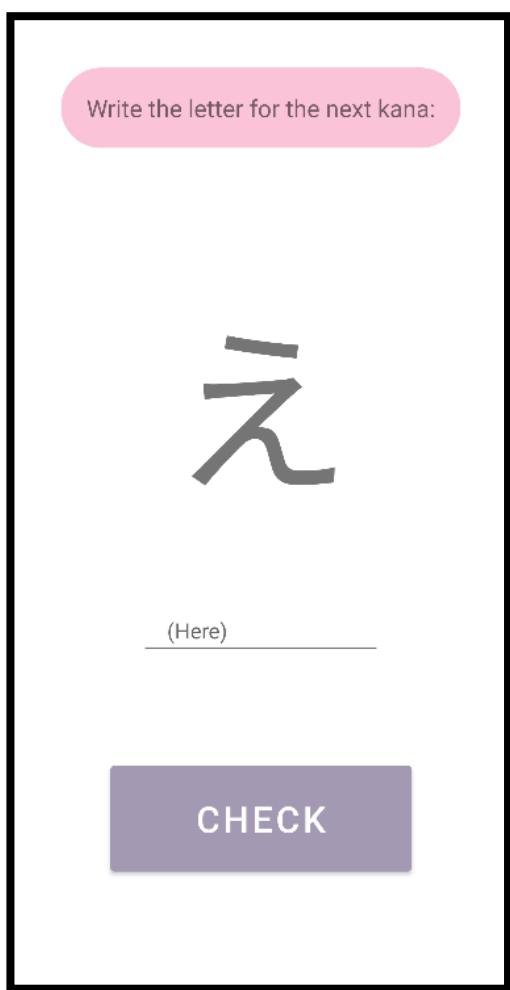
Como se puede apreciar, se utiliza la información recibida por medio de **Intent** para obtener el número de páginas de la lección en cuestión.

4.1.6. Pantalla de Prácticas: Practice Activities.

Tras finalizar las lecciones, la siguiente pantalla a acceder son las prácticas que necesita realizar el usuario para subir de nivel y hacer crecer a “Tanuki”.

Hay dos tipos de pantalla de práctica:

- Una de **carácter obligatoria**, el usuario necesita completar estas prácticas para poder subir de nivel.
- Una más enfocada a **un desafío**, el usuario decide si completarla o no. Esto no afecta al nivel del usuario.

*obligatoria**opcional*

Práctica 01 (obligatoria):

Esta práctica consta de 5 “TextView” una que actúa de enunciado y el resto actúan como posibles respuestas. Todo realizado para una tener una estética visual agradable y fácil de manejo para el usuario. El funcionamiento está explicado en el apartado de “[MAIN SCREEN](#)”.

Las funcionalidades de la práctica dependen mucho de la base de datos local creada. Ya que las preguntas y opciones vienen de ella, y se cargan en la misma pantalla.

La forma para poder mostrar los datos en pantalla es hacer una **consulta** sobre las tablas de “**Questions**” y “**Options**” e insertar sus datos en una lista de objeto que contiene todos los campos necesarios. Y es a partir de este objeto la que se manipula en la actividad de las prácticas.

The screenshot shows a mobile application interface. At the top, there is a header with several columns: level, questionES, questionEN, questionCH, idQuestion, complete, retries, and email. Below this is a table with two rows. The first row has a 'level' value of 1, a 'questionES' value of 'Selecciona el hiragan', a 'questionEN' value of 'Select the correct hiragan', a 'questionCH' value of '选择正确的平假名来', an 'idQuestion' value of '0552a51e4-5eeb-43f0...', a 'complete' value of 0, and an 'email' value of 'prueba@gmail.com'. The second row is a header for a table with columns 'option', 'rights', and 'idQuestion'. It contains two rows of data: one with 'option' 'あ' and 'rights' 1, and another with 'option' 'い' and 'rights' 0.

La metodología utilizada para la consulta en tabla “Questions” es obtener todas las preguntas que **coincidan** con el **nivel** de la práctica y el **correo** del usuario:

```
public List<Question> getQuestionsByLevelAndEmail(int level, String email) {
    List<Question> questions = new ArrayList<>();

    SQLiteDatabase db = this.getReadableDatabase();
    String query = "SELECT * FROM " + TABLE_NAME + " WHERE " + COLUMN_LEVEL + "=? AND " + COLUMN_EMAIL + "=?";
    Cursor cursor = db.rawQuery(query, new String[]{String.valueOf(level), email});
```

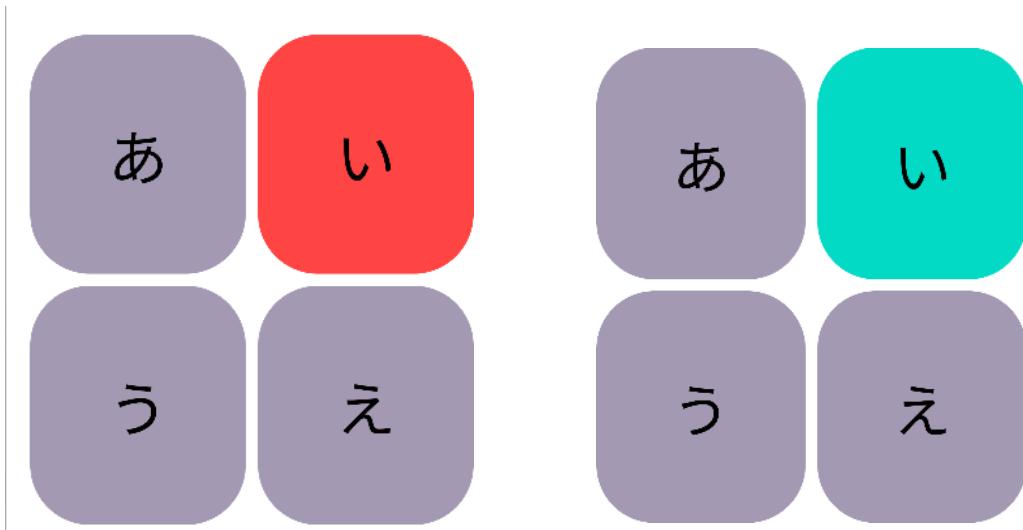
Una vez obtenida esta lista de objetos “Q”, se procede con la siguiente consulta, sobre la tabla de “Options” para recoger la lista de objetos “O”, que tengan el **mismo “idQuestion”** de todas las preguntas recogidas de la lista anterior “Q” y se añadirán a un objeto “T” que contenga las dos listas asociadas:

```
public List<Option> getOptionsByQuestionId(String idQuestion) {
    List<Option> options = new ArrayList<>();
    SQLiteDatabase db = this.getReadableDatabase();

    String query = "SELECT * FROM " + TABLE_NAME + " WHERE " + COLUMN_ID_QUESTION + "=?";
    Cursor cursor = db.rawQuery(query, new String[]{String.valueOf(idQuestion)});

    if (cursor != null && cursor.moveToFirst()) {
```

Tras obtener el objeto “T”, se puede controlar las selecciones del usuario e indicar si la opción seleccionada es correcta o no. Esto lo indica el campo “rights” de la tabla “Options” donde 0 es incorrecto (el fondo se pondrá en rojo) y 1 correcto (el fondo se pondrá en verde).



Tras acertar la opción se mostrará una animación y se recreará la actividad hasta que no haya más preguntas del nivel correspondiente en la lista del objeto “T”. Y el idioma mostrado de la pregunta es acorde al del seleccionado por el usuario guardado en el fichero de “SharePreference”.

```
if(getPosition() < listQuest.size()) {  
  
    if (!listQuest.get(getPosition()).isComplete()) {  
        SharedPreferences pref = getSharedPreferences( name: "Config", MODE_PRIVATE);  
  
        switch (pref.getString( key: "language", defaultValue: null)) {  
            case "es":  
                question.setText(listQuest.get(getPosition()).getQuestionES());  
                break;  
            case "zh":  
                question.setText(listQuest.get(getPosition()).getQuestionCH());  
                break;  
            default:  
                question.setText(listQuest.get(getPosition()).getQuestionEN());  
                break;  
        }  
  
        loadOptions(listQuest.get(getPosition()).getIdQuestion());  
    }  
}
```

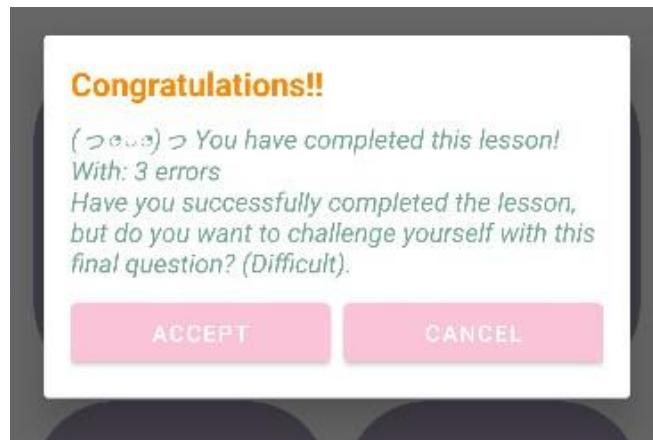
Se tiene en cuenta que al recrear la actividad se perdería todos los datos temporales de esa actividad por tanto para llevar el avance de las preguntas en la lista del objeto “T” se recurre a utilizar el “SharePreference” creando un valor contador que lleva el avance de las prácticas para su recreación en la misma pantalla. Esta se reinicia a 0 cada vez que comienza una práctica.

```
SharedPreferences pref = getSharedPreferences( name: "PRACTICE", MODE_PRIVATE);
SharedPreferences.Editor editor = pref.edit();

editor.putInt("cnt", pref.getInt( key: "cnt", defValue: 0) + 1);
editor.apply();
```

Y para agilizar toda la gestión llevada a cabo en la actividad de las prácticas las consultas a la base de datos y funciones complejas se realizan en hilo secundarios que no impiden el funcionamiento principal.

Al finalizar las pruebas se realiza los procesos de actualización del nivel de usuario y se muestra el mensaje al usuario para poder acceder a la práctica 2 de desafío. El mensaje mostrado es el siguiente:



Al aceptar se dirigirá a la siguiente práctica (no afecta al desbloqueo de niveles) y en cancelar vuelve al login y desbloquea el siguiente nivel.

Práctica 02 (desafío):

Esta práctica está compuesta de:

- Dos “TextView” que muestran la pregunta a responder.
- Un “editText” para introducir la respuesta.
- Un “Button” que permite su comprobación.

En comparación a la práctica 1, es **mucho más simple**. Ya que solo se carga una pregunta obtenida en un objeto creado en la misma actividad con una serie de 5 preguntas distintas (de ellas solamente una en pantalla), con una única respuesta sin recurrir a la base de datos. La pregunta que carga depende del nivel de la lección que accede el usuario. Los valores del objeto serán similares al siguiente:

```
Question question = new Question();
question.setQuestionEN("¿Cuál es la capital de Francia?");
question.setIdQuestion("1"+i);
question.setLevel(1);
question.setComplete(false);
quests.add(question);
```

Al accionar el botón se comprobará si la respuesta introducida por el usuario es correcta.

En caso correcto se mostrará una animación de haber completado la práctica con éxito y retrocede a la página principal. En caso contrario mostrará un mensaje.

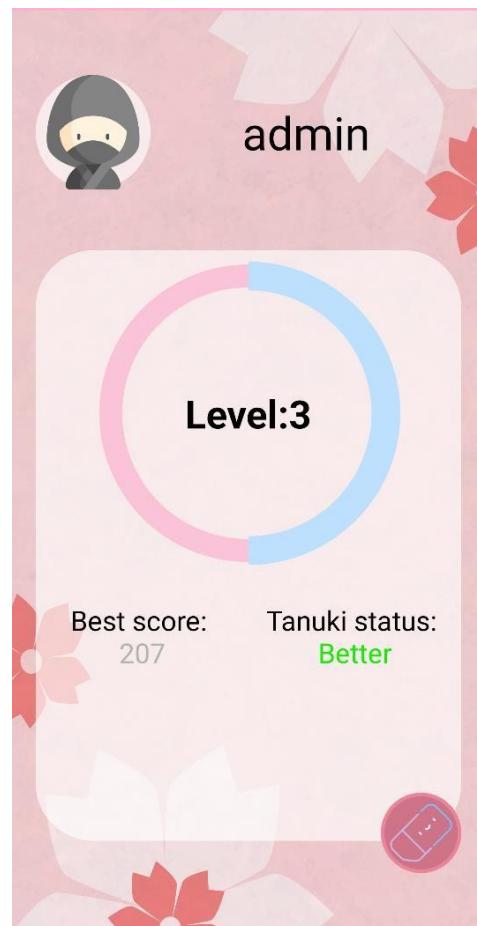
4.1.7. Pantalla de Usuarios: User Activity.

La pantalla del usuario muestra principalmente los siguientes elementos:

- Un “progressBar” que muestra el nivel de usuario, este destaca sobre los otros elementos de la pantalla.
- Un **ícono** cambiante por unos ofrecidos por la aplicación y se **permanecen guardados**.

- Un botón que permite la eliminación de la cuenta creada por el usuario.
- Y varios **textos** que muestra información relevante para el usuario: su nombre en formato grande, puntuación y el estado de “Tanuki” en **colores que cambian** según el progreso del usuario.

La pantalla se muestra de la siguiente forma:



Las funcionalidades añadidas en esta actividad son las siguientes:

- Icono personalizable.
- Progreso del nivel.
- Botón de borrar cuenta.

Icono personalizable:

Cambio de ícono, se le permite al usuario cambiar el ícono por unos que proporciona la aplicación por defecto y guardarlo para su próxima sesión.

Para lograrlo se **recurre a la base de datos** del Firebase sobre el campo “icon” donde se guarda el identificador que se asocia a las imágenes ofrecidas. Las opciones de íconos ofrecidas se muestran mediante un “**popupMenu**” las opciones se muestran en letras:



Para mostrar este menú se ha usado el siguiente código:

```
PopupMenu popupMenu = new PopupMenu(context: this, Icon, Gravity.END, popupStyleAttr: 0, R.style.PopupMenuStyle);
popupMenu.inflate(R.menu.icon_menu);

popupMenu.setOnMenuItemClickListener(item -> {
    switch (item.getItemId()) {
        case R.id.girl:
            Icon.setImageResource(R.drawable.user_icon1);
            saveIcon(0);
            break;
        case R.id.boy:
            Icon.setImageResource(R.drawable.user_icon2);
            saveIcon(1);
            break;
        case R.id.ninja:
            Icon.setImageResource(R.drawable.user_icon3);
            saveIcon(2);
            break;
        case R.id.grandpa:
            Icon.setImageResource(R.drawable.user_icon4);
            saveIcon(3);
            break;
    }
});
```

Tras seleccionar un ícono, se **actualiza** su valor por el id del elemento en la base de datos de Firebase. Se logra mediante una **consulta** empleando el “**Query**” sobre el **correo** del usuario.

Si la consulta ha sido completada con éxito se actualiza el campo “icon” de la tabla, esto se consigue usando el “**QueryDocumentSnapshot**” con la función de “**update()**”:

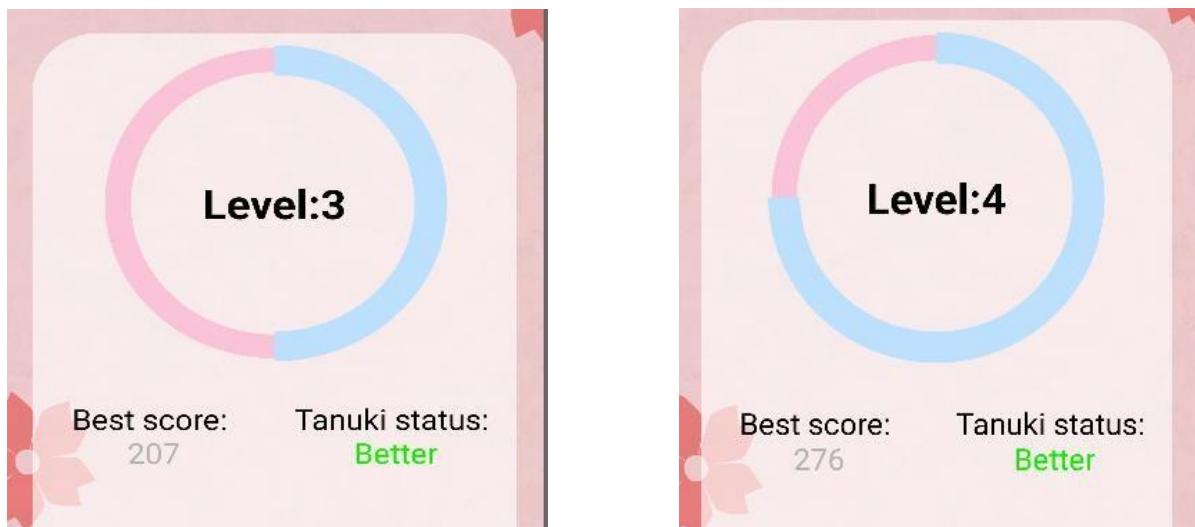
```
Query query = usersRef.whereEqualTo(field: "email", getUserEmail());
query.get().addOnCompleteListener(task -> {
    if (task.isSuccessful()) {
        for (QueryDocumentSnapshot document : task.getResult()) {
            String documentId = document.getId();
            usersRef.document(documentId).update(field: "icon", icon)
                .addOnSuccessListener(aVoid -> {
                    Toast.makeText(context, text: "Se ha guardado el ícono", Toast.LENGTH_SHORT).show();
                    recreate();
                })
                .addOnFailureListener(e -> {
                    Toast.makeText(context, text: "No se ha podido guardar el ícono", Toast.LENGTH_SHORT).show();
                });
        }
    } else {
        Toast.makeText(context, text: "No se ha podido guardar el ícono", Toast.LENGTH_SHORT).show();
    }
});
```

Y de manera similar se obtiene el nombre del usuario.

Progreso de nivel:

El nivel del usuario se muestra por texto en grande sincronizado con el “CircularProgressBar” ofrecido por el repositorio de “mikhaellopez”.

El nivel es obtenido de la base de datos del “Firebase” mediante una consulta con “Query” y se actualizará dependiendo de las lecciones completadas por el usuario. Lo mismo ocurre con las informaciones del estado de “Tanuki” y la puntuación máxima.



El código usado para lograrlo es el siguiente:

Mediante el uso de “Query” se obtiene de la base de datos de Firebase los datos necesarios para mostrar en la pantalla y para la **sincronización del “progressBar”** con el nivel se usa el método “**setProgress()**”.

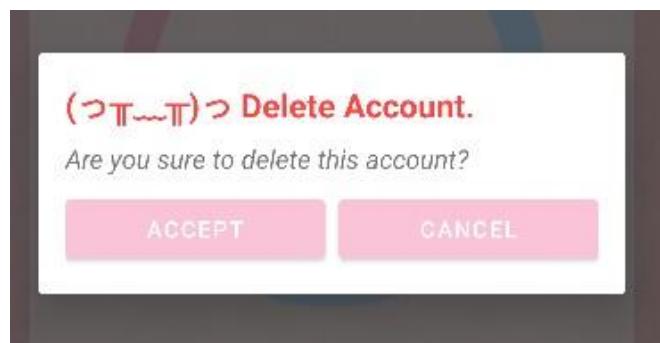
```
public void loadUserData() {  
    Query query = usersRef.whereEqualTo( field: "email", email);  
  
    query.get().addOnCompleteListener(task -> {  
        if (task.isSuccessful()) {  
            QuerySnapshot querySnapshot = task.getResult();  
            if (querySnapshot != null && !querySnapshot.isEmpty()) {  
                DocumentSnapshot document = querySnapshot.getDocuments().get(0);  
  
                String userName = document.getString( field: "name");  
                String GoogleUserName = document.getString( field: "nombre");  
                int icon = document.getLong( field: "icon").intValue();  
                int level = document.getLong( field: "level").intValue();  
  
                showUserData(userName, GoogleUserName, icon, level);  
            }  
        }  
    });  
}
```

Borrado de cuenta:



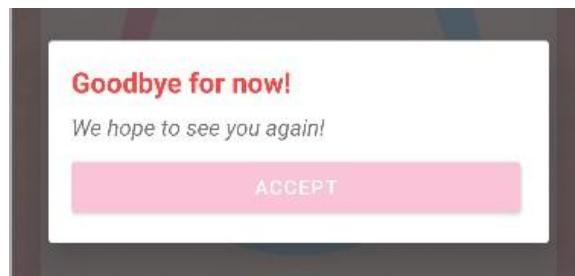
La aplicación ofrece la posibilidad de eliminar la cuenta creada, pulsando en el botón de borrado que se muestra con forma circular en la pantalla.

Al accionar en el botón se le presentará al usuario un mensaje de confirmación:

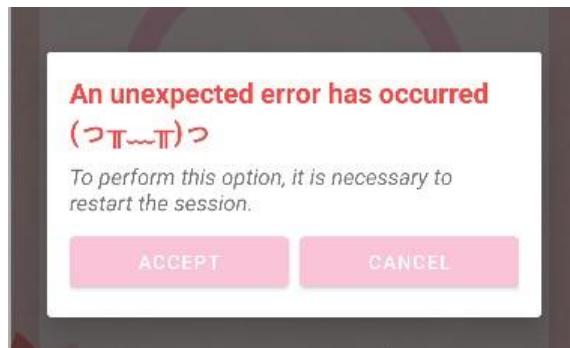


Al confirmar se le mostrará al usuario un mensaje indicando si el proceso solicitado se ha realizado con éxito o no. Los mensajes para mostrar son:

- Borrado con éxito, y al aceptar sale de la aplicación.



- Borrado sin éxito, se le pedirá al usuario volver a iniciar sesión para poder completar con el borrado. (Es principalmente usado para los usuarios con sesión con cuenta Google).



El código usado para realizar la eliminación de la cuenta se recurre a **los métodos** ofrecidos por Firebase con la función “**delete()**” para poder llamar a esta función se necesita una **autentificación** de la sesión iniciada en la app obtenida **por “FirebaseAuth”**. Esto es para el caso de un usuario iniciado sesión con correo y contraseña.

```
FirebaseAuth.getInstance().signInWithEmailAndPassword(getUserEmail(), getUserPassword())
    .addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
            if (user != null) {
                user.delete()
                    .addOnCompleteListener(task1 -> {
                        if (task1.isSuccessful()) {
                            DeleteUserFromDataBase(getUserEmail());
                        } else {
                            ErrorMessage( type: 0);
                        }
                    });
            }
        }
    });
}
```

Para el caso de borrar una cuenta iniciada sesión con cuenta Google, es necesario establecer una instancia similar a la establecida al iniciar el login. Tras ello, el proceso es igual al anterior.

```
private static void DeleteGoogleUser() {
    GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken("289735160416-nbt9sig9pai0fgofmq503gfnirvljqn1.apps.googleusercontent.com")
        .requestEmail()
        .build();
    GoogleSignInClient googleSignInClient = GoogleSignIn.getClient(context, gso);

    googleSignInClient.signOut()
        .addOnCompleteListener((Activity) context, task -> {
            FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        });
}
```

También se contempla **el borrado de todos los datos relacionados con este usuario** en las bases de datos. Para borrar en Firebase se usa funciones que proporciona como el “Query” y “delete()” contados en los anteriores puntos y el borrado en la base de datos local se usa las **funciones declaradas en las clases “Helper”** correspondientes a cada tabla.

La implementación en código es el siguiente:

```
private static void DeleteLocalData() {  
    //Elimino todas las preguntas asociadas a esta cuenta. (en local).  
    userSQLiteHelper.deleteByEmail(getUserEmail());  
  
    questionSQLiteHelper.deleteByEmail(getUserEmail());  
  
    optionsSQLiteHelper.deleteOptionsByIdQuestions(  
        questionSQLiteHelper.getIdQuestionsByEmail(getUserEmail()));  
}
```

Un ejemplo de código de la clase “Helper” de la tabla “Question” para eliminar columnas asociadas al correo del usuario:

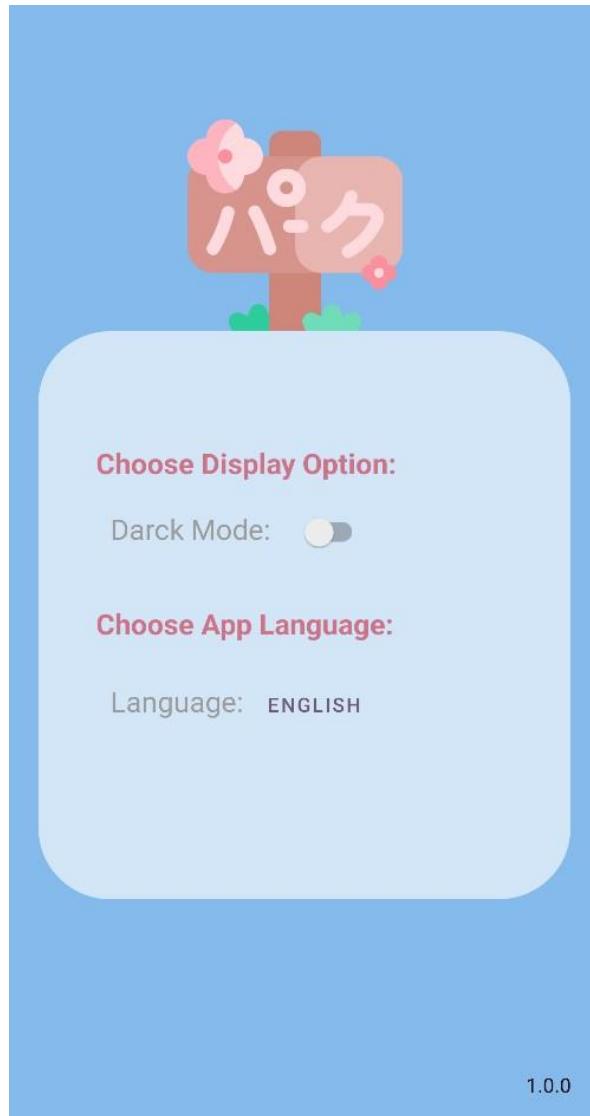
```
public void deleteByEmail(String email) {  
    SQLiteDatabase db = getWritableDatabase();  
  
    db.delete(TABLE_NAME, whereClause: COLUMN_EMAIL + "=?", new String[]{email});  
    db.close();  
}
```

Se tendrá en cuenta borrar la sesión en el fichero “SharedPreference” tras haber borrado la cuenta del usuario.

4.1.8. Pantalla de Configuración: Settings Activity.

La pantalla de Configuración simplemente muestra las dos opciones que el usuario puede cambiar de la aplicación siendo:

- El lenguaje, la aplicación permite: **español, inglés y chino** (simplificado).
- El **modo nocturno**, establecido por un “Switch” con un estilo simple y elegante.



Cambio de Idioma:

En la selección de idiomas al seleccionar el botón se muestra un “**popupMenu**” con las opciones de idiomas habilitad

```
//OPCIÓN: SELECCIONAR IDIOMA.
language.setOnClickListener(v -> showLanguageMenu());

@SuppressWarnings("NonConstantResourceId")
public void showLanguageMenu(){
    PopupMenu popupMenu = new PopupMenu( context: this, language, Gravity.END, popupStyleAttr: 0, R.style.PopupMenuStyle);
    popupMenu.inflate(R.menu.language_menu);

    popupMenu.setOnMenuItemClickListener(item -> {
        switch (item.getItemId()){
            case R.id.action_english:
                setLanguage("en");
                break;
            case R.id.action_spanish:
                setLanguage("es");
                break;
            case R.id.action_chinese:
                setLanguage("zh");
                break;
        }
        return false;
    });

    popupMenu.show();
}
```

El diseño del “**popupMenu**” es simple y se muestra de la siguiente forma:



Para lograr el cambio del lenguaje seleccionado se recurre a una nueva clase llamada “**LanguageHelper**” que gestiona el establecimiento del lenguaje con la función “**setLocale()**” mediante el contexto de la actividad obtenida permite el cambio del idioma según el código del idioma seleccionado por ejemplo para el español es “es”:

```
public static void setLocale(Context context, String languageCode) {  
    Locale locale = new Locale(languageCode);  
    Locale.setDefault(locale);  
  
    Configuration config = new Configuration();  
    config.locale = locale;  
  
    context.getResources().updateConfiguration(config, context.getResources().getDisplayMetrics());
```

Una vez seleccionado el lenguaje, la opción será guardado en un archivo “SharePreference” esto permite a la aplicación **recoger esta configuración y guardarla**. Al iniciar se establece estas configuraciones guardadas, evitando así la configuración constante.

Se tiene en cuenta en los casos que no obtiene valor en el archivo “SharePreference” para cargar la configuración se cogerá el lenguaje por defecto del dispositivo móvil del usuario.

```
public static String getLanguage(Context context) {  
    SharedPreferences preferences = context.getSharedPreferences(name: "Config", Context.MODE_PRIVATE);  
    String language = preferences.getString(key: "language", defValue: null);  
  
    //Si no hay idioma guardado uso el del sistema  
    if (language==null) {  
        language = Locale.getDefault().getLanguage();  
    }  
  
    return language;  
}
```

Modo Nocturno:

La opción de activar el modo nocturno se utiliza un “Switch”.

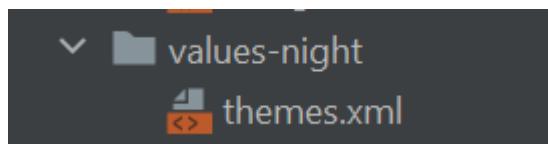
Al accionar el modo noche se recrea la pantalla de la actividad para mostrar los cambios efectuados y se guarda esta configuración en el archivo SharePreference con el mismo mecanismo que el de idioma.

```
//OPCIÓN: SELECCIONAR MODO NOCHE.
nightMode.setOnCheckedChangeListener(v -> {
    if (nightMode.isChecked()) {
        // Cambiar a modo nocturno
        AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
        recreate();
        setNightMode(true);

        saveConfig( lang: null, String.valueOf( b: true));
    } else {
        // Cambiar a modo normal
        AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
        recreate();
        setNightMode(false);

        saveConfig( lang: null, String.valueOf( b: false));
    }
});
```

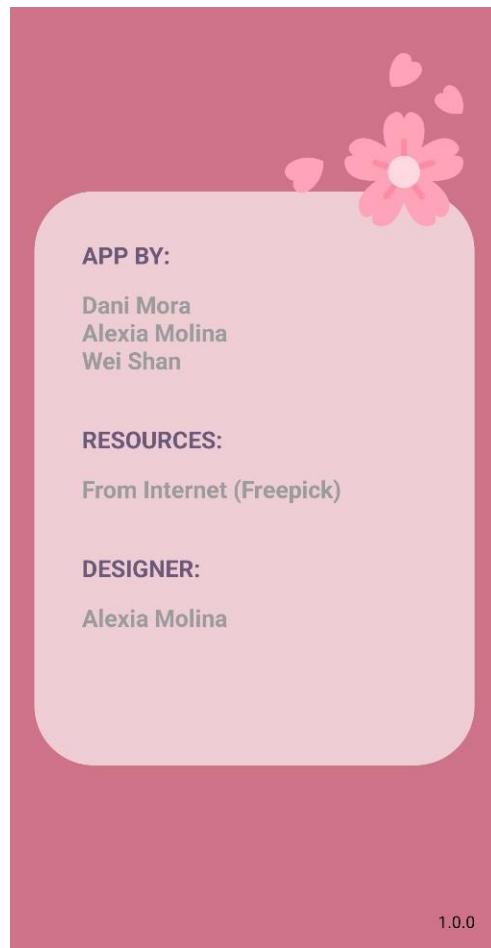
La función anterior recoge las configuraciones del sistema y establece las configuraciones de modo nocturno del sistema por defecto. Para que esto funcione se debe declarar un tema noche en el recurso del proyecto.



4.1.9. Pantalla de Información: About Activity.

La pantalla de información simplemente mostrará a los usuarios informaciones relevantes sobre los desarrolladores y la proveniencia de los recursos utilizados en la aplicación.

Toda la información se presentará con una determinada atractividad visual para mejorar la experiencia del usuario. (Sin información densa y pesada). Con una decoración simple y elegante.



4.1.10. Sobre la base de datos local y la carga de los campos de las tablas.

Para el proceso de inserción de los datos sobre todo en la base de datos local se ha optado por crear un archivo “json” con las estructuras requeridas coincidentes a las tablas creadas en SQLite.

Ya que esto permite **simplificar la cantidad de código necesaria** para la inserción de datos y una **mayor facilidad para la modificación** de ellas sin tener que tocar el código del proyecto.

Como el siguiente:

```
{ [ ]  
  "questions": [ [ ]  
    { [ ]  
      "email": null,  
      "level": 1,  
      "questionES": "Selecciona el hiragana correcto para 'a': ",  
      "questionEN": "Select the correct hiragana for 'a': ",  
      "questionCH": "选择正确的平假名来表示 'a': ",  
      "options": [ [ ]  
        { [ ]  
          "option": "\u00e1",  
          "right": true  
        },  
        { [ ]  
          "option": "\u00e3"  
        }  
      ]  
    }  
  ]  
}
```

Es necesario colocar el archivo “json” en la carpeta “**assets**” para poder ser cargado.

La función para poder cargar los datos de ese fichero usa la función “**AssetsManager**” para coger correctamente el archivo y su manejo.

Para la **lectura** del archivo se usa el método “**InputStream**” con la función “**read()**”. Con ello convertimos los bytes leídos en formato “String” (UTF-8).

```
private String loadMyJsonFromAsset(@NonNull AssetManager assetManager, String fileName) {  
    try {  
        InputStream inputStream = assetManager.open(fileName);  
        int size = inputStream.available();  
        byte[] buffer = new byte[size];  
        inputStream.read(buffer);  
        inputStream.close();  
        return new String(buffer, StandardCharsets.UTF_8);  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
    return null;  
}
```

Con este “String” obtenido se llama a una función de la clase creada “MyJsonParser” para pasar el “json” obtenido (en forma de String) a un objeto creado con anterioridad en el proyecto identificando cada elemento correspondiente.

```
public static LocalUser parseJsonToObjects(String jsonString) {  
    try {  
        JSONObject json0bject = new JSONObject(jsonString);  
        JSONArray jsonArray = json0bject.getJSONArray("questions");  
  
        List<Question> questions = new ArrayList<>();  
  
        for (int i = 0; i < jsonArray.length(); i++) {  
            JSONObject question0bj = jsonArray.getJSONObject(i);  
            int level = question0bj.getInt("level");  
            String questionES = question0bj.getString("questionES");  
            String questionEN = question0bj.getString("questionEN");  
            String questionCH = question0bj.getString("questionCH");  
            UUID uniqueId = UUID.randomUUID();  
            String idQuestion = i+String.valueOf(uniqueId);  
            boolean complete = question0bj.getBoolean("correct");  
            int retries = question0bj.getInt("retries");  
            String email = question0bj.getString("email");  
  
            JSONArray optionsArray = question0bj.getJSONArray("options");  
            List<Option> options = new ArrayList<>();  
    }  
}
```

Para la creación de las tablas se usan los métodos CREATE TABLE de SQL declarado en la clase “Helper” de cada tabla y se inicia su creación con el siguiente código. Se tiene en cuenta la verificación de la existencia de la tabla a crear.

4.2 Problemas encontrados y soluciones

Los errores principales que se han encontrado son:

- Problemas de **control de versiones**:

Ya que utilizamos GitHub como herramienta para el control de versiones y desarrollo organizado de la aplicación, surgen algunos problemas con la integración del IDE Android Studio al cambiar de dispositivo, clonar el proyecto...

- **git: fatal: detected dubious ownership (al intentar emularlo en Android Studio)**

- SOLUCION: git config --global --add safe.directory "directory".

(En este caso el propio IDE nos propone ejecutar este comando en la terminal propia para poder arreglar el error).

- Problemas en la **eliminación de una cuenta** de Google:

Para los usuarios iniciados con una cuenta Google, al **traspasar determinado tiempo** se les **caduca la sesión** iniciada **con Google** en la aplicación, pero como la aplicación tiene su “propia forma” de guardar sesión del usuario, al tener la sesión expirada de Google al querer borrar la cuenta, el servicio de Google no lo permite.

- Muestra siempre el mensaje de error al usuario al accionar en el botón de borrar cuenta.
- SOLUCION: tras surgir este problema de borrado de cuenta en vez de cerrar el dialogo que **obligue al usuario iniciar sesión de nuevo**. Para poder efectuar esta acción con éxito.

- Problemas en la **carga de datos** del usuario desde Firebase al **iniciar la sesión por primera vez**:

Al iniciar sesión para un nuevo usuario se guarda sus datos en el servidor de Firebase y enseguida accede a la pantalla principal donde se caga los datos del mismo usuario.

Al ser Firebase un **servicio cloud no siempre se leen y se escriben de forma instantánea**, se puede producir retrasos sobre todo en el caso anterior.

- Al entrar en la pantalla principal no se muestra el nombre de usuario, ni los botones correspondientes a las lecciones.
- SOLUCION: mostrar un dialogo de proceso durante la consulta a la base de datos del Firebase si aun así falla se fuerza un reinicio de la aplicación. Este reinicio como tiene la sesión del usuario guardado entra directamente a la pantalla principal y con este retardo se obtiene los datos correctamente de Firebase.

5 VALIDACIÓN DE LA SOLUCIÓN

5.1 Documentación descriptiva

Las principales pruebas realizadas para tener mayor control de los errores son:

PRUEBA 1:

En el registro y login se controla que no haya campos en vacío, contraseña inferior a 6 dígitos, correo ya asociado a una cuenta. Los mensajes que muestra son:

(いらっしゃいませ)

● Nombre Usuario
✉ Correo
🔒 Contraseña
🔒 Repita la contraseña
 Hay campos vacíos

(いらっしゃいませ)

● Nombre Usuario a
✉ Correo prueba@gmail.com
🔒 Contraseña
🔒 Repita la contraseña
 Las contraseñas introducidas no coinciden

(いらっしゃいませ)

● Nombre Usuario a
✉ Correo prueba@gmail.com
🔒 Contraseña .
🔒 Repita la contraseña .
 La contraseña debe no puede ser inferior a 6 dígitos

(いらっしゃいませ)

● Nombre Usuario a
✉ Correo prueba@gmail.com
🔒 Contraseña
🔒 Repita la contraseña
 El correo ya tiene una cuenta asociada

PRUEBA 2:

Para enviar el correo de recuperación de contraseña se verificará que no haya campos vacíos.



5.2 Problemas encontrados y justificación

Los principales problemas que se han encontrado y no se han podido solucionar del todo son:

- Problema excepcional de **conflictos de idioma** en algunos dispositivos:

Tras establecer un idioma seleccionado en la pantalla de configuración puede ocurrir que en algunos dispositivos no logren efectuar el cambio y aparezca algunas pantallas en el idioma seleccionado y otras por defecto del dispositivo.

Para solucionarlo se recurre a **mostrar un mensaje de alerta** que avisa al usuario si desea **reiniciar la aplicación** para establecer mejor las configuraciones seleccionadas.

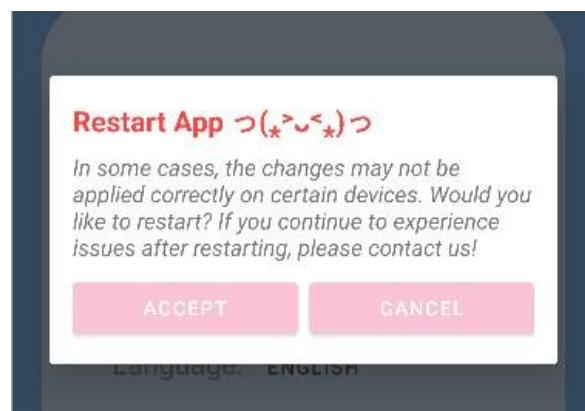
Este reinicio no es de carácter obligatorio, el usuario **puede cancelarlo**.

```
AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
builder.setView(dialogView);
AlertDialog alertDialog = builder.create();
alertDialog.setCancelable(false);
alertDialog.show();

switch(title){
    case "Restart App ↪(*>_<*)↪":
        dismiss.setVisibility(dialogView.VISIBLE);
        alertDialog.setCancelable(true);
        cancel.setOnClickListener(v -> {
            alertDialog.dismiss();
            restartApp();
        });

        dismiss.setOnClickListener(v -> {
            alertDialog.dismiss();
            Intent intent=new Intent( packageContext: NiHonActivity.this,MainActivity.class);
            startActivity(intent);
            finish();
        });
        break;
}
```

El mensaje mostrado es el siguiente:



Para poder reiniciar la aplicación se usa la siguiente función:

```
public void restartApp() {  
    Intent intent = getBaseContext().getPackageManager().getLaunchIntentForPackage(getApplicationContext()  
        [.getPackageName());  
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);  
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);  
    startActivity(intent);  
    finish();  
  
    // Finalizar el proceso actual  
    android.os.Process.killProcess(android.os.Process.myPid());  
    System.exit( status: 0);  
}
```

Más se cancela también la recreación de la actividad al accionar el efecto de cambio de sentido en pantalla, que pueda crear conflictividad entre lenguajes al recrear:

```
android:configChanges="keyboardHidden|orientation|screenSize"
```

La mayoría de las veces con el reinicio se soluciona el problema de conflictividad entre lenguajes.

Pero también hay casos excepcionales en que no tiene efecto y no se ha podido solucionar.

6 FUENTES

- Flaticon
- Coolors.co
- Stack Overflow
- Android Developers
- Firebase Docs
- ChatGPT
- GitHub
- Krita (para diseño y arreglo de iconos, imágenes y fondos)
- Lottie

7 ANEXO

Proyecto: [Aquí.](#)

Proyecto en Firebase: [Aquí.](#)