MemoryApp

Daniel Cuervo – DAM 2

Presentación del Proyecto

MemoryApp una App de Estudio que ayude a los usuarios a aprender, repasar y memorizar conceptos a través de listas de aprendizaje. Estas Listas se repasarán como *flashcards* usando un algoritmo de repetición y evaluación. Cada concepto tendrá un puntaje individual, y cada lista contará con un puntaje global que finalizado permite al usuario avanzar al siguiente nivel de los 3 totales (Fundamentos – Consolidación – Dominio). De esta manera la aplicación implementa y potencia uno de los principios clave de aprendizaje: la REPETICIÓN ESPACIADA.

Funcionamiento de las listas:

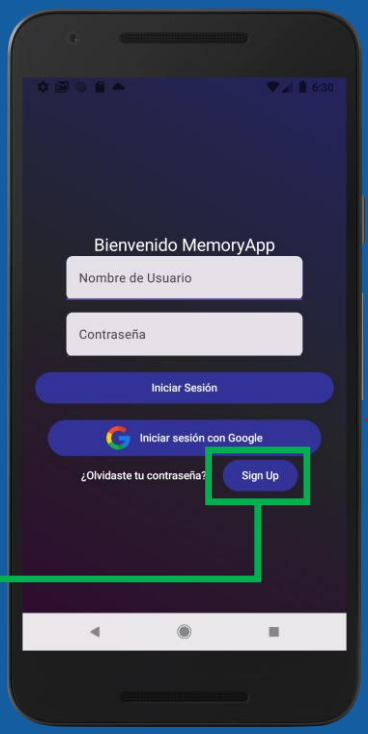
Las Listas de vocabulario/conceptos pueden ser creadas por cada usuario un la IA agregando varias Tarjetas a la Lista, cada tarjeta tiene dos caras (pregunta y respuesta); las listas se guardan y organizan dentro de un Tema el cual contiene diferentes listas.

Como se puede ver el principio general de repaso y aprendizaje son las *flashcards* pero existen tres tipos de listas:

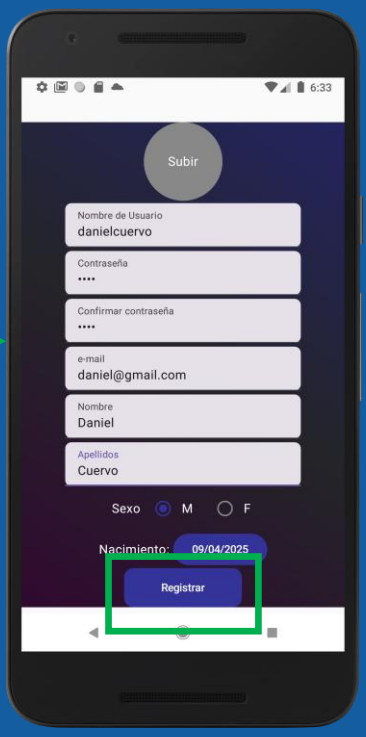
El usuario pueden ingresar registrándose o a través de la verificación de Google, también aporta sus datos personales y de registro, el sistema la asigna una puntuación base de 0 que irá aumentando a media que termine las listas de estudio.

Esta App se diseña en un principio para móviles Android, pero el objetivo final es que se pueda utilizar también en dispositivos IOS y Web; y formará parte del entorno de aprendizaje de www.imagine-iacademy.danielcuervox.com, mi página interactiva de enseñanza de idiomas, completando así con app exclusiva

Ventana Login y Registro



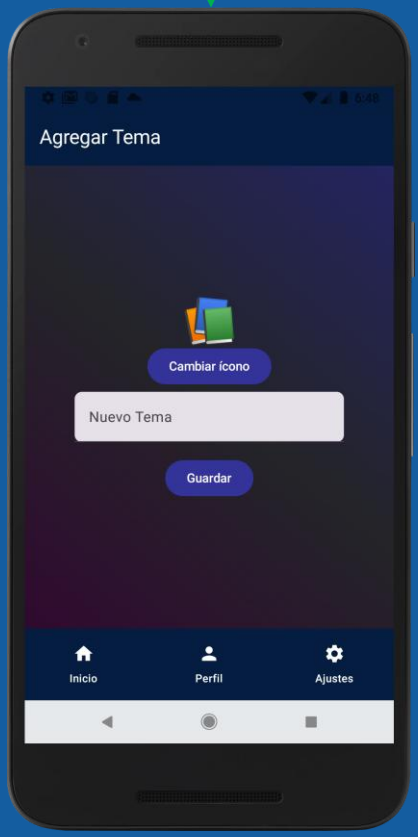
Esta es la primera ventana que ve el usuario cuando accede a MemoryApp, donde debe identificarse, puede ingresar con su cuenta de Google o creando un perfil para posteriormente ingresar con su nombre de usuario y contraseña.



Cuando se registra de manera manual debe ingresar todos los datos que se ven en la pantalla, y debe ser mayor de 6 años para que le valide la fecha, una vez estén todos los campos llenos se activa el botón de ‘Registrarse’

Ventana Main

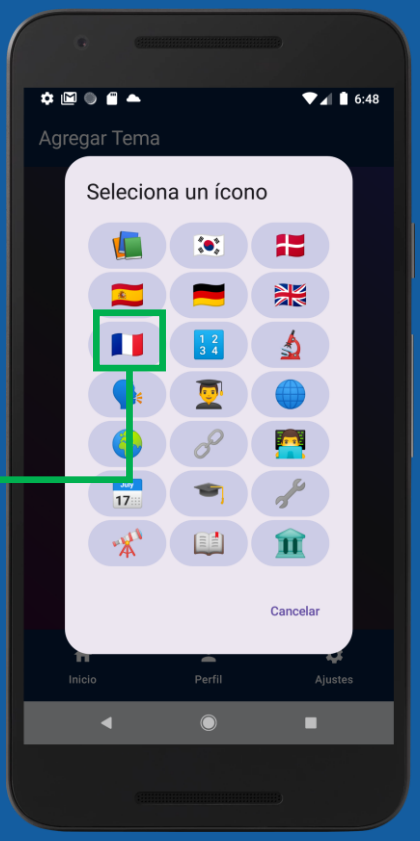
Una vez se ha identificado puede acceder a la ventana Main donde debe crear los Temas que van a contener las listas de Tarjetas

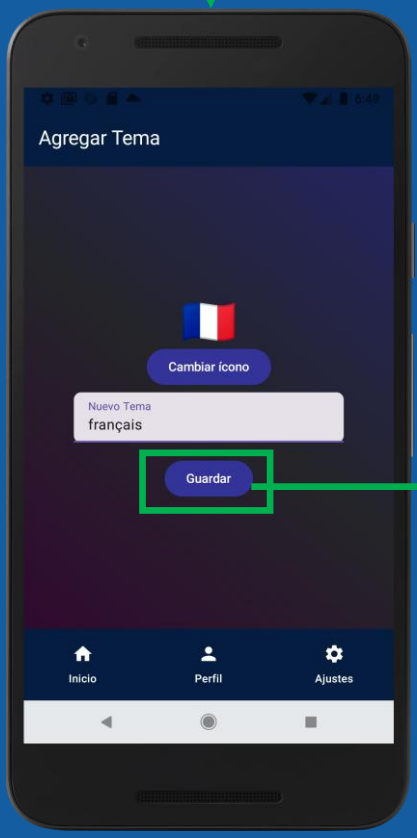


Para crear un nuevo tema sólo debe escribir el nombre y seleccionar un ícono.

Seleccionar un ícono

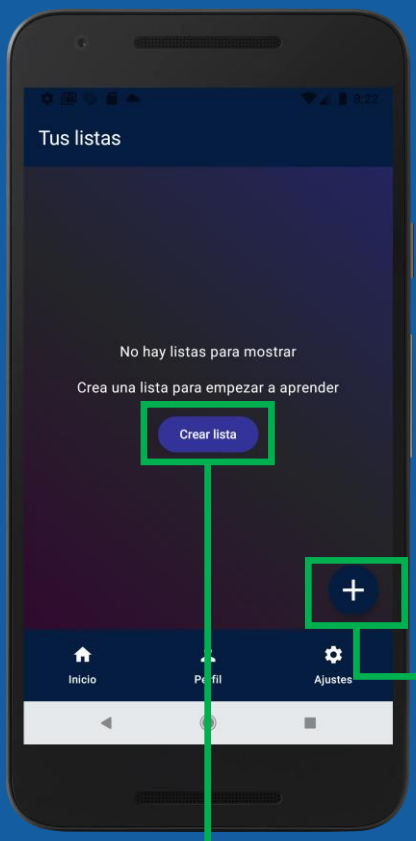
Este botón despliega una AlertDialog donde se ven diferentes íconos que contiene la App, donde puede seleccionar uno para costomizar el Tema



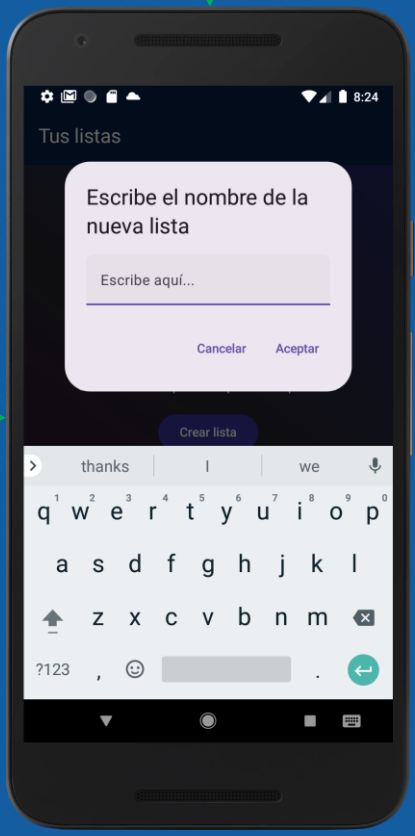


El tema crado aparecerá inmediatamente en la ventana Main

Ventana Temas



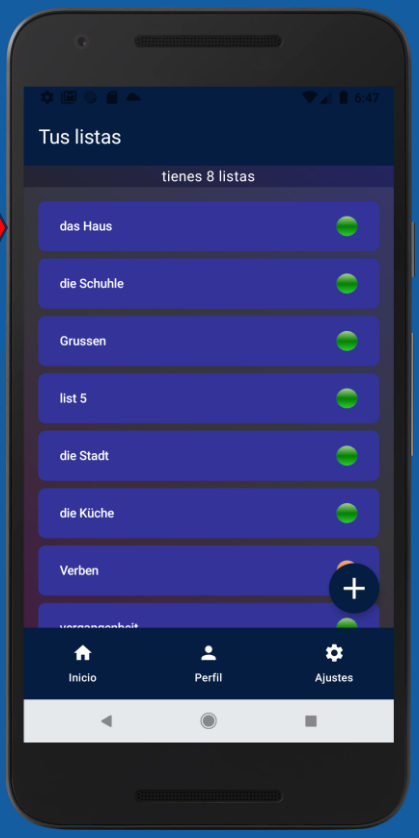
Inicialmente el usuario no tiene ningún tema creado, para crearlos lo puede hacer desde el botón que se muestra cuando hay cero temas o desde FloatActionButton



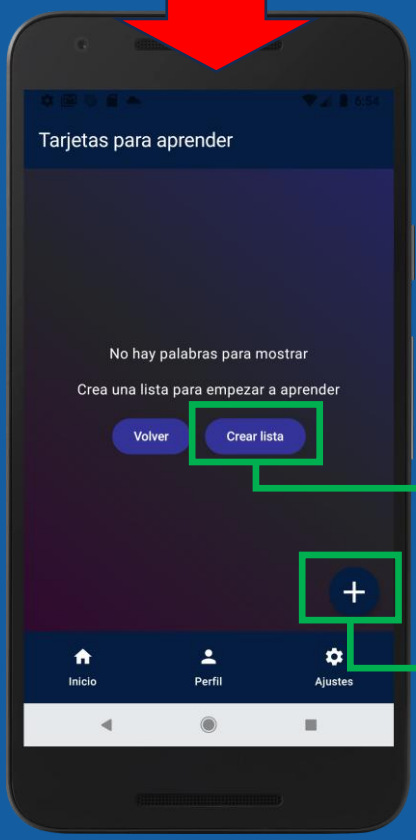
Para crear una Lista de Tarjetas de un Tema en particular se debe colocar el nombre.

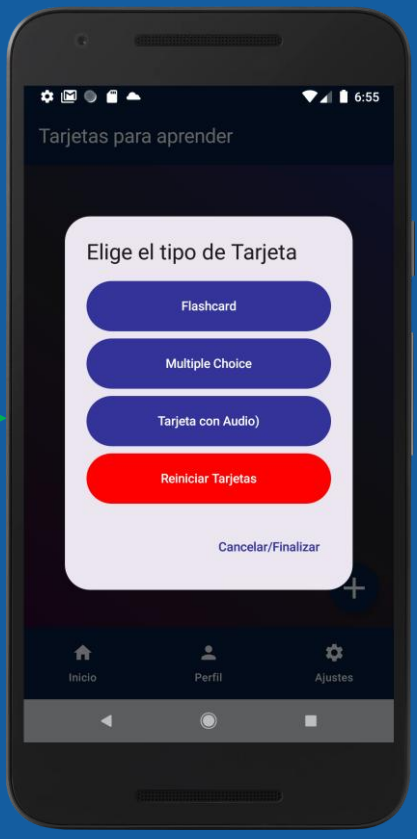
Crear las Flashcards de memorización

Se pueden crear tantas listas como se deseen, ejemplo: se crea tema Japonés y dentro de japonés están las litas: saludos, palabras de cortesía, verbos, despedidas, etc. Y cada lista tenía innumerables tarjetas a memorizar.

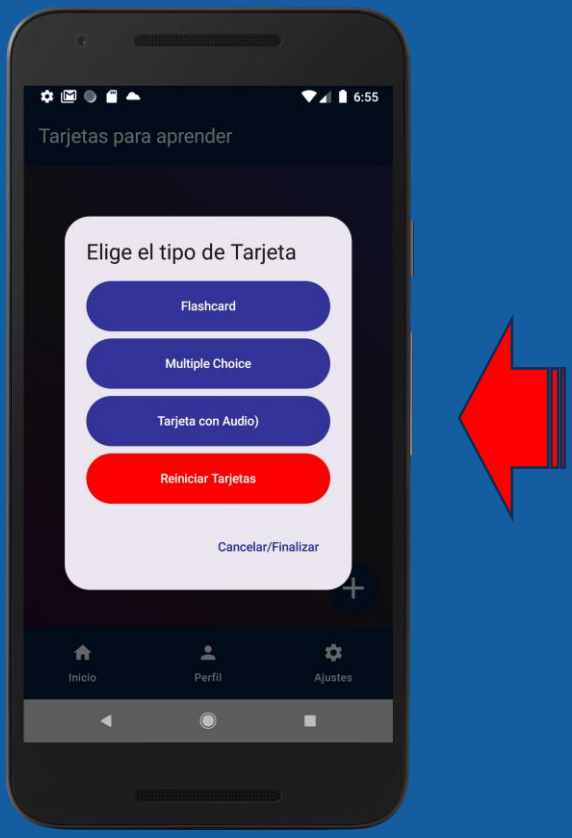


Cuando se crea una lista, por defecto está vacía, y se deben agregar tarjetas para memorizar, desde cualquiera de estos dos botones.





Crear las Flashcards de memorización



TIPOS DE TARJETA

Dentro del AlertDialog se pueden ver las diferentes opciones para las listas

1. Crear Flashcards normales, donde muestra la pregunta y la respuesta.

2. Multiple Choise: estas nos permite escribir la pregunta con la respuesta y 3 respuestas incorrectas, para la segunda fase se quiere que éstas las genere una IA.

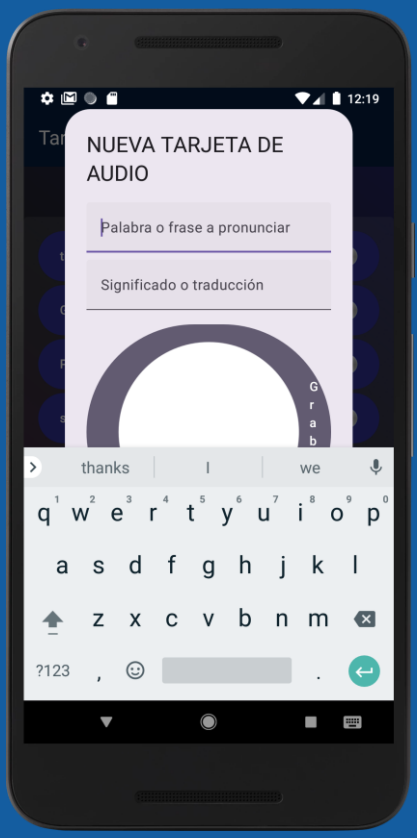
3. Tarjeta con Audio: se puede grabar un audio, muy útil para la pronunciación de idiomas.

4. ´También se puede borrar la puntuación de las tarjetas si se quiere volver a relializar la actividad de memorizarlas.

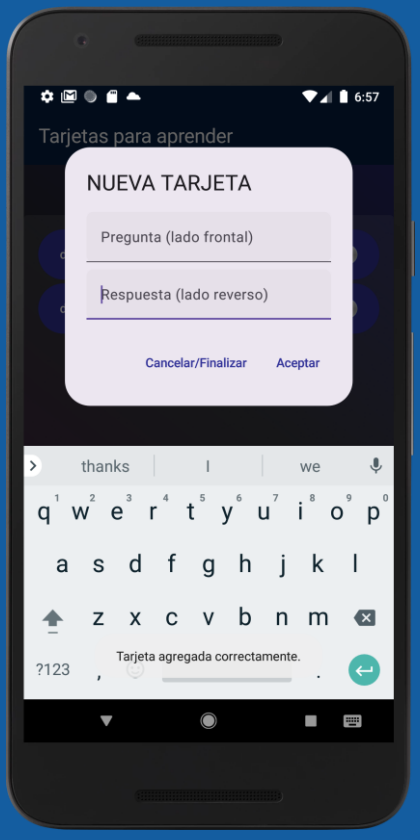
Crear las Flashcards de memorización

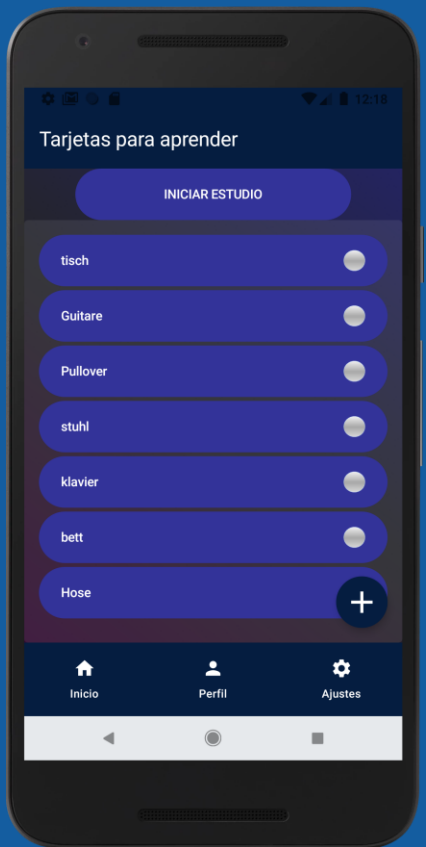
Para cada tarjeta se abre un AlertDialog donde se escribe la palabra y la pregunta o la palabra en otro idioma (pregunta) y español (repuesta). Y limpia los campos porque dentro de cada Lista se almacenan varias Tarjetas.

Crear Flashcards de memorización



También se puede agregar una tarjeta con audio, para ello el AlertDialog muestra un botón que se mantiene presionado para guardar el audio.

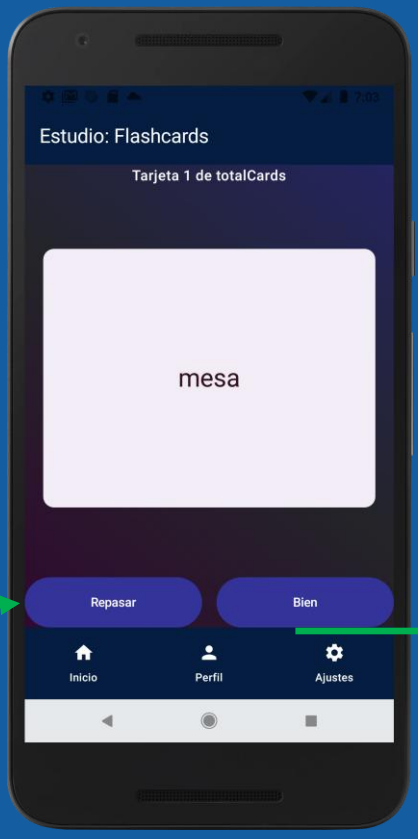




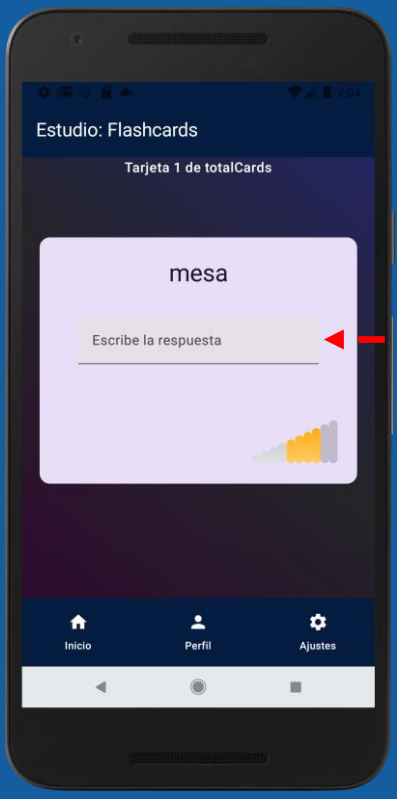
Cuando hace click en ‘Cancelar/Finalizar’ sale del AlertDialog y muestra las preguntas de todas las Tarjetas que hemos creado. Nótese que tiene un círculo gris, eso significa que tienen 0 puntos porque se acaban de crear; naranja: a partir de 4 puntos y verde a partir de 9 puntos, evidentemente estos puntos se logran realizando la actividad de memorización

Actividad de Memorización

1ra Actividad: Dentro de la ventana de iniciar actividad se muestra una palabra de la lista, esto se hace de manera aleatoria, el Usuario debe acordarse qué significaba, en 'ver Respuesta' gira la tarjeta y muestra la respuesta si el Usuario ha acertado debe dar en bien, en este caso el sistema suma un punto, si da en 'Repasar' el sistema le quita un punto. Esto se repite consecutivamente hasta que todas las tarjetas tenga puntuación de 4 cuando aparecerá, y ahí aparecerá un mesaje de finalización de la actividad.



Diseño de persistencia de la información

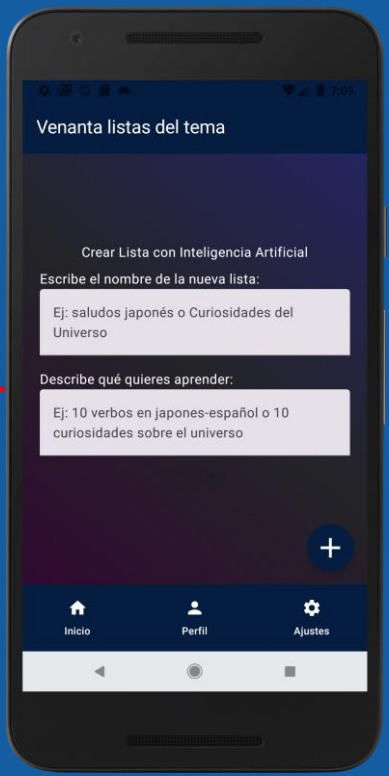


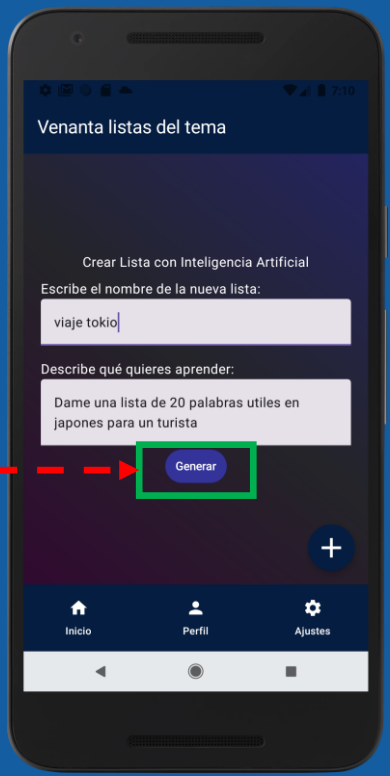
<u>3ra Actividad (enfocada al writing de idiomas)</u> : en esta actividad el Usuario debe escribir la repuesta, en donde comparará el campo de texto donde escribe el usuario y la respuesta almacenada. Si Coinciden sumará un punto, en cuyo caso suma un punto, la Actividad de Memorización termina cuando cada una de las tarjetas llega a 10 puntos.

Como se puede ver en la imagen, va mostrando en el gráfico el puntaje actual de cada tarjeta. Cuando termine todas las tarjetas sale un mensaje de felicitación

Crear una Lista con IA

Otra forma de crear listas es también a través de inteligencia artificial, donde le escribe el nombre de la Lista y un prompt donde describa la lista a crear.



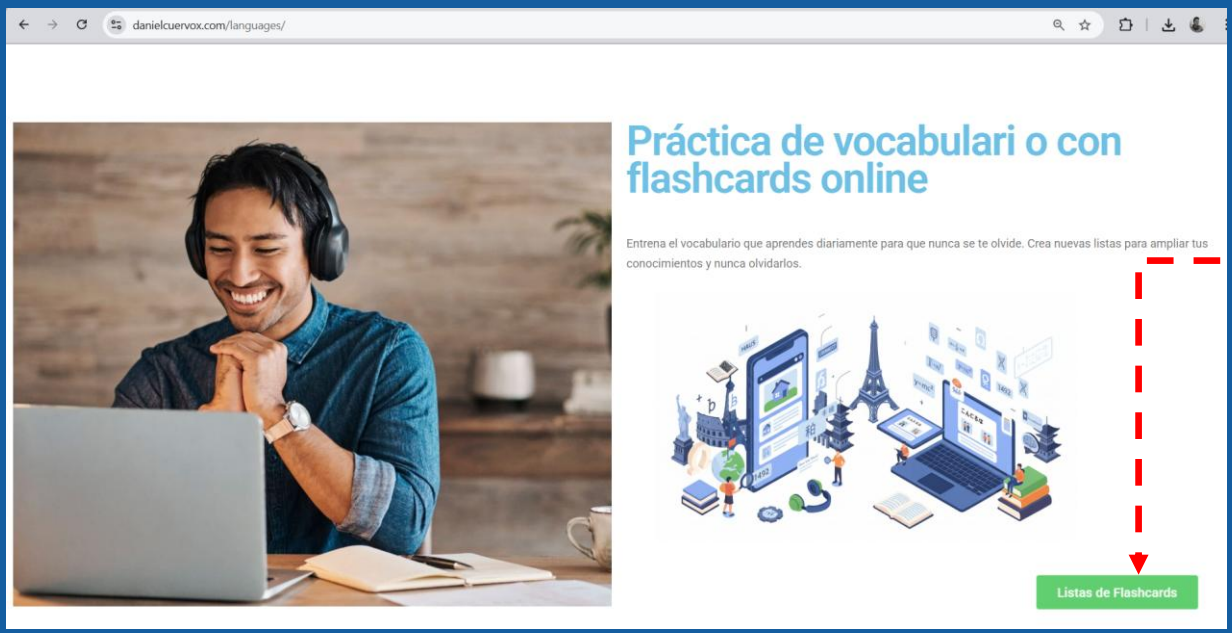


Una vez escribas el nombre de la lista y el prompt aparece el botón de generar y te muestra la lista que haya creado.

Web

El público inicial de mi aplicación son mis estudiantes de inglés, por eso dentro de mi página web aparecerá un link a una página donde el Usuario puede a las listas públicas y sus listas privadas.

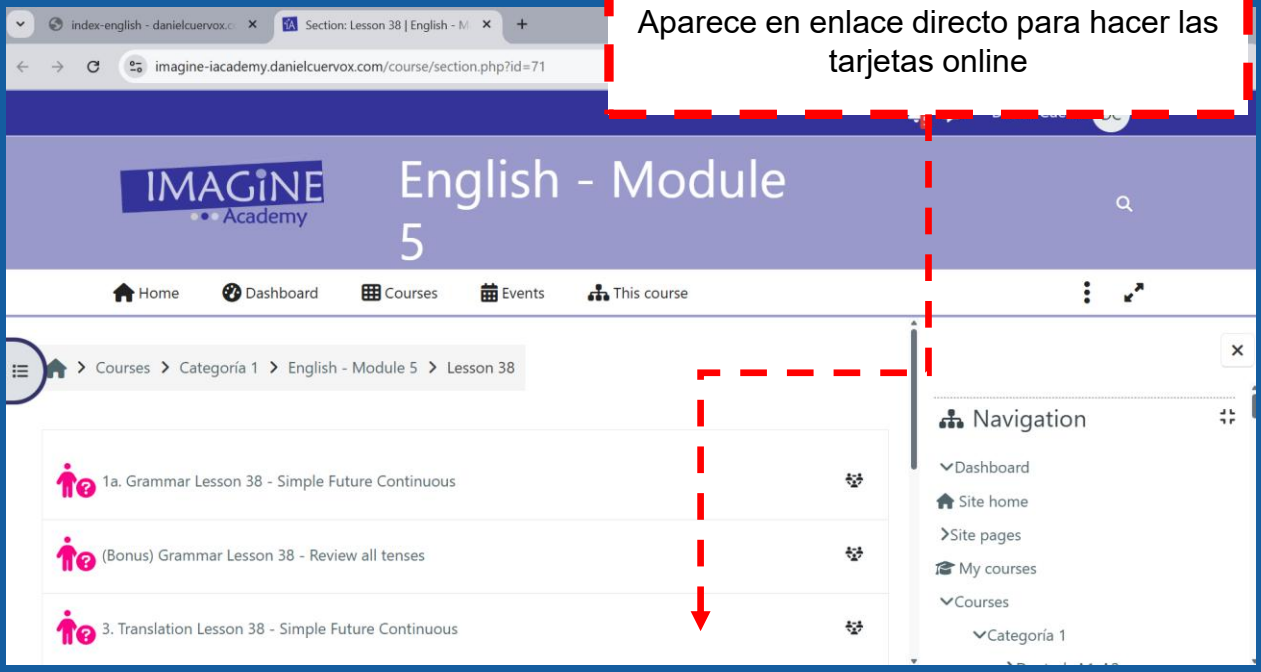
En el Landing page se encuentra el botón de acceso.



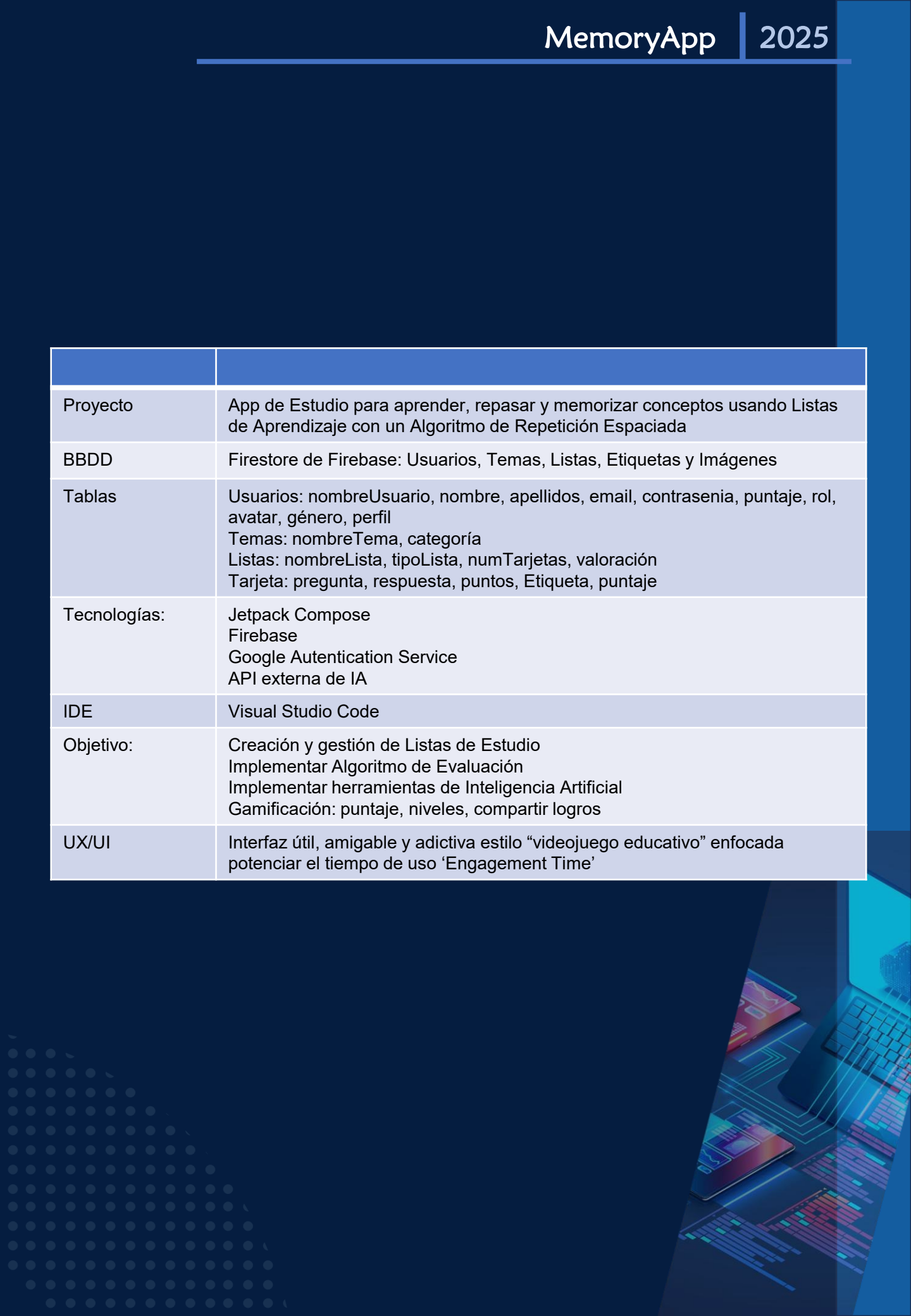
Diseño de persistencia de la información

Igualmente en cada una de las lecciones de la web tendrá el enlace directo a la lista de flashcards correspondiente.

Aparece en enlace directo para hacer las tarjetas online



Proyecto	App de Estudio para aprender, repasar y memorizar conceptos usando Listas de Aprendizaje con un Algoritmo de Repetición Espaciada
BBDD	Firestore de Firebase: Usuarios, Temas, Listas, Etiquetas y Imágenes
Tablas	Usuarios: nombreUsuario, nombre, apellidos, email, contrasenia, puntaje, rol, avatar, género, perfil Temas: nombreTema, categoría Listas: nombreLista, tipoLista, numTarjetas, valoración Tarjeta: pregunta, respuesta, puntos, Etiqueta, puntaje
Tecnologías:	Jetpack Compose Firebase Google Autentication Service API externa de IA
IDE	Visual Studio Code
Objetivo:	Creación y gestión de Listas de Estudio Implementar Algoritmo de Evaluación Implementar herramientas de Inteligencia Artificial Gamificación: puntaje, niveles, compartir logros
UX/UI	Interfaz útil, amigable y adictiva estilo “videojuego educativo” enfocada potenciar el tiempo de uso ‘Engagement Time’



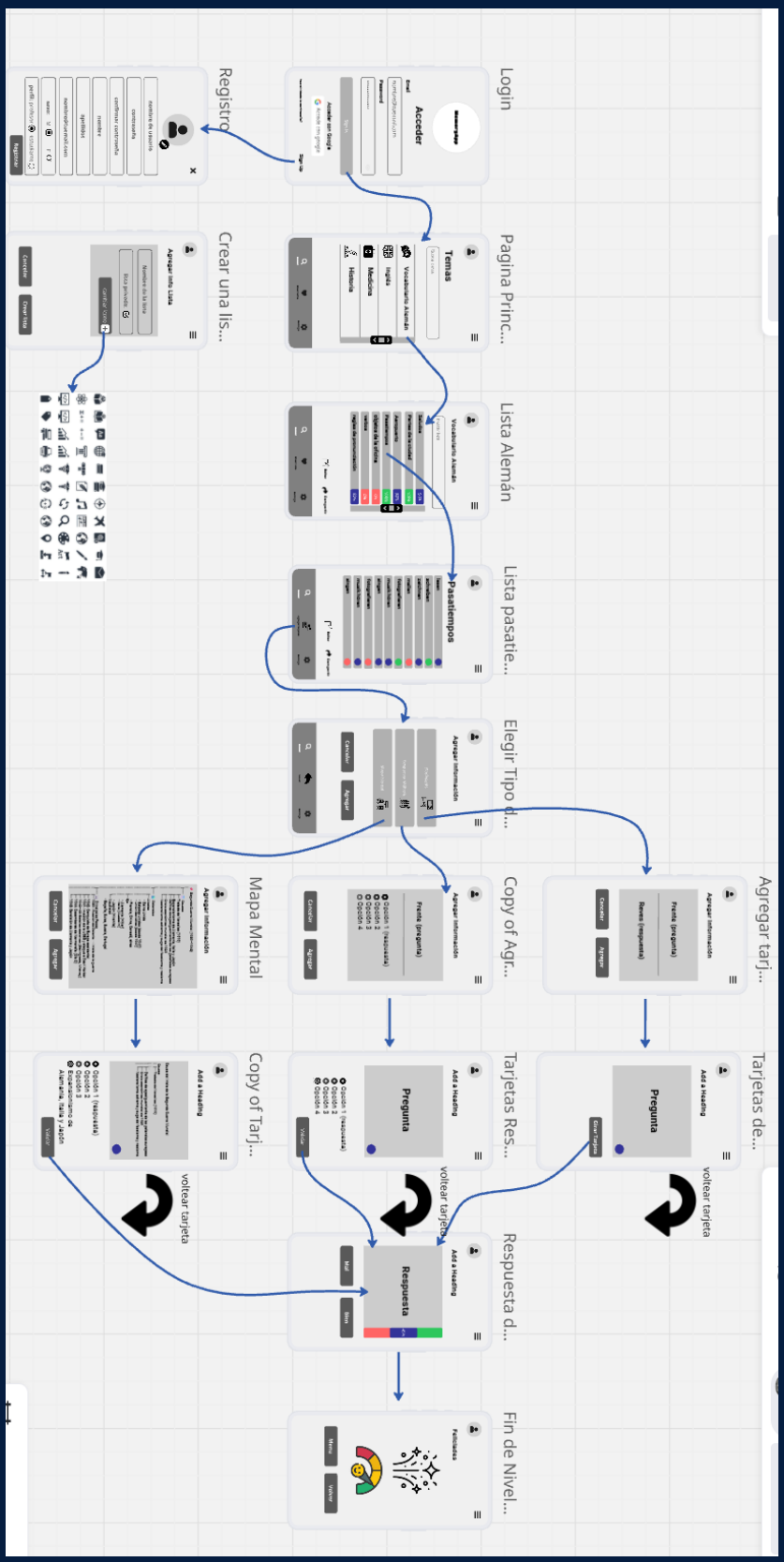
Análisis de la Solución

Mapa de Usuario

Elemento	MemoryApp: aplicación de flashcards para memorizar palabras y conceptos donde el estudiante se evalúa sumando o restando puntos, el algoritmo modifica la frecuencia que aparece cada flashcard.
Persona	“Alba, estudiante de inglés frustrada, está en un curso de inglés y busca constantemente app para practicar.
Escenario	Diariamente apunta las palabras nuevas en inglés que ve en el curso, además tiene que preparar diferentes exámenes, por eso ha utilizado Duolingo, actualmente tiene una evaluación de verbos irregulares.
Fases del Recorrido	1. Realiza una búsqueda en Google y Youtube sobre una app de memorización 2. Descarga MemoryApp para probarla 3. Utiliza MemoryApp creando su propia lista personaliza y descubre que también hay una comunidad que crea listas que puede descargar 4. Evalúa su progreso.
Acciones	Descarga la app, Acción: Realiza una sesión de 10 minutos, aprende los verbos irregulares con las 3 actividades propuestas en la App.
Pensamientos	Piensa que la app no es difícil de configurar, tiene una interfaz amigable y moderna y que en esta app puede almacenar el vocabulario que apunta cada día.
Emociones	Siente progreso y esperanza de encontrar una herramienta de repaso efectiva.
Puntos de Dolor	Aunque el registro y apuntar las palabras el toma un tiempo ve que hay una ventana cognitiva.
Oportunidades	Descubre que MemoryApp es una herramienta de un ecositema de aprendizaje más grande en imagine-iAcademy y decide apuntarse.

Navegabilidad

En este Enlace se puede ver el diagrama de navegabilidad más grande:



Herramientas Usadas

IDE de desarrollo, donde utilizo Compose el framework / tecnología de Android, la cual es UI declarativa, para desarrollar y gestionar la interfaz de mi aplicación MemoryApp.

Base de datos JSON en tiempo real que he decido utilizar debido a su amplio abanico de opciones: Firestore, Authenticator, RealTime Database y Storage, además de sincronización en tiempo real con todos los dispositivos conectados.

Es un cliente gráfico gratuito para Git, permite gestionar repositorios permite gestionar ramas, commits y merges de forma visual.

Es una herramienta de diseño online que utilizo para graficar la estructura y navegabilidad de la aplicación mostrando las ventanas, los botones, campos de texto e imágenes y la interacción que puede tener el usuario entre ventanas.

Google Drive lo utilizo como herramienta de respaldo para almacenar la documentación de trabajo escrito, imágenes y archivos .zip que genero periódicamente para el proyecto.

Utilizo la API-REST de Google para generar las Listas de palabras cuando el usuario crea una ListaIA. Permite que la creación de Listar o Tarjetas no sea solamente manual, además garantiza precisión en los datos.

Análisis de la Solución

Requisitos funcionales

Permitir que el usuario acceda a la App después de haberse registrado con su nombre de usuario y contraseña, o también puede acceder a la aplicación con Google, una vez allí puede editar su perfil de usuario agregando o editando su información personal.

El usuario puede crear Temas y agregar un ícono para personalizar el tema, seguidamente puede borrar o editar el tema. En este tema se almacenarán las Listas que considere pertenecientes al mismo. Puede agregar tantos temas como desee.

Dentro de los temas puede agregar las listas de ese tema, de igual maner esas listas pueden ser editadas o borradas.

Dentro de las listas es donde se crean las tarjetas de memorizar, el usuario puede crear flashcards tracionales (pregunta – respuesta), también puede crear flashcards generadas con inteligencia artificial. Puede seleccionar si tiene una sóla respuesta o es respuesta mútiple también generada con inteligencia Artificial.

Otra opción es crear tarjetas de preguntas basadas en un texto que genera un mapa conceptual, donde las preguntas son partes de ese mapa mental y el usuario debe seleccionar la respuesta correcta.

La Aplicación permite al usuario agregar sonido dentro de las tarjetas para que sea más ameno y realista el aprendizaje de idiomas.

El usuario puede compartir las listas creadas, y puede descargar listas creadas por otros usuarios para ahorrar tiempo.

Cuando el usuario realice la actividad con flashcards el usuario puede autoevaluarse, el algoritmo aumentará o disminurá el puntaje de cada tarjeta lo que modificará la frecuencia en la que son mostradas.

Requisitos no funcionales

El sistema debe permitir a usuario crear innumerables Temas, listas y tarjetas que considere para su posterior estudio, aumentando o disminuyendo el puntaje.

Debe permitir compartir las listas en la web para que también puedan ser usadas y descargadas por otros usuario, tanto de la versión Web de MemoryApp como de desde imagine-iAcadeny.

Debe permitir a usuario registrarse e ingresar con su nombre de usuario y contraseña o través de Google, pues los temas y listas son propias para ese usuario, y las que descargué se agregarán a su perfil.

Debe funcionar de manera fluida para que no haya *Lags* al ingreso, navegación y estudio con las *flashcards*.

La interfaz debe ser atractiva, debe verse moderna y profesional, la UI/UX es un aspecto clave a desarrollar y mejorar.

Conjuntamente la App debe ser robusta para que el usuario pueda costumizar en medida de lo posible su usuario y no se cierre repentinamente y haya pérdida de información.

Requisitos de información

El Sistema debe guardar las tarjetas creadas por el usuario, el contenido escrito no tiene un límite con respecto al número de caracteres, se plantea también para la segunda etapa poder subir imágenes y audio para las tarjetas.

También se debe guardar el puntaje de las Listas y las Flashcards cuando se crea y cuando el usuario realiza la actividad debe modificar su puntaje.

De igual manera el usuario debe ver su puntaje de las Listas y las Flashcards en la interfaz, por eso es crucial la recuperación de los datos.

Una vez se implemente se debe compartir, para eso se debe gestionar el formato de la exportación, serialización y el almacenamiento, esto se logrará serializándolo a JSON y las colecciones públicas de Firestore.

Actores

Usuario Registrado

Es el usuario final que se beneficia de las funcionalidades de la aplicación.

Identificación: Cualquier persona que descargue la aplicación, la instale, se registre y cree Flashcards.

Rol Principal: Crear Flashcards, aprender, memorizar, participar en la comunidad.

Funciones:

- Registrarse en la App.
- Crar Temas, Listas y Tarjetas.
- Estudiar las listas que ha creado.
- Personalizar su interfaz.
- Compartir las listas creadas.

Sistema (IA)

Es un sistema externo automatizado que provee servicios de valor al usuario.

Identificación: La **API de Inteligencia Artificial** (el modelo de lenguaje de gran escala o LLM) con la que te conectas para generar contenido.

Rol Principal: proveer el contenido estructurado de la petición del usuario con el Promt

Funciones Clave que Realiza:

- Recibir el prompt enviado por el usuario
- Devolver el contenido generado a la App

Desarrollador/Administrador

Es la persona que mantiene y mejora la aplicación en base a la experiencia de usuario del tráfico orgánico que vaya teniendo MemoryApp

Rol Principal: Mantener, moderar y escalar la aplicación.

Funciones Clave que Realiza:

- Monitorear el sistema (*troubleshooting* y *estadísticas de Firebase)*
- Optimizar el funcionamiento de la App y desarrollar la Segunda Etapa
- Gestionar el servidor la API.

Casos de Uso

Sistema

login

<include>

registro

<extend>

Login con google

Evalúa y modifica puntuación

Crear Tema

<include>

Crear lista

<include>

<include>

Crear flashcard

Usuario

Subir lista onine

<extend>

<extend>

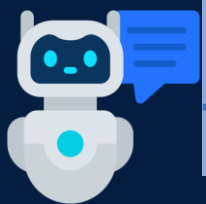
Descargar lista online

flashcard (repuesta müstiple)

<extend>

Mapa mental

Modificar perfil



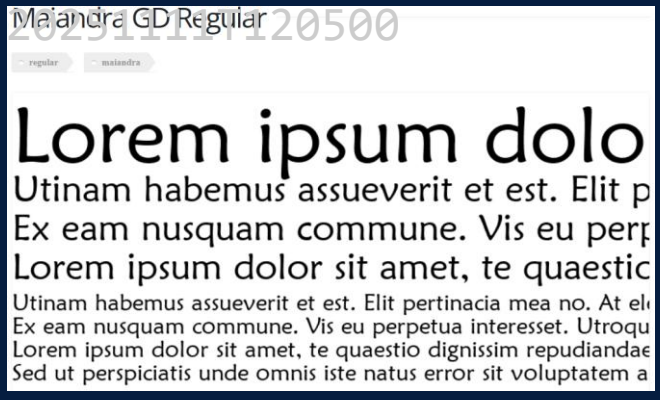
IA

Diseño de la Solución

Diseño de la interfaz de usuario y prototipos

Este es un punto crítico de la aplicación y son vitales para una larga vida del proyecto, tanto por el éxito comercial como la retención de los usuarios. El UX se enfoca en optimizar cómo se siente los sentimientos, funcionabilidad, facilidad de uso de la App y el UI en el aspecto visual (colores, botones, tipografía). MemoryApp no sólo vende la idea de crear flashcards, vende la idea de estudiar y memorizar sin esfuerzo ambos deben asegurar que la aplicación profesional.

Fuentes: Maiandra GD



Colores:

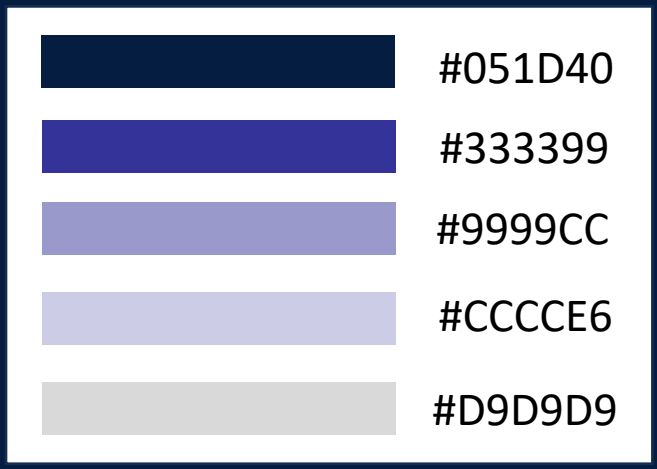
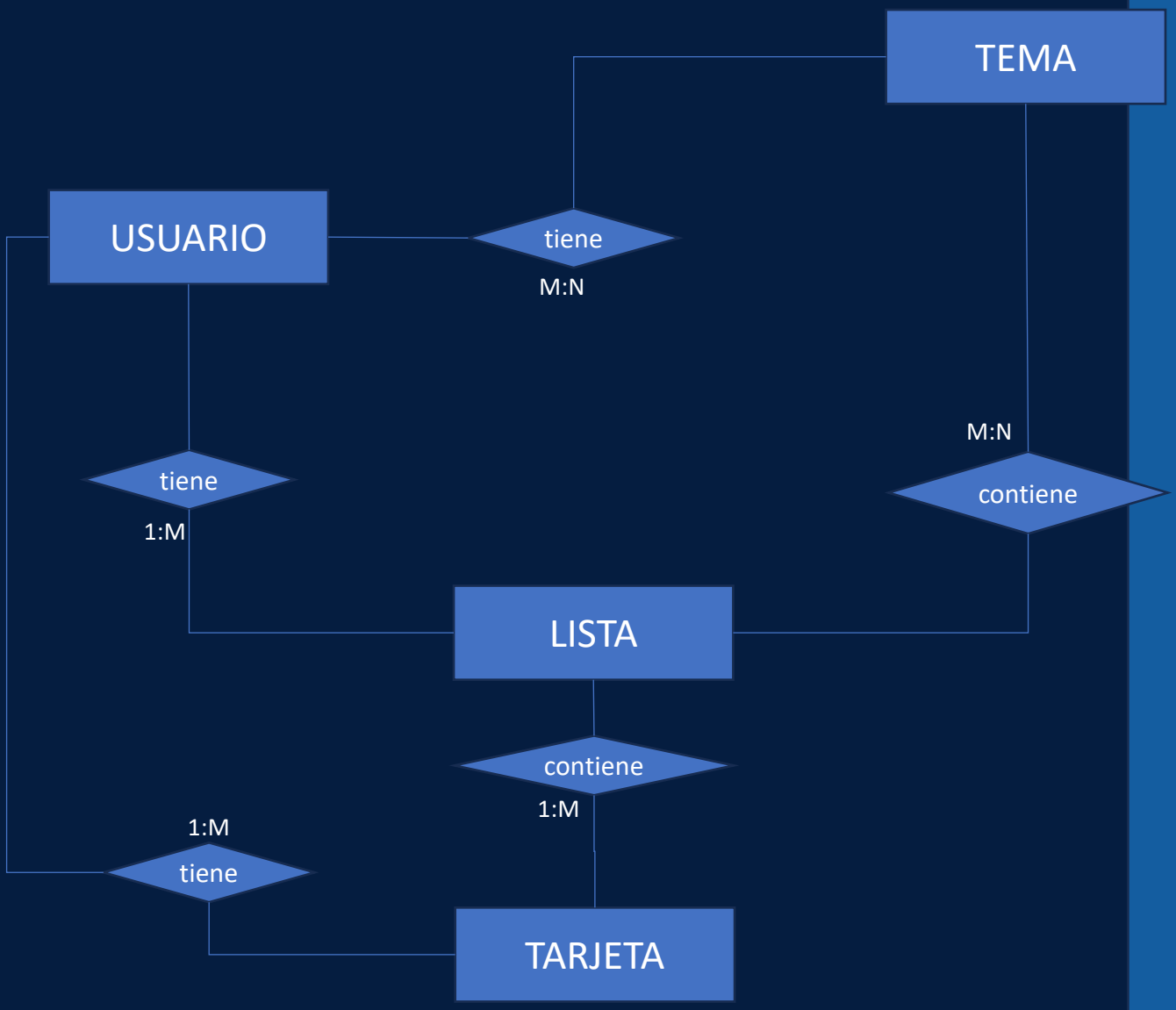


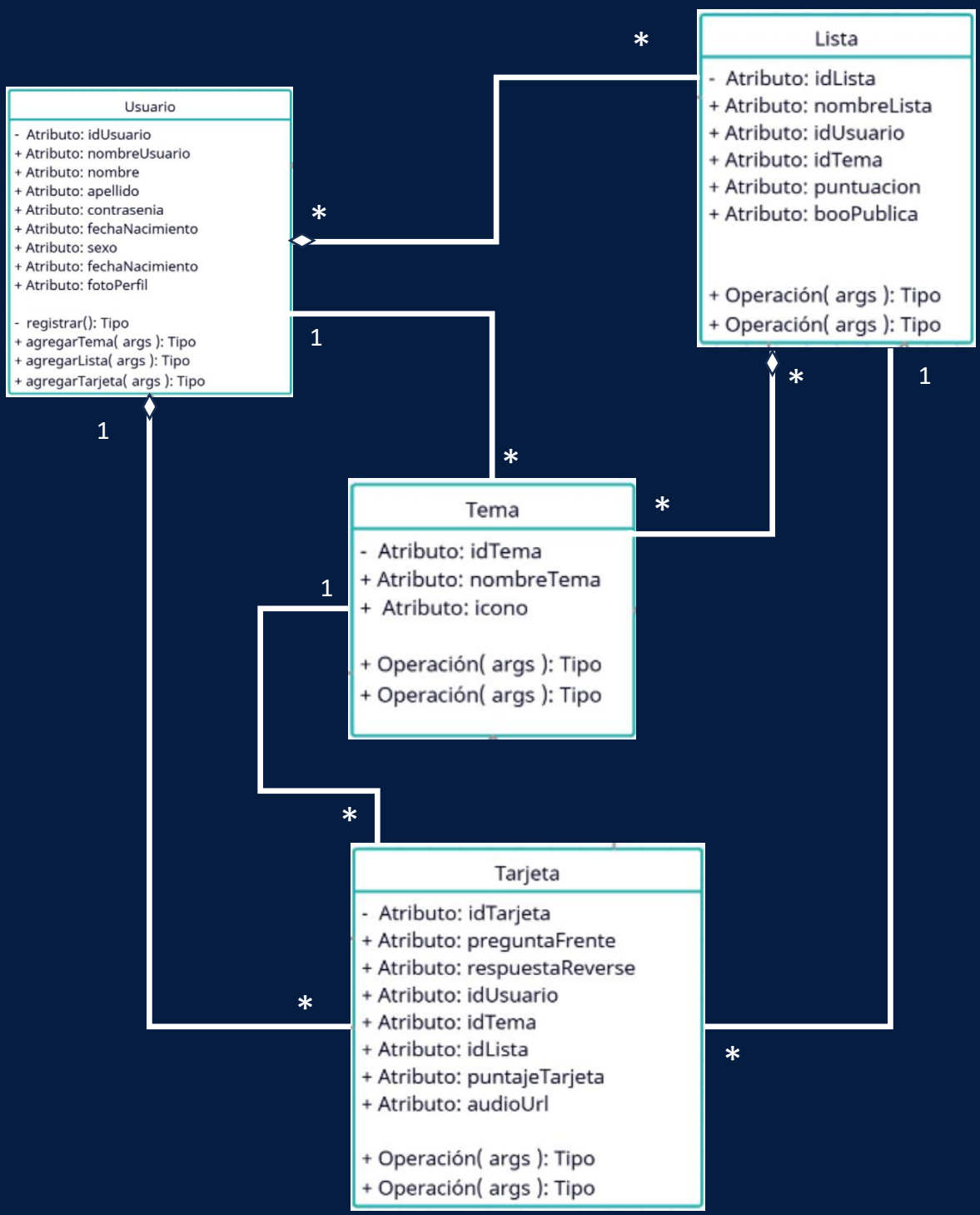
Diagrama de clases



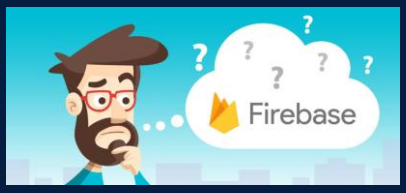
Explicación del Diagrama Entidad-Relación

Un usuario crea un o muchos temas, los temas continen uno o muchas listas, cada una de las listas contenen una o muchas tarjetas, al ser una Base de datos no relacional, para modificar la puntuación de cada Lista y Tarjeta se debe filtar por idUsuario, IdTema e idLista (necesario en el caso de la Tarjeta). También tenemos la entidad Lista compartida, donde el Usuario pude subir y descargar una lista y esta se agrega en un Tema.

Diagrama de Clases



Diseño de la persistencia de la información

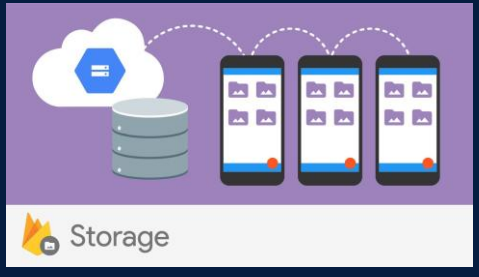


Para el proyecto de mi aplicación he decidido usar Firebase, porque me parece una herramienta muy potente y versátil, además de que te evita tener que crear la base de satos SQL tienes el respaldo de Google lo que me permite enfocarme en el desarrollo de mi aplicación.

Firebase es una plataforma desarrollada por Google que ofrece herramientas y servicios en la nube, además de otros servicios y todos dentro de un mismo entorno que además está optimizado para Android permitiendo que los datos se sincronicen y actualicen de forma casi instantánea sin necesidad crear o gestionar una API o un servidor.

Authenticator: tiene la integración integrada para las diferentes proveedores externos como: Google, Github, Facebook, Apple, Twitter, Outlook. Esto permite que el usuario pueda ingresar con un solo click sin necesidad de estar rellenando formularios de registro y lo más importante sin necesidad de estar recordando contraseñas.

Otra herramienta muy interesante es Storage, donde se pueden almacenar las fotos de perfil creadas por el usuario, los audios de las tarjetas y en un futuro para que el usuario pueda subir fotos o screenshots de temas que quiera estudiar y se almacenen en el Storage de Firebase.

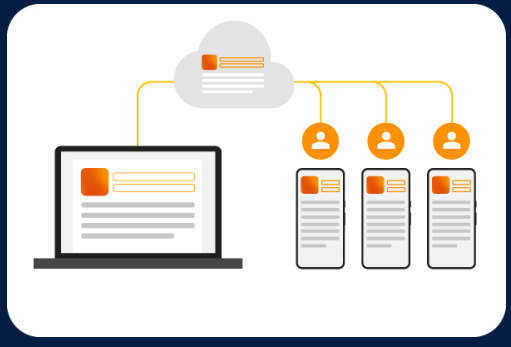


Diseño de la persistencia de la información

Con Analytics se puede medir el uso de la App, cómo los usuarios interactúan con la App, detectar fallos en tiempo real y analizar el rendimiento, todo esto se puede ver en:

- Número de usuarios en activo
- Dispositivos y Sistemas Operativos
- Sesiones y duración

Aunque en mi proyecto no tengo un chat, sí quiero resaltar RealTime Database de Firebase, muy utilizada para chat, no veo la necesidad de un chat en mi aplicación, ni siquiera un Chatbot de Inteligencia Artificial o la creación de listas colaborativas, pero es importante saber que existe esta herramienta si en algún punto de desarrollo de mi proyecto se pueda usar.

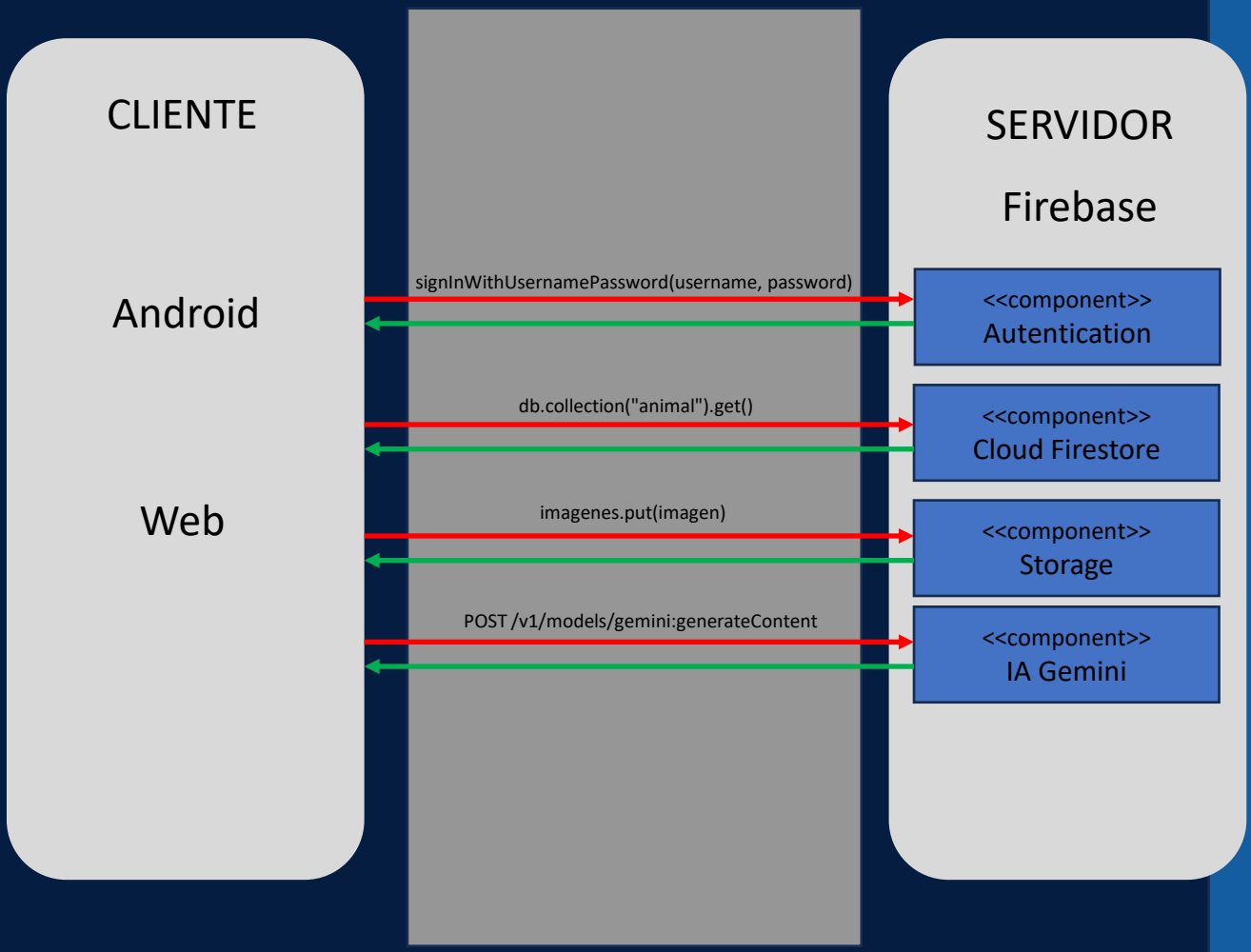


Además tiene un integración muy completa co Android, siendo ambos productos de Google,

La única que desventaja que puedo percibir es que te Google te ata a su ecosistema y si en algún momento se quieren hacer consultas muy complejas (joins y filtros combinados) Firebase no es la herramienta ideal, ya en ese caso se debe usar un backend propio o AWS/Azure.

Diagrama Cliente-Servidor

Este diagrama permite mostrar de forma visual como se comunica el cliente y el servidor dentro de la aplicación, mostrando la estructura técnica y los flujos de datos del sistema. En mi diagrama se muestra cómo actúa el cliente App/Web con el servidor Firebase (Firestone – Storage – API REST - Realtime) con el cliente y cómo se conectan entre sí utilizando HTTPS en cada



Justificacion Diagrama Cliente-Servidor

Authentication: este servicio ofrece la posibilidad de que el usuario se logue y acceda con un clic a la aplicación usando diferentes servicio a través de diferentes proveedores externos, actualmente para mi aplicación sólo tengo habilitado con Google.

RealTime Database: este servicio nos permitirá almacenar y acceder a toda la información que guarden los usuarios en la aplicación, por ejemplo cada uno de los atributos de nuestras clase como también las preguntas y respuestas de las tarjetas y los puntajes actualizándolos cuando sea el caso.

Storage: este servicio permitirá almacenar las imágenes dentro de las tarjetas que servirán también como preguntas o respuestas para la actividad de Memorización.

API-REST (Gemini): a través de un prompt enviado a la API-REST de Google Gemini se solicita la creación de un número de tarjetas con pregunta y respuesta. La respuesta generada por la API se serializa y convierte en formato JSON transformándose en Tarjetas, y finalmente se almacena la información en firebase para su uso.

Justificación tecnológica

Durante el periodo de análisis e investigación previo al desarrollo de la aplicación contemplé diferentes aplicaciones y tecnologías, teniendo en cuenta diferentes factores como conocimiento previo, usabilidad y documentación, escalabilidad y posibilidad de desarrollo multiplataforma. menciono algunas para justificar mi decisión final y justificación

Permite desarrollar aplicaciones multiplataforma usando un solo proyecto en .Net / C#. Tiene la carpeta Resources y Maui los optimiza para todos los dispositivos. Base de datos con SQLite, Realm, Azure y Firebase.

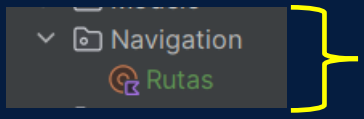
Es un Framework de aplicaciones móviles desarrollado por Meta (Facebook), también permite crear aplicaciones multiplataforma con un único código base en JavaScript/TypeScript. Base de datos con SQLite, Realm, Firebase, Watermelo.

Es un Framwork también para el desarrollo multiplataforma, usa Dart como lenguaje de programación, UI declarativa . Bases de datos: SQLite, Hive, Moor, Firebase, Supabase.

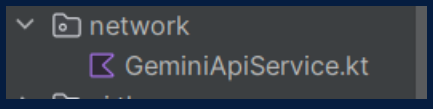
Elementos a implementar

Podemos ver una imagen donde muestra los diferentes clases y ventanas que tiene la aplicación

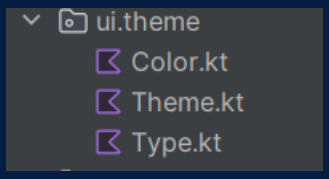
En este paquete se definen los Data Clases que representan las entidades fundamentales de la aplicación. También incluye 'ArticuloTienda' que contiene una estructura Map para gestionar los artículos del personaje.



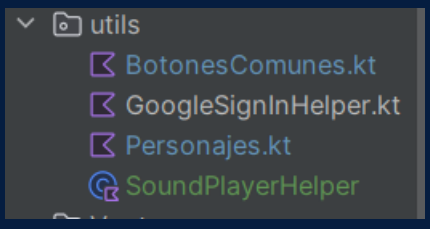
Contiene una Object Class donde se centraliza la definición de las rutas de las ventanas, de esta forma se gestiona la navegación de la aplicación permitiéndonos acceder de manera estática.



Este Package contiene la clase 'GeminiApiService' que realiza toda la comunicación con la API-REST



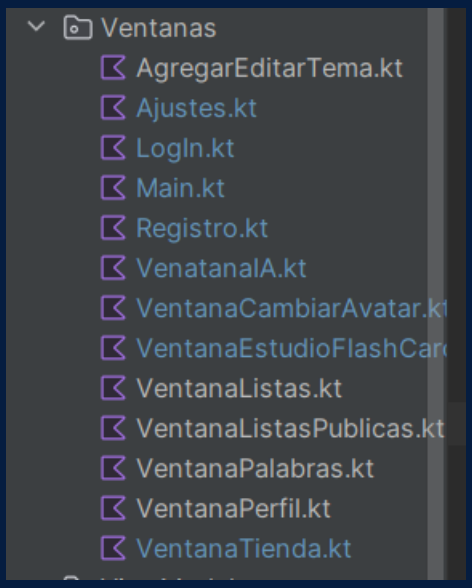
Contiene los estilos de color y fuentes utilizados en la aplicación.



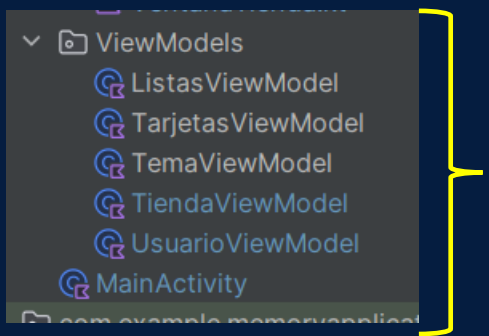
Utils contiene variedad de "Helpers" de herramientas específicas de la app como SoundPlayer, para la lógica de preproducción de sonido, o utilizados, 'BotonesComunes' su objetivo es encapsular funcionalidades específicas para no duplicar código.

Elementos a implementar

Podemos ver una imagen donde muestra los diferntes clases y ventanas que tiene la aplicación



Acá se pueden ver todas las ventanas o Screens composables que contiene la aplicación, como logIn, VentanaPrincipal o Tienda. Todos éstos son responsables de la estructura visual e interactúan con el ViewModel para mostrar datos.



Este es el pilar de la arquitectura MVVM. Contienen la lógica de negocio propia de cada Entidad de la aplicación. Su función es interactuar como intermediario entre las diferentes ventanas y con el Repository que lo comunica a la base de datos.

Implementación API REST

Creamos una APIKEY en Google AI Studio

Pegamos la implementación de Google GenerativeAI en el archivo Graddle

```
//implementation("com.google.ai.client.generativeai:generativeai:0.7.0")
//implementation("com.google.ai.client.generativeai:generativeai:0.10.0")
implementation(libs.generativeai)
```

También son necesarias otras implementaciones para serializar el archivo JSON que nos da la IA

```
// Retrofit para hacer las llamadas HTTP
implementation("com.squareup.retrofit2:retrofit:2.9.0")
// Convertidor para serialización/deserialización con Kotlinx JSON
implementation("com.jakewharton.retrofit:retrofit2-kotlinx-serialization-converter:1.0.0")
//  serializador de Kotlinx
implementation("org.jetbrains.kotlinx:kotlinx-serialization-json:1.6.0")
// Un interceptor para ver los logs de las llamadas
implementation("com.squareup.okhttp3:logging-interceptor:4.11.0")
```

En el archivo local.properties se debe pegar la ApiKey que nos da Google y se vincula con el Archivo de configuración Graddle.

```
#sdk.dir=C\:\\Users\\lezam\\AppData\\Local\\Android\\S
sdk.dir=C\:/Users/lezam/AppData/Local/Android/Sdk
# --- Mis claves personalizadas ---
GEMINI_API_KEY=AIzaSyB3iRCn_####################
```

```
buildConfigField( type = "String", name = "GEMINI_API_KEY", value = "\"$apiKey\"")
}
```

Implementación API REST

Creo una Clase TarjetaIA que es donde se va almacenar la información descerializada

```kotlin
@kotlinx.serialization.Serializable  3 Usages
data class TarjetaIA(
    val pregunta: String,
    val respuesta: String
)
```

Se crea una Clase 'ApiService' que es donde se escribe el modelo generativo que se va a usar y la configuración del mismo

```kotlin
interface GeminiApiService {  2 Usages
    //@POST("v1/models/gemini-1.5-pro-latest:generateContent")
    @POST( value = "v1/models/gemini-2.0-flash:generateContent")  2 Usages


    suspend fun generateContent(
        @Query( value = "key") apiKey: String,
        @Body request: GeminiRequest
    ): GeminiResponse
}

// --- Objeto para crear la instancia de Retrofit ---

object RetrofitClient {  3 Usages
    private const val BASE_URL = "https://generativelanguage.googleapis.com/"  1 Usage

    // Configura el cliente HTTP para poder ver los logs de las llamadas (muy útil para depurar)
    //private val httpClient = OkHttpClient.Builder()
        //.addInterceptor(HttpLoggingInterceptor().apply { level = HttpLoggingInterceptor.Level.BODY })
        //.build()

    private val logging = HttpLoggingInterceptor().apply {  1 Usage
        level = HttpLoggingInterceptor.Level.BODY
    }

    private val httpClient = OkHttpClient.Builder()  1 Usage
        .addInterceptor( interceptor = logging)
        .build()
```

Implementación API REST

En el Viewmodel se coloca el prompt donde especifica qué queremos obtener y en qué formato,

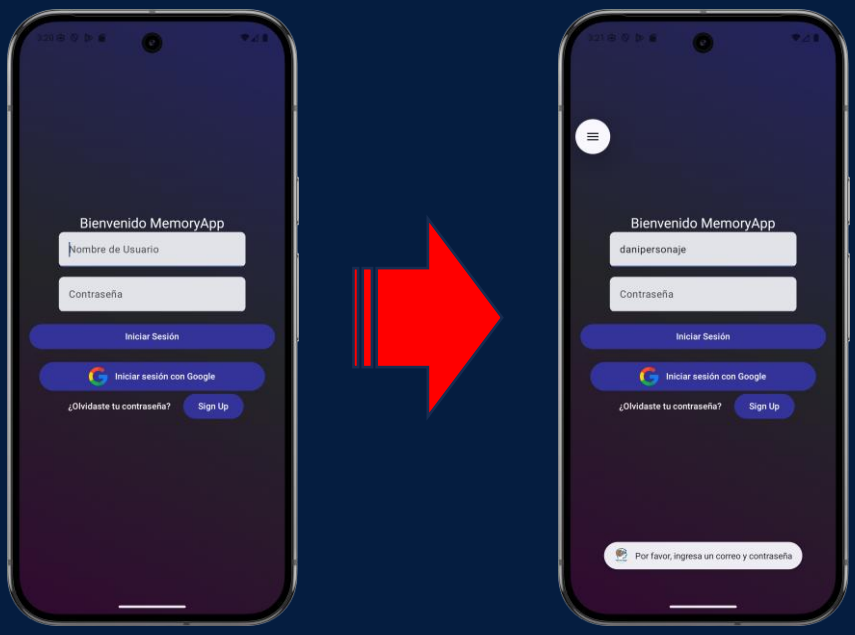
```
class ListasViewModel : ViewModel() {  14 Usages
    fun generarListaConIA(  1 Usage
        nombreLista: String,
        esPublia : Boolean,
        idUsuario: String,
        idTema: String,
        onResult: (Boolean, String) -> Unit
    ) {
        val apiKey = BuildConfig.GEMINI_API_KEY
        if (apiKey.isBlank()) {
            onResult(false, "API Key no encontrada. Revisa tu configuración.")
            return
        }

        viewModelScope.launch( context = Dispatchers.IO) {
            try {

                val promptCompleto = """
                Genera un array JSON para una app de flashcards. El tema es: "$prompt".
                La respuesta debe ser únicamente el array JSON, sin texto adicional, explicaciones o marcado de código.
                Cada objeto en el array debe tener dos claves: "pregunta" y "respuesta".

                Ejemplo de formato de salida para "capitales de países":
                [
                  {"pregunta": "¿Capital de Francia?", "respuesta": "París"},
                  {"pregunta": "¿Capital de Japón?", "respuesta": "Tokio"}
                ]
                """.trimIndent()
```

Asímismo tenemos en el Viewmodel el código para convertir el array de JSON que nos devuelve al IA, verifica que sea un JSON válido y lo serializa y lo convierte en una lista de objetos TarjetaIA que es lo que se almacena en Firebase.

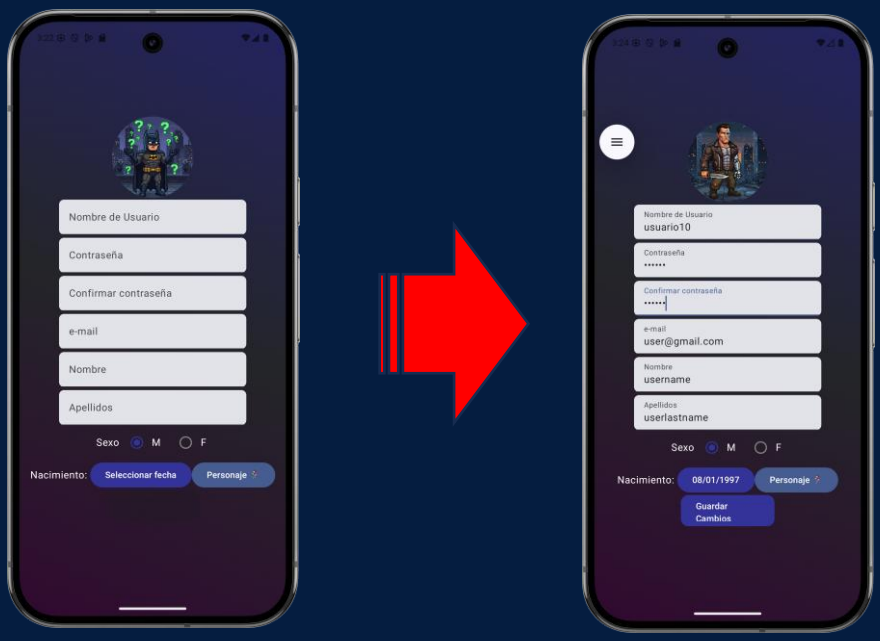
```
    // Llama a la API usando Retrofit
    Log.d(TAG,  msg = "Enviando petición REST a Gemini...")
    val response = RetrofitClient.apiService.generateContent(apiKey, requestBody)

    // Extrae el texto de la respuesta
    val part = response.candidates
        .firstOrNull()
        ?.content
        ?.parts
        ?.firstOrNull()

    val raw = part?.text ?: throw Exception( message = "La API no devolvió texto")

    // Limpiar ```json del inicio y fin si existe
    val cleanJson = raw
        .replace( oldValue = "```json",  newValue = "")
        .replace( oldValue = "```",  newValue = "")
        .trim()

    // Parsear JSON a lista de TarjetaIA
    val tarjetasIA: List<TarjetaIA> = try {
        kotlinx.serialization.json.Json.decodeFromString( string = cleanJson)
    } catch (e: Exception) {
        throw Exception( message = "La IA no devolvió un formato JSON válido.")
    }
```

Testeo y Plan de pruebas

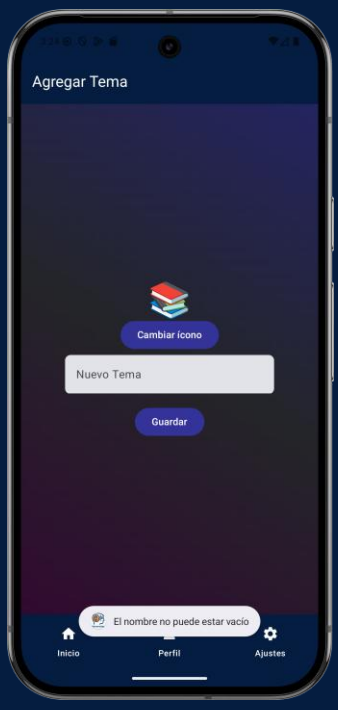
Prueba 1: se verifica que si el usuario intenta ingresar si un usuario y contraseña válido salte un mensaje de notificación. Así como también que esa contraseña pertenezca a ese usuario.



Prueba 2: Para el registro se verifica que el usuario ingrese el información mínima requerida además debe seleccionar un avatar y tener más de 6 años.



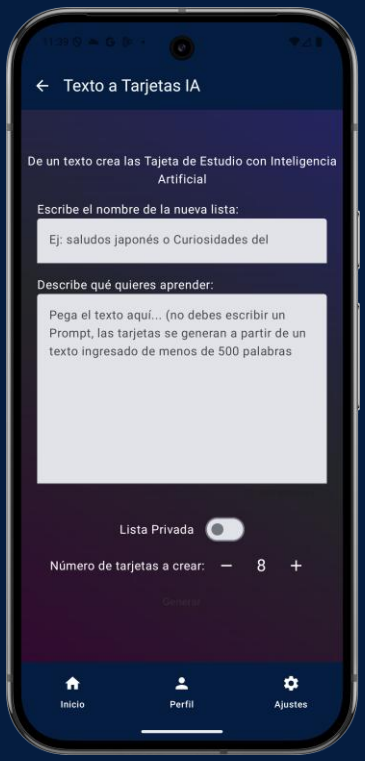
Prueba 3: Se verifica que el usuario tenga que escribir un nombre para crear un Tema y que ese nombre no esté siendo usado en su base de datos.



Prueba 4: Se controla que el usuario no pueda crear una lista sin nombre, ya que se activa el botón de Guardar cuando ha ingresado los campos.

Prueba 5: Se controla que el usuario tenga que escribir un nombre válido no usado para la lista IA y además tenga que escribir un prompt.

Prueba 6: Cuando el usuario va a crear listas de estudio desde un Texto se controla que no pueda tener más de 500 palabras para que no exceda el número de tokens y sature la IA.



Bibliografía

Precio/hora de un programador junior:
https://yeeply.com/blog/partners/precio-hora-programador-cuanto-cobra-desarrollador-de-software/

Cuánto cuesta subir una App en la Play/App store:
https://gooapps.es/2022/02/21/cuanto-cuesta-subir-una-aplicacion-a-una-app-store/

Se necesita un mac para programar para iOS
https://desarrolloweb.com/faq/necesito-mac-para-desarrollar-apps-ios

Ventajas de CloudStore
https://keepcoding.io/blog/cuales-son-las-ventajas-de-cloud-firestore/

Productos y Servicios de Firebase
https://keepcoding.io/blog/productos-de-firebase/

Mejor Ia para una App en Android Studio
https://developer.android.com/ai/overview?hl=es-419

Limites de almacenamiento de Storage de Firebase
https://firebase.google.com/pricing?hl=es-419