

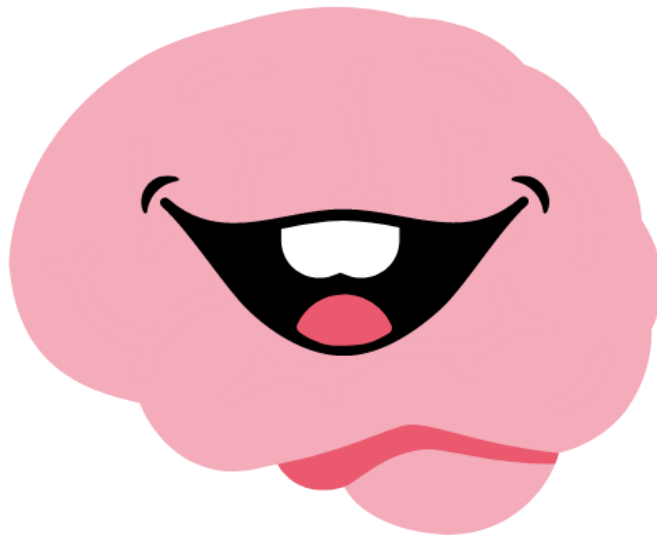


Trabajo de Fin de Grado

Desarrollo de Aplicaciones Multiplataforma

# FelizMente

Aplicación de estimulación cognitiva  
y mediación cultural para la tercera edad



Laurence Apuya

Alexandra Mendoza Robinska

Sarah Amselem Felices



## **1. Introducción**

Manuel Felices, el abuelo de Sarah, al igual que muchos de nuestros mayores, gozaba de una condición física e intelectual admirable. Sin embargo, cuando alcanzó los ochenta años empezó poco a poco a perder sus facultades, y conforme pasaban los años, el ritmo de deterioro se disparó.

Observar los cambios en su abuelo, así como el sufrimiento en su entorno, despertó en Sarah un deseo por mejorar la calidad de vida de aquellos que sufren deterioro cognitivo, y las personas que los acompañan.

Tras informarse acerca de productos digitales para prevenir y ralentizar el proceso de deterioro, observó que la mayoría de actividades propuestas en las aplicaciones resultaban anodinas, no pudiendo identificarse el paciente con ellas. Además, no había encontrado ninguna plataforma accesible que ofreciera contenido especializado.

Sarah creía que se podía hacer algo mejor y así surgió la idea de FelizMente.

FelizMente nace de la necesidad de ofrecer a los mayores con enfermedades neurodegenerativas un portal con contenidos curados, acorde a la memoria colectiva. Estas imágenes despiertan emociones en el paciente, así como un sentimiento de pertenencia e identidad, crucial para una fácil adhesión a la terapia. Estas prácticas se asocian al uso de la Terapia de Reminiscencia. Se pretende que el usuario pueda sentirse a gusto y ejercitar su mente de una manera amena, involucrándose con aquello que le importa.

El trabajo presentado no es más que el germen de un proyecto solidario que mejorará la vida de muchas personas.

En esta primera versión se ha implementado, además de un registro y un inicio de sesión, el acceso a vídeos de YouTube clasificados por categorías y un simpático juego de preguntas sobre personajes populares y cultura española.

## **Agradecimientos**

*“Estos dos últimos años han estado llenos de retos a nivel personal y, sobre todo, mental en los que he aprendido a gestionar la frustración a base de constancia y mucho humor.*

*La paciencia infinita de mi marido Alfonso y el resto de familiares y amigos, ha sido clave para afrontar esta etapa con un mínimo de cordura. Pero sin duda a los que más tengo que agradecer es tanto a los profesores, por la formación recibida, como a los compañeros por el compañerismo demostrado. Quiero hacer especial mención a Cristian Fernández Mirón, mi compañero de equipo durante estos dos años, ya que, sin su constante trabajo, paciencia y generosidad, no habría llegado hasta aquí.*

*Y por último agradecer a mis compañeros de proyecto, Laurence y Sarah, que me han acogido en esta recta final como parte de su equipo, demostrando una fe ciega en mí e impulsándome con sus palabras e ideas para dar siempre lo mejor de mí.”*

### **Alexandra Mendoza Robinska**

*“A mis abuelos: Pepi y Manolo.*

*A SARAIVA por su labor, y por haber cuidado tan bien de mi abuelo durante su último año de vida.*

*A mi tía María del Mar y a mi tío Diego.*

*A mi madre, por ser la persona más excepcional que he conocido.*

*A Javier, por haberme enseñado qué es el amor (y la programación).*

*A mi hermano Daniel y a mi familia política.*

*A Laurence y a Alexandra por haberme ayudado a sembrar la semilla de FelizMente.*

*A Guille, Luz e Iván, con los que he compartido penas, bromas y cotilleos durante mi paso por EDIX.*

*Y a las 800.000 personas que padecen Alzheimer en España.”*

### **Sarah Amselem Felices**

*“Agradecer primero a todo EDIX por tan gran trabajo y profesionalidad.  
A todos los compañeros que he conocido en el instituto, con particular alusión a mis  
compañeros de proyecto Alexandra y Sarah por ser de lo mejor en mi paso por EDIX.  
A mi familia y amigos que siempre confiaron y estuvieron a mi lado.  
Especial mención a Nesy, Rubén y Manu por ser los grandes pilares que me sustentan.  
A mi madre por todo su amor e infinita paciencia.  
Y como no, a mis mejores amigos Bunito y Bob, sin ellos no estaría aquí.”*

**Laurence Apuya Pangilinan**

## Glosario con palabras clave

**Activity:** se corresponde en Android con una pantalla de nuestra App. En realidad es un punto de entrada que Android puede cargar en cualquier momento.

**Accesibilidad cognitiva:** condición que han de satisfacer los textos, carteles, tecnología y pictogramas para que todas las personas puedan entenderlos con facilidad.

**Admin:** aplicación del proyecto para la administración de usuarios desde la cual se controla la gestión de creación, modificación y borrado de usuarios.

**Android Manifest:** fichero de tipo XML situado en la raíz de la aplicación. Se trata del archivo de configuración donde se pueden aplicar las configuraciones básicas de la app.

**API:** del inglés *application program interface*, es un conjunto de definiciones y protocolos que se usa para diseñar e integrar el software de las aplicaciones.

**APK:** paquete para el sistema operativo Android. Este formato es una variante del formato JAR de Java y se usa para distribuir e instalar componentes empaquetados para la plataforma Android para teléfonos inteligentes y tabletas.

**Bug:** error de software.

**Constraint Layout:** descendiente de *ViewGroup*, cuyo fin es permitir posicionar y redimensionar *views* de forma flexible a partir de una gran variedad de reglas de restricción.

**CRUD:** del inglés *Create, Read, Update and Delete*. Se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.

**DAO:** acrónimo de *Data Access Object*, es un patrón de *software* que se encarga del acceso a los datos, que básicamente tiene que ver con la gestión de diversas fuentes de datos y además abstrae la forma de acceder a ellos.

**EndPoint:** son cualquier punto que sea la parte final de una red.

**Enfermedad neurodegenerativa:** trastorno que afecta al sistema nervioso central (SNC). Se produce una pérdida neuronal progresiva en áreas determinadas del cerebro, que suele ir acompañada de cambios en la conducta.

**Estimulación cognitiva:** uso de actividades para mejorar o frenar el deterioro de las capacidades intelectuales residuales del paciente.

**GUI:** *Graphical User Interface* o interfaz gráfica de usuario.

**Guideline:** elemento propio de *Constraint Layout*. Este objeto se usa como ayuda para maquetar. Es imperceptible en el dispositivo. Puede ser horizontal o vertical y se puede situar en una posición fija o en relación a un porcentaje del ancho o alto de la pantalla.

**JSON:** acrónimo de JavaScript Object Notation, es un formato de texto sencillo para el intercambio de datos.

**Layout:** objeto que representa el espacio contenedor de todas las *views* dentro de la actividad. En él se define la estructura y el orden de los elementos.

**Mediación cultural:** acercamiento del patrimonio a la población general o a un sector específico de la misma.

**Proxy:** es un servidor, en nuestro caso una clase, que hace de intermediario en las peticiones de recursos que realiza un cliente (A) a otro servidor (C).

**Query:** consulta a una base de datos.

**Servicio REST:** Es una interfaz para conectar varios sistemas basados en el protocolo HTTP y nos sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como XML y JSON.

**Terapia de Reminiscencia:** técnica de estimulación cognitiva . Emplea estímulos que despiertan la memoria episódica autobiográfica. En este caso se aplica un enfoque orientado a la memoria colectiva, en lugar de aspectos específicos del individuo.

**URL:** Significa *Uniform Resource Locator* y es la dirección única y específica que se asigna a cada uno de los recursos disponibles de la *World Wide Web* para que puedan ser localizados por el navegador y visitados por los usuarios.

**View:** Una *view* es un objeto cuya clase es `android.view.View`. Es una estructura de datos cuyas propiedades contienen los datos de la capa, la información específica del área rectangular de la pantalla y permite establecer el *layout*. Una *view* tiene: *layout*, *drawing*, *focus change*, *scrolling*, etc.

**W3C:** el consorcio WWW o *World Wide Web Consortium* es una organización internacional que genera recomendaciones y estándares que aseguran el crecimiento de la *World Wide Web* a largo plazo.



# ÍNDICE

1.	Resumen	10
2.	Módulos formativos utilizados	11
2.1.	Programación	11
2.2.	Programación Multimedia y Dispositivos Móviles	11
2.3.	Programación de Servicios y Procesos	11
2.4.	Bases de Datos	11
2.5.	Acceso a Datos	11
2.6.	Entornos de Desarrollo	11
2.7.	Inglés Técnico	12
2.8.	Empresa e Iniciativa Emprendedora	12
2.9.	Lenguajes de Marcas y Sistemas de Gestión de la Información	12
3.	Herramientas y lenguajes empleados	13
3.1.	Java	13
3.2.	<i>Android Studio</i>	13
3.3.	<i>IntelliJ Idea</i>	13
3.4.	<i>Spring Framework / Spring Boot</i>	13
3.5.	<i>XAMPP</i>	14
3.6.	Librerías	14
3.6.1.	<i>Lombok</i>	14
3.6.2.	<i>Retrofit</i>	14
3.6.3.	<i>Glide</i>	15
3.6.4.	<i>android-youtube-player</i>	16
4.	Componentes del equipo y funciones	17
4.1.	Alexandra Mendoza	17
4.2.	Sarah Amselem	17
4.3.	Laurence Apuya	17
5.	Fases del proyecto	18
5.1.	El plan de Empresa	18
5.1.1.	Idea de Negocio	18
5.1.2.	<i>Business Model Canvas</i>	18
5.1.3.	Identidad de Marca	19

5.1.4.	Financiación	19
5.1.5.	Estudio de Mercado	20
5.2.	Casos de uso	21
5.3.	La arquitectura del proyecto	22
5.3.1.	Arquitectura de Admin	22
5.3.2.	Arquitectura de la aplicación Android	23
5.3.3.	Arquitectura del servicio REST	24
5.4.	La aplicación Android	25
5.4.1.	El diseño	25
5.4.1.1.	Accesibilidad	25
5.4.1.2.	La tablet como soporte ideal	26
5.4.1.3.	<i>ConstraintLayout</i>	27
5.4.2.	Otras funcionalidades	28
5.4.2.1.	<i>CardView</i>	28
5.4.2.2.	Ventanas de diálogo	28
5.4.2.3.	Funcionalidades especiales	29
5.5.	Servicio <i>REST</i>	31
5.6.	La aplicación <i>Admin</i>	32
5.7.	<i>Testing</i>	34
5.8.	La página web	35
6.	Conclusiones	36
7.	Bibliografía	38
8.	<i>Anexo</i> (manual de usuario)	39

## 1. Resumen

FelizMente es un portal de contenidos para la estimulación cognitiva de la tercera edad. Se hace uso de la Terapia de Reminiscencia, siendo estos contenidos parte del imaginario colectivo de los mayores de nuestro país.

La piedra angular del proyecto es una aplicación en Android pensada especialmente para ser utilizada en tablet, sin embargo, para su correcto funcionamiento es necesario que posea detrás una estructura que garantice el mismo.

El reto principal, además de incluir todos los módulos planteados para la aplicación, ha sido permitir que esta pudiera consumir un servicio REST y que este accediera a la base de datos. Esto es, hacer que coexistan los módulos de programación para móviles, servicios y acceso a datos.

El proyecto consta de cinco partes diferenciadas:

- La aplicación. Por la naturaleza de FelizMente, se han tenido en cuenta criterios de diseño orientados a la accesibilidad, creando botones para hacer más fácil e intuitivo su uso para personas que no están acostumbradas a las nuevas tecnologías y/o presentan dificultades de visión.
- Un servicio REST que sirve como conexión entre las aplicaciones y la base de datos.
- El Admin está pensado para dar asistencia a aquellas personas que por cualquier casuística no se puedan registrar en la app. Se podrá realizar la gestión de altas, así como borrar y modificar usuarios.
- La base de datos está creada en MySQL y recibe los datos que se guardarán desde la app a través del servicio REST.
- Por último, una *landing page* empleada como carta de presentación para potenciales usuarios.

## **2. Módulos formativos utilizados**

Se ha tenido presente en todos los pasos del desarrollo del proyecto la necesidad de cubrir la mayor parte de asignaturas del grado. Se mencionan todas aquellos cuyos conocimientos han sido utilizados.

### **2.1 Programación**

Como base principal del código se ha utilizado Java, en sus versiones 8 y 11.

### **2.2 Programación Multimedia y Dispositivos Móviles (PMDM)**

La aplicación se ha realizado dentro de *Android Studio*. Hemos tenido en cuenta las directrices de *Material Design* para hacer la aplicación más accesible.

### **2.3 Programación de Servicios y Procesos (PSP)**

Gracias a esta asignatura, se ha podido implementar un servicio REST y establecer diferentes *endpoints* para que, no solo la aplicación, sino también otros componentes, puedan consumir este servicio.

También se ha añadido una parte de encriptación de contraseñas que persisten en base de datos.

### **2.4 Bases de Datos**

La aplicación utiliza una base de datos sencilla para realizar consultas y persistencia.

### **2.5 Acceso a datos**

La API de persistencia de Java –JPA–, permite realizar operaciones sobre la base de datos utilizando Java directamente. Por supuesto, es necesario tener una base de conocimientos bien establecida del módulo de Base de Datos para poder trabajar con JPA.

### **2.6 Entornos de Desarrollo**

El entorno de desarrollo principal visto en el grado ha sido Eclipse. No obstante, para el proyecto se ha elegido IntelliJ Idea, pues es el utilizado en las respectivas empresas durante la realización de la Formación en Centros de Trabajo y aporta mejores ayudas a la hora de realizar código.

Además de esto se ha empleado GitHub como repositorio remoto principal . Realizando las diversas gestiones a través de la GUI integrada en ambos IDE utilizados ( IntelliJ y Android Studio) así como desde la terminal por medio de comandos.

En adición a lo anterior, se han incluido algunos test unitarios con la intención de más adelante asegurarse de que todo el código está cubierto por este tipo de tests. De esta manera se minimiza la probabilidad de que haya algún comportamiento inesperado del código.

## **2.7 Inglés Técnico**

El código, los comentarios, los commits y los dos *ReadMe* se han escrito en inglés.

## **2.8 Empresa e iniciativa emprendedora**

Este proyecto incluye diversos elementos de un plan de empresa ordinario.

## **2.9 Lenguajes de Marcas y Sistemas de Gestión de la Información**

Se ha realizado una *landing page* haciendo uso de los lenguajes HTML y CSS. También se ha empleado HTML para la personalización de las ventanas de diálogo de la aplicación, para modificar el tamaño y el color de la letra.

Asimismo, se hace uso de XML para la creación de los elementos y estructura de cada una de las *activities* o páginas de la aplicación en *Android Studio*. También se ha utilizado JSON para poder transferir información a través de servicios REST a la base de datos donde se aloja la información sobre los usuarios.

### **3. Herramientas/Lenguajes empleados**

A continuación, se enumeran todas las tecnologías utilizadas en el desarrollo del proyecto.

#### **3.1 Java**

Es el lenguaje principal sobre el que se desarrolla el proyecto. Se utiliza Java 11 en todos los componentes desarrollados fuera de Android y Java 8 para la aplicación.

En un momento inicial se barajó la posibilidad de utilizar *Flutter*, pero finalmente se desestimó en base a la curva de aprendizaje y del tiempo disponible.

#### **3.2 Android Studio**

Entorno elegido para desarrollar la aplicación en Android.

#### **3.3 IntelliJ Idea**

Entorno elegido para desarrollar la infraestructura de los servicios. En general, IntelliJ es más “inteligente” que Eclipse: mejores sugerencias, refactorizaciones y mayor número de plugins disponibles son algunas de las ventajas que posee este entorno.

También contribuye a nuestra elección el hecho de que tanto IntelliJ como Android Studio son herramientas del grupo JetBrains por lo que resulta muy cómodo ya que las interfaces son iguales y funcionan de forma tremendamente similar.

Al ser IntelliJ el entorno que utilizan los integrantes del equipo que ha realizado este proyecto, en el día a día en su trabajo, la curva de aprendizaje ha sido mucho más liviana ya que gran parte de sus características se han descubierto en el desempeño laboral. De esto es ejemplo la librería Lombok, que se menciona más abajo, que es inmensamente útil en la Programación Orientada a Objetos, ya que permite simplemente creando la clase del objeto y sus campos, poner unas anotaciones y genera automáticamente los constructores y los métodos getter y setter.

#### **3.4 Spring Framework / Spring Boot**

Una de las herramientas más utilizadas para crear servicios REST es Spring Boot que forma parte del Spring Framework. Spring ayuda al desarrollador a crear aplicaciones siendo la inyección de dependencias su característica principal.

### 3.5 XAMPP

XAMPP es el software elegido para la gestión de base de datos MySQL por su simplicidad y facilidad de uso.

### 3.6 Librerías

#### 3.6.1 Lombok

Lombok, a través de anotaciones, permite mejorar la legibilidad del código, elimina la repetitividad y supone un ahorro de tiempo al generar código en tiempo de compilación. Algunas de sus anotaciones básicas (tanto en Android como en la parte del REST) utilizadas son las siguientes:

- `@NoArgsConstructor`: genera un constructor vacío.
- `@AllArgsConstructor`: crea un constructor con todos los campos.
- `@Getter` y `@Setter` se emplean para crear los métodos *getter* y *setter* de todos los campos de un objeto.

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Admin {
    private int id;
    private String username, email, password;
}
```

*Ejemplo de las anotaciones Lombok en la clase Admin*

#### 3.6.2 Retrofit

Librería empleada en Android para poder realizar las peticiones a la API REST que se ha desarrollado. Las peticiones se realizan de forma asíncrona.

```

private void checkIfExistsAndLogin(String user, String pass) {
    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://10.0.2.2:8080/felizmente/")
        .addConverterFactory(GsonConverterFactory.create(
            new GsonBuilder().serializeNulls().create()
        ))
        .build();
    UserApiService userApiService = retrofit.create(UserApiService.class);
    User u = new User(user, pass);
    Call<User> call = userApiService.searchCoincidence(u);
    call.enqueue(new Callback<User>() {
        @Override
        public void onResponse(Call<User> call, Response<User> response) {
            if (response != null && response.body() != null) {
                if (response.isSuccessful()) {
                    User u = (User) response.body();
                    Log.d("tag: \"User is:\", u.toString());
                    Toast.makeText(context, text: "Login con éxito", Toast.LENGTH_LONG).show();
                    startActivity(new Intent(context, MainActivity.class));
                }
            } else {
                Log.d("tag: \"404\", msg: \"User not found\");
                Toast.makeText(context, text: "Login incorrecto. Revise sus credenciales.", Toast.LENGTH_LONG).show();
                emailBox.setText("");
                passBox.setText("");
                emailBox.requestFocus();
            }
        }
    });
    @Override
    public void onFailure(Call<User> call, Throwable t) {
        if(t != null)
        {
            t.printStackTrace();
        }
    }
}

```

*Uso de retrofit para las llamadas al servicio. Las llamadas se resuelven de forma asincrónica*

### 3.6.3 Glide

Librería utilizada para el módulo de preguntas y respuestas llamado “Ponte a prueba” .

En este módulo se realizan preguntas sencillas de cultura general todas ellas acompañadas de fotografías.

En un principio se consideró alojar los archivos .png dentro de la aplicación, pero finalmente se optó por Glide, ya que permite acceder a imágenes a través de una dirección web. El aspecto remoto implica que la primera vez que se accede a dicha imagen haya cierto retraso en su carga, pero una vez terminada la primera carga, se guarda la imagen en caché, lo que hace que sea mucho más eficiente.

```

Glide.with( activity: this)
    .load(quizModalArrayList.get(pos).getUrl())
    .into(imageQuestion);

```

*Código que introduce las imágenes obtenidas por URL en un ImageView*



### 3.6.4 android-youtube-player

Librería *open source* creada por Pier Francesco Soffriti . Incluye un reproductor de vídeos de YouTube con opciones de personalización para la interfaz.



*Ejemplo de un reproductor de Youtube*

## 4. Componentes del equipo y funciones

En el reparto de tareas se ha intentado que existiese un equilibrio posible entre la carga de trabajo y las motivaciones e intereses de cada integrante del equipo. A continuación, se exponen los integrantes del equipo y el trabajo realizado por cada uno de ellos.

### 4.1 Alexandra Mendoza

Creación de la aplicación en Android y diseño y desarrollo del código de las siguientes incluyendo la parte de XML y de Java: *Splash, Login, Registration, Main, Photos* y *QuizTypes*. Además ha llevado el peso de dirigir y orquestar al equipo, organizando las reuniones y asegurándose del cumplimiento de los tiempos de entrega para cada fase del proyecto.

### 4.2 Sarah Amselem:

Al aportar la idea original, ha intentado extender su visión e ideas al resto del grupo sobre la imagen que tenía de la aplicación. Ha trabajado especialmente en la aplicación de estilos, diseño y de la implementación de los módulos de Video y Music. Además ha aportado, junto a los elementos del plan de empresa, la landing page de la idea.

### 4.3 Laurence Apuya

Un trabajo enfocado totalmente en el *backend*, en la lógica del servicio REST y de la aplicación Admin, prestando especial atención en que las aplicaciones realicen adecuadamente las peticiones al servicio y de las persistencias.

A pesar de los roles y trabajos diferenciados de cada uno, cada integrante del grupo ha realizado la labor de consultor, supervisor y salvaguarda del correcto funcionamiento del proyecto. No se han tomado decisiones e implementaciones que no hayan sido consensuados por el resto del equipo.

## 5. Fases del proyecto

### 5.1 El Plan de Empresa

Dada la naturaleza no lucrativa del proyecto, se ha optado por adoptar los elementos del Plan de Empresa que se han considerado pertinentes.

#### 5.1.1 Idea de Negocio

La idea de negocio queda plasmada en el primer apartado introductorio de la memoria.

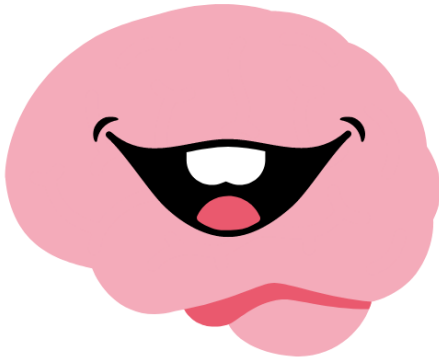
#### 5.1.2 Business Model Canvas

El Business Model Canvas es una herramienta muy útil para la gestión estratégica de cualquier proyecto. Consta de nueve piezas clave interconectadas.

Alianzas clave	Actividades clave	Propuestas de valor	Customer relationships	Segmentos de clientes
Centros de día / residencias Profesionales de la gerontología Instituciones que apoyen el emprendimiento social Campañas de publicidad Instituciones culturales	Estimulación cognitiva Mediación cultural Promoción	Información rápida y práctica Diseño accesible. Lectura fácil. Contenidos estimulantes que favorezcan la adhesión a la terapia Mediación cultural: facilitamos la cultura a los usuarios	Soporte técnico Redes Sociales Talleres y eventos Sistema de feedback, reseñas y puntuación	Mayores de todas las clases sociales que experimenten deterioro cognitivo Familiares y entorno Residencias y centros de día
	Recursos clave		Canales	
	Recursos intelectuales Desarrolladores/ Diseñadores Contenidos APP		Comunicación: WEB propia, RRSS, SEO, SEM, banners en webs afines Entrega: Apple Store, Google Play, WEB	
Estructura de costes			Fuentes de ingresos	
Recursos de formación	Marketing	Mantenimiento de la aplicación y de la web	Donaciones	Ayudas al emprendimiento social

*Business Model Canvas de FelizMente*

### 5.1.3 Identidad de marca



Para la identidad de marca se ha decidido adoptar una estética retro.

Destaca en el símbolo la influencia de Estudios Moro, creadores de personajes entrañables de la España del siglo XX como son Naranjito, Kinito o la familia Telerín.

*Símbolo de FelizMente.*

### 5.1.4 Financiación

En cuanto a la rentabilidad del producto, si bien en un principio se contempló cierto interés lucrativo, pronto consideramos que la aplicación debería ser gratuita, dado el interés social de la misma. Además, los contenidos se extraen de YouTube. El equipo defiende que internet es de todos.

En un futuro se hablaría con instituciones públicas como RTVE (Radio Televisión Española) o Canal Sur, para acceder a sus fondos. También se podría contactar con instituciones privadas del entretenimiento para que cedan sus contenidos de manera solidaria.

Para financiar el mantenimiento de la aplicación se recurrirá a la comercialización de productos digitales accesibles, así como a campañas de recaudación y organización de eventos.

### 5.1.5 Estudio de mercado

El mercado a intervenir es el mercado de las aplicaciones para la estimulación cognitiva, que se encuentra en constante crecimiento. Hasta que desde la medicina se descubra una manera de frenar el deterioro cognitivo por completo, el mercado seguirá expandiéndose, por suerte o por desgracia.

A pesar de que la oferta por parte de entidades de prestigio y/o con una base científica sólida va en aumento, se confía en que FelizMente puede hacerse un hueco.

Hasta ahora ninguna iniciativa en este mercado le ha dado tanta importancia a la mediación cultural.

### Análisis DAFO

El análisis DAFO es una matriz que se emplea para obtener una visión general de la situación actual de una empresa, un proyecto o incluso un individuo.

Consta de las siguientes partes:

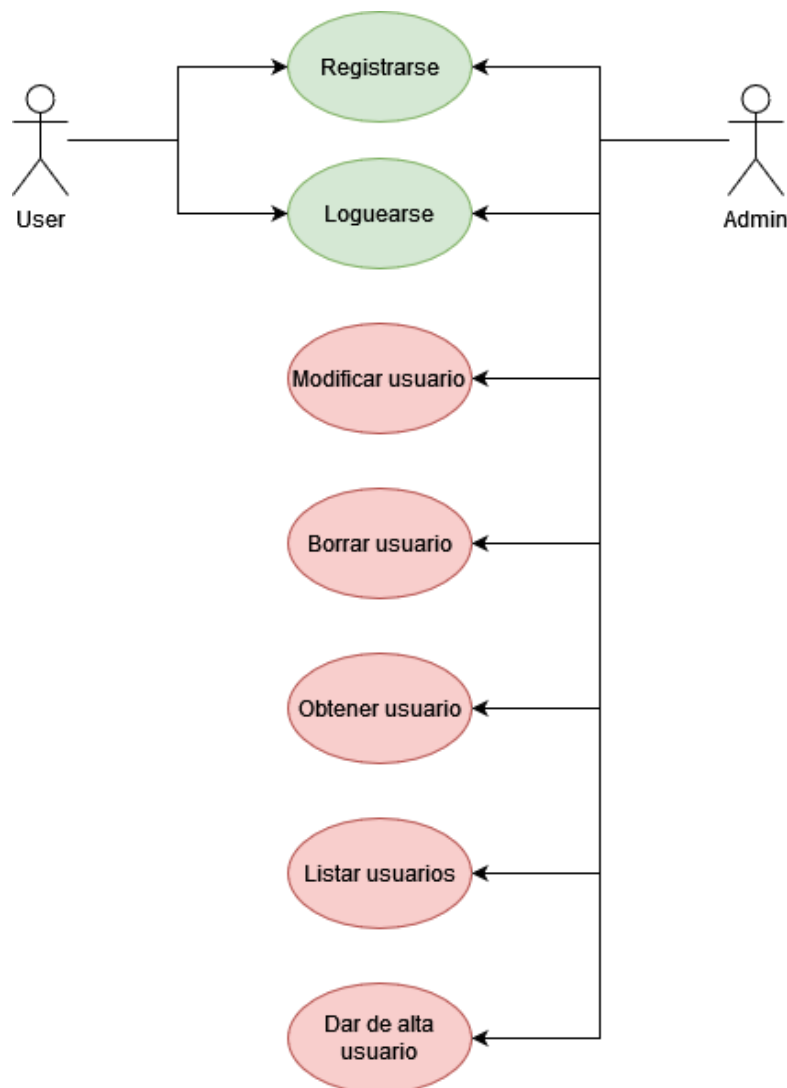
1. Debilidades: factor negativo e interno. Puntos débiles y aspectos desfavorables.
2. Amenazas: factor negativo y externo. Situaciones del entorno que puedan constituir un peligro para el proyecto.
3. Fortalezas: factor positivo e interno. Ventaja competitiva.
4. Oportunidades: factor positivo y externo. Circunstancias externas favorables.



## 5.2 Casos de uso

Los casos de uso, de acuerdo a IBM, “es un artefacto que define una secuencia de acciones que da lugar a un resultado de valor observable. Los casos de uso proporcionan una estructura para expresar requisitos funcionales en el contexto de procesos empresariales y de sistema. Los casos de uso pueden representarse como un elemento gráfico en un diagrama y como una especificación de caso de uso en un documento textual”.

Los casos de uso que se exponen a continuación muestran cuales son los requisitos funcionales y los actores principales del proyecto de FelizMente: un usuario que podrá registrarse e iniciar sesión y un ADMIN que, aparte de las anteriores operaciones, realiza las operaciones de CRUD.

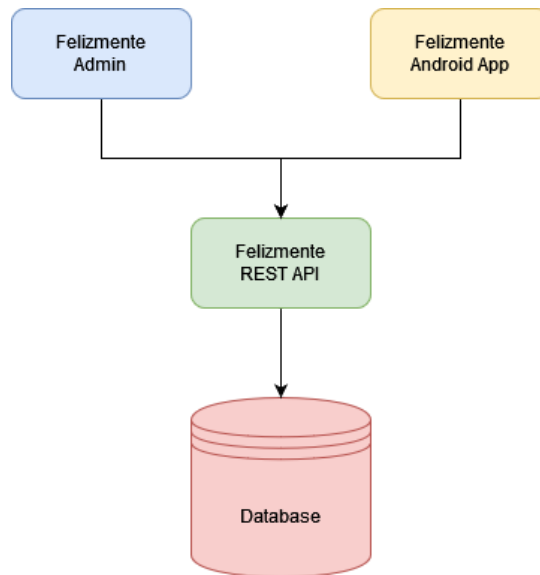


Casos de uso de FelizMente

### 5.3 La arquitectura del proyecto

A la hora de desarrollar una aplicación, es importante tener una visión global de todos los componentes y servicios.

Para la app de FelizMente, se ha seguido la siguiente arquitectura:

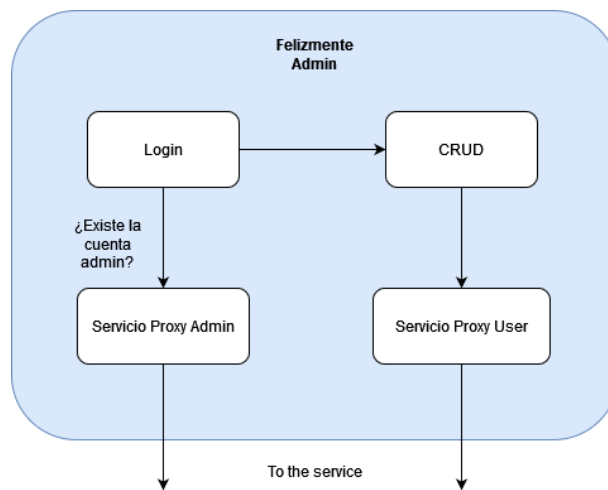


*Arquitectura a gran escala del proyecto*

Es decir, se han creado dos componentes Admin y Android App que consumirá los servicios REST, que es el encargado de realizar las operaciones oportunas a la base de datos. A continuación, se presentarán las arquitecturas de los distintos módulos, vistos desde más cerca.

#### 5.3.1 Arquitectura de Admin

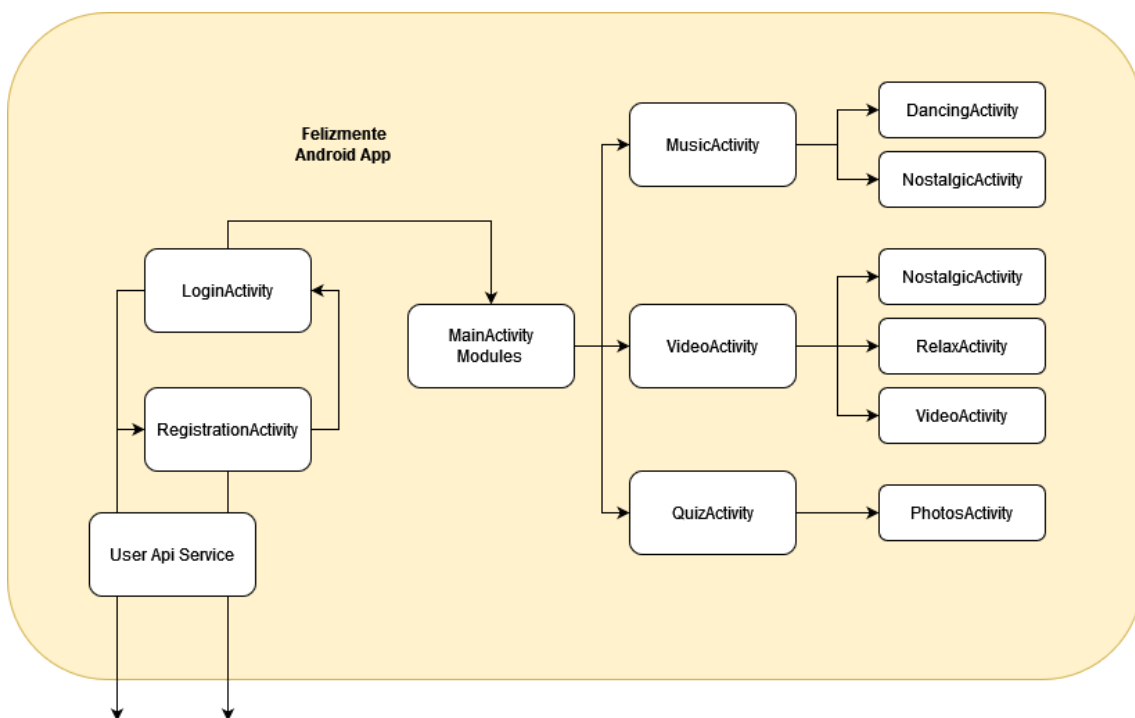
El inicio de sesión llamará a un *proxy* del servicio que consultará si existe o no una cuenta de administrador. En caso afirmativo, procederá a la siguiente pantalla donde podrá realizar las operaciones de CRUD. Este módulo será el encargado de recibir las peticiones y hacer las operaciones en la base de datos.



*Arquitectura de la aplicación Admin*

### 5.3.2 Arquitectura de la aplicación Android

La arquitectura de la aplicación Android, algo más extensa, es la siguiente:



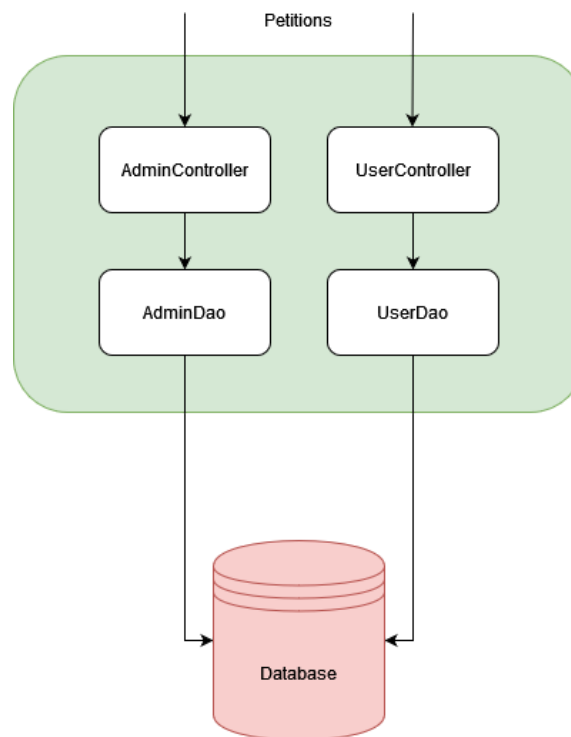
*Arquitectura de la aplicación Android*



Como se puede observar, existirá un *login* que también permitirá registrarse. Ambas *activities* comunicarán con un “*User API Service*” que será el encargado de realizar las peticiones al servicio REST. Después, se presentará una pantalla principal donde se le hará elegir al usuario entre los tres módulos existentes: música, vídeos o un pequeño juego de preguntas y respuestas.

### 5.3.3 Arquitectura del servicio REST

Finalmente, la arquitectura del servicio REST:



*Arquitectura del servicio*

Este módulo recibe las peticiones que pueden llegar al controlador del administrador o del usuario, dependiendo del *endpoint* al que se acceda. Estos controladores llamarán a los respectivos DAO que serán los que finalmente realizarán las operaciones a la base de datos.

## 5.4 La aplicación Android

### 5.4.1 El diseño

#### 5.4.1.1 Accesibilidad

La accesibilidad es el aspecto primordial de la aplicación, ya que el público objetivo son personas mayores. Se ha priorizado la accesibilidad cognitiva a través de un diseño sencillo, con el mínimo número de elementos posibles e instrucciones sencillas para que la experiencia del usuario resulte fácil e intuitiva. Por esta misma razón, se han tenido en cuenta los siguientes aspectos clave:

- Se han utilizado tanto elementos de tipo “*CardView*” con imágenes de estilo retro, como botones grandes para cambiar de pantalla y ayudar a usuarios con escasa experiencia en las nuevas tecnologías a desenvolverse dentro de la aplicación.
- La combinación de los colores elegidos, se ha realizado teniendo en cuenta la calificación AAA, por la cual se establece el mayor nivel de contraste entre colores, lo cual facilita la legibilidad de los textos en pantallas.



Paleta de colores elegidos y muestra de letra Roboto Regular y los tamaños implementados.

- La letra empleada para los textos de toda la aplicación ha sido Roboto Regular, una única tipografía sencilla, moderna y accesible. Además los tamaños de letra también han sido establecidos cuidadosamente para asegurarse de que puedan ser leídos con facilidad:
  - Títulos principales: 48sp
  - Subtítulos: 32 sp
  - Texto en las tarjetas o “card views”: 28sp
  - Resto de texto: 24sp

Se ha tratado de seguir de la manera más precisa posible las directrices de dos instituciones clave en el tema, como son W3C y Material Design.

#### 5.4.1.2 La tablet como soporte ideal

La tablet es el dispositivo preferido entre la tercera edad, pues ofrece la portabilidad y el aspecto táctil del *smartphone* con una resolución de pantalla mayor.

Para asegurarnos de que la aplicación solo sea utilizable en dispositivos tablet, se ha añadido el siguiente código al *Android Manifest*, lo cual establece que los dispositivos que se admiten son aquellos de pantalla grande o muy grande y cuyo lado más pequeño tenga como mínimo 600 dp (densidad de píxeles). Esto limita la descarga de la aplicación en Google Store para que esté disponible únicamente para aquellos dispositivos cuyo tamaño de pantalla cumpla con las condiciones aquí expuestas:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.felizmente">

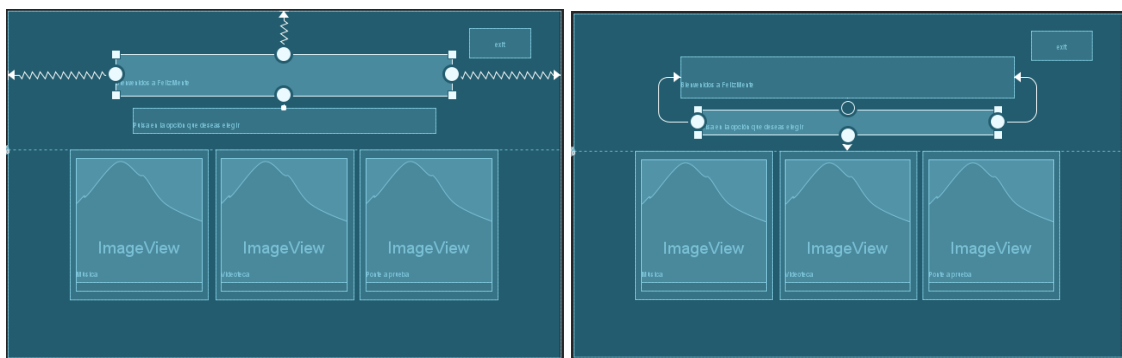
    <supports-screens
        android:smallScreens="false"
        android:normalScreens="false"
        android:largeScreens="true"
        android:xlargeScreens="true"
        android:requiresSmallestWidthDp="600"
    />
</manifest>
```

*Restricción de dispositivos en los que la aplicación puede ser descargada desde Google Play Store*

Además, se ha decidido mantener la orientación horizontal para mayor comodidad y aprovechamiento de la pantalla teniendo en cuenta el reproductor de vídeo, aunque también para evitar confusión y facilitar el manejo de la aplicación por personas de avanzada edad.

#### 5.4.1.3 ConstraintLayout

Desde un principio se decidió que la aplicación se desarrollaría con el tipo de diseño Constraint Layout, el cual funciona a base de “*constraints*” o restricciones y se adapta fácilmente a diferentes tamaños de pantalla. Cada uno de los elementos que se incluyan en el archivo XML para incluirlo en el diseño, estará anclado a otro elemento al menos en dos puntos, siempre siendo al menos uno de ellos vertical y otro horizontal.



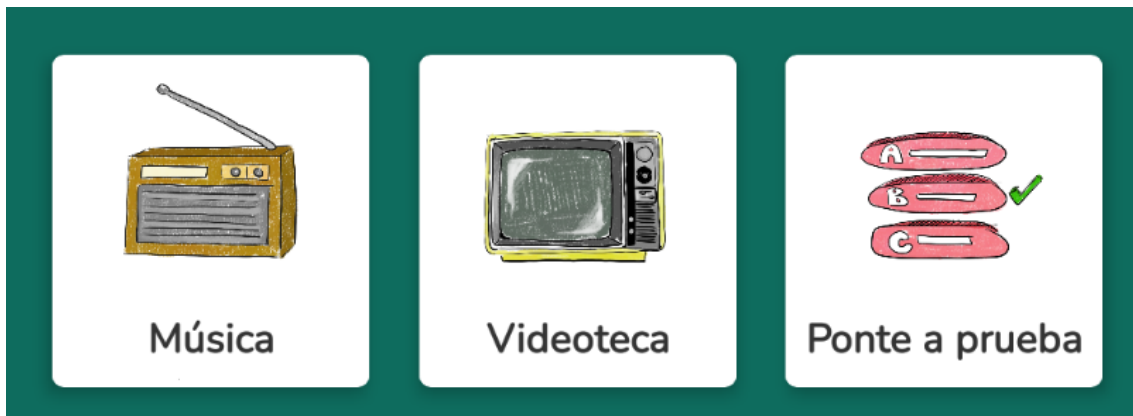
*Ejemplos de las restricciones verticales y horizontales establecidas para el Constraint Layout*

Como elemento de ayuda para cuadrar correctamente las posiciones de los componentes se ha utilizado el elemento *guideline* horizontal, situándose al 40% de la pantalla del eje vertical, para que así la distribución de los elementos sea proporcionalmente similar en los diferentes tamaños de pantalla. Este elemento de ayuda es sólo visible en la fase de diseño, siendo invisible en el dispositivo desde el que se utilice la aplicación.

## 5.4.2 Otras funcionalidades

### 5.4.2.1 CardView:

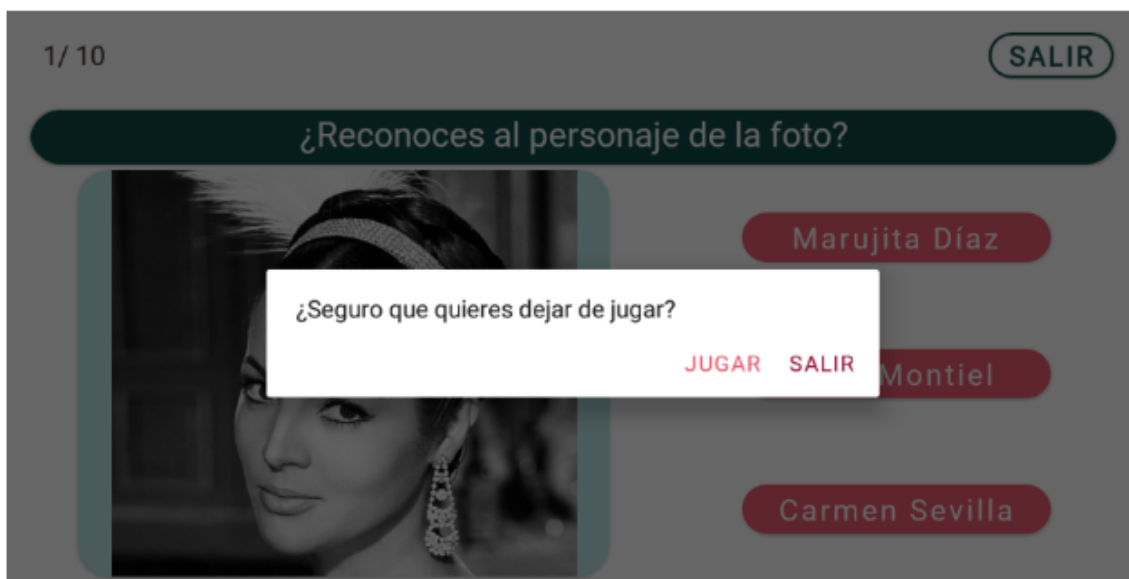
Tipo de vistas utilizadas para el acceso a cada uno de los módulos y sus correspondientes categorías. El funcionamiento es como el de un botón grande para facilitar la accesibilidad y se han diseñado ilustraciones con imágenes representativas de estilo retro. Una vez se haya pulsado en cualquier punto de la CardView, se pasa directamente a la página de la categoría seleccionada.



*Cada uno de estos “botones” se ha realizado con un CardView.*

### 5.4.2.2 Ventanas de diálogo:

Se han creado diversas ventanas de diálogo para ciertas funcionalidades de la aplicación: para salir del juego y para confirmar la salida de la aplicación.



*Ejemplo de ventana de diálogo utilizada.*

Debido a que el tamaño de la letra era pequeño y se quería aplicar también color para diferenciar las opciones (en el caso de la imagen, “JUGAR” o “SALIR”), se recurrió a la utilización del objeto Html y su método “.fromHtml” para modificar el tamaño de la letra y modificar los colores mediante etiquetas propias de este tipo de lenguaje de marcas:

```
public void exitQuiz(View v){
    Intent intent = new Intent( packageContext: this, MainActivity.class);

    AlertDialog dialog = new AlertDialog.Builder( context: this)
        .setMessage(Html.fromHtml( source: "<big>¿Seguro que quieres dejar de jugar?</big>",
            Html.FROM_HTML_MODE_LEGACY))
        .setPositiveButton(Html.fromHtml( source: "<big><font color='#99173C'>Salir</font></big>",
            Html.FROM_HTML_MODE_LEGACY) , (dialogInterface, i) -> startActivity(intent))
        .setNegativeButton(Html.fromHtml( source: "<big><font color='#EA596E'>Jugar </font></big>",
            Html.FROM_HTML_MODE_LEGACY) , listener: null)
        .create();
    dialog.show();
}
```

*Ejemplo de etiquetas propias del lenguaje HTML para mejorar el diseño de las ventanas de diálogo.*

#### 5.4.2.3 Funcionalidades especiales

Se comprobó que al dar al botón de atrás en el dispositivo Android, el comportamiento de la aplicación por defecto, hacía que se volviese siempre a la pantalla anterior, lo cual podría resultar confuso. Esto se observó particularmente en el caso de estar en el juego de preguntas y respuestas, al poder salir del juego mediante el botón de “salir” esto llevaba a la página principal, una vez aquí sí se daba al botón de atrás se volvía al juego, comportamiento que se quería evitar.

Se decidió controlar esta casuística añadiendo a todas las pantallas intermedias una nueva funcionalidad que hiciera que el botón atrás de Android, llevase a la pantalla inmediatamente anterior. De esta manera, en el ejemplo comentado anteriormente, ya no se volvería a la pantalla del juego, es más en el caso en concreto de la pantalla principal, se abrirá una ventana de diálogo para confirmar si se quiere cerrar la aplicación. Sin embargo, si se accediese al módulo de música y posteriormente a ambas categorías musicales, una vez de vuelta en música, el botón de atrás de Android llevará a la página principal y no a la última visitada.

## **El reproductor de vídeo.**

En principio se había recurrido a la API oficial de YouTube. No obstante, tras un poco de investigación se halló una alternativa interesante: la librería de Pier Francesco Soffriti, disponible en GitHub.

Este recurso facilita una *View* sencilla que puede integrarse fácilmente en cada *Activity/Fragment*.

La librería es un envoltorio de la *IFrame Player API* que corre dentro de una *WebView*, por lo tanto no se requiere la aplicación de YouTube en el dispositivo y no se infringen los términos de servicio.

La YouTube Android Player API lleva años sin actualizarse y presenta numerosos *bugs*. además es poco flexible en cuanto a personalización. En este caso la alternativa ha resultado ser una opción más fiable.

Si bien en la API proporcionada por Google había métodos específicos para la integración con playlists alojadas en YouTube, el reproductor de Soffritti funciona de otra manera: las listas de reproducción son creadas en una clase llamada *VideoIdsProvider*, pasándole un array con los identificadores de los vídeos que se mostrarán en cada canal.

Se puede encontrar el *id* de cada vídeo en la URL del mismo, en los caracteres tras “watch?v=”.

A continuación, se implementa el método para reproducir el siguiente vídeo en orden aleatorio tras pulsar el botón correspondiente.

## 5.5 Servicio REST

Esta parte mezcla las asignaturas de Programación de Servicios y Procesos y Acceso a Datos, ya que combina un servicio que es el encargado además de persistir usando JPA. Este servicio al crearse comprueba si ya existe la tabla de datos de administrador y, en caso negativo, la crea con tres administradores por defecto. Además también crea la tabla de usuarios.

Dos han sido los retos principales de este módulo: por un lado el crear las *queries* necesarias y deseadas y además persistir las contraseñas usando la encriptación.

```
String username = user.getUsername();

User uAux = em.createQuery( q1String: "select user from User user where user.email = :username", User.class)
    .setParameter( name: "username", username)
    .getSingleResult();

try {
    if (uAux != null) {
        encryptor.init(Cipher.DECRYPT_MODE, scytale);
        byte [] passwordBytes = encryptor.doFinal(Base64.getDecoder().decode(uAux.getPassword()));
        String passwordString = new String (passwordBytes);
        if (user.getPassword().equals(passwordString)) {
            return uAux;
        }
    }
}
```

*Busca al usuario y en caso de encontrarse, descripta la contraseña. Si coincide, se considera al usuario registrado.*

El servicio cuenta con dos clases de tipo Controller para usuarios y admins, que se encargan de recibir las peticiones por los distintos *endpoints* definidos para finalmente mandarlos a sus respectivos DAO. Además, contiene los objetos de encriptación que serán los encargados de encriptar las contraseñas a la hora de la persistencia, como de descriptar a la hora de buscar coincidencias en las contraseñas para el inicio de sesión. La clase Main es la encargada de crearlos y la que se los entrega a los DAO para que sean utilizados por estos.



```

context = SpringApplication.run(FelizMenteApplication.class, args);
Databasedao dataBaseDao = context.getBean( name: "databasedao", Databasedao.class);
AdminDao adminDao = context.getBean( name: "adminDao", AdminDao.class);
UserDao userDao = context.getBean( name: "userDao", UserDao.class);

try {
    KeyGenerator generator = KeyGenerator.getInstance("AES");
    SecretKey scytale = generator.generateKey();
    Cipher encryptor = Cipher.getInstance("AES");
    encryptor.init(Cipher.ENCRYPT_MODE, scytale);
    adminDao.setEncryptor(encryptor);
    adminDao.setScytale(scytale);
    userDao.setEncryptor(encryptor);
    userDao.setScytale(scytale);
    if (!dataBaseDao.alreadyWithAdmins()) {
        dataBaseDao.initDataBase(encryptor);
    }
} catch (Exception e) {
    e.printStackTrace();
}

```

*Código fuente que crea los objetos de encriptación y se envía a los respectivos DAO*

Cabe mencionar una cosa muy importante: los objetos que se encargan de la encriptación se generan al levantar el servicio. Si el servicio se apagase y se volviese a levantar, se generarían otros objetos de encriptación totalmente diferentes y la tabla quedaría invalidada al haber sido generada las contraseñas con la encriptación anterior. Se es consciente de esto: en un entorno real, el servicio estaría alojado en un servidor o en la nube donde estos objetos de encriptación no se pierden. Igualmente, es necesario guardar una copia de seguridad de estos objetos de encriptación o profundizar más en el tema y elegir una implementación adecuada.

## 5.6 La aplicación Admin

La motivación de añadir esta aplicación es la de que de forma externa se puedan gestionar los usuarios. Si alguien tuviese problemas con el inicio de sesión o cualquier otro tipo de problema, es conveniente tener una aplicación que tenga las operaciones de CRUD:

- Dar de alta un usuario
- Dar de baja un usuario
- Buscar un usuario
- Eliminar un usuario
- Listar todos los usuarios

Contará con dos clases Proxy: una para ADMIN y otra para los usuarios. El servicio *proxy* de los administradores se utiliza única y exclusivamente para el inicio de sesión y autenticación del administrador. La otra clase *proxy* será la encargada de realizar las peticiones que el administrador desee a nuestro servicio REST.

```
@Service
public class AdminProxyService {
    public static final String URL = "http://localhost:8080/felizmente/admins/";

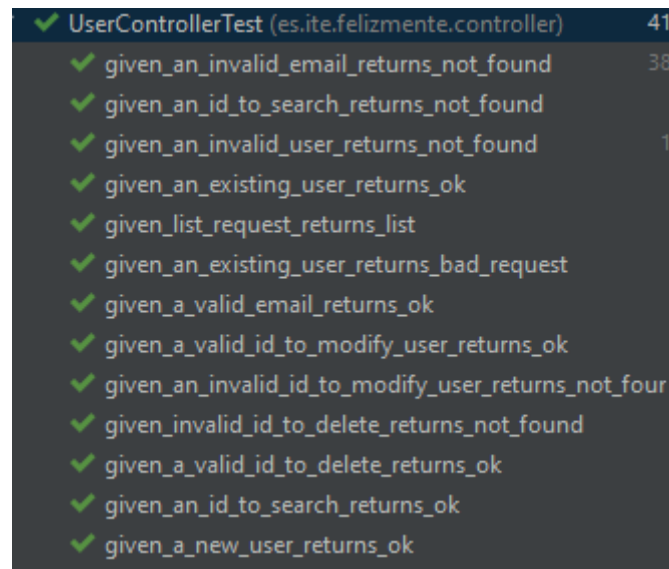
    @Autowired
    private RestTemplate restTemplate;

    public Admin get(Admin a) {
        try {
            String completeURL = URL+"login";
            ResponseEntity<Admin> re = restTemplate.postForEntity(completeURL, a, Admin.class);
            HttpStatus hs = re.getStatusCode();
            if (hs == HttpStatus.OK) {
                return re.getBody();
            } else {
                System.out.println("Not admitted");
                return null;
            }
        } catch (HttpClientErrorException e) {
            System.out.println("get -> User not found: " + a.toString());
            System.out.println("get -> Response code: " + e.getStatusCode());
            return null;
        }
    }
}
```

*Ejemplo de servicio proxy de la clase Admin*

## 5.7 Testing

Se han realizado test unitarios para ciertas partes del código. Si bien se ha aprendido y realizado tests unitarios en el grado, el proyecto requería de tests unitarios para un servicio, convirtiéndolo en algo más complejo.



✓ UserControllerTest (es.ite.felizmente.controller)	417
✓ given_an_invalid_email_returns_not_found	386
✓ given_an_id_to_search_returns_not_found	1
✓ given_an_invalid_user_returns_not_found	16
✓ given_an_existing_user_returns_ok	1
✓ given_list_request_returns_list	1
✓ given_an_existing_user_returns_bad_request	2
✓ given_a_valid_email_returns_ok	1
✓ given_a_valid_id_to_modify_user_returns_ok	3
✓ given_an_invalid_id_to_modify_user_returns_not_found	1
✓ given_invalid_id_to_delete_returns_not_found	1
✓ given_a_valid_id_to_delete_returns_ok	1
✓ given_an_id_to_search_returns_ok	1
✓ given_a_new_user_returns_ok	2

*Tests unitarios realizados para el controlador de user*

Por ello, no se han realizado tests de todo el código si no que se ha realizado en las partes que podía ser más sencillo.

La idea principal es que hay que *mockear* aquellos objetos que la clase necesita y luego inyectárselas. Los objetos que son *mockeados* no necesitan inicializarse ya que el programa se encarga de darles un valor totalmente aleatorio. Lo que si que hay que definir es el comportamiento de estos objetos en tiempo de ejecución del test y “decirle” qué tiene que hacer cuando se encuentre el método del objeto y qué espera retornar.

```

@RunWith(MockitoJUnitRunner.class)
public class UserControllerTest {
    @InjectMocks
    private UserController userController;
    @Mock
    private UserDao userDao;

    @BeforeEach
    public void setUp() { MockitoAnnotations.openMocks( testClass: this); }

    @Test
    public void given_a_valid_email_returns_ok() {
        when(userDao.search(any())).thenReturn(User.builder().build());

        var response : ResponseEntity<User> = userController.getUser(anyString());
        assertNotNull(response);
        verify(userDao, times( wantedNumberOfInvocations: 1)).search(any());
    }
}

```

*Ejemplo de test unitario usando mockito*

## 5.8 La página web

Por último, se ha maquetado una landing page sencilla, empleando HTML y CSS.

Desde ella se puede descargar la APK de la aplicación, así como abrir el proveedor de correo electrónico por defecto para enviar un mensaje al equipo de FelizMente.

## 6. Conclusiones

Al tener uno de los integrantes la idea principal, es tentador pensar que desarrollar un proyecto de este calibre pueda resultar fácil. Nada más lejos de la realidad.

Si bien es cierto que la idea es clara, surgen las dudas de su implementación y desarrollo. Algunas preguntas han sido: ¿Cómo de accesible ha de ser la aplicación? ¿Sólo en tablet o también para móviles? ¿Qué información recoger en la base de datos? ¿Merece la pena el esfuerzo de conectarlo a un servicio o es demasiado complicado y se deja directamente conectado a la base de datos? ¿Cuántos módulos implementar con el tiempo del que se dispone? Estas y muchas otras preguntas han rondado, no solo al principio, sino durante todas las fases del proyecto. Gracias al trabajo en equipo muchos (sino todos) los objetivos se han conseguido. Por supuesto, siempre existe margen para la mejora, sin embargo la sensación general es la de haber realizado un buen trabajo dentro del tiempo disponible.

Tras un proceso de debate se marcaron tres objetivos principales:

1. Desarrollar la aplicación con  $N$  módulos, en función del estado del proyecto.
2. Accesibilidad de la aplicación.
3. Crear un servicio sobre el que la app pudiera realizar peticiones HTTP.

Haciendo retrospectiva, se puede asegurar que se han cumplido los objetivos de forma satisfactoria. Se ha tenido claro desde el primer momento que la implementación de  $N$  módulos iba a depender del tiempo disponible. Finalmente se han implementado tres módulos: uno con dos submódulos y los otros dos, con dos submódulos.

En cuanto a la accesibilidad de la aplicación, posee *píldoras* de accesibilidad suficientes para empezar, pero que en el futuro se pueda seguir mejorando.

Finalmente, se ha podido realizar la conexión de la app al servicio. Las conexiones, si bien son sencillas, es todo un logro para el equipo dado que se requería mucha investigación. Poder conectar las asignaturas de PMDM, PSP y AD es más que motivo suficiente de satisfacción. Además, ha habido tiempo para poder implementar la aplicación de complemento Admin que sirva para gestionar de forma externa a los usuarios de la aplicación.

Cabe realizar una mención especial al uso de Git. El uso de repositorios puede ser tedioso y puede dar algo de miedo su uso, pero todos los integrantes del equipo han demostrado saber manejarse con esta tecnología. No conforme con eso, todos han entendido la gran importancia que esta puede resultar a la hora de desarrollar proyectos en el futuro.

Existen puntos en el proyecto que podrían mejorarse. Algunos de estos puntos son:

- Añadir aspectos de accesibilidad para la población no lectora (“*Text to Speech*” e imágenes más explicativas).
- Añadir más categorías y fomentar el aspecto modular de manera que el cuidador pueda activar o desactivar módulos en función de los intereses del usuario.
- Ahondar en el estudio de la cultura popular española para mejorar la calidad de los contenidos.
- Programar sincronización con pulsadores para usuarios con discapacidad motriz, ya sea por parálisis o enfermedad de Parkinson.
- Incluir recursos de “Lectura Fácil”.
- Diseño e implementación de más actividades relacionadas con la vida cotidiana
- Refinar el diseño de la app, consiguiendo una UI elegante y accesible
- Mejorar y refactorizar el código
- Mejorar la encriptación
- Ampliar la base de datos
- Crear un servicio de orquestamiento de peticiones con hilos
- Aumentar la cobertura con tests unitarios

Esta lista podría seguir porque, como ya se ha mencionado antes, siempre hay margen para sacar una mejor versión. Como equipo creemos firmemente que, en general, se han cumplido la mayoría de los objetivos propuestos y se está satisfecho con el trabajo realizado.

## 7. Bibliografía

<https://www.w3schools.com/>

<https://www.baeldung.com/java-password-hashing>

<https://java2blog.com/validate-password-java/>

<https://pierfrancesco-soffritti.medium.com/how-to-play-youtube-videos-in-your-android-app-c40427215230>

<https://square.github.io/retrofit/>

<https://programacionymas.com/blog/consumir-una-api-usando-retrofit>

<https://medium.com/contraslashsas/retrofit-para-android-desde-0-1c8be830a1af>

<https://engineering.creativesociety.mx/retrofit-cliente-http-para-android/>

<https://www.sen.es/saladeprensa/pdf/Link280.pdf>

<https://www.redaccionmedica.com/secciones/neurologia/en-espana-hay-800-000-personas-con-alzheimer-y-seran-el-doble-en-20-anos-3170>

<https://developer.android.com/guide/topics/ui/look-and-feel?hl=es-419>

<https://stackoverflow.com/questions/10540646/designing-an-android-tablet-only-app>

<https://www.infogerontologia.com/>

<https://www.ibm.com/docs/es/elm/6.0.3?topic=requirements-defining-use-cases>

## Anexos

### Manual de usuario del proyecto

La aplicación está alojada en el siguiente repositorio:

<https://github.com/lapuya/FelizMente>

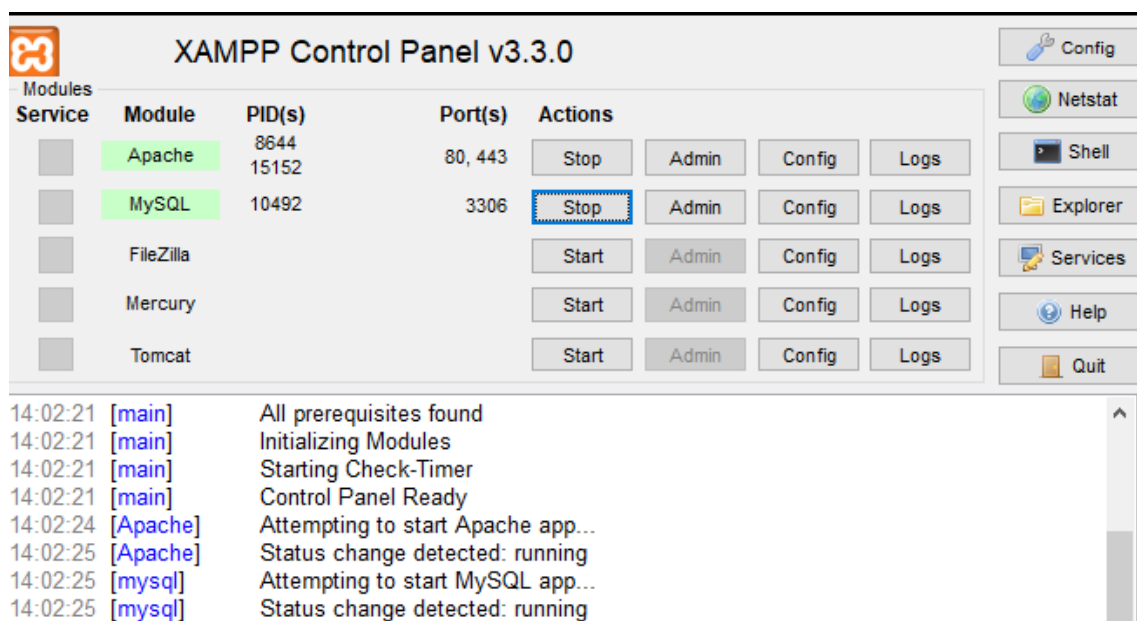
A continuación se muestran las pautas para su correcto funcionamiento.

#### Pre-requisitos:

- XAMPP
- IDE con JAVA 11
- Android Studio

#### Pasos:

1. Descargarse todos los repositorios en: (url)
2. Encender XAMPP



*XAMPP encendido*

3. Pulsar el botón “Admin” de la parte MySQL. Se abrirá una página correspondiente al *localhost*. Darle al botón “nueva” para crear una nueva base de datos y llamarlo “felizmente”. Esta no tendrá ninguna tabla.



4. Levantar el servicio REST del proyecto “FelizMente\_Rest”. Esto generará dos tablas sencillas: admin y users, con ADMIN teniendo valores cargados por defecto.

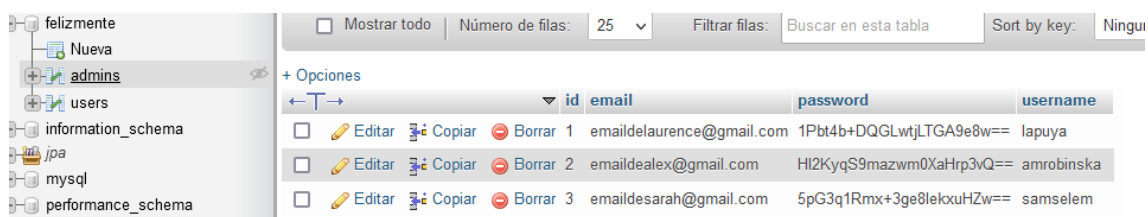
```

...
:: Spring Boot :: (v2.0.0)

2022-05-20 14:03:28.307 INFO 21172 --- [main] es.ite.felizmente.FelizMenteApplication : Starting FelizMenteApplication using Java 11 on DESKTOP-152HT1S with PID 21172 (F:\eclipse\workspace\FelizMente_Rest\target
2022-05-20 14:03:28.308 INFO 21172 --- [main] es.ite.felizmente.FelizMenteApplication : No active profile set, falling back to 1 default profile: "default"
2022-05-20 14:03:28.707 INFO 21172 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-05-20 14:03:28.711 INFO 21172 --- [main] org.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-05-20 14:03:28.711 INFO 21172 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.40]
2022-05-20 14:03:28.771 INFO 21172 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-05-20 14:03:28.772 INFO 21172 --- [main] o.s.c.s.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 439 ms
2022-05-20 14:03:28.928 INFO 21172 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-05-20 14:03:28.934 INFO 21172 --- [main] es.ite.felizmente.FelizMenteApplication : Started FelizMenteApplication in 0.809 seconds (JVM running for 1.228)
2022-05-20 14:03:29.011 INFO 21172 --- [main] org.hibernate.jpa.internal.util.LogHelper : HHN000004: Processing PersistenceUnitInfo [name: felizmente]
2022-05-20 14:03:29.039 INFO 21172 --- [main] org.hibernate.Version : HHN0000412: Hibernate ORM core version 5.4.21.Final
2022-05-20 14:03:29.103 INFO 21172 --- [main] org.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
2022-05-20 14:03:29.164 WARN 21172 --- [main] org.hibernate.orm.connections.pooling : HHN10001002: Using Hibernate built-in connection pool (not for production use!)
2022-05-20 14:03:29.166 INFO 21172 --- [main] org.hibernate.orm.connections.pooling : HHN10001005: using driver [com.mysql.cj.jdbc.Driver] at URL [jdbc:mysql://localhost:3306/felizmente]
2022-05-20 14:03:29.166 INFO 21172 --- [main] org.hibernate.orm.connections.pooling : HHN10001001: Connection properties: (user=root)
2022-05-20 14:03:29.166 INFO 21172 --- [main] org.hibernate.orm.connections.pooling : HHN10001003: AutoCommit mode: false
2022-05-20 14:03:29.167 INFO 21172 --- [main] org.hibernate.orm.connections.pooling : HHN0000115: Hibernate connection pool size: 20 (min=1)
2022-05-20 14:03:29.226 INFO 21172 --- [main] org.hibernate.dialect.Dialect : HHN0000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
2022-05-20 14:03:29.488 INFO 21172 --- [main] org.hibernate.orm.connections.access : HHN10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator
2022-05-20 14:03:29.518 INFO 21172 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHN0000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2022-05-20 14:03:29.585 INFO 21172 --- [main] org.hibernate.orm.connections.pooling : HHN10001008: Cleaning up connection pool [jdbc:mysql://localhost:3306/felizmente]

```

*Servicio REST en ejecución*



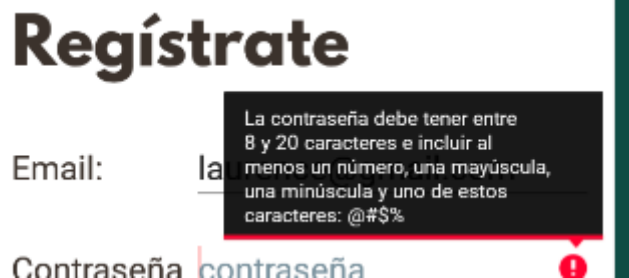
	id	email	password	username
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	1	emaildelaurence@gmail.com	1Pbt4b+DQGLwtjLTGA9e8w==	lapuya
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	2	emaildealex@gmail.com	Hi2KyqS9mazwm0XaHrp3vQ==	amrobinska
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	3	emaildesarah@gmail.com	5pG3q1Rmx+3ge8l6xuxHZw==	samsalem

*Base de datos con valores admin predeterminados*

5. Ejecutar la aplicación Android “FelizMente”. Después del Splash, se dará la opción de iniciar sesión o registrarte

#### a. Registro

- i. No se permite ninguno de los campos vacíos.
- ii. La contraseña debe tener entre 8 y 20 caracteres, tener un número, una mayúscula y un símbolo especial.



*Restricción de contraseña en la app*

iii. Tras registrarse exitosamente, se le llevará a la pantalla para iniciar sesión.

b. Inicio de sesión

i. En caso de que el usuario exista en la base de datos, se habrá iniciado la sesión con éxito e irá a la ventana principal.



*Menú principal*

6. La aplicación posee 3 módulos principales: música, vídeos y acertijos.

a. Música, que consta de dos submódulos: canciones enfocadas a la nostalgia y canciones para bailar. Elegir uno de estos canales te llevará al reproductor, donde tendrá dos botones: salir y siguiente vídeo.



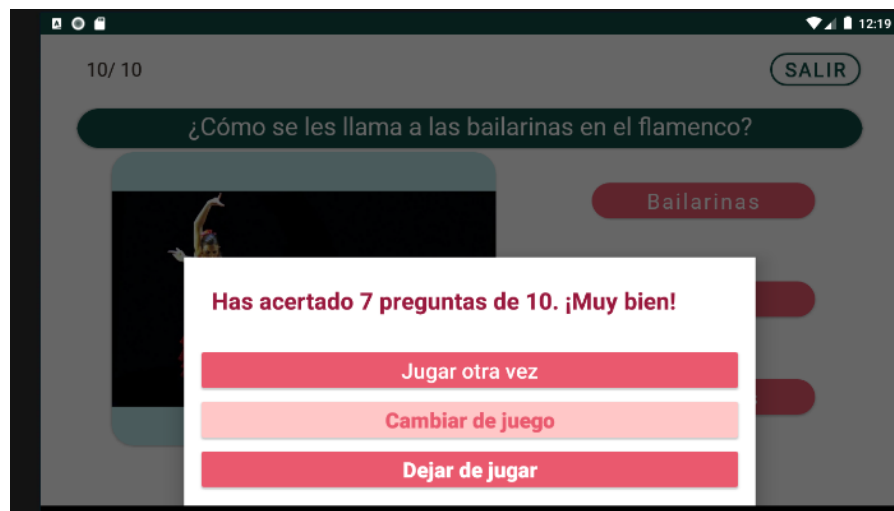
*Ejemplo de un canal*

- b. Vídeos, que sigue la misma estructura que música pero con tres submódulos: relajación, recuerdos y humor.



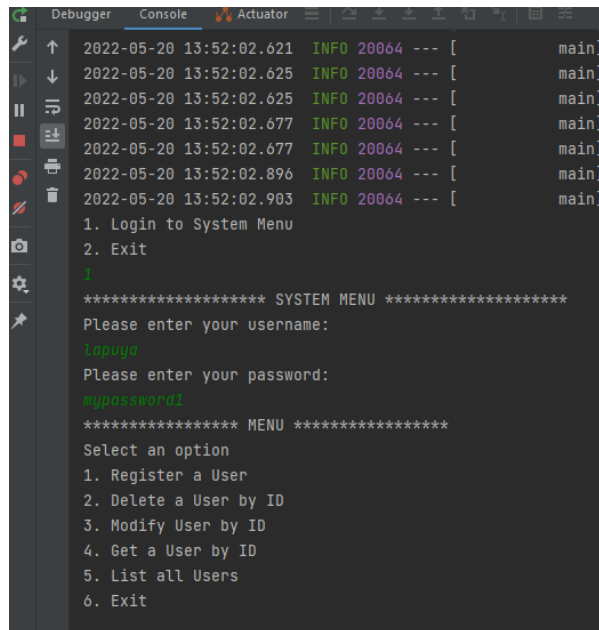
*Menú de videoteca*

- c. Preguntas tipo test, con dos módulos: cultura española y personajes. Al final de la pantalla se mostrará cuántas se han acertado y si se quiere volver a jugar o salir.



*Ejemplo de quiz*

7. Ejecutar la aplicación Admin del proyecto “FelizMente\_Admin”
  - a. Se le pedirá que inicie sesión y a continuación se le mostrará la pantalla CRUD.



```

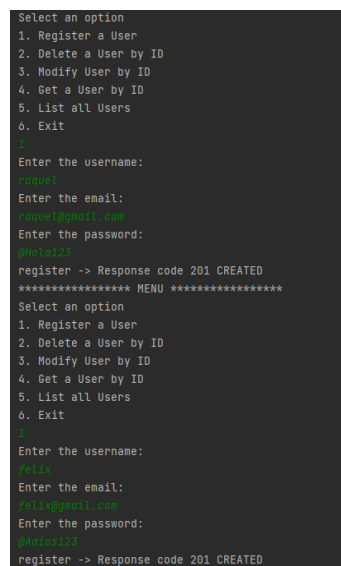
Debugger Console
2022-05-20 13:52:02.621 INFO 20064 --- [main]
2022-05-20 13:52:02.625 INFO 20064 --- [main]
2022-05-20 13:52:02.625 INFO 20064 --- [main]
2022-05-20 13:52:02.677 INFO 20064 --- [main]
2022-05-20 13:52:02.677 INFO 20064 --- [main]
2022-05-20 13:52:02.896 INFO 20064 --- [main]
2022-05-20 13:52:02.903 INFO 20064 --- [main]

1. Login to System Menu
2. Exit
3
***** SYSTEM MENU *****
Please enter your username:
lapuysa
Please enter your password:
mypassword1
***** MENU *****
Select an option
1. Register a User
2. Delete a User by ID
3. Modify User by ID
4. Get a User by ID
5. List all Users
6. Exit

```

*Inicio de sesión y muestra del menú CRUD*

- b. A continuación algunas imágenes sobre las operaciones que podrá realizar. Cualquier usuario dado de alta por esta vía podrá iniciar sesión en la aplicación Android con dicho usuario.



```

Select an option
1. Register a User
2. Delete a User by ID
3. Modify User by ID
4. Get a User by ID
5. List all Users
6. Exit
1
Enter the username:
raquel
Enter the email:
Raquel@gmail.com
Enter the password:
Raquel123
register -> Response code 201 CREATED
***** MENU *****
Select an option
1. Register a User
2. Delete a User by ID
3. Modify User by ID
4. Get a User by ID
5. List all Users
6. Exit
1
Enter the username:
felix
Enter the email:
Felix@gmail.com
Enter the password:
Felix123
register -> Response code 201 CREATED

```

*Damos de alta dos usuarios: raquel y felix*

```

4. Get a User by ID
5. List all Users
6. Exit
$
User(id=2, username=raquel, email=raquel@gmail.com, password=aUJT/k+d/CgHBsonVy537g==)
User(id=3, username=felix, email=felix@gmail.com, password=3t6h+t8DDpgTMPv2ef3mJA==)
***** MENU *****
Select an option
1. Register a User
2. Delete a User by ID
3. Modify User by ID
4. Get a User by ID
5. List all Users
6. Exit
$
Enter the id of the user:
$
New Username:
tomas
New email:
tomas@gmail.com
New password:
qHolo123
***** MENU *****
Select an option
1. Register a User
2. Delete a User by ID
3. Modify User by ID
4. Get a User by ID
5. List all Users
6. Exit
$
User(id=2, username=raquel, email=raquel@gmail.com, password=aUJT/k+d/CgHBsonVy537g==)
User(id=3, username=tomas, email=tomas@gmail.com, password=aUJT/k+d/CgHBsonVy537g==)

```

*Listamos y cambiamos a felix por tomas y comprobamos el cambio*



*Cualquier usuario registrado via Admin podrá iniciar sesión en la app*

**NOTA:** El servicio no debe apagarse durante las pruebas de ejecución. En caso de, por cualquier motivo, apagarse, hay que borrar las tablas de admin y users y volver a levantarlo. Esto se debe a que en cada ejecución se crean los diferentes objetos de encriptación para las contraseñas, no siendo iguales en ningún momento.