



# NIMESUKI

Desarrollo de Aplicaciones

Multiplataforma

Eloy Castro Saleta

Tutor: Fernando Vázquez Sánchez

# ÍNDICE

1) INTRODUCCIÓN .....	3
2) DESCRIPCIÓN Y ÁMBITO .....	4
3) OBJETIVOS .....	5
4) HERRAMIENTAS Y TECNOLOGÍAS .....	6
5) MODELO ENTIDAD-RELACIÓN .....	8
6) DIAGRAMA DE CLASES .....	9
7) DIAGRAMA DE PAQUETES .....	10
8) DIAGRAMAS DE CASOS DE USO .....	11
9) ANÁLISIS DE CÓDIGO .....	17
10) DIFICULTADES Y SOLUCIONES .....	22
11) PRUEBAS Y CONTROL DE ERRORES .....	25
12) MANUAL DE INSTALACIÓN .....	27
13) MANUAL DE USUARIO .....	30
14) PLANIFICACIÓN TEMPORAL .....	33
15) DAFO .....	34
16) ANÁLISIS PRESUPUESTARIO .....	35
17) MEJORAS A FUTURO .....	40
18) CONCLUSIONES .....	42
19) BIBLIOGRAFÍA .....	43
20) REPOSITORIO .....	45

# **1) INTRODUCCIÓN**

El presente documento indaga todo lo relativo al Proyecto del Grado Superior en Desarrollo de Aplicaciones Multiplataforma, también conocido como DAM, en la que se pusieron en práctica todos los conocimientos adquiridos durante el ciclo.

Dicho Proyecto se compone de un servicio REST, una aplicación de escritorio y una aplicación móvil. El servicio REST hará de intermediario entre la base de datos y la aplicación móvil. La aplicación de escritorio se conecta directamente a la base de datos, sin emplear el servicio REST, y permite gestionar todos sus datos. Por otra parte, la aplicación móvil consiste en la gestión personal de los animes favoritos del usuario.

La elección para desarrollar la aplicación surgió de mi gusto personal por este tipo de contenido audiovisual. Tras varios años en este mundillo, he podido observar como muchas personas gestionan los animes que han visto. La organización es algo que muchísimas personas necesitan en muchísimos ámbitos, especialmente en este, y aún hoy muchas de ellas siguen usando herramientas anticuadas como el bloc de notas, las hojas de cálculo o incluso papel y lápiz.

Es cierto que hay aplicaciones con un objetivo similar, pero también reciben bastantes críticas. Por estas críticas, y por no dar el paso al cambio, muchas personas se quedan estancadas en estas herramientas precarias, que no están pensadas para cubrir sus verdaderas necesidades. Me parece que es un desperdicio que tantas personas que disfrutan de estas obras no utilicen aplicaciones enfocadas a este nicho, el cual ha ido creciendo día a día.

Este tipo de aplicaciones les permite organizarse y llevar un seguimiento de sus series favoritas, siendo una gran ventaja para quienes suelen dejar una serie a medias y, meses después, desean retomarla pero ya no recuerdan en qué punto se quedaron, si merece la pena continuarla o si, de hecho, les pareció tan buena que quieren volver a verla desde el inicio.

## **2) DESCRIPCIÓN Y ÁMBITO**

Este proyecto consiste en el desarrollo de una aplicación móvil y otra de escritorio. La idea principal es cubrir tanto el ámbito personal como el administrativo, ya que cada aplicación está orientada a un perfil de usuario diferente.

La aplicación de escritorio está pensada para el usuario administrador (aunque diseñada también para usuarios normales), que se encargará de gestionar el contenido de la base de datos. Está diseñada para utilizarse desde un solo ordenador, lo cual permite tener un mayor control sobre los datos y la seguridad.

La aplicación móvil va dirigida al usuario final, el cual podrá consultar un catálogo de animes, ver la información detallada de cada uno de estos, añadirlos a una lista personal de favoritos, llevar un seguimiento de los episodios que ha visto y valorarlos. La comunicación con la base de datos se realiza exclusivamente a través del servicio REST, lo que permite garantizar cierta independencia entre cliente y servidor, así como una estructura más limpia y modular.

La idea del proyecto y su desenvolvimiento se han ido construyendo en base a las reuniones con el tutor, quien ha orientado las decisiones técnicas, y también a la experiencia propia como usuario de aplicaciones similares.

Aunque en esta versión inicial se ha planteado una estructura sencilla, el proyecto está diseñado para poder ampliarse a futuro, tanto en número de usuarios como en funcionalidades.

### **3) OBJETIVOS**

El objetivo principal de este proyecto es desarrollar una solución multiplataforma compuesta por una aplicación móvil enfocada en el seguimiento personalizado de animes, y una aplicación de escritorio orientada a la gestión administrativa. A través de este sistema se pretende ofrecer una experiencia fluida y centralizada, donde los usuarios puedan explorar y organizar contenido, mientras que desde la parte administrativa se gestionan los datos necesarios para el funcionamiento del sistema.

La finalidad de esta aplicación es proporcionar una herramienta ligera, accesible y personalizada para aficionados al anime, que deseen llevar un seguimiento de los títulos que ven, anotar su progreso, puntuar y guardar favoritos. A diferencia de plataformas ya existentes como MyAnimeList o AniList, que dependen de APIs externas, este proyecto propone una solución con control total sobre la base de datos y los datos del usuario, sin necesidad de registros externos ni conexión con servicios de terceros, aunque aún no se descarta esa opción a futuro.

Además, se busca que el sistema sea fácilmente ampliable, tanto a nivel funcional como estructural, con una arquitectura modular basada en servicios REST, lo que permite separar las capas de cliente/servidor y favorecer la escalabilidad.

Aunque existen aplicaciones que permiten gestionar listas de animes, la mayoría están centralizadas en grandes plataformas con diseños complejos y funciones innecesarias para un usuario promedio. Este proyecto apuesta por una alternativa simple, enfocada a un uso más personal donde se valore más la experiencia del usuario y la facilidad de mantenimiento.

Desde el punto de vista académico, esta aplicación sirve como ejercicio práctico para integrar múltiples competencias del ciclo formativo, como el desarrollo de interfaces gráficas, el uso y gestión de bases de datos y la implementación de servicios web.

## 4) HERRAMIENTAS Y TECNOLOGÍAS

### → **Apache Netbeans:**

IDE gratuito y de código abierto, diseñado principalmente para Java con acceso a más idiomas, utilizado para la creación del servicio REST y la aplicación de escritorio.

**Versión:** IDE 25



### → **Android Studio:**

IDE oficial para el desarrollo de aplicaciones Android, utilizado para la creación de la aplicación móvil en Java.

**Versión:** Meerkat Feature Drop | 2024.3.2



### → **MySQL Server:**

Sistema de gestión de bases de datos relacional utilizado para almacenar y gestionar los datos.

**Versión:** 8.0



### → **MySQL Workbench:**

Herramienta visual para el diseño, modelado, administración y mantenimiento de bases de datos MySQL, usada para gestionar las bases de datos del proyecto.

**Versión:** 8.0.38



### → **Bruno:**

Herramienta para pruebas y depuración de APIs REST, similar a Postman, que permite realizar solicitudes HTTP y analizar respuestas durante el desarrollo de servicios web.

**Versión:** 2.3.0



→ **GIT for Windows:**

Herramienta de control de versiones que permite gestionar el código fuente mediante Git, utilizando una interfaz de línea de comandos.

**Versión:** 2.49.0



→ **Git Hub:**

Plataforma de alojamiento de repositorios Git para el gestión, almacenamiento y colaboración del código fuente, así como para manejar imágenes y otros archivos del proyecto.



→ **Spring Initializr:**

Herramienta web utilizada para la creación del servicio REST en Java, permitiendo generar de forma rápida la estructura base del proyecto Spring Boot con las dependencias necesarias.



→ **PlantUML:**

Herramienta para la creación de diagramas UML a partir de texto plano. Facilita la documentación visual de sistemas y procesos.



→ **Launch4j:**

Herramienta que permite convertir archivos .jar de aplicaciones Java en archivos ejecutables .exe para Windows.

**Versión:** 3.50



→ **Inno Setup:**

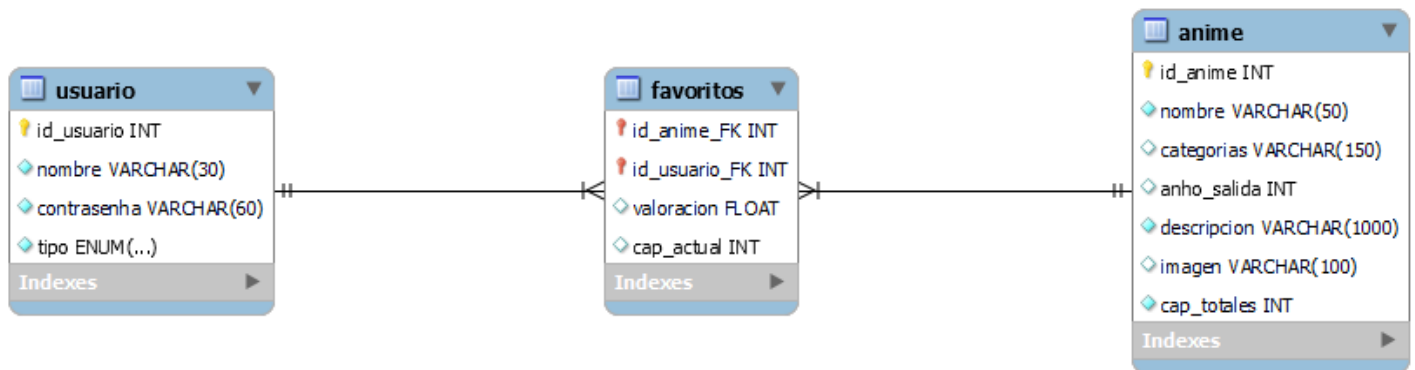
Asistente para la creación de instaladores para aplicaciones Windows.

Permite empaquetar archivos, configurar accesos directos, etc.

**Versión:** 6.4.3

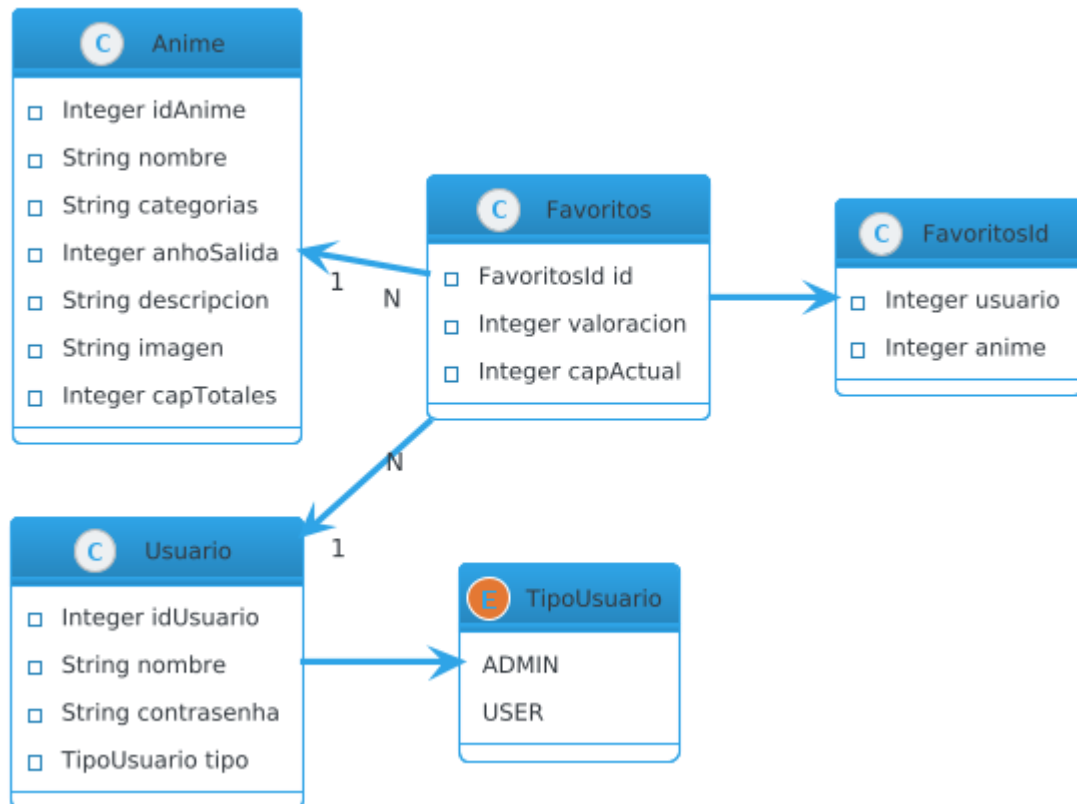


## 5) MODELO ENTIDAD-RELACIÓN





## 6) DIAGRAMA DE CLASES



## 7) DIAGRAMAS DE PAQUETES

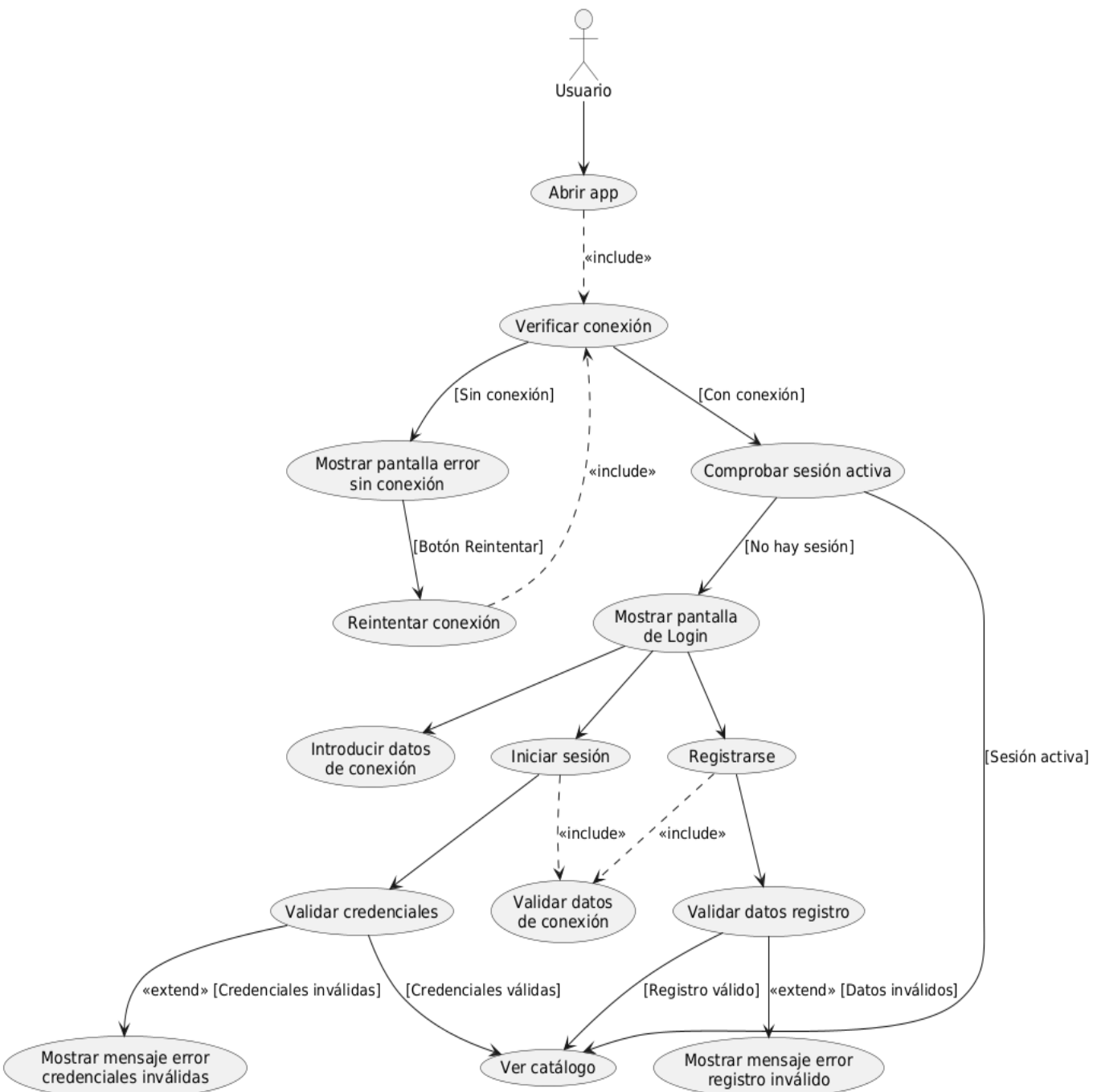


## 8) DIAGRAMAS DE CASOS DE USO

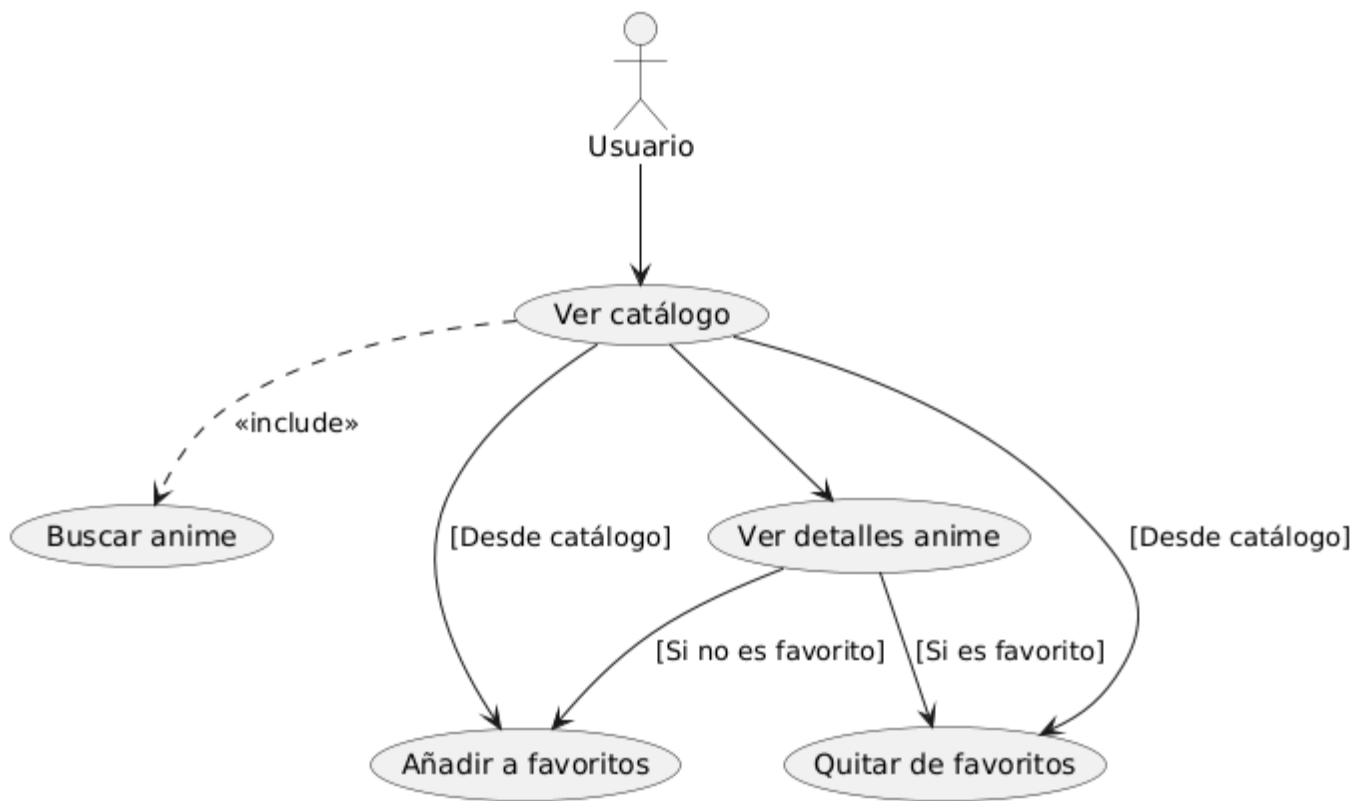
Los diagramas de casos de uso son una herramienta ampliamente utilizada para representar cómo interactúan los distintos actores con un sistema. Estos diagramas muestran escenarios específicos en los que los actores se relacionan con el sistema para lograr un objetivo concreto. Generalmente, estos casos se ilustran mediante diagramas que se complementan con descripciones escritas.

A continuación, se enseñarán diversos diagramas de casos de uso de la aplicación móvil desde el punto de vista del usuario, simples e intuitivos para cualquier lector, con el fin de enseñar el flujo de la aplicación:

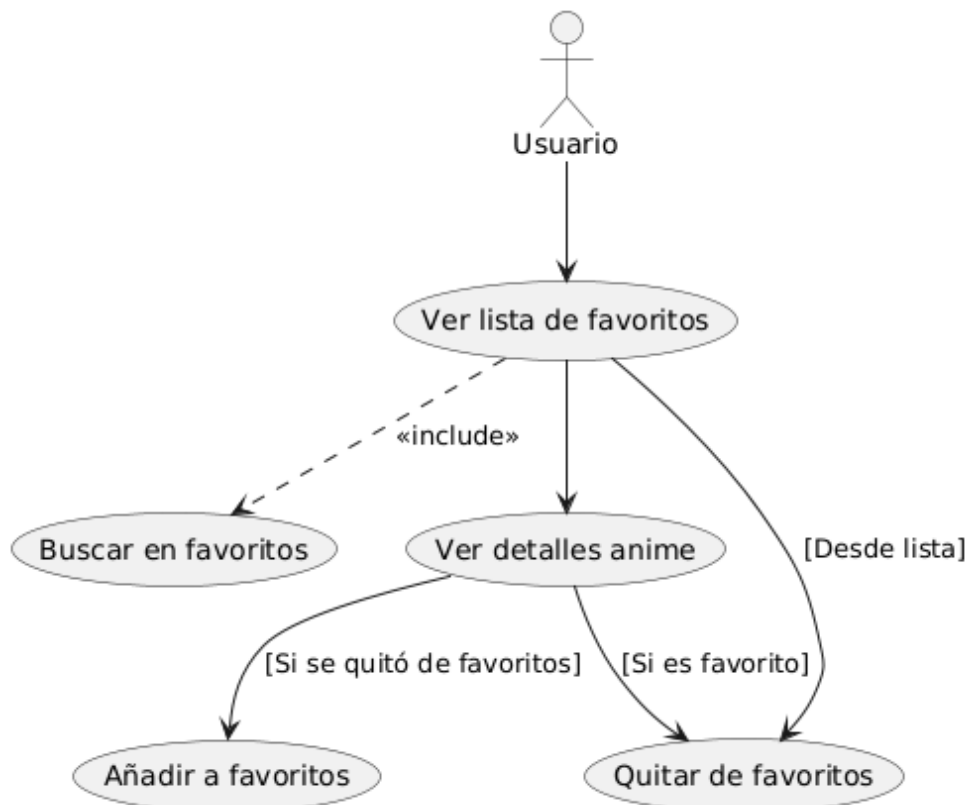
### Inicio de la aplicación



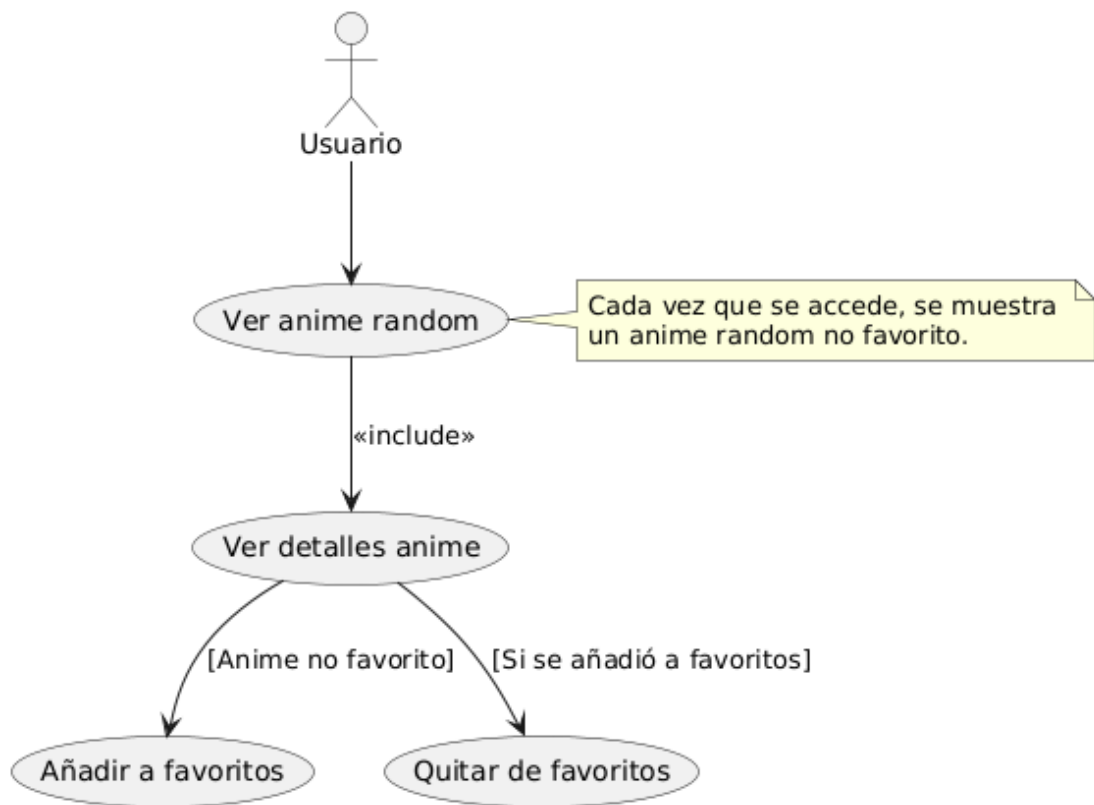
## Catálogo de animes



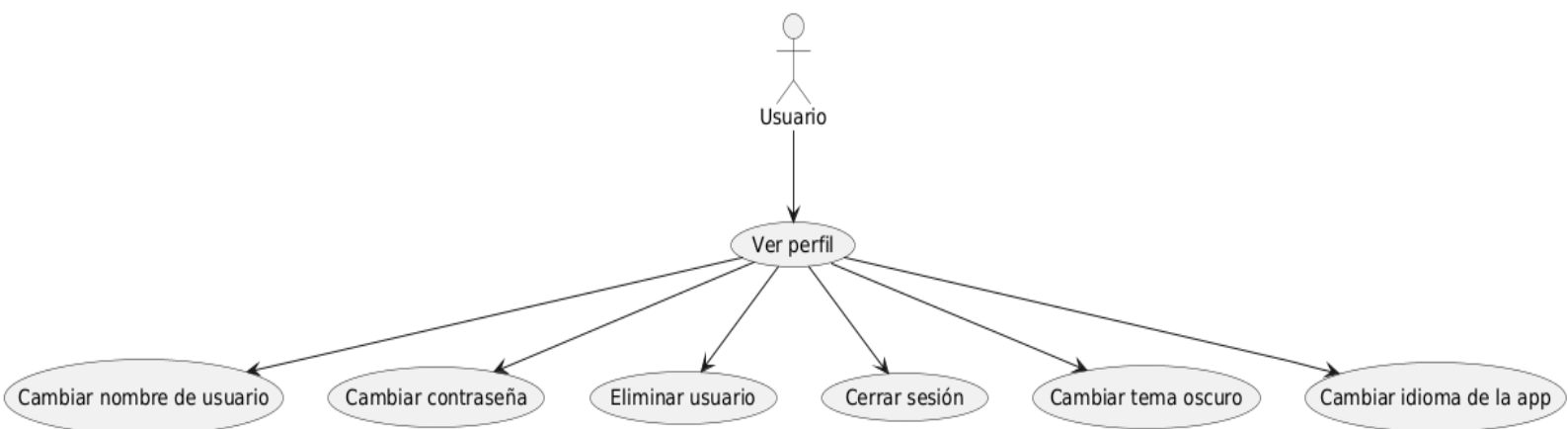
## Lista de animes favoritos



## Anime random

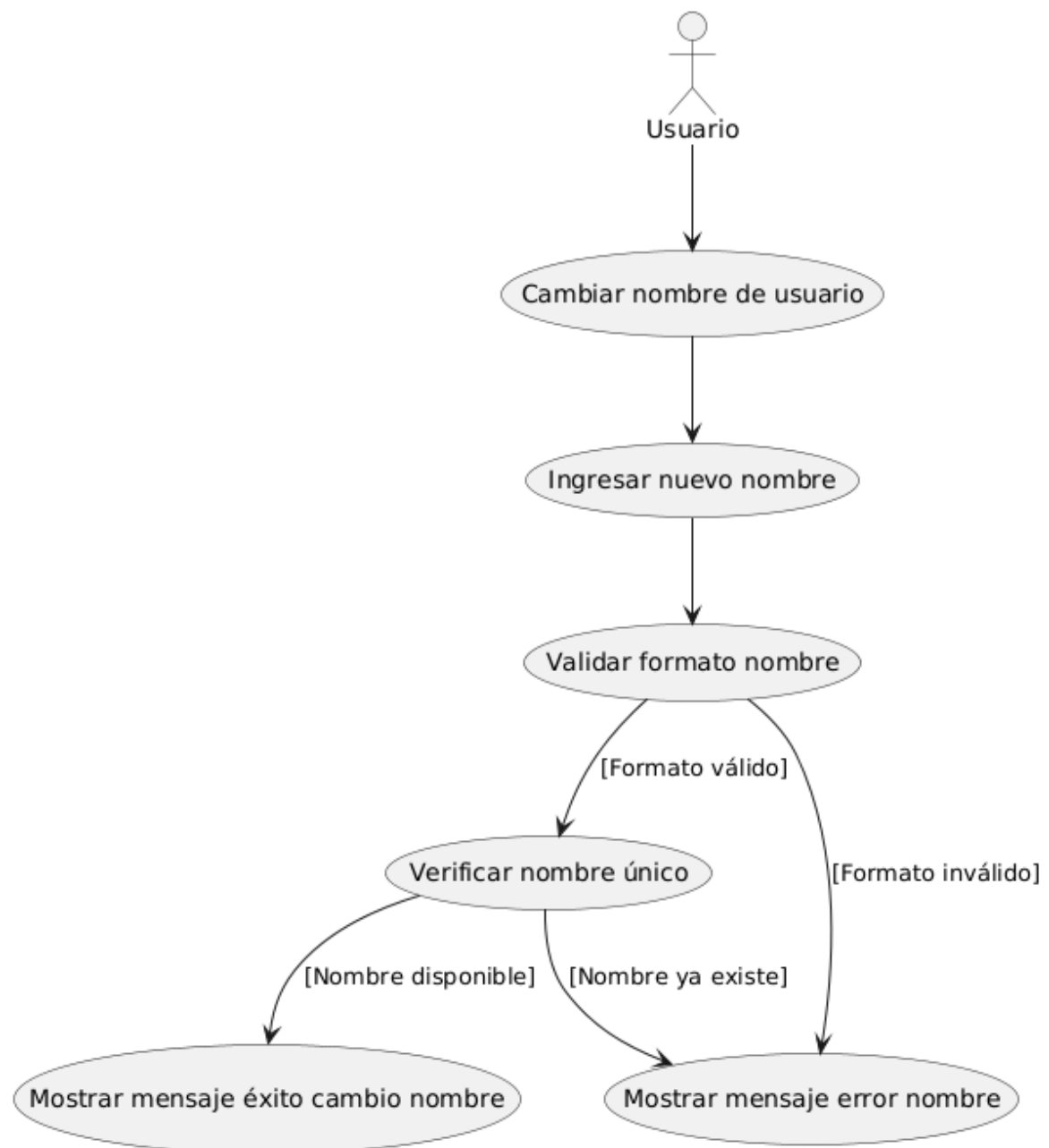


## Perfil

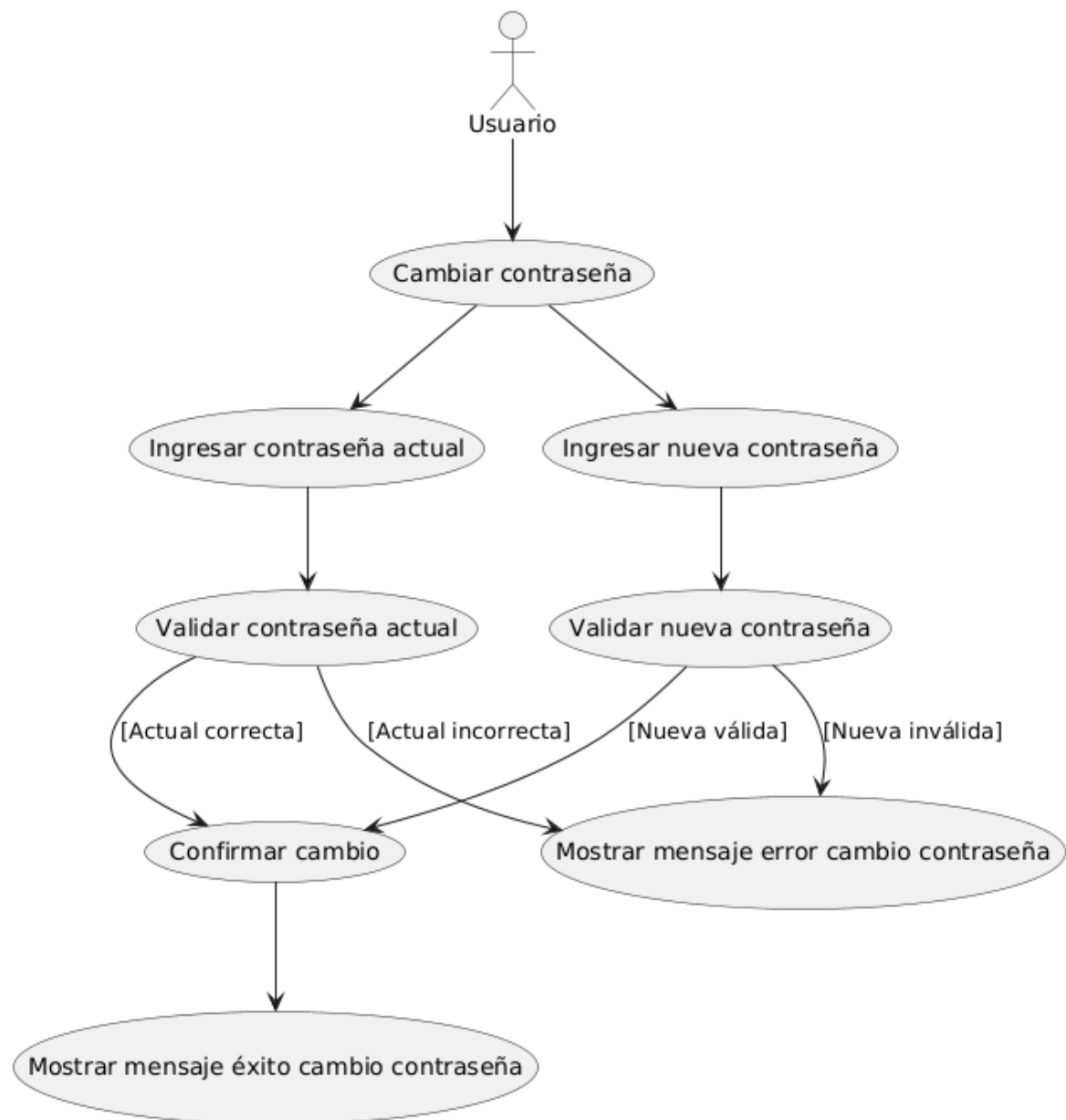


## Subprocesos perfil

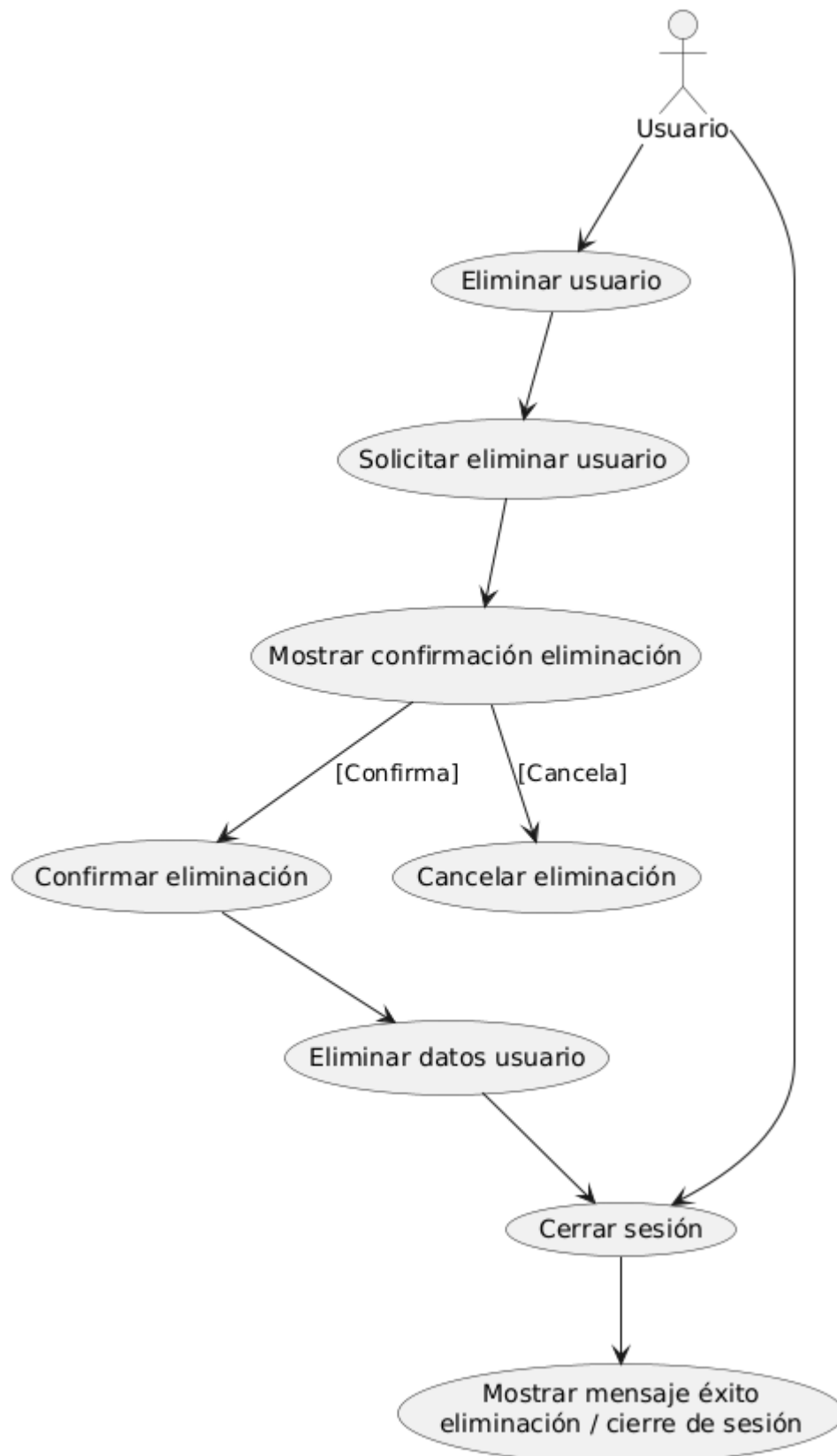
### 1. Cambiar nombre usuario



## 2. Cambiar contraseña



### 3. Eliminar usuario / Cerrar Sesión





## 9) ANÁLISIS DE CÓDIGO

BroadcastReceiver y Calendar para el envío de una notificación semana y otro

BroadcastReceiver para reprogramar la notificación cada vez que se lance:

```
public class WeeklyAnimeNotificationReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
            if (ContextCompat.checkSelfPermission(context, android.Manifest.permission.POST_NOTIFICATIONS)
                != PackageManager.PERMISSION_GRANTED) {
                return;
            }
        }

        NotificationManager notificationManager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
        String channelId = "anime_channel";

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel = new NotificationChannel(
                channelId,
                name: "Notificaciones semanales",
                NotificationManager.IMPORTANCE_DEFAULT
            );
            notificationManager.createNotificationChannel(channel);
        }

        Intent launchIntent = new Intent(context, SplashActivity.class);
        launchIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);

        PendingIntent contentIntent = PendingIntent.getActivity(
            context,
            requestCode: 0,
            launchIntent,
            flags: PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_IMMUTABLE
        );

        NotificationCompat.Builder builder = new NotificationCompat.Builder(context, channelId)
            .setSmallIcon(R.drawable.ic_notificacion_prueba)
            .setContentTitle(";A POR OTRA SEMANA DE NUEVOS EPISODIOS!")
            .setContentText(";No te pierdas tus animés favoritos esta semana!")
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)
            .setContentIntent(contentIntent)
            .setColor(ContextCompat.getColor(context, R.color.pastelBackground))
            .setAutoCancel(true);

        notificationManager.notify(id: 100, builder.build());
    }
}

public class BootReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (Intent.ACTION_BOOT_COMPLETED.equals(intent.getAction())) {
            ListaAnimesView.scheduleWeeklyAnimeNotification(context);
        }
    }
}
```

```

public static void scheduleWeeklyAnimeNotification(Context context) { 2 usages
    SharedPreferences prefs = context.getSharedPreferences( name: "MisPreferencias", MODE_PRIVATE);
    boolean alreadyScheduled = prefs.getBoolean( key: "alarm_scheduled", defValue: false);

    if (alreadyScheduled) return;

    Calendar calendar = Calendar.getInstance();
    calendar.set(Calendar.DAY_OF_WEEK, Calendar.FRIDAY);
    calendar.set(Calendar.HOUR_OF_DAY, 18);
    calendar.set(Calendar.MINUTE, 0);
    calendar.set(Calendar.SECOND, 0);
    calendar.set(Calendar.MILLISECOND, 0);

    if (calendar.getTimeInMillis() < System.currentTimeMillis()) {
        calendar.add(Calendar.WEEK_OF_YEAR, amount: 1);
    }

    Intent intent = new Intent(context, WeeklyAnimeNotificationReceiver.class);
    PendingIntent pendingIntent = PendingIntent.getBroadcast(
        context, requestCode: 1, intent, flags: PendingIntent.FLAG_IMMUTABLE | PendingIntent.FLAG_UPDATE_CURRENT);

    AlarmManager alarmManager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);

    alarmManager.setRepeating(
        AlarmManager.RTC_WAKEUP,
        calendar.getTimeInMillis(),
        intervalMillis: AlarmManager.INTERVAL_DAY * 7,
        pendingIntent
    );

    prefs.edit().putBoolean("alarm_scheduled", true).apply();
}

```

Administrador de favoritos a través del nombre del anime para la diferenciación en el catálogo:

```
public class FavoritosManager { 21 usages
    private static final String PREFS_NAME = "MisPreferencias"; 1 usage
    private static final String KEY_FAVORITOS = "favoritos_cache"; 2 usages

    private SharedPreferences prefs; 3 usages
    private Gson gson; 3 usages

    public FavoritosManager(Context context) { 7 usages
        prefs = context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
        gson = new Gson();
    }

    public void guardarFavoritos(ArrayList<String> favoritos) { 7 usages
        String json = gson.toJson(favoritos);
        prefs.edit().putString(KEY_FAVORITOS, json).apply();
    }

    public ArrayList<String> cargarFavoritos() { 6 usages
        String json = prefs.getString(KEY_FAVORITOS, defValue: null);
        if (json == null) return new ArrayList<>();
        Type type = new TypeToken<ArrayList<String>>() {
        }.getType();
        return gson.fromJson(json, type);
    }
}
```

## Number Picker para la actualización del capítulo actual:

```
public class PickerLayoutManager extends LinearLayoutManager { 6 usages

    private RecyclerView recyclerView; 9 usages
    private OnItemSelectedListener callback; 3 usages

    public PickerLayoutManager(Context context) { super(context, HORIZONTAL, reverseLayout: false); }

    public void setOnItemSelectedListener(OnItemSelectedListener callback) { 2 usages
        this.callback = callback;
    }

    @Override
    public void onAttachedToWindow(@NonNull RecyclerView view) {
        super.onAttachedToWindow(view);
        recyclerView = view;
        new LinearSnapHelper().attachToRecyclerView(recyclerView);
    }

    @Override
    public void onLayoutChildren(RecyclerView.Recycler recycler, RecyclerView.State state) {
        super.onLayoutChildren(recycler, state);
        scaleDownChildren();
    }

    @Override
    public int scrollHorizontallyBy(int dx, RecyclerView.Recycler recycler, RecyclerView.State state) {
        int scrolled = super.scrollHorizontallyBy(dx, recycler, state);
        scaleDownChildren();
        return scrolled;
    }

    private void scaleDownChildren() { 2 usages
        int mid = getWidth() / 2;

        for (int i = 0; i < getChildCount(); i++) {
            View child = getChildAt(i);
            if (child == null) continue;

            int childMid = (getDecoratedLeft(child) + getDecoratedRight(child)) / 2;
            float distance = Math.abs(mid - childMid);

            float d = Math.min(distance / (float) mid, 1f);
            float scale = 1f - d * 0.65f;
            scale = Math.max(0.4f, scale);

            child.setScaleX(scale);
            child.setScaleY(scale);
        }
    }

    @Override
    public void onScrollStateChanged(int state) {
        super.onScrollStateChanged(state);

        if (state == RecyclerView.SCROLL_STATE_IDLE && recyclerView != null && callback != null) {
            int recyclerViewCenterX = recyclerView.getWidth() / 2;

            int minDistance = recyclerView.getWidth();
            int selectedPosition = -1;

            for (int i = 0; i < recyclerView.getChildCount(); i++) {
                View child = recyclerView.getChildAt(i);
                int childCenterX = (getDecoratedLeft(child) + getDecoratedRight(child)) / 2;
                int distance = Math.abs(childCenterX - recyclerViewCenterX);
                if (distance < minDistance) {
                    minDistance = distance;
                    selectedPosition = recyclerView.getChildAdapterPosition(child);
                }
            }

            if (selectedPosition != -1) {
                int finalSelectedPosition = selectedPosition;
                recyclerView.post(() -> callback.onItemSelected(finalSelectedPosition));
            }
        }
    }

    public interface OnItemSelectedListener { 2 usages
        void onItemSelected(int position); 1 usage
    }
}
```

```

public class PickerAdapter extends RecyclerView.Adapter<PickerAdapter.PickerViewHolder> { 8 usages

    private ArrayList<String> data = new ArrayList<>(); 4 usages
    private int selectedItem = -1; 5 usages
    private Context context; 4 usages

    public PickerAdapter(Context context) { this.context = context; }

    public void setData(ArrayList<String> data) { 2 usages
        this.data.clear();
        this.data.addAll(data);
        notifyDataSetChanged();
    }

    public void setSelectedItem(int position) { 4 usages
        int oldPosition = selectedItem;
        selectedItem = position;
        if (oldPosition != -1) notifyItemChanged(oldPosition);
        notifyItemChanged(selectedItem);
    }

    public int getSelectedItem() { return selectedItem; }

    @NonNull
    @Override
    public PickerViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.row_view_slider_item, parent, attachToRoot: false);
        return new PickerViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull PickerViewHolder holder, int position) {
        holder.tvItem.setText(data.get(position));

        if (position == selectedItem) {
            holder.tvItem.setTextColor(ContextCompat.getColor(context, R.color.pastelAccent));
        } else {
            holder.tvItem.setTextColor(ContextCompat.getColor(context, R.color.pastelGray));
        }
    }

    @Override
    public int getItemCount() { return data.size(); }

    static class PickerViewHolder extends RecyclerView.ViewHolder { 4 usages
        TextView tvItem; 4 usages

        public PickerViewHolder(@NonNull View itemView) { 1 usage
            super(itemView);
            tvItem = itemView.findViewById(R.id.tv_item);
        }
    }
}

```

## 10) DIFICULTADES Y SOLUCIONES

Una de las dificultades fue el tener que poner los datos de conexión a mano en las apps, por lo que opté por el uso de un Entity Manager dinámico, el cual se crea pasándole la IP y el nombre y contraseña del usuario de la base de datos. En los services se creará para que en los repository se haga una llamada createQuery.

```
/**
 *
 * @author eloy.castro
 */
public class DynamicDataSourceFactory {

    public static DataSource create(String host, String username, String password) {
        HikariDataSource ds = new HikariDataSource();
        ds.setJdbcUrl("jdbc:mysql://" + host + ":3306/nimesuki?serverTimezone=UTC");
        ds.setUsername(username);
        ds.setPassword(password);
        ds.setDriverClassName("com.mysql.cj.jdbc.Driver");
        return ds;
    }
}

/**
 *
 * @author eloy.castro
 */
public class DynamicEntityManagerFactory {

    private static final Map<String, EntityManagerFactory> cache = new ConcurrentHashMap<>();

    public static EntityManager createEntityManager(DataSource dataSource, String key) {
        EntityManagerFactory emf = cache.computeIfAbsent(key, k -> {
            LocalContainerEntityManagerFactoryBean factoryBean = new LocalContainerEntityManagerFactoryBean();
            factoryBean.setDataSource(dataSource);
            factoryBean.setPackagesToScan("proyecto.nimesuki.modelo");
            factoryBean.setJpaVendorAdapter(new HibernateJpaVendorAdapter());

            Map<String, Object> props = new HashMap<>();
            props.put("hibernate.hbm2ddl.auto", "none");
            props.put("hibernate.dialect", "org.hibernate.dialect.MySQL8Dialect");

            factoryBean.setJpaPropertyMap(props);
            factoryBean.afterPropertiesSet();

            return factoryBean.getObject();
        });

        return emf.createEntityManager();
    }
}
```

Un problema que tuve fue al querer hacer la descripción expandible ya que quería añadir el texto para mostrar y ocultar pegado a la descripción, lo cual fue bastante difícil, pero lo solucioné gracias a la información obtenida y específicamente al uso del `SpannableString`, `ClickableSpan` y el `TextPaint`:

```
private void setupExpandableText(MaterialTextView textView, String descripcion, int maxLines) { 1 usage
    String mostrar = "Mostrar";
    String ocultar = "Ocultar";
    String puntos = "...";

    textView.setMaxLines(Integer.MAX_VALUE);

    textView.getViewTreeObserver().addOnGlobalLayoutListener(new ViewTreeObserver.OnGlobalLayoutListener() {
        @Override
        public void onGlobalLayout() {
            textView.getViewTreeObserver().removeOnGlobalLayoutListener( victim: this);

            TextPaint paint = textView.getPaint();
            int lineHeight = textView.getLineHeight();
            int maxHeight = lineHeight * maxLines;

            Runnable updateText = new Runnable() {
                @Override
                public void run() {
                    if (isExpanded) {
                        String textoCompleto = descripcion + " " + ocultar;
                        SpannableString spannable = new SpannableString(textoCompleto);

                        int start = textoCompleto.lastIndexOf(ocultar);
                        int end = textoCompleto.length();

                        ClickableSpan clickableSpan = new ClickableSpan() {
                            @Override
                            public void onClick(@NonNull View widget) {
                                isExpanded = false;
                                run();
                            }

                            @Override
                            public void updateDrawState(@NonNull TextPaint tp) {
                                super.updateDrawState(tp);
                                tp.setColor(ContextCompat.getColor(textView.getContext(), R.color.pastelPrimary));
                                tp.setUnderlineText(true);
                                tp.setFakeBoldText(true);
                            }
                        };

                        spannable.setSpan(clickableSpan, start, end, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
                        spannable.setSpan(new StyleSpan(Typeface.BOLD), start, end, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);

                        textView.setText(spannable);
                        textView.setMovementMethod(LinkMovementMethod.getInstance());
                    }
                }
            };
        }
    });
}
```

Otro problema fue que se quisieran modificar datos muy rápido, sin tiempo para que se actualicen correctamente los datos, generando un error, por lo que opté por el uso de progress bar, dándole tiempo para cargar los datos.

Al obtener un anime random si se quería salir de la vista generaría un error al querer añadir datos a un fragmento que no existía ya, por lo que opté por bloquear tanto el menú de navegación como los gestos para salir:

```
private void enableBottomBar(boolean enable) { 2 usages
    for (int i = 0; i < bottomNavigationView.getMenu().size(); i++) {
        bottomNavigationView.getMenu().getItem(i).setEnabled(enable);
    }
}

getOnBackPressedDispatcher().addCallback( owner: this, new OnBackPressedCallback( enabled: true) {
    @Override 5 usages
    public void handleOnBackPressed() {
        if (!bloquearVolver) {
            setEnabled(false);
            getOnBackPressedDispatcher().onBackPressed();
        }
    }
});
bloquearVolver = true;
obtenerYMostrarAnimeRandom();
```

Por último, el material design no te deja jugar mucho con la personalización, por lo que tuve que cambiar varios componentes pasándose un style propio, o usando otros componentes, como el endIconDrawable para las contraseñas, porque cortaba el borde de los textLayout, por lo que opté por usar imageButton.

```
<style name="NoRippleBottomNav" parent="Widget.MaterialComponents.BottomNavigationView">
    <item name="itemRippleColor">@android:color/transparent</item>
    <item name="backgroundTint">@android:color/transparent</item>
    <item name="itemIconTint">@color/selector_icon</item>
    <item name="itemTextColor">@color/selector_text</item>
    <item name="android:layout_alignParentBottom">true</item>
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">75dp</item>
    <item name="labelVisibilityMode">labeled</item>
    <item name="itemPaddingBottom">20dp</item>
</style>

<style name="CustomRatingBarTheme" parent="Theme.MaterialComponents.DayNight">
    <item name="colorControlActivated">@color/pastelAccent</item>
</style>

<style name="TransparentToolbarMenu" parent="">
    <item name="android:backgroundTint">@android:color/transparent</item>
</style>
```



## 11) PRUEBAS Y CONTROL DE ERRORES

### Toast error:



19:54

Nombre de Usuario

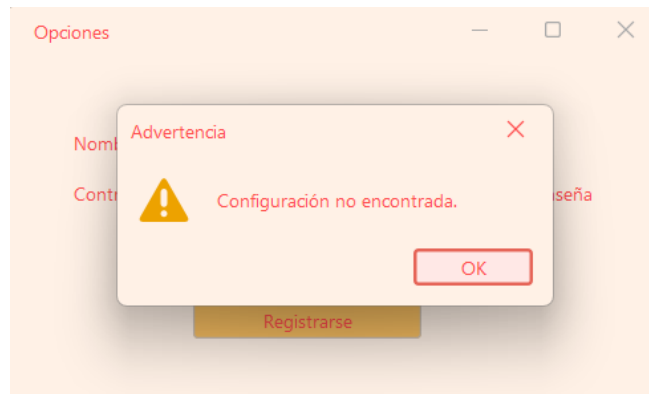
Contraseña

Iniciar sesión

Registrarse

Usuario no encontrado

### JOptionPane error:



Opciones

Advertencia

Configuración no encontrada.

OK

Registrarse

### Control de datos:



Usuarios

Id: 1

Nombre: Eloy

Contraseña: \*\*\*\*\*

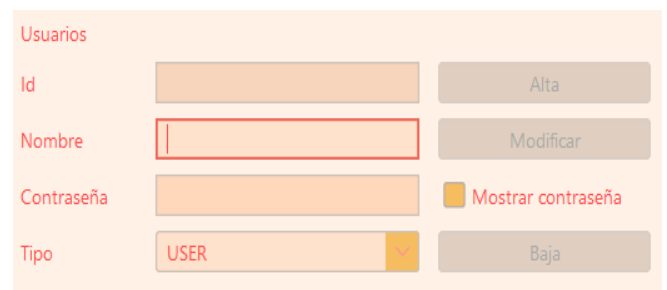
Tipo: USER

Alta

Modificar

Mostrar contraseña

Baja



Usuarios

Id:

Nombre:

Contraseña:

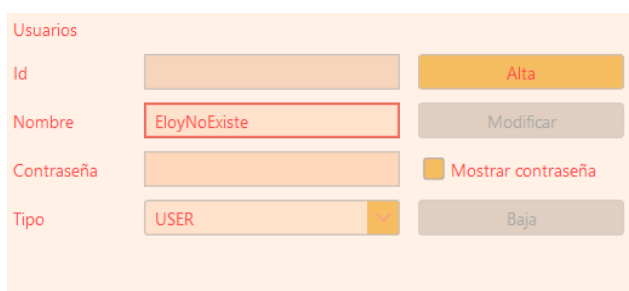
Tipo: USER

Alta

Modificar

Mostrar contraseña

Baja



Usuarios

Id:

Nombre: EloyNoExiste

Contraseña:

Tipo: USER

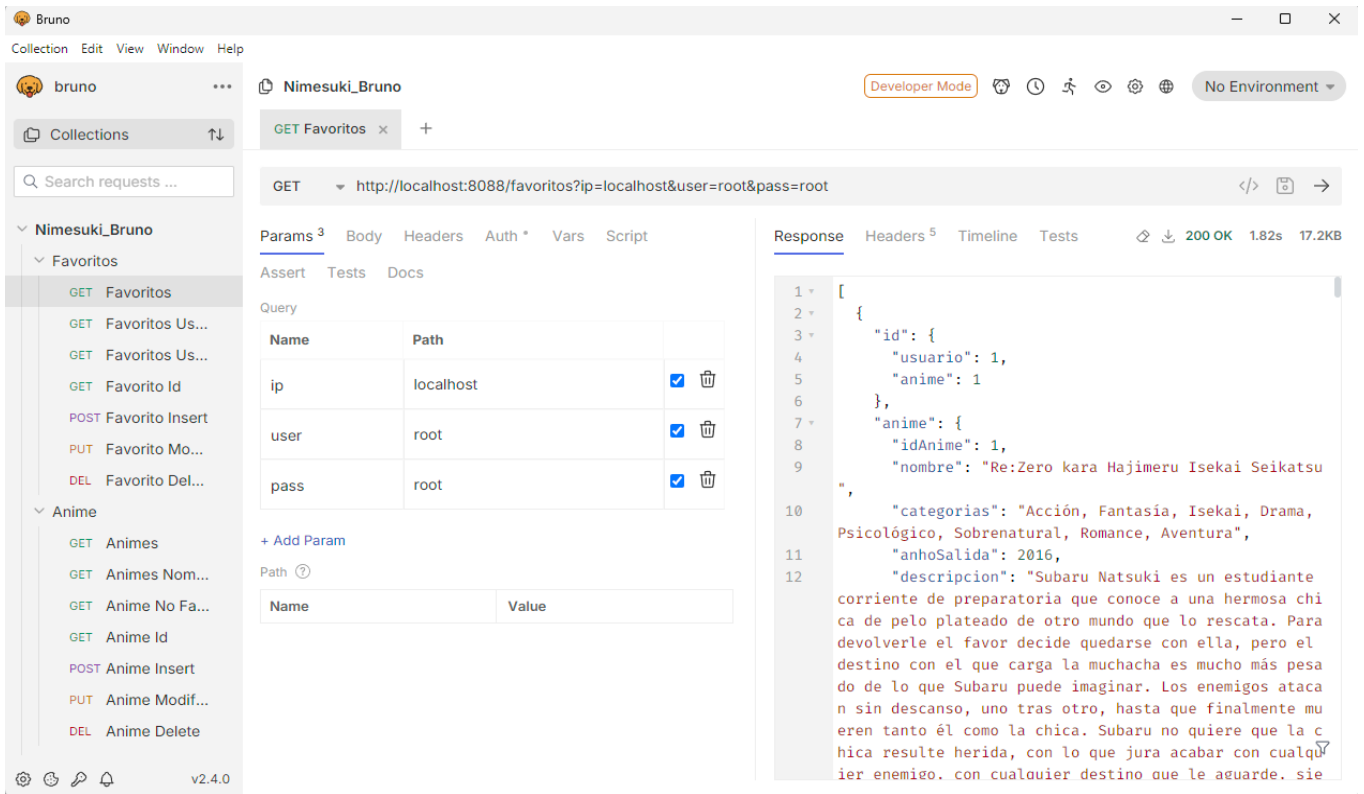
Alta

Modificar

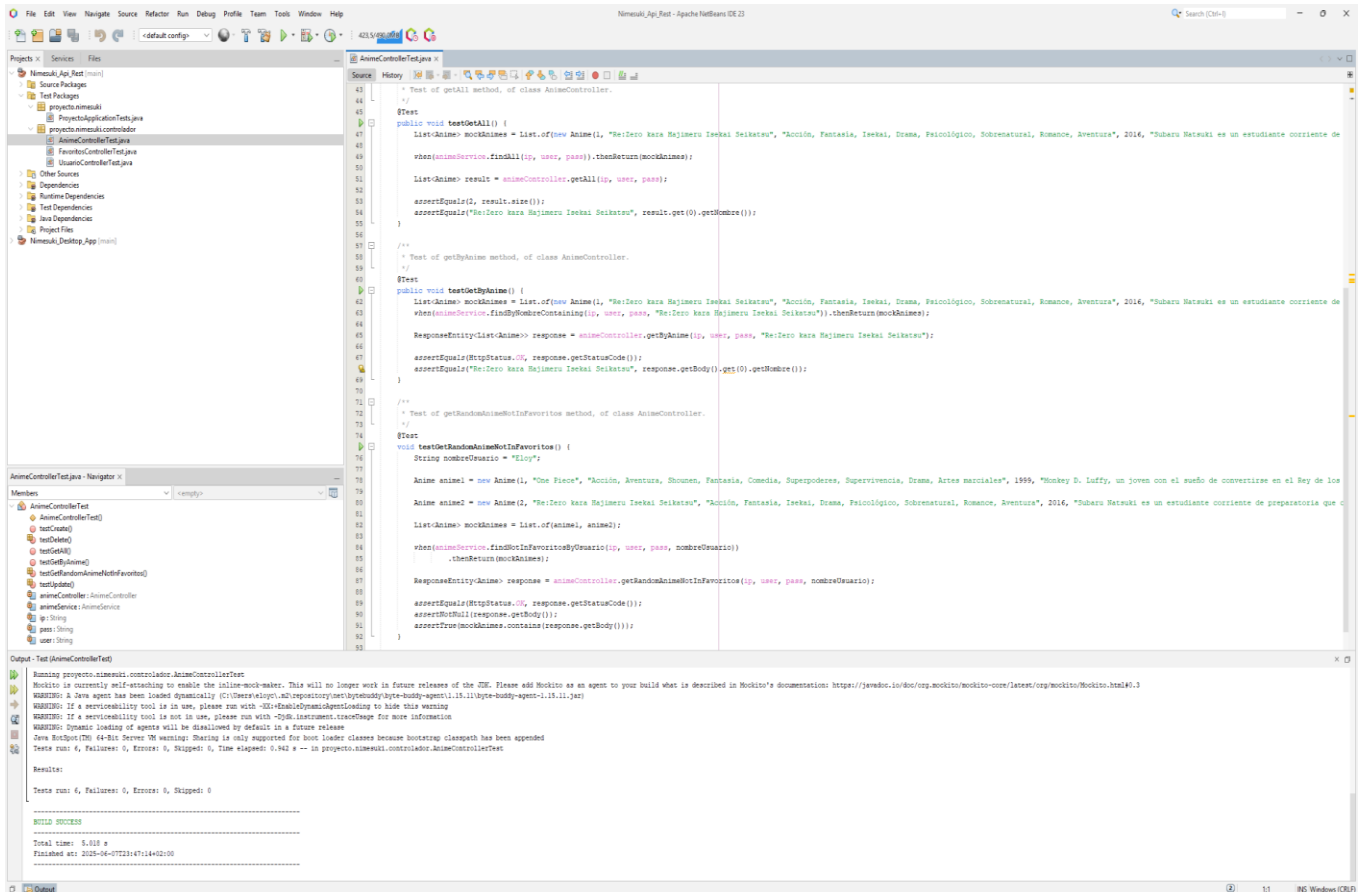
Mostrar contraseña

Baja

**Bruno:**



### Pruebas JUnit:



## 12) MANUAL DE INSTALACIÓN

Todos los archivos necesarios, incluyendo el código fuente, ejecutables (.jar, .exe, .apk) y el script de creación de la base de datos, se encuentran disponibles en el repositorio de GitHub al final de la documentación.

Requisitos generales:

- Sistema operativo: Windows, MacOS, Linux
- Memoria RAM: 8GB (16GB recomendado)
- Espacio disponible: Al menos 2GB de espacio
- Soporte para máquinas virtuales

### BD y MV

La base de datos se ejecuta en una máquina virtual, ejecutada por ejemplo desde VirtualBox, siendo necesario establecer los datos para asignar una IP. Como recomendación usaría host only. Para permitir el acceso remoto a esta desde otras aplicaciones, es necesario modificar el archivo configuración de MySQL **my.cnf** o **my.ini**. Dentro de este archivo, en la sección **[mysqld]**, se debe escribir esto:  
**bind-address = 0.0.0.0**

Este cambio permite aceptar conexiones entrantes desde cualquier dirección IP. Además, es necesario un usuario en MySQL con host **"%"** y con todos los privilegios.

Puede producirse un error al intentar modificar el archivo de configuración debido a restricciones de permisos. En ese caso, será necesario conceder al usuario los privilegios necesarios para editarlo. Esto puede implicar cambiar el propietario del archivo o establecer control total.

## Servicio REST

Requisitos:

- JDK: Versión 17 o superior
- Base de datos: MySQL 8 o compatible
- Puerto libre: 8088

Requiere acceso a red: Sí, conexión con la base de datos remota.

El servicio REST se ejecuta localmente en el equipo, si se quisiera usar en la máquina virtual habría que cambiar la IP asignada en la app móvil de 10.0.2.2 a la misma que la base de datos. Además, se encuentra configurado para escuchar en el puerto 8088.

Puede ejecutarse desde un entorno como NetBeans, IntelliJ o cualquier otro IDE compatible, aunque también está disponible como archivo .jar en el repositorio para su ejecución directa.

## App Escritorio

Requisitos:

- JDK: 17 o superior
- Sistema operativo: Windows, MacOS, Linux

La aplicación de escritorio puede ejecutarse como un .jar o .exe. La app permite añadir los datos de conexión en tiempo de ejecución guardándose en un .properties, gracias al uso de un Entity Manager dinámico, pasando los datos en la llamada al REST.

Opcionalmente, estos valores pueden establecerse editando el archivo .properties que acompaña al ejecutable. No es necesario ya que se pueden introducir mediante la aplicación, y será necesario guardarlos manualmente en cada sesión.

La app puede ejecutarse desde NetBeans, IntelliJ o cualquier otro IDE compatible

## App Móvil

Requisitos:

- IDE recomendado: Android Studio
- Versión mínima: Android 13
- Espacio libre: 200MB

La app móvil está desarrollada para Android y disponible como archivo .apk en el repositorio. Puede instalarse en un dispositivo físico (vía USB usando ADB reverse para conectar los puertos) o ejecutarse desde un emulador Android.

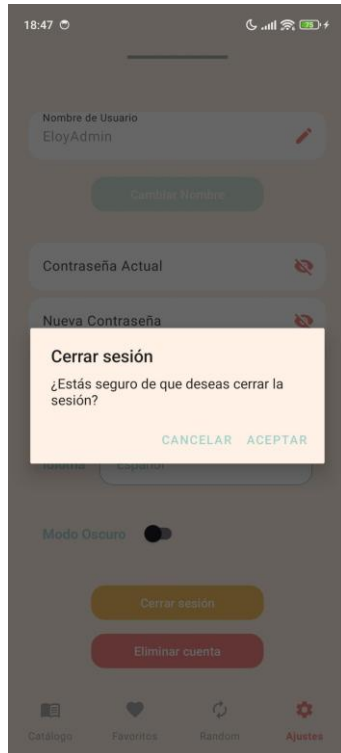
Cuando se ejecuta en un emulador, debe tenerse en cuenta que la IP localhost hace referencia al propio emulador, no al PC. Por eso, para conectarse al servicio desde el emulador, se debe utilizar la IP 10.0.2.2, a no ser que el servicio esté en la máquina virtual, en ese caso se usaría la misma IP que la base de datos, como ya comenté anteriormente.

Desde la app se introducirán los datos de conexión desde el login y se almacenarán en preferencias. Si aún no hay datos en preferencias se pondrán unos por defecto, pero hará falta guardarlos para su correcto funcionamiento.

## 13) MANUAL DE USUARIO

### App Móvil





## App Escritorio

Login:

Opciones

Configuración

Admin access

Nombre

Contraseña

Ver contraseña

Iniciar Sesión

Registrarse

Configuración:

IP

192.168.56.100

Usuario

root

Contraseña

\*\*\*\*

Cancelar

Guardar

Contraseña ADMIN:

Contraseña secreta

¿Cuál es la contraseña secreta?

OK

Cancel

## Vista ADMIN:

Opciones

Salir
Report
Consultas

Nombre
Contraseña
Tipo

Alta
Modificar
Mostrar contraseña
Baja

Animes
Id
Nombre
Categorías
Año Salida
Descripción
Imagen
Capítulos totales

Cambiar Nombre
Alta
Modificar
Baja

Favoritos
Anime
Usuario
Valoración
Capítulo Actual

Re:Zero kara Hajimeru Isekai Seikatsu
Eloy
5
25

Alta
Modificar
Baja

## Consultas:

Opciones

Volver

Usuario
Valoración

Ejecutar

Listado de animes

Ejecutar

Anime
Capítulos

Ejecutar

Listado de usuarios

Ejecutar

EloyAdmin | Re:Zero kara Hajimeru Isekai Seikatsu 3rd Season | 0.0  
EloyAdmin | Naruto | 0.0  
EloyAdmin | Naruto Shippuden | 0.0  
EloyAdmin | One Piece | 0.0  
Eloycorme3 | Re:Zero kara Hajimeru Isekai Seikatsu | 0.0  
Eloycorme3 | Re:Zero kara Hajimeru Isekai Seikatsu: Memory | 0.0  
Eloycorme3 | Re:Zero kara Hajimeru Isekai Seikatsu 2nd Season | 0.0  
Eloycorme3 | Re:Zero kara Hajimeru Isekai Seikatsu 3rd Season | 0.0  
Eloycorme3 | One Piece | 0.0

One Piece  
Death Note  
Fullmetal Alchemist  
Fullmetal Alchemist: Brotherhood  
Kimetsu no Yaiba  
Kimetsu no Yaiba - Tren Infinito  
Kimetsu no Yaiba - Distrito Rojo  
Kimetsu no Yaiba - Aldea de los herreros  
Kimetsu no Yaiba - Entrenamiento de los Pilares

One Piece | 1126  
Death Note | 37  
Fullmetal Alchemist | 51  
Fullmetal Alchemist: Brotherhood | 64  
Kimetsu no Yaiba | 26  
Kimetsu no Yaiba - Tren Infinito | 1  
Kimetsu no Yaiba - Distrito Rojo | 18  
Kimetsu no Yaiba - Aldea de los herreros | 11  
Kimetsu no Yaiba - Entrenamiento de los Pilares | 8

Eloy  
EloyAdmin  
Eloycorme3  
EloyUser  
EloyAdministrador  
Elox  
Eloo  
EloPrueba1

## Reports:

## Vista Usuario:

Opciones

Volver

Nombre
Año Inicio
Año Fin

Usuario
Límite

Generar Report
Generar Report

Opciones

Salir

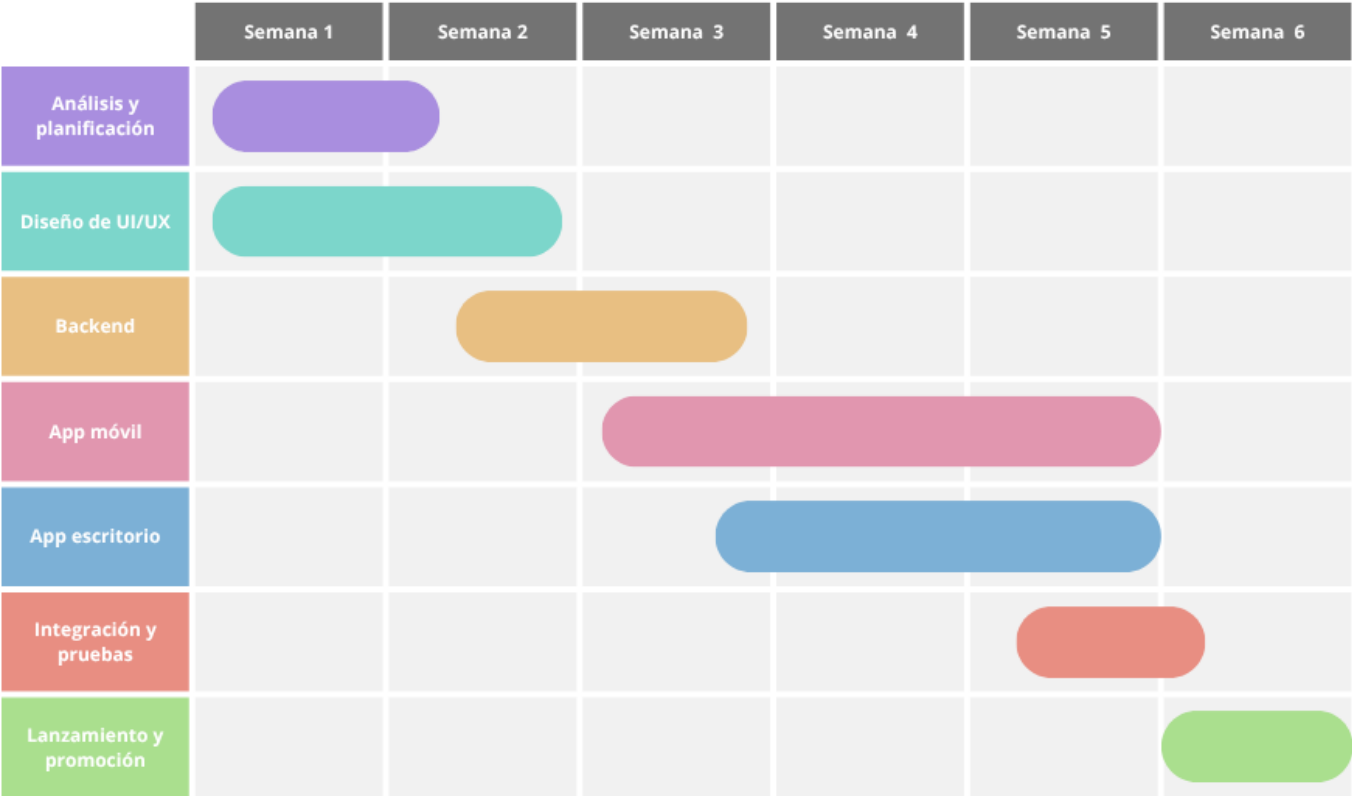
Nombre
Contraseña
Mostrar contraseña

Cambiar nombre
Cambiar contraseña
Eliminar Usuario

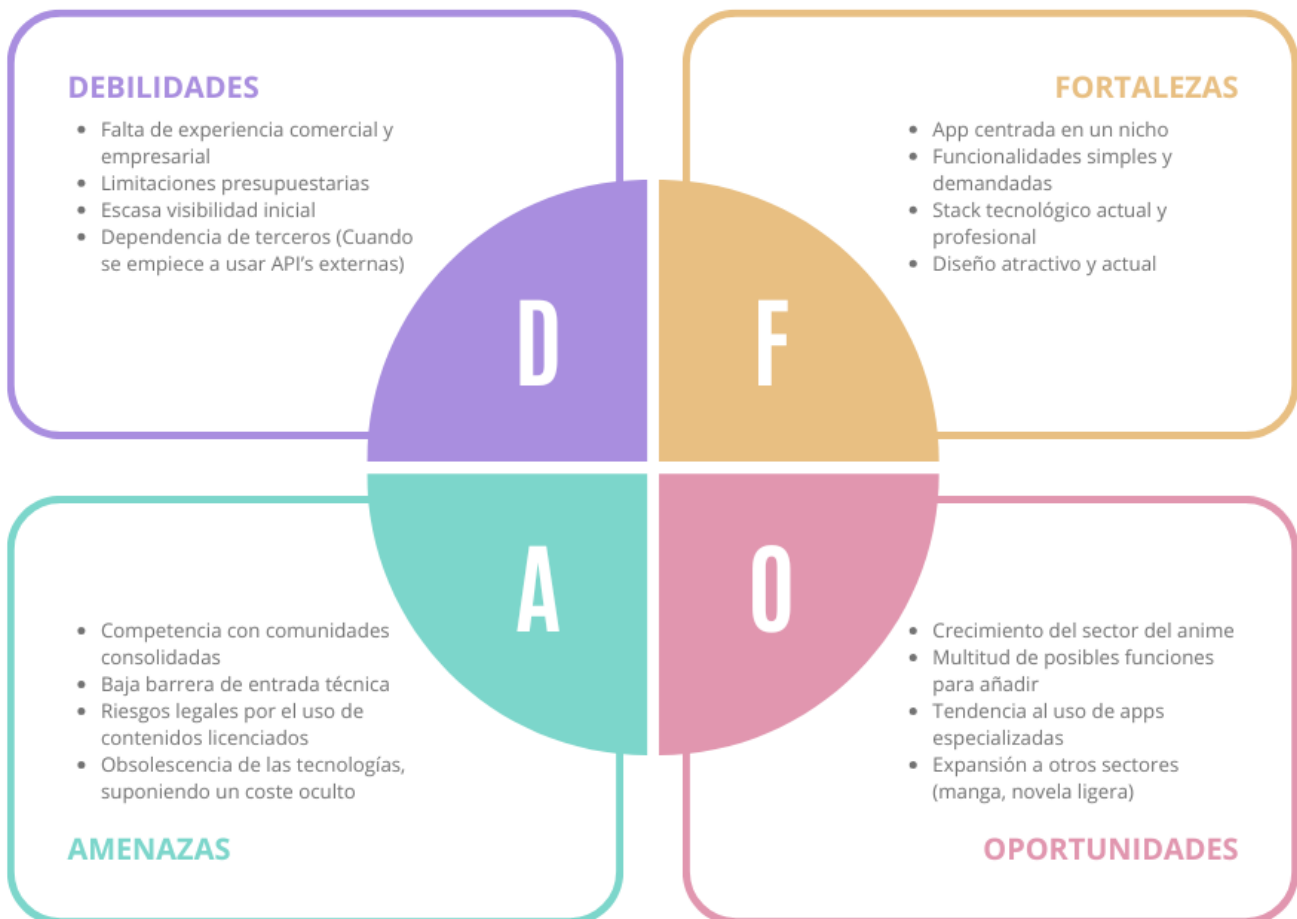


# 14) PLANIFICACIÓN TEMPORAL

Diagrama de Gantt



## 15) DAFO



## 16) ANÁLISIS PRESUPUESTARIO

### Actividades clave

La siguiente tabla detalla las actividades clave del proyecto, con una breve descripción, responsables y frecuencia estimada.

Desarrollo de la aplicación móvil y su mantenimiento
Desarrollo de la aplicación de escritorio
Desarrollo del backend y su integración en el frontend
Gestión de la base de datos
Diseño de la UI/UX
Testing y aseguramiento de calidad
Marketing digital
Soporte al usuario
Gestión técnica
Administración y finanzas

## Plan de inversiones

La siguiente tabla recoge las inversiones necesarias, su precio de adquisición, vida útil estimada, valor residual y amortización anual correspondiente.

ACTIVO NO CORRIENTE				
Elementos del inmovilizado	Precio de adquisición	Vida útil	Valor residual	Amortización anual *
Local (remoto)	0€	∞	0€	0€
Mobiliario (3 sillas y 3 escritorios)	1.050€	10	50€	100€
Equipos informáticos (3 pcs y 5 portátiles)	9.600€	4	0€	2.400€
Teléfonos para pruebas (2 uds)	600€	3	0€	200€
Monitores (4 uds, uno extra para el seguimiento de los procesos)	650€	5	0€	80€
Aplicaciones informáticas	450€	1	0€	450€
Dominio y host	210€	1	0€	210€
<b>TOTALES</b>	12.560€			3440€

## Costes fijos

La siguiente tabla muestra un presupuesto de costes fijos mensuales para el primer año de operación.

Costes fijos	Importe mensual
Gasto en sueldos (trabajadores y socios)	9.100€
Gasto en seguridad social (trabajadores y socios)	3.030€
Suministros	150€
Seguros	50€
Asesoría contable/legal	100€
Gastos financieros (cuota del préstamo)	566€
Gasto en promoción y marketing	300€
Hosting y servidores	100€
Material oficina	50€
<b>TOTAL</b>	<b>13.446€</b>

## Costes variables

Estos costes dependen del volumen de ingresos y/o usuario.

Costes variables	
Comisión tienda de apps (Google Play)	15%
Comisiones de pasarelas de pago (Paypal)	3%
Publicidad basada en afiliados	5%
Otros costes variables	2%

## Gasto en sueldos

Puesto	Trabajador o Socio	Sueldo bruto mes	TOTAL ANUAL
Desarrollador Android	Trabajador	2.000€	24.000€
Desarrollador Backend	Trabajador	2.000€	24.000€
Desarrollador de Escritorio	Trabajador	1800€	21.600€
Diseñador UI/UX	Trabajador	1.700€	20.400€
Responsable de Marketing	Trabajador	1.300€	15.600€
Soporte / Administrativo	Trabajador	1.300€	15.600€
CTO	Socio	0€	0€
CEO	Socio	0€	0€
TOTAL		9.100€	120.800€

## Gasto en Seguridad Social

Trabajador	% Cotización mensual	Cuota mensual S. Social	TOTAL ANUAL
Desarrollador Android	30%	600€	7.200€
Desarrollador Backend	30%	600€	7.200€
Desarrollador de Escritorio	30%	540€	6.480€
Diseñador UI/UX	30%	510€	6.120€
Responsable de Marketing	30%	390€	4.680€
Soporte / Administrativo	30%	390€	4.680€
CTO	0%	0€	0€
CEO	0%	0€	0€
TOTAL		3.030€	36.360€

## 17) MEJORAS A FUTURO

A pesar de que, en mi opinión, el proyecto ha alcanzado las metas que tenía en mente y una funcionalidad sólida, también quiero mencionar algunas mejoras e implementaciones aspiro a incorporar, pero que por tiempo y/o recursos no me fue posible, o simplemente preferí no hacerlas en este momento. Estas mejoras e ideas son con el principal objetivo de mejorar la experiencia de usuario y ampliar el alcance de la aplicación.

Algunas de estas mejoras son:

- Uso de HTTPS en lugar de HTTP, para garantizar una conexión segura y proteger la información de los usuarios.
- Integrar una API externa o ampliar la propia para obtener datos más completos.
- Crear un apartado específico para las categorías de los animes, en lugar de mostrarlas como texto simple, permitiendo así filtrar los animes y obtener una mejor organización.
- Desarrollar múltiples estados para los animes que el usuario sigue, más allá de favoritos. Por ejemplo: terminados, en espera, dropeados, etc.
- Mejorar la interfaz de usuario para hacerla más atractiva e intuitiva en distintos dispositivos.
- Optimizar la aplicación para llegar a más versiones de Android.
- Desarrollar una página web complementaria que funcione junto con la aplicación móvil para brindar una experiencia más completa y accesible.
- Utilizar Firebase para la gestión de autenticación, base de datos y notificaciones.
- Sección dedicada a noticias diarias del mundo del anime, manteniendo a los usuarios informados sobre lanzamientos, eventos y novedades.
- Ampliar el catálogo para incluir no solo animes, sino también mangas y novelas ligeras, ofreciendo un catálogo más completo y diverso.
- Organizar las series de anime y sus contenidos relacionados, separándolos por temporadas y uniendo sus mangas o novelas ligeras correspondientes para facilitar el acceso y la navegación.
- Cambiar el idioma de programación a Kotlin y así en un futuro llegar a IOS.



- Descarga de contenidos para utilizarlos en modo offline.
- Creación de grupos y foros para fomentar la interacción social.
- Implementación de avatares para fotos de perfil.
- Calendario de simulcasts y próximos lanzamientos.
- Mayor personalización de notificaciones.
- Sección de comentarios en cada serie.
- Solucionar la generación de reports en .jar
- etc...

## **18) CONCLUSIONES**

Llevar a cabo este proyecto ha sido un reto importante, pero también una gran oportunidad para poner en práctica todo lo aprendido. La experiencia ha sido muy completa, ya que he tenido que tocar cada parte del desarrollo del proyecto, haciéndome una idea de cómo se trabaja en cada área.

Además, este proyecto me sirvió para entender mejor la relación entre el cliente y el servidor, y la importancia de que todo esté bien coordinado para que funcione como debe. También me ayudó a ser más ordenado en el desarrollo y a tener en cuenta detalles que antes podía pasar por alto, como la gestión de errores o la experiencia del usuario.

Durante el desarrollo surgieron algunos problemas, sin embargo, poder resolverlos me dio confianza en mis propias capacidades.

En resumen, este proyecto me ha servido no solo para fortalecer mis conocimientos, sino también para crecer como desarrollador. He aprendido a ser más autónomo, a planificar mejor y a ver el desarrollo de software desde una perspectiva más completa.

## 19) BIBLIOGRAFÍA

### Github:

→ <https://github.com>

### ChatGPT:

→ <https://chatgpt.com>

### MVN Repository:

→ <https://mvnrepository.com>

### Spring Initializr:

→ <https://start.spring.io>

### Diseño:

→ FlatLaf

◆ <https://www.formdev.com/flatlaf/components>

◆ <https://www.formdev.com/flatlaf/properties-files>

→ Material Design

◆ <https://m3.material.io>

### Uso dispositivo vía USB:

→ Android

◆ <https://developer.android.com/studio/run/device?hl=es-419>

→ ADB

◆ <https://stackoverflow.com/questions/46138780/adb-reverse-tcp-not-working-on-android-connected-remotely>

### General:

→ Transiciones

◆ <https://developer.android.com/develop/ui/views/animations/transitions/start-activity?hl=es-419>

→ Forzar orientación

◆ <https://stackoverflow.com/questions/79096836/androidscreenorientation-portrait-is-deprecated>

→ SplashScreen

◆ <https://developer.android.com/develop/ui/views/launch/splash-screen?hl=es-419#java>

- Api REST
  - ◆ <https://biblus.us.es/bibing/proyectos/abreproy/92380/fichero/TFG-2380-MONAGO.pdf>
- Deshabilitar BottomNavigationView
  - ◆ <https://stackoverflow.com/questions/46539871/how-to-disable-bottomnavigationview-click-and-touch>
- Notificaciones
  - ◆ <https://developer.android.com/develop/ui/views/notifications/build-notification?hl=es-419>
- JOptionPane con JPasswordField
  - ◆ <https://stackoverflow.com/questions/9924289/hide-data-using-joptionpane>
- Number Picker
  - ◆ <https://sazib.medium.com/horizontal-number-picker-android-704bdd868e36>
- Desactivar vibración BottomNavigationView
  - ◆ <https://stackoverflow.com/questions/4670130/android-disable-default-vibration-of-onlongclick>
- Quitar tooltip BottomNavigationView
  - ◆ <https://stackoverflow.com/questions/57688720/disable-tooltiptext-in-bottomnavigationview>
- Cambios e idioma deprecated
  - ◆ <https://stackoverflow.com/questions/56057127/language-not-changing-in-app-android-studio>
- onBackPressed deprecated
  - ◆ <https://stackoverflow.com/questions/72634225/onBackPressed-is-deprecated-what-is-the-alternative>
- JAR with dependencies
  - ◆ <https://stackoverflow.com/questions/574594/how-can-i-create-an-executable-runnable-jar-with-dependencies-using-maven>

## 20) REPOSITORIO

Github: <https://github.com/Eloycorne3/Proyecto>