
# Documentación

Rocío Palao
Mohamed Asidah
Joaquín Ayllón
Roberto Blázquez
Fran Toribio

# Tabla de contenido


Documentación ..... 1

Introducción ..... 3

Objeto del proyecto ..... 3

Requisitos iniciales ..... 3

Requisitos funcionales ..... 1

Requisitos de información ..... 1

Descripción de la solución ..... 1

Organización y gestión del proyecto ..... 2

Actores ..... 2

Estructura interna ..... 2

Roles y responsabilidades ..... 2

Gestión del proyecto ..... 3

Planificación temporal ..... 3

Evolución del plan ..... 3

Evaluación ..... 3

Resumen del presupuesto ..... 4

Análisis y diseño del sistema ..... 4

Estimación de tamaño y esfuerzos ..... 4

Planes de Gestión del proyecto ..... 5

Gestión de plazos ..... 5

Gestión de costes ..... 5

Gestión de calidad ..... 5

Gestión de recursos humanos ..... 5

Gestión de riesgos ..... 5

Especificaciones del sistema ..... 5

## Introducción

**AnimeList**, es un proyecto educativo de software desarrollado por alumnos del IES Luis Vives de Leganés del Ciclo Formativo de Grado Superior “Desarrollo de Aplicaciones Multiplataforma”.

Este proyecto forma parte del resultado de aprendizaje del módulo de Programación. Pretende fomentar y evaluar el trabajo en equipo y poner en práctica todo lo aprendido durante el curso.

Los requisitos técnicos mínimos del proyecto y los contenidos básicos están fijados por **José Luis González Sánchez**, profesor del ciclo y tutor del grupo.

Se pide simular cómo sería un desarrollo de software en un entorno laboral real aplicando técnicas actualmente demandadas por el mercado.

La aplicación pretende emular una comunidad virtual de catalogación de anime y manga. Proporciona a sus usuarios un sistema basado en listas para organizar y puntuar los animes elegidos.

## Objeto del proyecto

Con la realización del proyecto se pretende conseguir los requisitos mínimos exigibles a un trabajo asignado al equipo.

## Requisitos iniciales

Los requisitos iniciales, fijados por el profesor, se basan en las buenas prácticas aprendidas en el módulo Entornos de Desarrollo.

Se deben seguir las fases del desarrollo de una aplicación (Análisis, Diseño, Codificación, Pruebas, Documentación, Explotación y Mantenimiento).

En el desarrollo de la aplicación se usa el IDE de la empresa Jet Brains ***INTELLI J***, con licencias educativas conseguidas por el responsable del departamento de informática

del centro y profesor de los grados de Entornos de Desarrollo y Programación, José Luis González Sánchez.

Para la correcta comunicación del equipo y el buen desarrollo conjunto se trabaja con **GitHub**, servicio basado en la nube que aloja un sistema de control de versiones (VCS) llamado Git. Éste nos permite a los desarrolladores del equipo colaborar y realizar cambios en el proyecto compartido, a la vez que mantenemos un seguimiento detallado de su progreso.

Al comienzo de cada día, se debe realizar una pequeña reunión para describir el trabajo realizado del día anterior y qué tareas vamos a adjudicarnos para la jornada.

La comunicación entre miembros del equipo se realiza principalmente con Discord, que es un servicio de mensajería instantánea freeware de chaz de voz VoIP, video y texto.

Para el reparto y seguimiento de tareas usamos una metodología basada en CANVAS que consiste literalmente en asignar tareas en post-it's, pegados a una cartulina, e ir pasándolas de columna. El tablón consta de cinco columnas:

**Backlog**, tareas por realizar.
**To Do**, tareas para realizar.
**In Progress**, tareas realizándose.
**Testing**, código en fase de test.
**Done**, tareas realizadas.

A parte del tablón físico ubicado en el aula de la clase, hacemos un seguimiento virtual con el software de administración de proyectos **Trello** con interfaz web.

En el desarrollo de la aplicación seguimos las siguientes tablas de requisitos funcionales y de información.

## Requisitos funcionales

| ID | TIPO | FUENTE | PRIORIDAD | DESCRIPCION | VERSION | COMPLEJIDAD | PRUEBAS | ACEPTACION |
|---|---|---|---|---|---|---|---|---|
| RPA1 | Producto como Admin | Cliente | Media | Crear nuevo anime | | Media | Crear un nuevo anime | Poder crearse un anime nuevo en el sistema |
| RPA2 | Producto como Admin | Cliente | Media | Eliminar anime existente | | Baja | Eliminar un anime existente | Eliminar del sistema un anime seleccionado |
| RPA3 | Producto como Admin | Cliente | Media | Eliminar usuario existente | | Media | Eliminar un usuario ya existente | Poder eliminar un usuario del sistema |
| RPA4 | Producto como Admin | Cliente | Media | Modificiar animes existentes | | Alta | Modificar un anime ya existente | Poder cambiar los datos de un anime |
| RPA5 | Producto como Admin | Cliente | Media | Modificar usuarios existentes | | Alta | Modificar los datos de un usuario | Poder modificar un usuario que ya existe |
| RP1 | Producto | Cliente | Alta | Debe existir un usuario admin | | Media | Poder logearse como admin | El admin podrá hacer las tareas propias de un administrador |
| RP2 | Producto | Cliente | Alta | Login de usuario | | Media | Login con usuario ya existente | Poder iniciar sesión con un |

| ID | TIPO | FUENTE | PRIORIDAD | DESCRIPCION | VERSION | COMPLEJIDAD | PRUEBAS | ACEPTACION |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | usuario ya existente |
| RP3 | Producto | Cliente | Alta | Sign-up de usuarios | | Alta | Crear un nuevo usuario | Se debe poder crear nuevos usuarios en el sistema |
| RP4 | Producto | Cliente | Alta | Añadir animes a la lista de vistos | | Baja | Agregar animes a la lista del usuario | Poder agregar cualquier tipo de anime |
| RP5 | Producto | Cliente | Media | Escribir reseñas sobre los animes | | Baja | Escribir una reseña sobre un anime | Poder escribir una reseña de un anime que tenga en la lista el usuario |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| RP6 | Producto | Cliente | Alta | Darle una puntuación numérica a un anime | | Baja | Introducir una puntuación para un anime | Poder puntuar en el rango adecuado un anime en la lista del usuario |
| RP7 | Producto | Cliente | Alta | Listar todos los animes de la base de datos | | Alta | Mostrar todos los animes | Poder listar todos los animes que tenemos |
| RP9 | Producto | Cliente | Baja | Limitar el numero de animes mostrados y dividirlos en páginas | | Alta | Listar el número de animes en varias páginas | No se muestran todos los animes en una tanda |
| RP10 | Producto | Cliente | Media | Filtrar por géneros seleccionados por el usuario | | Alta | Solo se muestran los animes de los géneros seleccionados | Mostrar los géneros de uno en uno |
| RP11 | Producto | Cliente | Alta | Seleccionar un anime para mostrar sus datos ampliados | | Media | Mostrar en una pantalla a parte los datos de un anime | Solo mostrar los datos de un anime |
| RP12 | Producto | Cliente | Media | Mostrar mi lista de animes | | Media | Mostrar los animes del usuario | Se muestran correctamente los animes que el usuario ha guardado |
| RP13 | Producto | Cliente | Alta | Modificar la lista del usuario | | Media | Cambiar el contenido de la lista del usuario | Poder eliminar y volver a añadir animes |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| RP14 | | | Alta | Mostrar estadísticas de toda la colección (especificados en los requisitos de información) | | Media | Las estadisicas coinciden con los datos | Al menos el top 10 |
| RP15 | | | Baja | Generar un informe de usuario en HTML con toda su lista | | Alta | Generar el informe de un usuario | Al menos mostrar los nombres y la puntuación que le dio |
| RP16 | Producto | Cliente | Alta | Buscar un anime ingresando su nombre en inglés o escrito con alfabeto romano | | Alta | Buscar un anime y que solo muestre ese | Al buscar un anime no se muestran otros |
| RP17 | Producto | Cliente | Alta | Eliminar anime de la lista del usuario | | Media | Eliminar un anime de la lista | Poder eliminar animes de la lista |
| RI1 | Implementación | Equipo | Alta | Desarrollar el programa usando Java 17 y Kotlin. | | Media | | |
| RI2 | | | Alta | Guardar los datos en una base de datos SQLlite | | Media | Mantener los datos de una sesión a otra | Todos los datos de una sesión se guardan |
| RI3 | | | Alta | Exportar los datos de animes a un JSON | | Media | Exportar los datos de los animes creados a un JSON | Se puede exportar la base de animes |
| RI4 | | | Alta | Exportar los datos de usuarios a un JSON | | Media | Exportar los datos de los usuarios a un JSON | Se puede exportar la tabla de usuarios |

# Requisitos de información

La información se recopila de la base de datos en tiempo de ejecución y, al acabar el programa, se actualiza la base de datos y se genera un fichero JSON con todos los datos.

Usuarios.

Los datos de los usuarios serán:

| DATO | TIPO | RESTRICCION |
|---|---|---|
| Nombre de usuario | Cadena de caracteres | Mínimo 4 caracteres máximo 20. Único |
| Contraseña | Cadena de caracteres hasheada | Mínimo 8, máximo 20, una mayúscula mínimo |
| ID | UUID | Único y no accesible al usuario |
| Correo electrónico | Cadena de caracteres | Cumpliendo el standard RFC para direcciones email |
| Fecha de nacimiento | Tipo Date | Formato español: Dia/mes/año |
| Fecha de alta | Tipo Date | Formato español: Dia/mes/año |
| Lista de animes | Tipo lista | Lista de id’s de anime |

Reviews.

Las reviews se almacenaran con el ID del usuario que ha puesto la review y la ID del anime:

| DATO | TIPO | RESTRICCION |
|---|---|---|
| ID | UUID | |
| ID_Anime | UUID | el id de un anime ya existente |
| ID_Usuario | UUID | Unico relacionado con un usuario |
| puntuacion | numero real | del 1 al 10 con un decimal |
| Review | Cadena caracteres | máximo de 500 caracteres |

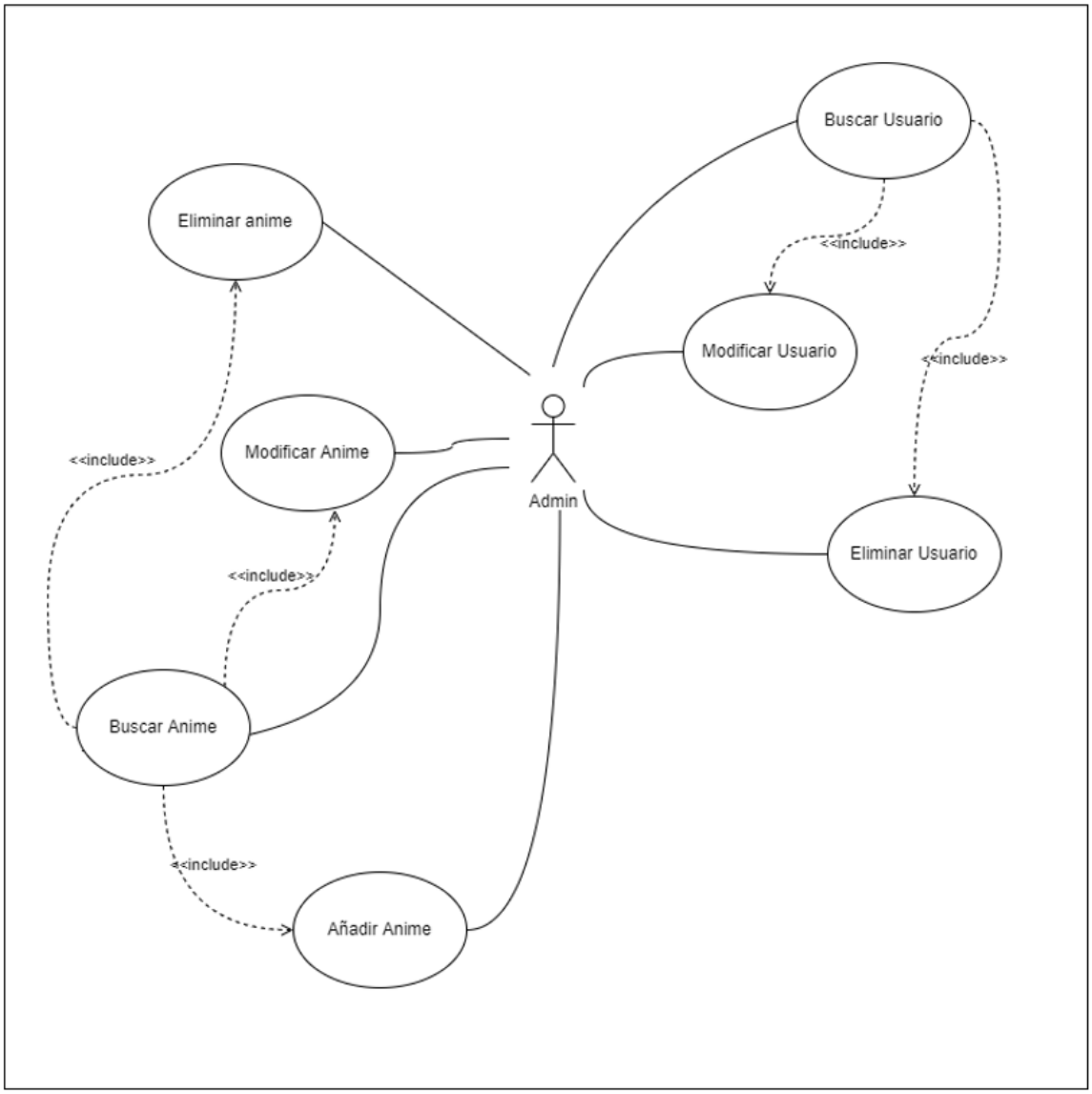
Animes.

Los datos de los animes serán:

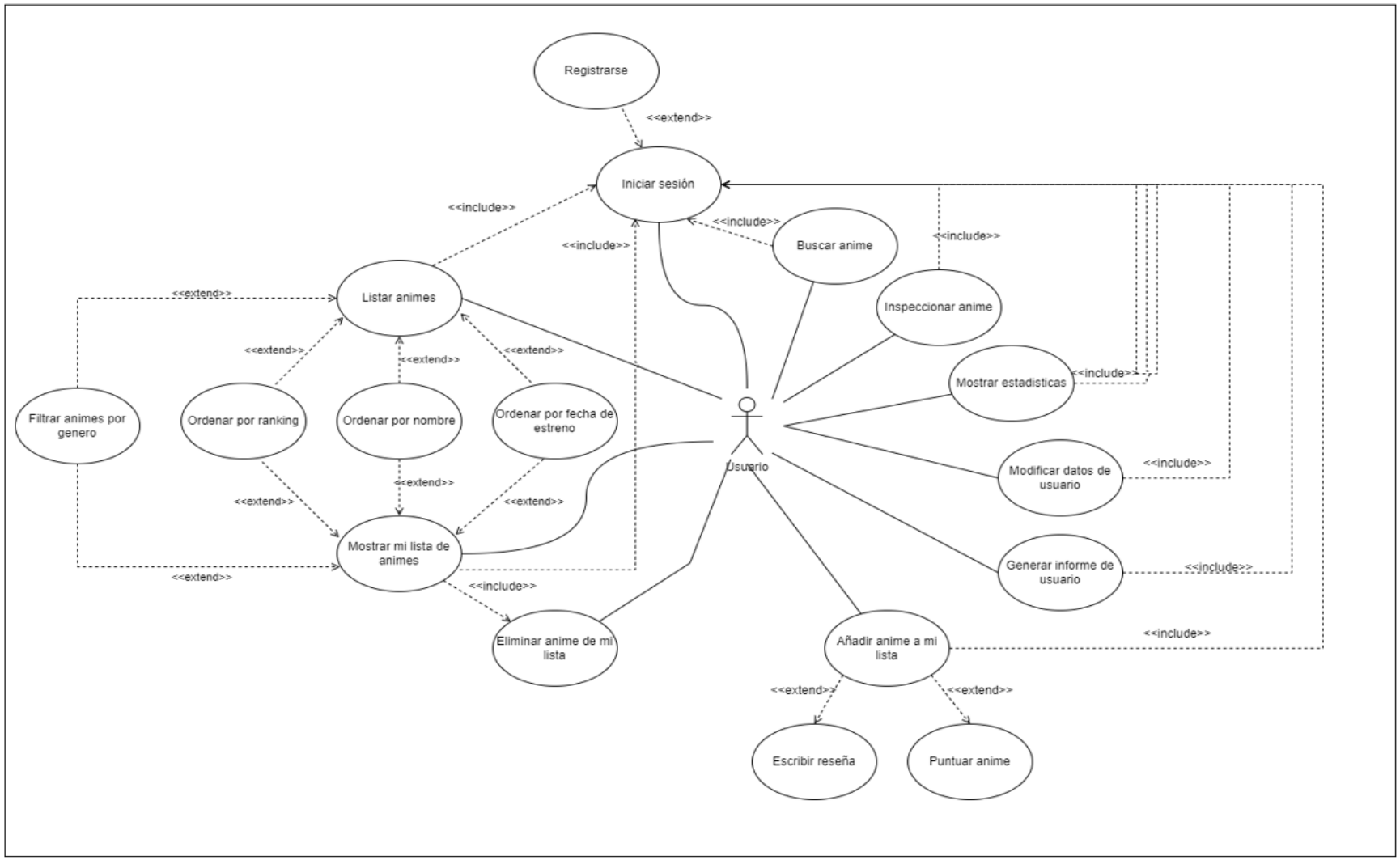
| DATO | TIPO | RESTRICCION |
|---|---|---|
| ID | Numero entero | Mayor que 0 y sin repeticiones |
| Titulo original | Cadena caracteres | No nula |
| Titulo inglés | Cadena de caracteres | Mayor que dos caracteres |
| Tipo | Cadena de caracteres | No nula |
| Numero episodios | Numero entero | Mayor que 0 |
| Estado | Cadena de caracteres | No nula |
| Emisión | Tipo Date | Formato español: Dia/Mes/Año |
| Estreno | Tipo Date | Formato español: Dia/Mes/Año |
| Duración de episodio | Numero entero | Mayor que 0 |
| Calificación por edades | Cadena de caracteres | No nula |
| Puntuación | Numero decimal (double) | Redondeado a 2 decimales. Mayor que 0 |
| Número de usuarios que han puntuado | Numero entero | Numero positivo |
| Ranking | Numero entero | Mayor que 0 y como máximo el tamaño de la tabla |
| Popularidad | Numero entero | Mayor que 0 |
| Genero | Lista de cadenas de caracteres | Mínimo una cadena de caracteres |

# Descripción de la solución

Casos de uso del administrador



Casos de uso de usuario



**Modelos:**

## Anime:

Los animes son el producto de nuestro programa, contienen entre otras cosas, el nombre del anime, la fecha de publicación, su nombre en inglés, el género, cuántos episodios tiene.

## Usuario:

En nuestro programa tenemos usuarios que para utilizar dicho programa, deben registrarse, cada usuario puede tener una lista con sus animes e incluso puede añadirle puntuación y/o comentario al anime que quiera de su lista.

## Review:

Las reviews contienen las opiniones de los usuarios sobre un anime en concreto, los usuarios pueden hacer reviews de los animes que tengan añadidos en su lista

**Repositorios:**

## AnimeList Repository:

Es el repositorio encargado de guardar los animes que está siguiendo un usuario.
Utiliza la interfaz ICRUDAnimeList, lo único que guardamos en este repositorio son los identificadores únicos del anime a añadir y del usuario que está logueado.

## Anime Repository:

Repositorio encargado del manejo de los animes que se salvaran en la base de datos, utiliza la interfaz de CRUDRepository ya que implementa todas las utilidades básicas de un CRUD y añade una búsqueda por título en los animes. Utiliza el DatabaseManager ofrecido por Jose Luis como dependencia para manejar la base de datos.

## Reviews Repository:

Es el repositorio encargado del manejo de las reviews y puntuaciones creadas por un usuario sobre un anime, al igual que el repositorio anterior utiliza una interfaz, IRepositoryReview que implementa todas las funciones de este repositorio.

## User Repository:

Es el repositorio encargado del manejo de los usuarios, tanto administradores como usuarios normales, utiliza la interfaz CRUDRepository ya que al igual que anime repository implementa todas las utilidades básicas de un CRUD.

**Patrones utilizados.**
En el proyecto hemos utilizado:
Patrón Fachada, ya que los usuarios sólo ven la vista, no ven todos los procesos que van ocurriendo por detrás. Implementado con todas las interfaces.
Patrón Singleton, hemos usado este patrón para que se nos devuelva siempre la misma instancia de una clase. Implementado en el DataBaseManager y en el DependenciesManager.
Patrón Observer, utilizado para los cambios que pueden ocurrir al borrar, crear o cambiar un elemento. Implementado en los repositorios del proyecto.

**Técnicas SOLID**
Principio de responsabilidad única
Principio de abierto/cerrado
Principio de sustitución de Liskov
Principio de segregación de Interfaces: La hemos utilizado en el proyecto ya que cada interfaz sólo implementa sus métodos necesarios.
Principio de inversión de dependencias: Utilizada para las clases, ya que dependen de las abstracciones.

## Organización y gestión del proyecto

## Actores

## Estructura interna

## Roles y responsabilidades

El equipo de trabajo está formado por una **Jefa de Proyecto***, Rocío Palao*, un **Product Owner**, *Mohamed Asidah*, dos **Programadores Junior** *Roberto Blázquez* y *Joaquín Ayllón* y un **Becario** *Francisco Toribio.*

Como Jefa de Proyecto, Rocío Palao es la encargada de la orientación en el reparto de tareas y la comunicación directa con el tutor.

Mohamed Asidah, Product Owner, se asegura de que el desarrollo del proyecto se ciña fielmente a los requisitos preestablecidos.

Todos los miembros del equipo se responsabilizan de implementar un código de calidad, comentado, testado, abierto a ampliaciones y mantenible en el tiempo.

Cada integrante del equipo tiene acceso a la modificación del código siempre para su mejora y aporte de soluciones. También se realizan aportaciones a la documentación para la correcta descripción del trabajo.

## Gestión del proyecto

### Planificación temporal

El plazo de consecución y entrega del proyecto es de 30 días naturales que acabarán el Martes 31 de Mayo.

En la primera semana del proyecto se exige tener los diagramas de casos de uso y los prototipos de las vistas.

En la segunda semana del proyecto deben estar implementados los CRUD del programa y realizados tanto los repositorios como la base de datos. Al final de la semana tendrá lugar una reunión entre la Jefa de Proyecto y el tutor.

Al inicio de la tercera semana habrá una evaluación de consecución de objetivos por parte del tutor.

### Evolución del plan

Los plazos exigibles se han ido cumpliendo rigurosamente, no exentos de dificultades, aportando las soluciones a los retos tecnológicos que van apareciendo según crece la aplicación.

### Evaluación

Tras una primera evalución intermedia en la tercera semana de proyecto se detectan por parte del tutor y CEO del proyecto los siguientes defectos en la aplicación:

- Fallo al registrarse -> El programa no lanza mensaje de error al no introducir ningún dato.

- Fallo al no cambiar la foto de perfil -> El programa no da la posibilidad de cambiar la foto de perfil del usuario en su vista.
- Fallo en la vista principal del usuario -> El programa no muestra la tabla de animes seleccionados en la vista principal del usuario.
- Fallo en la usabilidad e interpretación de usuario -> Falta de información en los labels de algunos registros.

Faltando 10 días para la evaluación definitiva, el equipo continúa con la corrección de errores y completando el resto de implementaciones pendientes.

## Resumen del presupuesto

Al ser un Proyecto educativo, AnimeList no tiene dotación económica por lo que se hará una estimación de los costes.

Consideramos el equipo de informáticos de AnimeList como una pequeña empresa de software formada por sus cinco integrantes. La empresa va a dedicar todos los recursos en exclusiva al proyecto durante el plazo de un mes.
Simulando costes de mercado, podríamos añadir a la contabilidad del proyecto los siguientes gastos:

- Equipos informáticos
- Sueldos y salarios
- Cargas sociales
- Beneficio empresarial
- Amortizaciones
- Depreciaciones
- Propiedad intelectual
- Obligaciones a largo plazo
- Deudas con entidades de crédito
- Gastos financieros
- Impuestos corrientes
- Tributos
- Otros gastos de gestión corriente
- Impuestos diferidos

## Análisis y diseño del sistema

## Estimación de tamaño y esfuerzos

Planes de Gestión del proyecto

Gestión de plazos

Gestión de costes

Gestión de calidad

Gestión de recursos humanos

Gestión de riesgos

Especificaciones del sistema

.