AnimeFetch

Manual técnico

Alumno: Jorge Pacheco Castro
Módulo: Programación Multimedia y Dispositivos Móviles
Curso: 2º CFGS DAM 24/25

Índice

1. Introducción

Justificación de la necesidad del proyecto:

En la actualidad, el consumo de contenido de anime ha crecido exponencialmente a nivel mundial. Los usuarios buscan aplicaciones y plataformas que centralicen todo tipo de anime y les permitan acceder a su contenido favorito de manera rápida y eficiente. Sin embargo, existen ciertas limitaciones en las aplicaciones actuales, como la dispersión de catálogos, la falta de una interfaz intuitiva y la ausencia de opciones personalizadas que mejoren la experiencia de usuario.

Este proyecto tiene como objetivo ser un **directorio de anime**, donde el usuario puede estar al tanto de sus animes favoritos, pero también descubrir nuevos a través de recomendaciones, consultar los más populares o esperados y, si aún no ha quedado satisfecho, realizar búsquedas personalizadas de manera sencilla. La propuesta está dirigida a los amantes del anime, desde novatos hasta fanáticos acérrimos, que buscan una plataforma en la que puedan acceder a su contenido favorito y gestionar sus notas (seguimiento de episodios, impresiones generales, etc.) de una manera eficiente.

El software proporciona opciones como la selección de idioma para títulos (inglés/japonés), tema adaptativo (claro u oscuro) y el sistema de gestión de notas mencionado anteriormente.

Propósito del software:

AnimeFetch ofrece una plataforma unificada, donde los usuarios pueden informarse sobre todo tipo de anime, cambiar la apariencia de la app, plasmar sus impresiones y disfrutar de una experiencia fluida y agradable.

2. Análisis del problema

a. Problemática

Los usuarios de anime se enfrentan a varios problemas en las plataformas actuales:

1. **Fragmentación de contenido**: Los usuarios a menudo se ven obligados a usar múltiples plataformas para acceder a su contenido favorito de anime, lo que genera una experiencia de usuario desconectada y dispersa.
2. **Interfaz poco amigable**: Muchas aplicaciones no son intuitivas o fáciles de navegar, lo que puede hacer que los usuarios abandonen la plataforma o tengan dificultades para encontrar contenido de manera eficiente.
3. **Poca personalización**: Muchas aplicaciones de anime no permiten personalizar la experiencia según las preferencias del usuario.
4. **Escasez de funcionalidades**: Muchas aplicaciones de anime no permiten importar listas o integrar recomendaciones basadas en los hábitos de consumo del usuario. Los usuarios a menudo tienen que realizar un seguimiento manual de qué episodios o series están viendo.

AnimeFetch busca resolver estos problemas al ofrecer una solución centralizada, fluida, personalizable y accesible, con el fin de mejorar la experiencia general de los usuarios de anime.

b. Clientes potenciales

Los clientes potenciales para **AnimeFetch** son principalmente:

1. **Fanáticos del anime**: Personas apasionadas por el anime, que consumen contenido de manera regular y buscan una plataforma eficiente para descubrir y seguir sus series favoritas.
2. **Usuarios novatos**: Aquellos nuevos en el mundo del anime que están buscando una plataforma accesible y fácil de usar para comenzar su viaje en el universo del anime.
3. **Usuarios que buscan personalización**: Personas que valoran la posibilidad de personalizar su experiencia (cambio de temas, idioma de títulos, notas sobre animes, etc.).
4. **Millennials y Generación Z**: Estas generaciones son las más propensas a consumir contenido de anime y están habituadas a la tecnología móvil. Suelen buscar soluciones accesibles en sus dispositivos móviles.

c. Análisis DAFO

DAFO es una herramienta que permite analizar las fortalezas, debilidades, oportunidades y amenazas de un proyecto o empresa. A continuación, se aplica al análisis de **AnimeFetch**:

Fortalezas	Debilidades
• Plataforma centralizada. • Personalización. • Interfaz intuitiva. • Transiciones suaves.	• Dependencia de licencias. • Necesidad de alianzas con proveedores de contenido.
Oportunidades	**Amenazas**
• Crecimiento global del anime. • Expansión a otros dispositivos.	• Competencia. • Regulaciones y derechos de autor.

d. Monetización y beneficios

Modelo de monetización:

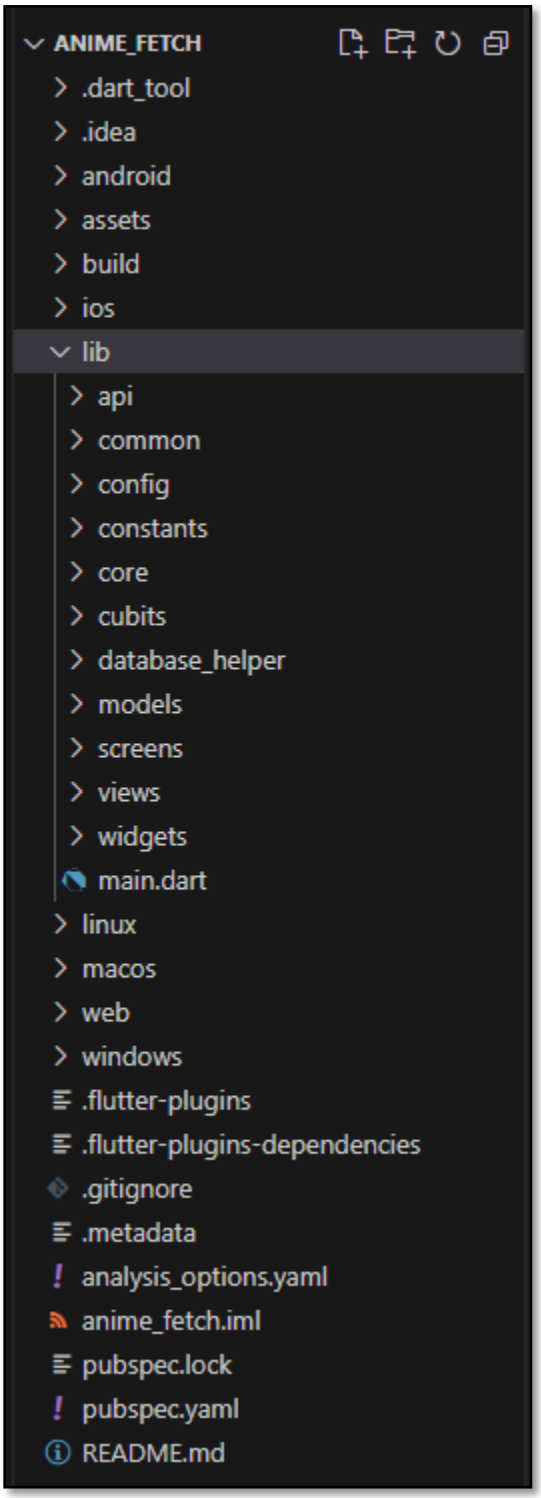
- **Publicidad**: Incluir anuncios dentro de la aplicación. Los anuncios pueden ser de diferentes tipos, como banners o anuncios en video.
- **Afiliados y colaboraciones**: Colaborar con marcas de merchandising de anime, estudios de animación o plataformas de eventos para generar ingresos adicionales mediante promociones, anuncios o productos relacionados con el anime.
- **Ventas dentro de la app**: Ofrecer productos relacionados con el anime, como mangas, figuras coleccionables, entradas para eventos, etc.

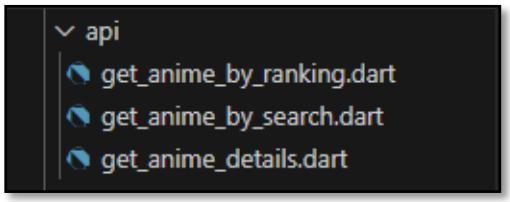
Beneficios:

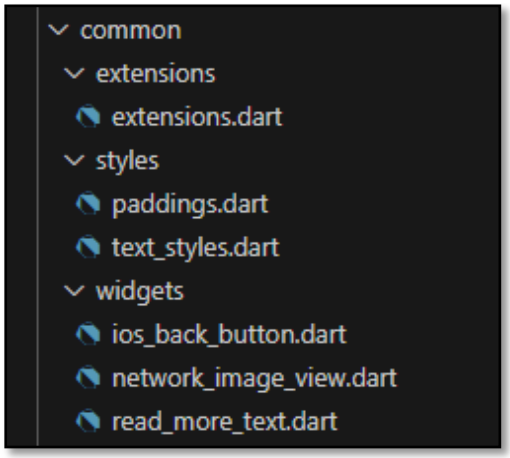
- **Para los usuarios**: Una experiencia centralizada, personalizada y de fácil acceso para seguir todo su contenido de anime. Además, el acceso a una plataforma que entiende sus gustos y preferencias.
- **Para los desarrolladores**: Una oportunidad de negocio viable en un mercado en crecimiento, con múltiples formas de generar ingresos.
- **Para los proveedores de contenido**: AnimeFetch podría ofrecerles un canal adicional para distribuir su contenido a una audiencia global.

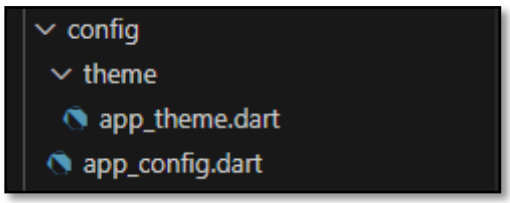
3. Diseño de la solución

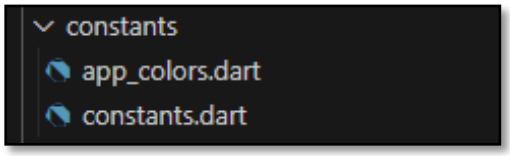
a. Estructura del proyecto

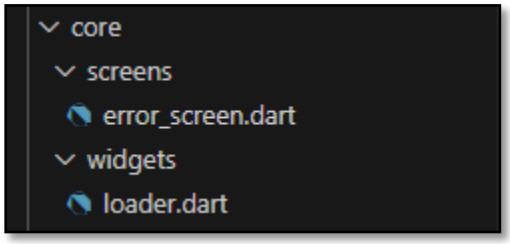


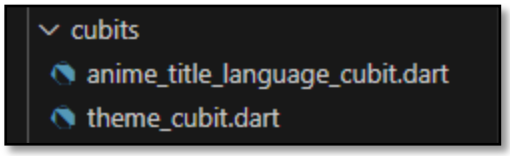


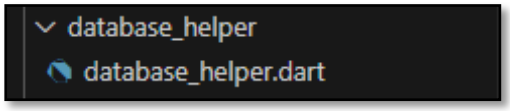


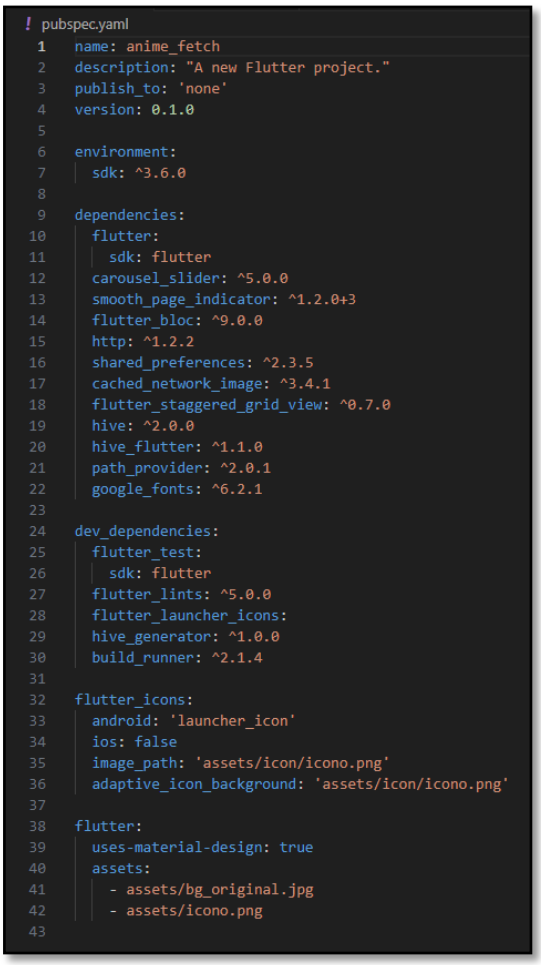


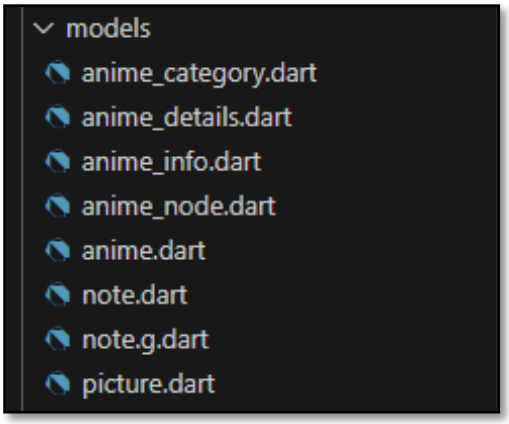


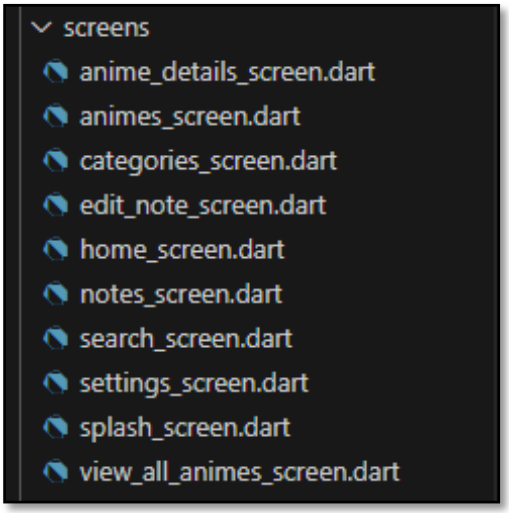


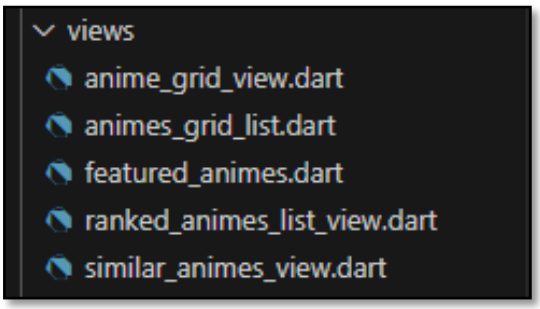


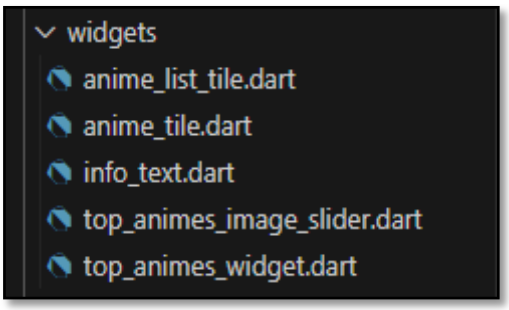












b. Explicación del código realizado

api	Se encarga de gestionar la comunicación con fuentes externas para obtener información sobre animes. Actúa como un módulo de conexión con la API, gestionando las solicitudes HTTP y el procesamiento de respuestas en formato JSON para integrarlas con la interfaz de la aplicación.
common	Contiene componentes reutilizables y configuraciones esenciales para la aplicación. Se divide en tres submódulos: • extensions: formateo de datos. • styles: configuraciones de diseño.

	• widgets: para la interfaz de usuario. Organiza los elementos reutilizables del proyecto, mejorando la escalabilidad y el mantenimiento del código.
config	Centraliza la configuración global de la aplicación, asegurando coherencia en el diseño y comportamiento del sistema. Se divide en dos submódulos principales: • app_theme: gestiona los temas de la aplicación. • app_config: contiene la clave de API. Mantiene la organización de las configuraciones globales y la estética de la aplicación.
constants	Agrupa valores constantes utilizados en toda la aplicación. Mejora la organización y mantenimiento del código.
core	Encapsula componentes que garantizan una mejor gestión de errores y una transición fluida en la carga de datos. ErrorScreen se encarga de mostrar una pantalla de error cuando ocurre un problema, mientras que Loader define un widget de carga, que muestra un indicador de progreso mientras se realizan operaciones en segundo plano.
cubits	Contiene la lógica de gestión de estado de la aplicación utilizando **Flutter Bloc (Cubit)**. Es responsable de manejar cambios en la configuración de la aplicación, específicamente el idioma de los títulos de anime y el tema visual. • anime_title_language_cubit: ✓ Permite alternar entre distintos idiomas (inglés/japonés) para los títulos de los animes obtenidos desde la API. ✓ Mantiene el estado del idioma seleccionado y notifica a los widgets cuando hay un cambio. ✓ Se utiliza en la configuración de la aplicación para que el usuario pueda elegir su preferencia. • theme_cubit: ✓ Mantiene el estado actual del tema (claro u oscuro). ✓ Permite alternar entre los modos y persiste la selección del usuario para futuras sesiones. ✓ Se utiliza en la configuración para ofrecer una opción de personalización visual.
database_helper	Se encarga de gestionar la configuración y preferencias del usuario a nivel local utilizando **SharedPreferences**, una solución de almacenamiento ligero basada en pares clave-valor.

	Esta clase permite conservar datos importantes, como la preferencia de tema (claro/oscuro) y el idioma de los títulos de anime, incluso después de cerrar la aplicación.
models	Define las estructuras de datos utilizadas en la aplicación. Estas clases representan los distintos tipos de información manejados en la app, como detalles de animes, categorías y notas. Un aspecto clave de este módulo es el uso de **Hive**, una base de datos NoSQL rápida y ligera utilizada para almacenar y gestionar notas de usuario de manera local. • note: ✓ Representa una nota creada por el usuario dentro de la app. ✓ Utiliza HiveObject para permitir el almacenamiento y la manipulación eficiente de notas dentro de la base de datos local de la aplicación. ✓ Permite a los usuarios guardar anotaciones sobre animes, ya sea recomendaciones, episodios pendientes, teorías, etc. • note.g: ✓ Archivo generado automáticamente por Hive para manejar la serialización y deserialización de objetos Note. ✓ Necesario para que Hive pueda leer y escribir objetos Note en la base de datos local. ✓ No debe modificarse manualmente, ya que se genera mediante build_runner.
screens	Contiene las pantallas principales de la aplicación. Estas pantallas representan la interfaz gráfica y la interacción del usuario con las distintas funcionalidades, como la exploración de animes, la gestión de notas y la configuración de preferencias.
views	Contiene una serie de vistas que son responsables de organizar y mostrar los datos de manera visualmente atractiva. Estas vistas se utilizan principalmente para mostrar listas de animes, animes recomendados, animes similares, entre otras representaciones visuales de contenido.
widgets	Contiene varios componentes reutilizables y modulares para la construcción de la interfaz de usuario dentro de la aplicación. Estos widgets permiten que la UI sea más eficiente y organizada, reutilizando piezas de código y asegurando que el diseño sea coherente a través de diferentes pantallas.

	Desde la presentación de listas de animes hasta el carrusel de los animes más destacados, estos widgets contribuyen a una experiencia de usuario fluida y bien estructurada.

c. Problemas encontrados, propuestas de mejora

A lo largo del desarrollo, han surgido diversas **dificultades** relacionadas con la gestión de recursos estáticos (assets), la optimización de la interfaz de usuario o la configuración de dependencias, amén de aquellas relacionadas con la implementación de elementos no vistos en clase como el uso de Hive para la persistencia de datos o la gestión de estados mediante la librería BLoC.

No obstante, también han supuesto la oportunidad de aprender cosas nuevas y volver a refrescar conceptos, al final las cosas van saliendo con tiempo, mucho ensayo y error y, sobre todo, paciencia.

Como **propuestas de mejora**, considero que se podrían añadir (o pulir) ciertas funcionalidades como:

Implementación de Favoritos y Notificaciones: Actualmente, la app no permite marcar animes como favoritos o recibir notificaciones sobre actualizaciones.

¿Cómo se podría mejorar?

- ✓ Agregando una función para guardar animes favoritos usando Hive.
- ✓ Implementando *Firebase Cloud Messaging (FCM)* para notificaciones sobre nuevos episodios o recomendaciones personalizadas.

Optimización de carga de datos: Los datos de animes se cargan cada vez que se entra a una pantalla, lo que puede generar tiempos de espera prolongados.

¿Cómo se podría mejorar?

- ✓ Implementar *caching* con Hive o shared_preferences para almacenar resultados previos y reducir el número de peticiones a la API.
- ✓ Utilizar *lazy loading* en listas para mejorar el rendimiento.
- ✓ Implementar *Pagination* en las listas de animes para no cargar todos los elementos de una vez.

Implementación de un mejor sistema de búsqueda: La búsqueda actual solo funciona con coincidencias exactas y no tiene filtros avanzados.

¿Cómo se podría mejorar?

- ✓ Implementando una búsqueda más avanzada con *Algolia* o *ElasticSearch*[1].
- ✓ Agregando filtros por género, año de lanzamiento, puntuación, etc.

[1] Motores de búsqueda avanzados diseñados para mejorar la velocidad y precisión de búsquedas en aplicaciones y sitios web.

4. Documentación de la solución

Los códigos tanto de la API (API Key) como de la aplicación están subidos a un repositorio en GitHub, para obtener acceso al código simplemente debes tener una cuenta creada en la plataforma, ya que al ser un repositorio de tipo público no es necesario solicitar ningún permiso adicional.

Repositorio con el código de la aplicación AnimeFetch:
https://github.com/jpaccas075/ProgMultimedia2425

5. Enlaces de interés

Mi perfil de GitHub

Documentación Flutter: https://docs.flutter.dev/
Documentación Dart: https://dart.dev/docs

API utilizada para los datos de los animes:
https://myanimelist.net/apiconfig/references/api/v2

Dependencias y plugins utilizados:
https://pub.dev/packages/build_runner
https://pub.dev/packages/cached_network_image
https://pub.dev/packages/carousel_slider
https://pub.dev/packages/flutter_bloc
https://pub.dev/packages/flutter_launcher_icons
https://pub.dev/packages/flutter_staggered_grid_view
https://pub.dev/packages/google_fonts
https://pub.dev/packages/hive
https://pub.dev/packages/hive_flutter
https://pub.dev/packages/hive_generator
https://pub.dev/packages/http
https://pub.dev/packages/path_provider
https://pub.dev/packages/shared_preferences
https://pub.dev/packages/smooth_page_indicator