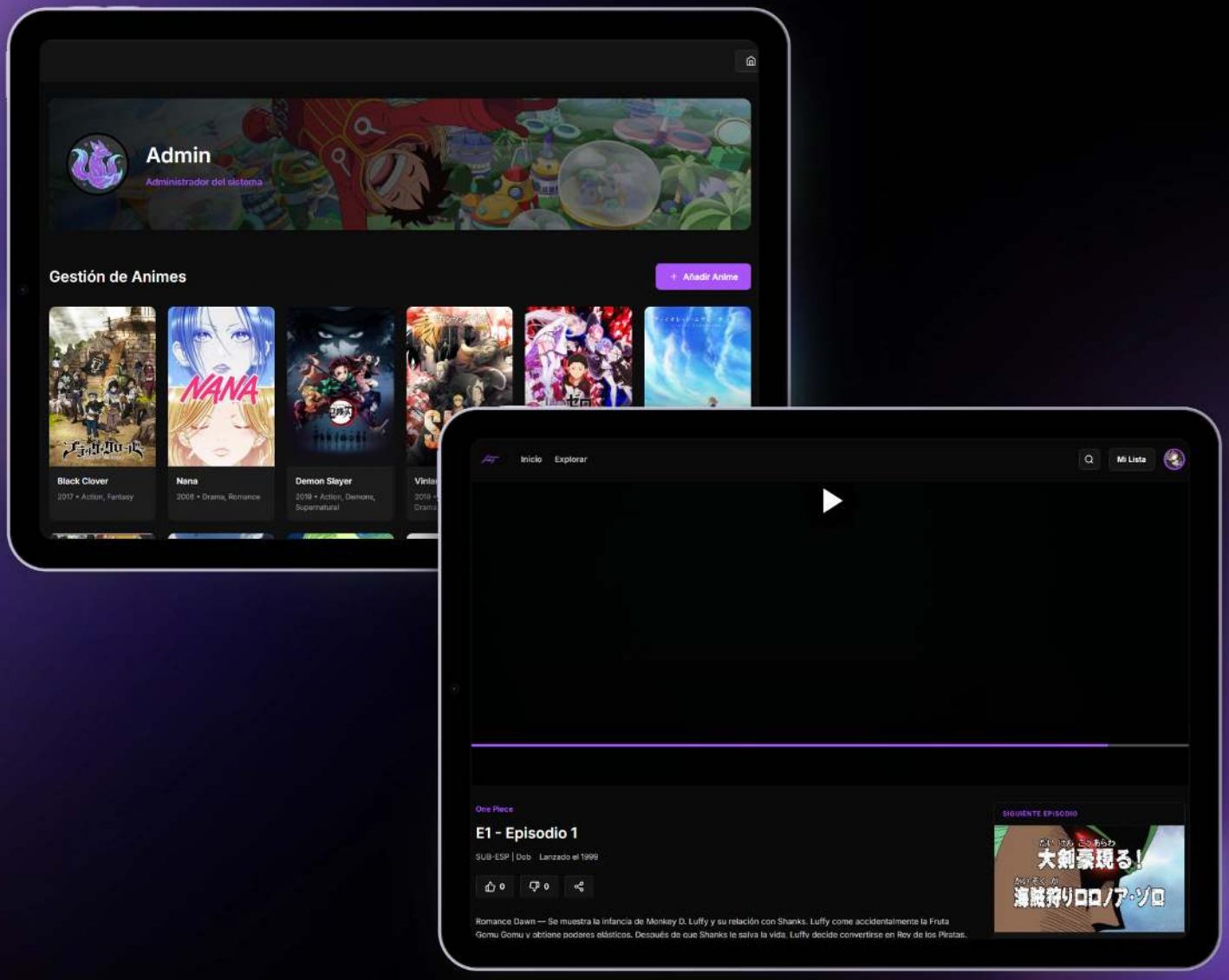

CC BY NC

ANITOKI

CFGS Desenvolupament d'Aplicacions Web
Juan e Izan DAW-2B

Resum del projecte:

El proyecto **AniToki** se centra en el desarrollo de una plataforma web de streaming de anime realizada por dos estudiantes. Su nombre combina "Ani", de anime, y "Toki", que significa tiempo en japonés, haciendo referencia a la posibilidad de acceder al anime en cualquier momento y lugar. La plataforma tiene como objetivo principal centralizar series y películas de anime en un entorno organizado, intuitivo y fácil de usar, mejorando la experiencia de los aficionados que actualmente encuentran el contenido disperso en múltiples fuentes como Crunchyroll, sin acceso completo al catálogo sin suscripción de pago.

El proyecto ofrece una aplicación web accesible que permite a los usuarios registrarse, explorar un catálogo de anime completo y reproducir episodios directamente desde el navegador. Entre los objetivos específicos se incluyen la implementación de un sistema de búsqueda y filtrado eficiente, la creación de listas de seguimiento personalizadas, y la gestión de roles de usuario y administrador mediante un panel de control integrado con dashboard propio.

La metodología seguida se basa en el desarrollo web utilizando **Python** y **Django** en el backend, **Vue.js** como framework de frontend, **PostgreSQL** como base de datos en producción, **Figma** para el diseño de la interfaz, y **Git** y **GitHub** para el control de versiones. El resultado es una plataforma funcional desplegada en producción, que cubre las funcionalidades principales previstas y deja como línea de trabajo futura la parte social y de comunidad.

Paraules clau:

- Anime
- Streaming
- Plataforma web
- Django
- Python
- Vue.js
- Listas de seguimiento
- PostgreSQL

Abstract:

The **AniToki** project focuses on the development of a web-based anime streaming platform built by two students. Its name combines "Ani," from anime, and "Toki," meaning time in Japanese, referring to the ability to access anime anytime and anywhere. The main goal of the platform is to centralize anime series and movies in an organized, intuitive, and user-friendly environment, improving the experience for fans who currently find content scattered across multiple sources such as Crunchyroll, where access to the full catalog requires a paid subscription.

The project provides an accessible web application that allows users to register, explore a complete anime catalog, and stream episodes directly from the browser. Specific objectives include implementing an efficient search and filter system, creating personalized watchlists, and managing user and administrator roles through an integrated control panel with a dedicated dashboard.

The methodology is based on web development using **Python** and **Django** for the backend, **Vue.js** as the frontend framework, **PostgreSQL** as the production database, **Figma** for interface design, and **Git** and **GitHub** for version control. The result is a fully functional platform deployed in production, covering the main planned features, with the social and community features left as future work.

Keywords:

- Anime
- Streaming
- Web platform
- Django
- Python
- Vue.js
- Watchlists
- PostgreSQL

ÍNDICE

1 Introducción 1
- 1.1 Contexto 1
- 1.2 Justificación 1
- 1.3 Objetivos 2
- 1.4 Estrategia y planificación del proyecto 2
- 1.5 Metodología de trabajo 3
- 1.6 Estudi econòmic i pressupostari 4

2 Descripción del proyecto 5
- 2.1 Análisis de requisitos 5
- 2.2 Tecnologías 6
- 2.3 Estructura del proyecto 9
- 2.4 Descripción de los componentes 9
- 2.5 Definición de las funcionalidades 11

3 Otros capítulos 13
- 3.2 Decisiones de diseño 13
- 3.3 Retos y soluciones encontradas 13
- 3.4 Documentación de pruebas y resultados 14
- 3.5 Reflexión sobre alternativas y decisiones 14
- 3.6 Orientación para el desarrollo de una plataforma web 15

4 Conclusiones 17
- 4.1 Conclusiones generales del proyecto 17
- 4.2 Consecución de los objetivos 17
- 4.3 Valoración de la metodología y planificación 18
- 4.4 Visión de futuro 18

5. Glosario 19

6. Bibliografia 20

7 Annexos 21

1 Introducción

AniToki consiste en el desarrollo de una plataforma web de streaming de anime que permite a los usuarios acceder a series y películas de forma organizada y accesible desde cualquier dispositivo con conexión a internet. El nombre del proyecto combina “Ani”, anime, y “Toki”, que significa tiempo en japonés, haciendo referencia a la posibilidad de disfrutar del contenido en cualquier momento y lugar.

El objetivo principal del proyecto es mejorar la experiencia de los aficionados al anime mediante una aplicación web intuitiva que integre un catálogo completo, sistemas de búsqueda y filtrado, reproducción directa de episodios y funcionalidades de personalización, como las listas de seguimiento.

1.1 Contexto

En la actualidad, el consumo de contenido audiovisual en streaming ha crecido de forma notable en los últimos años, especialmente entre el público joven. En el caso del anime, la oferta actual está repartida entre varias plataformas, cada una con sus limitaciones: Crunchyroll, por ejemplo, restringe gran parte de su catálogo a usuarios de pago y no siempre dispone de los títulos que los aficionados buscan. Por otro lado, plataformas como AniList o MyAnimeList permiten descubrir y valorar anime, pero no incluyen reproducción directa. Esta situación obliga a los usuarios a saltar de una plataforma a otra para seguir sus series, sin encontrar un sitio que lo centralice todo.

Como aficionados al anime desde hace años, los integrantes del equipo han vivido de primera mano esta fragmentación y la falta de una plataforma que combine un catálogo amplio, reproducción directa y una parte social con comentarios y valoraciones de la comunidad.

1.2 Justificación

AniToki nace de la idea de crear la plataforma que los propios desarrolladores habrían querido tener como usuarios. El objetivo era combinar lo que más gustaba de referencias como Crunchyroll y AniList: un catálogo accesible sin tantas restricciones de pago y un espacio donde la comunidad pueda opinar y valorar los títulos.

Durante el desarrollo se tuvo que priorizar la parte de streaming, que resultó más compleja de lo esperado, y dejar la parte social —comentarios y valoraciones— como línea de trabajo futura. A nivel académico, el proyecto ha permitido aplicar conocimientos de desarrollo web fullstack adquiridos durante las prácticas, donde se trabajó en un proyecto real desde cero hasta el despliegue, pasando por backend, frontend y base de datos.

1.3 Objetivos

Desarrollar una plataforma web de streaming de anime que sea funcional, accesible e intuitiva, permitiendo a los usuarios explorar un catálogo completo de series y películas, reproducir episodios directamente desde el navegador y gestionar preferencias de visualización mediante listas personalizadas.

Objetivos específicos

- Implementar un sistema de registro e inicio de sesión seguro para usuarios, incluyendo recuperación y cambio de contraseña.
- Crear un catálogo de anime con información detallada sobre series y episodios.
- Desarrollar un sistema de reproducción de episodios mediante streaming integrado en la plataforma, utilizando vídeos MP4 como material de muestra.
- Permitir la creación y gestión de listas de seguimiento personalizadas por cada usuario.
- Incorporar un sistema de búsqueda y filtrado eficiente para facilitar la localización de contenido.
- Diseñar una interfaz responsive que se adapte a diferentes dispositivos, garantizando una navegación clara y cómoda.
- Gestionar roles de usuario y administrador mediante un panel de control integrado, que incluye un dashboard para el administrador desde el que puede añadir, editar y eliminar animes y episodios directamente desde la propia aplicación.
- Añadir un sistema básico de valoración mediante likes y dislikes en animes y episodios.
- Garantizar la seguridad y confidencialidad de los datos de los usuarios, incluyendo contraseñas cifradas y comunicaciones protegidas.

1.4 Estrategia y planificación del proyecto

Para la ejecución del proyecto, se han considerado varias estrategias: adaptar una plataforma existente, integrar servicios de terceros o desarrollar un producto completamente nuevo. La estrategia elegida fue desarrollar la plataforma desde cero, lo que permitía tener control total sobre la arquitectura, las funcionalidades y el diseño, asegurando que el resultado final se ajustara a lo que se tenía en mente desde el principio.

La planificación del desarrollo se dividió en dos fases principales. En primer lugar, se diseñó toda la interfaz de la aplicación en Figma, definiendo previamente tanto el aspecto visual como la estructura general antes de comenzar a programar. Una vez finalizado el diseño, se utilizó la inteligencia artificial como apoyo para trasladar esa propuesta visual a la aplicación web, lo que permitió agilizar considerablemente el inicio del desarrollo. A partir de ese punto, el proceso avanzó de manera más fluida y progresiva, incorporando nuevas funcionalidades según las necesidades que iban surgiendo durante el proyecto.

En cuanto a la gestión de tareas, inicialmente se optó por utilizar Taiga, aunque pronto se descartó debido a que no resultaba especialmente práctico para un equipo formado únicamente por dos personas. Finalmente, el seguimiento del trabajo se realizó de una forma más sencilla: las tareas pendientes se organizaban en un documento de texto y el progreso del proyecto se controlaba directamente mediante los commits realizados en GitHub, donde cada integrante subía los avances desarrollados.

1.5 Metodología de trabajo

La metodología seguida fue práctica y adaptada a las características del equipo. Al compartir el mismo entorno de trabajo, la coordinación se realizaba de manera directa y presencial, mostrando continuamente los avances realizados, resolviendo dudas en el momento y decidiendo los siguientes pasos sin necesidad de reuniones formales ni herramientas externas de comunicación.

La distribución de tareas se organizó de forma natural en función de las necesidades pendientes y de la experiencia de cada integrante. En el principio se planificó el diseño de cómo se vería la plataforma luego debido a una mayor experiencia previa adquirida durante el periodo de prácticas, uno de los miembros asumió las partes más técnicas y complejas relacionadas con el backend y la arquitectura de la aplicación. Por otro lado, el otro integrante se centró principalmente en los componentes visuales y la documentación inicial del proyecto. A medida que el desarrollo avanzaba, ambos participaron conjuntamente en la implementación de nuevas funcionalidades dentro de la aplicación.

Aunque no se siguió una metodología ágil estricta como Scrum, sí se aplicaron algunos de sus principios de manera informal. El trabajo se desarrolló de forma incremental, priorizando las funcionalidades más importantes y adaptándose continuamente a los cambios y a los problemas que surgían durante el proceso de desarrollo.

1.6 Estudi econòmic i pressupostari

El proyecto AniToki se ha desarrollado íntegramente con herramientas gratuitas o de código abierto, lo que ha permitido mantener el coste económico en cero durante todo el proceso.

Herramientas y tecnologías utilizadas

- **Desarrollo:** Python, Django, Vue.js, Vite, Visual Studio Code, Git y GitHub, todas ellas herramientas gratuitas.
- **Diseño:** Figma, utilizando su versión gratuita.
- **Base de datos:** SQLite3 durante el desarrollo y PostgreSQL en producción, ambas soluciones gratuitas.
- **Despliegue:** Render, mediante su plan gratuito, donde la aplicación se encuentra desplegada y funcionando correctamente.
- **Asistencia con IA:** GitHub Copilot Pro con los modelos Claude Sonnet 4.5 y GPT-4.1. El plan Pro de Copilot, con un coste de 10 dólares mensuales, fue el único gasto económico real asociado al proyecto.

Decisión sobre la inversión económica

Desde el inicio del proyecto se decidió no realizar ninguna inversión económica. Al tratarse de un trabajo académico con una presentación concreta, no resultaba necesario asumir costes relacionados con hosting de pago, dominios u otras herramientas premium. Render permitió desplegar la aplicación de manera gratuita y obtener un resultado completamente funcional, suficiente para los objetivos planteados en el contexto académico.

Viabilidad futura

En caso de querer desarrollar el proyecto de forma profesional o comercial, el principal obstáculo económico serían las licencias de contenido. Los derechos de distribución de anime tienen un coste muy elevado, lo que convierte el proyecto en una propuesta comercial poco viable. A ello habría que añadir los costes derivados de un hosting de pago, un dominio propio y una infraestructura más escalable.

Por este motivo, AniToki se concibe exclusivamente como un proyecto académico y de aprendizaje, sin intención de transformarse en una plataforma comercial real.

2 Descripción del proyecto

El proyecto **AniToki** consiste en el desarrollo de una plataforma web de streaming de anime que permite acceder a series y películas.Organizada y accesible desde cualquier dispositivo con conexión a internet. La plataforma está diseñada para integrar un catálogo completo, reproducción directa de episodios, funcionalidades de personalización como listas de seguimiento y gestión de roles de usuario y administrador.

2.1 Análisis de requisitos

Los requisitos del proyecto se han definido para garantizar que la plataforma cumpla con sus objetivos funcionales y de calidad. El desarrollo se considerará finalizado cuando se cumplan estos requisitos.

2.1.1 Requisitos funcionales

RF1: Permitir el registro de nuevos usuarios mediante correo electrónico y contraseña.

RF2: Facilitar el inicio de sesión y la gestión del perfil de usuario.

RF3: Gestionar el catálogo de anime, incluyendo información sobre títulos, episodios, géneros y estado de visualización.

RF4: Reproducir episodios mediante streaming directamente desde la plataforma.

RF5: Permitir a los usuarios añadir anime a listas de seguimiento personalizadas.

RF6: Marcar episodios como los vistos.

RF7: Gestionar roles de usuario y administrador mediante un panel de control integrado.

2.1.2 Requisitos no funcionales

RNF1: La interfaz debe ser accesible desde dispositivos móviles y de escritorio (diseño responsive).

RNF2: Las páginas deben cargarse en menos de tres segundos bajo condiciones normales de conexión.

RNF3: Las contraseñas de los usuarios se cifrarán utilizando algoritmos seguros.

RNF4: La plataforma debe ser escalable y soportar múltiples usuarios simultáneamente.

RNF5: Todas las comunicaciones deben estar protegidas mediante HTTPS.

2.2 Tecnologías

2.2.1 Comparativa de las tecnologías valoradas

Durante la fase de análisis se valoraron distintas tecnologías para el desarrollo de la plataforma, considerando criterios como facilidad de implementación, seguridad, escalabilidad, curva de aprendizaje y compatibilidad con el entorno académico.

Backend

Para el desarrollo del backend se valoraron principalmente dos tecnologías: Django y Spring Boot. Finalmente, se seleccionó Django debido a la experiencia previa de uno de los integrantes del equipo, quien había trabajado anteriormente en un proyecto fullstack desarrollado desde cero utilizando este framework. Esta experiencia previa permitió reducir significativamente la curva de aprendizaje y acelerar el desarrollo inicial del proyecto.

Django ofrece múltiples ventajas para una plataforma como AniToki, especialmente por incluir funcionalidades integradas como el sistema de autenticación, cifrado seguro de contraseñas, panel de administración automático y un ORM robusto para la gestión de bases de datos relacionales.

Spring Boot fue descartado debido a su mayor complejidad de configuración y a una curva de aprendizaje más elevada. Aunque es una solución ampliamente utilizada en entornos empresariales, no aportaba ventajas suficientes para el alcance académico y funcional del proyecto.

Frontend

Para el frontend se analizaron Vue.js y React. Se eligió Vue.js 3 porque era el framework trabajado durante el ciclo formativo, lo que permitía al equipo desarrollar la interfaz con mayor rapidez. Entre las librerías utilizadas destacan Vue Router para la navegación entre páginas, HLS.js para la reproducción de vídeo en streaming y Chart.js para la visualización de datos en el panel de administración. Como herramienta de compilación y servidor de desarrollo se utilizó Vite, por su rapidez y optimización del entorno frontend. React fue descartado porque el equipo no contaba con experiencia previa suficiente.

Arquitectura

Desde las fases iniciales del proyecto se decidió separar completamente el frontend del backend, evitando el uso de plantillas tradicionales de Django. Esta decisión permitió adoptar una arquitectura desacoplada basada en API REST, facilitando la escalabilidad, el mantenimiento y la posibilidad de reutilizar el backend en futuras aplicaciones móviles o clientes externos.

La comunicación entre frontend y backend se realiza mediante peticiones HTTP a través de una API REST desarrollada con Django REST Framework. Durante el desarrollo, el servidor Vite se configuró con un sistema proxy para redirigir automáticamente las peticiones al backend Django ejecutado en el puerto 8000, simplificando la integración entre ambas partes del sistema.

Base de datos

Durante las primeras fases del desarrollo se utilizó SQLite3 debido a su simplicidad de configuración y facilidad para prototipar rápidamente funcionalidades básicas. Posteriormente, de cara al despliegue y producción, la plataforma migró a PostgreSQL.

PostgreSQL fue seleccionado por tratarse de un sistema gestor de bases de datos relacional robusto, escalable y ampliamente recomendado para proyectos desarrollados con Django. Además, ofrece mejor soporte para consultas complejas, relaciones avanzadas e integridad de datos.

Streaming

El sistema de streaming representó el apartado con mayor carga de investigación técnica, ya que ninguno de los integrantes había trabajado previamente con tecnologías de reproducción multimedia. Se valoraron HLS y MPEG-DASH, seleccionándose HLS por su amplia compatibilidad con navegadores modernos y mejor documentación. La reproducción se implementó mediante HLS.js en el frontend. Para el material de muestra, los vídeos se sirven en formato MP4.

Control de versiones

Se usó Git y GitHub desde el principio. Se valoró GitLab como alternativa, pero al estar ya familiarizados con GitHub y ser suficiente para el alcance del proyecto, no había motivo para cambiar.

Asistencia con inteligencia artificial

Durante el desarrollo del proyecto se utilizaron herramientas de inteligencia artificial como apoyo en varias fases del trabajo. Se recurrió a GitHub Copilot Pro con el modelo Claude Sonnet 4.5 y a GPT-4.1 tanto para resolver dudas técnicas como para automatizar partes repetitivas del desarrollo. En todo momento los integrantes actuaron como guías del proceso: definían qué se necesitaba, revisaban el resultado y tomaban las decisiones. La IA funcionó como una herramienta más dentro del flujo de trabajo.

2.2.2 Tecnologías escogidas

Tras el análisis comparativo, las tecnologías seleccionadas para el desarrollo del proyecto son las siguientes:

Backend

1. Lenguaje principal: Python
2. Framework: Django 4.2+
3. Base de datos: SQLite3 en desarrollo y PostgreSQL en producción (psycopg2-binary)
4. Librerías: django-cors-headers para gestión de CORS, python-decouple para variables de entorno y requests para peticiones HTTP

Frontend

1. Framework: Vue.js 3.5.13
2. Enrutamiento: Vue Router 4.5.0
3. Build tool: Vite 6.0.5
4. Reproducción de vídeo: HLS.js 1.6.16
5. Visualización de datos: Chart.js 4.5.1

Arquitectura

1. Monorepo con separación frontend/backend
2. API RESTful mediante endpoints de Django
3. Proxy de Vite configurado para redirigir las peticiones /api al servidor Django

Otras herramientas

1. Control de versiones: Git y GitHub
2. Diseño de interfaz: Figma
3. Entorno de desarrollo: Visual Studio Code
4. Asistencia con IA: GitHub Copilot (Claude Sonnet 4.5) y GPT-4.1

2.3 Estructura del proyecto

La arquitectura de AniToki se basa en un modelo cliente-servidor estructurado en tres capas principales:

1. **Capa de presentación (Frontend)**
   Interfaz web accesible desde el navegador. Permite registro, navegación por catálogo, reproducción y gestión de listas.

2. **Capa lógica (Backend – Django)**
   Gestiona autenticación, permisos, lógica de negocio, consultas a base de datos y control de acceso al streaming.

3. **Capa de datos (Base de datos PostgreSQL)**
   Almacena usuarios, animes, episodios, listas de seguimiento y comentarios.

Esta estructura permite una separación clara de responsabilidades, facilitando el mantenimiento y escalabilidad.

2.4 Descripción de los componentes

2.4.1 Componente 1: Interfaz de usuario (Frontend)

Descripción:
Es la capa visible del sistema. Permite la interacción directa del usuario con la plataforma.

Funcionalidad:

- Registro e inicio de sesión.
- Visualización del catálogo.
- Reproducción de episodios.
- Gestión de listas de seguimiento.

Tecnologías:
HTML, CSS, Vue.js 3, Vite y diseño responsive.

2.4.2 Componente 2: Servidor de aplicación (Backend)

Descripción:
Gestiona la lógica del sistema y coordina la comunicación entre la interfaz y la base de datos.

Funcionalidad:

- Autenticación y autorización de usuarios.
- Gestión de roles (usuario/administrador).
- Administración del catálogo.
- Control de acceso a los recursos de streaming.

Tecnologías:
Python y Django.

2.4.3 Componente 3: Base de datos

Descripción:
Sistema encargado del almacenamiento persistente de la información.

Funcionalidad:

- Almacenar datos de usuarios.
- Gestionar relaciones entre animes y episodios.
- Guardar listas de seguimiento.

Tecnología:
PostgreSQL.

2.4.4 Componente 4: Sistema de streaming

Descripción: Permite la reproducción de episodios.

Funcionalidad: Gestiona la distribución de contenido.

Tecnología: HLS.js

2.5 Definición de las funcionalidades

En este apartado se describen de forma detallada las funcionalidades que ofrece la plataforma AniToki. Para cada funcionalidad se explica qué permite hacer, el proceso conceptual que sigue y el estado actual de implementación.

2.5.1 Funcionalidad 1: Registro e inicio de sesión

Descripción:
Permite a los usuarios crear una cuenta y acceder a la plataforma de forma segura.

Proceso conceptual:

1. El usuario completa el formulario de registro con datos personales (usuario, correo, contraseña).
2. El sistema valida la información y comprueba duplicados.
3. La contraseña se almacena de forma cifrada mediante el sistema de hashing de Django.
4. En el inicio de sesión, el sistema verifica credenciales y crea una sesión autenticada para el usuario.

Estado: implementada totalmente.

2.5.2 Funcionalidad 2: Gestión del catálogo de animes

Descripción:
Permite consultar los animes disponibles, acceder a fichas detalladas y listar episodios de cada serie o película.

Proceso conceptual:

1. El usuario accede al catálogo desde la interfaz web.
2. El backend consulta la base de datos y devuelve la información de los animes.
3. Se muestran elementos como título, sinopsis, género, número de episodios y estado de visualización.
4. Al seleccionar un anime, se abre la ficha con todos los episodios disponibles.

Estado: implementada totalmente.

2.5.3 Funcionalidad 3: Reproducción de episodios

Descripción:
Permite reproducir los episodios mediante streaming adaptativo directamente desde el navegador.

Proceso conceptual:

1. El usuario selecciona un episodio desde la ficha de anime.
2. El backend valida los permisos del usuario.
3. El reproductor carga el archivo HLS

Estado: implementada totalmente.

2.5.4 Funcionalidad 4: Listas de seguimiento personalizadas

Descripción:
Permite a los usuarios guardar animes y controlar su progreso de visualización.

Proceso conceptual:

1. El usuario añade un anime a su lista de seguimiento.
2. Se crea una relación entre el usuario y el anime en la base de datos.
3. Se actualiza el progreso de episodios vistos.
4. El usuario puede consultar y modificar su lista en cualquier momento.

Estado: implementada totalmente.

2.5.5 Funcionalidad 6: Gestión de roles y permisos

Descripción:
Controla los permisos de los usuarios según su tipo (administrador, estándar).

Proceso conceptual: El backend verifica permisos antes de permitir el acceso a cada funcionalidad. Los administradores pueden gestionar el catálogo mediante el dashboard integrado.

Estado: implementada totalmente.

3 Otros capítulos

Justificación de las decisiones tomadas durante el desarrollo de AniToki. Alternativas valoradas, análisis técnicos y aspectos relevantes que complementan la memoria del proyecto.

3.2 Decisiones de diseño

Se detallan las decisiones de diseño de interfaz y arquitectura:

1. **Diseño responsive:** Se eligió un enfoque mobile-first para asegurar compatibilidad con todos los dispositivos.

2. **Arquitectura cliente-servidor:** Se separó frontend y backend para mejorar escalabilidad y mantenimiento.

3. **Seguridad:** Se aplicaron medidas como HTTPS, cifrado de contraseñas con Django.

4. **Panel de administración:** Se desarrolló un dashboard propio integrado en la aplicación para que el administrador pueda gestionar el catálogo.

3.3 Retos y soluciones encontradas

Reproducción de vídeo

Uno de los primeros retos fue encontrar una forma viable de mostrar los episodios. Inicialmente usábamos enlaces externos de YouTube para cargar los vídeos, pero el problema era que YouTube detectaba las peticiones como automatizadas si el usuario no estaba logueado en el navegador, lo que impedía la reproducción. La solución llegó de forma casual: viendo a un amigo ver One Piece desde One Pace, una página de fans que sube capítulos resumidos y editados de la serie, nos dimos cuenta de que ofrecía la opción de descarga directa en MP4. A partir de ahí usamos esos archivos como material de muestra para la plataforma. Para el resto de animes seguimos usando enlaces externos, y para One Piece usamos los MP4 descargados directamente de One Pace.

Integración frontend y backend

La comunicación entre Vue.js y Django mediante la API REST no supuso un problema técnico grave. Hubo errores puntuales relacionados con CORS o con endpoints mal conectados, pero en todos los casos el problema era identificable leyendo el error y se resolvía rápido. Más que retos técnicos complejos, fueron descuidos normales del desarrollo día a día.

Control de versiones

El reto más destacado a nivel de equipo fue un problema con Git: en un momento del desarrollo, uno de los integrantes ejecutó git push --force y eliminó todos los commits del repositorio. La solución fue reconstruir los commits desde el código local y crear una rama separada para el compañero, aunque finalmente se descartó porque generaba más confusión que otra cosa. A partir de ese momento se explicó cómo trabajar correctamente con Git y no volvió a haber ningún problema similar.

3.4 Documentación de pruebas y resultados

Se incluye documentación conceptual de pruebas realizadas para validar funcionalidades:

1. **Pruebas de usuario:** registro, login, reproducción de episodios y listas de seguimiento.
2. **Pruebas de rendimiento:** tiempos de carga, consumo de recursos, estabilidad del streaming.
3. **Pruebas de seguridad:** roles y cifrado de contraseñas.
4. **Pruebas de compatibilidad:** navegadores (Chrome, Firefox, Edge, Safari) y dispositivos móviles.

3.5 Reflexión sobre alternativas y decisiones

Cada decisión técnica y de diseño se tomó considerando:

- Impacto en la experiencia de usuario.
- Facilidad de mantenimiento y escalabilidad.
- Integración con tecnologías existentes.
- Viabilidad técnica dentro del tiempo y recursos disponibles.

3.6 Orientación para el desarrollo de una plataforma web

Cualquier persona que quiera desarrollar una plataforma web como AniToki puede hacerlo si entiende qué decisiones tiene que tomar y por qué. La inteligencia artificial ha reducido enormemente la barrera técnica, pero lo importante no es que la IA escriba el código, sino que quien desarrolla entienda qué está haciendo y por qué lo hace así.

El punto de partida: definir qué se quiere construir

Antes de tocar ninguna tecnología, quien quiera desarrollar una plataforma similar debería tener claro qué problema quiere resolver y qué funcionalidades son imprescindibles. No hace falta tenerlo todo definido desde el principio, pero sí una idea central sólida. En el caso de AniToki, la idea era centralizar anime en un solo lugar con reproducción directa y gestión de listas. Todo lo demás vino después.

Elegir el backend

El backend es la parte del sistema que gestiona la lógica, los datos y la seguridad. Quien empiece a investigar encontrará opciones como Django, FastAPI, Express o Spring Boot, entre otras. La clave no es elegir la más potente, sino la que mejor encaje con el nivel de experiencia, el tiempo disponible y los requisitos del proyecto. Vale la pena buscar cuál tiene mejor documentación, qué incluye de serie y cuánto tiempo llevaría aprender lo necesario para arrancar.

Elegir el frontend

El frontend es lo que ve y con lo que interactúa el usuario. Existen frameworks como Vue.js, React o Angular, cada uno con su filosofía y curva de aprendizaje. Quien investigue debería fijarse en qué tan activa es su comunidad, qué librerías tiene disponibles y si se integra bien con el backend elegido. También es importante decidir si se quiere una arquitectura separada, donde el frontend y el backend son proyectos independientes que se comunican mediante una API, o si se prefiere un enfoque más integrado.

La base de datos

Toda plataforma necesita almacenar datos. Quien se adentre en este apartado encontrará dos grandes familias: bases de datos relacionales como PostgreSQL o MySQL, y bases de datos no relacionales como MongoDB. Para una plataforma con usuarios, catálogos y listas de seguimiento, una base de datos relacional suele ser la opción más natural. Lo importante es entender cómo se relacionan los datos entre sí antes de elegir ninguna tecnología concreta.

El diseño de la interfaz

Antes de escribir código de interfaz, conviene tener claro cómo va a verse y funcionar la aplicación. Herramientas como Figma permiten diseñar pantallas sin necesidad de programar, lo que ayuda a visualizar el resultado final y detectar

problemas de usabilidad antes de que cuesten tiempo de desarrollo. Quien se salte este paso suele acabar rehaciendo partes del frontend que podrían haberse resuelto en el diseño.

Funcionalidades especiales

Dependiendo del tipo de plataforma, habrá funcionalidades que requieran investigación específica. En el caso de AniToki, la reproducción de vídeo fue el mayor reto, ya que implica entender cómo funciona el streaming, qué protocolos existen y cómo se procesan y sirven los archivos de vídeo. Quien quiera implementar algo similar debería investigar conceptos como HLS, segmentación de vídeo y reproductores compatibles con el navegador antes de decidir cómo abordarlo.

El despliegue

Desarrollar en local es solo la mitad del trabajo. Quien quiera que su aplicación sea accesible desde cualquier lugar necesita entender cómo funciona el despliegue: dónde se aloja el servidor, cómo se configura el entorno de producción y qué diferencias hay respecto al entorno de desarrollo. Plataformas como Render, Railway o Vercel permiten desplegar proyectos de forma gratuita y son un buen punto de partida para entender este proceso sin asumir costes.

El control de versiones

Cualquiera que desarrolle un proyecto, ya sea en solitario o en equipo, debería usar un sistema de control de versiones desde el primer día. Git permite registrar cada cambio en el código, volver a versiones anteriores si algo sale mal y trabajar en paralelo sin pisarse el trabajo. Quien empiece a investigar encontrará plataformas como GitHub o GitLab donde alojar el repositorio de forma remota. Lo importante no es solo instalarlo y usarlo, sino entender cómo funciona: qué es un commit, qué implica trabajar con ramas y, sobre todo, qué comandos pueden ser destructivos si se usan sin saber lo que hacen. Un error con el historial de versiones puede costar horas de trabajo recuperable o, en el peor caso, código perdido definitivamente.

El papel de la inteligencia artificial

La IA puede acelerar enormemente el desarrollo, desde generar estructuras base hasta resolver dudas técnicas en el momento. Sin embargo, quien la use sin entender lo que genera acabará con código que no sabe mantener ni depurar. La forma correcta de aprovecharla es como una herramienta de apoyo: plantearle problemas concretos, revisar lo que produce y asegurarse de entender por qué funciona antes de integrarlo en el proyecto.

4 Conclusiones

4.1 Conclusiones generales del proyecto

El desarrollo de AniToki ha permitido consolidar los conocimientos teóricos y prácticos adquiridos en el ámbito del desarrollo web, bases de datos y gestión de usuarios. La creación de una plataforma de streaming de anime funcional y accesible ha proporcionado experiencia en la implementación de control de roles, interfaces responsive y gestión de contenido multimedia.

Desde el punto de vista académico, el proyecto ha reforzado la comprensión de la arquitectura cliente-servidor, el uso de frameworks modernos como Django y Vue.js y la integración de bases de datos relacionales. Profesionalmente, ha permitido desarrollar habilidades en planificación de proyectos, trabajo en equipo y diseño de interfaces orientadas al usuario.

Además, el proyecto ha puesto en práctica la importancia de la organización, el control de versiones y la planificación técnica, mostrando cómo se combinan distintas tecnologías.

4.2 Consecución de los objetivos

El objetivo de desarrollar una plataforma web de streaming de anime funcional y accesible se ha alcanzado, ya que se ha implementado un frontend y un backend integrados que permiten el registro de usuarios, el acceso al catálogo completo y la reproducción de episodios directamente desde la plataforma.

1. El registro e inicio de sesión seguro se ha logrado con éxito, utilizando la autenticación de Django, que asegura que las credenciales se almacenen de forma cifrada.

2. El catálogo completo de animes se ha implementado satisfactoriamente mediante PostgreSQL, organizando tablas de animes, episodios y listas de seguimiento.

3. El reproductor de vídeo funciona correctamente mediante HLS.js, permitiendo la reproducción de episodios desde el navegador.

4. Las listas de seguimiento se han implementado totalmente, permitiendo a los usuarios agregar animes, marcar episodios como vistos y gestionar su progreso.

5. El sistema de búsqueda y filtrado se ha completado, con filtros dinámicos en el frontend que realizan consultas optimizadas en el backend.

6. La gestión de roles de usuario y administrador está plenamente operativa, con un dashboard propio desde el que el administrador puede añadir, editar y eliminar contenido.

7. La funcionalidad que quedó fuera del alcance final fue la parte social: comentarios y valoraciones de la comunidad. Sí se implementó un sistema básico de likes y dislikes. El resto queda como línea de trabajo futura.

4.3 Valoración de la metodología y planificación

La metodología utilizada fue práctica y flexible, permitió adaptaciones rápidas ante imprevistos y facilitó la coordinación entre los dos integrantes del equipo. Aunque no se siguió Scrum de forma estricta, el enfoque incremental funcionó bien para un equipo pequeño con tecnologías parcialmente nuevas.

Algunas funcionalidades requirieron más tiempo del previsto, especialmente todo lo relacionado con el streaming y la investigación de tecnologías no conocidas previamente. La metodología ágil permitió ajustar prioridades sin comprometer el resultado final.

4.4 Visión de futuro

Algunas funcionalidades quedan pendientes de implementar o mejorar, lo que ofrece oportunidades de expansión del proyecto:

- Optimización del streaming adaptativo para distintas calidades y condiciones de red.
- Desarrollo de la parte social: comentarios y valoraciones de la comunidad al estilo AniList.
- Desarrollo de versión móvil nativa para Android/iOS.
- Sistema de recomendaciones personalizadas basado en historial de visualización.
- Expansión de roles y funciones administrativas para moderación avanzada de contenido.

Estas mejoras permitirán que AniToki evolucione hacia una plataforma más completa y competitiva.

5. Glosario

HLS (HTTP Live Streaming): Protocolo de streaming adaptativo que permite reproducir vídeo en tiempo real ajustando la calidad según la velocidad de conexión del usuario.

Django: Framework web en Python que facilita el desarrollo rápido de aplicaciones, proporcionando autenticación de usuarios, gestión de bases de datos mediante ORM y panel de administración.

PostgreSQL: Sistema de gestión de bases de datos relacional utilizado para almacenar información sobre usuarios, animes, episodios y listas de seguimiento.

Vue.js: Framework de JavaScript para el desarrollo de interfaces de usuario basado en componentes reutilizables.

Responsive: Diseño de interfaces que se adapta automáticamente a diferentes tamaños de pantalla y dispositivos.

ORM (Object-Relational Mapping): Técnica que permite interactuar con bases de datos relacionales usando objetos de programación.

Streaming adaptativo: Tecnología que ajusta la calidad del vídeo de forma automática según el ancho de banda disponible.

Git y GitHub: Git es un sistema de control de versiones distribuido. GitHub es una plataforma que facilita la colaboración y el almacenamiento del repositorio remoto.

Vite: Herramienta de compilación y servidor de desarrollo para proyectos frontend, conocida por su rapidez.

API REST: Interfaz de programación que permite la comunicación entre frontend y backend mediante peticiones HTTP estándar.

Scrum / Metodología ágil: Enfoque de desarrollo iterativo e incremental que permite ajustar prioridades y adaptarse a cambios durante la ejecución.

Backend: Parte del sistema que gestiona la lógica, bases de datos y sirve contenido al frontend.

Frontend: Parte del sistema que interactúa directamente con el usuario, mostrando la interfaz y contenidos multimedia.

Lista de seguimiento: Funcionalidad que permite a los usuarios almacenar animes, marcar episodios como vistos y llevar un control de su progreso.

JWT (JSON Web Token): Estándar para transmitir información segura entre cliente y servidor, usado para autenticación.

Usuario estándar: Usuario registrado que accede al contenido y funcionalidades básicas de la plataforma.

6. Bibliografia

1. Django Software Foundation. Django Documentation. https://docs.djangoproject.com/ (consultado el 15/02/2026)

2.PostgreSQL Global Development Group. PostgreSQL Documentation. https://www.postgresql.org/docs/ (consultado el 15/02/2026)

3.Apple Inc. HTTP Live Streaming (HLS) Overview. https://developer.apple.com/streaming/ (consultado el 15/02/2026)

4. Figma Inc. Figma – Design and Prototyping Tool. https://www.figma.com/ (consultado el 15/02/2026)

5. GitHub. GitHub Docs – Version Control and Collaboration. https://docs.github.com/ (consultado el 15/02/2026)

6. Vue.js. Vue.js Documentation. https://vuejs.org/ (consultado el 15/02/2026)

7. Vite. Vite Documentation. https://vitejs.dev/(consultado el 15/02/2026)

7 Annexos

No se incluyen anexos en esta memoria. Toda la información técnica está reflejada en los README.md en los repositorios.