Instituto Puig Castellar
Ciclo formativo: DAM
Grado: CFGM / CFGS
Crédito de Síntesis / Proyecto intermodular

Inazuma Legacy

Proyecto de Desarollo

Autor/a/es: Víctor López Ruiz
Tutor: Luis Elia
Curso: 2025-2026
Fecha de entrega: 17/5/2026

Resumen del proyecto

Inazuma Legacy es una aplicación dedicada al universo de Inazuma Eleven, centrada en el anime y los videojuegos de la saga. La temática del proyecto es la gestión y consulta de información sobre temporadas, equipos, jugadores y partidos, con un componente social para fans. El objetivo principal es ofrecer una plataforma centralizada, intuitiva e interactiva donde los usuarios puedan informarse, crear plantillas personalizadas y comunicarse mediante chats individuales y grupales. La metodología seguida se basa en el análisis de requisitos funcionales y no funcionales, la definición de roles de usuario, el diseño del modelo de datos y de las interfaces, y una planificación técnica que incluye pruebas continuas de cada funcionalidad implementada. Como conclusiones, el proyecto demuestra que es viable construir una herramienta completa para la comunidad fan, que mejora el acceso a la información y fomenta la interacción social, aunque es necesaria una buena planificación del tiempo, gestión de riesgos e iteración constante para lograr una versión estable y usable.

Palabras clave

Inazuma Eleven, Spring Boot, WebSockets, Inteligencia Artificial, PostgreSQL, Desarrollo Full-stack.

Abstract

Inazuma Legacy is an application dedicated to the Inazuma Eleven universe, focusing on the anime and video games of the series. The project's theme is the management and consultation of information about seasons, teams, players, and matches, with a social component for fans. The main objective is to offer a centralized, intuitive, and interactive platform where users can stay informed, create custom rosters, and communicate through individual and group chats. The methodology followed is based on the analysis of functional and non-functional requirements, the definition of user roles, the design of the data model and interfaces, and technical planning that includes continuous testing of each implemented feature. In conclusion, the project demonstrates that it is feasible to build a comprehensive tool for the fan community, improving access to information and fostering social interaction, although good time management, risk management, and constant iteration are necessary to achieve a stable and usable versions.)

Keywords

Inazuma Eleven, Spring Boot, WebSockets, Artificial Intelligence, PostgreSQL, Full-stack Development.

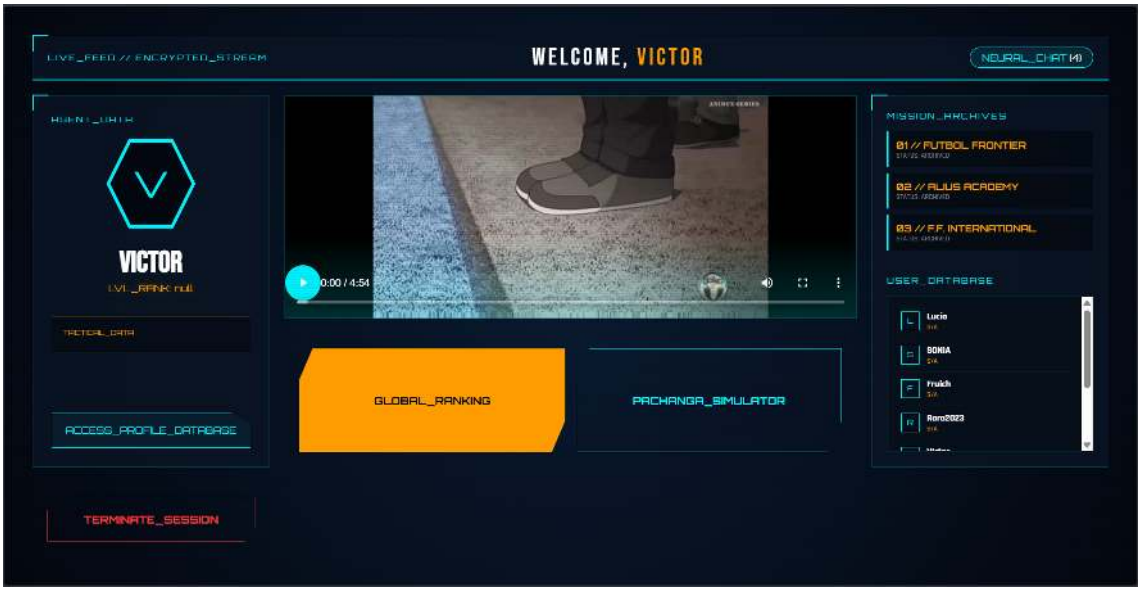
Índice

1. Presentación del proyecto

1.1 Introducción

Inazuma Legacy es un proyecto de software desarrollado en Java con Spring Boot que integra, en un entorno de ejecución robusto, la gestión de datos y la interacción social para la comunidad de Inazuma Eleven. El sistema permite el análisis detallado de la evolución técnica y estadística de los jugadores del club Raimon durante sus tres primeras temporadas, ofreciendo herramientas de personalización como un creador de equipos y un sistema dinámico de logros que gamifica la experiencia del usuario.

La aplicación destaca por la implementación de funcionalidades de comunicación en tiempo real, incluyendo chats privados y grupales con sistemas de reacción, así como un asistente virtual basado en Inteligencia Artificial. Mediante el uso de WebSocket con STOMP y la integración de APIs externas, este desarrollo demuestra una alta capacidad de procesamiento de datos y una interfaz intuitiva, consolidándose como una herramienta integral que combina la gestión de bases de datos con entornos colaborativos modernos dentro del ecosistema Java.



"Vista inicio de mi web"

1.2 Contexto

El proyecto **Inazuma Legacy** se enmarca en el ámbito del desarrollo de aplicaciones web con fines informativos y de gestión de comunidades digitales. Se sitúa en un contexto donde el consumo de contenido multimedia y la cultura del "fandom" exigen herramientas de interacción más sofisticadas que las simples bases de datos estáticas. En este sentido, el proyecto nace como una respuesta a la necesidad de integrar servicios de comunicación modernos e inteligencia artificial en entornos de consulta documental sobre la franquicia Inazuma Eleven.

El ámbito tecnológico que rodea al proyecto es el del ecosistema **Java con Spring Boot**, poniendo el foco en la creación de interfaces interactivas y el procesamiento de datos en tiempo real. El proyecto no solo aborda la preservación de la información de las temporadas clásicas del club Raimon, sino que también explora la convergencia entre el software tradicional y las nuevas tecnologías de asistencia virtual (IA). Este contexto permite demostrar cómo una aplicación web desarrollada con Java y Spring Boot puede evolucionar hacia un entorno social y gamificado, adaptándose a las demandas actuales de los usuarios digitales.

1.3 Justificación

La elección de este proyecto no responde únicamente a un interés personal en la franquicia, sino a la oportunidad de resolver desafíos técnicos avanzados del ciclo de DAM. La necesidad de gestionar una gran cantidad de datos estadísticos volátiles y la exigencia de implementar una comunicación bidireccional en tiempo real (WebSockets) justifican el uso de un ecosistema robusto como Spring Boot. El reto principal reside en centralizar información dispersa bajo una arquitectura profesional (MVC) que garantice la escalabilidad y la integridad de los datos.

"Vista detallada de la selección de jugadores mediante temporadas."

1.4 Objetivos

El objetivo principal de este proyecto es el desarrollo de una plataforma web integral con **Java y Spring Boot** que unifique la gestión documental histórica con herramientas de comunicación de vanguardia. Se pretende transformar la consulta de datos estáticos en una experiencia interactiva mediante una arquitectura que permita al usuario explorar la trayectoria del club Raimon a la vez que interactúa con una comunidad activa y un asistente inteligente, rompiendo así la pasividad de las soluciones de información convencionales.

Para alcanzar esta visión, se plantean metas específicas centradas en la funcionalidad y el rendimiento:

- En primer lugar, la creación de un sistema indexado para la consulta técnica de jugadores y un módulo de gestión de alineaciones personalizadas.
- En segundo lugar, la implementación de un sistema de mensajería en tiempo real mediante **WebSocket con STOMP** que soporte chats privados y grupales con interacción dinámica, integrado con un asistente de IA generativa.
- Finalmente, el diseño de un motor de gamificación basado en un sistema de logros que asegure una navegación profunda y transversal por todas las capacidades del software.

2. Estrategia y planificación

2.1 Estrategia de desarrollo y viabilidad

Para la realización de este proyecto se ha optado por una metodología de desarrollo **iterativa e incremental**, inspirada en los principios **Agile**. Este enfoque ha permitido priorizar la construcción de un núcleo funcional sólido para, posteriormente, integrar de forma modular los componentes de

mayor complejidad, como el sistema de chat con **WebSocket** y el asistente de **Inteligencia Artificial**. Esta estrategia facilita la validación constante de cada módulo dentro del entorno de desarrollo, asegurando que las nuevas funcionalidades no comprometan la estabilidad del sistema global.

El ciclo de vida del software se ha dividido en fases de análisis de requerimientos, diseño de arquitectura de datos, codificación modular en **Java** y pruebas de integración de APIs externas. Mediante esta segmentación, se ha garantizado un control de calidad riguroso sobre el flujo de información y la interfaz de usuario, permitiendo ajustar el diseño de manera ágil según los retos técnicos surgidos durante la implementación de los servicios en tiempo real.

2.2 Metodología de trabajo

La gestión de las tareas se ha articulado mediante un tablero de planificación tipo **Kanban**, clasificando las funcionalidades según su estado de desarrollo (pendiente, en curso, en pruebas y finalizado). Esta organización visual ha sido clave para mantener la trazabilidad del proyecto y optimizar los tiempos de entrega de cada módulo. Asimismo, se ha utilizado **Git** como sistema de control de versiones, permitiendo una gestión eficiente del código y la posibilidad de revertir cambios o experimentar con nuevas librerías sin afectar a la rama principal del proyecto.

El entorno de trabajo se ha centrado en el IDE **IntelliJ IDEA**, aprovechando sus herramientas de depuración avanzada y la gestión de dependencias para agilizar la programación. El uso de métodos sistemáticos de trabajo, sumado a una documentación constante del código, ha permitido que el desarrollo de **Inazuma Legacy** haya sido ordenado y escalable, transformando los requisitos iniciales en una aplicación funcional y coherente con los objetivos establecidos.

Setmana	Data	Hi ha hagut avanç?	Activitats realitzades	Tasques a treballar	Dificultats trobades	Solucions realitzades	Comentaris generals per part del grup	Comentaris del tutor
1	21/9/25	☑	Proposta de Projecte	Definir la teva idea de projecte y quines eines faràs servir	Dir quines eines fare servir durant el projecte	Posar-ne de eines temporals fins que trobi eines definitives	No faig el projecte en grup	
2	28/9/25	☑	Proposta de Projecte	Arreglar comentaris de la proposta	Trobar eines noves per a solucionar els comentaris	Cercar per internet	No faig el projecte en grup	
3	5/10/25	☑	Corregir comentaris de la proposta	Acabar de corregir comentaris de la porposta	Cap	Pensar hi una altra forma de redactarles i/o noves ideas	No faig el projecte en grup	
4	12/10/25	☑	2ª correcio de la porposta	Corregir la segona tanda de comentaris de la proposta	Cap	Pensar-hi una altra forma de redactarles i/o noves ideas	No faig el projecte en grup	

"Metodología que he usado para planificarme"

2.3 Planificación

La cronología del proyecto se ha dividido en cuatro fases principales, distribuidas a lo largo del ciclo escolar, para garantizar un desarrollo equilibrado de todas las funcionalidades. La primera fase se dedicó exclusivamente a la **investigación y análisis**, donde se definió la estructura de datos de los jugadores y se investigó la integración de APIs de terceros para la IA. Posteriormente, se inició la fase de **diseño y arquitectura**, centrada en la creación de la interfaz de usuario y la jerarquía de clases en Java, asegurando que el sistema de chat y el creador de equipos tuvieran una base lógica eficiente y escalable.

La segunda mitad del calendario se centró en el **desarrollo activo e integración**, dedicando una parte significativa del tiempo a la programación del servidor de chat y el procesamiento de las respuestas del asistente virtual. Finalmente, se reservó un periodo de **pruebas y depuración** (*debugging*) para resolver conflictos en la integración del WebSocket y la conexión con la API de OpenAI, además de pulir la experiencia de usuario. Esta distribución temporal ha permitido cumplir con los hitos establecidos, asegurando que la entrega final de **Inazuma Legacy** sea un producto completo y funcional.

"Diagrama de Gantt"

3. Análisis

3.1 Casos de uso

Para definir el comportamiento funcional de **Inazuma Legacy**, se ha realizado un modelado basado en casos de uso que identifica la interacción entre los diferentes actores y las funcionalidades del software. Basándonos en la lógica de negocio y la seguridad implementada con **Spring Security**, se han definido dos perfiles principales con permisos claramente diferenciados. El primero es el **Usuario**

Invitado, cuyo alcance es meramente consultivo, y el segundo es el **Usuario Registrado**, que tras superar el proceso de autenticación desbloquea las capacidades de persistencia e interacción social. A continuación, se desglosan las funcionalidades específicas que componen la plataforma:

- **Enciclopedia y Diccionario de Jugadores:** Sistema de consulta de la base de datos que permite filtrar a los personajes por temporada, posición y estadísticas tácticas, facilitando el acceso a la información técnica de la franquicia.
- **Gestión de Perfiles de Usuario:** Espacio personal donde el usuario registrado puede administrar sus datos, personalizar su cuenta y visualizar su progreso dentro de la web.
- **Simulación de Partidos (Pachangas):** Algoritmo de juego que permite realizar enfrentamientos virtuales entre usuarios o contra la IA, calculando el resultado en base a las estadísticas de ataque, defensa y rapidez de los jugadores seleccionados.
- **Sistema de Chat en Tiempo Real:** Servicio de mensajería instantánea implementado con el protocolo WebSockets (STOMP y SockJS) que permite la comunicación fluida en salas públicas y chats privados sin recarga de página.
- **Ranking Global y Competitivo:** Tabla de clasificación dinámica que ordena a los usuarios según su actividad y rendimiento en las simulaciones, fomentando la competitividad.
- **Asistente Virtual con IA:** Integración de la API de OpenAI para ofrecer un experto en el universo Inazuma capaz de responder dudas complejas y generar contexto para el usuario.

Este diseño asegura que cada funcionalidad tenga una correspondencia directa en los *endpoints* del backend, garantizando la integridad de los datos y una experiencia de usuario fluida según su estado en el sistema.

3.2 Requisitos funcionales

Los requisitos funcionales detallan las capacidades específicas que **Inazuma Legacy** ofrece a sus usuarios, asegurando que todas las necesidades detectadas en el análisis previo queden cubiertas por la lógica del software. Para una mayor claridad, se presentan de forma desglosada:

- **RF1 - Consulta Indexada de Jugadores:** El sistema implementa un motor de búsqueda y filtrado que permite a los usuarios acceder a la base de datos de personajes. La información está organizada de forma relacional, permitiendo visualizar estadísticas técnicas (ataque,

defensa, rapidez) y datos biográficos clasificados por las distintas temporadas de la franquicia.

- **RF2 - Sistema de Simulación (Pachangas):** La aplicación incluye una lógica de juego que permite realizar enfrentamientos virtuales. El sistema procesa las estadísticas de los jugadores seleccionados y genera un resultado basado en algoritmos de comparación de atributos, permitiendo la competitividad entre perfiles.
- **RF3 - Mensajería Instantánea Bidireccional:** Se ha implementado un servicio de chat dinámico basado en el protocolo **WebSocket (STOMP)**. Este requisito asegura una comunicación fluida y en tiempo real, tanto en canales globales como en chats privados, eliminando la necesidad de refrescar la interfaz.
- **RF4 - Consultoría Inteligente mediante IA:** Gracias a la integración de la **API de OpenAI**, el sistema actúa como un experto en el universo de la serie. El asistente es capaz de interpretar lenguaje natural para responder dudas sobre tácticas, historia o personajes, manteniendo el contexto de la sesión.
- **RF5 - Escalafón y Ranking Global:** El sistema calcula y muestra una tabla de clasificación actualizada donde los usuarios compiten según su actividad y éxito en las simulaciones, incentivando el uso continuo de la plataforma.

3.3 Requisitos no funcionales

Por otro lado, los requisitos no funcionales (RNF) establecen las cualidades del software en términos de eficiencia, seguridad y usabilidad. En este proyecto, se han priorizado los siguientes aspectos técnicos:

- **Rendimiento y latencia:** Se garantiza una baja latencia en la entrega de mensajes gracias a la optimización del protocolo STOMP y una conexión estable y eficiente con la base de datos **PostgreSQL**.

- **Integridad y Seguridad:** Se ha puesto especial énfasis en la consistencia de los datos y en la protección de la información del usuario mediante el cifrado de sesiones proporcionado por **Spring Security**.
- **Usabilidad y Estética:** Diseño de una interfaz de usuario (UI) intuitiva que respeta la identidad visual de la franquicia, asegurando que la navegación sea fluida y la aplicación se mantenga estable en los principales navegadores web.
- **Escalabilidad:** La arquitectura modular del backend permite la futura implementación de nuevas funcionalidades sin comprometer el rendimiento del núcleo del sistema.

3.4 Análisis de alternativas tecnológicas

Durante la fase de diseño se evaluaron diversas alternativas para el desarrollo de la plataforma, considerando lenguajes como **Python** o **C#**. Aunque Python ofrece una integración nativa sencilla con bibliotecas de IA, se optó por **Java con Spring Boot** debido a su robustez, su ecosistema web maduro y su arquitectura orientada a objetos, la cual es ideal para gestionar la complejidad del sistema de chat y la jerarquía de clases de los jugadores. Asimismo, se prefirió el uso del IDE **IntelliJ IDEA** frente a Eclipse por su gestión eficiente de dependencias y sus herramientas avanzadas de análisis de código, que han permitido un desarrollo más ágil y con menos errores estructurales.

En cuanto al sistema de comunicación y la inteligencia artificial, se consideró la implementación de bases de datos documentales en lugar de SQL y bots de respuesta estática. No obstante, se decidió apostar por una arquitectura basada en **APIs de IA generativa** para ofrecer una experiencia más natural y adaptativa, y por un sistema de chat basado en **WebSocket con STOMP** que permitiera la interacción en tiempo real con reacciones. Esta combinación de tecnologías asegura que **Inazuma Legacy** no sea solo una aplicación de consulta, sino una herramienta web moderna que aprovecha las capacidades del software actual para ofrecer un rendimiento óptimo a sus usuarios.

Tecnología	**Pros**	**Contras**	**Decisiones**
Python (Django/Flask)	Excelente integración con librerías de IA y sintaxis sencilla.	Menor eficiencia en la gestión de hilos (*multithreading*) para chats masivos.	Descartado
Node.js (Express)	Muy rápido para comunicación en	Gestión de tipos débil y arquitectura menos	Descartado

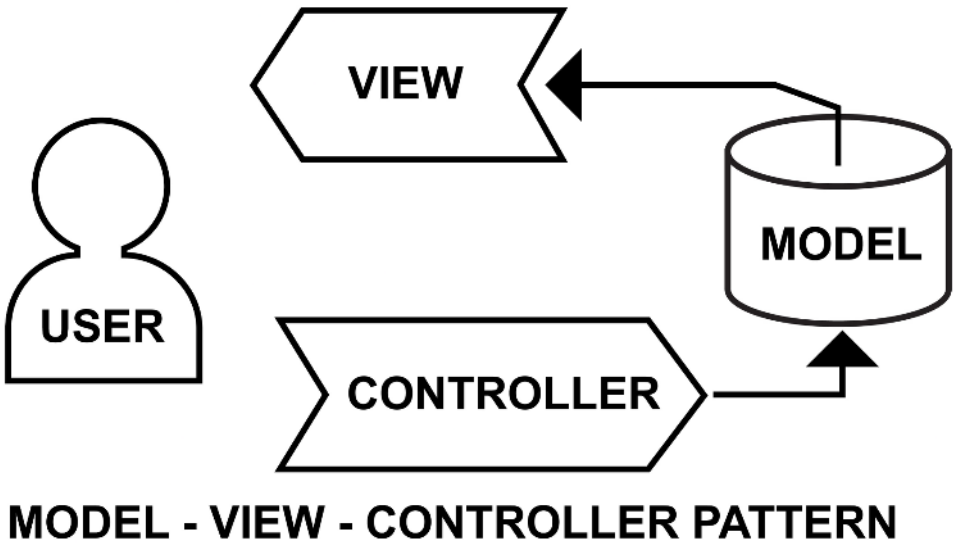
	tiempo real (WebSockets).	estructurada para un backend complejo.	
Java (Spring Boot)	Arquitectura robusta, seguridad nativa (Spring Security) y alta escalabilidad.	Curva de aprendizaje más elevada y configuración inicial compleja.	ELEGIDO

4. Diseño

4.1 Arquitectura del sistema

Para el desarrollo de **Inazuma Legacy**, se ha implementado una arquitectura de tres capas basada en el patrón **MVC (Modelo-Vista-Controlador)**, que permite una separación clara entre la lógica de negocio, los datos y la interfaz de usuario.

- **Capa de Presentación (Front-end):** Utiliza **Thymeleaf** como motor de plantillas para renderizar vistas HTML5 dinámicas, combinado con CSS3 y JavaScript para gestionar la interactividad en tiempo real mediante **WebSockets**.
- **Capa de Negocio (Back-end):** Gestionada por **Spring Boot**, que actúa como el núcleo del sistema. Se encarga de procesar las peticiones de los clientes, gestionar la seguridad mediante **Spring Security** y coordinar la integración con la API de **OpenAI**.
- **Capa de Datos (Persistencia):** Se utiliza **Spring Data JPA** para interactuar con una base de datos relacional **PostgreSQL**, garantizando la integridad y consistencia de toda la información relativa a jugadores, usuarios y mensajes.

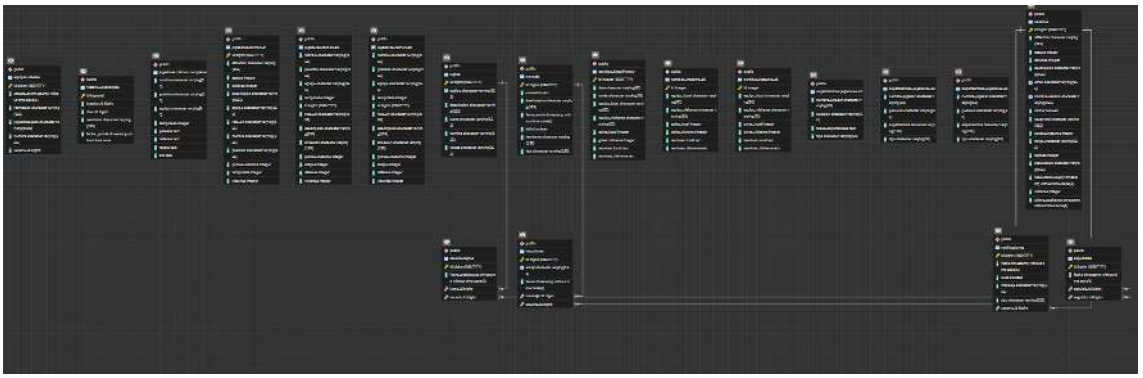


"Diagrama Entidad-Relación: Estructura de la base de datos PostgreSQL y relaciones de integridad"

4.2 Modelo de datos

El diseño de la base de datos es de tipo relacional y se ha normalizado para optimizar las consultas y evitar la redundancia de información. Las entidades principales que integran el sistema son:

- **Usuario:** Almacena las credenciales cifradas y el rol de acceso (Gestionado por Spring Security).
- **Jugador:** Contiene toda la información estadística, la posición en el campo y la descripción táctica detallada.
- **Equipo/Temporada:** Relaciona a los jugadores con sus respectivos clubes y los momentos cronológicos de la serie.
- **Mensaje/Chat:** Gestiona la persistencia de las conversaciones y las reacciones de los usuarios en tiempo real.



“Diagrama Entidad-Relación: Estructura del esquema relacional en PostgreSQL e integridad de datos entre entidades”

4.3 Diseño de interfaz

El diseño visual se ha concebido bajo una estética "**Cyberpunk / HUD**", utilizando contrastos de colores neón (azul cian y naranja) sobre fondos oscuros para simular una terminal táctica avanzada. Los pilares del diseño son:

- **UX (Experiencia de Usuario):** Se ha priorizado la navegación intuitiva con una barra de navegación fija y un sistema de "retorno" que mantiene el contexto del usuario (como la persistencia de la temporada seleccionada).
- **Diseño Responsive:** La interfaz se adapta a diferentes resoluciones mediante *Media Queries*, asegurando que la base de datos de jugadores sea consultable tanto en escritorio como en dispositivos móviles.

- **Componentes Visuales:** Se utilizan líneas de escaneo (*scan-lines*), rejillas digitales y animaciones CSS para reforzar la inmersión en el universo tecnológico de Inazuma Eleven.

“Interfaz de usuario con estética HUD: Implementación de diseño responsive y paleta de colores neón.”

5. Desarrollo

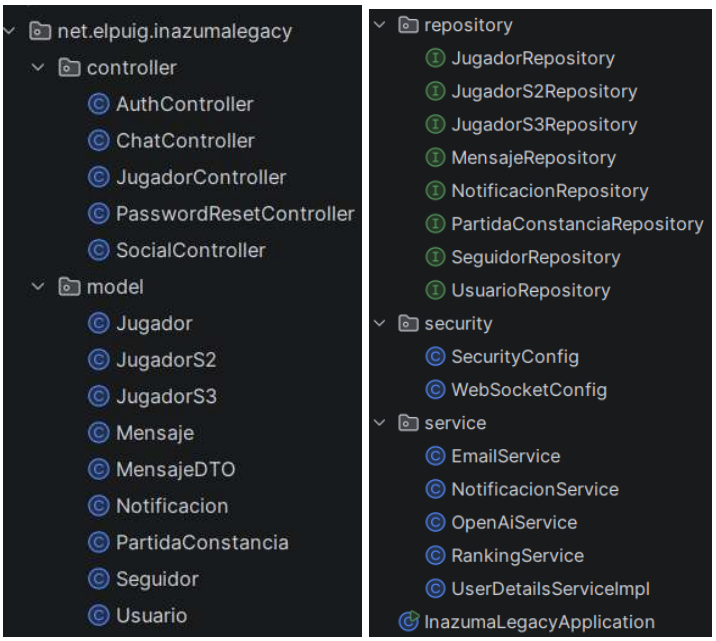
5.1 Estructura del proyecto

La implementación técnica de **Inazuma Legacy** se ha realizado bajo el ecosistema de **Spring Boot**, utilizando **Maven** como gestor de dependencias y construcción. Se ha adoptado una arquitectura en capas que separa claramente la responsabilidad de cada componente, facilitando la mantenibilidad y permitiendo una escalabilidad modular del sistema.

A continuación, se detalla la organización interna del código y la función crítica de cada paquete dentro del flujo de datos:

- **com.inazuma.controller (Capa de Presentación):** Actúa como el punto de entrada de la aplicación. Aquí se gestionan los controladores de **Spring MVC** (como ChatController, PlayerController y AuthController…). Su función es interceptar las peticiones HTTP, gestionar el intercambio de mensajes mediante protocolos **STOMP/WebSocket** y retornar las vistas o respuestas JSON correspondientes.
- **com.inazuma.model (Capa de Datos):** Contiene las entidades de **JPA** que representan fielmente el esquema de la base de datos PostgreSQL. En esta capa se definen las relaciones (OneToMany, ManyToMany) entre usuarios, jugadores y equipos, utilizando anotaciones de **Lombok** para reducir el código redundante y mejorar la legibilidad.
- **com.inazuma.repository (Capa de Persistencia):** Implementa interfaces que extienden de JpaRepository. Esta capa permite realizar operaciones CRUD (Crear, Leer, Actualizar, Borrar) de forma eficiente y segura, abstrayendo la complejidad de las consultas SQL mediante el uso de **Spring Data JPA**.
- **com.inazuma.service (Capa de Negocio):** Es el "cerebro" de la aplicación. Aquí reside la lógica compleja, como el algoritmo de simulación de partidos (Pachangas), el procesamiento de mensajes del chat y la gestión de las llamadas a la **API de OpenAI**. Al separar la lógica del controlador, se garantiza que el código sea testeable y reutilizable.
- **com.inazuma.config (Capa de Configuración):** Incluye las clases necesarias para el despliegue del sistema, como la configuración de **Spring Security** (seguridad y cifrado), la habilitación de los *endpoints* de WebSockets y los parámetros de conexión con el entorno cloud en **Railway**.

- **resources/templates y static (Frontend):** Se utiliza **Thymeleaf** como motor de plantillas para renderizar contenido dinámico desde el servidor. Los archivos estáticos gestionan la identidad visual del proyecto mediante **CSS con estética Cyberpunk**, animaciones de interfaz y la lógica de cliente en JavaScript para la comunicación en tiempo real.



“Estructura de paquetes del proyecto Inazuma Legacy siguiendo el estándar de arquitectura Spring Boot.”

5.2 Implementación de funcionalidades

En este apartado se detallan las soluciones técnicas aplicadas para cumplir con los objetivos del proyecto, centrándose en los módulos que han supuesto un mayor reto de programación y lógica de negocio. Para garantizar la robustez de **Inazuma Legacy**, se han implementado las siguientes funcionalidades principales:

- **Comunicación en Tiempo Real mediante WebSockets:** Se ha integrado el protocolo **STOMP** sobre WebSockets para permitir una interacción bidireccional permanente entre el cliente y el servidor. A diferencia del protocolo HTTP convencional, esta arquitectura utiliza un *Message Broker* interno en Spring Boot que gestiona diferentes destinos de mensajería (`/topic` para salas globales y `/user` para chats privados). Esto asegura que los paquetes de datos se entreguen de forma instantánea a los destinatarios suscritos, eliminando la latencia y la necesidad de refrescar la interfaz de usuario.
- **Servicio de Consultoría Inteligente (GPT-4o):** Se ha desarrollado un componente específico en la capa de servicio que encapsula las peticiones a la **API de OpenAI**. El reto técnico residió en el *Prompt Engineering*, diseñando un contexto de sistema que define la personalidad y el conocimiento técnico del asistente sobre la franquicia. El servicio procesa las consultas de forma asíncrona, manteniendo la coherencia temática y devolviendo respuestas que se renderizan dinámicamente en el frontend mediante JavaScript.
- **Seguridad Perimetral y Gestión de Roles:** La protección de los *endpoints* y la lógica de negocio se fundamenta en una configuración avanzada de **Spring Security**. Se han implementado filtros de autenticación e interceptores que validan el estado de la sesión en cada petición. Mediante el uso de un `AccessDeniedHandler` personalizado y utilidades de **Thymeleaf Extras**, el sistema sirve contenido dinámico de forma segura, redirigiendo a los usuarios no autenticados y protegiendo las funcionalidades de escritura en la base de datos (como el creador de equipos o el historial).
- **Motor de Simulación y Lógica de "Pachangas":** Se ha programado un algoritmo de juego que procesa las estadísticas almacenadas en la base de datos para calcular resultados de encuentros virtuales. Este módulo interactúa directamente con la capa de servicio para comparar atributos de ataque, defensa y rapidez de los jugadores seleccionados, generando una experiencia competitiva basada en los datos técnicos de los personajes.
- **Persistencia y Relaciones Complejas (JPA):** Para la gestión del historial y el ranking dinámico, se ha utilizado **Spring Data JPA** sobre una base de datos **PostgreSQL**. Se han definido relaciones `@ManyToOne` y `@ManyToMany` que vinculan a cada usuario con sus

alineaciones personalizadas y su registro de partidos. Esta estructura permite realizar consultas optimizadas para actualizar el escalafón global de usuarios en tiempo real según su rendimiento y actividad.

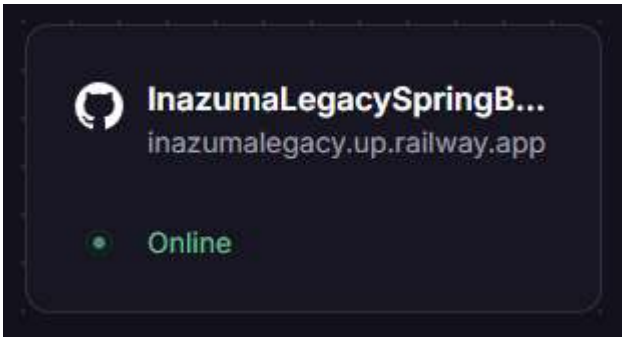
```java
package net.elpuig.inazumalegacy.security;

import ...

@Configuration  Victor
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Override  1 usage  Victor
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker( ...destinationPrefixes: "/topic", "/queue", "/user");
        config.setApplicationDestinationPrefixes("/app");
        config.setUserDestinationPrefix("/user");
    }

    @Override  1 usage  Victor
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint( ...paths: "/ws-inazuma").setAllowedOriginPatterns("*").withSockJS();
    }
}
```

“Implementación de la clase WebSocketConfig para la gestión de mensajes mediante el protocolo STOMP y SockJS.”

5.3 Pruebas

Para asegurar que **Inazuma Legacy** es una plataforma fiable y funcional, se ha llevado a cabo un plan de pruebas integral que abarca desde la lógica interna del código hasta la experiencia del usuario final:

- **Pruebas Unitarias y de Lógica:** Se ha verificado el correcto funcionamiento de los controladores y servicios de **Spring Boot**. Se han realizado pruebas específicas sobre el sistema de filtrado de jugadores por temporada y posición, asegurando que la base de datos responda con los registros exactos solicitados.
- **Pruebas de Integración con APIs:** Se ha testeado la conectividad con la API de **OpenAI** para garantizar que el asistente virtual reciba el contexto adecuado y devuelva respuestas coherentes. Asimismo, se ha validado la persistencia de datos en **PostgreSQL** mediante operaciones **CRUD** (Crear, Leer, Actualizar, Borrar) en el creador de equipos.
- **Pruebas de Tiempo Real (WebSocket):** Se han utilizado diversas instancias del navegador simultáneamente para comprobar que el protocolo **STOMP** distribuye los mensajes y las reacciones del chat sin retardos ni pérdida de paquetes.
- **Pruebas de Despliegue y Entorno:** Mediante los registros (*logs*) de **Railway**, se ha monitorizado el proceso de *build* y *deploy* para identificar posibles conflictos con las variables de entorno o la conexión remota a la base de datos.



“Monitorización de logs y validación del despliegue continuo en el entorno de producción."

6. Conclusiones

6.1 Conclusiones generales

El desarrollo de **Inazuma Legacy** ha sido una experiencia muy enriquecedora tanto a nivel académico como profesional. A nivel técnico, el proyecto ha permitido profundizar en la arquitectura web con **Spring Boot**, comprendiendo cómo se estructuran las capas de una aplicación real: controladores, servicios, repositorios y seguridad.

Paralelamente, la integración de APIs externas como la de **OpenAI** y la implementación del chat en tiempo real con **WebSocket** han supuesto un reto que ha ampliado significativamente la capacidad de trabajar con tecnologías modernas y asíncronas. Además, el proyecto ha obligado a tomar decisiones en todas las capas del desarrollo, desde el diseño de la base de datos con **PostgreSQL** hasta la maquetación del *frontend* con **Thymeleaf**, ofreciendo una visión global y transversal de lo que supone construir una aplicación web completa de principio a fin.

Profesionalmente, esta experiencia refuerza la capacidad de gestionar un proyecto real de forma autónoma, tomando decisiones tecnológicas justificadas y resolviendo problemas complejos de integración, lo que supone una preparación sólida para la etapa laboral en el sector IT.

6.2 Consecución de objetivos

La gran mayoría de los objetivos planteados al inicio del proyecto se han alcanzado satisfactoriamente. El sistema de consulta de jugadores, el chat en tiempo real, el ranking y el modo de partida (*pachanga*) funcionan de manera correcta y completa.

El único objetivo que no se ha podido completar de forma plena ha sido el **sistema de recuperación de contraseña** por correo electrónico. Aunque la funcionalidad está parcialmente implementada con el envío de *tokens* por email, no ha sido posible estabilizarla del todo dentro del plazo disponible. Esta limitación no afecta al núcleo funcional de la aplicación, pero es un aspecto que se considera prioritario de cara a futuras versiones.

6.3 Valoración de la metodología y planificación

El seguimiento de la planificación ha sido satisfactorio en términos generales, aunque a lo largo del desarrollo han surgido cambios y retrasos inevitables. Los principales retrasos se han producido en la **integración de funcionalidades complejas** como el WebSocket y la conexión con la API externa de IA, que han requerido más tiempo de investigación y depuración (*debugging*) del previsto inicialmente.

A pesar de estos ajustes, el proyecto se ha podido finalizar dentro del plazo establecido, entregando un producto funcional y coherente con los objetivos marcados al inicio del ciclo.

6.4 Visión de futuro

Inazuma Legacy tiene un recorrido de mejora y ampliación considerable. En primer lugar, sería necesario estabilizar y completar las funcionalidades que han quedado pendientes, especialmente el **sistema de recuperación de contraseña**, para garantizar una experiencia de usuario completa y robusta.

Más allá de las correcciones, la línea de futuro más ambiciosa sería el desarrollo de una **aplicación móvil nativa** para Android e iOS, que permitiera a los usuarios acceder a la plataforma de forma optimizada desde cualquier dispositivo. Otras mejoras que se añadirían con más tiempo serían un sistema de partidas más elaborado, funcionalidades sociales adicionales como eventos o grupos temáticos, y una expansión del catálogo de jugadores para cubrir más temporadas de la franquicia Inazuma Eleven.

7. Glosario

- **API (Application Programming Interface):** Conjunto de definiciones y protocolos que permiten la comunicación entre diferentes sistemas de software. En este proyecto se utiliza para conectar la aplicación con el servicio de IA de OpenAI.
- **Spring Boot:** Framework de Java que facilita la creación de aplicaciones web con configuración mínima, proporcionando un servidor integrado y gestión automática de dependencias.
- **Spring Security:** Módulo de Spring que gestiona la autenticación y la autorización de los usuarios dentro de la aplicación.
- **Thymeleaf:** Motor de plantillas para Java que permite generar páginas HTML5 dinámicas desde el servidor.
- **WebSocket:** Protocolo de comunicación bidireccional que permite el intercambio de datos en tiempo real entre el servidor y el cliente, utilizado en el sistema de chat de la aplicación.
- **STOMP (Simple Text Oriented Messaging Protocol):** Protocolo de mensajería que se utiliza sobre WebSocket para estructurar el envío y recepción de mensajes en el chat.
- **SockJS:** Biblioteca JavaScript que proporciona una capa de compatibilidad para WebSocket en navegadores que no lo soportan de forma nativa.
- **JPA (Java Persistence API):** Especificación de Java para la gestión de datos relacionales mediante ORM, que permite trabajar con la base de datos utilizando objetos Java.
- **ORM (Object-Relational Mapping):** Técnica que permite convertir datos entre el sistema de tipos de un lenguaje orientado a objetos y una base de datos relacional.
- **PostgreSQL:** Sistema de gestión de bases de datos relacional de código abierto utilizado para almacenar todos los datos de la aplicación.
- **Lombok:** Biblioteca de Java que reduce el código repetitivo generando automáticamente *getters*, *setters* y constructores mediante anotaciones.
- **IA (Inteligencia Artificial):** En el contexto de este proyecto, hace referencia al uso del modelo GPT-4o de OpenAI para responder preguntas de los usuarios dentro del chat.
- **Pachanga:** Modo de juego de la aplicación que permite a los usuarios enfrentarse entre sí basándose en sus estadísticas (ataque, defensa, rapidez).
- **Token:** Cadena de texto única y temporal generada por el sistema para verificar la identidad de un usuario, utilizada en el proceso de recuperación de contraseña.
- **Backend:** Parte del software que se ejecuta en el servidor y gestiona la lógica de negocio, el acceso a la base de datos y la seguridad.
- **Frontend:** Parte del software que se ejecuta en el navegador del usuario y define la interfaz visual con la que este interactúa.

8. Bibliografía

[1] **Spring Boot Documentation.** *Spring Boot Reference Documentation.* Pivotal Software. Consultado en noviembre de 2025. Disponible en: https://docs.spring.io/spring-boot/docs/current/reference/html/

[2] **OpenAI.** *OpenAI API Reference.* OpenAI. Consultado en febrero de 2026. Disponible en: https://platform.openai.com/docs/api-reference

[3] **Baeldung.** *Introduction to Spring Security.* Consultado en abril de 2025. Disponible en: https://www.baeldung.com/spring-security-intro

[4] **Baeldung.** *Using WebSocket to build an interactive web application.* Consultado en abril de 2025. Disponible en: https://www.baeldung.com/websockets-spring

[5] **Thymeleaf.** *Thymeleaf Official Documentation.* Consultado en marzo de 2025. Disponible en: https://www.thymeleaf.org/documentation.html

[6] **PostgreSQL Global Development Group.** *PostgreSQL Documentation.* Consultado en febrero de 2025. Disponible en: https://www.postgresql.org/docs/

[7] **Project Lombok.** *Lombok Official Documentation.* Consultado en marzo de 2025. Disponible en: https://projectlombok.org/features/

9. Anexos

Este proyecto no incluye anexos adicionales. Toda la información relevante sobre la arquitectura, las funcionalidades y el proceso de desarrollo queda recogida dentro de los capítulos de la presente memoria.

Nota sobre el uso de inteligencia artificial

En el desarrollo del proyecto Inazuma Legacy, se ha realizado un uso consciente y responsable de herramientas de inteligencia artificial generativa, principalmente Gemini y ChatGPT, como soporte al proceso de creación y depuración.

El uso de estas herramientas se ha centrado en los siguientes aspectos:

- Asistencia en la programación: Se han utilizado para la resolución de errores puntuales en la configuración de Spring Security y en la implementación del protocolo STOMP/WebSockets, así como para optimizar fragmentos de código Java y vistas en Thymeleaf.
- Diseño y Maquetación: Han servido de soporte para la generación de la estética visual "Cyberpunk" del CSS y para la estructuración de la documentación técnica de esta memoria.
- Integración de la API de OpenAI: Se ha consultado la mejor manera de estructurar los *prompts* de contexto para el asistente virtual del proyecto, con el fin de asegurar respuestas precisas sobre el universo de Inazuma Eleven.
- Declaración de autoría: Todo el contenido generado o sugerido por la IA ha sido revisado críticamente, corregido y validado por el autor del proyecto.

Licencia