Institut Puig Castellar
Santa Coloma de Gramenet

BitHub

(Projecte de desenvolupament)

CFGS Desenvolupament d'Aplicacions Web

Autores: Adrián Luque y Justin Alvarez
Grupo: Daw2b
CFGS Desenvolupament d'Aplicacions Web

Resumen del proyecto:

Este proyecto se basa en el desarrollo de una plataforma web educativa con estilo arcade, diseñada para alojar diversos minijuegos interactivos. La finalidad principal del proyecto es ayudar a los usuarios a mejorar y asentar sus conocimientos técnicos en informática y fundamentos de programación.

El objetivo de este proyecto es crear una plataforma completa desde cero, implementando los conocimientos adquiridos durante el grado. Esto abarca tanto el desarrollo web orientada al usuario, como la creación de una arquitectura de servidor y gestión de bases de datos. Para conseguir este objetivo se utiliza una metodología ágil utilizando Kanban para la gestión de tareas y Github para el control de versiones.

Como conclusión, este proyecto ha permitido profundizar en el desarrollo web, concretamente en la separación de tecnologías de cliente (Frontend) y servidor (Backend). Este proyecto además continuará en constante desarrollo con nuevas actualizaciones y la incorporación de nuevos minijuegos una vez sacada su versión inicial.

Palabras clave(entre 4 y 8):

Minijuegos de programación, Aprender a programar jugando, Juegos de lógica, Retos de código, Arcade para programadores, Practicar algoritmos

Abstract (in English, 250 words or less):

This project is based on the development of an arcade-style educational web platform, designed to host various interactive minigames. The main purpose of the project is to help users improve and consolidate their technical knowledge in computer science and programming fundamentals.

The objective of this project is to create a complete platform from scratch, implementing the knowledge acquired during the degree. This encompasses both user-oriented web development, as well as the creation of a server architecture and

database management. To achieve this goal, an agile methodology is used, utilizing Kanban for task management and GitHub for version control.

In conclusion, this project has allowed for a deeper understanding of web development, specifically in the separation of client (Frontend) and server (Backend) technologies. Furthermore, this project will continue in constant development with new updates and the addition of new minigames once its initial version is released.

Keywords (entre 4 i 8):

Programming minigames, Learning to code through play, Logic games, Coding challenges, Arcade for programmers, Algorithm practice.

Índice

1 Introducción

1.1 Contexto

Este proyecto se basa en el desarrollo de una plataforma web interactiva con estética arcade, la cual aloja un conjunto de minijuegos educativos, tales como Logic Slots, Code-Jack 21, Decibit y LogicLink. El sistema integra mecánicas de juguetización , gestión de progreso mediante un sistema de puntuación y paneles de administración.

La elección de este enfoque responde al auge de la juguetización y de la educación y entretenimiento en el ámbito del aprendizaje. En los últimos años, el aprendizaje basado en el juego ha demostrado ser una de las metodologías más efectivas para asentar conocimientos lógicos y técnicos. La interacción constante, la superación de retos y la obtención de recompensas inmediatas logran transformar conceptos teóricos de programación e informática en una experiencia dinámica, reduciendo la frustración y aumentando el interés de los usuarios.

A nivel tecnológico, uno de los pilares del proyecto es la separación del sistema en dos grandes bloques: la parte visual del cliente (Frontend) y la lógica del servidor (Backend). El uso de herramientas modernas como Vue.js en el frontend permite que los minijuegos funcionen de forma rápida y fluida en el navegador, sin tiempos de carga molestos. Por otro lado, el servidor desarrollado en Java con Spring Boot se encarga de todo el trabajo invisible: proteger los datos de inicio de sesión, almacenar el progreso (los "bytes") en la base de datos de MySQL y registrar las estadísticas de las partidas. Toda la comunicación entre ambos lados se hace a través de una API REST, lo que mantiene el código ordenado y facilita mucho poder añadir nuevos juegos en el futuro sin romper lo que ya funciona.

1.2 Justificación

La elección de este proyecto responde a varios factores, destacando en primer lugar la creciente popularidad de los minijuegos web y del aprendizaje interactivo. Este tipo de aplicaciones se caracteriza por ofrecer partidas rápidas y entretenidas, mediante unas mecánicas que fomentan el volver a jugar. Los entornos educativos basados en el juego se han abierto un gran espacio en el sector gracias a su capacidad para enseñar de forma dinámica y evitar la monotonía, convirtiendo a este medio en una herramienta muy demandada. Por ello, el proyecto busca contribuir a este catálogo ofreciendo una experiencia atractiva y accesible para los usuarios.

Esta decisión se fundamenta, además, en el objetivo de construir un sistema completo. Este enfoque permite diseñar y programar sistemas propios, abarcando tanto la interfaz visual como la gestión del servidor y la base de datos, garantizando así que el resultado final se adapte exactamente a las necesidades de la plataforma y de sus futuros jugadores.

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar una página web la cual sirva a usuarios que tienen poco conocimiento de informática a aprender algunas de las bases sobre este tema, con un sistema de puntaje el cual sirva a los usuarios a motivarlos para que tengan mas ganas de aprender para así estar lo más arriba posible de la clasificación

1.3.2 Objetivos específicos

- Implementar un sistema de inicio de sesión y de registro para los usuarios.
- Desarrollar 4 juegos con la finalidad de que sirvan de aprendizaje de cara a los usuarios.
- Crear una base de datos funcional para poder controlar usuarios y sus datos, datos de los juegos, estadísticas.
- Desarrollar una interfaz de usuario clara, atractiva y responsive para cualquier dispositivo.
- Aplicar medidas de seguridad básicas (hash de contraseñas, protección SQL Injection, validaciones).
- Integrar todas las partes del sistema con un funcionamiento estable y eficiente.
- Elaborar la documentación tècnica correspondente.

1.4 Estrategia y planificación del proyecto

Decisión estratégica:

Desde el inicio del proyecto, se optó por desarrollar la plataforma web y los minijuegos completamente desde cero, prescindiendo del uso de plantillas prefabricadas o gestores de contenido externos. Esta decisión permite mantener un control absoluto sobre la arquitectura del sistema, asegurando que la integración entre la parte visual y el servidor se adapte exactamente a las necesidades de la plataforma, como la gestión del saldo de puntos ("bytes") y el registro de estadísticas en los paneles de administración.

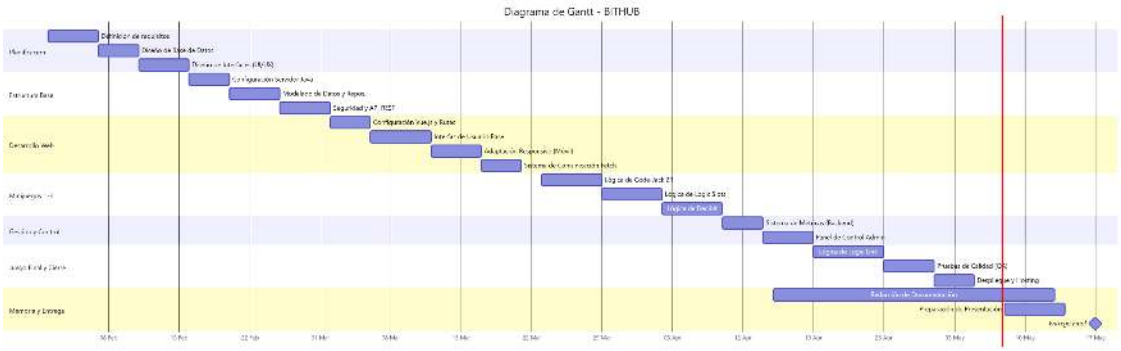
Estudio de viabilidad:

En cuanto a la viabilidad temporal, se trata de un proyecto ambicioso. El desarrollo de una página web completa y de varios minijuegos funcionales suele requerir plazos extensos de trabajo, convirtiendo la gestión del tiempo en el principal reto a superar durante su creación. A nivel económico, el proyecto no está pensado inicialmente como un producto comercial directo para generar ingresos mediante su venta. Sin embargo, su enfoque gratuito y educativo lo hace altamente viable para llegar a un público amplio. Debido a la popularidad de las plataformas de aprendizaje interactivo, la web se puede dar a conocer de manera rápida y eficaz, creando una base de usuarios sólida y dejando el terreno preparado para la publicación de futuras actualizaciones o la integración de nuevos juegos.

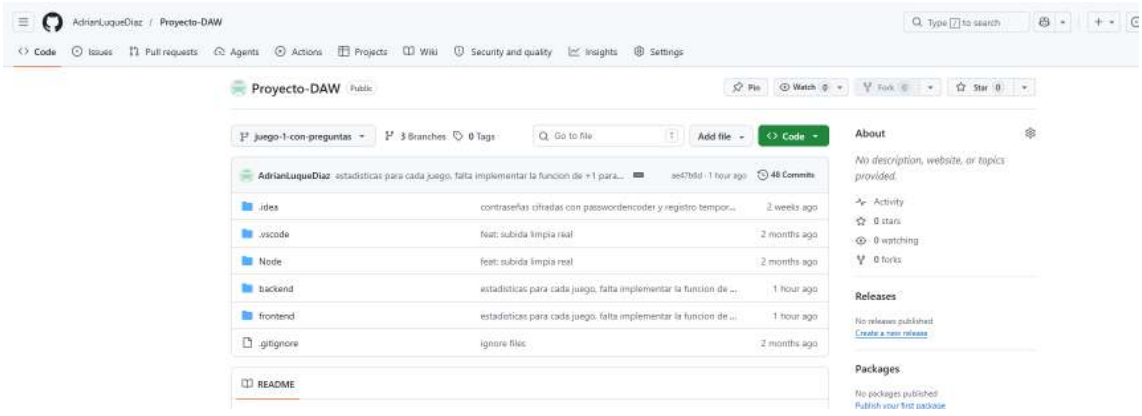
1.5 Metodología de trabajo

Para gestionar el desarrollo del proyecto se ha optado por una metodología organizada y estructurada, utilizando herramientas digitales para facilitar la planificación temporal y el seguimiento continuo del trabajo:

. **Diagrama de Gantt:** Para la planificación y el seguimiento de las tareas se ha elaborado un diagrama de Gantt. Esta herramienta gráfica permite organizar el proyecto en diferentes fases a lo largo del tiempo, estableciendo fechas de inicio y fin para cada actividad. De este modo, se obtiene una visión clara y global del progreso, lo que facilita la gestión del tiempo y ayuda a garantizar el cumplimiento de los plazos establecidos para la entrega final. (Nota: Aquí añadirías la figura de tu diagrama de Gantt).



. **GitHub:** Para el control de versiones del código se ha utilizado GitHub. Esta plataforma resulta fundamental en el desarrollo, ya que permite crear un repositorio centralizado con todo el código de la página web y de los minijuegos. Gracias a esto, se mantiene un historial seguro, ordenado y detallado de todos los cambios realizados, permitiendo volver a versiones anteriores si ocurre algún error durante la programación.



1.6 Estudio económico y presupuestario

El desarrollo de la plataforma BitHub ha priorizado el uso de tecnologías Open Source (código abierto) y servicios en la nube de capa gratuita para minimizar el impacto financiero inicial. No obstante, para dotar al proyecto de un enfoque realista y empresarial, se ha elaborado un presupuesto simulado que contempla tanto los

recursos materiales (hardware e infraestructura) como el coste de la mano de obra del equipo de desarrollo.

1.6.1 Costes de Infraestructura, Software y Hardware

La mayor parte de las herramientas de software utilizadas no han supuesto ningún coste. El coste principal en este apartado recae sobre la amortización del equipo físico utilizado por los desarrolladores y los gastos de despliegue web.

Concepto / Herramienta	***Tipo***	***Coste Estimado***
Visual Studio Code / IntelliJ	*IDE de Desarrollo*	*0,00 €*
Git y GitHub	*Control de Versiones*	*0,00 €*
Java (Spring Boot) / Node.js	*Entorno Servidor*	*0,00 €*
Vue.js / Vite / CSS	*Framework Frontend*	*0,00 €*
MySQL	*Base de Datos*	*0,00 €*
Vercel / Railway	*Hosting y Dominio (Anual)*	*20,00 €*
Estación de trabajo (PC 1)	*Hardware*	*750,00 €*
Estación de trabajo (PC 2)	*Hardware*	*700,00 €*
TOTAL INFRAESTRUCTURA		***1.470,00 €***

1.6.2 Costes de Personal (Mano de obra)

Para calcular el valor económico del desarrollo del software, se ha estimado el tiempo invertido en las distintas fases del ciclo de vida del proyecto (planificación, backend, frontend, pruebas y documentación).

Asumiendo el rol de dos desarrolladores Junior Full-Stack trabajando en el proyecto durante un periodo de 6 meses, el coste salarial se distribuiría de la siguiente manera basándonos en el mercado laboral actual:

Perfil Profesional	***Salario Mensual Estimado***	***Meses de dedicación***	***Coste Total***
Desarrollador 1 (Full-Stack)	*1.400,00 €*	*6*	*8.400,00 €*
Desarrollador 2 (Full-Stack)	*1.400,00 €*	*6*	*8.400,00 €*
TOTAL PERSONAL			***16.800,00 €***

(Se estima que los costes de mantenimiento, soporte y actualizaciones de la plataforma tendrían un valor de entre 300 € y 600 € anuales una vez finalizado el desarrollo).

1.6.3 Presupuesto Total del Proyecto

Para la estimación de los costes salariales reflejados en este presupuesto, se han tomado como referencia los datos públicos de portales de empleo en España (como Glassdoor e InfoJobs) para perfiles de desarrollador Junior. Por su parte, los costes asociados a la infraestructura y alojamiento se han calculado a partir de las tarifas de los proveedores en la nube utilizados.

*Sumando los recursos técnicos (1.470 €) y la dedicación del equipo de desarrollo (16.800 €), el coste total final para la ejecución y puesta en marcha del proyecto asciende a **18.270 €**. Este presupuesto permitiría a un cliente real evaluar la viabilidad de financiar un proyecto de estas características en un entorno profesional.*

2 Descripción del proyecto

2.1 Análisis de requisitos

2.1.1 Requisitos funcionales

- *El sistema debe permitir el registro de nuevos usuarios mediante la **validación de datos**.*

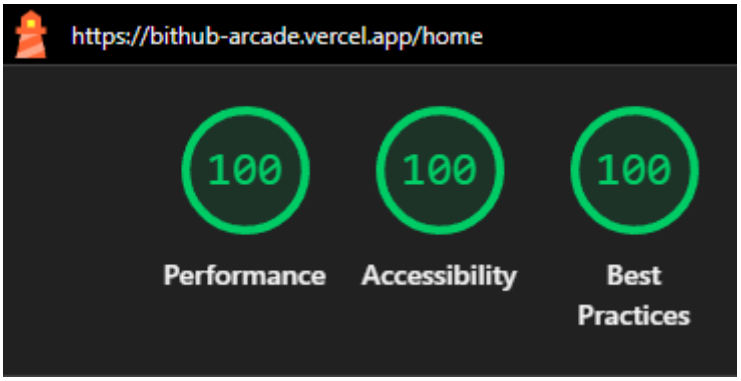
- *El sistema debe permitir el **inicio** y **cierre de sesión** de forma segura para los usuarios registrados (contraseñas hasheadas, borrar datos de la memoria del navegador, validación de casillas del registro).*
- *Cada usuario debe disponer de un **saldo inicial** de puntos virtuales ("bytes") al registrarse.*
- *El usuario debe poder **acceder** y **jugar** a los diferentes minijuegos educativos de la plataforma (Logic Slots, Code-Jack 21, DeciBit, Logic Link.).*
- *Los minijuegos deben funcionar con reglas lógicas claras y generar **resultados aleatorios** coherentes cuando la mecánica lo requiera (por ejemplo al barajar opciones o cartas).*
- *El sistema debe registrar las partidas jugadas y actualizar automáticamente las ganancias o pérdidas de bytes de cada usuario.*
- *El usuario debe poder consultar en pantalla su saldo actual de "bytes" y su nombre de jugador.*
- *El administrador debe tener acceso a un panel de control específico para visualizar estadísticas de los juegos y gestionar los datos de cada juego.*

2.1.2 Requisitos no funcionales

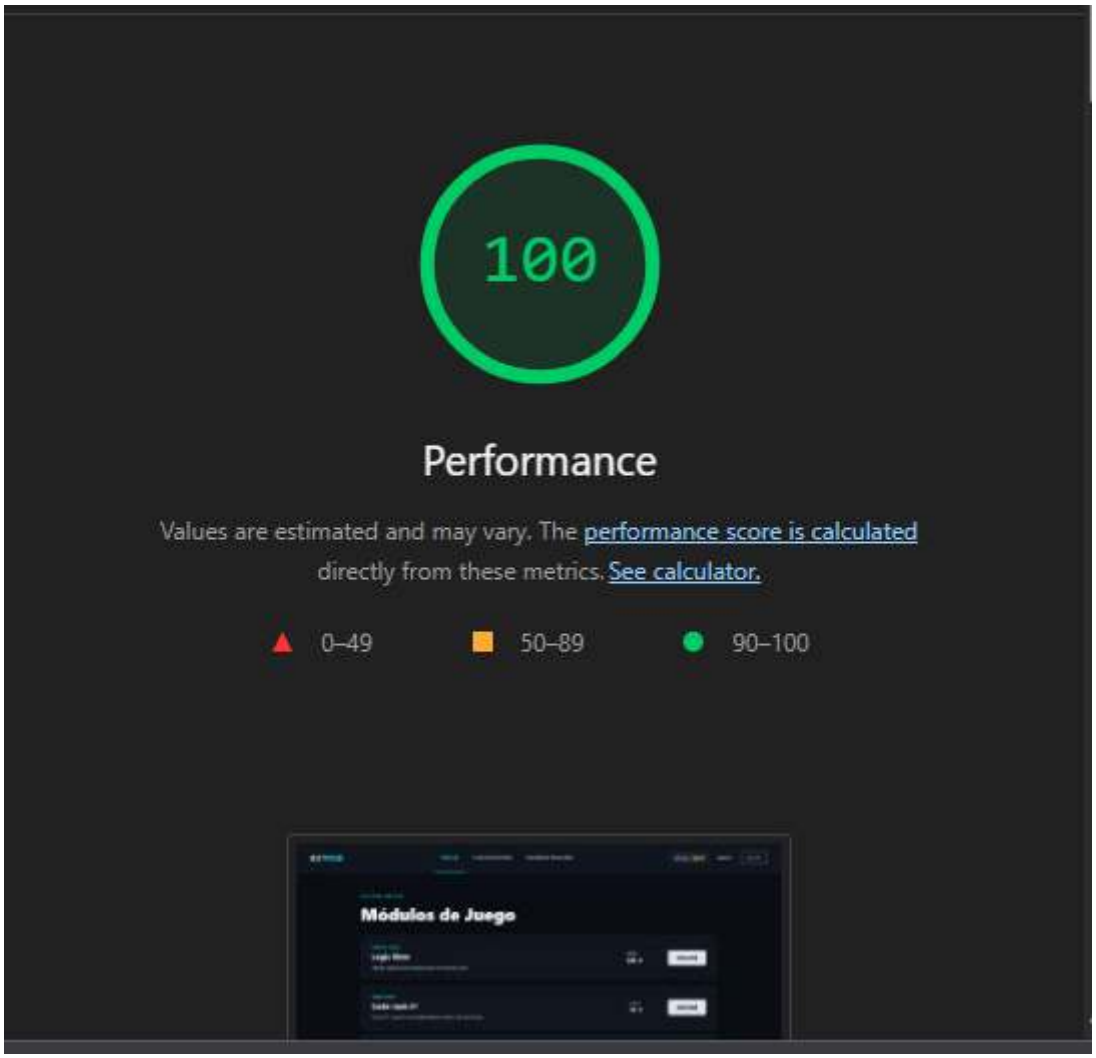
- *La aplicación debe ser **segura**, protegiendo obligatoriamente las **contraseñas** de los usuarios en la base de datos mediante **encriptación**.*
- *El sistema backend debe estar **protegido** contra **vulnerabilidades** y **ataques** comunes, como la inyección SQL. (gracias a Spring Boot).*
- *La interfaz visual (Frontend) debe ser **atractiva**, **intuitiva** y **fácil** de navegar para usuarios con pocos conocimientos informáticos.*
- *La plataforma web debe ser **adaptativa** (responsive) para que sea **accesible** y **visualmente** correcta desde diferentes tamaños de pantalla y dispositivos.*
- *El sistema debe garantizar una buena experiencia de usuario, con **tiempos** de carga **rápidos** y transiciones con **poco tiempo** de **espera** entre los minijuegos y los menús. (más explicado en el nuevo punto 2.1.3).*
- *El código fuente debe estar **estructurado** separando **cliente** y **servidor** para que resulte fácil de mantener y escalar en futuras actualizaciones.*

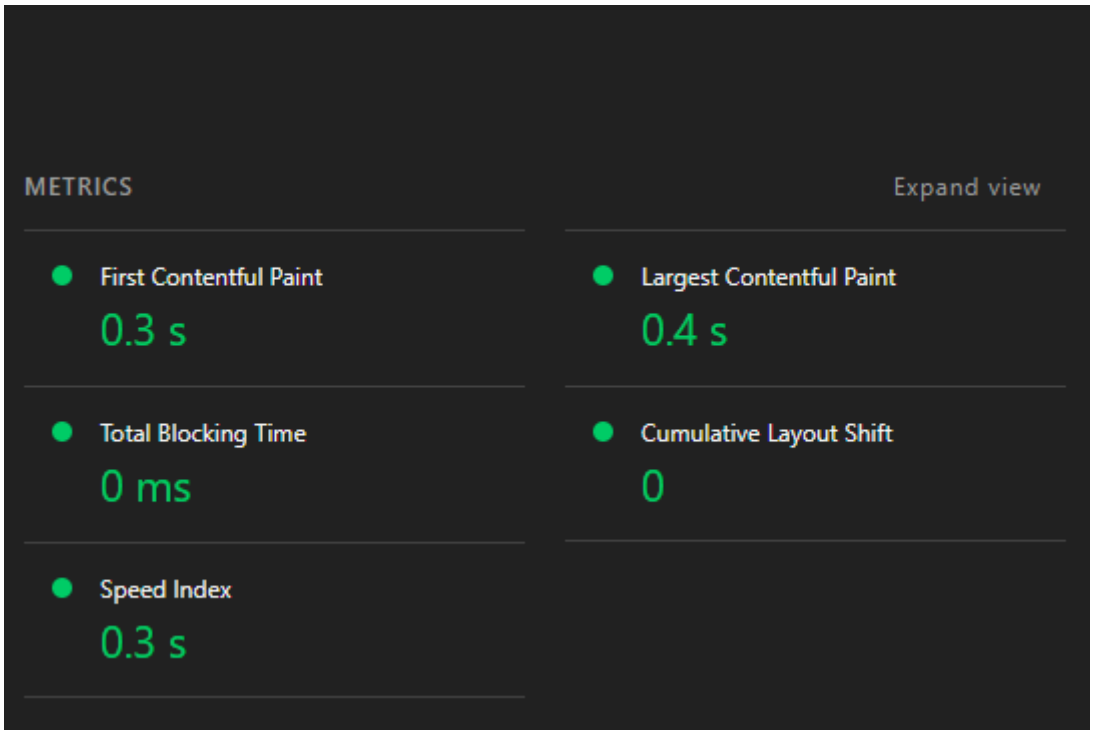
2.1.3 Evaluación de Rendimiento y Accesibilidad (Pruebas Lighthouse)

Para corroborar todo lo que se ha dicho como requisitos no funcionales, se ha utilizando la herramienta profesional Lighthouse. Los resultados obtenidos corroboran lo dicho en los requisitos no funcionales:

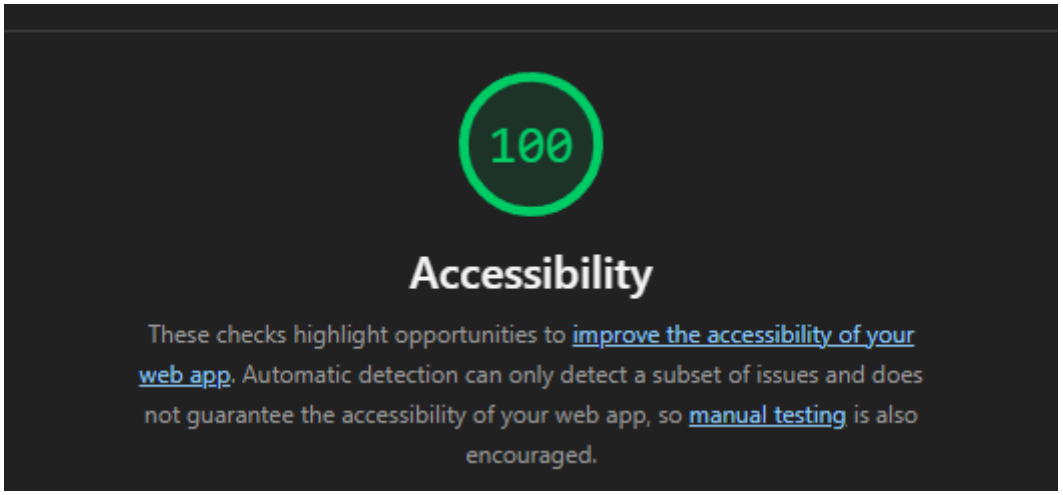


- ***Rendimiento (100/100):*** *Se ha alcanzado la puntuación máxima una vez desplegada la aplicación. Destacando un Tiempo de Bloqueo Total (TBT) de 0 ms y un despliegue del contenido principal (LCP) de 0.4 s. Con esto se garantiza que el sistema carga de manera ultrarrápida, no presenta congelamientos y responde de forma instantánea a cualquier interacción del usuario.*

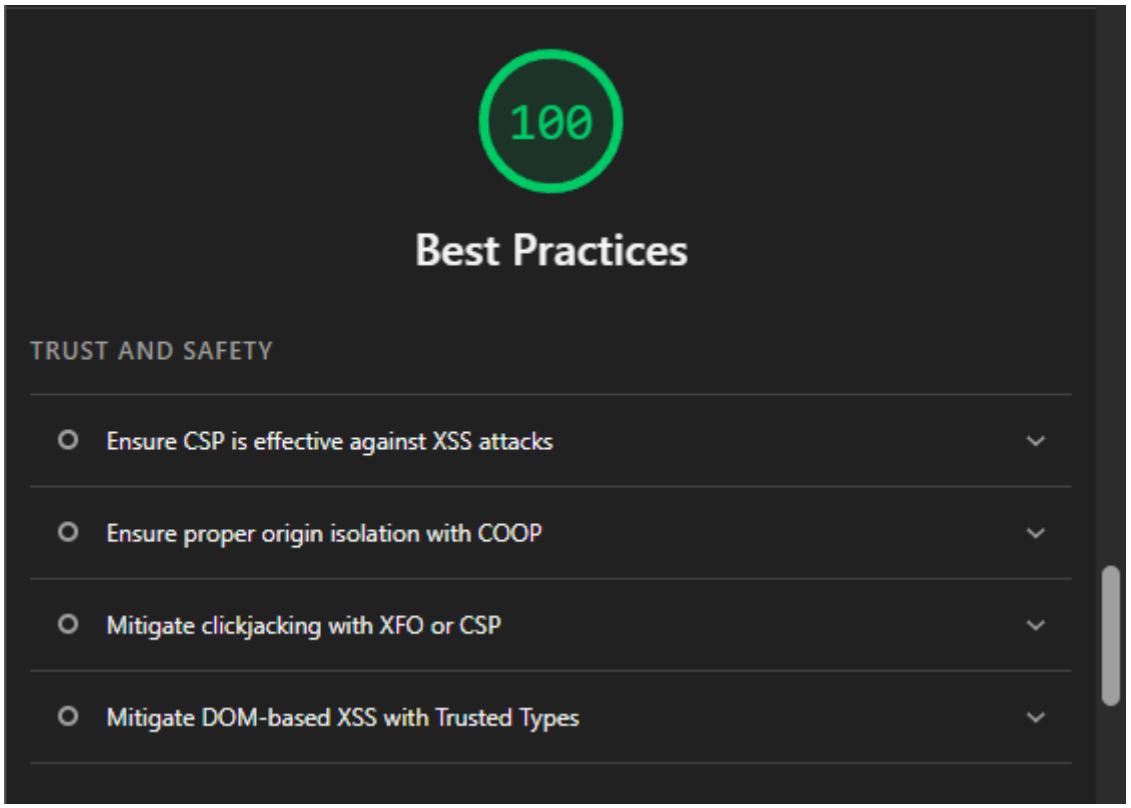




- ***Accesibilidad (~95/100 de media):*** *Se observan puntuaciones que van entre los 90 y los 100 puntos dependiendo de la página en la que estes. De este modo se confirma que la interfaz es intuitiva y accesible, resulta utilizable en dispositivos móviles y es compatible con tecnologías para sordos.*



- ***Buenas Prácticas (100/100):*** *Se ha obtenido la calificación más alta. Mediante este resultado se certifica que el proyecto se encuentra construido sobre buena arquitectura, hace uso de APIs modernas, asegura conexiones seguras mediante HTTPS (gracias a subirlo a un hosting), evita la implementación de librerías obsoletas y mantiene una consola del navegador completamente libre de errores.*



Para ofrecer un enfoque más exhaustivo sobre la usabilidad del proyecto, se ha elaborado un informe de pruebas de usabilidad, el cual puede consultarse en el apartado de Anexos.

2.1.4 Estudio previo de viabilidad tecnológica

Antes de iniciar el desarrollo técnico, se ha hecho necesaria una fase de investigación centrada en los siguientes puntos clave:

- ***Investigación sobre sistemas de autenticación:*** *Se ha analizado el funcionamiento de las sesiones de usuario y la persistencia de datos en el cliente (Frontend). Esto ha permitido asegurar que, una vez iniciado el acceso, la sesión se mantenga activa mientras el usuario navega por los distintos menús y juegos gracias a la gestión del estado de manera local mediante un almacén de autenticación (authStore).*
- ***Análisis de estructuras de bases de datos:*** *Se han estudiado modelos de bases de datos relacionales para organizar de forma eficiente la información de los usuarios, los retos de los minijuegos y el registro de estadísticas. Se han definido relaciones clave como la vinculación entre usuarios y sus puntuaciones, el historial de estadísticas y la estructura necesaria para que el servidor almacene y suministre los datos de los juegos (como las preguntas y respuestas correctas).*
- ***Comparación de frameworks y tecnologías:*** *Se ha realizado una comparativa entre distintas herramientas de desarrollo, optando finalmente por Vue.js para el frontend y Spring Boot para el backend, debido a su compatibilidad y facilidad para crear una api rest (el puente entre el frontend y el backend).*
- ***Estudio de medidas de seguridad web:*** *Se han analizado las vulnerabilidades más comunes, como la inyección SQL, para entender cómo*

las herramientas modernas (como los repositorios de Spring) protegen el sistema de forma automática.

2.2 Tecnologías

2.2.1 Comparativa de las tecnologías valoradas

Antes de empezar a programar, se han comparado diferentes opciones para ver cuál encaja mejor en una plataforma arcade educativa:

- **PHP / Node.js / Java (Spring Boot):** Se valoró PHP por ser muy fácil de usar y Node.js por ser más moderno. Sin embargo, se ha elegido **Java con Spring Boot** porque es muy seguro para crear la comunicación del servidor (API REST) y permite aplicar directamente la base aprendida durante el ciclo formativo.
- **MySQL / PostgreSQL:** Se comparó MySQL con PostgreSQL, que es una opción muy potente para proyectos enormes. Al final, se escogió **MySQL** porque es más sencilla, funciona muy bien para lo que necesita este proyecto y se conecta fácilmente con Spring Boot.
- **HTML/CSS/JS puro / Frameworks frontend (Vue.js):** Programar con código puro ayuda a controlar cada detalle desde cero, pero se ha decidido usar **Vue.js** porque permite avanzar mucho más rápido creando componentes reutilizables (como los menús o los botones). Además, hace que la web cargue de forma fluida (como una SPA) y facilita guardar el estado de los jugadores, como los puntos y la sesión abierta.

2.2.2 Tecnologías utilizadas

El conjunto de tecnologías con las que se ha desarrollado el proyecto es el siguiente:

Frontend (Interfaz del usuario)

- ***Vue.js:** Se encarga de mostrar la parte visual (menús, juegos) y permite que la web funcione de forma rápida sin tener que recargar la página entera cada vez que se hace clic.*
- ***Vite:** Herramienta que hace que el entorno de programación funcione mucho más rápido a la hora de compilar el código.*
- ***Pinia:** Se utiliza para guardar información temporal en el navegador (como el nombre del usuario conectado o sus bytes) sin tener que consultar a la base de datos de forma constante.*
- ***TypeScript:** Añade reglas al lenguaje JavaScript para detectar fallos y errores mientras se escribe el código.*

Backend (Servidor)

- ***Java y Spring Boot:** Son el motor principal del programa. Se encargan de recibir las peticiones de la página web, procesar la lógica de los juegos y devolver las respuestas correctas.*
- ***Spring Security:** Se ocupa de la seguridad del servidor, bloqueando accesos no autorizados y encriptando las contraseñas de los jugadores.*

Base de datos y Herramientas

- ***MySQL:*** *Sistema de base de datos donde se guarda de forma definitiva toda la información importante (usuarios, estadísticas, retos).*
- ***GitHub:*** *Plataforma que se ha usado para guardar copias de seguridad del código en la nube y mantener un registro de los cambios.*

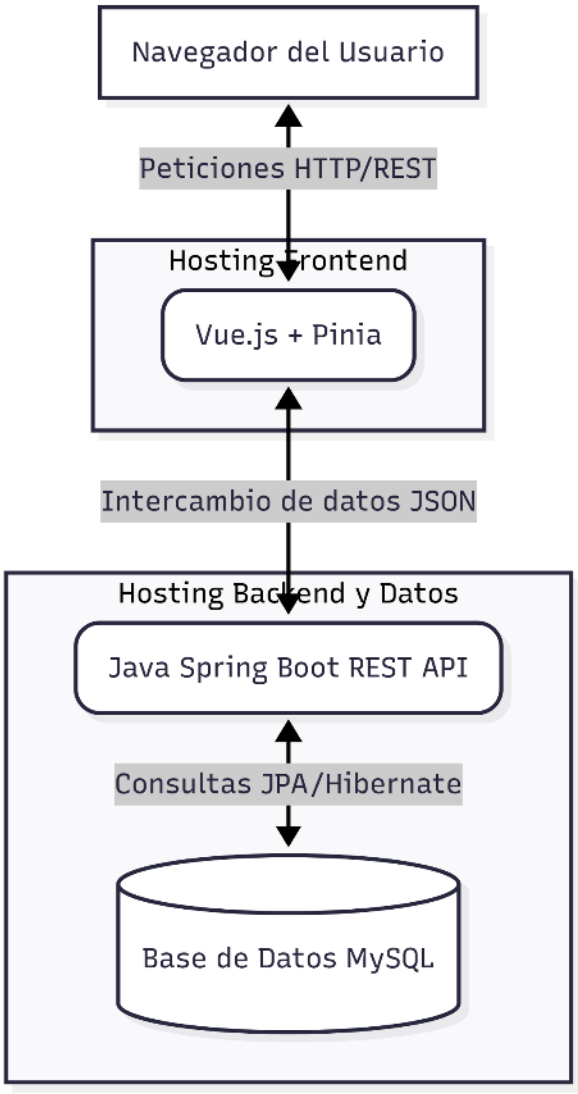
2.3 Estructura del proyecto

El proyecto se divide en dos grandes bloques independientes: el Frontend y el Backend. Esta separación permite que el código sea más fácil de mantener y organizar.

El sistema sigue una arquitectura distribuida basada en el modelo Cliente-Servidor, diseñada para separar las responsabilidades y mejorar la escalabilidad. La plataforma BitHub se divide en tres capas principales interconectadas:

- ***Capa de Presentación (Frontend):*** *Desarrollada con Vue.js y alojada en la plataforma Vercel. Es la encargada de renderizar la interfaz gráfica, gestionar el estado del usuario con Pinia y consumir los datos del servidor.*
- ***Capa de Negocio (Backend):*** *Desarrollada con Java Spring Boot y alojada en la infraestructura de Railway. Expone una API REST que procesa la lógica central de los minijuegos, valida las reglas de seguridad y gestiona los saldos (Bytes).*
- ***Capa de Datos (Base de Datos):*** *Consiste en un motor relacional MySQL, alojado también en un servicio independiente dentro de Railway. Se encarga de la persistencia de toda la información (usuarios, estadísticas, preguntas y cartas).*

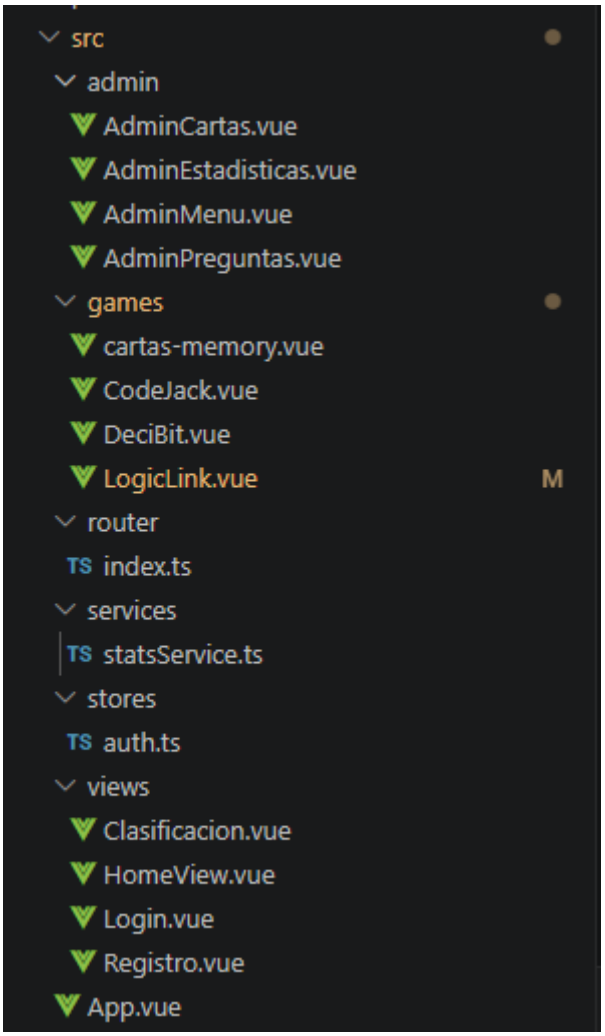
La comunicación entre la capa de presentación (Frontend) y la capa de negocio (Backend) se realiza de forma asíncrona mediante peticiones HTTP , intercambiando la información de forma segura en formato JSON.



2.3.1 Estructura del Frontend (Vue.js)

La parte visual se organiza siguiendo la estructura recomendada por Vue 3, separando la lógica de los datos de los elementos visuales:

- ***src/views/**: Contiene las páginas principales de la aplicación (Home, Login, Registro, Clasificación y las pantallas de cada minijuego).*
- ***src/stores/**: Aquí se ubica el authStore, que se encarga de gestionar los datos de la sesión del usuario.*
- ***src/services/**: Contiene las funciones que se comunican con el servidor (variable de entorno).*
- ***src/router/**: Es el archivo que define las rutas de la web, permitiendo que el usuario cambie de página de forma fluida.*
- ***src/admin/**: Contiene las páginas de administrador (Paneles de admin, estadísticas).*
- ***src/auth/**: Contiene el archivo de configuración para el LocalStorage.*

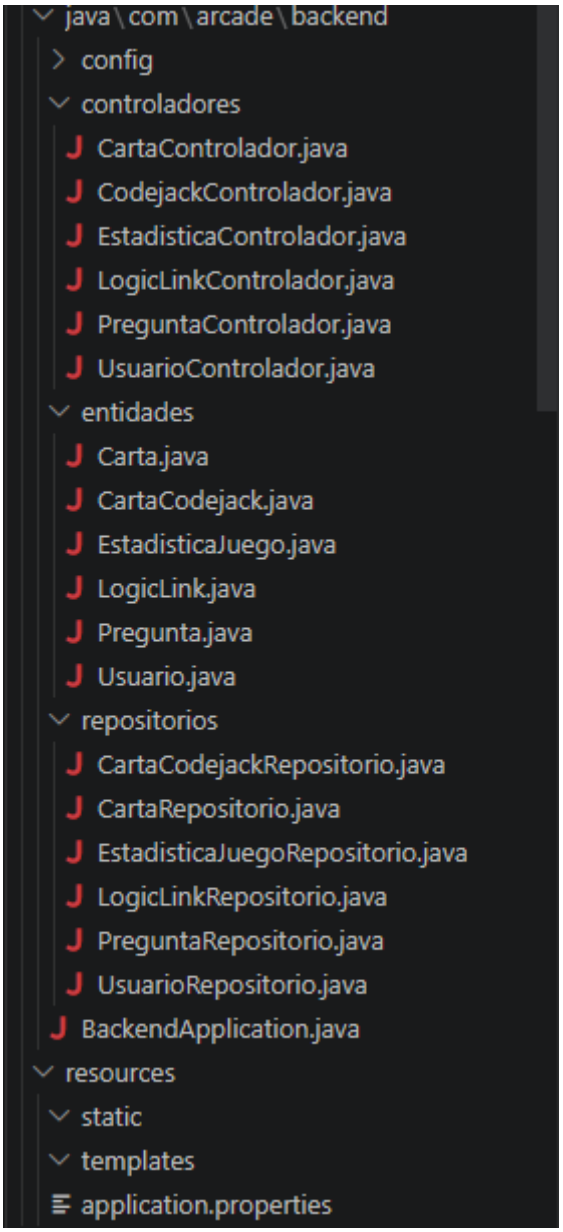


2.3.2 Estructura del Backend (Spring Boot)

El servidor se ha estructurado siguiendo el patrón de capas:

- ***controladores/**: Actúan como los puntos de entrada de la API REST. Archivos como UsuarioControlador.java o LogicLinkControlador.java reciben las peticiones de la página web, ejecutan la lógica necesaria y devuelven la respuesta al usuario.*
- ***entidades/**: Definen la estructura de los objetos del programa y establecen cómo se mapean y corresponden exactamente con las tablas de la base de datos MySQL.*
- ***repositorios/**: Contienen las interfaces que utilizan Spring Data JPA para comunicarse directamente con la base de datos, facilitando las operaciones de información cómo encontrar el nombre de algún juego, encontrar el id de una pregunta…*
- ***resources/**: Carpeta de Spring Boot que contiene el archivo application.properties, donde se sitúan los parámetros de configuración, como las credenciales de acceso a MySQL y el puerto del servidor.*
- ***config/**: Contiene las clases de configuración del servidor, como los permisos de conexión para permitir que el frontend se comunique con la api sin bloqueos*

de seguridad, aparte es quien se encarga de hashear la contraseña del usuario en la base de datos.



2.4 Descripción de los componentes

La plataforma se ha dividido en diferentes piezas para garantizar que el sistema opere de forma eficiente. A continuación, se detalla, de forma esquemática y conceptual, la función principal de cada uno de los componentes desarrollados.

2.4.1 Componente 1: Sistema de usuarios y seguridad

Este componente se encarga de gestionar el registro, el inicio de sesión y la seguridad de las credenciales de los jugadores.

- ***Funcionamiento:*** *Valida los datos introducidos y mantiene recordada la sesión mientras el usuario navega por la plataforma.*
- ***Tecnologías implicadas:*** *Utiliza* ***Spring Security*** *en el backend (configurado con BCryptPasswordEncoder) para encriptar las contraseñas antes de guardarlas, mientras que el uso de Spring Data JPA previene ataques de inyección SQL. En el frontend, utiliza* ***Pinia (authStore)*** *respaldado por el* ***LocalStorage*** *del navegador para mantener la sesión activa sin perder los datos al recargar la página.*

2.4.2 Componente 2: Motor de minijuegos

Es lo principal de la plataforma. Implementa la lógica de cada uno de los retos, generando las pruebas, validando las respuestas del jugador y calculando las recompensas o penalizaciones.

- ***Funcionamiento:*** *Se compone de diferentes módulos de juego (como Logic Link para sintaxis de código, DeciBit para conversiones matemáticas o CodeJack para lógica con cartas). Al finalizar una partida, envía una petición al servidor para actualizar el saldo de bytes del jugador.*
- ***Tecnologías implicadas:*** *Desarrollado íntegramente con componentes de* ***Vue.js*** *y* ***TypeScript*** *para la lógica de interfaz, comunicándose mediante los controladores de* ***Spring Boot****.*

2.4.3 Componente 3: Base de datos y Estadísticas

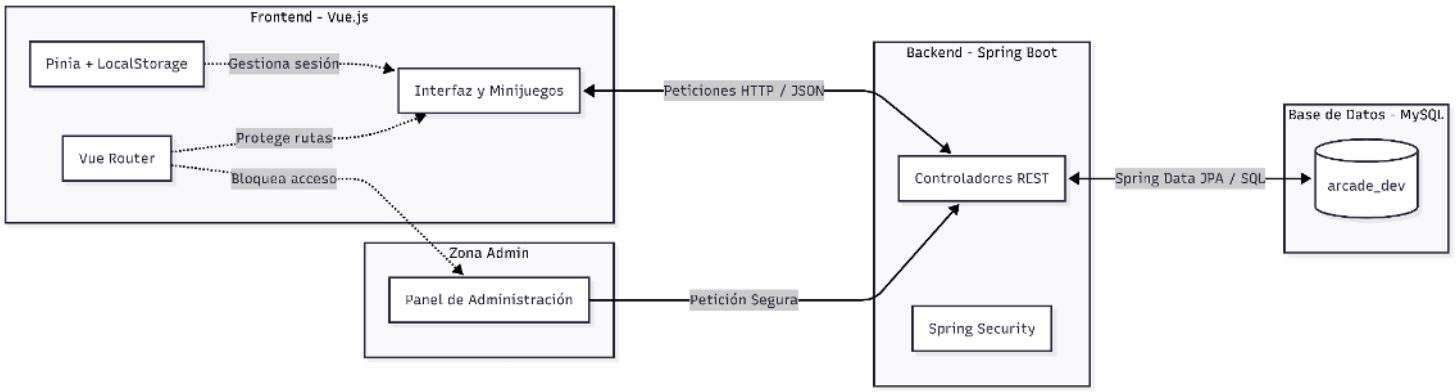
Se encarga de almacenar y suministrar de forma segura y estructurada toda la información vital del sistema.

- ***Funcionamiento:*** *Guarda los datos de los usuarios, el contenido de los retos (preguntas, respuestas, valores…) y registra el historial de las partidas. Además, este componente procesa los datos para generar la pantalla de Clasificación, calculando el ranking global de los jugadores según su saldo de bytes.*
- ***Tecnologías implicadas:*** *Utiliza el motor relacional* ***MySQL****. La comunicación se realiza mediante* ***Spring Data JPA****, lo que permite al backend extraer la información sin necesidad de escribir sentencias SQL manuales difíciles.*

2.4.4 Componente 4: Panel de Administración

Es un módulo de acceso restringido diseñado para el mantenimiento de la plataforma.

- ***Funcionamiento:*** *Proporciona una interfaz gráfica desde la cual los usuarios con rol de admin pueden realizar operaciones de gestión, consultar estadísticas globales de uso y administrar el contenido de las tarjetas y preguntas de los minijuegos.*
- ***Tecnologías implicadas:*** *Utiliza* ***Vue.js*** *(en la carpeta src/admin/) para las vistas de gestión. El acceso a estas pantallas está protegido en el frontend mediante los Navigation Guards de* ***Router*** *de vue definidos en el archivo index.ts, los cuales bloquean la carga del componente si el usuario no tiene el rol de admin especificado en la base de datos.*



2.5 Definición de las funcionalidades

En este apartado se describen de forma conceptual y detallada las funcionalidades principales que ofrece la plataforma.

2.5.1 Funcionalidad 1: Registro y autenticación de usuarios

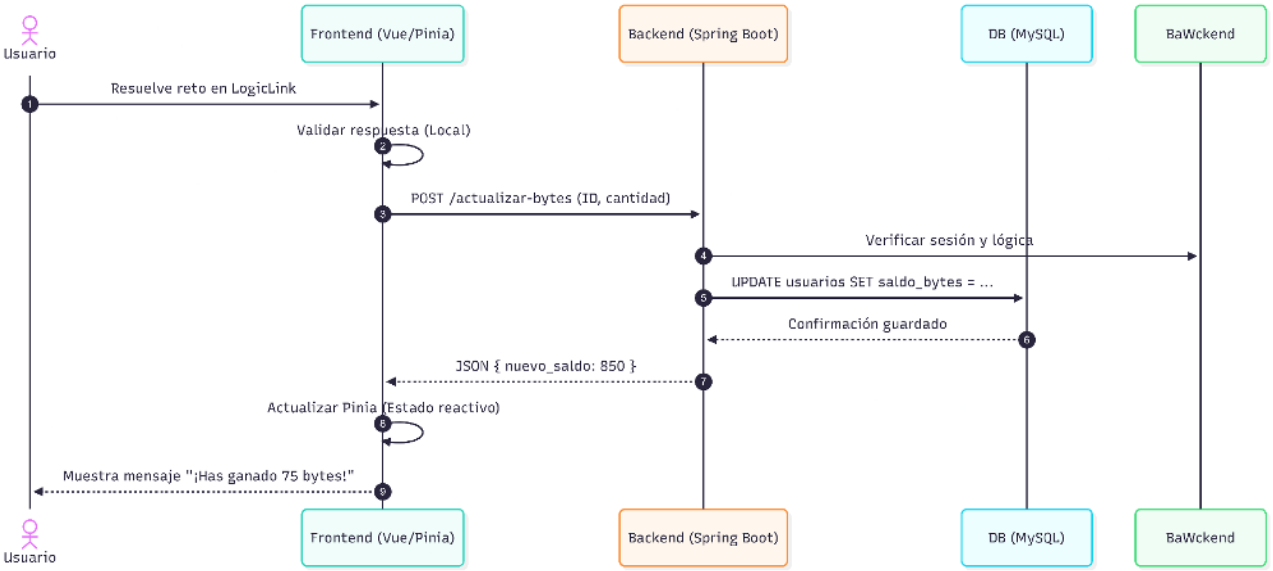
Permite a los nuevos usuarios crear una cuenta personal y acceder a la plataforma de forma segura para que su progreso quede almacenado.

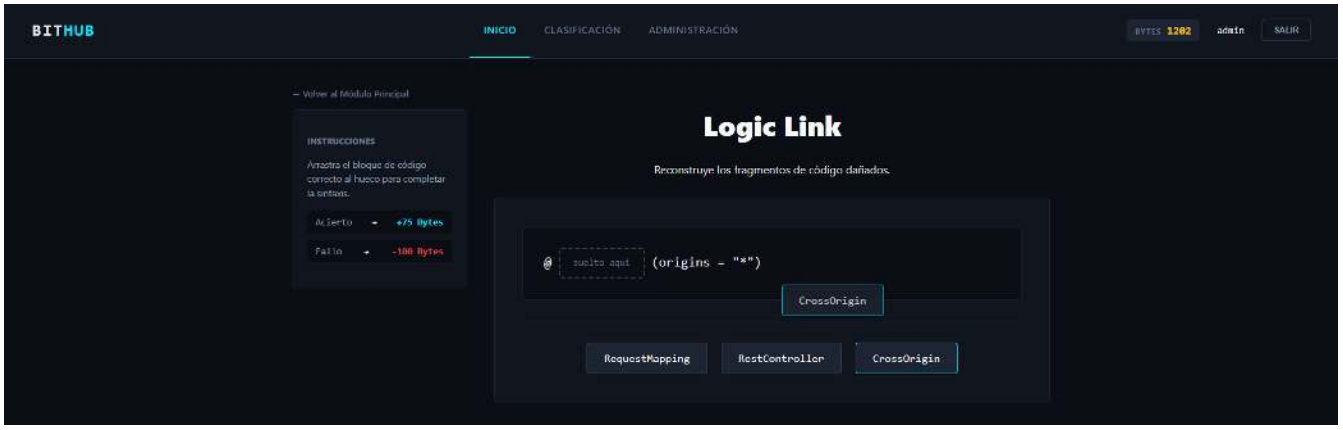
- ***Proceso:*** *El usuario introduce su alias, correo y contraseña en una interfaz que cuenta con validación en el lado del cliente. Al registrarse, los datos se envían mediante una petición POST al backend, donde la contraseña se encripta mediante el algoritmo BCrypt antes de guardarse en MySQL. Al iniciar sesión, el sistema valida las credenciales y, tras el éxito, almacena el estado del jugador en el* ***authStore*** *(Pinia) y lo hace persistente mediante* ***LocalStorage****, permitiendo la navegación por rutas privadas sin cierres de sesión inesperados.*
- ***Estado:*** *Funcionalidad* ***completamente implementada****.*

2.5.2 Funcionalidad 2: Participación en los minijuegos (Arcade)

Es el núcleo del proyecto y permite al usuario interactuar con los distintos retos educativos para poner a prueba sus conocimientos técnicos y ganar recompensas.

- ***Proceso:*** *El frontend solicita al servidor los datos de los retos (extraídos de tablas como logiclink o preguntas_arcade). El usuario interactúa con la interfaz mediante mecánicas de arrastrar y soltar o selección múltiple. El sistema evalúa la respuesta en tiempo real: si es correcta, se muestra el mensaje de* ***"Sintaxis Correcta"****; si es errónea, aparece* ***"Error de Compilación"****. Al finalizar, el sistema calcula la ganancia o pérdida de bytes y actualiza automáticamente el perfil del usuario en la base de datos mediante una llamada a la API.*
- ***Estado:*** *Implementación* ***completa*** *para los juegos actuales, diseñada de forma modular para permitir la integración de nuevos juegos en el futuro.*





2.5.3 Funcionalidad 3: Gestión de saldo y clasificación global

Permite fomentar la competitividad entre los usuarios mediante la visualización de los bytes ganados.

- ***Proceso:*** *El sistema mantiene el saldo de "bytes" actualizado de forma automática en la cabecera de la aplicación. Cuando el usuario accede a la sección de "Clasificación", el frontend realiza una petición al backend, el cual consulta la tabla de usuarios en MySQL y devuelve la lista ordenada de mayor a menor puntuación. La interfaz procesa estos datos para hacer un ranking dinámico que se actualiza tras cada partida.*
- ***Estado:*** *Funcionalidad* ***completamente implementada****.*

2.5.4 Funcionalidad 4: Panel de administración y gestión de contenidos

Proporciona herramientas a los usuarios con rol de administrador para el mantenimiento y actualización de la plataforma desde la propia interfaz web.

- ***Proceso:*** *El acceso a esta funcionalidad está restringido mediante los Navigation Guards de* ***Vue Router*** *definidos en index.ts, que expulsan a cualquier usuario que no tenga el rol de administrador. Una vez dentro, el administrador dispone de una interfaz para realizar operaciones* ***CRUD*** *pero sin la función de actualizar (Crear, Leer y Borrar) sobre el contenido de los juegos,*

permitiendo añadir nuevas preguntas o eliminar retos obsoletos sin necesidad de modificar el código o acceder manualmente a la base de datos.

- ***Estado:*** *Funcionalidad* ***completamente implementada****.*

2.6 Definición de las tareas (Pruebas de validación)

En este apartado se detallan las pruebas realizadas para comprobar el funcionamiento de los componentes y asegurar que la solución cumple con los requisitos técnicos establecidos.

2.6.1 Prueba 1: Sistema de autenticación de usuarios

- **Objetivo:** Comprobar el funcionamiento correcto del sistema de registro y el inicio de sesión.
- **Componentes utilizados:** Backend (Java Spring Boot), Base de datos (MySQL) y componentes reactivos del frontend (Vue.js).
- Proceso:
  - Implementación de una interfaz de registro basada en inputs enlazados a variables reactivas (v-model).
  - Validación en el lado del cliente para comprobar que no faltan credenciales obligatorias antes de interactuar con la API.
  - Envío de los datos (fetch con método POST) y gestión visual de los estados de respuesta (éxito, error de servidor o error de conexión).
  - Implementación de la encriptación de contraseñas mediante BCrypt en el servidor antes del guardado en la base de datos.
  - Prueba de inicio de sesión con credenciales correctas e incorrectas, verificando que el usuario es redirigido correctamente.
- **Conclusión:** El sistema de autenticación funciona correctamente, evita el envío de datos incompletos y garantiza la seguridad de las contraseñas.

2.6.2 Prueba 2: Funcionamiento del motor de minijuegos

- **Objetivo:** Verificar que los minijuegos cargan la información, evalúan las respuestas y funcionan sin errores visuales.
- **Componentes utilizados:** Frontend (Componentes Vue.js), Backend (Controladores) y Base de datos.
- Proceso:
  - Implementación de la interfaz interactiva (sistema de arrastrar, soltar seleccionar…).
  - Conexión con el servidor para extraer las preguntas aleatorias de la base de datos.
  - Ejecución de múltiples partidas de prueba forzando aciertos y errores.
  - Comprobación de los mensajes visuales: **"Sintaxis Correcta"** para los aciertos y **"Error de Compilación"** para los fallos.
- **Conclusión:** Los juegos reaccionan de forma adecuada a las acciones del jugador, ofreciendo una buena experiencia.

2.6.3 Prueba 3: Gestión del saldo virtual (Bytes) y Clasificación

- **Objetivo:** Comprobar que el saldo se actualiza exactamente después de cada partida y que el ranking muestra el orden correcto.
- **Componentes utilizados:** Base de datos, Backend y sistema de estado global (Pinia).
- Proceso:

    - Realización de partidas ganando y perdiendo bytes según la lógica de cada juego.
    - Verificación de que el saldo se actualiza al instante en la cabecera de la web sin necesidad de recargar.
    - Revisión manual en MySQL para confirmar que el nuevo saldo se ha guardado de forma permanente.
    - Consulta de la página de Clasificación para comprobar que los usuarios se ordenan de mayor a menor puntuación.
- **Conclusión:** El sistema gestiona correctamente los puntos virtuales y genera un ranking fiable en tiempo real.

2.6.4 Prueba 4: Panel de administración y seguridad de rutas

- **Objetivo:** Validar que solo los administradores pueden acceder a las herramientas de gestión.
- **Componentes utilizados:** Frontend (Vue Router), Backend y Base de datos.
- Proceso:
    - Intento de acceso directo a la URL de administración con un usuario con rol de jugador, comprobando la expulsión automática.
    - Inicio de sesión con una cuenta de administrador.
    - Navegación por el panel y realización de pruebas creando y borrando preguntas de los minijuegos.
- **Conclusión:** El panel es seguro, los bloqueos en index.ts funcionan y los cambios en la gestión se reflejan inmediatamente en la plataforma.

3 Implementación y Diseño del Sistema

3.1 Catálogo de Minijuegos

3.1.1 Logic Slots

3.1.1.1 ¿De qué trata el juego?

Este juego trata de revelar y emparejar las cartas mediante el emparejamiento de conceptos relacionados (por ejemplo una carta que sea un String, hay que emparejar con otra carta que contenga texto).

A diferencia de el juego tradicional, no basta con encontrar las parejas, al conseguir esto, el sistema lanza al usuario una pregunta técnica sobre las cartas que ha emparejado, el objetivo de esta pregunta es validar que el usuario es capaz de consolidar el acierto. Ha sido pensado para que ayude a los usuarios a reforzar el aprendizaje de conceptos de la programación, haciendo que sea competitivo con un sistema de puntos para motivar a los usuarios.

3.1.1.2 ¿Cómo funciona el juego?

3.1.1.2.1. Concepto y objetivo

Al cargar el componente, el sistema comprueba el saldo de Bytes del usuario. Si el jugador dispone del mínimo requerido (100 Bytes en este caso), se procede a restar el coste mínimo a su saldo. En caso contrario, el acceso es denegado y se redirige al usuario al menú principal. Tras la validación, se ejecutan peticiones (se hace en segundo plano sin detener ni bloquear cualquier ejecución) para obtener desde la base de datos el banco de preguntas y el conjunto de cartas disponibles.

3.1.1.2.2. Preparación del Entorno de Juego

Una vez recibidos los datos de la API, se ordena aleatoriamente tanto a la posición de las cartas en el tablero como a las opciones de respuesta de cada pregunta. Las cartas se cargan con estados iniciales de "oculto" y "no resuelto", mientras que las preguntas quedan almacenadas en memoria, vinculadas a sus respectivas parejas mediante un identificador común (id_pareja).

3.1.1.2.3. Lógica de Emparejamiento y Memoria

El flujo de juego permite la selección de hasta dos cartas simultáneamente. Una vez seleccionadas dos cartas, se comparan sus identificadores de pareja:

- ***Si los identificadores no coinciden:** Se aplica una penalización económica al saldo del usuario y las cartas vuelven a su posición original.*
- ***Si los identificadores coinciden:** Se bloquea el tablero y se activa la siguiente fase de validación (las preguntas).*

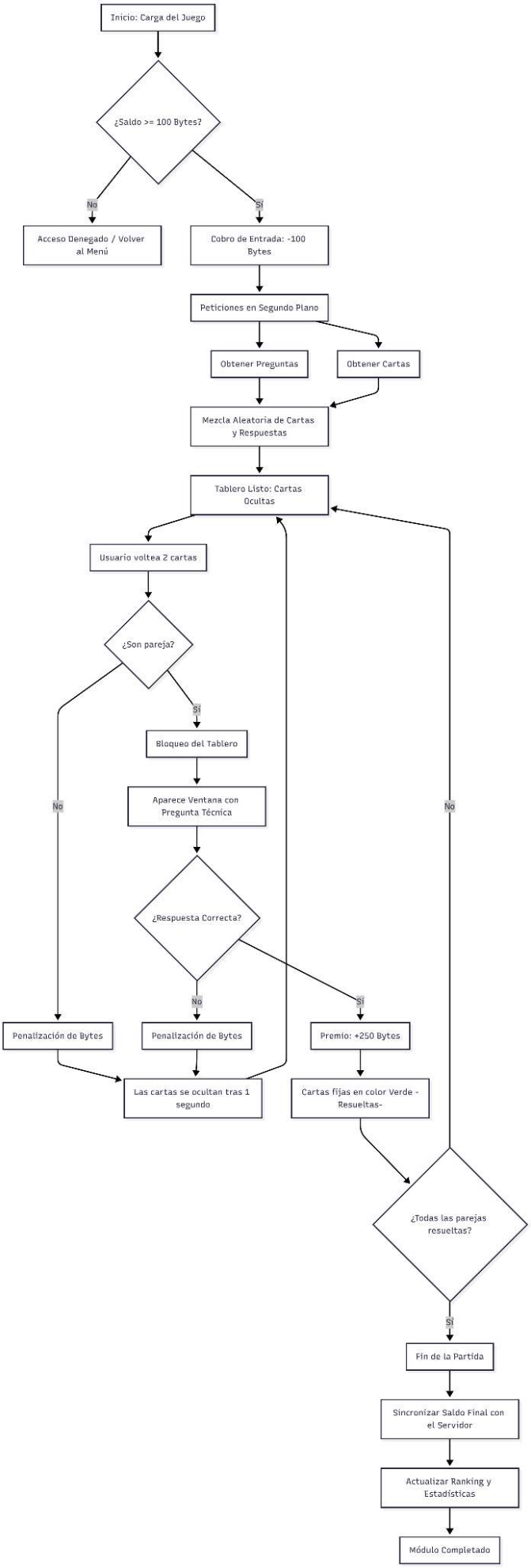
3.1.1.2.4. Fin de partida y reparto de recompensas

Al confirmarse una pareja, se despliega una ventana con una pregunta técnica vinculada directamente con la pareja de cartas encontradas.

- ***Selección y Mezcla:*** *El sistema busca la pregunta que comparte el mismo id_pareja. Tanto las preguntas como las respuestas se hacen aleatorias para que cada partida sea diferente.*
- ***Resolución por Acierto:*** *Si el usuario elige la respuesta correcta, las cartas se marcan como "resueltas", permanecen visibles en verde y el usuario recibe una bonificación de 250 Bytes.*
- ***Resolución por Fallo:*** *Si la respuesta es incorrecta, se aplica una penalización económica. Como castigo adicional, las cartas se giran de nuevo, obligando al usuario a recordar su posición y volver a emparejarlas para intentar responder la pregunta otra vez.*

3.1.1.2.5. Sincronización y Finalización

A lo largo de la partida, cada cambio en el saldo de bytes se actualiza de forma local a la par que se envía el cambio a la base de datos. Una vez que todas las cartas del tablero han sido resueltas, el sistema detecta el fin de la partida y hace una última petición de actualización de saldo al servidor. Esta operación sincroniza el saldo total de Bytes obtenido y registra la actividad en el servicio de estadísticas global, actualizando el perfil del usuario y su posición en la clasificación general.


Inicio: Carga del Juego
¿Saldo >= 100 Bytes?
No
Sí
Acceso Denegado / Volver al Menú
Cobro de Entrada: -100 Bytes
Peticiones en Segundo Plano
Obtener Preguntas
Obtener Cartas
Mezcla Aleatoria de Cartas y Respuestas
Tablero Listo: Cartas Ocultas
Usuario voltea 2 cartas
¿Son pareja?
Sí
No
Bloqueo del Tablero
Aparece Ventana con Pregunta Técnica
¿Respuesta Correcta?
No
Sí
Penalización de Bytes
Penalización de Bytes
Premio: +250 Bytes
Las cartas se ocultan tras 1 segundo
Cartas fijas en color Verde - Resueltas-
No
¿Todas las parejas resueltas?
Sí
Fin de la Partida
Sincronizar Saldo Final con el Servidor
Actualizar Ranking y Estadísticas
Módulo Completado

3.1.2 Code-Jack 21

3.1.2.1 ¿De qué trata el juego?

Este es un juego de estrategia y probabilidad. El objetivo es extraer datos (robar cartas) para acumular la mayor cantidad de memoria posible, representada en GB, acercándose al límite de 21 GB sin sobrepasar.

El jugador compite contra el sistema central (la banca). Si el jugador excede los 21 GB, se produce un "Desbordamiento de memoria" y pierde automáticamente, si el usuario llega a plantarse antes de llegar a la cifra de 21 GB, el sistema central intentará superar la cifra del usuario sin sobrepasar el límite.

Este juego está diseñado para que los usuarios pongan a prueba su capacidad de conversión de unidades de medida. El desafío reside en que no todos los datos se presentan en GB; las unidades pueden variar entre MB, KB, etc…, obligando al jugador a calcular mentalmente la equivalencia para no exceder el límite de memoria permitido.

3.1.2.2 ¿Cómo funciona el juego?

3.1.2.2.1. Inicio y cobro de entrada

Al cargar el módulo, el sistema verifica que el usuario tenga al menos 50 Bytes para poder entrar. Una vez dentro, se realiza una petición al servidor para obtener la "baraja" de datos. Esta baraja contiene objetos con etiquetas de texto y valores numéricos.

3.1.2.2.2. Dinámica de juego (Robar y plantarse)

El juego se basa en un estado que permite dos acciones principales:

- ***Extraer Datos:** El sistema selecciona una carta aleatoria de la baraja y la añade a la mano del jugador.*

  Visualmente, cada carta muestra un valor expresado en distintas unidades de medida (MB, KB, GB, etc.). Este valor sirve para orientar al usuario y permitirle realizar mentalmente la conversión correspondiente durante la partida.

  Sin embargo, internamente cada carta dispone de un valor real almacenado en la base de datos, expresado únicamente en GB. Este valor no es visible para el jugador y es el que utiliza el sistema para calcular la suma total de la mano y comprobar que no supere el límite de 21.

  Si el total acumulado supera los 21 GB, el estado de la partida cambia a "perdido" y se aplica automáticamente una penalización económica al jugador.

- ***Detener Extracción:** El jugador fija su puntuación actual. En este momento, la banca toma el control y extrae cartas automáticamente hasta que su memoria sea igual o mayor a 17 GB.*

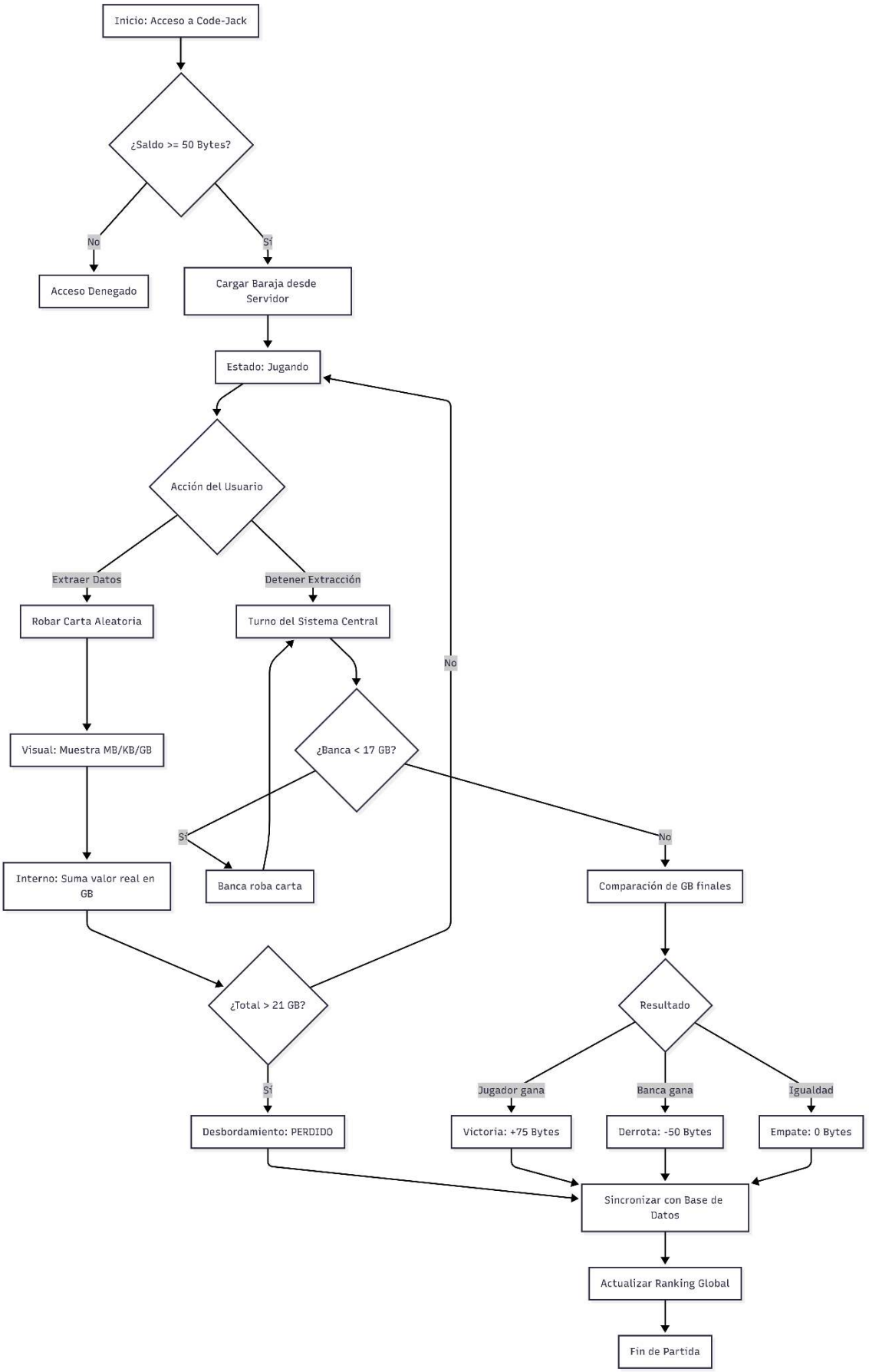
3.1.2.2.3. Resolución y Validación de Recursos

Una vez que ambos han finalizado sus jugadas, el sistema compara los resultados:

- ***Victoria:*** *El jugador gana si su memoria es mayor que la de la banca o si la banca se excede de 21 GB. Recibe una bonificación de 75 Bytes.*
- ***Derrota:*** *El jugador pierde si la banca tiene una puntuación mayor o si el jugador excedió los 21 GB. Se penaliza con 50 Bytes.*
- ***Empate:*** *Si ambos obtienen la misma puntuación, no hay intercambio de Bytes.*

3.1.2.2.4. Sincronización de Datos

Cada resultado se comunica al servidor mediante una petición PUT. Esto asegura que el saldo de Bytes del usuario y sus estadísticas de partida se actualicen en la base de datos, permitiendo que el progreso se refleje en la clasificación.


Inicio: Acceso a Code-Jack
¿Saldo >= 50 Bytes?
No
Sí
Acceso Denegado
Cargar Baraja desde Servidor
Estado: Jugando
Acción del Usuario
Extraer Datos
Detener Extracción
Robar Carta Aleatoria
Turno del Sistema Central
Visual: Muestra MB/KB/GB
¿Banca < 17 GB?
No
Sí
Sí
No
Interno: Suma valor real en GB
Banca roba carta
Comparación de GB finales
¿Total > 21 GB?
Resultado
Sí
Jugador gana
Banca gana
Igualdad
Desbordamiento: PERDIDO
Victoria: +75 Bytes
Derrota: -50 Bytes
Empate: 0 Bytes
Sincronizar con Base de Datos
Actualizar Ranking Global
Fin de Partida

3.1.3 DeciBit

3.1.3.1 ¿De qué trata el juego?

Este juego está diseñado para que el usuario mejore su agilidad mental en la conversión de sistemas numéricos, el objetivo es que el usuario convierta un numero que esta en formato decimal, a su representación en binario, utilizando un panel de 4 bits (8,4, 2, 1).

El juego reta al usuario a que resuelva esta conversión antes de que el temporizador de 10 segundos llegue a 0.

3.1.3.2 ¿Cómo funciona el juego?

3.1.3.2.1. Acceso y Validación de Entrada

Para acceder al juego, el sistema verifica que el usuario cuente con el saldo mínimo de 10 Bytes. Si el saldo es suficiente, se permite la entrada del usuario; de lo contrario el sistema deniega el acceso y redirige al usuario al menú principal.

3.1.3.2.2. Mecánica de Conversión

Una vez iniciada la partida, el funcionamiento se divide en los siguientes pasos lógicos:

- *Generación del Objetivo: El sistema genera un número aleatorio entre 0 y 15.*
- *Interacción con Bits: El usuario dispone de 4 interruptores visuales que representan los valores posicionales del sistema binario: 8, 4, 2 y 1. Al hacer clic en cada "caja de bit", el usuario cambia el valor entre 0 y 1.*
- *Gestión del Tiempo: Se activa un temporizador de 10 segundos. Si el tiempo se agota antes de que el usuario envíe su respuesta, el sistema realiza la comprobación automáticamente, resultando en un error si no se hace la combinación correcta.*

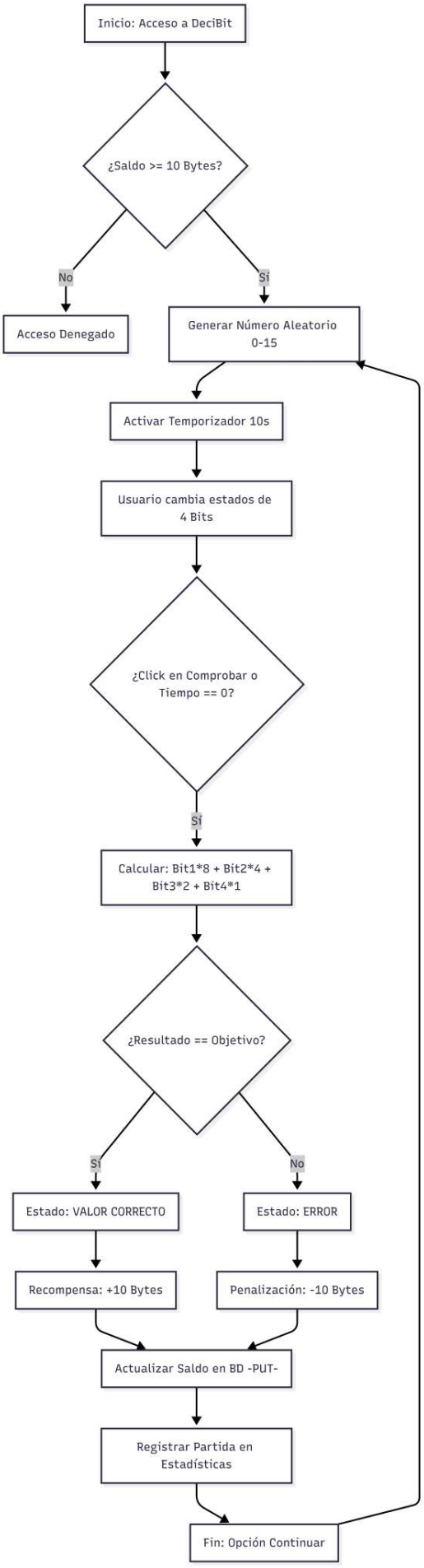
3.1.3.2.3 Validación y Recompensas

Al presionar el botón "Comprobar" o al agotarse el tiempo, el sistema calcula el valor decimal de los bits activados:

- *Acierto: Si la suma de los bits coincide con el número objetivo y queda tiempo en el reloj, el sistema muestra "VALOR CORRECTO" y recompensa al usuario con 10 Bytes.*
- *Fallo: Si la suma no coincide o el tiempo se agota, se muestra el mensaje "ERROR" y se penaliza al usuario restando 10 Bytes de su saldo.*

3.1.3.2.4 Sincronización y Persistencia

Tras cada intento, el sistema realiza una petición de actualización al servidor para modificar el saldo de Bytes en la base de datos. Finalmente, se utiliza el servicio de estadísticas para registrar la partida y actualizando la clasificación global.


Inicio: Acceso a DeciBit
¿Saldo >= 10 Bytes?
No
Sí
Acceso Denegado
Generar Número Aleatorio 0-15
Activar Temporizador 10s
Usuario cambia estados de 4 Bits
¿Click en Comprobar o Tiempo == 0?
Sí
Calcular: Bit1*8 + Bit2*4 + Bit3*2 + Bit4*1
¿Resultado == Objetivo?
Sí
No
Estado: VALOR CORRECTO
Estado: ERROR
Recompensa: +10 Bytes
Penalización: -10 Bytes
Actualizar Saldo en BD -PUT-
Registrar Partida en Estadísticas
Fin: Opción Continuar

3.1.4 CodeLink

3.1.4.1 ¿De qué trata el juego?

Este es un juego de diseñado para que el usuario practique la organización de ideas lógicas y el reconocimiento de fragmentos de código. El objetivo principal es identificar qué pieza es la correcta para completar la línea de código que aparece incompleta en la pantalla.

Para que la experiencia sea cómoda en cualquier dispositivo, el juego ofrece dos formas de interactuar:

- ***Selección táctil**: Ideal para móviles y también pensada para ordenadores, donde se toca la pieza y luego el hueco donde debe encajar.*
- ***Arrastrar y soltar**: Pensada para ordenadores, permitiendo al usuario mover físicamente el bloque de código hasta su destino.*

3.1.4.2 ¿Cómo funciona el juego?

3.1.4.2.1 Inicio y carga del reto

El módulo requiere un coste de entrada de 75 Bytes. El sistema realiza una comprobación de bytes antes de conceder permiso al usuario, si el usuario no alcanza este saldo, se le redirige automáticamente al menú de inicio.

3.1.4.2.2 Dinámica de arrastrar y soltar (Drag & Drop)

Al empezar, el juego solicita automáticamente al servidor la lista de acertijos disponibles. Cada reto que recibe contiene los siguientes elementos:

- ***Código inicial**: Es la primera parte de la frase o instrucción de programación.*
- ***Código final**: Es la parte que cierra la instrucción.*
- ***Respuesta correcta**: Es la pieza faltante que encaja perfectamente y da sentido a las otras dos partes.*
- ***Opciones falsas**: Son respuestas que parecen correctas a simple vista, pero que contienen pequeños errores para poner a prueba tu atención.*

3.1.4.2.3 Mecánica de Resolución (Multiplataforma)

El juego detecta el tipo de dispositivo del usuario para adaptar la experiencia:

- *Escritorio (PC): Utiliza la API nativa de HTML5 de Drag and Drop. El usuario debe arrastrar una de las tres opciones disponibles y soltarla en el "hueco" central.*
- *Móvil/Táctil: Implementa una lógica de selección por pasos. El usuario toca una pieza para marcarla y luego toca el hueco para colocarla.*

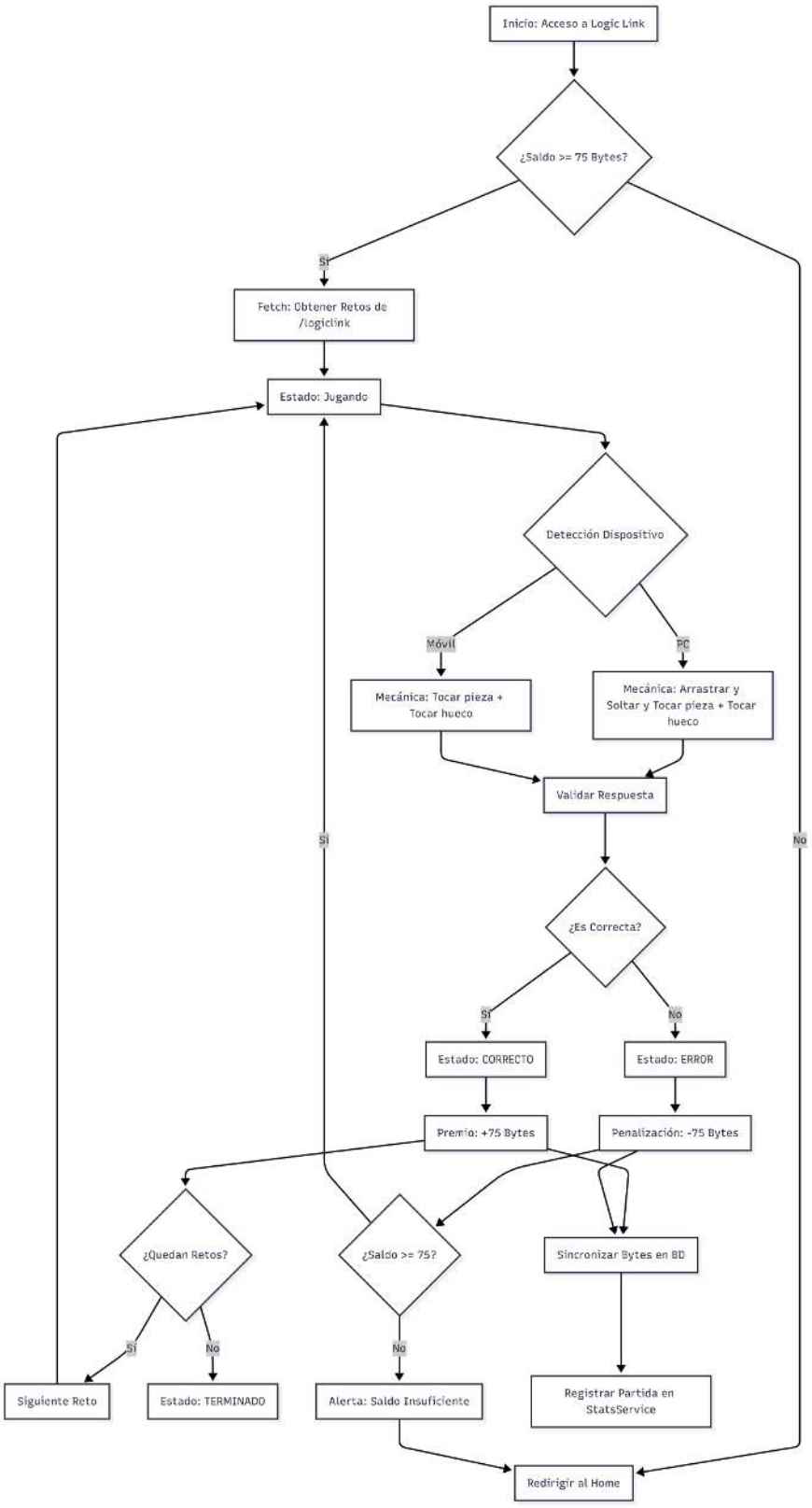
3.1.4.2.4 Validación y Economía del Sistema

Una vez colocada la pieza, el sistema valida la respuesta contra la base de datos:

- *Sintaxis Correcta: El estado cambia a "Correcto", el hueco se ilumina en verde y se otorgan +75 Bytes. Se avanza automáticamente al siguiente reto tras 1 segundo.*
- *Error de Sintaxis: El estado cambia a "Error", el hueco se ilumina en rojo y se restan -75 Bytes. El sistema verifica si el usuario aún tiene saldo para continuar; de lo contrario, finaliza la sesión de juego.*

3.1.4.2.5 Finalización

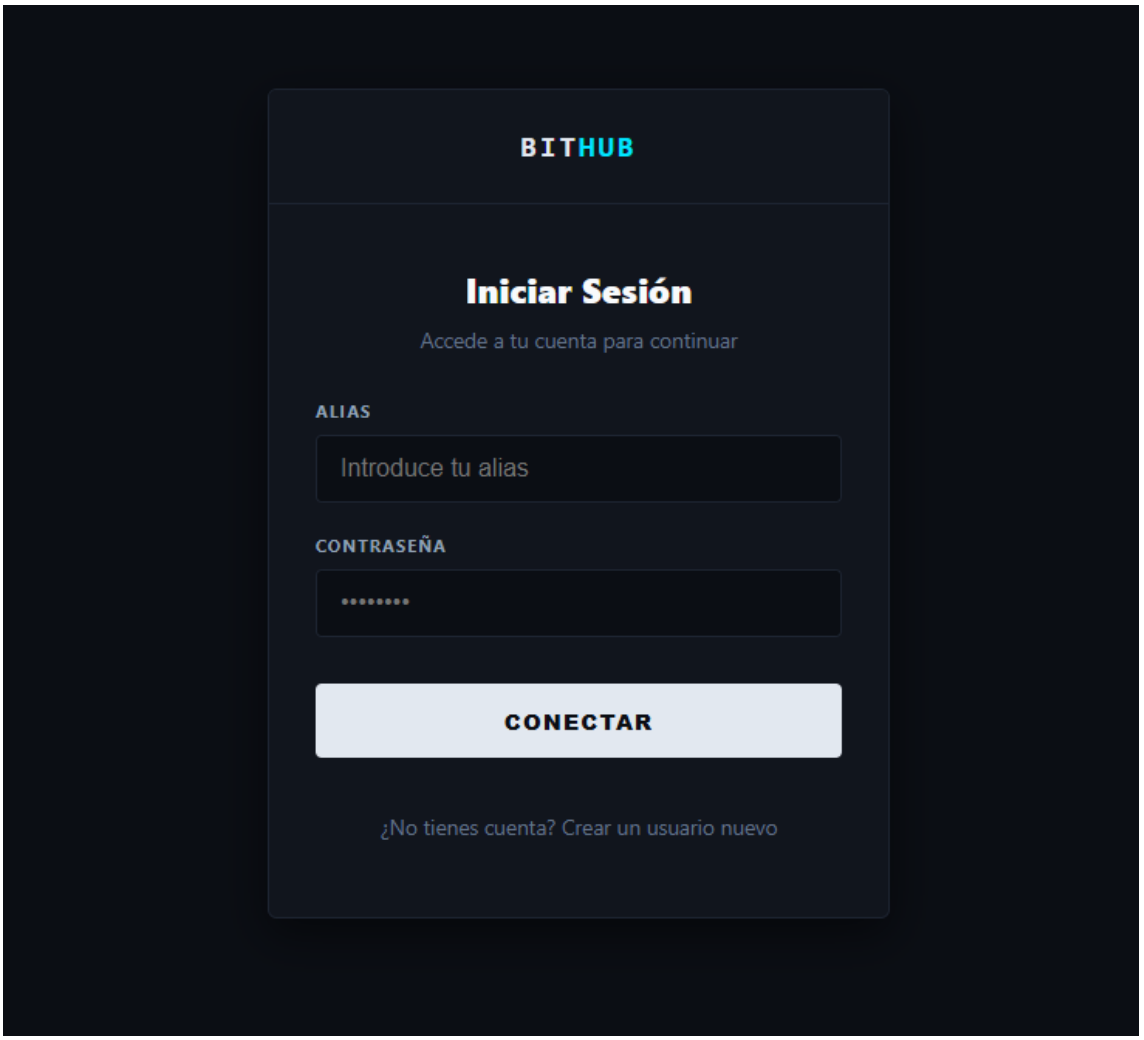
Cuando se completan todos los retos cargados desde el servidor, el sistema muestra un mensaje de "Sistemas Restaurados" y ofrece la posibilidad de reiniciar la base de retos.

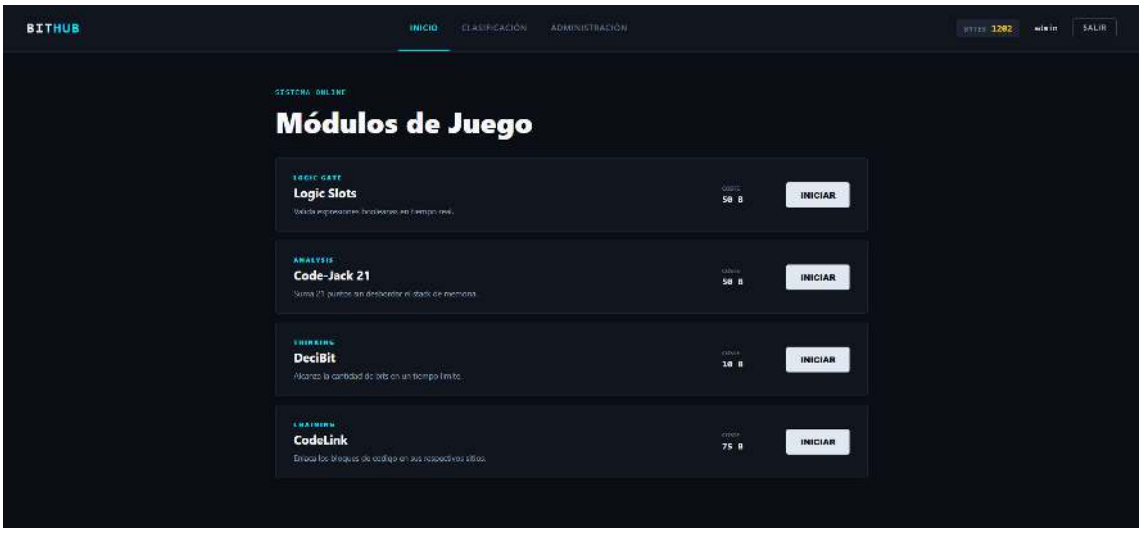

Inicio: Acceso a Logic Link
¿Saldo >= 75 Bytes?
Sí
Fetch: Obtener Retos de /logiclink
Estado: Jugando
Detección Dispositivo
Móvil
PC
Mecánica: Tocar pieza + Tocar hueco
Mecánica: Arrastrar y Soltar y Tocar pieza + Tocar hueco
Validar Respuesta
Sí
No
¿Es Correcta?
Sí
No
Estado: CORRECTO
Estado: ERROR
Premio: +75 Bytes
Penalización: -75 Bytes
¿Quedan Retos?
¿Saldo >= 75?
Sincronizar Bytes en BD
Sí
No
No
Siguiente Reto
Estado: TERMINADO
Alerta: Saldo Insuficiente
Registrar Partida en StatsService
Redirigir al Home

3.2 Diseño de la Interfaz y Experiencia de Usuario

Para el desarrollo visual de la plataforma, se ha priorizado una experiencia de usuario inmersiva que fomente la participación.

- ***Alternativas valoradas:*** *Se debatió entre utilizar un diseño corporativo y minimalista estándar, o un diseño temático.*
- ***Decisión tomada y justificación:*** *Se ha optado por una estética **"Retro-Arcade" (Modo Oscuro)**. Se han utilizado fondos oscuros, tipografías monoespaciadas para el texto de los juegos y colores de contraste (como el cian para los aciertos y el rojo para los errores). Esta decisión técnica se tomó porque reduce la fatiga visual del usuario y encaja perfectamente con la temática de "minijuegos", mejorando la retención y el interés del jugador.*

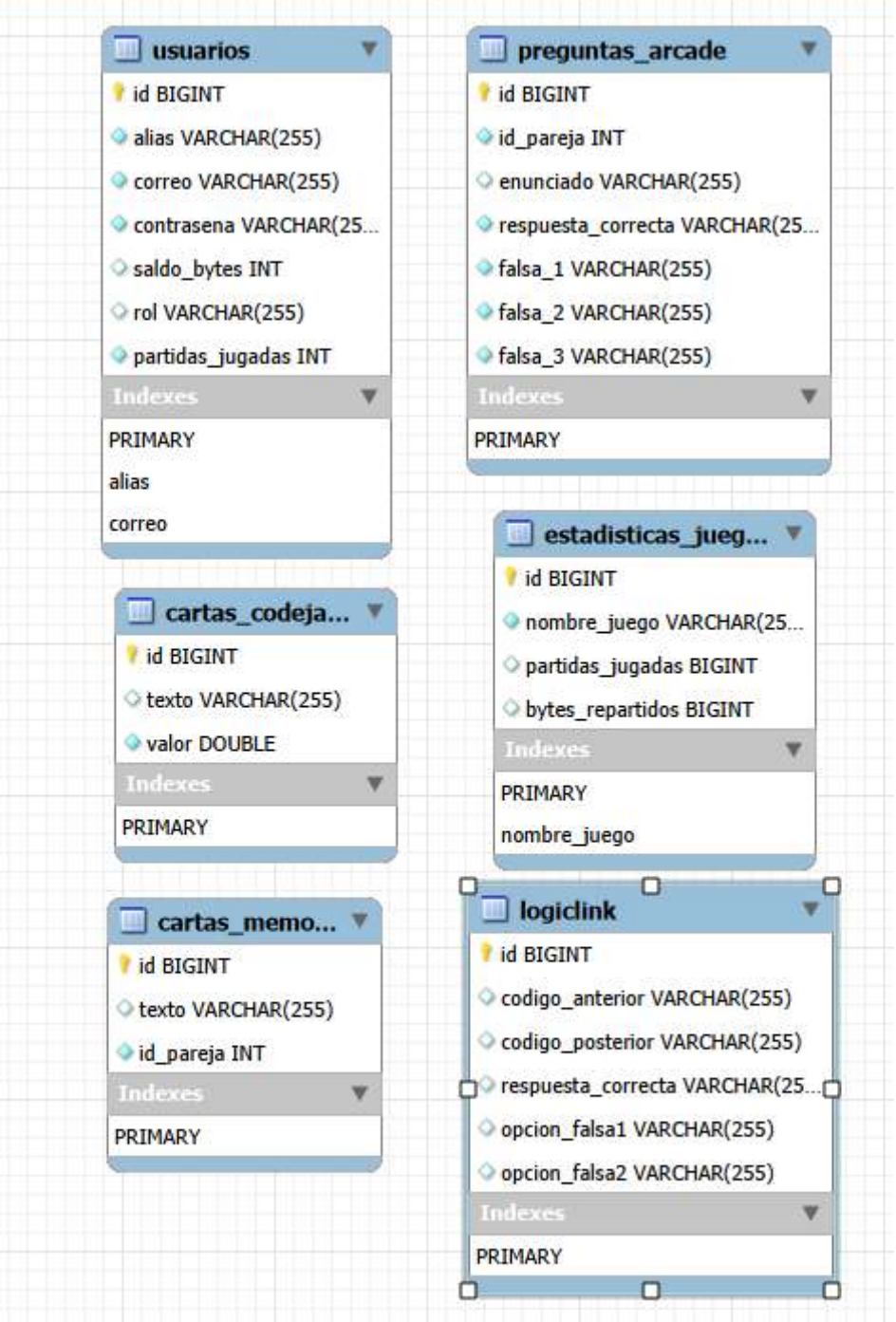




3.3 Modelo de Datos

Para garantizar la organización y escalabilidad de la información almacenada en MySQL, se ha diseñado una estructura de base de datos relacional (arcade_dev) dividida en entidades específicas para cada funcionalidad.

- ***Descripción de la estructura:*** *El núcleo del sistema es la tabla usuarios, que almacena las credenciales de acceso, el rol y el progreso global del jugador (saldo_bytes y partidas_jugadas). Para las estadísticas de la página, la tabla estadisticas_juegos registra la popularidad y el impacto económico de cada minijuego (contabilizando partidas jugadas y bytes repartidos). Finalmente,los juegos están creados en tablas independientes según su mecánica, logiclink (para los fragmentos de código a completar), preguntas_arcade (para los retos tipo test con opciones múltiples), y las tablas de datos para los juegos de cartas (cartas_codejack y cartas_memory).*



3.4 Despliegue del Hosting

Para que la página sea accesible a través de internet, se ha llevado a cabo el despliegue del proyecto. La arquitectura del despliegue se ha dividido en dos plataformas gratuitas, garantizando conexiones seguras (https): backend y la base de datos se aloja en **Railway**, mientras que el frontend se despliega mediante **Vercel**.

3.4.1 Despliegue del Backend y Base de Datos (Railway)

Railway es una plataforma de alojamiento en la nube que facilita la subida de aplicaciones backend sin configuraciones complejas de servidores. Los pasos realizados para su puesta en marcha son los siguientes:

1. **Creación del proyecto:** Se accede a la plataforma Railway y se vincula la cuenta de GitHub del administrador. Se crea un nuevo proyecto en blanco.
2. **Provisión de la Base de Datos:** Dentro del entorno del proyecto, se añade un nuevo servicio seleccionando el motor de base de datos relacional que tengas. La plataforma aprovisiona el servicio y genera automáticamente las credenciales de acceso.
3. **Despliegue de la API (Java Spring Boot):** Se incorpora un segundo servicio al proyecto, vinculado directamente al repositorio de GitHub que contiene el código del Backend.
4. **Configuración de Variables de Entorno:** En los ajustes del servicio del Backend, se introducen las variables del sistema (las credenciales, URL y puerto generados en el paso 2) para que la API pueda comunicarse internamente con la Base de Datos.
5. **Generación del dominio público:** Finalmente, en la pestaña de red (Networking), se genera un dominio público. Esta URL será la vía de comunicación a la que el Frontend enviará las peticiones HTTP.

3.4.2 Despliegue del Frontend (Vercel)

Vercel es una plataforma optimizada para hospedar aplicaciones y frameworks de frontend de manera rápida e integrada. El proceso de despliegue consta de los siguientes pasos:

1. **Conexión del repositorio:** Se ingresa a la plataforma Vercel iniciando sesión con GitHub. Se selecciona la opción de "Añadir Nuevo Proyecto" y se importa el repositorio que contiene el código fuente de Vue.js.
2. **Detección del marco de trabajo:** La plataforma analiza el código y detecta automáticamente que se trata de un proyecto desarrollado con Vue y el empaquetador Vite, auto configurando los comandos de compilación necesarios.
3. **Enlace con el Backend:** En la sección de Variables de Entorno, se añade una nueva variable denominada VITE_API_BASE_URL. Como valor, se introduce de forma estricta la URL pública generada por Railway en el apartado anterior.
4. **Despliegue y publicación:** Se procede a la compilación (*Deploy*). Tras finalizar el proceso, Vercel proporciona una URL pública, cifrada y lista para ser utilizada por los usuarios finales.

3.4.3 Integración y Despliegue

Gracias a la infraestructura descrita, el proyecto cuenta con un flujo de **Integración y Despliegue**. Esto significa que el proceso de actualización está automatizado: cada vez que se realiza una modificación en el código y se sube al repositorio principal de GitHub, tanto Vercel como Railway detectan los cambios de forma automática. Automáticamente, descargan la nueva versión, la compilan y actualizan la aplicación, sin necesidad de realizar ninguna intervención manual en los servidores ni interrumpir el servicio a los usuarios.

3.5 Responsive

¿Cómo lo hemos aplicado?

- *Adaptación de la Interfaz: En una pantalla de ordenador, los menús y paneles se muestran extendidos (en horizontal). En un móvil, esos mismos elementos se apilan uno debajo del otro (en vertical) para que no tengas que hacer "zoom" para leer.*

- *Contenedores Flexibles: Utilizamos estructuras que se estiran o encogen como si fueran de goma. Si la pantalla es pequeña, las tarjetas de los juegos (como Logic Slots o DeciBit) se ajustan para ocupar todo el ancho disponible sin salirse de los bordes.*

- *Mecánicas Duales: No solo cambia lo que se ve, sino cómo se juega. Por ejemplo, en Logic Link, el sistema detecta si estás en PC para dejarte "arrastrar" piezas, o si estás en móvil para permitirte "tocar y colocar".*

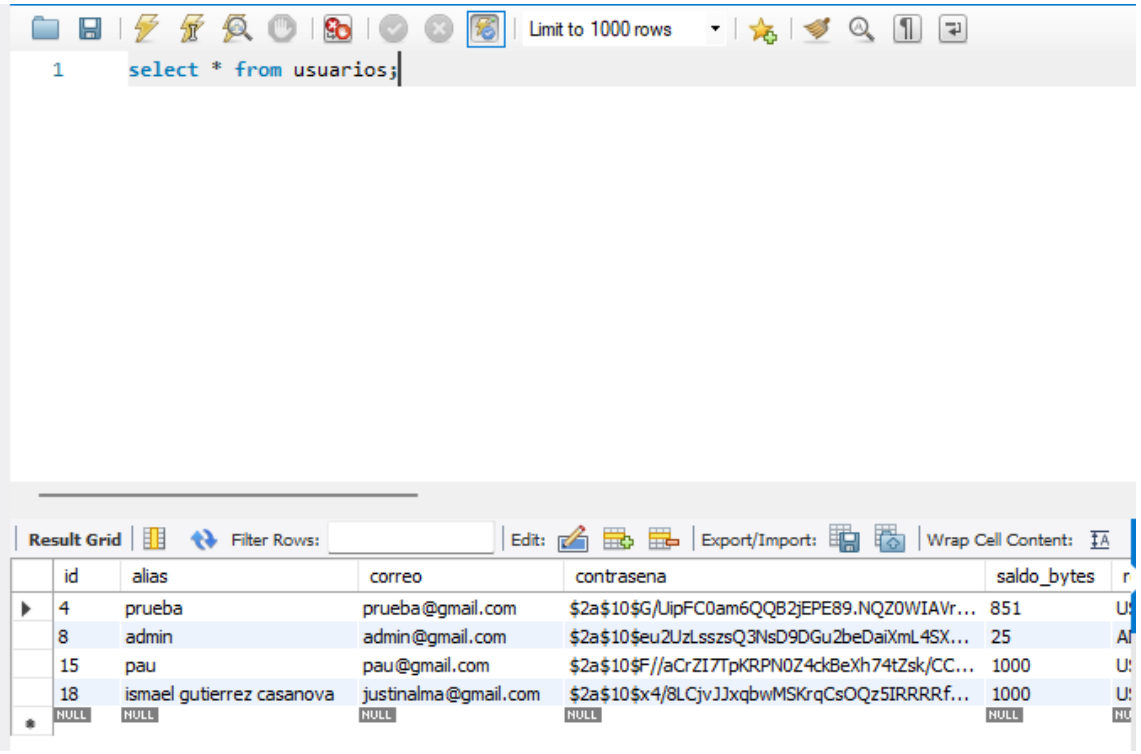
- *Optimización de Menús: La cabecera, que en PC muestra todos los enlaces (Inicio, Clasificación, Perfil), en pantallas pequeñas se simplifica o se reorganiza para no saturar el espacio visual del jugador.*

3.6 Seguridad

3.6.1 Cifrado de Credenciales y Seguridad en el Servidor

En el entorno del Backend, la seguridad de los accesos está gestionada a través de la configuración de *Spring Security* (SecurityConfig). Con el fin de prevenir de proteger las contraseñas mediante un cifrado de credenciales.

Las contraseñas de los usuarios en ningún caso se almacenan en texto plano. En su lugar, el sistema hace uso del algoritmo de encriptación BCrypt, el cual genera un *hash* criptográfico antes de guardar la información en la base de datos. De este modo, la verificación de credenciales durante el inicio de sesión se realiza comparando hashes, garantizando que ni siquiera los administradores del sistema tengan acceso a las contraseñas originales.

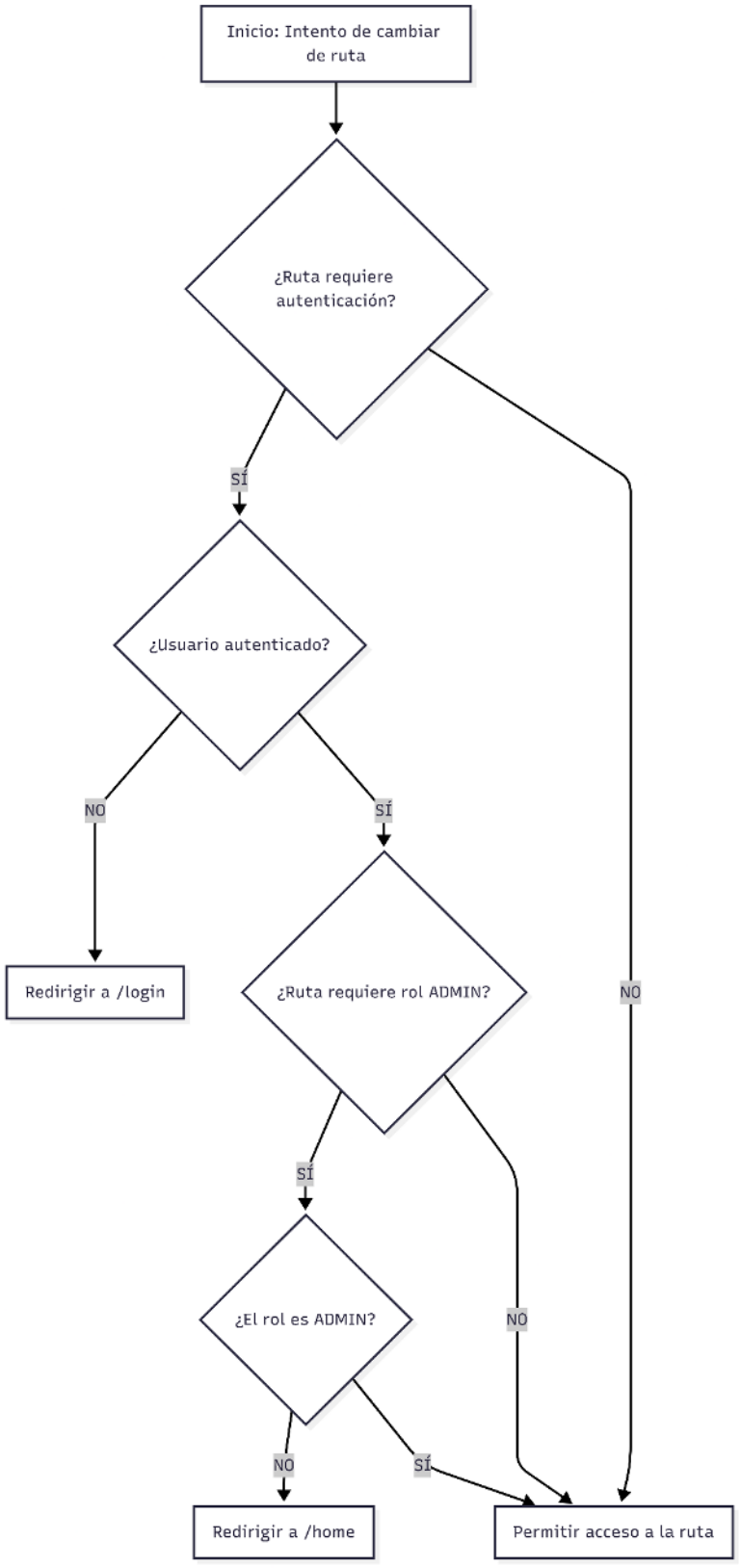


3.6.2 Control de Acceso y Protección de Rutas en el Cliente

Para la capa de presentación (Frontend), se ha diseñado un sistema de protección de navegación utilizando *Vue Router* en conjunto con el gestor de estados de autenticación global (Store).

El control de acceso se hace mediante la asignación de metadatos a cada ruta de la aplicación, definiendo dos niveles de restricción: requiereAutenticacion (para usuarios registrados) y requiereAdmin (para administradores). La evaluación de estos metadatos se ejecuta mediante un guardián de navegación global (router.beforeEach), el cual intercepta y analiza cada petición de cambio de vista antes de renderizarla.

El comportamiento lógico de este interceptor se describe en el siguiente flujo de decisiones:



Desglose de las reglas de intercepción:

- **Bloqueo a invitados:** Si se detecta un intento de acceso a una ruta protegida y el sistema no registra un rol de usuario activo en el estado de autenticación,

la petición es denegada y se redirige automáticamente a la pantalla de inicio de sesión (/login).

- **Protección de rutas de administración:** Si un usuario autenticado con un rol estándar intenta acceder a los paneles de gestión de contenido o estadísticas (/admin, /preguntas, /cartas, etc.), el guardián verifica la ausencia del rol ADMIN, deniega el acceso y lo redirige de forma segura a la interfaz principal (/home).
- **Prevención de enlaces rotos:** Se ha configurado una ruta comodín de captura global (/:pathMatch(.*)*). Si un usuario manipula la URL del navegador e introduce una ruta inexistente, el sistema intercepta el error y fuerza una redirección al punto de entrada seguro (/login), evitando la exposición de interfaces vacías o errores técnicos.

4 Conclusiones

4.1 Conclusiones generales del proyecto

*Tras haber finalizado el desarrollo de **BitHub**, la conclusión principal a la que se llega es que crear una plataforma de este tipo es bastante más complicado de lo que se planteó al principio. Se ha aprendido mucho sobre cómo conectar la parte visual con el servidor, ya que se ha visto que si la base de datos no se comunica bien con la pantalla la aplicación no sirve para nada.*

El proyecto ha servido para entender cómo funciona una aplicación real, desde que se piensa la idea hasta que se ve funcionando en el navegador. Además, se ha comprobado que ver los juegos terminados y que los puntos suban de verdad motiva mucho a seguir programando. Al final, lo que se buscaba era que aprender a programar no fuera un aburrimiento y se considera que con los juegos diseñados se ha conseguido bastante bien ese objetivo.

4.2 Consecución de los objetivos

Si se repasa lo que se propuso hacer al inicio, se puede decir que se han logrado prácticamente todas las metas:

- ***Desarrollo de la web completa:** Logrado. La web funciona, se puede navegar por ella y no se han detectado errores críticos.*
- ***Funcionamiento de los minijuegos:** Logrado. El juego de Logic Link y los demás cargan bien, avisan si hay fallos y reparten los puntos correctamente.*
- ***Seguridad y cuentas de usuario:** Logrado. Se pueden crear cuentas, entrar con contraseña y se ha conseguido que nadie que no sea administrador entre a la parte de gestión.*
- ***Puntos y ranking en directo:** Logrado. Los bytes suben y bajan al momento y se ha conseguido que la lista de clasificación se actualice sola.*

4.3 Valoración de la metodología y planificación

Sobre la organización, se ha visto que al principio se seguía muy bien el calendario, pero luego con la parte del frontend surgieron más líos de los que se esperaban. Al final, el calendario que se hizo se quedó un poco corto porque algunas cosas de los puntos y la lógica de los juegos costaron más tiempo de la cuenta.

Como análisis, se admite que la base de datos se podría haber hecho con más relaciones, pero se decidió hacerlo así para ir más rápido y controlar los posibles errores desde el código de Java. Al final, gracias a usar GitHub y a ir guardando el código poco a poco, no se ha perdido información y se ha podido sacar el proyecto adelante a pesar de los baches que han ido saliendo.

4.4 Visión de futuro

Aunque la web ya funciona correctamente, se han visto muchas cosas que se podrían añadir si se dispusiera de más tiempo:

- ***Modo multijugador:** Estaría muy bien que se pudiera jugar contra un amigo a la vez para ver quién acaba antes los retos.*
- ***Niveles y medallas:** Que según los puntos que se tengan, se den medallas o se suba de nivel para que los usuarios quieran jugar más.*
- ***Más variedad de juegos:** Añadir juegos de otros lenguajes o cosas de ciberseguridad, para no quedarse solo en la lógica de código.*
- ***Sistema de chat:** Que los usuarios pudieran hablar entre ellos o darse pistas sobre cómo pasar los niveles más difíciles.*

5. Glosario

API REST: *Se trata de la interfaz que se utiliza para que el navegador y el servidor se puedan comunicar entre ellos enviando y recibiendo información.*

Backend: *Se refiere a toda la parte lógica y de servidor de la aplicación. Se ha encargado de gestionar los datos y la seguridad desde atrás para que todo funcione.*

BCrypt: *Se trata de un sistema de cifrado que se utiliza para que las contraseñas de los usuarios no se guarden tal cual, sino que se conviertan en un código seguro.*

CRUD (Create, Read, Update, Delete): *Son las siglas que se utilizan para referirse a las operaciones básicas de Crear, Leer, Actualizar y Borrar datos en cualquier aplicación.*

Frontend: *Se trata de la parte visual de la aplicación, es decir, todo lo que el usuario ve en la pantalla y donde se interactúa con los juegos.*

HTTP: *Es el protocolo que se utiliza para que la información pueda viajar por internet entre el cliente y el servidor de forma correcta.*

JPA (Java Persistence API): *Se trata de una tecnología que se utiliza en el servidor para trabajar con la base de datos de forma más sencilla, sin tener que escribir código SQL manualmente todo el tiempo.*

JSON: *Es el formato que se utiliza para enviar los datos (como los puntos o el nombre de usuario) entre el servidor y el navegador de manera ligera.*

LocalStorage: *Se trata de un pequeño espacio de memoria en el navegador que se utiliza para guardar datos de forma que la sesión no se cierre al recargar la página.*

MySQL: *Es el sistema que se ha elegido para montar la base de datos y guardar toda la información de los jugadores y las preguntas de los juegos.*

Pinia: *Se trata del sistema que se utiliza para gestionar los datos de la sesión en el frontend y que la web sepa en todo momento qué usuario está jugando.*

SPA (Single Page Application): *Se refiere a las aplicaciones de una sola página que cargan una vez y luego van cambiando el contenido sin que la web tenga que recargarse entera cada vez.*

Spring Boot: *Es la herramienta de Java que se ha usado para crear todo el sistema del servidor y que la aplicación sea estable y segura.*

Vue.js: *Se trata del sistema que se ha utilizado para crear toda la parte visual y conseguir que los juegos sean interactivos y rápidos.*

6. Bibliografia

Documentación técnica oficial:

*[1] **Vue.js.** The Progressive JavaScript Framework. Disponible en: https://vuejs.org/ [Visitado: mayo de 2026].*

*[2] **Spring Boot.** Build anything with Spring Boot. Disponible en: https://spring.io/projects/spring-boot [Visitado: mayo de 2026].*

*[3] **Pinia.** The intuitive store for Vue.js. Disponible en: https://pinia.vuejs.org/ [Visitado: mayo de 2026].*

*[4] **MySQL.** MySQL Documentation. Disponible en: https://dev.mysql.com/doc/ [Visitado: mayo de 2026].*

*[5] **MDN Web Docs.** Resources for Developers (HTML, CSS, JavaScript). Disponible en: https://developer.mozilla.org/ [Visitado: mayo de 2026].*

Guías y recursos de aprendizaje:

*[6] **Baeldung.** Spring Boot Tutorials. Disponible en: https://www.baeldung.com/spring-boot [Visitado: mayo de 2026].*

*[7] **Stack Overflow.** Foro de consulta técnica para desarrolladores. Disponible en: https://stackoverflow.com/ [Visitado: mayo de 2026].*

*[8] **W3Schools.** Web Development Tutorials. Disponible en: https://www.w3schools.com/ [Visitado: mayo de 2026].*

Herramientas de diseño y diagramación:

*[9] **Mermaid Live Editor.** Generación de diagramas mediante código. Disponible en: https://mermaid.live/ [Visitado: mayo de 2026].*

*[10] **Canva.** Herramienta de diseño gráfico para interfaces y logotipos. Disponible en: https://www.canva.com/ [Visitado: mayo de 2026].*

7 Anexos

Dado el carácter autocontenido de la presente memoria, la mayor parte de la información técnica, arquitectónica y de uso ha sido detallada a lo largo de los capítulos principales. No obstante, se adjuntan a continuación los recursos externos del proyecto para su revisión, evaluación y prueba.

7.1. Repositorios de Código Fuente *El código fuente íntegro de la aplicación se encuentra alojado en la plataforma GitHub, dividido en dos repositorios independientes para mantener la separación de responsabilidades entre la interfaz y la lógica del servidor:*

- ***Frontend (Vue.js) y Backend (Java Spring Boot):*** *https://github.com/AdrianLuqueDiaz/Proyecto-DAW/tree/juego-1-con-preguntas*

7.2. Despliegue en Producción *El entorno de producción se encuentra accesible de forma pública a través de internet. Se puede acceder a la plataforma final y probar todas las funcionalidades descritas en este documento a través del siguiente enlace:*

- ***Plataforma web (BitHub):*** *https://bithub-arcade.vercel.app/*

7.3. Test de usabilidad *Para ofrecer un enfoque más exhaustivo sobre la usabilidad del proyecto, hemos elaborado el siguiente informe:*

https://docs.google.com/spreadsheets/d/1Ba2CY5XzO-9rSinadsX53VfG5WItLohES8w43kt_xBY/edit?usp=sharing