Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias.
Ingeniería de Sistemas y Computación.
Proyecto de Grado.

Diseño y estructuración de una plataforma web educativa que brinde una experiencia de aprendizaje atractiva e inmersiva para los estudiantes de un curso universitario de empleabilidad.

Santiago Grisales Zamora

Director: Dr. Gerardo Mauricio Sarria Montemiranda

29-08-2025

Santiago de Cali, 29-08-2025.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria Montemiranda
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo.

Por medio de la presente me permito informarle que el estudiante de Ingeniería de Sistemas y Computación Santiago Grisales Zamora (cod: 8956256) trabaja bajo mi dirección en el proyecto de grado titulado "Diseño y estructuración de una plataforma web educativa que brinde una experiencia de aprendizaje atractiva e inmersiva para los estudiantes de un curso universitario de empleabilidad.".

Atentamente,

Digitally signed by
Gerardo M. Sarria M.
Date: 2026.01.31
12:15:56 -05'00'

Dr. Gerardo Mauricio Sarria Montemiranda

Santiago de Cali, 29-08-2025.

Señores
Pontificia Universidad Javeriana Cali.
Dr. Gerardo Mauricio Sarria Montemiranda
Director Carrera de Ingeniería de Sistemas y Computación.
Cali.

Cordial Saludo.

Me permito presentar a su consideración el proyecto de grado titulado "Diseño y estructuración de una plataforma web educativa que brinde una experiencia de aprendizaje atractiva e inmersiva para los estudiantes de un curso universitario de empleabilidad." con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el proyecto de grado y posteriormente optar al título de Ingeniero de Sistemas y Computación.

Al firmar aquí, doy fe que entiendo y conozco las directrices para la presentación de trabajos de grado de la Facultad de Ingeniería aprobadas el 26 de Noviembre de 2009, donde se establecen los plazos y normas para el desarrollo del anteproyecto y del trabajo de grado.

Atentamente,

Santiago Grisales Zamora

Firmado digitalmente por Santiago Grisales Zamora
Fecha: 2026.01.26 08:12:48 -05'00'

Santiago Grisales Zamora
Código: 8956256

Agradecimientos

Agradezco a la Pontificia Universidad Javeriana de Cali por el apoyo académico y los recursos brindados para el desarrollo de este trabajo de grado.

Expreso un especial agradecimiento a mi director(a) de trabajo de grado por su orientación, acompañamiento y aportes durante todo el proceso de investigación y desarrollo del proyecto.

Asimismo, agradezco a la Oficina Institucional de Prácticas Profesionales por haberme permitido participar en la realización de mi práctica profesionales y trabajo de grado en un problema real que tenían.

Finalmente, agradezco a mi familia por su apoyo constante, compresión y motivación durante toda mi carrera académica. . De manera especial, agradezco al Ingeniero Carlos Felipe Palacio Lozano y a mis amigos del grupo Toxics por su apoyo, colaboración y motivación durante este proceso.

Resumen

Actualmente, el curso Ruta de Empleabilidad, implementado por la Oficina Institucional de Prácticas a través del LMS Brightspace, tiene como objetivo fortalecer las habilidades blandas y preparar a los estudiantes para su inserción laboral. No obstante, el curso presenta diversas limitaciones, entre ellas: la falta de interactividad y gamificación, la sobreabundancia de preguntas, la dificultad para realizar un seguimiento y evaluación eficiente, así como la limitada capacidad para almacenar y visualizar el progreso de los estudiantes.

Ante esta situación, se llevó a cabo un proceso de evaluación e investigación sobre el uso de LMS, paquetes SCORM y APIs de integración que pudieran ofrecer una solución para mejorar la dinámica del curso. Sin embargo, debido a la rigidez, escaso soporte técnico, limitaciones de diseño y dificultades técnicas del LMS Brightspace, se decidió, en conjunto con la oficina institucional de prácticas y con el apoyo del Centro de Sistemas de Información (CSI) de la universidad, desarrollar una aplicación web interactiva y gamificada. Esta aplicación permitirá a los estudiantes realizar las actividades de manera dinámica e interactiva, incorporando elementos gamificados como mapas, recompensas, progreso, puntajes y rankings. De esta forma, se busca que la experiencia de aprendizaje sea más inmersiva y motivadora, logrando que los estudiantes disfruten del proceso mientras desarrollan habilidades clave para su futura vida profesional. Adicionalmente, para el equipo administrativo de la oficina, que anteriormente debía dedicar gran parte de su tiempo a evaluar manualmente el trabajo de aproximadamente 600 estudiantes por semestre, la implementación de esta herramienta representará una mejora, pues la aplicación automatizará la evaluación y recolección de datos, permitiendo que el equipo se enfoque principalmente en el seguimiento del progreso y la generación de informes.

A futuro, esta aplicación web tendrá el potencial de consolidarse como un ecosistema educativo gamificado, proyectando la incorporación de nuevas actividades dinámicas que enriquezcan la experiencia formativa de los estudiantes. Asimismo, busca optimizar el manejo administrativo, facilitando un monitoreo detallado y eficiente del desempeño estudiantil. La proyección a mediano y largo plazo es que esta herramienta pueda estar disponible para toda la universidad, permitiendo que los estudiantes accedan a ella desde semestres tempranos y comiencen a prepararse con anticipación para los retos del mundo laboral.

Abstract

Currently, the Employability Pathway course, implemented by the Institutional Internship Office through the Brightspace LMS, aims to strengthen soft skills and prepare students for entering the workforce. However, the course has several limitations, including a lack of interactivity and gamification, an excessive number of questions, difficulties in conducting efficient monitoring and evaluation, and a limited capacity to store and visualize student progress.

Given this situation, an evaluation and research process was conducted on the use of Learning Management Systems (LMS), SCORM packages, and integration APIs that could offer a solution to improve the course dynamics. However, due to the rigidity, limited technical support, design limitations, and technical difficulties of the Brightspace LMS, it was decided, in conjunction with the institutional internship office and with the support of the university's Information Systems Center (CSI), to develop an interactive and gamified web application. This application will allow students to complete activities dynamically and interactively, incorporating gamified elements such as maps, rewards, progress tracking, scores, and rankings. This approach aims to make the learning experience more immersive and motivating, ensuring that students enjoy the process while developing key skills for their future professional lives. Additionally, for the office's administrative team, which previously had to dedicate a significant amount of time to manually evaluating the work of approximately 600 students per semester, the implementation of this tool will represent an improvement. The application will automate the evaluation and data collection, allowing the team to focus primarily on monitoring progress and generating reports.

Looking ahead, this web application has the potential to become a gamified educational ecosystem that incorporates new dynamic activities to enrich the student learning experience. It also aims to optimize administrative management, facilitating detailed and efficient monitoring of student performance. The medium- and long-term vision is for this tool to be available to the entire university, allowing students to access it from early semesters and begin preparing for the challenges of the professional world.

Índice general

Introducción

La preparación de los estudiantes para enfrentar los desafíos y situaciones del ambiente laboral es una prioridad de las universidades; por esta razón, la Oficina Institucional de Prácticas Profesionales ofrece como prerrequisito para matricular la materia Práctica Profesional, un curso llamado 'Equipamiento para el trabajo'. El curso se dicta actualmente de forma presencial en la universidad durante una semana con horarios fijos, lo que dificulta la asistencia de algunos estudiantes. Como es un prerrequisito obligatorio, la oficina de prácticas profesionales creó una alternativa virtual llamada Ruta de empleabilidad, alojada en Brightspace, que ofrece los mismos conocimientos del curso Equipamiento para el trabajo. Este curso ha funcionado de manera correcta durante al menos 2 años en la universidad, logrando que una gran cantidad de estudiantes lo utilicen como alternativa de manera satisfactoria; sin embargo, ha generado algunos problemas para la oficina. Los problemas radican principalmente en que el curso es muy monótono, poco gráfico e intuitivo, lo que hace que los estudiantes no se sientan totalmente atraídos a este, y por tanto terminen resolviendo las actividades del curso de manera incorrecta o directamente no lo quieran hacer; otro problema principal es la falta de automatización de revisión de actividades de los estudiantes, pues como se tiene el curso actualmente, es necesario revisar actividades una a una de aproximadamente 600 estudiantes que participan del curso por semestre, lo cual lleva un gasto físico y de tiempo arduo ya que dedican alrededor de 8 horas semanales solamente a calificar. Es por esto que se decidió iniciar con la creación de un proyecto de desarrollo de software para implementar la 'Ruta de empleabilidad' como un curso totalmente reformado en cuanto a calidad gráfica y dinámica, nuevas actividades interactivas y la automatización de resultados para un mejor manejo de la información con la oficina institucional de prácticas profesionales.

Capítulo 1

Descripción del Problema

1.1. Planteamiento del Problema

El curso 'Ruta de empleabilidad' que usa la oficina de prácticas institucionales para promover y brindar apoyo en el acceso al campo laboral a los estudiantes, se encuentra activo hace aproximadamente 2 años, en donde los estudiantes han tenido la oportunidad de participar del curso de forma virtual, resolviendo las actividades que se les proponen de la misma forma que las realizarían para cualquier asignatura de la universidad. El curso ha cumplido su función a lo largo de varias generaciones de estudiantes, quienes han completado y presentado exitosamente las distintas actividades. Sin embargo, tanto la Oficina Institucional de Prácticas Profesionales como los estudiantes que lo han cursado han identificado algunos aspectos que requieren mejoras.

Uno de los principales problemas que se ha identificado es la falta de interactividad y dinamismo en el curso como se muestra en la figura 1.1 donde se encuentra un mapa con acciones limitadas, botones que llevan a enlaces incorrectos y carencia de dinamismo. A pesar de que la narrativa del curso simula un viaje por continentes, donde los estudiantes deberían adquirir experiencia del campo laboral en cada uno de ellos, el diseño visual actual no logra transmitir esta experiencia inmersiva. Los estudiantes no sienten que están 'viajando' o progresando por diferentes áreas del conocimiento, sino que simplemente navegan por una serie de actividades virtuales sin cohesión ni atractivo visual. El diseño no los invita a explorar ni les ofrece elementos interactivos que los motiven a seguir adelante en su viaje.

Otro problema es que las actividades del curso se limitan a la resolución de archivos PDF planos como se observa en la figura 1.2, donde los estudiantes deben responder preguntas y enviarlas para su evaluación. Esta metodología, que ha sido funcional, resulta monótona y no estimula el compromiso o la motivación de los estudiantes. La repetición de este tipo de actividades fomenta una experiencia estática que no aprovecha las ventajas que un entorno virtual interactivo y gamificado podría ofrecer [4]. Como resultado, los estudiantes perciben el curso como una asignatura más, sin características que lo distingan o que lo hagan relevante a su preparación para el mundo laboral.

En la figura 1.3 se observa que la Oficina de Prácticas enfrenta dificultades en la gestión y evaluación del curso. Actualmente, la revisión de las actividades se realiza de manera manual, lo que implica que los responsables deben revisar los resultados de los estudiantes de forma individual. Este proceso además de tedioso, consume mucho tiempo, ya que se trata de un proceso de revisión de al menos 48 actividades, 600 estudiantes por semestre y una dedicación de al menos 8 horas

semanales únicamente por calificar, lo que afecta la eficiencia del departamento y también puede retrasar el feedback que los estudiantes reciben sobre su progreso. La falta de automatización en el seguimiento y evaluación del curso complica la capacidad del departamento para monitorear eficazmente el desempeño de los estudiantes y proporcionar una experiencia educativa fluida.

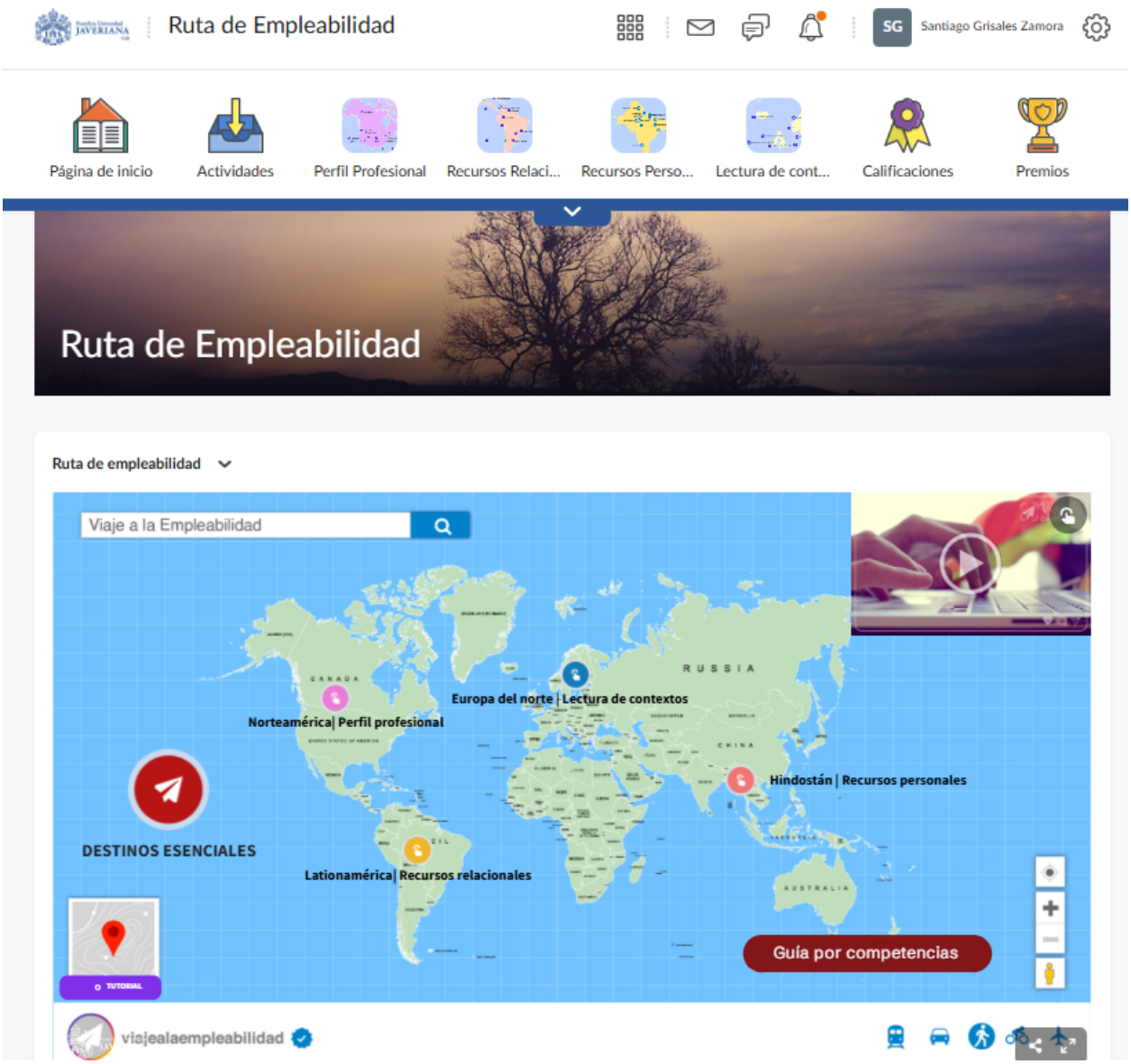


Figura 1.1: Carencia de interactividad

Figura 1.2: Actividades en pdf

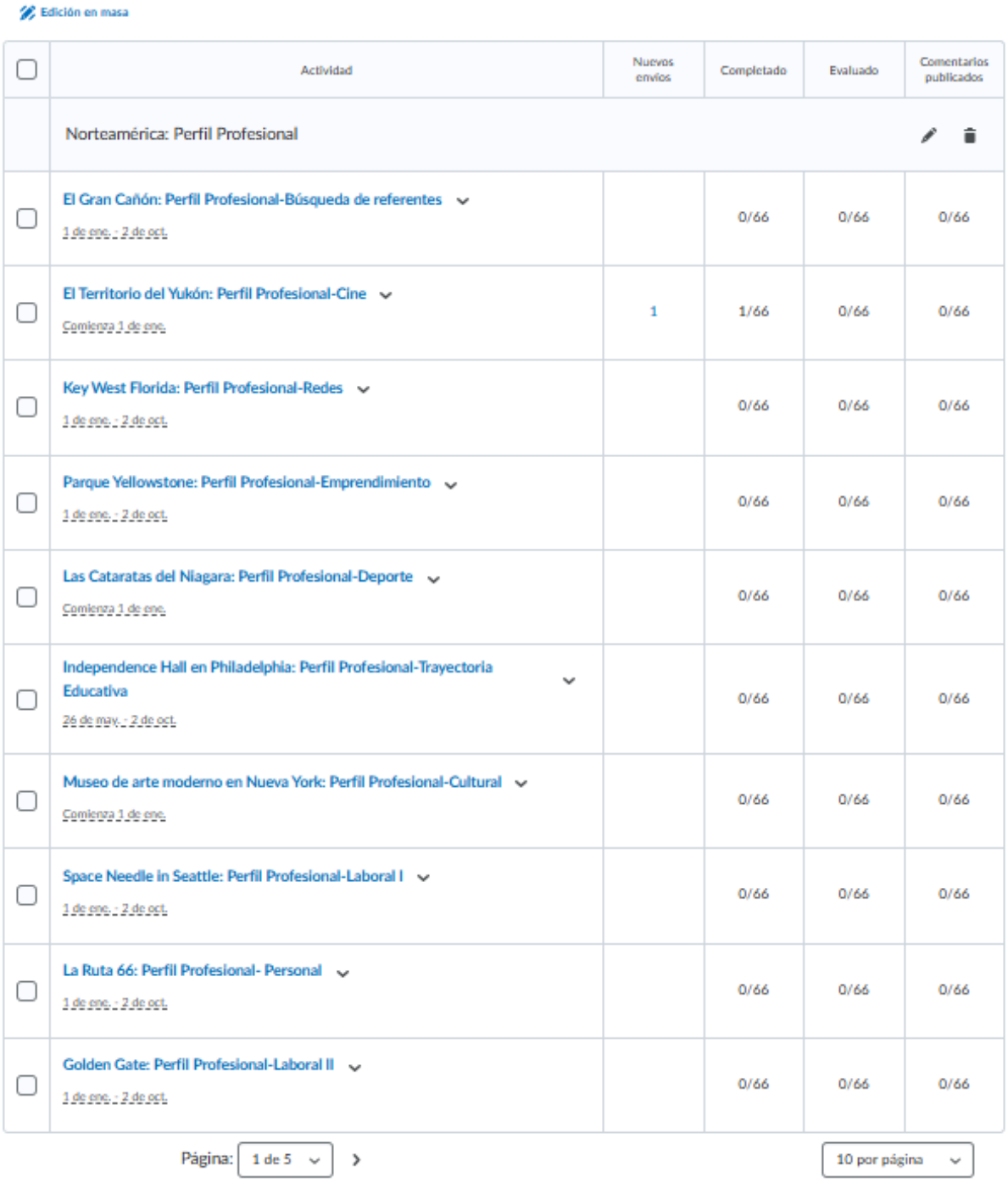


Figura 1.3: Calificar actividades

1.1.1. Formulación

¿Cómo construir y estructurar una aplicación web educativa que brinde una experiencia de aprendizaje más atractiva e inmersiva para los estudiantes del curso 'Ruta de empleabilidad' y, además, ayude a la oficina institucional de prácticas profesionales a automatizar la calificación de las actividades del aplicativo para disminuir la carga de trabajo y enfocarse en una mejor logística del curso?

1.1.2. Sistematización

1. ¿Qué tecnologías y frameworks se definirán y utilizarán para el desarrollo del backend y frontend?

2. ¿Qué criterios de diseño se aplicarán en la creación de la interfaz gráfica del aplicativo para asegurar una experiencia interactiva y accesible tanto para estudiantes como para administradores?

3. ¿Cómo se gestionará la autenticación y autorización de estudiantes y administradores, asegurando la protección de datos y el acceso controlado?

4. ¿Cómo se diseñará e integrará el sistema de persistencia de datos, y qué mecanismos se implementarían para permitir el análisis del progreso estudiantil y facilitar la gestión académica?

5. ¿De qué forma se evaluará la aplicación en términos de usabilidad, participación de los estudiantes y eficiencia administrativa?

1.2. Objetivos

1.2.1. Objetivo General

Diseñar y estructurar una aplicación web educativa interactiva y con diseño gamificado que integre funcionalidades de autenticación, gestión de contenidos, seguimiento del progreso y evaluación automatizada, orientada a fortalecer el aprendizaje autónomo y la motivación de los estudiantes del curso 'Ruta de empleabilidad', apoyando la gestión académica y permitiendo evaluar la usabilidad del aplicativo.

1.2.2. Objetivos Específicos

1. Definir la arquitectura tecnológica de la aplicación web educativa, especificando sus principales componentes y frameworks.

2. Diseñar la interfaz interactiva del aplicativo con elementos de gamificación.

3. Desarrollar un prototipo funcional de la aplicación web que integre módulos de autenticación, gestión de contenidos, seguimiento del progreso y evaluación automatizada.

4. Crear una base de datos que permita registrar y consultar el desempeño de los estudiantes y la información administrativa del curso.

5. Realizar pruebas piloto con dos tipos de usuarios para recopilar información sobre su uso y funcionamiento.

1.3. Justificación

El uso de aplicaciones web interactivas y con un diseño gamificado ha demostrado ser eficaz para mejorar la motivación, la participación y el compromiso de los usuarios, lo que impulsa aún más su nivel de aprendizaje [3]. Este tipo de gamificación en entornos educativos implementa elementos típicos de los juegos como puntos, niveles, recompensas, entornos interactivos y gráficos dinámicos, lo cual facilita la asimilación de contextos y el desarrollo de habilidades como la atención, colaboración y resolución de problemas [1].

El uso de elementos gamificados en el aprendizaje no solo aumenta la motivación de los estudiantes, sino que también promueve un sentido de logro y progreso, lo que puede ayudar a reducir aspectos como la ansiedad y aumentar la eficiencia del trabajo [2]. Además, un entorno de aprendizaje gamificado puede fomentar la interacción social y la colaboración entre estudiantes, habilidades indiscutiblemente necesarias en el ámbito laboral [5], lo cual es importante ya que es justo una de las metas del propio curso.

Es por esto que, dadas las circunstancias actuales del curso que no permiten aprovechar las ventajas al máximo y las críticas que surgen debido a su rigidez, es necesario hacer una reestructuración y mejora hacia los diseños gamificados y dinámicos, los cuales podrían mejorar la experiencia de aprendizaje y hacerla más atractiva y relevante para los estudiantes [6].

1.4. Delimitaciones y Alcances

1.4.1. Delimitaciones

1. La aplicación estará diseñada específicamente para el ámbito académico y exclusivamente para el curso 'Ruta de empleabilidad' de la oficina institucional de prácticas.

2. La aplicación será enfocada principalmente en 2 tipos de usuarios: estudiantes (quienes participarán del curso, realizarán actividades, recibirán calificaciones y podrán ver su progreso), y los administradores (encargados de monitorear el progreso de los estudiantes, gestionar actividades y evaluar los resultados de los informes generados).

3. Se utilizarán frameworks y librerías web para garantizar una experiencia fluida e interactiva.

4. El desarrollo de la aplicación será el siguiente:

    - Frontend: El frontend de la aplicación será una interfaz gráfica responsive y gamificada que ofrecerá una experiencia interactiva para los estudiantes. Estará compuesta de

un módulo de introducción, un módulo de información o perfil del usuario, el módulo para realizar las actividades y la finalización y aprobación del curso con su respectivo certificado.

- Backend: Implementación del sistema de autenticación para estudiantes y administradores según su rol, con protección de rutas y gestión de usuarios. Además, se incluye el manejo y almacenamiento de los datos, específicamente los resultados de las actividades de cada estudiante, en una base de datos Oracle.

5. Se priorizó una sistematización que permita futuras mejoras y expansión a más cursos sin comprometer el rendimiento.

6. El desarrollo se realizó dentro del período establecido por el proyecto de grado.

1.4.2. Alcances

1. Ingreso y autenticación segura teniendo conexión con los servicios de la universidad.

2. Módulo de gestión para administradores, donde podrán visualizar y descargar los progresos de los estudiantes.

3. Interfaz de usuario atractiva e interactiva, con un mapa visual que represente la ruta de aprendizaje.

1.4.3. Entregables

1. Prototipo de la aplicación web estructurada con una interfaz gráfica funcional e interactiva, el sistema de autenticación e identificación de roles y simulación del progreso en las actividades.

2. Documento detallado sobre el proceso de diseño de la aplicación, las decisiones de diseño y las herramientas utilizadas.

1.5. Resultados esperados

1. Prototipo funcional del aplicativo donde usuarios tanto estudiantes como administradores puedan interactuar con la interfaz realizan las acciones que les corresponden dentro del aplicativo.

2. Se realizarán distintas pruebas técnicas y de usuario para validar el correcto funcionamiento del sistema, además de una información de los posibles errores.

3. El aplicativo cumplirá con las normativas de privacidad y protección de datos, asegurando que la información personal de los estudiantes sea manejada de manera ética y confidencial.

4. Se proporcionará una documentación que incluya diagramas del sistema y descripciones de las funcionalidades implementadas.

1.6. Metodología

1.6.1. Tipo de Estudio

El proyecto es un estudio experimental de desarrollo de software con enfoque mixto. La naturaleza experimental del proyecto radica en la creación y validación de una aplicación web gamificada que transforme la experiencia de aprendizaje del curso Ruta de Empleabilidadz automatice los procesos de evaluación, sirviendo como herramienta de apoyo para los administradores de la Oficina Institucional de Prácticas Profesionales en la gestión eficiente del curso. Para el desarrollo y validación del proyecto se utilizó una metodología ágil adaptada, específicamente una combinación de principios de Scrum y desarrollo iterativo e incremental. Esta metodología se estructuró en entregables de desarrollo de 2 semanas, donde cada iteración producía un incremento funcional del sistema y se mostraban los resultados al equipo de trabajo para aprobación y mejoras.

1.6.2. Actividades realizadas

1. Se analizaron las necesidades y requisitos que fueron usados para establecer la estructura del aplicativo.

    - Investigación sobre la gamificación y el aprendizaje en línea: Se revisaron estudios y artículos que abordaran temas como la gamificación en la educación y su impacto en la motivación y el aprendizaje.
    - Se realizaron reuniones con estudiantes, tutores y administrativos que tuvieron contacto con el curso anterior y se recopiló información sobre sus expectativas y necesidades.
    - Se definió y documentó los requisitos funcionales y no funcionales.

2. Se diseñó la arquitectura para el back y frontend de la aplicación, incluyendo el sistema de evaluación de actividades.

    - Se definieron los componentes principales de la aplicación, su interacción y flujo de datos.
    - Se crearon mockups, presentaciones y prototipos de la interfaz de usuario.
    - Se definió cómo se almacenará la información de los usuarios, las actividades del curso y sus resultados.
    - Se definieron las estrategias y sistema para evaluar las actividades de acuerdo a los aciertos, errores y tiempo en que se resuelvan.

3. Se desarrollaron los scripts necesarios para el sistema interactivo, manejo y guarda de información.

    - Se seleccionaron las herramientas y tecnologías adecuadas para realizar el proyecto en su totalidad, teniendo en cuenta el frontend, backend, almacenamiento de usuario, protección de rutas, arquitectura de la aplicación y su despliegue.

- Se desarrolló el código por componentes de las características definidas como gamificación y diseño, gestión de usuarios y almacenamiento de actividades.
- Se desarrolló la interfaz de usuario gamificada.

4. Se realizaron pruebas de validación y de usuario, para recopilar información sobre el desempeño y la experiencia en general.
5. Se preparó el entorno de producción configurando el servidor y la infraestructura necesaria para el desarrollo correcto de la aplicación.
6. Se realizó la documentación sobre el progreso y el desarrollo de trabajo.
    - Se documentó el proceso de desarrollo.

Capítulo 2

Desarrollo del Proyecto

2.1. Marco de Referencia

2.1.1. Marco Teórico

2.1.1.1. Gamificación en la educación

- Gamificación: La gamificación hace referencia al uso de elementos y diseño de videojuegos en contextos como la educación con el objetivo de aumentar la motivación, el compromiso y la participación de los usuarios. En el ámbito educativo, la gamificación busca transformar el proceso de aprendizaje en una experiencia más atractiva y dinámica, utilizando mecánicas como puntos, niveles, recompensas y narrativas que fomentan la interacción y el aprendizaje activo [1].
- Elementos de gamificación [/]:
  - Logros: Todo tipo de comentarios que elogian las acciones específicas de los jugadores. Algunos ejemplos y sinónimos son insignias, medallas y trofeos.
  - Competencia: Cuando dos o más jugadores compiten entre sí por un objetivo común. Algunos ejemplos y sinónimos son jugador contra jugador, marcadores y conflicto.
  - Ruta guiada: Decisiones que el jugador debe tomar para avanzar en el juego. Algunos ejemplos y sinónimos son juicios y elecciones forzadas (no confundir con narrativa).
  - Narrativa: Orden de los eventos que ocurren en un juego. Estas son decisiones influenciadas por las acciones de los jugadores. Algunos ejemplos y sinónimos son las estrategias que el jugador usa para superar un nivel.
  - Objetivos: Guía las acciones de los jugadores. Cuantificables o espaciales, de corto a largo plazo. Algunos ejemplos y sinónimos son misiones, aventuras e hitos.
  - Puntos: Unidad utilizada para medir el rendimiento de los usuarios. Algunos ejemplos y sinónimos son puntuaciones, número de bajas y puntos de experiencia.
  - Progreso: Esto permite a los jugadores ubicar sus avances dentro del juego.
  - Actividades: Desafíos dentro del juego que deberían hacer pensar al jugador.
  - Reputación: Títulos que el jugador acumula dentro del juego.
  - Información de usuario: Información visible utilizada por el jugador, relacionada con sus resultados dentro del juego.

2.1.1.2. Interactividad en entornos de aprendizaje

- Importancia de la interactividad: La interactividad en entornos de aprendizaje se refiere a la capacidad de los estudiantes para relacionarse activamente con el contenido, sus compañeros, sus progresos y resultados, generando un aprendizaje más profundo y significativo. Esta puede manifestarse a través de actividades prácticas y el uso de tecnología con resultados inmediatos [6].

- Diseño de experiencias interactivas: Tipos de implementación de evaluación [8].

  - Prueba rápida virtual.
  - Ensayos tipo documento de word.
  - Entrevistas simuladas en persona y virtual.
  - Trabajo colaborativo.
  - Pruebas virtuales escritas y de selección.
  - Tableros de discusión.
  - Herramientas de blog con procesos secuenciales.
  - Actividades con retroalimentación inmediata.

2.1.1.3. Tecnologías educativas y herramientas de aprendizaje

- Las aplicaciones web en la educación son plataformas digitales que facilitan el acceso a recursos educativos, la interacción entre estudiantes y docentes, y la gestión del aprendizaje. Estas herramientas permiten la creación de entornos de aprendizaje flexibles y accesibles, donde los estudiantes pueden participar en actividades interactivas, recibir retroalimentación y colaborar con sus compañeros [4].

- Herramientas como Kahoot, Google Forms, Mentimeter, Educaplay y demás, son aplicaciones dinámicas e interactivas que cuentan con las herramientas adecuadas para dar retroalimentación automática y seguimiento personalizado a los resultados de los usuarios [9].

2.1.1.4. Arquitectura limpia

La arquitectura limpia es un conjunto de principios y patrones de diseño de software propuestos por el desarrollador y fundador Robert C. Martin. Esta fue presentada por primera vez por su fundador durante una serie de charlas y conferencias alrededor del año 2010. La idea principal de la arquitectura es separar la lógica de negocio de la infraestructura, de forma que las preocupaciones se distribuyan en diferentes capas definidas con reglas estrictas que permitan interactuar controladamente entre sí como se muestra en la figura 2.1. Esta arquitectura combina los principios de la arquitectura hexagonal, la Onion y otras variantes [15].

- Capa de entidades: Entidades centrales y objetos de negocio del dominio. Son estructuras que encapsulan los datos principales del negocio. Son el núcleo de la aplicación y puden ser variables, constantes, clases y objetos del negocio.
- Capa de casos de uso: Contiene la lógica de la aplicación y coordina la interacción entre las entidades del dominio. Los casos de uso encapsulan la lógica de negocio específica para cada función o acción que debe realizar la aplicación.
- Capa de adaptadores: Son las interfaces que los Casos de Uso necesitan para interactuar con el mundo externo como la interfaz de usuario (UI), la base de datos y otros servicios externos. Estos convierten los datos del exterior en un formato que las capas internas puedan entender y viceversa.
- Capa de interfaces externas: Esta capa contiene el código que interactúa con bibliotecas, frameworks y herramientas externas. Es la capa más externa de la arquitectura. Bases de datos, ORM, sistema de archivos, dispositivos, API externas, HTTP, CLI, API REST.

2.1.1.5. Bases de datos relacionales

Una base de datos relacional es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Las bases de datos relacionales se basan en el modelo relacional, una forma intuitiva y directa de representar datos en tablas. En una base de datos relacional, cada fila en una tabla es un registro con una ID única, llamada clave. Las columnas de la tabla contienen los atributos de los datos y cada registro suele tener un valor para cada atributo, lo que simplifica la creación de relaciones entre los puntos de datos.

Estructura de las bases de datos relacionales
El modelo relacional significa que las estructuras de datos lógicas (las tablas de datos, las vistas y los índices) están separadas de las estructuras de almacenamiento físicas. Gracias a esta separación, los administradores de bases de datos pueden gestionar el almacenamiento físico de datos sin que eso influya en el acceso a esos datos como estructura lógica.

La distinción entre lógico y físico se aplica también a las operaciones de base de datos, que son acciones que permiten a las aplicaciones manipular los datos y las estructuras de la base de datos. Con las operaciones lógicas, las aplicaciones pueden especificar el contenido que necesitan, mientras que las operaciones físicas determinan cómo se debe acceder a esos datos y llevan a cabo la tarea.
El modelo relacional es sencillo pero potente, y lo utilizan organizaciones de todos los tipos y tamaños para una gran variedad de aplicaciones con datos. Las bases de datos relacionales se usan para rastrear inventarios, procesar transacciones de comercio electrónico, administrar cantidades enormes y esenciales de información de clientes y mucho más. Las bases de datos relacionales se pueden emplear para cualquier aplicación de datos en la que los puntos de datos se relacionen entre sí y deban gestionarse de forma segura, conforme a normas y de un modo uniforme [17].

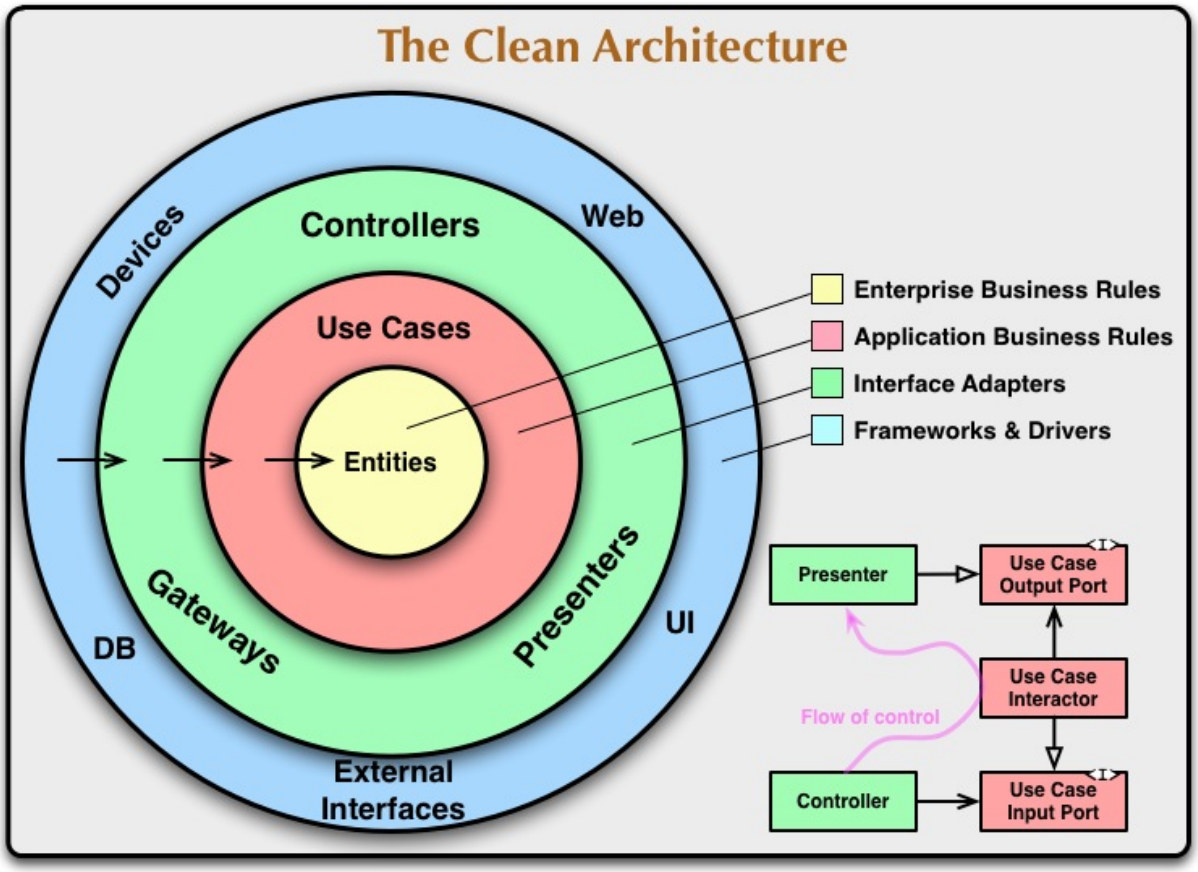


Figura 2.1: Clean architecture

2.1.1.6. Tailwind CSS

Tailwind CSS es un framework de CSS de código abierto que proporciona clases de utilidad de bajo nivel para construir diseños personalizados directamente en el HTML. Su enfoque permite un desarrollo más rápido y consistente, ya que los desarrolladores pueden construir interfaces complejas combinando múltiples clases utilitarias sin necesidad de escribir CSS personalizado. Tailwind CSS incluye un sistema de diseño robusto con espaciado, colores, tipografía y breakpoints responsivos predefinidos, facilitando la creación de interfaces modernas y responsivas [26].

2.1.1.7. SQLAlchemy

SQLAlchemy es una biblioteca de mapeo objeto-relacional (ORM) para Python que proporciona una interfaz de alto nivel para interactuar con bases de datos relacionales. SQLAlchemy permite a los desarrolladores trabajar con bases de datos utilizando objetos Python en lugar de escribir consultas SQL directamente. SQLAlchemy soporta múltiples motores de base de datos, incluyendo PostgreSQL, MySQL, SQLite y Oracle, proporcionando una capa de abstracción que facilita la portabilidad del código entre diferentes sistemas de gestión de bases de datos [27].

2.1.1.8. API Rest

Un API es un conjunto de definiciones, protocolos y herramientas que permite la comunicación y el intercambio de datos entre diferentes aplicaciones o sistemas de software. Las APIs facilitan la interoperabilidad, ya que exponen ciertas funcionalidades de una aplicación o servicio para que otros programas puedan utilizarlas sin necesidad de conocer los detalles internos de su implementación. Gracias a estas es posible integrar sistemas heterogéneos, automatizar procesos y habilitar nuevas funcionalidades de forma eficiente, segura y controlada.

Ahora, REST es un estilo de arquitectura de software propuesto por Roy Fielding en el año 2000, el cual establece un conjunto de principios y restricciones para el diseño de servicios web escalables, modulares y mantenibles. Entre sus características principales están que los recursos son identificados mediante URLs, las solicitudes no dependen de un contexto almacenado en el servidor, las respuestas pueden ser almacenadas en caché para mejor rendimiento y que su arquitectura puede estar compuesta por varias capas.

Por lo que, API REST es una interfaz de programación de aplicaciones que implementa los principios de la arquitectura REST para permitir la comunicación entre sistemas a través de internet, utilizando comúnmente el protocolo HTTP. Este tipo de API expone recursos que pueden ser accedidos y manipulados mediante operaciones definidas, donde cada recurso es representado por una URL única y sus interacciones se realizan mediante los métodos HTTP apropiados. El uso de API REST tiene mucha relevancia en el contexto de las arquitecturas orientadas a servicios y el desarrollo de servicios en la nube [18].

2.1.1.9. Json Web Token

En el contexto de arquitecturas orientadas a servicios en sistemas basados en microservicios, los mecanismos de autenticación y autorización adquieren un papel central en la seguridad de las aplicaciones. Este tipo de arquitecturas, al estar compuestas por múltiples servicios independientes expuestos a través de interfaces públicas, incrementan la superficie de ataque disponible para posibles amenazas persistentes avanzadas,, transformando el panorama de riesgos de seguridad. Por esta razón, la implementación adecuada de patrones de autenticación y autorización se considera un componente esencial dentro de cualquier programa de madurez en seguridad de software.

JSON Web Token es un estándar abierto utilizado como mecanismo de autenticación y autorización en arquitecturas distribuidas. Un JWT es un token compacto, autocontenido y firmado digitalmente, que permite transmitir información de forma segura entre las partes involucradas en un sistema. El proceso de autenticación con JWT generalmente sigue los siguientes pasos [19]:

- El usuario proporciona sus credenciales a un servidor de autenticación.
- Si las credenciales son válidas, el servidor emite un JWT firmado que incluye los claims necesarios.
- El cliente almacena el token y lo envía al encabezado HTTP Authorization en cada petición.
- Cada microservicio que recibe la solicitud valida la firma del token y, en función de los claims contenidos, permite o deniega el acceso

2.1.1.10. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacionales de código abierto, orientado a objetos y de propósito general desarrollado inicialmente en la Universidad de California, Berkeley. PostgreSQL es especialmente valorado por su confiabilidad, rendimiento y capacidad para manejar cargas de trabajo complejas, siendo ampliamente utilizado en aplicaciones empresariales, sistemas web y aplicaciones de análisis de datos [29].

2.1.1.11. Docker

Docker es una plataforma de contenedores de código abierto que permite a los desarrolladores empaquetar aplicaciones y sus dependencias en contenedores ligeros y portátiles. Los contenedores Docker proporcionan un entorno de ejecución aislado y consistente, garantizando que las aplicaciones se ejecuten de manera idéntica en diferentes entornos, desde desarrollo hasta producción. Este permite un despliegue más eficiente, escalabilidad horizontal y una gestión simplificada de dependencias, facilitando el desarrollo, pruebas y despliegue de aplicaciones en entornos distribuidos [30].

2.1.1.12. Frameworks

Los frameworks de desarrollo son herramientas que proporcionan estructuras predefinidas, bibliotecas, convenciones y patrones de diseño que simplifican y aceleran el proceso de construcción de software. Estos frameworks permiten a los desarrolladores enfocarse en la lógica de negocio de las aplicaciones, reduciendo la complejidad técnica de tareas comunes como el manejo de rutas, la gestión de estado, la conexión a bases de datos o la renderización de interfaces de usuario.

Los frameworks de frontend están diseñados para facilitar el desarrollo de la interfaz de usuario y la experiencia del usuario en aplicaciones web. Estos frameworks gestionan la lógica de presentación, la interacción del usuario y, cada vez más, el manejo del estado de la aplicación en el lado del cliente. Frameworks como React, Angular, Vue, Svelte y Blazor representan algunos de los paradigmas declarativos más influyentes en la actualidad.

Los frameworks de backend proporcionan los cimientos para desarrollar la lógica de negocio, la gestión de datos y la comunicación entre servicios en el lado del servidor. Estos frameworks gestionan aspectos esenciales como la gestión de rutas, conexiones de base de datos, lógica de negocio, comunicación entre microservicios, escalabilidad y manejo de protocolos de autenticación y seguridad. Entre los frameworks de backend más populares destacan Node.js (con Express.js), Django, Spring Boot, ASP.NET Core, Laravel y Ruby on Rails, cada uno optimizado para diferentes lenguajes de programación y entornos de desarrollo [20].

2.1.1.13. Pydantic

Pydantic es una biblioteca de Python para la validación de datos y configuración de aplicaciones que utiliza anotaciones de tipo de Python para definir modelos de datos. Pydantic es especialmente popular en el desarrollo de APIs web con FastAPI, donde se utiliza para validar datos de entrada, estructurar respuestas y generar documentación automática de la API. La biblioteca soporta tipos complejos, validaciones personalizadas y conversiones automáticas de tipos, facilitando el desarrollo de aplicaciones robustas y mantenibles [34].

2.1.1.14. Screaming architecture

Screaming Architecture es un principio de diseño de software introducido por Robert C. Martin en su libro 'Clean Architecture' que establece que la estructura de una aplicación debe 'gritar' o comunicar claramente su propósito y dominio de negocio. Esta arquitectura no es solo una técnica de organización de código, sino una filosofía que prioriza la comunicación clara de intención sobre la conveniencia técnica. Es la manifestación del principio de que 'el código es comunicación' comunicación entre desarrolladores, stakeholders y el futuro mantenimiento del sistema [23].

Antes del Screaming Architecture:

- La estructura refleja patrones técnicos con decisiones tecnológicas y no de negocio.

- Los nuevos desarrolladores tienen una curva de aprendizaje más grande debido a que tardan mucho en entender el propósito real del proyecto.
- El código se vuelve difícil de mantener porque no comunica su propósito.

Después del Screaming Architecture:

- La estructura refleja y comunica inmediatamente el dominio de negocio.
- Los patrones técnicos están al servicio del dominio.
- La arquitectura evoluciona con el negocio, no con las tecnologías.

Principios fundamentales.

1. Principio de revelación de intención: La arquitectura debe revelar la intención del sistema de manera inmediata. Un desarrollador que ve la estructura por primera vez debe poder decir: 'Ah, esto es un sistema de enseñanza para la empleabilidad' sin ver una sola línea de código.
2. Principio de independencia de framworks: Los frameworks son herramientas, no la razón de ser del sistema. La arquitectura debe poder cambiar de framework sin afectar la lógica de negocio.
3. Principio de separación de preocupaciones:
   - De negocio: Reglas, entidades, casos de uso, clases.
   - De técnicas: Persistencia, escalabilidad, UI, frameworks, APIs.
4. Principio de inversión de dependencias: Las dependencias apuntan hacia el dominio de negocio, no hacia los detalles técnicos.

2.1.1.15. TypeScript

TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft que extiende JavaScript agregando tipado estático opcional, clases, interfaces y otras características de programación orientada a objetos. El compilador de TypeScript transpila el código TypeScript a JavaScript puro, permitiendo que se ejecute en cualquier entorno que soporte JavaScript. Las principales ventajas de TypeScript incluyen la detección temprana de errores durante el desarrollo, mejor documentación del código a través de tipos, y herramientas de desarrollo más avanzadas que facilitan el mantenimiento y la escalabilidad de aplicaciones complejas [25].

2.1.1.16. ORM (Objeto relacional)

ORM (Object-Relational Mapping) es una técnica de programación que permite a los desarrolladores trabajar con bases de datos relacionales utilizando objetos de programación orientada a

objetos en lugar de escribir consultas SQL directamente. El ORM actúa como una capa de abstracción que mapea automáticamente las tablas de la base de datos a clases de objetos, las filas a instancias de objetos, y las columnas a atributos de objetos. Los ORMs proporcionan funcionalidades como mapeo automático de relaciones, gestión de transacciones, caché de consultas y generación automática de consultas SQL optimizadas, mejorando la productividad del desarrollo y reduciendo la complejidad del código relacionado con la persistencia de datos [28].

2.1.1.17. Postman

Postman es una plataforma de desarrollo de APIs que proporciona herramientas para diseñar, desarrollar, probar y documentar APIs RESTful. La aplicación permite a los desarrolladores crear y enviar solicitudes HTTP a endpoints de API, visualizar respuestas, automatizar pruebas y colaborar en el desarrollo de APIs. La plataforma es ampliamente utilizada en el desarrollo de software para probar APIs durante el desarrollo, validar la funcionalidad de endpoints y facilitar la integración entre diferentes servicios [31].

2.1.1.18. CRUD

CRUD es un acrónimo que representa las cuatro operaciones fundamentales de persistencia de datos en sistemas de gestión de bases de datos: Create (Crear), Read (Leer), Update (Actualizar) y Delete (Eliminar). Estas operaciones constituyen la base de la manipulación de datos en aplicaciones de software. Create se refiere al ingreso de nuevos registros en la base de datos, Read implica la consulta y recuperación de datos existentes, Update permite la modificación de registros previamente almacenados, y Delete se refiere a la eliminación de registros de la base de datos [32].

2.1.1.19. DTO (Data Transfer Objects)

Los Data Transfer Objects (DTOs) son objetos de programación que se utilizan para transferir datos entre diferentes capas o componentes de una aplicación, especialmente en arquitecturas de software complejas. Estos encapsulan datos relacionados en una sola estructura, facilitando la comunicación entre diferentes partes del sistema sin exponer la estructura interna de los objetos de dominio. Los DTOs son particularmente útiles en aplicaciones web para estructurar las respuestas de APIs, validar datos de entrada y separar la lógica de presentación de la lógica de negocio [33].

2.2. Trabajos Relacionados

1. Aplicación web para evaluación formativa universitaria basada en competencias: En este artículo se describe el Sistema de Evaluación Web desarrollado para la Universidad del Valle (SEUV). Las principales características de este sistema son: 1) se propone como una aplicación de apoyo a la educación virtual, 2) permite la evaluación formativa de los alumnos, 3) todos los ítems del banco de preguntas se clasifican bajo competencias específicas y transversales, y

4) permite establecer la relación entre la jerarquía de conocimientos de los currículos universitarios y las áreas temáticas definidas para los Exámenes de Calidad de Educación Superior en Colombia (ECAES). La arquitectura de este sistema se constituye de cinco sub-módulos, donde cada uno se encarga de diferentes etapas del proceso de evaluación [10].

2. Developing interactive educational engineering software for the world wide web with Java: Este artículo ilustra el diseño y la implementación de un applet de Java para su uso en programas de ingeniería de propulsión. El applet Simulador de Turbinas de Gas de Java proporciona un entorno gráfico interactivo que permite la construcción y el análisis rápidos y eficientes de sistemas arbitrarios de turbinas de gas. El sistema de simulación combina una interfaz gráfica de usuario y un método de análisis aerotermodinámico de turbinas de gas transitorio, promediado espacialmente, ambos completamente codificados en lenguaje Java. El paquete combinado proporciona herramientas analíticas, gráficas y de gestión de datos que permiten al estudiante construir y controlar simulaciones dinámicas de turbinas de gas mediante la manipulación de objetos gráficos en la pantalla del ordenador [11].

3. A gamified web based system for computer programming learning: La disponibilidad de herramientas de evaluación automatizada para tareas de programación informática puede ser una ventaja significativa en la educación en Ciencias de la Computación. Los sistemas que ofrecen este tipo de servicio se basan en una interfaz que permite administrar las tareas (ejercicios para entrenar habilidades de programación) y mostrar los resultados, acompañados de retroalimentación significativa. Para obtener estos resultados, aplican técnicas que van desde el análisis estático de la corrección del programa hasta la evaluación basada en pruebas. Estos sistemas también pueden apoyar la Programación Competitiva, reconocida por su valor educativo. Desarrollamos el sistema 2TSW, que facilita la corrección automatizada de tareas de programación informática en un entorno web gamificado [12].

4. EDUMAT: herramienta web gamificada para la enseñanza de operaciones elementales: Esta investigación propone un método para la enseñanza de operaciones elementales basadas en la gamificación y las tecnologías de la información y la comunicación con el objetivo de promover las mejores prácticas en el contexto de la educación y mejorar el rendimiento de los estudiantes de de educación básica en el área de matemáticas, centrado la división con sustracción sucesiva, que incluye las cuatro operaciones elementales (suma, resta, multiplicación y división) en un solo procedimiento. El método se implantó en una aplicación web con un entorno interactivo y didáctico donde a través del juego se puso a prueba las destrezas y el conocimiento que los estudiantes adquirían en el aula. La intención, fue verificar mediante una prueba de implantación en un entorno educativo si el método es realmente efectivo y si fomenta el interés de incluir estrategias didácticas de aprendizaje en las aulas de clase [13].

5. Plataforma web gamificada para el entorno universitario: Este artículo presenta una propuesta de plataforma de gamificación basada en web y pensada principalmente para su uso dentro del entorno universitario, teniendo en consideración los distintos tipos de usuario que pueden existir dentro de la misma. En este artículo se presentarán los elementos que formarán parte

de la plataforma, la estructura con la que cuenta su modelo de datos, y la visión que se podrá experimentar por parte de un usuario dentro de una actividad de un curso gamificado en ella disfrutando de mecánicas y herramientas que es difícil de ver en plataformas de gamificadas en entorno universitario [14].

Capítulo 3

Diseño del Proyecto

3.1. Bitácora de desarrollo del proyecto

3.1.1. Inmersión en la problemática | 15/09/24 - 15/12/24

En este tiempo se asignó el proyecto con el enfoque principal de darle un vistazo muy por encima a los objetos visuales y funcionales que se encontraban en el curso alojado en el LMS de Brightspace de la universidad, para que de esta forma se pudieran identificar las problemáticas que se habían comentado al iniciar el proyecto, investigar herramientas que permitieran buscar la solución a las problemáticas y, además, saber si eran viables para poder construirse durante el tiempo de realización del trabajo de grado.
Durante este tiempo de inmersión, se intentó recopilar información del desarrollo del curso como se tenía montado en Brightspace, para poder guiarse y tener una idea de cómo proceder con el proyecto; sin embargo, no se logró encontrar gran información ya que quienes montaron el curso desarrollaron cada una de las ventanas de visualización en una herramienta de contenidos visuales llamada Genially, la cual era trabajo meramente de diseño y no era posible ni se tenían los diseños para poderlos moldear y modificar a la forma que se requería por parte de la administración del departamento.
Buscando alguna guía que permitiera enfocar el proyecto, surgió el departamento 'Centro Magis', los cuales tenían información acerca de la construcción del curso, ya que dentro de sus funciones en la universidad se encuentra la de implementar cursos en Brightspace para las distintas asignaturas del medio universitario, y también habían estado en contacto con las personas que montaron el curso para el departamento de prácticas. Por este motivo, se tuvieron bastantes reuniones con ellos, en donde se habló de los distintos cursos que habían implementado anteriormente, y de las ventajas que tenía el dejar el curso 'Ruta de empleabilidad' alojado en Brightspace. Siguiendo con las reuniones, el centro Magis brinda los permisos de administrador dentro de un curso de prueba que tenían de la 'Ruta de empleabilidad' y también crean un curso en Brightspace desde cero para poder modificarlo y configurarlo para hacer distintas pruebas.
Después de la información recibida y el espacio de trabajo con los cursos a disposición, se inició con el proceso de aprender cómo funcionaban los cursos en Brightspace, qué estaba detrás de la estructura de cada curso y si al fin y al cabo se podrían aplicar conocimientos de la carrera de ingeniería de sistemas y no simplemente implementar un curso en un LMS con imágenes diseñadas en Genially. Afortunadamente, y con ayuda del centro Magis, se logró descubrir que se podían construir los cursos con código HTML y diseño CSS para la construcción de ventanas de visualización, y se decidió seguir ese camino para la construcción del mismo. De este modo, se mantuvo la codificación

en HTML y CSS para poder ir presentando avances al equipo de trabajo. El primer resultado que se consiguió se puede ver en la figura 3.1.

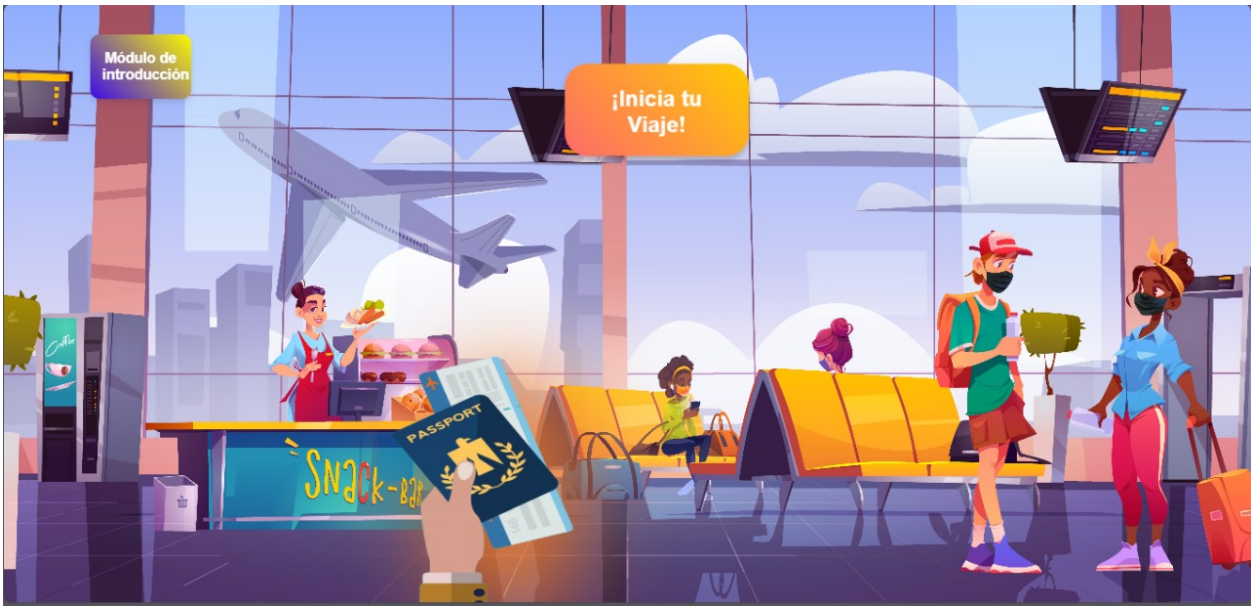


Figura 3.1: Primer avance del aplicativo

3.1.2. Inicio en forma del proyecto | 10/02/25 - 05/03/25

Durante el tiempo intersemestral se propuso la idea de continuar con el proyecto, haciendo uso de los cursos en Brightspace para los cuales el Centro Magis había otorgado permisos, con el fin de replicar y/o adaptar contenidos al desarrollo del curso para la oficina de prácticas. Sin embargo, aparecieron algunas problemáticas: el código que se venía desarrollando en HTML y CSS resultaba muy complejo de integrar en el LMS de Brightspace, ya que la plataforma es bastante rígida en cuanto a diseño y, al momento de incorporar nuevos estilos, exige el uso de tecnologías de desarrollo específicas que se desconocían y no se habían considerado durante las primeras etapas del diseño. Esto impidió que se pudiera implementar de una forma apropiada el proyecto en el LMS, lo cual empezó a generar varias dudas sobre la viabilidad del enfoque que se había pensado en un inicio.
Se decidió hablar con el equipo de trabajo, quienes también manifestaron cierto descontento con el LMS, ya que durante el tiempo que habían tenido el curso en funcionamiento, habían querido realizar varios cambios, pero sus solicitudes no fueron atendidas de manera adecuada y, en general, sentían que el curso nunca logró funcionar de forma óptima. Ante esta problemática, se decidió agendar distintas reuniones tanto con el Centro Magis como con el Centro de Servicios Informáticos (CSI) de la Universidad Javeriana de Cali, con el objetivo de discutir la situación y buscar posibles soluciones.

Dentro de las reuniones que se hicieron con ambas oficinas, se propuso una idea en que se había estado trabajando durante el tiempo intersemestral: el desarrollo de un aplicativo web propio para la Oficina Institucional de Prácticas Profesionales, que funcionara como alternativa al curso Ruta de Empleabilidad. La propuesta buscaba ofrecer una solución original, más flexible y adecuada a las necesidades detectadas al inicio del proyecto. En un principio, la idea no se aceptó de buena manera, ya que las entidades consideraban que el hacer un aplicativo completamente desde cero significaría mucho tiempo a dedicar y, por lo tanto, se decidió darle más oportunidades a brightspace para lograr montar el proyecto. El resultado obtenido en este tiempo se puede ver en las figuras 3.2 y 3.3.

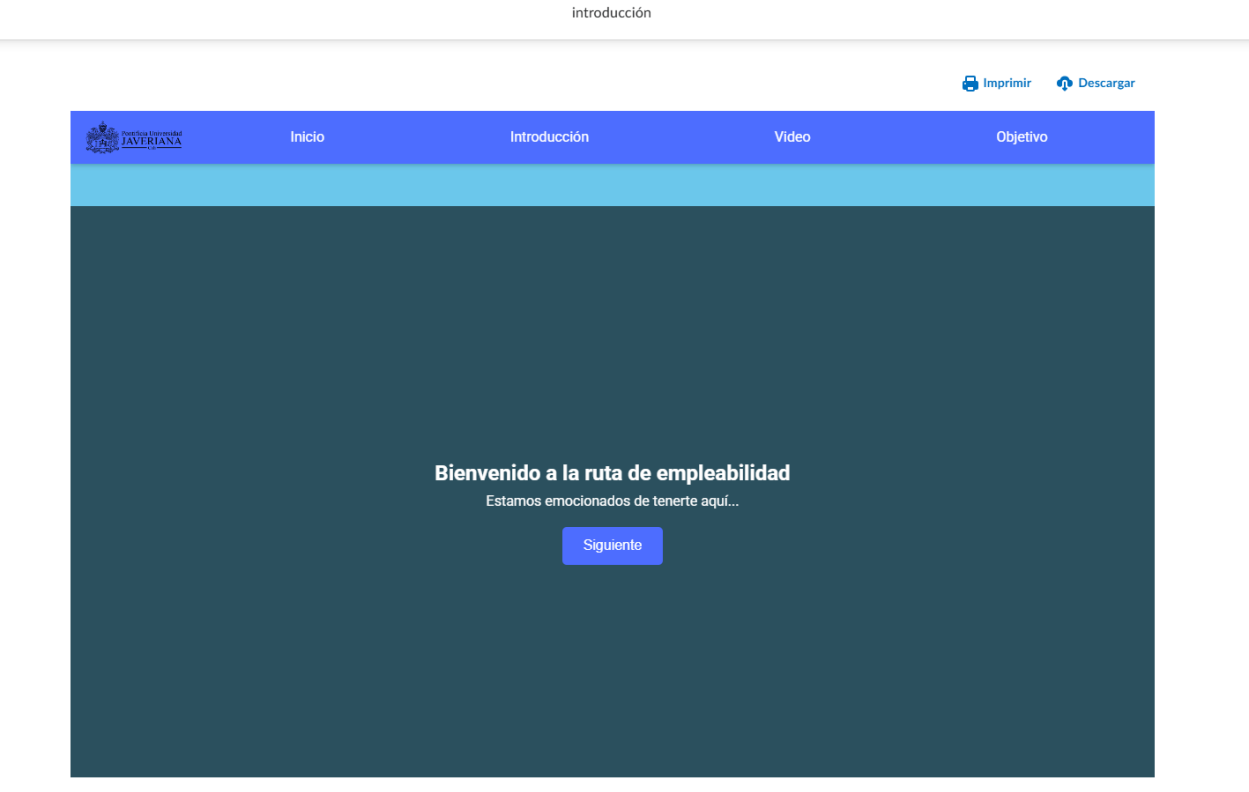


Figura 3.2: Prueba 1 del proyecto en Brightspace

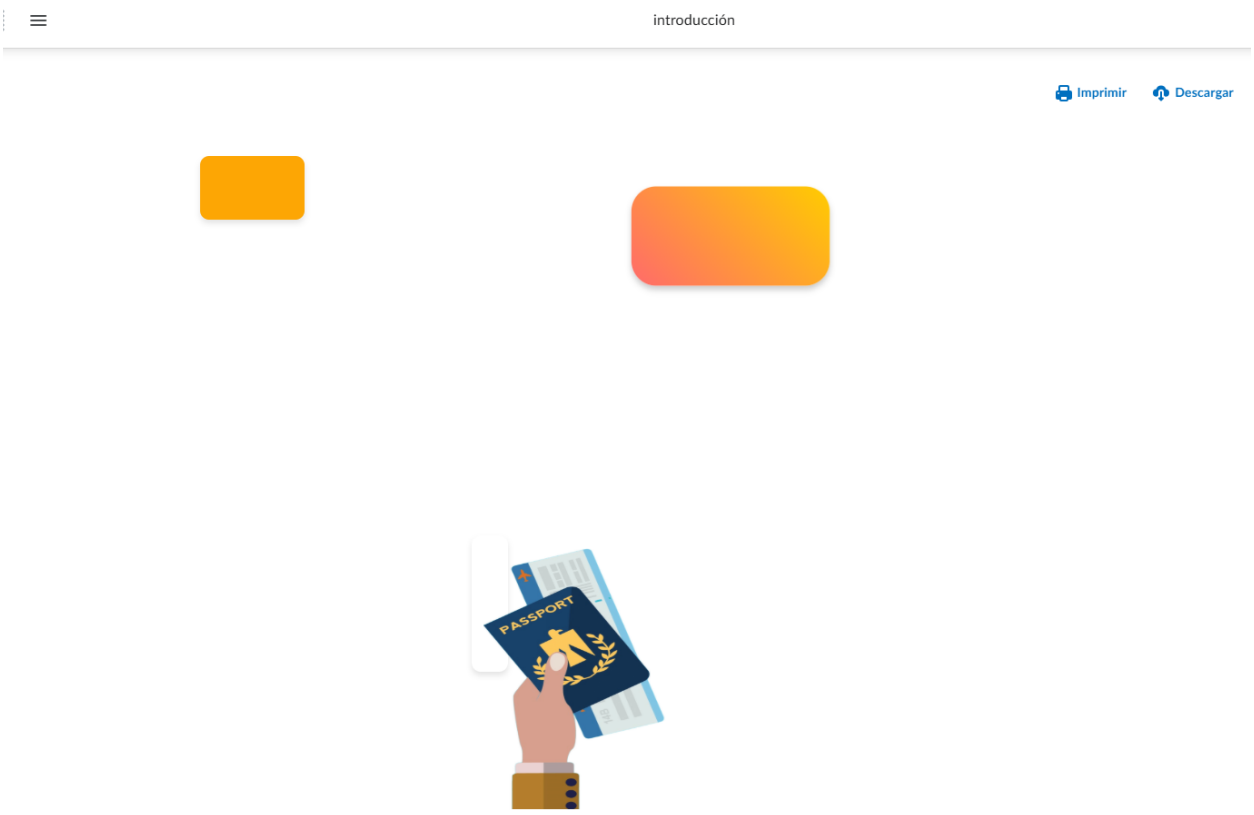


Figura 3.3: Prueba 2 del proyecto en Brightspace

3.1.3. Decisión del aplicativo | 06/03/25 - 10/04/25

Dados los resultados obtenidos y con una sensación de incertidumbre, se decidió reunirse nuevamente con el equipo de trabajo y con el CSI para tomar una decisión definitiva. Aunque, para el Centro Magis la idea de utilizar el LMS Brightspace como ecosistema para albergar el curso era apropiada, en la práctica no lograba resolver las necesidades específicas del departamento; por esta razón, y contando con todos los permisos necesarios, se acordó finalmente desarrollar un aplicativo web desde cero. Este incluiría la implementación completa del backend, el frontend, la arquitectura del sistema y la gestión del almacenamiento de información, con el objetivo de asegurar un funcionamiento óptimo y alineado con los requerimientos del curso.

Con una idea más clara de lo que se quería hacer, se empezó a construir el proyecto por módulos, procurando tenerlos diferenciados y desarrollarlos uno a uno con el mayor nivel de detalle posible. Partiendo de esto, se inició primero con el módulo llamado 'Home'. En este módulo se buscaba dejar una imagen de fondo de un aeropuerto para tratar de dar la sensación de viaje, un botón funcional

que desplegara un pasaporte y en este se mostrara la información resumida del estudiante que tenía la sesión abierta, un botón de introducción que debía redireccionar a un módulo de introducción el cual contextualizaría a los estudiantes de cómo moverse a través de todo el aplicativo y a su vez, explicarle todo con respecto a cómo funcionaría toda la lógica del curso y por último, un botón de inicio de viaje, el cual te redirecciona a un mapamundi donde se encuentra la ruta guiada que mostraría cada una de las actividades. El resultado del primer módulo 'Home' se puede ver en la figura 3.4.

Figura 3.4: Módulo Home aplicativo

El siguiente módulo en ser desarrollado fue el Módulo de Introducción. Este tenía como propósito servir como un apartado inicial obligatorio que los estudiantes debían revisar antes de acceder a las actividades del curso. Su función principal era contextualizar al estudiante, brindándole una visión general de los procesos, herramientas y funcionalidades que estarían disponibles a lo largo de su recorrido, con el fin de que se sintieran más familiarizados con el entorno del aplicativo. Este módulo estaba compuesto por un carrusel de información, que podía ser navegado mediante botones para recorrer el contenido paso a paso. Además, se incluyó una barra de navegación (navbar) que permitía moverse fácilmente entre las distintas secciones presentadas. El Módulo de Introducción quedó estructurado como se ve en la figura 3.5.

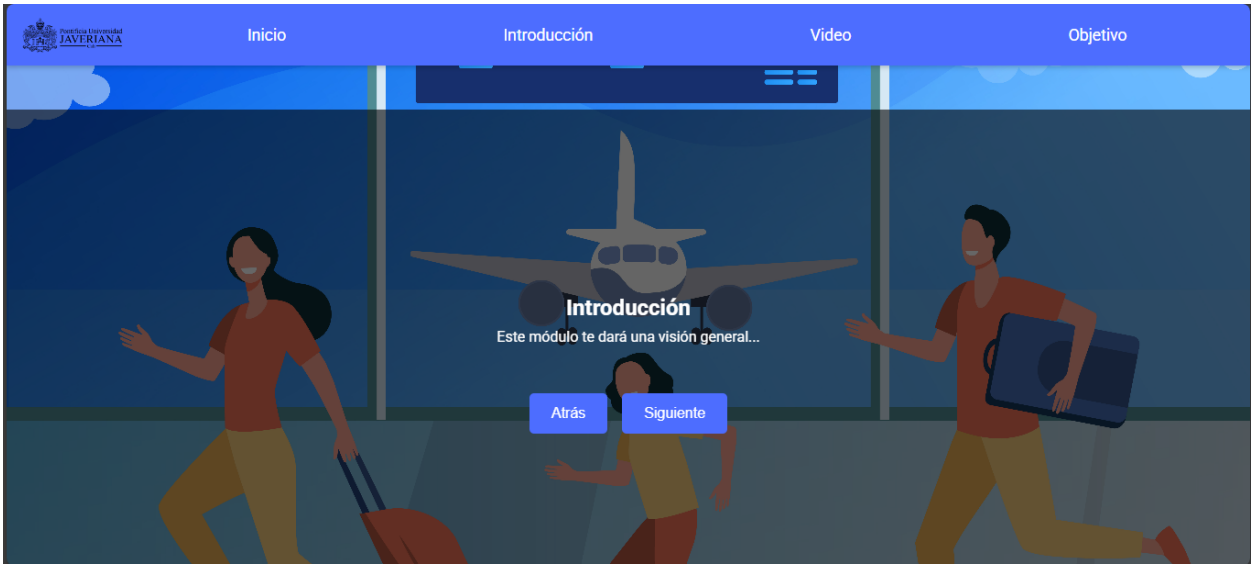


Figura 3.5: Módulo de Introducción aplicativo

3.1.4. Desarrollo de primeras actividades | 06/03/25 - 10/04/25

Una de las problemáticas que se presentaba al inicio del proyecto era que las actividades que se tenían eran demasiado estáticas, llenas de texto y que dada la cantidad de actividades que se tenían, representaba bastante tiempo la valoración individual y apropiada para cada una de ellas. Es por esto que el apartado de las actividades representaba un punto bastante importante en el desarrollo del proyecto y se estuvo estructurando cuáles serían las apropiadas para solventar la problemática presentada.

Afortunadamente, durante el tiempo en que se intentaba montar el curso en Brightspace, desde el Centro Magis se comentó sobre una plataforma web llamada Educaplay, la cual ofrecía actividades dinámicas y permitía una calificación automática basada en los resultados obtenidos por cada estudiante. Estas actividades podían integrarse directamente en Brightspace [21].
Educaplay contaba con un catálogo bastante amplio de actividades lúdicas e interactivas, lo cual pareció una solución interesante al grupo de trabajo. Se decidió comentar la alternativa con el equipo de trabajo, con la intención de utilizarla como alojador de las actividades y medio para obtener los resultados de los estudiantes. Esta opción representaba una gran ventaja, ya que evitaba tener que desarrollar las actividades desde cero, diseñar una arquitectura específica y resolver la gestión del almacenamiento de datos. Además, la universidad contaba con los recursos necesarios para adquirir la plataforma. Sin embargo, aunque Educaplay funcionaba muy bien para LMS como Brightspace, Moodle, entre otros, se descubrieron inconvenientes importantes: al comunicar la solicitud con el equipo de desarrolladores, informaron que no contaban con una API que permitiera alojar y guardar

el progreso de los estudiantes dentro de plataformas externas. Esto significaba que no era posible integrarla adecuadamente con un aplicativo web completamente independiente como el que se estaba construyendo. Por esta razón, la opción de usar Educaplay fue finalmente descartada.

Tomando como inspiración algunas de las actividades disponibles en la plataforma Educaplay, se seleccionaron dos que servían como base para el diseño de nuestras propias versiones. La primera fue una sopa de letras, ya que cumplía con los criterios clave: era dinámica, permitía la interacción del estudiante y podía ser calificada automáticamente al finalizar. Esta actividad fue construida como una cuadrícula de letras generada aleatoriamente, en la que el estudiante debía encontrar un conjunto de palabras ocultas. Además, se incluyeron elementos que añadían un componente de reto, tales como un contador de tiempo, un sistema de puntuación y un número limitado de vidas. Estos factores no solo hacían la experiencia más lúdica, sino que también incentivaban la competencia y el rendimiento. La primera actividad interactiva quedó estructurada como se ve en la figura 3.6.

Figura 3.6: Primera actividad interactiva HTML

La segunda actividad desarrollada fue un crucigrama, ya que cumplía con criterios similares a los de la sopa de letras: era interactiva, permitía la autoevaluación y fue bien recibida por el departamento. Una vez aprobada, se dio inicio a su construcción. Para esta actividad se utilizó una cuadrícula similar a la anterior, pero adaptada a las características propias de un crucigrama. Se definieron claramente los espacios vacíos, en los que no se podía escribir, y los espacios habilitados para que los estudiantes completaran las palabras correctas. Además, se incorporó un botón de pistas que facilitaba el avance en caso de dificultad, y un panel de estadísticas que mostraba el tiempo, el número de vidas disponibles y el puntaje acumulado. La segunda actividad interactiva quedó estructurada como se ve en la figura 3.7.

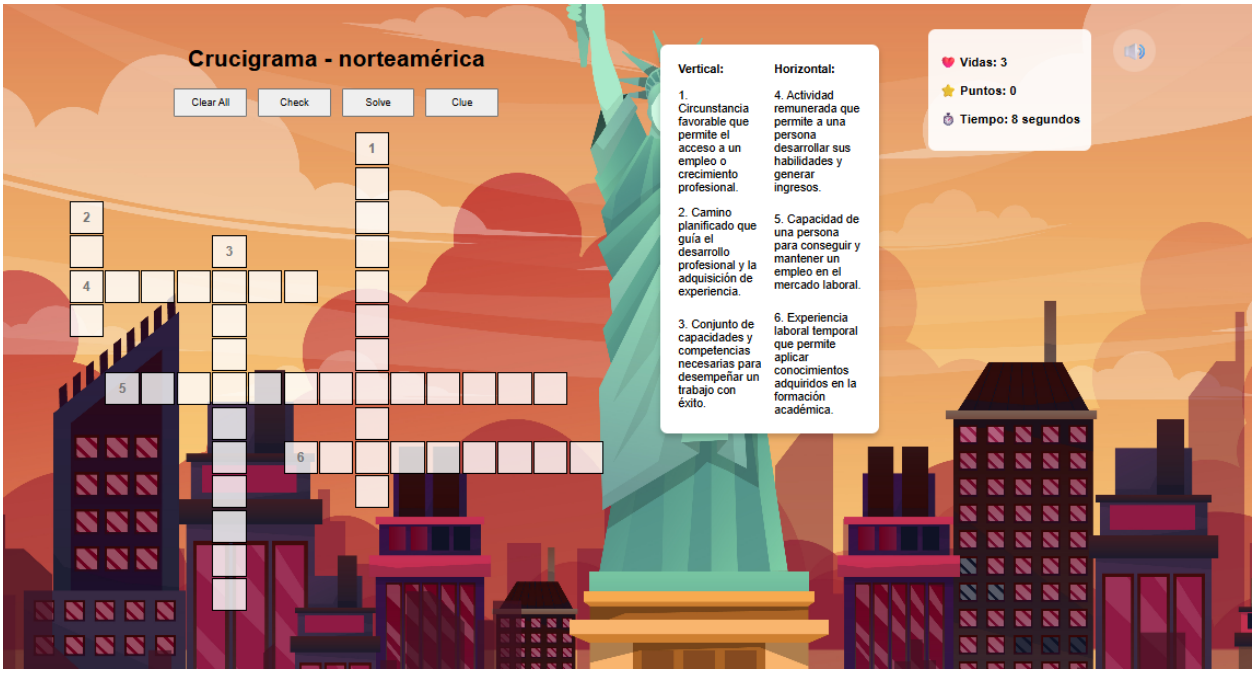


Figura 3.7: Segunda actividad interactiva HTML

De esta forma quedaba establecida la base de diseño para el proyecto en HTML; el nuevo enfoque era pulir diseños, tomar ideas de arreglos y apuntar a hacer la parte del backend que permitiría la funcionalidad de todos estos diseños.

3.1.5. Cambio en el diseño por uso de nuevas tecnologías | 10/04/25 - 10/05/25

Durante esta etapa, la base de diseño en HTML fue bien recibida por el equipo de trabajo, el CSI y las personas encargadas del área administrativa de la oficina. Sin embargo, conforme avanzaba en el desarrollo de detalles visuales, se percató de que lograr una responsividad adecuada se volvía una necesidad importante. Adaptar los diseños a múltiples dispositivos requería implementar estrategias como media queries, que permitieran escalar y ajustar la resolución de los elementos gráficos. En muchos casos, también se recurría a soluciones poco escalables o de 'fuerza bruta' para lograr que los componentes encajaran visualmente de forma correcta [22].

Por esta razón, se decidió presentar una nueva propuesta centrada en el uso de React, una librería de JavaScript y orientada a la construcción de interfaces de usuario por módulos. React ofrecía una gestión más robusta de los elementos visuales y facilitaba la reutilización de componentes. Además, se propuso integrar Tailwind CSS, una librería de utilidades que permitía trabajar con estilos prediseñados, como barras de navegación, botones, tarjetas, etc., lo cual aceleraría el proceso de maquetación visual y mantendría la coherencia estética del aplicativo.

Una de las principales ventajas de este nuevo enfoque era la posibilidad de construir cada parte del sistema como un módulo independiente. Esto permitiría asignar tiempos específicos de desarrollo a cada sección, enfocarse en los detalles particulares y mejorar el mantenimiento general del código. Por ejemplo, en el caso de las actividades, se diseñó una base reutilizable para la sopa de letras, que serviría como plantilla para implementar cinco de las diez actividades del curso. Gracias a esto, no era necesario desarrollar cada actividad desde cero, sólo era necesario modificar el contenido base y adaptarlo a cada caso, optimizando así tiempo y esfuerzo en el desarrollo.

De esta forma se presentaría el primer módulo de inicio construido con Javascript, React y Tailwind para los diseños, teniendo como base el anterior diseño que se tenía en HTML, y logrando mantener la interactividad, los buenos efectos y detalles pulidos. Este módulo de inicio también contaba con una barra de navegación las cuales llevaban a módulos como el de introducción y el de progreso. El de introducción era muy parecido al que se tenía anteriormente, pero manejando un poco el mismo diseño de este módulo de inicio y el de progreso; es una sección en la que se puede ver la información completa del estudiante. El resultado se puede apreciar en la figura 3.8.

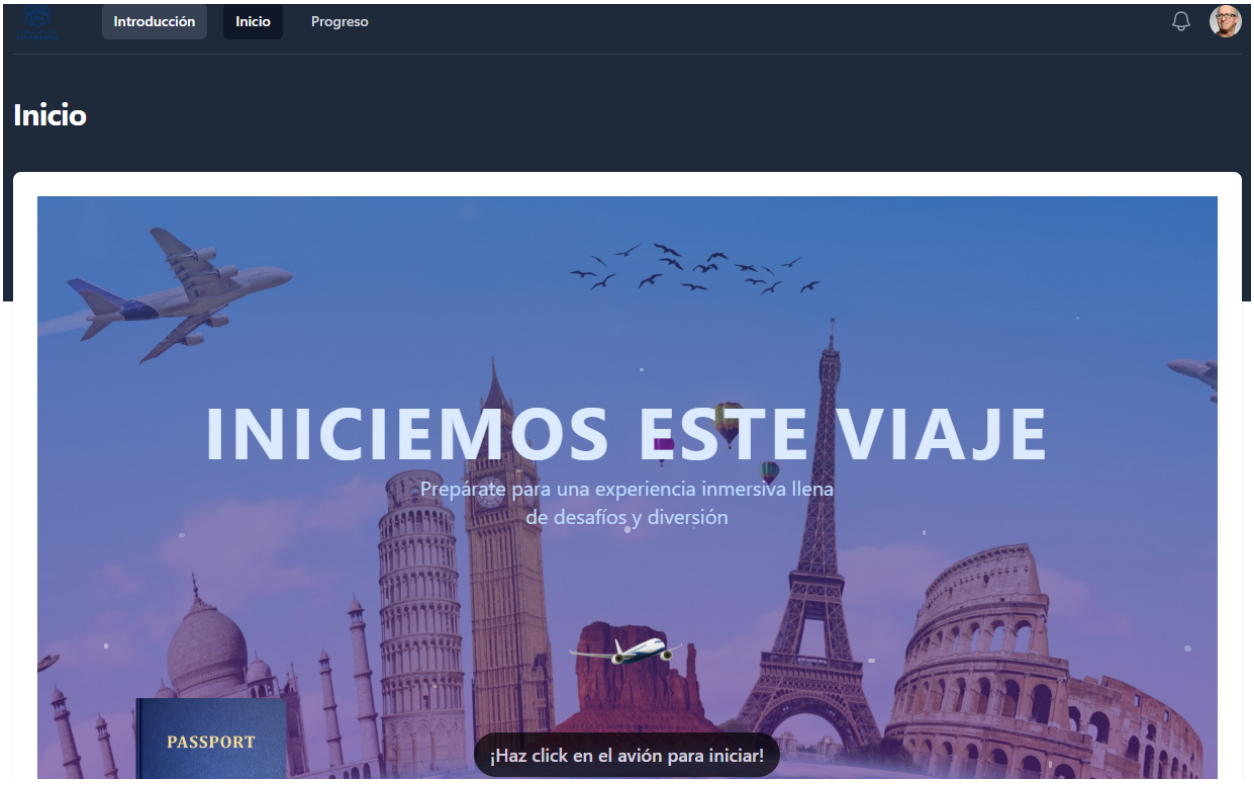


Figura 3.8: Módulo de inicio en React

El siguiente módulo en desarrollarse fue uno de los más importantes del proyecto: el Módulo de Mapa. Este componente tenía como objetivo principal mostrar visualmente toda la ruta de actividades que los estudiantes debían completar a lo largo de su recorrido formativo dentro del aplicativo. Fue uno de los módulos más complejos de implementar, ya que requería la incorporación de animaciones personalizadas que permitieran guiar al estudiante a través de la ruta de manera visual e intuitiva. La idea central era que el mapa mostrara un camino progresivo, iluminado con colores que marcaran el trayecto ya recorrido y destacaran la siguiente actividad a realizar. Estas animaciones debían actualizarse a medida que el estudiante completaba cada actividad. Además, como se puede apreciar en la figura 3.9, aparecían unos círculos que, como tal no eran las actividades, si no, los destinos donde los estudiantes llegarían y dentro de cada uno de ellos se encontrarían las actividades. Así quedó estructurado el Módulo de Mapa:

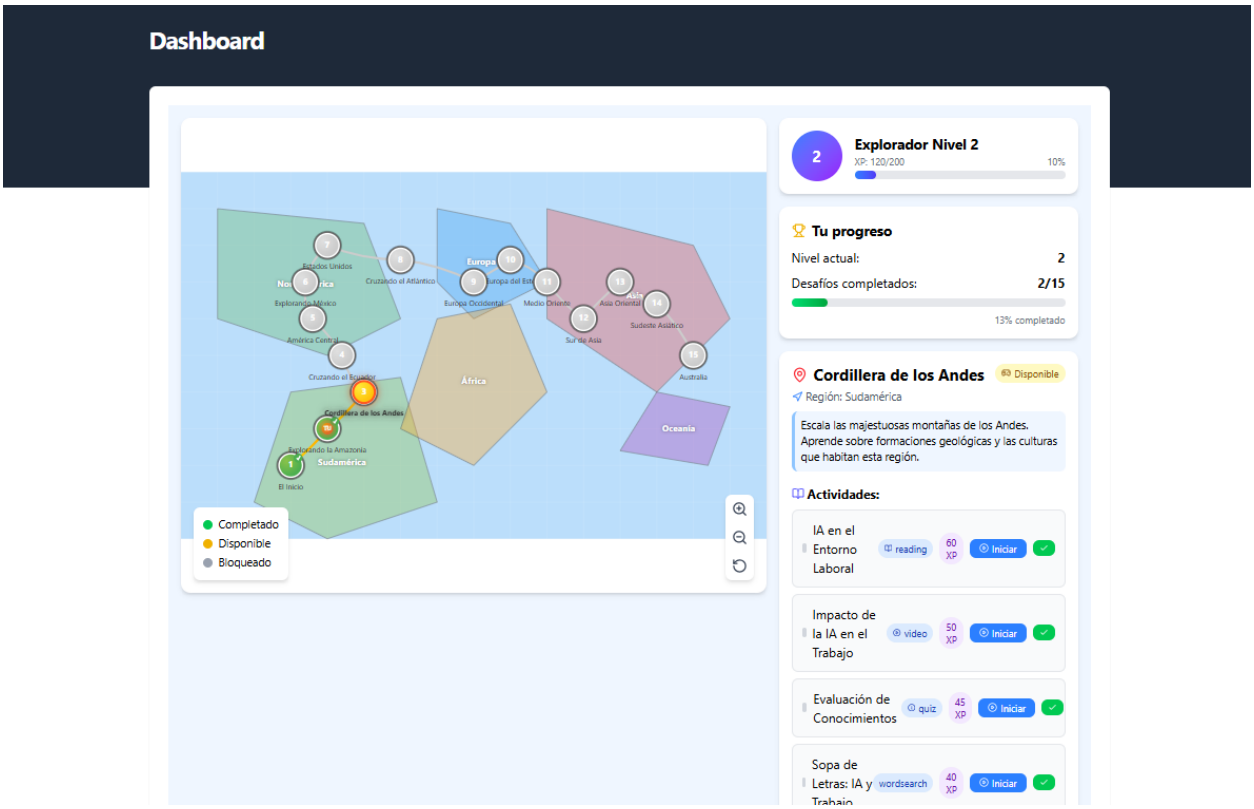


Figura 3.9: Módulo de Mapa en React

Por último, se abordó el desarrollo del Módulo de Actividades, las cuales, si bien conservarían la estructura de algunas diseñadas en HTML, debían adaptarse a los nuevos estilos y componentes definidos en la interfaz desarrollada con React y Tailwind. Además, se incorporaron nuevos tipos de actividades más sencillas, pensadas para complementar y completar los diferentes destinos dentro del recorrido del estudiante. Estas actividades incluían:

- Videos interactivos, que consistían en contenidos de YouTube que los estudiantes debían visualizar por una cantidad de segundos para que se consideraran aprobadas.
- Lecturas dinámicas, presentadas mediante un carrusel de información que los estudiantes debían deslizar completamente para acceder a todo el contenido.
- Exámenes tipo quiz, resueltos directamente en el sistema mediante selección de respuestas.
- Y, finalmente, las actividades anteriormente desarrolladas: la sopa de letras y el crucigrama, ya adaptadas al nuevo entorno visual e integradas funcionalmente al sistema.

Algunas de las actividades interactivas del aplicativo quedaron como se ve en las figuras 3.10 y 3.11.

Figura 3.10: Actividad sopa de letras en React

Figura 3.11: Actividad crucigrama en React

3.2. El uso de Screaming architecture para el diseño en el frontend

La decisión de adoptar Screaming Architecture se fundamentó en dos aspectos clave: primero, la necesidad de mantener consistencia arquitectónica entre frontend y backend, ya que Martin aborda ambos conceptos en su obra como parte de una estrategia integral de diseño de software. Segundo, la necesidad de crear un código que comunicara claramente su propósito desde el primer vistazo, especialmente importante en un contexto donde múltiples tipos de usuarios (desarrolladores, administradores académicos, estudiantes) necesitan interactuar con el sistema.

Una de las características de esta arquitectura es que su dominio no revela qué tipo de aplicación es, ya que utiliza convenciones genéricas como componentes, vistas, servicios y utilidades. Esta organización se basa en responsabilidades técnicas genéricas que podrían aplicarse a cualquier aplicación React, sin reflejar las especificaciones del dominio o las necesidades específicas del curso

de empleabilidad. Se organiza por responsabilidades técnicas (API, components, views) como se puede observar en la Figura 3.12, donde cada carpeta contiene elementos relacionados por su función técnica en lugar de su función en el negocio. Los nombres de las carpetas son términos estándar del ecosistema React que no comunican el propósito específico de la aplicación.

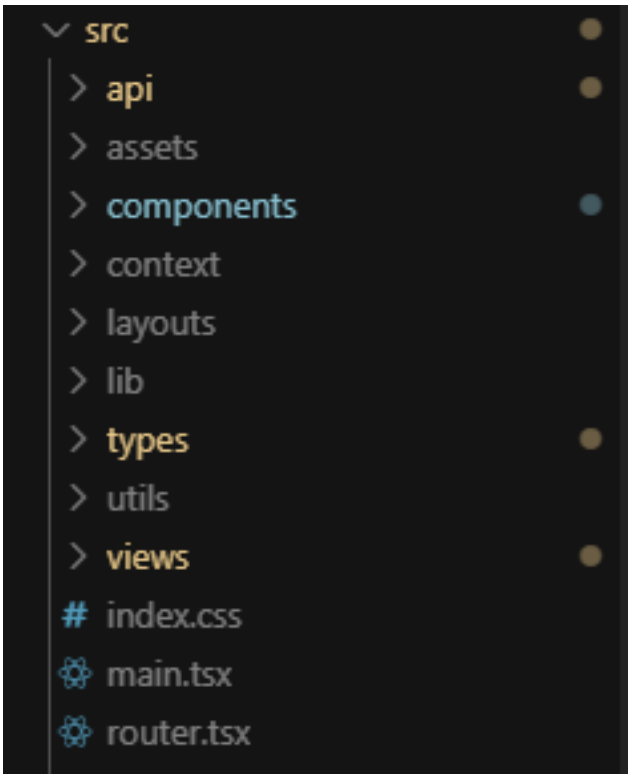


Figura 3.12: Primera fase de estructuración de carpetas

Esta evolución representa un paso significativo hacia la implementación de Screaming Architecture, donde la organización del código comienza a reflejar las necesidades específicas del negocio educativo en lugar de seguir únicamente patrones genéricos. Se identifica que hay dos tipos principales de usuarios: administradores y estudiantes, cada uno con necesidades, permisos y funcionalidades diferentes. Esta diferencia del dominio se refleja directamente en la estructura de carpetas y la organización de componentes como se ve en la Figura 3.13, permitiendo que cualquier desarrollador que examine el código entienda inmediatamente que se trata de un sistema con roles diferenciados. Se revelan los conceptos específicos del negocio que son fundamentales para entender el sistema: la gestión, que incluye la administración de estudiantes, el seguimiento de progreso y la generación de reportes académicos; la ruta de aprendizaje, que representa el viaje educativo estructurado que los estudiantes deben completar; y la gamificación, que incorpora elementos lúdicos y interactivos para motivar el aprendizaje. Se puede entender el flujo de usuario solo mirando la estructura, ya que la organización de carpetas y archivos como se ven en las Figuras 3.14 y 3.15 refleja directamente el recorrido que un estudiante o administrador seguiría al utilizar la aplicación.

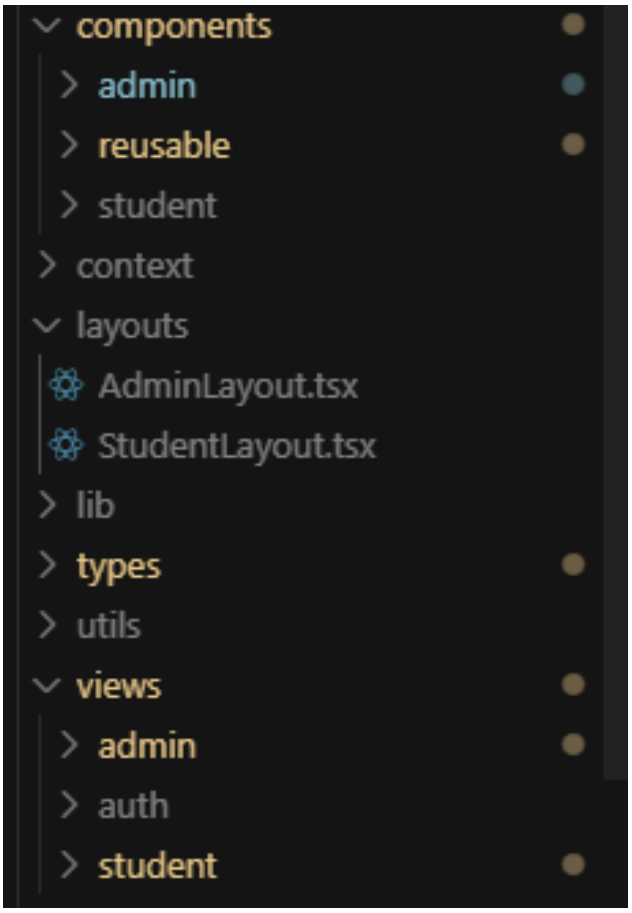


Figura 3.13: Segunda fase de estructuración de carpetas

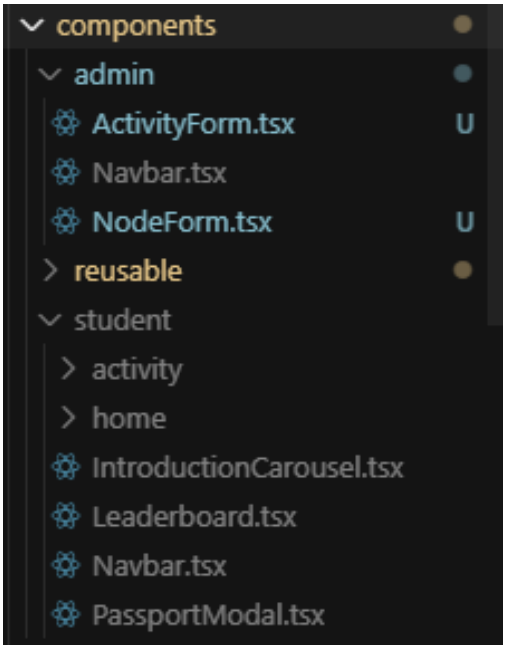


Figura 3.14: Tercera .1 fase de estructuración de carpetas

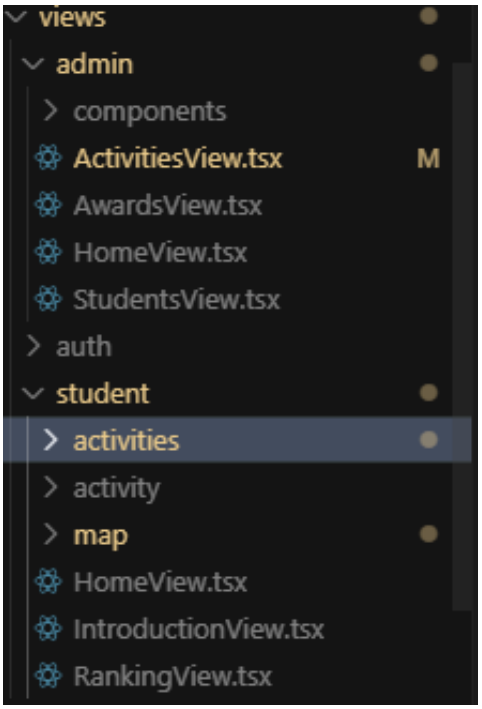


Figura 3.15: Tercera .2 fase de estructuración de carpetas

3.3. El uso de Clean Architecture para el backend

La aplicación implementa la Clean Architecture organizando el código en capas centradas con responsabilidades específicas y dependencias bien definidas. La decisión de implementar Clean Architecture se fundamentó en la necesidad de crear un sistema que evolucione con el negocio y no con las tecnologías. Esta arquitectura permite que las reglas de negocio fundamentales del curso de empleabilidad permanezcan estables independientemente de los cambios en frameworks, bases de datos o interfaces de usuario [24].

3.3.1. Principios fundamentales implementados:

Dependencias hacia el interior: Todas las dependencias fluyen hacia las capas internas del dominio. Los routers de FastAPI dependen de los casos de uso, los casos de uso dependen de las interfaces del dominio, y las implementaciones de infraestructura dependen de las definiciones del dominio. FastAPI fue seleccionado como framework principal del backend ya que la validación automática de tipos mediante Pydantic reduce los errores en tiempo de ejecución y mejora la robustez del sistema. Además, es compatible con la arquitectura limpia al permitir la inyección de dependencias y la separación clara entre la capa de entrada y la lógica de negocio, facilitando la implementación de los principios de Clean Architecture.

Funciones específicas por capa: La capa de entrada maneja comunicación HTTP y serialización, la capa de aplicación contiene lógica de casos de uso, la capa de dominio define entidades y reglas de negocio, y la capa de infraestructura implementa acceso a datos y servicios externos.

Aislamiento de capas internas: Las entidades del dominio (User, Activity, Node, UserActivityProgress, UserNodeProgress) no conocen tecnologías específicas y contienen únicamente lógica de negocio.

Independencia de tecnologías: Los casos de uso (ManageAuthenticationUseCase, ManageActivitiesUseCase, ManageNodesUseCase, ManageMapProgressUseCase) no dependen de tecnologías específicas, permitiendo cambiar implementaciones sin afectar la lógica de negocio.

3.3.2. Estructura específica del proyecto:

La aplicación estructura las capas con routers de FastAPI y DTOs en la entrada, casos de uso y ApplicationContainer en la aplicación, entidades del modelo educativo e interfaces de puertos en el dominio, y repositorios, servicios de autenticación y mappers en la infraestructura. Esta arquitectura permite cambios en la interfaz o base de datos sin afectar la lógica de negocio, facilita pruebas unitarias mediante mocking de dependencias, y permite evolución tecnológica sin impactar las reglas de negocio fundamentales del sistema. En el backend desarrollado con Python y FastAPI, se implementa la convención snake_case para nombrar variables, funciones, métodos y atributos de clases, siguiendo las guías de estilo PEP 8 de Python.

3.3.3. Capa de dominio

La capa de dominio constituye la capa más interna de la aplicación, conteniendo la lógica de negocio pura y representando el corazón del sistema. Esta capa es completamente independiente de cualquier tecnología externa.

Modelos de dominio: El proyecto implementa entidades de negocio (User, Activity, Node, UserActivityProgress, UserNodeProgress) que representan los conceptos fundamentales del sistema, encapsulando las reglas y comportamientos específicos de su dominio.

Características de la capa de dominio:

- **Uso de @dataclass:** Las entidades utilizan el decorador @dataclass de Python para representar datos de manera clara y concisa, generando automáticamente métodos como init, repry eq. Se puede ver reflejada en la Figura 3.16

- **Independencia de frameworks:** Los modelos no importan ni dependen de frameworks externos como FastAPI o SQLAlchemy, garantizando que las reglas de negocio permanezcan puras.

- **Lógica esencial de negocio:** Cada entidad contiene la lógica fundamental del dominio educativo, incluyendo validaciones de tipos de contenido, reglas de autenticación y roles, y cálculos de progreso y puntajes.

- **Aislamiento de infraestructura:** Los modelos no conocen la existencia de bases de datos o APIs REST, no contienen anotaciones de mapeo y no manejan aspectos de persistencia o comunicación.

- **Interfaces de puertos:** La capa define interfaces abstractas (UserPort, ActivityPort, NodePort) que establecen contratos para la comunicación con capas externas sin crear dependencias directas.

Esta implementación garantiza que el núcleo del sistema sea robusto, mantenible y evolutivo, permitiendo que las reglas de negocio fundamentales permanezcan estables independientemente de los cambios tecnológicos en las capas externas.

```python
@dataclass
class Activity:
    id: UUID
    node_id: UUID
    title: str
    description: Optional[str]
    content_type: str
    order_index: int
    is_required: bool
    estimated_duration_minutes: Optional[int]
    parameters: Dict[str, Any]
    created_at: datetime
    updated_at: datetime
    is_active: bool
    node: Optional[Node] = None
```

Figura 3.16: Dataclass. Capa de dominio.

Puertos (Gateways): Representan las interfaces abstractas.
Características de los puertos:

- **Clases abstractas que definen qué hacer, no cómo:** El proyecto implementa interfaces abstractas como UserPort, ActivityPort y NodePort que definen únicamente los métodos requeridos por la lógica de negocio sin especificar su implementación concreta un ejemplo de estas clases se puede ver en la Figura 3.17.
- **Contratos claros que especifican métodos y tipos de retorno:** Las interfaces definen métodos específicos con tipos de retorno claros, como find_user_by_username() que retorna un objeto User, get_activities() que retorna una lista de Activity, y get_nodes() que retorna una lista de Node.
- **No sabe de base de datos o APIs:** Las interfaces están completamente aisladas de tecnologías específicas, sin referencias a PostgreSQL, SQLAlchemy o FastAPI, permitiendo que la lógica de negocio defina sus necesidades sin crear dependencias hacia tecnologías externas.
- **Implementación en el proyecto:** El sistema implementa estas interfaces en la capa de dominio, donde UserPort define contratos para operaciones de usuarios, ActivityPort establece contratos para gestión de actividades, y NodePort define contratos para operaciones de nodos de aprendizaje.

```python
class ActivityPort(ABC):
    @abstractmethod
    async def get_activities(self, user_id: Optional[int] = None) -> List[
        pass

    @abstractmethod
    async def get_activity_by_id(self, activity_id: str) -> Activity:
        pass

    @abstractmethod
    async def create_activity(self, activity: Activity) -> Activity:
        pass

    @abstractmethod
    async def update_activity(self, activity: Activity) -> Activity:
        pass

    @abstractmethod
    async def delete_activity(self, activity_id: UUID) -> None:
        pass
```

Figura 3.17: Clase abstracta. Capa de dominio.

Características de los casos de uso:

- **La lógica de la aplicación orquesta operaciones complejas:** El proyecto implementa casos de uso específicos como ManageAuthenticationUseCase el cual coordina el proceso completo de autenticación, ManageActivitiesUseCase que gestiona operaciones de actividades educativas, ManageNodesUseCase que maneja nodos de aprendizaje, y ManageMapProgressUseCase que orquesta la obtención del progreso completo del mapa educativo. Un ejemplo de estos se puede ver en la Figura 3.18.

- **Al inyectar dependencias, se reciben puertos como parámetros:** Los casos de uso implementan inyección de dependencias recibiendo instancias de puertos como parámetros, dependiendo de abstracciones en lugar de implementaciones concretas.

- **Se implementa la lógica específica de la aplicación:** Cada caso de uso implementa lógica específica del dominio, incluyendo procesamiento de tokens encriptados, validación de credenciales LDAP, generación de tokens JWT, filtrado de actividades según contexto de autenticación, cálculo de estados de completado y generación de métricas de progreso.

```python
class ManageActivitiesUseCase:
    def __init__(self, activity_port: ActivityPort):
        self.activity_port = activity_port

    async def get_activities(self, user_id: Optional[int] = None) -> List[Activity]:
        print(f"DEBUG: ManageActivitiesUseCase.get_activities() llamado con user_id: {user_id}")
        activities = await self.activity_port.get_activities(user_id)
        print(f"DEBUG: ManageActivitiesUseCase.get_activities() devolviendo {len(activities)} actividades")
        return activities
```

Figura 3.18: Casos de uso. Capa de dominio.

3.3.4. Capa de aplicación

La capa de aplicación configura y conecta todas las capas del sistema utilizando inyección de dependencias, estableciendo el mecanismo de orquestación que permite la comunicación entre los diferentes componentes de la arquitectura limpia.

Componentes usados en el proyecto:

- **Contenedor de dependencias:** El proyecto implementa el ApplicationContainer como el componente central que configura cómo se conectan todas las piezas del sistema, utilizando la biblioteca dependency-injector para gestionar la inyección de dependencias. En la Figura 3.19 se puede ver el uso del componente 'ApplicationContainer'.

- **Configuración centralizada donde se define todas las dependencias:** El Application-Container centraliza la configuración de todas las dependencias, definiendo implementaciones de puertos del dominio, casos de uso y servicios de infraestructura en un único punto de configuración.

- **El framework inyecta todas las dependencias:** El sistema utiliza inyección automática para proporcionar las implementaciones correctas a cada componente, eliminando la necesidad de crear manualmente instancias y gestionar dependencias.

- **Buen control y flexibilidad al cambiar implementaciones:** La configuración centralizada permite cambiar fácilmente implementaciones sin afectar el resto del sistema, facilitando la evolución y adaptación a diferentes entornos.

```python
class ApplicationContainer(containers.DeclarativeContainer):
    config = providers.Configuration()

    authentication_token = providers.Configuration()

    jwt_settings = providers.Singleton(
        JWTSettings
    )

    user_repository = providers.Singleton(
        UserRepository
    )

    authentication_client = providers.Factory(
        AuthenticationClient,
        token=authentication_token
    )

    jwt_token_service = providers.Singleton(
        JwtTokenService,
        jwt_settings=jwt_settings
    )

    user_adapter = providers.Factory(
        UserAdapter,
        user_repository=user_repository
    )
```

Figura 3.19: Contenedor de aplicaciones. Capa de aplicación.

3.3.5. Capa de infraestructura

La capa de infraestructura implementa las interfaces definidas en el dominio, constituyendo la capa más externa que interactúa con el mundo real, incluyendo bases de datos, servicios externos y sistemas de autenticación.

Componentes implementados en el proyecto:

- **Adaptador PostgreSQL:** El proyecto implementa adaptadores específicos para PostgreSQL que materializan los puertos del dominio, incluyendo UserRepository, ActivityRepository y NodeRepository como las implementaciones principales de los puertos del dominio.

- **Convierte los llamados del dominio en operaciones de base de datos:** Los repositorios traducen las llamadas de los casos de uso en operaciones SQL específicas de PostgreSQL, manejando consultas complejas con JOINs para obtener actividades, validación de credenciales y operaciones CRUD para nodos.
- **Convierte entre entidades de la base de datos y modelos de dominio:** El proyecto implementa mappers especializados que transforman entre entidades de base de datos y modelos de dominio, manejando conversión de tipos de datos, mapeo de relaciones y transformación de estructuras como content_type y parameters en formato JSON.
- **Maneja los errores traduciéndolos desde la base de datos hasta el dominio:** Los adaptadores implementan manejo robusto de errores que traduce excepciones específicas de PostgreSQL en errores comprensibles para el dominio, capturando violaciones de restricciones y problemas de integridad referencial.
- **Servicios adicionales de infraestructura:** El proyecto implementa JwtTokenService para tokens JWT y servicios de autenticación LDAP, complementando los adaptadores de base de datos mientras mantienen la separación de responsabilidades de la arquitectura limpia.

El uso de un adaptador de actividades usado en el proyecto puede verse reflejado en la Figura 3.20.

```python
class ActivityAdapter(ActivityPort):
    def __init__(self, activity_repository: ActivityRepository):
        self.activity_repository = activity_repository

    async def get_activities(self, user_id: Optional[int] = None) -> List[Activity]:
        print(f"DEBUG: ActivityAdapter.get_activities() llamado con user_id: {user_id}")
        activities_entities = await self.activity_repository.get_activities(user_id)
        print(f"DEBUG: ActivityAdapter.get_activities() - entidades obtenidas: {len(activities_entities)}")
        activities = [ActivityMapper.entity_to_domain(entity) for entity in activities_entities]
        print(f"DEBUG: ActivityAdapter.get_activities() - actividades mapeadas: {len(activities)}")
        return activities
```

Figura 3.20: Adaptador de actividades. Capa de infraestructura.

Entidades de la base de datos: Las entidades de base de datos constituyen la representación ORM de las tablas físicas en PostgreSQL, implementando el mapeo objeto-relacional que permite la interacción entre el código y la base de datos relacional. La estructura del componente 'ActivityEntity' puede verse en la figura 3.21.

Características de implementación:

- **El mapeo ORM representa las tablas de la BD:** El proyecto implementa entidades ORM utilizando SQLAlchemy que representan directamente las tablas físicas. UserEntity mapea la

tabla 'users', ActivityEntity representa la tabla 'activities', y NodeEntity mapea la tabla 'nodes', utilizando la clase Base de SQLAlchemy y definiendo __tablename__ para especificar nombres exactos de tablas.

- **Las relaciones definen los joins y llaves foráneas:** El proyecto implementa relaciones entre entidades utilizando SQLAlchemy relationship(). SQLAlchemy fue elegido a diferencia de otros ORM, porque ofrece un control granular sobre las consultas y la gestión de bases de datos que es esencial para un sistema con lógica de negocio compleja como el curso de empleabilidad. UserEntity establece relaciones uno a machos con entidades de progreso, ActivityEntity mantiene relación muchos a uno con NodeEntity, y NodeEntity establece relaciones con entidades de progreso, definiendo automáticamente los JOINs necesarios y garantizando integridad referencial.
- **Usa el ORM para persistencia de datos:** Las entidades implementan persistencia utilizando session.query(), session.add(), session.commit() y session.delete().

```python
class ActivityEntity(Base):
    __tablename__ = 'activities'

    id = Column(UUID, primary_key=True, server_default=func.gen_random_uuid())
    node_id = Column(UUID, ForeignKey('nodes.id', ondelete='CASCADE'), nullable=False)
    title = Column(String(255), nullable=False)
    description = Column(Text)
    content_type = Column(String(50), nullable=False)
    order_index = Column(Integer, nullable=False, default=0)
    is_required = Column(Boolean, default=True)
    estimated_duration_minutes = Column(Integer)
    parameters = Column(JSONB, nullable=False, server_default='{}')
    created_at = Column(TIMESTAMP(timezone=True), server_default=func.now())
    updated_at = Column(TIMESTAMP(timezone=True), server_default=func.now())
    is_active = Column(Boolean, default=True)

    # Relationships
    node = relationship("NodeEntity", back_populates="activities")
    user_progress = relationship("UserActivityProgressEntity", back_populates="activity", cascade="all,
```

Figura 3.21: Entidad de actividades. Capa de infraestructura.

Mappers: Los mappers constituyen componentes especializados que facilitan la transformación de datos entre las entidades de base de datos y los modelos de dominio, asegurando la correcta conversión de estructuras y tipos de datos entre las diferentes capas de la arquitectura.

Características de implementación:

- **Conversión bidireccional entre las entidades y el dominio:** El proyecto implementa mappers que realizan conversión bidireccional entre entidades ORM y modelos de dominio. ActivityMapper convierte ActivityEntity a Activity manejando content_type, parameters y fechas, NodeMapper realiza conversiones para NodeEntity, y UserMapper convierte UserEntity, transformando roles y estados de activación.

- **Carga objetos relacionados:** Los mappers implementan la carga de objetos relacionados para mantener la consistencia de las relaciones del dominio. ActivityMapper carga automáticamente la entidad Node relacionada, NodeMapper carga actividades relacionadas, y UserMapper puede cargar progreso de actividades y nodos, optimizando el rendimiento y manteniendo los datos correctamente.

- **Transforma los datos para adaptarse entre capas:** Los mappers implementan transformaciones específicas para adaptar datos entre convenciones de base de datos y dominio. ActivityMapper (el cual se puede ver su estructura en la figura 3.22), transforma is_active en estado de completado calculado, NodeMapper convierte información de ubicación geográfica, y UserMapper adapta roles y permisos, asegurando que cada capa reciba los datos de forma apropiada.

```python
class ActivityMapper:
    @staticmethod
    def entity_to_domain(entity: ActivityEntity) -> Activity:
        node = None
        try:
            if hasattr(entity, 'node') and entity.node:
                node = Node(
                    id=entity.node.id,
                    title=entity.node.title,
                    description=entity.node.description,
                    created_at=entity.node.created_at,
                    updated_at=entity.node.updated_at,
                    created_by=entity.node.created_by,
                    is_active=entity.node.is_active,
                    location=entity.node.location,
                    region=entity.node.region,
                    order_index=entity.node.order_index,
                )
        except Exception:
            # Si hay error al cargar la relación node, simplem
            node = None

        return Activity(
```

Figura 3.22: Mapeador de actividades. Capa de infraestructura.

Repositories: Los repositorios constituyen la implementación concreta de los puertos del dominio, proporcionando operaciones de acceso a datos que interactúan directamente con la base de datos PostgreSQL mediante SQLAlchemy. En la Figura 3.23 se puede ver la estructura del repositorio para las actividades.

Características de implementación:

- **Operaciones CRUD: Create, Read, Update, Delete:** El proyecto implementa repositorios completos que proporcionan todas las operaciones CRUD necesarias. UserRepository, ActivityRepository y NodeRepository implementan métodos como find_user_by_username(), get_activities(), get_activity_by_id(), create_activity(), update_activity(), delete_activity(), y métodos especializados como get_user_activity_progress() para consultas específicas del dominio educativo.

- **Usan joins y filtros:** Los repositories implementan consultas optimizadas utilizando JOINs entre activities y nodes para obtener información completa en una sola consulta, filtros eficientes por username y email para autenticación, y JOINs complejos entre entidades de progreso para optimizar el rendimiento mediante consultas SQL bien estructuradas.

- **Abren y cierran conexiones correctamente:** Los repositories implementan gestión adecuada de conexiones utilizando el patrón de manejo de sesiones de SQLAlchemy, con manejo de excepciones que garantiza el cierre correcto de conexiones.

- **Para eliminación de datos se marcan como inactivos:** El proyecto implementa soft delete marcando elementos como inactivos mediante is_active = False, preservando integridad histórica de datos, permitiendo recuperación de elementos eliminados, y manteniendo relaciones referenciales intactas con filtros automáticos en consultas.

```python
class ActivityRepository:
    async def get_activities(self, user_id: Optional[int] = None) -> List[ActivityEntity]:
        db_session = next(get_db_session())
        try:
            activity_entities = db_session.query(ActivityEntity) \
                .options(joinedload(ActivityEntity.node)) \
                .filter(ActivityEntity.is_active == True) \
                .order_by(ActivityEntity.node_id, ActivityEntity.order_index) \
                .all()

            print(f"📋 GET /activities - Encontradas {len(activity_entities)} actividades activas")
            for activity in activity_entities:
                print(f"   - {activity.id}: {activity.title} (is_active: {activity.is_active})")

            # Si se proporciona un user_id, agregar información de progreso
            if user_id:
                for activity in activity_entities:
                    progress = db_session.query(UserActivityProgressEntity) \
                        .filter(UserActivityProgressEntity.activity_id == activity.id,
                                UserActivityProgressEntity.user_id == user_id) \
                        .first()

                    if progress:
                        # Agregar el estado de completado como atributo dinámico
                        activity.completed = progress.status == 'completed'
                    else:
                        activity.completed = False
```

Figura 3.23: Repositorio de actividades. Capa de infraestructura.

3.3.6. Capa de puntos de entrada (entrypoints)

La capa de puntos de entrada maneja la comunicación con el mundo exterior, específicamente a través de HTTP, proporcionando la interfaz REST que permite a los clientes interactuar con el sistema educativo mediante endpoints bien definidos.

Componentes implementados en el proyecto:

FastAPI Routers: El proyecto implementa routers específicos de FastAPI organizados por funcionalidad: AuthRouter para autenticación (/auth/), ActivityRouter para actividades educativas (/activities/), NodeRouter para nodos de aprendizaje (/nodes/), y MapRouter para progreso del mapa (/map/). El enrutador para obtener las actividades se puede ver en la Figura 3.24.

Características de implementación:

- **Manejo HTTP donde se procesan requests y responses:** Los routers implementan manejo completo de métodos HTTP (GET, POST, PUT, DELETE) con códigos de estado apropiados (200, 401, 404, 500) y endpoints específicos para cada funcionalidad del sistema educativo.

- **En la inyección de dependencias se reciben casos de uso:** Los endpoints implementan inyección de dependencias mediante @inject, recibiendo automáticamente instancias de casos de uso configurados en el ApplicationContainer, permitiendo que los routers se enfoquen en manejo HTTP mientras delegan lógica de negocio.

- **Usan DTOs para validar datos:** El proyecto implementa DTOs específicos (AuthRequestDto, CreateActivityRequestDto, UpdateActivityRequestDto, ActivityResponseDto) que utilizan Pydantic para validación automática de tipos, restricciones y enumeraciones predefinidas, garantizando que solo datos válidos lleguen a la lógica de negocio.

- **Para autenticar se manejan tokens y autorización:** Los endpoints implementan sistema robusto de autenticación que maneja tokens encriptados, JWT, y tokens externos, con validación mediante Authorization header y manejo de errores HTTP 401, permitiendo acceso tanto autenticado como anónimo según el contexto.

- **Convierten modelos de dominio a DTOs de respuesta:** Los endpoints implementan conversión automática entre modelos de dominio y DTOs de respuesta utilizando métodos estáticos como from_domain(), garantizando respuestas HTTP estructuradas y consistentes mientras mantienen separación entre lógica de negocio y presentación de datos.

```python
@activity_router.get("/")
@inject
async def get_activities(
    authorization: Optional[str] = Header(None),
    activities_use_case: ManageActivitiesUseCase =
        Depends(Provide[ApplicationContainer.manage_activities_use_case]),
    auth_use_case: ManageAuthenticationUseCase =
        Depends(Provide[ApplicationContainer.manage_authentication_use_case]),
    jwt_service: JwtTokenService =
        Depends(Provide[ApplicationContainer.jwt_token_service])
):
    user_id = None

    if authorization and authorization.startswith("Bearer "):
        try:
            token = authorization.split(" ")[1]
            print(f"DEBUG: Token recibido en /activities: {token}")

            # Si el token parece ser un token de autenticación externa (username:password)
            if ':' in token and not token.count('.') == 2:
                print("DEBUG: Token parece ser de autenticación externa en /activities")
                try:
                    username = token.split(":")[0]
                    print(f"DEBUG: Username extraído en /activities: {username}")

                    user_info = await auth_use_case.user_port.find_user_by_username(username)
                    user_id = user_info.id
```

Figura 3.24: Obtener actividades. Capa de puntos de entrada.

DTOs (Objetos de transferencia de información): Los DTOs constituyen objetos especializados que facilitan la transferencia de datos entre las capas de la aplicación, proporcionando validación automática y estructuración consistente de la información que fluye a través de los endpoints de la API REST.

Características de implementación:

- **Valida tipos y restricciones automáticamente:** El proyecto implementa DTOs utilizando Pydantic que proporcionan validación automática de tipos de datos y restricciones en tiempo de ejecución. AuthRequestDto valida que el campo token sea de tipo string, CreateActivityRequestDto implementa validaciones específicas como min_length=1, max_length=255 para título, ge=0 para order_index, ge=1 para estimated_duration_minutes, y validación de UUID para node_id.

- **Valores predefinidos para campos específicos:** Los DTOs implementan valores predefinidos y enumeraciones para campos específicos del dominio. ContentTypeEnum define valores predefinidos para tipos de contenido ("lecture", "word_search", çrossword", "quiz", "video"), CreateActivityRequestDto utiliza default_factory=dict para parameters, UpdateActivityRequestDto implementa campos opcionales con valor None para actualizaciones parciales.

Response DTOs (Objetos de transferencia de información): Los Response DTOs constituyen objetos especializados que estructuran y serializan las respuestas de la API, convirtiendo los modelos de dominio en formatos JSON apropiados para la comunicación con los clientes del sistema. Se puede apreciar el DTO para actividades en la Figura 3.25.

Características de implementación:

- **Convierten modelos de dominio a diccionarios JSON:** El proyecto implementa Response DTOs que transforman automáticamente los modelos de dominio en estructuras JSON con seriales. ActivityResponseDto.from_domain() convierte objetos Activity en diccionarios JSON con campos como id, title, description, content_type, parameters y metadatos de auditoría. NodeResponseDto.from_domain() convierte objetos Node en respuestas JSON, AuthResponseDto.from_success_authentication() estructura respuestas de autenticación con información de usuario y token, y MapProgressResponseDto.from_domain_data() convierte datos complejos de progreso en estructuras JSON, garantizando un correcto formato bien estructurado para las respuestas HTTP.

- **Incluyen objetos relacionados:** Los Response DTOs implementan la inclusión de objetos relacionados para proporcionar información completa y contextual. ActivityResponseDto incluye información del nodo relacionado mediante is_active calculado, NodeResponseDto incluye actividades relacionadas, MapProgressResponseDto incluye información anidada de nodos con actividades y estados de progreso, y AuthResponseDto incluye información completa del usuario con roles, optimizando consultas al reducir requests necesarios y mejorando la experiencia del usuario con datos contextuales.

```python
class ActivityResponseDto(BaseModel):
    id: UUID
    node_id: UUID
    title: str
    description: Optional[str] = None
    content_type: str
    order_index: int
    is_required: bool
    estimated_duration_minutes: Optional[int] = None
    parameters: Dict[str, Any]
    created_at: datetime
    updated_at: datetime
    is_active: bool
    node: Optional[NodeResponseDto] = None

    @classmethod
    def from_domain(cls, activity):
        try:
            print(f"DEBUG: ActivityResponseDto.from_domain() - procesando actividad: {activity.title}")
            node_dto = None
            if activity.node:
                node_dto = NodeResponseDto.from_domain(activity.node)

            result = cls(
                id=activity.id,
                node_id=activity.node_id,
                title=activity.title
```

Figura 3.25: Objeto de transferencia de información para actividades. Capa de puntos de entrada.

Capítulo 4

Implementación del Proyecto

La implementación del proyecto 'Ruta de Empleabilidad' representa la materialización de todos los análisis, diseños y decisiones arquitectónicas desarrolladas en los capítulos anteriores. Este capítulo documenta el proceso de construcción de la aplicación web gamificada, desde la configuración inicial del entorno de desarrollo hasta el despliegue final de un sistema funcional.

El desarrollo se estructuró en dos áreas: el frontend, responsable de la experiencia de usuario y la interfaz gamificada, y el backend, encargado de la lógica de negocio, la gestión de datos y la seguridad del sistema. Ambas áreas fueron implementadas utilizando tecnologías modernas y patrones arquitectónicos que garantizan la mantenibilidad, escalabilidad y robustez de la aplicación.

En este capítulo se detallan los aspectos técnicos más relevantes de la implementación, incluyendo la configuración del stack tecnológico, la arquitectura de componentes del frontend, la implementación de la API REST en el backend, el diseño y configuración de la base de datos, y finalmente, la integración y despliegue del sistema completo. Cada sección incluye ejemplos de código, diagramas explicativos y capturas de pantalla que ilustran el resultado final de la implementación.

4.1. Configuración del entorno de desarrollo

4.1.1. Stack tecnológico

El stack tecnológico seleccionado para el desarrollo de la aplicación fue seleccionado considerando los requerimientos específicos del proyecto, la experiencia del equipo de desarrollo y las mejores prácticas de la industria. Para el frontend, se optó por React 18.x como biblioteca principal, complementada con TypeScript para proporcionar tipado estático y mejorar la robustez del código. La estilización se implementó utilizando Tailwind CSS, una biblioteca de utilidades que permite un desarrollo rápido y consistente de interfaces de usuario responsivas y modernas. En el backend, se seleccionó FastAPI como framework principal de Python, debido a su rendimiento excepcional y soporte nativo para operaciones asíncronas. Para la gestión de la base de datos, se implementó SQLAlchemy como ORM, facilitando la interacción con la base de datos PostgreSQL de manera eficiente y segura. La autenticación y autorización se maneja mediante JWT (JSON Web Tokens), proporcionando un sistema de seguridad robusto y escalable que permite la gestión de sesiones de usuario de manera eficiente.

4.1.2. Configuración de herramientas

La configuración de las herramientas de desarrollo se realizó considerando la compatibilidad entre versiones y la estabilidad del entorno de trabajo. Para el desarrollo frontend, se configuró

Node.js en su versión 18.x, que proporciona soporte LTS (Long Term Support) y características modernas de JavaScript que son esenciales para el desarrollo con React y TypeScript. En el lado del backend, se implementó Python en su versión 3.11+, aprovechando las mejoras de rendimiento y las nuevas características del lenguaje que optimizan el desarrollo de APIs REST. La siguiente es una lista de las herramientas y sus versiones:

- **Node.js:** Versión 18.x para desarrollo frontend
- **Python:** Versión 3.11+ para desarrollo backend
- **Base de datos:** PostgreSQL 15.X
- **IDE:** Visual Studio Code con extensiones.

4.2. Implementación del frontend

La implementación del frontend representa la capa de presentación de la aplicación, donde se materializa la experiencia de usuario gamificada diseñada para los estudiantes. Esta sección aborda el desarrollo de los componentes React que conforman la interfaz interactiva, desde el sistema de autenticación hasta las actividades y su progreso. En el frontend desarrollado con React y TypeScript, se utiliza la convención lowerCamelCase para nombrar variables, funciones y propiedades de componentes, manteniendo consistencia con las convenciones estándar del ecosistema TypeScript.

4.2.1. Sistema de autenticación y autorización

El sistema de autenticación y autorización constituye el punto de entrada fundamental de la aplicación, garantizando que solo usuarios autorizados puedan acceder a los recursos y funcionalidades correspondientes a su rol. La implementación incluye una interfaz de autenticación intuitiva que permite a los estudiantes y administradores iniciar sesión de manera segura desde el portal de servicios de la universidad. El sistema gestiona los tokens JWT de manera eficiente, almacenándolos de forma segura en el navegador y renovándolos automáticamente cuando sea necesario para mantener la sesión activa. La gestión de roles de usuario se implementa mediante un sistema de autorización basado en permisos, donde cada usuario tiene asignado un rol específico (estudiante o administrador) que determina las funcionalidades y recursos a los que puede acceder.

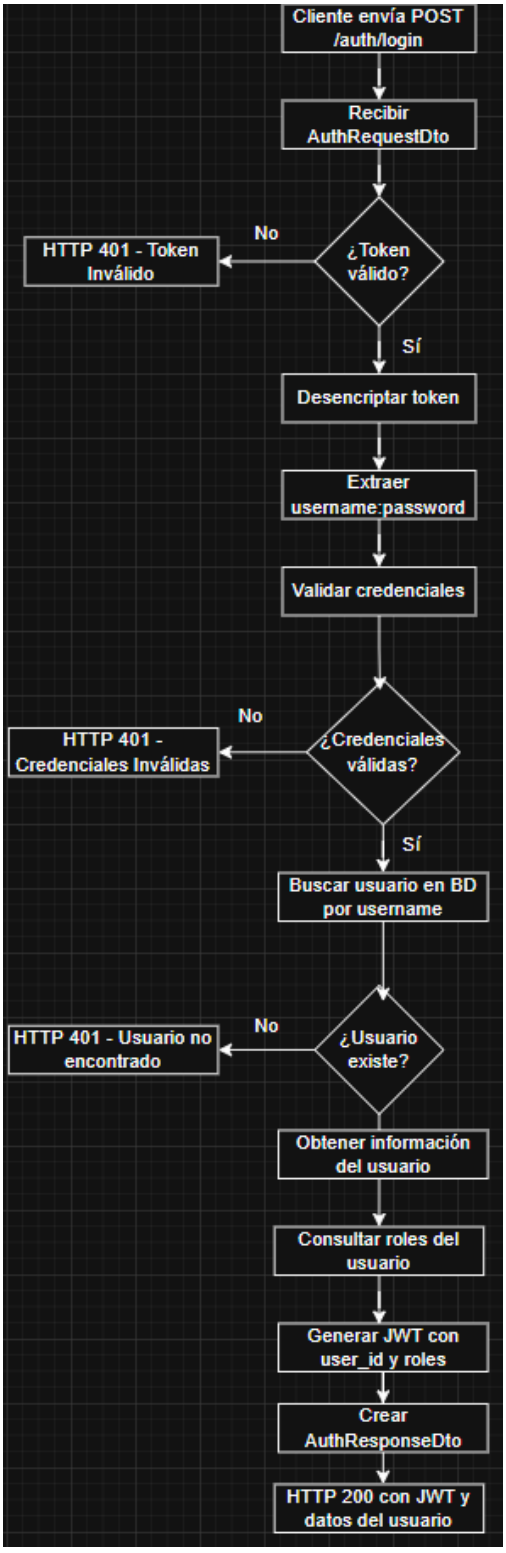


Figura 4.1: Diagrama de flujo de ingreso al aplicativo

4.2.2. Explicación del diagrama de flujo de Autenticación

El diagrama de flujo de autenticación implementa un sistema de autenticación basado en tokens JWT con validación en múltiples capas. El proceso inicia cuando el cliente frontend envía una petición POST al endpoint /auth/login con un AuthRequestDto que contiene las credenciales del usuario. El backend primero valida la integridad y formato del token recibido, y si es válido, procede a desencriptarlo para extraer el username y password. Posteriormente, valida estas credenciales contra los criterios del sistema y, si son correctas, busca al usuario en la base de datos utilizando el username proporcionado. Una vez confirmada la existencia del usuario, el sistema obtiene su información completa y consulta los roles asociados a su cuenta. Con esta información, genera un nuevo JWT que incluye el user_id y los roles del usuario, crea un AuthResponseDto con todos los datos necesarios, y devuelve una respuesta HTTP 200 exitosa que contiene el token JWT y la información del usuario. Este flujo garantiza la seguridad mediante validaciones secuenciales en cada capa (token, credenciales, existencia del usuario), maneja errores específicos con códigos HTTP 401 para cada tipo de falla (token inválido, credenciales incorrectas, usuario no encontrado), y proporciona la información necesaria para la autorización basada en roles en el frontend, permitiendo que la aplicación React determine automáticamente si el usuario debe ser dirigido a la interfaz de estudiante o administrador según sus permisos.

4.2.3. Actividades interactivas implementadas

Las actividades interactivas constituyen un componente importante de la experiencia de aprendizaje gamificada, diseñadas para transformar el contenido tradicional en experiencias dinámicas. Cada actividad fue desarrollada considerando los principios de gamificación, la interactividad y evaluación automática, permitiendo a los estudiantes aprender de manera activa mientras reciben retroalimentación inmediata sobre su desempeño. Las actividades siguen un esquema dinámico de implementación, en el cual, toda la información para construir una actividad es almacenada en un campo JSON en la base de datos, lo que permite obtener diferentes estructuras de parámetros para las diferentes actividades existentes sin construir tablas o modelos adicionales, simplificando la lógica y tiempo de ejecución de las consultas.

Para todas las actividades se diseñó un componente reutilizable que específica un listado de parámetros, y con ese listado crea dinámicamente la actividad.

4.2.3.1. Sopa de letras

La actividad de sopa de letras implementa un algoritmo de generación procedural que utiliza técnicas de programación dinámica para crear cuadrículas adaptativas para la distribución de elementos. El sistema emplea un generador de números pseudoaleatorios para garantizar la variabilidad entre sesiones, implementando un patrón de reproducibilidad controlada que permite la validación de resultados mientras mantiene la diversidad en la experiencia del usuario. A continuación se describe la especificación para el almacenamiento de los parámetros necesarios para construir la actividad de tipo sopa de letras en la base de datos:

```
{
  "title": "string",
  "words": "string[]",
  "gridSize": "number",
  "initialLives": "number"
}
```

Algoritmo 4.1: Especificación parámetros sopa de letras

Donde, el campo título (title) que es un campo de tipo texto corresponde al título de la actividad mostrado, el campo palabras que una lista de texto (words) que corresponde a las palabras a encontrar en la sopa de letras, el campo gridSize (tamaño de malla) que es un campo numérico que determina la cantidad de columnas y filas de la sopa de letras, por último, el campo initialLives (vida inicial) que corresponde a la cantidad de oportunidades que tiene un usuario para completar la actividad antes de tener que repetirla.

4.2.3.2. Crucigrama

El crucigrama implementa una arquitectura de interfaz reactiva que gestiona la interacción del usuario mediante un sistema que escucha eventos optimizado para dispositivos táctiles y de entrada por teclado. La implementación utiliza un patrón para manejar los eventos de clic en celdas individuales, donde cada celda actúa como un componente de estado independiente que mantiene su propio ciclo de vida y validación. Esta actividad cuenta con un sistema de validación en tiempo real que permite ingresar la palabra completa en las casillas para poder determinar si es o no correcta. A continuación se describe la especificación para el almacenamiento de los parámetros necesarios para construir la actividad de tipo crucigrama en la base de datos:

```
{
  "title": "string",
  "words": [
    {
      "clue": "string",
      "word": "string",
      "startCol": "number",
      "startRow": "number",
      "direction": "string"
    }
  ],
  "gridSize": "number",
  "initialLives": "number"
}
```

Algoritmo 4.2: Especificación parámetros crucigrama

El campo título (title) que es un campo de tipo texto corresponde al título de la actividad mostrado al estudiante. El campo palabras (words) que es una lista de objetos corresponde a las palabras del crucigrama, donde cada objeto contiene la pista (clue) como texto descriptivo, la palabra (word) como respuesta, las coordenadas de inicio (startCol y startRow) como valores numéricos

que determinan la posición en la grilla, y la dirección (direction) como texto que indica si la palabra va horizontal (.ªcross") o vertical ("down"). El campo gridSize (tamaño de malla) que es un campo numérico determina la cantidad de columnas y filas del crucigrama. Por último, el campo initialLives (vidas iniciales) que corresponde a la cantidad de oportunidades que tiene un estudiante para completar la actividad antes de tener que reiniciar.

4.2.3.3. Visualización de vídeos

La actividad de visualización de vídeos implementa una integración directa con YouTube mediante el uso de la API oficial de YouTube que permite embebir reproductores de video de manera segura y controlada. El sistema utiliza iframes para cargar el contenido multimedia, estableciendo un canal de comunicación entre el reproductor y la aplicación para monitorear el comportamiento del usuario en tiempo real. El sistema de control de tiempo implementa un mecanismo de seguimiento que registra continuamente el progreso de reproducción del video. El algoritmo utiliza eventos nativos del reproductor para detectar cuando el usuario inicia, pausa, avanza o rebobina el contenido. A continuación se describe la especificación para el almacenamiento de los parámetros necesarios para construir la actividad de tipo vídeo en la base de datos:

```
{
  "title": "string",
  "videoUrl": "string",
  "description": "string",
  "requiredWatchTime": "number"
}
```

Algoritmo 4.3: Especificación parámetros vídeo

Donde, el campo título (title) que es un campo de tipo texto corresponde al título del video educativo mostrado al estudiante. El campo URL del video (videoUrl) que es un campo de tipo texto contiene la dirección web donde está alojado el archivo de video que el estudiante debe reproducir. El campo descripción (description) que es un campo de tipo texto proporciona información sobre el contenido del video, explicando qué aprenderá el estudiante al verlo. Por último, el campo tiempo requerido de visualización (requiredWatchTime) que es un valor numérico representa la cantidad mínima de segundos que el estudiante debe ver del video para considerar completada la actividad, permitiendo verificar que ha dedicado suficiente tiempo al contenido educativo.

4.2.3.4. Lecturas

La actividad de lecturas implementa un sistema de navegación por carrusel que utiliza componentes de interfaz reactivos para presentar contenido educativo estructurado en secciones modulares. El sistema emplea un patrón de paginación virtual que carga dinámicamente el contenido según la posición del usuario, optimizando el rendimiento y reduciendo el tiempo de carga inicial. El módulo de carrusel interactivo implementa un sistema de navegación basado en eventos que permite la transición fluida entre secciones mediante animaciones CSS3 y transiciones suaves. A continuación

se describe la especificación para el almacenamiento de los parámetros necesarios para construir la actividad de tipo lectura en la base de datos:

```
{
  "title": "string",
  "slides": [
    {
      "title": "string",
      "content": "string"
    }
  ],
  "description": "string"
}
```

Algoritmo 4.4: Especificación parámetros lectura

El campo título (title) que es un campo de tipo texto corresponde al título principal de la actividad de lectura mostrado al estudiante. El campo diapositivas (slides) que es una lista de objetos corresponde a las diferentes secciones o páginas del material de lectura, donde cada objeto contiene un título (title) como texto que identifica el tema de la diapositiva, y el contenido (content) como texto que incluye el material educativo, vocabulario, ejemplos y ejercicios de práctica que el estudiante debe leer y estudiar. Por último, el campo descripción (description) que es un campo de tipo texto proporciona una breve explicación del objetivo y contenido de la actividad de lectura para orientar al estudiante sobre lo que aprenderá.

4.2.3.5. Quiz

La actividad de quiz implementa un sistema de evaluación adaptativa basado en preguntas de selección múltiple que utiliza algoritmos de calificación automática para evaluar la comprensión del estudiante sobre los contenidos del curso. El sistema emplea un patrón de flujo de preguntas secuencial que presenta las cuestiones de manera ordenada, manteniendo el estado de respuestas mediante un sistema de gestión de estado reactivo. El módulo de evaluación automática implementa un algoritmo de calificación instantánea que procesa las respuestas del usuario mediante validación de claves de respuesta almacenadas en la configuración JSON de la actividad. A continuación se describe la especificación para el almacenamiento de los parámetros necesarios para construir la actividad de tipo quiz en la base de datos:

```
{
  "title": "string",
  "questions": [
    {
      "id": "string",
      "options": "string[]",
      "question": "string",
      "explanation": "string",
      "correctAnswer": "number"
    }
  ],
```

```
  "description": "string",
  "requireAllCorrect": "boolean"
}
```

Algoritmo 4.5: Especificación parámetros quiz

Donde, el campo título (title) que es un campo de tipo texto corresponde al título del quiz mostrado al estudiante. El campo preguntas (questions) que es una lista de objetos corresponde a las preguntas del quiz, donde cada objeto contiene un identificador único (id) como texto, las opciones de respuesta (options) como una lista de textos que representan las alternativas disponibles, la pregunta (question) como texto con el enunciado a responder, la explicación (explanation) como texto que proporciona información adicional sobre la respuesta correcta, y la respuesta correcta (correctAnswer) como número que indica el índice de la opción correcta en el array de opciones. El campo descripción (description) que es un campo de tipo texto proporciona una breve explicación del propósito del quiz. Por último, el campo requireAllCorrect (requerir todas correctas) que es un valor booleano determina si el estudiante debe responder correctamente todas las preguntas para aprobar la actividad o si se permite un margen de error.

4.3. Implementación del backend

La implementación del backend es la responsable de gestionar toda la lógica de negocio, la seguridad de los datos y la comunicación entre el frontend y la base de datos. Esta sección aborda el desarrollo del servidor utilizando FastAPI como framework principal, implementando los principios de Clean Architecture que soporten las funcionalidades complejas de la aplicación gamificada.

4.3.1. Clean Architecture

Diagrama. Arquitectura por capas de Clean Architecture El diagrama completo de la arquitectura por capas implementada en el proyecto se encuentra disponible en: https://drive.google.com/file/d/1S8_mqzdpXtNZkLhqW1HynT4vGhY92zzR/view?usp=sharing
Como se puede observar en el diagrama, se tiene implementado en el proyecto el patrón de arquitectura Clean Architecture implementada en el proyecto sigue un patrón de diseño que organiza el código en capas concéntricas, donde cada capa tiene responsabilidades específicas y las dependencias fluyen hacia el interior, manteniendo el núcleo del sistema independiente de tecnologías externas. En la capa más externa se encuentran los puntos de entrada, que incluyen los routers de FastAPI como AuthRouter, ActivityRouter, NodeRouter y MapRouter, junto con los DTOs que manejan la validación y estructuración de datos de entrada y salida. Estos componentes actúan como la interfaz entre el mundo exterior y la aplicación, procesando requests HTTP, validando datos y convirtiendo respuestas del dominio en formatos JSON apropiados para los clientes.
La capa de aplicación contiene el ApplicationContainer que gestiona la inyección de dependencias, y los casos de uso como ManageAuthenticationUseCase, ManageActivitiesUseCase, ManageNodesUseCase y ManageMapProgressUseCase que orquestan las operaciones de negocio. Esta capa actúa como el coordinador que recibe las solicitudes de los routers, utiliza los puertos del dominio para

acceder a funcionalidades, y devuelve respuestas estructuradas.
En el corazón del sistema se encuentra la capa de dominio, que contiene las entidades de negocio como User, Activity, Node, UserActivityProgress y UserNodeProgress, junto con las interfaces abstractas (puertos) como UserPort, ActivityPort y NodePort. Esta capa define las reglas de negocio fundamentales del sistema educativo, las entidades que representan los conceptos del dominio, y los contratos que deben cumplir las implementaciones externas.
La capa más externa es la infraestructura (Infrastructure Layer), que implementa las interfaces definidas en el dominio y interactúa con el mundo real. Esta capa incluye los repositorios como UserRepository, ActivityRepository y NodeRepository que implementan los puertos del dominio, las entidades de base de datos que mapean las tablas de PostgreSQL, los mappers que convierten entre entidades de base de datos y modelos de dominio, y servicios externos como JwtTokenService y LDAP. Los repositorios traducen las llamadas del dominio en operaciones SQL, los mappers manejan la conversión bidireccional de datos entre capas, y los servicios externos proporcionan funcionalidades como autenticación y generación de tokens.
El flujo de dependencias garantiza que las capas internas no dependan de las externas, permitiendo que el dominio permanezca puro e independiente de tecnologías específicas. Cuando un router recibe una request, utiliza un caso de uso que a su vez utiliza un puerto del dominio, y la infraestructura implementa ese puerto proporcionando la funcionalidad concreta.

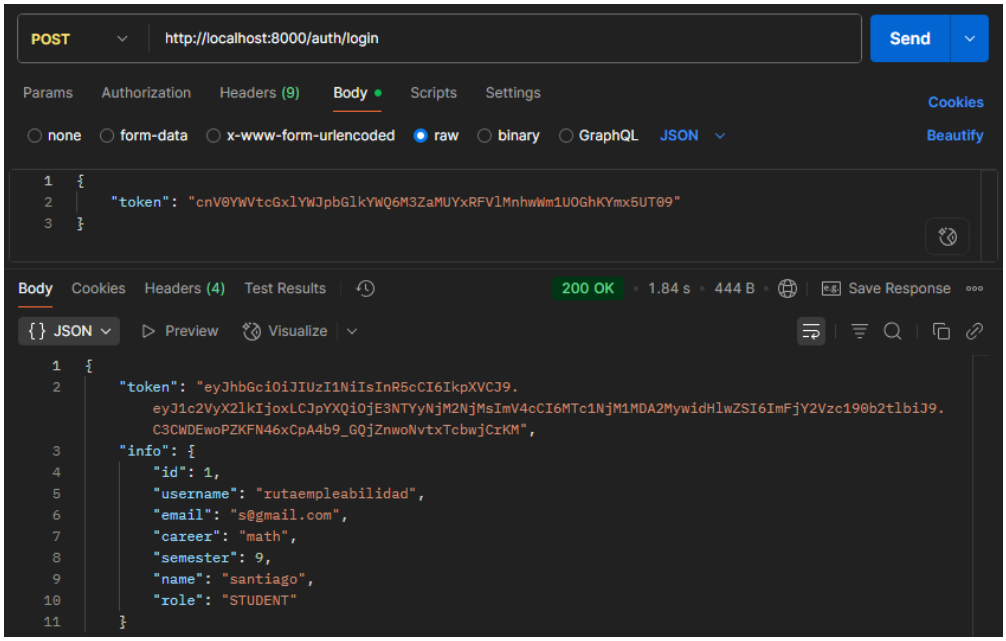
4.3.2. API REST implementada

La API REST implementada constituye la interfaz de comunicación principal entre el frontend y el backend. La implementación utiliza FastAPI como framework principal, aprovechando sus características de validación de tipos y soporte nativo para operaciones asíncronas que optimizan el rendimiento de la aplicación.

4.3.3. Endpoint de autenticación

- **POST /auth/login**

El endpoint POST /auth/login permite la autenticación de usuarios mediante un token encriptado. Recibe un objeto JSON con el campo token de tipo string que contiene las credenciales encriptadas. La aplicación implementa un sistema de autenticación seguro que procesa tokens encriptados y genera automáticamente un token JWT con información del usuario y sus roles. Lo anterior puede verse reflejado en la prueba de Postman 2. El sistema incluye un mecanismo de control de acceso basado en roles, un tiempo de expiración de 1 hora para las sesiones, y manejo adecuado de errores retornando códigos HTTP apropiados. Además, es compatible con múltiples formatos de token, proporcionando flexibilidad en los mecanismos de autenticación soportados. Esta implementación cumple con los estándares de seguridad web modernos y garantiza un proceso de autenticación eficiente y confiable para el correcto funcionamiento de las funcionalidades posteriores de la aplicación.



Postman 1: Imagen endpoint POST /auth/login

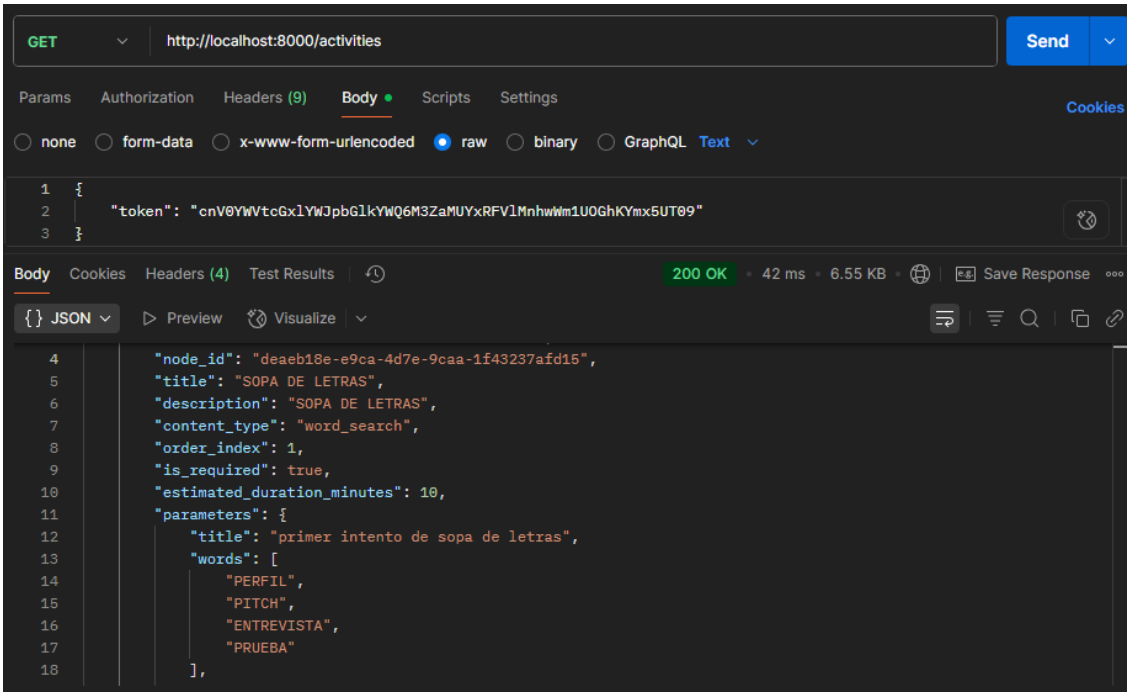
4.3.4. Manejo de actividades

- **GET /activities**

El endpoint GET /activities permite obtener todas las actividades disponibles en el sistema. Recibe un header de autorización opcional que puede contener un token Bearer para identificar al usuario autenticado, y retorna un array JSON con la información completa de cada actividad. Cada actividad retornada contiene información detallada, incluyendo identificador único, título, descripción, tipo de contenido (lectura, búsqueda de palabras, crucigrama, cuestionario o video), índice de orden, duración estimada, parámetros específicos y estado de completado. Además, el sistema proporciona metadatos como fechas de creación y actualización, y un flag de estado activo para control de visibilidad. La prueba en postman para este endpoint puede verse en la imagen 1. Esta implementación garantiza un acceso flexible a las actividades del sistema, permitiendo tanto consultas públicas como personalizadas según el nivel de autenticación del usuario.

- **GET /activities/{id}**

El endpoint GET /activities/id permite obtener una actividad específica mediante su identificador único. Recibe como parámetro de ruta el ID de la actividad en formato UUID y retorna un objeto JSON con la información completa de la actividad solicitada. a prueba en postman para este endpoint puede verse en la imagen 3. La aplicación implementa un sistema de consulta individual de actividades que procesa identificadores únicos y valida la existencia de la actividad en la base de datos. El sistema incluye un mecanismo de manejo de errores robusto que retorna un código



Postman 2: Imagen endpoint GET /activities

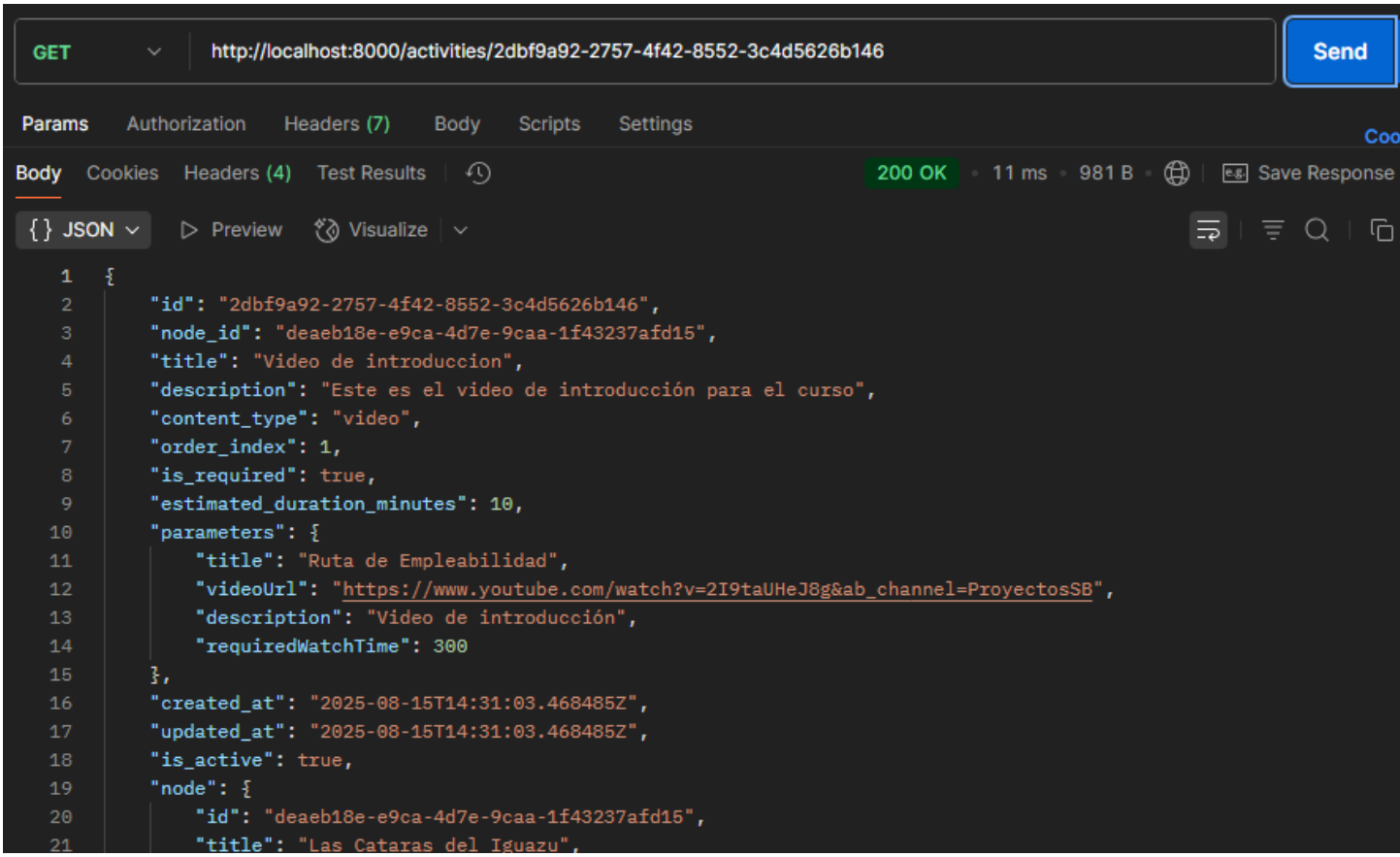
HTTP 404 (Not Found) cuando la actividad especificada no existe en el sistema, proporcionando un mensaje descriptivo que incluye el ID de la actividad no encontrada. Cada actividad retornada contiene información detallada.

- **POST /activities/{id}**

El endpoint POST /activities permite crear una nueva actividad en el sistema. Recibe un objeto JSON con los datos de la actividad a crear, incluyendo título, descripción, tipo de contenido, índice de orden, duración estimada, ID del nodo y parámetros específicos. a prueba en postman para este endpoint puede verse en la imagen 4. La aplicación implementa un sistema de validación robusto mediante esquemas Pydantic que verifica restricciones como longitud del título, valores numéricos positivos y tipos de contenido válidos. El sistema proporciona manejo detallado de errores, retornando códigos HTTP apropiados para JSON inválido, errores de validación y errores internos.

- **POST /progress/{activityId}**

El endpoint POST /progress/{activityId} permite registrar el progreso de un usuario en una actividad existente. Una muestra de esto se puede ver en la imagen 5. Recibiendo como una variable en el path el id de la actividad en la base de datos, y en el cuerpo de la solicitud los metadatos necesarios para calcular el score dependiendo del tipo de actividad. El sistema utiliza el token proporcionado para identificar al usuario autenticado y asociar correctamente el progreso registrado con su perfil, garantizando la integridad y seguridad de los datos almacenados. Una vez procesada la información,

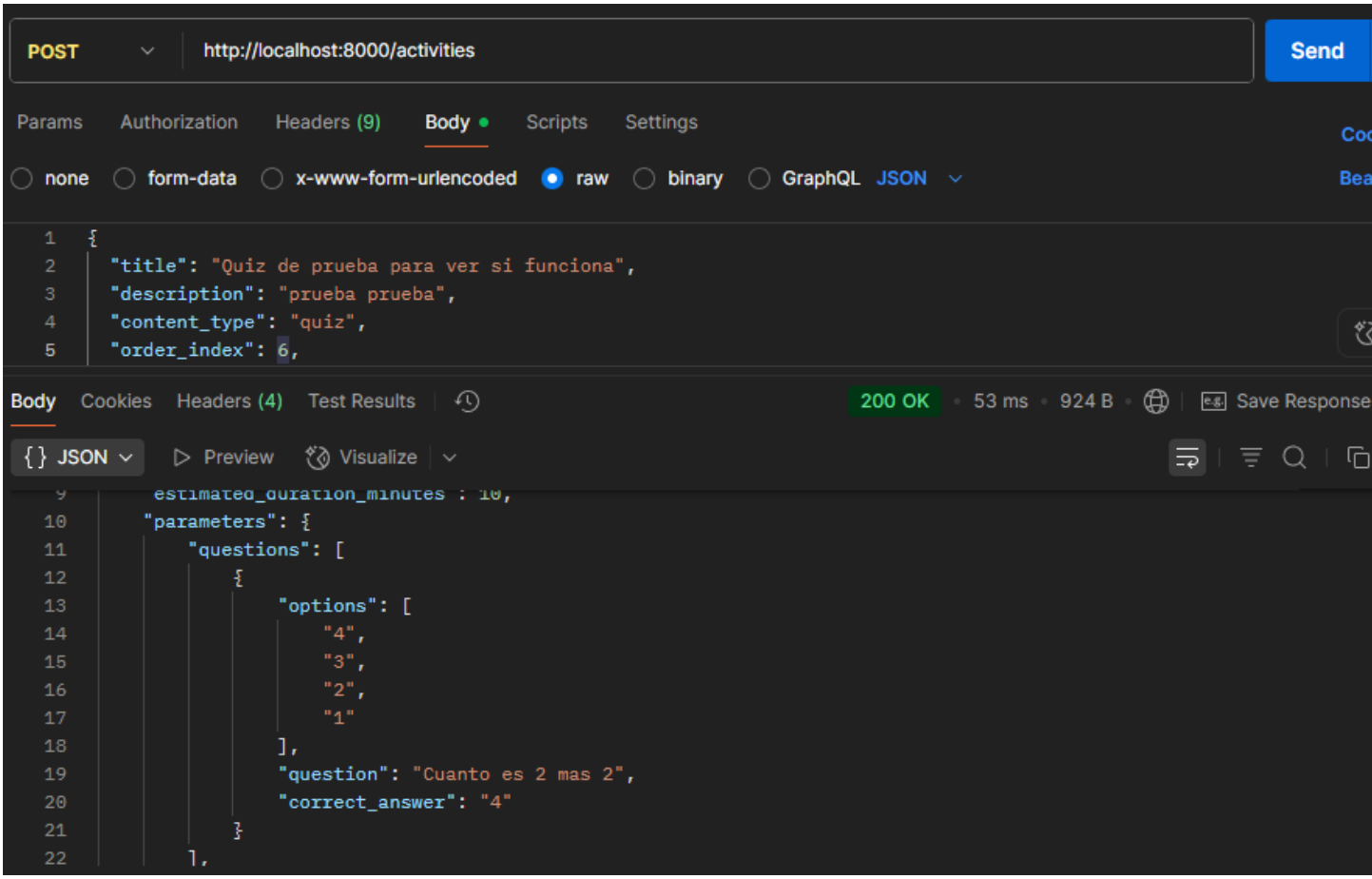


Postman 3: Imagen endpoint GET /activities/{id}

el endpoint retorna una respuesta que confirma el registro exitoso del progreso junto con los detalles actualizados de la actividad completada.

- **GET /progress/activityId**

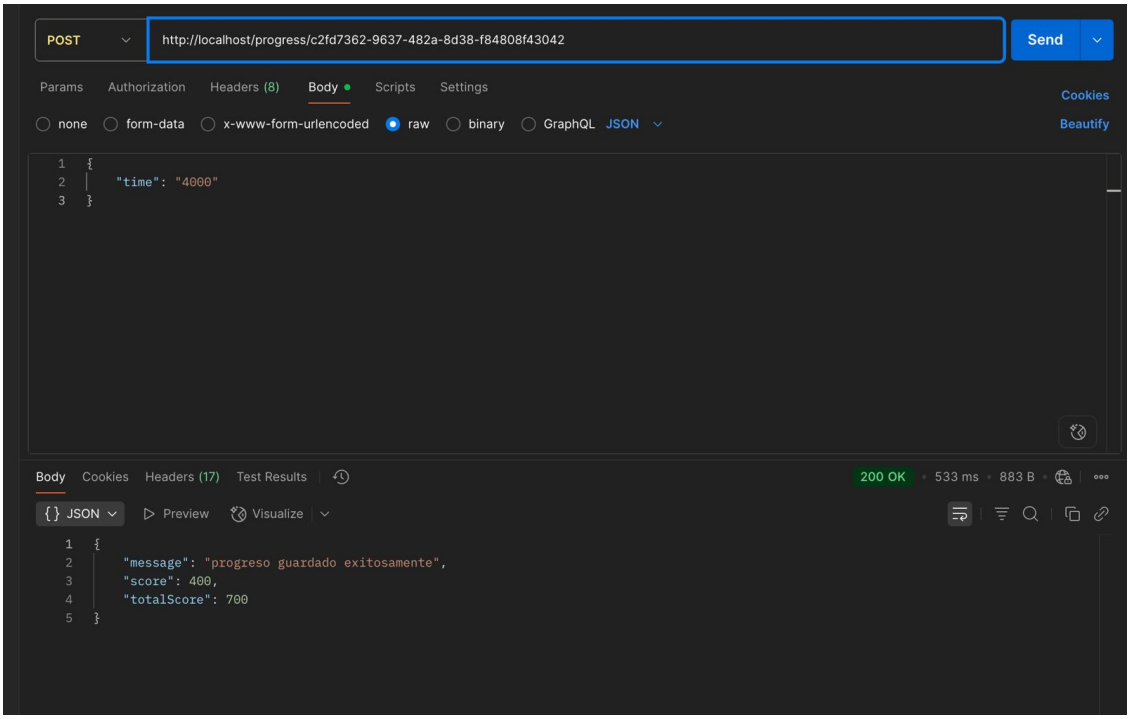
El endpoint GET /progress/activityId permite consultar el progreso específico de un usuario en una actividad determinada. Este endpoint requiere autenticación mediante un token de usuario incluido en el header de autorización y utiliza el identificador de la actividad como parámetro de ruta para localizar el registro correspondiente. La implementación de este endpoint en postman puede verse en la imagen 6. El sistema procesa la petición extrayendo el ID del usuario desde el token de autenticación y combinándolo con el ID de actividad proporcionado para realizar una búsqueda precisa en la base de datos. Una vez localizado el registro de progreso, el servicio retorna una respuesta JSON completa que incluye todos los detalles del progreso del usuario, incluyendo el estado de completitud, puntuación obtenida, datos de finalización con respuestas detalladas, tiempos de inicio y finalización, así como las marcas temporales de creación y actualización del registro.



Postman 4: Imagen endpoint POST /activities/{id}

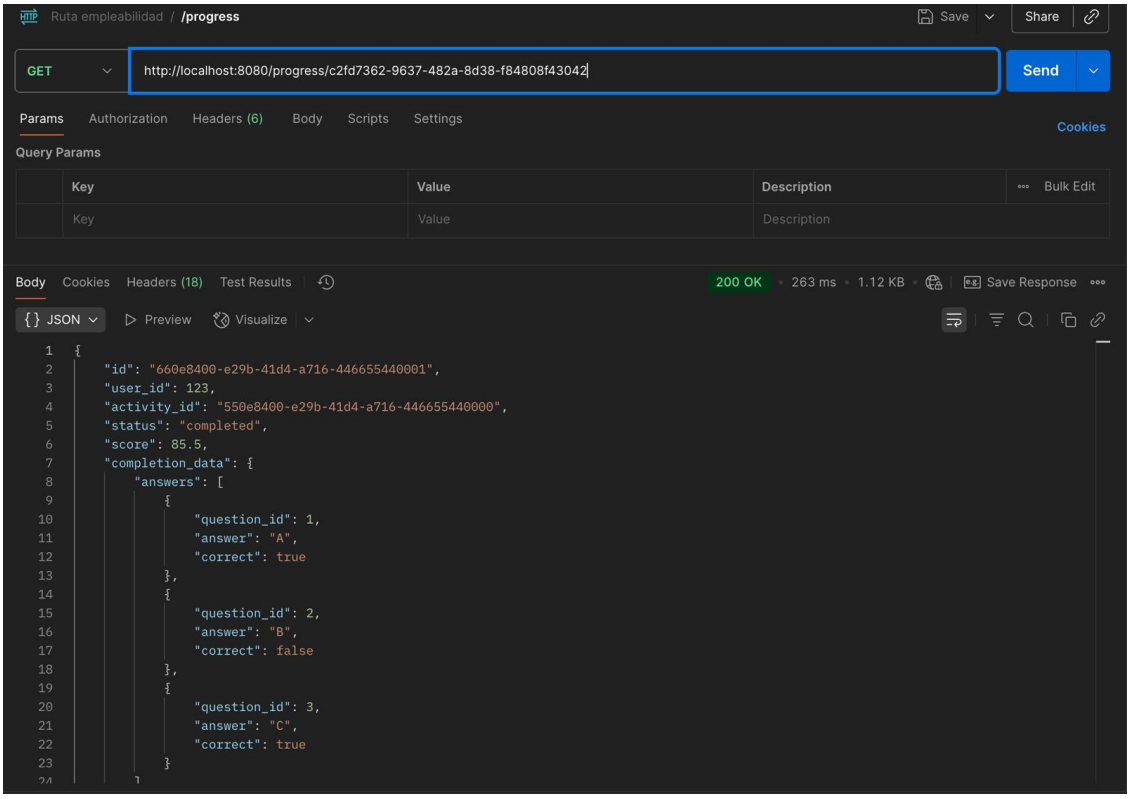
4.3.4.1. Sistema de calificación automática

El sistema de calificación automática implementa algoritmos especializados para cada tipo de actividad, considerando múltiples variables que van más allá de la simple verificación de respuestas correctas, para ofrecer una evaluación integral y justa del aprendizaje. Para las actividades de sopa de letras, el sistema evalúa no solo la precisión en encontrar las palabras correctas, sino también el tiempo empleado en completar la actividad y el número de vidas utilizadas. En el caso de los crucigramas, la calificación automática verifica cada letra ingresada en tiempo real, proporcionando retroalimentación inmediata que permite a los estudiantes corregir errores antes de finalizar la actividad. El sistema considera el número de intentos realizados, el tiempo total empleado y la precisión final. Para las actividades de visualización de videos, el sistema implementa un algoritmo sofisticado que monitorea el tiempo efectivo de visualización, detectando comportamientos como pausar o adelantar. Las actividades de lectura utilizan un sistema de seguimiento que registra el tiempo dedicado a cada sección del carrusel, validando que el estudiante haya navegado completamente por



Postman 5: Imagen endpoint POST /progress/activityId

todo el contenido antes de considerar la actividad como completada. Para los quizzes, la calificación automática evalúa tanto la precisión de las respuestas como el tiempo de respuesta, considerando que respuestas rápidas y correctas indican un mejor dominio del contenido.



Postman 6: Imagen endpoint GET /progress/activityId

4.4. Base de datos

4.4.1. Esquema de la base de datos

La aplicación utiliza una base de datos PostgreSQL con un esquema relacional el cual se puede ver en la imagen 4.4.1 y está compuesto por cinco tablas principales diseñadas para gestionar usuarios, nodos de aprendizaje, actividades educativas y el progreso de los usuarios en el sistema.

- La tabla users almacena la información fundamental de cada usuario registrado, incluyendo identificador único, nombre de usuario, nombre completo, correo electrónico, rol asignado y metadatos de auditoría. El campo is_active permite controlar el estado de activación de las cuentas.
- La tabla nodes representa los módulos o unidades de aprendizaje, conteniendo título, descripción, ubicación geográfica, índice de orden y metadatos de auditoría. El campo created_by establece relación con la tabla de usuarios para identificar el creador de cada nodo.
- La tabla activities almacena las actividades educativas individuales que componen cada nodo, incluyendo título, descripción, tipo de contenido, duración estimada y parámetros específicos. El campo node_id establece la relación jerárquica con la tabla de nodos.

- La tabla user_activity_progress implementa el seguimiento del progreso individual de cada usuario en actividades específicas, registrando estado de completado, puntaje obtenido y metadatos de auditoría.
- La tabla user_node_progress registra el progreso general de cada usuario en los nodos completos, almacenando estado de completado, puntaje final y porcentaje de completado.

El esquema implementa un sistema de relaciones que permite el seguimiento jerárquico del progreso educativo, desde actividades individuales hasta nodos completos, facilitando la generación de reportes detallados y la personalización de la experiencia de aprendizaje según el progreso de cada usuario.

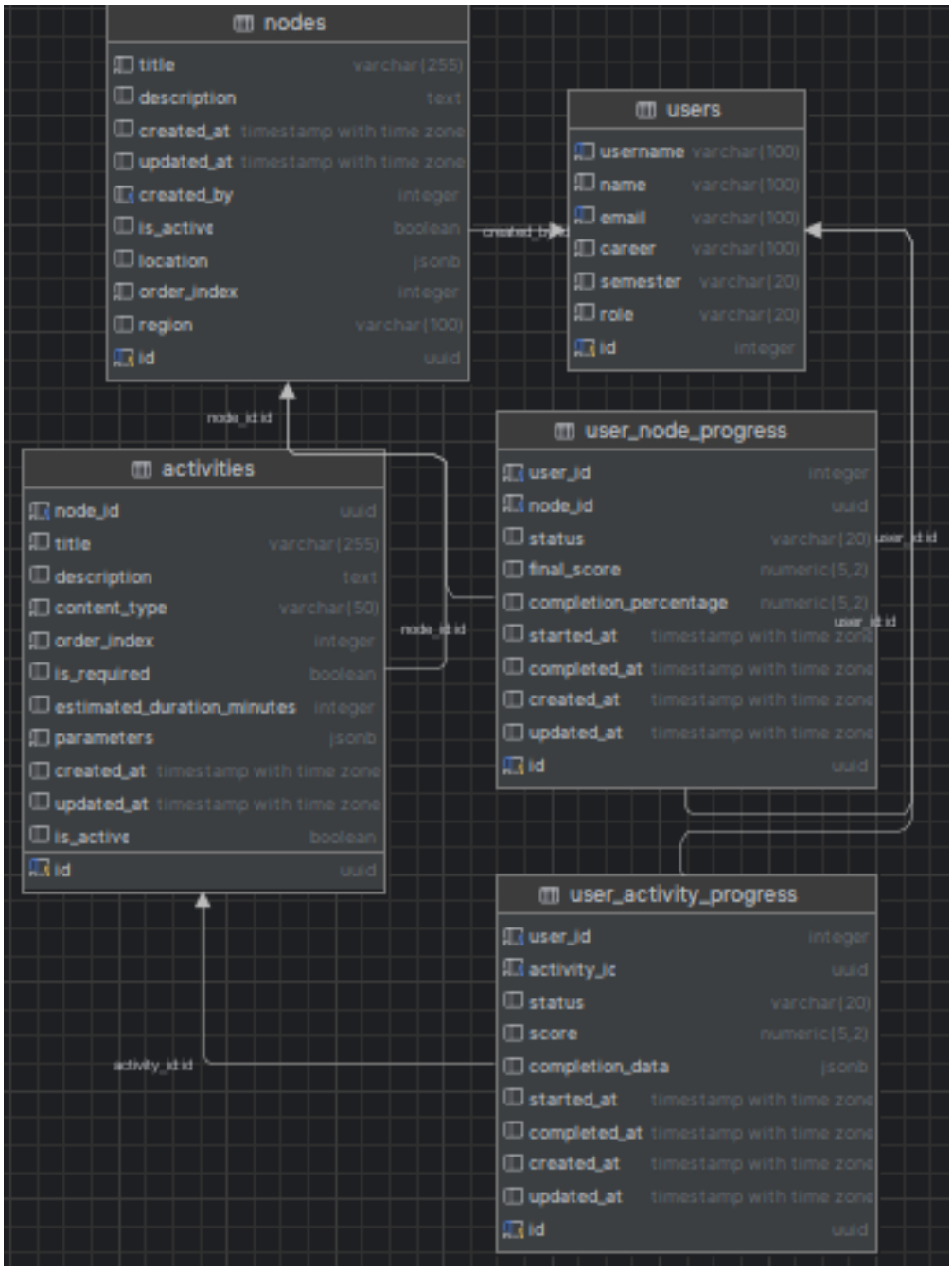


Diagrama 4.4.1: Imagen endpoint GET /activities

Capítulo 5

Pruebas y Validación

5.1. Matriz de Pruebas por Módulo

5.1.1. Módulo Autenticación

ID	MÓDULO	Método/Función	Descripción	Datos de Entrada	Resultado Esperado
UT-001	Autenticación	POST /auth/login	Login con token válido	{'token': 'usuario:password'}	Status 200, JWT token generado
UT-002	Autenticación	POST /auth/login	Login con token inválido	{'token': 'usuario:wrongpassword'}	Status 401, 'Authentication failed'
UT-003	Autenticación	POST /auth/login	Login sin token	{}	Status 422, 'Validation error'
UT-004	Autenticación	POST /auth/mobility-login	Login mobility con token válido	{'token': 'encrypted_token'}	Status 200, JWT con roles
UT-005	Autenticación	POST /auth/mobility-login	Login mobility con token inválido	{'token': 'invalid_encrypted'}	Status 401, 'Authentication failed'
UT-006	Autenticación	GET /auth/auth-callback	Obtener callback de autenticación	Sin parámetros	Status 200, mensaje de callback
UT-007	Autenticación	GET /auth/me	Verificar token JWT válido	Header: 'Bearer valid_jwt'	Status 200, información del usuario
UT-008	Autenticación	GET /auth/me	Verificar token externo válido	Header: 'Bearer usuario:password'	Status 200, información del usuario
UT-009	Autenticación	GET /auth/me	Verificar sin header Authorization	Sin header	Status 401, 'Authorization header missing'
UT-010	Autenticación	GET /auth/me	Verificar formato header inválido	Header: 'Invalid token'	Status 401, 'Invalid authorization header format'

ID	MÓDULO	Método/Función	Descripción	Datos de Entrada	Resultado Esperado
UT-011	Autenticación	GET /auth/me	Verificar token JWT inválido	Header: 'Bearer invalid_jwt'	Status 401, 'Invalid token'
UT-012	Autenticación	GET /auth/me	Verificar token externo inválido	Header: 'Bearer usuario:wrongpass'	Status 401, 'Invalid token'

5.1.1.1. Análisis del módulo de autenticación

- **Flujos Exitosos (UT-001, UT-004)** UT-001 confirma que el sistema procesa correctamente tokens 'usuario:password' válidos, generando JWTs y retornando código 200. UT-004 verifica el flujo de autenticación mobility con tokens encriptados, confirmando que el sistema puede desencriptar tokens, validar credenciales, consultar roles y generar JWTs con información de roles específica.
- **Manejo de Errores de Credenciales (UT-002, UT-005)** UT-002 confirma que el sistema rechaza tokens 'usuario:wrongpassword' retornando código 401, mientras que UT-005 verifica que el sistema maneja adecuadamente tokens encriptados corruptos o inválidos, manteniendo la seguridad del flujo mobility.
- **Validación de Datos de Entrada (UT-003)** La prueba UT-003 valida la capacidad del sistema al enviar un objeto JSON vacío . El sistema debe detectar la ausencia del campo 'token' requerido y retornar código 422, confirmando que las validaciones de entrada previenen el procesamiento de solicitudes malformadas.
- **Verificación Exitosa (UT-007, UT-008)** UT-007 confirma que el sistema puede decodificar y validar JWTs válidos, mientras que UT-008 verifica la compatibilidad con tokens externos en formato 'usuario:password', demostrando la flexibilidad del sistema para múltiples mecanismos de autenticación.
- **Manejo de Errores de Autorización (UT-009, UT-010)** Las pruebas UT-009 y UT-010 validan el manejo de problemas de autorización. UT-009 confirma que el sistema detecta la ausencia del header Authorization retornando código 401, mientras que UT-010 verifica que el sistema valida el formato correcto 'Bearer <token>' y rechaza formatos inválidos.
- **Validación de Tokens Inválidos (UT-011, UT-012)** UT-011 confirma que el sistema rechaza JWTs malformados o expirados, mientras que UT-012 verifica que el sistema mantiene la seguridad con tokens externos inválidos, retornando código 401 en ambos casos.

Los resultados de las pruebas del módulo de autenticación demuestran que el sistema implementa un mecanismo de autenticación seguro que maneja correctamente múltiples flujos de autenticación, incluyendo autenticación genérica con tokens 'usuario:password', autenticación mobility con tokens encriptados, verificación de tokens JWT, y compatibilidad con tokens externos.

5.1.2. Módulo Actividades

ID	MÓDULO	Método/Función	Descripción	Datos de Entrada	Resultado Esperado
ACT-001	Actividades	GET /activities	Obtener todas las actividades sin autenticación	Sin parámetros	Status 200, lista de actividades
ACT-002	Actividades	GET /activities	Obtener actividades con token JWT válido	Header: 'Bearer valid_jwt'	Status 200, lista con progreso del usuario
ACT-003	Actividades	GET /activities	Obtener actividades con token externo válido	Header: 'Bearer usuario:password'	Status 200, lista con progreso del usuario
ACT-004	Actividades	GET /activities	Obtener actividades con token inválido	Header: 'Bearer invalid_token'	Status 200, lista sin progreso específico
ACT-005	Actividades	GET /activities/activity_id	Obtener actividad por ID válido	UUID válido	Status 200, actividad específica
ACT-006	Actividades	GET /activities/activity_id	Obtener actividad con ID inexistente	UUID inexistente	Status 404, 'Activity not found'
ACT-007	Actividades	GET /activities/activity_id	Obtener actividad con ID inválido	String inválido	Status 422, 'Validation error'
ACT-008	Actividades	POST /activities	Crear actividad válida	{'title': 'Test', 'description': 'Test', 'content_type': 'lecture', 'order_index': 1, 'estimated_duration_minutes': 30, 'node_id': 'uuid', 'parameters': {}}	Status 200, actividad creada
ACT-009	Actividades	POST /activities	Crear actividad sin título	{'description': 'Test', 'content_type': 'lecture', 'order_index': 1, 'estimated_duration_minutes': 30, 'node_id': 'uuid'}	Status 422, 'Validation error'

ID	MÓDULO	Método/Función	Descripción	Datos de Entrada	Resultado Esperado
ACT-010	Actividades	POST /activities	Crear actividad con content_type inválido	{'title': 'Test', 'description': 'Test', 'content_type': 'invalid', 'order_index': 1, 'estimated_duration_minutes': 30, 'node_id': 'uuid'}	Status 422, 'Validation error'
ACT-011	Actividades	POST /activities	Crear actividad con order_index negativo	{'title': 'Test', 'description': 'Test', 'content_type': 'lecture', 'order_index': -1, 'estimated_duration_minutes': 30, 'node_id': 'uuid'}	Status 422, 'Validation error'
ACT-012	Actividades	POST /activities	Crear actividad con duración inválida	{'title': 'Test', 'description': 'Test', 'content_type': 'lecture', 'order_index': 1, 'estimated_duration_minutes': 0, 'node_id': 'uuid'}	Status 422, 'Validation error'
ACT-013	Actividades	POST /activities	Crear actividad con node_id inválido	{'title': 'Test', 'description': 'Test', 'content_type': 'lecture', 'order_index': 1, 'estimated_duration_minutes': 30, 'node_id': 'invalid-uuid'}	Status 422, 'Validation error'
ACT-014	Actividades	PUT /activities/activity_id	Actualizar actividad existente	{'title': 'Updated Title'}	Status 200, actividad actualizada
ACT-015	Actividades	PUT /activities/activity_id	Actualizar actividad inexistente	UUID inexistente	Status 500, 'Error updating activity'
ACT-016	Actividades	PUT /activities/activity_id	Actualizar actividad con datos inválidos	{'order_index': -1}	Status 422, 'Validation error'
ACT-017	Actividades	DELETE /activities/activity_id	Eliminar actividad existente	UUID válido	Status 200, 'Activity deleted successfully'

ID	MÓDULO	Método/Función	Descripción	Datos de Entrada	Resultado Esperado
ACT-018	Actividades	DELETE /activities/activity_id	Eliminar actividad inexistente	UUID inexistente	Status 500, 'Error deleting activity'

5.1.2.1. Análisis del módulo de actividades

- **Consultas Exitosas (ACT-001, ACT-002, ACT-003, ACT-005)** ACT-001 confirma que el endpoint permite acceso público sin autenticación, retornando una lista completa de actividades. ACT-002 y ACT-003 verifican que el sistema adapta la respuesta según el tipo de token (JWT o externo), incluyendo información de progreso del usuario cuando las credenciales son válidas. ACT-005 valida la consulta específica por ID, confirmando que el sistema puede recuperar actividades individuales mediante UUIDs válidos.

- **Manejo de Errores de Consulta (ACT-004, ACT-006, ACT-007)** ACT-004 confirma que el sistema maneja tokens inválidos de manera elegante, retornando actividades sin progreso específico en lugar de fallar completamente. ACT-006 verifica que el sistema detecta UUIDs inexistentes y retorna código 404 apropiado, mientras que ACT-007 confirma la validación de formatos de UUID inválidos retornando código 422.

- **Creación Exitosa (ACT-008)** La prueba ACT-008 valida el flujo principal de creación de actividades, confirmando que el sistema puede procesar datos válidos completos y crear actividades con todos los campos requeridos, retornando código 200 y la actividad creada.

- **Validaciones de Datos de Entrada (ACT-009 a ACT-013)** ACT-009 confirma que el sistema requiere campos obligatorios como título, ACT-010 verifica la validación de enumeraciones para content_type, ACT-011 y ACT-012 validan restricciones numéricas, y ACT-013 confirma la validación de formato UUID para node_id. Todas estas pruebas retornan código 422, confirmando que el sistema previene la creación de actividades con datos inválidos.

- **Operaciones Exitosas (ACT-014, ACT-017)** ACT-014 confirma que el sistema puede actualizar actividades existentes con datos parciales, mientras que ACT-017 verifica que el sistema implementa soft delete marcando actividades como inactivas en lugar de eliminarlas físicamente.

- **Manejo de Errores en Operaciones (ACT-015, ACT-016, ACT-018)** ACT-015 y ACT-018 confirman que el sistema maneja UUIDs inexistentes retornando código 500, mientras que ACT-016 verifica que las validaciones de datos se aplican también en operaciones de actualización.

Los resultados demuestran que el módulo de actividades implementa un sistema de gestión de contenido que maneja correctamente operaciones CRUD completas con validaciones exhaustivas de datos de entrada.

5.1.3. Módulo Nodos

ID	MÓDULO	Método/Función	Descripción	Datos de Entrada	Resultado Esperado
NODE-001	Nodos	GET /nodes	Obtener todos los nodos	Sin parámetros	Status 200, lista de nodos con actividades
NODE-002	Nodos	GET /nodes/node_id	Obtener nodo por ID válido	UUID válido	Status 200, nodo específico
NODE-003	Nodos	GET /nodes/node_id	Obtener nodo con ID inexistente	UUID inexistente	Status 404, 'Node not found'
NODE-004	Nodos	GET /nodes/node_id	Obtener nodo con ID inválido	String inválido	Status 422, 'Validation error'
NODE-005	Nodos	POST /nodes	Crear nodo válido	{'title': 'Test Node', 'description': 'Test Description', 'location': {'x': 10.5, 'y': 20.3}, 'region': 'Test Region'}	Status 200, nodo creado
NODE-006	Nodos	POST /nodes	Crear nodo sin título	{'description': 'Test', 'location': {'x': 10.5, 'y': 20.3}, 'region': 'Test Region'}	Status 422, 'Validation error'
NODE-007	Nodos	POST /nodes	Crear nodo sin descripción	{'title': 'Test Node', 'location': {'x': 10.5, 'y': 20.3}, 'region': 'Test Region'}	Status 422, 'Validation error'
NODE-008	Nodos	POST /nodes	Crear nodo sin ubicación	{'title': 'Test Node', 'description': 'Test Description', 'region': 'Test Region'}	Status 422, 'Validation error'

ID	MÓDULO	Método/Función	Descripción	Datos de Entrada	Resultado Esperado
NODE-009	Nodos	POST /nodes	Crear nodo sin región	{'title': 'Test Node', 'description': 'Test Description', 'location': {'x': 10.5, 'y': 20.3}}	Status 422, 'Validation error'
NODE-010	Nodos	POST /nodes	Crear nodo con coordenadas inválidas	{'title': 'Test Node', 'description': 'Test Description', 'location': {'x': 'invalid', 'y': 20.3}, 'region': 'Test Region'}	Status 422, 'Validation error'
NODE-011	Nodos	POST /nodes	Crear nodo con título muy largo	{'title': 'A'*256, 'description': 'Test Description', 'location': {'x': 10.5, 'y': 20.3}, 'region': 'Test Region'}	Status 422, 'Validation error'
NODE-012	Nodos	PUT /nodes/node_id	Actualizar nodo existente	{'title': 'Updated Node Title'}	Status 200, nodo actualizado
NODE-013	Nodos	PUT /nodes/node_id	Actualizar nodo inexistente	UUID inexistente	Status 500, 'Error updating node'
NODE-014	Nodos	PUT /nodes/node_id	Actualizar nodo con datos inválidos	{'title': ''}	Status 422, 'Validation error'
NODE-015	Nodos	DELETE /nodes/node_id	Eliminar nodo existente	UUID válido	Status 200, 'Node deleted successfully'
NODE-016	Nodos	DELETE /nodes/node_id	Eliminar nodo inexistente	UUID inexistente	Status 500, 'Error deleting node'

5.1.3.1. Análisis del módulo de nodos

- **Consultas Exitosas (NODE-001, NODE-002)** NODE-001 demuestra que el endpoint permite obtener todos los nodos del sistema, retornando una lista completa que incluye las actividades relacionadas. NODE-002 prueba la consulta específica por ID, evidenciando que el sistema puede recuperar nodos individuales mediante UUIDs válidos.

- **Manejo de Errores de Consulta (NODE-003, NODE-004)** NODE-003 muestra que el sistema detecta UUIDs inexistentes y retorna código 404 apropiado, mientras que NODE-004 establece la validación de formatos de UUID inválidos retornando código 422.
- **Creación Exitosa (NODE-005)** La prueba NODE-005 examina el flujo principal de creación de nodos, demostrando que el sistema puede procesar datos válidos completos y crear nodos con todos los campos requeridos, incluyendo información de ubicación geográfica y región, retornando código 200 y el nodo creado.
- **Validaciones de Campos Obligatorios (NODE-006 a NODE-009)** NODE-006 establece que el sistema requiere el campo título, NODE-007 verifica la obligatoriedad de la descripción, NODE-008 asegura que la ubicación geográfica sea obligatoria, y NODE-009 confirma que la región es un campo requerido. Todas estas pruebas retornan código 422, evidenciando que el sistema previene la creación de nodos con datos incompletos.
- **Validaciones de Formato y Restricciones (NODE-010, NODE-011)** NODE-010 prueba que el sistema valida el formato de coordenadas geográficas, rechazando valores no numéricos. NODE-011 verifica la restricción de longitud máxima para el título (255 caracteres), asegurando que no se acepten títulos demasiado largos.
- **Operaciones Exitosas (NODE-012, NODE-015)** NODE-012 demuestra que el sistema puede actualizar nodos existentes con datos parciales, mientras que NODE-015 verifica que el sistema implementa soft delete marcando nodos como inactivos en lugar de eliminarlos físicamente.
- **Manejo de Errores en Operaciones (NODE-013, NODE-014, NODE-016)** NODE-013 y NODE-016 muestran que el sistema maneja UUIDs inexistentes retornando código 500, mientras que NODE-014 asegura que las validaciones de datos se aplican también en operaciones de actualización, rechazando títulos vacíos.

Los resultados demuestran que este módulo implementa un sistema de gestión de nodos que maneja correctamente operaciones CRUD completas con validaciones exhaustivas de datos de entrada.

5.1.4. Módulo mapa

ID	MÓDULO	Método/Función	Descripción	Datos de Entrada	Resultado Esperado
MAP-001	MAPA	GET /map/progress	Obtener progreso con token JWT válido	Header: 'Bearer valid_jwt'	Status 200, progreso del mapa

ID	MÓDULO	Método/Función	Descripción	Datos de Entrada	Resultado Esperado
MAP-002	MAPA	GET /map/progress	Obtener progreso con token externo válido	Header: 'Bearer usuario:password'	Status 200, progreso del mapa
MAP-003	MAPA	GET /map/progress	Obtener progreso sin autenticación	Sin header Authorization	Status 401, 'Authentication required'
MAP-004	MAPA	GET /map/progress	Obtener progreso con token inválido	Header: 'Bearer invalid_token'	Status 401, 'Authentication required'
MAP-005	MAPA	GET /map/progress	Obtener progreso con JWT sin user_id	Header: 'Bearer jwt_without_userid'	Status 401, 'Authentication required'
MAP-006	MAPA	GET /map/progress	Obtener progreso con token externo inválido	Header: 'Bearer usuario:wrongpass'	Status 401, 'Authentication required'

5.1.4.1. Análisis del módulo mapa

- **Consultas Exitosas (MAP-001, MAP-002)** MAP-001 demuestra que el sistema puede procesar tokens JWT válidos y retornar el progreso completo del mapa interactivo, incluyendo información de nodos, actividades y métricas de progreso del usuario. MAP-002 prueba la compatibilidad con tokens de autenticación externa en formato 'usuario:password', evidenciando que el sistema mantiene los mecanismos de autenticación para el acceso a información de progreso.

- **Manejo de Errores de Autenticación (MAP-003 a MAP-006)** MAP-003 muestra que el sistema requiere autenticación obligatoria, rechazando solicitudes sin header Authorization y retornando código 401. MAP-004 verifica que el sistema maneja tokens inválidos de manera consistente, retornando el mismo código de error para tokens malformados. MAP-005 prueba la validación de contenido de tokens JWT, asegurando que contengan el user_id necesario para procesar el progreso. MAP-006 establece que el sistema valida también tokens externos, rechazando credenciales incorrectas y manteniendo la seguridad del endpoint.

Los resultados permiten ver que el módulo de mapa implementa un sistema apropiado de consulta de progreso educativo que requiere autenticación obligatoria y maneja múltiples tipos de tokens de manera segura.

5.2. Pruebas de usabilidad y efectividad del aplicativo

5.2.1. Prueba de administrador

Se implementó una prueba de usabilidad contextual dirigida a un usuario experto con experiencia previa en la gestión de la ruta de empleabilidad, utilizando un protocolo de evaluación basado en tareas para validar la funcionalidad de creación de actividades. La sesión de evaluación fue documentada mediante grabación de pantalla y está disponible en el siguiente enlace: https://drive.google.com/file/d/1x09NYehFfyiE2WHZISyhHU3FD5dHgK6b/view?usp=sharing
El análisis del material audiovisual revela que el módulo administrativo cumple con los criterios de usabilidad establecidos mediante la implementación de un flujo de navegación intuitivo que permite al administrador acceder al panel administrativo gracias a la autenticación por roles, acceder al apartado de creación de actividades y destinos, configurar personalizadamente cada una de las actividades y validar que la actividad fue creada al acceder al apartado de mapa en la vista de los estudiantes. La efectividad funcional del sistema se demuestra mediante la completitud de la tarea sin errores de usuario o fallos del sistema, validando que la arquitectura de componentes reutilizables y el sistema de configuración basado en JSON operan correctamente para la creación de actividades personalizadas.

5.2.2. Prueba de estudiante

Se ejecutó una prueba de usabilidad con usuarios finales dirigida a un estudiante universitario utilizando un protocolo de evaluación de efectividad que midió la completitud de tareas y la satisfacción del usuario. La sesión de evaluación fue documentada mediante grabación de pantalla y está disponible en el siguiente enlace: https://drive.google.com/file/d/1Qs3o6fiDtDw3sOaxRCKNz3sOQQxOei_j/view?usp=sharing
De acuerdo al material anterior, se puede apreciar una prueba muy corta donde el estudiante resuelve la actividad creada por el administrador de forma éxitosa y se puede la motivación que consigue por resolver la prueba en su totalidad. De esta forma se logra evaluar la efectividad de la actividad creada, y que a pesar de ser corta, mantiene la atención del estudiante.

Capítulo 6

Conclusiones

6.1. Conclusiones

En este proyecto el objetivo era diseñar y estructurar una plataforma web educativa gamificada que transformara la experiencia visual y de aprendizaje del curso 'Ruta de Empleabilidad' de la Pontificia Universidad Javeriana Cali. Al final, se logró desarrollar una aplicación que responde de manera destacable a las necesidades identificadas en el curso tradicional, demostrando que la gamificación y las tecnologías web modernas pueden revolucionar la forma en que los estudiantes interactúan con el contenido educativo.

De tal forma que mediante esta serie de experimentos y desarrollos se logró mejorar la experiencia visual con elementos gamificados en entornos educativos, transformando una experiencia estática y monótona en un viaje interactivo y motivador que simula situaciones del mundo laboral real. También se logró evidenciar las diferencias principales entre los sistemas tradicionales de gestión de aprendizaje (LMS) y las aplicaciones web personalizadas, siendo los primeros sistemas rígidos con limitaciones de diseño y los segundos plataformas flexibles que permiten implementar experiencias completamente personalizadas y adaptadas a las necesidades específicas del dominio educativo.

Otro punto notorio que se evidencia en el desarrollo del proyecto es el cambio de enfoque realizado en la arquitectura del sistema, optando por implementar Clean Architecture en el backend y Screaming Architecture en el frontend, representando los datos y la lógica de negocio de una forma totalmente diferente a los sistemas tradicionales, pasando de una estructura técnica genérica a una organización que comunica inmediatamente el propósito y dominio específico del sistema educativo.

En conclusión, el proyecto proporciona una comparativa entre los sistemas tradicionales de gestión de aprendizaje y las aplicaciones web gamificadas personalizadas, pero más allá de ello permite darse cuenta del potencial que tiene la gamificación en general para casi todo tipo de contextos educativos donde se requiera motivación y participación activa de los estudiantes, pudiendo transformar experiencias de aprendizaje estáticas en aventuras interactivas que para los estudiantes serían prácticamente imposibles de experimentar en entornos tradicionales y ahorrando tiempo, recursos y esfuerzo al automatizar procesos administrativos complejos.

Como opinión personal, puedo decir que este proyecto realmente fue un reto bastante grande que en un principio se minimizó mucho su magnitud. Al ser un trabajo para una sola persona terminó siendo bastante complejo, enfrentando distintos impedimentos tanto de conocimiento técnico, limitaciones de tiempo y desafíos de comunicación con diferentes stakeholders de la universidad. Sin embargo, se hizo lo mejor posible con los recursos disponibles y quedan muchos aprendizajes valiosos sobre el desarrollo de aplicaciones educativas, la implementación de arquitecturas limpias y la importancia de la experiencia de usuario en entornos gamificados. Realmente se generó cariño

por el proyecto al ver cómo una idea inicial se transformaba en una herramienta que podría impactar positivamente la experiencia de aprendizaje de cientos de estudiantes, demostrando que la tecnología bien aplicada puede hacer una diferencia significativa en la educación universitaria.

6.2. Futuro trabajo

Como trabajo futuro, se plantea la expansión y evolución de la plataforma educativa gamificada hacia un ecosistema más completo y diversificado. Se contempla la integración de nuevas actividades interactivas que aprovechen tecnologías emergentes, como entrevistas en tiempo real con inteligencia artificial que simulen situaciones laborales auténticas y sistemas de evaluación basados en machine learning que proporcionen retroalimentación personalizada a los estudiantes.

El proyecto tiene el potencial de evolucionar hacia una solución educativa integral que trascienda el curso específico de empleabilidad, convirtiéndose en una plataforma general que pueda ser adoptada por otros departamentos de la universidad para diferentes cursos y programas académicos. Esta expansión interna podría servir como base para desarrollar una solución educativa que pueda ser comercializada a otras instituciones educativas, transformando la iniciativa en una oportunidad de emprendimiento que contribuya a la modernización de la educación superior en Colombia y Latinoamérica.

Bibliografía

1. Deterding, S., Dixon, D., Khaled, R., and Nacke, L. (2011). From game design elements to gamefulness: defining "gamification". In Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments (pp. 9-15). ACM.

2. Alsadoon, Hamadah. (2023). The Impact of Gamification on Student Motivation and Engagement: An Empirical Study. Dirasat: Educational Sciences. 50. 386-396. 10.35516/edu.v50i2.255.

3. J. Hamari, J. Koivisto and H. Sarsa, "Does Gamification Work? – A Literature Review of Empirical Studies on Gamification,"2014 47th Hawaii International Conference on System Sciences, Waikoloa, HI, USA, 2014, pp. 3025-3034, doi: 10.1109/HICSS.2014.377. keywords: Context;Psychology;Databases;Educational institutions;Games;Libraries;gamification;motivational affordance;persuasive technology;motivation;HCI,

4. Kapp, Karl. (2012). The gamification of learning and instruction: Game-based methods and strategies for training and education. San Francisco, CA: Pfeiffer.

5. Ortiz-Colón, A. M., Jordán, J., and Agredal, M. (2018). Gamificación en educación: una panorámica sobre el estado de la cuestión. Educação e pesquisa, 44, e173773.

6. Zainuddin, Zamzami and Chu, Samuel and Shujahat, Muhammad and Perera, Corinne. (2020). The impact of gamification on learning and instruction: A systematic review of empirical evidence. Educational Research Review. 30. 100326. 10.1016/j.edurev.2020.100326.

7. A. M. Toda et al., .ª Taxonomy of Game Elements for Gamification in Educational Contexts: Proposal and Evaluation,"2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT), Maceio, Brazil, 2019, pp. 84-88, doi: 10.1109/ICALT.2019.00028.

8. Park, J. Y. (2008). iLED: interactive learning experience design. Journal of Online Learning and Teaching, 4(3), 357-370.

9. Pianucci, I., and Tenutto, M. (2020). Potencial de la tecnología digital para la retroalimentación efectiva en diferentes momentos evaluativos. In Congreso Iberoamericano (pp. 1-11).

10. Gaviria, B. F., Narváez, F. U., and Bustos, J. J. (2009). Aplicación web para evaluación formativa universitaria basada en competencias. Revista Educación en Ingeniería, 4(8), 1-12.

11. Reed, J. A., and Afjeh, A. A. (1998). Developing interactive educational engineering software for the World Wide Web with Java. Computers and Education, 30(3-4), 183-194.

12. Polito, G., and Temperini, M. (2021). A gamified web based system for computer programming learning. Computers and Education: Artificial Intelligence, 2, 100029.

13. Sanabria, L. F. M., and Ordoñez, L. M. V. (2019). EDUMAT: herramienta web gamificada para la enseñanza de operaciones elementales. Campus virtuales, 8(2), 9-17.

14. de-la-Mata-Moratilla, S., Castillo-Martinez, A., and Gutierrez-Martínez, J. M. (2021). Plataforma web gamificada para el entorno universitario. In ATICA2021: Aplicación de Tecnologías de la Información y Comunicaciones Avanzadas y Accesibilidad (pp. 448-455). Universidad de Alcalá.

15. Isha, V. P. (2023, November). An Approach to Clean Architecture for Microservices Using Python. In 2023 7th International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS) (pp. 1-5). IEEE.

16. Lano, K., Tehrani, S. Y. (2023). Introduction to Software Architecture: Innovative Design Using Clean Architecture and Model-driven Engineering. Springer Nature.

17. "¿Qué es una base de datos relacional?," Oracle.com, 2014. https://www.oracle.com/co/database/what-is-a-relational-database.

18. Ehsan, A., Abuhaliqa, M. A. M., Catal, C., and Mishra, D. (2022). RESTful API testing methodologies: Rationale, challenges, and solution directions. Applied Sciences, 12(9), 4369.

19. Barabanov, A., and Makrushin, D. (2020). Authentication and authorization in microservice-based systems: survey of architecture patterns. arXiv preprint arXiv:2009.02114.

20. R. Ollila, N. Mäkitalo and T. Mikkonen, 'Modern Web Frameworks: A Comparison of Rendering Performance,' in Journal of Web Engineering, vol. 21, no. 3, pp. 789-813, May 2022, doi: 10.13052/jwe1540-9589.21311.

21. Taylor, L. (2018). Educaplay. The School Librarian, 66(4), 214.

22. Gardner, B. S. (2011). Responsive web design: Enriching the user experience. Sigma Journal: Inside the Digital Ecosystem, 11(1), 13-19.

23. Martin, R. C. (2017). Clean architecture: a craftsman's guide to software structure and design (Chapter 21). Prentice Hall Press.

24. Martin, R. C. (2017). Clean architecture: a craftsman's guide to software structure and design (Chapter 22). Prentice Hall Press.

25. Goldberg, J. (2022). Learning TypeScript. .º'Reilly Media, Inc.".

26. Gerchev, I. (2022). Tailwind CSS. SitePoint Pty Ltd.

27. Bayer, M. (2012). SQLAlchemy. The architecture of open source applications, 2.

28. Ireland, C., Bowers, D., Newton, M., and Waugh, K. (2009). Understanding object-relational mapping: A framework based approach. Int J Adv Softw, 2, 202-16.

29. Worsley, J., and Drake, J. D. (2002). Practical PostgreSQL. .ºReilly Media, Inc.".

30. Anderson, C. (2015). Docker [software engineering]. IEEE software, 32(3), 102-c3.

31. Kore, P. P., Lohar, M. J., Surve, M. T., and Jadhav, S. (2022). API Testing Using Postman Tool. International Journal for Research in Applied Science and Engineering Technology, 10(12), 841-43.

32. Sabbag Filho, N. (2024). Best practices for using DTOs (Data Transfer Objects) in a clean architecture. Leaders Tec, 1(26).

33. Narayanan, P. K. (2024). Getting Started with Data Validation using Pydantic and Pandera. In Data Engineering for Machine Learning Pipelines: From Python Libraries to ML Pipelines and Cloud Platforms (pp. 163-196). Berkeley, CA: Apress.