PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Unidad Académica de Formación Técnica y Tecnológica - PUCE TEC

SISTEMA PARA LA GESTIÓN DE CONTENIDO Y SERVICIOS EDUCATIVOS "ENGLISH FOR EVERYONE"

Proyecto de titulación previo a la obtención del título de: Tecnología Superior en Desarrollo de Software

Autores: Martín Ismael Moreira Carbo y Maykel Said Navarrete Maldonado

Tutor: Carlos Miguel Cárdenas Riofrio

Quito, Ecuador

2026

Dedicatoria

Said Navarrete:

Dedico el presente trabajo a mi familia, que ha sido el soporte de todas mis metas y objetivos realizados, han sido mi fuente de inspiración y motivación constante. Agradezco mucho sus palabras de aliento y haber creído en mi durante este trayecto.

Para Geanella, mi hermanita, a quien considero capaz de cualquier cosa, pues en ella veo una mujer firme y confiable. Espero que esto le demuestre que ella no tiene ningún límite siempre que se lo proponga, especialmente siendo la señorita virtuosa que tanto me llena de orgullo día a día.

Agradezco también a Iván Vivas, que ha sido un compañero para mi familia en tiempos difíciles y ahora un pilar fundamental, ha mostrado ser un hombre ejemplar, dedicado a las personas que ama y me ha enseñado más de lo que un padre pudo haberme mostrado acerca de cuidar a sus seres queridos.

A mi madre, que es la luz de mi vida, le doy mis más profundos agradecimientos, pues ella ha sido quien me ha enseñado todo lo que soy hoy en dia, me ha dispuesto de más de lo necesario con su amor inagotable y sacrificado por su familia. Este logro no sería posible de no ser por su constancia y acompañamiento a lo largo de mi existencia.

También a mi compañero, Martín Moreira, que ha sido un buen amigo, tanto en la carrera, como por fuera de ella, que hizo que este trabajo sea posible.

Finalmente, quisiera agradecer a mi hermana, Allison Navarrete, a quien estimo y quiero con todo mi corazón, ella ha sido la base de mi formación, porque sin su apoyo, no hubiera llegado tan lejos. Le deseo mis mayores bendiciones, por cuánto ha demostrado ser siempre una mujer fuerte, responsable y con ansias de autosuperación. Gracias, porque el día de hoy este trabajo no es solo mío, sino también el fruto de todo tu esfuerzo, has mostrado ser una gran amiga y espero que esto retribuya una pequeña parte de todo tu sacrificio. Te quiero con toda mi alma.

Martín Moreira:

Dedico este trabajo con gran cariño a mi familia, a mi madre Elisa Carbo quien desde siempre ha formado los cimientos de la persona que soy hoy, proporcionando respaldo e inspiración inigualable. Sin importar las dificultades que nos arroje la vida, no he conocido mujer más fuerte, capaz y amorosa. No tengo palabras que hagan justicia a la inmensidad de tu amor y tu paciencia. A mi hermana Andrea que ha sido compañera, amiga, y una de las pocas personas que aguanta los monólogos y discursos sobre un tema u otro que a veces me consumen. He visto como has enfrentado dificultad tras dificultad, con quejas y objeciones, pero sin ceder. Quiero que sepas que gracias a tu ejemplo repongo fuerzas y siento que puedo enfrentarme a los obstáculos que vengan.

A mi abuelita Rosa Elisa quien quizás indirectamente y sin saberlo me inspiró a tomar acción y volverme a poner en pie. Aunque no recuerdes todas nuestras aventuras sabes que toda la vida hemos sido compañeros.

A mi padre Juan Moreira a quien tengo en mente todos los días y me asombro cuando descubro a través de mis propias experiencias lo duro que trabajaste toda tu vida y el gran impacto que tuviste sobre la gente a tu alrededor.

A Tamara Mancero, amiga querida para toda mi familia que nos ha brindado apoyo, consejo y ánimo en tantas ocasiones. No puedo pensar lo diferente que serían nuestras vidas si las relaciones de amistad formadas hace tantos años no existieran.

A Said Navarrete, amigo. Al iniciar esta carrera no esperaba encontrarme con una persona con quien pueda compartir tantos intereses, historias y opiniones. No deja de sorprenderme tu voluntad de trabajo y el esfuerzo que demuestras día a día.

Quiero también agradecer a todos mis primas, primos, tías, tíos y familia extendida con quienes hemos compartido tantos eventos, risas y llantos a lo largo de los años. Sé que siempre puedo contar con ustedes y espero sepan que puedan contar conmigo también.

Tabla de contenidos

DECLARACIÓN Y AUTORIZACIÓN

Yo, Navarrete Maldonado Maykel Said con C.I. 1750966093 autor(a) del trabajo de titulación, titulado: “Sistema para la gestión de contenido y servicios educativos – English for Everyone”, previa a la obtención del título de Tecnólogo en Desarrollo de Software en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 4 de febrero del 2026

Maykel Said Navarrete Maldonado

C.I. 1750966093

DECLARACIÓN Y AUTORIZACIÓN

Yo, Moreira Carbo Martín Ismael con C.I. 1715115638 autor(a) del trabajo de titulación, titulado: “Sistema para la gestión de contenido y servicios educativos – English for Everyone”, previa a la obtención del título de Tecnólogo en Desarrollo de Software en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 4 de febrero del 2026

Martín Ismael Moreira Carbo

C.I. 1715115638

Introducción

La transformación digital en el sector educativo ha roto barreras de costos y alcance para diversos individuos en quienes está la motivación de desarrollar nuevas habilidades. No obstante, pese a las variadas ofertas disponibles en cuanto al idioma, los canales existentes, han afectado seriamente a las instituciones tradicionales y profesionales independientes, quienes buscan sumarse a estos nuevos medios para ofrecer sus servicios sin las limitantes físicas y a su vez aspirar una audiencia más amplia.

El presente proyecto “English For Everyone” un emprendimiento de enseñanza del idioma inglés en Quito, surge como una respuesta a la necesidad de compartir y distribuir material para la enseñanza del inglés. A pesar de los esfuerzos del proyecto de brindar contenido educativo de alta calidad, la dependencia del docente a herramientas con alcances dispersos supone una carencia de integración tecnológica. Esta falta de una plataforma centralizada no solo fragmenta la experiencia del estudiante, sino que también limita la capacidad del docente de proveer un servicio eficiente.

Conforme se desarrolle el presente trabajo, se describirán las etapas para la realización de este, incluyendo el levantamiento de requisitos, el diseño de la arquitectura. Posteriormente se hará énfasis en el proceso de implementación de la solución de la plataforma, así como pruebas que validen la seguridad y estabilidad del producto final. Por última instancia, se realizará el análisis y la justificación de las herramientas empleadas, así como el impacto que la solución propuesta tiene sobre los servicios de “English for Everyone”

Capítulo I

Levantamiento de Requisitos y Diseño del Sistema

En este capítulo se describe el proceso llevado a cabo para la identificación y recopilación de las necesidades del sistema web requerido por “English for Everyone”. Este análisis permite la definición precisa de los requisitos funcionales y no funcionales, los cuales sirven como base en el diseño de la arquitectura y el desarrollo de la plataforma, asegurando la distribución del material educativo.

1.1 Levantamiento de requisitos

En la definición de las necesidades específicas del sistema, se llevó a cabo una entrevista a Felipe Pérez, dueño del emprendimiento English for Everyone. El diálogo permitió distinguir los puntos críticos en la gestión actual de sus servicios, destacando principalmente la dificultad de proporcionar material didáctico y la organización de este. A partir de la información recolectada se estableció el flujo que debería llevar el sistema y los principales requisitos a solventar.

Requisitos funcionales:

- RF01 - Gestión de autenticación: La aplicación debe permitir el registro e inicio de sesión de usuarios mediante correo electrónico y contraseña.
- RF02 - Módulo de administración de contenidos: El docente debe poder cargar, editar y eliminar el material didáctico en los formatos establecidos (jpg, pdf, docx, png).
- RF03 – Organización y categorización de contenidos: El docente debe tener la capacidad de crear categorías temáticas con imágenes representativas, a fin de agrupar los recursos educativos.
- RF04 - Internacionalización de textos: La aplicación debe presentar una interfaz con soporte bilingüe en inglés y español aplicando el uso de

herramientas de internacionalización interactivas que permitan que los usuarios seleccionen el lenguaje de su preferencia.

- RF05 - Separación de roles para control de accesos: Los usuarios de la aplicación deben ser asignados un rol de “usuario” automáticamente durante el registro de cuenta, protegiendo de esta manera las funcionalidades administrativas de la aplicación.
- RF06 - Visualización del contenido: Los visitantes podrán observar el catálogo de categorías y acceder a los recursos didácticos asociados a cada una de las temáticas.

Requisitos no funcionales:

- RNF01 - Página para rutas no encontradas: Los usuarios deben ser presentados con una página de rutas no encontradas personalizada con código de error HTTP 404 para proteger accesos no autorizados a contenidos no implementados o no autorizados.
- RNF02 – Single Page Application: La página debe ser implementada siguiendo una arquitectura de aplicación de una sola página (SPA) implementada en React para minimizar el tiempo de respuesta y carga entre vistas.
- RNF03 – CORS (Cross-Origin Resource Sharing): La aplicación define una política de uso compartido de recursos entre orígenes que restrinja el acceso a la API únicamente a los dominios autorizados, garantizando la seguridad en la comunicación entre el frontend y el backend.
- RNF04 - Validación de datos: La aplicación implementa un sistema de validación de datos para sanear y validar entradas en todas las rutas de escritura de la API.

- RNF05 - Limitación de Velocidad: La aplicación debe integrar un método de protección contra ataques de fuerza bruta y abuso de API, salvaguardando la estabilidad y funcionalidad del sistema.
- RNF06 – Mantenibilidad y escalabilidad: La aplicación aplica buenas prácticas de desarrollo, código documentado y patrones de diseño para facilitar actualizaciones y mantenimiento.
- RNF07 – Logs y gestión de errores: La aplicación debe integrar funcionalidades de registro de eventos y errores, con el fin de monitorear y agilizar la depuración de fallos y reconocer ataques.
- RNF08 - Diseño responsive: La aplicación debe presentar un diseño responsive adaptable para usuarios accediendo a través de dispositivos móviles.

1.2 Diseño del Sistema

La presente sección presentará el diseño integral del sistema desde su concepción arquitectónica hasta los detalles de implementación, despliegue y mantenimiento. Se explica cómo se estructuran y relacionan sus componentes principales para cumplir con los requisitos funcionales y no funcionales establecidos. También se incluyen prototipos y mockups elaborados durante la fase de diseño los cuales fueron utilizados para validar la propuesta con el cliente y guiar todo el proceso de desarrollo.

1.2.1 Arquitectura general

La aplicación se concibió siguiendo un modelo de arquitectura cliente-servidor. La página web forma parte de la capa de presentación que cumple la responsabilidad de facilitar la interacción con los usuarios y el consumo de la API a través de peticiones HTTP. Se encarga también de la validación de formularios y el manejo de ingreso de información.

El backend, desarrollado en Express.js constituye la capa de aplicación del sistema que se encarga del manejo de la lógica de negocio, la implementación de endpoints

RESTful para la comunicación con el frontend, la gestión de sesiones y autenticación de usuarios y otras implementaciones de seguridad como el cifrado de contraseñas.

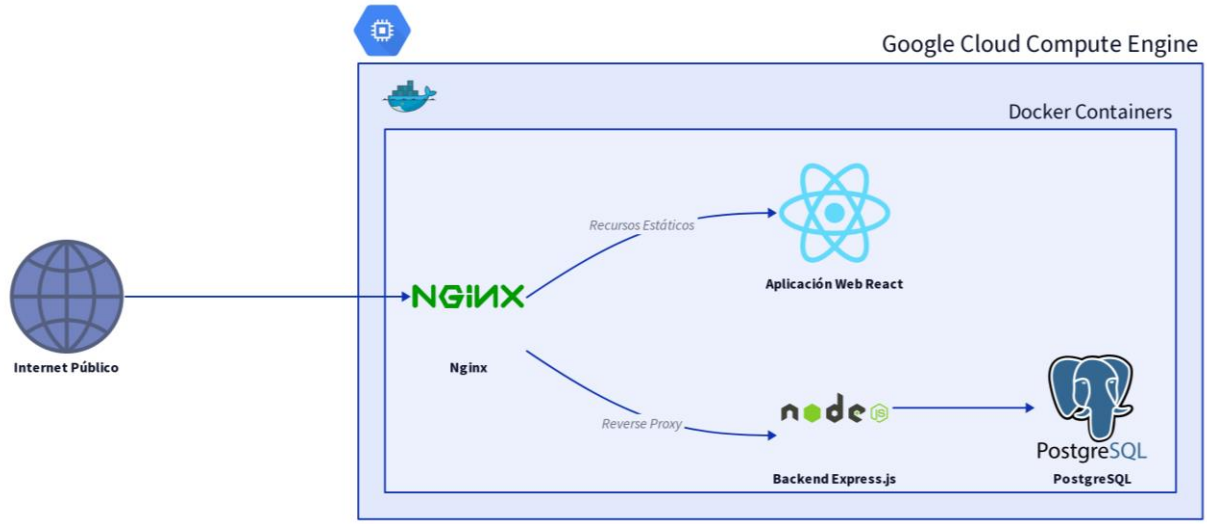
Finalmente, la capa de datos se encuentra desarrollada utilizando una base de datos relacional implementada en PostgreSQL con la responsabilidad de garantizar la persistencia de datos de usuarios y el contenido distribuido por la aplicación. La capa de datos se encuentra vinculada a la capa de aplicación a través de Prisma, un cliente de Mapeo Objeto-Relacional (ORM) para asegurar la mantenibilidad de la aplicación y la protección de la información.

1.2.2 Modelado del sistema

Durante la fase de planteamiento y diseño se elaboraron diagramas arquitectónicos para representar el flujo de trabajo de la aplicación y las interacciones entre sus componentes.

Figura 1

Diagrama general de arquitectura de la aplicación.



1.3 Herramientas utilizadas

Durante la etapa de desarrollo de la aplicación se seleccionaron y emplearon las siguientes tecnologías y herramientas:

Visual Studio Code: Como editor de texto para el desarrollo de todo el código de la aplicación.

React: Constituye la librería de componentes reutilizables que fueron elaborados para el diseño visual de la interfaz de usuario o frontend.

Node.js: Para la implementación de la lógica de negocio de la aplicación, constituyendo un API RESTful que recibe las peticiones enviadas por parte del frontend, y almacenando la información en la base de datos.

Better-Auth: Better-auth es un “framework de autenticación y autorización para TypeScript” (Better-Auth, s. f.) que se adapta a una multitud de implementaciones de software. Se usó para el desarrollo del sistema de autenticación y manejo de roles de la aplicación.

Google Compute Engine: Parte de la plataforma de servicios en la nube proporcionados por Google, Compute Engine es una plataforma de IaaS (Infraestructura como Servicio) que permite crear máquinas virtuales accesibles a través de internet. Fue empleado para desplegar la aplicación y sus componentes en la internet pública.

PostgreSQL: Como la base de datos relacional empleada para almacenar la información de usuarios, sesiones, y recursos digitales disponibles en la aplicación.

Postman: Herramienta de pruebas de APIs que fue utilizada durante la etapa de desarrollo para verificar el correcto funcionamiento de la aplicación.

Prisma ORM: Herramienta de Mapeo Objeto-Relacional que simplifica el acceso e interacciones con la base de datos de manera rápida y segura.

D2Lang: Lenguaje interactivo para la elaboración de diagramas arquitectónicos de sistemas.

Docker: Como plataforma de contenedorización de aplicaciones, fue empleado para el rápido despliegue y reproducibilidad de los componentes funcionales de la aplicación tanto durante las fases de desarrollo como despliegue.

TailwindCSS: Framework de CSS para facilitar la elaboración de estilos en el frontend, permite rápida iteración y experimentación para una experiencia de aplicación más amigable.

1.4 Metodología

Para el desarrollo del proyecto se seleccionó Scrum como marco de trabajo ágil, fundamentado en las necesidades del equipo de desarrollo. “Scrum ayuda a los equipos a abordar proyectos complejos dividiendo el trabajo en ciclos iterativos más pequeños llamados sprints.” (Atlassian, s. f.)

Esta estructura facilita la detección temprana de problemas y permite realizar ajustes durante el desarrollo, lo cual reduce los riesgos asociados con desviaciones tardías del plan original.

Scrum además promueve una comunicación transparente entre los miembros del equipo gracias a sus ceremonias de revisión y retrospectivas. Esto impulsa la coordinación del equipo y fomenta la mejora continua durante los ciclos de desarrollo iterativos.

1.4.1 Estructura y roles del equipo

Durante la etapa de planificación del proyecto se determinaron los siguientes roles para el marco de trabajo:

Product Owner: Felipe Pérez (English for Everyone)

Su rol implicó definir y priorizar las funcionalidades de la aplicación según las necesidades de su emprendimiento para asegurar que el equipo de desarrollo trabaje en las tareas de mayor valor y entregue un producto que se apegue a las metas de English for Everyone.

Scrum Master: Carlos Cárdenas (PUCE)

Como docente tutor del presente proyecto, se encargó de supervisar el proceso de desarrollo y documentación de la aplicación, eliminando impedimentos y verificando que cumpla con los requisitos establecidos durante la planificación.

Dev Team: Martín Moreira y Said Navarrete

El equipo se encargó de elaborar, codificar y entregar todos los incrementos de la aplicación durante los ciclos de desarrollo. También fueron responsables de la realización de pruebas, implementación y despliegue del producto.

1.5 Limitantes y alcance funcional de la aplicación

En consideración de la estabilidad y funcionalidad de la aplicación web, es importante resaltar los siguientes aspectos y condiciones:

- Acceso a endpoints de creación de contenido: El acceso a los endpoints de categorización y carga de recursos está limitado exclusivamente al usuario con rol de administrador, denegando cualquier intento de uso por parte de usuarios no autorizados.
- Registro e Inicio de Sesión: La autenticación y registro de usuarios está únicamente disponible para credenciales correspondientes a correo electrónico y contraseña gestionados con cifrado y tokens de autenticación.
- Carga y subida de archivos: Con el objetivo de prevenir saturación en el servidor y optimizar la velocidad, la subida de archivos, ya sea en formato de imagen o documento, está limitada a 25 megabytes por carga.
- Disponibilidad y conectividad: La aplicación requiere conexión a internet estable para su funcionamiento, debido a su despliegue en servicios de nube.

Capitulo II

Construcción de la aplicación

El presente capítulo desarrolla el proceso realizado en la construcción de la aplicación web, detallando de manera clara y organizada las fases de desarrollo e implementación de cada una de las funcionalidades descritas, así como las tecnologías empleadas para el desarrollo.

2.1 Estructura general del desarrollo

El proyecto empleó una arquitectura cliente-servidor, que se describe como un patrón de diseño que diferencia las responsabilidades para cada componente funcional de la aplicación web. Por el lado del cliente, este se encarga de solicitar, enviar y recibir información a través de una interfaz simplificada para el uso del usuario; por otro lado, el servidor abstrae toda la lógica del sistema, así como el flujo de datos, optimizando las operaciones y reduciendo el tiempo de espera.

Entre las tecnologías aplicadas para la construcción del proyecto, se incluye a React para cubrir las necesidades de una interfaz responsiva e intuitiva, incluyendo la división de las funcionalidades en componentes específicos y especializados. Este apartado está alojado en un contenedor Docker corriendo dentro de sí un servidor con Nginx, encargado de redirigir el tráfico de las solicitudes hacia el backend y evitando exposiciones innecesarias.

Por otro lado, el backend se desarrolló empleando TypeScript y la ayuda de Express.js, que es un framework que permite el desarrollo de APIs siguiendo el estándar RESTful, asegurando así que el procesamiento de la lógica del negocio sea robusta y escalable. Para el apartado de datos, se implementó una base de datos PostgreSQL junto con PrismaORM para el modelado y acceso a datos, esto con el fin de simplificar la manipulación de los datos y la integridad de los mismos en cada operación.

2.2 Organización del Proyecto

La aplicación web, se encuentra estructurada en un esquema de microservicios contenerizados con ayuda de Docker, facilitando su despliegue e interacción en una red gestionada con Docker Compose.

Contenedor Frontend (Página Web – React + Nginx)

- Autenticación y recuperación de usuario administrador mediante Better-Auth.
- Página de inicio con los contenidos y categorías establecidos por el docente
- Interfaz pública de acceso a recursos educativos.
- Panel de administración
- Formulario de creación de apartados de contenido y carga de recursos.
- Proxy Inverso y acceso a archivos estáticos moderado por Nginx

Contenedor Backend (API – Typescript + Express)

- Administración de usuarios y roles con endpoints de better-auth
- API RESTFul para la creación y gestión de recursos educativos y categorías disponibles.
- Apartado de recuperación de usuario y contraseña con Resend.
- Uso de PrismaORM para el acceso a datos y conexión con la base de datos PostgreSQL.

Contenedor Base de Datos (PostgreSQL)

- Gestión de datos de usuarios, sesiones, cuentas e información relacionada a los recursos y categorías.
- Configuración de volúmenes para persistencia de datos

2.3 Desarrollo del Frontend

El frontend de la aplicación constituye la interfaz presentada a los usuarios finales del producto y sigue una estructura basada en componentes, elementos pequeños reutilizables que forman los bloques de interacción y presentación.

Estos componentes fueron elaborados con la librería de JavaScript React y personalizados con el uso de TailwindCSS, un framework que permite aplicar estilos a los elementos de una aplicación web mediante clases, para minimizar el uso de archivos css independientes y promover la iteración constante y experimentación.

React además permite el uso de ReactRouter, una librería que provee de capacidades de enrutamiento entre las diferentes vistas o pantallas. Esto permite que el frontend se transforme en una aplicación de una sola página (SPA). Las SPA minimizan la carga del servidor y generan ventajas en cuanto a rendimiento y velocidad. Todo esto promueve una mejor experiencia para el usuario final.

Al ser una página web con el objetivo de promover el aprendizaje del idioma inglés, la aplicación fue planeada desde los cimientos para integrar una interfaz bilingüe que se adapte a las necesidades de los usuarios. Esto se consiguió a través de la integración de i18next, un framework de internacionalización para JavaScript. I18next permite definir “locales” o listas de cadenas de texto asociadas con un idioma, para poder actualizar de forma dinámica los contenidos de texto, tanto por detección automática del lenguaje del dispositivo que accede a la página como por input del usuario.

Una vez planteados los requisitos y criterios de aceptación de la interfaz, se procedió a preparar las principales vistas que la aplicación presenta, haciendo hincapié en la usabilidad y simplicidad del contenido. Se utilizó una paleta de colores reducida, determinada con aprobación del cliente, y se agregaron textos informativos acerca de los servicios ofrecidos por el emprendimiento English for Everyone y su proyecto “LinguaLab”.

En su implementación final, la aplicación web se encuentra distribuida en los siguientes componentes:

- Home: La vista principal de la aplicación, como landing page principal de los usuarios. Muestra información corta sobre English for Everyone y una descripción general de cada categoría de recurso la cual puede ser personalizada por el usuario administrador.
- Navbar: En la parte superior se muestra una barra de navegación que muestra enlaces para inicio de sesión y personalización del lenguaje de la página.
- El componente de inicio de sesión se despliega como un pop-up y permite el registro de usuarios, inicio de sesión y recuperación de contraseñas.
- Panel de administración: En el caso de que el usuario con sesión activa tenga el rol de administrador, el navbar mostrará un enlace al panel de administración el cual despliega opciones de personalización del contenido de la aplicación y permite que el administrador gestione las categorías y recursos que son presentados a los usuarios.
- Páginas de categoría: Cada categoría detallada en la página principal tiene una página de detalle donde se muestra todos los recursos de aprendizaje asociados con esta. Cada recurso muestra un enlace de descarga.
- Contenido no disponible: Se trata una página de uso general para contenido no disponible, esta muestra el código de error HTTP asociado (404) e informa a los usuarios. Permite además asegurar las rutas de acceso restringido y previene ataques de fuerza bruta a directorios, o de navegación forzada por agentes maliciosos.

Figura 2

Mockup página principal.

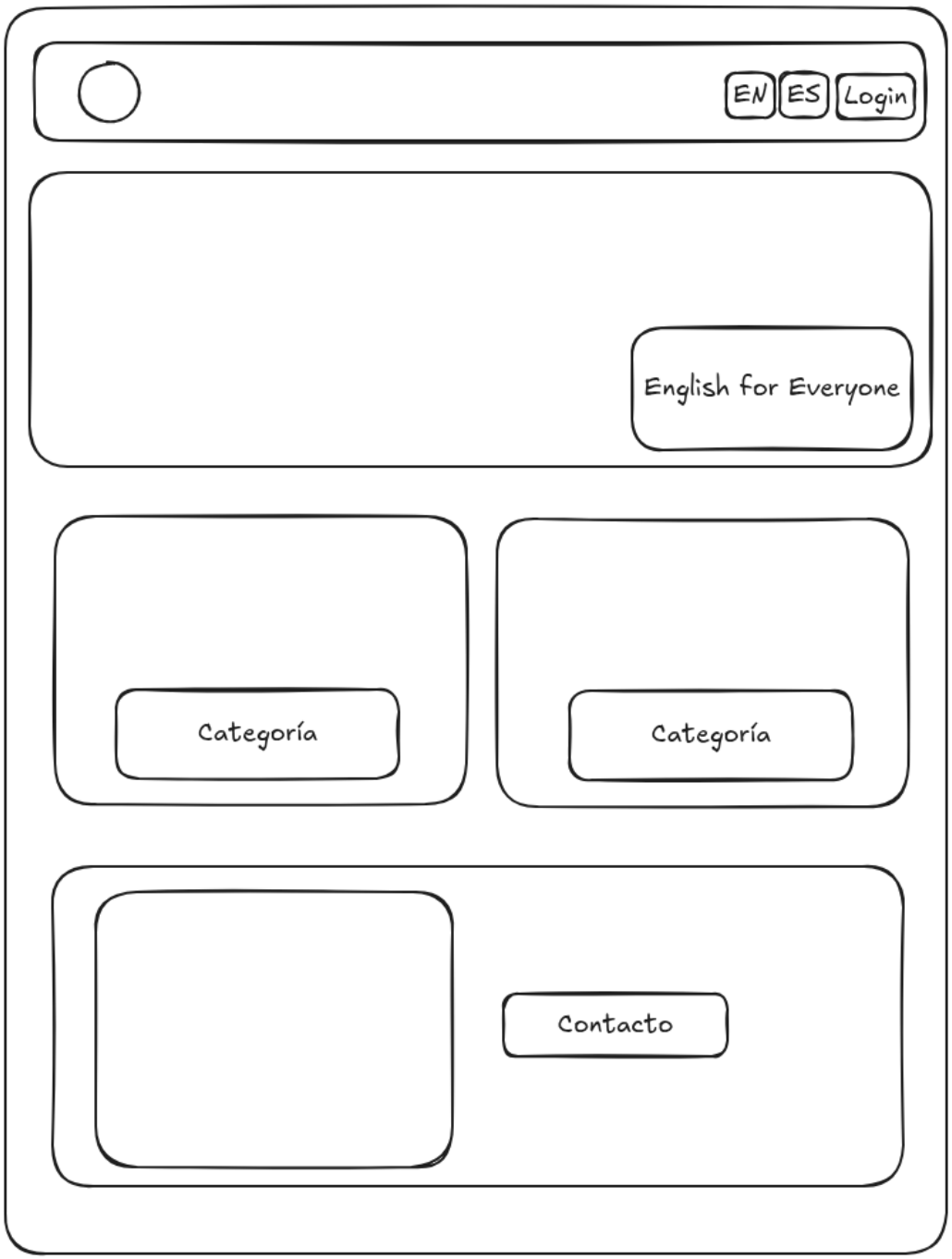


Figura 3

Mockup página de recursos

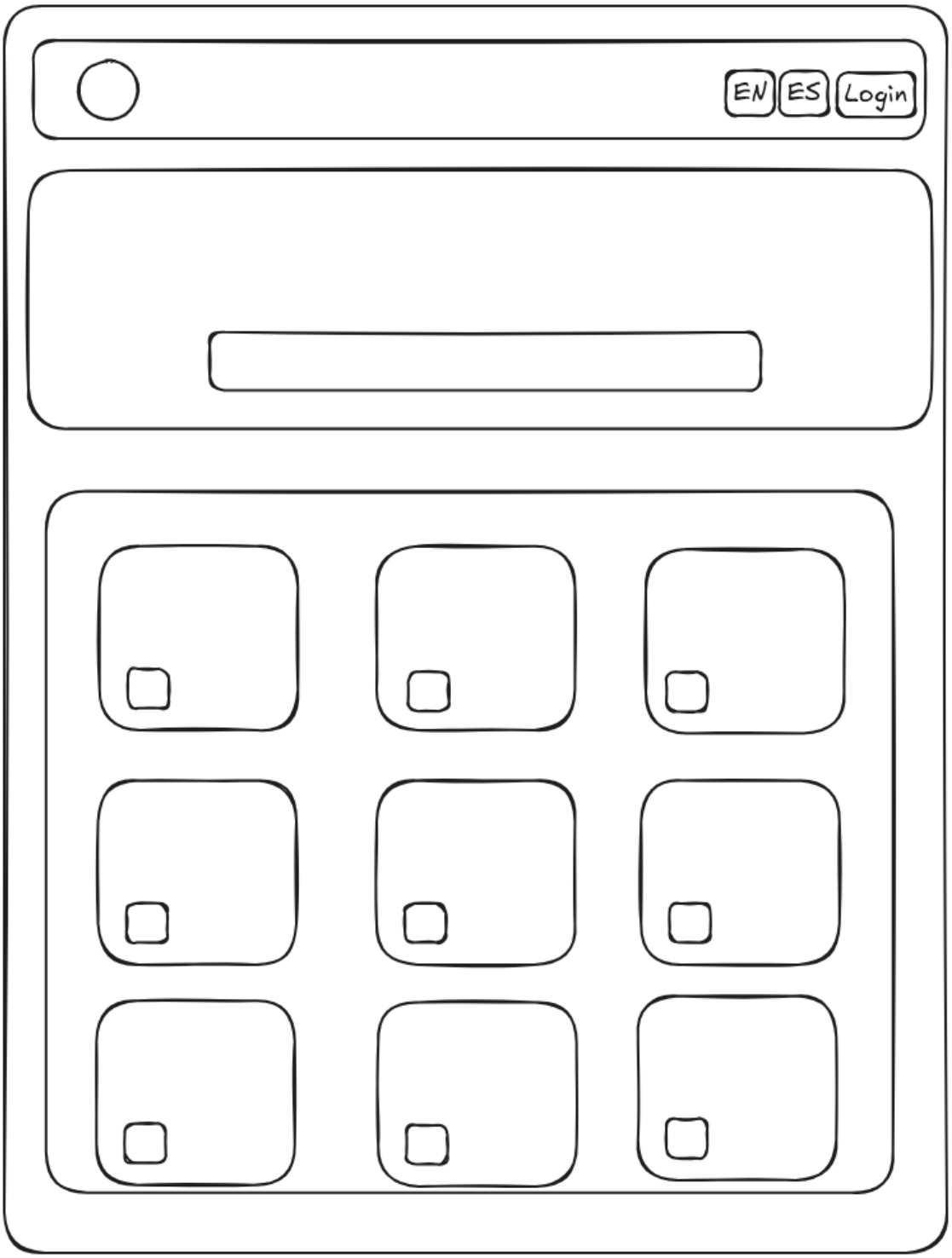
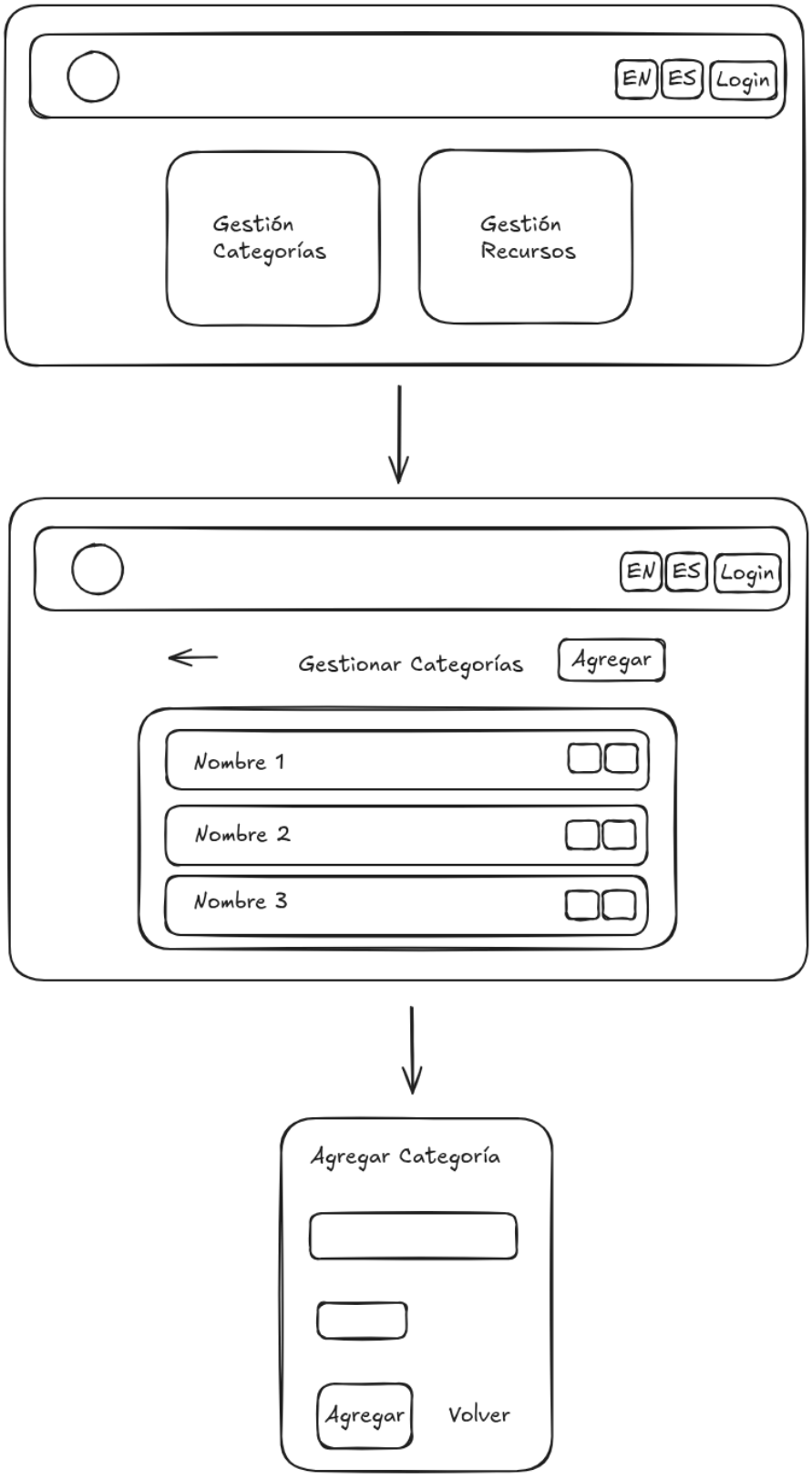


Figura 4

Flujo de uso panel administración



Para la mantenibilidad del código, los elementos con funciones o apariencia similares como formularios de ingreso de información y contenedores de tipo card conforman los componentes reutilizables. Se integra además elementos interactivos para la subida de archivos de estilo drag-and-drop, y validaciones de seguridad para prevenir vulnerabilidades de ingresos.

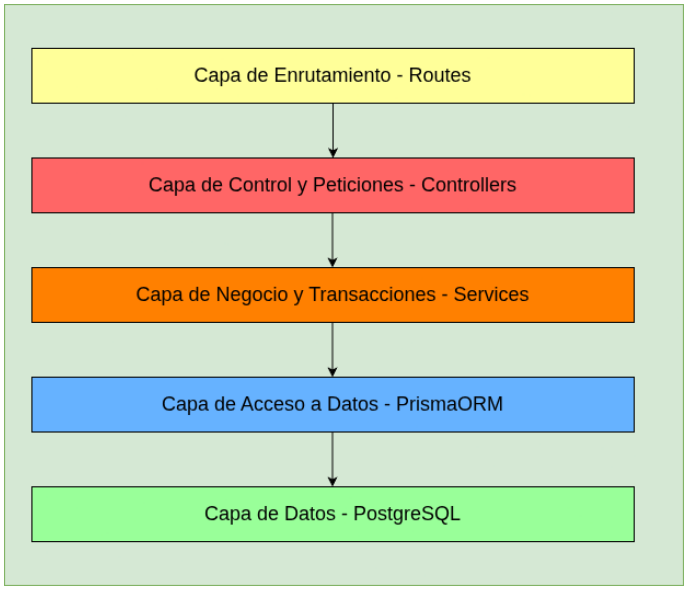
2.4 Desarrollo del Backend

La construcción del lado del servidor está cimentada en el uso de TypeScript y Node.js, con el objetivo de asegurar la integridad de los datos recibidos y procesados a través del tipado estático que proporciona TypeScript, así mismo, la implementación de interfaces ayuda a sanitizar las entradas y salidas de información al momento de construir la aplicación, reduciendo significativamente los riesgos que involucran los errores al momento de la ejecución de las operaciones del sistema.

Por otra parte, la implementación del framework Express.js recae en la simplificación de la construcción de la API REST, dado que define un estándar de trabajo al separar las responsabilidades del sistema en capas tal como se muestra en la siguiente figura.

Figura 5

Diagrama de arquitectura por capas



Esto asegura que la aplicación no comprometa operaciones, dando paso a una estructura mucho más sencilla de mantener y utilizar.

Uno de los puntos a resaltar, es el uso de Better Auth. Esta librería de javascript, abstrae toda la funcionalidad de autenticación y registro de usuarios con configuraciones mínimas y personalizadas, al mismo tiempo, define sus propias tablas en la base de datos y se encarga de la creación de tokens, por lo que se pudo establecer la combinación de correo electrónico y contraseña como credenciales de acceso rápidamente.

Tomando en cuenta todo lo mencionado anteriormente, la construcción del API recae en los endpoints expuestos para el acceso y manejo de la información de categorías que se encargan de ordenar los recursos. En cuanto al apartado de creación de recursos, sus endpoints son el núcleo operativo de la aplicación llevando a cabo las siguientes responsabilidades.

- Obtener los datos enviados por el usuario administrador desde la interfaz de creación de recursos y su validación por campos.
- Almacenamiento en la base de datos PostgreSQL.
- Procesar archivos y documentos tipo pdf, docx y diversos formatos de imagen, generando rutas absolutas y nombres únicos de archivos para evitar duplicados.
- Acceso a los archivos mediante llamadas http, organizando los recursos en lotes para evitar sobrecargas y agilizar las operaciones a nivel de la capa de datos.
- Asociación de los recursos a las categorías puestas a disposición por el docente.
- Acceso exclusivo a solicitudes PUT, DELETE, POST al administrador.

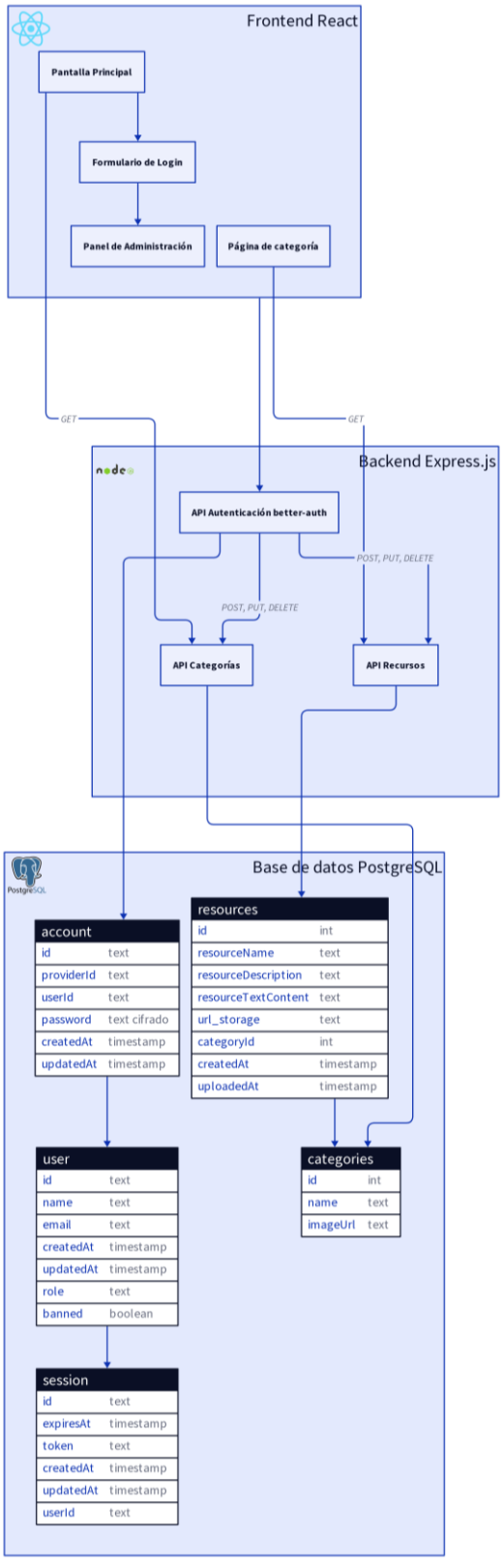
Cabe resaltar que la responsabilidad de carga de archivos recae sobre un middleware construido utilizando el paquete multer, restringiendo el formato de archivos y estandarizando el tamaño del archivo para reducir el estrés del lado del servidor. En particular, estas limitaciones no se reducen únicamente a preservar la operatividad de la aplicación, sino que fortalecen la seguridad al reducir los posibles vectores de ataque con la implantación de scripts y archivos maliciosos.

Dada la naturaleza de estas operaciones, se implementó un sistema de auditoría basado en la librería Winston. Este componente permite monitorear el comportamiento de la API en tiempo real mediante un logger personalizado, categorizando eventos por niveles de criticidad. Se priorizó el registro de errores internos y de acceso no autorizado, facilitando la identificación de incidentes en los endpoints más sensibles.

A continuación, se indicará de forma visual la interacción de la aplicación y las rutas empleadas con la interfaz de usuario.

Figura 6

Diagrama de rutas del API



2.6 Control de versiones y pruebas iniciales

Durante la etapa de desarrollo, la aplicación se dividió en dos módulos correspondientes al frontend y backend. Cada módulo fue asignado un repositorio independiente manejado a través de Git siguiendo el modelo de trabajo GitFlow en el cual se separa el desarrollo en ramas main, develop y feature. Cada componente o característica implementado durante el desarrollo se separó en su respectiva rama feature la cual posterior a pruebas de unitarias y revisión de código fue acoplada a la rama develop.

A su vez, la rama develop una vez funcional fue integrada con la rama main correspondiendo a la versión de producción de la aplicación que luego fue desplegada en la máquina virtual en la nube, permitiendo el acceso a la aplicación desde la internet pública

Este flujo de trabajo facilitó la colaboración e integración de los módulos del proyecto minimizando riesgos de conflictos o pérdidas de información.

Previo al desarrollo de la interfaz de frontend, se realizó una implementación de pruebas de backend para poder verificar la conexión básica del servidor y configurar el manejo de autenticación con el cliente de Better-Auth. Esta serie de pruebas fueron ejecutadas utilizando Postman y YaaK como clientes de testeo de APIs.

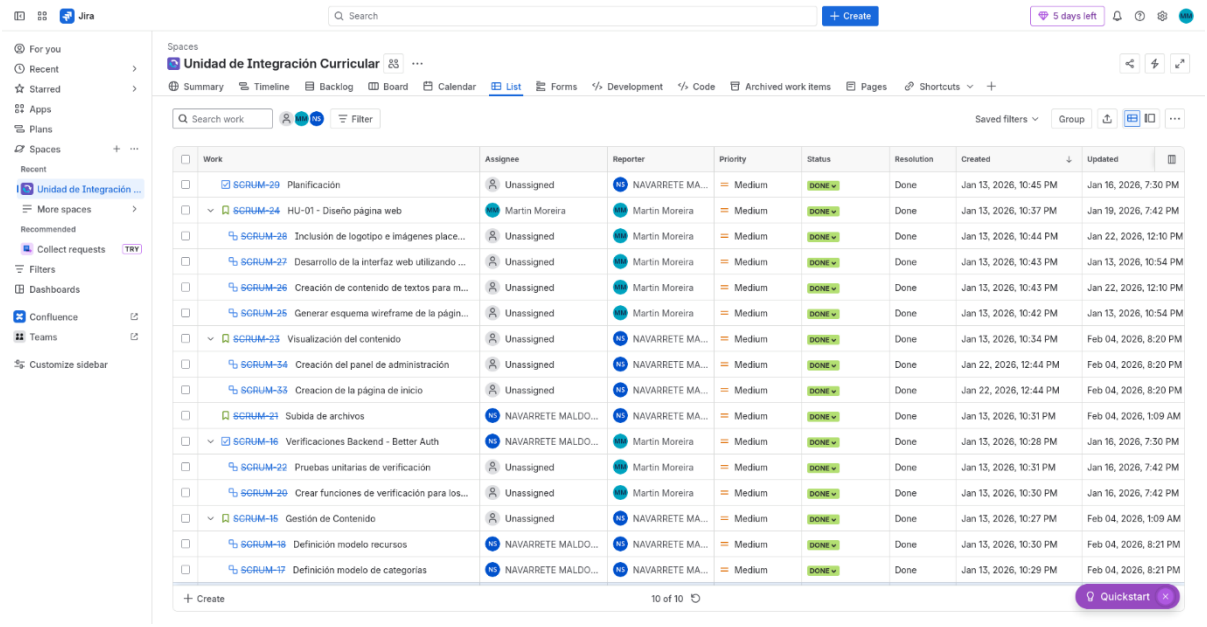
2.7 Aplicación y manejo de la metodología de desarrollo Scrum

Para la gestión del proceso de desarrollo y mejor organización de la metodología previamente determinada, se utilizó la herramienta de trabajo Jira. Se preparó un proyecto de desarrollo de software y se configuraron los requisitos funcionales y no funcionales a manera de Historias de Usuario con sus respectivas incidencias y criterios de aceptación.

Ya generado un backlog, Jira permitió configurar el ciclo de Sprints, proporcionando al equipo de trabajo una lista clara de trabajo a ser realizado.

Figura 7

Tablero de lista de trabajo en Jira



Capitulo III

Pruebas, Estabilización y Despliegue

En este apartado, se detallarán las diferentes pruebas a las que fue sometida la aplicación para detectar su actividad y comportamiento, lo que permitió identificar su fiabilidad, cómo se comporta individualmente cada componente, así como su reacción ante situaciones de estrés y sobrecarga.

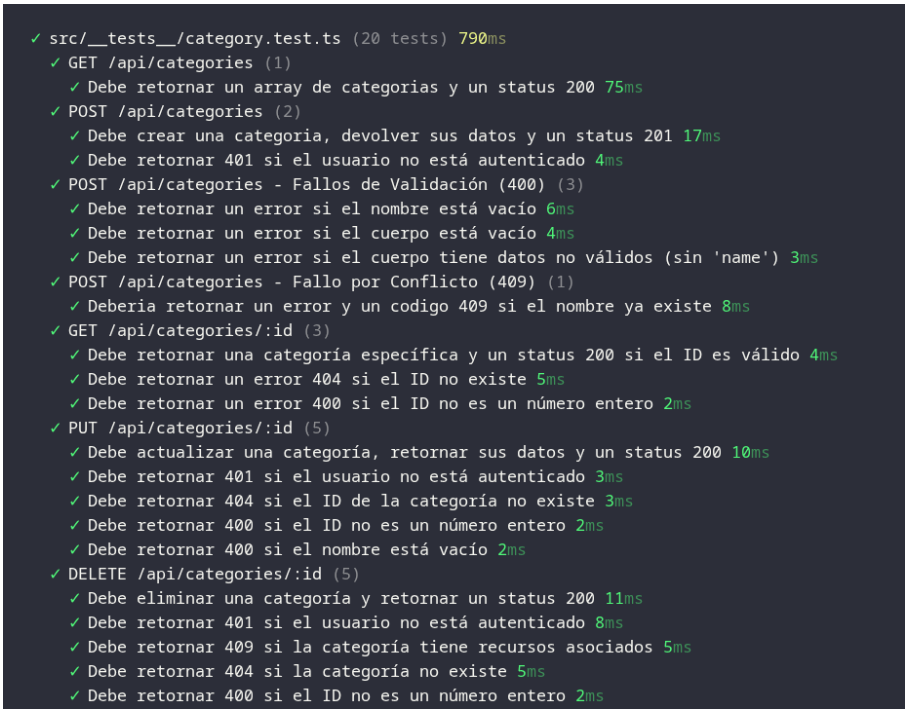
Tomando en cuenta lo mencionado anteriormente, estas pruebas han sido aplicadas al backend especialmente, utilizando herramientas como vitest, que ayudó a realizar pruebas unitarias y facilitar el desarrollo definiendo previamente el comportamiento de las solicitudes, sus entradas/salidas y pruebas propias para identificar la salud de la API.

Por otro lado, también se vió necesario el apoyo de K6, una herramienta de testing desarrollada con Golang, que es liviana, sencilla de configurar y capaz de realizar pruebas en diferentes hilos para reducir el tiempo empleado en el testeo. Esta herramienta en específico fue empleada para identificar posibles cuellos de botella y deficiencias de rendimiento.

3.1 Pruebas Unitarias

Figura 8

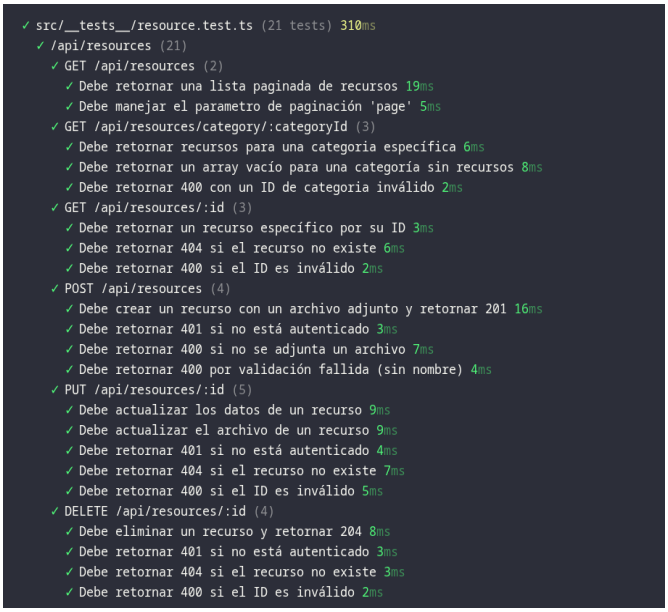
Pruebas unitarias para el módulo de categorías



Como se aprecia en la imagen previa, las pruebas unitarias aportaron información útil, especialmente en el momento del desarrollo, dado que especificaban los errores presentes en la API, yendo desde accesos denegados en rutas protegidas y métodos HTTP sin autenticación previa, hasta la identificación de códigos de error adecuados e ingresos incorrectos de información.

Figura 9

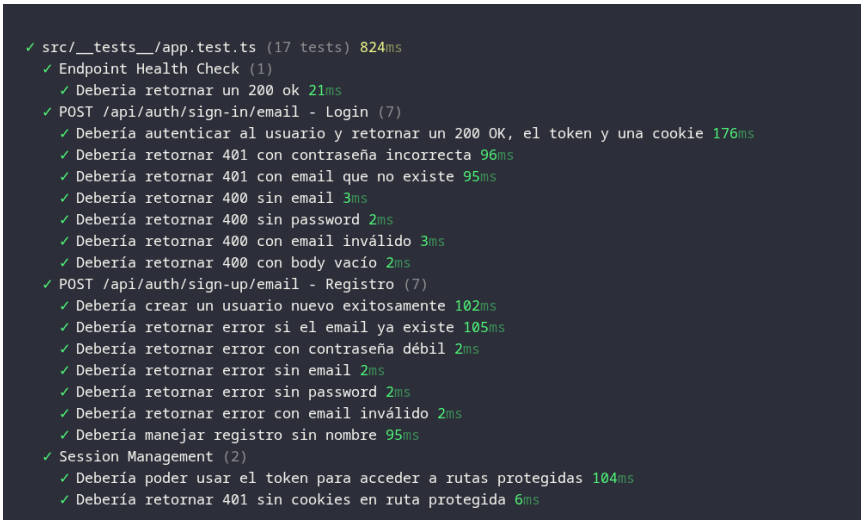
Pruebas unitarias para el módulo de recursos



De forma similar a las pruebas anteriores, el apartado de recursos brindó información sumamente importante, pero a diferencia de las pruebas realizadas para el apartado de categorías, la especificación y restricciones representan parte importante para evitar manipulación de la información mediante accesos no autorizados.

Figura 10

Pruebas unitarias empleadas para funciones generales



Nota: La figura solo muestra las pruebas realizadas para identificar la salud de la API y manejo de usuarios.

3.2 Tests de Rendimiento

Figura 11

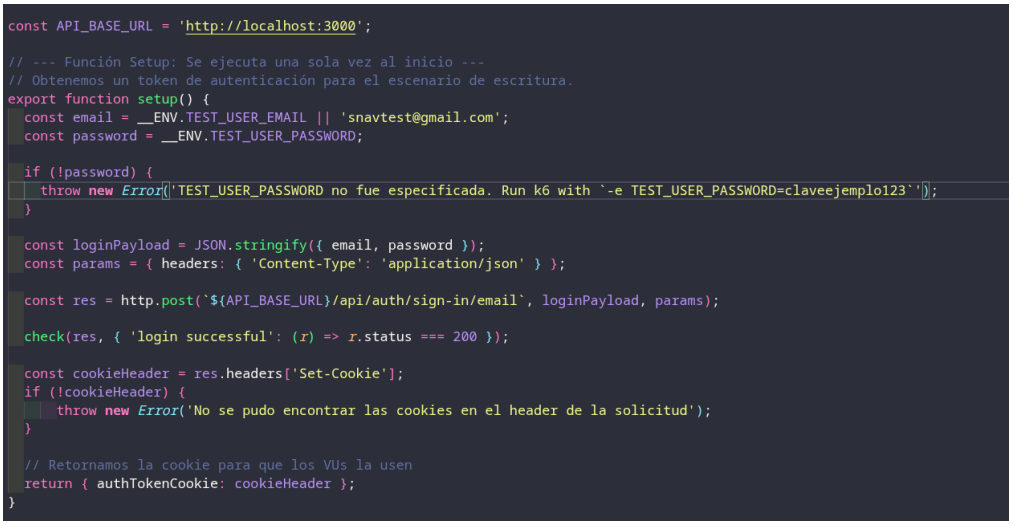
Parametrización de los tests

```
export const options = {
  thresholds: {
    'http_req_failed': ['rate<0.05'],
    'http_req_duration{scenario:public_reads}': ['p(95)<300'],
    'http_req_duration{scenario:admin_writes}': ['p(95)<1000'],
  },
  scenarios: {
    public_reads: {
      executor: 'constant-arrival-rate',
      rate: 30,
      timeUnit: '1s',
      duration: '7m',
      preAllocatedVUs: 20,
      maxVUs: 100,
      exec: 'publicReads',
    },
    admin_writes: {
      executor: 'per-vu-iterations',
      vus: 10, // 10 usuarios "admin" concurrentes
      iterations: 20, // Cada admin creará 20 categorías
      maxDuration: '5m',
      exec: 'adminWrites', // Llama a la función adminWrites
    }
  }
};
```

- Para la configuración de k6, se especificaron parámetros de usuarios, cantidad de solicitudes y el tiempo de ejecución.

Figura 12

Definición de la función núcleo del test



```
const API_BASE_URL = 'http://localhost:3000';

// --- Función Setup: Se ejecuta una sola vez al inicio ---
// Obtenemos un token de autenticación para el escenario de escritura.
export function setup() {
  const email = __ENV.TEST_USER_EMAIL || 'snavtest@gmail.com';
  const password = __ENV.TEST_USER_PASSWORD;

  if (!password) {
    throw new Error('TEST_USER_PASSWORD no fue especificada. Run k6 with `-e TEST_USER_PASSWORD=claveejemplo123`');
  }

  const loginPayload = JSON.stringify({ email, password });
  const params = { headers: { 'Content-Type': 'application/json' } };

  const res = http.post(`${API_BASE_URL}/api/auth/sign-in/email`, loginPayload, params);

  check(res, { 'login successful': (r) => r.status === 200 });

  const cookieHeader = res.headers['Set-Cookie'];
  if (!cookieHeader) {
      throw new Error('No se pudo encontrar las cookies en el header de la solicitud');
  }

  // Retornamos la cookie para que los VUs la usen
  return { authTokenCookie: cookieHeader };
}
```

Debido a que la API, maneja autenticación, es necesario recolectar la información de la sesión e inyectarla en cada solicitud para evitar accesos denegados para las acciones http restringidas.

Figura 13

Funciones para pruebas de rendimiento

```
// --- Escenario 1: Lecturas Públicas ---
export function publicReads() {
  const res = http.get(`${API_BASE_URL}/api/categories`);
  check(res, {
    'GET /api/categories status 200': (r) => r.status === 200
  }, { scenario: 'public_reads' });
  sleep(Math.random() * 2); // Espera aleatoria hasta 2s
}

// --- Escenario 2: Escrituras de Administrador ---
export function adminWrites(data) {
  const categoryName = `K6 Test Category ${__VU}-${__ITER}`;
  const createPayload = JSON.stringify({ name: categoryName });
  const params = {
    headers: {
      'Content-Type': 'application/json',
      'Cookie': data.authTokenCookie,
    },
  };

  const res = http.post(`${API_BASE_URL}/api/categories`, createPayload, params);
  check(res, {
    'POST /api/categories status is 201': (r) => r.status === 201,
  }, { scenario: 'admin_writes' });

  sleep(Math.random() * 5 + 1); // Espera entre 1 y 6 segundos
}
```

Figura 14

Resultados de la primera ronda de pruebas – Acceso único a capa de datos

```
  THRESHOLDS

    http_req_duration{scenario:admin_writes}
    ✓ 'p(95)<1000' p(95)=12.32ms

    http_req_duration{scenario:public_reads}
    ✓ 'p(95)<300' p(95)=1.68ms

    http_req_failed
    ✓ 'rate<0.05' rate=0.13%


  TOTAL RESULTS

    checks_total.......: 12781  30.283594/s
    checks_succeeded...: 99.86% 12764 out of 12781
    checks_failed......: 0.13%  17 out of 12781

    ✓ login successful
    × POST /api/categories status is 201
      ↳  91% — ✓ 183 / × 17
    ✓ GET /api/categories status 200
```

Figura 15

Revisión de Logs de Sistema Para Identificación de Errores

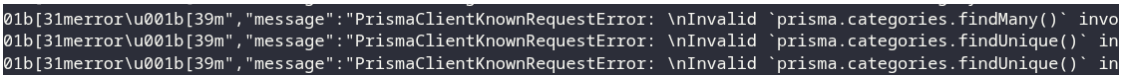
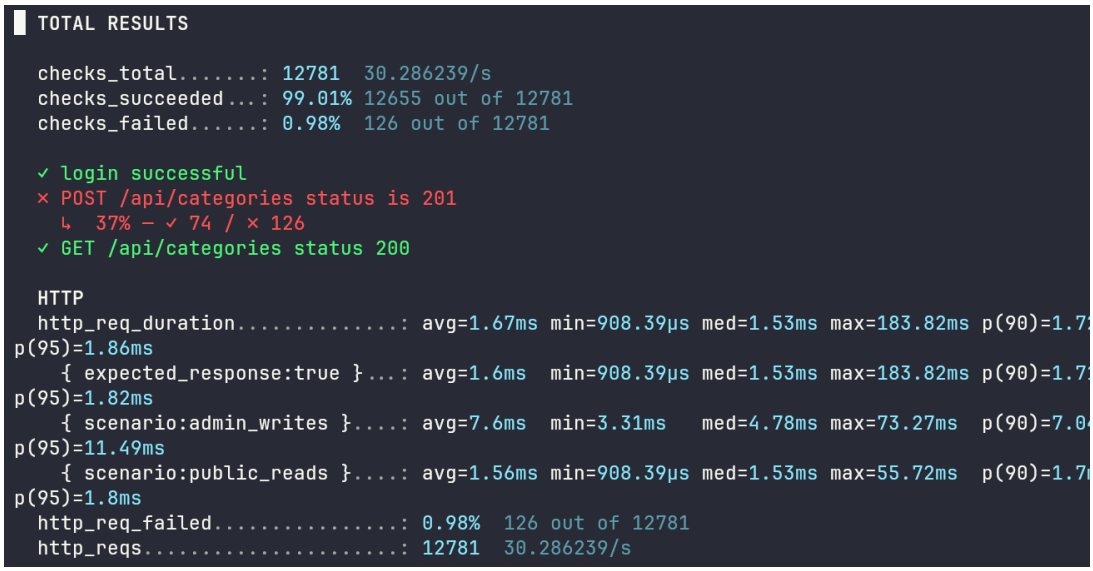


Figura 16

Resultados de la segunda ronda de pruebas – Acceso múltiple a capa de datos



Las pruebas de rendimiento especificadas para la API buscan explorar el comportamiento del código ante situaciones de estrés, más específicamente en las que han sido identificadas como las acciones más utilizadas (lectura y escritura de datos). En el caso de la API esta respondió de buena manera en la primera ronda, pero sufrió un decaimiento en la segunda ronda, arrojando resultados que indican un cambio de un 90% hasta un 37% de operabilidad bajo los parámetros definidos en un comienzo, lo que permite apreciar que el código se encuentra bien optimizado y que las consultas de administrador que sufrieron un bloqueo en las transacciones respondieron bien incluso bajo múltiples solicitudes.

No obstante, se puede diferenciar que existieron errores apreciables en los logs de sistema, en el que consultas básicas tuvieron fallos debido a un cuello de botella existente en la cantidad de accesos disponibles desde la base de datos hacia la capa de datos

gestionada por el ORM prisma. Por lo que se ejecutaron nuevamente las pruebas para determinar algún cambio.

En cuanto a lo explicado en el punto anterior, en la segunda ronda de pruebas, el rendimiento decayó hasta un apreciable 37%, solo con el cambio de accesos a la capa de datos. Esto deja en claro dos cosas especialmente, siendo la primera de ellas que los fallos no se ven estrictamente ligados a deficiencias en el código o a consultas no optimizadas, sino que son limitaciones provenientes de la capacidad del hardware de acceder a la base de datos, y, por último, el hecho de que prisma actúa como un limitador de carga, que no es un defecto en sí, sino más bien una consecuencia de la librería por evitar fallos críticos.

Finalmente, la evidencia revela que solucionar el cuello de botella provocado por Prisma mediante la ampliación de accesos solo dio paso a un bloqueo más crítico a nivel de infraestructura. Esto aclara que el problema no es la configuración de la librería, sino la capacidad física del servidor para procesar múltiples transacciones simultáneas. Por tanto, el sistema requiere una gestión de cargas más eficiente que actúe como amortiguador entre las peticiones entrantes y la escritura en disco.

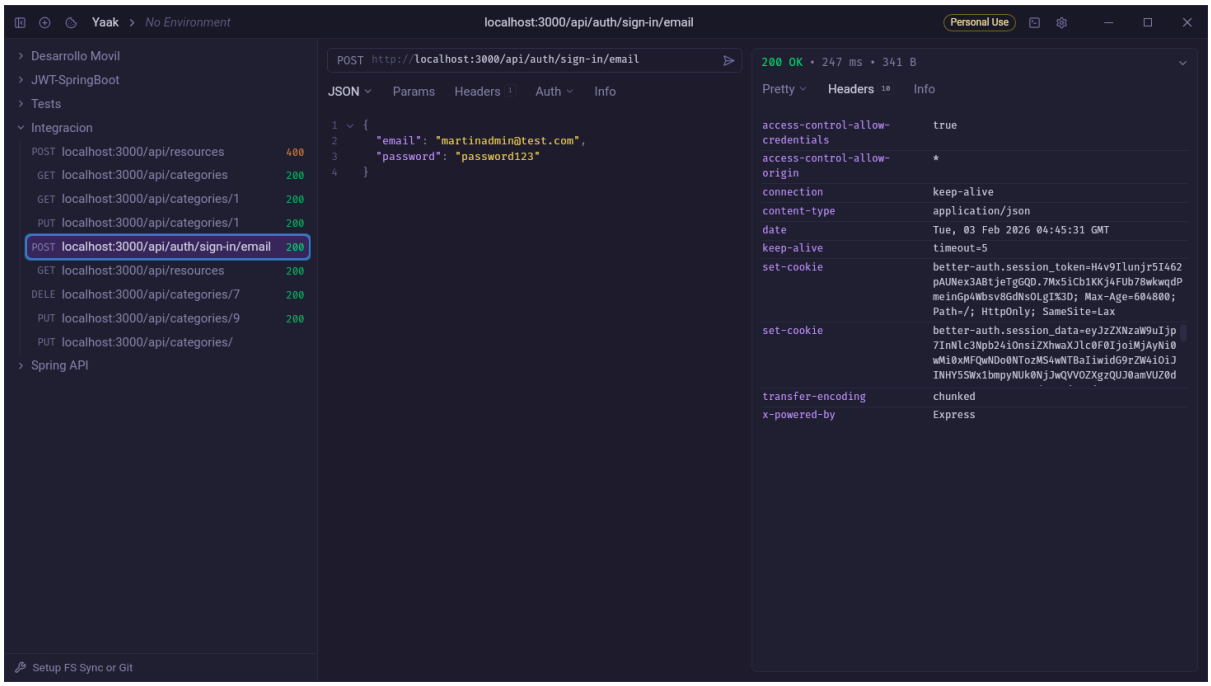
3.3 Tests con cliente de APIs

Las pruebas realizadas con clientes de testeo de APIs como YaaK permitieron comprobar la correcta operación del servidor de backend. Se pudo verificar el uso de headers de autenticación y el correcto manejo de datos enviados y recibidos por las peticiones en formato JSON.

Este tipo de tests fueron instrumentales durante el proceso de integración con el frontend y proporcionaron información valiosa para la depuración de errores.

Figura 17

Test de endpoints utilizando Yaak



3.4 Validaciones y pantallas de frontend

Figura 18

Pantalla principal de bienvenida de la aplicación web.

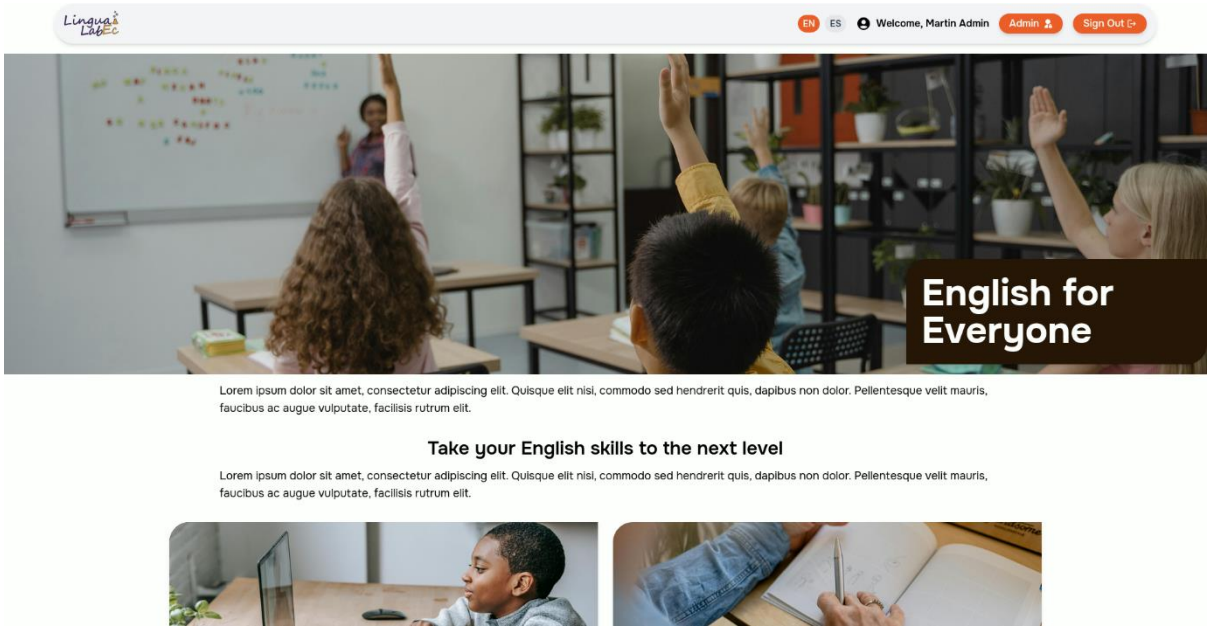


Figura 19

Panel de administración y gestión de categorías

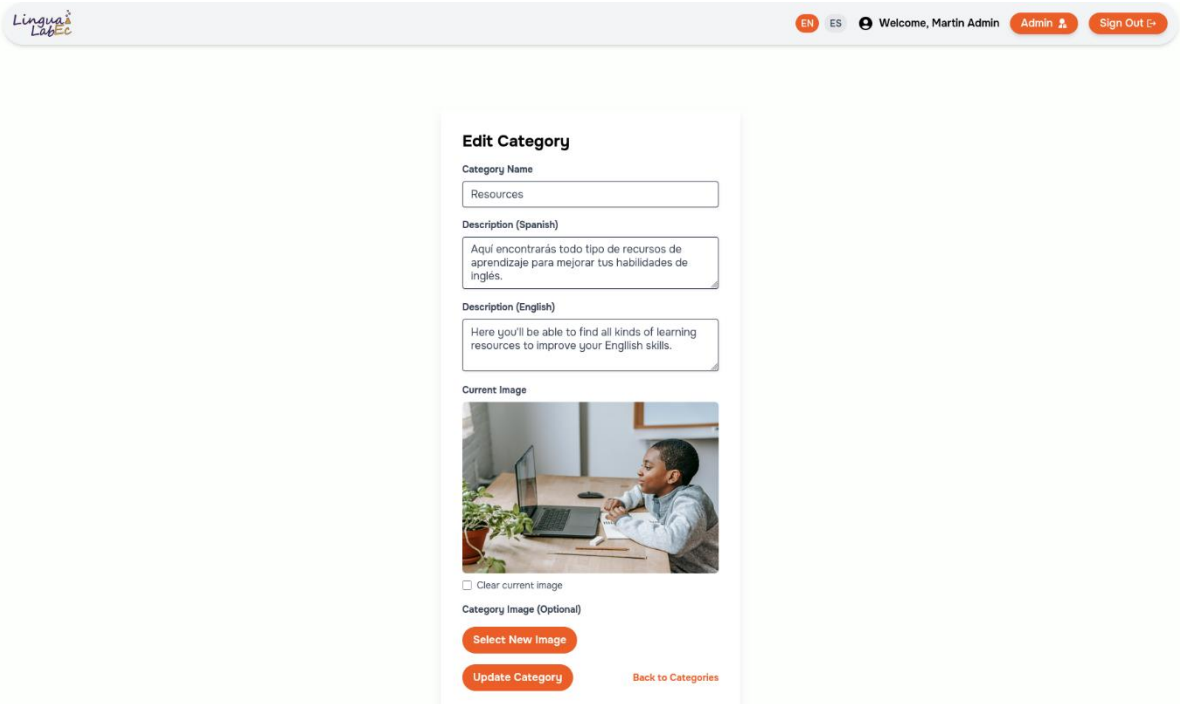


Figura 20

Pantalla de categoría con sus recursos asociados

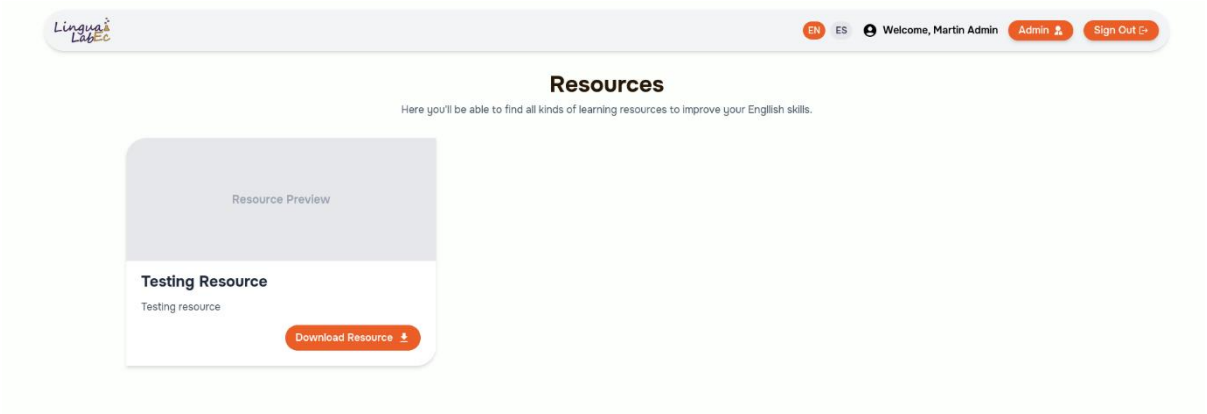


Figura 21

Interfaz con diseño responsive

3.5 Despliegue

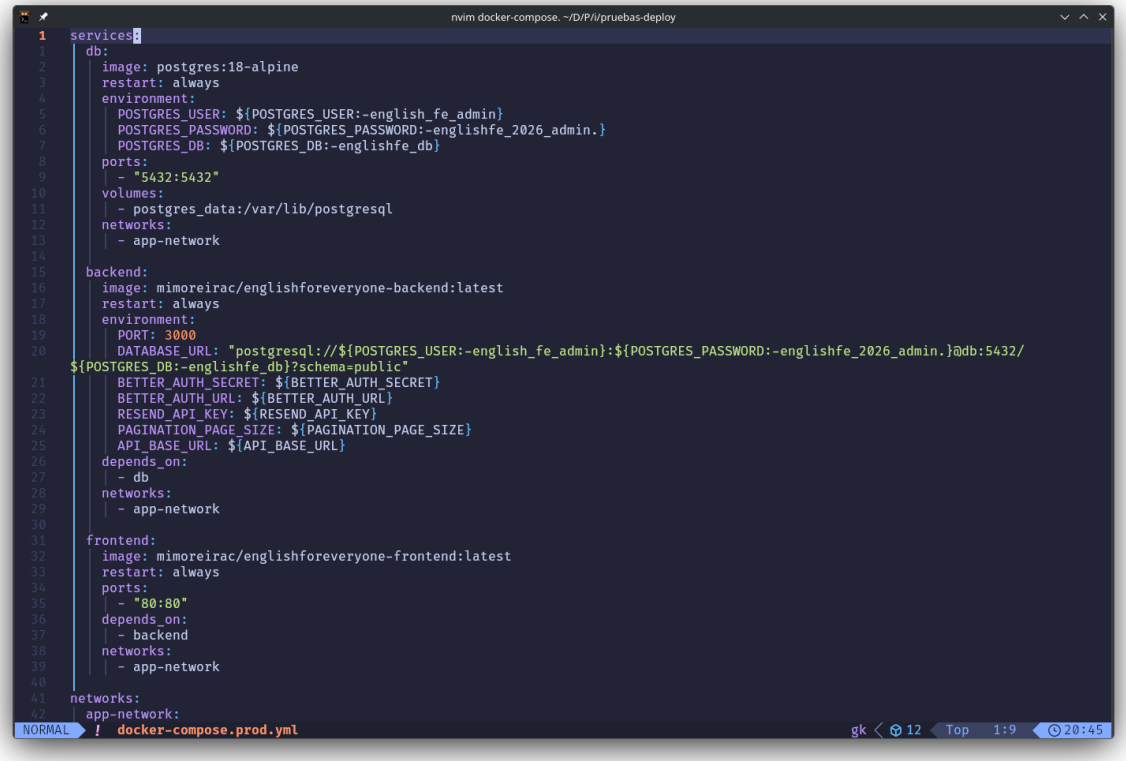
Concluido el ciclo de desarrollo y habiendo validado el correcto funcionamiento de la aplicación se procedió a la fase de preparación para su despliegue final. Siguiendo la misma estructura de componentes del proyecto se empleó Docker para generar imágenes contenedorizadas de las capas de frontend, backend y base de datos.

Las imágenes fueron construidas utilizando una máquina de desarrollo local para de esta manera reducir el impacto de rendimiento sobre el servidor privado virtual (VPS) y publicadas en el registro de imágenes público Docker Hub. Esto permite la configuración de un VPS con especificaciones básicas, minimizando así los costos incurridos del despliegue y operación.

El contenedor de frontend integra también el paquete Nginx, que cumple una doble función de servidor, permitiendo que usuarios de la aplicación web tengan acceso a los archivos estáticos de la aplicación, y de proxy inverso redirigiendo el tráfico del internet público hacia el servidor API de backend.

Figura 22

Archivo docker-compose que especifica la configuración de contenedores final de la aplicación.



Para mejor mantenibilidad a futuro durante la operación de la aplicación, se

configuraron varias reglas de firewall a través de la consola interactiva de la plataforma Google Cloud. Estas reglas son una capa adicional adicional de seguridad a nivel del servidor y restringen el acceso de tráfico público entrante a puertos no autorizados mientras que permiten que el equipo de desarrollo tenga acceso en caso de que sea necesario realizar mantenimientos o cambios en la aplicación.

Figura 23

Panel de observación del servidor privado virtual (VPS) y su rendimiento

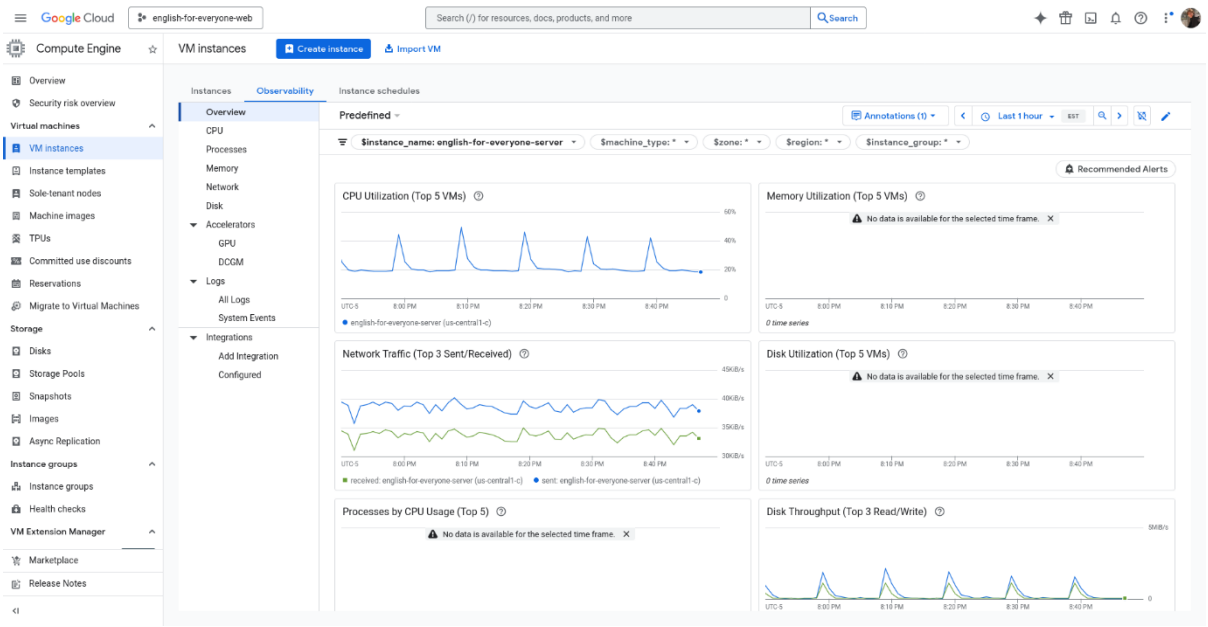
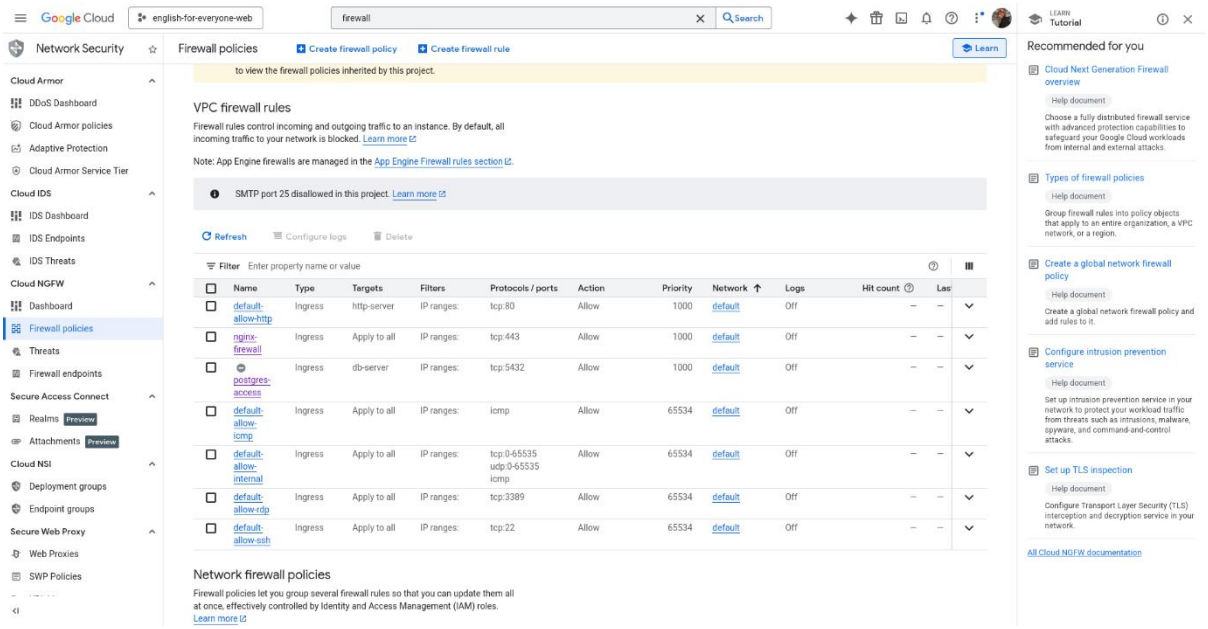


Figura 24

Configuración de Firewall en la Consola de Google Cloud



3.6 Disaster Recovery Plan

Como parte del despliegue en la plataforma de Google Cloud, también se consideraron los mecanismos y procesos necesarios para garantizar la continuidad operativa del sistema ante posibles eventos que vulneren la integridad de la infraestructura y comprometan la información almacenada en el servidor.

Se seleccionó el servicio integrado "Backup and DR" de Google Cloud como la mejor opción puesto que permite la creación automática de snapshots o capturas instantáneas del estado del servidor. Estos snapshots pueden incluir los datos almacenados en disco y las configuraciones del sistema con una frecuencia de respaldo que se verá determinada según las necesidades del cliente.

En caso de fallo crítico del servidor el plan de acción consiste en:

- Evaluar el incidente para determinar el alcance y la naturaleza del fallo.
- Seleccionar el snapshot más reciente previo al incidente.
- Utilizar el snapshot seleccionado para crear una nueva instancia del VPS.
- Comprobar la integridad de los datos y el funcionamiento del sistema.
- Redireccionar el tráfico web configurando las políticas de firewall ya existentes.
- Realizar un reporte sobre el incidente y explorar avenidas de mejora para prevenir situaciones similares a futuro.

3.7 Alojamiento y operación

Finalizado el proceso de despliegue de la aplicación, esta se encuentra disponible al internet a través de la dirección IP pública http://35.193.20.213/

Se contempla asociar esta dirección IP con un dominio registrado con autorización del cliente para su operación a futuro. El uso del dominio registrado permitirá además utilizar un protocolo seguro HTTPS con emisión de certificados.

Conclusiones

1. La aplicación, English For Everyone, resultó ser un proyecto viable y eficaz para poder resolver la problemática referente a la dispersión y falta de centralización de los recursos educativos del emprendimiento, integrando en una misma plataforma la gestión, distribución y visualización de los contenidos.
2. A través de las diversas funcionalidades implementadas se dotó al docente de una herramienta que gestiona eficazmente sus materiales educativos y que la organización temática sea tanto sencilla, como estructurada.
3. La estructura arquitectónica y el seguimiento de buenas prácticas de desarrollo aportaron solidez al sistema, dotando a la aplicación de una estabilidad que facilita su crecimiento. De este modo, se garantiza que futuros mantenimientos o actualizaciones se realicen de manera fluida sin interrumpir las operaciones críticas
4. La interfaz aporta a la experiencia del usuario con simplicidad y retroalimentación constante, lo que no solo facilita la labor pedagógica, sino que optimiza el proceso de aprendizaje del estudiante, permitiendo centrarse en mejorar habilidades específicas.
5. El soporte bilingüe interactivo y el diseño adaptable de la aplicación responden directamente a las necesidades del sector educativo moderno. Estas características no solo alinean la plataforma con su propósito de enseñanza de idiomas, sino que también aseguran que el contenido sea accesible desde cualquier dispositivo, ampliando el alcance potencial del emprendimiento.

Recomendaciones

- Dada la base técnica en React, se sugiere considerar el desarrollo de versiones nativas para iOS y Android utilizando React Native. Esto permitiría aprovechar funciones específicas de los dispositivos, como notificaciones push para recordatorios de estudio, mejorando la retención y el compromiso del usuario.
- Para potenciar el aprendizaje del idioma, se recomienda añadir elementos interactivos como sistemas de medallas, tablas de clasificación o desafíos diarios. Estas dinámicas aumentan la motivación del estudiante y transforman la plataforma en un entorno de aprendizaje más dinámico y participativo.
- Ante un crecimiento previsto en el volumen de materiales didácticos, se recomienda explorar la integración de servicios de almacenamiento en la nube (como AWS S3 o Google Cloud Storage). Esto asegurará que la carga de archivos pesados, como videos en alta resolución, no comprometa la velocidad de respuesta del sistema.

Referencias

Atlassian. (s. f.). *¿Qué es scrum? Una guía para el marco de trabajo ágil*. Recuperado 31 de enero de 2026, de https://www.atlassian.com/es/agile/scrum

Better-Auth. (s. f.). *Better Auth*. Recuperado 31 de enero de 2026, de https://better-auth.com

Anexos

Repositorio privado para el desarrollo del frontend en GitHub

Repositorio privado para el desarrollo del backend en GitHub