ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE UN SISTEMA PARA EL APRENDIZAJE DE PROGRAMACIÓN EN NIÑOS MENORES A 12 AÑOS

BACKEND

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE

DILAN ALEXANDER FLORES QUIMBIA

DIRECTOR: VANESSA KATHERINE GUEVARA BALAREZO

DMQ, marzo 2024

CERTIFICACIONES

Yo, **DILAN ALEXANDER FLORES QUIMBIA** declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

DILAN ALEXANDER FLORES QUIMBIA

dilan.flores@epn.edu.ec

dilanflores.21@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por **DILAN ALEXANDER FLORES QUIMBIA**, bajo mi supervisión.

Ing. VANESSA GUEVARA

DIRECTOR

vanessa.guevarav@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

DILAN ALEXANDER FLORES QUIMBIA

DEDICATORIA

Para mi familia, quienes con apoyo incondicional me motivaron constantemente y con amor, me enseñaron valiosas lecciones.

AGRADECIMIENTO

Agradezco sinceramente a las personas que contribuyeron en mi desarrollo personal y académico. Quiero expresar mi gratitud a mis padres, cuya influencia y apoyo han sido parte indispensable en mi formación personal y fuente constante de motivación. Asimismo, agradezco a mis profesores, cuya orientación ha sido clave en la formación académica. Estoy agradecido por su dedicación, guía y participación en este viaje, para lograr la victoria que presenta la realización de este sueño.

ÍNDICE DE CONTENIDO

RESUMEN

El presente proyecto aborda la brecha digital en la comprensión y aplicación de la programación desde las primeras etapas de la educación, mediante la creación de un componente *backend* centrado en el aprendizaje progresivo de programación para niños menores de 12 años. Para guiar este desarrollo, se emplea la metodología ágil SCRUM y se estructura según el Modelo-Vista-Controlador (MVC), utilizando JavaScript para facilitar la manipulación de datos. Además, se utiliza el f*ramework* Express para gestionar rutas, respuestas HTTP y autenticación de usuarios. La elección de la base de datos MongoDB permite almacenar información de perfiles de usuarios, actividades y funcionalidades asociadas. Este enfoque tiene como objetivo proporcionar oportunidades educativas accesibles, fomentando un aprendizaje progresivo y motivador en niños desde una edad temprana a través de un *backend* estructurado y funcional.

La estructura del presente documento comienza con la identificación del problema, definiendo los objetivos para su resolución. Posteriormente, se establecen los límites dentro del alcance del proyecto y se detallan los conocimientos necesarios para abordar el tema. A continuación, se describe la metodología empleada para guiar el desarrollo del proyecto, seguido de la presentación de los *endpoints* generados en cada iteración, junto con sus respectivas pruebas. Finalmente, se concluye con el despliegue en producción y se exponen las conclusiones y recomendaciones obtenidas a lo largo del proceso.

PALABRAS CLAVE: aprendizaje progresivo**,** *backend,* Express, JavaScript, MongoDB, programación para niños, oportunidades educativas.

ABSTRACT

The present project addresses the digital divide in understanding and applying programming from the early stages of education, through the creation of a backend component focused on the progressive learning of programming for children under 12 years old. To guide this development, the agile SCRUM methodology is employed and structured according to the Model-View-Controller (MVC), using JavaScript to facilitate data manipulation. Additionally, the Express framework is utilized to manage routes, HTTP responses, and user authentication. The choice of MongoDB database allows for storing information on user profiles, activities, and associated functionalities. This approach aims to provide accessible educational opportunities, fostering progressive and motivating learning in children from an early age through a structured and functional backend.

The structure of this document begins with the identification of the problem, defining the objectives for its resolution. Subsequently, the project's scope is established, and the necessary knowledge to address the issue is detailed. Next, the methodology employed to guide the project's development is described, followed by the presentation of the endpoints generated in each iteration, along with their respective tests. Finally, the document concludes with the deployment in production and the presentation of the conclusions and recommendations obtained throughout the process.

KEYWORDS: progressive learning, *backend*, Express, JavaScript, MongoDB, programming for children, educational opportunities.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La brecha digital y educativa en Ecuador, combinada con la falta de acceso a programas de aprendizaje de programación en escuelas, crea una disparidad significativa en las habilidades que los niños desarrollan desde temprana edad. En un mundo cada vez más digitalizado, esta carencia puede resultar en una desventaja para los niños en términos de competencias futuras y oportunidades laborales [1].

Las escuelas fiscales, especialmente en comunidades rurales, enfrentan desafíos específicos en el acceso a programas de aprendizaje de programación para niños menores de 12 años [2]. La edad óptima para el desarrollo cerebral y la mejora de la capacidad de aprendizaje se sitúa entre los 7 y 8 años, cuando los niños y niñas son más creativos y pueden comprender conceptos más complejos. A los 10 años, se produce un incremento en el desarrollo intelectual referente a la curiosidad, la capacidad lógica y el desarrollo de primeros hobbies [3]. Es durante este período que resulta oportuno fomentar habilidades y promover un pensamiento más digital, ampliando las posibilidades profesionales que los niños pueden imaginar, como complemento a profesiones tradicionales como ser profesor o médico, incluyendo roles en robótica o programación [1].

La educación en programación para niños menores de 12 años es crucial para proporcionar habilidades cognitivas y prepararlos para un mundo tecnológico en constante evolución. Esto incluye aspectos como computación, pensamiento crítico, creatividad, comunicación, colaboración y carácter [4].

El objetivo principal de este proyecto es desarrollar un sistema para el aprendizaje de programación dirigida a niños menores de 12 años, centrándose en el *backend*. Se busca proporcionar a padres (tutores) e hijos una herramienta que estimule el pensamiento lógico, la resolución de problemas y la creatividad a través de la programación. Además, se establece un entorno educativo que facilita el aprendizaje intuitivo de los fundamentos de la programación.

1.1 Objetivo general

Desarrollar un *backend* de un sistema para el aprendizaje de programación en niños menores a 12 años.

1.2 Objetivos específicos

1. Levantar los requerimientos necesarios para el desarrollo del *backend*.
2. Diseñar la arquitectura del *backend* basada en el Modelo-Vista-Controlador (MVC).
3. Diseñar el modelo de base de datos no relacional de acuerdo con los requerimientos establecidos.
4. Codificar los *endpoints* para el *backend* según los roles especificados.
5. Verificar el correcto funcionamiento de los *endpoints* mediante pruebas.
6. Desplegar el *backend* en un ambiente en la nube.

1.3 Alcance

En la actualidad, el aprendizaje de programación se ha convertido en una habilidad esencial en la era digital, ya que contribuye significativamente a la resolución de problemas y al desarrollo de soluciones aplicables en diversos campos como la tecnología, la medicina y la ingeniería. Aunque muchas escuelas aún no han integrado la programación en su currículo, la demanda es considerable en nuestro entorno. Por lo tanto, es crucial que los niños comiencen a adquirir conocimientos en programación desde una edad temprana utilizando recursos diseñados específicamente para ellos [5].

El desarrollo de sitios web abarca tres áreas clave, una de las cuales se centra en la programación del *backend*. Esta área es responsable de controlar la funcionalidad esencial del sitio, incluyendo la lógica necesaria para gestionar tareas como la carga de archivos, el acceso mediante usuario y contraseña, el almacenamiento seguro de información, entre otras funciones esenciales. En cuanto a los lenguajes de programación, estos varían e incluyen Python, Ruby y JavaScript [6].

En esta iniciativa, se busca implementar un componente *backend* dirigido a niños menores a 12 años después de su análisis, desarrollo, pruebas e implementación.

El objetivo es mejorar sus conocimientos en un entorno educativo, especialmente en el ámbito de la tecnología, con el fin de enriquecer su formación. Se identifican tres perfiles de usuario: Administrador, Tutor y Niño, cada uno con módulos y funcionalidades distintas, como se detalla a continuación:

Para usuario con perfil Administrador se generan:

- *Endpoints* que permiten restablecer contraseña.
- *Endpoints* que permiten iniciar sesión y cerrar sesión.
- *Endpoints* que permiten administrar actividades.
- *Endpoints* que permiten eliminar en cascada: Tutor, niños, inscripciones, progreso y logros

Para usuario con perfil Tutor se generan:

- *Endpoints* que permiten registrar cuenta.
- *Endpoints* que permiten restablecer contraseña.
- *Endpoints* que permiten iniciar sesión y cerrar sesión.
- *Endpoints* que permiten visualizar el perfil de usuario.
- *Endpoints* que permiten administrar niños.
- *Endpoints* que permiten restablecer contraseña de niños
- *Endpoints* que permiten visualizar actividades disponibles.
- *Endpoints* que permiten inscribir niños en actividades.
- *Endpoints* que permiten visualizar actividades en las que el niño está inscrito

Para usuario con perfil Niño se generan:

- *Endpoints* que permiten iniciar sesión y cerrar sesión.
- *Endpoints* que permiten visualizar el perfil de usuario.
- *Endpoints* que permiten visualizar actividades en las que el niño está inscrito
- *Endpoints* que permiten registrar y visualizar progreso en las actividades

- *Endpoints* que permiten modificar perfil de usuario.
- *Endpoints* que permiten registrar y visualizar logros personales

1.4 Marco Teórico

Ingeniería de *software*

La ingeniería de *software* y la ingeniería web desempeñan un papel crucial en el diseño y desarrollo eficiente de sistemas que facilitan el proceso de aprendizaje.

La ingeniería de *software* es una disciplina que se enfoca en la aplicación de diversas metodologías y herramientas para el desarrollo exitoso de *software*, estructurando eficientemente cada fase del proceso [7].

Por otro lado, la ingeniería web se destaca por su uso de métodos sistemáticos para el desarrollo efectivo de programas de alta calidad en la *World Wide Web* (WWW) [8].

Desde el análisis de requerimientos hasta la implementación y mantenimiento, el proceso de creación de *software* se convierte en una herramienta esencial para lograr un aprendizaje efectivo de programación. Las etapas de análisis, diseño, desarrollo, pruebas e implementación se llevan a cabo para asegurar el desarrollo de un *software* de calidad [9].

JavaScript

JavaScript es un lenguaje de programación popular, utilizado para codificar no solo del lado del cliente, sino también en el lado del servidor en el entorno Node.js. Dado que JavaScript tiene un enfoque tanto en el lado del servidor como en el lado del cliente, estos funcionan juntos de manera dinámica [10].

La programación en un servidor web se construye usando web *frameworks*, lo que facilita la realización de operaciones como recuperar datos de una base de datos y presentarlos en una página, validar campos con datos introducidos del usuario, guardar en la base de datos, etc. [11].

Framework Express

Express es un web *framework* popular, codificado en JavaScript y alojado en el entorno de ejecución Node.js. El *framework* crea un "esqueleto" en el que es posible

agregar rutas específicas para el sitio. Además, presenta documentación apropiada para el uso de Mongo como base de datos para sitios web, con la declaración de objetos de esquemas y modelos, tipo de datos para los campos, validaciones básicas y formas para acceder a los modelos de datos [12].

MongoDB

MongoDB, como sistema de base de datos NoSQL, almacena la información en colecciones y documentos, aportando la capacidad de almacenar y recuperar datos de manera eficiente y flexible [13].

MongoDB Atlas es una base de datos popular para el desarrollo y la creación de prototipos, también se utiliza como proveedor de servicios. Esta base de datos es de uso gratuito y fácil de configurar: definición de esquemas, definición y creación de modelos, validaciones integradas y personalizadas [14].

Proceso de *software*

El proceso en *software* es un conjunto de actividades, métodos, prácticas y tecnologías que son aplicables a cualquier proyecto de *software*

El proceso de *software* contiene una serie de pasos que permite llevar a cabo el análisis, diseño, codificación, pruebas y mantenimiento [15].

- Análisis: Se definen los requisitos funcionales y no funcionales del sistema.
- Diseño de arquitectura y la estructura del sistema, así como los componentes y la presentación de información a usuarios.
- Codificación: En base al diseño se implementa el sistema.
- Pruebas: Al finalizar la codificación, el sistema pasa por una serie de pruebas para verificar el funcionamiento de componentes individuales.
- Mantenimiento: Monitoreo activo del rendimiento del componente *backend.*

Enfoque de calidad

El enfoque de calidad es fundamental para asegurar que el componente cumpla con los estándares en términos de funcionalidad, usabilidad, rendimiento y seguridad. Se sigue un proceso estructurado y se aplican principios de calidad en todas las etapas del desarrollo para garantizar que el sistema cumpla con las necesidades y expectativas de los usuarios. [15]

2 METODOLOGÍA

En el marco de este proyecto de Integración Curricular, se emplea un enfoque de estudio de caso que permite una exploración detallada de un tema específico. Este método se caracteriza por describir, comparar, evaluar y comprender diferentes aspectos de un problema, siendo comúnmente utilizado en temas de investigación social, educativa, clínica y empresarial [16].

Considerando los aspectos mencionados, este proyecto de aprendizaje en niños menores de 12 años se establece en un ambiente educativo que involucra la investigación y análisis de la problemática de la brecha digital y educativa en las escuelas fiscales de Ecuador. Además, se aborda la baja competitividad en un entorno tecnológico, identificando la relevancia de intervenir en estas áreas desde una edad temprana.

La selección de la edad menor a 12 años se fundamenta en una ventana de oportunidad óptima para el aprendizaje de habilidades cognitivas y, especialmente, de programación. En este contexto, el desarrollo del *backend* de un sistema de aprendizaje de programación adquiere una relevancia especial al buscar cultivar una mayor agilidad mental en niños para comprender algoritmos, conceptos matemáticos, etc., desde una edad temprana.

2.1 Metodología de Desarrollo

Para el desarrollo del *backend* en el tema de aprendizaje de programación en niños menores de 12 años, se emplea una metodología ágil debido a su flexibilidad y capacidad de adaptación a los cambios que puedan surgir durante el proyecto.

Específicamente, se utiliza la metodología ágil SCRUM por su capacidad para adaptarse de manera óptima al proyecto, especialmente en el ámbito del desarrollo web, permitiendo un desarrollo eficiente dentro de un plazo apropiado de tiempo. SCRUM se basa en iteraciones continuas que permiten construir un producto en cada una de ellas, priorizando las necesidades de los usuarios [17].

Roles

SCRUM se fundamenta en tres roles que detallan las responsabilidades y obligaciones de los miembros del equipo para ejecutar eficazmente el trabajo. Esta

estructura define una colaboración y asignación de tareas dentro del equipo [18]. A continuación, se describen los roles asignados en el desarrollo del *backend.*

Equipo de desarrollo (*Development Team)*

El equipo de desarrollo está compuesto por las personas encargadas de realizar el trabajo. Este equipo se encarga de desarrollar el proyecto y entregar el trabajo de acuerdo con los objetivos establecidos en cada *Sprint.* Además, se enfoca en mantener una transparencia del trabajo [18].

Propietario del producto (*Product Owner)*

El propietario del producto es responsable de conocer los requisitos de los clientes o usuarios a los que va dirigido el proyecto. A partir de estos requisitos, el propietario del producto crea y gestiona el *backlog* del producto. Debe tener un buen entendimiento de las necesidades del usuario final y presentar las prioridades del trabajo en función de estas necesidades [18].

Experto en SCRUM (SCRUM *Master)*

El experto en SCRUM es el encargado de unir al equipo y garantizar que el proyecto se realice correctamente. Ayuda al propietario del producto a definir el valor del proyecto y al equipo de desarrollo a entregar los requisitos necesarios para alcanzar los objetivos establecidos [18].

En la **Tabla 2. 1** se presenta las personas responsables de cada rol en la metodología SCRUM.

Tabla 2. 1: Roles SCRUM

Rol	Nombre
Product Owner	Ing. Vanessa Guevara
SCRUM *Master*	Ing. Vanessa Guevara
Development Team	Dilan Alexander Flores Quimbia

Artefactos

Los artefactos de SCRUM constituyen una pieza fundamental que captura la esencia de las acciones ejecutadas y las tareas realizadas a lo largo del proyecto. Estos elementos no son solo registros, sino que representan información crucial para comprender y detallar el producto en desarrollo [19]. En el entorno del *backend*, se implementan los siguientes artefactos:

Recopilación de Requerimientos

En el proceso de desarrollo de un proyecto de *software*, los requerimientos son fundamentales para definir de manera clara y sin ambigüedad lo que se quiere lograr. Esta fase marca el punto de partida para la planificación, reflejando las necesidades del usuario final. Además, establece una base sólida para el desarrollo, permitiendo la verificación del cumplimiento de los objetivos propuestos [20]. En la **Tabla 2. 2** se muestra el formato empleado para la recopilación de requerimientos y la lista completa de los requerimientos se encuentra disponible en el **ANEXO II**.

Tabla 2. 2: Recopilación de requerimientos

RECOPILACION DE REQUERIMIENTOS		
TIPO DE SISTEMA	**ID – RR**	**ENUNCIADO DEL ITEM**
backend	**RR-001**	Como usuario Tutor, se requiere crear *endpoints* para: • Registrar cuenta

Historias de Usuario

Las historias de usuario se presentan como una descripción general de una función desde la perspectiva del usuario. Su objetivo principal es expresar de manera clara cómo dicha función genera un valor significativo para el usuario final [21]. En la **Tabla 2. 3** se presenta el formato empleado en las Historias de Usuario, la lista completa se encuentra disponible en el **ANEXO II**.

Tabla 2. 3: Formato Historias de Usuario

HISTORIA DE USUARIO	
Identificador: HU-001	**Usuario:** Tutor
Nombre Historia: Registrar cuenta	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Media
Iteración asignada: 1	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para que el usuario con perfil Tutor pueda registrarse.	
Observación: Los usuarios con perfil Tutor necesitan registrar su cuenta para acceder a los *endpoints* asignados. Para el usuario con perfil Administrador se registra una cuenta sin un *endpoint* correspondiente.	

Product Backlog

El *Backlog* de un producto se representa con una lista de tareas ordenadas por prioridad, a las cuales se les asigna el tiempo necesario en la fase de desarrollo. Esta lista considera el orden de las tareas, las prioridades asignadas e incluso aquellas que no serán directamente visibles para el usuario [22]. En la **Tabla 2. 4** se muestra el formato empleado para el *Product Backlog* y la tabla completa está disponible en el **ANEXO II**.

Tabla 2. 4: Formato para el *Product Backlog*

ELABORACION DEL *PRODUCT BACKLOG*				
HU	**HISTORIA DE USUARIO**	**ITERACION**	**ESTADO**	**PROIRIDAD**
HU-001	Registrar cuenta	1	Terminado	Media

Sprint Backlog

En el *Sprint Backlog* se define un tiempo específico para el desarrollo de cada tarea con el propósito de priorizar las tareas a llevar a cabo. Además, se describe las tareas y actividades planificadas para cada *Sprint*, considerando aquellas

necesarias para completarse dentro del plazo del *Sprint* [23]. En la **Tabla 2. 5** se muestra el formato empleado para el *Sprint Backlog* y la tabla completa está disponible en el **ANEXO II**.

Tabla 2. 5: Formato de *Sprint Backlog*

ELABORACION DEL *SPRINT BACKLOG*						
ID-SB	**NOMBRE**	**MÓDULO**	**HU**	**HISTORIAS DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB-000	Configuración del ambiente de desarrollo	N/A	N/A	N/A	• Recopilación y definición de los requerimientos • Creación de la estructura del proyecto *backend* • Diseño e implementación de base de datos NoSQL • Definición de roles de usuarios con sus respectivas funcionalidades	**10 H**

2.2 Diseño de la arquitectura

La arquitectura de *software* se refiere a patrones o directrices que guían a los desarrolladores, analistas y demás roles durante la creación de un programa, proporcionando un camino claro y efectivo para lograr el cumplimiento de los requisitos específicos de la aplicación [24].

En la arquitectura de *software*, se estructura la aplicación. Mediante el análisis de requisitos, se diseñan patrones y se determinan servidores, tecnologías y bases de datos para abordar de manera eficaz el problema en cuestión. [24].

Arquitectura de Datos

La arquitectura de datos se refiere a la estructura y organización de los datos dentro de un sistema o una organización. Se trata de un marco que define cómo se recopilan, almacenan, gestionan, y utilizan los datos en un entorno determinado. Esto incluye la identificación de los tipos de datos, la definición de las relaciones entre ellos y los métodos de almacenamiento. En la **Figura 2. 1** se muestra el Modelado de la Base de Datos del Sistema.

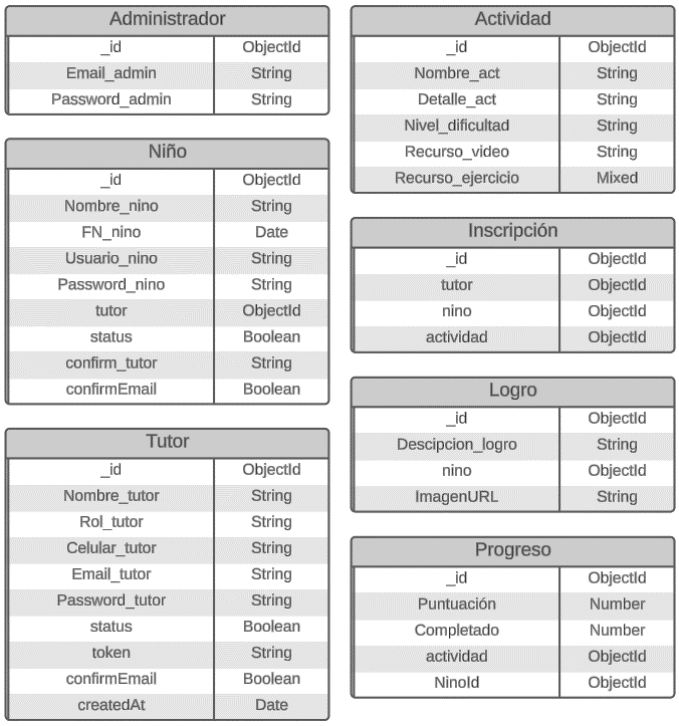


Figura 2. 1: Modelado de Base de Datos

Patrón arquitectónico

El patrón arquitectónico Modelo-Vista-Controlador (MVC) es ampliamente utilizado en el diseño de *software*. Este patrón integra el modelo para la gestión de datos, el controlador para la lógica que ajusta el modelo y/o las vistas según las interacciones del usuario, y, finalmente, las vistas que, en el caso de un componente *backend*, definen las rutas de los *endpoints* [25]. El MVC adopta la idea de reutilizar código y separar conceptos, con el objetivo de facilitar el desarrollo y mantenimiento [26].

En la **Figura 2. 2** se muestra la implementación del patrón arquitectónico aplicado al componente *backend* desarrollado en este proyecto.

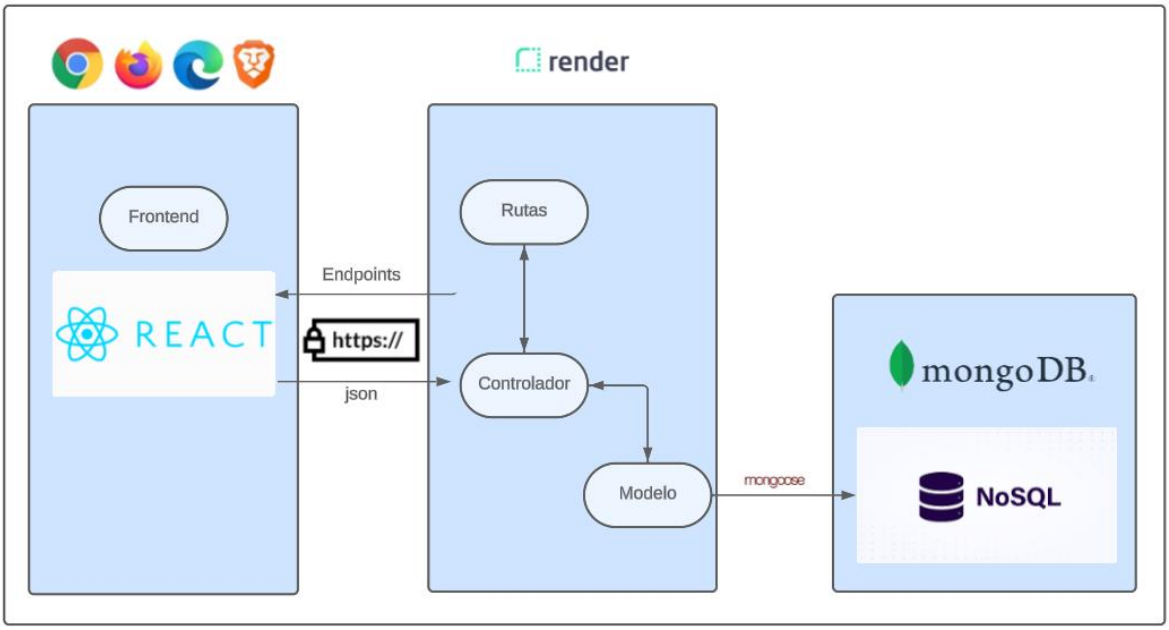


Figura 2. 2: Patrón de arquitectura - *backend*

2.3 Herramientas de desarrollo

Para el desarrollo de aplicaciones web o de *software*, el *backend*, también conocido como "lado del servidor", comprende una serie de componentes esenciales. Estos componentes incluyen servidores, lógica de aplicación, *frameworks*, bases de datos y diversas herramientas que colaboran entre sí para construir y respaldar el funcionamiento del *backend* de la aplicación web o de *software* [27].

En la **Tabla 2. 6** se visualizan las herramientas utilizadas en el *backend* del presente proyecto.

Tabla 2. 6: Herramientas para el desarrollo - *backend*

Herramienta	Justificación
Visual Studio Code	IDE y editor de código fuente que ofrece compatibilidad integrada con JavaScript y Node.js. Proporciona una amplia gama de extensiones que se utilizan en el desarrollo del proyecto *backend* [28].
GitHub y Git	Git permite un eficiente control de versiones del código, mientras que GitHub permite alojar repositorios Git y el código del proyecto [29].

Npm	Herramienta que permite gestionar paquetes de Node.js, facilitando un desarrollo rápido y fácil en el proyecto *backend* [30].
MongoDB	Base de datos NoSQL que permite la gestión de los datos consumidos desde el *backend.* Destacada por su flexibilidad, rendimiento y adaptabilidad a diversos casos de uso [14].
Render.com	Servicio de alojamiento en la nube para desplegar el proyecto *backend.* Se vincula con el repositorio GitHub, actualizándose automáticamente con cada *push* generado [31].

Librerías

Las librerías desarrolladas para diversos lenguajes de programación, incluyendo JavaScript, son herramientas fundamentales que permiten a los desarrolladores evitar la creación de funcionalidades desde cero en cada proyecto, ya sea en el *backend* o en el *frontend*. Estas librerías proporcionan código reutilizable con una serie de funcionalidades estándar que son comúnmente utilizadas [32]. En la **Tabla 2. 7** se presenta un listado de las librerías empleadas durante el desarrollo de los diferentes *endpoints* de este proyecto.

Tabla 2. 7: Librería para el desarrollo - *backend*

Herramienta	Justificación
bcrypt.js	Permite gestionar de manera segura las contraseñas mediante su encriptación [33].
Cors	Paquete de Node.js que permite habilitar el control de acceso a recursos desde diferentes orígenes dentro del proyecto [34].
Dotenv	Gestiona las variables de entorno desde archivo env, protegiendo así las variables sensibles [34].
express	*Framework* de aplicaciones web utilizado para construir y desarrollar el *backend* [35].

express-session	Middleware para Express.js que permite manejar sesiones en el lado del servidor [36].
fs-extra	Agrega funciones adicionales para la manipulación y gestión de sistemas de archivos [37].
jsonwebtoken	Permite implementar y verificar JSON Web Tokens (JWT) [38].
moment	Permite manipular y formatear fechas y horas en JavaScript [39].
mongodb	Controlador oficial para que Node.js se conecte a MongoDB [40].
mongoose	Herramienta de modelado de datos de objetos (ODM) para MongoDB [34].
nodemailer	Simplifica el proceso de envío de correos electrónicos mediante Node.js [34].
passport	Middleware de autenticación para Node.js que simplifica y gestiona el proceso de autenticación [34].
passport-local	Estrategia de autenticación local que permite desarrollar un Middleware de Passport [38].
nodemon	Herramienta de desarrollo que permite reiniciar automáticamente Node.js al detectar cambios realizados en el código [41].

3 RESULTADOS

En esta sección se presentan los logros alcanzados en el desarrollo e implementación de los diferentes *endpoints* en cada *Sprint*, proporcionando una visión detallada de la evolución del proyecto. Se expone la verificación de la lógica del *backend*, describiendo las diferentes pruebas realizadas. Finalmente, se presenta el despliegue en producción, donde se visualizan los resultados obtenidos.

3.1 *Sprint* 0. Configuración del ambiente de desarrollo

Para el *Sprint* 0 se detallan las siguientes tareas:

- Recopilación y definición de los requerimientos
- Creación de la estructura del proyecto *backend*
- Diseño e implementación de base de datos NoSQL
- Definición de roles de usuarios con sus respectivas funcionalidades

Recopilación y definición de los requerimientos

Implementación de *endpoints* para registrar cuentas

Para el usuario con el perfil de Tutor, el *backend* crea *endpoints* específicos para su registro, los cuales contienen campos predeterminados.

Implementación de *endpoints* para restablecer la contraseña

Para el usuario con el perfil de Tutor y Administrador, el *backend* crea *endpoints* específicos para restablecer sus contraseñas mediante campos preestablecidos.

Implementación de *endpoints* para el inicio de sesión y cierre de sesión de usuarios

Para el usuario con el perfil de Tutor, Administrador y Niño, el *backend* crea *endpoints* específicos para su inicio de sesión, los cuales contienen campos predefinidos. Además, el *backend* crea *endpoints* para su cierre de sesión.

Implementación de *endpoints* para la administración de usuarios Niño

Para el usuario con el perfil de Tutor, el *backend* crea *endpoints* específicos para administrar usuarios con el perfil de Niño (Crear, visualizar, modificar, eliminar), asociados a sus cuentas respectivas y utilizando campos preestablecidos.

Implementación de *endpoints* para la visualización de perfiles de usuario

Para el usuario con el perfil de Tutor y Niño, el *backend* crea *endpoints* específicos para la visualización de sus perfiles correspondientes.

Implementación de *endpoints* para restablecer contraseña de usuario con perfil de Niño

Para el usuario con el perfil de Tutor, el *backend* crea *endpoints* específicos para restablecer contraseña de usuario con perfil de Niño asociado a sus cuentas respectivas y mediante campos preestablecidos.

Implementación de *endpoints* para visualizar actividades disponibles

Para el usuario con el perfil de Tutor, el *backend* crea *endpoints* específicos para visualizar las actividades disponibles.

Implementación de *endpoints* para la inscripción de usuario con perfil de Niño

Para el usuario con el perfil de Tutor, el *backend* crea *endpoints* específicos para la inscripción de niños asociados a su cuenta respectiva y con requerimientos predefinidos.

Implementación de *endpoints* para la visualización de actividades en las que el niño está inscrito

Para el usuario con el perfil de Tutor y Niño, el *backend* crea *endpoints* específicos para la visualización de actividades en las que el usuario con perfil de Niño esté previamente inscrito.

Implementación de *endpoints* para el registro y visualización de progreso en actividades

Para el usuario con el perfil de Niño, el *backend* crea *endpoints* específicos para su registro de progreso, mediante campos predefinidos; Además, el *backend* crea *endpoints* para la visualización de su respectivo progreso registrado.

Implementación de *endpoints* para la administración de actividades

Para el usuario con el perfil de Administrador, el *backend* crea *endpoints* específicos para administrar actividades (Crear, visualizar, modificar, eliminar), los cuales contienen campos predeterminados.

Implementación de *endpoints* para la eliminación en cascada de registros de usuarios

Para el usuario con el perfil de Administrador, el *backend* crea *endpoints* específicos para la eliminación de Tutores y datos asociados (Niños, inscripciones, progreso y logros), mediante requerimientos predefinidos.

Implementación de *endpoints* para la modificación de perfil de usuario

Para el usuario con el perfil de Niño, el *backend* crea *endpoints* específicos para la modificación de su perfil correspondiente, mediante campos preestablecidos.

Implementación de *endpoints* para el registro y visualización de logros personales

Para el usuario con el perfil de Niño, el *backend* crea *endpoints* específicos para su registro de logros personales, mediante requerimientos predefinidos.; Además, el *backend* crea *endpoints* para la visualización de sus respectivos logros registrados.

Estructura del proyecto *backend*

La estructura arquitectónica del proyecto *backend*, de acuerdo con el modelo-vista-controlador (MVC) presentado anteriormente, se detalla en la **Figura 3. 1.** Se ha implementado utilizando la herramienta Visual Studio Code, la cual incluye diversos archivos de configuración que siguen este patrón. Estos archivos organizan adecuadamente los *endpoints* utilizados para el desarrollo del componente *backend.*

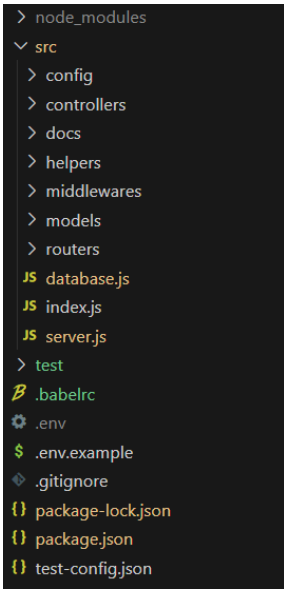


Figura 3. 1: Estructura del proyecto *backend*

Diseño e implementación de base de datos NoSQL

En este proyecto de *backend* se ha utilizado la base de datos NoSQL MongoDB Atlas. En ella, se han implementado los modelos detallados en la **Figura 3. 2** para establecer colecciones específicas que respalden las diversas funcionalidades, garantizando un almacenamiento que promueva la integridad de los datos y una gestión eficiente de la información.

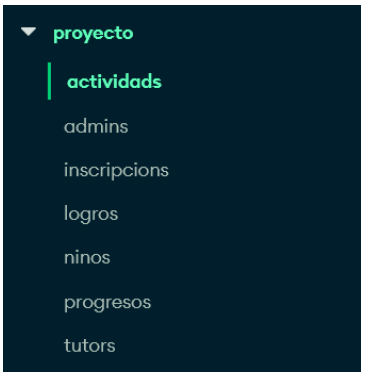


Figura 3. 2: Creación de Base de Datos

Definición de roles de usuarios con sus respectivas funcionalidades

En el desarrollo del proyecto *backend*, se han establecido tres roles de usuario, con uno con funcionalidades específicas. Esta estructura permite mantener un control de acceso preciso y gestionar eficientemente los diverso *endpoints*, como se detalla en la **Figura 3. 3.**

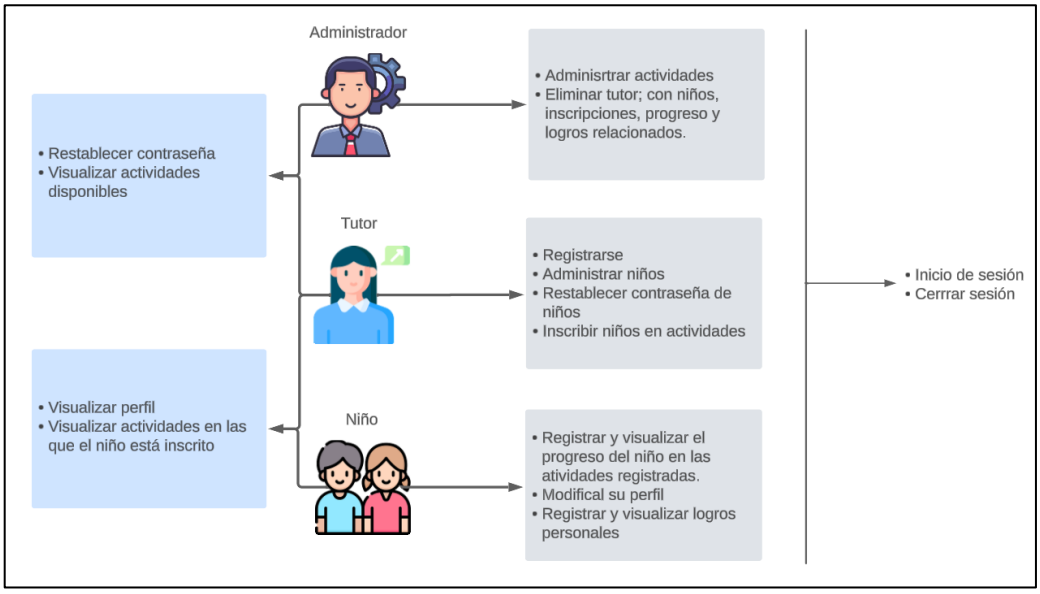


Figura 3. 3: Roles de usuario y sus funcionalidades

3.2 *Sprint* 1. Resultados del diseño e implementación de *endpoints* para registro, inicio de sesión, recuperación de contraseña y perfil de usuario

Para el *Sprint* 1, se presenta las tareas correspondientes para el diseño e implementación de *endpoints* que permiten:

- Registrar cuenta
- Restablecer contraseña
- Iniciar sesión y cerrar sesión
- Visualizar perfil
- Restablecer contraseña de niños
- Modificar perfil

Diseño e implementación de *endpoints* para registrar cuenta

Para el usuario con perfil de Tutor, se han desarrollado *endpoints* públicos de tipo POST destinados a su registro. Estos *endpoints* están diseñados para recibir campos predefinidos que el usuario completa con su información. Posteriormente, estos datos son validados y, una vez que cumplan con un formato apropiado, se crea exitosamente el nuevo usuario, como se observa en la **Figura 3. 4.**

Además, con el fin de verificar el registro del usuario se han implementado *endpoints* para la validación del correo electrónico, como se presenta en la **Figura 3. 5.** Finalmente, en la **Figura 3. 6** se visualizan los resultados de la prueba unitaria realizada para este caso. En el **ANEXO II** se encuentra información adicional.

Figura 3. 4: Registro - Tutor

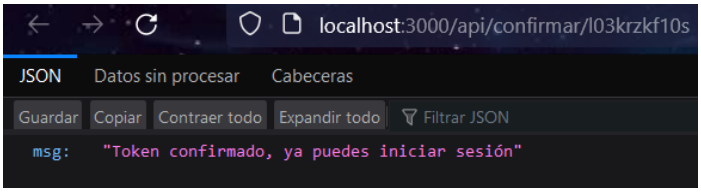


Figura 3. 5: Confirmación de correo electrónico - Tutor

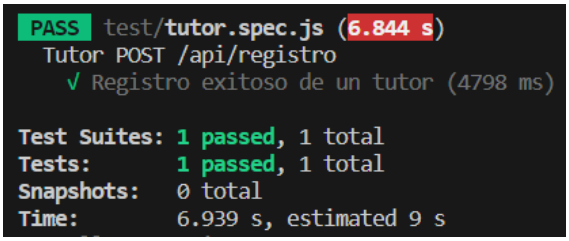


Figura 3. 6: Resultado de prueba unitaria registro - Tutor

Diseño e implementación de *endpoints* para restablecer contraseña

Para el usuario con el perfil de Administrador, se han desarrollado *endpoints* públicos de tipo POST destinados al restablecimiento de su contraseña. Estos *endpoints* requieren que el usuario ingrese su correo electrónico, como se muestra en la **Figura 3. 7,** y posteriormente complete la información con la nueva contraseña, como se presenta en la **Figura 3. 8.** Si la validez de estos datos cumple un formato adecuado, se restablece exitosamente la contraseña. Además, para confirmar la intención del usuario de restablecer su contraseña, se han implementado *endpoints* para la validación del correo electrónico, como se observa en la **Figura 3. 9.** Finalmente, en la **Figura 3. 10** se visualizan los resultados de la prueba unitaria realizada para este caso. En el **ANEXO II** se encuentra información adicional.

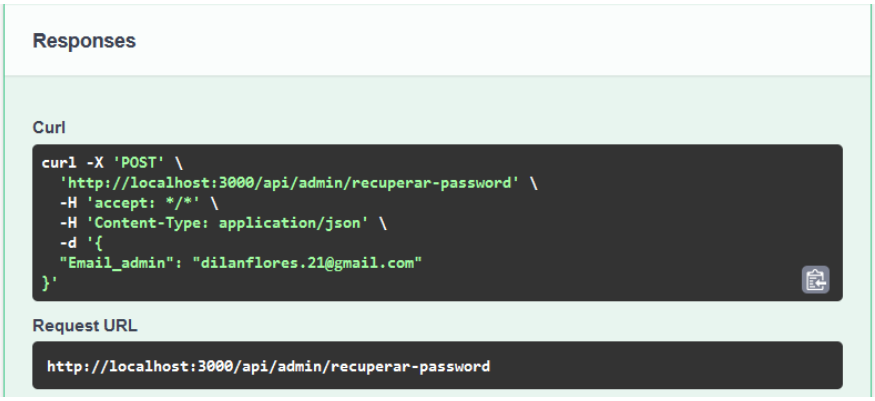


Figura 3. 7: Restablecer contraseña - Administrador

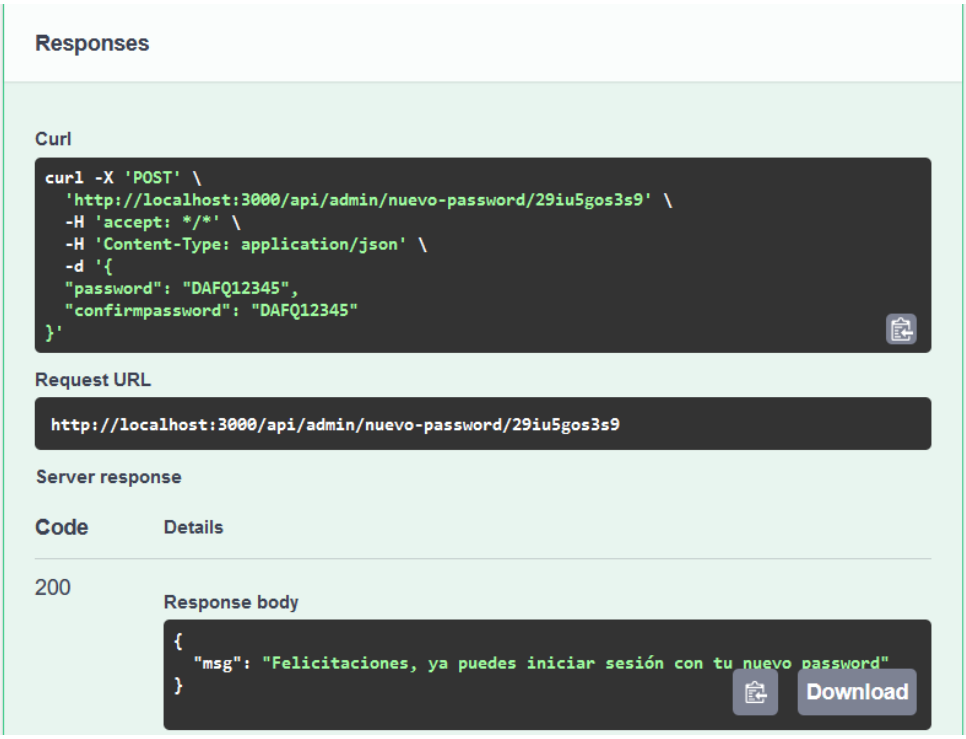


Figura 3. 8: Nueva contraseña – Administrador

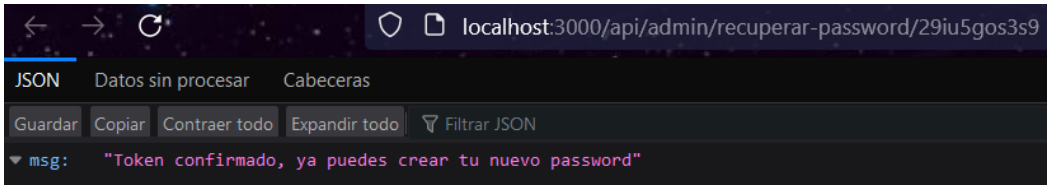


Figura 3. 9: Confirmación de restablecimiento de contraseña - Administrador

Figura 3. 10: Resultado de prueba unitaria restablecer contraseña – Administrador

Para el usuario con el perfil de Tutor, se han desarrollado *endpoints* públicos de tipo POST destinados al restablecimiento de su contraseña. Estos *endpoints* requieren que el usuario ingrese su correo electrónico, como se muestra en la **Figura 3. 11** y posteriormente complete la información con la nueva contraseña, como se presenta en la **Figura 3. 12.** Si la validez de estos datos cumple un formato adecuado, se restablece exitosamente la contraseña. Además, para confirmar la

intención del usuario de restablecer su contraseña, se han implementado *endpoints* para la validación del correo electrónico, como se observa en la **Figura 3. 13.** Finalmente, en la **Figura 3. 14** se visualizan los resultados de la prueba unitaria realizada para este caso. En el **ANEXO II** se encuentra información adicional.

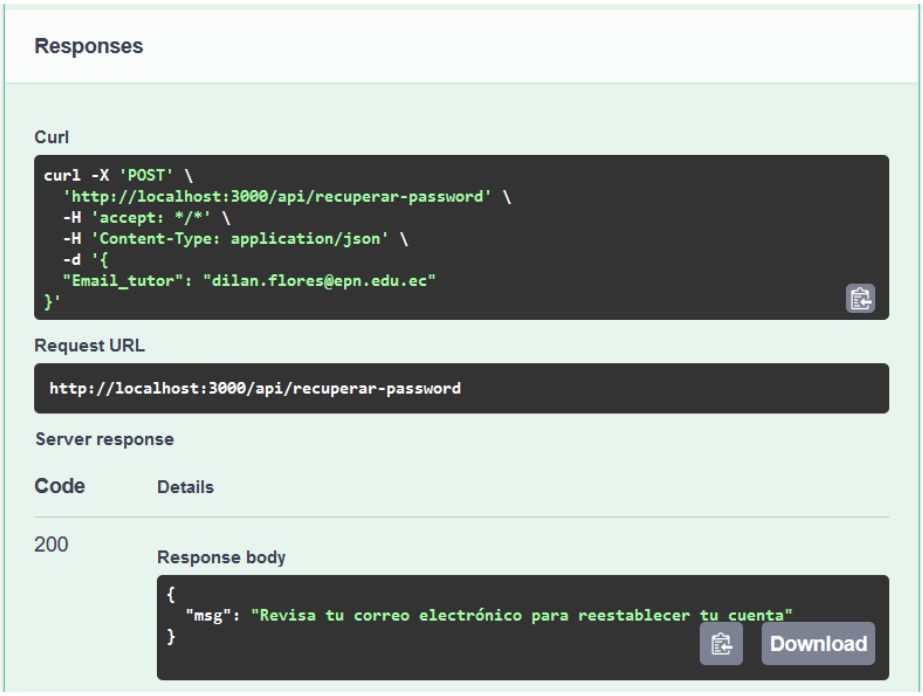


Figura 3. 11: Restablecer contraseña - Tutor

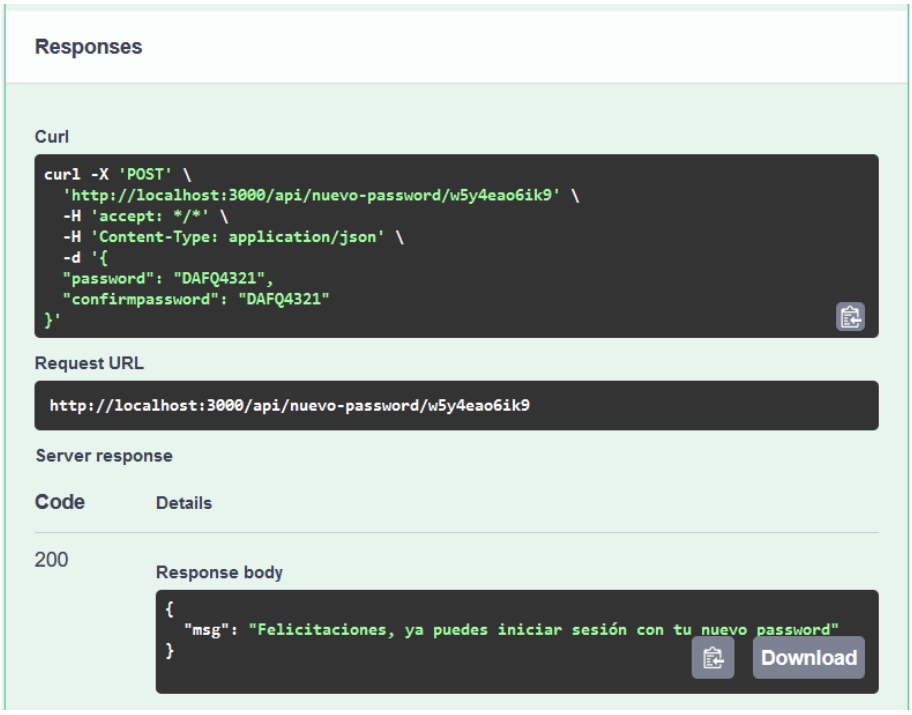


Figura 3. 12: Nueva contraseña – Tutor

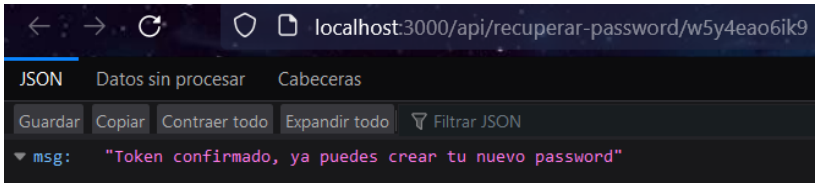


Figura 3. 13: Confirmación de restablecimiento de contraseña - Tutor

Figura 3. 14: Resultado de prueba unitaria restablecer contraseña - Tutor

Diseño e implementación de *endpoints* para iniciar sesión y cerrar sesión

Para el usuario con el perfil de Administrador, se implementaron *endpoints* públicos de tipo POST para su inicio de sesión, mediante campos predefinidos que el usuario completa con su información correspondiente, tal como se visualiza en la **Figura 3. 15.** Los resultados de la prueba unitaria están detallados en la **Figura 3. 16.**

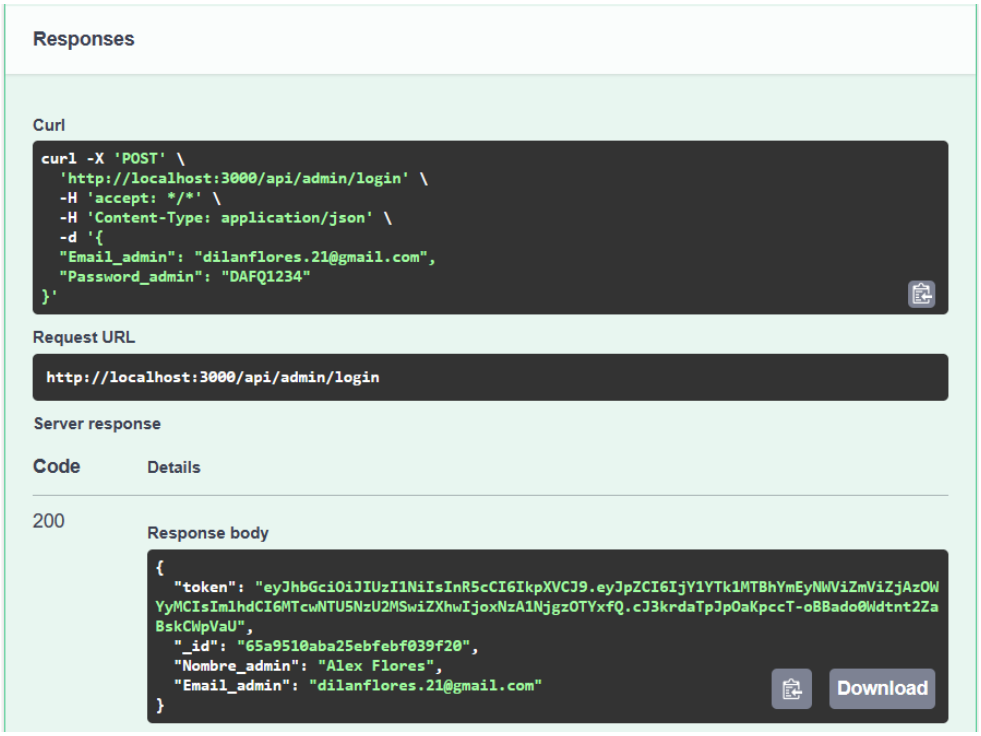


Figura 3. 15: Inicio de sesión - Administrador

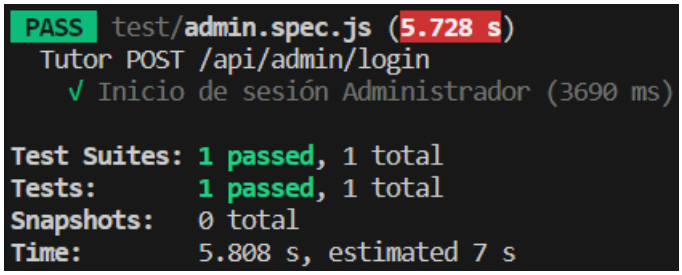


Figura 3. 16: Resultado de prueba unitaria Inicio de sesión – Administrador

Para el usuario con el perfil de Tutor, se implementaron *endpoints* públicos de tipo POST para su inicio de sesión, mediante campos definidos que el usuario completa con su información correspondiente, tal como se visualiza en la **Figura 3. 17.** Los resultados de la prueba unitaria están detallados en la **Figura 3. 18.**

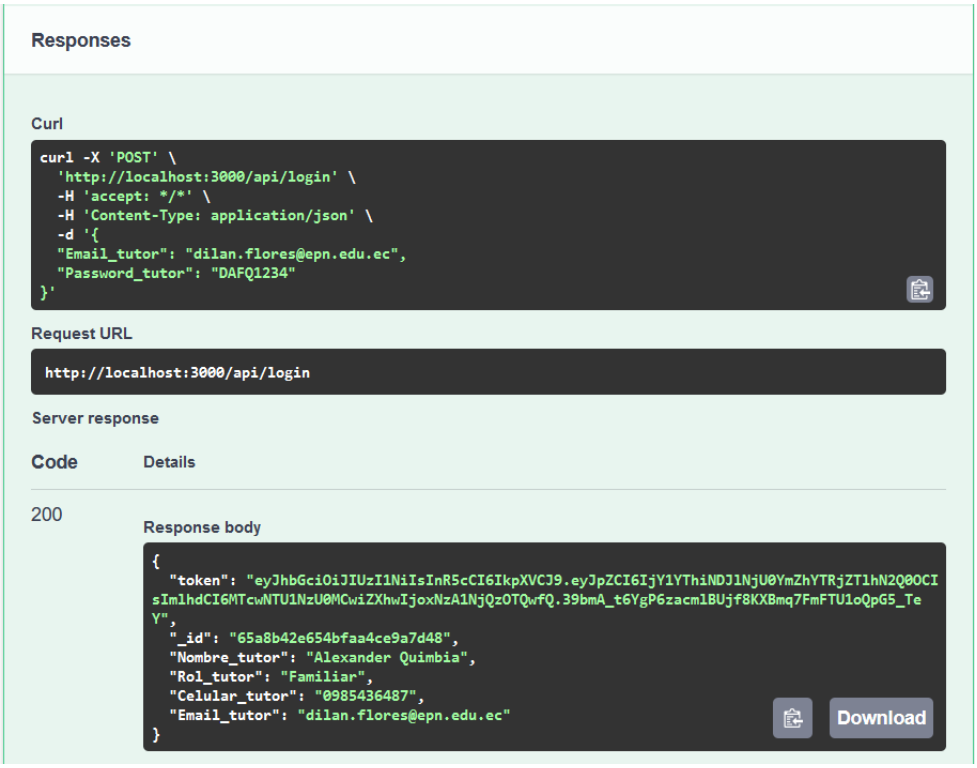


Figura 3. 17: Inicio de sesión - Tutor

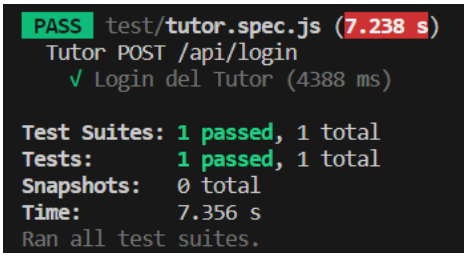


Figura 3. 18: Resultado de prueba unitaria inicio de sesión – Tutor

Para el usuario con el perfil de Niño, se implementaron *endpoints* públicos de tipo POST para su inicio de sesión, mediante campos predefinidos que el usuario completa con su información correspondiente, tal como se visualiza en la **Figura 3. 19.** Los resultados de la prueba unitaria para este caso están detallados en la **Figura 3. 20.**

En cuanto al cierre de sesión, se crearon *endpoints* para este propósito que permiten al usuario Administrador, Tutor y Niño finalicen su sesión de manera efectiva. En el **ANEXO II** se encuentra información adicional.

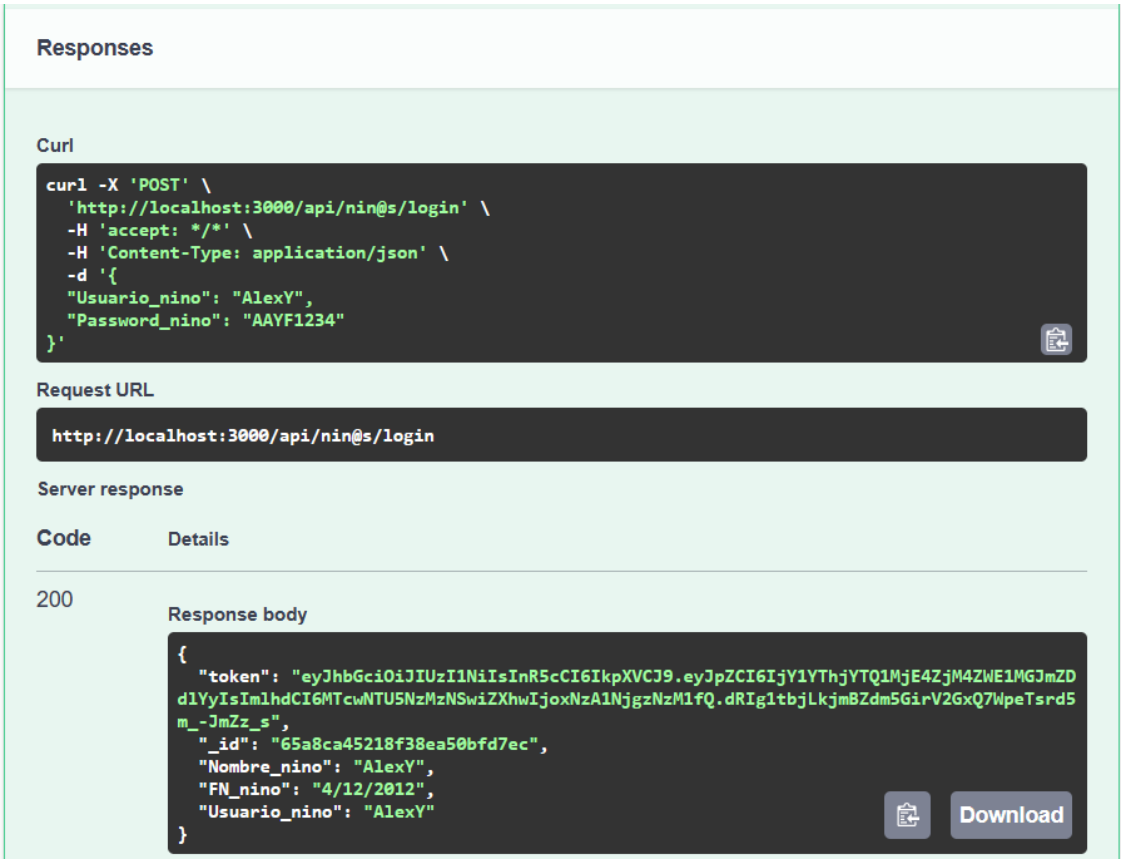


Figura 3. 19: Inicio de sesión - Niño

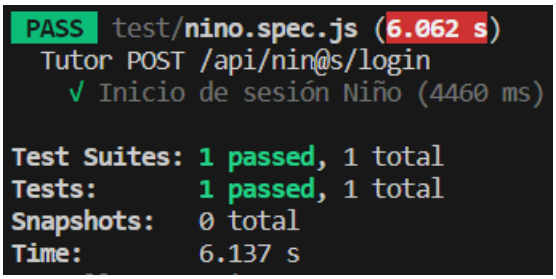


Figura 3. 20: Resultado de prueba unitaria inicio de sesión - Niño

Diseño e implementación de *endpoints* para visualizar perfil

Para el usuario con el perfil de Tutor, se implementaron *endpoints* privados de tipo GET para la visualización de su perfil correspondiente, mediante datos requeridos. Una vez que el usuario completa exitosamente el proceso de inicio de sesión, accede a su propia información mediante su ID, como se expone en la **Figura 3. 21.** Los resultados de la prueba unitaria están detallados en la **Figura 3. 22.**

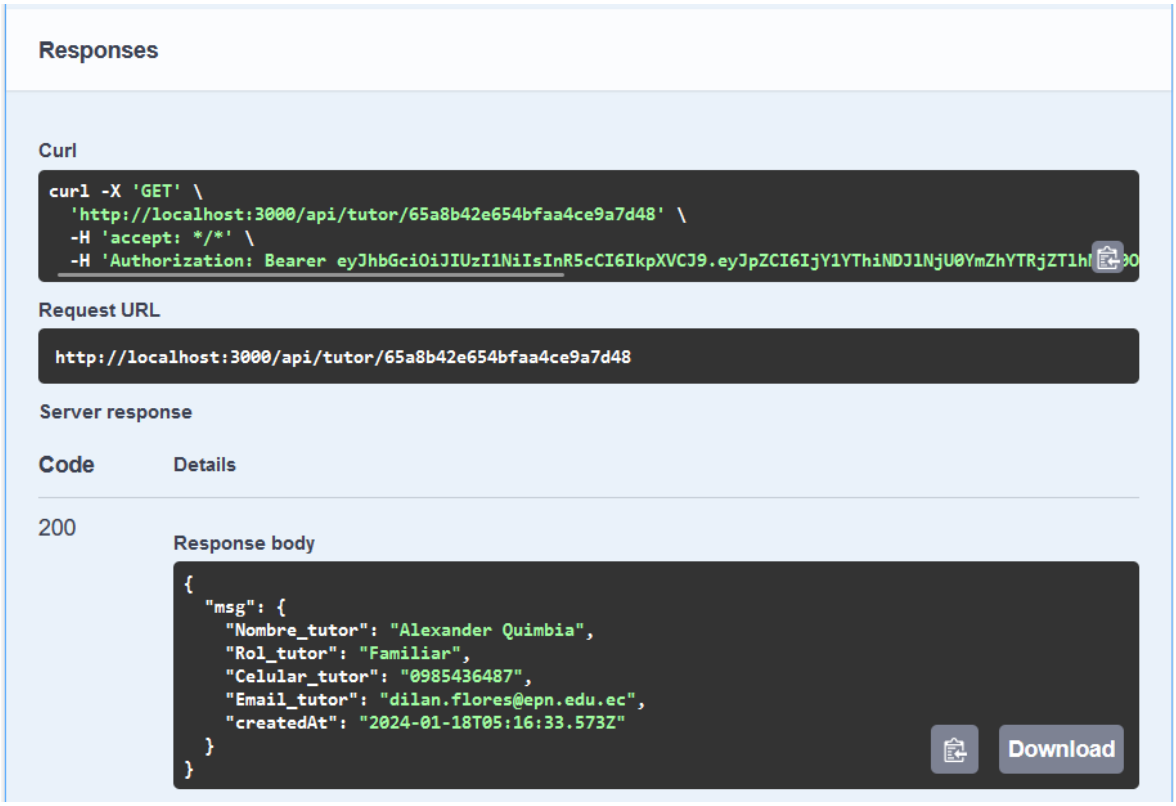


Figura 3. 21: Visualización perfil - Tutor

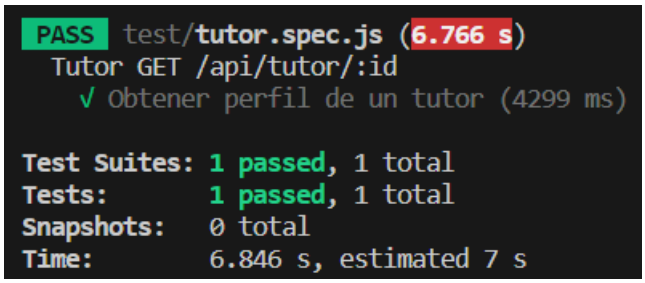


Figura 3. 22: Resultado de prueba unitaria visualización perfil – Tutor

Para el usuario con el perfil de Niño, se implementaron *endpoints* privados de tipo GET para la visualización de su perfil correspondiente, con los datos requeridos. Una vez que el usuario completa exitosamente el proceso de inicio de sesión,

accede a su propia información mediante su ID, como se expone en la **Figura 3. 23.** Los resultados de la prueba unitaria para este caso están detallados en la **Figura 3. 24**. En el **ANEXO II** se encuentra información adicional.

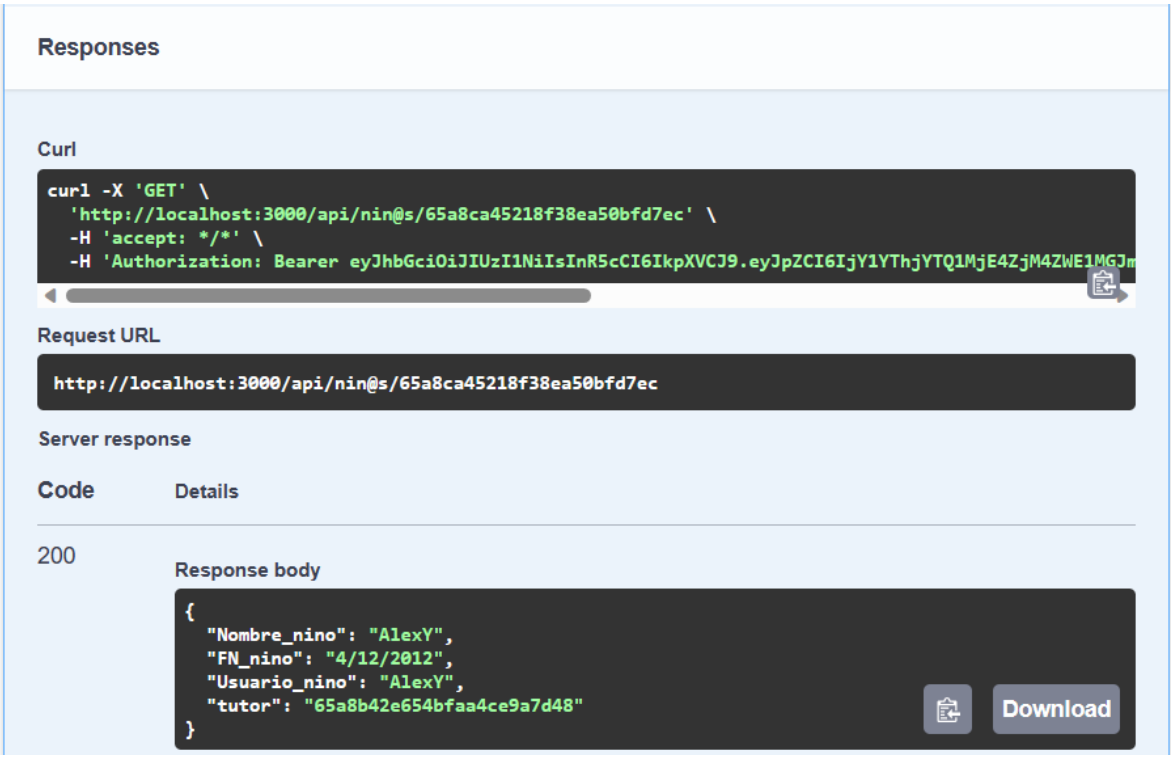


Figura 3. 23: Visualización de perfil - Niño

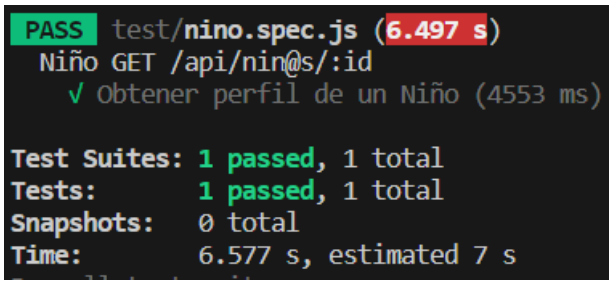


Figura 3. 24: Resultado de prueba unitaria visualización perfil – Niño

Diseño e implementación de *endpoints* para restablecer la contraseña de niños

Para el usuario con el perfil de Tutor, se han desarrollado *endpoints* privados de tipo POST destinados al restablecimiento de la contraseña de usuarios con el perfil de Niño asociados a su cuenta respectiva. Estos *endpoints* requieren que el usuario ingrese un usuario niño, como se muestra en la **Figura 3. 25,** y posteriormente complete la información con la nueva contraseña, como se presenta en la **Figura**

3. 26. Si la validez de estos datos cumple un formato adecuado, se restablece exitosamente la contraseña del usuario Niño. Además, para confirmar la intención de restablecer la contraseña, se han implementado *endpoints* para la validación del correo electrónico del usuario Tutor, como se observa en la **Figura 3. 27**. Finalmente, en la **Figura 3. 28** se visualizan los resultados de la prueba unitaria realizada para este caso. En el **ANEXO II** se encuentra información adicional.

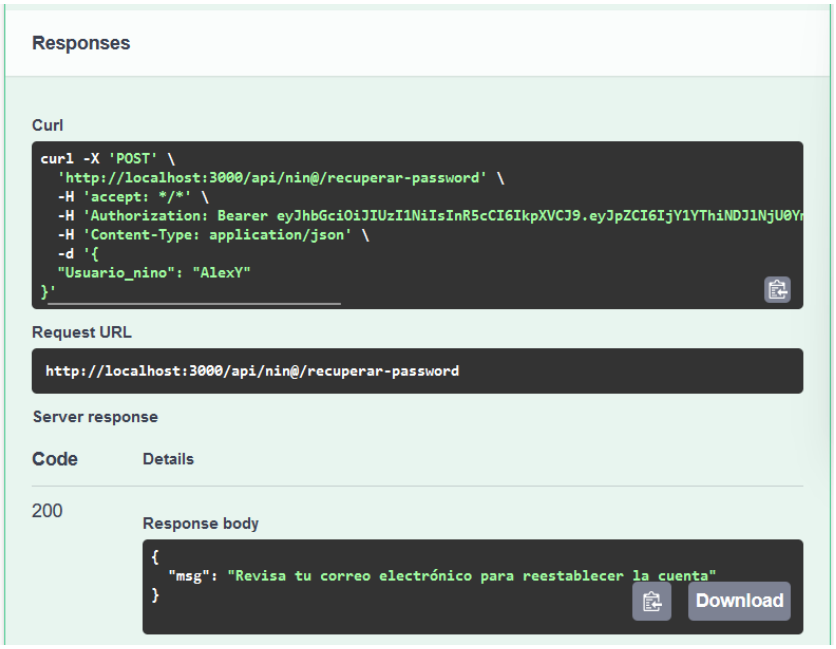


Figura 3. 25: Restablecer contraseña – Niño

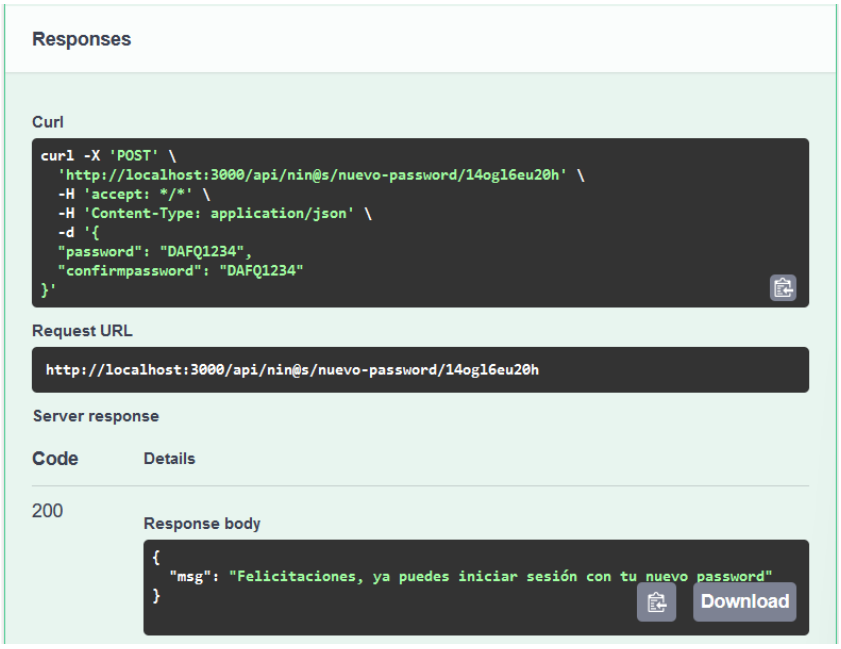


Figura 3. 26: Nueva contraseña - Niño

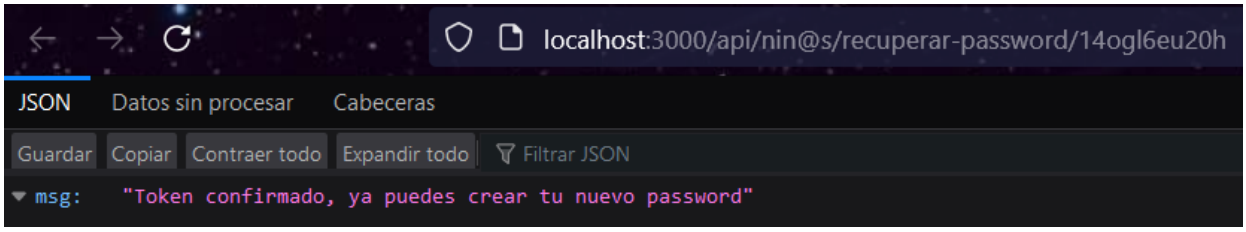


Figura 3. 27: Confirmación de restablecimiento de contraseña - Niño

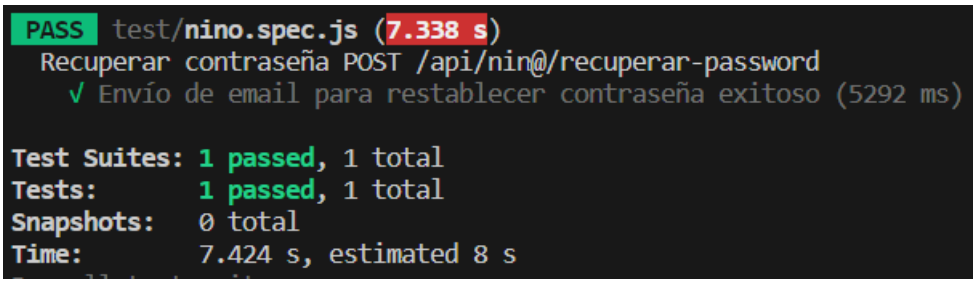


Figura 3. 28: Resultado de prueba unitaria restablecer contraseña – Niño

Diseño e implementación de *endpoints* para modificar perfil

Para el usuario con el perfil de Niño, se implementaron *endpoints* privados de tipo GET para la modificación de su perfil, mediante campos predefinidos que el usuario completa con su información. Una vez que el usuario completa exitosamente el proceso de inicio de sesión, accede a campos destinados para modificar su información, identificado mediante su Id, como se expone en la **Figura 3. 29.** Los resultados de la prueba unitaria para este caso están detallados en la **Figura 3. 30.** En el **ANEXO II** se encuentra información adicional.

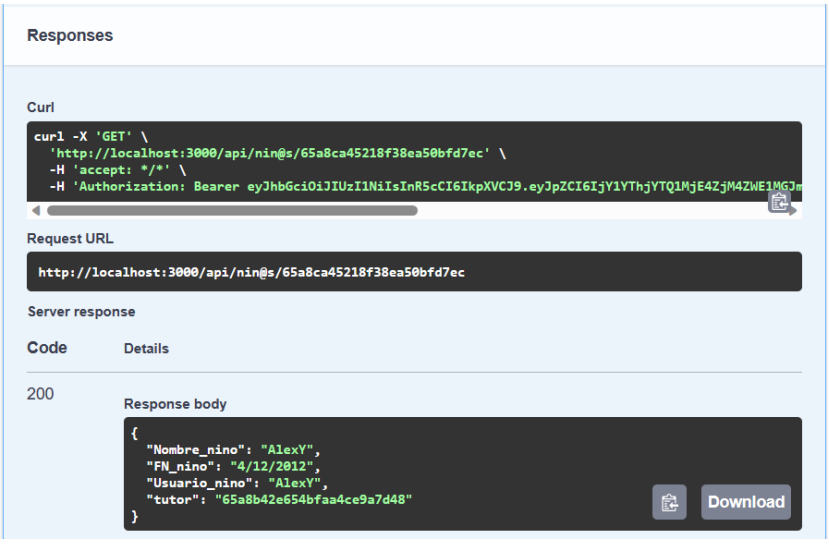


Figura 3. 29: Modificación de perfil - Niño

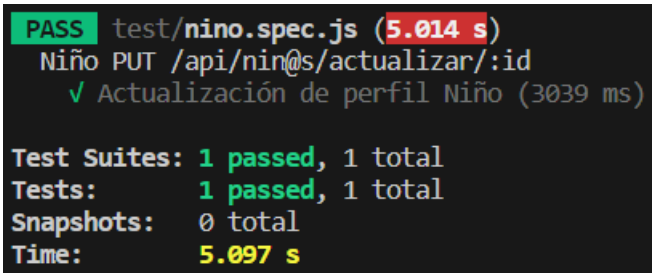


Figura 3. 30: Resultado de prueba unitaria modificación de perfil - Niño

3.3 *Sprint* 2. Resultados del diseño e implementación de *endpoints* para administrar niños y actividades

Para el *Sprint* 2, se presenta las tareas correspondientes para el diseño e implementación de *endpoints* que permiten:

- Administrar niños
- Visualizar actividades disponibles
- Inscribir niño en actividades
- Visualizar actividades en las que el niño está inscrito
- Administrar actividades
- Registrar y visualizar el progreso del niño en las actividades

Diseño e implementación de *endpoints* para administrar niños (Crear, visualizar, actualizar y eliminar)

Para el usuario con el perfil de Tutor, se implementaron *endpoints* privados de tipo POST para el registro de usuarios con el perfil de Niño asociados a su cuenta respectiva, mediante campos preestablecidos que el usuario Tutor completa con información del usuario Niño. Estos datos son validados y, una vez que cumplen con un formato adecuado, se registra exitosamente el nuevo usuario, como se visualiza en la **Figura 3. 31.** Además, con el fin de verificar la creación de un usuario Niño, se ha implementado un *endpoint* para que el usuario Tutor confirme la creación de una nueva cuenta, detallada en la **Figura 3. 32.** Para finalizar esta parte del registro, en la **Figura 3. 33** se presentan los resultados de la prueba unitaria. En el **ANEXO II** se encuentra información adicional.

En la administración de usuarios con el Perfil de Niño, también se implementaron *endpoints* privados de tipo GET, que, una vez finalizado exitosamente el proceso de inicio de sesión, permiten visualizar la información correspondiente mediante un ID registrado. Además, se implementaron *endpoints* privados de tipo PUT para actualizar información y *endpoints* privados de tipo DELETE para eliminar un usuario Niño especificado mediante su ID.

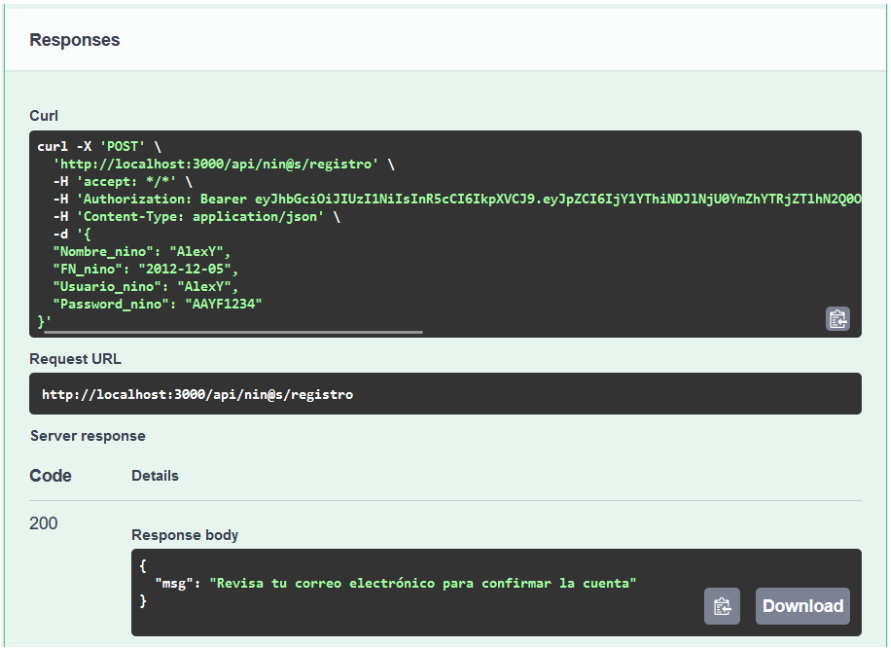


Figura 3. 31: Registro de usuario Niño - Tutor

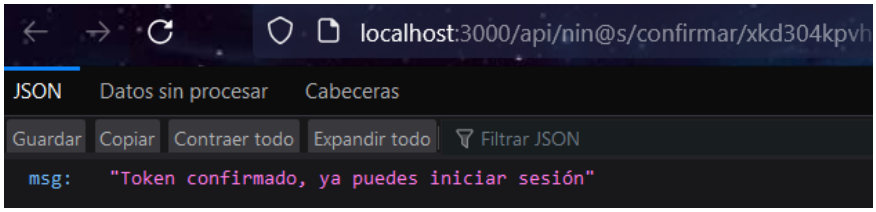


Figura 3. 32: Confirmación de registro usuario Niño - Tutor

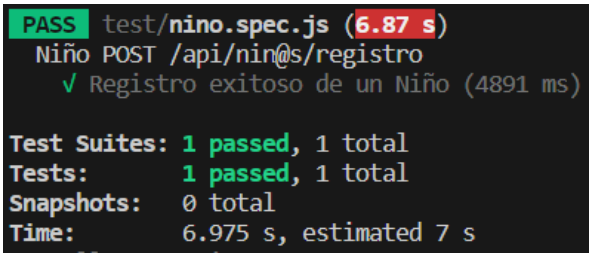


Figura 3. 33: Resultados de prueba unitaria registro de usuario Niño - Tutor

Diseño e implementación de *endpoints* para visualizar actividades disponibles

Para el usuario con el perfil de Tutor, se implementaron *endpoints* privados de tipo GET para la visualización de actividades registradas. Una vez que el usuario completa exitosamente el proceso de inicio de sesión, accede a la información de las actividades disponibles, como se expone en la **Figura 3. 34.** Los resultados de la prueba unitaria para este caso están detallados en la **Figura 3. 35.** En el **ANEXO II** se encuentra información adicional.

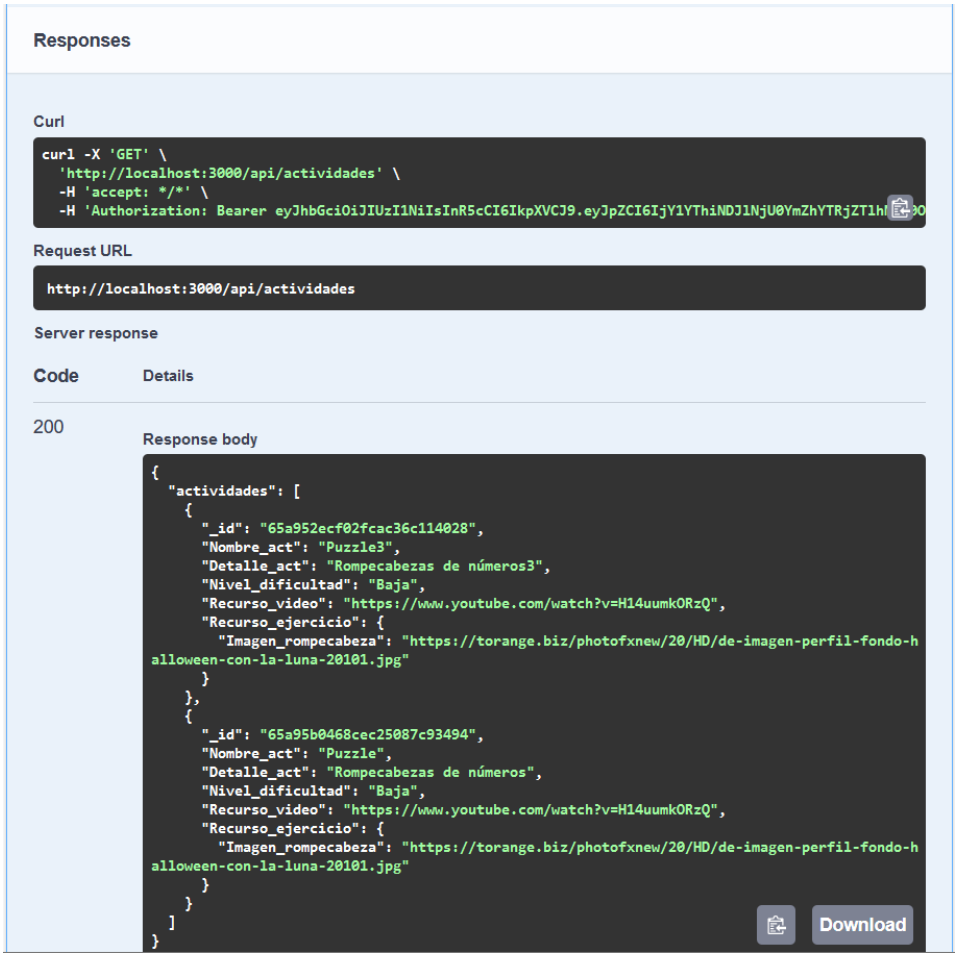


Figura 3. 34: Visualización de actividades

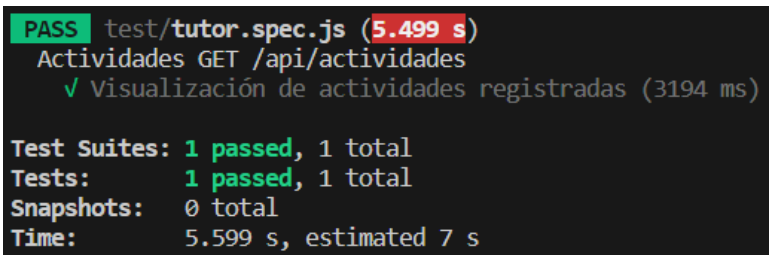


Figura 3. 35: Resultado de prueba unitaria visualización de actividades

Diseño e implementación de *endpoints* para inscribir niños en actividades

Para el usuario con el perfil de Tutor, se implementaron *endpoints* privados de tipo POST para el registro de niños en actividades disponibles, mediante la identificación de ID tanto de un usuario Niño como de una actividad registrada. Una vez que el usuario completa exitosamente el proceso de inicio de sesión, puede acceder a este procedimiento, como se visualiza en la **Figura 3. 36.** Además, en esta parte de inscripción, en la **Figura 3. 37** se presentan los resultados de la prueba unitaria para este caso. En el **ANEXO II** se encuentra información adicional.

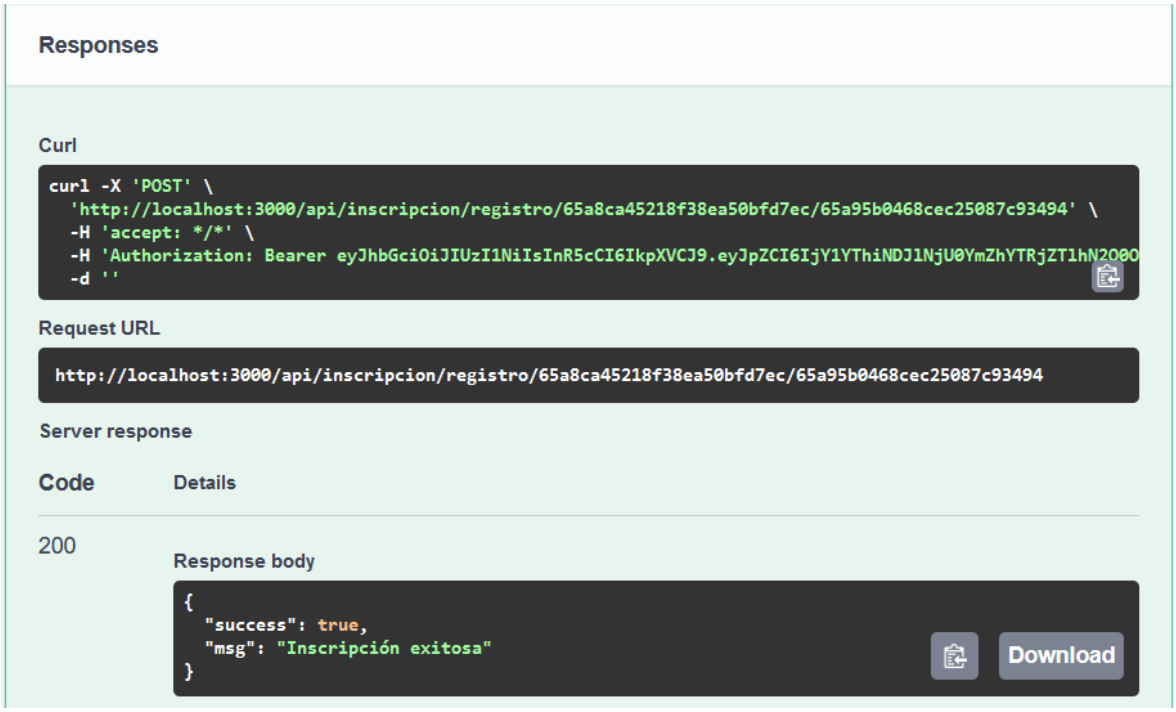


Figura 3. 36: Inscripción de usuario Niño en actividad - Tutor

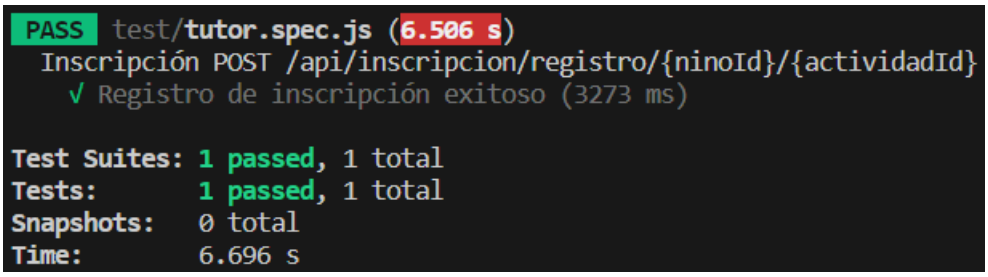


Figura 3. 37: Resultado de prueba unitaria inscripción de usuario Niño en actividad – Tutor

Diseño e implementación de *endpoints* para visualizar actividades en las que el niño está inscrito.

Para el usuario con el perfil de Tutor y Niño, se implementaron *endpoints* privados de tipo GET para la visualización de actividades inscritas por un usuario Tutor para un usuario Niño. Una vez que el usuario completa exitosamente el proceso de inicio de sesión, accede a la información de actividades correspondientes, como se muestra en la **Figura 3. 38.** Los resultados de la prueba unitaria para este caso están detallados en la **Figura 3. 39.** En el **ANEXO II** se encuentra información adicional.

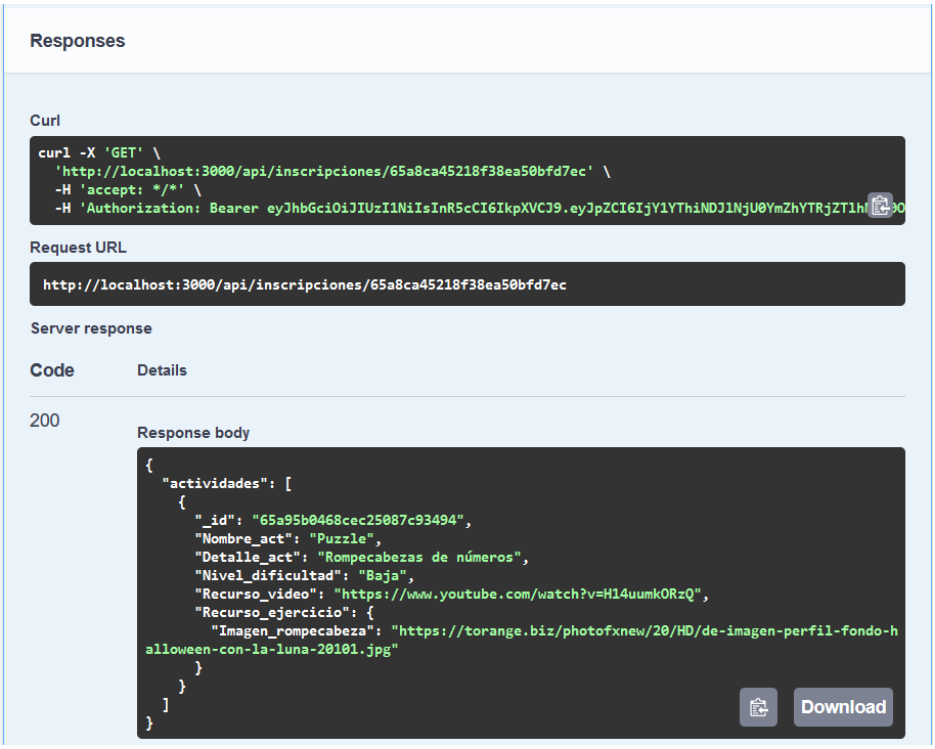


Figura 3. 38: Visualización de actividades en las que un usuario Niño está inscrito

Figura 3. 39: Resultado de prueba unitaria visualización de actividades en las e las que un usuario Niño está inscrito

Diseño e implementación de *endpoints* para administrar actividades

Para el usuario con el perfil de Administrador, se implementaron *endpoints* privados de tipo POST para el registro de actividades, mediante campos preestablecidos que el usuario Tutor completa y, una vez que cumplen con un formato adecuado, se registra exitosamente una nueva actividad, como se visualiza en la **Figura 3. 40.** Además, en esta parte de registro, en la **Figura 3. 41** se presentan los resultados de la prueba unitaria para este caso. En el **ANEXO II** se encuentra información adicional.

En la administración de actividades, también se implementaron *endpoints* privados de tipos GET, que una vez finalizado exitosamente el proceso de inicio de sesión se visualiza la información correspondiente a la actividad por medio de un ID registrado. Además, se han desarrollado *endpoints* privados de tipo PUT, diseñados para la actualización de información, y *endpoints* privados de tipo DELETE, con el propósito de eliminar un registro de actividad por medio de un ID registrado.

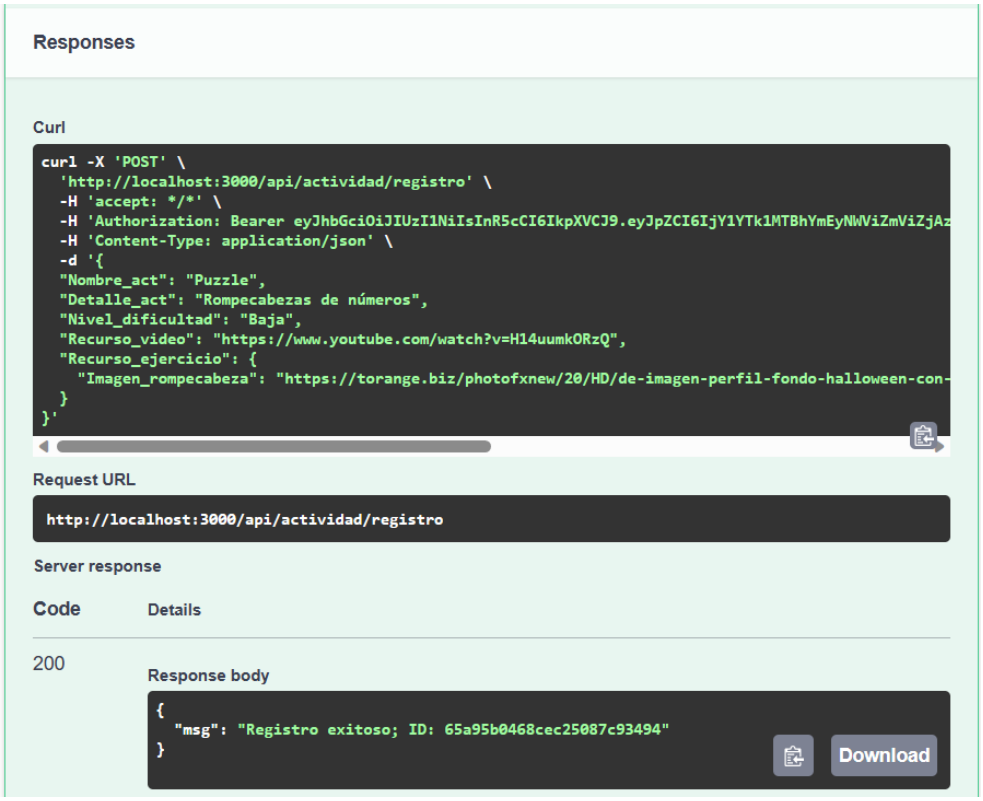


Figura 3. 40: Registro de actividad - Administrador

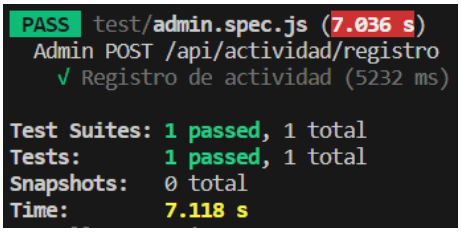


Figura 3. 41: Resultado de prueba unitaria de registro de actividad – Administrador

Diseño e implementación de *endpoints* para registrar y visualizar el progreso del niño en las actividades

Para el usuario con el perfil de Niño, se implementaron *endpoints* privados de tipo POST para el registro del progreso en actividades inscritas. Una vez que el usuario completa exitosamente el proceso de inicio de sesión, puede acceder a los *endpoints* para registrar su progreso basándose en los resultados obtenidos durante una actividad específica, como se muestra en la **Figura 3. 42.** Los resultados de la prueba unitaria para este caso están detallados en la **Figura 3. 43.** En el **ANEXO II** se encuentra información adicional.

En cuanto a la visualización del progreso, una vez que el usuario con el perfil de Niño ha completado satisfactoriamente el proceso de inicio de sesión tiene acceso a *endpoints* privados de tipo GET diseñados específicamente para este propósito.

Figura 3. 42: Registro de proceso - Niño

Figura 3. 43: Resultado de prueba unitaria registro de progreso – Niño

3.4 *Sprint* 3. Resultados del diseño e implementación de *endpoints* para eliminar en cascada y ver logros

Para el *Sprint* 3, se presenta las tareas correspondientes para el diseño e implementación de *endpoints* que permiten:

- Registrar y visualizar logros personales
- Eliminar en cascada: Tutor, niños, inscripciones, progreso y logros

Diseño e implementación de *endpoints* para registrar y visualizar logros personales

Para el usuario con el perfil de Niño, se implementaron *endpoints* privados de tipo POST para el registro de logros personales mediante condiciones predefinidas. Una vez que el usuario completa exitosamente el proceso de inicio de sesión, puede acceder a los *endpoints* para registrar sus logros personales, siempre y cuando cumpla con las condiciones predefinidas, como se muestra en la **Figura 3. 44.** Los resultados de la prueba unitaria para este caso están detallados en la **Figura 3. 45.** En el **ANEXO II** se encuentra información adicional.

En cuanto a la visualización de los logros personales, una vez que el usuario con el perfil de Niño ha completado satisfactoriamente el proceso de inicio de sesión tiene acceso a *endpoints* privados de tipo GET diseñados específicamente para este propósito.

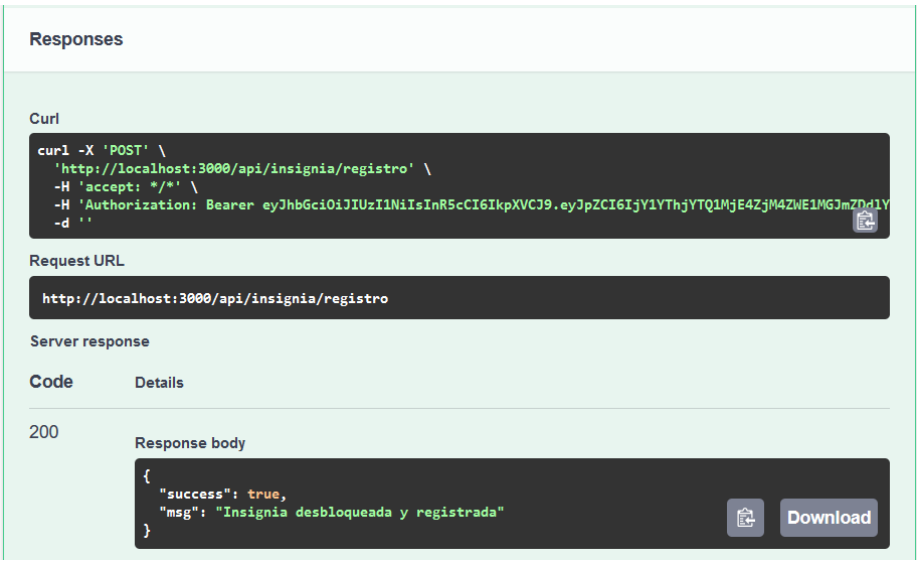


Figura 3. 44: Registro de logro - Niño

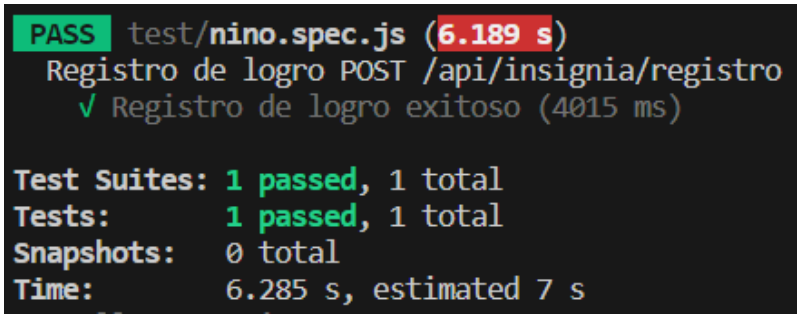


Figura 3. 45: Resultado de prueba unitaria registro de logro – Niño

Diseño e implementación de *endpoints* para que eliminar en cascada: tutor, niños, inscripciones, progreso y logros

Para el usuario con el perfil de Administrador, se han implementado *endpoints* de tipo DELETE destinados a la eliminación completa de registros, utilizando el ID de un usuario Tutor que incluye usuarios Niño a su cargo, inscripciones correspondientes, registros de progreso y logros asociados. Una vez que el usuario ha completado satisfactoriamente el proceso de inicio de sesión puede acceder a los *endpoints* para este caso, como se presenta en la **Figura 3. 46.** Además, en la **Figura 3. 47,** se detallan los resultados correspondientes a estos *endpoints.* En el **ANEXO II** se encuentra información adicional.

Figura 3. 46: Eliminación de tutor y datos asociados - Administrador

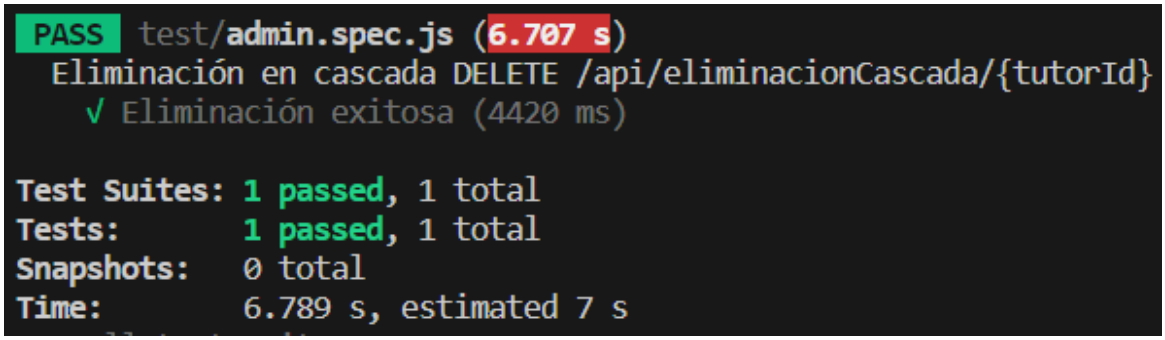


Figura 3. 47: Resultado de prueba unitaria eliminación de tutor y datos asociados - Administrador

3.5 *Sprint* 4. Resultados de las pruebas al componente *backend*

Para el *Sprint* 4, se presenta las siguientes tareas:

- Pruebas unitarias.
- Pruebas de rendimiento.
- Pruebas de carga.

Pruebas unitarias

El proceso de ejecución de pruebas unitarias tiene como objetivo evaluar la funcionalidad de un código específico y asegurar su calidad. Durante este proceso, se prueban unidades pequeñas de código en base a su funcionalidad individual para identificar posibles fallos en áreas específicas. Estas pruebas permiten solucionar eficazmente cualquier error identificado de manera aislada, además de verificar si el código se comporta según lo previsto [42].

En el presente proyecto, se han utilizado las herramientas de prueba "Jest" y "SuperTest" para la realización de pruebas unitarias y de integración, específicamente para el entorno Node.js. En la **Figura 3. 48** se presenta la parte de código correspondiente a la visualización de una actividad registrada mediante su ID, siempre y cuando el usuario complete exitosamente el proceso de inicio de sesión. Además, el resultado de la prueba unitaria para este caso se muestra en la **Figura 3. 49,** confirmando la implementación exitosa del caso. En el **ANEXO II** se encuentra información adicional.

```
describe('Visualizar una actividad GET /api/actividad/{id}', () => {
    test('Visualizar una actividad', async () => {
        // Datos del usuario administrador
        const adminData = {
            Email_admin: "dilanflores.21@gmail.com",
            Password_admin: "DAFQ12345"
        };
        // Realiza una solicitud POST para obtener el token de acceso
        const loginResponse = await request(API)
            .post('/api/admin/login')
            .send(adminData);

        // Extrae el token de acceso del cuerpo de la respuesta
        const token = loginResponse.body.token;
        // Id de una actividad
        const id = "65a95b0468cec25087c93494";
        // Simular solicitud GET para visualizar la actividad
        const response = await request(API)
            .get(`/api/actividad/${id}`)
            .set('Authorization', `Bearer ${token}`);

        console.log(response.status); // Imprime el código de respuesta
        console.log(response.body) // Actividad
        // Verificar que la respuesta sea exitosa (código 200)
        expect(response.status).toBe(200);
    });
});
```

Figura 3. 48: Código de prueba unitaria visualizar actividad - Administrador

```
  console.log
    200

      at Object.log (test/admin.spec.js:126:17)

  console.log
    {
      _id: '65a95b0468cec25087c93494',
      Nombre_act: 'Puzzle',
      Detalle_act: 'Rompecabezas de números',
      Nivel_dificultad: 'Baja',
      Recurso_video: 'https://www.youtube.com/watch?v=H14uumkORzQ',
      Recurso_ejercicio: {
        Imagen_rompecabeza: 'https://torange.biz/photofxnew/20/HD/de-imagen-perfil-fondo-
halloween-con-la-luna-20101.jpg'
      }
    }

      at Object.log (test/admin.spec.js:127:17)

 PASS  test/admin.spec.js (6.108 s)
  Visualizar una actividad GET /api/actividad/{id}
    √ Visualizar una actividad (4046 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        6.202 s
```

Figura 3. 49: Resultado de prueba unitaria visualizar actividad – Administrador

Según los resultados de la prueba para este caso, se confirma un resultado exitoso con el código de estado 200, lo que indica que la prueba ha sido aprobada satisfactoriamente. Esto significa que la visualización de la actividad registrada, mediante su ID, cuando el usuario ha completado exitosamente el proceso de inicio de sesión se ha realizado de manera correcta y sin errores. El código de estado 200 es una indicación estándar en HTTP que representa una respuesta exitosa, lo que confirma que la funcionalidad probada se comporta según lo esperado y cumple con los criterios de aceptación establecidos.

Pruebas de rendimiento

Estas pruebas se centran en evaluar cómo se comporta el sistema bajo condiciones normales de uso y carga. El objetivo principal es medir el tiempo de respuesta, la velocidad y la eficiencia del sistema al procesar solicitudes típicas de usuarios en un entorno de producción. Para llevar a cabo las pruebas de rendimiento en este proyecto *backend*, se utilizó Apache JMeter para simular el acceso a diversos *endpoints* y evaluar su rendimiento, considerando los roles de usuario definidos (Tutor, Niño y Administrador). [43]. Los resultados obtenidos de estas pruebas se detallan en la **Figura 3. 50.**

Etiqueta	# Muestras ↑	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec
Login Tutor	250	531	218	810	44,67	0,00%	1,3/sec	0,75	0,38
Login Niño	250	669	445	827	38,74	0,00%	1,3/sec	0,67	0,37
Login Administrador	250	634	403	785	33,82	0,00%	1,3/sec	0,68	0,39
Perfil Niño	250	390	211	847	116,58	0,00%	1,3/sec	0,45	0,53
Perfil Tutor	250	894	343	1186	115,03	0,00%	1,3/sec	0,54	0,53
Visualizar actividades	250	885	476	1493	77,25	0,00%	1,3/sec	0,75	0,50
Visualizar Inscripciónes	250	1575	657	2024	208,36	0,00%	1,3/sec	0,75	0,54
Visualizar progreso	250	1020	429	1409	185,09	0,00%	1,3/sec	0,52	0,53
Visualizar logro	250	772	323	1211	130,33	0,00%	1,3/sec	0,61	0,51
Total	2250	819	211	2024	346,48	0,00%	11,2/sec	5,66	4,23

Figura 3. 50: Resultados de pruebas de rendimiento

Los resultados de las pruebas de rendimiento realizadas en los 9 *endpoints* muestran valores satisfactorios. Es especialmente destacable el hecho de que el porcentaje de error sea del 0%, lo que evidencia una alta estabilidad y confiabilidad en los *endpoints* del sistema bajo carga normal. Este resultado indica que el sistema pudo manejar la carga de manera eficiente y sin errores significativos durante las pruebas, lo que es fundamental para garantizar una experiencia de usuario consistente y fiable.

Pruebas de carga

Las pruebas de carga se enfocan en evaluar cómo se comporta el sistema bajo condiciones extremas o máximas de carga, superando los límites normales de uso. El objetivo es verificar la capacidad del sistema para manejar una gran cantidad de solicitudes simultáneas y anticipar posibles problemas de rendimiento o recursos bajo presión. [44]

En el desarrollo de este proyecto, se utilizó la herramienta Artillery para llevar a cabo las pruebas de carga. En la **Figura 3. 51** se presenta el código para la ejecución de estas pruebas, y en la **Figura 3. 52** se detallan los resultados de carga para este caso. Además, otras pruebas realizadas están documentadas en el **ANEXO II** para una referencia adicional.

```
    "phases": [
        {
            "duration": 60,
            "arrivalRate": 5
        }
    ]
},
"scenarios": [
    {
        "flow": [
            {
                "post": {
                    "url": "/api/login",
                    "json": {
                        "Email_tutor": "dilan.flores@epn.edu.ec",
                        "Password_tutor": "DAFQ1234"
                    },
                    "capture": {
                        "json": "$.token",
                        "as": "authToken"
                    }
                }
            }
        ]
    }
]
```

Figura 3. 51: Código de prueba de carga

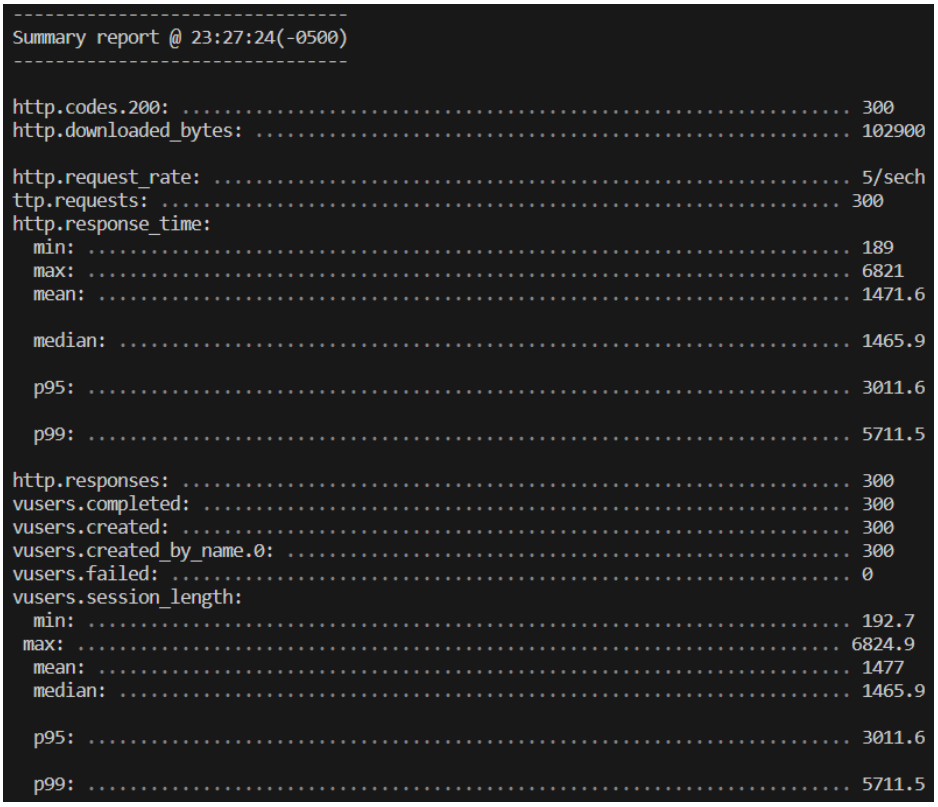


Figura 3. 52: Resultados de ejecución de pruebas de carga

Los resultados obtenidos de las pruebas de carga realizadas para este caso reflejan un comportamiento satisfactorio del componente *backend.* La aprobación del 100% de códigos de respuesta 200 indica que el sistema pudo manejar la carga de manera eficiente, lo que respalda su fiabilidad y robustez. Estos resultados son cruciales para garantizar que el sistema pueda mantener un rendimiento óptimo incluso bajo condiciones de alta demanda.

Además, las pruebas de carga proporcionaron información valiosa para identificar posibles cuellos de botella y áreas de mejora en el sistema. Estos hallazgos permiten optimizar el sistema y mejorar su capacidad de manejo de carga.

3.6 *Sprint* 5. Componente *backend* en entorno de producción

Para el *Sprint* 5, se realizó la siguiente tarea:

- Despliegue del *backend* en Render

Despliegue del *backend* en Render

Después de finalizar el desarrollo y completar las pruebas del proyecto *backend,* se optó por utilizar el servicio en la nube Render para el despliegue en producción. La documentación correspondiente al proyecto ahora está disponible en la siguiente URL:

https://epn-backend.onrender.com/doc

4 CONCLUSIONES

- Una recopilación precisa de requerimientos es esencial para el desarrollo y cumplimiento eficiente de los objetivos planteados. Al definir claramente las necesidades del proyecto desde el principio, se establece una guía para el proceso de desarrollo preciso.
- Con la implementación de la metodología SCRUM, se ha organizado de manera eficiente el desarrollo del componente *backend*, estableciendo un orden adecuado para su desarrollo y asegurando el cumplimiento de los objetivos planteados en plazos adecuados.
- El uso del patrón Modelo-Vista-Controlador (MVC) proporcionó una estructura lógica de desarrollo para los *endpoints* y la configuración del proyecto, facilitando su comprensión y mantenimiento.
- La codificación de los *endpoints* incluyó especificaciones de acceso para usuarios determinados, contribuyendo así a uno de los pilares fundamentales de la seguridad de la información: la confidencialidad.
- En el diseño de la base de datos con MongoDB Atlas, se implementaron datos con tipos específicos y se consideraron restricciones adecuadas para mantener la integridad de los datos.
- La ejecución de las diferentes pruebas ha garantizado el buen funcionamiento de los diferentes *endpoints*, identificando y corrigiendo posibles errores antes del desplegar en un entorno de producción.
- El despliegue del proyecto en un entorno de producción en la nube, como Render, ofrece beneficios significativos. Al utilizar un servidor externo en la nube, se evita la carga de mantenimiento, como la infraestructura y actualización de *software*, lo cual representa una manera adecuada para visualizar la funcionalidad del proyecto desarrollado.

5 RECOMENDACIONES

- Es importante establecer claramente los roles de usuario y asignar las funcionalidades correspondientes a cada tipo para mantener la confidencialidad de la información y garantizar un acceso controlado según los permisos asignados.
- Se recomienda utilizar eficientemente herramientas, metodologías y arquitecturas como SCRUM y el patrón MVC para organizar el tiempo y los archivos de manera efectiva, garantiza una escritura de código fluida y mejora la productividad del equipo de desarrollo.
- Se recomienda realizar pruebas durante todo el desarrollo del proyecto para verificar el buen funcionamiento de cada componente y corregir posibles errores antes de avanzar al desarrollo del siguiente componente. Esto ayuda a identificar problemas temprano y garantiza la calidad del *software* final.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] M. Castro. “El mundo programado por los niños”. GK. Accedido el 25 de diciembre de 2023. [En línea]. Disponible: https://gk.city/2020/10/25/beneficios-aprender-programacion-ninos-ninas

[2] “Presidente Guillermo Lasso inauguró la nueva Unidad Educativa Malchinguí – Ministerio de Educación”. Ministerio de educación. Accedido el 25 de diciembre de 2023. [En línea]. Disponible: https://educacion.gob.ec/presidente-guillermo-lasso-inauguro-la-nueva-unidad-educativa-malchingui/

[3] Dra. E. Martínez. “Desarrollo físico e intelectual de 6 a 12 años - canalSALUD”. Blog Salud MAPFRE. Accedido el 25 de diciembre de 2023. [En línea]. Disponible: https://www.salud.mapfre.es/salud-familiar/ninos/crecimiento-y-desarrollo-nino/desarrollo-fisico-de-6-a-12-anos/#4_Desarrollo_del_cerebro

[4] “Universitario capacitan a niños en programación”. Revista Líderes. Accedido el 25 de diciembre de 2023. [En línea]. Disponible: https://progracademy.org/category/noticias

[5] M. Caparrós. “Programación: una habilidad esencial en la era digital - Smile and Learn”. Smile and Learn. Accedido el 2 de enero de 2024. [En línea]. Disponible: https://www.smileandlearn.com/programacion-una-habilidad-esencial-en-la-era-digital

[6] R. Valdés. “Herramientas para Aprender Programación Web desde Niños – Robin Blog”. Escuela en Línea para Niños y Jóvenes | Robin – Robin. Accedido el 2 de enero de 2024. [En línea]. Disponible: https://robinacademy.com/programacion-web-ninos

[7] Á. Tumbaco. “Estudia el pregrado en Ingeniería de software - UNEMI”. UNEMI. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: https://www.unemi.edu.ec/index.php/carreras-presencial/ingenieria-de-software/#:~:text=La%20Ingeniería%20de%20Software%20es,del%20software%20en%20una%20organización

[8] Max. “Ingeniería web: Qué es, características y todo lo que debes saber”. Carreras Universitarias. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: https://micarrerauniversitaria.com/c-ingenieria/ingenieria-web

[9] Certus. “Descubre en qué es el desarrollo de software | Certus”. Certus Blog | Carreras Técnicas Profesionales. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: https://www.certus.edu.pe/blog/consiste-desarrollo-software

[10] "¿Qué es JavaScript? - Aprende desarrollo web | MDN”. MDN Web Docs. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[11] “Programación lado servidor - Aprende desarrollo web | MDN”. MDN Web Docs. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: https://developer.mozilla.org/es/docs/Learn/Server-side

[12] “Express Web Framework (Node.js/JavaScript) - Aprende desarrollo web | MDN”. MDN Web Docs. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs

[13] DataScientest. “MongoDB: todo sobre la base de datos NoSQL orientada a documentos”. Formation Data Science | DataScientest.com. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: https://datascientest.com/es/mongodb-todo-sobre-la-base-de-datos-nosql-orientada-a-documentos#:~:text=ingeniería%20de%20datos.-

,MongoDB%20es%20una%20base%20de%20datos%20NoSQL%20orientada%20a%20documentos,almacenan%20como%20colecciones%20y%20documentos

[14] “Express Tutorial Part 3: Using a Database (with Mongoose) - Learn web development | MDN”. MDN Web Docs. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose

[15] J. Rea, “Venus: Construcción de una herramienta I-CASE para diseño OO, y su Entorno de Apoyo a Proyectos Integrado (EAPI)”, tesis, Univ. Am. Puebla, 2004. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rea_c_ji/

[16] C. Ortega. “¿Qué es un estudio de caso y cómo realizarlo?” QuestionPro. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: https://www.questionpro.com/blog/es/que-es-un-estudio-de-caso

[17] C. Drumond. “¿Qué es scrum? [+ Cómo empezar]”. Atlassian. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: https://www.atlassian.com/es/agile/scrum

[18] D. West. “Funciones de scrum de metodología ágil”. Atlassian. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: https://www.atlassian.com/es/agile/scrum/roles

[19] C. Harris. “Obtén más información sobre los artefactos del scrum ágil”. Atlassian. Accedido el 26 de diciembre de 2023. [En línea]. Disponible: https://www.atlassian.com/es/agile/scrum/artifacts

[20] J. R. Estévez, “La ingeniería de requisitos en el desarrollo de aplicaciones informáticas”, Revista cubana de informática médica, vol. 12, núm. 2, p. 375, 2020.

[21] M. Rehkopf. “Historias de usuario”. Atlassian. Accedido el 3 de enero de 2024. [En línea]. Disponible: https://www.atlassian.com/es/agile/project-management/user-stories

[22] D. Radigan. “Explicación del backlog del producto [+ ejemplos]”. Atlassian. Accedido el 3 de enero de 2024. [En línea]. Disponible: https://www.atlassian.com/es/agile/scrum/backlogs

[23] M. Rehkopf. “Todo lo que necesitas saber sobre los sprints de scrum”. Atlassian. Accedido el 4 de enero de 2024. [En línea]. Disponible: https://www.atlassian.com/es/agile/scrum/sprints

[24] D. Forero. “Qué es la arquitectura de software: más allá de la programación”. Platzi. Accedido el 4 de enero de 2024. [En línea]. Disponible: https://platzi.com/blog/que-es-arquitectura-de-software/

[25] “MVC - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN”. MDN Web Docs. Accedido el 6 de enero de 2024. [En línea]. Disponible: https://developer.mozilla.org/es/docs/Glossary/MVC

[26] Colaboradores de los proyectos Wikimedia. “Modelo–vista–controlador - Wikipedia, la enciclopedia libre”. Wikipedia, la enciclopedia libre. Accedido el 6 de enero de 2024. [En línea]. Disponible: https://es.wikipedia.org/wiki/Modelo–vista–controlador

[27] A. Ken. “Backend: ¿Qué es y para qué sirve?” Gluo. Accedido el 6 de enero de 2024. [En línea]. Disponible: https://www.gluo.mx/blog/backend-que-es-y-para-que-sirve

[28] “Visual Studio: IDE y Editor de código para desarrolladores de software y Teams”. Visual Studio. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://visualstudio.microsoft.com/es/#vscode-section

[29] E. Castellanos. “Git vs GitHub – ¿Qué es el Control de Versiones y Cómo Funciona?” freeCodeCamp.org. Accedido el 15 de enero de 2024. [En línea].

Disponible: https://www.freecodecamp.org/espanol/news/git-vs-github-what-is-version-control-and-how-does-it-work/

[30] “¿Qué es npm? Una Introducción al Gestor de Paquetes de Node”. Kinsta®. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://kinsta.com/es/base-de-conocimiento/que-es-npm/#:~:text=npm%20es%20un%20gestor%20de,nosotros%20mismos%20durante%20el%20desarrollo.

[31] “Cloud Application Hosting for Developers”. Render.com. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://render.com/

[32] M. Rolfo. “Lo mejor de las Librerías de JavaScript”. Codigoencasa.com. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://codigoencasa.com/librerias-de-javascript/#:~:text=Las%20librerías%20proporcionan%20muchas%20funcionalidades,una%20aplicación%20basada%20en%20JS.

[33] R. Fanizzi. “Registro & Login con nodejs (Jwt, bcrypt) , express, y MySQL (Backend)”. Medium. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://medium.com/@ricardo.fanizzi/registro-login-con-node-js-express-y-mysql-backend-7eea7440837a

[34] S. Bonisteel. “13 Bibliotecas Node.js Para Potenciar Tus Proyectos”. Kinsta®. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://kinsta.com/es/blog/bibliotecas-node-js/

[35] “¿Qué es Express.js? Todo lo que Debes Saber”. Kinsta®. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://kinsta.com/es/base-de-conocimiento/que-es-express/

[36] CodeNerd. “Express.Js Session explained super easy with example”. Medium. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://medium.com/@zakiazizi1841992/express-js-session-explained-super-easy-with-example-908d8bc4bef9

[37] J. Richardson. “fs-extra”. npm. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://www.npmjs.com/package/fs-extra

[38] D. Poza. “Autenticación vía JWT en express.Js (parte 1)”. Creatividad digital. Scientia potentia est - David Poza. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://davidinformatico.com/jwt-express-js-passport

[39] “Moment.js”. Moment.js. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://momentjs.com/

[40] “MongoDB Node.js Driver”. GitHub.io. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://mongodb.github.io/node-mongodb-native/

[41] I. Lopez. “nodemon”. Byspel Tech. Accedido el 15 de enero de 2024. [En línea]. Disponible: https://byspel.com/nodemon/

[42] “¿Qué son las pruebas unitarias?: explicación de las pruebas unitarias en AWS”. Amazon Web Services, Inc. Accedido el 21 de enero de 2024. [En línea]. Disponible: https://aws.amazon.com/es/what-is/unit-testing/

[43] “La guía 2024 de JMeter: Tutorial de pruebas de carga y rendimiento”. LoadView. Accedido el 24 de enero de 2024. [En línea]. Disponible: https://www.loadview-testing.com/es/la-guia-definitiva-de-jmeter-tutorial-de-pruebas-de-carga-y-rendimiento/

[44] G. Lee. “Pruebas de carga de back-end frente a la interfaz de usuario web”. LoadView. Accedido el 21 de enero de 2024. [En línea]. Disponible: https://www.loadview-testing.com/es/blog/pruebas-de-carga-de-back-end-frente-a-la-interfaz-de-usuario-web/

7 ANEXOS

En esta sección se presenta la recopilación de anexos que integran el desarrollo del presente proyecto.

ANEXO I. Certificado de Originalidad

ANEXO II. Manual de usuario.

ANEXO III. Manual de instalación

ANEXO IV. Despliegue en entorno de producción

ANEXO I

CERTIFICADO DE ORIGINALIDAD

TRABAJO DE INTEGRACIÓN CURRICULAR

Quito, D.M. 12 de febrero de 2024

De mi consideración:

Yo, VANESSA KATHERINE GUEVARA BALAREZO, en calidad de Director del Trabajo de Integración Curricular titulado DESARROLLO DE UN BACKEND asociado al DESARROLLO DE UN SISTEMA PARA EL APRENDIZAJE DE PROGRAMACIÓN EN NIÑOS MENORES A 12 AÑOS elaborado por el estudiante DILAN ALEXANDER FLORES QUIMBIA de la carrera en TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE, certifico que he empleado la herramienta antiplagio "TURNITIN" para la revisión de originalidad del documento escrito (sin anexos) producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 10%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el informe generado por la herramienta Turnitin.

Atentamente,

Vanessa Guevara

Docente Ocasional a Tiempo Completo

ESFOT

ANEXO II

Recopilación de requerimientos

Los requerimientos recolectados para el presente proyecto *backend* se reflejan en la **Tabla 1.**

Tabla 1: Recopilación de Requerimientos

RECOPILACION DE REQUERIMIENTOS		
TIPO DE SISTEMA	**ID – RR**	**ENUNCIADO DEL ITEM**
backend	**RR-001**	Como usuario Tutor, se requiere crear *endpoints* para: • Registrar cuenta
	RR-002	Como usuario Tutor y Administrador, se requiere generar *endpoints* para: • Restablecer contraseña
	RR-003	Como usuario Administrador, Tutor y Niño, se requiere generar *endpoints* para: • Iniciar sesión • Cerrar sesión
	RR-004	Como usuario Tutor y Niño, se requiere generar *endpoints* para: • Visualizar perfil
	RR-005	Como usuario Tutor, se requiere generar *endpoints* para: • Administrar niños (Crear perfil, visualizar perfil, modificar perfil, eliminar perfil)
	RR-006	Como usuario Tutor, se requiere generar *endpoints* para: • Restablecer contraseña de niños asociados a su cuenta
	RR-007	Como usuario Tutor requiere generar *endpoints* para: • Visualizar las actividades disponibles

	RR-008	Como usuario Tutor, se requiere generar *endpoints* para: • Inscribir a niños asociados a su cuenta en actividades específicas
	RR-009	Como usuario Tutor y Niño, se requiere generar *endpoints* para: • Visualizar las actividades en las que el niño está inscrito.
	RR-010	Como usuario Niño, se requiere generar *endpoints* para: • Registrar y visualizar el progreso del niño en las actividades
	RR-011	Como usuario Administrador, se requiere generar *endpoints* para: • Administrar las actividades (Crear, visualizar, modificar y eliminar)
	RR-012	Como usuario Administrador, se requiere generar *endpoints* para: • Eliminar en cascada: Tutor, niños, inscripciones, progreso y logros.
	RR-013	Como usuario Niño, se requiere generar *endpoints* para: • Modificar perfil
	RR-014	Como usuario Niño, se requiere generar *endpoints* para: • Registrar y visualizar logros personales

Historias de usuario

Con base a la recopilación de requerimientos, se detalla la creación de Historias de Usuario desde la **Tabla 2** hasta la **Tabla 15.**

Tabla 2: Historia de usuario 1 - Registrar cuenta

HISTORIA DE USUARIO	
Identificador: HU-001	**Usuario:** Tutor
Nombre Historia: Registrar cuenta	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Media
Iteración asignada: 1	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para que el usuario con perfil Tutor pueda registrarse.	
Observación: Los usuarios con perfil Tutor necesitan registrar su cuenta para acceder a los *endpoints* asignados. Para el usuario con perfil Administrador se registra una cuenta sin un *endpoint* correspondiente.	

Tabla 3: Historia de usuario 2 - Recuperar contraseña

HISTORIA DE USUARIO	
Identificador: HU-002	**Usuario:** Tutor y Administrador
Nombre Historia: Restablecer contraseña	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Media
Iteración asignada: 1	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para que el usuario con perfil Tutor y Administrador restablezcan la contraseña.	
Observación: Los correos electrónicos de los usuarios con perfil Tutor y Administrador necesitan estar registrados para acceder a los *endpoints* mencionados. Para recuperar la contraseña se enviará un correo electrónico al usuario respectivo.	

Tabla 4: Historia de usuario 3 - Inicio sesión, cerrar sesión

HISTORIA DE USUARIO	
Identificador: HU-003	**Usuario:** Administrador, Tutor y Niño
Nombre Historia: Iniciar sesión, cerrar sesión	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Media

Iteración asignada: 1
Responsable: Dilan Flores
Descripción: El *backend* permite generar *endpoints* para: • Iniciar Sesión • Cerrar Sesión
Observación: Los usuarios con perfil Administrador, Tutor y Niño necesitan estar registrados para acceder a los *endpoints* mencionados.

Tabla 5: Historia de usuario 4 - Visualizar perfil

HISTORIA DE USUARIO	
Identificador: HU-004	**Usuario:** Tutor, Niño
Nombre Historia: Visualizar perfil	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Media
Iteración asignada: 1	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para visualizar el perfil respectivo.	
Observación: Los usuarios con perfil Tutor y Niño una vez iniciado sesión, pueden acceder a *endpoints* para visualizar sus perfiles	

Tabla 6: Historia de usuario 5 - Administrar niños

HISTORIA DE USUARIO	
Identificador: HU-005	**Usuario:** Tutor
Nombre Historia: Administrar niños	
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Media
Iteración asignada: 2	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para: • Administrar niños (CRUD)	
Observación: Los usuarios con perfil Tutor una vez iniciado sesión, pueden acceder a *endpoints* para crear, visualizar, modificar y eliminar usuarios con perfil Niño.	

Tabla 7: Historia de usuario 6 - Modificar contraseña de niños

HISTORIA DE USUARIO	
Identificador: HU-006	**Usuario:** Tutor
Nombre Historia: Restablecer contraseña de niños	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Media
Iteración asignada: 1	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para: • Restablecer contraseña de niños asociados al tutor	
Observación: Los usuarios de perfil Niños necesitan estar registrados para acceder a los *endpoints* mencionados. Para realiza el proceso de restablecer la contraseña y cambiar la misma, se enviará un correo electrónico al Tutor.	

Tabla 8: Historia de usuario 7 - Visualizar actividades disponibles

HISTORIA DE USUARIO	
Identificador: HU-007	**Usuario:** Tutor
Nombre Historia: Visualizar actividades disponibles	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Media
Iteración asignada: 2	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para: • Visualizar las actividades disponibles	
Observación: Los usuarios con perfil Tutor una vez iniciado sesión, pueden acceder a los *endpoints* y visualizar las actividades disponibles.	

Tabla 9: Historia de usuario 8 - Inscripción niños en actividades

HISTORIA DE USUARIO	
Identificador: HU-008	**Usuario:** Tutor
Nombre Historia: Inscribir niños en actividades	
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Media
Iteración asignada: 2	

Responsable: Dilan Flores
Descripción: El *backend* permite generar *endpoints* para: • Inscribir a niños asociados al tutor en actividades específicas
Observación: Los usuarios con perfil Tutor una vez iniciado sesión, pueden agregar actividades a niños asociados a su cuenta.

Tabla 10: Historia de usuario 9 - Actividades en las que el usuario niño está inscrito

HISTORIA DE USUARIO	
Identificador: HU-009	**Usuario:** Tutor y Niño
Nombre Historia: Visualizar actividades en las que el niño está inscrito	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Media
Iteración asignada: 2	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para: • Visualizar las actividades en las que el niño está inscrito.	
Observación: Los usuarios con perfil Tutor y Niño una vez iniciado sesión, pueden acceder a los *endpoints* para visualizar las actividades en las que el niño está inscrito. El usuario Tutor puede visualizar la información respectiva de niños asociados a su cuenta.	

Tabla 11: Historia de usuario 10 - Progreso de niño

HISTORIA DE USUARIO	
Identificador: HU-010	**Usuario:** Niño
Nombre Historia: Registrar y visualizar el progreso del niño en las actividades	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Media
Iteración asignada: 2	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para: • Registrar y visualizar el progreso del niño en las actividades.	
Observación: Los usuarios con perfil de Niño una vez iniciado sesión, pueden acceder a los *endpoints* para registrar y visualizar su progreso en las actividades.	

Tabla 12: Historia de usuario 11 - Administrar actividades

HISTORIA DE USUARIO	
Identificador: HU-011	**Usuario:** Administrador
Nombre Historia: Administrar actividades	
Prioridad en el negocio: Alta	**Riesgo en el desarrollo:** Media
Iteración asignada: 2	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para: • Administrar las actividades (CRUD)	
Observación: Los usuarios con perfil Administrador una vez iniciado sesión, pueden acceder a los *endpoints* para registrar, visualizar, modificar y eliminar actividades.	

Tabla 13: Historia de usuario 12 - Eliminar en cascada: usuario Tutor y datos asociados

HISTORIA DE USUARIO	
Identificador: HU-012	**Usuario:** Administrador
Nombre Historia: Eliminar en cascada: Tutor, niños, inscripciones, progreso y logros	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Media
Iteración asignada: 3	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para: • Eliminar en cascada: Tutor, niños, inscripciones, progreso y logros	
Observación: Los usuarios con perfil Administrador una vez iniciado sesión, pueden eliminar tutores y sus registros en cascada. Es decir, todos los registros que tengan como *Foreign Key (FK)* el ID del Tutor, y, el ID de los niños asociados a la cuenta respectiva del Tutor.	

Tabla 14: Historia de usuario 13 - Modificar perfil

HISTORIA DE USUARIO	
Identificador: HU-013	**Usuario:** Niño
Nombre Historia: Modificar perfil	

Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Media
Iteración asignada: 1	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para: • Modificar perfil	
Observación: Los usuarios con perfil de Niño una vez iniciado sesión, pueden modificar su propio perfil.	

Tabla 15: Historia de usuario 14 - Logros personales

HISTORIA DE USUARIO	
Identificador: HU-014	**Usuario:** Niño
Nombre Historia: Registrar y visualizar logros personales	
Prioridad en el negocio: Media	**Riesgo en el desarrollo:** Media
Iteración asignada: 3	
Responsable: Dilan Flores	
Descripción: El *backend* permite generar *endpoints* para: • Registrar y visualizar logros personales	
Observación: Una vez que los usuarios con perfil de Niño inicien sesión, se registrarán sus logros cuando cumplan exitosamente ciertas condiciones en el proceso, como completar un número determinado de actividades u obtener buenas puntuaciones.	

Product Backlog

En la **Tabla 16** se presenta el *Product Backlog* en base a las Historias de Usuario.

Tabla 16: *Product Backlog*

ELABORACION DEL *PRODUCT BACKLOG*				
HU	**HISTORIA DE USUARIO**	**ITERACION**	**ESTADO**	**PROIRIDAD**
HU-001	Registrar cuenta	1	Terminado	Media

HU-002	Restablecer contraseña	1	Terminado	Media
HU-003	Iniciar sesión, cerrar sesión	1	Terminado	Media
HU-004	Visualizar perfil	1	Terminado	Media
HU-005	Administrar niños	2	Terminado	Alta
HU-006	Restablecer contraseña de niños	1	Terminado	Media
HU-007	Visualizar actividades disponibles	2	Terminado	Media
HU-008	Inscribir niños en actividades	2	Terminado	Alta
HU-009	Visualizar actividades en las que el niño está inscrito	2	Terminado	Media
HU-010	Registrar y visualizar progreso de niño	2	Terminado	Media
HU-011	Administrar actividades	2	Terminado	Alta
HU-012	Eliminar en cascada: Tutor, niño, actividad y datos asociado	3	Terminado	Media
HU-013	Modificar perfil	1	Terminado	Media
HU-014	Registrar y visualizar logros personales	3	Terminado	Media

Sprint Backlog

En la **Tabla 17** se detalla el desarrollo de cada *Sprint* del presente proyecto, con las funcionalidades específicas designadas para cada usuario y el tiempo de ejecución de cada *Sprint.*

Tabla 17: *Sprint Backlog*

ELABORACION DEL *SPRINT BACKLOG*						
ID-SB	**NOMBRE**	**MÓDULO**	**HU**	**HISTORIAS DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB-000	Configuración del ambiente de desarrollo	N/A	N/A	N/A	• Recopilación y definición de los requerimientos • Creación de la estructura del proyecto *backend* • Diseño e implementación de base de datos NoSQL • Definición de roles de usuarios con sus respectivas funcionalidades	**10 H**
SB-001	Diseño e implementación de *endpoints* para registro, inicio de sesión,	Módulo de registro	HU 001	Registrar cuenta	• Diseño e implementación de *endpoints* para registrar cuenta del Tutor. • Validación de los datos requeridos. • Registro en la base de datos.	**80 H**

	recuperación de contraseña y perfil de usuario	Módulo de recuperar contraseña	HU 002	Restablecer contraseña	• Diseño e implementación de *endpoints* para que el usuario Tutor y Administrador restablezcan su contraseña. • Validación de los datos requeridos. • Registro en la base de datos	
		Módulo de inicio de sesión	HU 003	Iniciar sesión y cerrar sesión	• Diseño e implementación de *endpoints* para que el usuario Tutor, Niño y Administrador inicien sesión y cierren sesión. • Validación de los datos requeridos.	
		Módulo de Visualizar usuario	HU 004	Visualizar perfil	• Diseño e implementación de *endpoints* para que el usuario Tutor y Niño visualicen su perfil. • Validación de los datos requeridos.	
		Módulo de recuperar contraseña de niños	HU 006	Restablecer contraseña de niños	• Diseño e implementación de *endpoints* para que el usuario Tutor restablezca la contraseña de niños asociados a su cuenta. • Validación de los datos requeridos. • Registro en la base de datos.	

		Módulo de modificar perfil	HU 013	Modificar perfil	• Diseño e implementación de *endpoints* para que el usuario Niño modifique su perfil • Validación de los datos requeridos. • Registro en la base de datos.	
SB-002	Diseño e implementación de *endpoints* para administrar niños y actividades	Módulo de administrar niños	HU 005	Administrar niños	• Diseño e implementación de *endpoints* para que el usuario Tutor administre niños (Crear, visualizar, actualizar y eliminar). • Validación de los datos requeridos. • Registro en la base de datos	**80 H**
		Módulo de visualizar actividades disponibles	HU 007	Visualizar actividades disponibles	• Diseño e implementación de *endpoints* para que el usuario Tutor visualice actividades disponibles. • Validación de los datos requeridos.	

		Módulo de inscribir niños en actividades	HU 008	Inscribir niño en actividades	• Diseño e implementación de *endpoints* para que el usuario Tutor inscriba niños en actividades. • Validación de los datos requeridos. • Registro en la base de datos.	
		Módulo de visualizar actividades en las que el niño está inscrito	HU 009	Visualizar actividades en las que el niño está inscrito	• Diseño e implementación de *endpoints* para que el usuario Tutor y Niño visualice las actividades en las que el niño está inscrito. • Validación de los datos requeridos.	
		Módulo de administrar actividades	HU 011	Administrar actividades	• Diseño e implementación de *endpoints* para que el usuario Administrador administre actividades (crear, visualizar, actualizar y eliminar). • Validación de los datos requeridos. • Registro en la base de datos	

		Módulo de Registrar y visualizar el progreso de niño	HU 010	Registrar y visualizar el progreso del niño en las actividades	• Diseño e implementación de *endpoints* para que el usuario Niño visualice el progreso en el que se ha registrado automáticamente al ingresar a realizar una actividad. • Validación de los datos requeridos. • Registro en la base de datos.	
SB-003	Diseño e implementación de *endpoints* para eliminar en cascada y ver logros	Módulo de registrar y visualizar logros personales	HU 014	Registrar y visualizar logros personales	• Diseño e implementación de *endpoints* para que el usuario Niño visualice sus logros personales, en los que se ha registrado automáticamente al completar con éxito algunos requisitos definidos. • Validación de los datos requeridos. • Registro en la base de datos.	**30 H**
		Módulo de Eliminar en cascada: Tutor, niños, inscripciones, progreso y logros	HU 012	Eliminar en cascada: Tutor, niños, inscripciones, progreso y logros	• Diseño e implementación de *endpoints* para que el usuario Administrador elimine al tutor, niños, inscripciones, progreso y logros. • Validación de los datos requeridos. • Registro en la base de datos.	

SB-004	Pruebas al componente *backend*	• Pruebas unitarias • Pruebas de rendimiento • Pruebas de carga	**10 H**
SB-005	Componente *backend* en entorno de producción	• Despliegue del *backend* en Render	**10 H**
Documentación		• Trabajo de integración curricular • Anexos	**20 H**
Total			**240 H**

Pruebas

Pruebas Unitarias

Las pruebas unitarias pendientes para los *endpoints* del proyecto *backend* se detallan a continuación:

En la **Figura 1** se muestra el código de la prueba unitaria correspondiente a la modificación de una actividad, una vez que un usuario Administrador haya completado exitosamente el proceso de inicio de sesión. En la **Figura 2** se detallan los resultados de la prueba unitaria de este caso.

En la **Figura 3** se presenta el código de la prueba unitaria destinada a la eliminación de una actividad. Los resultados del caso se muestran en la **Figura 4.**

```
describe('Actualizar una actividad PUT /api/actividad/actualizar/{id}', () => {
    test('Actualizar una actividad', async () => {
        // Datos del usuario administrador
        const adminData = {
            Email_admin: "dilanflores.21@gmail.com",
            Password_admin: "DAFQ12345"
        };
        // Realiza una solicitud POST para obtener el token de acceso
        const loginResponse = await request(API)
            .post('/api/admin/login')
            .send(adminData);

        // Extrae el token de acceso del cuerpo de la respuesta
        const token = loginResponse.body.token;
        // Id de una actividad existente
        const id = "65a95b0468cec25087c93494";
        // Datos para actualizar la actividad
        const actividadActualizada = {
            Nombre_act: "Rompecabezas2",
            Detalle_act: "Rompecabezas tipo 2",
            Nivel_dificultad: "Alta",
            Recurso_video: "https://www.youtube.com/watch?v=NuevoVideo"
        };

        // Simular solicitud PUT para actualizar la actividad
        const response = await request(API)
            .put(`/api/actividad/actualizar/${id}`)
            .set('Authorization', `Bearer ${token}`)
            .send(actividadActualizada);

        console.log(response.status); // Imprime el código de respuesta
        console.log(response.body); // Mensaje de actualización exitosa
        expect(response.status).toBe(200);
    });
});
```

Figura 1: Código de prueba unitaria modificación actividad - Administrador

```
  console.log
    200

      at Object.log (test/admin.spec.js:165:17)

  console.log
    { msg: 'Actualización exitosa de la actividad' }

      at Object.log (test/admin.spec.js:166:17)

 PASS  test/admin.spec.js (5.447 s)
  Actualizar una actividad PUT /api/actividad/actualizar/{id}
    √ Actualizar una actividad (2738 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        5.657 s
```

Figura 2: Resultado de prueba unitaria modificar actividad - Administrador

```
describe('Eliminar una actividad DELETE /api/actividad/eliminar/{id}', () => {
    test('Eliminar una actividad', async () => {
        // Datos del usuario administrador
        const adminData = {
            Email_admin: "dilanflores.21@gmail.com",
            Password_admin: "DAFQ12345"
        };
        // Realiza una solicitud POST para obtener el token de acceso
        const loginResponse = await request(API)
            .post('/api/admin/login')
            .send(adminData);

        // Extrae el token de acceso del cuerpo de la respuesta
        const token = loginResponse.body.token;
        // Id de una actividad existente
        const id = "65a952ecf02fcac36c114028";
        // Simular solicitud DELETE para eliminar la actividad
        const response = await request(API)
            .delete(`/api/actividad/eliminar/${id}`)
            .set('Authorization', `Bearer ${token}`);

        console.log(response.status); // Imprime el código de respuesta
        console.log(response.body); // Mensaje de eliminación exitosa
        // Verificar que la respuesta sea exitosa (código 200)
        expect(response.status).toBe(200);
    });
});
```

Figura 3: Código de prueba unitaria eliminación de actividad - Administrador

```
  console.log
    200

      at Object.log (test/admin.spec.js:192:17)

  console.log
    { msg: 'Eliminación exitosa de actividad' }

      at Object.log (test/admin.spec.js:193:17)

 PASS  test/admin.spec.js (5.723 s)
  Eliminar una actividad DELETE /api/actividad/eliminar/{id}
    √ Eliminar una actividad (3336 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        5.832 s, estimated 6 s
```

Figura 4: Resultado de prueba unitaria eliminación de actividad – Administrador

Para la administración de usuarios Niño, el usuario Tutor debe completar exitosamente el proceso de inicio de sesión para realizar acciones correspondientes, como visualizar, modificar y eliminar un usuario Niño. A continuación, se detallan las pruebas unitarias pendientes para estas acciones:

En la **Figura 5** presenta el código de la prueba unitaria para la visualización de un usuario Niño por parte del usuario Tutor. Los resultados respectivos se muestran en la **Figura 6.**

En la **Figura 7** se detalla el código de la prueba unitaria para modificar un usuario Niño por parte del usuario Tutor. Los resultados correspondientes se encuentran en la **Figura 8.**

En la **Figura 9** se muestra el código de la prueba unitaria para eliminar un usuario Niño por parte del usuario Tutor. Los resultados de esta prueba se presentan en la **Figura 10.**

```
describe('Visualizar usuario Niño GET /api/nin@s/{id}', () => {
    test('Visualización exitosa de usuario Niño', async () => {
        // Datos del usuario tutor para iniciar sesión y obtener el token
        const tutorData = {
            Email_tutor: "dilan.flores@epn.edu.ec",
            Password_tutor: "DAFQ1234"
        };
        // Realiza una solicitud POST para obtener el token de acceso del
        const loginResponse = await request(API)
            .post('/api/login')
            .send(tutorData);

        // Extrae el token de acceso del cuerpo de la respuesta
        const token = loginResponse.body.token;
        // Id de un niño existente (debes tener un id válido en tu base de
        const idNino = "65ae072d608dc3e4f51e43f3";
        // Realiza una solicitud GET a la ruta para visualizar un Niño uti
        const response = await request(API)
            .get(`/api/nin@s/${idNino}`)
            .set('Authorization', `Bearer ${token}`);

        console.log(response.status); // Imprime el código de respuesta
        console.log(response.body); // Datos del Niño
        // Verificar que la respuesta sea exitosa (código 200)
        expect(response.status).toBe(200);
    });
});
```

Figura 5: Código de prueba unitaria visualizar usuario Niño - Tutor

```
  console.log
    200

      at Object.log (test/tutor.spec.js:226:17)

  console.log
    {
      Nombre_nino: 'AlexY',
      FN_nino: '4/12/2012',
      Usuario_nino: 'AlexY',
      tutor: '65ac9cd61e0a095830534822'
    }

      at Object.log (test/tutor.spec.js:227:17)

 PASS  test/tutor.spec.js (5.622 s)
  Visualizar usuario Niño GET /api/nin@s/{id}
    √ Visualización exitosa de usuario Niño (3215 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        5.791 s, estimated 8 s
```

Figura 6: Resultado de prueba unitaria visualizar usuario Niño - Tutor

```
describe('Actualizar perfil de Niño PUT /api/nin@s/actualizar/{id}', () => {
    test('Actualización exitosa de perfil de Niño', async () => {
        // Datos del usuario tutor para iniciar sesión y obtener el token
        const tutorData = {
            Email_tutor: "dilan.flores@epn.edu.ec",
            Password_tutor: "DAFQ1234"
        };
        // Realiza una solicitud POST para obtener el token de acceso del tut
        const loginResponse = await request(API)
            .post('/api/login')
            .send(tutorData);

        // Extrae el token de acceso del cuerpo de la respuesta
        const token = loginResponse.body.token;
        // Id de un niño existente (debes tener un id válido en tu base de da
        const idNino = "65ae072d608dc3e4f51e43f3";
        // Datos del Niño a actualizar
        const ninoDataActualizar = {
            Nombre_nino: "Alex :)",
            FN_nino: "2013-08-02",
            Usuario_nino: "AlexAlex"
        };
        // Realiza una solicitud PUT a la ruta para actualizar el perfil de u
        const response = await request(API)
            .put(`/api/nin@s/actualizar/${idNino}`)
            .set('Authorization', `Bearer ${token}`)
            .send(ninoDataActualizar);

        console.log(response.status); // Imprime el código de respuesta
        console.log(response.body); // Mensaje de actualización exitosa
        // Verificar que la respuesta sea exitosa (código 200)
        expect(response.status).toBe(200);
    });
});
```

Figura 7: Código de prueba unitaria modificar usuario Niño -Tutor

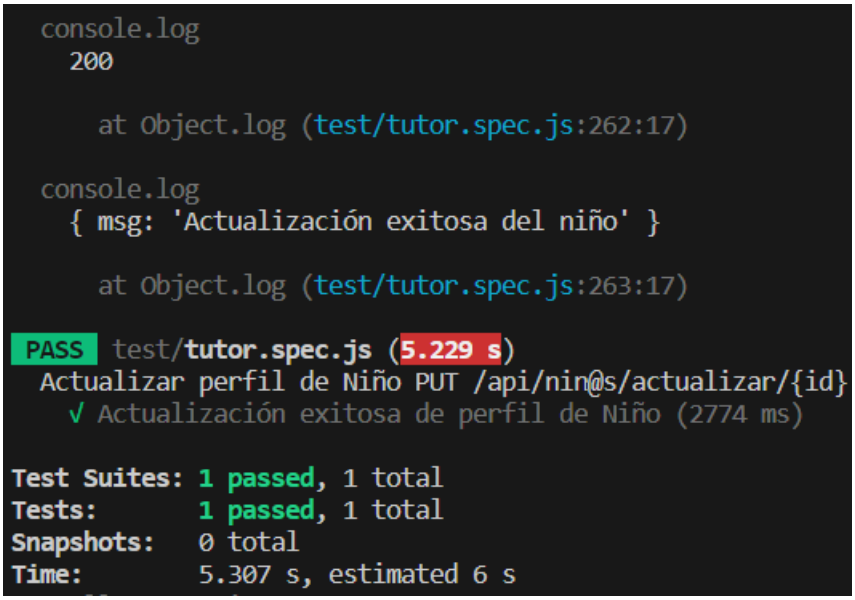


Figura 8: Resultado de prueba unitaria modificar usuario Niño - Tutor

```
describe('Eliminar perfil de Niño DELETE /api/nin@s/eliminar/{id}', () => {
    test('Eliminación exitosa de perfil de Niño', async () => {
        // Datos del usuario tutor para iniciar sesión y obtener el token
        const tutorData = {
            Email_tutor: "dilan.flores@epn.edu.ec",
            Password_tutor: "DAFQ1234"
        };
        // Realiza una solicitud POST para obtener el token de acceso del tutor
        const loginResponse = await request(API)
            .post('/api/login')
            .send(tutorData);

        // Extrae el token de acceso del cuerpo de la respuesta
        const token = loginResponse.body.token;
        // Id de un niño existente (debes tener un id válido en tu base de datos)
        const idNino = "65ae072d608dc3e4f51e43f3";
        // Realiza una solicitud DELETE a la ruta para eliminar el perfil de un N
        const response = await request(API)
            .delete(`/api/nin@s/eliminar/${idNino}`)
            .set('Authorization', `Bearer ${token}`);

        console.log(response.status); // Imprime el código de respuesta
        console.log(response.body); // Mensaje de eliminación exitosa
        // Verificar que la respuesta sea exitosa (código 200)
        expect(response.status).toBe(200);
    });
});
```

Figura 9: Código de prueba unitaria eliminación usuario Niño -Tutor

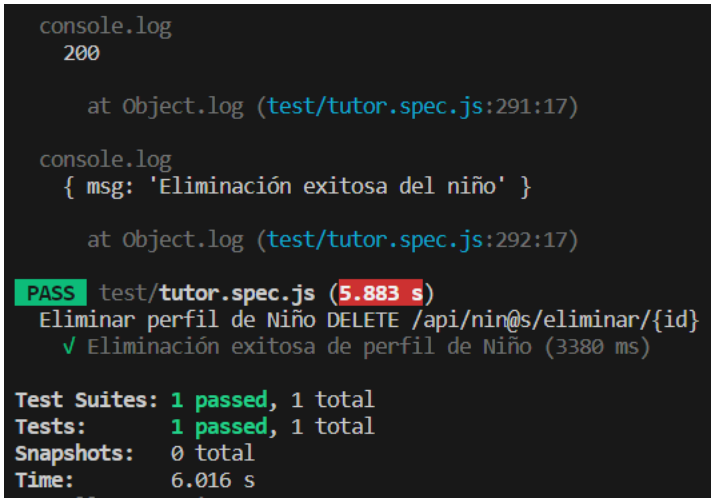


Figura 10: Resultado de prueba unitaria eliminar usuario Niño – Tutor

Para la parte de logros, un usuario Niño completa exitosamente el proceso de inicio de sesión y puede acceder a la parte de visualizar su progreso en actividades inscritas. A continuación, se presentan las pruebas unitarias pendientes para este caso:

En la **Figura 11** se muestra el código de la prueba unitaria para la visualización del progreso en actividades inscritas por parte del usuario Niño. Los resultados de esta prueba se presentan en la **Figura 12.**

```
describe('Obtener progreso de una actividad GET /api/progreso/{actividadID}', () => {
    test('Obtener progreso exitoso', async () => {
        const ninoData = {
            Usuario_nino: "AlexAlex",
            Password_nino: "AAYF1234"
        };
        // Realiza una solicitud POST para obtener el token de acceso del niño
        const loginResponse = await request(API)
            .post('/api/nin@s/login')
            .send(ninoData);

        // Extrae el token de acceso del cuerpo de la respuesta
        const token = loginResponse.body.token;
        const actividadID = "65a95b0468cec25087c93494"; // ID de la actividad de ejemp
        // Simular solicitud GET para obtener el progreso de la actividad
        const response = await request(API)
            .get(`/api/progreso/${actividadID}`)
            .set('Authorization', `Bearer ${token}`);

        console.log(response.status); // Imprime el código de respuesta
        console.log(response.body); // Datos de progreso obtenidos
        // Verificar que la respuesta sea exitosa (código 200)
        expect(response.status).toBe(200);
    });
});
```

Figura 11: Código de prueba unitaria visualizar progreso - Niño

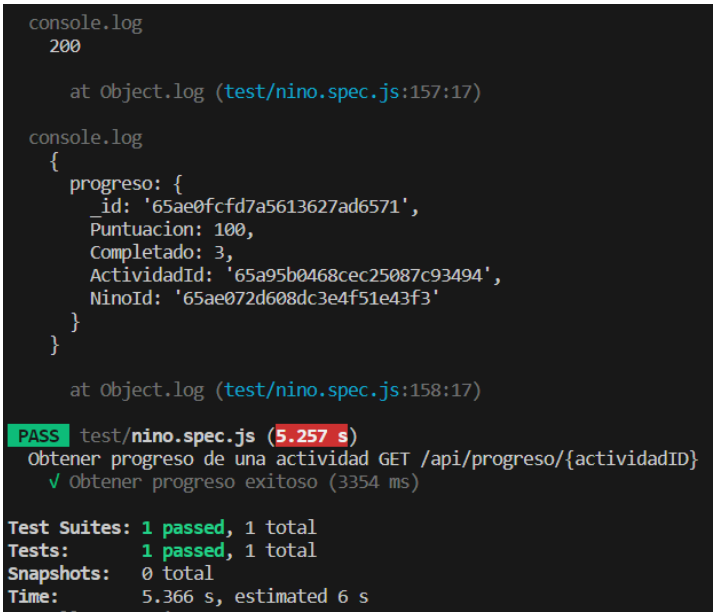


Figura 12: Resultado de prueba unitaria visualizar progreso - Niño

Para la parte de logros, un usuario Niño completa exitosamente el proceso de inicio de sesión y puede acceder a la visualización de sus logros personales en base a condiciones previamente definidas. A continuación, se presentan las pruebas unitarias pendientes para este caso:

En la **Figura 13** se muestra el código de la prueba unitaria para la visualización de logros personales por parte del usuario Niño. Los resultados de esta prueba se presentan en la **Figura 14.**

```
describe('Visualizar logros GET /api/insignias', () => {
    test('Visualización de logros exitosa', async () => {
        const ninoData = {
            Usuario_nino: "AlexAlex",
            Password_nino: "AAYF1234"
        };

        // Realiza una solicitud POST para obtener el token de acceso del niño
        const loginResponse = await request(API)
            .post('/api/nin@s/login')
            .send(ninoData);

        // Extrae el token de acceso del cuerpo de la respuesta
        const token = loginResponse.body.token;
        // Simular solicitud GET para obtener los logros del niño
        const response = await request(API)
            .get('/api/insignias')
            .set('Authorization', `Bearer ${token}`);

        console.log(response.status); // Imprime el código de respuesta
        console.log(response.body); // Lista de logros del niño
        // Verificar que la respuesta sea exitosa (código 200)
        expect(response.status).toBe(200);
    });
});
```

Figura 13: Código de prueba unitaria visualizar logros personales - Niño

Figura 14: Resultado de prueba unitaria visualizar logros personales - Niño

Pruebas de Carga

Las pruebas de carga realizadas para los *endpoints* del proyecto *backend* se enfocaron en evaluar el rendimiento bajo condiciones de alta demanda, especialmente para el usuario con perfil de Tutor. Una vez que este usuario completa el proceso de inicio de sesión, se accede a los *endpoints* para visualizar tanto su propio perfil como el perfil de un usuario niño asociado a su cuenta.

En la **Figura 15** se presenta el código utilizado para estas pruebas, mientras que en la **Figura 16** se detallan los resultados de la ejecución. Se realizaron un total de 900 peticiones, todas las cuales recibieron respuestas exitosas. Este resultado indica que el sistema pudo manejar la carga de manera eficiente, proporcionando una experiencia de usuario consistente incluso bajo condiciones de alta demanda.

```
{
    "post": {
        "url": "/api/login",
        "json": {
            "Email_tutor": "dilan.flores@epn.edu.ec",
            "Password_tutor": "DAFQ1234"
        },
        "capture": {
            "json": "$.token",
            "as": "Token"
        }
    }
},
{
    "get": {
        "url": "/api/tutor/65b1effa9ba67757b54c9b37",
        "headers": {
            "Authorization": "Bearer {{ Token }}"
        }
    }
},
{
    "get": {
        "url": "/api/nin@s/65b203552f4a736e21289aae",
        "headers": {
            "Authorization": "Bearer {{ Token }}"
        }
    }
}
```

Figura 15: Código de prueba de carga - Tutor

```
--------------------------------
Summary report @ 23:04:33(-0500)
--------------------------------

http.codes.200: ..................................................... 900
http.downloaded_bytes: .............................................. 186300
http.request_rate: .................................................. 15/sec
http.requests: ...................................................... 900
http.response_time:
  min: .............................................................. 177
  max: .............................................................. 1431
  mean: ............................................................. 387.9
  median: ........................................................... 361.5
  p95: .............................................................. 620.3
  p99: .............................................................. 727.9
http.responses: ..................................................... 900
vusers.completed: ................................................... 300
vusers.created: ..................................................... 300
vusers.created_by_name.0: ........................................... 300
vusers.failed: ...................................................... 0
vusers.session_length:
  min: .............................................................. 796.8
  max: .............................................................. 2718
  mean: ............................................................. 1170.9
  median: ........................................................... 1130.2
  p95: .............................................................. 1652.8
  p99: .............................................................. 1901.1
```

Figura 16: Resultado de prueba de carga - Tutor

Se realizaron pruebas de carga para el usuario con perfil de Niño en el proyecto *backend.* Una vez que este usuario completa el proceso de inicio de sesión, tiene acceso a los *endpoints* para visualizar su perfil, progreso y logros respectivos.

En la **Figura 17** se muestra el código utilizado para estas pruebas, mientras que en la **Figura 18** se detallan los resultados de la ejecución. Se realizaron un total de 1200 peticiones, todas las cuales recibieron respuestas exitosas. Este resultado indica que el sistema pudo manejar la carga generada por estas operaciones de manera efectiva, asegurando un rendimiento consistente para el usuario Niño incluso bajo condiciones de alta demanda.

```
{
    "post": {
        "url": "/api/nin@s/login",
        "json": {
            "Usuario_nino": "alexy",
            "Password_nino": "AAYF1727"
        },
        "capture": {
            "json": "$.token",
            "as": "TokenNino"
        }
    }
},
{
    "get": {
        "url": "/api/nin@s/65b203552f4a736e21289aae",
        "headers": {
            "Authorization": "Bearer {{ TokenNino }}"
        }
    }
},
{
    "get": {
        "url": "/api/progreso/65a95b0468cec25087c93494",
        "headers": {
            "Authorization": "Bearer {{ TokenNino }}"
        }
    }
},
{
    "get": {
        "url": "/api/insignias",
        "headers": {
            "Authorization": "Bearer {{ TokenNino }}"
        }
    }
}
```

Figura 17: Código de prueba de carga - Niño

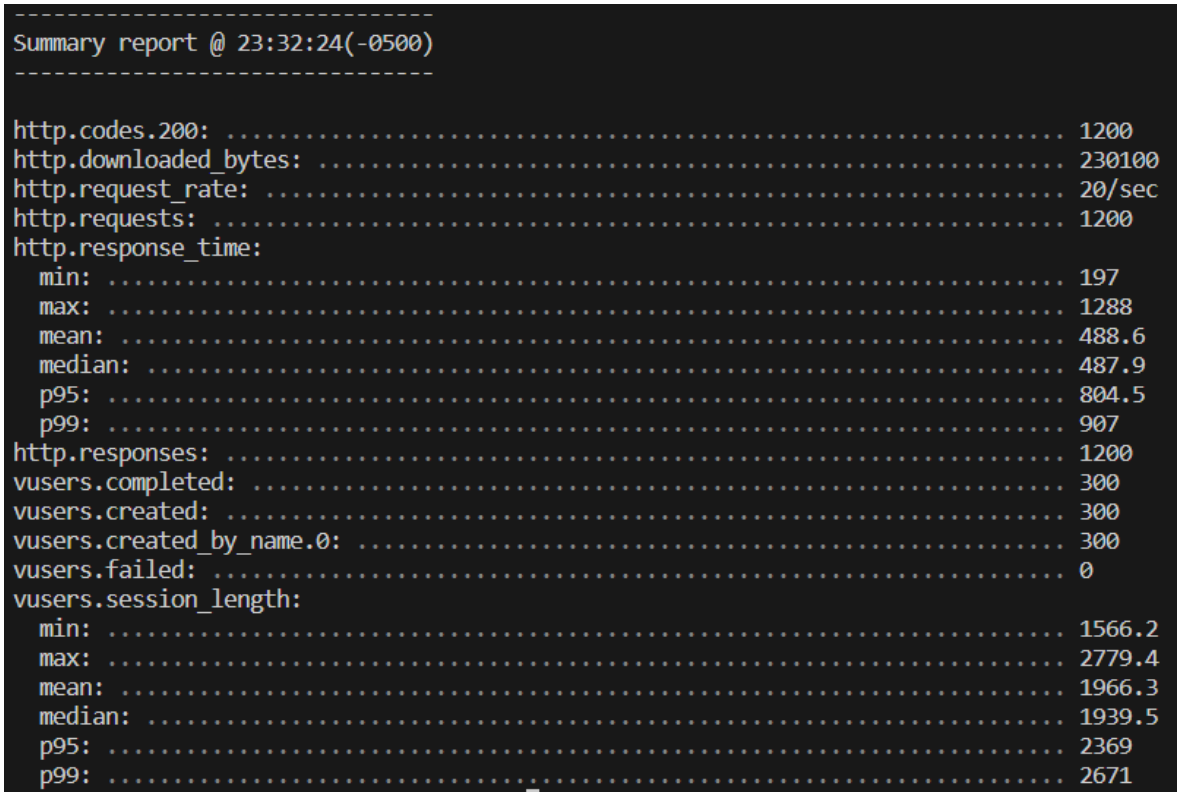


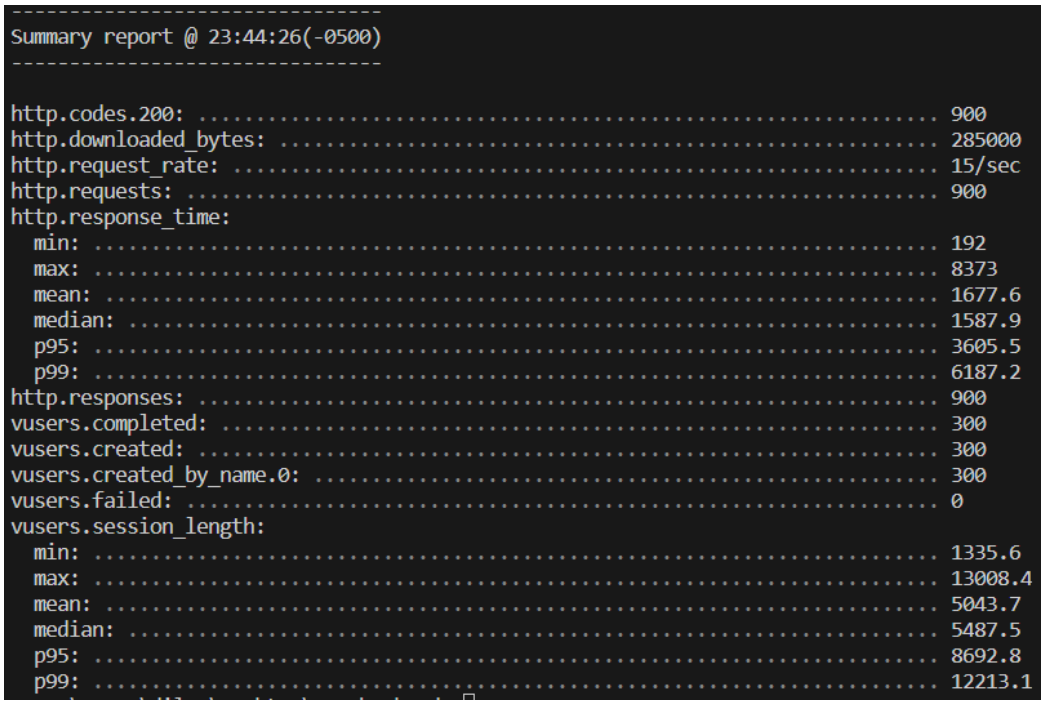
Figura 18: Resultados de la prueba de carga – Niño

Se realizaron pruebas de carga para el usuario con perfil de Administrador en el proyecto *backend.* Una vez que este usuario completa el proceso de inicio de sesión, tiene acceso a los *endpoints* para visualizar el detalle de una actividad y todas las actividades registradas.

En la **Figura 19** se muestra el código utilizado para estas pruebas, mientras que en la **Figura 20** se detallan los resultados de la ejecución. Se realizaron un total de 900 peticiones, todas las cuales recibieron respuestas exitosas. Este resultado indica que el sistema pudo manejar la carga generada por estas operaciones de manera efectiva, asegurando un rendimiento consistente para el usuario Administrador incluso bajo condiciones de alta demanda.

```
{
    "post": {
        "url": "/api/admin/login",
        "json": {
            "Email_admin": "dilanflores.21@gmail.com",
            "Password_admin": "DAFQ12345"
        },
        "capture": {
            "json": "$.token",
            "as": "TokenAdmin"
        }
    }
},
{
    "get": {
        "url": "/api/actividades",
        "headers": {
            "Authorization": "Bearer {{ TokenAdmin }}"
        }
    }
},
{
    "get": {
        "url": "/api/actividad/65a95b0468cec25087c93494",
        "headers": {
            "Authorization": "Bearer {{ TokenAdmin }}"
        }
    }
}
```

Figura 19: Código de prueba de carga - Administrador



```
-------------------------------
Summary report @ 23:44:26(-0500)
-------------------------------

http.codes.200: ..................................................... 900
http.downloaded_bytes: .............................................. 285000
http.request_rate: .................................................. 15/sec
http.requests: ...................................................... 900
http.response_time:
  min: .............................................................. 192
  max: .............................................................. 8373
  mean: ............................................................. 1677.6
  median: ........................................................... 1587.9
  p95: .............................................................. 3605.5
  p99: .............................................................. 6187.2
http.responses: ..................................................... 900
vusers.completed: ................................................... 300
vusers.created: ..................................................... 300
vusers.created_by_name.0: ........................................... 300
vusers.failed: ...................................................... 0
vusers.session_length:
  min: .............................................................. 1335.6
  max: .............................................................. 13008.4
  mean: ............................................................. 5043.7
  median: ........................................................... 5487.5
  p95: .............................................................. 8692.8
  p99: .............................................................. 12213.1
```

Figura 20: Resultado de prueba de carga – Adminsitrador

ANEXO III

En esta sección se presenta la URL de manual de usuario del presente proyecto *backend:*

https://www.youtube.com/watch?v=DQ54SGomKfs

ANEXO IV

En esta sección se presenta el enlace al repositorio GitHub para acceder al contenido del código completo, además de la documentación del presente proyecto *backend* en un entorno de producción.

Entorno de producción:

https://epn-backend.onrender.com/doc/

Credenciales Administrador

- Correo electrónico: user.adm.epn@gmail.com
- Contraseña: adm123EPN

Credenciales Tutor

- Correo electrónico: user.tutor.epn@gmail.com
- Contraseña: tutor123EPN

Credenciales Niño

- Nombre de usuario: NinoEPN
- Contraseña: nino1234EPN

Repositorio GitHub:

https://github.com/dilan-flores/EPN_backend