Desarrollo de *Backend* para aplicación web para mejorar la rapidez y comprensión lectora de estudiantes de nivel medio en Guatemala

Michael Steven Chan González

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Desarrollo de *Backend* para aplicación web para mejorar la rapidez y comprensión lectora de estudiantes de nivel medio en Guatemala

Trabajo de graduación presentado por Michael Steven Chan González para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala,

2024

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Desarrollo de *Backend* para aplicación web para mejorar la rapidez y comprensión lectora de estudiantes de nivel medio en Guatemala

Trabajo de graduación presentado por Michael Steven Chan González para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala,

2024

Vo.Bo.:

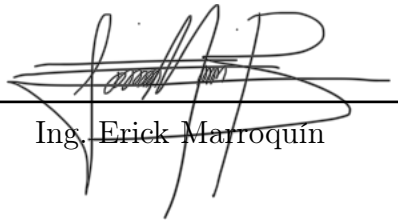
(f) _____

Ing. Diana Ortiz

Tribunal Examinador:

(f) _____

Ing. Diana Ortiz



(f) _____

Ing. Erick Marroquín

(f) _____

Ing. Douglas Barrios

Fecha de aprobación: Guatemala, 18 de abril de 2024

Índice

Lista de figuras

Lista de cuadros

Resumen

El presente trabajo se realiza con la finalidad de desarrollar una interfaz de programación que pueda ser consumida por una aplicación web orientada a mejorar el rendimiento de velocidad de lectura y comprensión lectora en estudiantes de nivel primario en Guatemala.

El ambiente utilizado para el desarrollo de la interfaz de programación ha sido Laravel, el cual se maneja sobre el lenguaje de programación PHP. La interfaz ha sido alojada en una máquina virtual de Amazon Web Services, implementando el protocolo SSL para la seguridad de las solicitudes. El código ha sido almacenado en el manejador de versiones GitHub, ahí mismo se ha implementado la herramienta dependabot que se encarga de analizar las bibliotecas utilizadas e indicar si alguna posee alguna vulnerabilidad.

El desarrollo del proyecto se realizó bajo el marco de trabajo de SCRUM. SCRUM es una metodología ágil que ha permitido una gestión eficiente del trabajo y una entrega iterativa del producto. La implementación de la interfaz y la aplicación web se logró gracias a la estructura proporcionada por SCRUM, que se enfoca en equipos auto-organizados, entregas rápidas y una comunicación fluida entre los miembros. Además se realizaron pruebas de carga y estrés, determinando la capacidad del sistema y los costos necesarios para una mejor infraestructura, de esta manera asegurar la eficiencia de las solicitudes.

Abstract

The current work is being carried out to develop an application programming interface (API) intended for consumption by a web application aimed at improving reading speed and comprehension among primary-level students in Guatemala.

The environment utilized for the API development has been Laravel, which operates on the PHP programming language. The interface has been hosted on an Amazon Web Services (AWS) virtual machine, implementing the SSL protocol for request security. The code has been stored in the version control system GitHub, where the Dependabot tool has been implemented to analyze the libraries used and indicate if any vulnerabilities are present.

The project was developed within the SCRUM framework. SCRUM is an agile methodology that allowed for efficient work management and iterative product delivery. The implementation of the interface and web application was achieved through the structure provided by SCRUM, emphasizing self-organized teams, swift deliveries, and seamless communication among members. Additionally, load and stress tests were conducted to determine the system's capacity and the costs required for an improved infrastructure, ensuring the efficiency of requests.

CAPÍTULO 1

Introducción

Para un buen desarrollo de un estudiante es importante que pueda desarrollar una alta velocidad de lectura y comprensión lectora. La velocidad de lectura representa la cantidad de palabras que una persona es capaz de leer en un minuto durante una lectura natural, sin tomar en cuenta otros aspectos como la comprensión del mismo. Por otro lado, la comprensión lectora presenta la capacidad del lector para extraer y construir significados a partir de la lectura del texto (Almutairi, 2018).

Una alta velocidad de lectura junto a un buen nivel de comprensión lectora, conlleva a que el estudiante logre un mejor desenvolvimiento en todo el ámbito académico. Ya que se logra absorber una mayor cantidad de información en un tiempo reducido, resultando en una mejora directa en materias como matemáticas y literatura (Andina, 2006)

En el caso de Guatemala, estas dos habilidades presentan una evidente deficiencia, ya que en los resultados de prueba de lectura realizados por el ministerio de educación en el año 2021 únicamente un 32 % de los estudiantes evaluados presentan un buen desempeño (Perez y Dominguez, 2022). Por otra parte, datos indican que un porcentaje muy reducido de la población total de Guatemala (16.5 millones de habitantes), probablemente menor al 3 por ciento lee por placer, a diferencia de países como Argentina o Chile que se mantiene en 70 por ciento. Y ante este dato, se entiende que menos del 3.0 por ciento entiende a su totalidad lo que está leyendo.

Sabiendo estos datos, se entiende que la comprensión lectora es un tema que en Guatemala se ha minimizado a pesar de ser una habilidad importante para el desarrollo de estudiantes a cualquier nivel, pero especialmente en el nivel medio.

Ante los problemas de subdesarrollo de estas habilidades cognitivas en Guatemala presentados, se propone el presente proyecto, el cual tiene como objetivo lograr que los niveles de velocidad de lectura y comprensión lectora puedan llegar a ser duplicados mediante pruebas y retos interactivos, dentro de una aplicación web diseñada, desarrollada y montada durante la segunda mitad del año 2022.

CAPÍTULO 2

Antecedentes

2.1. Progrentis

Progrentis es una plataforma de "destrezas digitales" que busca el desarrollo del pensamiento creativo, que es la capacidad de generar a partir del correcto uso de la información, nuevas ideas que llenan una necesidad, elevando los niveles de comprensión lectora y enseñando a buscar y filtrar toda la información de una mejor manera. Está dirigida a niños y adolescentes en edad escolar y ofrece una variedad de herramientas y recursos pedagógicos. Ofrece una amplia variedad de recursos y actividades diseñadas para mejorar el rendimiento académico y potenciar el desarrollo cognitivo..

Progrentis basa su funcionamiento en una metodología basada en el juego para involucrar a los estudiantes y hacer el aprendizaje más interactivo y entretenido, desarrollando habilidades cognitivas clave, como la atención, la memoria, la percepción visual y auditiva, la lógica, entre otras. De igual manera, utiliza técnicas de personalización para adaptar el contenido y las actividades de aprendizaje según el nivel y las áreas de interés de cada estudiante. La plataforma realiza un seguimiento del progreso de cada estudiante a lo largo del tiempo, lo que permite a los padres y educadores ver cómo están mejorando en diferentes áreas cognitivas. (Progrentis, 2022).

El problema de la implementación de esta herramienta dentro de Guatemala es su precio elevado, los precios de suscripciones dentro de Guatemala suelen estar por encima de los Q300 y las cuentas deben estar ligadas a un centro educativo, esto conlleva a que no sea accesible a toda la población guatemalteca, especialmente establecimientos del sector público.

2.2. iStart

iSTART enseña estrategias de autoexplicación a través de una serie de videos y ejercicios prácticos basados en juegos. Las estrategias de autoexplicación permiten a los estudiantes

desarrollar una comprensión más profunda de los conceptos complejos explicados en el texto. Además, los estudiantes que se autoexplican los textos tienen más éxito en la resolución de problemas, son más propensos a generar inferencias y construyen modelos mentales más coherentes.

iStart es la tecnología educativa enfocada en comprensión lectora que tiene la mayor cantidad de estudios y publicaciones, que dan cuenta de su eficacia, es istart (Interactive Strategies Training for Active Reading and Thinking) creada por Danielle McNamara y su equipo (McNamara, O'Reilly, Rowe, Boonthum y Levinstein, 2007; Snow, Jacovina, Jackson, y McNamara, 2016). Como objetivo de istart tienen mejorar la comprensión lectora de estudiantes de secundaria a través de la enseñanza y entrenamiento de cinco estrategias: monitoreo, puente, predicción, elaboración y parafraseo. Los estudiantes deben leer partes de textos científicos y luego elaborar auto explicaciones acerca de esas piezas de información, utilizando alguna de las estrategias aprendidas.

El sistema está diseñado para apoyar el aprendizaje efectivo, aumentar la motivación para aprender y mantener el compromiso durante una interacción a largo plazo con el sistema. Se han realizado múltiples estudios para evaluar los beneficios de usar iSTART. Los usuarios del sistema han demostrado aumentos significativos tanto en conocimiento como en rendimiento durante y después de su uso, la suscripción a esta plataforma se encuentra en USD5.

El problema con esta herramienta es que originalmente ha sido desarrollada en inglés y la traducción al español aún no se encuentra optimizada, de igual manera posee una interfaz muy antigua y poco amigable al usuario, como se puede evidenciar en la siguiente figura.

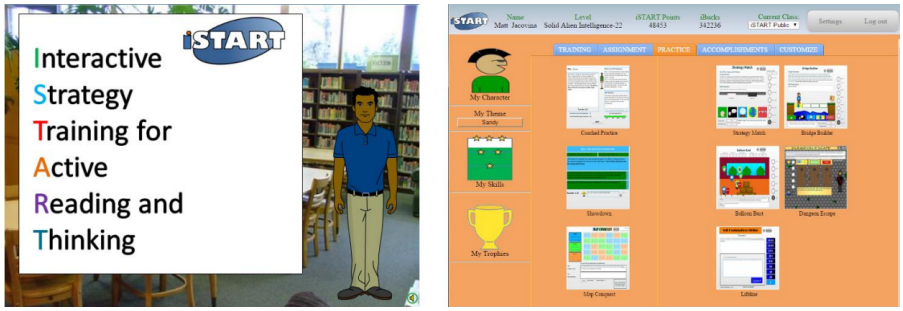


Figura 1: Imágenes de iStart: video lección (izquierda) y menú de acceso a los juegos (a la derecha).

2.3. Comprende

También existe la herramienta 'Comprende' (Soto, Gutiérrez de Blume, Rodríguez, Asún, Figueroa, y Serrano, 2019) es una tecnología educativa que está diseñada para enseñar y entrenar estrategias de comprensión lectora en alumnos de 5° a 7° año de primaria. Para ello cuenta con dos fases: Lecciones y Entrenamiento. En la fase de Lecciones, un agente animado enseña a aplicar las estrategias de vocabulario contextual, inferencias puente, integración de ideas y comprensión global.

La herramienta Comprende utiliza dos principales técnicas

Estrategia de Puente (Bridging Strategy): Esta estrategia se refiere a dar un enfoque o método utilizado para ayudar a los lectores a conectar la información presente en el texto con su conocimiento previo o con otros conceptos relevantes. Las estrategias de puente suelen ser útiles para mejorar la comprensión al facilitar la integración de nueva información con el conocimiento existente.

Juicios de Sentimiento de Saber (Feeling of Knowing Judgments): Esto se refiere a realizar evaluaciones subjetivas que los lectores hacen sobre su propio conocimiento o comprensión de un tema particular mientras están leyendo. Por ejemplo, un lector podría evaluar si siente que sabe lo suficiente sobre un tema o si necesita más información para comprenderlo completamente, la plataforma no incluye información acerca de los precios de la plataforma.

El problema con esta herramienta es no presentar un formato basado en juegos, por lo cual no llega a ser entretenido para los niños y en poco tiempo perdemos su atención.

2.4. Cuadro comparativo

Criterios	Progrentis	iStart	Comprende	Objetivo
Interfaz de usuario amigable	✓		✓	✓
Precio accesible para estudiantes de Guatemala		✓	✓	✓
Aprendizaje basado en juegos, con actividades interactivas para mantener la atención de los estudiantes	✓	✓		✓

Cuadro 1: Cuadro comparativo de herramientas existentes y el objetivo que se ha desarrollado.

CAPÍTULO 3

Justificación

Para un desarrollo óptimo en la educación de un estudiante, la capacidad de leer a un ritmo rápido mientras se tiene comprensión de textos extensos, llegan a ser indispensables para un buen nivel de desarrollo en el ámbito académico. Por un lado, ser capaz de leer a alta velocidad puede tener una influencia positiva en qué tan bien el lector comprende un texto así como qué tanto recuerda del mismo. Por otra parte, un alto nivel de comprensión lectora resulta en un mejor desempeño académico en materias como matemáticas y lenguaje (Andina, 2006).

Debido al rápido desarrollo de las tecnologías digitales en conjunto con el aumento en la cantidad de información disponible, una persona enfrenta el problema de la sobrecarga de información y, en consecuencia, la incapacidad de procesar cualitativamente dicha información. La situación actual desafía las habilidades de lectura tradicionales y crea la necesidad de formas más rápidas, avanzadas y efectivas de procesar la información textual. (Merina, 2021)

El desarrollo de una herramienta que mejore la comprensión lectora y la velocidad de lectura en los niños es fundamental por varias razones:

Habilidad fundamental: La lectura es una habilidad fundamental para el aprendizaje en todas las áreas. Mejorar la comprensión lectora y la velocidad de lectura en los niños les permite acceder a información, aprender de manera efectiva y desenvolverse en el mundo académico.

Éxito académico: La capacidad de comprender lo que se lee y de leer rápidamente impacta directamente en el rendimiento académico. Los niños con buenas habilidades de lectura tienden a tener un mejor desempeño en todas las materias.

Autonomía y empoderamiento: Una mejor comprensión lectora les otorga a los niños la autonomía para aprender de manera independiente y explorar el mundo a través de la lectura. Además, los niños que leen bien pueden acceder a información valiosa por sí mismos, lo que les empodera en su aprendizaje.

Desarrollo de habilidades cognitivas: La lectura mejora la capacidad cognitiva de los niños, fortaleciendo áreas como la atención, la memoria, la concentración y el pensamiento crítico.

Fomenta la imaginación y la creatividad: La lectura les brinda a los niños la oportunidad de sumergirse en mundos imaginarios y desarrollar su creatividad.

Mejora la comunicación: Una buena comprensión lectora se traduce en una mejor habilidad para expresarse tanto oralmente como por escrito.

Preparación para el futuro: En un mundo cada vez más digitalizado, donde la información se obtiene principalmente a través de la lectura en dispositivos electrónicos, es crucial que los niños desarrollen habilidades sólidas de comprensión lectora y velocidad para enfrentar los desafíos futuros.

Por todas estas razones, el desarrollo de herramientas efectivas que mejoren la comprensión lectora y la velocidad de lectura en los niños no solo beneficia su desempeño actual, sino que también establece las bases para su éxito académico y personal a lo largo de sus vidas (Merina, 2021).

Esta aplicación web presentará lecciones interactivas que permitirán a los estudiantes de nivel medio incrementar su rapidez de lectura a la vez que mejoran su capacidad para analizar, comprender y retener la información que consumen de manera textual.

Tomando en cuenta los antecedentes mencionados en el capítulo anterior, ninguna de las herramientas ya existentes cumple con todas las expectativas para poder ser implementada con éxito en la región de Guatemala, con esto en mente se busca desarrollar una aplicación web accesible por todos los niños del país y que posea una interfaz amigable con una interacción por medio de juegos para mantener la atención del niño todo el tiempo.

Para el funcionamiento de esta aplicación web se necesita una interfaz de programación (API) la cual representará el "*backend*" de la aplicación, encargada de recibir solicitudes y controlar el manejo de la base de datos que se estará manejando.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Desarrollar API Restful que pueda ser consumido por herramienta web que logre incrementar la rapidez de lectura y mejorar la comprensión lectora de los estudiantes de nivel medio de Guatemala mediante lecciones interactivas.

4.2. Objetivos específicos

- Examinar los tiempos de respuesta de la aplicación con el fin de lograr respuestas eficientes mediante una estructura de base de datos ótima.
- Comparar resultados de sujetos de prueba realizando pruebas al inicio y al final para identificar la mejora del individuo utilizando la aplicación.
- Indentificar la capacidad del servidor utilizando pruebas de estrés a la aplicación con el fin de establecer límites para un funcionamiento óptimo.
- Calcular los costos requeridos para el funcionamiento de la aplicación dentro del servidor basándose en los precios actuales del mercado.

CAPÍTULO 5

Marco teórico

5.1. Velocidad de lectura

La velocidad lectora se refiere a la cantidad de palabras que una persona consigue leer por minuto durante una lectura natural de un texto sin comprometer su nivel de comprensión del mismo (Rayner, et al., 2016). Cabe mencionar, que la velocidad de lectura se mide en la lectura de un texto completo, no del resultado de hacer skimming o scanning de un texto, los cuales consisten en echar un vistazo a los párrafos claves de un texto y escanear un texto para encontrar un dato o palabra específica, respectivamente.

La velocidad de lectura se mide en palabras leídas por minuto, el promedio para estudiantes de nivel medio se encuentra entre doscientas y trescientas palabras por minuto (Hasbrouck y Tindal, 2017).

5.2. Comprensión lectora

La comprensión lectora es una de las habilidades cognitivas más importantes que puede desarrollar el ser humano para concretar aprendizajes en cualquier ámbito. Monroy y Gómez, la definen como el entendimiento de textos leídos por una persona permitiéndole la reflexión, el análisis, y la interpretación de lo leído con el conocimiento previo (Monroy y Gómez, 2009).

Esta habilidad está desarrollada de manera distinta en cada persona. Lo que indica que existen múltiples niveles de comprensión lectora, estos niveles se pueden identificar haciendo uso de ciertas métricas. A pesar de que no existen métricas estandarizadas para la comprensión lectora de un individuo, uno de los exámenes más aceptados globalmente es el exámen PISA (por sus siglas en inglés «Program for International Student Assessment ») desarrollado por la OCDE (Por sus siglas en ingés « Organisation for Economic Co-operation and Development ») que mide a través de sus estándares, la capacidad lectora no sólo en el ámbito académico, sino en situaciones varias (OCDE/INCE, 2000). El exámen mide la

capacidad lectora en tres escalas:

- Obtención de información: muestra la capacidad de los estudiantes para localizar información en un texto.
- Interpretación de textos: muestra la capacidad para construir significados y hacer inferencias a partir de la información escrita.
- Reflexión y evaluación: muestra la capacidad del alumno para relacionar el texto con sus conocimientos, sus ideas y sus experiencias.

Las escalas mencionadas anteriormente se pueden utilizar como base para desarrollar pruebas que permitan a las personas mejorar su capacidad lectora.

5.3. Problemas relacionados con una velocidad de lectura deficiente

Tener una capacidad lectora deficiente, según García, Jiménez, González y Jiménez-Suárez (2015) se define como la dificultad de poder decodificar el lenguaje escrito, y no tener la posibilidad de interpretar correctamente el mensaje que está transmitiendo el autor en sus escritos. Ante esta problemática, se han desencadenado diferentes problemas que actúan como causas y consecuencias de esta deficiencia lectora.

Estos problemas se han catalogado como trastornos de la lectura, y estos ocurren cuando una persona tiene problemas para leer palabras. La dislexia es un tipo de trastorno de la lectura, que afecta específicamente la velocidad de lectura de una persona. Las personas disléxicas tienen una inteligencia convencional pero leen a niveles más bajos de lo esperado. (Real Academia Española, 2022)

La mayoría de los problemas de la capacidad lectora, se presentan en la niñez. Aunque en algunos casos las personas pierden la capacidad de leer después de un derrame cerebral o alguna lesión neurológica. De acuerdo a estas problemáticas de salud, el tipo de trastorno de la lectura se llama alexia. (American Speech-Language-Hearing Association, 2017)

La Real Academia Española define el término "analfabetismo" como la falta de instrucción elemental en un país, referida especialmente al número de sus ciudadanos que no saben leer, ni escribir. Por lo tanto, un analfabeto es aquel que no posee conocimiento acerca de las letras. (Uría, 2005). A pesar de los logros y progresos en el ámbito educativo, se debe reconocer que el analfabetismo aún no ha podido ser erradicado. La permanencia de este mismo indica que no se trata de un problema sencillo ni de simple solución; por el contrario, es un problema complejo, profundo y relacionado con las condiciones producto de la pobreza y desigualdad, problemas que también son persistentes en Guatemala.

El analfabetismo suele persistir en las áreas marginadas y en aquellos grupos sociales los cuales no tienen acceso a gran cantidad de bienes y servicios que formalmente todos los guatemaltecos deberían poseer. Su persistencia está relacionada con la reproducción de cierta estructura de la sociedad guatemalteca y de las disimilitudes sociales. Los ámbitos

culturales, sociales e incluso lingüísticos condicionan los procesos educativos y, por supuesto, restringen a su vez los alcances de la alfabetización.

En cualquier investigación que pretenda examinar a fondo la tenencia o no de aptitudes básicas sobre los elementos de la lectoescritura, se debe tomar en cuenta la dicotomía histórica entre el analfabetismo y el alfabetismo, debido a que estos términos contrapuestos desempeñarán una comprensión eficiente sobre la evolución y definición de este problema social. Por esta razón, a continuación se presentarán conceptos de alfabetismo. El diccionario de la Real Academia Española sugiere que el alfabetismo es el conocimiento básico de la lectura y escritura y a su vez define "alfabetizar" como la acción de enseñar a alguien a leer y escribir.

La Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura (UNESCO) afirma que la alfabetización es un derecho fundamental y que a su vez, este derecho se torna en la base del conocimiento y aprendizaje que cualquier persona obtiene a lo largo de toda su vida y que es esencial tanto en el desarrollo social como en el propio ser humano. Sugiere al mismo tiempo que la alfabetización es un recurso imprescindible en el desarrollo de la sociedad mundial.

5.4. Problemas relacionados con una comprensión lectora deficiente

La comprensión lectora de una persona se ve afectada por diversos factores, entre ellos nivel de estudios de los padres, zona socioeconómica, zona geográfica, rural o urbana, entre otras (Perez, 2014).

Personas con un nivel de comprensión lectora bajo suelen presentar las mismas dificultades, como lo son problemas con la falta de memoria, memoria defectuosa, pobre vocabulario, falta de decodificación de las palabras o frases, interpretación incorrecta respecto a las tareas dadas, inseguridad o autoestima baja, falta de interés o motivación para realizar actividades. Todo esto influye a que la persona no desarrolle un juicio crítico, al no decodificar bien la información de las noticias crea desinformación y una divulgación de noticias incompletas o no ciertas.

La baja comprensión lectora tiene como consecuencias la poca comprensión y aprendizaje de los temas y lecciones impartidas dentro de las clases, tomando en cuenta que la lectura va en conjunto de la escritura, la deficiencia de ambos conlleva a un bajo rendimiento estudiantil y alta dificultad para adaptarse a un adecuado aprendizaje en cualquier nivel educativo (Suarez, et al., 2010).

5.5. Situación actual de Guatemala

Las disparidades socioeconómicas pueden impactar la calidad de la educación, ya que algunos sectores de la población tienen acceso a mejores recursos educativos y oportunidades que otros, muchos guatemaltecos, especialmente en áreas rurales y comunidades indígenas,

tienen un acceso limitado a la educación formal. La falta de acceso a escuelas de calidad y recursos educativos dificulta el desarrollo de habilidades de lectura.

La falta de libros, bibliotecas y materiales educativos adecuados puede afectar negativamente la práctica de la lectura y la comprensión lectora entre la población guatemalteca.

De acuerdo con el Informe Nacional de Graduandos del 2019, realizado por la Dirección General de Evaluación e Investigación Educativa (Digeduca), del Mineduc, con base en el examen de lectura y matemática de 157 mil 318 estudiantes, a nivel nacional, como requisito para extender el título del ciclo diversificado, 62.97 % no logró alcanzar los niveles de desempeño de lectura y el 37.03 % fue calificado como satisfactorio y excelente. "Este fenómeno del bajo logro obtenido se asocia con el escaso hábito lector que tienen los estudiantes", señala. En el 2010, el porcentaje de logro fue de 22.39 % (Mineduc, 2019).

En este informe se señala, además, que 20.47 % de los estudiantes no ha leído ningún libro por placer; 31.5 %, solo uno, y 23.2 %, dos. Apenas el 4.01 % ha leído más de seis libros. De este último porcentaje, 63.9 % alcanzó el logro en lectura.

5.6. API (Application Programming Interface)

Se le denomina API a un software intermediario entre dos aplicaciones que permite la comunicación entre ellas. Una API establece el contenido que es requerido del consumidor, denominado como una llamada, y el contenido que debe devolver el servidor, denominado como una respuesta (Red Hat, 2020).

En el momento en el que un API utiliza el protocolo HTTP para realizar las solicitudes se considera una API RESTful. Luego pueden ser utilizadas las funciones de GET, PUT, POST y DELETE para interactuar que se refiere a leer, actualizar, crear y borrar recursos (Gillis, 2020).

5.6.1. Alojamiento de API

Para que una API pueda ser accedida por cualquier persona en Internet, debe ser alojada dentro de un servidor. La entidad que se encarga de alojar el API dentro del servidor también es el responsable de mantener una infraestructura estable y asegurar que el API pueda ser accedido por los usuarios deseados. Los usuarios pueden acceder al API realizando solicitudes hacia una URL específica. Las API son alojadas usualmente para proveer un acceso a datos o funcionalidades a otros desarrolladores o aplicaciones (Nzubechukwu, 2023).

5.7. Arquitectura REST

La arquitectura REST ha sido desarrollada por Roy Fielding en el año 2000 para el desarrollo web, esta arquitectura se encuentra basada en el protocolo de comunicación HTTP. Consta de un listado de reglas que rigen el diseño de una API, un API que se ve definido por

una arquitectura REST es llamada una API RESTful. Un sistema basado en arquitectura RESTful utiliza los métodos de HTTP para definir las acciones que se aplicarán a los recursos del sistema. Los métodos HTTP definen las acciones:

5.7.1. Método GET

El método GET se traduce en recuperar cualquier tipo de información en forma de identidad identificada por el URL al que se le solicita. Si el URL al que se solicita hace referencia a un proceso de producción de datos, son los datos procesados los que se tendrán como respuesta en la entidad que se devuelve. La respuesta a una solicitud GET puede ser almacenada en caché únicamente si cumple con los requisitos para almacenar en caché HTTP (IBM, 2021).

5.7.2. Método POST

El método POST es utilizado para realizar una solicitud al servidor permitiendo que se acepte una agrupación de datos por medio de un cuerpo (body) y nada en el URL a comparación del método GET, suele ser utilizado para registrar información, o acciones como solicitar información bajo ciertos parámetros (IBM, 2021).

5.7.3. Método PUT

Es similar al método de petición POST, con la diferencia de que PUT puede ser ejecutado varias veces teniendo el mismo resultado, caso contrario a POST que cada vez que se ejecuta realiza la agregación de un nuevo objeto, ya que el método PUT es utilizado para la actualización de información ya existente (IBM, 2021).

5.7.4. Método DELETE

Este método permite la eliminación de un recurso en específico, de igual manera como el método PUT, este puede ser utilizado varias veces y el resultado no cambiará, ya que únicamente se elimina el recurso indicado, y no eliminará uno diferente cada vez que se utilice (IBM, 2021).

5.8. Instancia en la nube

Una instancia es una máquina virtual que se encuentra activa en la nube. Dentro de esta aplicación se pueden implementar servidores que corren una aplicación. Una API puede ser montada dentro de una instancia en la nube, permitiendo un fácil acceso por medio de un dominio (Shivang, s.f).

Una instancia en la nube será utilizada para poder alojar el API y nuestra base de datos para que pueda ser accedida por cualquier servicio.

5.9. Respuestas posibles del servidor

Al realizar solicitudes a un servidor, este se encuentra obligado a responder a cada una de las peticiones realizadas por un cliente. Las respuestas pueden ser desde un texto plano, un maquetado HTML, un objeto JSON, hasta archivos con un poco más de complejidad, como imágenes, videos, audios, pdfs, etc.

También se debe tomar en cuenta que al utilizar, ya sea, el protocolo HTTP o HTTPS, existen diferentes tipos de estatus para notificar el estado de una solicitud. estos estatus son representados por medio de un código numérico, siendo éstos códigos llamados status code.

Estos código abarcan un rango de números enteros, dentro del rango de 100 a 599, pueden ser agrupados dentro de 5 categorías:

5.9.1. Respuestas informativas (100 - 199)

- 100 Continue: El servidor ha recibido los encabezados de solicitud y el cliente debería continuar enviando el cuerpo de la solicitud.
- 101 Switching Protocols: El servidor acepta el cambio propuesto por el cliente en los protocolos y está listo para iniciar la comunicación.

5.9.2. Respuestas satisfactorias (200 - 299)

- 200 OK: Indica que la solicitud se ha procesado correctamente y que el servidor está enviando el contenido solicitado. Es el código de respuesta más común cuando todo está funcionando correctamente.
- 201 Created: Se utiliza cuando una solicitud ha tenido éxito y ha llevado a la creación de un nuevo recurso. Es comúnmente utilizado en respuestas a peticiones POST que crean un nuevo objeto en el servidor.
- 202 Accepted: La solicitud ha sido aceptada para procesamiento, pero el procesamiento no ha sido completado.
- 204 No Content: Indica que la solicitud se ha completado con éxito, pero no hay contenido para enviar en la respuesta. Esto es común en solicitudes donde no es necesario enviar datos de vuelta al cliente.

5.9.3. Redirecciones (300 - 399)

- 300 Multiple Choices: Indica que la solicitud tiene más de una respuesta posible.

- 301 Moved Permanently: El recurso solicitado se ha movido permanentemente a una nueva ubicación.
- 302 Found: El recurso solicitado se encuentra temporalmente en una ubicación diferente.
- 304 Not Modified: Indica que la versión actual del recurso en caché se puede usar.

5.9.4. Errores por parte de los clientes (400 - 499)

- 400 Bad Request: Indica que la solicitud del cliente es incorrecta o defectuosa. Puede deberse a una sintaxis inválida en la solicitud o a parámetros faltantes.
- 401 Unauthorized: Significa que se requiere autenticación para acceder al recurso solicitado. El cliente debe proporcionar credenciales válidas (por ejemplo, nombre de usuario y contraseña) para continuar.
- 403 Forbidden: Indica que el servidor ha entendido la solicitud, pero se niega a autorizarla. A diferencia del código 401, la autenticación no hará ninguna diferencia.
- 404 Not Found: Indica que el servidor no ha encontrado el recurso solicitado. Es el código que generalmente se muestra cuando una URL no es válida o no existe.

5.9.5. Errores de los servidores (500 - 599)

- 500 Internal Server Error: Se utiliza cuando se produce un error en el servidor que impide que la solicitud se complete. Puede deberse a un fallo en la lógica del servidor, problemas con bases de datos, entre otros.
- 502 Bad Gateway: Se produce cuando un servidor actuando como puerta de enlace o proxy recibe una respuesta no válida del servidor ascendente al que accede.
- 503 Service Unavailable: Indica que el servidor no puede manejar la solicitud en ese momento. Puede deberse a sobrecarga temporal, mantenimiento del servidor u otras circunstancias.
- 504 Gateway Timeout: Similar al 502, pero indica que el servidor ascendente no ha respondido a tiempo.
- 505 HTTP Version Not Supported: Indica que la versión HTTP utilizada en la solicitud no es compatible con el servidor.

Este listado representa en general los código de estado HTTP. Existen más codigos y subcódigos específicos que se utilizan para situaciones más detalladas. Cada código de estado tiene un propósito específico y se utiliza para comunicar el resultado de una solicitud entre el servidor y el cliente (mdn, 2023).

Al momento de consumir el API, las respuestas del servidor nos servirán como guía para saber si los requests que realizamos se están ejecutando de manera correcta, o si en su caso tenemos un error poder identificar el punto en el cual se está originando el problema.

5.10. Base de datos

Una base de datos es una recopilación de información organizada, o una estructura de datos, que suele ser almacenada dentro de un sistema informático o de manera electrónica. Usualmente las bases de datos son controladas por un sistema de gestión de bases de datos.

Existen diferentes tipos de bases de datos, sin embargo, la que actualmente es la más utilizada suele manejarse como estructuras de filas y columnas en una serie de tablas para el aumento de la eficacia del procesamiento y la consulta de datos. De esta manera se puede acceder, gestionar, modificar, actualizar, controlar y organizar fácilmente los datos. Gran parte de las bases de datos utilizan un lenguaje de consulta estructurada (SQL) para escribir y consultar datos (Oracle, 2022).

Para el proyecto se estará utilizando una base de datos relacional, la cual será montada dentro de la instancia en la nube, de esta manera será accesible en todo momento.

5.11. Secure Sockets Layer (SSL)

SSL (Secure Sockets Layer) es un protocolo de seguridad que tiene como finalidad establecer una capa de encriptación para proteger la comunicación entre un cliente y un servidor en internet. SSL ha sido reemplazado por TLS (Transport Layer Security), pero a menudo se usa el término SSL de manera más general para referirse al uso de certificados TLS/SSL para asegurar la conexión establecida entre un navegador web y un servidor web (Digicert, 2023).

El objetivo principal de SSL es garantizar que los datos transmitidos entre el cliente y el servidor estén protegidos y que no puedan ser interceptados o modificados por terceros malintencionados. Cuando un sitio web utiliza SSL, los datos se cifran antes de ser enviados a través de internet, y solo pueden ser descifrados por el servidor que posee el certificado SSL correspondiente (Digicert, 2023).

La importancia del uso del protocolo SSL radica cuando se manejan datos confidenciales, como contraseñas, información personal o datos de pago. Al proteger la comunicación entre el cliente y el servidor, SSL ayuda a prevenir ataques de intercepción, como el robo de información (como credenciales de inicio de sesión o tarjetas de crédito) durante la transmisión (Digicert, 2023).

Para utilizar SSL en un sitio web, es necesario obtener un certificado SSL válido emitido por una Autoridad de Certificación (CA). Este certificado vincula la identidad del sitio web con una clave criptográfica pública y contiene información sobre el dominio del sitio y el propietario. Una vez que el certificado SSL está instalado en el servidor web, el tráfico entre el navegador del cliente y el servidor web estará encriptado y protegido (Globalsign, 2020).

Para configurar SSL en un servidor web, se deben seguir algunos pasos básicos que dependen del software específico del servidor utilizado (por ejemplo, Apache, Nginx, IIS). Generalmente, estos pasos implican generar una solicitud de firma de certificado (CSR), enviarla a la CA para obtener el certificado SSL, configurar el servidor web para usar el

certificado y asegurarse de que las reglas de redireccionamiento y enrutamiento estén correctamente configuradas para que el sitio funcione con SSL (Globalsign, 2020).

5.12. Pruebas de estrés

Las pruebas de estrés en una API (Application Programming Interface) son un tipo de prueba de rendimiento que tiene como objetivo evaluar la capacidad y estabilidad de una API bajo una carga extrema de solicitudes y tráfico. Durante estas pruebas, se simulan situaciones de uso intensivo y se incrementa gradualmente la carga de trabajo para evaluar cómo la API responde y si es capaz de mantener un rendimiento óptimo y estable en situaciones de alta demanda (Gillis, 2023).

El propósito principal de las pruebas de estrés en una API es identificar posibles cuellos de botella, puntos débiles o problemas de rendimiento en la aplicación subyacente que podría afectar la experiencia del usuario o la disponibilidad del servicio. Estas pruebas también permiten ajustar y optimizar la configuración del servidor y la infraestructura para mejorar la capacidad de manejo de la API bajo una carga elevada (Gillis, 2023).

5.13. Control de versiones

El control de versiones permite a los desarrolladores llevar un seguimiento y manejo de los cambios realizados a un proyecto de software. Mientras más crece un proyecto, es mucho más esencial el uso de control de versiones. El control de versiones se basa en dos principios: Branching, que permite al usuario duplicar parte del código fuente (llamado repositorio). Permitiendo al desarrollador realizar cambios de manera segura a esa parte del código sin llegar a afectar al resto del proyecto.

Merging, consiste que al momento de que los desarrolladores tengan su parte de código funcionando perfectamente, estos puedan hacer un merge y unificar este código con el código fuente y hacerlo oficial (Kinsta, 2022).

En el proyecto una herramienta de control de versiones nos será de gran ayuda para llevar un registro del trabajo, así como tener una copia de seguridad para acceder a ella en cualquier momento ante cualquier inconveniente, de igual manera para estar actualizando nuestro proyecto en la instancia en la nube se facilita de gran manera al estar utilizando control de versiones, ya que podremos obtener la versión de código que se encuentre en nuestro repositorio, en lugar de pasar el código de manera manual.

5.14. SCRUM

SCRUM es un marco de trabajo ágil utilizado principalmente en el desarrollo de software, aunque su aplicación se ha extendido a otros ámbitos. Su objetivo principal es gestionar proyectos complejos de manera eficiente y adaptable, priorizando la entrega de valor de

forma incremental. SCRUM se basa en principios empíricos de control de procesos, lo que significa que se enfoca en la adaptación continua a medida que se obtiene más información (Schwaber y Sutherland, 2017).

SCRUM es un enfoque de gestión de proyectos que se centra en la flexibilidad y la adaptación continua a medida que se desarrolla el proyecto. Se basa en tres pilares fundamentales: transparencia, inspección y adaptación (Sutherland, 2014).

El marco de SCRUM se basa en roles definidos, eventos y artefactos. Algunos de los conceptos clave incluyen:

- Roles:

  Scrum Master: Es responsable de garantizar que el equipo comprenda y siga los principios y prácticas de SCRUM.

  Product Owner: Representa los intereses del cliente y es responsable de gestionar el backlog del producto y priorizar las funcionalidades.

  Equipo de Desarrollo: Son los encargados de entregar el producto final.

- Eventos:

  Sprint Planning: Reunión para planificar el trabajo que se realizará durante el sprint.

  Daily Scrum: Reunión diaria corta donde el equipo de desarrollo sincroniza sus actividades.

  Sprint Review: Reunión al final del sprint para inspeccionar el trabajo completado y adaptar el backlog del producto en consecuencia.

  Sprint Retrospective: Reunión al final del sprint donde el equipo reflexiona sobre su desempeño y busca formas de mejorar en el próximo sprint.

- Artefactos:

  Product Backlog: Lista prioritaria de todas las funcionalidades pendientes.

  Sprint Backlog: Lista de tareas que el equipo de desarrollo se compromete a completar durante el sprint.

CAPÍTULO 6

Metodología

6.1. Investigación

A lo largo de las primeras dos semanas se ha realizado una investigación bibliográfica con el fin de obtener información respecto al problema y cómo esto afecta dentro de la población guatemalteca, entre el tipo de información que se considera están: precedentes, estudios previos, soluciones desarrolladas y cualquier otro tipo de material que pueda ser de utilidad para solucionar el problema. Se delimitan claramente las necesidades y las soluciones que se pueden desarrollar.

6.2. Diseño de pruebas

Haciendo uso de la información recolectada en el paso anterior, se desarrollaron pruebas que se pasaron a los usuarios con el fin de cumplir el objetivo principal del proyecto: ayudar a mejorar su capacidad y comprensión lectora. En conjunto con estas pruebas, se definieron los casos de éxito y fracaso de las pruebas.

6.3. Ejecución de las pruebas y recolección de datos

Se realizaron las pruebas piloto con un grupo reducido de usuarios con el fin de verificar si son efectivas en alcanzar el objetivo planteado. Este paso también contempla el rediseño de las pruebas en caso de que no sean efectivas.

6.4. Planeación de casos de uso de la aplicación

Una vez que se ha validado la efectividad de las pruebas, se convirtieron estas pruebas en casos de uso de la aplicación. En este paso se contempla el diseño de casos de uso para características ajenas a las pruebas, tales como: registro de usuarios, autenticación de usuarios, edición de perfil, entre otros. Sabiendo los casos de uso se determinó el stack de tecnología que mejor se adapte a las necesidades de la aplicación.

6.5. Desarrollo de API haciendo uso de la metodología SCRUM

Se desarrolló el API haciendo uso de la metodología SCRUM, dejando un registro de los avances realizados en cada sprint. Implementando todas las funcionalidades necesarias para la aplicación web.

6.6. Creación de instancia en AWS

Creación de una instancia en AWS, creada con Ubuntu configuración de apache.

6.7. Implementación de API dentro de instancia

Se configura el API dentro de la instancia de AWS, habilitando todos los puertos necesarios e implementando apache.

6.8. Implementación de protocolo de seguridad para las solicitudes al API

Se realizó la configuración con certbot dentro de la instancia para que las solicitudes sean por medio de HTTPS y no HTTP al implementar el protocolo SSL.

6.9. Estudio de rendimiento en etapa de pruebas

Durante la etapa de pruebas se estudiará el rendimiento del API y se realizarán los cambios necesarios para cubrir los errores encontrados dentro de las pruebas.

6.10. Análisis de resultados

Se analizarán los resultados obtenidos en las pruebas a usuarios para corroborar si efectivamente hubo una mejora en la capacidad y comprensión lectora.

6.11. Cronograma de actividades

	Semana							
	Enero				Febrero			
Investigación	X	X						
Diseño de las pruebas			X					
Ejecución de las pruebas y recolección de datos				X	X			
Planeación de casos de uso de la aplicación						X		
Selección de stack a utilizar							X	
Desarrollo de API							X	X

Cuadro 2: Cronograma de actividades.

	Semana							
	Marzo				Abril			
Desarrollo de API	X							
Creación de instancia AWS		X						
Implementación de protocolo de seguridad			X					
Estudio de rendimiento en etapa de pruebas				X	X	X	X	X
Análisis de resultados								X

Cuadro 3: Cronograma de actividades.

CAPÍTULO 7

Tecnologías utilizadas

7.1. Laravel

La decisión más importante a la hora de desarrollar una API es la elección del framework a utilizar. Para ello se evaluaron los siguientes criterios:

- Funcionalidades nativas del framework: En este criterio tomamos en cuenta todo lo que podemos realizar sin la necesidad de intalar paquetes y librerias extras, lo cual conlleva a mayor tiempo de desarrollo,
- Velocidad del ciclo de desarrollo: En este criterio consideramos la practicidad del framework a la hora de desarrollar, esto permite una mayor comodidad y velocidad en el desarrollo.
- Soporte de versiones: En este criterio vemos que tanto mantenimiento reciben los frameworks en sus diferentes versiones y si manejan versiones *Long term support*, la cuales son versiones que se comprometen a brindarle soporte por un tiempo más extendido que las demás versiones.
- ORM: En este apartado comparamos los *Object-Relational Mapping*, la herramienta encargada de facilitar la interacción entre aplicaciones orientadas a objetos y bases de datos relacionales.
- Documentación y comunidad: Un factor muy importante es la documentación que existe en internet y la comunidad que existe trabajando para brindar apoyo en foros.

Criterios	Laravel	Django	Express	Spring
Funcionalidades nativas del framework	Amplio, con sistema de enrutamiento y controladores, interfaz de línea de comandos llamada Artisan, programación orientada a eventos, sistema de usuarios, junto a validaciones de correo y autenticación de dos pasos.	Sistema de administración para gestionar modelos de bases de datos.	Enrutador incorporado, gestión de usuarios, autenticación y autorización.	Contenedor IoC que gestiona la creación y administración de objetos y sus dependencias.
Velocidad del ciclo de desarrollo	Lenguaje interpretado y actualización en vivo.	Lenguaje interpretado y actualización en vivo.	Lenguaje interpretado y actualización en vivo.	Lenguaje compilado, lento de ejecutar.
Soporte de versiones	Versión LTS activa	Versión LTS activa	Ciclo de vida de soporte menos formalizado	Versión LTS activa
ORM	Laravel incluye Eloquent ORM, que es un ORM elegante y expresivo que facilita el mapeo de objetos PHP a tablas de bases de datos relacionales.	Django incluye su propio ORM incorporado que facilita el mapeo de objetos Python a tablas de bases de datos relacionales.	Express no incluye ORM nativo com parte de su framework.	Hibernate: ORM utilizado en Spring, madura y ampliamente adoptada que facilita el mapeo de objetos Java a tablas de bases de datos.
Documentación y comunidad	La documentación oficial de Laravel es altamente elogiada por su calidad y accesibilidad. Es completa y bien organizada, con guías detalladas, tutoriales, referencias de API y ejemplos de código.	La documentación oficial de Django es reconocida por su calidad y exhaustividad. Está bien estructurada y cubre todos los aspectos del framework.	La documentación oficial de Express.js es clara y concisa, proporcionando información detallada sobre cómo utilizar el framework, incluyendo ejemplos de código y descripciones de API	Spring ofrece una documentación oficial extensa y bien organizada en su sitio web. Esta documentación incluye guías detalladas, tutoriales, referencias de API y ejemplos de código.

Cuadro 4: Cuadro comparativo de características de frameworks para el desarrollo web.

Luego de comparar las diferentes opciones que existen, se decide trabajar con Laravel, un framework de desarrollo web de código abierto escrito en el lenguaje de programación PHP. Fue creado por Taylor Otwell en 2011 y ha ganado una gran popularidad en la comunidad de desarrollo web debido a su facilidad de uso, su enfoque en la simplicidad y la elegancia del código, y sus características avanzadas como el enrutamiento, la autenticación, la gestión de sesiones y la generación de vistas.

Laravel se basa en el patrón de arquitectura de software MVC (Modelo Vista Controlador), que separa los aspectos de la aplicación relacionados con la lógica de negocios, la presentación y la interacción del usuario. Además, ofrece una amplia gama de características que permiten a los desarrolladores construir aplicaciones web de alta calidad y escalables de manera rápida y eficiente.

Algunas de las características principales de Laravel incluyen:

- Routing: Laravel ofrece un sistema de enrutamiento flexible y fácil de usar que permite definir rutas para diferentes URL y métodos HTTP.
- Migraciones de base de datos: Laravel incluye una herramienta de migración de base de datos que permite a los desarrolladores crear y modificar tablas de bases de datos de manera programática.
- Eloquent ORM: Laravel utiliza un mapeo objeto-relacional (ORM) llamado Eloquent que simplifica la interacción con bases de datos y elimina la necesidad de escribir consultas SQL directamente.
- Plantillas Blade: Laravel ofrece un motor de plantillas llamado Blade que permite a los desarrolladores crear y reutilizar componentes de vista de manera eficiente.
- Autenticación: Laravel incluye una capa de autenticación integrada que permite a los desarrolladores agregar autenticación y autorización a sus aplicaciones de manera fácil y rápida.
- Testing: Laravel incluye un conjunto completo de herramientas para pruebas unitarias y de integración que facilitan la tarea de escribir y ejecutar pruebas de calidad para su código.
- Modularidad: Laravel está diseñado para ser modular y extensible, lo que significa que los desarrolladores pueden agregar fácilmente nuevas funcionalidades a través de complementos y paquetes.

Laravel es una excelente opción para los desarrolladores que buscan un framework de desarrollo web moderno, fácil de usar y altamente escalable que les permita construir aplicaciones web de alta calidad de manera rápida y eficiente. Sus características avanzadas y su enfoque en la simplicidad y la elegancia del código lo convierten en una herramienta poderosa para cualquier proyecto web.

Las alternativas de frameworks más conocidos son Django, ExpressJS, Ruby on Rails, Flask, ASP .NET. Sin embargo se ha decidido utilizar Laravel debido a las funcionalidades que ya se incorporan por defecto, sin necesidad de instalar librerías, como lo es todo el manejo de usuarios y autenticación, o como lo podría ser el manejo de CORS.

7.2. PHP

PHP es un lenguaje interpretado de propósito general ampliamente usado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores. Es también el módulo Apache más popular entre las computadoras que utilizan Apache como servidor web.

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

Aunque todo en su diseño está orientado a facilitar la creación de página web, es posible crear aplicaciones con una interfaz gráfica para el usuario, utilizando la extensión PHP-Qt o PHP-GTK. También puede ser usado desde la línea de órdenes, de la misma manera como Perl o Python pueden hacerlo; a esta versión de PHP se la llama PHP-CLI (Command Line Interface). Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el script solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Mediante extensiones es también posible la generación de archivos PDF, Flash, así como imágenes en diferentes formatos. Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como UNIX (y de ese tipo, como Linux o Mac OS X) y Windows, y puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI.

PHP es una alternativa a las tecnologías de Microsoft ASP y ASP.NET, a ColdFusion de la compañía Adobe, a JSP/Java de Sun Microsystems, y a CGI/Perl.

7.3. SQL

El lenguaje de consulta estructurado (SQL Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar de una forma sencilla información de interés de una base de datos, así como también hacer cambios sobre la misma.

Criterios	Relacional	No relacional
Modelo de datos	Tabular	Clave-valor, documento o gráfico
Tipo de datos	Estructurado	Estructurados, semiestructurados y sin estructurar
Integridad de los datos	Alta, con total conformidad con ACID	Modelo de coherencia eventual
Rendimiento	Se ha mejorado al agregar más recursos al servidor	Se ha mejorado al agregar más nodos de servidor
Escalado	El escalado horizontal requiere estrategias de administración de datos adicionales	El escalado horizontal es sencillo

Cuadro 5: Cuadro comparativo de características de bases de datos relacionales en comparación con bases de datos no relacionales.

Se ha trabajado con bases de datos estructuradas debido a la naturaleza de los datos, permitiendo agregaciones fáciles sobre grandes volúmenes de datos, además de representar un sistema más robusto y con mayor documentación.

7.3.1. MySQL

Criterios	MySQL	SQLite	PostgreSQL	Oracle
Tipo de licencia	Código abierto (GPL)	Dominio público	Código abierto (PostgreSQL)	Comercial
Soporte ACID	Si	Si	Si	Si
Soporte de versiones	Bueno	Limitado	Bueno	Muy bueno
Complejidad de instalación	Moderada	Baja	Moderada	Alta

Cuadro 6: Cuadro comparativo de características de sistemas de gestión de bases de datos relacionales.

Se ha decidido utilizar MySQL debido a que éste funciona mejor ante funcionalidades no tan complejas y además debido a que el lenguaje de programación que se decidió utilizar ha sido PHP y MySQL combina de gran manera con este lenguaje, ya que los dos crecieron de igual manera en tiempos similares, inclusive dentro de las oficinas de MySQL se contrataron programadores específicamente para que MySQL trabaje correctamente de la mano con

PHP.

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

MySQL es muy utilizado en aplicaciones web como MediaWiki, Amazon, Yahoo, Flickr o Drupal; en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, siendo esta una de las razones por la cuál se optó por utilizar esta herramienta.

Se utilizó esta herramienta debido a su gran disponibilidad y escalabilidad, además de un fácil manejo de seguridad utilizando la encriptación SSL.

7.4. Postman

Postman es una herramienta de colaboración y desarrollo de API (Interfaces de programación de aplicaciones) que permite a los desarrolladores probar, documentar y compartir fácilmente API. Es una aplicación de escritorio que se puede utilizar en Windows, macOS y Linux.

Existen diferentes alternativas cono Testfully, Insomia, Hoppscotch, Apidog o extensiones dentro de editores de texto, sin embargo se ha seleccionado Postman debido a que es la herramienta standarizada para las pruebas de API RESTful y la facilidad al exportar e importar espacios de trabajo.S

Postman ofrece una interfaz de usuario intuitiva que permite a los desarrolladores crear, enviar y recibir solicitudes HTTP y HTTPS, lo que les permite probar fácilmente la funcionalidad de sus API. Además, ofrece una serie de características avanzadas que permiten a los desarrolladores colaborar en tiempo real y automatizar tareas repetitivas.

7.5. AWS - Amazon Web Services

Al decidir que infraestructura utilizaremos es importante considerar los siguientes criterios:

- Costos: Analizar donde se tienen los precios que mejor se adapten a la situación que nos encontremos, en este caso se considera la disponibilidad de servicios gratuitos.
- Regiones de disponibilidad: Debemos asegurarnos que existan zonas de disponibilidad cercanas a los lugares donde deseamos utilizar el servicio.

- Catálogo de servicios: En este criterio consideramos todos los servicios que tienen a dispocisión las diferentes herramientas.
- Fiabilidad: Es importante que el servicio que tengamos nos brinde la seguridad que están disponibles en todo momento, para ello consideramos los *service-level agreement (SLA)* lo cual representa un compromiso por parte del proveedor hacia los usuarios, asegurando un porcentaje mínimo de tiempo sobre el que el sistema no fallará.
- Popularidad: También es importante considerar la presencia de los servicios en el mercado, si presenta mayor popularidad existirá mayor documentación y ayuda por parte de la comunidad.

Criterios	AWS	Google Cloud	Azure	Digital Ocean
Costo	Cuenta con servicio gratis en los servicios más básicos durante un año	Brinda 300USD que pueden ser utilizables en los primeros 3 meses	Servicios populares básicos gratis por un año	Brinda 200USD utilizables en los primeros 2 meses
Regiones de disponibilidad	24 regiones, 77 zonas de disponibilidad	26 regiones, 77 zonas de disponibilidad	60+ regiones	14 centros de datos, 45 data centers
Catálogo de servicios	El más amplio de todos, con variedad de servicios en todas las categorías	Amplio, con énfasis en servicios de datos y análisis	Amplio, con enfoque en servicios empresariales	Amplio, con énfasis en simplicidad y facilidad de uso
Fiabilidad (SLA)	Ofrece SLA para la mayoría de los servicios, con altos niveles de disponibilidad	SLA con altos niveles de disponibilidad, respaldado por una infraestructura robusta	SLA con altos niveles de disponibilidad, especialmente para servicios principales	Ofrece SLA, pero con enfoque en simplicidad, con niveles aceptables de disponibilidad
Popularidad	41.5 % del mercado	3.0 % del mercado	29.4 % del mercado	<1 % del mercado

Cuadro 7: Cuadro comparativo de características de sistemas de gestión de bases de datos relacionales.

AWS es el mayor ejemplar en cuanto computación en la nube nos referimos, siendo sus mayores competidores Azure y Google Computing. Todas poseen sus ventajas y desventajas a la hora de compararlas, sin embargo AWS gana en las áreas más cruciales.

Principalmente siendo que cuenta con la mayor cantidad de servicios, teniendo más de 200 activos, siguiendo Azure con más de 0 y luego Google Computing con más de 60. Esto permite que al momento de desear realizarse cualquier tipo de escalabilidad AWS sea más propenso a tener herramientas adaptadas para la funcionalidad que se desea implementar.

De igual manera tiene a su disposición más zonas de disponibilidad de servidores, por lo cual si se desea conectar hacia un servidor, la probabilidad de que se encuentre uno cercano

a la zona es más probable. Además al ser la más estandarizada en el mercado, es la que posee la mayor cantidad de documentación.

7.6. Amazon EC2

Amazon Elastic Compute Cloud (EC2) es un servicio web de Amazon Web Services (AWS) que ofrece a los usuarios la capacidad de ejecutar aplicaciones en máquinas virtuales escalables en la nube. Es decir, Amazon EC2 permite crear y gestionar instancias de servidores virtuales en la nube, con la posibilidad de aumentar o disminuir la capacidad de procesamiento según se requiera.

Amazon EC2 se basa en el concepto de instancias, que son unidades de cómputo virtual que se ejecutan en la nube de AWS. Estas instancias pueden ser configuradas con diferentes tipos de procesadores, memoria, almacenamiento y otros recursos, según las necesidades del usuario. Además, Amazon EC2 permite escalar fácilmente la capacidad de cómputo, ya sea aumentando o disminuyendo el número de instancias en ejecución.

7.7. Amazon S3

Amazon S3 (Simple Storage Service) es un servicio de almacenamiento en la nube de Amazon Web Services (AWS) que ofrece una solución escalable, segura y de alta disponibilidad para el almacenamiento y recuperación de datos en la nube. Es uno de los servicios más populares de AWS y es utilizado por empresas de todos los tamaños y sectores.

Amazon S3 se basa en el concepto de objetos, que son unidades de datos que pueden contener desde pequeños archivos hasta grandes conjuntos de datos estructurados. Estos objetos se almacenan en "buckets.º depósitos, que son contenedores de almacenamiento virtual en la nube. Los usuarios pueden crear, modificar y eliminar buckets, así como subir y descargar objetos a y desde los mismos.

7.8. Nginx

Nginx es un servidor web de código abierto de alta performance que se utiliza para alojar y servir sitios web. Es conocido por su capacidad para manejar grandes cantidades de tráfico y su escalabilidad, lo que lo convierte en una opción popular para sitios web de alto tráfico.

Nginx fue diseñado para ser un servidor web ligero y rápido, con una arquitectura modular que permite a los desarrolladores personalizar y ampliar sus capacidades. Nginx utiliza una arquitectura basada en eventos y un modelo de proceso asincrónico que permite manejar múltiples conexiones de manera eficiente y minimizar el uso de recursos. Además de servir como un servidor web, Nginx también se puede utilizar como un proxy inverso, lo que significa que se puede configurar para actuar como un intermediario entre el usuario y el servidor web final. Esto permite a los desarrolladores optimizar el rendimiento y la seguridad de los

sitios web, ya que Nginx puede actuar como un firewall de aplicaciones web, equilibrador de carga, caché de contenido y proxy de SSL/TLS.

7.9. Certbot

Certbot es una herramienta gratuita y de código abierto para la gestión de certificados SSL/TLS, desarrollada por la Electronic Frontier Foundation (EFF). Es un cliente de automatización de la Autoridad de Certificación Let's Encrypt, que proporciona certificados SSL/TLS gratuitos para la creación de conexiones seguras en la web.

Los certificados SSL/TLS son esenciales para proteger la información que se transfiere entre un servidor web y los usuarios finales, ya que cifran los datos en tránsito y garantizan que la conexión sea auténtica. Certbot simplifica el proceso de obtención y renovación de estos certificados mediante la automatización de las tareas de verificación de la propiedad del dominio y la configuración del servidor web.

Certbot soporta una amplia variedad de sistemas operativos y servidores web, incluyendo Apache, Nginx y otros. Al utilizar Certbot, los usuarios pueden generar automáticamente los certificados SSL/TLS para sus sitios web, configurar sus servidores web para usarlos, y programar la renovación automática de los mismos antes de que expiren.

7.10. Github

Github es un servicio basado en la nube que ayuda a los desarrolladores a almacenar y llevar un manejo de su código. Siendo su principal objetivo llevar seguimiento y control de los cambios realizados al código.

7.11. Dependabot

Dependabot es una herramienta integrada en Github para mantener las dependencias actualizadas y seguras en un proyecto. Ayuda a los equipos de desarrollo a estar al tanto de las actualizaciones y evitar problemas de seguridad o incompatibilidades con versiones más nuevas de las dependencias. Permite a los desarrolladores centrarse en el desarrollo de nuevas características y corrección de errores, mientras Dependabot maneja la tarea de mantener las dependencias actualizadas (Github, 2023).

7.12. Locust

Locust es una herramienta que se encarga de crear sets de funciones de prueba para simular una carga alta de usuarios. Esto ayuda a determinar el principal punto de quiebre en términos de eficiencia, seguridad y manejo de cargas. Para las pruebas de carga, Locust

utiliza Python, además de presentar los resultados de las pruebas en un dashboard (Echout, 2022).

Se ha utilizado Locust sobre otras herramientas por la sólida documentación que ésta posee, posibilitando un fácil entendimiento e implementación. Además de contar con una interfaz agradable con el usuario, brindando dashboards, visualizaciones y reportes que resumen el proceso de pruebas de carga (Echout, 2022).

CAPÍTULO 8

Resultados

8.1. Estructura de Base de Datos

La siguiente estructura muestra las diferentes tablas en las que se almacena la información y cómo los los diferentes datos se conectan entre sí.

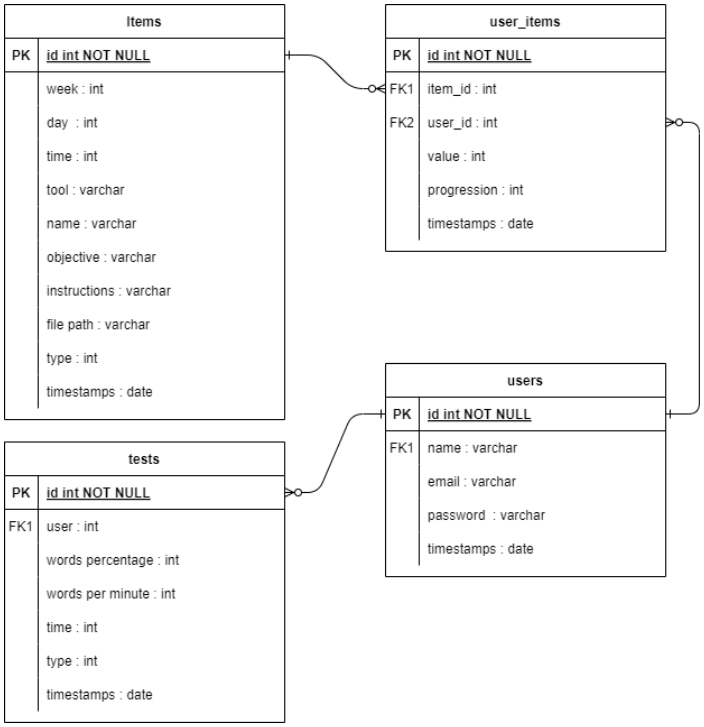


Figura 2: Estructura de Base de Datos

8.1.1. Tabla 1 *Items*

Esta tabla representa todos los diferentes ejercicios que se realizan a lo largo de las semanas dentro del programa. Se encarga de almacenar todos los datos necesarios, junto a los documentos necesarios para la realización de los mismos.

Esta tabla consta de las siguientes entidades:

- ID: Identificador de tipo *id*. Valor entero sin signo, utilizado como identificador único.
- Week: Dato de tipo *unsignedTinyInteger*, utilizado para establecer a qué semana pertenece el ejercicio.
- Day: Dato de tipo *unsignedTinyInteger*, utilizado para establecer a qué día pertenece el ejercicio.
- Time: Dato de tipo *unsignedSmallInteger*, utilizado para establecer cuánto es el tiempo de duración del ejercicio.
- Tool: Dato de tipo *String*, utilizado para saber qué herramientas son necesarias para la realización del ejercicio, este campo no es obligatorio.
- Name: Dato de tipo *String*, utilizado para guardar el nombre del ejercicio.
- Objective: Dato de tipo *Text*, utilizado para almacenar el objetivo del ejercicio.
- Instructions: Dato de tipo *Text*, utilizado para almacenar las instrucciones del ejercicio.
- File path: Dato de tipo *String*, utilizado para almacenar el enlace al documento dentro del almacenamiento S3 necesario para la realización del ejercicio.
- Type: Dato de tipo *Enum*, utilizado para saber de qué tipo es el ejercicio.
- Timestamps: Dato de tipo *Timestamps*, utilizado para guardar la fecha de creación y fechas de edición del ejercicio.

8.1.2. Tabla 2 *Users*

En esta tabla se encuentran todos los datos de los usuarios que estarán utilizando el sistema.

Esta tabla consta de las siguientes entidades:

- ID: Identificador de tipo *id*. Valor entero sin signo, utilizado como identificador único.
- Name: Dato de tipo *String*, utilizado para guardar el nombre del usuario.
- Email: Dato de tipo *String*, utilizado para guardar el correo electrónico del usuario. Este dato tiene que ser único.

- Password: Dato de tipo *String*, utilizado para guardar la contraseña del usuario, el dato ya se encuentra previamente encriptado.
- Timestamps: Dato de tipo *Timestamps*, utilizado para guardar la fecha de creación y fechas de edición del usuario.

8.1.3. Tabla 3 *User_items*

Esta tabla es del tipo *many-to-many*, el cual se encarga de relacionar a cada usuario con sus respectivos ejercicios a realizar, de igual manera en esta tabla se guardarán los resultados de los usuarios en cada uno de los ejercicios y el progreso que tienen.

Esta tabla consta de las siguientes entidades:

- ID: Identificador de tipo *id*. Valor entero sin signo, utilizado como identificador único.
- User: Dato de tipo *Foreign Key*, que hace referencia al usuario que realiza el ejercicio.
- Value: Dato de tipo *unsignedSmallInteger*, utilizado para guardar la calificación del usuario en el ejercicio.
- Progression: Dato de tipo *Enum*, utilizado para indicar el progreso que tiene el usuario en el ejercicio.
- Timestamps: Dato de tipo *Timestamps*, utilizado para guardar la fecha de creación y fechas de edición de la entrada en la tabla.

8.1.4. Tabla 4 *Tests*

En esta tabla se guardan los datos de los resultados de los usuarios en los diferentes exámenes a realizar a lo largo del uso del programa.

Esta tabla consta de las siguientes entidades:

- ID: Identificador de tipo *id*. Valor entero sin signo, utilizado como identificador único.
- User: Dato de tipo *Foreign Key*, que hace referencia al usuario que realiza el examen.
- Words_percentage: Dato de tipo *unsignedTinyInteger*, representa el porcentaje de respuestas correctas que obtuvo el usuario en el exámen.
- Words_per_minute: Dato de tipo *unsignedSmallInteger*, representa la cantidad de palabras por minuto que logró leer el usuario en el exámen.
- Time: Dato de tipo *unsignedSmallInteger*, utilizado para almacenar la cantidad de tiempo que le ha tomado al usuario realizar la lectura.
- Type: Dato de tipo *unsignedTinyInteger*, utilizado para indicar el tipo de exámen que ha realizado el usuario.

- Timestamps: Dato de tipo *Timestamps*, utilizado para guardar la fecha de creación y fechas de edición de la entrada en la tabla.

8.2. Endpoints

Todos los endpoint reciben y retornan únicamente objetos JSON con la siguiente información:

8.2.1. Register

Request de tipo Post utilizado para registrar un nuevo usuario

Parámetros a recibir	**Datos de retorno**
Nombre	Token de usuario registrado
Correo electrónico	
Contraseña	
Confirmación de contraseña	

Cuadro 8: Datos de endpoint Register.

8.2.2. Login

Request de tipo Post utilizado para identificar a un usuario

Parámetros a recibir	**Datos de retorno**
Correo electrónico	Token de usuario identificado
Contraseña	

Cuadro 9: Datos de endpoint Login.

8.2.3. CreateItem

Request de tipo Post utilizado para identificar a un usuario

Parámetros a recibir	**Datos de retorno**
Nombre de ejercicio	Objeto de ejercicio creado
Objetivo de ejercicio	
Instrucciones de ejercicio	
Herramienta a utilizar en el ejercicio	
Día al que pertenece el ejercicio	
Semana al que pertenece el ejercicio	
Tiempo que dura el ejercicio	
Tipo de ejercicio	
Enlace hacia el documento del ejercicio	

Cuadro 10: Datos de endpoint createItem.

8.2.4. CreateUserItem

Request de tipo Post utilizado para crear la relación entre usuario y ejercicio

Parámetros a recibir	**Datos de retorno**
Usuario que realiza el ejercicio	Objeto de relación creada
Ejercicio que se está ejecutando	
Puntaje	
Pregresión del ejercicio	

Cuadro 11: Datos de endpoint CreateUserItem.

8.2.5. GetItems

Request de tipo Get utilizado para obtener los ejercicios a los que está asignado el usuario

Parámetros a recibir	**Datos de retorno**
Usuario que realiza la solicitud	Lista de ejercicios a los que el usuario está asignado

Cuadro 12: Datos de endpoint GetItems.

8.2.6. CurrentUser

Request de tipo Get utilizado para obtener todos los datos del usuario que lo solicita

Parámetros a recibir	**Datos de retorno**
Usuario que realiza la solicitud	Nombre del usuario
	Correo electrónico
	Ha completado el tutorial
	Ha completado el test inicial
	Ha completado el test final
	Tiene acceso al test final

Cuadro 13: Datos de endpoint CurrentUser.

8.2.7. CreateTest

Request de tipo Post utilizado para crear un nuevo examen realizado por el usuario

Parámetros a recibir	**Datos de retorno**
Usuario que realiza el examen	Objeto de examen creado
Porcentaje de palabras	
Palabras por minuto	
Tiempo	
Tipo de examen	

Cuadro 14: Datos de endpoint CreateTest.

8.2.8. Dashboard

Request de tipo Get utilizado para obtener el progreso por semana del usuario en cuestión

Parámetros a recibir	**Datos de retorno**
Usuario que realiza la solicitud	Información del progreso de todas las semanas

Cuadro 15: Datos de endpoint Dashboard.

8.2.9. Weeks

Request de tipo Get utilizado para obtener información del avance de ejercicios de una semana en específico del usuario que lo solicita

Parámetros a recibir	**Datos de retorno**
Usuario que realiza la solicitud	Dada la semana, se brinda información específica del avance por día y ejercicio
Semana de la cual deseamos obtener la información	

Cuadro 16: Datos de endpoint Weeks.

8.2.10. Progress

Request de tipo Post utilizado para actualizar el progreso de un usuario cuando termina un ejercicio

Parámetros a recibir	**Datos de retorno**
Usuario que realiza la solicitud	Valores antiguos del ejercicio
Id del ejercicio que se está avanzando	Valores nuevos del ejercicio
Calificación obtenida	

Cuadro 17: Datos de endpoint Progress.

8.2.11. FinalAvailability

Request de tipo Get que indica si el usuario actual puede acceder al examen final

Parámetros a recibir	**Datos de retorno**
Usuario que realiza la solicitud	Verdadero o falso según si el usuario tiene acceso al exámen final

Cuadro 18: Datos de endpoint FinalAvailability.

8.2.12. Results

Request de tipo Get que retorna los resultados del usuario en sus respectivos exámenes

Parámetros a recibir	**Datos de retorno**
Usuario que realiza la solicitud	Valores de la prueba de inicio
	Valores de la prueba final
	Mejoras que se presentaron

Cuadro 19: Datos de endpoint Results.

8.3. Resultados en sujetos de prueba

Dentro del programa se realiza una prueba de diagnóstico inicial y una prueba de diagnóstico final luego de realizar todas las pruebas establecidas. Estas pruebas proveen textos a los usuarios para luego realizar una serie de preguntas relacionadas al texto para medir el nivel de comprensión lectora y la velocidad de lectura basada en la longitud del texto y el tiempo que se tarda en completarlo, medido en palabras por minuto.

Las pruebas se realizaron a 10 jóvenes bajo el permiso de sus padres, los resultados han sido los siguientes:

Sujeto	**Porcentaje de mejora en comprensión lectora**	**Porcentaje de mejora en palabras por minuto**
Usuario 1	32.84	13.08
Usuario 2	14.10	8.53
Usuario 3	39.29	33.08
Usuario 4	28.21	1.39
Usuario 5	58.93	7.39
Usuario 6	0	6.44
Usuario 7	14.10	15.92
Usuario 8	28.21	8.13
Usuario 9	32.84	5.15
Usuario 10	0	4.37
Promedio	**27.66**	**10.35**

Cuadro 20: Resultados en sujetos de prueba.

CAPÍTULO 9

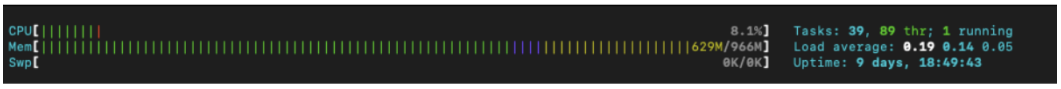
Rendimiento

Para el hosting o alojamiento del api se ha utilizado una instancia EC2 de AWS de tipo t2.micro la cual consta de las siguientes especificaciones:

- 1 vCPU
- 1.0 GB de memoria RAM
- Velocidad de reloj de 3.3 GHz
- Arquitectura de CPU i386

El desempeño logrado ha sido el siguiente:

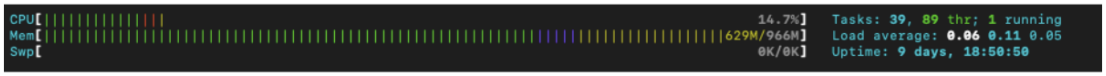
9.1. 10 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	41	0	110	150	380	138	108	385	173	0.5	0
GET	/api/final_availability	62	0	120	140	370	130	108	375	20	0.1	0
GET	/api/items	59	0	460	630	880	465	294	885	76445	0.5	0
GET	/api/results	48	0	110	130	180	116	106	176	450	0.3	0
POST	/api/sanctum/token	10	0	1000	1400	1400	1032	714	1438	57	0	0
POST	/api/test	61	0	98	110	290	106	94	292	25	0.8	0
GET	/api/user	50	0	120	170	500	137	111	501	301	0.5	0
GET	/api/users	47	0	100	120	430	118	100	428	2198	0.2	0
	Aggregated	**378**	**0**	**120**	**460**	**1000**	**201**	**94**	**1438**	**12330**	**2.9**	**0**

Figura 3: Rendimiento con 10 usuarios

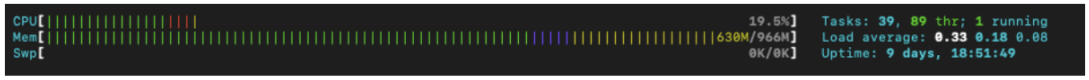
9.2. 20 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	164	0	110	130	370	124	108	385	173	1.1	0
GET	/api/final_availability	206	0	120	140	450	133	108	641	20	1	0
GET	/api/items	198	0	430	560	1100	462	289	1320	76445	0.9	0
GET	/api/results	195	0	110	130	580	128	105	591	450	0.9	0
POST	/api/sanctum/token	20	0	1000	1600	1600	1065	714	1585	57	0	0
POST	/api/test	195	0	99	110	240	104	93	292	25	0.8	0
GET	/api/user	193	0	120	140	560	136	109	599	301	0.8	0
GET	/api/users	190	0	100	120	340	112	99	428	2198	0.8	0
	Aggregated	**1361**	**0**	**110**	**420**	**1000**	**186**	**93**	**1585**	**11564**	**6.3**	**0**

Figura 4: Rendimiento con 20 usuarios

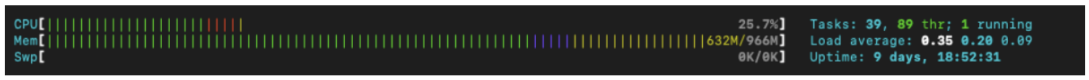
9.3. 30 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	324	0	120	130	310	122	106	385	173	1.7	0
GET	/api/final_availability	351	0	120	150	450	134	108	641	20	1	0
GET	/api/items	340	0	440	560	1100	461	289	1401	76445	1.7	0
GET	/api/results	325	0	110	140	460	131	105	591	450	1	0
POST	/api/sanctum/token	30	0	1000	1400	1600	1023	599	1585	57	0	0
POST	/api/test	344	0	100	110	300	107	93	608	25	0.9	0
GET	/api/user	326	0	120	140	500	134	109	599	301	1.2	0
GET	/api/users	362	0	110	130	620	124	98	827	2198	1.7	0
	Aggregated	**2402**	**0**	**120**	**420**	**880**	**184**	**93**	**1585**	**11284**	**9.2**	**0**

Figura 5: Rendimiento con 30 usuarios

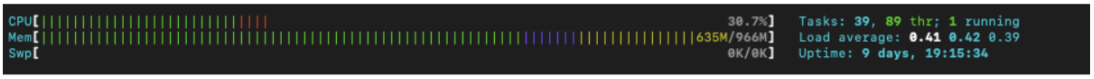
9.4. 40 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	233	0	120	150	2800	195	108	3346	173	2	0
GET	/api/final_availability	233	0	120	140	450	147	109	3221	20	2	0
GET	/api/items	227	0	410	580	1200	456	293	1644	76445	2.2	0
GET	/api/results	247	0	110	140	2700	198	104	3418	450	1.4	0
POST	/api/sanctum/token	40	0	2500	4300	4400	2632	825	4399	57	0	0
POST	/api/test	222	0	100	130	1300	150	92	2384	25	0.6	0
GET	/api/user	206	0	120	140	1700	165	110	3092	301	1.5	0
GET	/api/users	221	0	110	150	3000	198	97	3293	2198	2.1	0
	Aggregated	**1629**	**0**	**120**	**450**	**3300**	**275**	**92**	**4399**	**11089**	**11.8**	**0**

Figura 6: Rendimiento con 40 usuarios

9.5. 50 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	276	0	120	160	3000	187	109	3354	173	2.1	0
GET	/api/final_availability	265	0	130	160	1500	176	109	4225	20	2.4	0
GET	/api/items	283	0	440	760	3400	560	292	4168	76445	2.9	0
GET	/api/results	274	0	120	140	3300	182	105	3713	450	1.6	0
POST	/api/sanctum/token	50	0	3300	5100	5600	3312	1111	5564	57	0	0
POST	/api/test	270	0	110	150	1500	161	93	3325	25	2.7	0
GET	/api/user	302	0	130	180	3800	251	111	4146	301	1.7	0
GET	/api/users	267	0	110	160	3700	214	99	4141	2198	1.7	0
	Aggregated	**1987**	**0**	**120**	**490**	**4200**	**326**	**93**	**5564**	**11322**	**15.1**	**0**

Figura 7: Rendimiento con 50 usuarios

9.6. 60 usuarios

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	352	0	120	150	960	148	108	1425	173	2.4	0
GET	/api/final_availability	383	0	130	160	1300	160	107	1543	20	1.9	0
GET	/api/items	391	0	420	570	1200	456	290	1690	76445	3.1	0
GET	/api/results	379	0	120	160	1200	157	106	1390	450	2.4	0
POST	/api/sanctum/token	60	0	1300	1800	2400	1352	618	2389	57	0	0
POST	/api/test	396	0	110	140	1200	142	92	1383	25	2.5	0
GET	/api/user	348	0	130	160	890	160	108	1392	301	2.7	0
GET	/api/users	401	0	110	140	820	134	96	1394	2198	2.9	0
	Aggregated	**2710**	**0**	**120**	**450**	**1500**	**221**	**92**	**2389**	**11487**	**17.9**	**0**

Figura 8: Rendimiento con 60 usuarios

9.7. 70 usuarios

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	482	0	130	160	1300	162	105	1659	173	2.9	0
GET	/api/final_availability	480	0	130	160	870	152	109	1289	20	3.8	0
GET	/api/items	449	0	410	560	1700	458	288	2079	76445	3.1	0
GET	/api/results	489	0	120	160	1300	163	101	1461	450	3.2	0
POST	/api/sanctum/token	70	0	1400	2100	2500	1456	650	2505	57	0	0
POST	/api/test	494	0	110	130	1200	133	93	1755	25	2.8	0
GET	/api/user	470	0	130	160	1300	159	110	1779	301	2.9	0
GET	/api/users	464	0	110	140	980	147	96	1768	2198	3	0
	Aggregated	**3398**	**0**	**130**	**430**	**1600**	**220**	**93**	**2505**	**10540**	**21.7**	**0**

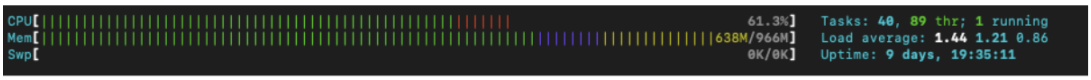
Figura 9: Rendimiento con 70 usuarios

9.8. 80 usuarios

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	410	0	130	270	1800	224	108	2308	173	3.6	0
GET	/api/final_availability	390	0	140	210	1800	211	110	2279	20	4.2	0
GET	/api/items	346	0	390	790	2100	502	294	2598	76445	2.8	0
GET	/api/results	366	0	120	190	1900	209	102	2327	450	3.1	0
POST	/api/sanctum/token	80	0	1800	2500	10000	1925	941	10215	57	0	0
POST	/api/test	390	0	110	170	1800	164	92	2245	25	3.6	0
GET	/api/user	398	0	130	210	1800	202	111	2232	301	4.1	0
GET	/api/users	385	0	120	170	1500	179	99	1878	2198	4	0
	Aggregated	**2765**	**0**	**130**	**490**	**2200**	**286**	**92**	**10215**	**10009**	**25.4**	**0**

Figura 10: Rendimiento con 80 usuarios

9.9. 90 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	565	0	130	180	1900	184	107	2461	173	2.9	0
GET	/api/final_availability	586	0	130	180	1700	180	108	2285	20	5.3	0
GET	/api/items	584	0	370	500	2200	445	286	2765	76445	3.6	0
GET	/api/results	544	0	120	160	1900	188	104	2430	450	3.7	0
POST	/api/sanctum/token	90	0	1700	2600	2800	1734	630	2834	57	0	0
POST	/api/test	558	0	110	140	1700	157	92	2323	25	4.3	0
GET	/api/user	566	0	140	180	1900	196	111	2123	301	4.5	0
GET	/api/users	541	0	120	160	1900	171	96	2372	2198	4	0
	Aggregated	**4034**	**0**	**130**	**410**	**2200**	**253**	**92**	**2834**	**11496**	**28.3**	**0**

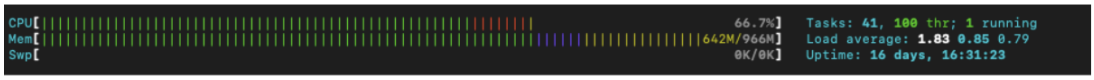
Figura 11: Rendimiento con 90 usuarios

9.10. 100 usuarios

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	567	0	130	200	2800	244	107	3250	173	4.6	0
GET	/api/final_availability	539	0	140	200	2700	245	110	3335	20	4.1	0
GET	/api/items	584	0	400	670	3200	520	292	3694	76445	4.2	0
GET	/api/results	593	0	130	200	2600	222	103	3250	450	4	0
POST	/api/sanctum/token	100	0	2600	3500	4800	2471	681	4849	57	0	0
POST	/api/test	553	0	110	160	2600	188	92	3156	25	4.2	0
GET	/api/user	565	0	140	200	2800	239	111	3298	301	5.4	0
GET	/api/users	528	0	120	190	2700	204	98	3325	2198	4.7	0
	Aggregated	**4029**	**0**	**140**	**490**	**3100**	**323**	**92**	**4849**	**11509**	**31.2**	**0**

Figura 12: Rendimiento con 100 usuarios

9.11. 110 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	589	0	140	210	2600	232	107	2814	173	4.4	0
GET	/api/final_availability	648	0	140	220	2100	226	110	2812	20	5.7	0
GET	/api/items	584	0	390	580	2400	494	291	3046	76445	5.1	0
GET	/api/results	588	0	130	210	2300	212	105	2741	450	4.4	0
POST	/api/sanctum/token	110	0	1800	2900	3400	1902	695	3587	57	0	0
POST	/api/test	600	0	110	180	2500	237	92	2749	25	5.4	0
GET	/api/user	601	0	150	210	2500	233	111	2755	301	4.9	0
GET	/api/users	615	0	120	190	2000	198	98	2719	2198	5	0
	Aggregated	**4335**	**0**	**140**	**490**	**2600**	**302**	**92**	**3587**	**10744**	**34.9**	**0**

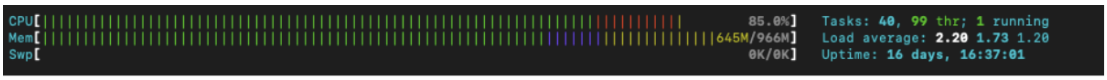
Figura 13: Rendimiento con 110 usuarios

9.12. 120 usuarios

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	713	0	140	210	3000	250	107	3348	173	5.2	0
GET	/api/final_availability	632	0	150	230	2800	252	108	3184	20	4.6	0
GET	/api/items	712	0	400	620	3100	515	289	3654	76445	6	0
GET	/api/results	631	0	130	200	2700	254	104	3354	450	5.8	0
POST	/api/sanctum/token	120	0	2000	3400	3900	2128	610	4009	57	0	0
POST	/api/test	697	0	110	190	2700	210	92	3348	25	5.8	0
GET	/api/user	723	0	150	220	2300	230	110	3181	301	5.7	0
GET	/api/users	732	0	130	200	2900	245	97	3448	2198	5	0
	Aggregated	**4960**	**0**	**150**	**480**	**3000**	**325**	**92**	**4009**	**11431**	**38.1**	**0**

Figura 14: Rendimiento con 120 usuarios

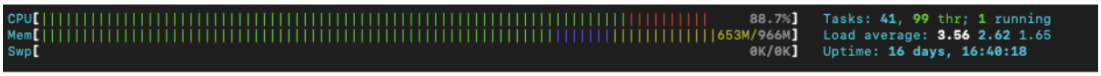
9.13. 130 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	626	0	150	470	3300	325	108	3613	173	6.4	0
GET	/api/final_availability	621	0	160	270	3100	290	108	3355	20	5.3	0
GET	/api/items	681	0	440	990	3500	635	295	4727	76445	5.1	0
GET	/api/results	666	0	140	270	2800	268	104	3620	450	7.6	0
POST	/api/sanctum/token	130	0	2000	3700	4500	2214	629	4633	57	0	0
POST	/api/test	663	0	120	280	3300	283	92	3512	25	5.9	0
GET	/api/user	594	0	160	370	3300	347	110	3701	301	5.5	0
GET	/api/users	637	0	140	330	3200	295	97	3431	2198	4.2	0
	Aggregated	**4618**	**0**	**160**	**870**	**3400**	**404**	**92**	**4727**	**11711**	**40**	**0**

Figura 15: Rendimiento con 130 usuarios

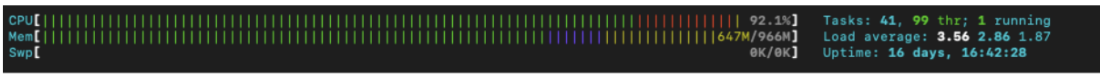
9.14. 140 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	820	0	160	270	3200	266	108	3917	173	7.4	0
GET	/api/final_availability	817	0	170	280	3000	291	108	3636	20	7.6	0
GET	/api/items	758	0	420	660	3600	556	291	3968	76445	6.4	0
GET	/api/results	752	0	150	270	3400	308	107	3908	450	4.8	0
POST	/api/sanctum/token	140	0	2200	3900	4600	2303	620	4726	57	0	0
POST	/api/test	780	0	130	250	3200	278	93	3795	25	5.1	0
GET	/api/user	781	0	170	290	3000	274	109	3607	301	6.4	0
GET	/api/users	750	0	140	260	3300	285	97	3511	2198	5.2	0
	Aggregated	**5598**	**0**	**170**	**520**	**3400**	**371**	**93**	**4726**	**10781**	**42.9**	**0**

Figura 16: Rendimiento con 140 usuarios

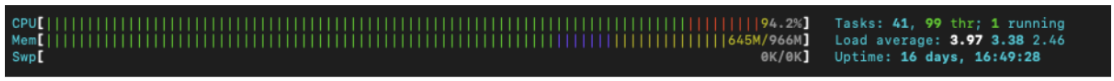
9.15. 150 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	2008	0	180	310	2900	258	108	4389	173	6.4	0
GET	/api/final_availability	2046	0	190	330	2900	263	109	4248	20	7.3	0
GET	/api/items	2074	0	440	610	3300	529	287	4692	76445	7.2	0
GET	/api/results	2032	0	160	280	3000	242	102	4381	450	8.6	0
POST	/api/sanctum/token	150	0	2400	4200	4800	2510	611	4944	57	0	0
POST	/api/test	1979	0	140	270	3000	236	91	4427	25	6.3	0
GET	/api/user	2001	0	190	310	2400	254	109	4286	301	6.4	0
GET	/api/users	1894	0	150	290	2800	230	96	4445	2198	5.4	0
	Aggregated	**14184**	**0**	**190**	**470**	**3200**	**312**	**91**	**4944**	**11610**	**47.6**	**0**

Figura 17: Rendimiento con 150 usuarios

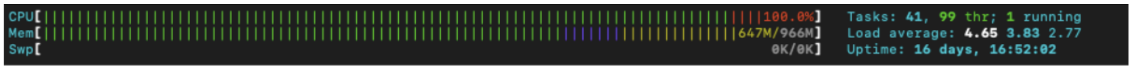
9.16. 160 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	1038	0	200	400	3300	370	107	4068	173	7.7	0
GET	/api/final_availability	1077	0	210	390	3400	342	108	3934	20	5.5	0
GET	/api/items	1027	0	520	830	4100	675	302	6293	76445	7.8	0
GET	/api/results	1047	0	180	350	3300	310	104	3989	450	7	0
POST	/api/sanctum/token	160	0	2400	4800	8900	2805	616	8921	57	0	0
POST	/api/test	1022	0	160	320	3200	295	92	4134	25	7.5	0
GET	/api/user	1079	0	210	390	3300	345	111	4079	301	6.4	0
GET	/api/users	1085	0	180	360	3400	334	98	4390	2198	6.7	0
	Aggregated	**7535**	**0**	**210**	**650**	**3700**	**432**	**92**	**8921**	**10873**	**48.6**	**0**

Figura 18: Rendimiento con 160 usuarios

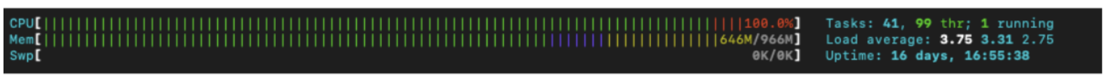
9.17. 170 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	858	0	230	420	4600	413	107	4810	173	7	0
GET	/api/final_availability	903	0	240	490	4300	436	110	4825	20	7.5	0
GET	/api/items	890	0	530	860	4400	726	292	5066	76445	7.6	0
GET	/api/results	883	0	210	470	4400	436	105	4540	450	7.1	0
POST	/api/sanctum/token	170	0	2400	5000	6600	2803	614	6614	57	0	0
POST	/api/test	899	0	190	390	4400	370	95	4728	25	7.3	0
GET	/api/user	923	0	240	460	4500	443	110	4781	301	7.1	0
GET	/api/users	829	0	210	440	4300	392	99	4679	2198	7.8	0
	Aggregated	**6355**	**0**	**250**	**830**	**4600**	**523**	**95**	**6614**	**11130**	**51.4**	**0**

Figura 19: Rendimiento con 170 usuarios

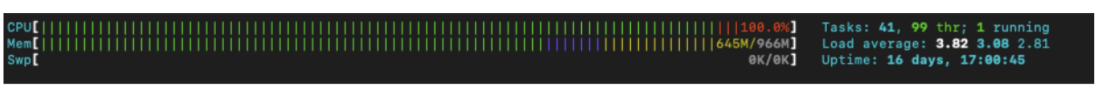
9.18. 180 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	1064	0	310	520	4300	457	111	4979	173	6.7	0
GET	/api/final_availability	1094	0	320	570	4600	510	117	5318	20	8.3	0
GET	/api/items	1080	0	600	830	4600	737	311	5464	76445	6.1	0
GET	/api/results	1043	0	300	530	4500	469	108	5181	450	6.5	0
POST	/api/sanctum/token	180	0	2700	5100	5900	2940	630	6143	57	0	0
POST	/api/test	1114	0	270	490	4500	433	96	5298	25	8.7	0
GET	/api/user	1083	0	330	610	4700	540	112	5347	301	8.7	0
GET	/api/users	1078	0	290	520	4700	466	101	5215	2198	9.6	0
	Aggregated	**7736**	**0**	**340**	**760**	**4700**	**573**	**96**	**6143**	**11113**	**54.6**	**0**

Figura 20: Rendimiento con 180 usuarios

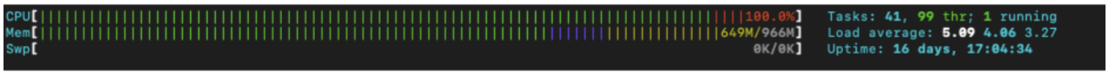
9.19. 190 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	1063	0	440	700	4500	629	112	5483	173	8.4	0
GET	/api/final_availability	1022	0	450	690	4500	640	112	5317	20	7.9	0
GET	/api/items	1060	0	720	980	5000	916	311	5808	76445	7.7	0
GET	/api/results	1039	0	440	700	4900	638	111	5555	450	8.6	0
POST	/api/sanctum/token	190	0	2900	5400	8900	3157	730	8991	57	0	0
POST	/api/test	1105	0	400	620	4600	542	96	5484	25	7.1	0
GET	/api/user	1048	0	460	700	4600	633	128	5394	301	6.3	0
GET	/api/users	1016	0	440	710	4900	678	106	5472	2198	8.9	0
	Aggregated	**7543**	**0**	**470**	**940**	**5000**	**730**	**96**	**8991**	**11175**	**54.9**	**0**

Figura 21: Rendimiento con 190 usuarios

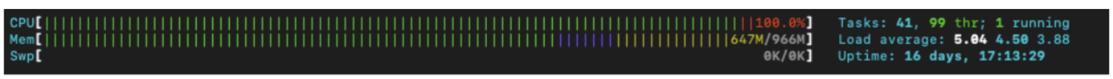
9.20. 200 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	1061	0	630	890	5700	796	122	6410	173	8.2	0
GET	/api/final_availability	1037	0	660	1000	5600	877	130	6234	20	7.4	0
GET	/api/items	1138	0	940	1300	5900	1130	341	7019	76445	8.2	0
GET	/api/results	1044	0	630	900	5700	847	106	6320	450	6.5	0
POST	/api/sanctum/token	200	0	2900	6100	7100	3385	617	7379	57	0	0
POST	/api/test	1147	0	590	860	5300	733	96	6276	25	9.2	0
GET	/api/user	1100	0	660	950	5800	866	123	6580	301	7.1	0
GET	/api/users	1073	0	630	910	5500	814	100	6507	2198	8.5	0
	Aggregated	**7800**	**0**	**680**	**1200**	**5900**	**932**	**96**	**7379**	**11590**	**55.1**	**0**

Figura 22: Rendimiento con 200 usuarios

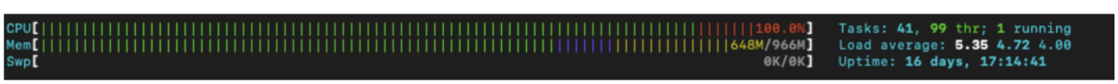
9.21. 210 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	1330	0	790	1100	5400	1009	325	6572	173	8.3	0
GET	/api/final_availability	1283	0	790	1100	5300	940	396	6405	20	7.1	0
GET	/api/items	1305	0	1100	1400	6200	1258	700	6809	76445	8.5	0
GET	/api/results	1328	0	770	1000	5800	988	220	6389	450	7.7	0
POST	/api/sa /api/results	210	0	3200	6400	7000	3501	689	7459	57	0	0
POST	/api/test	1264	0	750	980	5500	893	258	6269	25	6.6	0
GET	/api/user	1294	0	800	1100	5500	966	390	6317	301	7.4	0
GET	/api/users	1332	0	770	1000	5700	926	357	6487	2198	7.4	0
	Aggregated	**9346**	**0**	**810**	**1200**	**5900**	**1054**	**220**	**7459**	**11125**	**53**	**0**

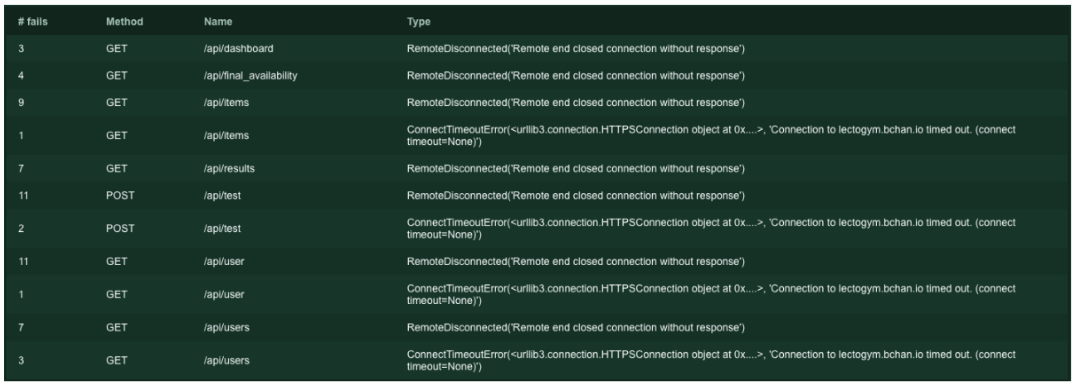
Figura 23: Rendimiento con 210 usuarios

9.22. 220 usuarios



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/dashboard	472	3	120	2500	68000	1716	109	72200	172	3.1	0
GET	/api/final_availability	455	4	130	2100	54000	1826	109	68092	20	2.4	0
GET	/api/items	465	10	420	1700	68000	2607	293	75486	74801	3	0
GET	/api/results	504	7	120	1800	52000	1548	105	67942	444	2.3	0
POST	/api/sanctum/token	199	0	2800	5600	6500	3124	615	6489	57	0	0
POST	/api/test	501	13	100	2500	68000	2811	92	75580	24	2.1	0
GET	/api/user	489	12	130	1900	65000	2228	109	75510	294	1.3	0
GET	/api/users	476	10	110	2800	68000	2556	97	75595	2152	2.4	0
	Aggregated	**3561**	**59**	**130**	**3300**	**62000**	**2238**	**92**	**75595**	**10190**	**16.6**	**0**

Figura 24: Rendimiento con 220 usuarios



# fails	Method	Name	Type
3	GET	/api/dashboard	RemoteDisconnected('Remote end closed connection without response')
4	GET	/api/final_availability	RemoteDisconnected('Remote end closed connection without response')
9	GET	/api/items	RemoteDisconnected('Remote end closed connection without response')
1	GET	/api/items	ConnectTimeoutError(<urllib3.connection.HTTPSConnection object at 0x...>, 'Connection to lectogym.bchan.io timed out. (connect timeout=None)')
7	GET	/api/results	RemoteDisconnected('Remote end closed connection without response')
11	POST	/api/test	RemoteDisconnected('Remote end closed connection without response')
2	POST	/api/test	ConnectTimeoutError(<urllib3.connection.HTTPSConnection object at 0x...>, 'Connection to lectogym.bchan.io timed out. (connect timeout=None)')
11	GET	/api/user	RemoteDisconnected('Remote end closed connection without response')
1	GET	/api/user	ConnectTimeoutError(<urllib3.connection.HTTPSConnection object at 0x...>, 'Connection to lectogym.bchan.io timed out. (connect timeout=None)')
7	GET	/api/users	RemoteDisconnected('Remote end closed connection without response')
3	GET	/api/users	ConnectTimeoutError(<urllib3.connection.HTTPSConnection object at 0x...>, 'Connection to lectogym.bchan.io timed out. (connect timeout=None)')

Figura 25: Error con 200 usuarios

9.23. Resumen de pruebas de rendimiento

Usuarios	**Request por segundo**	**%CPU**	**Promedio tiempo respuesta** (ms)
10	2.9	8.1	201
20	6.3	14.7	186
30	9.2	19.5	184
40	11.8	25.7	275
50	15.1	30.7	326
60	17.9	37.1	221
70	21.7	46.6	220
80	25.4	57.7	286
90	28.3	61.3	253
100	31.2	66	323
110	34.9	66.7	302
120	38.1	74.7	325
130	40	85	404
140	42.9	88.7	371
150	47.6	92.1	312
160	48.6	94.2	432
170	51.4	100	523
180	54.6	100	573
190	54.9	100	730
200	55.1	100	932
210	53	100	1054
220	16.6	100	2238

Cuadro 21: Resumen de pruebas de rendimiento.

9.24. Gráfica de *requests* por segundo según cantidad de usuarios

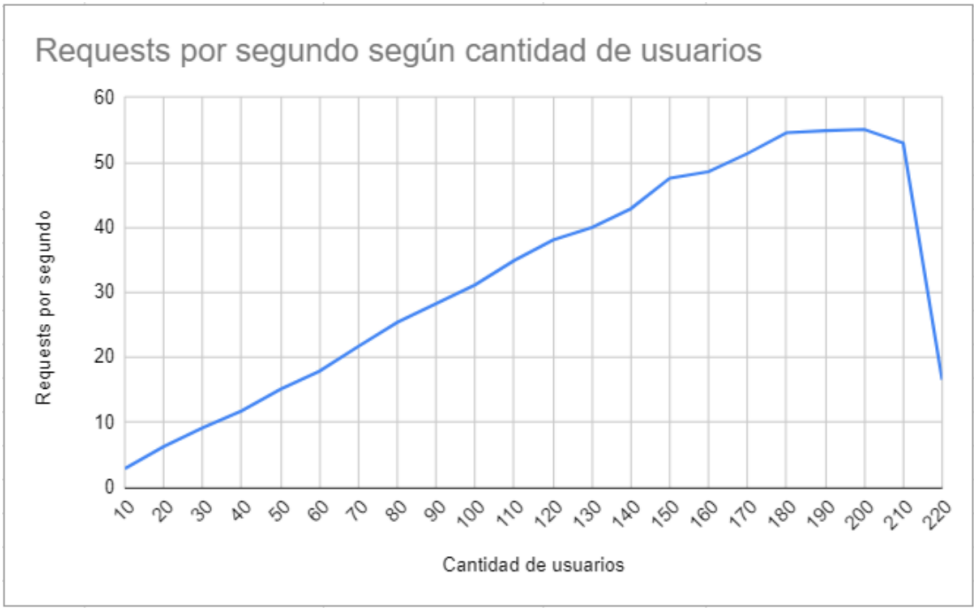


Figura 26: *Requests* por segundo según cantidad de usuarios

La cantidad de requests por segundo presenta un crecimiento lineal hasta que el sistema llega a los 180 usuarios, luego de ello se mantiene constante, hasta los 220 usuarios, donde se presenta una caída abrupta.

9.25. Gráfica de porcentaje de uso de CPU según cantidad de usuarios

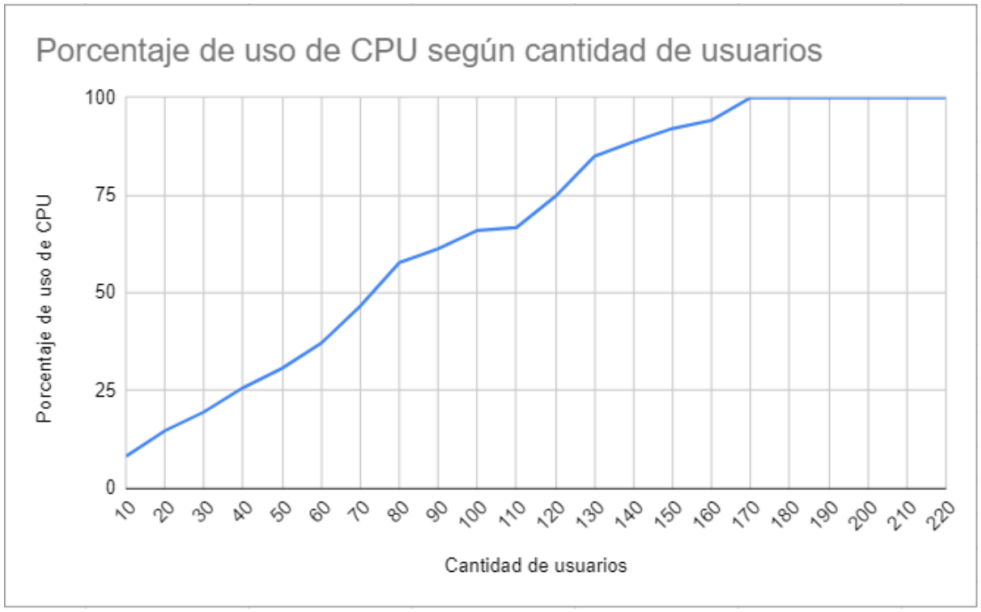


Figura 27: Porcentaje de uso de CPU según cantidad de usuarios

La CPU de la máquina virtual de igual manera presenta un crecimiento lineal, siendo a los 170 usuarios que llega a su límite utilizando el 100 del CPU.

9.26. Gráfica de promedio de tiempos de respuesta según cantidad de usuarios

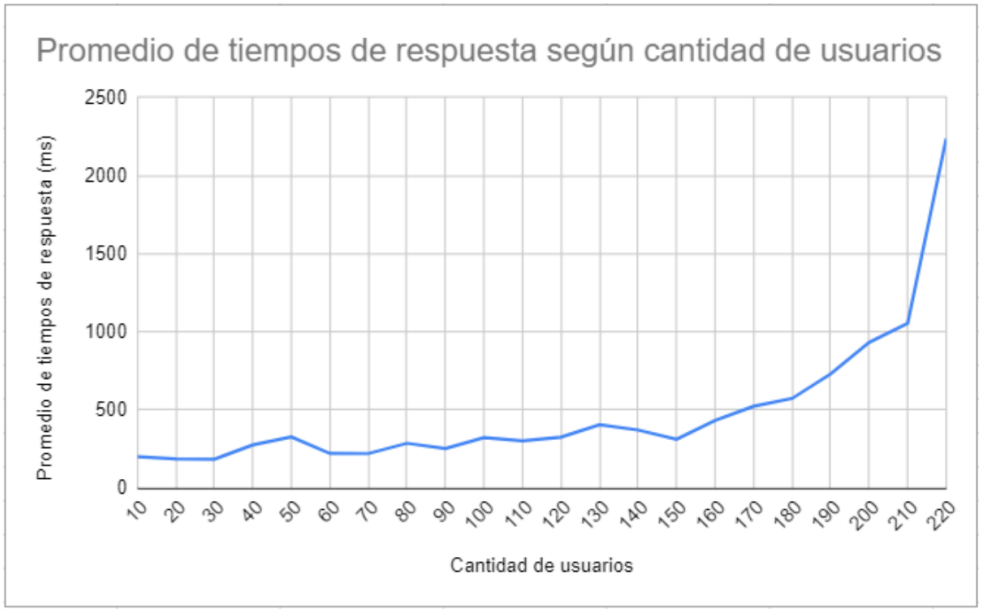


Figura 28: Promedio de tiempos de respuesta según cantidad de usuarios

El promedio de tiempos de respuesta se presenta de manera constante por debajo de los 500 ms, con un aumento muy leve, luego de los 160 usuarios activos la creciente de tiempos de respuesta empieza a ser bastante más notoria, llegando hasta los 2200 ms con 220 usuarios, momento en el cual se empezaron a presentar las primeras fallas.

Por lo cual se establece que con la configuración actual del sistema, siendo alojado en la instancia t2micro de aws, la máxima capacidad de usuarios simultáneos de manera óptima es de 150 realizando aproximadamente 45 solicitudes por segundo entre todos, siendo un punto en el que los tiempos de respuesta son menores a los 500 ms y la máquina virtual aún se mantiene con alrededor del 90 % del CPU en uso, sin llegar a su máximo.

9.27. Costos del sistema y escalabilidad

Actualmente el API se encuentra alojado en una máquina virtual de tipo t2micro, la cual tiene un costo de 0.0116 dólares por hora, siendo 8.47 dólares por mes, se tiene como propuesta para la escalabilidad del proyecto, implementar un tipo de escalabilidad ágil, siendo más específicos una escalabilidad horizontal, AWS cuenta ya cuenta con la facilidad de implementar esta tecnología, en la cual según la carga que presenta el CPU se pueden crear o disminuir nuevas instancias. La manera en la que esto funciona es estableciendo parámetros de uso de CPU, por ejemplo, se establece que al llegar al 90 % se crea una nueva instancia, de esta manera disminuyendo el uso en 45 % cada una, de igual manera en caso contrario, si se llega a 20 % disminuir las instancias. De esta manera el cobro se realizará según la cantidad de instancias que se activen por hora.

CAPÍTULO 10

Seguridad del sistema

10.1. Implementación de SSL

Para la implementación del protocolo SSL se ha utilizado la herramienta de Certbot, para ello es necesario tener un dominio debido a que los certificados de SSL únicamente pueden ser generados utilizando un dominio y no con una IP.

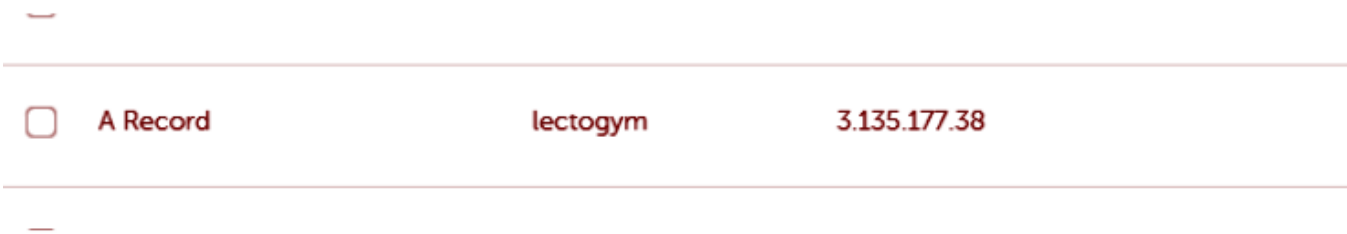


Figura 29: IP de servidor asignado a dominio lectogym.bchan.io en namecheap.com

10.2. Control de vulnerabilidaades de dependencias

Para el control de vulnerabilidades se ha utilizado la herramienta implementada en GitHub llamada Dependabot, éste se encarga de tomar los archivos "composer.lock" y "composer.json" donde se encuentran enlistadas todas las dependencias del proyecto y se realiza el análisis para detectar cualquier anomalía relacionada a la versión de alguna de las dependencias.

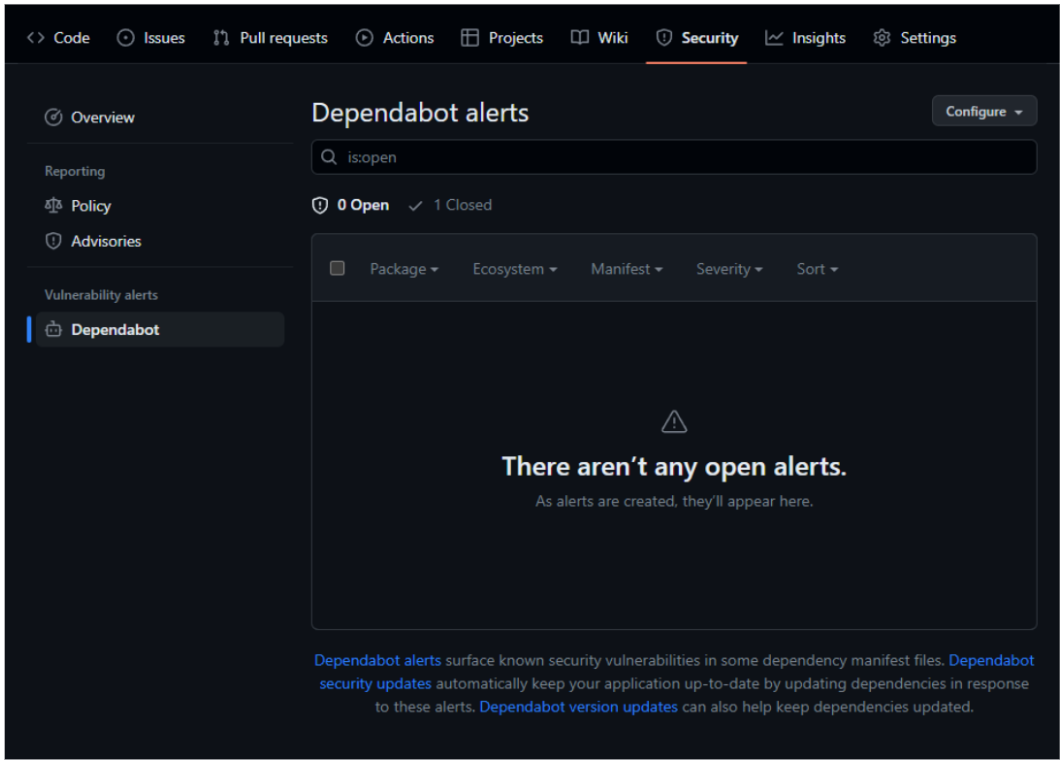


Figura 30: Panel de Dependabot

En la figura se muestra el panel de Dependabot al momento de existir alguna vulnerabilidad relacionada con una de las dependencias se mostrará en el panel, indicando el nivel de severidad, siendo los niveles *low*, *moderate*, *high* y *critical*. Dependabot se encuentra activa todo el tiempo, al momento de encontrar una vulnerabilidad se mostrará dentro del panel y enviará una notificación por medio de correo electrónico.

10.3. Protección ante SQL-injection

El propio framework utilizado posee distintas funcionalidades que se encargan de fortalecer la seguridad de la aplicación, entre ellas las más importantes se encuentran las siguientes:

Se realiza una configuración del límite de solicitudes que puede realizar un usuario o una ip en específico.

```php
/**
 * Configure the rate limiters for the application.
 *
 * @return void
 */
protected function configureRateLimiting()
{
    RateLimiter::for('api', function (Request $request) {
        return Limit::perMinute(1000)->by($request->user()?->id ?: $request->ip());
    });
}
```

Figura 31: Configuración de límite de solicitudes por usuario o por IP

De igual manera, las contraseñas de los usuarios se encuentran encriptadas, de esta manera es imposible acceder a los datos de los usuarios ni siquiera teniendo acceso a la base de datos.

```php
$user = User::create([
    'name' => $request->name,
    'email' => $request->email,
    'password' => Hash::make($request->password),
]);
```

Figura 32: Encriptación de datos dentro de la base de datos

CAPÍTULO 11

Conclusiones

- Un buen diseño de base de datos ha sido fundamental para poder abstraer la manera en la que se almacenan los datos y construir una arquitectura eficiente y segura, donde se tengan fuentes únicas de fidelidad, logrando tiempos de respuesta de las solicitudes en promedio menores a los 500ms.
- Las pruebas de diagnóstico realizados a los sujetos de prueba presentaron en los usuarios una mejora en su comprensión lectora en promedio de un 27.66 % y una mejora en su velocidad de lectura en promedio de un 10.35 %.
- Se implementaron medidas de seguridad dentro de la instancia de AWS para garantizar solicitudes seguras a la API. Esto incluye configuraciones de seguridad como el uso de protocolos SSL, políticas de acceso, certificados, entre otras prácticas de seguridad para proteger la integridad de los datos y la comunicación. Con los recursos que posee el servidor montado se establece un límite de 150 usuarios activos simultáneamente para un funcionamiento eficiente.
- Bajo las especificaciones actuales del servidor, el mantenerlo activo presenta un costo de 0.0116 dólares por hora, 8.47 dólares por mes. Planteando como posibilidad una escalibilidad horizontal.

CAPÍTULO 12

Recomendaciones

- Previo a implementar nuevas funcionalidades al proyecto debemos dedicarle el suficiente tiempo al diseño de la arquitectura, de esta manera nos aseguramos de optimizar los datos que almacenaremos para no tener problemas en los que la misma información se encuentra representada en dos lugares a la vez y luego tener que cambiar la arquitectura varias veces.
- Al momento de trabajar en el API es importante poder alojarla lo antes posible, de esta manera el equipo de desarrollo que se encuentre trabajando en el Front-end puede tener acceso a ella en todo momento e indicar si los request se encuentran trabajando de la manera que se desea y realizar cambios si así son requeridos.
- Ya con el API montada es importante conectar la base de datos del servidor a nuestra herramienta de visualización de bases de datos que utilicemos, DBeaver en el caso de este proyecto, de esta manera podemos visualizar en tiempo real todas las tablas y datos de la base de datos para llevar un monitoreo de la base de datos en tiempo real.
- La implementación de herramientas de seguridad como Dependabot también es de gran importancia que puedan ser implementadas lo antes posible, ya que esto se encarga de analizar si alguna de las librerías que se están utilizando poseen alguna vulnerabilidad, de ser así poder cambiarlo, si esto se implementa demasiado tarde, corremos con el riesgo de tener una vulnerabilidad en una librería que conlleva demasiado trabajo sustituirla.
- La base de datos posee bastante información por defecto para que pueda funcionar, debemos procurar tener una copia de la base de datos o un seeder en cuestión para poder recuperar la base de datos de manera fácil en caso de cualquier incoveniente.

CAPÍTULO 13

Bibliografía

Almutairi, Nouf Rashdan. (2018). *Effective Reading Strategies for Increasing the Reading Comprehension Level of Third-Grade Students with Learning Disabilities.* Western Michigan University.

Andina, Zully. (2006). *Importancia de la comprensión y velocidad de lectura en el rendimiento en los cursos de literatura e idioma español.* Universidad de San Carlos Guatemala.

American Speech-Language-Hearing Association. (2017). *Disorders of reading and writing.* Extraido de: https://www.asha.org/Practice-Portal/Clinical-Topics/Written-Language-Disorders/Disorders-of-Reading-and-Writing/

Bilaya, A. (2021). Speed Reading as a Psychological Problem. Extraído de https://www.e3s-conferences.org/articles/e3sconf/pdf/2021/34/e3sconf_uesf2021_07062.pdf

Diccionario Real Academia Española (actualización 2021). *Diccionario de la lengua española, Edición del Tricentenario en línea.* Recuperado el 12 de mayo de 2023 desde https://dle.rae.es/analfabetismo

Digicert (2023). *WHAT IS SSL, TLS & HTTPS?* Recuperado el 30 de julio de 2023 de https://www.digicert.com/what-is-ssl-tls-and-https

Dyson, Mary Clare y Haselgrove, Mark. (2020). *The effects of reading speed and reading patterns on the understanding of text read from screen. Journal of Research in Reading.* Extraído de https://www.researchgate.net/publication/227760582_The_effects_of_reading_-speed_and_reading_patterns_on_the_understanding_of_text_read_from_screen

Echout, Mohamed. (2022). *What Is Locust Load Testing?* Recuperado el 29 de julio de 2023 de https://www.blazemeter.com/blog/locust-load-testing

El analfabetismo.Plan educativo Universidad Nacional Autónoma de México (UNAM). Ciudad de México. http://www.planeducativonacional.unam.mx/CAP_00/Text/00_08a.html

Esquivel, 2018. Revista Científica del SEP Vol. 1 Año 2018, pp. 81-91. José Adrián Esquivel Sarceño, licenciado en Pedagogía y Ciencias de la Educación y Magíster en Desarrollo Rural.

García, Eduardo, *et al.* (2015). *Problemas de comprensión en el alumnado de Educación Primaria y Educación Secundaria Obligatoria: un estudio de prevalencia en español.* European Journal of investigation in health, psychology and education, 3(2), 113-123.Recuperado de https://bit.ly/2OZutPH

Gillis, Alexander y Bigelow, Stephen. (2020). *REST API (RESTful API).* Recuperado el 18 de mayo de 2023 de https://www.techtarget.com/searchapparchitecture/definition/RESTful-API

Gillis, Alexander y Bigelow, Stephen. (2023). *REST API (RESTful API).* Recuperado el 18 de mayo de 2023 de https://www.techtarget.com/searchapparchitecture/definition/performance-testing

Github (2023). *About Dependabot version updates.* Recuperado el 30 de julio de 2023 de https://docs.github.com/en/code-security/dependabot/dependabot-version-updates/about-dependabot-version-updates

Globalsign (2020). *What is Secure Sockets Layer?* Recuperado el 30 de julio de 2023 de https://www.globalsign.com/en/ssl-information-center/what-is-ssl

Hasbrouck, Jan. & Tindal, Gerald. (2017). *An update to compiled ORF norms (Technical Report No. 1702).* Eugene, OR, Behavioral Research and Teaching, Universidad de Oregon.

IBM, (2021). *Métodos PUT, POST Y DELETE.* Recuperado el 12 de agosto de 2023 de https://www.ibm.com/docs/es/mam/7.6.1?topic=api-put-post-delete-methods

Kinsta. (2022). *What is GitHub? An introduction to Github.* Recuperado el 11 de agosto de 2023 de https://kinsta.com/knowledgebase/what-is-github/

McNamara, D., *et al.* (2007). *istart: A web-based tutor that teaches self-explanation and metacognitive reading strategies.* En D.S. McNamara (Ed.), Reading comprehension strategies: Theories, interventions, and technologies (pp. 397-420). Erlbaum.

mdn, (2023). *HTTP response status codes.* Recuperado el 14 de agosto de 2023 de https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Merina, Elena. (2021). *La Importancia de la Comprensión Lectora en los Niños. El Método Montessori.* Recuperado el 12 de diciembre de 2022 de https://elmetodomontessori.com/la-importancia-de-la-comprension-lectora-en-los-ninos/

Monroy, José y Gómez, Blanca. (2009). *Comprensión lectora.* Revista Mexicana de Orientación Educativa, 6(16), 37-42. Universidad Autónoma de Tlaxcala. Recuperado el 13 de mayo de 2023, de http://pepsic.bvsalud.org/scielo.php?script=sci_arttext&pid=S1665-75272009000100008&lng=pt&tlng=es.

Nzubechukwu, Clement. (2023). *What is a hosted API?* Recuperado el 5 de agosto de 2023 de https://www.quora.com/What-is-a-hosted-API

OCDE/ Instituto Nacional de Calidad y Evaluación (INCE). (2000). *Proyecto PISA. La medida de los conocimientos y destrezas de los alumnos: Un nuevo marco para la evaluación.* Madrid.

Oracle (2022). *¿Qué es una base de datos?.* Recuperado el 18 de mayo de 2023 de https://www.oracle.com/mx/database/what-is-database/

Pérez, Cesar. y Dominguez, Andrea. (2022). *Solo 17 % de graduandos obtiene buen desempeño en Matemáticas y 32 % en Lectura, según evaluación de Educación.* Recuperado el 2 de junio de 2023 de https://www.prensalibre.com/guatemala/comunitario/solo-17-de-graduandos-obtiene-buen-desempeno-en-matematicas-y-32-en-lectura-segun-evaluacion-de-educacion/

Pérez, Elena. (2014). *Comprensión lectora VS Competencia lectora: qué son y qué relación existe entre ellas.* Investigaciones Sobre Lectura, (1), 65-74.

Peronard, Marianne, *et al.* (1998). *Comprensión de textos escritos: de la teoría a la sala de clases (1. ed.).* Santiago de Chile: Andrés Bello

PIRLS | IEA.nl. (2019). *Progress in International Reading Literacy Study (PIRLS).* Recuperado 22 de marzo de 2023, de https://www.iea.nl/studies/iea/pirls

Progrentis (2022). *¿Qué es Progrentis? Función y beneficios.* Recuperado el 12 de mayo de 2023, de https://soporteapps.progrentis.com/support/solutions/articles/12000025665–qué-es-progrentis-

Rayner, Keith., *et al.* (2016). *So Much to Read, So Little Time: How Do We Read, and Can Speed Reading Help?* Psychological Science in the Public Interest, 17(1), 4–34. https://doi.org/10.1177/1529100615623267

Real Academia Española y Asociación de Academias de la Lengua Española. «dislexia». Diccionario de la lengua española (23.ª edición).

Schwaber, Ken., & Sutherland, Jeff. (2017). *The Scrum Guide.* Recuperado el 28 de abril de 2024, de Scrum.Org.

Shivan. (s. f.). *What Is an Instance In Cloud Computing? – A Thorough Guide.* Recuperado el 18 de mayo de 2023 de https://www.scaleyourapp.com/what-is-an-instance-in-cloud-computing-a-thorough-guide/

Sutherland, Jeff. (2014). Scrum: The Art of Doing Twice the Work in Half the Time. Currency.

Soto, Christian., *et al.* (2019). Impact of bridging strategy and feeling of knowing judgments on reading comprehension using comprende: An educational technology. TechTrends, 63(5), 570-582. doi: http://dx.doi.org/10.1007/s11528-019-00383-5.

Suárez, Angel., Moreno, Juan Manuel, y Godoy, Maria Jose (2010). Vocabulario y comprensión lectora: Algo más que causa y efecto. Álabe: Revista de Investigación sobre Lectura y Escritura, 1, 0-10.

Uría, Lucia. (2005). El analfabetismo en América Latina con especial atención a la situación boliviana. Tesis presentada en el Departamento de Idiomas. Universidad de Ciencias

Aplicadas de Colonia. Colonia (Alemania), pp. 4-10.

Velásquez, Marisol.; Cornejo, Carolina. y Roco, Angel. (2008), Evaluación de la competencia lectora en estudiantes de primer año de carreras del área humanísticas y carreras del área de la salud en tres universidades del consejo de rectores 34(1), 123-138.

Repositorio Github con el proyecto: https://github.com/mschangtaitai/FinalProject.git