UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

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

Trabajo de graduación presentado por Luis Alejandro Urbina Hernández para optar al grado académico de Licenciado en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Guatemala,

2023

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

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

Trabajo de graduación presentado por Luis Alejandro Urbina Hernández para optar al grado académico de Licenciado en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Guatemala,

2023

Vo.Bo.:

(f) _____

Ing. Diana María Ortiz Naranjo

Tribunal Examinador:

(f) _____

Ing. Diana María Ortiz Naranjo

(f) _____

Ing. Angelica Gabriela Rocha Guevara

(f) _____

MSc. Douglas Leonel Barrios Gonzalez

Fecha de aprobación: Guatemala, 23 de mayo de 2023.

Prefacio

Agradezco a Dios por todo lo que me ha dado, a mis padres y a mi abuela por su apoyo y amor, a todos los miembros de la fundación Juan Bautista Gutiérrez por proveerme la oportunidad de estudiar mi carrera, a mi asesora Diana Ortiz por su ayuda y guianza en este trabajo, a Christian Porras por los recursos provistos para desarrollar este proyecto, y a todos los que en algún momento creyeron en mí.

Índice

Lista de figuras

Lista de cuadros

Resumen

En la población estudiantil de Guatemala, tanto la velocidad de lectura, como el nivel de comprensión lectora se encuentran subdesarrollados. Este problema representa un obstáculo significativo para el desarrollo académico y personal de los estudiantes, así como para la competitividad y el desarrollo del país. Por lo tanto, se propuso una solución a este problema a través del desarrollo de una aplicación web interactiva que permita a diez estudiantes de nivel medio en Guatemala mejorar su velocidad de lectura y nivel de comprensión lectora mediante retos interactivos. Los resultados de la implementación de la aplicación web demostraron una mejora significativa tanto en la velocidad de lectura como en el nivel de comprensión lectora de los estudiantes. En promedio, los estudiantes mejoraron su nivel de comprensión lectora en un 24.85 % y aumentaron su velocidad de lectura, o palabras leídas por minuto, en un 10.35 %. Basado en estos resultados, se concluyó que los ejercicios interactivos, en conjunto con la aplicación web desarrollada, fueron efectivos para mejorar la velocidad de lectura y el nivel de comprensión lectora en estudiantes de nivel medio en Guatemala. Para obtener mejores resultados, se recomienda: incrementar el tamaño de la muestra de estudiantes utilizada, desarrollar más retos interactivos, dar seguimiento por más tiempo a los estudiantes, incorporar un tutor en la aplicación, e implementar nuevas funcionalidades a la aplicación mediante el desarrollo de una aplicación móvil o una aplicación web progresiva.

Abstract

In the student population of Guatemala, both reading speed and reading comprehension levels are underdeveloped. This problem represents a significant obstacle to the academic and personal development of students, as well as to the competitiveness and development of the country. Therefore, a solution to this problem was proposed through the development of an interactive web application that allows ten middle school students in Guatemala to improve their reading speed and comprehension level through interactive challenges. The results of the implementation of the web application demonstrated a significant improvement in both the reading speed and comprehension level of the students. On average, students improved their reading comprehension level by 24.85 % and increased their reading speed, or words read per minute, by 10.35 %. Based on these results, it was concluded that the interactive exercises, in conjunction with the developed web application, were effective in improving the reading speed and comprehension level of middle school students in Guatemala. For better results, it is recommended to increase the size of the student sample used, develop more interactive challenges, follow up with students for a longer period, incorporate a tutor in the application, and implement new functionalities to the application through the development of a mobile application or progressive web application.

CAPÍTULO 1

Introducción

Dos de las habilidades cognitivas más importantes que los estudiantes pueden desarrollar son la rapidez de lectura y la comprensión lectora. La rapidez de lectura es la cantidad de palabras que una persona puede leer por minuto durante una lectura natural de un texto sin comprometer su nivel de comprensión del mismo (Rayner et al., 2016). La comprensión lectora es la capacidad que tiene una persona para extraer y construir significado a través de la lectura de un texto (Almutairi, 2018).

Estas habilidades son importantes, puesto que ser competente en ellas resulta en una ventaja significativa en el ámbito académico. Una rapidez de lectura avanzada está relacionada con un alto nivel de comprensión lectora (Langer, 1991), un mejor desempeño académico (Carver, 1990), y una mejor retención de información (Wright y Gronlund, 1984). Un alto nivel de comprensión lectora resulta en un mejor razonamiento y retención de información (Miller y Keenan, 2016). Además, está ligado con un mejor desempeño académico (Centro Nacional de Estadísticas Educativas, 2017), así como una alta probabilidad de desempeñarse en la universidad con éxito (Carlson y O'Brien, 2017).

En cuanto a la población estudiantil de Guatemala, estas dos habilidades se encuentran subdesarrolladas. Primero, los estudiantes de nivel medio presentan dificultades de comprensión y rapidez de lectura a nivel nacional. Segundo, según los resultados de la prueba de lectura de 2019, solo el 37.03 % de los estudiantes han desarrollado habilidades de lectura esperadas al concluir el bachillerato. Finalmente, Guatemala es uno de los países con mayor tasa de analfabetismo (18.5 %) a nivel latinoamericano.

Como una posible solución al subdesarrollo de estas habilidades cognitivas en Guatemala, se desarrolló el proyecto titulado LectoGym, que consiste en una aplicación web compuesta por tres elementos: *backend*, API, y *frontend*, siendo el último el enfoque del presente documento. El proyecto tiene como objetivo incrementar la cantidad de palabras leídas por minuto y el nivel de comprensión lectora inicial de diez estudiantes de nivel medio en Guatemala mediante retos interactivos en la ya mencionada aplicación web.

Esta aplicación presenta a los usuarios una serie de retos interactivos que les ayudan a

mejorar su rapidez de lectura y comprensión lectora progresivamente. Además, presenta a los usuarios pruebas de diagnóstico inicial y final para medir el estado de estas dos habilidades antes y después de realizar los retos. Finalmente, se muestran las métricas de calificación de las pruebas de diagnóstico: tiempo de lectura, porcentaje de preguntas contestadas correctamente, y palabras leídas por minuto. Aunado a estos resultados, se calcula y muestra el porcentaje de mejora (si existe) entre el porcentaje de preguntas contestadas correctamente y las palabras leídas por minuto previo a iniciar el programa y al finalizarlo. Los ejercicios interactivos y la aplicación web desarrollada resultaron ser efectivos en mejorar la rapidez de lectura y el nivel de comprensión lectora de los 10 estudiantes de prueba.

CAPÍTULO 2

Antecedentes

Se han desarrollado algunas alternativas que tienen como fin ayudar a las personas a mejorar sus capacidades de rapidez y comprensión lectora. Algunas de estas alternativas son:

- Progrentis: es una plataforma online que permite a niños de entre cinco y dieciséis años mejorar sus habilidades cognitivas mediante ejercicios y juegos interactivos. Los ejercicios de la plataforma están basados en principios de neurociencia y están diseñados de manera que los niños los encuentren entretenidos y divertidos. Los ejercicios se adaptan al nivel y progreso de cada niño, y la dificultad de estos aumenta a medida que el niño mejora (Progrentis, 2021).
- Innova Reading: es una plataforma web que permite a sus usuarios, que pueden ser de cualquier edad, leer cinco veces más rápido e incrementar su atención, comprensión, y retención de información. La plataforma utiliza ejercicios neurovisuales combinados con tutorías psicopedagógicas en clases pregrabadas para cumplir este fin (Innova Reading, 2021).
- Curso de lectura rápida: ofrecido por la Universidad Rafael Landívar de Guatemala, es un curso virtual que promete mejorar la rapidez de lectura, comprensión lectora, e interpretación de la información de los estudiantes. Este es un curso virtual, con una duración de veinte horas, que incluye recursos virtuales, videoconferencias, y evaluaciones para garantizar la mejora que ofrece a los alumnos (Universidad Rafael Landívar [URL], 2019).

CAPÍTULO 3

Justificación

Para un estudiante, especialmente en el nivel medio, ser capaz de leer a altas velocidades y comprender textos extensos, resultan ser habilidades indispensables para desarrollarse de forma correcta en el ámbito académico (Willingham, 2006).

Por una parte, ser capaz de leer rápidamente permite a los estudiantes procesar información de mejor forma, cubrir material eficazmente, y ser más productivo. Una rapidez de lectura avanzada está relacionada con un alto nivel de comprensión lectora (Langer, 1991), un mejor desempeño académico (Carver, 1990), y una mejor retención de información (Wright y Gronlund, 1984).

Por otra parte, un alto nivel de comprensión lectora involucra habilidades de captura, análisis e interpretación de información escrita. Un alto nivel de comprensión lectora está correlacionado con un mejor razonamiento y retención de información (Miller y Keenan, 2016). Además, está ligado con un mejor desempeño académico (Centro Nacional de Estadísticas Educativas, 2017), así como una alta probabilidad de desempeñarse en la universidad con éxito (Carlson y O'Brien, 2017).

Los estudiantes en Guatemala, independientemente del nivel académico, evidencian dificultades de comprensión y rapidez de lectura (Cien, 2019). Según la página de la Dirección General de Evaluación e Investigación Educativa (Digeduca) del Ministerio de Educación, en la prueba de lectura de 2019, el logro en Lectura satisfactorio por parte de estudiantes de diversificado fue de 37.03 % (Digeduca, 2019). Lo que significa que, de cada cien estudiantes a punto de graduarse, sólo treinta y siete han desarrollado las habilidades esperadas en lectura al concluir la carrera. Resultados como estos son ocasionados por las múltiples fallas en el sistema educativo guatemalteco, entre ellas: ausencia de una cultura de lectura inculcada por profesores, altos niveles de analfabetismo, carencia de escuelas, y falta de programas que motiven la lectura.

Una velocidad de lectura deficiente en conjunto con un bajo nivel de comprensión lectora resulta en problemas serios para los estudiantes de Guatemala (Gonzalez, 2016). Entre estos problemas se pueden listar: analfabetismo parcial o total, falta de memoria, pobre

vocabulario, interpretación incorrecta de tareas, inseguridades, baja autoestima, entre otros (Secretaría de Planificación y Programación de la Presidencia de Guatemala [SEGEPLAN], 2015). Partiendo de estos problemas, surge la necesidad de mejorar estas dos habilidades imprescindibles para el desarrollo académico de los estudiantes de nivel medio en Guatemala (Gonzalez, 2016).

El proyecto en cuestión busca ayudar a diez estudiantes de nivel medio a mejorar su capacidad de lectura y su nivel de comprensión lectora. La importancia del proyecto se puede ver reflejada en los beneficios que ocasiona el desarrollo de estas dos habilidades, entre ellos: mejor capacidad de retención de información, razonamiento, y resolución de problemas, alta capacidad de comprensión lectora, una mejor probabilidad de éxito en la universidad y, en general, un mejor desempeño académico.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Incrementar la cantidad de palabras leídas por minuto y el nivel de comprensión lectora inicial de 10 estudiantes de nivel medio en Guatemala mediante retos interactivos en una aplicación web.

4.2. Objetivos específicos

- Desarrollar retos interactivos que permitan a estudiantes mejorar sus capacidades de lectura y comprensión lectora.
- Diseñar e implementar el *front-end* de una aplicación web que presente a los usuarios tareas interactivas que les permitan mejorar su rapidez y comprensión lectora.
- Generar pruebas unitarias para la aplicación desarrollada que garanticen el correcto funcionamiento de los componentes utilizados en la misma.

CAPÍTULO 5

Marco teórico

5.1. Rapidez de lectura

La rapidez de lectura 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). La rapidez de lectura es una métrica utilizada para la lectura de un texto completo, no del resultado de hacer *skimming* (leer los títulos párrafos claves de un texto) o *scanning* (escanear un texto para encontrar una palabra específica) del mismo.

La rapidez de lectura se mide en palabras leídas por minuto, el promedio para estudiantes de nivel medio se encuentra entre 200 y 300 palabras por minuto (Hasbrouck y Tindal, 2017). Se considera que un individuo cuenta con una rapidez de lectura avanzada si es capaz de leer más de esa cantidad de palabras en un minuto.

El poseer una rapidez de lectura avanzada es una ventaja considerable para estudiantes de todos los niveles académicos, incluyendo a estudiantes de nivel medio. Una rapidez de lectura avanzada está relacionada con un alto nivel de comprensión lectora (Langer, 1991), una mejor retención de información (Wright y Gronlund, 1984) y un mejor desempeño académico (Carver, 1990).

5.2. Comprensión lectora

La comprensión lectora se puede definir como la capacidad que tiene una persona para extraer y construir significado a través de la lectura de un texto (Almutairi, 2018).

Esta habilidad está desarrollada de manera distinta en cada persona. Lo que resulta en la existencia de múltiples niveles de comprensión lectora. La forma estándar de medir qué tan buena comprensión lectora tiene un individuo es mediante pruebas de comprensión lectora que contienen preguntas relacionadas al tema de la lectura. Así, el nivel de comprensión

lectora se determina mediante el cálculo del porcentaje de preguntas correctas respecto a las preguntas totales de la prueba (Dyson y Haselgrove, 2001).

Al igual que con la rapidez de lectura, un nivel alto de comprensión de lectura resulta ser beneficioso para estudiantes de cualquier nivel académico. Un alto nivel de comprensión lectora está correlacionado con un mejor razonamiento y retención de información (Miller y Keenan, 2016). Además, está ligado con un mejor desempeño académico (Centro Nacional de Estadísticas Educativas, 2017), así como una alta probabilidad de desempeñarse en la universidad con éxito (Carlson y O'Brien, 2017).

5.3. Problemas relacionados con una rapidez de lectura deficiente

El impacto de contar con una rapidez de lectura deficiente va mucho más allá de no ser capaz de leer rápidamente. Estudios han demostrado que esta condición está asociada con un mayor riesgo de abandono escolar, intentos de suicidio (Daniel et al., 2006), encarcelación (Christie y Yell, 2008), ansiedad, depresión (Carroll et al., 2005), y baja autoestima (McArthur et al., 2016).

Además de las consecuencias ya mencionadas, otros estudios demuestran que una rapidez de lectura deficiente en la niñez desemboca en un pobre rendimiento académico, delincuencia juvenil y comportamiento antisocial, dificultades psicosociales en la adultez, desempleo por incompetencia y/o un estatus ocupacional más bajo (Smart et al., 2017).

Finalmente, un estudio realizado por el Centro de Investigación de Discapacidades de Aprendizaje de Colorado se encontró que además de dificultades de lectura, los individuos con rapidez de lectura deficiente exhiben mayores tasas de dificultades académicas, depresión y trastorno de conducta de inicio en la adolescencia (Willcutt et al., 2007).

5.4. Problemas relacionados con una comprensión lectora deficiente

Así como un nivel deficiente de rapidez de lectura causa problemas para las personas, una comprensión lectora deficiente ocasiona ciertos problemas. La comprensión lectora de una persona se ve afectada por diversos factores, entre ellos nivel de estudios de los padres, zona socioeconómica, zona geográfica, entre otros (Pérez, 2014).

Personas con un nivel de comprensión lectora bajo suelen presentar dificultades como 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 en que la persona no desarrolle un juicio crítico.

Aunado a esto, la baja comprensión lectora tiene como consecuencias la poca comprensión y aprendizaje de los temas impartidos dentro de las clases, tomando en cuenta que la

lectura está directamente relacionada con la escritura, la deficiencia de ambos desemboca en un bajo rendimiento estudiantil y alta dificultad para adaptarse a un adecuado aprendizaje en cualquier nivel educativo (Suárez, et al., 2010).

5.5. Rapidez de lectura y comprensión lectora en Guatemala

En Guatemala, la población estudiantil, independientemente del nivel académico, evidencia dificultades de comprensión lectora y una rapidez de lectura deficiente (Cien, 2019).

Datos de la Dirección General de Evaluación e Investigación Educativa respecto a las pruebas nacionales de lectura y matemática realizadas por el Ministerio de Educación, muestran que el porcentaje de estudiantes que alcanzan los niveles de desempeño satisfactorio y excelente (nivel de logro) en estas pruebas ha empeorado y es bajo. Esto puede considerarse una causa para el estado decadente de la rapidez y comprensión lectora de la población estudiantil guatemalteca.

5.5.1. Primaria

Las evaluaciones de primaria realizadas por el MINEDUC se hicieron desde 2008 hasta 2014. En ellas, se evaluó tercero y sexto primaria en las materias de lectura y matemáticas. Los resultados muestran que entre 2008 y 2014, el nivel de logro en lectura para los alumnos de tercero primaria se mantuvo cerca del 50 %. Los resultados también muestran que el nivel de logro en lectura para los alumnos de sexto primaria pasó de ser 35.3 % en 2008 a ser 40.4 % en 2014, aumentando poco más de cinco puntos porcentuales (Cien, 2019).

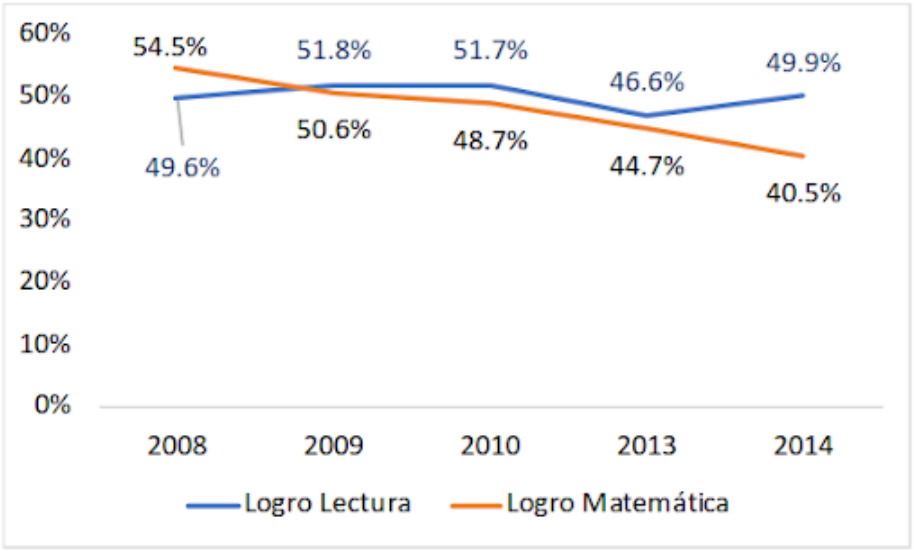


Figura 1: Niveles de logro en lectura y matemática, alumnos de tercero primaria, 2008 a 2014

(Cien, 2019)

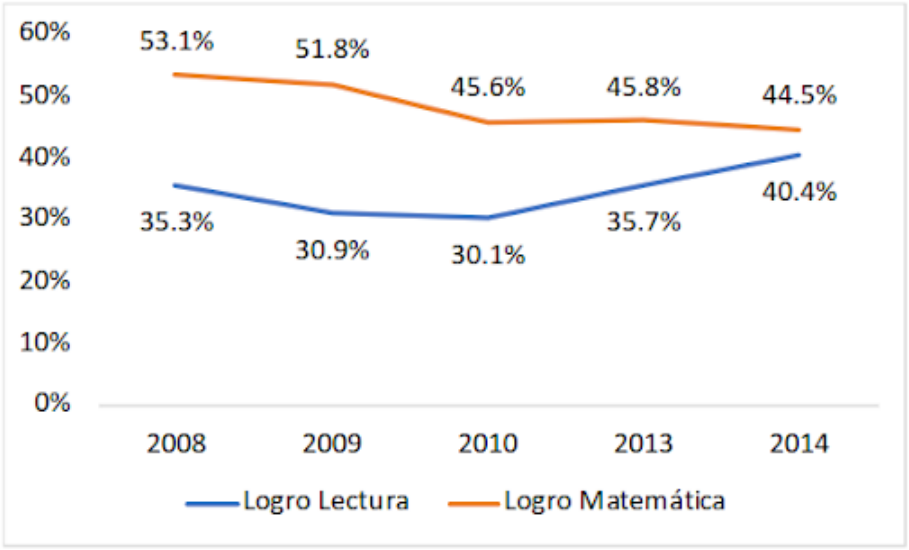


Figura 2: Niveles de logro en lectura y matemática, alumnos de sexto primaria, 2008 a 2014

(Cien, 2019)

5.5.2. Tercero básico

Las evaluaciones de tercero básico realizadas por el MINEDUC se hicieron en 2006, 2009, y 2013. Los resultados muestran que el logro en lectura se redujo 12 puntos porcentuales, pasando de 27 % en 2006 a 15 % en 2013.

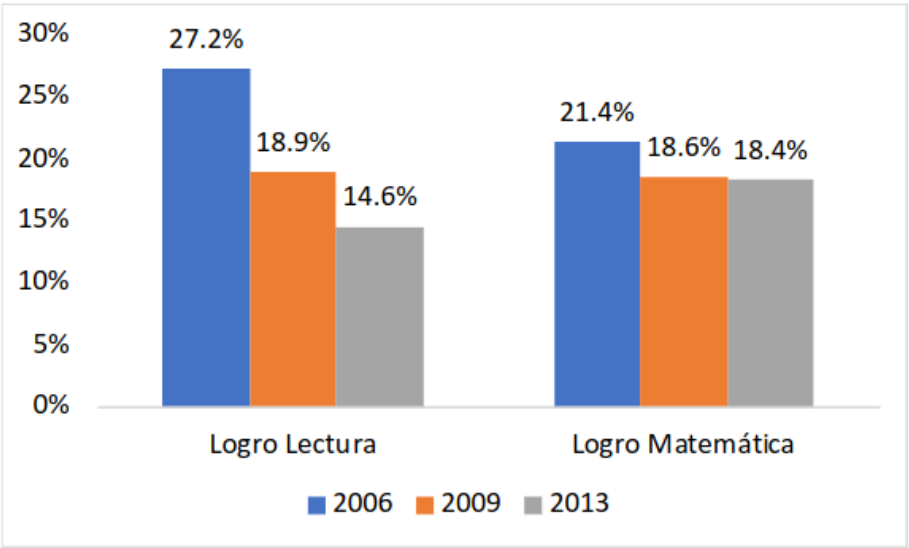


Figura 3: Niveles de logro en lectura y matemática, alumnos de tercero básico, 2006 a 2013

(Cien, 2019)

5.5.3. Graduandos

Las evaluaciones a graduandos realizadas por el MINEDUC se hicieron desde 2010 hasta 2018. Los resultados muestran que el logro en lectura mejoró 12 puntos porcentuales, de 22.4 % en 2010 a 34.8 % en 2018.

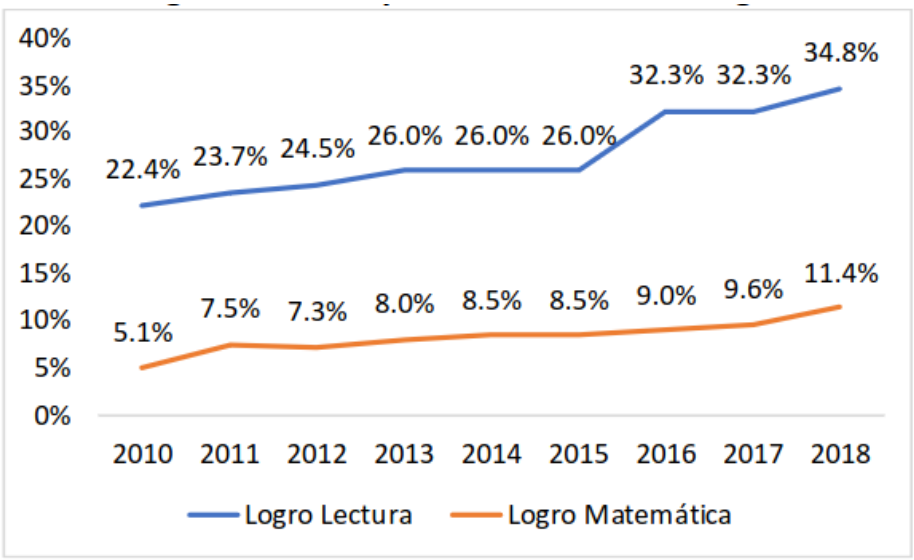


Figura 4: Niveles de logro en lectura y matemática, alumnos graduandos, 2010 a 2018

(Cien, 2019)

Con base en las gráficas anteriores, se puede evidenciar que el logro en lectura para los tres niveles estudiantiles anteriormente mencionados sigue siendo bastante bajo.

5.6. Rapidez de lectura y comprensión lectora en Guatemala en comparación con otros países

La participación de Guatemala en evaluaciones internacionales también confirma que aunque los punteos de logro en lectura han mejorado en los últimos años, aún se tienen desafíos importantes respecto a su capacidad lectora y comprensión de lectura. Recientemente, Guatemala ha participado en dos tipos de pruebas internacionales para evaluar los conocimientos de lectura de la población estudiantil:

Para el nivel de educación primaria, el Estudio Regional Comparativo y Explicativo (ERCE 2019). Para el nivel de educación básica, el Programa Internacional de Evaluación de los Estudiantes para el Desarrollo (PISA-D, 2018).

5.6.1. ERCE 2019

El Cuarto Estudio Regional Comparativo y Explicativo (ERCE 2019) es una iniciativa del Laboratorio Latinoamericano de Evaluación de la Calidad de la Educación (LLECE)

que busca evaluar logros de aprendizaje de estudiantes de tercero y sexto grado de primaria en las áreas de Matemática, Lenguaje (lectura y escritura) y para sexto grado en el área de Ciencias. Guatemala participó en conjunto con otros dieciséis países de Latinoamérica y el Caribe en esta evaluación (UNESCO, 2021).

En la Figura 5, se puede observar que, en la prueba de lectura para tercero primaria, el país obtuvo un puntaje promedio de 656 en el ERCE 2019, resultado que es considerablemente inferior al promedio regional, siendo la diferencia de -41 puntos. Para sexto grado, en la prueba de lectura, el país obtuvo un promedio de 645 puntos en el ERCE 2019, resultado que también es considerablemente inferior al promedio regional, siendo la diferencia de -51 puntos (UNESCO, 2021).

Grado	Área curricular	Puntaje promedio	Comparación con promedio de países del ERCE 2019	Brechas de género	Comparación con el TERCE 2013
3° grado	Lectura	656	-41*	A favor de las niñas	-22*
	Matemática	662	-36*	No hay diferencia	-10*
6° grado	Lectura	645	-51*	No hay diferencia	-33*
	Matemática	657	-40*	A favor de los niños	-15*
	Ciencias	661	-41*	No hay diferencia	-23*

Figura 5: Resultados de Guatemala en el ERCE 2019 por área y grado: Comparación con el promedio regional y con TERCE

(UNESCO, 2021)

5.6.2. PISA-D

El Programa Internacional de Evaluación de Estudiantes para el Desarrollo (PISA, por sus siglas en inglés), creado por la Organización para la Cooperación y el Desarrollo Económico (OCDE, por sus siglas en inglés), evalúa las competencias de jóvenes de quince años en lectura, matemática y ciencias, además de medir sus habilidades para aplicar lo que han aprendido en la escuela a situaciones de la vida real. Guatemala participó en el último PISA-D en 2018 en conjunto con otros siete países, incluidos Bhutan, Cambodia, Ecuador, Honduras, Paraguay, Senegal y Zambia (López Rivas, Mejía, Barrios Robles de Mejía, et al, 2018). Los resultados de PISA-D pueden ser comparados con los resultados de PISA 2015, que tuvo a setenta y dos países como participantes.

En la Figura 6 se puede notar que Guatemala obtuvo una media en Lectura de 369 puntos, con una desviación estándar de 72 en el logro de Lectura de PISA-D. Esto coloca a Guatemala por debajo del promedio de países de la OCDE (493) por 124 puntos y del promedio de países de Latinoamérica (406) por 37 puntos (López Rivas, Mejía, Barrios Robles de Mejía, et al, 2018).

Países	Lectura				Matemática				Ciencias			
	Promedio		Desviación Estándar		Promedio		Desviación Estándar		Promedio		Desviación Estándar	
	Media	Error estándar	Desviación Estándar	Error estándar	Media	Error estándar	Desviación Estándar	Error estándar	Media	Error estándar	Desviación Estándar	Error estándar
PISA-D												
Camboya	321	(2.1)	62	(1.1)	325	(2.7)	75	(1.5)	330	(1.9)	51	(1.1)
Ecuador	409	(3.4)	81	(1.8)	377	(3.1)	76	(1.5)	399	(2.9)	71	(1.4)
Guatemala	**369**	**(3.5)**	**75**	**(2.2)**	**334**	**(3.2)**	**69**	**(2.1)**	**365**	**(2.9)**	**62**	**(2.1)**
Honduras	371	(3.5)	75	(2.3)	343	(3.5)	77	(2.3)	370	(2.9)	62	(2.1)
Paraguay	370	(3.7)	83	(2.0)	326	(2.9)	68	(1.4)	358	(3.3)	72	(1.7)
Senegal	306	(1.8)	72	(1.5)	304	(2.6)	81	(1.7)	309	(1.8)	54	(1.1)
Zambia	275	(3.9)	71	(2.5)	258	(3.9)	72	(2.4)	309	(3.1)	57	(1.9)
Países de Latinoamérica												
Chile	459	(2.6)	88	(1.7)	423	(2.5)	85	(1.4)	447	(2.4)	86	(1.3)
México	423	(2.6)	78	(1.5)	408	(2.2)	75	(1.3)	416	(2.1)	71	(1.1)
Brasil	407	(2.8)	100	(1.5)	377	(2.9)	89	(1.7)	401	(2.3)	89	(1.3)
Colombia	425	(2.9)	90	(1.5)	390	(2.3)	77	(1.3)	416	(2.4)	80	(1.3)
Costa Rica	427	(2.6)	79	(1.6)	400	(2.5)	68	(1.4)	420	(2.1)	70	(1.2)
República Dominicana	358	(3.1)	85	(1.9)	328	(2.7)	69	(2.0)	332	(2.6)	72	(1.8)
Perú	398	(2.9)	89	(1.6)	387	(2.7)	83	(1.4)	397	(2.4)	77	(1.4)
Trinidad y Tobago	427	(1.5)	104	(1.3)	417	(1.4)	96	(1.2)	425	(1.4)	94	(1.1)
Uruguay	437	(2.5)	97	(1.6)	418	(2.5)	87	(1.7)	435	(2.2)	87	(1.3)
Promedios internacionales												
Promedio de países PISA-D	346	(1.2)	74	(0.7)	324	(1.2)	74	(0.7)	349	(1.0)	61	(0.6)
Promedio de países de la OCDE	493	(0.5)	96	(0.3)	490	(0.4)	89	(0.3)	493	(0.4)	94	(0.2)
Promedio de países de América Latina	406	(0.8)	86	(0.5)	379	(0.7)	78	(0.5)	398	(0.7)	76	(0.4)
Promedio de países de ingreso medio bajo	378	(1.1)	78	(0.6)	368	(1.1)	80	(0.7)	392	(0.9)	68	(0.6)
Promedio de países de ingreso medio alto	410	(0.8)	90	(0.4)	402	(0.7)	82	(0.4)	411	(0.6)	80	(0.3)

Figura 6: Desempeño medio de los estudiantes de Guatemala en lectura, ciencias y matemática, en PISA-D en comparación con países de Latinoamérica y promedios internacionales

(López Rivas, Mejía, Barrios Robles de Mejía, et al, 2018)

Resultados como estos son ocasionados por las múltiples fallas en el sistema socioeconómico y educativo guatemalteco, entre ellas: alta tasa de analfabetismo, ausencia de una cultura de lectura inculcada por profesores, carencia de escuelas, repitencia de estudiantes, y falta de programas que motiven la lectura (UNESCO, 2021).

5.7. Ejercicios y técnicas para mejora de rapidez y comprensión lectora

La rapidez y comprensión lectora han sido un tópico de interés para profesionales de diferentes campos, esto ha ocasionado que se hayan desarrollado ejercicios para la mejora de estas dos habilidades. A continuación, se presentan algunos de estos ejercicios.

5.7.1. Entrenamiento de movimiento de ojos

Este ejercicio consiste en entrenar los ojos para seguir una dirección determinada, usualmente de izquierda a derecha, para evitar fijaciones o regresiones. Al corregir los movimientos no deseados de los ojos, permite una mejora la rapidez de lectura, así como la eficacia de la misma (Solan et al., 2001).

5.7.2. Técnica de *finger-pacing*

Esta técnica se basa en utilizar el dedo índice o un lápiz para guiar los ojos del lector y controlar el ritmo de lectura de un texto. Al ayudar al lector a concentrarse en el texto y reducir movimientos no deseados de los ojos, esta técnica permite mejorar la rapidez y comprensión lectora (Cho y Ahn, 2014).

5.7.3. Lectura con taquistoscopio

Este tipo de lectura consiste en leer palabras a través de las rendijas de un taquistoscopio, de esa forma el lector recibe la información en forma de pequeños fragmentos en lugar de recibirla en su totalidad. Al reducir las fijaciones de ojos, este tipo de lectura puede mejorar tanto la rapidez y comprensión lectora (Nist y Simpson, 1990).

5.7.4. Reconocimiento rápido de palabras

Este ejercicio implica encontrar o reconocer palabras específicas en un bloque de texto. El reconocimiento rápido de palabras puede ayudar a mejorar la rapidez de lectura en individuos con dificultades para leer (Schneps et al., 2013).

5.7.5. Lectura activa

Más que una técnica, esta se refiere a un conjunto de técnicas, como tomar notas, escribir resúmenes, o subrayar partes importantes de un texto. Al promover una inmersión y procesamiento del texto de parte del lector, este conjunto de técnicas permite mejorar la comprensión lectora (Harris y Sipay, 1980).

5.8. Aplicación web

Una aplicación web es una aplicación de *software* que se basa en una arquitectura cliente-servidor que utiliza un navegador web como su programa cliente, y se conecta con un servidor alojado en Internet para intercambiar información (Shklar y Rosen, 2003).

Cabe mencionar que una aplicación web cuenta con ciertas características que la diferencian del resto de aplicaciones de *software*, entre ellas:

- Se ejecuta en un servidor web y es accedida a través de un navegador web.
- Es accesible desde cualquier dispositivo que cuente con un navegador web y conexión a internet (computadoras, teléfonos inteligentes, y tablets).
- No necesita ser descargada en el dispositivo desde el cual se quiere consultar, por lo tanto, no ocupa espacio en dicho dispositivo.
- Es lo suficientemente flexible para adaptarse a múltiples modelos de negocio y necesidades de los usuarios.
- Se puede desarrollar con una lista extensa de tecnologías, *frameworks*, y patrones de diseño (todos estos siendo independientes del dispositivo en el que se ejecuten).
- Presentan información dinámica basada en las interacciones del usuario, como parámetros en las solicitudes HTTP y eventos en el DOM.

(Felke-Morris, 2017)

Convencionalmente, se dice que una aplicación web se compone por tres elementos: *frontend*, *backend* y un API (Pressman y Maxim, 2015). El *frontend* es la parte interactiva de la aplicación, el *backend* se encarga manejar la lógica de negocio, y el API es una interfaz que permite que el *frontend* y el *backend* se comuniquen entre sí (Pressman y Maxim, 2015). Estos tres conceptos se explican a detalle más adelante.

5.9. *Backend*

El *backend* se refiere a la parte de una aplicación web que se encarga del procesamiento de datos, este se ejecuta en un servidor web y cumple varias funciones tales como: almacenar datos, realizar cálculos, y tomar decisiones basadas en la información obtenida de los usuarios o terceros, entre otras (Brown, 2017).

Existen muchos lenguajes de programación que se pueden utilizar para el desarrollo del *backend* de una aplicación, entre los más populares se encuentran: PHP, Ruby, y Python. Además, existen múltiples *frameworks* de los que se puede escoger, siendo tres de los más populares: Laravel (PHP), Ruby on Rails (Ruby), y Django (Python). Asimismo, existen varios gestores de bases de datos como MySql, PostgreSQL, MongoDB, entre otros (Mardan, 2014).

5.10. Web API

Un Web API o "Interfaz de Programación de Aplicaciones Web", por sus siglas en inglés, es la parte de una aplicación web encargada de comunicar el *frontend* y el *backend* entre sí. Se compone por un patrón de solicitudes y respuestas HTTP que se utilizan para permitir al servidor web proveer información al cliente y al cliente consultar, editar, o eliminar esa información del servidor (Jin, Sahni y Shevat, 2018). Cabe mencionar que la API es una parte en sí del *backend* de una aplicación, no algo ajeno como lo es el *frontend*.

Puesto que la API es una parte del *backend*, los lenguajes de programación y *frameworks* que se pueden utilizar para su desarrollo son los ya mencionados en la sección anterior: PHP, Ruby y Python; Laravel, Ruby on Rails, y Django (Mardan, 2014).

La decisión sobre el lenguaje, *framework* y demás tecnologías que se utilizan para el *backend* (y la API) de una aplicación depende de varios factores, tales como: el tipo de aplicación, los requerimientos de la misma, curva de aprendizaje de las tecnologías, escalabilidad, seguridad y costo, entre otros.

5.11. *Frontend*

El *frontend* es la parte de una aplicación web con la que los usuarios pueden interactuar. Este se conforma por la interfaz gráfica, que incluye elementos estáticos e interactivos, y el código asociado a dicha interfaz, usualmente HTML, CSS, y Javascript (Robbins, 2018).

5.11.1. HTML

El Lenguaje de Marcado de Hipertexto o HTML, por sus siglas en inglés, es un sistema que se utiliza para describir la estructura de un documento. Este lenguaje de marcado está conformado por una serie de etiquetas que pueden contener desde texto, imágenes, e hipervínculos, hasta formularios, tablas, y menús. Cabe mencionar que HTML es el lenguaje de marcado estándar para desarrollar aplicaciones web, y su versión más reciente es HTML5 (Robbins, 2018).

5.11.2. CSS

Las Hojas de Estilo en Cascada o CSS, por sus siglas en inglés, describen cómo el contenido de una página web se presenta al usuario. Esto incluye los colores, imágenes, tipografía, espacios, animaciones, entre otros. Además, también describen cómo se presenta el contenido en distintos dispositivos como tabletas y teléfonos inteligentes (Robbins, 2018).

5.11.3. Javascript

Javascript es un lenguaje de *scripting* que permite a los desarrolladores añadir interactividad a las páginas web. Por ejemplo: obtener y mostrar información de forma dinámica, validar errores en tiempo real, cambiar estilos dinámicamente, entre otros. Javascript interactúa directamente con el Modelo de Objetos del Documento o DOM, por sus siglas en inglés, que contiene todos los elementos de una página web que pueden ser manipulados (Robbins, 2018).

5.11.4. *Frameworks* para *frontend*

Un *framework* para *frontend* es un paquete de *software* conformado por librerías y herramientas que ayudan a los desarrolladores a construir interfaces de aplicaciones web. La principal ventaja que ofrecen este tipo de *frameworks* es la abstracción de la lógica para construir la interfaz de la aplicación, al ofrecer un modelo de componentes, plantillas, y una librería para manejar el estado de la aplicación (Fain y Moiseev, 2017).

Existen diferentes *frameworks* para *frontend*. A continuación, se presentan los tres más utilizados, en conjunto con sus ventajas y desventajas.

React

React es un *framework* de desarrollo web creado y mantenido por Facebook, Inc. Este *framework* se caracteriza por ser rápido, intuitivo, y declarativo (Stefanov y Adams, 2019).

Angular

Angular es un *framework* de código abierto basado en Javascript mantenido por Google. Es un *framework* robusto que se caracteriza por su amplia gama de librerías y herramientas a disposición del desarrollador (Fain y Moiseev, 2017).

Vue

Vue es un *framework* de código abierto creado por Evan You y mantenido por él en conjunto con otros miembros del proyecto. Vue se caracteriza por ser rápido, accesible, y versátil (Vue, 2023).

Ventajas	**Desventajas**
React es mantenido por Facebook, esto garantiza el soporte continuo al igual que la confiabilidad de las actualizaciones y parches (React, s.f.).	Las aplicaciones basadas en React pueden tener un peor desempeño que aquellas construidas con otros *frameworks*.
Al usar una sintaxis declarativa, escribir código y leer código de React es más sencillo, incluso para principiantes en el ámbito de desarrollo de *frontend* (React, s.f.).	Un proyecto desarrollado en React requiere mucho más código base (código que se utiliza de forma repetitiva para desarrollar una nueva aplicación). Esto ocasiona que crear y mantener aplicaciones en React requiera más tiempo y esfuerzo que con otros *frameworks* (Stefanov, 2016).
La comunidad de React es bastante extensa, por lo que hay bastantes recursos, documentación, y soporte disponible (React, s.f.).	Los proyectos desarrollados en React suelen ser pesados, y pueden impactar la velocidad de carga y el desempeño de la aplicación (Chinnathambi, 2018).
La curva de aprendizaje de React es considerablemente menor a la de otros *frameworks* para desarrollo web (Stefanov, 2016).	

Cuadro 1: Ventajas y desventajas de React

(elaboración propia)

Ventajas	**Desventajas**
Al ser mantenido por Google, los paquetes, parches y actualizaciones tienen la garantía de ser menos propensos a errores (Angular, 2023).	Angular tiene una curva de aprendizaje bastante pronunciada, especialmente para desarrolladores que están empezando en el desarrollo de *frontend* (Freeman, 2017).
Puesto que es uno de los *frameworks* más populares para desarrollo web, existe una vastedad de recursos, documentación, y soporte disponible (Angular, 2023).	Los proyectos en Angular suelen ser pesados, y pueden impactar la velocidad de carga (Chinnathambi, 2017).
La arquitectura de Angular obliga a los desarrolladores a convertir aplicaciones complejas en componentes más pequeños, esto resulta en código más reusable y mantenible (Chinnathambi, 2017).	La sintaxis de Angular puede resultar compleja para los desarrolladores nuevos, debido a lo extensa y poco intuitiva que es (Freeman, 2017).
Las aplicaciones desarrolladas en Angular son rápidas, debido a la forma en la que el *framework* maneja cómo se muestran los datos de una aplicación (Fain y Moiseev, 2017).	Debido a la arquitectura de Angular, puede resultar demasiado robusto para proyectos pequeños en los que la mayoría de sus capacidades no se aprovecharían (Fain y Moiseev, 2017).
El *framework* es bastante personalizable, lo que le permite adecuarse a las necesidades de los desarrolladores (Angular, 2023).	

Cuadro 2: Ventajas y desventajas de Angular

(elaboración propia)

Ventajas	**Desventajas**
Vue es uno de los *frameworks* más sencillos de aprender, especialmente para desarrolladores que están empezando en el ámbito de *frontend* (Vue, 2023).	Al no ser mantenido por una empresa como Facebook o Google, no se puede garantizar el mismo nivel de confiabilidad para las actualizaciones y parches.
Uno de sus enfoques es ser un *framework* ligero, lo que permite que las aplicaciones que lo usan sean más rápidas y eficientes (Vue, 2023).	Cuenta con menos librerías y *plugins* que otros *frameworks*, lo que puede hacer que el desarrollo de aplicaciones sea más laborioso y tardado (Skellie, 2019).
Puede ser usado para desarrollar proyectos pequeños (como aplicaciones de una sola página) y proyectos grandes (como aplicaciones de empresas) gracias a su versatilidad (Vue, 2023).	Debido al alcance del proyecto y el tamaño de la comunidad que lo apoya, puede ser más difícil encontrar recursos, documentación, y soporte sobre el *framework* (Krutov, 2019).
Puede ser integrado en proyectos existentes, incluso aquellos que ya utilizan otros *frameworks* como React o Angular (Vue, 2023).	

Cuadro 3: Ventajas y desventajas de Vue

(elaboración propia)

5.11.5. Manejo de estado

En las aplicaciones web, el manejo del estado es el proceso de mantener la información de una aplicación de manera consistente, predecible y confiable. Este proceso permite que todos los componentes (o los que lo necesiten) tengan conocimiento y acceso a dicha información en todo momento, garantizando así que la interfaz de usuario esté sincronizada con el estado actual. El estado de una aplicación se puede representar con cualquier estructura de datos. Sin embargo, la convención para la mayoría de aplicaciones y *frameworks* es representarla con un objeto o diccionario.

Existen varias librerías útiles para el manejo del estado del *frontend* de una aplicación web, a continuación, se presentan tres, una para cada uno de los *frameworks* para *frontend* presentados en la sección anterior.

Redux

Redux es una librería de código abierto de Javascript que permite manejar y centralizar el estado de una aplicación. La principal característica que ofrece esta librería es producir un estado predecible, centralizado, depurable, flexible, y fácil de probar (Redux, 2023).

NgRx

NgRx es una librería basada en Redux que provee manejo de estado reactivo para aplicaciones de Angular. Entre las características que ofrece se encuentran: manejo de estado local

y global, aislamiento de efectos secundarios, y herramientas para desarrolladores (NgRx, 2023).

Reactivity API

La API Reactivity provista en el paquete base de Vue ya incluye funcionalidades que permiten manejar el estado global en una aplicación. Las ventajas de esta API incluyen: manejo de estado global y local (incluido por defecto en Vue) y simplicidad en el diseño de la store (Vue, s.f.)

5.11.6. Manejo de efectos secundarios

En las aplicaciones web, el manejo de efectos secundarios se refiere a manejar acciones asíncronas como llamadas a un API, tareas que demoren un tiempo considerable, o modificar información compartida entre componentes. Por lo general, este proceso requiere que se añada un *middleware*, o código intermediario en español, a la aplicación que se encargue de interceptar y manejar los efectos secundarios asíncronos sin bloquear el flujo de la aplicación (Bugl, 2018).

Hay varias librerías disponibles que permiten manejar efectos secundarios asíncronos en aplicaciones web, a continuación, se presentan tres, una para cada uno de los *frameworks* para *frontend* presentados previamente.

Redux-Saga

Redux-Saga es una librería de Javascript que permite el manejo de efectos secundarios en aplicaciones que utilizan Redux. Esta librería se caracteriza por ser fácil de manejar, fácil de testear, y ejecutarse eficientemente (Redux-Saga, 2023).

NgRx effects

Los *effects* o efectos, en español, de la librería NgRx permiten el manejo de efectos secundarios en aplicaciones de Angular al aislarlos del resto de la aplicación. Este apartado de la librería ofrece varias ventajas como una mejora del flujo de la aplicación y mejor manejo de estado (NgRx, 2023).

Vuex

Vuex es un patrón de manejo de estado y librería para aplicaciones de Vue, la librería provee un almacenamiento centralizado además de manejo de efectos secundarios mediante acciones. Vuex es fácil de utilizar, no requiere mucho código base, y tiene una integración sencilla con Vue (Vuex, 2023).

5.11.7. Manejo de estilos

Como se ha mencionado anteriormente, los estilos definen cómo se verá una aplicación web para el usuario final. Existen varias formas, técnicas, y librerías que permiten manejar los estilos en una aplicación web, a continuación, se presentan algunas de las formas más populares para hacerlo.

Inline styling

Inline styling, o los estilos en línea en español, permiten escribir los estilos para un elemento directamente en la etiqueta HTML de dicho elemento, haciendo uso del atributo *"styles"* propio de las tags HTML. Los estilos en línea son fáciles de agregar y utilizar por lo que resultan útiles para estilizar aplicaciones pequeñas, sin embargo, son poco mantenibles a largo plazo, en especial para proyectos complejos (Freeman y Robins, 2018).

Styled components

Los *styled components*, o componentes estilizados en español, es una librería que permite añadir estilos (CSS) directamente en los componentes programados en Javascript (Hoy, 2020). Esta librería permite la modularización de estilos, evita el uso de archivos CSS externos, y facilita la creación de estilos. Sin embargo, el uso de esta librería puede incrementar el tamaño del *bundle* de las aplicaciones en las que se utilice.

Sass

Hojas de estilo sintácticamente asombrosas (SASS por sus siglas en inglés), es un preprocesador de CSS. Sass extiende las capacidades de CSS convencional con el uso de variables, funciones, y reglas anidadas. Esto permite que se puedan crear código leíble y mantenible, en menos tiempo, y con una menor cantidad de líneas (Chowdhury, 2019). A pesar de esto, el uso de Sass puede tener un impacto negativo en el tamaño del proyecto y además, necesita configuración previa para poderse ejecutar en un proyecto.

5.12. Principios de diseño

5.12.1. Color

El color es la propiedad que posee un objeto con respecto a la luz reflejada por sí mismo. La luz, en un sentido estricto, es la parte del espectro electromagnético que puede ser percibida por el ojo humano, con longitudes de onda entre 400 y 700 nanómetros (Nassau, 1998).

Cuando la luz en este rango de longitudes de onda interactúa con un objeto no transparente se divide en dos partes: las longitudes de onda reflejadas y las longitudes de onda

absorbidas. La combinación de longitudes de ondas que un objeto refleja y absorbe determinan el color que el ojo percibe. Esto se puede observar gráficamente en la Figura 7.

Figura 7: Absorción y reflexión de colores en objetos de color rojo, blanco y negro

(Nassau, 1998)

Círculo cromático

El círculo cromático es una representación circular que sirve para ordenar los colores de acuerdo con su matiz (Hard, 2013). Está compuesto por los colores primarios: rojo, amarillo, y azul, y los colores intermedios llenando los espacios entre estos. Los hay de dos formas, escalonados o en degradé. En el caso de los escalonados, el círculo se puede dividir en cualquier cantidad de colores, los más comunes tienen 12, 24, y 48 colores (Hard, 2013).

Figura 8: Círculo cromático de 24 escalones

(elaboración propia)

Psicología del color

El color tiene un impacto psicológico significativo en el ser humano. Puede alterar las emociones, sentimientos, y comportamientos del receptor. Las alteraciones que puede causar son variadas y dependen de muchos factores, por ejemplo: experiencias individuales, contexto en el que se percibe el color, y trasfondos culturales (Hogg y Garrow, 2003).

Diferentes colores causan diferentes sentimientos, emociones y comportamientos (Schloss y Palmer, 2021). Por ejemplo, el color rojo se asocia con peligro, pasión, y emoción, mientras que el color verde se asocia con paz, crecimiento, y fertilidad. Aunque estos sentimientos no

son universales y dependen de muchos factores, se han desarrollado gráficas que muestran una generalización de los sentimientos asociados con cada color (Hogg y Garrow, 2003). La Figura 8 es un ejemplo.

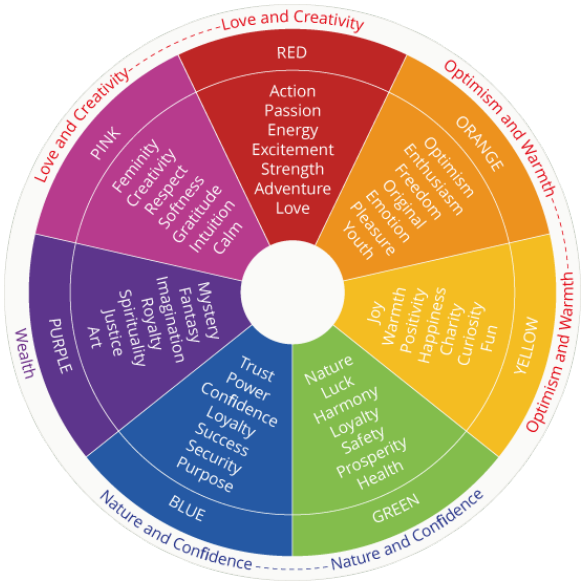


Figura 9: Rueda de colores y las emociones asociadas a cada color

(Clark-Keane, 2022)

Cuando se diseña una aplicación web, es importante considerar el impacto psicológico que los colores tienen en los usuarios y escoger los colores que transmitan las emociones y sentimientos deseados en los mismos (Kujala y Walsh, 2011). Asimismo, es importante no mostrar muchos o muy pocos colores al usuario ya que el hacerlo causará una mala experiencia al utilizar la aplicación (Kujala y Walsh, 2011).

Esquemas de colores

Un esquema de colores es un conjunto de colores del círculo cromático que se han escogido para un propósito específico. Por ejemplo: pintura de una casa, colores de un producto, o estilos de una aplicación. Existen varios tipos de esquemas de colores, a continuación se listan algunos de los más comunes:

Análogo: Un esquema complementario utiliza tres colores adyacentes o cualquiera de sus tintas y sombras en el círculo cromático. Este esquema es armonioso y unificado (Marks, 2019).

Complementario: Un esquema complementario está compuesto por colores opuestos en el círculo cromático. Este esquema tiene un alto contraste y energía (Marks, 2019).

Triádico: Un esquema triádico está compuesto por colores equidistantes en el círculo cromático. Este esquema provee un alto contraste en conjunto con armonía entre los colores (Marks, 2019).

Complementario dividido: Este tipo de esquema utiliza los dos colores adyacentes al complemento de un color en el círculo cromático. Este esquema provee una amplia variación de colores con alto contraste y armonía entre sí (Marks, 2019).

Tetrádico: Un esquema tetrádico está compuesto por dos pares de colores complementarios en el círculo cromático. Este esquema provee más variación de colores así como balance y alto contraste (Marks, 2019).

Estos esquemas se pueden ver gráficamente en la Figura 10.

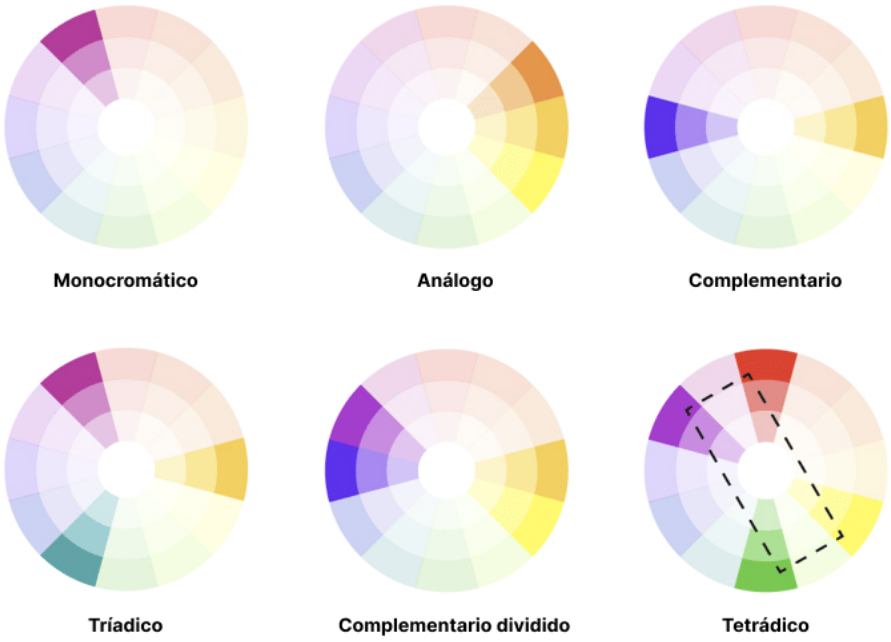


Figura 10: Esquemas de colores

(Gübelin, 2019)

5.12.2. Principios de Gestalt

Los principios de Gestalt son un conjunto de leyes que explican cómo la gente percibe y construye significado de la información visual que se les presenta (Buley, 2010). La psicología detrás de estos principios sostiene que los seres humanos naturalmente percibimos los objetos como patrones organizados y completos, en lugar de percibir sus partes individuales (Buley, 2010).

Estos principios se han utilizado extensamente para el diseño de productos tales como, presentaciones, interiores, logos, publicidad, libros, ropa, e interfaces gráficas (Liang, 2018). Hablando de esta última específicamente, el uso de estos principios permite crear interfaces balanceadas, fácilmente navegables, e intuitivas para el usuario.

A continuación, se presenta una definición de algunos de los principios de Gestalt en

conjunto con cómo se utilizan en el diseño de interfaces gráficas.

Principio de proximidad

Este principio establece que, en comparación, los objetos que están cerca entre sí parecen más relacionados que los objetos que están alejados y es más probable que se consideren como un todo (Liang, 2018). Este principio se utiliza para crear conexiones o reflejar la falta de conexión entre dos o más objetos, proveer estructura, y generar jerarquía. En la Figura 11 se muestran ejemplos de este principio, en ambos lados, se pueden dividir los elementos en dos columnas, independientemente del color de estos.

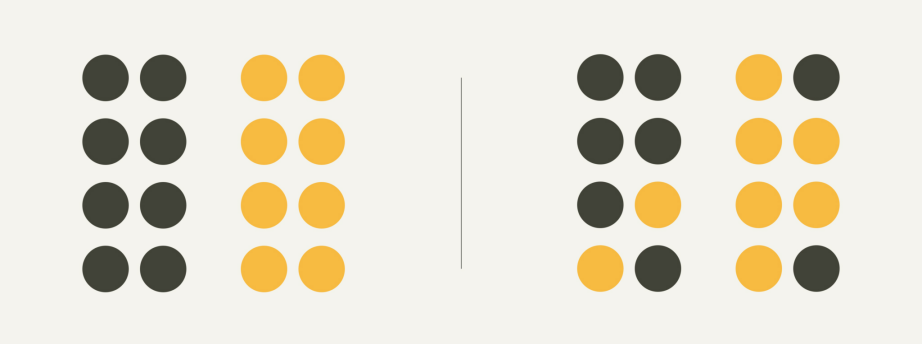


Figura 11: Ejemplos del principio de proximidad

(Giménez, 2018)

Principio de similaridad

Este principio establece que elementos con características comunes, como forma, color y orientación son percibidos como un grupo y están relacionados entre sí (Liang, 2018). Este principio se utiliza para crear patrones visuales, transmitir significado, y enfatizar información importante. La Figura 12 muestra un ejemplo del principio en cuestión, en el cual los cuadros rojos se asocian con un grupo distinto a los círculos amarillos.

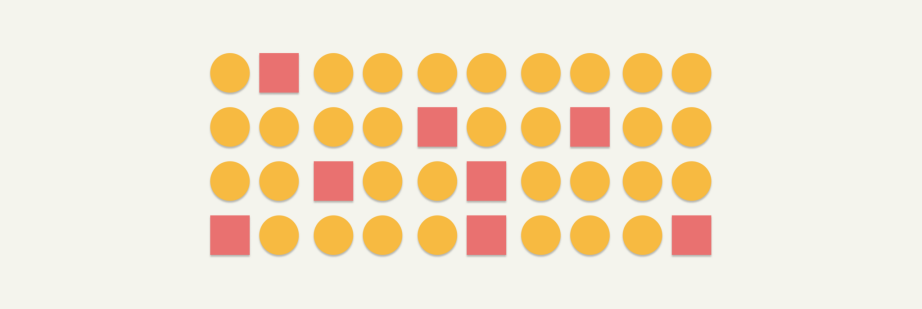


Figura 12: Ejemplo del principio de similaridad

(Giménez, 2018)

Principio de cierre

Este principio establece que hay una tendencia de percibir formas incompletas como completas al llenar la información que hace falta (Liang, 2018). Este principio se utiliza para crear interés y añadir profundidad. La Figura 13 muestra en ejemplo de este principio, los círculos incompletos (1 y 2) se perciben como parte del último círculo (3).

Figura 13: Ejemplo del principio de cierre

(Garcia Mingrone, 2022)

Principio de continuidad

Este principio establece que hay una tendencia de ver las líneas y los patrones como si continuaran en la misma dirección, incluso si estos son interrumpidos por otros elementos (Liang, 2018). Este principio se utiliza para crear un flujo visual, guiar los ojos del usuario, y construir un diseño cohesivo. La Figura 14 muestra un ejemplo de este principio, los ojos naturalmente siguen la línea horizontal recta, a pesar de que hay elementos que la interrumpen.

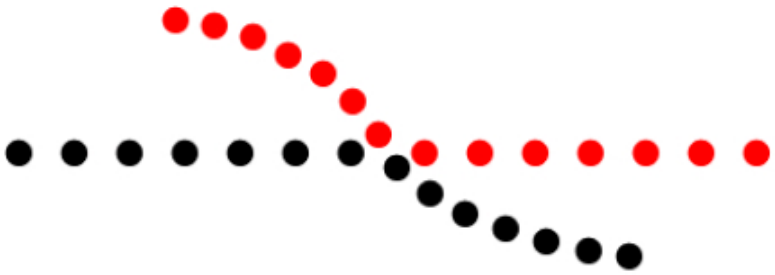


Figura 14: Ejemplo del principio de continuidad

(Giménez, 2018)

Principio de simetría

Este principio establece que hay una tendencia de percibir los objetos como simétricos y organizados a partir de un eje central (Liang, 2018). Este principio se utiliza para crear balance, armonía y orden. En la Figura 15 se muestran elementos alineados a la izquierda y derecha de un eje central como un ejemplo de este principio.

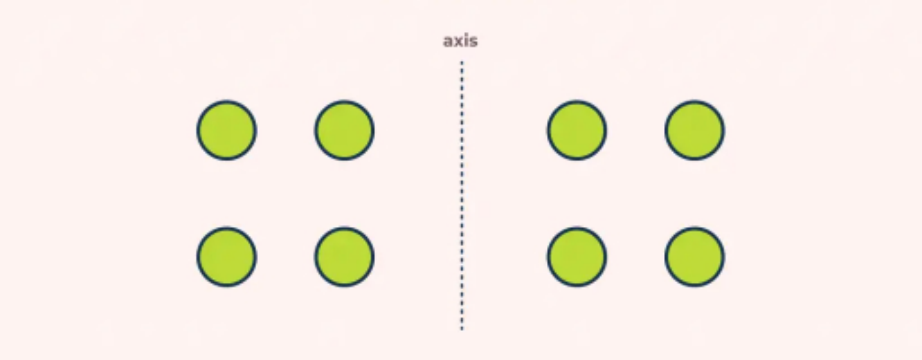


Figura 15: Ejemplo del principio de simetría

(Chapman, 2023)

5.13. Experiencia de usuario

La experiencia de usuario (comúnmente denotada por UX), se refiere a la experiencia de un usuario al utilizar un servicio o producto. Es una combinación de la percepción del usuario respecto a la facilidad de uso, simplicidad, e intuitividad del producto, entre otras características (Rosenfeld y Morville, 2019).

5.13.1. Buenas prácticas de UX en desarrollo de aplicaciones web

Hablando estrictamente de aplicaciones web, hay un conjunto de prácticas generales que se pueden implementar para mejorar la experiencia del usuario al utilizar la aplicación. A continuación se presentan algunas de ellas.

- Proveer mensajes de error claros y concisos al usuario.
- Evitar el desorden y elementos innecesarios en las pantallas de la interfaz.
- Utilizar lenguaje plano y sencillo a lo largo de la aplicación.
- Crear una jerarquía visual mediante tamaños, estilos, y espacios.
- Minimizar el ingreso de datos por parte del usuario para completar una tarea.
- Proveer instrucciones claras y concisas de cómo utilizar la interfaz.

- Utilizar navegación consistente y predecible a lo largo de la aplicación.
- Utilizar señalamiento visual para guiar a los usuarios a través de la interfaz.
- Perfeccionar el diseño de la aplicación mediante iteraciones basadas en pruebas con los usuarios.

(Krug, 2014).

5.14. Pruebas unitarias

Las pruebas unitarias son pruebas que se enfocan en probar una porción o unidad pequeña, simple, y aislada de código. Al concentrarse en una unidad pequeña de código, se garantiza que sean rápidas, repetibles, y fáciles de leer o editar. Su propósito es verificar que el código que está siendo probado se comporte como es esperado y garantizar que lo siga haciendo si se modifica alguna parte del mismo (Patton, 2006).

Existen diferentes *frameworks* para realizar pruebas unitarias en aplicaciones web, estas varían dependiendo del lenguaje de programación que se utilice, el *framework* utilizado, y del código que se quiera probar. A continuación, se presentan tres de las librerías más conocidas, una para cada *framework* presentado en la sección 5.11.4.

5.14.1. *Frameworks* para pruebas unitarias

Jasmine

Jasmine es un *framework* basado en el comportamiento para hacer pruebas de código en Javascript que viene incluido con Angular. Entre las principales ventajas que ofrece este *framework* se encuentran: una sintaxis simple y obvia que facilita la escritura de pruebas, independencia de otros *frameworks*, y un extenso conjunto de APIs que permiten crear pruebas complejas (como las que incluyen *mocking*, *spying*, y llamadas asíncronas) (Jasmine, s.f.).

Vitest

Vitest es un *framework* veloz para hacer pruebas de código de Vue. Entre las principales ventajas que ofrece este *framework* se pueden mencionar: una API intuitiva que lo hace fácil de aprender y utilizar, integración sencilla con otros *frameworks* de pruebas, y un poderoso set de características como soporte para *mocking*, *spying*, y pruebas de código asíncrono (Vitest, 2021).

Jest y Enzyme

Jest es un *framework* para hacer pruebas de código escrito en Javascript enfocado en la simplicidad. Puede ser considerado el *framework* estándar para aplicaciones web puesto que la mayoría de *frameworks* están escritos utilizando a Jest como base. Además, Jest funciona con proyectos que utilizan Angular, React, Vue, entre otros. Entre las principales ventajas de este *framework* se encuentran: configuración casi nula para incluirlo en proyectos, buen desempeño en la ejecución de las pruebas, y un API rica en características que permite realizar prácticamente cualquier tipo de prueba (Jest, 2023).

Enzyme es una utilidad de prueba de JavaScript para React que facilita el desarrollo de pruebas para la salida u *output* de componentes de React. Entre las ventajas principales de Enzyme se pueden mencionar: manipulación fácil de componentes, opciones de pruebas flexibles, y soporte y mantenimiento continuo por parte de contribuyentes (Enzyme, s.f.).

Jest se utiliza en conjunto con Enzyme para realizar pruebas unitarias en proyectos que utilizan React.

5.14.2. Cobertura

En el ámbito de pruebas de código, la cobertura mide el número de líneas de código ejecutadas durante una *suite* de pruebas en un programa (Mustafa et al., 2013) La mayoría de las herramientas de prueba de código expresan la cobertura como un porcentaje de las instrucciones, ramas, funciones y líneas que son cubiertas por las pruebas unitarias desarrolladas. La cobertura es un indicador importante de la calidad del *software* y una parte esencial del mantenimiento del *software*. En general, se puede decir que mientras más altos sean los porcentajes de cobertura para un proyecto, mejor calidad tiene (Shahid et al., 2011). No existe un mínimo de cobertura obligatorio, sin embargo, 80 % de cobertura en instrucciones, ramas, funciones, y líneas es generalmente aceptado como el porcentaje ideal al cual apuntar (Rutledge, 2021).

5.15. Postman

Postman es una plataforma para construir, utilizar y probar APIs. Ofrece una amplia gama de características como:

- Fácil almacenamiento, catalogación, y colaboración de APIs en una plataforma central.
- Extenso conjunto de herramientas para diseño, desarrollo, pruebas, y documentación de APIs.
- Flexibilidad para que las organizaciones puedan aplicar sus prácticas de diseño al crear APIs.
- Capacidad de colaboración y organización a través de espacios de trabajo compartidos.

- Integración con aplicaciones importantes de la industria para mejorar el proceso de desarrollo de APIs.

(Postman, 2023)

En el ámbito del *frontend*, Postman se utiliza para probar y entender las rutas que se encuentran disponibles en la API provista por los desarrolladores del *backend*. Esto permite hacerse una idea de cómo obtener, almacenar, y manejar la información de la base de datos en el estado de la aplicación.

5.16. Control de versiones

El control de versiones, también conocido como control de código fuente, es una práctica del desarrollo de *software* para dar seguimiento y manejar cambios realizados en el código fuente de un proyecto (GitLab, 2023).

Esta práctica permite a los desarrolladores ver el historial de un archivo a lo largo del tiempo, regresar a una versión anterior si es necesario, proteger el código de daños irreparables, experimentar con adiciones de código sin peligro, agilizar el proceso de desarrollo, y finalmente, contar con una única fuente de verdad en cuanto a código se refiere (GitLab, 2023).

5.16.1. Github

Github es el servicio de alojamiento más grande para repositorios de Git. Github permite a los desarrolladores almacenar y compartir repositorios, colaborar con otros desarrolladores, revisar código, entre otras funcionalidades (Chacon y Straub, 2014). Con más de 100 millones de desarrolladores, 4 millones de organizaciones, y 330 millones de repositorios, Github se ha convertido en la plataforma estándar para el control de versiones en los últimos años (GitHub, 2023).

5.16.2. *Commits*

Un *commit* es una captura de un repositorio de Git en un punto en el tiempo. Los *commits* son creados por el desarrollador cuando quiere añadir cambios al código y son subidos al repositorio de git posteriormente mediante el comando de Git "*push*"(Chacon y Straub, 2014).

5.16.3. *Branches*

Una *branch* es un conjunto de *commits* consecutivos a los que se les asigna un nombre. Las *branches* son creadas por el desarrollador cuando va a trabajar en una nueva característica

o corregir un error. Conforme se avanza en el desarrollo se van subiendo *commits* a dicha *branch* (Chacon y Straub, 2014).

5.16.4. *Pull Requests*

Una *pull request* es una forma de iniciar una discusión respecto a un conjunto de cambios que se han realizado a un repositorio. Una *pull request* usualmente se crea cuando el desarrollador ha concluido los cambios en una *branch* y quiere incorporarlos en la *branch* principal del repositorio en el que está trabajando (Chacon y Straub, 2014).

5.16.5. Github *actions*

Las Github *actions* son una plataforma de integración y despliegue continuos (ver secciones 5.17 y 5.18 para más información sobre estos dos conceptos) provista por Github que permite a los desarrolladores automatizar la construcción, pruebas, y despliegue de sus proyectos. Las acciones están descritas por *workflows* que se encargan de ejecutar las instrucciones provistas por desarrolladores cuando cierto evento ocurre (GitHub, 2023).

5.16.6. Github *workflows*

Un *workflow* es un proceso automático configurable soportado por Github que ejecuta uno o más trabajos. Estos pueden realizar todo tipo de tareas, por ejemplo: construir y probar una *pull request*, correr las pruebas unitarias de un proyecto, o publicar el código fuente de una aplicación (GitHub, 2023).

Los *workflows* tienen tres componentes básicos: uno o más eventos que activan el workflow, uno o más trabajos que tienen cierta cantidad de pasos, y los pasos de cada trabajo que ejecutan código o acciones (GitHub, 2023).

5.16.7. Github *pages*

Github *pages* es un servicio de alojamiento web ofrecido por Github. Este servicio permite a los usuarios de Github publicar sitios web estáticos directamente desde sus repositorios. Las principales ventajas de utilizar este servicio son la simplicidad, el bajo costo, y la integración fácil con el control de versiones así como con Github actions (GitHub Pages, 2023).

5.17. Integración continua

Continuous integration o integración continua en español, es una práctica de desarrollo de *software* en la que se automatiza la integración de cambios en el código fuente desarrollados por los contribuyentes. Cada integración de código debe ser probada y verificada previo a

formar parte del código final para así poder detectar errores de integración y evitarlos en un ambiente de producción (Humble y Farley, 2010).

En la mayoría de proyectos alojados en Github, se utilizan las Github *actions* y los Github *workflows* para llevar a cabo el proceso de integración continua.

5.18. Despliegue continuo

Continuous deployment o despliegue continuo en español, es una práctica de desarrollo de *software* en la que se automatiza el despliegue de cambios en el código fuente a un ambiente de producción. Esto usualmente se realiza mediante un proceso de que publica los cambios a un ambiente de pruebas, los prueba, y si pasan las pruebas, los despliega a un ambiente de producción (Humble y Farley, 2010).

En la mayoría de proyectos alojados en Github, se utilizan las Github *actions* y los Github *workflows* en conjunto con Github *pages* para llevar a cabo el proceso de despliegue continuo.

CAPÍTULO 6

Metodología

6.1. Investigación

Se realizó una investigación acerca de las principales soluciones que se han desarrollado para ayudar a la mejora de la comprensión lectora y rapidez de lectura en Guatemala. Algunas de estas soluciones son: Progrentis, Innova Reading y el Curso de lectura rápida ofrecido por la Universidad Rafael Landívar de Guatemala. De estas tres alternativas, se extrajeron características que se consideraron importantes para el desarrollo de la aplicación propia. Entre estas características se pueden mencionar: guiar a los estudiantes a través de todo el proceso de mejora en la aplicación, diseñar los ejercicios de manera que los estudiantes los encuentren entretenidos, y hacer la aplicación web sencilla, fácil de usar, y orientada en el usuario.

Además, se investigaron ejercicios y técnicas para mejorar la comprensión lectora y rapidez de lectura que al haber sido probados por sujetos de estudio hayan resultado en una mejora en cualquiera de estas dos áreas. Entre estos destacan: entrenamiento de movimiento de ojos, técnica de *finger-pacing* (lectura con los dedos), lectura con taquistoscopio, reconocimiento rápido de palabras, y lectura activa. Estos ejercicios y técnicas se usaron como base para el diseño de los ejercicios que formarían parte de la aplicación.

Aunado a esto, se contactó a Christian Porras, un emprendedor guatemalteco que desarrolló un programa para mejorar las habilidades de lectura de personas de cualquier edad. Este programa tiene una duración de 10 semanas y cuenta con más de 60 ejercicios cognitivos distribuidos en dos folletos. Estos ejercicios, en conjunto con clases guiadas impartidas por un tutor, permiten a los estudiantes mejorar su comprensión lectora y rapidez lectora. Desde su creación en 2017, este programa ha permitido a cientos de miembros mejorar su comprensión lectora y rapidez de lectura. Christian autorizó el uso de conceptos y ejercicios de su programa para construir la aplicación web desarrollada como parte de este proyecto.

6.2. Diseño de pruebas de diagnóstico y ejercicios

Como se ha mencionado anteriormente, la comprensión lectora y la rapidez de lectura se miden de forma distinta. Por un lado, la comprensión lectora se mide por el porcentaje de respuestas correctas en una prueba de comprensión lectora. Por otro lado, la rapidez de lectura se mide en palabras leídas por minuto.

El proyecto en cuestión busca no solo medir estas dos habilidades, también busca ayudar a los estudiantes a mejorarlas. Por lo que esta fase se dividió en dos subfases.

La primera subfase consistió en desarrollar las pruebas de diagnóstico inicial y final necesarias para medir la comprensión lectora y la rapidez de lectura de los estudiantes. A continuación se presentan las dos pruebas desarrolladas.

Prueba de diagnóstico inicial

La prueba de diagnóstico inicial fue diseñada para medir la capacidad lectora de los estudiantes previo a utilizar la aplicación web. Esencialmente, es un examen conformado por tres pasos. El primer paso es la lectura de un cuento corto titulado "El precio del humo". El segundo paso es una serie de 9 preguntas de opción múltiple respecto a la lectura de la fase anterior. El tercer paso es una tabla donde se muestran los tres resultados de la prueba: porcentaje de respuestas correctas, tiempo de lectura en segundos, y palabras leídas por minuto. Las pantallas asociadas a esta prueba se presentan en la sección de resultados.

Prueba de diagnóstico final

La prueba de diagnóstico final fue diseñada para medir la capacidad lectora de los estudiantes luego de utilizar la aplicación web. Al igual que la prueba de diagnóstico inicial, es un exámen conformado por tres pasos. El primer paso es la lectura de un cuento corto titulado "Galletitas". El segundo paso es una serie de 9 preguntas de opción múltiple respecto a la lectura de la fase anterior. El tercer paso es una tabla donde se muestran los tres resultados de la prueba: porcentaje de respuestas correctas, tiempo de lectura en segundos, y palabras leídas por minuto. Las pantallas asociadas a esta prueba se presentan en la sección de resultados.

La segunda subfase consistió en combinar los ejercicios y técnicas para mejora de rapidez y comprensión lectora investigados previamente, en conjunto con los ejercicios del programa LectoGym para desarrollar ejercicios interactivos que les permitan a los estudiantes mejorar su comprensión lectora y rapidez de lectura. Cabe mencionar que los ejercicios están fuertemente inspirados en algunos de los ejercicios creados por Christian Porras, con algunos cambios menores.

Ejercicios

En total se desarrollaron 5 ejercicios: el primero, titulado "seguimiento de flechas" está diseñado para el entrenamiento de movimiento de ojos; el segundo, titulado "percepción de palabras", está diseñado para entrenar al cerebro a identificar palabras rápidamente; el tercero y cuarto, titulados "encontrar palabras" y "encontrar antónimos" respectivamente, están diseñados para entrenar al cerebro a encontrar palabras en bloques de texto de manera eficaz; el quinto, está diseñado para practicar la lectura activa. Estos ejercicios se explican más a detalle en la sección de resultados.

Además de definir las pruebas de diagnóstico y los ejercicios, se definió el proceso que debería seguir un usuario para completar el uso de la aplicación.

Primero, los usuarios deben realizar una prueba de diagnóstico inicial. Luego, se deben someter a un proceso de 4 semanas, para cada día de la semana, tienen que realizar los 5 ejercicios anteriormente mencionados. Estos 5 ejercicios se deben realizar 3 veces por día, idealmente, una en la mañana, una en la tarde, y una en la noche. Una vez que hayan completado 2 semanas de ejercicios, pueden realizar la prueba de diagnóstico final para saber cuánto han mejorado en comparación con la prueba de diagnóstico inicial.

Cabe mencionar que para cada semana, los ejercicios del 1 al 5 varían en cuanto a contenido, dificultad, y complejidad, lo cual permite que los estudiantes mejoren sus capacidades de comprensión lectora y rapidez de lectura progresivamente. Estas variaciones se presentan en la sección de resultados.

6.3. Diseño de aplicación web

6.3.1. Paleta de colores

Para el diseño de la aplicación web, se comenzó por determinar la paleta de colores de la aplicación. Esta paleta cuenta con 6 categorías de colores:

Cabe mencionar que las categorías de colores primario, secundario, advertencia, y texto tienen en realidad dos tipos de colores: uno claro y uno oscuro, que permiten que la paleta sea más flexible y variada.

- Primario: utilizado en mayor proporción en la aplicación, como en la barra de navegación, botones, y acciones.
- Secundario: utilizado en menor proporción en la aplicación, como para acentuar algún elemento.
- Advertencia: utilizado para informar sobre errores o advertir sobre alguna acción, como llenar un campo de un formulario incorrectamente.
- Texto: utilizado para el texto de la aplicación.
- Fondo: utilizado para el fondo de las pantallas de la aplicación.

Primero, se escogieron los colores primarios de la paleta. Debido a la naturaleza de la aplicación, se pensó en un color que estuviera asociado con confianza, seguridad, y simplicidad. Al hacer un análisis de la Figura 8, se encontró que el color asociado con estas emociones era el color azul. Por lo tanto, se escogió el color azul (#266CAE en código hexadecimal) como el color primario claro de la aplicación. Además, se escogió el color Tinte Índigo (#194A77) como el color primario oscuro de la aplicación.

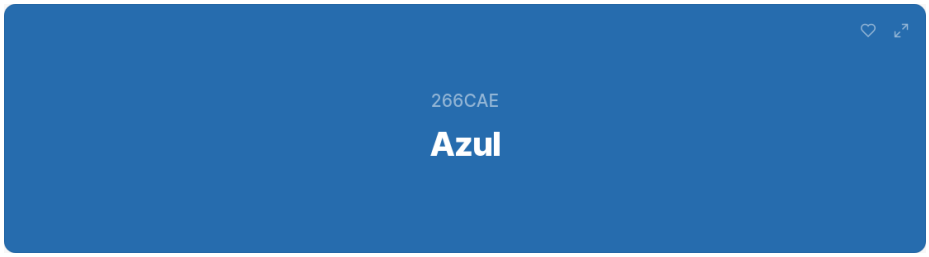


Figura 16: Color primario claro (#266CAE en código hexadecimal)

Figura 17: Color primario oscuro (#194A77 en código hexadecimal)

Luego, se escogió el esquema de colores análogo para determinar el color secundario de la aplicación. Esto debido a que se buscó una apariencia armoniosa, ordenada, y unificada para los colores de la aplicación. El color adyacente que se obtuvo fue una combinación entre azul y verde, luego de explorar un poco las distintas tintas y sombras, se encontró el color Azul Tiffany (#ADDECE en código hexadecimal) que se estableció como el color secundario claro de la aplicación. Además, escogió una sombra de ese mismo color como color secundario oscuro (#95D4C0 en código hexadecimal).

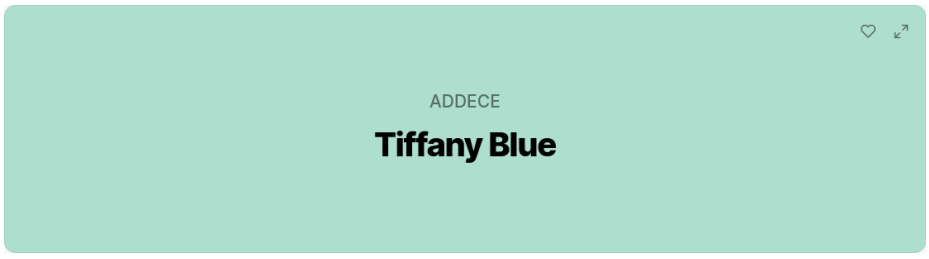


Figura 18: Color secundario claro (#ADDECE en código hexadecimal)

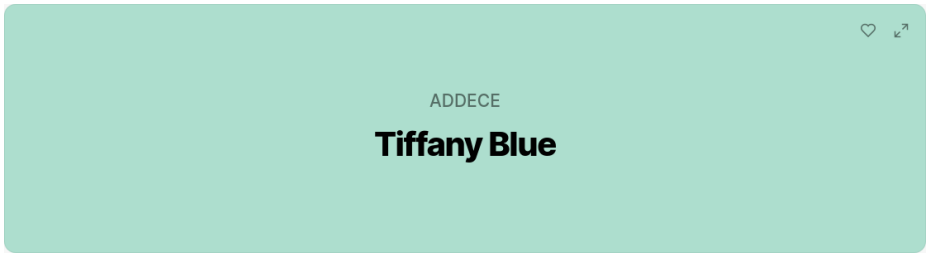


Figura 19: Color secundario oscuro (#95D4C0 en código hexadecimal)

Luego, se estableció el color Rojo Claro (#FF7B7B en código hexadecimal) como el color de advertencia claro, y el color Rojo Imperial (#FF4848 en código hexadecimal) como el color de advertencia oscuro. Esto debido a la necesidad de que el color de advertencia fuera llamativo e imponente.

Figura 20: Color de advertencia claro (#FF7B7B en código hexadecimal)

Figura 21: Color de advertencia oscuro (#FF4848 en código hexadecimal)

Para los colores de texto, se buscaron colores que tuvieran la máxima legibilidad y contraste al ser usados con otros colores. Por lo tanto, se escogió blanco (#FFFFFF en código hexadecimal) como el color de texto claro, y jet (#2D2D2D en código hexadecimal) como el color de texto oscuro.

Figura 22: Color de texto claro (#FFFFFF en código hexadecimal)

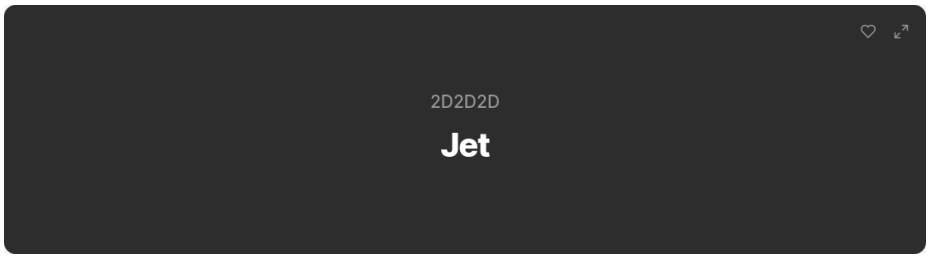


Figura 23: Color de texto oscuro (#2D2D2D en código hexadecimal)

Finalmente, para el color de fondo se escogió el blanco (#FFFFFF en hexadecimal).

6.3.2. Fuente

La fuente seleccionada fue Inter, de Google. Para ello se consideraron varios factores: primero, el tipo público al cual está dirigida la aplicación (jóvenes); segundo, el medio en el que se utiliza la aplicación (laptops y computadoras); tercero, el estilo de la fuente (Inter se caracteriza por ser moderna, limpia, y legible).

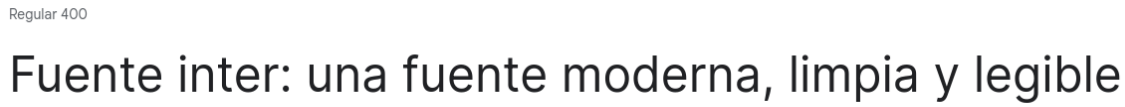


Figura 24: Texto en fuente Inter

(elaboración propia)

6.3.3. Aplicación de principios de Gestalt

Luego de definir la paleta de colores y la fuente de la aplicación, se definió cómo se utilizarían los principios de Gestalt en el diseño de la misma.

El principio de proximidad se utilizó para representar elementos iguales como un grupo, por ejemplo: ítems de un menú o campos de un formulario.

El principio de similaridad se utilizó para agrupar elementos similares en un grupo mental en la cabeza del usuario. Por ejemplo, se asignó una misma fuente para todos los tipos de texto de la aplicación.

El principio de cierre se utilizó para crear elementos incompletos que llamen la atención y generen interés en el usuario. Por ejemplo, barras de carga e indicadores de progreso.

El principio de continuidad se utilizó para guiar los ojos del usuario a lo largo de la aplicación mediante el orden y la colocación de los elementos como tarjetas, botones, y campos.

El principio de simetría se utilizó para mantener orden y armonía a lo largo de la aplicación al alinear los elementos horizontalmente y verticalmente utilizando tablas y cuadrículas.

6.4. Prototipado

Tomando como base los elementos definidos en la sección de diseño, se creó un prototipo de la aplicación en Figma, una aplicación web para desarrollo de interfaces gráficas. Este prototipo se utilizó para presentar a algunos de los usuarios de prueba los estilos de la aplicación y las funcionalidades básicas. A continuación se listan los *mockups*, o maquetas en español, de las pantallas incluidas en el prototipo.

6.4.1. Pantallas del prototipo

Registro

Esta pantalla solicita a un visitante ingresar sus datos para registrarse en la aplicación.

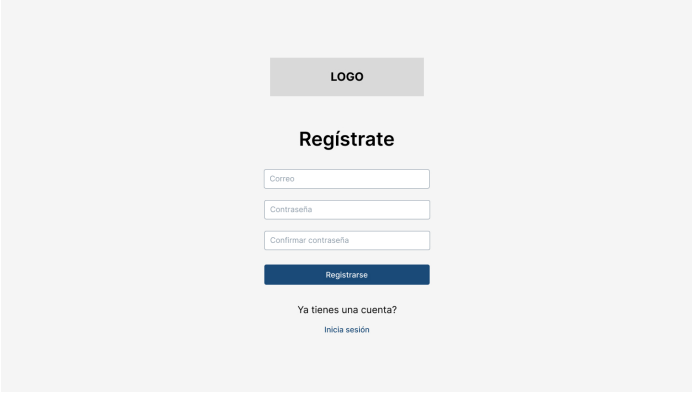


Figura 25: *Mockup* de pantalla de registro

(elaboración propia)

Inicio de sesión

Esta pantalla solicita a un visitante ingresar sus credenciales para iniciar sesión en la aplicación.

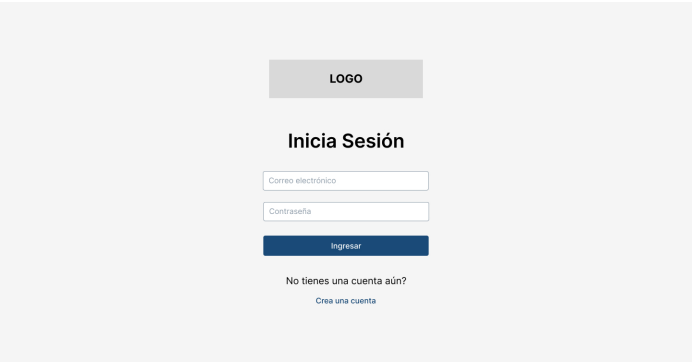


Figura 26: *Mockup* de pantalla de inicio de sesión

(elaboración propia)

Tutorial

Esta pantalla muestra a los usuarios nuevos un tutorial que los guía a través de la aplicación y sus funcionalidades.

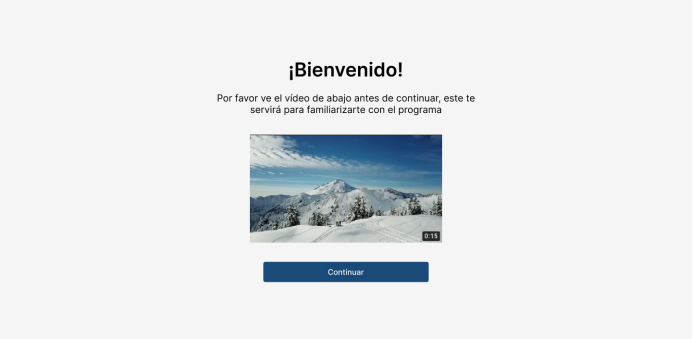


Figura 27: *Mockup* de pantalla tutorial

(elaboración propia)

Prueba de diagnóstico inicial

Esta pantalla cuenta con tres pasos: primero, muestra al usuario instrucciones sobre la prueba de diagnóstico inicial; segundo, presenta al usuario la lectura de la prueba de diagnóstico inicial; y tercero, muestra las preguntas de opción múltiple que el usuario debe responder.

Figura 28: *Mockup* del paso 1 de la prueba de diagnóstico inicial (elaboración propia)

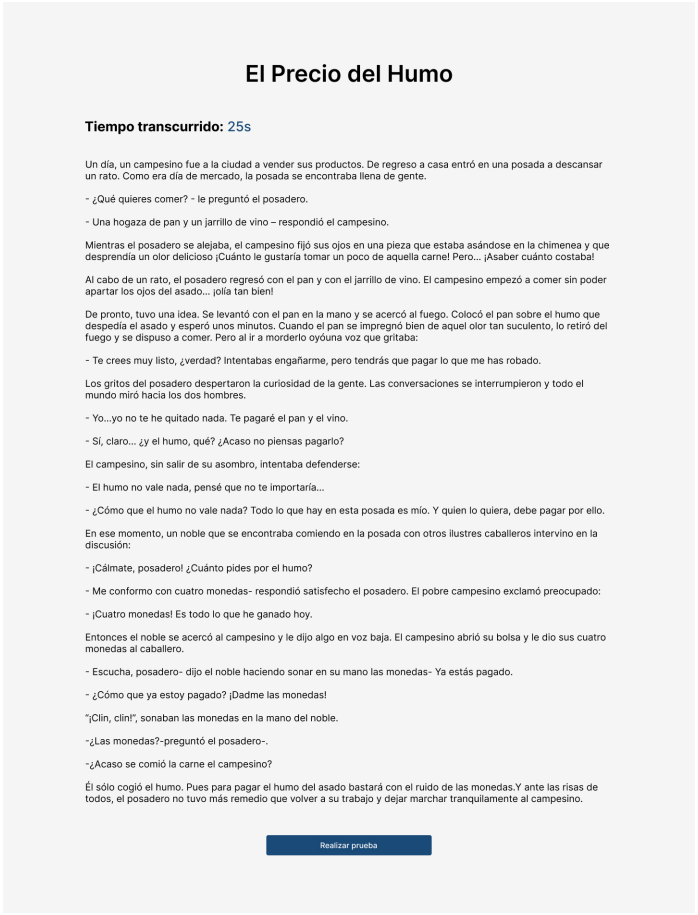


Figura 29: *Mockup* del paso 2 de la prueba de diagnóstico inicial (elaboración propia)

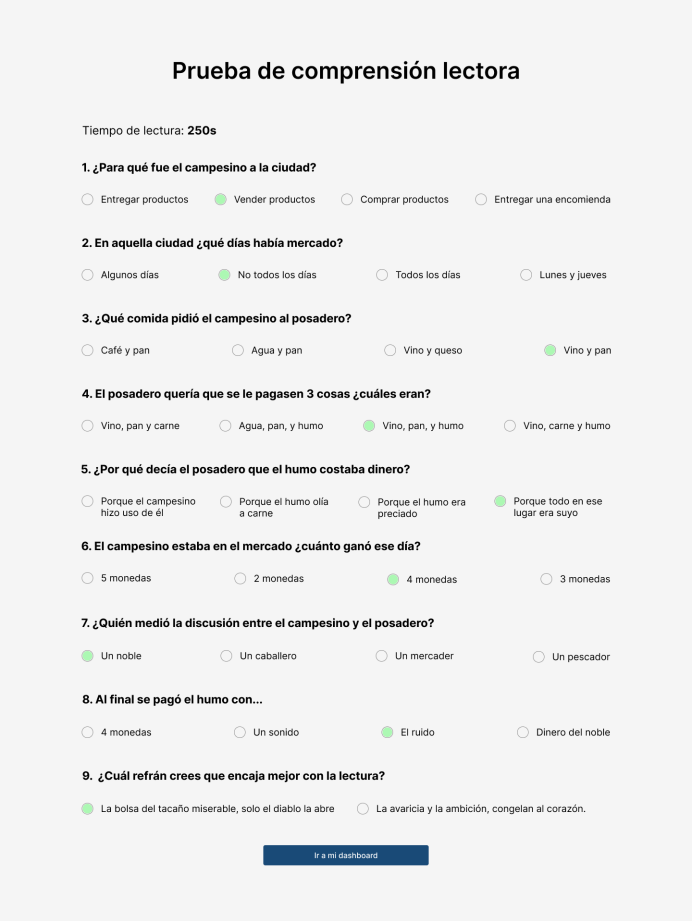


Figura 30: *Mockup* del paso 3 de la prueba de diagnóstico inicial

(elaboración propia)

Dashboard

Esta pantalla muestra al usuario las 4 semanas de ejercicios así como información relevante de cada una, incluyendo el progreso de ejercicios que ha realizado en la semana.

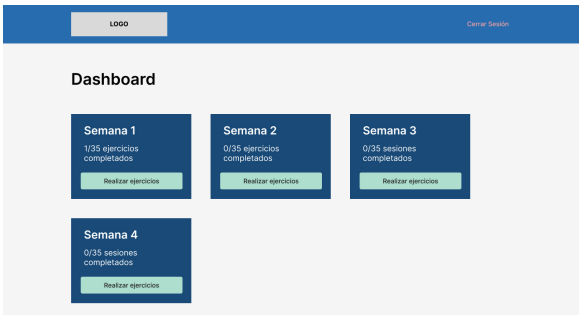


Figura 31: *Mockup* de la pantalla *dashboard*

(elaboración propia)

Semana

Esta pantalla muestra al usuario todos los ejercicios que el usuario debe realizar en una semana. Cada ejercicio cuenta con información relevante y un indicador circular que representa la cantidad de veces que han realizado cada ejercicio.

Figura 32: *Mockup* de la pantalla semana

(elaboración propia)

Ejercicio

Esta pantalla muestra al usuario un ejercicio que debe realizar. En ella, se incluyen detalles como las instrucciones, herramientas que debe utilizar, imagen del ejercicio, y un formulario para subir su respuesta. Esta pantalla varía dependiendo del ejercicio que el usuario realice y la semana a la que corresponda dicho ejercicio.

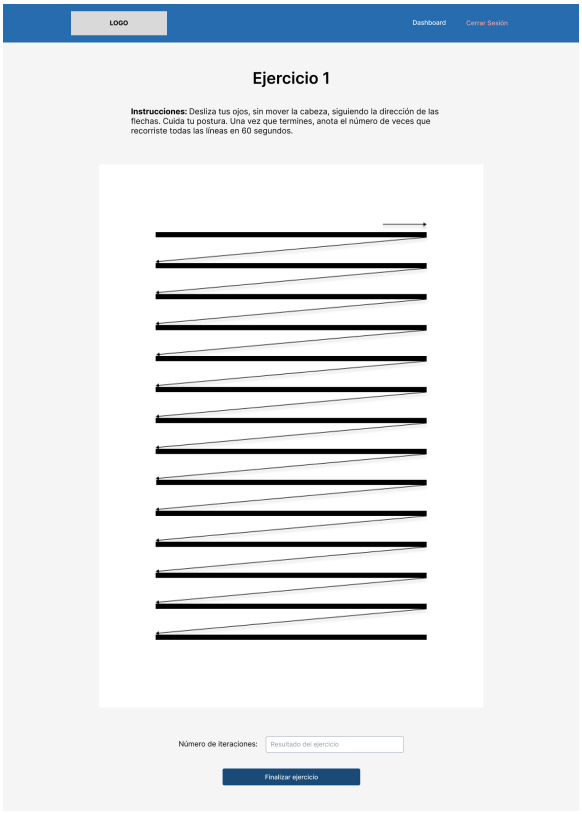


Figura 33: *Mockup* de la pantalla ejercicio 1 de la semana 1

(elaboración propia)

Prueba de diagnóstico final

Similar a la prueba de diagnóstico inicial, esta pantalla cuenta con tres pasos: primero, muestra al usuario instrucciones sobre la prueba de diagnóstico final; segundo, presenta al usuario la lectura de la prueba de diagnóstico final; y tercero, muestra las preguntas de opción múltiple que el usuario debe responder.

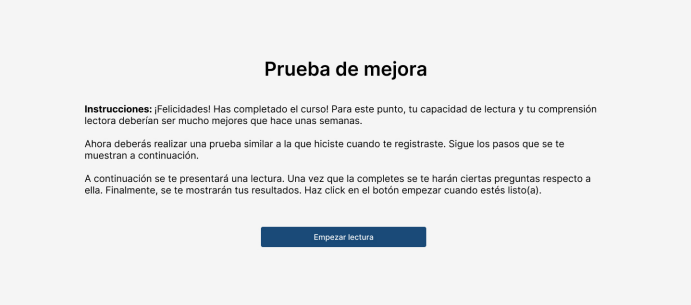


Figura 34: *Mockup* del paso 1 de la prueba de diagnóstico final

(elaboración propia)

Figura 35: *Mockup* del paso 2 de la prueba de diagnóstico final (elaboración propia)

Figura 36: *Mockup* del paso 3 de la prueba de diagnóstico final (elaboración propia)

Resultados

Esta pantalla muestra al usuario los resultados de sus pruebas de diagnóstico inicial y final: porcentaje de respuestas correctas, tiempo de lectura, y palabras leídas por minuto. Además, muestra la mejora del usuario en forma de porcentaje.

Figura 37: *Mockup* de la pantalla de resultados

(elaboración propia)

Luego de culminar con el diseño del prototipo, se le añadió funcionalidad. Es decir, se modificaron los botones, vínculos, y pantallas de manera que un usuario pudiera navegar de la forma en la que lo haría en la aplicación final (ver prototipo funcional en anexo 12.1). Posteriormente, se le mostró este prototipo a 3 potenciales usuarios finales y se les pidió su retroalimentación respecto a cada pantalla. A continuación se listan las recomendaciones o solicitudes realizadas, organizadas por pantalla.

6.4.2. Retroalimentación de prototipo por pantalla

Registro

Los usuarios encontraron esta pantalla bastante sencilla y entendible. Sin embargo, comentaron que se veía muy vacía y poco llamativa.

Inicio de sesión

Al ser bastante similar a la pantalla de registro, la retroalimentación respecto a esta pantalla fue casi igual a la pantalla de registro. Los usuarios comentaron sobre lo vacía y poco llamativa que era la página.

Tutorial

No hubo comentarios respecto a esta pantalla.

Prueba de diagnóstico inicial

Para esta pantalla, los usuarios proveyeron comentarios notables: primero, que la lectura podría tener un fondo de otro color debido a que el que no lo tuviera lo hacía difícil de leer; segundo, que la pantalla de las preguntas resultaba abrumadora, debido que todas las preguntas se muestran al mismo tiempo; tercero, que esperaban que se les mostraran los resultados de la prueba al concluir, y no ocurrió.

Dashboard

No hubo comentarios respecto a esta pantalla.

Semana

Para esta pantalla, los usuarios comentaron que encontraban la pantalla muy llena, es decir, estaban abrumados por la cantidad de elementos y opciones que se les presentaron. Algunos de ellos sugirieron que solo se mostraran los ejercicios del día actual y los que ya han realizado en lugar de mostrar todos.

Ejercicio

Para esta pantalla los usuarios no tuvieron comentarios respecto al diseño. Sin embargo, comentaron que se debería añadir un cronómetro para los ejercicios en la misma pantalla, puesto que medir el tiempo por su cuenta les pareció ineficiente.

Prueba de diagnóstico final

La retroalimentación de esta pantalla fue muy similar a la de la pantalla de prueba de diagnóstico inicial con la excepción de que no comentaron sobre ver sus resultados. Esto debido a que se les mostró la pantalla de resultados inmediatamente después de terminar la prueba.

Resultados

Para esta pantalla, los usuarios comentaron sobre posiblemente simplificar la explicación de los porcentajes que muestran su mejora, ya que los encontraron difíciles de comprender.

Además, mencionaron que la intensidad de los colores rojo y verde en la pantalla resultaba incómoda a la vista.

La retroalimentación proporcionada por los usuarios de prueba se tomó en consideración para el desarrollo de la aplicación web. La mayoría de los comentarios y sugerencias tuvieron efecto en la versión final de la aplicación (que se presenta más adelante). Esto fue un factor determinante para que los usuarios encontraran la aplicación fácil de usar, intuitiva, y amigable.

6.5. Selección de tecnologías a utilizar

Una vez que se estableció el diseño básico de la aplicación, se prosiguió a escoger las tecnologías que se utilizarían para desarrollar la aplicación. Específicamente, para el *backend* y el *frontend*.

Backend

El *backend* de la aplicación fue desarrollado por Michael Chan, estudiante de Ciencia de la Computación y Tecnologías de la Información de la Universidad del Valle de Guatemala. La selección de tecnologías para esta parte de la aplicación se dejó a su elección.

Para el desarrollo del *backend* se utilizaron tres herramientas distintas. Para el manejo de bases de datos se utilizó el gestor de bases de datos MySQL, que se caracteriza por ser fácil de usar, altamente escalable, y contar con buen soporte por parte de la comunidad de desarrolladores que lo usan. Para la comunicación entre *frontend* y *backend* se construyó una API REST, utilizando PHP como lenguaje en conjunto con el *framework* Laravel. Esta decisión se tomó con base en factores como: facilidad de uso del *framework*, familiaridad con el lenguaje de programación, y la facilidad que ofrece Laravel para escalar aplicaciones cuando sea necesario.

Frontend

La selección de las tecnologías para desarrollar *frontend* se dividió en 6 partes. A continuación se lista cada una de ellas.

Framework: Para la elección de qué *framework* utilizar, se hizo una comparación entre los tres *frameworks* investigados tomando en cuenta factores como: facilidad de uso del *framework*, curva de aprendizaje, adaptabilidad a las necesidades de la aplicación, soporte y actualizaciones, y tamaño de la comunidad.

React cuenta con una sintaxis declarativa, lo cual lo hace más fácil de usar que Angular. Aunado a esto, se tenía un amplio conocimiento y varios años de experiencia utilizando React, lo que representa una curva de aprendizaje menor que las otras dos alternativas. Tanto React como Vue están diseñados para crear aplicaciones relativamente sencillas. Sin embargo, al ser mantenido por Facebook, React cuenta con mejor soporte y actualizaciones.

Finalmente, de las tres alternativas, React es la que tiene la comunidad más grande. Por estas razones, se decidió utilizar React para el desarrollo del *frontend* de la aplicación.

Manejo de estado: Una vez que se escogió el *framework*, la elección de la librería de manejo de estado se simplificó significativamente, puesto que se tuvo que escoger la librería diseñada para el *framework* escogido previamente (React). Por lo tanto, la librería seleccionada para el manejo de estado fue Redux.

Manejo de efectos secundarios: Al igual que con la elección de la librería de manejo de estado, la elección de la librería de manejo de efectos secundarios se basó en la elección del *framework* seleccionado para el desarrollo del *frontend* (React). Por lo tanto, la librería seleccionada para el manejo de efectos secundarios fue Redux-Saga.

Manejo de estilos: Para la elección del manejo de estilos, se hizo una comparación entre las diferentes técnicas y herramientas investigadas previamente, tomando en cuenta factores como: complejidad, mantenibilidad, y eficiencia en creación de código.

Sass está diseñado para funcionar de forma muy similar a las hojas de estilos convencionales, por lo que es menos complejo que una alternativa como los *styled components* (que cuentan con una sintaxis y estructura distinta). Además, Sass ofrece mejor mantenibilidad debido a la organización del código en archivos separados de la lógica de los componentes, a diferencia de los *styled components* y los estilos en línea. Finalmente, Sass permite generar código más rápido debido a la estructura de las hojas de estilos y la sintaxis similar a la ofrecida por las hojas de estilos convencionales. Por estas razones, se decidió utilizar Sass para el manejo de estilos.

Control de versiones: La elección de la tecnología para el control de versiones se basó en la popularidad, soporte, y adopción de la comunidad de desarrolladores de las plataformas disponibles. Github sobresale por encima de todas las demás herramientas de la misma índole. Por lo que se escogió como la herramienta a utilizar para el manejo del control de versiones.

Pruebas unitarias: Al igual que con la elección de las librerías de manejo de estado y del manejo de efectos secundarios, la elección de las librerías de pruebas unitarias se basó en la elección del *framework* utilizado para el desarrollo del *frontend* (React). Por lo tanto, las librerías seleccionadas para la realización de pruebas unitarias fueron Jest y Enzyme.

6.6. Desarrollo de la aplicación

El proceso de desarrollo de la aplicación fue coordinado por María Isabel Ortiz, estudiante de Ciencia de la Computación y Tecnologías de la Información de la Universidad del Valle de Guatemala haciendo uso de la metodología SCRUM. Su planeación fue importante para garantizar la construcción de la aplicación de forma ordenada, coherente, y ágil. Además, permitió que los encargados del *backend* y el *frontend* puedan colaborar de forma eficiente en el desarrollo de la aplicación al proveer un panel de control centralizado, un canal de comunicación abierto, y un programa a seguir.

Para el desarrollo de la aplicación, se empezó por crear un repositorio en Github para el

proyecto de React (ver anexo 12.2).

Una vez que se tuvo el repositorio, se prosiguió a hacer pruebas a la API proporcionada por Michael Chan, explorando las rutas expuestas en la API y enviando solicitudes para poder generar y estudiar las respuestas enviadas.

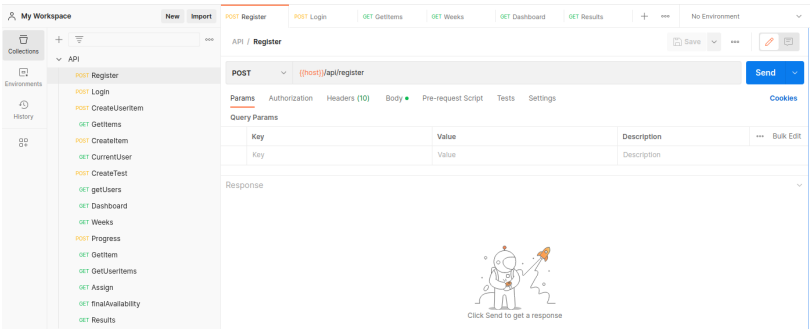


Figura 38: Inicio de postman con las rutas expuestas por la API en el panel izquierdo

(elaboración propia)

Luego, se inició el proyecto de React y se descargaron las librerías necesarias para el desarrollo de la aplicación como Redux y Redux-Saga.

Con base en las respuestas obtenidas de parte de la API, se diagramó el estado de la aplicación, definiendo los diferentes valores que tendría este objeto, por ejemplo: una lista con las diferentes semanas de la aplicación, o un objeto que representaría al ejercicio actual que el usuario estuviera realizando.

Con el estado ya diagramado, se prosiguió a desarrollar las características de la aplicación. Para cada característica se siguió el mismo procedimiento, descrito a continuación.

Primero, se programó la lógica de Redux asociada a la característica. Esto incluye los tipos, las acciones (funciones que representan la intención de alterar el estado), los reductores (funciones encargadas de actualizar el estado de la aplicación) y las sagas (funciones encargadas de manejar los efectos secundarios) asociadas. La Figura 39 muestra cómo interactúan entre sí estos cuatro elementos.

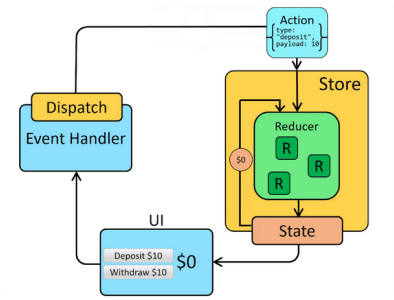


Figura 39: Flujo de datos de Redux

(Redux, 2023)

Segundo, se programaron los componentes (las unidades mínimas de funcionalidad de la aplicación) asociadas a la característica en conjunto con los estilos asociados a ellos. Cabe mencionar que los componentes no están conectados al estado de la aplicación, estos simplemente reciben atributos de sus componentes padres y los renderizan. Entre los componentes programados se pueden mencionar: botones, tarjetas, contadores, cronómetros, preguntas, entre otros. La Figura 40 muestra un ejemplo de un componente de la aplicación.

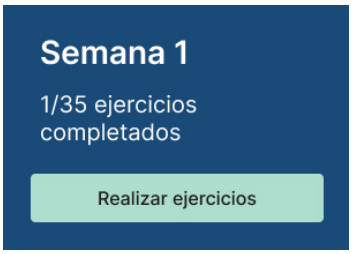


Figura 40: Componente "módulo semana" de la aplicación

(elaboración propia)

Tercero, se desarrollaron las pantallas asociadas a la característica incorporando la lógica de Redux y los componentes previamente desarrollados. Además, para cada pantalla se creó la conexión con el estado de la aplicación para mantener la información consistente entre pantallas. Entre las pantallas programadas se pueden mencionar: registro, inicio de sesión, *dashboard*, semana, entre otras. La Figura 41 muestra un ejemplo de una pantalla de la aplicación.

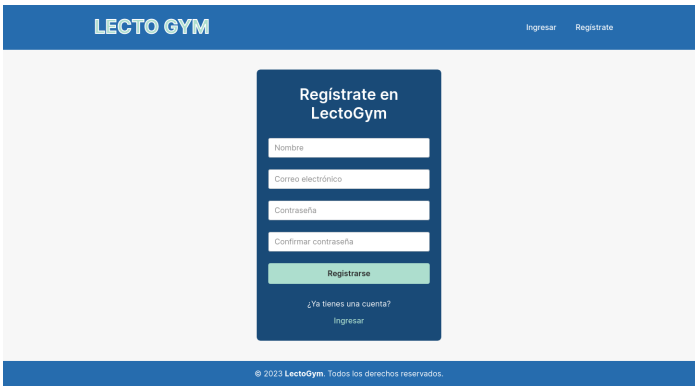


Figura 41: Pantalla de registro de la aplicación

(elaboración propia)

Una vez que se terminó de desarrollar todas las características de la aplicación, se añadieron las pruebas unitarias para garantizar su correcto funcionamiento. Las pruebas unitarias se enfocaron en tres áreas de la aplicación: componentes, para garantizar la renderización adecuada de la interfaz de usuario; lógica de Redux, para garantizar la creación y actualización correcta del estado de la aplicación; y funciones útiles, para garantizar el retorno de valores correctos con cada llamada. Para la aplicación completa, se buscó tener una cobertura mayor a 80 % en instrucciones, ramas, y líneas. Aunado a esto, se buscó tener una cobertura mayor a 60 % en funciones. La diferencia entre el porcentaje de cobertura para funciones y el resto de métricas se debe a que la mayoría del código en React son funciones

que usualmente no vale la pena probar debido a que son parte del código base.

6.7. Control de versiones

Para el control de versiones del proyecto se partió de una *branch* principal en el repositorio de Github. Para cada una de las características de la aplicación, se creó una nueva *branch* a partir de la *branch* principal. A esta nueva *branch* se agregaron *commits* que representaban un pequeño avance en el código de la característica. Luego de completar todos los cambios necesarios en la *branch*, se creó una *pull request* para incorporar los cambios de la nueva *branch* a la *branch principal*. A la *branch principal* se añadió una restricción para que todas las *pull requests* deban ser revisadas por un segundo desarrollador antes de ser incorporadas a la *branch* principal.

6.8. Integración continua

Para la integración continua, se desarrolló un *workflow* de Github que se ejecuta de forma automática cada vez que se realiza un *pull request* hacia la branch principal. Este *workflow* ejecuta las pruebas unitarias, calcula la cobertura, y verifica que las instrucciones, ramas, y líneas de código cuenten con un porcentaje de cobertura mayor a 80 % además de que las funciones cuenten con un porcentaje de cobertura mayor a 60 %. Además, bloquea el poder incorporar los cambios a la *branch* principal si estos requisitos no se cumplen.

En la Figura 42 se aprecian las dos restricciones aplicadas a la *branch* principal (revisión por otro desarrollador y requisitos de cobertura). Se puede ver que aunque los requisitos de cobertura se han cumplido (ícono verde), la *pull request* aún necesita ser revisada por un segundo desarrollador (ícono rojo). Por lo tanto, la *branch* asociada a esta *pull request* no se puede incorporar a la *branch* principal.

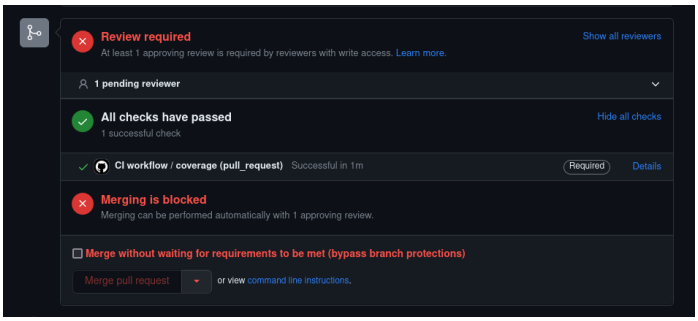


Figura 42: *Pull request* con requerimientos sin aprobar

(elaboración propia)

En la Figura 43, se puede observar una *pull request* que ha cumplido ambos requisitos para que la *branch* asociada pueda ser incorporada a la *branch* principal.

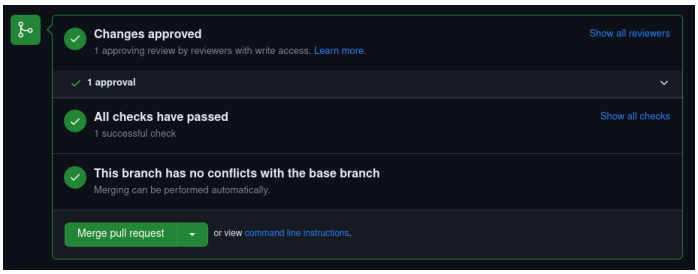


Figura 43: *Pull request* con todos los requerimientos aprobados

(elaboración propia)

6.9. Despliegue continuo

Para el despliegue continuo, se desarrolló un *workflow* de Github que se ejecuta de forma automática cada vez que se incorporan cambios en la *branch* principal. Este *workflow publica* los archivos de la aplicación a Github *pages* con la URL https://virtualmonkey.github.io/lecto-gym. En la Figura 44 se puede observar el *workflow* en funcionamiento luego de incorporar cambios a la *branch* principal.

Figura 44: *Workflow* de despliegue continuo ejecutado correctamente

(elaboración propia)

6.10. Pruebas a usuarios y recolección de datos

Para las pruebas a usuarios se seleccionó una muestra de 10 estudiantes de nivel medio entre 13 y 17 años. A los padres de cada estudiante se les solicitó la colaboración de sus hijos mediante una carta de autorización. Una vez que los padres firmaron la carta (ver ejemplo de la carta firmada en anexo 12.13), se le proveyó a cada uno de los estudiantes un usuario y contraseña para ingresar a la aplicación y se les solicitó empezar a utilizarla.

El proceso de cada usuario inició con realizar la prueba de diagnóstico inicial. Luego, se les presentaron todos los ejercicios que debían realizar. Se buscó que los estudiantes realizaran 5 ejercicios por día, 7 días a la semana, por 2 semanas consecutivas (por cuestiones de tiempo, se decidió acortar la duración del programa dos semanas). Para ello, se tuvo contacto directo con ellos con el fin de recordarles sobre la realización de los ejercicios a diario. Una vez que culminaron todos los ejercicios, realizaron la prueba de diagnóstico final y se calcularon sus resultados.

CAPÍTULO 7

Resultados

7.1. Rapidez de lectura y comprensión lectora en estudiantes

Luego de que los usuarios realizaran la prueba de diagnóstico final, se calculó el porcentaje de mejora (cantidad de aumento dividida la cantidad inicial, multiplicado por cien) en su rapidez de lectura y comprensión de lectura. Para ello, se compararon los resultados en palabras leídas por minuto y porcentaje de respuestas correctas obtenidos en la prueba de diagnóstico inicial con los obtenidos en la prueba de diagnóstico final. Estos resultados se extrajeron directamente de la pantalla de resultados de cada usuario en la aplicación (ver anexos 12.3 al 12.12).

El Cuadro 4 muestra información de cada usuario de la aplicación, en conjunto con los porcentajes de respuestas correctas (%) y palabras por minuto (PPM) en las pruebas de diagnóstico inicial y final. Además, muestra el porcentaje de mejora en comprensión lectora y en palabras leídas por minuto para cada usuario.

A continuación se provee una descripción de cada una de las columnas presentadas en el Cuadro 4:

- **Columna 1:** No. de estudiante
- **Columna 2:** Nombre de usuario
- **Columna 3:** Edad (años)
- **Columna 4:** % de respuestas correctas en prueba de diagnóstico inicial
- **Columna 5:** PPM en prueba de diagnóstico inicial
- **Columna 6:** % de respuestas correctas en prueba de diagnóstico final

- **Columna 7:** PPM en prueba de diagnóstico final
- **Columna 8:** % de mejora en comprensión lectora
- **Columna 9:** % de mejora en palabras por minuto

Columna 1	Columna 2	Columna 3	Columna 4	Columna 5	Columna 6	Columna 7	Columna 8	Columna 9
1	usuario 1	17	67	237	89	308	32.84	13.08
2	usuario 2	17	78	211	89	229	14.10	8.53
3	usuario 3	17	56	130	78	173	39.29	33.08
4	usuario 4	16	78	216	100	219	28.21	1.39
5	usuario 5	13	56	176	89	189	58.93	7.39
6	usuario 6	17	100	202	100	215	0	6.44
7	usuario 7	15	78	201	89	233	14.10	15.92
8	usuario 8	16	78	209	100	226	28.21	8.13
9	usuario 9	17	67	194	89	204	32.84	5.15
10	usuario 10	15	78	183	78	191	0	4.37

Cuadro 4: Resultados de porcentaje de comprensión lectora y palabras leídas por minuto en pruebas de diagnóstico inicial y final, y porcentajes de mejora en ambas métricas, por estudiante

(elaboración propia)

7.2. Ejercicios para mejora de rapidez de lectura y comprensión lectora

Con ayuda de Christian Porras, se diseñaron 5 ejercicios interactivos para ayudar a los estudiantes a mejorar su rapidez de lectura y comprensión lectora. Todos los ejercicios están basados en un ejercicio o técnica existente que ha probado ser eficaz en la mejora de estas dos habilidades. Además, cada ejercicio tiene 4 variaciones, una para cada semana

del programa. Cada variación incrementa la dificultad del ejercicio al variar su estructura (ejercicio 1), contenido (ejercicios 2, 3, y 4), o la complejidad del texto (ejercicio 5). A continuación se presentan los ejercicios diseñados y sus variantes.

7.2.1. Ejercicio 1: seguimiento de flechas

Basado en el ejercicio de entrenamiento de movimiento de ojos (Solan et al., 2001) y en la técnica de *finger-pacing* (Cho y Ahn, 2014), este ejercicio consiste en presentar al estudiante una serie de líneas negras paralelas organizadas verticalmente. Luego, se le pide que colocando su dedo índice sobre las líneas negras, deslice sus ojos sin mover la cabeza siguiendo la dirección de las flechas y recorra las líneas tantas veces como pueda en 60 segundos. Finalmente, se le solicita que anote la cantidad de veces que recorrió todas las líneas en ese período.

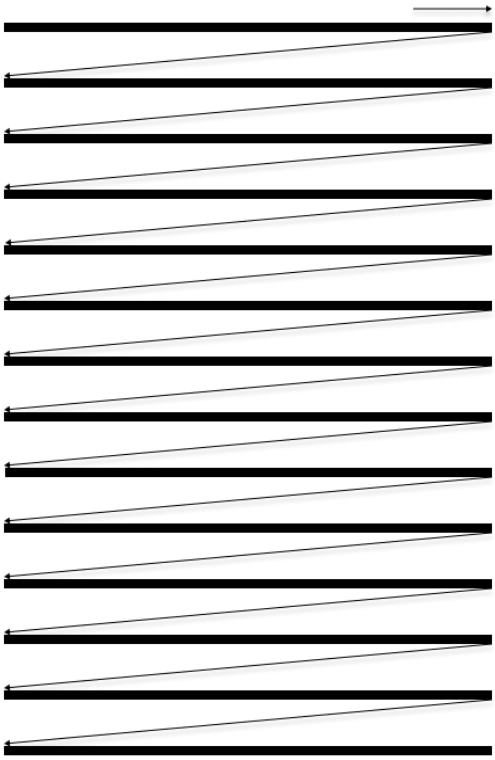


Figura 45: Contenido del ejercicio 1, primera variante

(elaboración propia)

Este ejercicio incrementa su dificultad al variar la estructura en la que se presentan las líneas. En la Figura 46 se puede apreciar cómo las primeras dos variantes del ejercicio

presentan una sola columna de líneas, mientras que en las últimas dos semanas se presentan tres columnas, las primeras dos de estas con líneas más cortas y juntas entre sí.

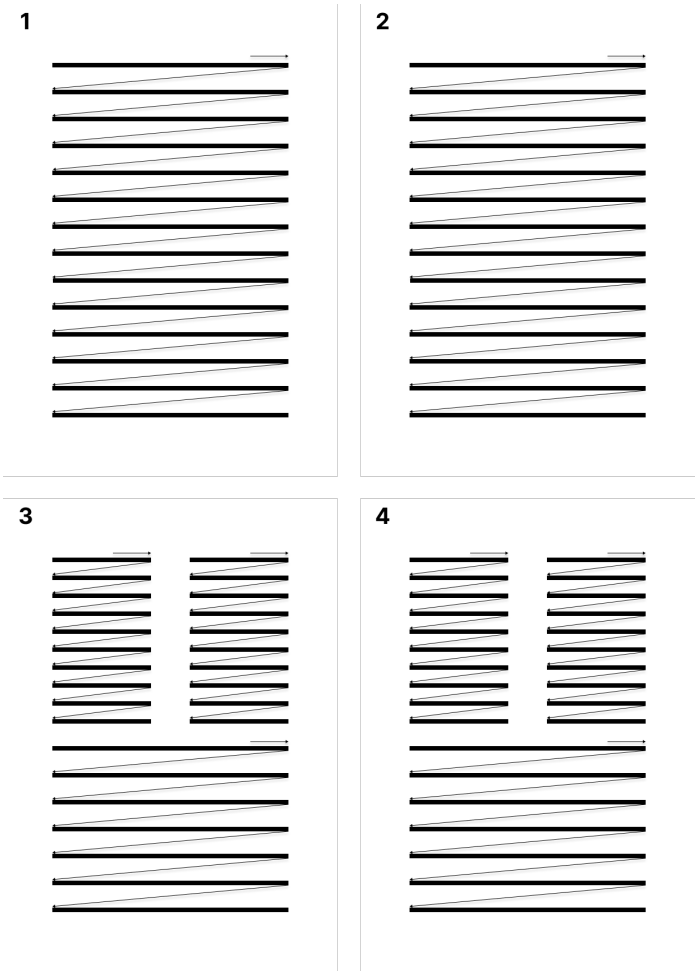


Figura 46: 4 variantes del ejercicio 1, etiquetadas con el número de semana a la que corresponde cada una

(elaboración propia)

7.2.2. Ejercicio 2: percepción de palabras

Basado en la lectura con taquistoscopio (Nist y Simpson, 1990), este ejercicio consiste en presentar al estudiante una serie de columnas de palabras organizadas verticalmente. Luego, se le pide que se ayude con el cursor para escanear cada una de las palabras en las columnas de arriba a abajo, empezando por la columna que se encuentra más a la izquierda. Finalmente, se le solicita que anote la cantidad de columnas que logró completar en 60 segundos.

Figura 47: Contenido del ejercicio 2, primera variante

(elaboración propia)

Este ejercicio incrementa su dificultad al variar las palabras en cada columna. En la Figura 48 se puede apreciar cómo la primera variante del ejercicio tiene palabras cortas. La segunda variante tiene palabras más largas y complejas. La tercera variante mezcla palabras cortas y largas ordenadas por longitud de forma ascendente y descendente, alternando entre estas dos. La cuarta y última variante mezcla palabras cortas y largas pero varía el patrón de ordenamiento al empezar con palabras cortas, usar palabras largas en medio, y terminar con palabras cortas.

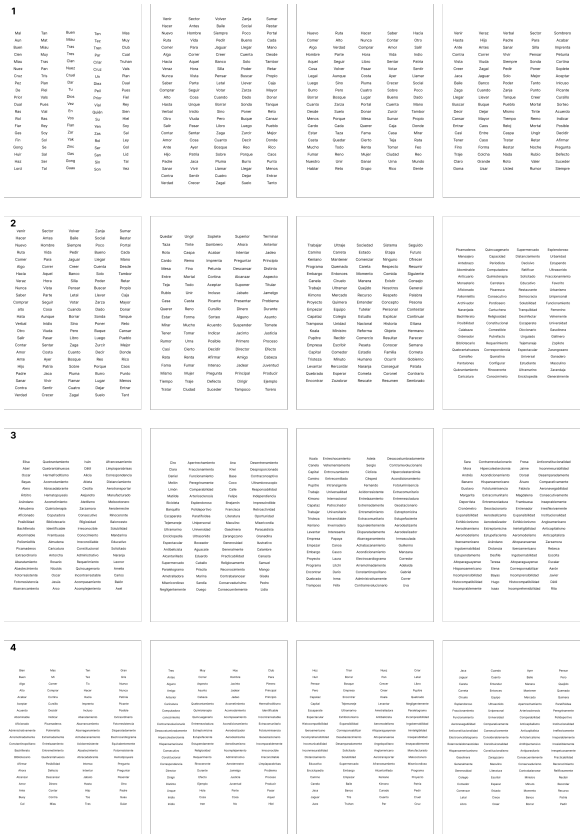


Figura 48: 4 variantes del ejercicio 2 etiquetadas con el número de semana a la que corresponde cada una

(elaboración propia)

7.2.3. Ejercicio 3: encontrar palabras

Basado en el ejercicio de reconocimiento rápido de palabras (Schneps et al., 2013), este ejercicio consiste en presentar al estudiante una serie de filas con palabras, cada fila tiene del lado izquierdo una palabra y del lado derecho un bloque de palabras aleatorias en las que hay algunas instancias o palabras relacionadas a la palabra del lado izquierdo. Luego, se le pide que para cada palabra busque en el bloque correspondiente de la derecha todas las ocurrencias de dicha palabra o las palabras relacionadas lo más rápido posible. Finalmente, se le solicita que anote cuánto tardó en encontrar todas las palabras en su bloque correspondiente.

Computadora	conocimiento, acomodamiento, computadora, palabras, maremoto, festival, espectacular, universal, escritorio, mar, lluvia, universal, computadora, risa, maremoto, computadora, pared, agenda, papel, maremoto, impresora, caer, lapicero, espectacular, vaso, universal, espectacular, celular, zapatos, tierno.
Maremoto	conocimiento, acomodamiento, computadora, palabras, maremoto, festival, espectacular, universal, escritorio, mar, lluvia, universal, computadora, risa, maremoto, computadora, pared, agenda, papel, maremoto, impresora, caer, lapicero, espectacular, vaso, universal, espectacular, celular, zapatos, tierno.
Espectacular	conocimiento, acomodamiento, computadora, palabras, maremoto, festival, espectacular, universal, escritorio, mar, lluvia, universal, computadora, risa, maremoto, computadora, pared, agenda, papel, maremoto, impresora, caer, lapicero, espectacular, vaso, universal, espectacular, celular, zapatos, tierno.
Universal	conocimiento, acomodamiento, computadora, palabras, maremoto, festival, espectacular, universal, escritorio, mar, lluvia, universal, computadora, risa, maremoto, computadora, pared, agenda, papel, maremoto, impresora, caer, lapicero, espectacular, vaso, universal, espectacular, celular, zapatos, tierno.
Abominable	favorito, patio, gradas, plantas, flores, abominable, carro, gasolinera, lluvia, llantas, instrumento, abominable, favorito, caricatura, bachillerato, columna, instrumento, favorito, horizontal, abominable, gasolinera, favorito, lapicero, abominable, tapón, instrumento, cable, impresora, silla, mesa, instrumento,
Instrumento	favorito, patio, gradas, plantas, flores, abominable, carro, gasolinera, lluvia, llantas, instrumento, abominable, favorito, caricatura, bachillerato, columna, instrumento, favorito, horizontal, abominable, gasolinera, favorito, lapicero, abominable, tapón, instrumento, cable, impresora, silla, mesa, instrumento,
Gasolinero	favorito, patio, gradas, plantas, flores, abominable, carro, gasolinera, lluvia, llantas, instrumento, abominable, favorito, caricatura, bachillerato, columna, instrumento, favorito, horizontal, abominable, gasolinera, favorito, lapicero, abominable, tapón, instrumento, cable, impresora, silla, mesa, instrumento,

Figura 49: Contenido del ejercicio 3, primera variante

(elaboración propia)

Este ejercicio incrementa su dificultad al variar las palabras a encontrar y las palabras en el bloque donde hay que encontrar dichas palabras. En la Figura 50 se puede apreciar que las palabras a encontrar y las palabras en los bloques de texto son distintas en cada variante.

Figura 50: 4 variantes del ejercicio 3 etiquetadas con el número de semana a la que corresponde cada una

(elaboración propia)

7.2.4. Ejercicio 4: encontrar antónimos

Basado en el ejercicio de reconocimiento rápido de palabras (Schneps et al., 2013), este ejercicio consiste en presentar al estudiante una serie de filas con palabras, cada fila tiene del lado izquierdo una palabra y del lado derecho un bloque de palabras aleatorias en las que hay algunas instancias de antónimos correspondientes a la palabra del lado izquierdo. Luego, se le pide que para cada palabra busque en el bloque correspondiente de la derecha todas las ocurrencias de los antónimos de dicha palabra lo más rápido posible. Finalmente, se le solicita que anote cuánto tardó en encontrar todas las palabras en su bloque correspondiente.

Cerca	extremo, lejano, sentencia, gigante, huérfano, distante, opaco sucio, libro, foco, nervioso, lejos, biblioteca, celular, luminoso
Pequeño	gigantesco, ganador, limpio, impaciente, famoso, lejos, grande, cantar, automóvil, distante, gigante, canción, sencillo, enorme.
Brillante	impóluto, comparar, sucio, nervioso, invierno, música, enorme, aseado, oscuro, abuelos, interesante, bachiller, caballeroso.
Elegante	fachudo, impaciente, lejano, trapajoso, obscuro, limpio, número, libro, diminuto, zaparastroso, nervioso, victoriasos, distante, hermoso, mamaracho
Perder	luminoso, éxito, biblioteca, victoria, nervioso, foco, libro, sucio, triunfar, dispensable, huérfano, ganar, sentencia, lejano, extremo.
Sucio	conocido, impóluto, canción, desesperado, distante, automóvil, cantar, brillante, lejos, famoso, impaciente, limpio, ganador, obscuro.
Paciente	impaciente, cabelleroso, bachiller, irritable, abuelos, árbol, aseado, enorme, música, invierno, nervioso, triunfar, comparar, impóluto.

Figura 51: Contenido del ejercicio 4, primera variante

(elaboración propia)

Al igual que el ejercicio anterior, este ejercicio incrementa su dificultad al variar las palabras de las cuales se debe encontrar el antónimo y las palabras que hay en el bloque donde hay que encontrar los antónimos. En la Figura 52 se puede apreciar que las palabras de las cuales se debe encontrar el antónimo y las palabras en los bloques de texto son distintas en cada variante.

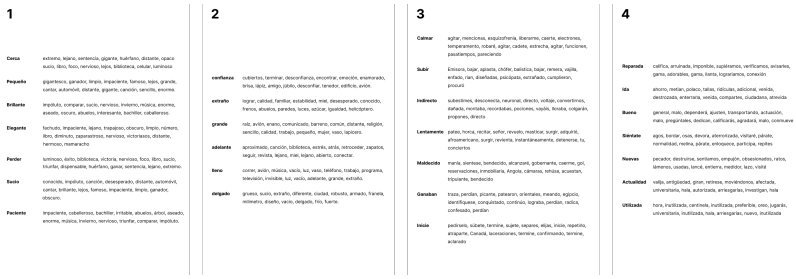


Figura 52: 4 variantes del ejercicio 4 etiquetadas con el número de semana a la que corresponde cada una

(elaboración propia)

7.2.5. Ejercicio 5: lectura

Basado en el conjunto de técnicas conocidas como lectura activa, este ejercicio consiste en presentar al estudiante una lectura corta. Luego, se le pide que lea el texto solo con los ojos y sin mover la cabeza. Finalmente, se le solicita que anote cuánto tardó en leer el texto.

INSTRUCCIONES PARA DAR CUERDA AL RELOJ

Piensa en esto: cuando te regalan un reloj te regalan un pequeño infierno florido, una cadena de rosas, un calabozo de aire. No te dan solamente el reloj, que los cumplas muy felices y esperamos que te dure porque es de buena marca, suizo con áncora de rubies; no te regalan solamente ese menudo picapedrero que te atarás a la muñeca y pasearás contigo. Te regalan -no lo saben, lo terrible es que no lo saben-, te regalan un nuevo pedazo frágil y precario de ti mismo, algo que es tuyo pero no es tu cuerpo, que hay que atar a tu cuerpo con su correa como un bracito desesperado colgándose de tu muñeca.

Te regalan la necesidad de darle cuerda todos los días, la obligación de darle cuerda para que siga siendo un reloj; te regalan la obsesión de atender a la hora exacta en las vitrinas de las joyerías, en el anuncio por la radio, en el servicio telefónico. Te regalan el miedo de perderlo, de que te lo roben, de que se te caiga al suelo y se rompa. Te regalan su marca, y la seguridad de que es uma marca mejor que las otras, te regalan la tendencia de comparar tu reloj con los demás relojes. No te regalan un reloj, tú eres el regalado, a ti te ofrecen para el cumpleaños del reloj.

Julio Cortázar, Cuentos de Cortázar

Figura 53: Contenido del ejercicio 5, primera variante

(elaboración propia)

Este ejercicio incrementa su dificultad al variar la complejidad y longitud del texto a leer. En la Figura 54 se puede apreciar cómo las variantes van aumentando en longitud y en complejidad en cada una de las semanas.

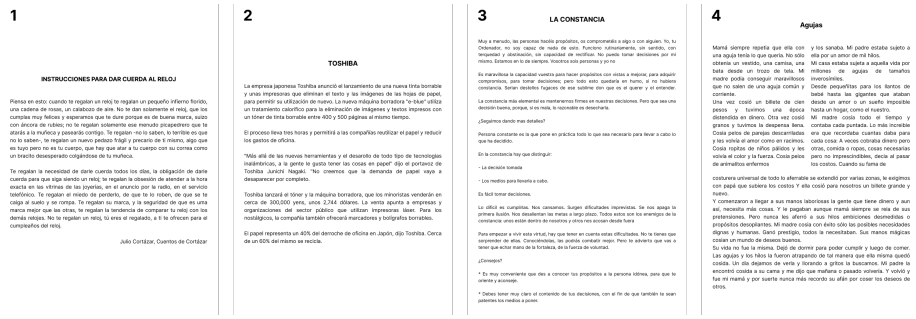


Figura 54: 4 variantes del ejercicio 5 etiquetadas con el número de semana a la que corresponde cada una

(elaboración propia)

7.3. Aplicación Web

Se diseñó y desarrolló el *frontend* de una aplicación web con tareas y retos interactivos que permitieran a los usuarios mejorar su rapidez de lectura y comprensión lectora.

El diseño del *frontend* de la aplicación fue realizado utilizando principios de teoría del color como la psicología del color, el círculo cromático, y esquemas de colores. Además, se utilizaron principios de Gestalt como el principio de proximidad, el principio de cierre, y el principio de similaridad, entre otros. Aunado a esto, se consideraron buenas prácticas de experiencia de usuario como proveer mensajes de error claros y concisos, utilizar lenguaje sencillo, y proveer instrucciones claras sobre cómo utilizar la aplicación. Finalmente, se tomaron en consideración las recomendaciones y solicitudes de parte de los usuarios de prueba para mejorar la aplicación al hacerla más sencilla, comprensible, y orientada al usuario.

El desarrollo del *frontend* de la aplicación se realizó utilizando React como *framework*, Redux para el manejo de estado, Redux-Saga para el manejo de efectos secundarios, y Sass para el manejo de estilos. Además de contar con pantallas estándar de una aplicación web como las de registro, inicio de sesión, y *dashboard*, la aplicación cuenta con pantallas que permiten a los usuarios mejorar el estado de sus habilidades de lectura. A continuación se muestran las pantallas y sus funcionalidades:

7.3.1. Registro

Esta pantalla solicita a un visitante ingresar su nombre, correo electrónico, contraseña, y confirmación de contraseña para poder crear una cuenta. Una vez que el usuario llena los campos, la aplicación realiza una validación de estos. Esta revisa que todos los campos estén llenos, que el correo electrónico sea válido, que las contraseña contenga un número, una letra, y más de 8 caracteres, y que las contraseñas coincidan. Si hay algún error en algún campo, se le da retroalimentación al usuario indicando el campo y el tipo de error. Si no hay ningún error, se envía una solicitud al API para registrar un nuevo usuario. Si se crea

el usuario exitosamente, se redirige al usuario autenticado a la pantalla “tutorial”, en caso contrario se muestra un error.

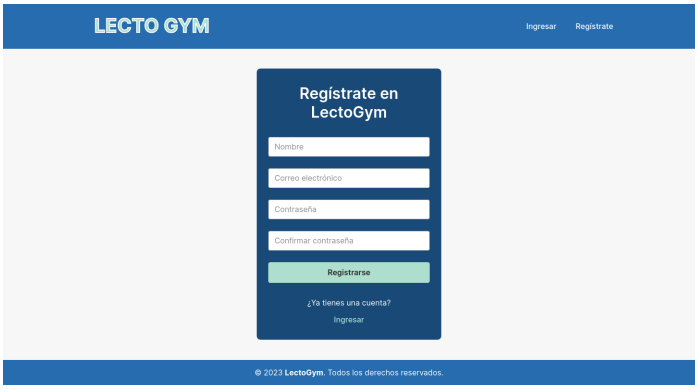


Figura 55: Pantalla “registro”

(elaboración propia)

7.3.2. Inicio de sesión

Esta pantalla solicita a un visitante ingresar sus credenciales (correo y contraseña) para iniciar sesión en la aplicación. Una vez que el usuario llena los campos, la aplicación realiza una validación de estos. Luego, revisa que todos los campos estén llenos, y que el correo electrónico sea válido. Si hay algún error en algún campo, se le da retroalimentación al usuario indicando el campo y el tipo de error. Si no hay ningún error, se envía una solicitud al API para ingresar sesión con las credenciales provistas. Si se ha iniciado sesión correctamente, hay tres posibles pantallas a las que se puede redirigir al usuario. Si no ha terminado el tutorial, se le redirige a la pantalla “tutorial”. Si ha terminado el tutorial, se le redirige a la pantalla “prueba de diagnóstico inicial”, y si ya ha terminado ambos, se le redirige a la pantalla “*dashboard*”. En caso de que haya un error, se le muestra el error.

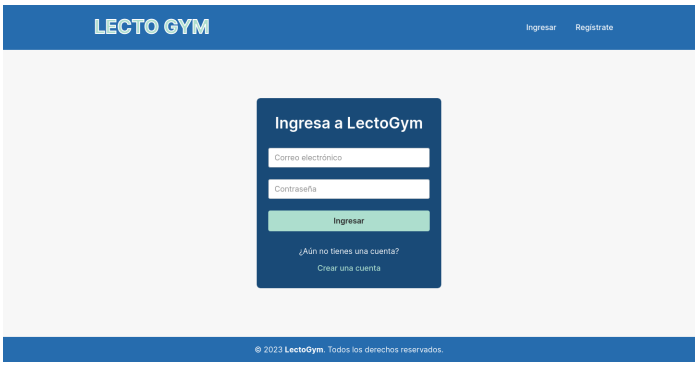


Figura 56: Pantalla “inicio de sesión”

(elaboración propia)

7.3.3. Tutorial

Esta pantalla muestra al usuario tres cosas: primero, una bienvenida a la aplicación; segundo, un video de aproximadamente 5 minutos en el que se explica el funcionamiento y las funcionalidades de la aplicación; tercero: un botón que el usuario puede cliquear una vez que haya terminado de ver el tutorial y desee continuar. Una vez que el usuario cliquea este botón, se le redirige a la pantalla "prueba de diagnóstico inicial". En caso de que el usuario cierre sesión previo a cliquear el botón, se le vuelve a mostrar la misma pantalla la siguiente vez que inicie sesión.

Figura 57: Pantalla "tutorial"

(elaboración propia)

7.3.4. Prueba de diagnóstico inicial

Esta funcionalidad cuenta con cuatro pantallas. Primero, se muestra al usuario instrucciones sobre la prueba de diagnóstico inicial y un botón para empezar la parte de la lectura de la prueba.

Figura 58: Primera pantalla de la prueba de diagnóstico inicial

(elaboración propia)

Segundo, se le presenta al usuario un cronómetro que inicia automáticamente, la lectura de la prueba de diagnóstico inicial, y un botón que debe cliquear al culminar la lectura para contestar las preguntas.

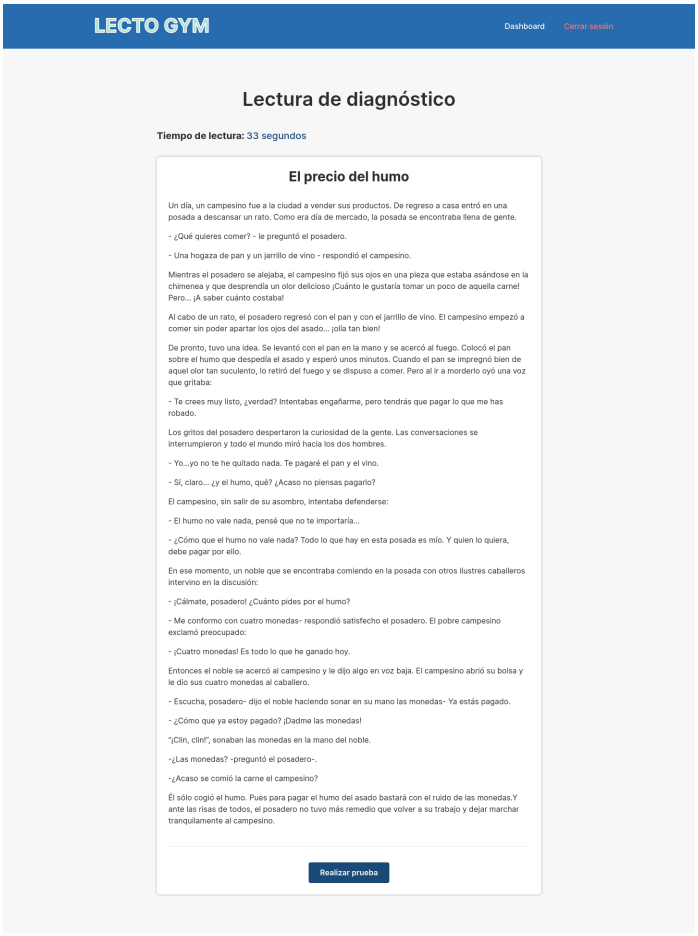


Figura 59: Segunda pantalla de la prueba de diagnóstico inicial

(elaboración propia)

Tercero, se le muestra al usuario una por una las preguntas de opción múltiple respecto a la lectura en conjunto con las posibles respuestas, además de un botón para avanzar a la siguiente pregunta o finalizar el examen cuando haya respondido todas las preguntas.

Figura 60: Tercera pantalla de prueba de diagnóstico inicial mostrando la primera pregunta

(elaboración propia)

Figura 61: Tercera pantalla de prueba de diagnóstico inicial mostrando la novena y última pregunta

(elaboración propia)

Finalmente, se le muestra al usuario los resultados de la prueba de diagnóstico inicial: porcentaje de respuestas correctas, tiempo de lectura, y palabras leídas por minuto. Además, se le presenta al usuario un botón que le permite dirigirse a su *dashboard*.

Figura 62: Cuarta pantalla de prueba de diagnóstico inicial mostrando los resultados de la prueba

(elaboración propia)

7.3.5. *Dashboard*

Esta pantalla muestra al usuario las 4 semanas de ejercicios ofrecidos por la aplicación. Cada semana puede tener uno de tres estados: en progreso, que indica que los ejercicios de esa semana aún no se han culminado; completada, que indica que los ejercicios de esa semana se han completado; y finalmente, inhabilitada, que indica que la semana no está habilitada debido a que los ejercicios de la semana anterior no se han completado. Además, cada semana contiene el progreso de los ejercicios y un botón que al ser cliqueado redirige al usuario a la pantalla "semana". Esta pantalla cuenta con validaciones que le permiten habilitar las semanas conforme el usuario termina las semanas anteriores, de manera que los usuarios no puedan adelantarse o hacer ejercicios en un orden no deseado.

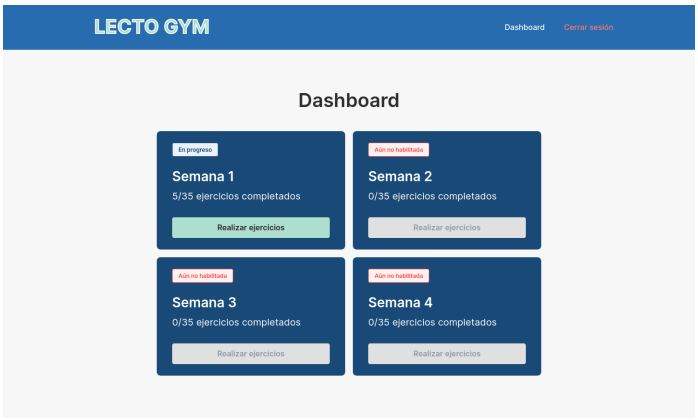


Figura 63: Pantalla "*dashboard*" mostrando diferentes estados posibles de las semanas

(elaboración propia)

7.3.6. Semana

Esta pantalla muestra al usuario los ejercicios correspondientes a una semana. Cada ejercicio contiene el nombre del ejercicio, el tipo, las herramientas a usar, y un indicador de progreso circular, que mediante colores y números le hace saber al usuario qué iteración del ejercicio ha completado (0, 1, 2, o 3). Además, cada ejercicio contiene un botón que al ser cliqueado redirige al usuario a la pantalla "ejercicio". Cabe mencionar, que los ejercicios cuentan con validaciones para que no se puedan realizar más de tres veces.

Cada día de la semana cuenta con una etiqueta de progreso que puede ser: en progreso, cuando aún no se han terminado las tres iteraciones de cada ejercicio correspondiente al día; y completado, cuando se da el caso opuesto. Aunado a esto, esta pantalla cuenta con restricciones para que al usuario solo se le muestren los días que ya ha completado, y el día que se encuentra en progreso. Esto reduce la carga visual para el usuario y le permite enfocarse en el día que se encuentra en progreso.

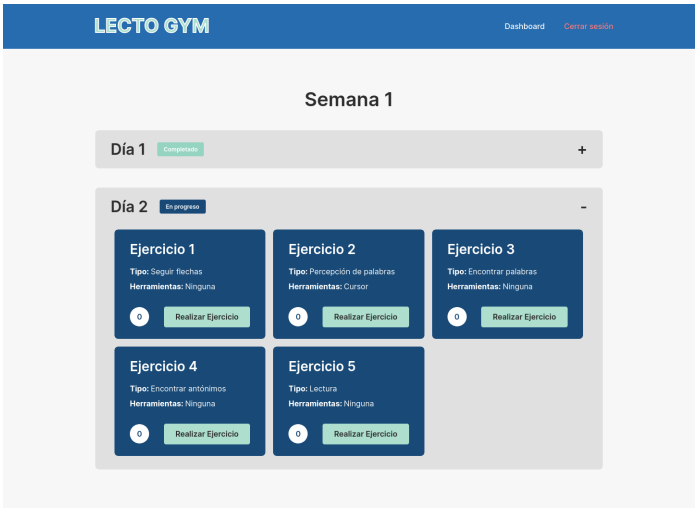


Figura 64: Pantalla "semana" mostrando días completados (colapsados) y el día en progreso (expandido)

(elaboración propia)

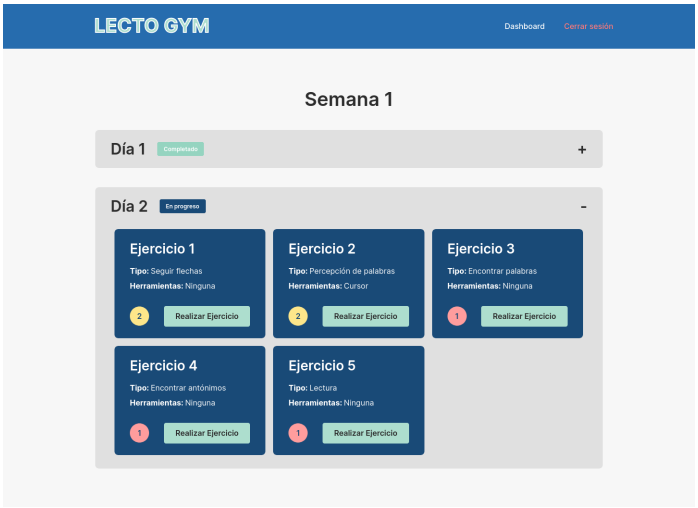


Figura 65: Pantalla "semana" mostrando diferentes indicadores de progreso en diferentes ejercicios del mismo día

(elaboración propia)

7.3.7. Ejercicio

Esta pantalla muestra al usuario un ejercicio que debe realizar. En ella, se incluyen: nombre del ejercicio, instrucciones, lista de herramientas a utilizar, un contador regresivo o cronómetro dependiendo del tipo de ejercicio, un campo para ingresar el resultado del ejercicio que también varía dependiendo del tipo de ejercicio, y un botón para finalizar el ejercicio.

Para los ejercicios "seguimiento de flechas" y "percepción de palabras", en esta pantalla

se presenta un contador regresivo con el tiempo establecido en 60 segundos (Figura 66). Además, se le solicita al usuario que llene el campo del resultado y luego cliquee el botón que se le presenta para finalizar el ejercicio.

Para los ejercicios “encontrar palabras”, “encontrar antónimos”, y “lectura”, en esta pantalla se presenta un cronómetro con controles para iniciar, detener, y reiniciar el tiempo (Figura 67). Además, el tiempo transcurrido al realizar el ejercicio se calcula automáticamente y se llena el campo del resultado con ese tiempo, por lo que solo se le solicita que cliquee el botón que se le presenta para finalizar el ejercicio.

Esta pantalla cuenta con dos tipos de validaciones: primero, no permite que el usuario finalice el ejercicio a menos que el contador regresivo haya llegado a 0 o que el cronómetro haya sido iniciado y detenido; segundo, que para los ejercicios en los que se solicita que el usuario llene el campo de resultado, este no se encuentre vacío.

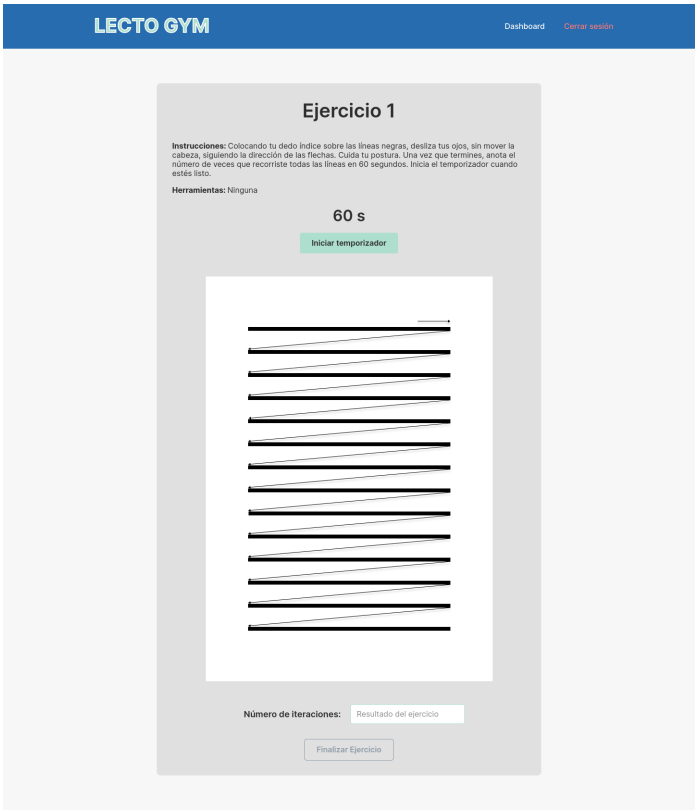


Figura 66: Pantalla “ejercicio” con contador regresivo

(elaboración propia)

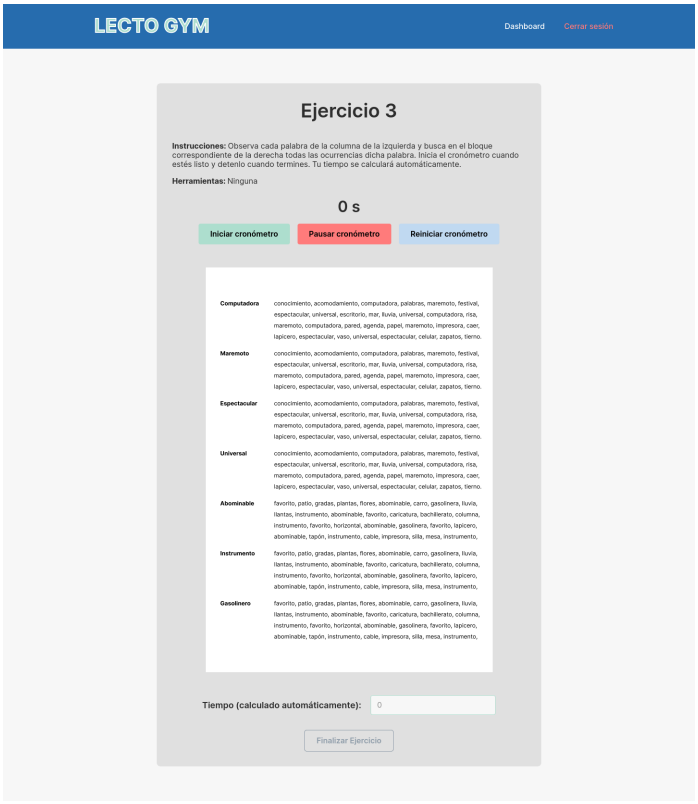


Figura 67: Pantalla "ejercicio" con cronómetro

(elaboración propia)

7.3.8. Prueba de diagnóstico final

Al igual que la prueba de diagnóstico inicial, esta funcionalidad cuenta con cuatro pantallas. Primero, se muestra al usuario instrucciones sobre la prueba de diagnóstico final y un botón para empezar la parte de la lectura de la prueba.

Figura 68: Primera pantalla de la prueba de diagnóstico final

(elaboración propia)

Segundo, se le presenta al usuario un cronómetro que inicia automáticamente, la lectura

de la prueba de diagnóstico final, y un botón que debe cliquear al culminar la lectura para contestar las preguntas.

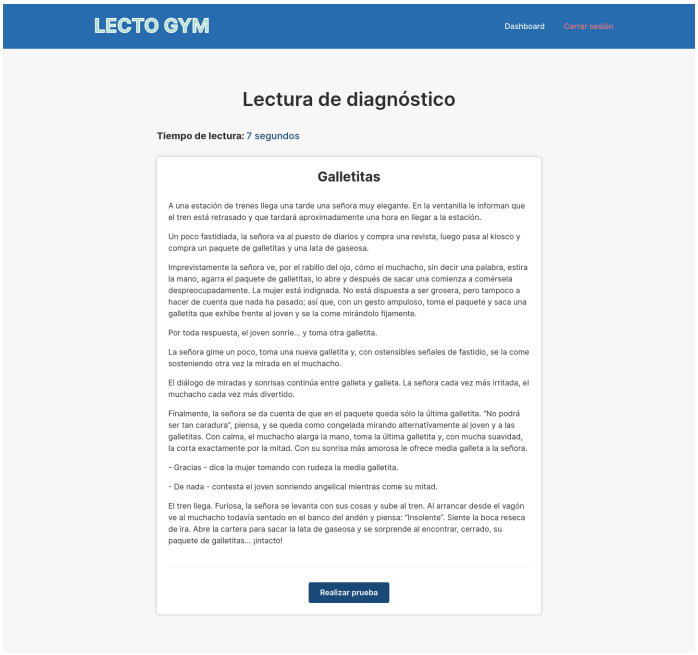


Figura 69: Segunda pantalla de la prueba de diagnóstico final

(elaboración propia)

Tercero, se le muestra al usuario una por una las preguntas de opción múltiple respecto a la lectura en conjunto con las posibles respuestas, además de un botón para avanzar a la siguiente pregunta o finalizar el examen cuando haya respondido todas las preguntas.

Figura 70: Tercera pantalla de prueba de diagnóstico final mostrando la primera pregunta

(elaboración propia)

Figura 71: Tercera pantalla de prueba de diagnóstico final mostrando la novena y última pregunta

(elaboración propia)

Finalmente, se le muestra al usuario los resultados de la prueba de diagnóstico final: porcentaje de respuestas correctas, tiempo de lectura, y palabras leídas por minuto. Además, se le presenta al usuario un botón que le permite dirigirse a la pantalla "resultados".

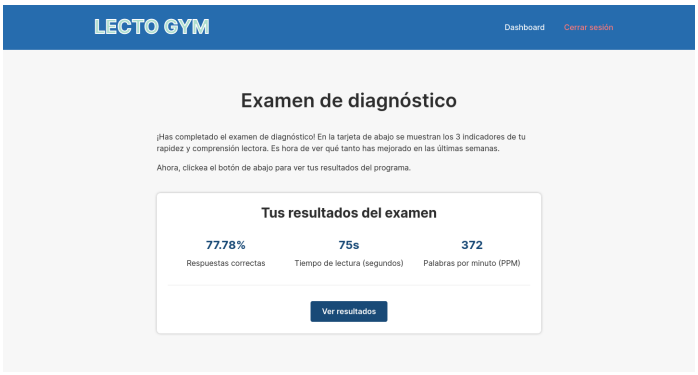


Figura 72: Cuarta pantalla de prueba de diagnóstico final mostrando los resultados de la prueba

(elaboración propia)

7.3.9. Resultados

Esta pantalla muestra al usuario tres tarjetas: la primera contiene los resultados de la prueba inicial (porcentaje de respuestas correctas, tiempo de lectura, y palabras leídas por minuto); la segunda contiene los resultados de la prueba final (porcentaje de respuestas correctas, tiempo de lectura, y palabras leídas por minuto); la tercera contiene los porcentajes de mejora de la comprensión lectora y rapidez de lectura (palabras leídas por minuto) del usuario, calculados con base en la comparación entre los resultados de las tarjetas 1 y 2. Además, esta pantalla contiene un botón que le permite al usuario regresar a su *dashboard*.

Figura 73: Pantalla "resultados"

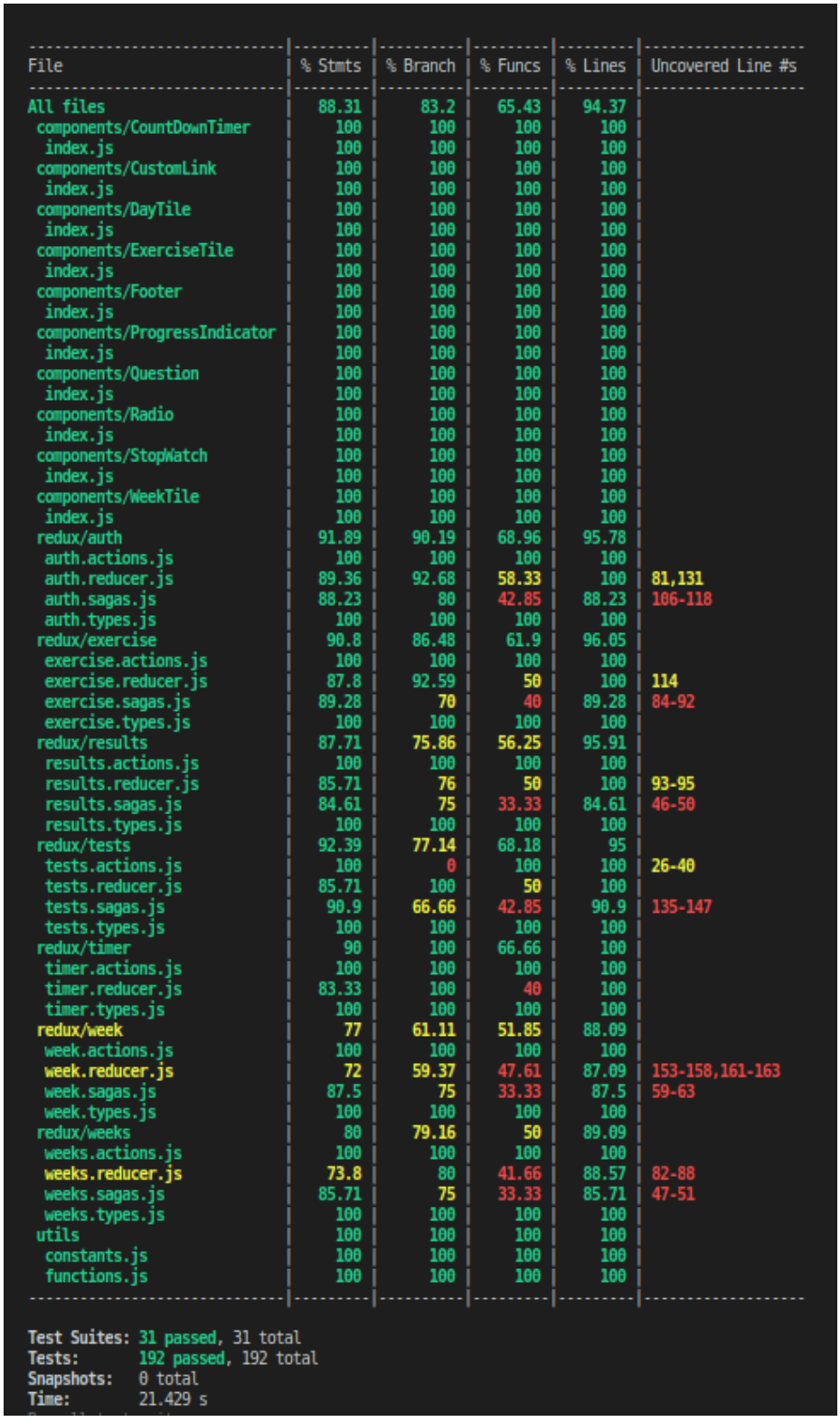
(elaboración propia)

El control de versiones para el desarrollo de la aplicación se realizó haciendo uso de Github. Con ayuda de los *workflows* y las *actions* de Github se implementaron prácticas de integración continua y despliegue continuo para automatizar la ejecución de pruebas unitarias y la publicación de la aplicación a un ambiente de producción.

7.4. Pruebas unitarias y cobertura

Además del diseño y desarrollo de la aplicación, se crearon pruebas unitarias para los componentes, la lógica de Redux, y las funciones de la aplicación. Basado en el estándar aceptado para aplicaciones (Rutledge, 2021), se buscó una cobertura mayor a 80 % en instrucciones, ramas, y líneas, y 60 % en funciones. Estos porcentajes de cobertura se garantizaron mediante la implementación un *workflow* de Github que evita que se incorporen cambios a la rama principal en caso de que los criterios establecidos de cobertura no se cumplan.

En la Figura 74 se puede ver el reporte de cobertura del código de la aplicación. Cada fila representa un archivo o una carpeta, exceptuando la primera fila que se refiere a todos los archivos en conjunto. En la primera columna titulada "*File*" se muestra el nombre del archivo al cual pertenecen las pruebas. En la segunda columna titulada " *% Stmts*" se muestra el porcentaje de instrucciones cubiertas para dicho archivo o carpeta. En la tercera columna titulada " *% Branch*" se muestra el porcentaje de ramas cubiertas para dicho archivo o carpeta. En la cuarta columna titulada " *% Funcs*" se muestra el porcentaje de funciones cubiertas para dicho archivo o carpeta. En la quinta columna titulada " *% Lines*" se muestra el porcentaje de líneas cubiertas para dicho archivo o carpeta. La última columna muestra las líneas que no se cubrieron para cada archivo.



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	88.31	83.2	65.43	94.37	
components/CountDownTimer	100	100	100	100	
index.js	100	100	100	100	
components/CustomLink	100	100	100	100	
index.js	100	100	100	100	
components/DayTile	100	100	100	100	
index.js	100	100	100	100	
components/ExerciseTile	100	100	100	100	
index.js	100	100	100	100	
components/Footer	100	100	100	100	
index.js	100	100	100	100	
components/ProgressIndicator	100	100	100	100	
index.js	100	100	100	100	
components/Question	100	100	100	100	
index.js	100	100	100	100	
components/Radio	100	100	100	100	
index.js	100	100	100	100	
components/StopWatch	100	100	100	100	
index.js	100	100	100	100	
components/WeekTile	100	100	100	100	
index.js	100	100	100	100	
redux/auth	91.89	90.19	68.96	95.78	
auth.actions.js	100	100	100	100	
auth.reducer.js	89.36	92.68	58.33	100	81,131
auth.sagas.js	88.23	80	42.85	88.23	106-118
auth.types.js	100	100	100	100	
redux/exercise	90.8	86.48	61.9	96.05	
exercise.actions.js	100	100	100	100	
exercise.reducer.js	87.8	92.59	50	100	114
exercise.sagas.js	89.28	70	40	89.28	84-92
exercise.types.js	100	100	100	100	
redux/results	87.71	75.86	56.25	95.91	
results.actions.js	100	100	100	100	
results.reducer.js	85.71	76	50	100	93-95
results.sagas.js	84.61	75	33.33	84.61	46-50
results.types.js	100	100	100	100	
redux/tests	92.39	77.14	68.18	95	
tests.actions.js	100	0	100	100	26-40
tests.reducer.js	85.71	100	50	100	
tests.sagas.js	90.9	66.66	42.85	90.9	135-147
tests.types.js	100	100	100	100	
redux/timer	90	100	66.66	100	
timer.actions.js	100	100	100	100	
timer.reducer.js	83.33	100	40	100	
timer.types.js	100	100	100	100	
redux/week	77	61.11	51.85	88.09	
week.actions.js	100	100	100	100	
week.reducer.js	72	59.37	47.61	87.09	153-158,161-163
week.sagas.js	87.5	75	33.33	87.5	59-63
week.types.js	100	100	100	100	
redux/weeks	80	79.16	50	89.09	
weeks.actions.js	100	100	100	100	
weeks.reducer.js	73.8	80	41.66	88.57	82-88
weeks.sagas.js	85.71	75	33.33	85.71	47-51
weeks.types.js	100	100	100	100	
utils	100	100	100	100	
constants.js	100	100	100	100	
functions.js	100	100	100	100	

Test Suites: 31 passed, 31 total
Tests: 192 passed, 192 total
Snapshots: 0 total
Time: 21.429 s

Figura 74: Reporte de cobertura del código de la aplicación

(elaboración propia)

CAPÍTULO 8

Discusión de resultados

El objetivo principal planteado fue incrementar la cantidad de palabras leídas por minuto y el nivel de comprensión lectora inicial de diez estudiantes de nivel medio en Guatemala, mediante retos interactivos en una aplicación web. Para ello, se diseñaron ejercicios interactivos inspirados en ejercicios y técnicas para la mejora de la rapidez y comprensión lectora. Estos ejercicios se incorporaron en una aplicación web que se los presentó a los usuarios de forma ordenada, guiada y sencilla. Con el fin de calcular la mejora de los usuarios, se establecieron pruebas de diagnóstico para medir la rapidez de lectura, representada en palabras por minuto (Hasbrouck y Tindal, 2017), y la comprensión lectora, representada en porcentaje de respuestas correctas (Dyson y Haselgrove, 2001) antes y después de utilizar la aplicación.

El Cuadro 4 muestra, en la columna titulada " % de mejora en comprensión lectora", que ocho de los diez estudiantes mejoraron su comprensión lectora luego de utilizar la aplicación, mientras que dos mantuvieron su nivel de comprensión lectora inicial. En promedio, los estudiantes mejoraron en un 24.85 % su nivel de comprensión lectora. En el Cuadro 4 también se muestra, en la columna titulada " % de mejora en palabras por minuto", que los diez estudiantes aumentaron su cantidad de palabras leídas por minuto luego de utilizar la aplicación. En promedio, los estudiantes aumentaron en un 10.35 % la cantidad de palabras leídas por minuto. Estos resultados evidencian un incremento significativo tanto en la cantidad de palabras leídas por minuto como en el nivel de comprensión lectora de los estudiantes.

El primer objetivo específico planteado fue desarrollar retos interactivos que permitieran a estudiantes mejorar sus capacidades de lectura y comprensión lectora. Para ello, con ayuda de Christian Porras, se diseñaron cinco ejercicios interactivos basados en diferentes ejercicios y técnicas utilizados para mejorar la rapidez de lectura y comprensión de lectura. Para cada uno de estos ejercicios, se diseñaron cuatro variantes, resultando en veinte ejercicios totales. Cada variante incrementa la dificultad del ejercicio al variar su estructura (ejercicio 1),

contenido (ejercicios 2, 3, y 4) o la complejidad del texto (ejercicio 5).

El ejercicio 1, titulado "seguimiento de flechas", basado en el ejercicio de entrenamiento movimiento de ojos (Solan et al., 2001) y la técnica de *finger-pacing* (Cho y Ahn, 2014), permitió a los estudiantes mejorar su rapidez y comprensión lectora al reducir los movimientos no deseados de los ojos. El ejercicio 2, titulado "percepción de palabras", basado en la lectura con taquistoscopio (Nist y Simpson, 1990), permitió a los estudiantes mejorar su rapidez y comprensión lectora mediante la reducción de las fijaciones de los ojos al leer palabras. Los ejercicios 3 y 4, titulados "encontrar palabras" y "encontrar antónimos", basados en el ejercicio de reconocimiento rápido de palabras (Schneps et al., 2013), permitieron a los estudiantes mejorar su rapidez de lectura al entrenar sus ojos para encontrar y descartar palabras en un texto. Finalmente, el ejercicio 5, titulado "lectura", basado en el conjunto de técnicas conocidas como lectura activa, permitió a los estudiantes mejorar su comprensión lectora al proveer un medio para practicar la lectura de forma constante. Los resultados del Cuadro 4, discutidos previamente, demuestran que los ejercicios desarrollados contribuyeron a la mejora de la rapidez y comprensión lectora de los estudiantes.

El segundo objetivo específico planteado fue diseñar e implementar el *frontend* de una aplicación web que presente a los usuarios tareas interactivas que les permitan mejorar su rapidez y comprensión lectora. Se diseñó y desarrolló el *frontend* de una aplicación web que incorporó los ejercicios interactivos previamente desarrollados, con el fin de mejorar las habilidades de lectura y comprensión lectora de los usuarios.

Para el diseño del *frontend* de la aplicación se utilizaron principios de teoría del color (Hogg y Garrow, 2003; Hard, 2013; Marks, 2019), principios de Gestalt (Buley, 2010), y buenas prácticas de experiencia de usuario (Krug, 2014). Además, se tomaron en cuenta las recomendaciones y solicitudes de los usuarios de prueba para construir una aplicación sencilla, comprensible y orientada al usuario. Para el desarrollo del *frontend* de la aplicación, se utilizó React (Stefanov y Adams, 2019) como *framework* en conjunto con librerías como Redux (Redux, 2023), Redux-Saga (Redux-Saga, 2023) y Sass. La aplicación está formada por nueve pantallas: registro, inicio de sesión, tutorial, prueba de diagnóstico inicial, *dashboard*, semana, ejercicio, prueba de diagnóstico final, y resultados.

Los ejercicios interactivos fueron incorporados de manera correcta en la aplicación y los usuarios fueron capaces de utilizar, navegar y aprovechar todas las funcionalidades de esta. Esto, en conjunto con los resultados del Cuadro 4, mencionados previamente, demuestran que la aplicación contribuyó a la mejora en la rapidez y comprensión lectora de los usuarios.

El tercer objetivo específico fue generar pruebas unitarias para la aplicación, que garanticen el correcto funcionamiento de los componentes utilizados en la misma. Para ello, se crearon pruebas unitarias para los componentes, la lógica de Redux y las funciones de la aplicación. Tomando en cuenta que el estándar aceptado, pero no obligatorio, de cobertura es de 80 % o más (Rutledge, 2021), se buscó una cobertura mayor a 80 % en instrucciones, ramas y líneas, y 60 % en funciones. En la Figura 74 se puede observar que para los archivos para los cuales se desarrollaron pruebas unitarias, se alcanzó un 88.31 % de cobertura en instrucciones, un 83.2 % de cobertura en ramas, y un 94.37 % en líneas. Además, se alcanzó un 65.43 % en funciones. Estos porcentajes corresponden al estándar aceptado, comprobando que cada parte de código analizada funciona de forma correcta en diferentes escenarios.

CAPÍTULO 9

Conclusiones

Los ejercicios interactivos de la aplicación web, combinados con su *frontend*, contribuyeron a la mejora de la velocidad y comprensión lectora de diez estudiantes de nivel medio que hicieron uso de la aplicación, como lo demuestra el aumento en la cantidad de palabras leídas por minuto y el nivel de comprensión lectora de estos.

Los cinco ejercicios interactivos desarrollados y sus variantes contribuyeron a la mejora de las capacidades de rapidez de lectura y comprensión lectora de los estudiantes que los llevaron a cabo durante dos semanas.

La interfaz simple, entendible, y centrada en el usuario de la aplicación permitió que los estudiantes fueran capaces de utilizar todas las funcionalidades de esta para completar tanto las pruebas de diagnóstico final e inicial así como los ejercicios.

El uso de React como *framework* y de librerías como Redux, Redux-Saga y Sass, junto con la creación de pruebas unitarias para componentes, lógica de Redux y funciones garantizaron el correcto funcionamiento de los componentes utilizados en la aplicación web.

Los porcentajes de cobertura alcanzados en instrucciones, ramas, líneas y funciones en las pruebas unitarias correspondieron al estándar aceptado, lo que confirmó que cada pieza de código analizada funciona correctamente en diferentes escenarios.

CAPÍTULO 10

Recomendaciones

Para incrementar la validez de los resultados, se recomienda aumentar el tamaño de la muestra de estudiantes. Es posible que un tamaño de muestra mayor a diez estudiantes provea evidencia más significativa respecto a la efectividad de la aplicación en la mejora de la rapidez y comprensión lectora de los estudiantes.

Para mejorar los resultados obtenidos, se recomienda expandir la cantidad de ejercicios desarrollados. Estos nuevos ejercicios podrían enfocarse en áreas específicas de comprensión de lectura o velocidad que no se abordaron inicialmente.

Para obtener mejores datos, se recomienda someter a los estudiantes al programa por más de dos semanas. Un seguimiento mayor puede resultar en un mayor incremento de las de la rapidez y comprensión lectora de los estudiantes. Además, puede proveer datos más significativos respecto a la efectividad de la aplicación.

Para mejorar el control sobre los usuarios, se recomienda incorporar un tutor que pueda dar seguimiento y proveer retroalimentación personalizada de manera periódica a los usuarios.

Para mejorar el *frontend* de la aplicación y sus funcionalidades, se recomienda utilizar una versión web de la aplicación, o bien, una aplicación web progresiva, que incluya funcionalidades como notificaciones push, capacidad de funcionar sin internet, y capacidad de ser consultada en cualquier dispositivo.

CAPÍTULO 11

Bibliografía

1. Almutairi, N. (2018). *Effective Reading Strategies For Increasing The Reading Comprehension Level Of Third-Grade Students With Learning Disabilities*. Western Michigan University.

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

3. Andina, Z. (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.

4. Angular. (2023). *Agular Features*. https://angular.io/features.

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

6. Brown, E. (2017). *Web Development with Node and Express: Leveraging the JavaScript Stack*. O'Reilly Media.

7. Bugl, D. (2018). *Learning Redux: The Simplest Way to Manage Your State*. Packt Publishing Ltd.

8. Buley, L. (2010). *The User Experience Team of One: A Research and Design Survival Guide*. Rosenfeld Media.

9. Carlson, S., y O'Brien, E. (2017). *Strong Reading Skills in Freshman Year Associated With College Success*. UC Berkeley News. https://news.berkeley.edu/2017/09/20/strong-reading-skills-in-freshman-year-associated-with-college-success/

10. Carroll, J. M., Maughan, B., Goodman, R. y Meltzer, H. (2005). Literacy difficulties and psychiatric disorders: evidence for comorbidity. *J. Child Psychol. Psychiatry 46*.

11. Carver, R. P. (1990). *Reading rate: A review of research and theory*. San Diego: Academic Press.

12. Chacon, S. and Straub, B. (2014) *Pro Git*. Apress.

13. Chapman, C. (2023). *Explorando los principios Gestalt del diseño*. https://www.toptal.com/designers/ui/exploring-the-gestalt-principles-of-design.

14. Chinnathambi, K. (2017). *Learning Angular: A Hands-On Guide to Angular 2 and Angular 4*. O'Reilly Media.

15. Chinnathambi, K. (2018). *Learning React, Second Edition*. O'Reilly Media.

16. Cho, B. Y. y Ahn, S. (2014). The effects of finger reading and pacing on reading performance and comprehension. *Journal of Research in Reading, 37(1)*, 71-88.

17. Chowdhury, A. (2019). *React 16 essentials: Unleash the power of React 16 to build stunning modern web applications*. Packt Publishing Ltd.

18. Christie, C. y Yell, M. (2008). Preventing youth incarceration through reading remediation: issues and solutions. *Read. Writ. Quart. 24*, 148–176.

19. Cien. (2019). *El Sistema educativo en Guatemala*. https://cien.org.gt/wp-content/uploads/2019/05/Educacio%CC%81n-y-Tecnologi%CC%81a-documento-final.pdf

20. Clark-Keane, C. (2022). *8 ways to use color psychology in marketing (with examples)*. https://www.wordstream.com/blog/ws/2022/07/12/color-psychology-marketing.

21. Daniel, S., Walsh, A., Goldston, D., Arnold, E., Reboussin, B., y Wood, F. (2006). Suicidality, School Dropout, and Reading Problems Among Adolescents. *Journal of Learning Disabilities, 39*(6), 507–514.

22. Dyson, M. y Haselgrove, M. (2001). The influence of reading speed and line length on the effectiveness of reading from screen. *International Journal of Human-Computer Studies, 54*(4), 585–612.

23. Dyson, M. y Haselgrove, M. (2020). The effects of reading speed and reading patterns on the understanding of text read from screen. *Journal of Research in Reading, 23*(2).

24. Enzyme. (s.f.). *Enzyme*. https://enzymejs.github.io/enzyme/.

25. Fain, Y., y Moiseev, A. (2017). *Angular Development with Typescript*. Manning Publications.

26. Felke-Morris, T.A. (2017). *Web development & design foundations with HTML5*. Pearson.

27. Freeman, A. (2017). *Pro Angular 4*. Apress

28. Freeman, A., y Robins, M. (2018). *Head First HTML and CSS: A learner's guide to creating standards-based web pages*. O'Reilly Media.

29. Garcia Mingrone, A. (2022). *Qué es la ley de cierre de la Gestalt y ejemplos*. https://www.psicologia-online.com/que-es-la-ley-de-cierre-de-la-gestalt-y-ejemplos-6290.html.

30. García, E., Jiménez, J., González, D., y Jiménez-Suárez, E. (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.

31. Giménez, A. (2018). *Principio de continuidad - Leyes de usabilidad. Gestalt.* https://www.aunitz.net/ley-13-principio-de-continuidad/.

32. Giménez, A. (2018). *Principio de proximidad - Leyes de usabilidad. Gestalt.* https://www.aunitz.net/ley-06-principio-de-la-proximidad/.

33. GitHub Pages. (2023). *Websites for you and your projects.* https://pages.github.com/.

34. GitHub. (2023). *About workflows.* https://docs.github.com/en/actions/using-workflows/about-workflows.

35. GitHub. (2023). *Let's build from here.* https://github.com/about.

36. GitHub. (2023). *Understanding GitHub actions.* https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions.

37. GitLab. (2023). *What is version control?* https://about.gitlab.com/topics/version-control/.

38. Gómez, L. (1998). *Comprensión de textos escritos: de la teoría a la sala de clases* (1. ed.). Santiago de Chile: Andrés Bello.

39. González, J. (2016). *Comprensión Lectora en Estudiantes de Tercer Grado de Educación Primaria en Guatemala.* Tecnológico de Monterrey, Campus Ciudad de Guatemala. https://repositorio.tec.mx/bitstream/handle/11285/568771/DocsTec_6536.pdf?sequence=1.

40. Gübelin, P. (2019). *Combinación de colores.* https://www.piagubelin.com/blog/combinaci.

41. Harris, A. J., y Sipay, E. R. (1980). *How to increase reading ability: A guide to developmental and remedial methods.* Longman.

42. Hasbrouck, J. y Tindal, G. (2017). *An update to compiled ORF norms (Technical Report No. 1702).* Eugene, OR, Behavioral Research and Teaching, Universidad de Oregon.

43. Hogg, M. A., y Garrow, J. S. (2003). The psychology of color symbolism. Journal of Fashion Marketing and Management. *An International Journal, 7*(3), 205-216.

44. Hoy, J. (2020). *Fullstack React: The Complete Guide to ReactJS and Friends.* Newline.

45. Humble, J. y Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.* Addison-Wesley Professional.

46. INNOVA READING. (2021). *Conoce a Innova Reading.* https://innovareading.com/acerca-de/

47. Jackson, M.D. y McClelland, J.L. (1979). Processing determinants of reading speed. *Journal of experimental psychology: general, 108*(2).

48. Jasmine. (s.f.). *Jasmine. Behavior-Driven JavaScript.* https://jasmine.github.io/.

49. Jest. (2023). *Jest.* https://jestjs.io/.

50. Jin, B., Sahni, S. y Shevat, A. (2018). *Designing Web APIs.* O'Reilly Media, Inc.

51. Krug, S. (2014). *Don't make me think, revisited: A common sense approach to web usability.* New Riders.

52. Krutov, P. (2019). *Vue.js 2 Design Patterns and Best Practices: Build enterprise-ready, modular Vue.js applications with Vuex and Nuxt.* Packt Publishing.

53. Kujala, S., y Walsh, T. (2011). Designing for impact: the value of aesthetics in interface design. *Applied Ergonomics, 42*(4).

54. Langer, P. (1991). Speed reading courses and their effect on reading rate and comprehension. *Journal of Reading, 34*(8), 594-597.

55. Liang, Y. (2018). Application of gestalt psychology in product human-machine interface design. *IOP Conference Series: Materials Science and Engineering, 392*, 1.

56. López Rivas, O.H., Canto Mejía, H., Barrios Robles de Mejía, M.E. et al. (2018). *Guatemala en PISA-D. Programa Internacional de Evaluación de Estudiantes.* Ministerio de Educación de Guatemala. https://www.mineduc.gob.gt/digeduca/documents/pisa/InformePISADGuatemala.pdf.

57. Lupton, E. (2010). *Thinking with type: a critical guide for designers, writers, editors and students.* Princeton Architectural Press.

58. Mardan, A. (2014). *Full Stack JavaScript: Learn Backbone.js, Node.js, and MongoDB.* Apress.

59. Marks, T. (2019). *Color Harmony Compendium: A Complete Color Reference for Designers of All Types.* Rockport Publishers.

60. McArthur, G., Castles, A., Kohnen, S. y Banales, E. (2016). Low self-concept in poor readers: prevalence, heterogeneity, and risk. *Peer. J., 4*, e2669.

61. McNamara, D., O'Reilly, T., Rowe, M., Boonthum, C. y Levinstein, I.B. (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. Erlbaum.

62. Merina, E. (2021). *La Importancia de la Comprensión Lectora en los Niños.* El Método Montessori. https://elmetodomontessori.com/la-importancia-de-la-comprension-lectora-en-los-ninos/.

63. Miller, B., y Keenan, J. (2016). *Why Reading Comprehension Is Critical to Student Success.* American Psychological Association. https://www.apa.org/education/k12/reading-comprehension.

64. Monroy, José., y Gómez, Blanca. (2009). Comprensión lectora. *Revista Mexicana de Orientación Educativa, 6*(16), 37-42. Recuperado el 13 de mayo de 2023.

65. Mustafa, S., Deris, S. y Mohamad, R. (2013). *Comparative Evaluation of Automatic Test Case Generation Methods*. Technology, Education, and Science of International Conference.

66. Nassau, K. (1998). *Color for science, art and technology*. Elsevier.

67. National Center for Education Statistics. (2017). *The Nation's Report Card: 2017 National Assessment of Educational Progress (NAEP) Reading Assessment*. U.S. Department of Education. https://www.nationsreportcard.gov/reading_math_2017_highlights/.

68. NgRx. (2019). *Effects*. https://v7.ngrx.io/guide/effects.

69. NgRx. (2023). *What is NgRx?*. https://ngrx.io/docs.

70. Nist, S. L., y Simpson, M. L. (1990). The effects of reading comprehension strategy training on cognitive strategies and recall. *Journal of Educational Psychology, 82*(2), 331-335.

71. 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.

72. Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura. [UNESCO]. (2021). Estudio regional comparativo y explicativo (ERCE 2019). *Reporte nacional de resultados - Guatemala*. https://www.mineduc.gob.gt/digeduca/documents/pisa/InformePISADGuatemala.pdf.

73. Patton, R. (2006). *Software testing*. Sams Publishing.

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

75. PIRLS | IEA.nl. (2019). *IEA*. https://www.iea.nl/studies/iea/pirls.

76. Postman. (2023). *Build APIs together*. https://www.postman.com/.

77. Powell, T. (2002). *Web Design: The Complete Reference*. Osborne.

78. Pressman, R. S., y Maxim, B. R. (2015). *Ingeniería del software: un enfoque práctico* (Séptima ed.). McGraw-Hill.

79. Progrentis (2017). *¿Qué es Progrentis?*. http://soporteapps.progrentis.com/support/solutions/articles/12000025665–qu%C3%A9-es-progrentis.

80. Progrentis. (2021). *Desarrollada para estudiantes del siglo XXI*. https://www.progrentis.com/.

81. Rayner, K., Schotter, E., Masson, M., Potter, C., y Treiman, R. (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.

82. React. (s. f.). *React*. Recuperado de https://react.dev/.

83. Redux. (2023). *Redux - a predictable state container for JS Apps.* https://redux.js.org/.

84. Redux-Saga. (2023) *Redux-Saga: an intuitive Redux side effect manager.* https://redux-saga.js.org/.

85. Robbins, J. N. (2018). *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics*. O'Reilly Media.

86. Rosenfeld, L., y Morville, P. (2015). *Information Architecture: For the Web and Beyond*. O'Reilly Media.

87. Rutledge, M. (2021). *What is code coverage and why is it important?.* https://www.atlassian.com/continuous-delivery/software-testing/code-coverage.

88. Schloss, K. B., y Palmer, S. E. (2021). Color and emotion: 20 years of progress. *Psychology of Aesthetics, Creativity, and the Arts, 15*(2).

89. Schneps, M. H., Brockmole, J. R., Sonnert, G., y Pomplun, M. (2013). Learning to read in the digital age: A new opportunity for reading research. *Journal of Learning Disabilities, 46*(1), 21-34.

90. Secretaría de Planificación y Programación de la Presidencia de Guatemala (SEGEPLAN) (2015). *Estrategia Nacional para el Desarrollo de la Lectura.* http://www.segeplan.gob.gt/downloads/senplades/programas-proyectos/lectura/estrategia-nacional-para-el-desarrollo-de-la-lectura.pdf.

91. Shahid, M., Ibrahim, S. y Mahrin, M. (2011). *A Study on Test Coverage in Software Testing.*

92. Shklar, L., Rosen, R. (2003). *Web Application Architecture: principles, protocols and practices*. John Wiley & Sons, Ltd.

93. Skellie, R. (2019). *Practical Vue.js: Build a Real-World Web App with Vue, Vuex and Axios*. Apress.

94. Smart, D., Youssef, G., Sanson, A., Prior, M., Toumbourou, J. y Olsson, C. (2017). Consequences of childhood reading difficulties and behaviour problems for educational achievement and employment in early adulthood. *British Journal of Educational Psychology, 87*(2), 288-308.

95. Solan, H. A., Shelley-Tremblay, J., Hansen, P. C., y Larson, S. (2001). Effectiveness of computerized visual training in the treatment of developmental dyslexia: A review. *Optometry-Journal of the American Optometric Association, 72*(10), 719-726.

96. Stefanov, S., y Adams, C. R. (2019). *React: Up & Running: Building Web Applications*. O'Reilly Media.

97. Stefanov, Stoyan. *React: Up & Running: Building Web Applications*. O'Reilly Media, 2016.

98. Suárez, A., Moreno, J. M., y Godoy, M. J. (2010). Vocabulario y comprensión lectora: Algo más que causa y efecto. *Álabe: Revista de Investigación sobre Lectura y Escritura, 1*, 0-10.

99. Tidwell, J. (2010). *Designing Interfaces: Patterns for Effective Interaction Design.* O'Reilly Media.

100. Universidad Rafael Landívar [URL]. (2019). *Lectura rápida, principios para mejorar la velocidad de lectura.* https://cursoslibres.url.edu.gt/cursos/lectura-rapida-2/.

101. Velásquez, M., Cornejo, C. y Roco, A. (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. *Estudios Pedagógicos, 34*(1), 123-138.

102. Vitest. (2021). *Vitest - blazing fast unit test framework.* https://vitest.dev/.

103. Vue. (s.f.). *State management.* https://vuejs.org/guide/scaling-up/state-management.html#simple-state-management-with-reactivity-api.

104. Vuex. (2023). *What is Vuex?* https://vuex.vuejs.org/.

105. Willcutt, E. G., Betjemann, R. S., Pennington, B. F., Olson, R. K., DeFries, J. C., y Wadsworth, S. J. (2007). Longitudinal Study of Reading Disability and Attention-Deficit/Hyperactivity Disorder: Implications for Education. *Mind, Brain, and Education, 1*(4), 181–192.

106. Willingham, D. T. (2006). The usefulness of brief instruction in reading comprehension strategies. *American Educator, 30*(4), 39-45.

107. Wright, C. R., y Gronlund, N. E. (1984). The effect of training in a speed-reading technique on reading rate and comprehension. *Journal of Reading Behavior, 16*(1), 67-80.

CAPÍTULO 12

Anexos

12.1. Link del prototipo funcional

https://www.figma.com/proto/RQoeOdhVl5cK67Z6mdXe0i/LectoGym-App?node-id=1-2&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=1%3A2.

12.2. Link del repositorio de Github del proyecto

https://github.com/virtualmonkey/lecto-gym.

12.3. Pantalla de resultados del usuario 1 en la aplicación

12.4. Pantalla de resultados del usuario 2 en la aplicación

12.5. Pantalla de resultados del usuario 3 en la aplicación

12.6. Pantalla de resultados del usuario 4 en la aplicación

12.7. Pantalla de resultados del usuario 5 en la aplicación

12.8. Pantalla de resultados del usuario 6 en la aplicación

12.9. Pantalla de resultados del usuario 7 en la aplicación

12.10. Pantalla de resultados del usuario 8 en la aplicación

12.11. Pantalla de resultados del usuario 9 en la aplicación

12.12. Pantalla de resultados del usuario 10 en la aplicación

12.13. Ejemplo de carta de autorización firmada por padre o tutor de un usuario

Guatemala, 18 de marzo de 2023

Estimados padre de familia:

María Isabel Ortiz, Michael Chan, y Luis Urbina, estudiantes de la carrera de Ciencias de la computación y Tecnologías de la Información de la Universidad del Valle de Guatemala nos dirigimos a usted para solicitar su apoyo, en conjunto con el de su hijo, en nuestro proyecto de graduación. En concreto, con la aplicación que hemos desarrollado como parte de este llamada *"LectoGym"*.

"LectoGym" es una aplicación que tiene como objetivo ayudar a los estudiantes en un rango de 14 a 17 años de edad a mejorar su comprensión y rapidez lectora de una manera lúdica y entretenida. La aplicación está diseñada para adaptarse a las necesidades de cada estudiante y brindar un plan de estudios completo que los ayudará a mejorar su capacidad lectora en tres semanas.

Para poder completar el trabajo de graduación, nosotros necesitamos realizar pruebas con estudiantes de dicho rango de edad. Entiéndase, su hijo. Y es aquí donde necesitamos su ayuda como padres de familia.

Les pedimos su colaboración para permitir que sus hijos utilicen la aplicación LectoGym. Recalcando que los datos a recabar son confidenciales. Estos datos solo se utilizarán para efectos de probar la aplicación y redactar el informe del trabajo de graduación. Estamos seguros de que sus hijos disfrutarán la experiencia y será de mucha ayuda para su futuro como profesionales.

Agradecemos de antemano su consideración y apoyo en este proyecto. Si tienen alguna pregunta o inquietud, no dude en ponerse en contacto con nosotros a nuestros correos electrónicos institucionales: ort18176@uvg.edu.gt, cha18562@uvg.edu.gt, urb18473@uvg.edu.gt.

Atentamente,

Isabel Ortiz Naranjo

Luis Alejandro Urbina

Michael Steven Chan

Firma de conformidad del padre de familia o tutor:

Nombre del joven a participar: