

**PROTOTIPO WEB QUE GENERA EXÁMENES USANDO
PROCESAMIENTO DEL LENGUAJE NATURAL PARA EL CURSO DE
INGLÉS IV DE INGENIERÍA DE SISTEMAS EN LA UNIVERSIDAD
DEL VALLE SEDE TULUÁ**

Luis Eduardo Albarán Vélez

**Universidad del Valle
Facultad de ingeniería
Escuela de ingeniería de sistemas y computación
Tuluá
2021**

**PROTOTIPO WEB QUE GENERA EXÁMENES USANDO
PROCESAMIENTO DEL LENGUAJE NATURAL PARA EL CURSO DE
INGLÉS IV DE INGENIERÍA DE SISTEMAS EN LA UNIVERSIDAD
DEL VALLE SEDE TULUÁ**

Luis Eduardo Albarán Vélez

Código: 201667464

luis.albaran@correounivalle.edu.co

**Documento presentado como requisito parcial para la obtención de
grado de Ingeniero de Sistemas**

Director

Joshua David Triana Madrid, Ing

joshua.triana@correounivalle.edu.co

Universidad del Valle

Facultad de ingeniería

Escuela de ingeniería de sistemas y computación

Tuluá

2021

Trabajo de grado presentado por
Luis Eduardo Albarán Vélez
Como requisito parcial para la obtención del título de Ingeniero de Sistemas

Joshua David Triana Madrid, Msc.
Director

Jurado 1

Jurado 2

Agradecimientos

A mi familia por el apoyo incondicional en todos los momentos de mi vida.

A profesores que me guiaron en el aprendizaje.

A mi director de trabajo de grado por la confianza depositada.

Índice general

Agradecimientos	I
Índice general	II
Índice de figuras	IV
Índice de cuadros	V
Abstract	VII
Resumen	VIII
Introducción	IX
1. Contexto y objetivos	1
1.1. Planteamiento del problema	2
1.1.1. Descripción del problema	2
1.1.2. Formulación del problema	2
1.2. Objetivos	3
1.2.1. Objetivo general	3
1.2.2. Objetivos específicos	3
1.3. Antecedentes	3
1.4. Justificación del problema	4
1.4.1. Justificación social	4
1.4.2. Justificación académica	4
1.4.3. Justificación económica	4
1.5. Resultados esperados	5
1.6. Información sobre capítulos	6
2. Marco referencial	7
2.1. Marco de antecedentes	8
2.2. Marco teórico	10
2.2.1. Natural language processing (NLP)	10
2.2.2. Software como servicio (SaaS)	12
2.2.3. Metodologías ágiles de desarrollo de software	12
2.2.4. Manifiesto Ágil	12
2.2.5. Metodología de desarrollo Kanban	13
2.3. Marco conceptual	15
3. Recolección de información	16
3.1. Introducción	16
3.1.1. Documentos recolectados	17
3.1.2. Algoritmos recolectados	17
3.1.3. Análisis	18

3.1.4. Resultados	21
4. Modelamiento	23
4.1. Generación de texto	23
4.2. Preguntas	25
4.2.1. Generación y respuesta de preguntas	25
4.2.1.1. Descripción y detalles del modelo	25
4.2.1.2. Parámetros y uso	27
4.2.1.3. Datos de entrenamiento	27
4.3. Adaptación e integración	27
4.3.1. Generación de texto	28
4.3.2. Generación de preguntas y respuestas	29
4.4. Despliegue	30
4.4.1. Modelos de prestación	30
4.4.1.1. Infraestructura como servicio (IaaS)	30
4.4.1.2. Plataforma como servicio (PaaS)	30
4.4.1.3. Software como servicio (SaaS)	30
4.4.2. Productos de Google Cloud Platform	31
4.4.2.1. IA y Aprendizaje automático	31
4.4.2.2. Bases de datos	31
4.4.2.3. Procesamiento	31
4.4.2.4. Herramientas para desarrolladores	31
4.4.2.5. Almacenamiento	31
4.4.3. Detalles y resultados	32
5. Desarrollo del prototipo web	33
5.1. Ingeniería	34
5.1.1. Metodología de desarrollo	34
5.1.2. Análisis	34
5.1.2.1. Historias de usuario	34
5.1.3. Diseño	36
5.1.3.1. Diagrama casos de uso	36
5.1.3.2. Modelo de base de datos	37
5.1.3.3. Diagrama de despliegue	38
5.2. Descripción del prototipo web	38
5.2.1. Interfaz de usuario	38
5.2.2. Usuarios	39
5.2.3. Módulo Generación	39
5.2.3.1. Submódulo Parámetros de Generación	39
5.2.3.2. Submódulo Revisión Generación	40
5.2.3.3. Submódulo Configuración conjunto de exámenes	41
5.2.4. Módulo Examen	42
5.2.5. Módulo Estadísticas	42
5.2.6. Módulo Calificaciones	42
5.2.6.1. Submódulo Lista de calificaciones	43
5.2.6.2. Submódulo Revisión de examen	44
5.3. Implementación	44
5.3.1. Detalles de la implementación	44

5.3.1.1. Backend	45
5.3.1.2. Frontend	46
5.3.1.3. Repositorio	47
6. Pruebas	48
6.1. Pruebas unitarias	48
6.1.1. Descripción	48
6.1.2. Detalles	49
6.2. Pruebas de regresión	51
6.2.1. Descripción	51
6.2.2. Detalles	51
6.3. Pruebas de usabilidad	53
6.3.1. Conclusiones	54
7. Conclusiones y trabajos futuros	55
7.1. Conclusiones	55
7.2. Trabajos futuros	56
Referencias	57
Anexos	60
Historias de usuario	60
Justificación selección de algoritmos	61
Documentos recolectados	61
Algoritmos recolectados	62
Prototipo de interfaz GQuestions	62
Manuales de usuario	62
Plan de pruebas	62
Encuestas	62

Índice de figuras

2.2.1.Encoder Decoder structure	11
3.1.1.Relevancia documentos recolectados	19
4.1.1.Ejemplo entrada activa GPT-2	24
4.1.2.Ejemplo 2 entrada activa GPT-2	25
5.1.1.Tablero Kanban Jira - GQuestions	34
5.1.2.Diagrama de casos de uso - GQuestions	36
5.1.3.Modelo Entidad Relación - GQuestions	37
5.1.4.Diagrama de despliegue - GQuestions	38
5.2.1.UI Homepage	39
5.2.2.UI Párametros de generación - docente	40
5.2.3.UI Revisión de generación - docente	41
5.2.4.UI Configuración examen - docente	41
5.2.5.UI Estadísticas - docente	42
5.2.6.UI Calificaciones	43
5.2.7.UI Listado de calificaciones	43
5.2.8.UI Revisión de examen	44
6.1.1.Modelo Generación - backend Django	50
6.1.2.Clase Generación de prueba unitaria - backend Django	50
6.1.3.Prueba unitaria etiqueta de campo - backend Django	50
6.1.4.Prueba unitaria longitud de campo - backend Django	51
6.1.5.Prueba unitaria valor por defecto - backend Django	51
6.2.1.Prueba de regresión Selenium - frontend React	53

Índice de cuadros

1.5.1.Resultados esperados de objetivos específicos	5
1.6.1.Estructura de capítulos	6
3.1.1.Tabla documentos recolectados	17
3.1.2.Algoritmos recolectados	18
5.1.1.Historia de usuario - GQuestions	35
6.1.1.Caso de prueba etiquetas de campos	49
6.2.1.Caso de prueba inicio de sesión	52
7.2.1.Historia de usuario 1	60
7.2.2.Historia de usuario 5	60
7.2.3.Historia de usuario 8	61
7.2.4.Historia de usuario 15	61

Abstract

In this degree work project, a web prototype is developed that seeks to help reduce fraud in English exams, through the implementation of Natural Language Processing algorithms with the integration of different tasks such as Text Generation and Question Answering.

Through the use of the Kanban development methodology, the development is carried out allowing to offer a tool for English teachers that supports the reduction of the identified problem, in addition to the use of cutting-edge technologies in the Web area and in Artificial Intelligence As Natural Language Processing, these technologies include Django (Python), ReactJS, GPT-2, Google Cloud Platform (GCP) and others used in the project.

Keywords – Natural Language Processing, Text generation, Question Answering, Kanban, GPT-2, GCP, Django, ReactJS.

Resumen

En el presente proyecto de trabajo de grado se desarrolla un prototipo web que busca ayudar a reducir los fraudes en exámenes de Inglés, a través de la implementación de algoritmos de Procesamiento de Lenguaje Natural con la integración de diferentes tareas como la generación de texto (Text Generation) y la respuesta a preguntas (Question Answering).

Mediante el uso de la metodología de desarrollo Kanban se lleva a cabo el desarrollo permitiendo ofrecer una herramienta para los docentes de Inglés que sirve de apoyo a la reducción del problema identificado, además el uso de tecnologías de vanguardia en el área Web y en Inteligencia Artificial como Procesamiento de Lenguaje Natural, entre estas tecnologías se encuentra Django (Python), ReactJS, GPT-2, Google Cloud Platform (GCP) y otras utilizadas en el proyecto.

Palabras clave – Procesamiento de Lenguaje Natural, Text generation, Question Answering, Kanban, GPT-2, GCP, Django, ReactJS

Introducción

El presente trabajo trata sobre la problemática de deshonestidad académica «fraudes en exámenes» por parte de los estudiantes presentada en instituciones educativas, específicamente en universidades, se busca abordar la problemática desde la aplicación de la Ingeniería de Sistemas. Para ello se pretende la utilización de técnicas de procesamiento del lenguaje natural (NLP) que permitan generar exámenes a partir de algoritmos que utilizan estas técnicas. NLP corresponde a un área de la Inteligencia Artificial que consiste en el estudio de la interacción de las computadoras y el lenguaje humano, la máquina aprende la sintaxis y el significado del lenguaje humano, lo procesa y le da la salida al usuario.

Los exámenes generados a partir del algoritmo adaptado para este propósito serán aplicados en el área de Inglés de la Universidad del Valle sede Tuluá, a través del desarrollo de un prototipo web que sigue el proceso correspondiente de desarrollo de software con metodologías ágiles.

Capítulo 1

Contexto y objetivos

Índice

1.1.	Planteamiento del problema	2
1.1.1.	Descripción del problema	2
1.1.2.	Formulación del problema	2
1.2.	Objetivos	3
1.2.1.	Objetivo general	3
1.2.2.	Objetivos específicos	3
1.3.	Antecedentes	3
1.4.	Justificación del problema	4
1.4.1.	Justificación social	4
1.4.2.	Justificación académica	4
1.4.3.	Justificación económica	4
1.5.	Resultados esperados	5
1.6.	Información sobre capítulos	6

1.1. Planteamiento del problema

1.1.1. Descripción del problema

La copia en los exámenes es un problema que afecta a toda institución educativa, a pesar de las medidas tomadas, no se logra controlar, muchos estudiantes se ven en la necesidad de recurrir a esta práctica deshonesta con el fin de aprobar el curso [18].

En el año 2011 un estudio realizado por Jara, Riascos y Romero [24] sobre los exámenes ICFES del 2009, se obtuvo como resultado que alrededor del 96 % de las aulas hubo una pareja de estudiantes sospechosos de copia, estos resultados fueron obtenidos a partir del análisis de todos los exámenes de ese año usando un algoritmo de detección de copias en exámenes, con el fin de comparar respuestas y detectar similitudes entre las pruebas resueltas.

En los últimos años el avance tecnológico de alguna forma ha facilitado que los estudiantes hagan mal uso de este recurso para realizar copia en exámenes, además de prácticas convencionales. Cuando se buscan razones sobre la motivación de realizar copia por parte de los estudiantes en los exámenes, se encuentra que algunas son: falta de tiempo, falta de estudio, dificultad de la asignatura, ausencia de autoridad por parte del docente, etc.

Los experimentos realizados por Gino, Ayal y Ariely [9] plantean que la deshonestidad académica puede ser influida por el individuo hacia otros individuos que se reconocen como parte del mismo círculo social, un comportamiento que se cumple dentro del contexto académico.

En el estudio realizado por Lina Martínez y Enrique Ramírez [16] se logró obtener datos sobre la frecuencia con que estudiantes universitarios cometan fraudes académicos, a partir de encuestas realizadas a 3.300 estudiantes de cuatro universidades de Colombia entre los años 2003 y 2013, estos resultados mostraron que más del 90 % de encuestados admitieron haber cometido fraude; entre las conductas relacionadas se identificó que dejarse copiar en un examen e incluir a alguien en un grupo sin haber trabajo son de las más comunes.

La Universidad representa el modelo a seguir en nuestra sociedad, por ende, es importante hallar soluciones que disminuyan la deshonestidad académica, además, los estudiantes que recurren a esta conducta tienen una alta probabilidad de comportarse igual en su vida profesional.

1.1.2. Formulación del problema

¿Cómo crear un mecanismo para los docentes que ayude a reducir los fraudes en exámenes por parte de los estudiantes en el curso de Inglés IV de Ingeniería de Sistemas en la Universidad del Valle sede Tuluá?

1.2. Objetivos

1.2.1. Objetivo general

- Desarrollar un prototipo web que genere exámenes de inglés usando procesamiento del lenguaje natural para docentes en el área de inglés de la Universidad del Valle sede Tuluá.

1.2.2. Objetivos específicos

1. Revisar literatura y estado del arte sobre algoritmos de procesamiento del lenguaje natural para determinar los algoritmos que serán usados.
2. Adaptar algoritmos de procesamiento del lenguaje natural para que generen exámenes con distintas preguntas a partir del tema escogido por el docente de inglés.
3. Desarrollar un prototipo web que implemente las técnicas de procesamiento de lenguaje natural usadas en el algoritmo.
4. Realizar pruebas de usabilidad al prototipo web y aplicar métricas para evaluar el algoritmo de Procesamiento de lenguaje natural.

1.3. Antecedentes

El presente trabajo trata sobre la problemática de deshonestidad académica «fraudes en exámenes» por parte de los estudiantes presentada en instituciones educativas, específicamente en universidades, se busca abordar la problemática desde la aplicación de la Ingeniería de Sistemas. Para ello se pretende la utilización de técnicas de procesamiento del lenguaje natural (NLP) que permitan generar exámenes a partir de algoritmos que utilizan estas técnicas. NLP corresponde a un área de la Inteligencia Artificial que consiste en el estudio de la interacción de las computadoras y el lenguaje humano, la máquina aprende la sintaxis y el significado del lenguaje humano, lo procesa y le da la salida al usuario.

Los exámenes generados a partir del algoritmo adaptado para este propósito serán aplicados en el área de Inglés de la Universidad del Valle sede Tuluá, a través del desarrollo de un prototipo web que sigue el proceso correspondiente de desarrollo de software con metodologías ágiles.

1.4. Justificación del problema

1.4.1. Justificación social

Esta propuesta busca beneficiar a docentes, estudiantes y la Universidad del Valle en sí, donde se aplicará la solución. Los docentes de inglés podrán contar con una herramienta automatizada que genera diferentes exámenes y que intenta reducir los fraudes durante los exámenes por parte de los estudiantes en el salón de clase.

1.4.2. Justificación académica

La propuesta se realiza con la motivación de aplicar los conocimientos aprendidos a lo largo de la carrera de Ingeniería de Sistemas en una problemática real, especialmente los cursos de Inteligencia artificial, Álgebra lineal, Desarrollo de software, Desarrollo de aplicaciones web, base de datos, Técnicas de pruebas de software, y otros cursos que no están directamente involucrados pero que han contribuido fuertemente a la capacidad de resolver problemas de modo sistemático.

1.4.3. Justificación económica

El prototipo web será desarrollado para ser ejecutado en la Universidad del Valle sede Tuluá para docentes de inglés, contar con una herramienta que facilite la realización de exámenes en el área de inglés podría aportar en lo económico ya que se está automatizando dicho proceso; además, el prototipo podría ser escalable a otras instituciones, en especial, instituciones académicas donde su actividad principal gira alrededor del aprendizaje del lenguaje Inglés, permitiendo realizar exámenes a estudiantes sobre diferentes temas en esta área que se quieran validar.

1.5. Resultados esperados

Nº	Objetivo específico	Resultado esperado
1	Revisar literatura y estado del arte sobre algoritmos de procesamiento del lenguaje natural para determinar los algoritmos que serán usados	Documento de posibles algoritmos que serán usados para adaptar el algoritmo que genera exámenes de inglés
2	Adaptar algoritmos de procesamiento del lenguaje natural para que generen exámenes con distintas preguntas a partir del tema escogido por el docente de inglés	<ul style="list-style-type: none"> - Obtener un algoritmo adaptado que genere exámenes de inglés por medio del Procesamiento de Lenguaje Natural - Modelo de generación de exámenes
3	Desarrollar un prototipo web que implemente las técnicas de Procesamiento de Lenguaje Natural usadas en el algoritmo	<ul style="list-style-type: none"> - Prototipo web integrado con el algoritmo de Procesamiento de Lenguaje Natural - Código fuente, artefactos de software, manual de usuario
4	Evaluar el prototipo web desarrollado	<ul style="list-style-type: none"> - Documento de plan de pruebas a realizar para evaluar el prototipo web - Análisis de los resultados obtenidos

Cuadro 1.5.1: Resultados esperados de objetivos específicos

Fuente: Elaboración propia

1.6. Información sobre capítulos

Capítulo	Descripción	Objetivo específico
Capítulo 2: Marco referencial	En este capítulo se explican diferentes conceptos que son indispensables en la comprensión del problema, estado del arte y hacia qué áreas se enfoca la solución	Ninguno
Capítulo 3: Recolección de información	En este capítulo se presenta la información recolectada y considerada como relevante para la solución del problema, específicamente documentos como artículos, tesis, etc y algoritmos de Procesamiento de Lenguaje Natural	1
Capítulo 4: Modelamiento	En este capítulo se presenta detalles sobre los algoritmos seleccionados, detalles como preparación de datos, entrenamiento, evaluación, implementación, despliegue, entre otros	2
Capítulo 5: Desarrollo del prototipo web	En este capítulo se presenta el proceso relacionado al de desarrollo del prototipo web	3
Capítulo 6: Pruebas	En este capítulo se presentan las pruebas realizadas al Prototipo Web, detalles y conclusiones de las pruebas	4
Capítulo 7: Conclusiones y trabajos futuros	En este capítulo se presentan las conclusiones finales y los posibles trabajos futuros del trabajo de grado	Ninguno

Cuadro 1.6.1: Estructura de capítulos

Fuente: Elaboración propia

Capítulo 2

Marco referencial

Índice

2.1.	Marco de antecedentes	8
2.2.	Marco teórico	10
2.2.1.	Natural language processing (NLP)	10
2.2.2.	Software como servicio (SaaS)	12
2.2.3.	Metodologías ágiles de desarrollo de software	12
2.2.4.	Manifiesto Ágil	12
2.2.5.	Metodología de desarrollo Kanban	13
2.3.	Marco conceptual	15

2.1. Marco de antecedentes

Con los años los fraudes realizados en exámenes por parte de estudiantes universitarios han estado presentes, por lo cual se han hecho diversas investigaciones, estudios e informes que abordan esta problemática.

La investigación Cheating and feeling honest: Committing and punishing analog versus digital academic dishonesty behaviors in higher education [8] realizada en 2016 con el objetivo de examinar la deshonestidad académica entre estudiantes universitarios, analizó casos del mismo cometidos por estudiantes, así como las razones de los comportamientos y la severidad de las sanciones por violaciones de la integridad académica. Este estudio se basó en Pavela's (1997) [21] tipos de deshonestidad académica (engaño, plagio, fabricación y facilitación), además utilizaron “el marco motivacional para cometer un acto de deshonestidad académica” realizado por Murdock y Anderman [19] y el “modelo de Mantenimiento del autoconcepto” realizado por Mazar, Amir y Ariely [17] para analizar las razones de los comportamientos deshonestos de los estudiantes. Los resultados de esta investigación mostraron que la deshonestidad analógica (tradicional) era más frecuente que la deshonestidad digital, según la investigación mencionada señalan que en Israel el 95 % de estudiantes admitieron haber cometido algún tipo de deshonestidad académica, ya sea virtual o presencial, de los cuales el 60 % participó en la copia de documentos y el 60 % durante exámenes, mientras que en Korea otro estudio señala que el 69 % de estudiantes admitieron haber participado en al menos un tipo de deshonestidad académica. Como argumentación por parte de los estudiantes, indicaron que la razón más frecuente de su deshonestidad académica fue la necesidad de mantener una visión positiva de sí mismo como persona honesta a pesar de violar los códigos éticos.

Por otra parte, la institución Defensor Universitario de la Universidad de Alcalá, Madrid - España, realizó un informe [6] en junio de 2018 donde su objetivo comprende analizar el fraude académico en los procesos de evaluación del aprendizaje, indagar en la legislación y el reglamento sobre el fraude y responder el interrogante principal: ¿Cómo contrarrestar el fraude académico en la Universidad de Alcalá?.

La manera de abordar la problemática en dicho informe es mayormente desde los aspectos legales, normativas y reglamentos, con el informe se expone el estado de la situación, las circunstancias que la rodean, además, plantean aspecto relevantes sobre fraudes académicos como: tipología, factores favorecedores, consecuencias, medidas preventivas, medidas para detecciones, procedimientos a seguir tras la detección.

Diversos estudios realizados demuestran que el problema sucede de manera más regular de lo que parece, exponiendo que más del 30 % de los universitarios admiten haber realizado fraudes académicos, alrededor del 50 % afirman haber hecho copia en exámenes, así que, la intención del Defensor Universitario es sacar a la luz un problema que como se mencionó

anteriormente se presenta con regularidad y acudir a normativas y/o reglamentos como un mecanismo efectivo.

Actualmente existen sistemas que monitorean plataformas de E-learning [1] donde las instituciones tienen implementado un LMS (Learning Management System), estos sistemas ofrecen herramientas para realizar evaluaciones tales como talleres, foros, tareas, ejercicios y exámenes , asimismo, brinda posibilidades de aplicación a cualquier tipo de prueba, como exámenes finales, pruebas de inglés, exámenes de admisión, entre otros. LMS brinda muchas ventajas pero consigo trae desventajas como las posibilidades de fraude o suplantación en un examen en línea.

Validar la identidad del estudiante cuando realiza un examen en línea es un desafío de seguridad debido a que ninguna plataforma LMS puede validar que efectivamente es el estudiante quien está realizando la prueba. Por esta razón nacen los servicios de eProctering [7], una implementación digital de Proctering que ha sido empleada en instituciones educativas, estos servicios constituyen un conjunto de herramientas de software, metodologías y protocolos usadas para monitorear la actividad del estudiante al realizar un examen en línea.

La solución de eProctering puede ser implementadas en plataformas Moodle que son usadas en Campus virtuales, la efectividad de estos servicios eProctering no son de especial interés para este trabajo, por lo cual no se hará énfasis en ello; sin embargo, se evidencia que existe una forma de abordar la problemática desde el campo del desarrollo de software como herramienta para afrontar la copia en exámenes virtuales y que es importante mencionarla. A pesar de existir estos servicios, suelen tener un costo algo elevado para las instituciones y supone algunas cuestiones de privacidad, como consecuencia una posible presión psicológica para el estudiante al estar siendo monitoreado durante su prueba.

Por este mismo camino de software de apoyo existen otros como Urkund [32] o Turnitin [30] que se usan para el mismo propósito de evitar fraudes académicos, otro como TestWe [27]que sirve para tomar el control del ordenador y evitar que personas externas puedan suplantar al estudiante. Estos software utilizan sistemas que permiten comparar exámenes entre sí y sistemas que hacen comparaciones con fuentes disponibles en Internet.

Después de hacer un recorrido por investigaciones, estudios, herramientas de software con el fin de conocer el estado del arte, se observa que la problemática es abordada por la parte de vigilar, controlar, ,detectar, aplicar reglas y normativas, entre otras acciones; no obstante, también se observa que no se aborda la problemática por la vía de la generación de exámenes que su finalidad sea tratar de reducir fraudes académicos en exámenes, ni mucho menos por medio de un software que brinde una solución por esta vía, es por esto que se pretende desarrollar un prototipo web que brinde dicha solución generando exámenes con diferentes preguntas usando técnicas procesamiento del lenguaje natural.

2.2. Marco teórico

2.2.1. Natural language processing (NLP)

El procesamiento del lenguaje natural es un área integral de la informática en la que se utilizan ampliamente el aprendizaje automático y la lingüística computacional. Este campo se ocupa principalmente de hacer que la interacción humana y la computadora sea fácil pero eficiente. La máquina aprende la sintaxis y el significado del lenguaje humano, lo procesa y le da la salida al usuario. El área de PLN implica la creación de sistemas informáticos para realizar tareas significativas con el lenguaje natural y comprensible para los humanos.

La razón por la que el procesamiento del lenguaje natural es tan importante en el futuro es que ayuda a construir modelos y procesos que toman trozos de información como entrada y en forma de voz o texto o ambos y los manipulan según el algoritmo dentro de la computadora. Por lo tanto, la entrada puede ser voz, texto o imagen, donde la salida de un sistema de PLN puede procesarse tanto en voz como en texto escrito [13] .

Algunos de los diferentes algoritmos desarrollados para aumentar la eficiencia del procesamiento del lenguaje en forma de texto son:

- Memoria a largo plazo a corto plazo:

LSTM significa memoria a corto plazo [11]. La red neuronal recurrente es el elemento principal del modelo LSTM. La red neuronal recurrente es un fragmento de red neuronal que puede recordar valores. LSTM es un tipo especial de red neuronal recurrente que puede recordar la entrada anterior en un intervalo de tiempo arbitrario y predecir la salida. Se utiliza para entrenar la máquina mediante conjuntos de entrada. Es uno de los modelos de aprendizaje en aprendizaje automático que se utiliza ampliamente en el procesamiento del lenguaje natural.

- Secuencia 2 Modelo de secuencia (Seq2Seq):

El modelo tradicional seq2seq contiene dos redes neuronales recurrentes, es decir, una red codificadora y una red decodificadora [4].

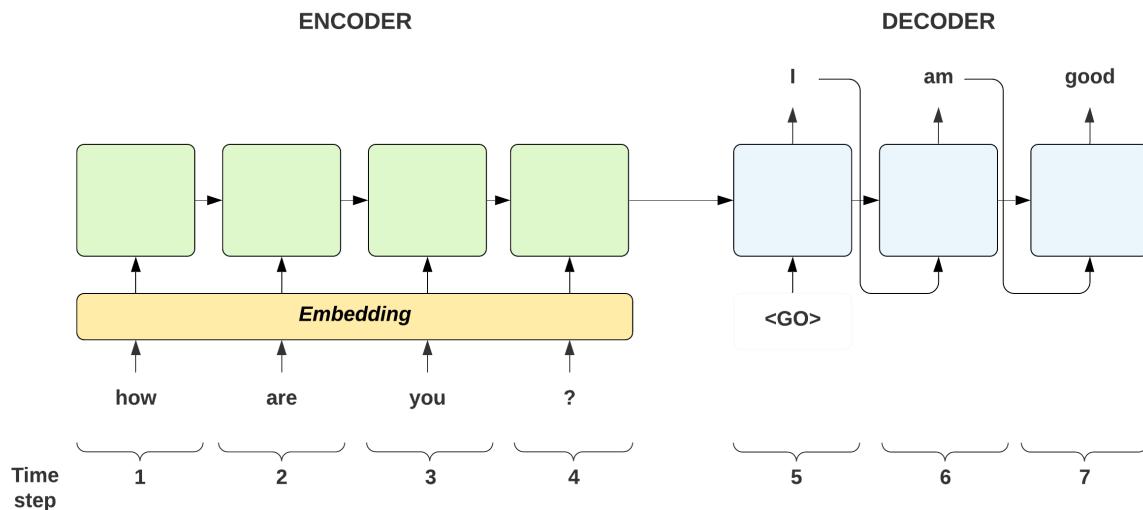


Figura 2.2.1: Encoder Decoder structure

Fuente: Elaboración propia

En primer lugar, la lista de vocabulario se crea y se compila mediante incrustaciones para que el modelo pueda identificar la sintaxis grammatical correcta. El conjunto de vocabulario se procesa para verificar las apariciones de las palabras y clasificar las palabras de uso frecuente, de uso poco frecuente y únicas en el vocabulario. A continuación, las palabras se reemplazan por ids. Según los id, la sugerencia de respuesta se decodifica y se da como salida.

- **Modelo de reconocimiento de entidad nombrada:**

Como sugiere el nombre, el reconocimiento de entidad con nombre se utiliza para identificar nombres relevantes, clasificar el nombre por la entidad a la que pertenecen. El modelo NER busca nombres de lugares, personas y otras entidades relevantes en el conjunto de datos de entrada que pueden estar en forma de texto o voz [33].

- **Modelo de gráfico de preferencias del usuario:**

El gráfico de preferencias del usuario se utiliza para crear un conjunto de opciones de usuario. Cuando el usuario elige repetidamente tiempos verbales específicos, adjetivos, conjunciones y preposiciones, etc., se crea un gráfico de preferencias de modo que cuando el usuario usa un tipo similar de oraciones, el modelo sugiere las siguientes palabras calculando la probabilidad [14]. Estas palabras se asignan entre sí, por lo que se crea un gráfico de preferencias para un usuario en particular.

- **Modelo de incrustación de palabras (Word Embedding):**

La incrustación de palabras se deriva del aprendizaje de características y el modelado del lenguaje en el procesamiento del lenguaje natural, donde las palabras y frases se mapean en vectores del gráfico de número real de preferencias.

- Basado en características extracción de oraciones utilizando reglas de inferencia difusas.
- Algoritmo basado en plantillas que usa resumen de texto automatic (text summarization):

El resumen automático de texto es la tarea de producir un resumen conciso y fluido mientras se conserva el contenido de la información clave y el significado general. [2].

2.2.2. Software como servicio (SaaS)

El software como servicio (SaaS) [12] es un nuevo paradigma de aplicaciones de software a las que se puede acceder a través de un navegador web o una interfaz de programa de aplicación (API) [3]. SaaS utiliza la infraestructura en la nube y la plataforma como servicio (PaaS) como base de recursos informáticos [29]. Los datos almacenados en SaaS están altamente disponibles para los consumidores de SaaS. Los servicios SaaS se ejecutan y operan en el centro de datos del proveedor de servicios [31].

2.2.3. Metodologías ágiles de desarrollo de software

El término ágil [15] surge como iniciativa de un conjunto de expertos en el área de desarrollo de software con el fin de optimizar el proceso de creación del mismo, el cual era caracterizado por ser rígido y con mucha documentación [28]. El punto de partida fue el manifiesto ágil, el cual es un documento donde se detalla todo lo que involucra la filosofía “ágil”.

2.2.4. Manifiesto Ágil

Este es un documento que engloba principios y valores que hacen diferente un proyecto de desarrollo de software ágil de uno en su forma tradicional.

Según el manifiesto ágil se valora a:

- El individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.
- Desarrollar software que funcione más que la documentación del mismo.
- La colaboración con el cliente más que la negociación de su contrato.
- Responde a los cambios más que seguir con el plan establecido [28].

Esta metodología ágil está regida además por doce principios [20] que ayudan a que el proceso de desarrollo se vuelva menos complejo y responda de manera oportuna a los

cambios que surgen a lo largo del mismo, siempre contando con el punto de vista del cliente:

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
2. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
3. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
5. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan confiar en ellos para conseguir finalizar el trabajo.
6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
7. El software que funciona es la medida principal de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y los usuarios deberían ser capaces de mantener una paz constante.
9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
12. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

2.2.5. Metodología de desarrollo Kanban

David Anderson fue pionero en implementar esta metodología para el desarrollo de software, en 2004 bajo la guía de Don Reinertsen utilizó Kanban en un proyecto de TI de Microsoft. Kanban consiste en mejorar el flujo de trabajo de un equipo, aumentando al mismo tiempo la productividad y la calidad del producto final. Kanban se engloba dentro de la denominada «metodología ágil» y, como tal, otorga una gran flexibilidad a los procesos de trabajo. Las tareas se dividen en pequeñas fases que se realizan de forma consecutiva y en base al siguiente lema: «Stop starting – start finishing». En lugar de comenzar nuevas tareas y realizarlas todas de una manera más o menos simultánea, deberá

llevarse a término cada una de las fases que las constituyen antes de empezar la siguiente [25].

Reglas:

Las tres principales reglas de Kanban según (Pérez, 2012) [22] son las siguientes:

1. **Visualizar el flujo de trabajo:** Dividir el trabajo en partes o tareas, escribirlas en una tarjeta y colocarla en la columna inicial, las columnas o estados pueden ser tantas como el equipo considere necesario que pase cada tarea. El objetivo primordial de esta regla es que el trabajo a realizar quede claro, visualizar en qué está trabajando cada miembro del equipo y que todos tengan algo que hacer, siempre teniendo en cuenta las prioridades de cada tarea.
2. **Determinar el límite del WIP (Work In Progress):** Limitar el número de tareas que se pueda realizar en cada estado del flujo de trabajo, independientemente de la magnitud del proyecto siempre hay una cantidad óptima. El principal objetivo de esta regla es detectar cuellos de botella fácilmente para buscar soluciones, muchas de las veces la solución más eficiente sería la colaboración del equipo que tenga procesos libres y pueda aceptar nuevos ítems.
3. **Controlar el tiempo en completar una actividad (Lead time):** Inicia desde su petición hasta su entrega, mientras que cycle-time inicia desde que una actividad comienza que finaliza, es decir, mide el rendimiento del proceso. Es indispensable optimizar estas métricas para el control y una mejora continua.

Tablero Kanban: Proporciona visibilidad de todo el proceso de software, mostrando el trabajo asignado a cada miembro del equipo, además indica las prioridades de cada tarea y resalta los cuellos de cuello de botella existentes, de este modo el equipo se concentra en los problemas que bloquean o impiden el proceso buscando una solución rápida, así se minimizará los defectos y se mantendrá un flujo estable. El uso de un tablero Kanban permitirá también equilibrar la demanda con el rendimiento del trabajo liberado por el equipo, consiguiendo un ritmo de desarrollo sostenible, un mayor rendimiento del equipo y elevar la calidad del producto final. De esta forma se reduce el tiempo del ciclo (lead time) y se incrementa la confianza del cliente al entregar avances de su producto regularmente (Guzmán, Islas, Corona, Méndez, 2014) [10].

Kanban combinado con otras metodologías: Kanban suele ser combinado con otras metodologías como herramienta de apoyo, de la misma forma en la que diversas metodologías y filosofías de desarrollo son, muchas veces, compatibles entre sí, permitiendo combinar y crear métodos que cumpla con necesidades del equipo y la organización. Un ejemplo de una metodología híbrida publicada es Scrumban que combina prácticas de Scrum y Kanban. En esta metodología se plantea a Kanban como una base sobre la cual uno puede, o no, colocar prácticas de Scrum según se considere necesario (Sanhueza, 2017)

[23].

Beneficios: Los principales beneficios para (Shore, 2010) [26] y (crisp, 2010) [5] son:

- Es muy flexible y permite detectar cualquier problema existente y ajustar el flujo de trabajo para obtener mejores resultados.
- Beneficia el flujo visual mediante tarjetas de colores distribuidas en el mismo tablero.
- La digitalización del tablero Kanban tiene la facilidad de acceder a su flujo desde cualquier sitio para comunicarse con el equipo de desarrolladores.
- Reduce el tiempo de espera y el dedicado a la asignación de tareas mediante el flujo constante de tareas.
- Visibilidad en tiempo real de los cuellos de botella.
- Desarrollo de software ágil sin la necesidad de tener que usar iteraciones de compromiso fijo de tiempo fijo como los sprints de SCRUM.

Herramientas Kanban: Jira, Kanbanize, Trello, MeisterTask, Taiga, monday.com, Clarizen, ProjectManager.com, Breeze

2.3. Marco conceptual

Deshonestidad académica: Actos individuales o colectivos en que se presenta como propio el conocimiento ajeno, tales como: copia de exámenes, tareas, trabajos o proyectos, plagio de textos, sustitución de personas en los exámenes, falsificación de documentos o datos, presentación de trabajos o proyectos elaborados por terceros y cualquier tipo de acción que atente contra la honestidad académica en el Instituto.

Sistemático: Aquello que respeta o se adapta a un sistema: un conjunto ordenado o estructurado de principios o elementos que se relacionan entre sí. El término proviene del latín tardío *systematicus*, a su vez derivado del griego *systēmatikós*.

Escalabilidad: Propiedad de aumentar la capacidad de trabajo de un programa sin alterar la capacidad de trabajo o tamaño de un sistema sin poner en riesgo el buen funcionamiento y la calidad del mismo. Cuando esto sucede, dicha propiedad recibe el nombre de “sistema escalable” o escalable.

Capítulo 3

Recolección de información

Índice

3.1. Introducción	16
3.1.1. Documentos recolectados	17
3.1.2. Algoritmos recolectados	17
3.1.3. Análisis	18
3.1.4. Resultados	21

En este capítulo se presenta toda la información recolectada y considerada como relevante en relación con la generación de texto y la generación de respuesta a preguntas.

3.1. Introducción

Se ha elaborado una tabla que consiste en cincuenta (50) documentos, principalmente artículos de los cuales se ha recolectado información relevante de cada uno con el fin de identificar las técnicas y algoritmos usados en investigaciones, desarrollo de modelos/sistemas, implementaciones u otros.

La elección de documentos se ha dividido en dos tipos de tareas principales de Procesamiento de Lenguaje Natural (PLN), la primera generación de texto (Text Generation) y la segunda generación de respuestas a preguntas (Question Answering), veinticuatro y veintiséis documentos respectivamente.

Los documentos seleccionados en su mayoría son del año 2008 hasta el presente, esta consideración se tuvo debido al acelerado avance en el campo de NLP hecho por investigadores, con ello el uso de técnicas más sofisticadas y mejores resultados en experimentos realizados.

La información recolectada e identificada como relevante para el trabajo actual, será fundamental para la implementación del modelo adaptado que generará exámenes de Inglés, a continuación se adjunta la tabla elaborada y posteriormente su análisis.

3.1.1. Documentos recolectados

En esta sección se presenta una parte de los cincuenta (50) documentos recolectados, se tuvo en cuenta información relevante como la siguiente:

- Técnicas de Procesamiento de Lenguaje Natural
- Ventajas
- Desventajas
- Logros
- Trabajos futuros
- Otros

Información Artículos - Generación de Texto y Respuesta a Preguntas Procesamiento del Lenguaje Natural									
Id	Nombre del documento	Autores	Año	Técnicas PLN	Info. adicional (Red neuronal, Dataset, estrategias, etc)	Resumen	Relevancia (1-5)	Ventajas	Desventajas
1	An Integrated Deep Generative Model for Text Classification and Generation	ZhengWang y QingbiaoWu	2018	Bag of words, Word embedding, Text label	Variational Autoencoder (VAE), LSTM	En este artículo, se propone un modelo integrado basado en VAE para manejar tareas de generación y clasificación de texto.	3	Sirve también para la generación de texto, utiliza explícitamente la información de la etiqueta durante la decodificación, resultados competitivos en ambas tareas	El modelo fue entrenado para la clasificación de texto
2	Recent advances of neural text generation: Core tasks, datasets, models and challenges	JIN HanQi, CAO Yue, WANG TianMing, XING XinYu & WAN XiaoJun	2020	No se especifican técnicas de PLN	Métricas de evaluación: Perplexity, BLEU (based n-gram), meteor, Self-BLEU Modelos: Seq2Seq, Transformers, GPT-2, BERT, VAE, GAN	Esta review tiene como objetivo proporcionar una síntesis actualizada de las tareas centrales en la generación de texto neuronal y las arquitecturas adoptadas para manejar estas tareas, y sobre los desafíos en la generación de texto neuronal.	2	Presenta los avances recientes en la generación de texto neuronal. - Guía y referencia para investigadores y profesionales en esta área.	No entran en detalle, ni mencionan técnicas específicas de PLN para la generación de texto
3	LSTM encoder-decoder with adversarial network for text generation from keyword	Dongju Park, Chang Wook Ahn	2018	Word Embedding, Skip gram, Word2Vec	NLG, GAN, RNN, LSTM, selfAttention, MLE, BLEU Score, COCO	En este artículo, se propone un modelo para generar texto a partir de una palabra determinada. El propósito es contribuir a ampliar la diversidad de textos generados a partir de una sola palabra.	4	Generación de texto realista basado en una palabra dada y un discriminador que puede distinguir entre datos generados y reales Uso de LSTM en lugar de RNN en el codificador y decodificador Mayor diversidad de oraciones	- Genera oraciones cortas
4	A Text Generation and Prediction System: Pretraining on New Corpora Using BERT and GPT-2	Yuanbin Qu, Peihan Liu, Wei Song, Lizhen Liu, Miaomiao Cheng	2020	No se especifican técnicas de PLN	GPT-2, BERT, Transformer	En este documento, se utilizan modelos pre-entrenados para completar algunas tareas de generación de texto chino, incluida la generación de oraciones largas, utilizando Transformer a partir de modelos pre-entrenados	4	Oraciones largas a partir de palabra clave Desempeño de GTP-2 de vanguardia Se describe el pseudocódigo del modelo de generación de texto a grandes rasgos	Para oraciones demasiadas largas comienza a volverse repetitivo (duplicaciones), se aclara que el problema se puede deber a que los corpus seleccionados no están estandarizados, por lo que el modelo no logra aprender todas las reglas del lenguaje. Entrenado para generar texto en chino

Cuadro 3.1.1: Tabla documentos recolectados

Fuente: Elaboración propia

La tabla completa se encuentra en los anexos.

3.1.2. Algoritmos recolectados

En esta sección se presenta una parte de los diez (10) algoritmos recolectados, se tuvo en cuenta información relevante de manera similar al ítem 3.1.1.

Recolección algoritmos - Prototipo web que genera exámenes usando Procesamiento del Lenguaje Natural para el curso de Inglés IV de Ingeniería de Sistemas										
#	Tema	Nombre	Autores	Descripción	Enlace	Técnicas PLN	Información adicional (Estrategias, red neuronal, Datasets, otros)	Relevancia (1-5)	Ventajas	Desventajas
1	Text Generation	GPT-2, Language Models are Unsupervised Multitask Learners	Radford, Alec and Wu, Jeff and Child, Rewon and Luan, David and Amodei, Dario and Sutskever, Ilya	Es un modelo de aprendizaje automático transformador, GPT-2 utiliza el aprendizaje profundo para traducir texto, responder preguntas, resumir pasajes y generar resultados de texto	https://github.com/openai/gpt-2	Tokenization, Word Embedding (word2vec), Byte Pair Encoding (BPE)	Transformers, self-Attention	4	- Puede ser reentrenado con otro conjunto de datos - Código abierto y permite hacer Fine tuning - Documentación detallada - Disponible en cuaderno de Google Colab para probar - Artículo en Medium	- Puede volverse repetitivo o sin sentido al generar pasajes largos (Más de dos párrafos)
2	Question Generation	Question Generation using transformers	Suraj Patil, Neeraj Varshney, Priyank Soni, Manuel Romero, Tom Wilde	Este proyecto está dirigido como un estudio de código abierto sobre la generación de preguntas con transformadores preentrenados (seq2seq) utilizando métodos sencillos de extremo a extremo sin tuberías muy complicadas	https://github.com/patil-suraj/question-generation	Tokenizer, Word Embedding, Masked LM (MLM), Next Sentence Prediction (NSP)	Transformers, T5, Seq2Seq, sQUDAD	4	-	-
3	Question generation	Question Generator	Adam Montogomerie	Question Generator es un sistema de PNL para generar preguntas de estilo de comprensión lectora a partir de textos como artículos de noticias o extractos de páginas de libros. El sistema está construido utilizando modelos previamente entrenados de HuggingFace Transformer. Tiene tres componentes: el generador de preguntas en sí mismo y el evaluador de control de calidad, que clasifica y filtra los pares de preguntas y respuestas en función de su aceptabilidad.	https://github.com/AMontogomerie/question_generator	Tokenizers	Transformers, T5, Seq2Seq, sQUDAD	4	- Permite cambiar algunos parámetros del modelo, como el número de preguntas, el estilo de preguntas - Fácil ejecución - Genera preguntas de opción múltiple a partir de texto	- No tiene bien especificada la estructura del modelo, sin embargo se podría obtener inspeccionando el código fuente
4	Question generation	Question Generation	Kristiyian Vachev	La idea es generar respuestas de opción múltiple a partir del texto, dividiendo este complejo problema en pasos más simples: Identifique las palabras clave del texto y utilícelas como respuestas a las preguntas. Reemplaza la respuesta de la oración con un espacio en blanco y úsala como base para la pregunta. Transforme la oración con un espacio en blanco para responder a una oración más similar a una pregunta. Genere distractores, palabras similares a la respuesta, como respuestas incorrectas.	https://github.com/KristiyianVachev/Question-Generation	Part of speech, Named entity, Word count	sQUDAD, TF-IDF score and cosine similarity, scikit-learn's Gaussian Naive Bayes	4	- Buena documentación - El algoritmo está dividido en pasos más sencillos y entendibles (modularidad y simplicidad)	- Las preguntas generadas no son realmente aptas para utilizar directamente en exámenes

Cuadro 3.1.2: Algoritmos recolectados

Fuente: Elaboración propia

La tabla completa se encuentra en los anexos.

3.1.3. Análisis

De acuerdo a la relevancia de cada documento se le ha asignado un valor de uno (1) a cinco (5) a consideración basándose en ventajas, desventajas, logros, entre otros, siendo cinco (5) muy relevante y uno (1) poco relevante. De acuerdo con el año de publicación se evidencia que después de haber realizado la selección de documentos, los publicados recientemente son los más relevantes en cuanto a resultados obtenidos.

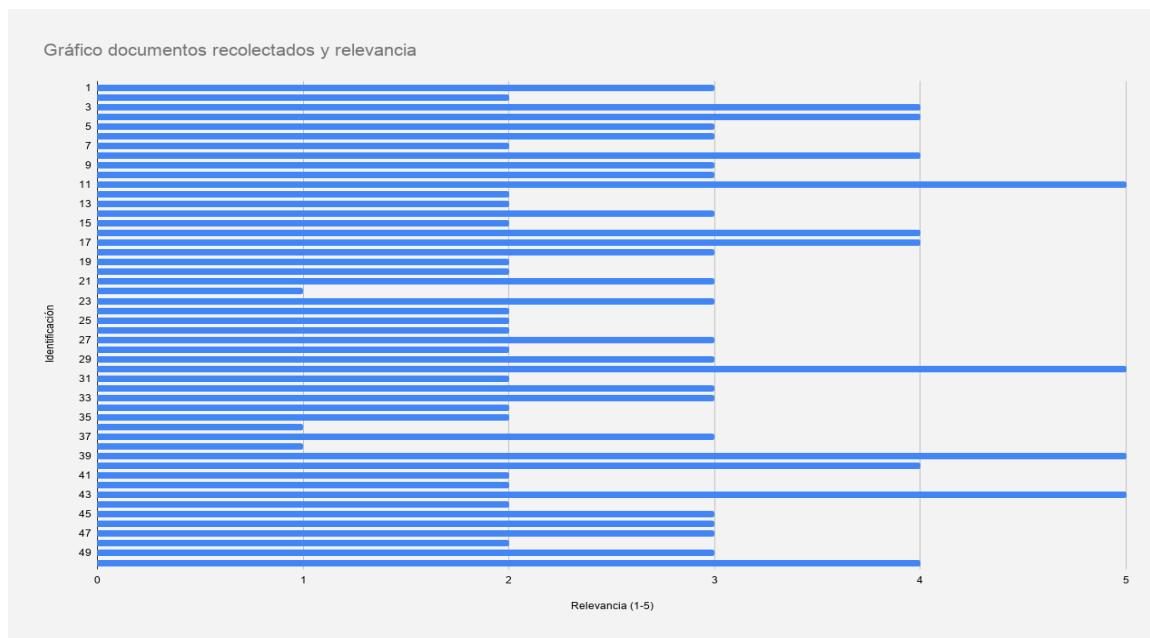


Figura 3.1.1: Relevancia documentos recolectados
Fuente: Elaboración propia

Entre los más relevantes se encuentra BERT (Bidirectional Encoder Representations from Transformers), una técnica de aprendizaje automático basada en Transformer para la formación previa del Procesamiento del Lenguaje Natural (PLN) desarrollada por Google [1]. Se puede evidenciar que la técnica BERT está presente en los documentos considerados más relevantes y en una cantidad importante de ellos, esto se debe a que los resultados obtenidos en los experimentos realizados por los autores de los artículos son prometedores.

Del mismo modo GPT-2 (Generative Pre-trained Transformer 2), una inteligencia artificial de código abierto creada por OpenAI que utiliza el aprendizaje profundo para diferentes tareas (Traducción, responder preguntas, resumir pasajes y generar resultados de texto), se encuentra citada en repetidas ocasiones, siendo el más prometedor entre los diferentes tipos de modelos propuestos en los artículos. Su sucesor GPT-3 no ha sido considerado como opción viable debido al acceso restringido a la API y al objetivo al que está enfocado su uso.

En cada documento analizado se logró recopilar las diferentes técnicas de Procesamiento de Lenguaje Natural con el fin de conocer en detalle la construcción de los modelos propuestos o investigaciones presentadas (teoría). Por tanto a continuación se describen las técnicas de PLN utilizadas según la información recopilada en la tabla para las tareas de generación de texto y respuesta a preguntas.

- **Tokenization:** La tokenización es el proceso de demarcar y posiblemente clasificar secciones de una cadena de caracteres de entrada. Los tokens resultantes luego se pasan a otra forma de procesamiento. El proceso se puede considerar una subtarea de analizar la entrada [1].

- **NLP Lower-Case**
- **StopWords Pre-Processing:** Palabras vacías es el nombre que reciben las palabras sin significado como artículos, pronombres, preposiciones, etc. que son filtradas antes o después del procesamiento de datos en lenguaje natural [].
- **Stemming:** Cortar el final o el comienzo de una palabra, teniendo en cuenta una lista de prefijos y sufijos comunes que se pueden encontrar en una palabra flexionada [].
- **Lemmatization:** Generalmente se refiere a hacer las cosas correctamente con el uso de un vocabulario y análisis morfológico de palabras, normalmente con el objetivo de eliminar solo las terminaciones flexivas y devolver la forma base o de diccionario de una palabra, lo que se conoce como lema [].
- **StopWords Pre-Processing:** Palabras vacías es el nombre que reciben las palabras sin significado como artículos, pronombres, preposiciones, etc. que son filtradas antes o después del procesamiento de datos en lenguaje natural [].
- **PoS Tagging (part-of-speech tagging):** Es el proceso de marcar una palabra en un texto (corpus) como correspondiente a una parte particular del discurso, basado tanto en su definición como en su contexto [].
- **Term Frequency:** frecuencia de término (o sea, la frecuencia de ocurrencia del término en la colección de documentos), es una medida numérica que expresa cuán relevante es una palabra para un documento en una colección [].
- **Word Embedding:** La incrustación de palabras es cualquiera de un conjunto de técnicas de aprendizaje de características y modelado del lenguaje en el procesamiento del lenguaje natural (PNL) donde las palabras o frases del vocabulario se asignan a vectores de números reales [].
- **Bag of words:** El modelo de bolsa de palabras es una representación simplificada utilizada en el procesamiento del lenguaje natural y la recuperación de información (IR). En este modelo, un texto (como una oración o un documento) se representa como la bolsa (multiset) de sus palabras, sin tener en cuenta la gramática e incluso el orden de las palabras, pero manteniendo la multiplicidad [].
- **Masked LM (MLM):** El modelado de lenguaje enmascarado es una tarea de llenar espacios en blanco, en la que un modelo usa las palabras de contexto que rodean un símbolo de máscara para intentar predecir cuál debería ser la palabra enmascarada [].
- **Next Sentence Prediction (NSP):** En el proceso de entrenamiento de BERT, el modelo recibe pares de oraciones como entrada y aprende a predecir si la segunda oración del par es la oración subsiguiente en el documento original [].

- **N-gram:** Un n-grama es una secuencia contigua de n elementos de una muestra dada de texto o habla. Los elementos pueden ser fonemas, sílabas, letras, palabras o pares de bases según la aplicación. Los n-gramas normalmente se recopilan de un texto o corpus de voz [].
- **Named Entity Recognition (NER):** El reconocimiento de entidad con nombre (NER) (también conocido como identificación de entidad (con nombre) , fragmentación de entidad y extracción de entidad) es una subtarea de extracción de información que busca ubicar y clasificar las entidades con nombre mencionadas en texto no estructurado en categorías predefinidas como persona nombres, organizaciones, ubicaciones, códigos médicos , expresiones de tiempo, cantidades, valores monetarios, porcentajes, etc. [] .

Para la generación de preguntas en la mayoría de modelos se utiliza SQuAD (Stanford Question Answering Dataset) para entrenar y evaluar los modelos. SQuAD es un conjunto de datos de comprensión de lectura, que consta de preguntas planteadas por los trabajadores de la red en un conjunto de artículos de Wikipedia, donde la respuesta a cada pregunta es un segmento de texto, o un intervalo, del pasaje de lectura correspondiente, o la pregunta podría ser incontestable [] .

Gracias a la revisión del estado del arte y recolección de información presentada en la tabla es posible tener una base sólida en la elección de técnicas y modelos a utilizar en el trabajo de grado actual (generación de exámenes de Inglés).

Adicionalmente, una vez obtenida las fuentes teóricas, se recopiló información de manera más específica sobre algoritmos que se basan en la teoría presente en la mayoría de documentos, para este propósito también se realiza una tabla asociada a 10 algoritmos de código abierto disponible en repositorios o plataformas reconocidas como Hugging Face [] .

Después de realizar una búsqueda exhaustiva de algoritmos de código libre disponibles en la Web que fueran potencialmente útiles para cumplir el primer objetivo propuesto, se puede concluir que, los modelos basados en Transformers ofrecen resultados de vanguardia tanto para la tarea de generación de texto como para la tarea de respuesta a preguntas.

3.1.4. Resultados

Tras realizar un análisis profundo sobre los artículos y algoritmos recolectados se describe a continuación los algoritmos seleccionados para la implementación en el prototipo web:

- **Generación de texto**

Para el algoritmo de generación de texto se seleccionó GPT-2 de Hugging Face [] principalmente por la diferencia de rendimiento en comparación con los otros algoritmos recopilados. Dicha diferencia representa casi el doble de tiempo de ejecución en igualdad de condiciones (entradas). Dado que la comparación se redujo

a dos algoritmos, el de Hugging Face (GPT-2) y un algoritmo encontrado en GitHub llamado GPT2-Pytorch [], ambos están claramente basados en GPT-2 lo cual implica que los textos generados sean similares al modelo original GPT-2 y que la diferencia se defina por el rendimiento de generación de texto.

El algoritmo proporcionado por Hugging Face es oficial respecto a GPT2-Pytorch que es una modificación de GPT-2 hecha por un desarrollador de la comunidad, de allí las diferencias en algunos aspectos como la calidad de los resultados, esto se debe a que, Hugging Face es un proveedor de código abierto reconocido en el campo de Procesamiento de Lenguaje Natural (PLN).

Se realizaron encuestas a un docente de Inglés con el fin de obtener una evaluación de un experto en cuanto a la calidad de los textos generados por ambos algoritmos, se compararon tiempos de respuesta para comparar el rendimiento, se tuvieron en cuenta las limitaciones y otros detalles, finalmente, los resultados obtenidos favorecieron a GPT-2 Hugging Face.

■ Generación de preguntas y respuestas

Para la selección del algoritmo de generación de preguntas y respuestas, se compararon dos algoritmos de código abierto para esta tarea, sin embargo, desde un inicio uno de los algoritmos presentaba mayor potencial de acuerdo al propósito del proyecto, ya que, de los recolectados es el único algoritmo capaz de generar preguntas y respuestas de opción múltiple y de oraciones (preguntas abiertas), es decir dos tipos de preguntas.

El tipo de preguntas es un factor importante para el propósito del proyecto, ya que, a mayor tipo de preguntas más posibilidades de generar exámenes variados para los estudiantes.

De acuerdo a las pruebas y posterior análisis realizado entre ambos algoritmos inicialmente seleccionados, se decidió seleccionar finalmente el algoritmo encontrado en GitHub llamado Question Generator [], debido a todas las ventajas obtenidas respecto a el propósito del proyecto expuestas anteriormente.

La justificación de selección de algoritmos completa se encuentra en los anexos.

Capítulo 4

Modelamiento

Índice

4.1. Generación de texto	23
4.2. Preguntas	25
4.2.1. Generación y respuesta de preguntas	25
4.3. Adaptación e integración	27
4.3.1. Generación de texto	28
4.3.2. Generación de preguntas y respuestas	29
4.4. Despliegue	30
4.4.1. Modelos de prestación	30
4.4.2. Productos de Google Cloud Platform	31
4.4.3. Detalles y resultados	32

En este capítulo se presenta detalles sobre los algoritmos seleccionados y mencionados en el capítulo 3, detalles como preparación de datos, entrenamiento, evaluación, implementación, despliegue, entre otros.

La palabra algoritmo, modelo y sistema en este capítulo se utilizan indistintamente.

4.1. Generación de texto

En esta sección se describe brevemente el funcionamiento del modelo generador GPT-2 que fue el seleccionado en el proyecto para la tarea de generar texto.

La forma más sencilla de ejecutar un modelo GPT-2 entrenado es permitir que divague por sí solo (lo que técnicamente se llama Generating Unconditional Samples); alternativamente, se le puede dar un mensaje para que genere sobre un tema determinado (también conocido como Generating Interactive Conditional Samples). En el caso de divagaciones, simplemente

se entrega el token de inicio y se hace que comience a generar palabras (el modelo entrenado usa <|endoftext|> como su token de inicio. En las siguientes figuras se mostrará como <s>). []

En el caso del proyecto al modelo GPT-2 se le pasó un mensaje inicial (oración) para que generara a partir de ese mensaje el texto (Generating Interactive Conditional Samples).

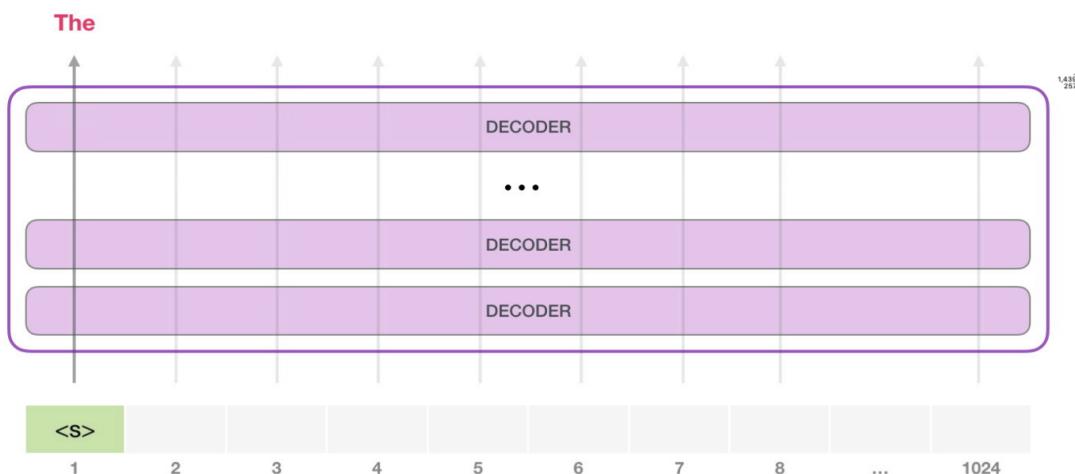


Figura 4.1.1: Ejemplo entrada activa GPT-2
Fuente: The Illustrated GPT-2 []

El modelo solo tiene un token de entrada, por lo que esa ruta sería la única activa. El token se procesa sucesivamente a través de todas las capas, luego se produce un vector a lo largo de ese camino. Ese vector se puede puntuar con el vocabulario del modelo (todas las palabras que conoce el modelo, 50.000 palabras en el caso de GPT-2). En este caso se selecciona el token con mayor probabilidad. GPT-2 tiene un parámetro llamado top-k utilizado para que el modelo considere el muestreo de palabras distintas de la palabra superior (cuando top-k = 1). []

En el siguiente paso, se agrega la salida del primer paso a la secuencia de entrada para que el modelo haga su próxima predicción:

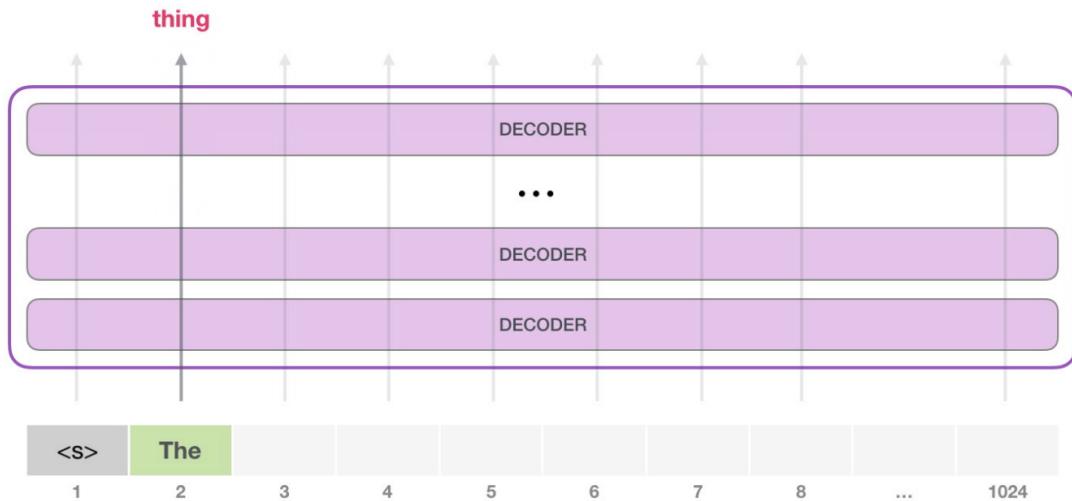


Figura 4.1.2: Ejemplo 2 entrada activa GPT-2

Fuente: The Illustrated GPT-2 [1]

La segunda ruta es la única que está activa en este cálculo. Cada capa de GPT-2 ha conservado su propia interpretación del primer token y lo usará para procesar el segundo token. GPT-2 no reinterpreta el primer token a la luz del segundo token.

4.2. Preguntas

En esta sección se describe aspectos importantes del algoritmo de generación de preguntas seleccionado.

El algoritmo de código abierto seleccionado [1] es capaz de realizar dos tareas (generación de preguntas y evaluación de preguntas para mayor calidad), en la sección 4.2.1.1 se describe el generador de preguntas y el evaluador de preguntas, a pesar de presentarse por separado, es importante denotar que ambas tareas se encuentran integradas en el algoritmo gracias a Adam Montgomerie, Ingeniero de Machine Learning; esto a través de dos modelos entrenados y desplegados en Hugging Face Transformers construidos por Adam Montgomerie [2].

4.2.1. Generación y respuesta de preguntas

4.2.1.1. Descripción y detalles del modelo

Es un algoritmo que permite generar preguntas con sus respuestas correspondientes a partir de textos del tipo comprensión lectora a partir artículos, pasajes extraídos de libros, entre otros. El algoritmo está construido en base a dos modelos de Hugging Face descritos a continuación:

- **t5-base-question-generator []**

Es un modelo generador de preguntas sequence to sequence (seq2seq) que toma una respuesta como entrada y da como salida una pregunta, está basado en un modelo pre-entrenado t5-base [].

Este es un modelo ajustado con conjuntos de datos de control de calidad como SQuAD , CoQA y MSMARCO. Fue entrenado durante 20 épocas sobre el conjunto de entrenamiento con taza de aprendizaje de 1e-3.

- **Uso:** El modelo toma respuestas concatenadas y el contexto como una secuencia de entrada y de este modo genera una oración de pregunta como secuencia de salida. El formato de entrada se organiza de la siguiente manera:

```
answer_token <answer-phrase>context_token <context-from-text>
```

La secuencia de entrada se puede codificar y pasar directamente al método `generate()` del modelo. La manera de obtener mejores resultados es generar gran cantidad de preguntas y después realizar un filtro utilizando el evaluador QA (bert-base-cased-qa-evaluator).

- **Limitaciones:** Las preguntas generadas pueden presentar sesgos y presentar incoherencia, si el contexto es excesivamente corto o si directamente no tiene contexto, o si la respuesta no coincide con el contexto.

- **bert-base-cased-qa-evaluator []**

Es un modelo que toma como entrada un par de preguntas y respuestas para generar un valor resultante que representa la predicción sobre si el par de preguntas y respuestas son válidas o no. Está basado en el modelo pre-entrenado BERT base cased []. Este evaluador QA fue diseñado inicialmente para el modelo visto previamente y poder evaluar las preguntas generadas de dicho modelo. El conjunto de datos de entrenamiento fueron RACE, SQuAD , CoQA y MSMARCO.

- **Uso:** El modelo sigue un formato basado en BertForSequenceClassification a diferencia de que se usa tanto pregunta y respuesta como las dos secuencias, el formato de entrada se organiza de la siguiente manera:

```
[CLS] <question>[SEP] <answer>[SEP]
```

El token **CLS** significa clasificación y se interpone al principio del par de oración de ejemplo de entrada, de otra manera, el token **SEP** significa separador y se utiliza para separar oraciones para la siguiente tarea de predicción de oraciones.

- **Limitaciones:** El modelo está entrenado para evaluar si una pregunta y una respuesta están relacionadas semánticamente, pero no puede determinar si una respuesta es efectivamente correcta o no.

4.2.1.2. Parámetros y uso

- Texto: Este parámetro puede ser pasado como argumento de línea de comando o como argumento de función `generate(text, num_questions=10)`. Es el contexto para la generación de preguntas, a partir del texto de entrada se extraen las entidades nombradas (NER).
- Número de preguntas: Este parámetro puede ser pasado como argumento de línea de comando o como argumento de función `generate(text, num_questions=10)`. Cuando el número de preguntas es demasiado grande, puede suceder que el modelo no genere suficientes preguntas, el número de preguntas depende directamente de la longitud del texto de entrada. Es importante denotar que la calidad de las preguntas puede reducir si el número de preguntas es grande, esto se debe a que, las preguntas pasan por el modelo evaluador y este devuelve las mejores preguntas.
- Estilo de preguntas: Este parámetro puede ser pasado como argumento de línea de comando o como argumento de función `generate(text, num_questions=10, answer_style=<style>)`. Los estilos de preguntas pueden ser oraciones (“sentences”), de opción múltiple (“multiple_choice”) o ambas (“all”).

4.2.1.3. Datos de entrenamiento

- **RACE:** Es un conjunto de datos de comprensión lectora a gran escala con más de 28.000 pasajes y casi 100.000 preguntas.
- **SQuAD:** Es un conjunto de datos de comprensión lectora, que consta de preguntas planteadas por los trabajadores de la red en un conjunto de artículos de Wikipedia, donde la respuesta a cada pregunta es un segmento de texto, o un intervalo , del pasaje de lectura correspondiente [].
- **CoQA:** Es un conjunto de datos a gran escala para la construcción de Sistemas de respuesta a preguntas conversacionales.
- **MSMARCO:** Es un conjunto de datos a gran escala centrado en la comprensión de lectura automática, la respuesta a preguntas y la clasificación de pasajes, la extracción de frases clave y los estudios de búsqueda conversacional [].

4.3. Adaptación e integración

En esta sección se presentan detalles de la adaptación o modificación de los algoritmos seleccionados, respondiendo a uno de los objetivos principales del proyecto actual.

La generación de texto es una tarea que se decidió incluir en el prototipo web para utilizar completamente Procesamiento de Lenguaje Natural, tanto texto como preguntas. El texto

generado por GPT-2 es una de las entradas al algoritmo de generación de preguntas y respuestas (se complementan).

4.3.1. Generación de texto

La plataforma Hugging Face ofrece una funcionalidad llamada API de inferencia acelerada (Accelerated Inference API) que permite integrar modelos previamente entrenados (o modelos propios privados) y desplegados en la plataforma a través de solicitudes HTTP simples, la inferencia que ofrecen es de 2 a 10 veces más rápida dependiendo del plan.

En el proyecto se utilizó esta API para realizar peticiones HTTP al modelo GPT-2 con el plan gratuito, este plan permite hasta 30.000 caracteres de entrada por mes y utiliza CPU acelerada para estas tareas, suficiente para el propósito del proyecto. Para el uso de GPU es necesario un plan de pago de inicio o empresarial.

Para empezar a hacer uso de esta API fue necesario crear una cuenta en la plataforma y posteriormente obtener un token que se envía en la solicitud para aprovechar funciones de aceleración. Con el plan gratuito cada respuesta de solicitud de un texto varía entre 8 y 15 segundos dependiendo de la longitud del texto.

En el cuerpo de solicitud se debe enviar el JSON de la siguiente forma:

```
{"inputs": <sentence>, "parameters": "max_length": <length>,
"num_return_sequences":<number>}
```

- Inputs: Inicio de oración, a partir de esta entrada se genera las demás palabras que conformarían el texto.
- Parameters: Como opciones acepta el número máximo de longitud que tendrá el texto generado y el número de textos que genera. El número máximo de longitud no puede sobrepasar el valor de 500 palabras, es una limitación por parte de la API que se tuvo en cuenta en el desarrollo del prototipo web.

La petición se debe realizar al siguiente dominio del servidor:

<https://api-inference.huggingface.co/models/gpt2>

El enlace debe tener un formato donde el último elemento debe ser el nombre del modelo desplegado en Hugging Face (en este caso .../gpt2).

Como se mencionó hace poco, en la solicitud HTTP se debe enviar un parámetro inicio de oración **<sentence>** para generar el texto. Fue necesario tener un conjunto de datos de oraciones como entrada al momento de generar los textos desde el prototipo web, para esto se realizó lo siguiente:

1. **Selección de conjunto de datos (dataset):** En este paso lo que se hizo fue revisar diferentes datasets que tuvieran textos de diversas áreas de interés (Ej:

Computer programming, algorithm design, operating systems, etc). Tras realizar la revisión de los diferentes datasets se escogió el dataset Web of Science (WOS) [] que es un conjunto de datos de clasificación de documentos que contiene 46,985 documentos con 134 categorías que incluyen 7 categorías principales.

2. **Selección de áreas o categorías:** En este paso se realizó un proceso sencillo utilizando Microsoft Excel, se utilizaron filtros para seleccionar manualmente las áreas netamente relacionadas a Ingeniería de Sistemas (se descartaron áreas de salud, política, otros), esto se debe a que, el proyecto está enfocado a un curso del programa de Ingeniería de Sistemas. Algunas de las áreas seleccionadas fueron: Data Structures, Parallel computing, Image processing, Machine learning, Problem-solving, Relational databases, Distributed computing, Structured storage, Computer vision, Operating systems, y muchos más (26). El resultado de esta selección se exportó en formato CSV y fueron aproximadamente 11.000 de los 46.985 textos iniciales.
3. **Limpieza de datos y extracción de oraciones:** En este paso se procedió a escribir un programa en Python que recibiera el CSV filtrado con el fin de eliminar textos que posiblemente tuvieran palabras inadecuadas y textos que iniciaran con enumeración (ej. “1. Machine learning is ...”). Una vez eliminados los textos inadecuados se procedió a sacar las oraciones de los textos originales, se redujo cada texto a sus diez primeras palabras (oración <sentence>) y se exportó en formato JSON 5.000 oraciones que serían las entradas al momento de generar textos desde el prototipo web con Javascript.

4.3.2. Generación de preguntas y respuestas

A diferencia de la generación de texto, el algoritmo de generación de preguntas y respuestas se tuvo que realizar un proceso más laborioso debido a que no estaba en ninguna plataforma que ofreciera el servicio API directamente. De este modo, se procedió a crear una API REST utilizando Flask que es un framework web para Python que proporciona funcionalidad para crear aplicaciones web, incluida la gestión de solicitudes HTTP.

Para obtener una respuesta a la solicitud en un formato adecuado para consumir los datos, se modificó uno de los archivos del algoritmo (questiongenerator.py) y se codificó para obtener una salida del tipo:

<insertar ejemplo respuesta API QA>

Se creó un proyecto por separado para este algoritmo, la descarga de este algoritmo incluye la descarga de los dos modelos que utiliza para su funcionamiento, es importante mencionar que por esta razón el tamaño en disco de los modelos era inicialmente grande, alrededor de 8GB, esto suponía un problema para el despliegue visto más adelante, sin embargo, se logró reducir a un tamaño en disco de 1.3GB utilizando tuberías (pipelines)

de Hugging Face. Las tuberías son una forma excelente y fácil de usar modelos para inferencia. Estas tuberías o pipelines son objetos que abstraen la mayor parte del código complejo de la biblioteca, ofreciendo una API simple dedicada a varias tareas, incluido el reconocimiento de entidades nombradas, el modelado de lenguaje enmascarado, el análisis de sentimientos, la extracción de características y la respuesta a preguntas [1].

En cuanto a rendimiento la ejecución de este algoritmo es de alta exigencia computacional, por lo que el autor recomienda el uso de GPU. Por limitaciones computacionales las pruebas de API tuvieron que hacerse en CPU y los tiempos de respuesta iban de 1 a 2 minutos por generación de 5 preguntas con longitud de texto de 350 palabras aproximadamente, en GPU esta misma tarea toma de 7 a 10 segundos. Para pruebas generales (no API) se utilizó Google Colab que permite aprovechar recursos computacionales como el uso gratuito de GPU.

4.4. Despliegue

En el despliegue se tuvieron en cuenta varias alternativas de servicios Cloud Computing, entre las que están Google Cloud Platform, IBM Watson, Amazon Web Services (AWS) y Microsoft Azure, todos estos servicios son de pago, sin embargo, Google Cloud ofrece tres meses de prueba con 300 dólares y por esta razón se eligió esta plataforma para desplegar el modelo de Generación de preguntas y respuestas.

Google Cloud platform (GCP) es una plataforma en la nube que permite crear, probar e implementar soluciones utilizando la infraestructura confiable y escalable de Google. GCP se ejecuta en la misma infraestructura que Google usa internamente para sus productos.

4.4.1. Modelos de prestación

4.4.1.1. Infraestructura como servicio (IaaS)

Permite a TI ejecutar máquinas virtuales sin tener que invertir o administrar esta infraestructura informática ellos mismos.

4.4.1.2. Plataforma como servicio (PaaS)

Basado en el modelo IaaS. Los clientes optan por todos los beneficios de IaaS, además de obtener una infraestructura subyacente, como sistemas operativos y middleware.

4.4.1.3. Software como servicio (SaaS)

Todo está disponible a través de la web: el proveedor aloja, administra y entrega toda la infraestructura, incluidas las aplicaciones.

4.4.2. Productos de Google Cloud Platform

En esta sección se describe brevemente algunos productos que ofrece GCP.

4.4.2.1. IA y Aprendizaje automático

- Vertex AI: Plataforma unificada para entrenar, alojar y administrar modelos de AA.
- Deep Learning VM Image: VM preconfiguradas para aprendizaje profundo.
- Contenedores Deep Learning: Contenedores previamente configurados para entornos de aprendizaje profundo

4.4.2.2. Bases de datos

- Cloud SQL: Base de datos completamente administrada para MySQL, PostgreSQL y SQL Server.
- Firebase Realtime Database: Base de datos NoSQL para almacenar y sincronizar datos en tiempo real.

4.4.2.3. Procesamiento

- App Engine: Plataforma de aplicaciones sin servidores para apps y backends.
- Compute Engine: Máquinas virtuales que se ejecutan en el centro de datos de Google.
- Cloud Run: Entorno completamente administrado para ejecutar apps alojadas en contenedores.

4.4.2.4. Herramientas para desarrolladores

- Artifact Registry: Almacena, administra y protege imágenes de contenedor y paquetes de lenguajes.
- Cloud Build: Plataforma de integración y entrega continuas.

4.4.2.5. Almacenamiento

- Archive Storage: Archivo de datos que ofrece velocidad de acceso en línea a muy bajo costo.
- Cloud Storage: Almacenamiento de objetos seguro, duradero y escalable.
- Filestore: Almacenamiento de archivos altamente escalable.

4.4.3. Detalles y resultados

De los servicios descritos anteriormente se utilizaron Cloud Run, Cloud Build y Cloud Storage.

Para realizar el despliegue en Cloud Run se tuvo que utilizar contenedores, específicamente Docker, al momento de compilar el servicio se verifica el archivo docker que hace referencia al proyecto.

En los directorios del proyecto deben estar los archivos main.py, DockerFile, requirements.txt, .gcloudignore.

La máquina virtual que ejecuta el algoritmo una vez está desplegado en Cloud Run utiliza CPU de 4 núcleos y 8GB, para utilizar GPU que sería lo ideal los cobros son altos y no es posible con la prueba gratuita; por esta razón la respuesta a las solicitudes API son tardías por limitaciones computacionales a pesar de estar corriendo en la nube (GCloud), sin embargo permitió tener el servicio disponible en la nube en vez de manera local y realizar peticiones HTTP utilizando el framework Flask.

Capítulo 5

Desarrollo del prototipo web

Índice

5.1.	Ingeniería	34
5.1.1.	Metodología de desarrollo	34
5.1.2.	Análisis	34
5.1.3.	Diseño	36
5.2.	Descripción del prototipo web	38
5.2.1.	Interfaz de usuario	38
5.2.2.	Usuarios	39
5.2.3.	Módulo Generación	39
5.2.4.	Módulo Examen	42
5.2.5.	Módulo Estadísticas	42
5.2.6.	Módulo Calificaciones	42
5.3.	Implementación	44
5.3.1.	Detalles de la implementación	44

En este capítulo se presenta todo el proceso de desarrollo del prototipo web con los artefactos de software, además, se hace referencia a la metodología de desarrollo, las tecnologías usadas, el diseño, los detalles de la implementación y otras consideraciones importantes.

5.1. Ingeniería

5.1.1. Metodología de desarrollo

La metodología de desarrollo de software aplicada fue Kanban, después de identificar el tipo de proyecto se determinó que era la metodología que más se ajustaba. En la sección 2.2.5 se explica la metodología Kanban con todos los beneficios que ofrece.

En la implementación se utilizó la plataforma Jira que permite visualizar el flujo de trabajo, establecer los límites de trabajo en curso y la medición del tiempo para cada tarea.

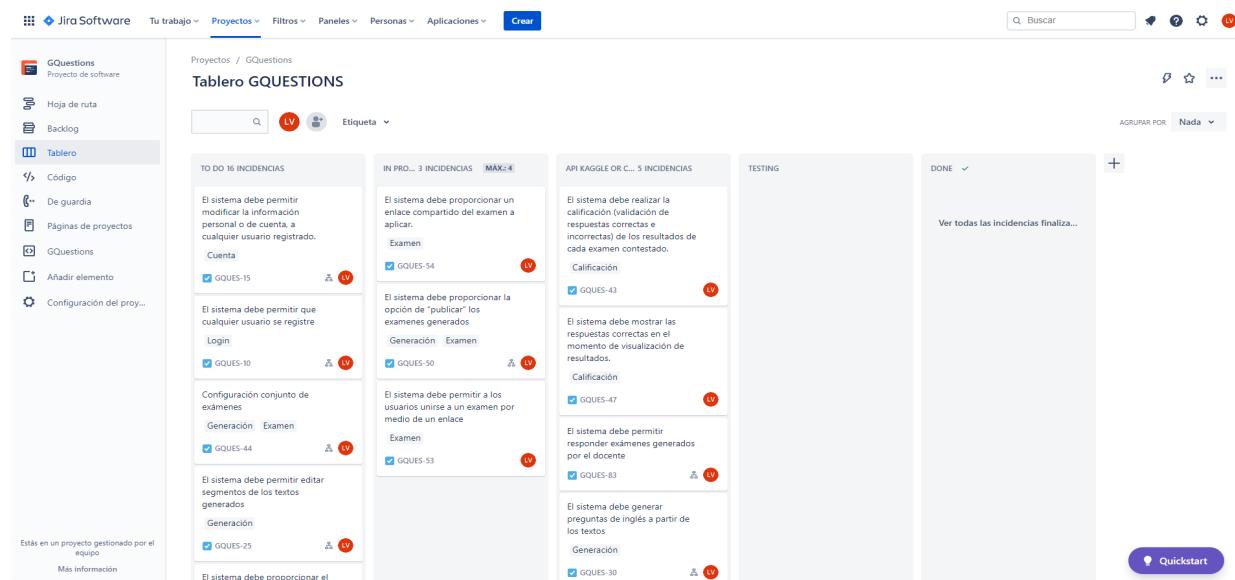


Figura 5.1.1: Tablero Kanban Jira - GQuestions

Fuente: Elaboración propia

5.1.2. Análisis

En esta fase se llevó a cabo el análisis de las necesidades identificadas de acuerdo al problema planteado y las soluciones a brindar por medio de un prototipo web, teniendo en cuenta los objetivos trazados y resultados esperados.

5.1.2.1. Historias de usuario

Con el fin de identificar las funciones del prototipo web que daban valor al cliente se elaboraron las historias de usuario, incluyendo la planeación de tareas a llevar a cabo. Se utilizó un formato que permitiera establecer una prioridad a cada historia de usuario de acuerdo a la importancia en el desarrollo.

Este formato contiene la siguiente información: autor, responsable, módulo al que pertenece, prioridad, descripción, validación y finalmente la fecha de creación.

Historia de usuario	
Número de historia: 17	
Autor: Luis Eduardo Albarán Vélez	
Módulo: Generación	Prioridad: Alta
Responsable: Luis Eduardo Albarán Vélez	
Descripción: Como docente de Inglés, quiero poder establecer una hora de inicio y finalización del examen	
Validación: Dado que el usuario se encuentre en el módulo de evaluación, cuando vaya a configurar el examen, entonces se le muestra diferentes opciones como las siguientes: - Fecha y hora de inicio, fecha y hora límite de finalización, fecha y hora de visualización de resultados	
Fecha de creación: 15-02-21	

Cuadro 5.1.1: Historia de usuario - GQuestions

Fuente: Elaboración propia

Las demás especificaciones de historias de usuario se encuentran en los anexos.

5.1.3. Diseño

En esta sección se presentan diferentes artefactos de software que son fundamentales para la construcción del producto, conformando la base y el plan de solución del prototipo web.

5.1.3.1. Diagrama casos de uso

Los diagramas de caso de uso son una descripción de los requerimientos funcionales de un sistema, en él se tienen en cuenta los actores, los escenarios y otros sistemas externos. Se empieza con un evento desde un actor hacia el sistema, es importante tener en cuenta que los casos de uso describen qué hace un sistema, pero no cómo.

Para tener claro el comportamiento del prototipo web se realizó el diagrama de casos de uso presentado en la siguiente figura.

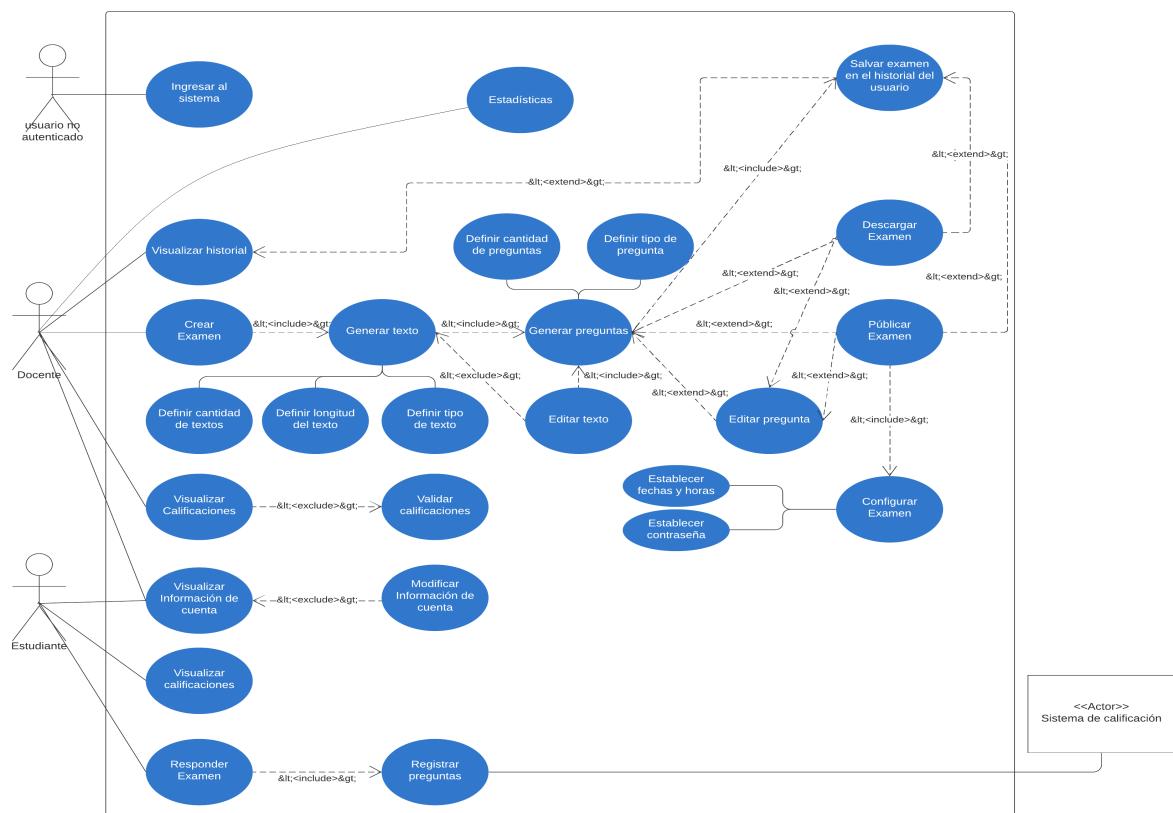


Figura 5.1.2: Diagrama de casos de uso - GQuestions

Fuente: Elaboración propia

5.1.3.2. Modelo de base de datos

Un modelo de base de datos es la estructura lógica que adopta la base de datos, incluyendo las relaciones y limitaciones que determinan cómo se almacenan y organizan, y cómo se accede a los datos. Así mismo, un modelo de base de datos también define qué tipo de operaciones se pueden realizar con los datos, es decir, que también determina cómo se manipulan los mismos, proporcionando también la base sobre la que se diseña el lenguaje de consultas. []

El modelo de base de datos del prototipo web se realizó bajo un modelo de base de datos relacional, en la figura 5.1.3 se presenta el modelo entidad-relación completo.

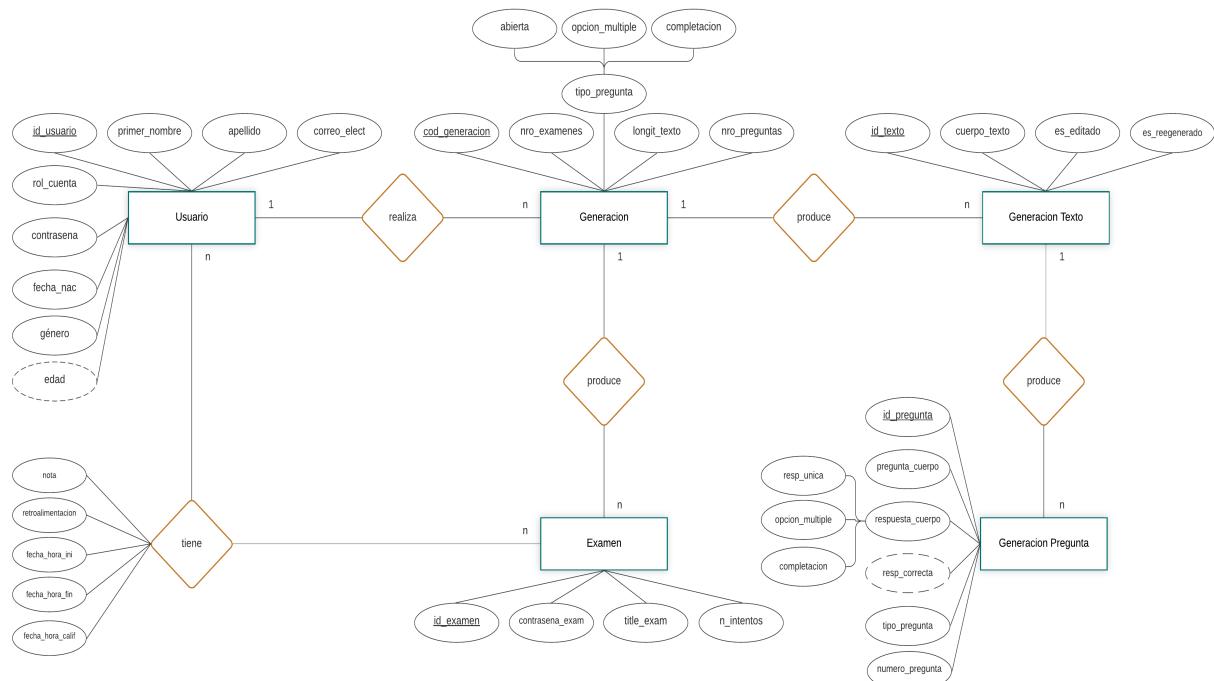


Figura 5.1.3: Modelo Entidad Relación - GQuestions

Fuente: Elaboración propia

5.1.3.3. Diagrama de despliegue

En el diagrama de despliegue se muestra la arquitectura de ejecución del prototipo web, en donde se incluyen nodos, entornos de ejecución tanto de software como de hardware y el middleware que los conecta. El fin de este diagrama es poder entender cómo se desplegará el sistema.

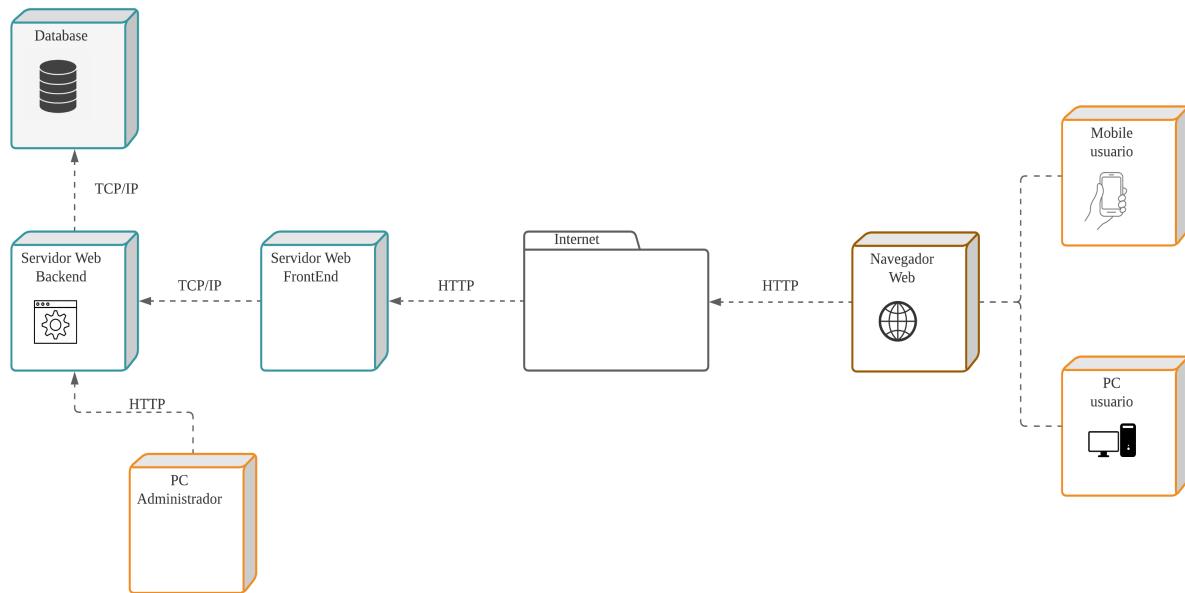


Figura 5.1.4: Diagrama de despliegue - GQuestions
Fuente: Elaboración propia

5.2. Descripción del prototipo web

En esta sección se presenta el prototipo web de manera global, el cual cuenta con cinco (5) módulos principales, generación, examen, estadísticas, calificaciones y usuarios.

El prototipo web está dividido en dos tipos de perfiles, docente y estudiante. El perfil docente es donde se encuentran la mayor parte de módulos debido a la orientación de la solución, este cuenta con los cinco módulos anteriormente mencionados, mientras que el perfil estudiante cuenta con tres (3) de los cinco (5) módulos.

En las siguientes secciones se describe cada módulo y submódulos con funcionalidades que puede realizar cada perfil, además, se describe y se presenta la interfaz gráfica de usuario.

5.2.1. Interfaz de usuario

Para la interfaz gráfica de usuario se comenzó con el diseño de un prototipo basado en las necesidades identificadas en la fase de análisis. Se utilizó la plataforma Figma la cual es

una herramienta de generación de prototipos, el resultado se encuentra en los anexos.

Como se mencionó en la sección 5.2 el prototipo web está compuesto por dos tipos de perfiles, el perfil docente tiene los módulos de generación, examen, calificaciones, estadísticas y usuarios (ajustes de cuenta), entre tanto, el perfil estudiante tiene los módulos examen, calificaciones y usuarios (ajustes de cuenta).



Figura 5.2.1: UI Homepage
Fuente: Elaboración propia

5.2.2. Usuarios

Este módulo contiene toda la información de los usuarios, realiza la gestión de los usuarios que se registran y de los usuarios que inician sesión, esta gestión permite visualizar y modificar la información personal del usuario.

5.2.3. Módulo Generación

Este módulo es el más importantes de GQuestions, contiene varios submódulos que permiten la generación de exámenes y que conforman tres series de pasos vistos en las siguientes secciones.

La palabra generación de aquí en adelante se refiere a el conjunto de exámenes y se utiliza de manera indistinta.

5.2.3.1. Submódulo Parámetros de Generación

Este submódulo representa el primer paso de la generación, permite configurar los parámetros de la generación entre los que están: cantidad de exámenes, longitud de

texto, cantidad de preguntas, tipos de preguntas y el área del texto a generar (ej. Machine learning).

Se puede visualizar el tiempo de espera promedio que tarda la generación de acuerdo a la configuración del mismo.

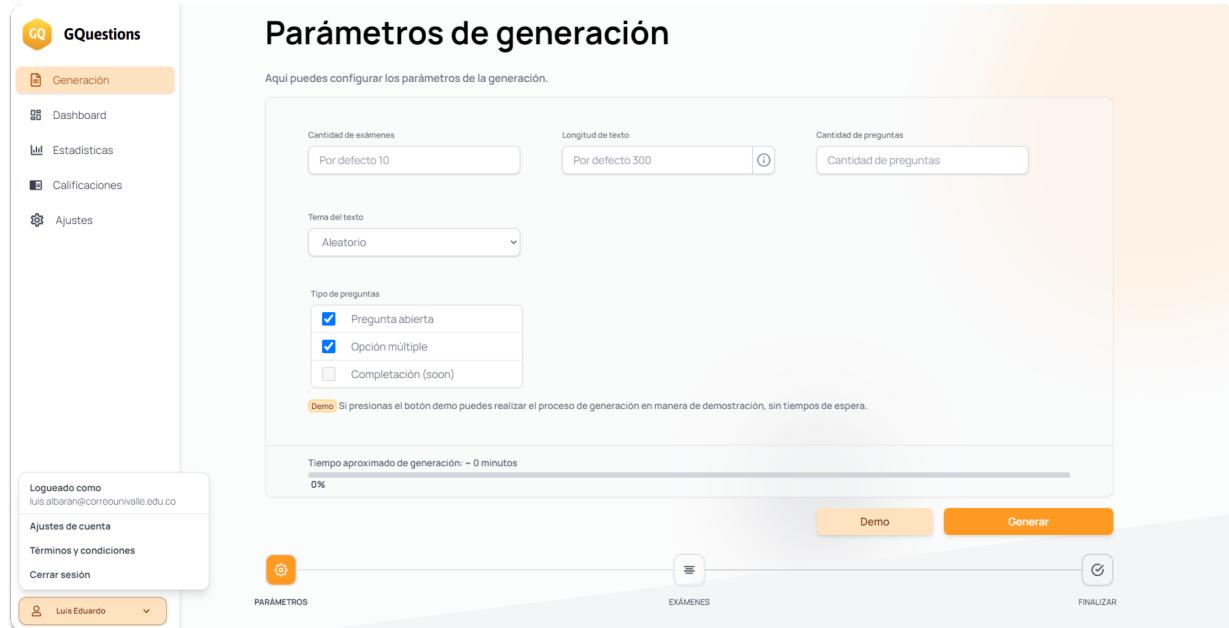


Figura 5.2.2: UI Párametros de generación - docente
Fuente: Elaboración propia

5.2.3.2. Submódulo Revisión Generación

Este submódulo representa el segundo paso de la generación, permite navegar por los exámenes generados, visualizar y modificar cada examen del conjunto, tanto texto base como preguntas generadas, además, permite volver a generar cualquier texto con sus preguntas en caso de no ser adecuado para el docente.

Figura 5.2.3: UI Revisión de generación - docente
Fuente: Elaboración propia

5.2.3.3. Submódulo Configuración conjunto de exámenes

Este submódulo representa el tercer y último paso de la generación, permite configurar el conjunto de exámenes en cuanto a disposición del mismo para los estudiantes, con campos como fecha y hora de inicio y finalización, contraseña, duración y nombre del conjunto de exámenes.

Figura 5.2.4: UI Configuración examen - docente
Fuente: Elaboración propia

5.2.4. Módulo Examen

Este módulo permite a los estudiantes responder los exámenes generados por el docente, accediendo a ellos por medio de un enlace y contraseña, además, se realiza la calificación del examen por parte del sistema. La interfaz gráfica de este módulo es muy similar a la del submódulo de la sección revisión de examen 5.2.6.2.

5.2.5. Módulo Estadísticas

Este módulo muestra la información del usuario relacionada a las generaciones, se puede visualizar el total de número de generaciones, exámenes, textos y preguntas generadas, además, se puede visualizar el número de generaciones por mes.

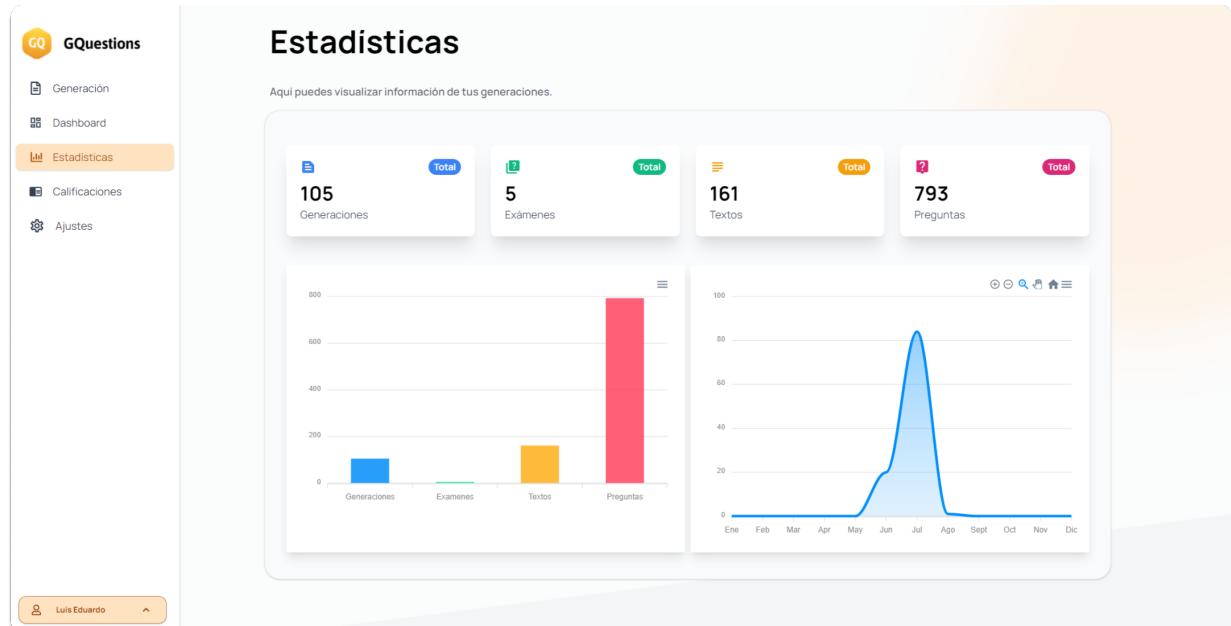


Figura 5.2.5: UI Estadísticas - docente

Fuente: Elaboración propia

5.2.6. Módulo Calificaciones

Este módulo permite visualizar el listado de exámenes que ha aplicado el docente, contiene varios submódulos que permiten revisar las calificaciones de los estudiantes.

TODOS LOS EXÁMENES	
Examen Goloud generation 1	Aplicado: Wed Jul 14 2021 11:16:05 Intentos: 11 Duración: 2 h
Examen Test 2 realtime	Aplicado: Mon Jul 05 2021 19:02:06 Intentos: 11 Duración: 2 h
Examen Generación realtime	Aplicado: Sun Jul 04 2021 18:27:38 Intentos: 11 Duración: 3 h
Examen Sin título	Aplicado: Sun Jul 04 2021 17:31:46 Intentos: 11 Duración: 2 h
Examen Sin título	Aplicado: Thu Jul 01 2021 11:55:00 Intentos: 11 Duración: 2 h
Examen Sin título	Aplicado: Tue Jun 29 2021 23:44:56 Intentos: 11 Duración: 2 h
Examen Sin título	Aplicado: Tue Jun 29 2021 23:41:38 Intentos: 11 Duración: 1 h

Luis Eduardo

Figura 5.2.6: UI Calificaciones
Fuente: Elaboración propia

5.2.6.1. Submódulo Lista de calificaciones

Este submódulo permite visualizar un listado de los estudiantes junto con las calificaciones correspondientes, el docente puede modificar la calificación en caso de ser necesario.

#	FECHA	ESTUDIANTE	ESTADO	CALIFICACIÓN	ACCIÓN
1	Wed Jun 16 2021 14:40:33	Luis Eduardo Albaran Vélez luisalvaraneav@gmail.com	Contestado	3.00	
2	Wed Jun 09 2021 20:43:55	Luis Eduardo Albaran Vélez luisalvaraneav@gmail.com	Contestado	2.30	
3	Tue Jun 01 2021 20:47:54	Luis Eduardo Albaran Vélez luisalvaraneav@gmail.com	Contestado	2.00	
4	-	-	Sin asignar	-	
5	-	-	Sin asignar	-	
6	Wed Jun 09 2021 06:00:00	Luis Eduardo Albaran Vélez luisalvaraneav@gmail.com	Contestado	3.46	
7	-	Luis Eduardo Albaran Vélez luisalvaraneav@gmail.com	Sin contestar	-	
8	-	Luis Eduardo Albaran Vélez luisalvaraneav@gmail.com	Sin contestar	-	

LUIS EDUARDO

Figura 5.2.7: UI Listado de calificaciones
Fuente: Elaboración propia

5.2.6.2. Submódulo Revisión de examen

Este submódulo permite revisar el examen resuelto, visualizar el texto base, las preguntas, las respuestas del estudiante, las respuestas correctas y la calificación de cada pregunta y global.

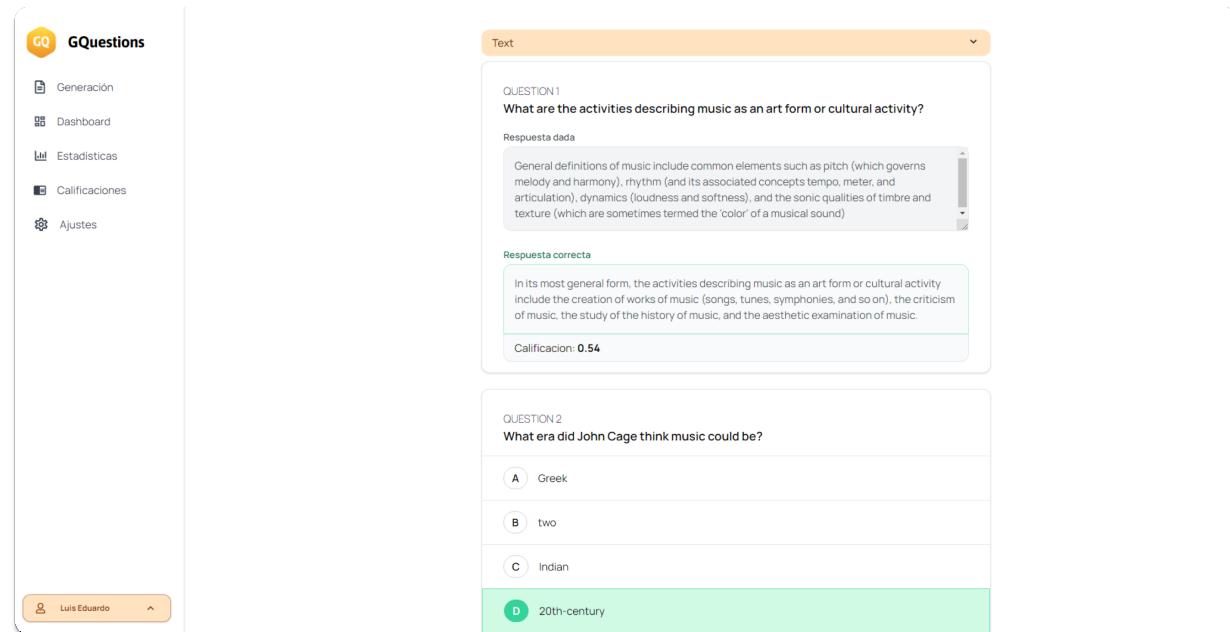


Figura 5.2.8: UI Revisión de examen
Fuente: Elaboración propia

5.3. Implementación

En esta sección se presentan los detalles más importantes de la implementación, se abordan temas relacionados a estrategias utilizadas, frameworks, librerías, lenguajes de programación y estructura del prototipo web.

5.3.1. Detalles de la implementación

Para la construcción del prototipo web se utilizaron diferentes frameworks y librerías, en cuanto al backend se utilizó Django, siendo este un framework web Python que fomenta el desarrollo rápido, limpio y práctico.

En cuanto al frontend se utilizó ReactJS, siendo esta una amplia librería Javascript que permite crear interfaces de usuario y facilita el desarrollo de aplicaciones web.

Como framework de diseño se utilizó Tailwind CSS que permite un desarrollo más ágil y optimizado, es de aplicación sencilla en el JSX o HTML ya que está basado en clases.

En cuanto a controlador de versiones se utilizó Git y remotamente GitHub, siendo una plataforma que ofrece servicios de hosting de repositorios, permitiendo alojar código fuente y mantener el control de versiones (VCS). El control de versiones hace referencia a un sistema que permite rastrear y gestionar cambios realizados en conjuntos de archivos, además, permite a los desarrolladores trabajar simultáneamente en el mismo proyecto.

En el desarrollo del proyecto cada vez que se añadían nuevas características, correcciones u otros cambios importantes se actualizaba el proyecto en GitHub. Es de mencionar que se manejó un único repositorio tanto para frontend como para backend.

Para probar las API creadas en Django se utilizó Postman que es una herramienta dirigida a desarrolladores web que permite el envío de peticiones HTTP REST a cualquier API sin necesidad del desarrollo de un cliente. Su uso fue de gran utilidad debido a las pruebas que permitieron comprobar el correcto funcionamiento del desarrollo backend.

En cuanto al entorno de desarrollo se utilizó el editor de texto Visual Studio Code, este permite trabajar con diferentes lenguajes de programación y aprovechar al máximo variedad de extensiones que ayudan a aumentar la productividad del desarrollador.

Los navegadores utilizados en el desarrollo fueron principalmente Google Chrome y Firefox.

Finalmente, en el despliegue se utilizó Heroku que es una plataforma de servicios de computación en la nube que permite entre otras cosas desplegar aplicaciones en diversos lenguajes de programación. Se crearon dos aplicaciones en Heroku, una para el frontend con ReactJS y otra para el backend con Django.

5.3.1.1. Backend

Como se mencionó previamente para el backend se utilizó Django, permitió implementar todo el modelamiento de base de datos y una de las partes más importantes del desarrollo del proyecto como lo es el servicio API REST, esto se llevó a cabo utilizando Django REST Framework.

Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. Suele considerarse como el contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita del consumidor (la llamada) y el que requiere el productor (la respuesta). []

Por otra parte, un servicio REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (Simple Object Access Protocol), que disponen de una gran capacidad pero también mucha complejidad. A veces es preferible una solución más sencilla de manipulación de datos como REST. []

El marco Django REST (DRF) es una biblioteca Python/Django de código abierto,

madura y bien soportada que tiene como objetivo crear API web sofisticadas. Es un conjunto de herramientas flexible y con todas las funciones con una arquitectura modular y personalizable que hace posible el desarrollo de puntos finales API simples y construcciones REST complicadas. []

El uso de Nested Relationships con Django REST facilitó considerablemente el consumo de las API construidas, ya que, permite obtener datos de las tablas que están relacionadas por medio de llaves foráneas, de esta manera una generación de conjunto de exámenes en el modelo representa la jerarquía mayor, contiene datos de los modelos de base de datos (tablas) de exámenes, calificaciones, y otros perteneciente a dicha generación.

Estructuralmente el desarrollo backend del prototipo web se realizó creando tres grandes directorios, lo que en Django se llamaría StartApp. Se creó un startapp para los usuarios, otro que contenía la parte raíz del proyecto y otro para lo relacionado a las generaciones.

5.3.1.2. Frontend

Se realizó un análisis previo donde se consideró las ventajas y desventajas de utilizar ciertos lenguajes de programación y frameworks frontend, se tuvo en cuenta opciones como el mismo framework Django para realizar todo el prototipo web (backend y frontend), también se considerarán AngularJS, Vue.js y ReactJS; después de valorar cada alternativa, finalmente se eligió ReactJS debido a una curva de aprendizaje más sencilla por haber trabajado con esta librería en ocasiones previas, además, al ser un equipo de trabajo pequeño y al tener las ventajas que ofrece ReactJS como rendimiento, amplia comunidad de desarrolladores, aprovechamiento del lenguaje de programación Javascript, entre otras.

La conjunción de ReactJS con el framework de diseño Tailwind CSS permitió personalización alta en la creación de componentes y así lograr el prototipo de UI creado en las fases iniciales, asimismo, ventajas como facilidad de utilizar clases en JSX para los estilos de los componentes y la optimización a través de PurgeCSS que remueve el código CSS no utilizado al momento de compilar.

Para el consumo API REST se utilizó Fetch de Javascript que permite realizar las peticiones de manera asíncrona usando promesas, esto ayudó a tener mayor flexibilidad y capacidad de control a la hora de realizar las llamadas al servidor, además es soportada de forma nativa en la mayoría de navegadores web.

Estructuralmente el desarrollo frontend del prototipo web se realizó separando en directorios individuales los elementos más importantes como las peticiones API, components, containers, assets, hooks, y tests, esto con el fin de tener un acceso más sencillo a la hora de realizar modificaciones y una mayor organización (estructura) del código.

5.3.1.3. Repositorio

El proyecto se encuentra en un repositorio de GitHub con todos los archivos de código y documentación relacionada:

<https://github.com/luisalvaran17/GQuestions>

Capítulo 6

Pruebas

Índice

6.1. Pruebas unitarias	48
6.1.1. Descripción	48
6.1.2. Detalles	49
6.2. Pruebas de regresión	51
6.2.1. Descripción	51
6.2.2. Detalles	51
6.3. Pruebas de usabilidad	53
6.3.1. Conclusiones	54

En este capítulo se presentan las pruebas realizadas al Prototipo Web GQuestions, en el backend se realizaron pruebas unitarias con Django TestCase y en el frontend se realizaron pruebas de regresión con Selenium WebDriver para comprobar el correcto funcionamiento. También se presentan las pruebas de usabilidad que se realizaron a través de encuestas y los resultados obtenidos.

6.1. Pruebas unitarias

6.1.1. Descripción

Las pruebas unitarias o unit testing lo que busca es corroborar el comportamiento correcto de una unidad de código.

Django proporciona un framework de pruebas con una pequeña jerarquía de clases que se basan en la librería unittest estándar Python, esta librería es adecuada tanto para pruebas unitarias como pruebas de integración, en el backend del prototipo web GQuestions se hicieron pruebas unitarias.

Django ofrece métodos y herramientas que ayudan a probar el comportamiento, estos permiten simular las solicitudes con inserción de datos de prueba e inspección de la respuesta a las solicitudes.

Para escribir pruebas existen clases base de prueba de Django como SimpleTestCase, TransactionTestCase, TestCase, LiveServerTestCase. En el backend se realizaron pruebas utilizando la clase base **TestCase**, se escribieron métodos por separado para verificar que los modelos del proyecto que hacen parte del diseño propio no tuviesen errores.

6.1.2. Detalles

Se probó aspectos del código escrito respecto a los modelos (base de datos) y no de bibliotecas o funcionalidades proporcionadas por Python o Django. Estos aspectos fueron el texto utilizado para etiquetas (atributos), tamaños de campos asignados y valores por defecto asignados.

A continuación, se presentan algunos casos de prueba y posteriormente la creación de estas pruebas en Django con TestCase.

Objetivo de la prueba	Verificar las etiquetas de texto en los atributos de modelos de base de datos en Django							
Técnicas	Uso de <code>django.test.TestCase</code> , se crean clases de prueba que crea una base de datos limpia y se ejecuta cada función de prueba en su propia transacción							
Casos de prueba	<table border="1"> <thead> <tr> <th>Casos de prueba</th></tr> </thead> <tbody> <tr> <td>Tipo de prueba: Unitaria (caja blanca)</td></tr> <tr> <td>Objetivo: Verificar el texto utilizado de etiquetas en los modelos</td></tr> <tr> <td>Caso 1</td></tr> <tr> <td>Descripción: Etiquetas correctas</td></tr> <tr> <td>Entradas: Registros de la base de datos de prueba</td></tr> <tr> <td>Salidas esperadas: Etiquetas correctas</td></tr> </tbody> </table>	Casos de prueba	Tipo de prueba: Unitaria (caja blanca)	Objetivo: Verificar el texto utilizado de etiquetas en los modelos	Caso 1	Descripción: Etiquetas correctas	Entradas: Registros de la base de datos de prueba	Salidas esperadas: Etiquetas correctas
Casos de prueba								
Tipo de prueba: Unitaria (caja blanca)								
Objetivo: Verificar el texto utilizado de etiquetas en los modelos								
Caso 1								
Descripción: Etiquetas correctas								
Entradas: Registros de la base de datos de prueba								
Salidas esperadas: Etiquetas correctas								
Resultados	Se ejecutaron todos los casos de prueba y todas las etiquetas declaradas en los modelos son correctas							
Observaciones								

Cuadro 6.1.1: Caso de prueba etiquetas de campos
Fuente: Elaboración propia

```

1 class GeneracionModel(models.Model):
2     id = models.CharField(primary_key=True, null=False, max_length=255)
3     n_examenes = models.SmallIntegerField(null=False)
4     longit_texto = models.IntegerField(default=200)
5     n_preguntas = models.SmallIntegerField(null=False)
6     inicio_oracion = models.CharField(max_length=30, null=False,
7                                       default='Aleatorio')
7     fecha_generacion = models.DateTimeField(auto_now_add=True, null=
8                                       False)
8     account = models.ForeignKey(Account, related_name='account',
9                                   on_delete=models.CASCADE)

```

Figura 6.1.1: Modelo Generación - backend Django

Fuente: Elaboración propia

En la figura anterior se presenta el modelo Generación definido en Django, aquí se probó las etiquetas de todos los campos, porque a pesar de no haberse especificado explícitamente la mayoría se tiene un diseño que dice cuáles deben ser estos valores. Si no se prueban los valores, entonces no se sabe que las etiquetas de los campos tienen efectivamente los valores deseados. De esta manera, sucede lo mismo con las longitudes de campo especificadas (`max_length`) y valores por defecto (`default`).

Para probar el modelo se comienza por crear una clase `GeneracionModelTest` que recibe como argumento `TestCase`. Se escribe un método de clase `setUpTestData` que permite la creación de datos de prueba iniciales.

```

1 class GeneracionModelTest(TestCase):
2     @classmethod
3     def setUpTestData(cls):
4         GeneracionModel.objects.create(
5             id='ff6eb8f3-38a1', n_examenes=10, n_preguntas=5, account)

```

Figura 6.1.2: Clase Generación de prueba unitaria - backend Django

Fuente: Elaboración propia

Una vez se tiene el método de clase `setUpTestData` se escriben los métodos de clase de pruebas, como sigue:

```

1 def test_n_examenes_label(self):
2     gen=GeneracionModel.objects.get(id='ff6eb8f3-38a1')
3     field_label = gen._meta.get_field('n_examenes').verbose_name
4     self.assertEquals(field_label, 'n_examenes')

```

Figura 6.1.3: Prueba unitaria etiqueta de campo - backend Django

Fuente: Elaboración propia

En este caso se verifica que el campo tenga el valor de etiqueta de campo correcta

`verbose_name`, de manera similar se escribieron los pruebas de tamaño de campo y valores por defecto.

```
1 def test_inicio_oracion_max_length(self):
2     generacion=GeneracionModel.objects.get(id='ff6eb8f3-38a1')
3     max_length = generacion._meta.get_field('inicio_oracion').
4     max_length
5     self.assertEquals(max_length, 30)
```

Figura 6.1.4: Prueba unitaria longitud de campo - backend Django
Fuente: Elaboración propia

```
1 def test_longit_text_default_value(self):
2     generacion=GeneracionModel.objects.get(id='ff6eb8f3-38a1')
3     default_value = generacion.longit_texto
4     self.assertEquals(default_value, 200)
```

Figura 6.1.5: Prueba unitaria valor por defecto - backend Django
Fuente: Elaboración propia

6.2. Pruebas de regresión

6.2.1. Descripción

La prueba de regresión es un tipo de prueba que se realiza para verificar que un cambio de código en el software no afecta la funcionalidad existente del producto. Esto es para asegurarse de que el producto funcione bien con nuevas funciones, correcciones de errores o cualquier cambio en la función existente. Los casos de prueba ejecutados previamente se vuelven a ejecutar para verificar el impacto del cambio [].

En este caso se utilizó Selenium WebDriver para realizar las pruebas de regresión en el prototipo web GQuestions.

Selenium WebDriver se utiliza para automatizar pruebas de aplicaciones web y verificar que funciona como se esperaba. Es compatible con muchos navegadores como Firefox, Chrome y Safari. Utilizando Selenium WebDriver, podemos automatizar las pruebas solo para aplicaciones web. No califica para aplicaciones basadas en ventanas y admite diferentes lenguajes de programación como Javascript, Java, Perl, PHP y otros para escribir scripts de prueba [].

6.2.2. Detalles

Se realizaron pruebas de regresión en todos los módulos y submódulos del frontend del prototipo web (Generación, Examen, Estadísticas, Usuarios, Calificaciones, más). Para

esto se documentaron inicialmente los casos de prueba.

El uso de este tipo de pruebas fue de gran utilidad, porque al realizar cambios en el código como nuevas características permitía ejecutar las pruebas automatizadas para verificar que no se hubiera visto afectada ninguna funcionalidad existente. Las pruebas automatizadas se refiere al diseño, desarrollo y ejecución de scripts a fin de que puedan realizarse esas pruebas sin intervención humana.

Objetivo de la prueba	Verificar el funcionamiento del módulo de generación con todas sus fases (submódulos)						
Técnicas	Uso de Selenium Webdriver, se crean acciones que simulan el comportamiento del usuario para comprobar el funcionamiento del módulo de generación						
Casos de prueba	<table border="1"> <thead> <tr> <th>Caso de prueba 1</th> </tr> </thead> <tbody> <tr> <td> Tipo de prueba: Objetivo: Verificar la generación de exámenes, configuración de publicación de exámenes y navegación por las fases de generación y exámenes generados </td> </tr> <tr> <th>Caso</th> </tr> <tr> <td> Descripción: Se configuró las acciones para navegar por el módulo de generación que consta de tres fases (Configuración de generación, revisión de generación y configuración de publicación de exámenes) </td> </tr> <tr> <td> Entradas: Generación (Cantidad de preguntas), Configuración examen (Contraseña exámenes, título de exámenes, duración de exámenes) </td> </tr> <tr> <td> Salidas esperadas: Publicación de exámenes exitosa </td> </tr> </tbody> </table>	Caso de prueba 1	Tipo de prueba: Objetivo: Verificar la generación de exámenes, configuración de publicación de exámenes y navegación por las fases de generación y exámenes generados	Caso	Descripción: Se configuró las acciones para navegar por el módulo de generación que consta de tres fases (Configuración de generación, revisión de generación y configuración de publicación de exámenes)	Entradas: Generación (Cantidad de preguntas), Configuración examen (Contraseña exámenes, título de exámenes, duración de exámenes)	Salidas esperadas: Publicación de exámenes exitosa
Caso de prueba 1							
Tipo de prueba: Objetivo: Verificar la generación de exámenes, configuración de publicación de exámenes y navegación por las fases de generación y exámenes generados							
Caso							
Descripción: Se configuró las acciones para navegar por el módulo de generación que consta de tres fases (Configuración de generación, revisión de generación y configuración de publicación de exámenes)							
Entradas: Generación (Cantidad de preguntas), Configuración examen (Contraseña exámenes, título de exámenes, duración de exámenes)							
Salidas esperadas: Publicación de exámenes exitosa							
Resultados	<p>Se ejecutaron todos los casos de prueba y se encontraron los siguientes errores:</p> <ul style="list-style-type: none"> - Error en la barra de progreso de la generación, específicamente en la visualización de acuerdo al porcentaje (75% mostrado como 100%) - Carácter “</s>” inesperado en las preguntas generadas - Textos de longitud menor a la especificada por el usuario 						
Observaciones	Errores medios						

Cuadro 6.2.1: Caso de prueba inicio de sesión

Fuente: Elaboración propia

En la implementación de los casos de prueba se escribieron funciones asíncronas en los diferentes archivos separados por módulos que ejecutara cada caso.

```
1 (async function LoginSuccessTest() {
2   const url = 'http://192.168.0.34:3000/'
3   const driver = await new Builder().forBrowser('chrome').build();
4   try {
5     await driver.get(url.toString() + 'login');
6     await driver.findElement(By.id('login')).sendKeys('test1@gmail.com');
7     await driver.findElement(By.id('password')).sendKeys('test1234');
8     await driver.findElement(By.className('btn-primary')).click();
9     let et = "Inicio - GQuestions";
10    await driver.wait(until.titleIs(et), 10000);
11    let at = await driver.getTitle()
12
13    const assertLogin = function (condition, message) {
14      if (!condition)
15        throw Error('Assert login failed: ' + (message || ''));
16      if (condition)
17        console.log('Test login successful');
18    };
19    assertLogin(et === at);
20
21    await driver.close();
22  } finally {
23  }
24 })();
```

Figura 6.2.1: Prueba de regresión Selenium - frontend React
Fuente: Elaboración propia

De la misma manera se realizaron el resto de casos de prueba con Selenium WebDriver, los casos de prueba completos se encuentran en los anexos.

6.3. Pruebas de usabilidad

Las pruebas de usabilidad o de UX son un método de prueba que permite medir qué tan amigable y fácil de usar es un software. Un pequeño grupo de usuario finales utiliza el software para exponer defectos de usabilidad.

El propósito de realizar estas pruebas es para verificar diferentes aspectos importantes:

- Estética y diseño
- Efectividad del sistema
- Eficiencia
- Facilidad de uso
- Exactitud

Para llevar a cabo las pruebas de usabilidad se realizaron dos tipos de encuestas, el primer tipo de encuesta dirigido a un docente del curso Lectura de textos académicos en Inglés IV y el segundo tipo de encuesta dirigido a estudiantes de Inglés. Ambos tipos de encuestas comparten algunas preguntas o similares.

Las encuestas realizadas se encuentran en los anexos.

6.3.1. Conclusiones

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

1. Se logró evidenciar que llevar un proceso adecuado de desarrollo de software aumenta las probabilidades de éxito en términos de lograr los objetivos trazados y dar solución al problema o necesidad inicial. Una planificación correcta, el uso de metodologías ágiles, herramientas de gestión, controladores de versiones, entre otros aspectos, influyen directamente.
2. El estado del arte Procesamiento de Lenguaje Natural se encuentra en un buen momento y en crecimiento acelerado, existen gran variedad de modelos sofisticados que con una buena integración dan solución a problemas del mundo real, como la generación de texto (Text Generation) y la respuesta a preguntas (Question Answering) en el proyecto.
3. Se logró evidenciar que el desarrollo de un software que genera exámenes de inglés utilizando tecnologías de Inteligencia Artificial, en este caso Procesamiento de Lenguaje Natural, puede ayudar a reducir el fraude en exámenes de inglés, debido a que, en efecto cada estudiante tiene un examen diferente basado en el texto generado.
4. La utilización de modelos de Procesamiento de Lenguaje Natural implica altos costos computacionales y su despliegue puede suponer diferentes limitaciones especialmente de rendimiento, tiempos de respuesta, disponibilidad, usabilidad, entre otras.
5. La realización de pruebas de software son indispensables, permitieron detectar errores que con una revisión humana no hubieran sido detectados a tiempo. Los casos de prueba fueron la base para la ejecución apropiada de pruebas, además, la automatización de éstas redujo considerablemente el tiempo que se fuera utilizado en sólo pruebas manuales.

7.2. Trabajos futuros

Durante el desarrollo del proyecto se pudo identificar diferentes aspectos que complementarían al prototipo web:

1. Implementar más tipos de preguntas en la generación de preguntas, por ejemplo, completación, cloze, verdadero o falso, entre otras.
2. Implementar la gamificación en el perfil estudiante y que a través de juegos trasladados a la parte educativa se pueda incentivar el conseguir mejores resultados, adquirir más conocimientos o mejorar en las habilidades del idioma Inglés.
3. Mejorar los tiempos de respuesta cuando se generan exámenes a través de las peticiones API.
4. Ampliar los temas de generación de exámenes a áreas de aprendizaje más específicas en el idioma Inglés.

Bibliografía

- [1] *¿Cómo evitar el fraude en los exámenes en línea con Moodle.* 2018. URL: <https://blog.edu-labs.co/moodle-fraude-examenes-proctoring/>.
- [2] Mehdi Allahyari y col. “Text Summarization Techniques: A Brief Survey”. En: *International Journal of Advanced Computer Science and Applications (IJACSA)* 8 (jul. de 2017), págs. 397-405. DOI: 10.14569/IJACSA.2017.081052.
- [3] Michael Armbrust y col. “A View of Cloud Computing”. En: *Commun. ACM* 53 (abr. de 2010), págs. 50-58. DOI: 10.1145/1721654.1721672.
- [4] Jan Chorowski y Navdeep Jaitly. “Towards Better Decoding and Language Model Integration in Sequence to Sequence Models”. En: ago. de 2017, págs. 523-527. DOI: 10.21437/Interspeech.2017-343.
- [5] C. Crisp. *Kanban.* 2010. URL: <https://www.crisp.se/gratis-material-ochguider/kanban>.
- [6] Universidad de Alcalá Defensor universitario. “El fraude académico en los procesos de evaluación de los aprendizajes: aspectos legales y actuaciones para contrarrestarlo”. En: (jun. de 2018). URL: <https://docplayer.es/93517418-El-fraude-academico-en-los-procesos-de-evaluacion-de-los-aprendizajes.html>.
- [7] EduLabs eProctering. *Certified Moodle Partner of Moodle Pty Ltd in Latin America plataforma online para los docentes y un software sin conexión destinado a los alumnos.* URL: <https://eprctoring.com/>.
- [8] Adi Friedman, Ina Blau y Yoram Eshet-Alkalai. “Cheating and Feeling Honest: Committing and Punishing Analog versus Digital Academic Dishonesty Behaviors in Higher Education”. En: *Interdisciplinary Journal of e-Skills and Lifelong Learning* 12 (ene. de 2016), págs. 193-205. DOI: 10.28945/3629.
- [9] Francesca Gino y Shahar Ayal. “Contagion and Differentiation in Unethical Behavior The Effect of One Bad Apple on the Barrel”. En: *Psychological science* 20 (abr. de 2009), págs. 393-8. DOI: 10.1111/j.1467-9280.2009.02306.x.

- [10] David Guzmán y col. “Metodología ágil Scrumban en el proceso de desarrollo y mantenimiento de software de la norma MoProSoft”. En: *Research in Computing Science* 79 (dic. de 2014), págs. 97-107. DOI: 10.13053/rcs-79-1-8.
- [11] Matthew Henderson y col. “Efficient Natural Language Response Suggestion for Smart Reply”. En: (mayo de 2017).
- [12] Mohammed Ikram y Farookh Hussain. “Software as a Service (SaaS) Service Selection Based on Measuring the Shortest Distance to the Consumer’s Preferences”. En: feb. de 2018, págs. 403-415. ISBN: 978-3-319-75927-2. DOI: 10.1007/978-3-319-75928-9_36.
- [13] Aditya Jain, Gandhar Kulkarni y Vraj Shah. “Natural Language Processing”. En: *International Journal of Computer Sciences and Engineering* 6 (ene. de 2018), págs. 161-167. DOI: 10.26438/ijcse/v6i1.161167.
- [14] Fangtao Li y col. “Deceptive Answer Prediction with User Preference Graph”. En: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, ago. de 2013, págs. 1723-1732. URL: <https://www.aclweb.org/anthology/P13-1169>.
- [15] Rina Elizabeth López Menéndez de Jiménez. “Metodologías ágiles de desarrollo de Software aplicadas a la gestión de proyectos empresariales”. En: (dic. de 2015). URL: <http://hdl.handle.net/10972/2905>.
- [16] Lina Martinez y R. Ramírez. “Fraude académico en universitarios en Colombia: ¿Qué tan crónica es la enfermedad?” En: *Educação e Pesquisa* 44 (jun. de 2017). DOI: 10.1590/s1517-9702201706157079.
- [17] Nina Mazar y On Amir. “The Dishonesty of Honest People: A Theory of Self-Concept Maintenance”. En: *Journal of Marketing Research* 45 (dic. de 2008). DOI: 10.1509/jmkr.45.6.633.
- [18] Angie Estéfany Mosquera Poveda. *La copia en los exámenes, un dilema del mal estudiante*. 2015. URL: <https://hdl.handle.net/20.500.12313/237>.
- [19] Tamera Murdock y Eric Anderman. “Motivational Perspectives on Student Cheating: Toward an Integrated Model of Academic Dishonesty”. En: *Educational Psychologist* 41 (sep. de 2006), págs. 129-145. DOI: 10.1207/s15326985ep4103_1.
- [20] K. Armijos Guillen P. Buñay E. Yépez. “Aplicación de la metodología kanban en el desarrollo del software para generación, validación y actualización de reactivos, integrado al sistema informático de control académico UNACH”. En: (feb. de 2020). URL: <http://dspace.unach.edu.ec/handle/51000/6457>.

- [21] Gary Pavela. “Applying the Power of Association on Campus: A Model Code of Academic Integrity”. En: *The Journal of college and university law* 24 (1997), págs. 97-118.
- [22] María José Pérez Pérez. “Guía comparativa de metodologías ágiles”. En: (2012). URL: <http://uvadoc.uva.es/handle/10324/1495>.
- [23] María José Pérez Pérez. “Metodología Lean-PMI para desarrollo de proyectos de software”. En: (2017). URL: <http://repositorio.udc.cl/jspui/handle/11594/2478>.
- [24] Mauricio Romero, Álvaro Riascos y Diego Jara. *Detección de copia: modelos de respuesta nominal, comparación de índices y pruebas múltiples de copia*. Documentos CEDE 008748. Universidad de los Andes - CEDE, mayo de 2011. URL: <https://ideas.repec.org/p/col/000089/008748.html>.
- [25] 11 IONOS España S.L.U. *¿Qué es Kanban?* URL: <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/que-es-kanban/>.
- [26] L. Shore. *Kanban Tool. ¿Por qué utilizar la metodología Kanban?* 2010. URL: <https://kanbantool.com/es/metodologia-kanban>.
- [27] TestWe. *Plataforma online para los docentes y un software sin conexión destinado a los alumnos*. 2018. URL: <https://testwe.eu/es/>.
- [28] E. López TP. Torres. “Metodologías ágiles en el desarrollo de software”. En: (nov. de 2003). URL: https://rua.ua.es/dspace/bitstream/10045/92767/1/tesis_santiago_leonardo_morales_cardoso.pdf.
- [29] Wei-Tek Tsai, Xiaoying Bai y Yu Huang. “Software-as-a-service (SaaS): Perspectives and challenges”. En: *Science China Information Sciences* 57 (mayo de 2014), págs. 1-15. DOI: 10.1007/s11432-013-5050-z.
- [30] Turnitin. *Educación con integridad, Software*. URL: <https://www.turnitin.com/es>.
- [31] Pasi Tyrväinen y Joona Selin. “How to Sell SaaS: A Model for Main Factors of Marketing and Selling Software-as-a-Service”. En: vol. 80. Jun. de 2011, págs. 2-16. DOI: 10.1007/978-3-642-21544-5_2.
- [32] Urkund. *Plagiarism prevention that simply works, Software*. URL: <https://www.urkund.com/>.
- [33] Adams Yu, Hongrae Lee y Quoc Le. “Learning to Skim Text”. En: (abr. de 2017).

Anexos

Historias de usuario

El documento adjunto se encuentra en la carpeta Diseño como:

- historias_de_usuario_GQuestions_TG.pdf

Historia de usuario	
Número de historia: 1	
Autor: Luis Eduardo Albarán Vélez	
Módulo: Generación	Prioridad: Alta
Responsable: Luis Eduardo Albarán Vélez	
Descripción: Como docente de inglés, quiero indicar la cantidad de exámenes a generar para aplicarlo a la cantidad de estudiantes a evaluar	
Validación: Dado que el usuario docente se encuentre en el módulo de generación, cuando vaya a generar los exámenes, entonces se le pide que indique la cantidad de exámenes a generar	
Fecha creación: 15/02/21	

Cuadro 7.2.1: Historia de usuario 1

Fuente: Elaboración propia

Historia de usuario	
Número de historia: 5	
Autor: Luis Eduardo Albarán Vélez	
Módulo: Generación	Prioridad: Media
Responsable: Luis Eduardo Albarán Vélez	
Descripción: Como docente de inglés, quiero poder editar cualquier segmento de los textos generados en caso de ser necesario para corregir algún error del texto	
Validación: Dado que el usuario se encuentre en el módulo de generación, cuando el texto sea generado, entonces se le permitirá modificar segmentos de los textos	
Fecha creación: 15/02/21	

Cuadro 7.2.2: Historia de usuario 5

Fuente: Elaboración propia

Historia de usuario	
Número de historia: 8	
Autor: Luis Eduardo Albarán Vélez	
Módulo: Generación	Prioridad: Baja
Responsable: Luis Eduardo Albarán Vélez	
Descripción: Como docente de inglés, quiero poder indicar la longitud de los textos a generar	
Validación: Dado que el usuario se encuentre en el módulo de generación, cuando vaya a generar los textos, entonces se le pedirá al usuario que indique la longitud de los textos a generar	
Fecha creación: 15/02/21	

Cuadro 7.2.3: Historia de usuario 8

Fuente: Elaboración propia

Historia de usuario	
Número de historia: 15	
Autor: Luis Eduardo Albarán Vélez	
Módulo: Calificación	Prioridad: Alta
Responsable: Luis Eduardo Albarán Vélez	
Descripción: Como docente de inglés, quiero que el prototipo califique los exámenes y muestre los resultados	
Validación: Dado que el usuario resuelva un examen, cuando el sistema da por terminado el examen, entonces verificará automáticamente las respuestas proporcionadas con las respuestas correctas y mostrará la calificación correspondiente	
Fecha creación: 15/02/21	

Cuadro 7.2.4: Historia de usuario 15

Fuente: Elaboración propia

Justificación selección de algoritmos

El documento adjunto se encuentra en la carpeta raíz como:

- justificacion_seleccion_algoritmos_GQuestions_TG.pdf

Documentos recolectados

El documento adjunto se encuentra en la carpeta raíz como:

- tabla_documentos_recolectados.pdf

Algoritmos recolectados

El documento adjunto se encuentra en la carpeta raíz como:

- tabla_algoritmos_recolectados.pdf

Prototipo de interfaz GQuestions

El documento adjunto se encuentra en la carpeta Diseño como:

- prototipo_de_interfaz_GQuestions_TG.pdf

Manuales de usuario

El documento adjunto se encuentra en la carpeta raíz como:

- manual_de_usuario_docentes_GQuestions.pdf
- manual_de_usuario_estudiantes_GQuestions.pdf

Documento plan de pruebas

El documento adjunto se encuentra en la carpeta raíz como:

- plan_de_pruebas_GQuestions.pdf

Encuestas

El documento adjunto se encuentra en la carpeta Encuestas como:

- encuesta_1_GQuestions_google_forms.pdf
- encuesta_2_GQuestions_google_forms.pdf
- encuesta_3_usabilidad_docente_GQuestions_google_forms.pdf
- encuesta_3_usabilidad_estudiantes_GQuestions_google_forms.pdf