

**UNIVERSIDAD MAYOR REAL Y PONTIFICIA DE
SAN FRANCISCO XAVIER DE CHUQUISACA**

FACULTAD DE CIENCIAS Y TECNOLOGÍA

CARRERA DE INGENIERÍA EN CIENCIAS DE LA

COMPUTACIÓN



PROYECTO DE GRADO

MODELO DE INTELIGENCIA ARTIFICIAL BASADO EN EL

PROCESAMIENTO DEL LENGUAJE NATURAL PARA LA

GENERACIÓN DE CÓDIGO CSS

Licenciatura en Ingeniería en Ciencias de la Computación

Universitario: Alex Tumiri Huanca

Docente: Ing. Oswaldo Velásquez Aroni

Sucre, Mayo de 2024

DECLARACIÓN DE DERECHOS DE AUTOR

Al presentar este Proyecto, como uno de los requisitos para la obtención del Grado Académico de Licenciatura en Ingeniería en Ciencias de la Computación de la Universidad Mayor Real y Pontificia de San Francisco Xavier de Chuquisaca, autorizo a la Dirección de Carrera de Ingeniería en Ciencias de la Computación y/o a la Biblioteca de la Universidad, para que se haga de este Informe un documento disponible para su lectura según las normas de la Universidad.

Asimismo, manifiesto mi acuerdo en que se utilice como material productivo dentro del Reglamento de Ciencia y Tecnología, siempre y cuando esta utilización no suponga ganancia económica, ni potencial.

También cedo a la Universidad de San Francisco Xavier de Chuquisaca los derechos de publicación de este proyecto o parte de ella, manteniendo derechos de autor, por un periodo de treinta meses después de su aprobación.

Alex Tumiri Huanca

C.U. 111-153

Sucre, Mayo de 2024

AGRADECIMIENTOS

A Dios por la constante inspiración.

A todos mis seres queridos por el apoyo incondicional en todo momento.

A mi Familia por todo su cariño y guía incondicional.

A mi Pareja por su constante apoyo incondicional.

A mis amigos y compañeros por la confianza y ayuda.

A mis Docentes por los conocimientos y valores brindados.

Y a la Universidad por una vez más acogerme en sus aulas.

DEDICATORIA

A mis padres, Galo y Mery, por brindarme su amor incondicional.

A mis hermanos, Anael y Luis, por haber depositado su confianza plena en mí.

A mi pareja Yolanda por brindarme el apoyo e inspiración.

A los ingenieros Oswaldo Velásquez Aroni y Carlos W. Pacheco Lora por su guía y enseñanza

*A todos mis seres queridos, que siempre me han apoyado en mis decisiones, les dedico mi
proyecto.*

RESUMEN

El presente proyecto de grado se enfoca en la creación y aplicación de un modelo de inteligencia artificial (IA) destinado a la generación de código CSS en base a etiquetas HTML para facilitar el trabajo de los desarrolladores de software. El objetivo principal es proporcionar una herramienta que aumente la eficiencia y reduzca el tiempo de desarrollo al generar código CSS para vistas y páginas web.

El proyecto se desarrolló utilizando tecnologías como PyTorch, Colab, Hugging Face y El modelo pre entrenado GPT-2 XL, para la implementación del modelo de IA. Se emplearon técnicas de aprendizaje profundo como fine tuning (ajuste fino) y procesamiento de lenguaje natural para entrenar el modelo en grandes conjuntos de datos de código CSS existentes. Esto permitió al modelo aprender patrones y reglas comunes utilizadas en el diseño de interfaces web.

Además, se creó una aplicación web que hace uso del modelo de IA previamente entrenado para generar código CSS en base a etiquetas HTML. La aplicación web se desarrolló utilizando Next.js, TypeScript, JavaScript y Tailwind CSS, aprovechando las capacidades de Next.js para la construcción de aplicaciones web robustas y eficientes. Además, se implementó una API desarrollada en JavaScript con Express para la comunicación entre la aplicación web y el modelo de IA.

Este proyecto ofrece una solución innovadora y práctica para agilizar el desarrollo frontend al proporcionar a los desarrolladores una herramienta de generación de código CSS utilizando inteligencia artificial basado en el campo del Procesamiento del Lenguaje Natural. La combinación de un modelo de IA avanzado con una aplicación web intuitiva y eficiente promete mejorar significativamente la productividad y la calidad del desarrollo de software frontend.

ÍNDICE DE CONTENIDO

CAPÍTULO I

INTRODUCCIÓN

1.1 Antecedentes.....	1
1.2 Situación Problemática.....	3
1.3 Problema Central	5
1.4 Abordaje de Solución	5
1.5 Objetivos.....	7
1.5.1 Objetivo General.....	7
1.5.2 Objetivos Específicos.....	7
1.6 Justificación	8
1.6.1 Justificación Tecnológica.....	8
1.6.2 Justificación Social.....	8

CAPÍTULO II

MARCO CONTEXTUAL

2.1. Análisis de la Situación Actual.....	9
2.1.1 Antecedentes Generales.....	9
2.1.2 Descripción de Tipos de Desarrolladores de Software.....	9

CAPÍTULO III

FUNDAMENTO TEÓRICO

3.1. Antecedente Teórico.....	15
3.1.1 Impacto de la Inteligencia Artificial en el Desarrollo de Software.....	15
3.1.2 Modelos Generadores de textos.....	16
3.1.2.1 ¿Qué es un modelo de Inteligencia Artificial?	18
3.1.3 MOVIMIENTO NO CODE	19
3.2. Marco Teórico Del Contexto.....	20
3.2.1 Lenguaje Css.....	20
3.2.2 La Inteligencia Artificial y sus áreas de aplicación.....	22
3.2.3 Redes Neuronales.....	24

3.2.3.1 ¿Cuáles son los tipos de redes neuronales?	29
3.2.4 Transformers.....	30
3.2.5 Red Transformer.....	31
3.2.6 Dataset.....	45
3.2.6.1 Recolección de Datos para el Dataset.....	47
3.2.6.2 Datasets de Entrenamiento, Prueba, Evaluacion.....	49
3.2.7 Tokenizacion (Tokenization).....	51
3.2.8 Incrustacion (Embedding).....	52
3.2.8.1 Tokenizacion VS Incrustacion.....	53
3.2.9 Fine-Tuning.....	55
3.2.9.1 Origen del Fine-Tuning en NLP.....	55
3.2.9.2 Transfer Learning (Aprendizaje por Transferencia).....	55
3.2.9.3 Fine Tuning vs Transfer Learning (Aprendizaje por Transferencia).....	55
3.3. Marco Teórico De Ingeniería	57
3.3.1 METODOLOGÍA DE DESARROLLO.....	57
3.3.1.1 ¿Qué es la programación extrema (XP)?	57
3.3.1.2 Ciclo de vida de la programación extrema (XP)	57
3.3.1.3 ¿Cuáles son las 12 prácticas de la programación extrema?	58
3.3.2 MODELO PRE ENTRENADO.....	60
3.3.2.1 GPT-2 XL	60
3.3.2.2 Parámetros (Parameters):.....	61
3.3.2.3 Cabezas (Heads):	62
3.3.2.4 Conjunto de datos (Datasets):	63
3.3.2.5 Accesibilidad al Modelo y otras consideraciones:	64
3.3.2.6 Mecanismos de Atención y Representaciones Distribuidas en GPT-2 XL	64
3.3.2.7 Detalles del Modelo GPT-2 XL.....	64
3.3.3 PYTORCH	69
3.3.4 NEXTJS PARA EL DESARROLLO WEB	71
3.3.5 PARADIGMA DE PROGRAMACIÓN	72
3.3.6 COLAB	73

3.3.7 VISUAL STUDIO CODE.....	73
3.3.8 MONGODB	74
3.3.9 POSTMAN	75

CAPÍTULO IV

METODOLOGÍA APLICADA AL PROYECTO

4.1. Metodología De La Investigación	76
4.1.1 Metodología General.....	76
4.1.2 Métodos, Técnicas y Herramientas.....	76
4.2. Metodología De Ingeniería	77
4.2.1 Desarrollo del Proyecto.....	77
4.2.2 Planificación del Proyecto.....	78

CAPÍTULO V

INGENIERÍA DEL PROYECTO

5.1. Fase de Exploración: Análisis Preliminar del Proyecto	79
5.1.1 PROCESO DE REQUERIMIENTOS	79
5.1.1.1 Identificación de Actores	79
5.1.1.2 Descripción de Actores.....	80
5.1.1.3 Requerimientos Funcionales.....	81
5.1.1.4 Requerimientos no Funcionales.....	83
5.2. Fase de Elaboración: Definición de La Arquitectura del Sistema.....	84
5.2.1 DIAGRAMA DE CASOS DE USO GENERAL	84
5.2.2 DIAGRAMA DE CASOS DE USO CLASIFICACIÓN POR PRIORIDAD	85
5.2.3 DIAGRAMA DE CASOS DE USO CLASIFICADOS POR FUNCIONALIDAD	86
5.2.4 DESCRIPCIÓN DE LOS CASOS DE USO	87
5.2.5 PROCESO DE ANÁLISIS Y DISEÑO	89
5.2.6 ANÁLISIS DE RIESGOS	89
5.2.7 DIAGRAMAS DE FRONTERA DEL MODELO	92
5.2.7.1 Diagrama Frontera del Administrador.....	92
5.2.7.2 Diagrama Frontera del Desarrollador Web	92
5.2.7.3 Diagrama Frontera del Desarrollador Frontend.....	93

5.2.8 Diagrama de Componentes del Sistema.....	93
5.3. Fase de Desarrollo	95
5.3.1 Datasets de Entrenamiento, Validación y Prueba.....	95
5.3.1.1 Diagrama de Flujo	95
5.3.1.2 Características del Dataset.....	96
5.3.1.3 Recolección de Datos	97
5.3.1.4 Procesamiento de Datos.....	102
5.3.1.5 División en conjunto de datos de entrenamiento, validación y prueba.....	103
5.3.2 Modelo de Inteligencia Artificial.....	104
5.3.2.1 Diagrama de Flujo.....	104
5.3.2.2 Implementación del Modelo.....	105
5.3.2.3 Tokenización e Incrustación.....	111
5.3.2.4 Fine-Tuning.....	113
5.3.2.5 Entrenamiento.....	118
5.3.2.6 Resultados de rendimiento del Modelo	133
5.3.2.7 Conclusiones con respecto a la Implementación del Modelo.....	138
5.3.3 API de servicios web para el Modelo.....	139
5.3.4 Sistema Generador de CSS.....	141
5.3.4.1 Estructura de Archivos de Sistema Generador de Código CSS.....	141
5.3.4.1 Barra de navegación vertical.....	142
5.3.4.2 Visualizador.....	143
5.3.4.3 Inicio de sesión.....	144
5.3.4.4 Historial.....	145
5.3.5 API de servicios web para el Sistema Generador de CSS.....	145
5.3.5.1 Estructura de Archivos de la API del Sistema Generador de Código CSS.....	146
5.3.5.2 CRUD de Usuarios.....	147
5.3.5.3 CRUD de Vistas.....	149
5.4. Fase de Pruebas.....	152
5.4.1 Plan de Pruebas.....	152

5.4.2 Cronograma de Ejecución.....	156
------------------------------------	-----

CAPÍTULO VI

ANÁLISIS DE RESULTADOS

6.1. Presentación de Resultados.....	157
6.1.1 Validaciones Orientadas al Cumplimiento de Objetivos.....	157
6.2. Plan de Puesta en Marcha.....	159
6.2.1 Hardware y Software.....	159
6.2.1.1 Hardware.....	160
6.2.1.2 Software.....	160
6.2.2 Capacitación de Usuarios.....	161
6.3. Costos.....	161
6.3.1 Costos de Desarrollo y Esfuerzo.....	161
CONCLUSIONES.....	162
RECOMENDACIONES.....	163
REFERENCIAS BIBLIOGRÁFICAS.....	164
BIBLIOGRAFÍA.....	168
GLOSARIO DE TÉRMINOS.....	169
ANEXOS.....	168
ANEXO A: ESTIMACIÓN DEL COSTO DEL SOFTWARE DESARROLLADO.....	173
ANEXO B: MANUAL DE USUARIO.....	175

ÍNDICE DE TABLAS

Tabla 1. Desarrollo del Proyecto	78
Tabla 2. Descripcion del Administrador.....	80
Tabla 3. Descripcion de Desarrollador Web	80
Tabla 4. Descripcion de Desarrollador Frontend.....	81
Tabla 5. Descripcion de los Casos de Uso.....	88
Tabla 6. Plan de Pruebas.....	152
Tabla 7. Prueba de unidad: Inicio de Sesion de Usuario	154
Tabla 8. Prueba de Integracion: Visualizador	155
Tabla 9. Prueba de Aceptacion: Caso de Uso : Gestionar Historial	155
Tabla 10. Requisitos minimos de Hardware.....	160
Tabla 11. Requisitos minimos de Software	160
Tabla 12. Calculo de Puntos Objeto	173
Tabla 13. Costos de material de escritorio, equipos y conectividad.....	174

ÍNDICE DE FIGURAS

fig 1. Arquitectura del Software de Generación Css.....	6
fig 2. Diagrama de Tipos de Desarrolladores.....	9
fig 3. Principales de Tipos de Desarrolladores en 2023.....	11
fig 4. Cuota de Mercado de los Sistemas Operativos de 2022.....	12
fig 5. Diagrama de Casos de uso de un Modelo de Inteligencia Artificial.....	14
fig 6. El Deep learning fortalece a otras técnicas de IA	16
fig 7. Diagrama de Proceso de Desarrollo de Chat GPT	17
fig 8. Arquitectura de una Red Neuronal Simple.....	29
fig 9. Diagrama de bloques general de una Red Transformer.....	32
fig 10. Embedding de entrada de la Red Transformer.....	33
fig 11. Codificador de posición de la Red Transformer	33
fig 12. El bloque de codificación de la Red Transformer.....	34
fig 13. Detalle del primer codificador en la Red Transformer	34
fig 14. La Relación Entre Cada Palabra De La Frase 'I Love Italian Food'	35
fig 15. Obtención De Los 'Queries', 'Keys' Y 'Values'	35
fig 16. Comparación De 'Queries' Y 'Keys' Y Obtención De Los Puntajes	36
fig 17. Aplicación De La Función 'Softmax' A La Matriz De Puntajes	37
fig 18. La Salida Del Bloque Atencional.....	37
fig 19. Los Multiples Bloques Atencionales.....	38
fig 20. Los Elementos del Bloque Residual.....	39
fig 21. La Etapa de Salida del Codificador: Una Red Neuronal Seguida de un Bloque Residual.....	39
fig 22. Salida Resultante Del Primer Codificador	40
fig 23. Elementos del Decodificador	41
fig 24. Momento Del Enmascaramiento	42
fig 25. El Resultado Del Enmascaramiento	43
fig 26. El Bloque Atencional Del Decodificador	44
fig 27. Múltiples Decodificadores Para Encontrar Relaciones Entre Palabras Y Grupos De Palabras A Diferentes Niveles	44

fig 28. Salida Del Bloque De Decodificación	45
fig 29. Tokenizacion, Incrustacion y Procesamiento del Modelo	54
fig 30. Fine Tuning Aplicado al Transformer para la Generacion de codigo CSS.....	56
fig 31. Grafico de modelos pre-entrenados de Hugging Face según sus parametros	61
fig 32. Grafico de modelos pre-entrenados de Hugging Face según sus Cabezas	63
fig 33. Rendimiento del Modelo GPT-2 XL en Tareas NLP de Comprension Lectora, Traducion, Resumen y Respuesta a Preguntas	69
fig 34. Resultados del Modelo GPT-2 XL.....	69
fig 35. Diagrama de Gantt	78
fig 36. Diagrama de Actores.....	79
fig 37. Diagrama General de Casos de Uso.....	84
fig 38. Diagrama de Casos de Uso del Usuario	85
fig 39. Diagrama de Casos de Uso del Administrador	85
fig 40. Diagrama de Casos de Uso de Generacion de codigo CSS	86
fig 41. Diagrama de Casos de Uso de Edicion de Codigo en Tiempo Real	86
fig 42. Diagrama Frontera del Administrador	92
fig 43. Diagrama Frontera del Desarrollador Web	92
fig 44. Diagrama Frontera del Desarrollador Frontend	93
fig 45. Diagrama de Componentes del Sistema.....	93
fig 46. Diagrama de Flujo del Desarrollo del Dataset de Entrenamiento, Validacion, Prueba .	95
fig 47. CSS Grid Generator	97
fig 48. Ultimate CSS Gradient Generator.....	98
fig 49. CSS CodeGenerators.....	98
fig 50. The Ultimate CSS Generator	99
fig 51. EnjoyCSS	99
fig 52. Repositorio de Bootstrap.....	100
fig 53. Repositorio de Normalize.css.....	100
fig 54. Repositorio de Materialize CSS	101
fig 55. Repositorio de Awesome CSS	101
fig 56. Ejemplo de Dataset	102
fig 57. Diagrama de Flujo del Desarrollo del Modelo IA	104

fig 58. Creacion del Dataset	107
fig 59. Creacion de Diccionario de rutas de los diferentes Datasets preprocesados	107
fig 60. Creacion de Dataset de Entrenamiento	108
fig 61. Cargando el Modelo GPT-2 XL.....	109
fig 62. Cargando el Tokenizador GPT2Tokenizer para la Tokenizacion del Dataset.....	111
fig 63. Creacion de Sub Datasets.....	113
fig 64. Creacion de la Metrica de Evaluacion	114
fig 65. Creacion de Funcion para computar las Metricas	115
fig 66. Implementacion de TrainingArguments	116
fig 67. Implementacion del Trainer	117
fig 68. Entrenamiento del Modelo	119
fig 69. Resultados del Entrenamiento del Modelo con 3 epochas.....	121
fig 70. Resultados del Entrenamiento del Modelo con 10 epochas.....	124
fig 71. Resultados del Entrenamiento del Modelo con 10 epochas y ajustes adicionales de hiperparametros	127
fig 72. Resultados del Entrenamiento del Modelo hasta la epoca 21	129
fig 73. Resultados del Entrenamiento del Modelo hasta la epoca 35	130
fig 74. Resultados del Entrenamiento del Modelo con Overfitting Alto Metrica Precision ...	134
fig 75. Resultados del Entrenamiento del Modelo con Overfitting Alto Metrica Perdida	134
fig 76. Resultados del Entrenamiento del Modelo con Overfitting Medio Metrica Perdida...	135
fig 77. Resultados del Entrenamiento del Modelo con Overfitting Medio Metrica Precision	135
fig 78. Resultados del Entrenamiento del Modelo con Overfitting Bajo Metrica Precision ...	136
fig 79. Resultados del Entrenamiento del Modelo con Overfitting Bajo Metrica Perdida.....	136
fig 80. Resultados del Entrenamiento del Modelo Finales Metrica Precision.....	137
fig 81. Resultados del Entrenamiento del Modelo Finales Metrica Perdida	137
fig 82. Estructura de archivos general del Sistema Generador de codigo CSS	141
fig 83. Componentes del Sistema Generador de codigo CSS.....	142
fig 84. Paginas del Sistema Generador de codigo CSS	142
fig 85. Estructura de archivos general de la API para el Sistema Generador de codigo CSS .	146
fig 86. Rutas de la API para Sistema Generador de codigo CSS	146
fig 87. Modelos de la API para Sistema Generador de codigo CSS	146

fig 88. Peticion POST de usuarios de la API.....	147
fig 89. Peticion GET de usuarios de la API.....	148
fig 90. Peticion PATCH de usuarios de la API	148
fig 91. Peticion DELETE de usuarios de la API	149
fig 92. Peticion POST de vistas de la API.....	150
fig 93. Peticion GET de vistas de la API	150
fig 94. Peticion PATCH de vistas de la API.....	151
fig 95. Peticion DELETE de vistas de la API.....	151
fig 96. Diagrama de Validacion del Plan de Pruebas	153
fig 97. Barra de Navegacion	157
fig 98. Visualizador de codigo HTML, CSS y Vista resultante	158
fig 99. Inicio de Sesion	158
fig 100. Historial de Estilos Generados	159

CAPÍTULO I

INTRODUCCIÓN

1.1 Antecedentes

El 12 de marzo de 1989, Tim Berners-Lee publicaba un "vago espacio" lo que más adelante se convertiría en un hito para que toda la humanidad pudiera compartir ideas y conocimientos a nivel mundial. La World Wide Web (la Web) nació en el CERN, el Centro Europeo de Física Nuclear, en Ginebra (Suiza), de la mano del ingeniero y físico británico Tim Berners-Lee como un sistema de intercambio de datos entre los 10.000 científicos que trabajaban en la institución. Hoy es una red inabordable e intangible de documentos, imágenes y protocolos que componen la telaraña de información que crece a pasos de gigante. (mdn, 2023)

Una página web es un documento electrónico accesible a través de un navegador web, que puede contener diferentes tipos de contenido como texto, imágenes, videos, enlaces y otros elementos multimedia. Las páginas web son la base de la experiencia en línea y pueden variar en complejidad, desde simples sitios de texto hasta aplicaciones web interactivas y complejas.

Se conoce como desarrollo web al proceso de crear y mantener un sitio web que sea funcional en internet, a través de diferentes lenguajes de programación, según el modelo y la parte de la página que corresponda. Cada sitio tiene una URL única que lo distingue de los demás en la red informática mundial.

Un sitio web puede clasificarse de diferentes formas. Para cuestiones de desarrollo web principalmente se divide en dos partes, aunque muchas veces pueden existir otras áreas de desarrollo.

El desarrollador Frontend se encarga de la parte que interactúa con el usuario, tanto en imagen como en función. Por ello está íntimamente relacionada con la experiencia del usuario (UX) y la interfaz de usuario (IU).

El desarrollador Backend se encarga de la parte que está en contacto directo con el servidor; es donde se aplica el código de programación para crear la estructura. Permanece en un segundo plano a cargo de la accesibilidad, actualización, bases de datos y cambios del sitio.

En la actualidad el lenguaje de diseño y programación CSS tiene muchas ventajas, características, librerías, API's y es utilizado en todos los navegadores y plataformas web del mundo, siendo así un estándar en el desarrollo de interfaces de usuario y en el desarrollo de aplicaciones web, el problema no reside en que la tecnología CSS sea una tecnología ineficiente para el desarrollo de software, o que tenga problemas de compatibilidad, o una ruta de aprendizaje compleja, todo lo contrario.

Los desarrolladores de software sobre todo especializados en la implementación de diseños e interfaces gráficas durante mucho tiempo han intentado hacer su trabajo más eficiente y productivo en esta exhaustiva búsqueda de eficiencia y productividad los mismos programadores y ingenieros de software, han desarrollado diversas técnicas y herramientas las cuales facilitan su trabajo a la hora de crear software.

Existen muchas librerías y herramientas de CSS que pueden ayudarnos a generar plantillas y diseños web, estas herramientas aumentan la productividad y eficiencia siendo algunas de estas Bootstrap, Materialize CSS, Tailwind CSS, Semantic UI, etc.

Según el sitio web BuiltWith, que rastrea el uso de tecnologías web en todo el mundo, Bootstrap es la librería de CSS más utilizada en la actualidad, con un 16% de participación de mercado. Le sigue Materialize CSS con un 0,9%, Foundation con un 0,8%, Tailwind CSS con un 0,6%, Semantic UI con un 0,4%, y Bulma con un 0,2%.
(BuiltWith, 2023)

Las herramientas anteriormente mencionadas también brindan un diseño estándar el cual es fácilmente reconocible siendo este el caso de Bootstrap por lo cual no agrega valor al diseño de la interfaz.

1.2 Situación Problemática

Los problemas encontrados al desarrollar y diseñar una web son:

- Los ingenieros de software se enfrentan constantemente a la complejidad de implementar un diseño web efectivo y la gran cantidad de líneas de código requeridas al desarrollar interfaces de usuario, diseño y estilos utilizando el lenguaje de diseño gráfico CSS. Muchos desarrolladores pueden encontrarse con un proceso de desarrollo lento y tedioso debido a la necesidad de escribir código repetitivo constantemente.
- Estas librerías, frameworks y APIs anteriormente mencionadas comúnmente carecen de flexibilidad a la hora de desarrollar ofreciendo muchas veces estándares de diseños con colores predeterminados y que para lograr el resultado deseado por el desarrollador se debe escribir más código e incluso tener más conocimientos sobre la librería que se esté utilizando, muchas veces el trabajo solo se incrementa tanto en líneas de código como en tiempo de desarrollo, reduciendo la productividad y aumentando la complejidad del desarrollo y obviamente aumentando también el tiempo de desarrollo, también es cierto que el uso excesivo de estas herramientas puede llevar a una falta de flexibilidad y creatividad en el diseño e implementación de una aplicación o sitio web. Por lo tanto, es importante que los desarrolladores encuentren el equilibrio adecuado entre el uso de estas herramientas y la implementación de soluciones personalizadas y creativas.
- El proceso de desarrollo de estilos CSS no solo depende, que el desarrollador de software conozca muy bien la tecnología y los conceptos de la misma si no que tenga conocimientos en el área de diseño de interfaces, el cual es importante para tener éxito con el desarrollo de la Interfaz de Usuario (UI) y Experiencia de usuario (UX), mientras que uno hace referencia a la experiencia y sensación del usuario, el otro está dirigido hacia un lado más racional de la navegación, también se deben considerar otros desafíos mayores como un diseño web efectivo, esto afecta el proceso de desarrollo de programadores sin experiencia y con poco conocimiento de la tecnología.

- Por lo general las diferentes herramientas para CSS que existen en la actualidad son muy diversas y muchas veces se obtiene con estas el mismo resultado, también pueden tener una curva de aprendizaje de larga duración, sin mencionar que se debe conocer muy bien los conceptos y sintaxis del lenguaje CSS, por esta razón no son alternativas factibles para desarrolladores sin experiencia con CSS o que se estén iniciando en el desarrollo de interfaces de usuario. Existen varias herramientas de CSS que pueden tener una curva de aprendizaje larga e ineficiente para algunos desarrolladores, dependiendo del nivel de experiencia y habilidades técnicas. Algunas de estas herramientas son los preprocesadores de CSS (Sass, Less, Stylus y PostCSS, etc) cada una de estas tecnologías tiene su propia sintaxis y estructura, que tienden a ser más complejas que el CSS convencional, lo que requiere tiempo y esfuerzo para aprender.
- El desarrollo de software con la tecnología CSS como con otras tecnologías tiende a ser repetitivo, poco intuitivo, ineficiente para el ingeniero de software a causa de las malas prácticas, desconocimiento de la tecnología CSS y otros factores que hacen del ingeniero de software no sea productivo a la hora de desarrollar software de buena calidad.

1.3 Problema Central

El uso de CSS en diseño web para principiantes en el desarrollo web es complejo y requiere de muchas líneas de código, también de librerías y frameworks estandarizadas que limitan la flexibilidad y creatividad. La curva de aprendizaje es pronunciada y la necesidad de comprender conceptos fundamentales aumentan los errores y la ineficiencia en la escritura y mantenimiento del código CSS, esto conduce a retrasos en el desarrollo y en la entrega de productos de calidad inferior.

1.4 Abordaje de la Solución

El desarrollo de un modelo de inteligencia artificial basado en el procesamiento del lenguaje natural que generará estilos CSS predefinidos y con características específicas, reconociendo el código HTML introducido por el desarrollador de software en una aplicación web intuitiva y fácil de usar que tendrá como motor de generación de código el modelo anteriormente mencionado, para reducir de manera notable el tiempo de desarrollo, aumentando la productividad y eficiencia del ingeniero de software, evitando que el ingeniero en cuestión escriba código repetitivo reduciendo así el tiempo de desarrollo con el cual el desarrollador de software podrá gestionar de manera más productiva su tiempo para optimizar y desarrollar un código de buena calidad y por consiguiente un mejor producto.

Se recopilará un gran volumen de datos de estilos CSS para el preprocesamiento de las diferentes características de la sintaxis del lenguaje CSS y clasificación de datos obtenidos para crear dos diferentes sets de datos que serán utilizados para el entrenamiento y pruebas de nuestro modelo generador de código CSS.

Como alternativa tecnológica para el desarrollo del modelo generador de código CSS se considera, luego de un análisis profundo e investigación sobre el estado actual del desarrollo de modelos de inteligencia artificial, es difícil ignorar el avance que se realizó en los últimos años en el área del Procesamiento del lenguaje natural (NLP) es una tecnología que brinda a las computadoras la capacidad interpretar, comprender el lenguaje humano esta tecnología será de vital utilidad para reconocer, clasificar y extraer texto para que el modelo pueda aprender la sintaxis del lenguaje de programación CSS.

La arquitectura Transformer de redes neuronales será utilizada para el entrenamiento y optimización del modelo, “[...]brinda la innovadora técnica del procesamiento en paralelo de la secuencia de parámetros introducidos a diferencia de las redes neuronales recurrentes donde los parámetros se procesan de forma serial”. (Sotaquirá, 2020).

Se implementarán los módulos de codificación y decodificación, así como las técnicas de embedding y los bloques atencionales, toda esta arquitectura es eficiente e innovadora en el desarrollo de modelos de procesamiento de lenguaje natural y generación de texto específicamente, optimizando y ajustando la arquitectura Transformer a nuestro caso en específico obtendremos como resultado un eficiente y potente modelo de generación de código CSS.

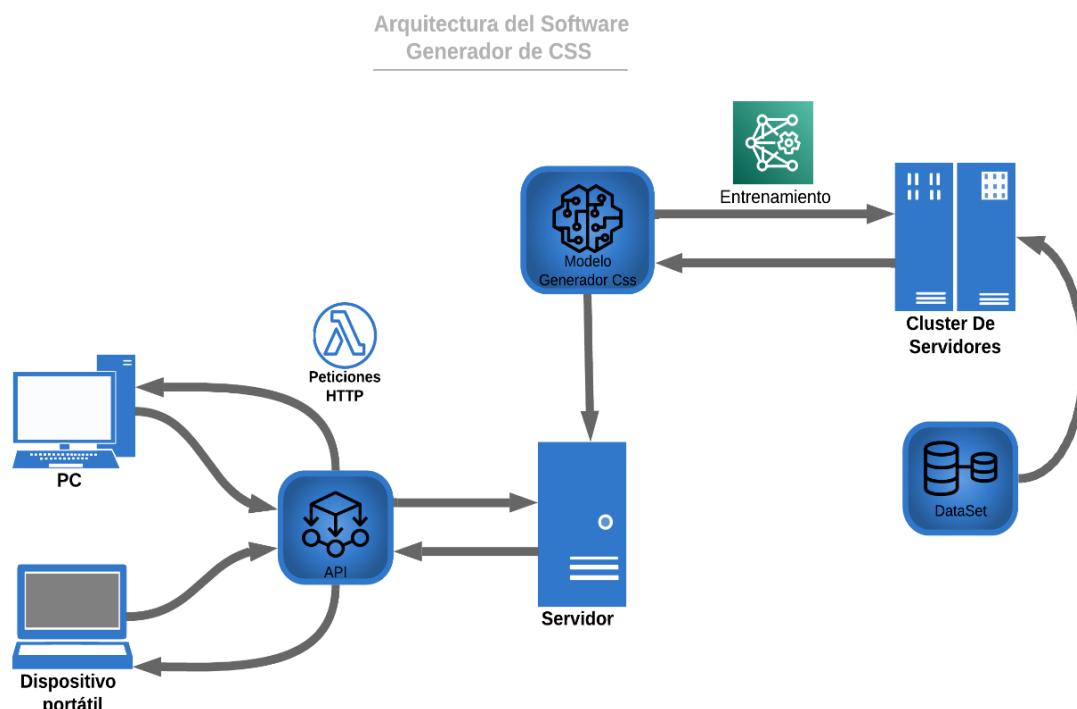


Figura 1. Arquitectura del Software de Generación CSS
Fuente: Elaboración Propia

1.5 Objetivos

1.5.1 Objetivo General

Desarrollar un modelo de inteligencia artificial basado en el campo del Procesamiento del Lenguaje Natural para la generación de código CSS predefinido, que facilite el trabajo de los desarrolladores de software, reduciendo así el tiempo de desarrollo al generar diseños predefinidos y personalizables.

1.5.2 Objetivos Específicos

- ✓ Análisis bibliográfico del lenguaje del procesamiento natural
- ✓ Diseñar un algoritmo de generación de código CSS
- ✓ Reducir la cantidad el código repetitivo y líneas de código CSS.
- ✓ Desarrollar un prototipo de aplicación web para generar código CSS y brinde una previsualización en tiempo real de vistas generadas en base a etiquetas HTML.
- ✓ Realizar entrenamiento y pruebas al Modelo de generación de código CSS.

1.6 Justificación

1.6.1 Justificación Tecnológica

Las herramientas que se utilizan en la actualidad no son suficientes para generar código Css intuitivo y con buen diseño dado que son generadores de código estándar los cuales no solo se concentran en una tecnología en particular si no en diferentes tecnologías, esto hace que estos modelos generadores de código sean más generalistas que especialistas, muchos de los generadores actuales son muy eficientes sin embargo no brindan una interfaz dedicada para el desarrollo de software e implementación de código y está claro que muchas de las herramientas en la actualidad generan código pero no generan otras alternativas en cuestión de vistas y ejemplos de cómo se visualizara la implementación de código Css una vez integrado a los proyectos de los desarrolladores de software y aunque en la actualidad día a día se van desarrollando plugins y diferentes aplicaciones para el uso de estos generadores de código. Esto nos brinda muchas alternativas y oportunidades para el desarrollo de modelos de Inteligencia artificial generadores de texto aplicando NLP (Procesamiento del Lenguaje Natural), tanto en el software como en la generación de código CSS.

La arquitectura Transformers de redes neuronales que se empleara para la optimización y entrenamiento de nuestro modelo Sistema de generación de código CSS mediante la entrada de código HTML.

1.6.2 Justificación Social

El desarrollo web, incluido el diseño y la implementación de CSS, puede ser abrumador para los principiantes debido a la complejidad y la cantidad de información que necesitan asimilar. Al ofrecer asistencia personalizada y contextualizada a través del modelo de inteligencia artificial, se puede ayudar a reducir la curva de aprendizaje y hacer que el proceso sea más accesible y menos intimidante.

Al eliminar las barreras para los programadores principiantes, brindando la oportunidad de contribuir con nuevas ideas y soluciones creativas al mundo del desarrollo web. Esto puede llevar a una mayor innovación en el diseño y la implementación de sitios web y aplicaciones, enriqueciendo la experiencia en línea para los ingenieros de software.

CAPÍTULO II

MARCO CONTEXTUAL

2.1. Análisis de la Situación Actual

2.1.1 Antecedentes Generales

El término "desarrollador frontend" comenzó a utilizarse de manera más prominente en la década de 2000, a medida que la web se volvía más compleja y se buscaba una manera de diferenciar las responsabilidades entre los desarrolladores que trabajaban en la parte visual e interactiva de los sitios web y aquellos que trabajaban en la lógica del servidor y la parte detrás de escena. (BuiltWith, 2023)

2.1.2 Descripción de Tipos de Desarrolladores de Software

En la actualidad hay diversos tipos de desarrolladores para nuestro caso de estudio presentaremos 11 diferentes tipos de desarrolladores de software en el siguiente organigrama.

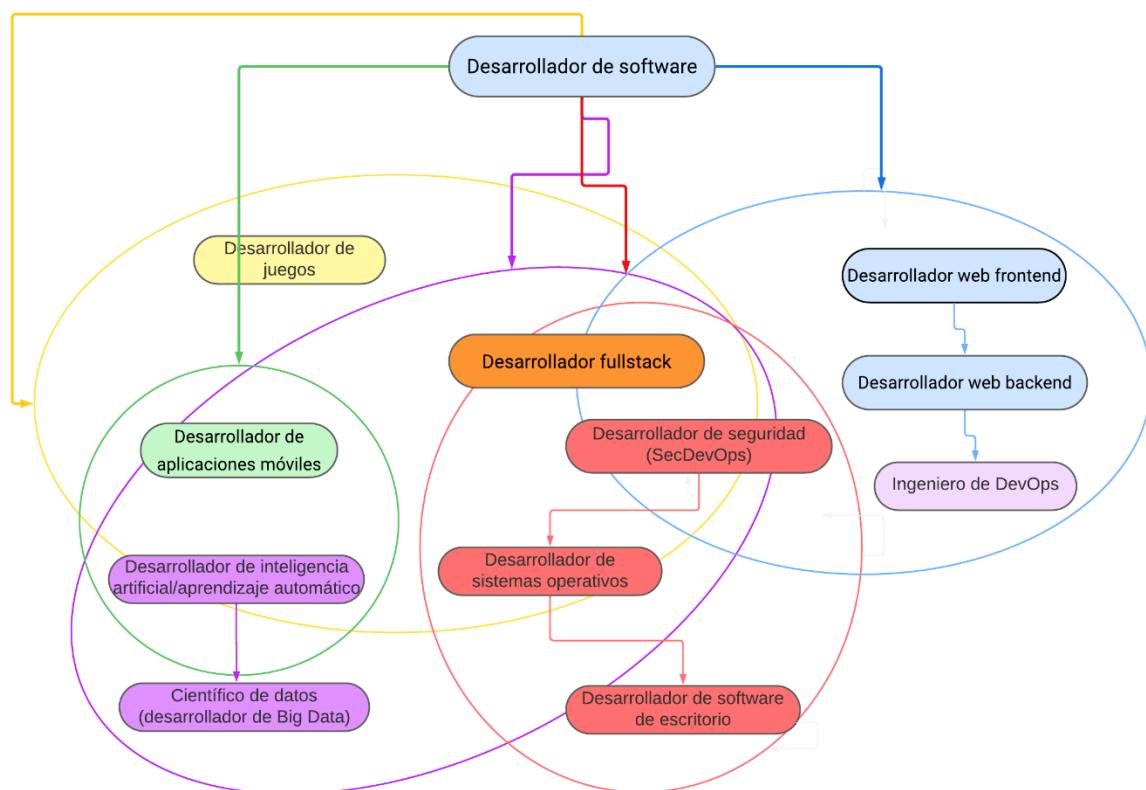


Figura 2. Diagrama de Tipos de Desarrolladores
Fuente: Elaboración Propia

Desarrollador web frontend: Un desarrollador frontend se centra en la interfaz de usuario de un sitio web. Utilizan sus conocimientos de HTML y CSS para controlar el aspecto y la sensación de un sitio, a menudo adaptándose a los distintos tamaños de pantalla de los navegadores móviles y de escritorio. Los desarrolladores de frontend suelen basarse en JavaScript para crear una experiencia receptiva para los usuarios. Estos desarrolladores suelen estar familiarizados con las bibliotecas y marcos de JavaScript que pueden acelerar la creación de aplicaciones dinámicas orientadas al cliente. El desarrollador frontend trabaja para mejorar el rendimiento del sitio web, optimizando las imágenes, el JavaScript y el marcado para que el tiempo de carga en los navegadores sea más rápido. La mayoría de los desarrolladores frontend también se centran en la optimización de los motores de búsqueda y la accesibilidad del sitio web. Todo esto requiere muchas habilidades, y el salario de un desarrollador frontend puede reflejarlo. (BuiltWith, 2023)

¿Los desarrolladores frontend son diseñadores web? Muchos desarrolladores frontend son también diseñadores, pero eso no es un requisito del título «frontend». Del mismo modo, muchos diseñadores web conocen bien el HTML y el CSS -y pueden utilizarlos en las maquetas- pero no se consideran desarrolladores. Para muchas organizaciones, el diseño forma parte de la marca que abarca medios más allá de la web. Independientemente de quién cree un diseño, el trabajo del desarrollador de frontend es darle vida en una página web y convertir la visión de una experiencia de usuario en una aplicación funcional. (mdn, 2023)

Desarrollador web backend: Un desarrollador de backend crea aplicaciones del lado del servidor que suelen requerir experiencia en el software del servidor web, las bases de datos y los sistemas operativos en los que se ejecutan. Un ejemplo de código abierto sería el sistema operativo Linux, un servidor web Nginx o Apache y una base de datos MariaDB o PostgreSQL. La línea entre el desarrollo del backend y el frontend se difumina con tecnologías como PHP, un lenguaje de scripting del lado del servidor que envía HTML a los navegadores en el frontend. PHP -que impulsa WordPress y otros CMS populares, así como marcos de desarrollo como Laravel- es el lenguaje de programación del lado del servidor más común de la web. Sin embargo, los desarrolladores del backend pueden utilizar tecnologías como C# y el marco .NET de Microsoft, Python, Java, Ruby on Rails o Node.js. Los desarrolladores de backend pueden esperar trabajar en colaboración con los miembros del equipo que se ocupan

del lado del cliente de un sitio web. El desarrollo del lado del servidor también puede incluir la creación de interfaces de programación de aplicaciones (API) que apoyen los servicios del lado del cliente, con menos necesidad de un acoplamiento estrecho de ambos lados. Si todo esto suena como tu tipo de trabajo, puede que te interesen los salarios típicos de los desarrolladores de backend. (BuiltWith, 2023)

Desarrollador web full-stack: Puede que hayas adivinado que un desarrollador full-stack hace el trabajo de los desarrolladores de frontend y backend. (Por desgracia, eso no significa que el salario típico de un desarrollador full-stack sea el doble que el de los demás) Aun así, dominar todos los niveles del desarrollo web parece tener sus recompensas. Una encuesta realizada en 2022 por StackOverflow entre los desarrolladores descubrió que «desarrollador full-stack» era la respuesta principal (casi el 47%) cuando se pedía a los encuestados que describieran sus funciones. (StackOverflow, 2023)

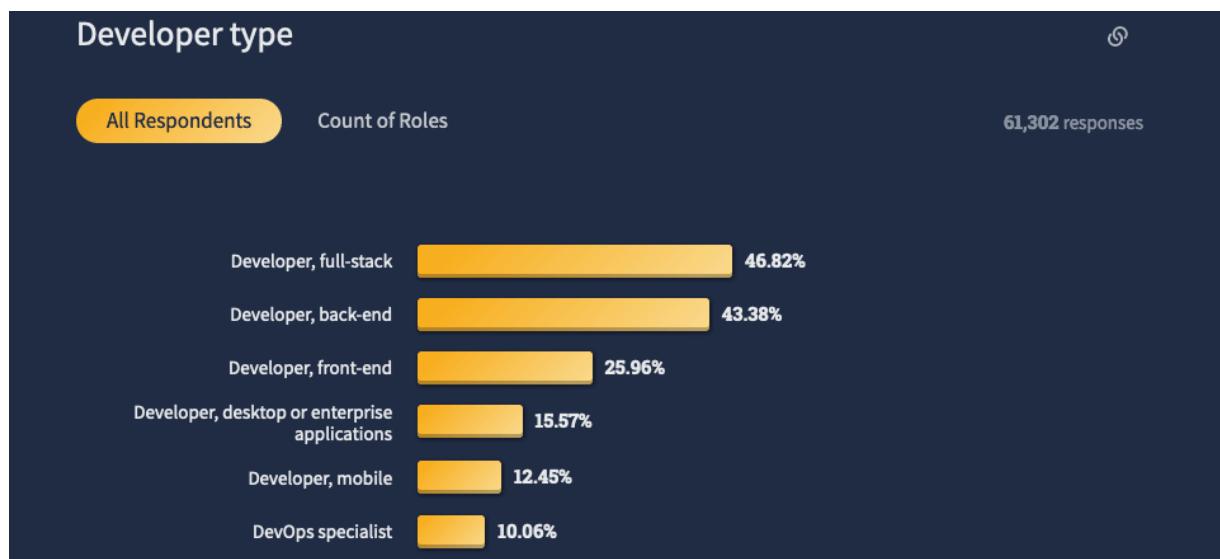


Figura 3. Principales tipos de desarrolladores en 2023.

Fuente: (StackOverflow, 2023)

Desarrollador de aplicaciones móviles: El término «aplicación» abarca mucho terreno en el desarrollo de software. Desde el escritorio hasta la web, las aplicaciones son las herramientas que hacen las cosas para los usuarios finales. Cuando se acorta a sólo «aplicación» – como en «¡Hay una aplicación para eso!» – tendemos a pensar primero en las aplicaciones que se ejecutan en smartphones, tabletas y otros dispositivos móviles. Y la creación de aplicaciones

móviles requiere un tipo de desarrollador especializado. Los desarrolladores de aplicaciones móviles están familiarizados con los kits de desarrollo de software (SDK) y las herramientas relacionadas que se utilizan para crear aplicaciones para el sistema operativo de un dispositivo, como iOS de Apple o Android de Google. Swift es el lenguaje de programación nativo para iOS, mientras que Java y Kotlin son los más utilizados para Android. Además, hay un gran ecosistema de herramientas de creación de aplicaciones, como Apache Cordova y NativeScript, que pueden convertir el código basado en JavaScript, HTML y CSS para la web en software móvil nativo. Cuando examinamos los salarios medios de los desarrolladores de aplicaciones, descubrimos que los que construyen para plataformas móviles ganan ligeramente más que los que se centran en el escritorio o la web. (Statcounter, 2023)

Desarrollador de software de escritorio: Los desarrolladores de escritorio crean aplicaciones de software que se ejecutan en ordenadores personales y estaciones de trabajo. Los desarrolladores suelen crear aplicaciones de escritorio para sistemas operativos específicos, por lo que la actividad en este campo refleja las cuotas de mercado relativas de plataformas como Windows de Microsoft, macOS de Apple, Chrome OS de Google y el sistema operativo de código abierto Linux. Utilizando datos recogidos de las visitas a sitios web de todo el mundo, Statcounter estimó en agosto de 2022 que Windows tenía una cuota de mercado del 74% entre los sistemas operativos de escritorio. (Statcounter, 2023)

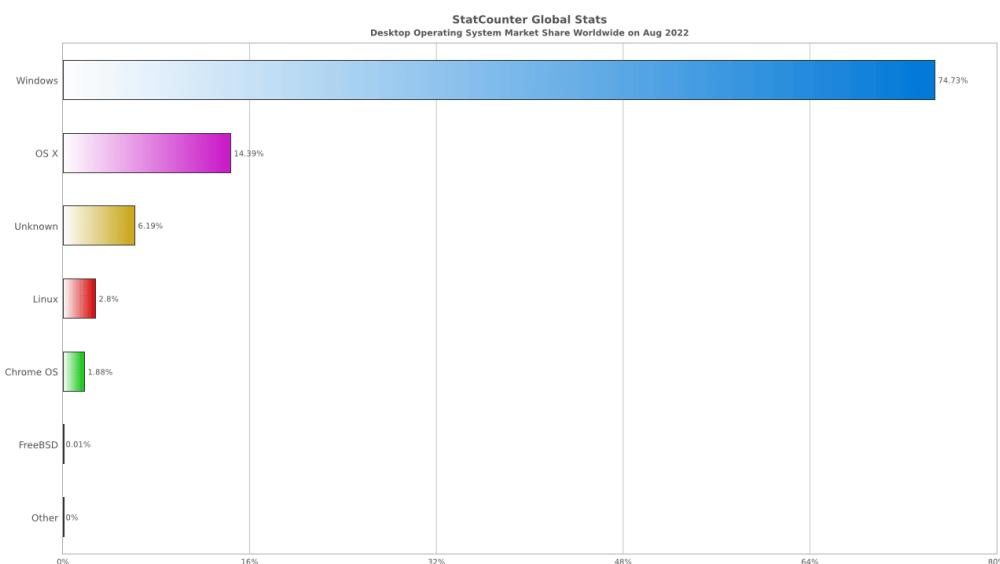


Figura 4. Cuota de mercado de los sistemas operativos, agosto de 2022.

Fuente: (Statcounter, 2023)

Es probable que los desarrolladores programen utilizando herramientas de entorno de desarrollo integrado (IDE) para editar, depurar y compilar rápidamente el código fuente. Los lenguajes más populares para la programación de aplicaciones de escritorio son C#, C++, Swift y Java.

El software de escritorio se ejecuta localmente, pero las aplicaciones modernas pueden utilizar la conectividad a Internet para tareas que van desde la actualización de productos hasta el intercambio de datos a través del almacenamiento en la nube.

También en la actualidad existe muchos tipos de software en los cuales la tecnología Transformers ha sido implementado brindando así un gran avance tecnológico y beneficios a las empresas que aplican los Transformers algunos de los softwares en los que se aplica esta tecnología en la actualidad son:

Motores de búsqueda: Los Transformers se utilizan en los motores de búsqueda para mejorar la comprensión de consultas de búsqueda y generar resultados más relevantes. Estos modelos ayudan a comprender el contexto y la intención detrás de las consultas de los usuarios, lo que mejora la precisión y la calidad de los resultados de búsqueda. (Nvidia, 2023)

Análisis de datos y minería de texto: Los transformers se aplican en el análisis de datos y la minería de texto para extraer información y conocimientos útiles de grandes conjuntos de datos no estructurados. Estos modelos ayudan en la clasificación de texto, agrupación, detección de anomalías y resumen automático de documentos. (Nvidia, 2023)

Reconocimiento de voz y procesamiento de audio: Los transformers también se utilizan en aplicaciones de reconocimiento de voz y procesamiento de audio. Estos modelos se utilizan para transcribir el habla en texto, realizar traducción de voz en tiempo real, mejorar la calidad del audio y realizar tareas de separación de fuentes de audio. (IBM, 2023)

Aplicaciones de recomendación y personalización: Los transformers se utilizan en sistemas de recomendación y personalización en aplicaciones como plataformas de streaming de música y video, servicios de comercio electrónico y redes sociales. Estos modelos ayudan a generar recomendaciones precisas y personalizadas para los usuarios en función de sus preferencias y comportamientos pasados. (Nvidia, 2023)

Seguridad y detección de fraudes: Los transformers se aplican en sistemas de seguridad y detección de fraudes para analizar grandes volúmenes de datos y detectar patrones anómalos o fraudulentos. Estos modelos pueden ayudar en la detección de spam, identificación de actividades sospechosas y protección contra ataques cibernéticos. (IBM, 2023)

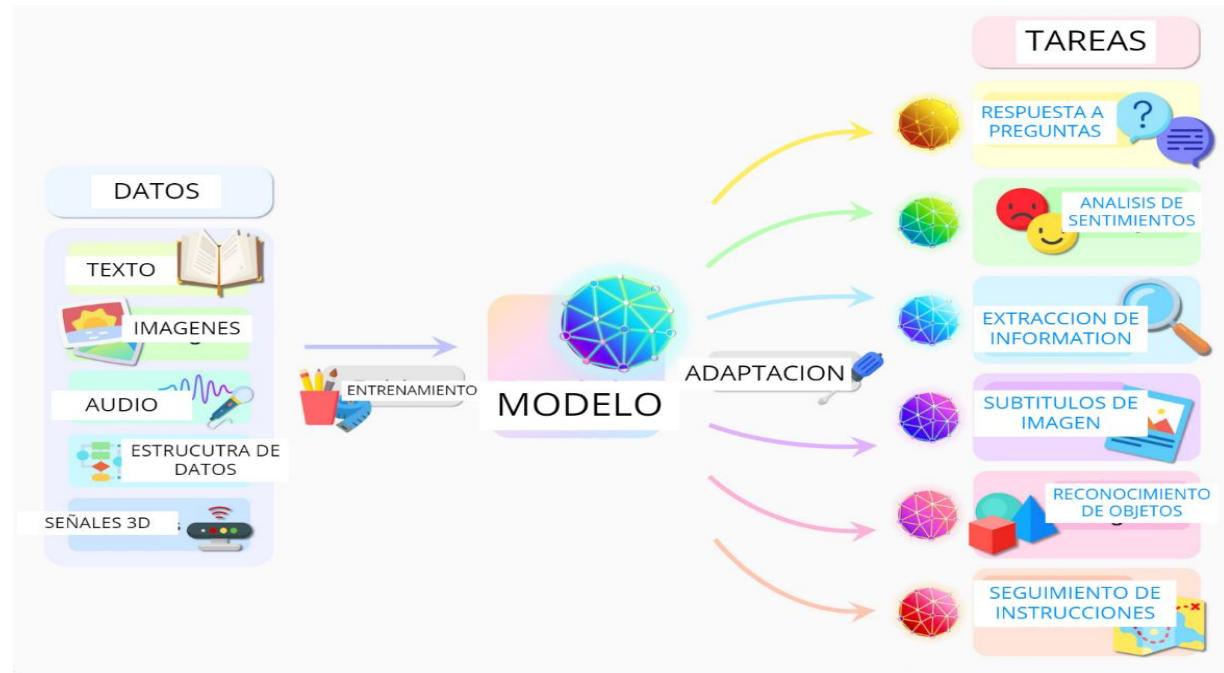


Figura 5. Diagrama de Casos de uso de un Modelo de Inteligencia Artificial

Fuente: (Nvidia, 2023)

CAPÍTULO III

FUNDAMENTO TEÓRICO

3.1. Antecedente Teórico

El desarrollo de software se refiere al proceso de crear, diseñar, programar, probar y mantener programas de computadora, aplicaciones móviles o sistemas informáticos. Es un proceso sistemático que involucra diversas etapas y actividades para construir software funcional y de calidad.

El desarrollo de software implica trabajar en conjunto con profesionales de diferentes disciplinas, como programadores, analistas de sistemas, diseñadores de interfaces, testers y gerentes de proyectos, entre otros. Estos profesionales colaboran para convertir los requisitos y necesidades de los usuarios en soluciones de software.

3.1.1 Impacto de la Inteligencia Artificial en el Desarrollo de Software

Durante décadas, los científicos han tratado de utilizar técnicas y algoritmos de inteligencia artificial para dotar a las computadoras con conocimiento y comportamiento similar al del ser humano. Aunque más sofisticadas que la programación tradicional, las técnicas utilizadas se han enfocado principalmente en crecer y mejorar manualmente la base de conocimiento del sistema, que siempre ha sido limitada. Un conocimiento limitado del dominio ha demostrado ser un mal sustituto para la experiencia de los humanos; es decir, los sistemas de IA son tan buenos como su programación (realizada manualmente por un humano). (Galvan, 2021)

Esta ola de inteligencia artificial impactará el trabajo de los desarrolladores de software, así que es importante estar preparados. Los desarrolladores deben entender en qué consisten dichas tecnologías y cómo pueden aplicarlas, tanto en el ciclo de vida de desarrollo de software como en las aplicaciones mismas. (Galvan, 2021)

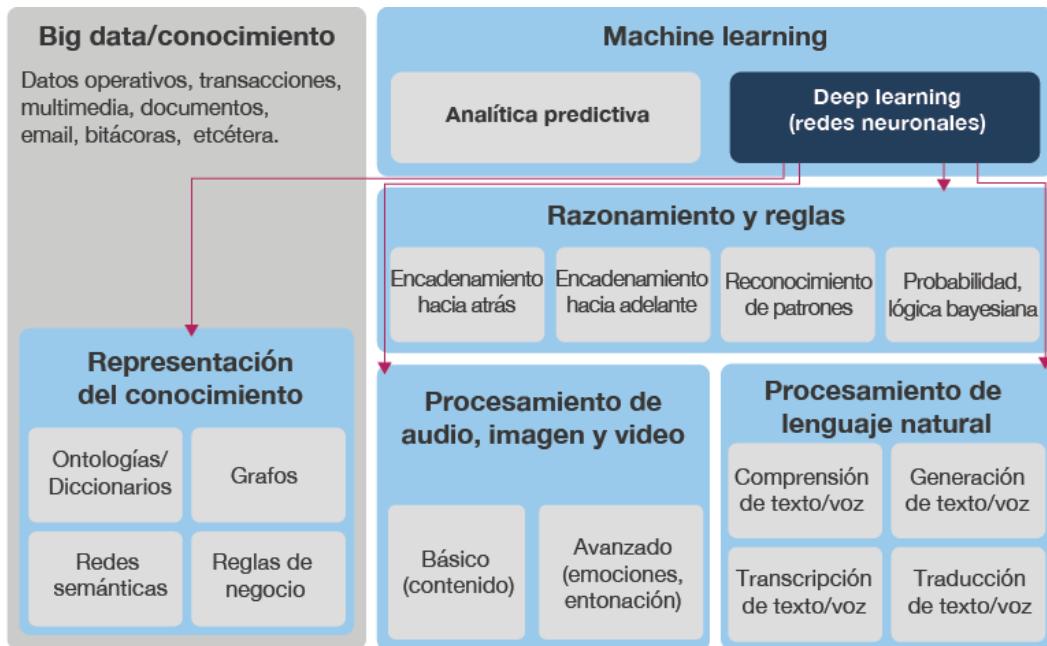


Figura 6. El Deep learning fortalece a otras técnicas de IA

Fuente: (Galvan, 2021)

3.1.2 Modelos Generadores de textos

En la actualidad hay muchas herramientas de inteligencia artificial de generación de texto y código algunas de estas son OpenAI Codex y GPT-3, DeepCode, Bard, ChatGPT que en la actualidad se encuentra en su cuarta versión y ha revolucionado la manera en que entendemos la inteligencia artificial y sus aplicaciones en nuestro diario vivir, incluso ha causado preocupación en la comunidad científica por el avance que la inteligencia ha tenido aproximadamente en tan solo cinco años. (marketing-branding, 2023)

También existen herramientas de autocompletado de código Kite, Tabnine siendo Github Copilot la herramienta más utilizada para el autocompletado de código se basa en la tecnología de inteligencia artificial y aprendizaje automático, específicamente en el modelo de IA GPT-3 de OpenAI, para ayudar a los desarrolladores a escribir código de manera más eficiente y productiva. (marketing-branding, 2023)

Chat GPT una de las herramientas más utilizadas en la actualidad que forma parte del mercado de aplicaciones que implementan Transformers no obtiene su información navegando por internet. Pero lo interesante de OpenAI es que, en un movimiento revolucionario, creó una interfaz de usuario que permite al público en general experimentar con ella directamente. (marketing-branding, 2023)

A grandes rasgos, los bots de chat como GPT funcionan con grandes cantidades de datos y técnicas informáticas para hacer predicciones para unir palabras de manera significativa. No sólo aprovechan una gran cantidad de vocabulario e información, sino que también entienden palabras en contexto. Esto les ayuda a imitar los patrones del habla mientras transmiten un conocimiento enciclopédico. (marketing-branding, 2023)

Para simplificar esto se muestra la siguiente figura:

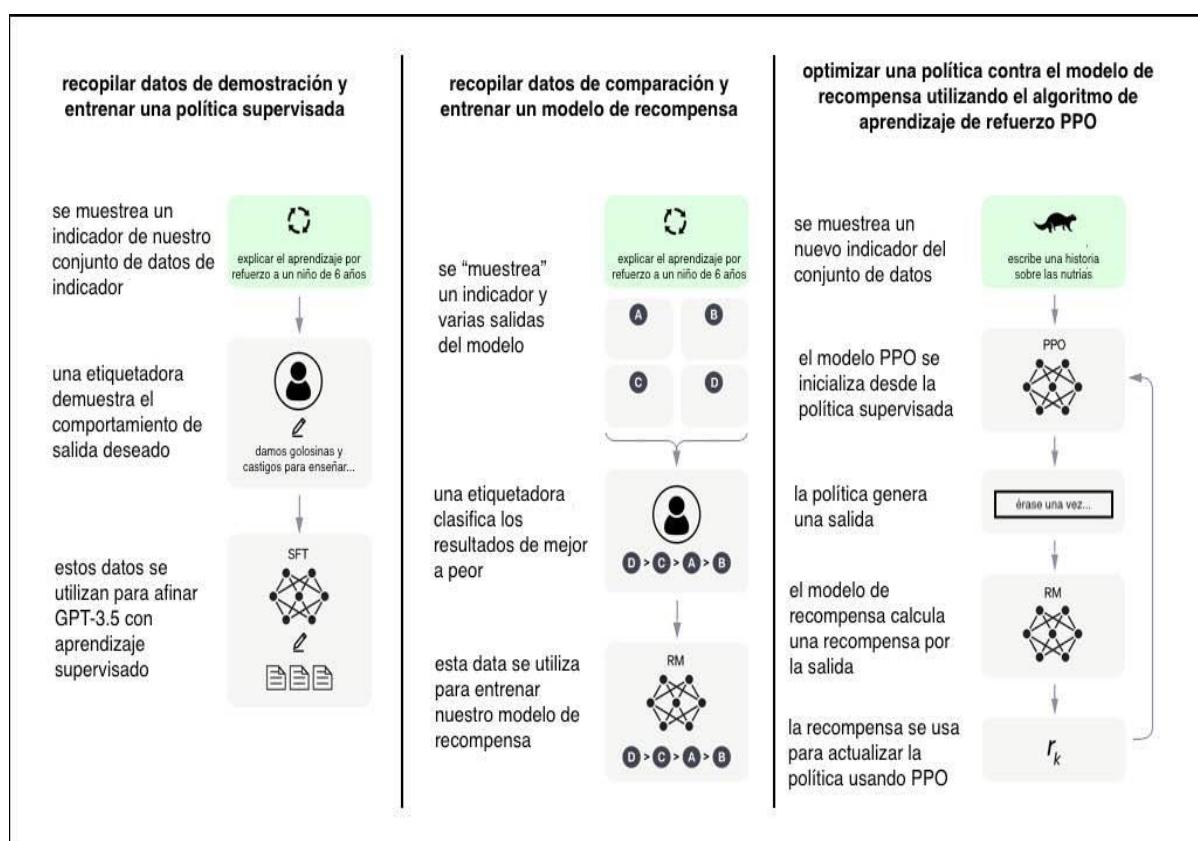


Figura 7. Diagrama de Proceso de Desarrollo de Chat GPT
Fuente: (marketing-branding, 2023)

3.1.2.1 ¿Qué es un modelo de Inteligencia Artificial?

Un modelo de inteligencia artificial (IA) es una representación matemática o computacional que se utiliza para simular el comportamiento inteligente de un sistema. Estos modelos pueden ser diseñados para llevar a cabo una variedad de tareas, desde reconocimiento de imágenes y procesamiento de lenguaje natural hasta toma de decisiones y control de sistemas autónomos. Los modelos de IA se construyen a partir de algoritmos y técnicas de aprendizaje automático, como redes neuronales artificiales, árboles de decisión, máquinas de vectores de soporte, entre otros. Estos modelos se entran con datos para aprender patrones y realizar predicciones o tomar decisiones en función de nuevos datos de entrada. (IBM, 2023)

Características de un modelo de Inteligencia Artificial:

1. Capacidad de aprendizaje: Los modelos de IA pueden aprender de los datos a través de técnicas de aprendizaje automático, como el aprendizaje supervisado, no supervisado o por refuerzo.
2. Adaptabilidad: Pueden adaptarse a nuevos datos y circunstancias, actualizando sus comportamientos y predicciones a medida que reciben más información.
3. Generalización: Los modelos pueden generalizar los patrones aprendidos a partir de un conjunto de datos para hacer predicciones precisas sobre datos nuevos y no vistos.
4. Automatización: Pueden realizar tareas de manera autónoma una vez que han sido entrenados y desplegados, lo que puede aumentar la eficiencia y reducir los costos en diversas aplicaciones.
5. Escalabilidad: Algunos modelos de IA pueden ser escalables, lo que significa que pueden manejar grandes volúmenes de datos y aumentar su capacidad de procesamiento a medida que se necesite.
6. Interpretabilidad: En algunos casos, los modelos de IA pueden proporcionar explicaciones sobre cómo llegaron a una decisión o predicción particular, lo que puede ayudar a comprender su funcionamiento y confiabilidad.
7. Robustez: Los modelos pueden ser diseñados para ser robustos frente a ruido en los datos, cambios en el entorno o ataques adversarios. (datascientest, 2023)

3.1.3 Movimiento no Code

No Code es una filosofía que quiere cambiar cómo se crean páginas web, aplicaciones o juegos. Sin código. Que cualquiera pueda hacer apps sin programar. (López, 2022).

Para crear una aplicación o página web de manera profesional hay que conocer determinados lenguajes de programación, emplearlos en la práctica escribiendo código, depurando ese código, compilándolo, probándolo en un entorno real o virtual y un largo etcétera. Pero el movimiento No Code quiere simplificar este proceso. ¿Por qué no convertir la tarea de programar una app en algo tan fácil como crear un documento de Word o una presentación de PowerPoint? (López, 2022)

Primero fue el movimiento Low Code. Este término fue acuñado por la empresa de investigación y asesoría empresarial Forrester Research. La idea detrás de este concepto es democratizar el acceso a la creación de aplicaciones y otros proyectos digitales que requieren código de por medio. No se trata de eliminar a los profesionales del código. Sino permitir que cualquier profesional sea capaz de programar sin código. O al menos, con el mínimo código posible. (López, 2022)

Con el tiempo, esta filosofía ha evolucionado y hoy se apuesta drásticamente por el concepto de No Code. El propósito es el mismo. Que no saber programar no sea un impedimento. De ahí la proliferación en los últimos años de herramientas, muchas de ellas online, que hacen de programar una tarea sencilla. Incluso soluciones profesionales como el Android Studio de Google siguen esta filosofía para que programadores y no programadores se sientan a gusto creando aplicaciones Android. (López, 2022)

Con código o con menús gráficos, como el editor de diseño, que permite añadir elementos gráficos en pantalla mientras el archivo XML correspondiente se va completando con el código. Para lograr su propósito, las plataformas y servicios No Code ofrecen plantillas y editores gráficos que convierten la experiencia de programar en una actividad más propia de un puzzle. Normalmente, mediante elementos modulares que gestionamos con el ratón y el teclado, la herramienta los traduce en código que nosotros no necesitamos editar manualmente. Eso sí, podemos ver al mismo tiempo ambas versiones, la gráfica y la textual. (López, 2022)

Otra particularidad del movimiento No Code es que unifica en sus herramientas todo el proceso desde que se diseña una aplicación o página web hasta que se publica online o en la tienda de aplicaciones correspondiente. Así, no sólo creamos la aplicación, también podemos probarla, integrar elementos como pasarelas de pago o servicios de terceros y, finalmente, la enviamos para que los usuarios de iOS y Android puedan descargarla en sus dispositivos. (López, 2022)

3.2. Marco Teórico del contexto

3.2.1 Lenguaje Css

En la actualidad el lenguaje CSS (Cascading Style Sheets) es fundamental en el desarrollo web y se utiliza ampliamente en todo el mundo, algunas de las tendencias y características más destacadas de CSS en la actualidad son el Desarrollo web moderno donde CSS cumple un rol fundamental, Responsive Web Design (Diseño web adaptable) CSS juega un papel vital en el diseño web adaptable, Animaciones 2D o 3D para la web, CSS Grid y Flexbox son dos sistemas de diseño en CSS que han revolucionado la forma en que se crea y se estructura el diseño de las páginas web, Herramientas y Frameworks de CSS que mejoran la eficiencia en el desarrollo, el diseño responsive, la reutilización de componentes y la compatibilidad con navegadores. (Peiró, 2020)

Los lenguajes de hojas de estilo surgieron con la introducción de Internet y el crecimiento exponencial del lenguaje HTML para la creación de documentos electrónicos. El organismo W3C (World Wide Web Consortium) es el encargado de crear todos los estándares relacionados con la web y fue el que propuso la creación de un lenguaje de hojas de estilos específico para el lenguaje HTML. Se escogieron dos propuestas: la CHSS (Cascading HTML Style Sheets) y la SSP (Stream-based Style Sheet Proposal). Entre finales de 1994 y 1995 se definió un nuevo lenguaje que tomaba lo mejor de cada propuesta y lo llamaron CSS (Cascading Style Sheets). (Peiró, 2020)

A principios de 1997, el W3C decide separar los trabajos del grupo de HTML en tres secciones: el grupo de trabajo de HTML, el grupo de trabajo de DOM y el grupo de trabajo de

CSS. La adaptación del lenguaje CSS por parte de los navegadores ha sido progresiva y ha requerido un largo tiempo. En la actualidad todos lo reconocen. (mdn, 2023)

El lenguaje CSS se ha convertido en una revolución en el ámbito del diseño web especialmente. Entre los beneficios que se encuentran al utilizar éste se destacan los siguientes:

Más precisión: Cuando se utiliza CSS, el tamaño y posicionamiento de los elementos que conforman la web será exacto. Se le puede indicar al navegador en qué píxel colocar una determinada imagen, así como las medidas de éstas. El CSS da mejor accesibilidad y estructura. Al combinar el lenguaje CSS y los marcadores descriptivos se posibilita que una web se vea correctamente puesto que la información se mantendrá estructurada y ordenada. (Peiró, 2020)

Mejora los tiempos de carga: Con la introducción del CSS se ha dividido contenido y apariencia, por lo que se obtienen archivos más ligeros. Esto es ventajoso para reducir los tiempos de carga del sitio en el navegador y bajar el volumen de tráfico del servidor que se ha escogido. (Peiró, 2020)

Preprocesador CSS: Un preprocesador CSS es un programa que te permite generar CSS a partir de la sintaxis (en-US) única del preprocesador. Existen varios preprocesadores CSS de los cuales escoger, sin embargo, la mayoría de preprocesadores CSS añadirán algunas características que no existen en CSS puro, como variables, mezclas, selectores anidados, entre otros. Estas características hacen la estructura de CSS más legible y fácil de mantener. (mdn, 2023)

Bootstrap es la librería de CSS más utilizada debido a su amplia comunidad de usuarios, su diseño y funcionalidad responsivos, su conjunto completo de componentes y estilos predefinidos, y su capacidad de personalización. (Bootstrap, 2023)

Estas características hacen que sea una herramienta muy útil para los desarrolladores que buscan crear diseños y aplicaciones web modernas y adaptativas. Bootstrap está constituido por una serie de archivos CSS y JavaScript responsables de asignar características específicas a los elementos de la página. (Author, 2020)

Sass y Less: Sass (Syntactically Awesome Style Sheets) y Less (Leaner CSS) son preprocesadores de CSS que permiten a los desarrolladores escribir CSS de manera más eficiente y organizada. (mdn, 2023)

CSS Grid y Flexbox: CSS Grid y Flexbox son herramientas de diseño web que permiten a los desarrolladores crear diseños de página más complejos y flexibles. (mdn, 2023)

Animaciones CSS: Las animaciones CSS permiten a los desarrolladores agregar efectos y animaciones a las páginas web. (mdn, 2023)

3.2.2 La Inteligencia Artificial y sus áreas de aplicación

En la actualidad, los desarrolladores de software desempeñan un papel fundamental en el avance y la aplicación de la inteligencia artificial (IA). A medida que la IA se ha vuelto cada vez más prominente en diversas industrias, los desarrolladores de software se han convertido en los encargados de crear y desplegar soluciones basadas en esta tecnología. (Peiró, 2020)

La IA es el estudio de los agentes que reciben percepciones del entorno y realizan acciones. (datascientest, 2023)

A continuación, se presentan algunas de las principales áreas en las que se aplica la IA en la actualidad:

Aprendizaje automático (Machine Learning): El aprendizaje automático es una rama de la IA que se basa en algoritmos y modelos para permitir a las máquinas aprender y mejorar a partir de datos. Los algoritmos de aprendizaje automático pueden identificar patrones y realizar predicciones o tomar decisiones basadas en conjuntos de datos históricos. Se utilizan técnicas como el aprendizaje supervisado, no supervisado y por refuerzo para entrenar modelos y permitirles aprender de manera autónoma. (datascientest, 2023)

Redes Neuronales Artificiales: Las redes neuronales artificiales son modelos inspirados en el funcionamiento del cerebro humano. Estas redes están compuestas por nodos interconectados llamados neuronas artificiales, que procesan y transmiten información. Las redes neuronales

son ampliamente utilizadas en tareas como el reconocimiento de imágenes y voz, la traducción automática y el procesamiento del lenguaje natural. (datascientest, 2023)

Procesamiento del Lenguaje Natural (NLP): El NLP es un campo de la IA que se centra en la interacción entre las computadoras y el lenguaje humano. Los sistemas de NLP permiten a las máquinas comprender, interpretar y generar texto de manera similar a como lo haría un ser humano. Esto se aplica en tareas como la traducción automática, la generación de texto, la clasificación de sentimientos y el procesamiento de voz a texto. (datascientest, 2023)

Visión por Computadora: La visión por computadora se ocupa del análisis, procesamiento e interpretación de imágenes y videos por parte de las máquinas. Utiliza algoritmos de IA para reconocer objetos, realizar seguimiento de movimientos, detectar rostros, analizar imágenes médicas, entre otras aplicaciones. Esto se aplica en campos como la conducción autónoma, la vigilancia de seguridad, la realidad aumentada, entre otros. (datascientest, 2023)

Robótica y Automatización: La IA se utiliza ampliamente en robots y sistemas automatizados para permitirles realizar tareas complejas y adaptarse a diferentes situaciones. Los robots equipados con IA pueden realizar tareas de ensamblaje, navegación, reconocimiento de objetos y colaboración con humanos. La automatización de procesos empresariales también se beneficia de la IA al utilizar algoritmos de aprendizaje automático para automatizar tareas repetitivas y mejorar la eficiencia. (datascientest, 2023)

Sistemas de Recomendación: Los sistemas de recomendación utilizan algoritmos de IA para analizar y comprender los patrones de comportamiento de los usuarios y proporcionar recomendaciones personalizadas. Estos sistemas se aplican en plataformas de comercio electrónico, servicios de streaming, redes sociales y otras aplicaciones en línea para ofrecer sugerencias de productos, música, películas, contenido y conexiones sociales relevantes para cada usuario. (datascientest, 2023)

Asistentes Virtuales y Chatbots: Los asistentes virtuales y chatbots utilizan IA para interactuar con los usuarios y brindar respuestas y servicios. Estos sistemas pueden entender y responder preguntas, realizar tareas específicas, brindar información y asistencia en tiempo real. Ejemplos conocidos son Siri de Apple, Google Assistant, Alexa de Amazon y chatbots de atención al cliente. (datascientest, 2023)

3.2.3 Redes Neuronales

Las redes neuronales son un modelo computacional inspirado en el sistema nervioso biológico. Estas redes están compuestas por unidades de procesamiento interconectadas, llamadas neuronas artificiales o nodos, que trabajan en conjunto para resolver problemas y realizar tareas de aprendizaje automático.

Una red neuronal es un método de la inteligencia artificial que enseña a las computadoras a procesar datos de una manera que está inspirada en la forma en que lo hace el cerebro humano. Se trata de un tipo de proceso de machine learning llamado aprendizaje profundo, que utiliza los nodos o las neuronas interconectados en una estructura de capas que se parece al cerebro humano. Crea un sistema adaptable que las computadoras utilizan para aprender de sus errores y mejorar continuamente. De esta forma, las redes neuronales artificiales intentan resolver problemas complicados, como la realización de resúmenes de documentos o el reconocimiento de rostros, con mayor precisión. (aws, 2022)

Las redes neuronales están presentes en varios casos de uso en muchos sectores, como los siguientes:

- Diagnóstico médico mediante la clasificación de imágenes médicas
- Marketing orientado mediante el filtrado de redes sociales y el análisis de datos de comportamiento
- Predicciones financieras mediante el procesamiento de datos históricos de instrumentos financieros
- Previsión de la carga eléctrica y la demanda de energía
- Proceso y control de calidad
- Identificación de compuestos químicos

A continuación, presentamos cuatro de las aplicaciones más importantes de las redes neuronales.

Visión artificial: La visión artificial es la capacidad que tienen las computadoras para extraer información y conocimientos de imágenes y videos. Con las redes neuronales, las

computadoras pueden distinguir y reconocer imágenes de forma similar a los humanos. La visión artificial tiene varias aplicaciones, como las siguientes: (aws, 2022)

- Reconocimiento visual en los vehículos autónomos para que puedan reconocer las señales de tráfico y a otros usuarios del camino
- Moderación de contenido para eliminar de forma automática los contenidos inseguros o inapropiados de los archivos de imágenes y videos
- Reconocimiento facial para identificar rostros y reconocer atributos como ojos abiertos, gafas y vello facial
- Etiquetado de imágenes para identificar logotipos de marcas, ropa, equipos de seguridad y otros detalles de la imagen

Reconocimiento de voz: Las redes neuronales pueden analizar el habla humana a pesar de los diferentes patrones de habla, el tono, el idioma y el acento. Los asistentes virtuales como Amazon Alexa y el software de transcripción automática utilizan el reconocimiento de voz para realizar tareas como las siguientes: (aws, 2022)

- Asistir a los agentes de los centros de llamadas y clasificar las llamadas de forma automática
- Convertir las conversaciones clínicas en documentación en tiempo real
- Subtitular con precisión videos y grabaciones de reuniones para aumentar el alcance del contenido

Procesamiento de lenguaje natural: El procesamiento de lenguaje natural (PLN) es la capacidad de procesar texto natural creado por humanos. Las redes neuronales ayudan a las computadoras a obtener información y significado a partir de los datos y los documentos de texto. El PLN está presente en varios casos de uso, entre los que se incluyen los siguientes: (aws, 2022)

- Chatbots y agentes virtuales automatizados
- Organización y clasificación automáticas de datos escritos
- Análisis de inteligencia empresarial de documentos con formato largo, como emails y formularios

- Indexación de frases clave que indican sentimientos, como los comentarios positivos y negativos en las redes sociales
- Resumen de documentos y producción de artículos para un tema determinado

Motores de recomendaciones: Las redes neuronales pueden hacer un seguimiento de la actividad del usuario para elaborar recomendaciones personalizadas. También pueden analizar todo el comportamiento de los usuarios y descubrir productos o servicios nuevos que interesen a un usuario específico. Por ejemplo, Curalate, una empresa emergente con sede en Filadelfia, ayuda a las marcas a convertir las publicaciones en las redes sociales en ventas. Las marcas utilizan el servicio de etiquetado inteligente de productos (IPT) de Curalate para automatizar la recopilación y la selección del contenido social que generan los usuarios. El IPT utiliza las redes neuronales para encontrar y recomendar de forma automática productos relevantes para la actividad del usuario en las redes sociales. Los consumidores no tienen que buscar en los catálogos en línea para encontrar un producto específico a partir de una imagen en las redes sociales. En cambio, pueden utilizar el etiquetado automático de productos de Curalate para comprar el producto con facilidad. (aws, 2022)

El cerebro humano es lo que inspira la arquitectura de las redes neuronales. Las células del cerebro humano, llamadas neuronas, forman una red compleja y con un alto nivel de interconexión y se envían señales eléctricas entre sí para ayudar a los humanos a procesar la información. De manera similar, una red neuronal artificial está formada por neuronas artificiales que trabajan juntas para resolver un problema. Las neuronas artificiales son módulos de software, llamados nodos, y las redes neuronales artificiales son programas de software o algoritmos que, en esencia, utilizan sistemas informáticos para resolver cálculos matemáticos. (aws, 2022)

Arquitectura de una red neuronal simple: Una red neuronal básica tiene neuronas artificiales interconectadas en tres capas:

Capa de entrada: La información del mundo exterior entra en la red neuronal artificial desde la capa de entrada. Los nodos de entrada procesan los datos, los analizan o los clasifican y los pasan a la siguiente capa. (aws, 2022) (Matich, 2001)

Capa oculta: Las capas ocultas toman su entrada de la capa de entrada o de otras capas ocultas. Las redes neuronales artificiales pueden tener una gran cantidad de capas ocultas. Cada capa oculta analiza la salida de la capa anterior, la procesa aún más y la pasa a la siguiente capa. (aws, 2022)

Capa de salida: La capa de salida proporciona el resultado final de todo el procesamiento de datos que realiza la red neuronal artificial. Puede tener uno o varios nodos. Por ejemplo, si tenemos un problema de clasificación binaria (sí/no), la capa de salida tendrá un nodo de salida que dará como resultado 1 o 0. Sin embargo, si tenemos un problema de clasificación multiclase, la capa de salida puede estar formada por más de un nodo de salida. (aws, 2022)

En una red neuronal artificial (ANN), los pesos y sesgos son parámetros ajustables que se utilizan para realizar el cálculo de las activaciones en cada capa de la red y, finalmente, producir una salida.

A continuación, brindaremos una definición matemática formal y conceptual de estos términos:

En una red neuronal artificial (ANN), los pesos y sesgos son parámetros ajustables que se utilizan para realizar el cálculo de las activaciones en cada capa de la red y, finalmente, producir una salida. Aquí tienes una explicación matemática y conceptual de estos términos:

Pesos (Weights):

Los pesos son valores numéricos que representan la fuerza y la dirección de la conexión entre las neuronas en dos capas consecutivas de una red neuronal. Cada conexión entre una neurona de la capa i y una neurona de la capa $i + 1$ está asociada con un peso. (Matich, 2001)

Matemáticamente: Si representamos las activaciones de las neuronas en la capa i como un vector $a^{(i)}$ y los pesos asociados con las conexiones hacia la capa $i + 1$ como una matriz $W^{(i)}$, entonces el cálculo de las activaciones en la capa $i + 1$ se realiza mediante una operación de multiplicación de matrices:

$$z^{(i+1)} = W^{(i)} \cdot a^{(i)}$$

Los pesos son los parámetros que se ajustan durante el proceso de entrenamiento de la red neuronal para minimizar alguna función de pérdida, generalmente mediante algoritmos de optimización como el descenso de gradiente. El ajuste de los pesos permite que la red neuronal aprenda a representar las relaciones entre las características de entrada y las salidas deseadas. (Matich, 2001)

Sesgos (Biases):

Los sesgos son términos adicionales que se agregan a las activaciones en cada capa de una red neuronal, independientemente de las entradas. Los sesgos permiten que la red tenga cierto grado de libertad para realizar desplazamientos o traslaciones en el espacio de las características. (Matich, 2001)

Matemáticamente: Si representamos los sesgos en la capa $i + 1$ como un vector $b^{(i+1)}$, entonces el cálculo de las activaciones en la capa $i + 1$ se modifica agregando el sesgo:

$$z^{(i+1)} = W^{(i)} \cdot a^{(i)} + b^{(i+1)}$$

Al igual que los pesos, los sesgos son parámetros ajustables que se modifican durante el entrenamiento para mejorar el rendimiento de la red neuronal. (Matich, 2001)

Arquitectura de una red neuronal profunda: Las redes neuronales profundas, o redes de aprendizaje profundo, tienen varias capas ocultas con millones de neuronas artificiales conectadas entre sí. Un número, denominado peso, representa las conexiones entre un nodo y otro. El peso es un número positivo si un nodo estimula a otro, o negativo si un nodo suprime a otro. Los nodos con valores de peso más altos tienen mayor influencia en los demás nodos. En teoría, las redes neuronales profundas pueden asignar cualquier tipo de entrada a cualquier tipo de salida. Sin embargo, también necesitan mucho más entrenamiento en comparación con otros métodos de machine learning. Necesitan millones de ejemplos de datos de entrenamiento en lugar de los cientos o miles que podría necesitar una red más simple. (aws, 2022)

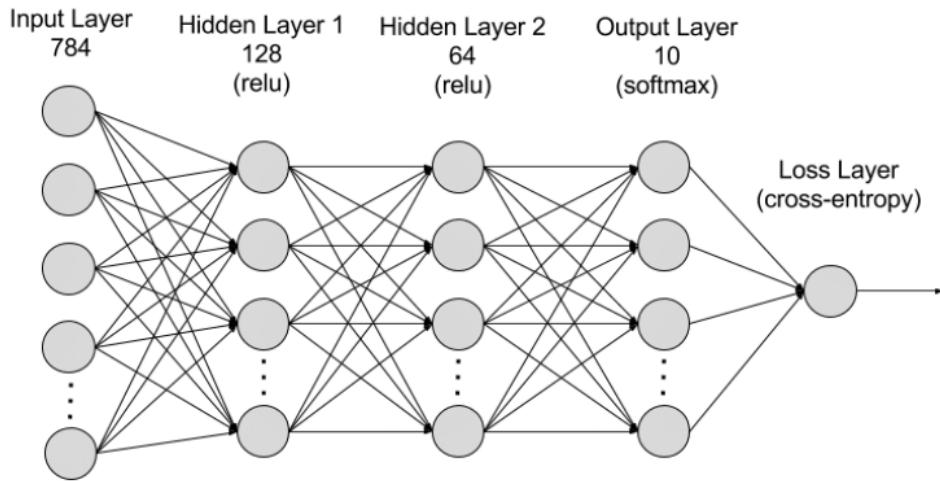


Figura 8. Arquitectura de una Red Neuronal Simple

Fuente: (aws, 2022)

3.2.3.1 ¿Cuáles son los tipos de redes neuronales?

Las redes neuronales artificiales pueden clasificarse en función de cómo fluyen los datos desde el nodo de entrada hasta el nodo de salida. A continuación, se indican varios ejemplos:

Redes neuronales prealimentadas: Las redes neuronales prealimentadas procesan los datos en una dirección, desde el nodo de entrada hasta el nodo de salida. Todos los nodos de una capa están conectados a todos los nodos de la capa siguiente. Una red prealimentada utiliza un proceso de retroalimentación para mejorar las predicciones a lo largo del tiempo. (aws, 2022)

Algoritmo de retropropagación: Las redes neuronales artificiales aprenden de forma continua mediante el uso de bucles de retroalimentación correctivos para mejorar su análisis predictivo. En pocas palabras, puede pensar en los datos que fluyen desde el nodo de entrada hasta el nodo de salida a través de muchos caminos diferentes en la red neuronal. Solo un camino es el correcto: el que asigna el nodo de entrada al nodo de salida correcto. Para encontrar este camino, la red neuronal utiliza un bucle de retroalimentación que funciona de la siguiente manera: (aws, 2022)

- Cada nodo intenta adivinar el siguiente nodo de la ruta.

- Se comprueba si la suposición es correcta. Los nodos asignan valores de peso más altos a las rutas que conducen a más suposiciones correctas y valores de peso más bajos a las rutas de los nodos que conducen a suposiciones incorrectas.
- Para el siguiente punto de datos, los nodos realizan una predicción nueva con las trayectorias de mayor peso y luego repiten el paso 1.

Redes neuronales convolucionales: Las capas ocultas de las redes neuronales convolucionales realizan funciones matemáticas específicas, como la síntesis o el filtrado, denominadas convoluciones. Son muy útiles para la clasificación de imágenes porque pueden extraer características relevantes de las imágenes que son útiles para el reconocimiento y la clasificación de imágenes. La forma nueva es más fácil de procesar sin perder características que son fundamentales para hacer una buena predicción. Cada capa oculta extrae y procesa diferentes características de la imagen, como los bordes, el color y la profundidad. (aws, 2022)

3.2.4 Transformers

¿Qué Es un Modelo Transformer?

Un modelo transformer es una red neuronal que aprende contexto y, por lo tanto, significado mediante el seguimiento de relaciones en datos secuenciales como las palabras de una oración. Los modelos transformer aplican un conjunto en evolución de técnicas matemáticas, llamadas atención o atención propia, para detectar formas sutiles en que los elementos de datos en una serie se influencian y dependen entre sí. (Nvidia, 2023)

Un Modelo Transformer es una arquitectura de red neuronal utilizada principalmente en el campo del Procesamiento del Lenguaje Natural (NLP). Fue introducido por primera vez en el artículo "Attention is All You Need" de Vaswani et al. en 2017 y desde entonces ha sido ampliamente adoptado en diversas aplicaciones de NLP. (Nvidia, 2023)

"Los modelos con mejor rendimiento también conectan el codificador y el decodificador mediante un conector de atención. mecanismo. Proponemos una nueva arquitectura de red simple, el Transformer, basada únicamente en mecanismos de atención, prescindiendo de la recurrencia y las convoluciones. Enteramente". (Ashish Vaswani, 2017)

Los Transformers se ha vuelto muy popular y ampliamente utilizada en la actualidad, especialmente en tareas relacionadas con el procesamiento del lenguaje natural (NLP, por sus siglas en inglés). En la actualidad los Transformers se utiliza en Modelos de lenguaje pre entrenados como BERT (Bidirectional Encoder Representations from Transformers) y GPT (Generative Pre-trained Transformer), se entrena con grandes cantidades de texto para aprender representaciones del lenguaje. (aws, 2022)

Chatbots y asistentes virtuales. Los Transformers se utilizan para desarrollar chatbots y asistentes virtuales que pueden mantener conversaciones coherentes y responder preguntas de manera inteligente. (aws, 2022)

Otros usos muy frecuentes de los Transformers son Traducción Automática, Resumen Automático, Generación de Texto y Código.

Existe una gran cantidad de profesionales usando esta tecnología en la actualidad algunos de estos son: Investigadores en IA utilizan transformers para desarrollar y mejorar modelos de procesamiento de lenguaje natural, Ingenieros de software, Empresas de tecnología, Desarrolladores de aplicaciones IA. (Nvidia, 2023)

3.2.5 Red Transformer

La Red Transformer descrita en el artículo de 2017 Attention is all you need, desarrollado por investigadores de Google, y nacieron inicialmente como una alternativa al problema de la traducción de texto de un idioma a otro. En estas redes la totalidad de la secuencia de entrada es procesada en paralelo por la red, a diferencia de las Redes Recurrentes en donde se procesan uno a uno (es decir de forma serial) los elementos de la secuencia.

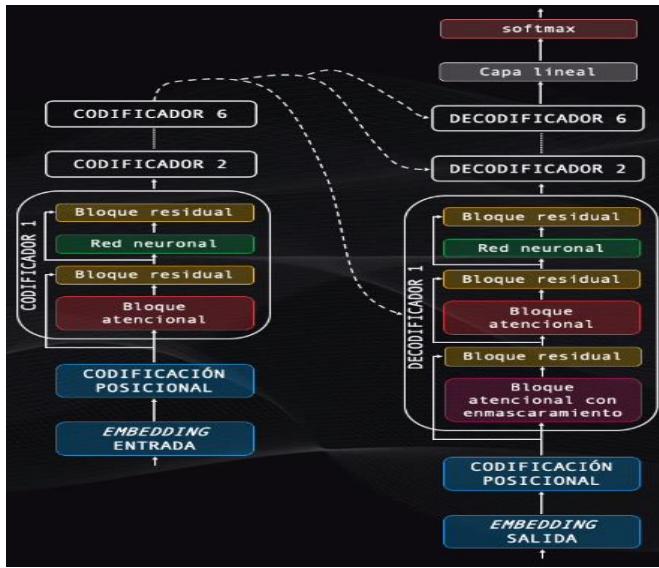


Figura 9. Diagrama de bloques general de una Red Transformer

Fuente: (Sotaquirá, 2020)

Esta secuencia es inicialmente convertida en una representación numérica usando un embedding. Después se añade una codificación de posición y los vectores resultantes ingresan a la etapa de codificación, que se encarga de extraer la información más relevante de la secuencia en su idioma original. La salida de esta etapa se conecta al decodificador, que toma esta información para generar secuencialmente el texto traducido al segundo idioma. Aunque tiene muchos elementos realmente es sencillo entender cómo funciona. Veamos entonces en detalle cada módulo de esta red. (Sotaquirá, 2020)

El embedding de entrada

Primero está el bloque embedding, que es simplemente un algoritmo que convierte el texto en una serie de vectores, o tokens, es decir en una representación numérica que puede ser “comprendida” por la red.



Figura 10. Embedding de entrada de la Red Transformer
Fuente: (Sotaquirá, 2020)

El codificador de posición



Figura 11. Codificador de posición de la Red Transformer
Fuente: (Sotaquirá, 2020)

Como la secuencia se procesa en paralelo es necesario indicarle a la red el orden en el que se encuentran las palabras dentro del texto. Esto se logra con el codificador de posición. Este codificador genera una serie de vectores que se sumarán a los tokens, y que indican la posición relativa de cada token dentro de la secuencia. Para esto se usan funciones senoidales para las posiciones pares, y cosenoidales para las impares, con lo que cada vector generado tendrá un patrón numérico único con la información de la posición. (Sotaquirá, 2020)

Codificación y el bloque atencional

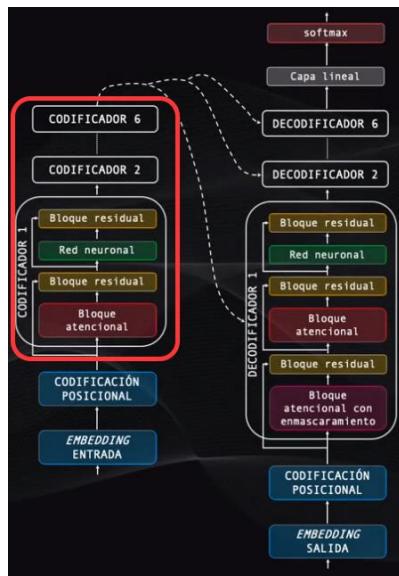


Figura 12. El bloque de codificación de la Red Transformer

Fuente: (Sotaquirá, 2020)

Ahora viene el bloque de codificación, que contiene seis codificadores, todos con una estructura idéntica. Analicemos en detalle uno de estos codificadores. Cada codificador tiene cuatro elementos: un bloque atencional, un bloque de conexión residual, una red neuronal y otro bloque de conexión residual:

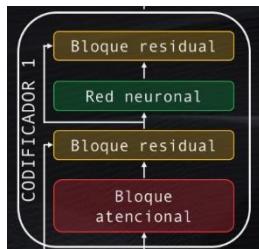


Figura 13. Detalle del primer codificador en la Red Transformer

Fuente: (Sotaquirá, 2020)

Veamos en detalle el bloque atencional, que es tal vez el más importante de toda la red, pues se encarga de analizar la totalidad de la secuencia de entrada (recordemos que la red la procesa de manera simultánea) y de encontrar relaciones entre varias palabras de esta secuencia. Por ejemplo, si el texto de entrada es “I love Italian food”, podemos ver que hay al menos dos posibles asociaciones entre palabras: el verbo “love” y el sujeto (“I”) y el sustantivo “food”

asociado al adjetivo “Italian”. Pero además entre estas dos frases (I Love e Italian Food) también hay una asociación:



Figura 14. La relación entre cada palabra de la frase 'I love Italian food'
Fuente: (Sotaquirá, 2020)

Así, lo que hace el bloque atencional es expresar numéricamente las relaciones que existen a diferentes niveles dentro de la secuencia, y luego codifica cada una de ellas con esta información del contexto, indicando así cuáles son los elementos del texto a los que se deben prestar más atención al momento de hacer la traducción. Esta es precisamente la manera como las redes transformer “comprenden” este contexto para codificar adecuadamente cada palabra. Para lograr esto en primer lugar los tokens se llevan simultáneamente a tres pequeñas redes neuronales, entrenadas para calcular los vectores “query”, “key” y “value”. Estos vectores son simplemente tres representaciones alternativas de los tokens originales: (Sotaquirá, 2020)

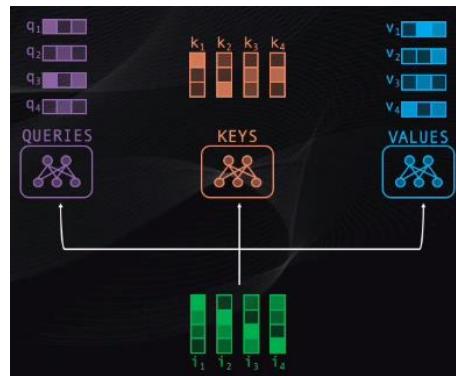


Figura 15. Obtención de los 'queries', 'keys' y 'values'
Fuente: (Sotaquirá, 2020)

Después de esto se toma el query de cada token y se compara con cada uno de los keys existentes. Esta comparación es simplemente una multiplicación de vectores, y con esto se obtendrá un puntaje que mide el grado de asociación entre pares de palabras.

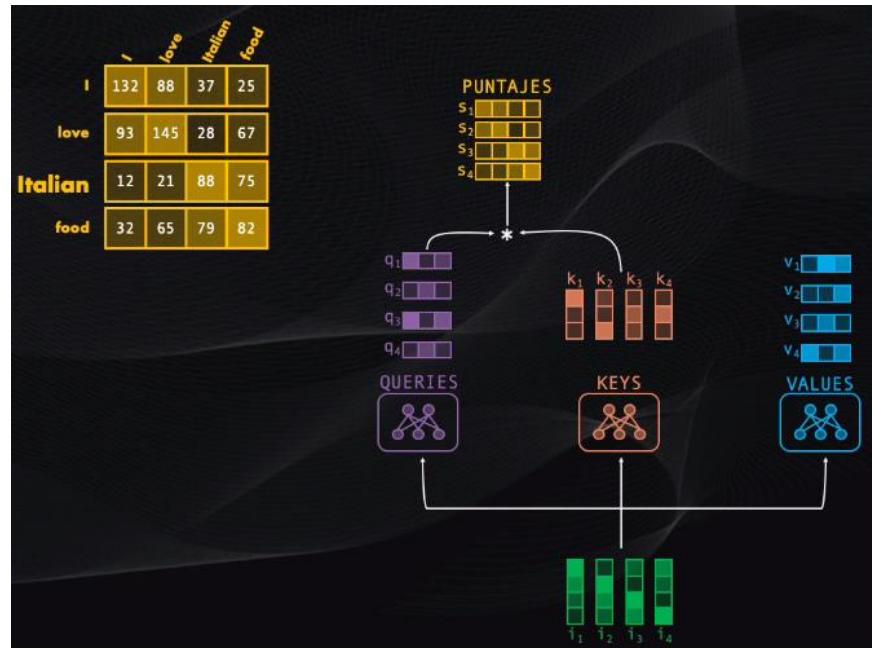


Figura 16. Comparación de 'queries' y 'keys' y obtención de los puntajes
Fuente: (Sotaquirá, 2020)

Así, para el caso de la frase que queremos traducir, si analizamos la palabra “Italian” los puntajes obtenidos indican que al codificar este token se le debería prestar más atención a la propia palabra “Italian” seguida por la palabra “food”, y se debería enfocar menos en las palabras “love” y “I”, que tienen los menores puntajes. La idea es ahora usar estos puntajes para ponderar cada uno de los vectores values, indicando así la importancia de cada palabra al momento de la codificación de los tokens. Para poder hacer esto se deben escalar los puntajes, dividiéndolos primero entre el tamaño de cada vector, y luego llevándolos a una función softmax. Esta función permite simplemente representar cada puntaje como una probabilidad entre cero y uno: (Sotaquirá, 2020)

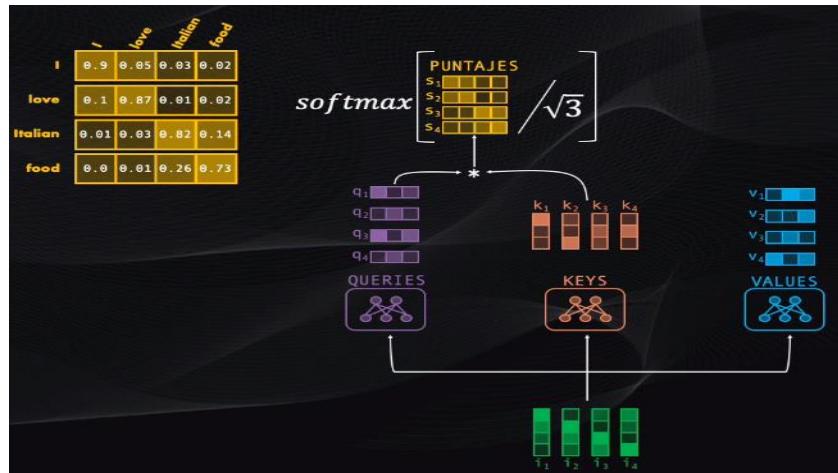


Figura 17. Aplicación de la función 'softmax' a la matriz de puntajes
Fuente: (Sotaquirá, 2020)

Un valor cercano a uno indica que la red debe prestarle más atención a ese token en particular, y un valor cercano a 0 que la palabra no es muy relevante. Finalmente, se debe condensar toda esta información resultante de la comparación en un solo vector por cada token. Así que tomamos la matriz de puntajes que acabamos de obtener y la multiplicamos por la matriz de values: el resultado serán cuatro nuevos tokens, que contendrán la codificación de la información de contexto más relevante para cada palabra de la secuencia: (Sotaquirá, 2020)

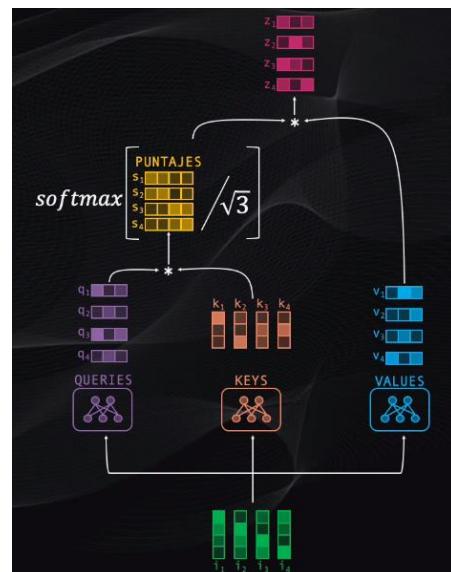


Figura 18. La salida del bloque atencional
Fuente: (Sotaquirá, 2020)

Así que, en resumen, el bloque atencional toma los tokens iniciales y codifica en los tokens resultantes los elementos de la secuencia a los que se debe dar más relevancia. Sin embargo, recordemos que en nuestra frase original encontramos, además de asociaciones entre palabras, asociaciones entre frases: para traducir la porción “Italian food” se necesita prestar atención a “I love”. Así que un solo bloque atencional no es suficiente. Al usar múltiples bloques atencionales es posible detectar y codificar asociaciones entre palabras y grupos de palabras a diferentes niveles. Las salidas de estos bloques se combinan en una última red neuronal que condensa toda la información resultante en un único vector para cada token de entrada: (Sotaquirá, 2020)

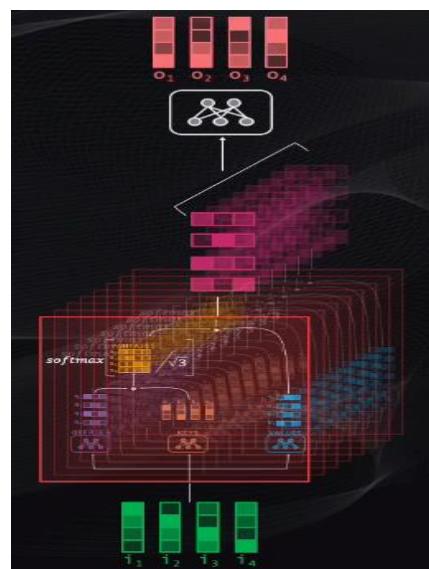


Figura 19. Los múltiples bloques atencionales
Fuente: (Sotaquirá, 2020)

Codificación y el bloque residual

A este bloque se llevan tanto la entrada como la salida del bloque atencional, y esto se hace pues la red es muy profunda y si tan solo se enviara la salida la información progresivamente se degradaría y esto dificultaría el entrenamiento y desempeño de la red. Esta etapa toma los dos datos, los suma y luego los normaliza para que tengan la escala adecuada requerida por el siguiente bloque: (Sotaquirá, 2020)

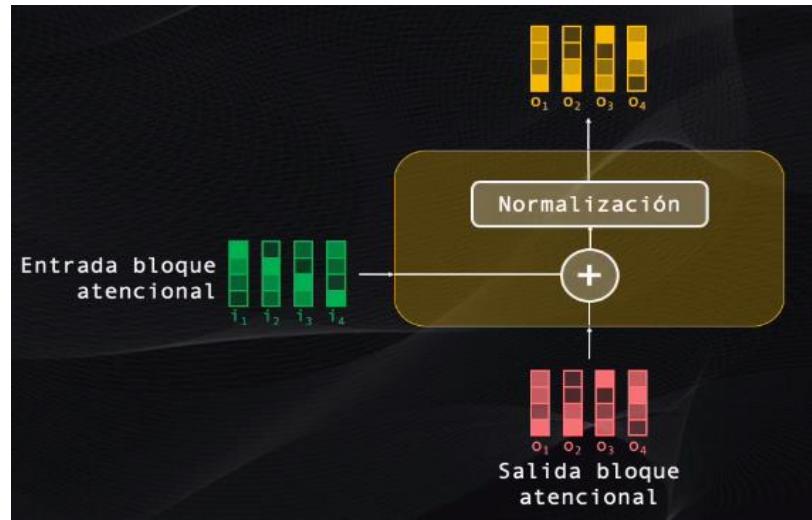


Figura 20. Los elementos del bloque residual

Fuente: (Sotaquirá, 2020)

Codificación: red neuronal y otro bloque residual

Después de esto tenemos una red neuronal seguida por un bloque residual:

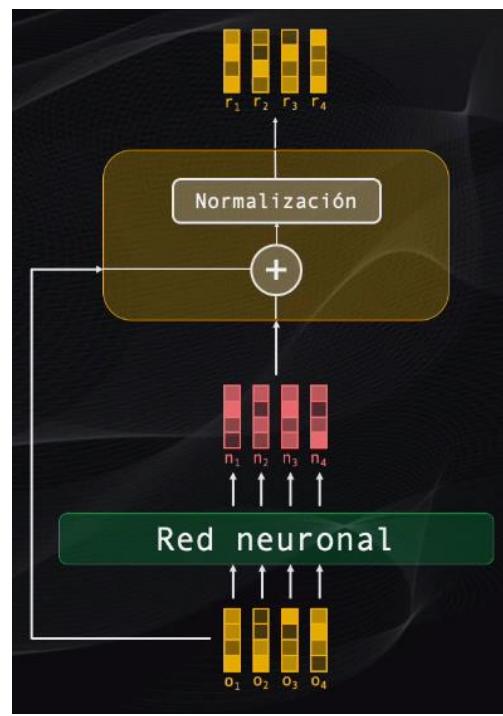


Figura 21. La etapa de salida del codificador: una red neuronal seguida de un bloque residual

Fuente: (Sotaquirá, 2020)

La red neuronal procesa en paralelo todos los vectores de la secuencia, tomando la información atencional de las capas anteriores y consolidándola en una única representación. La entrada y la salida de esta red neuronal son luego llevadas a un bloque residual que tiene exactamente las mismas características del bloque anterior: una suma seguida por una normalización de los datos.

Codificación: resultado final

Este bloque toma los tokens de entrada, los procesa en paralelo y entrega a la salida una representación que contiene información atencional sobre las diferentes relaciones entre palabras o grupos de palabras de la secuencia, importantes al momento de la traducción:

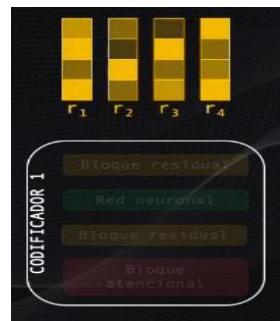


Figura 22. Salida resultante del primer codificador

Fuente: (Sotaquirá, 2020)

Y este proceso se repite para los codificadores restantes, que son idénticos en estructura al codificador que acabamos de analizar.

Decodificación

Ahora nos enfocamos el segundo bloque importante de la red transformer, que se encarga de hacer la traducción:

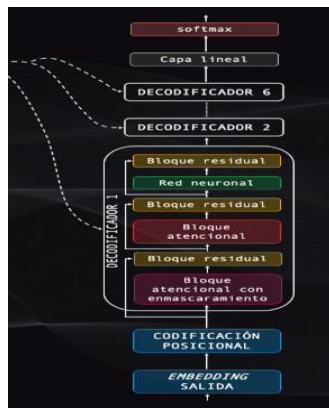


Figura 23. Elementos del decodificador

Fuente: (Sotaquirá, 2020)

En primer lugar, tenemos los bloques de embedding de salida y un codificador posicional, que cumplen exactamente la misma función de los bloques que vimos en la etapa de codificación. Luego viene el decodificador, que es muy similar al bloque de codificación: en total cuenta con 6 decodificadores, cada uno de ellos conectado al codificador, lo que permite conocer la información atencional de la entrada, en el idioma original, para poder realizar la traducción. Cada decodificador es similar a los bloques de codificación que vimos anteriormente: cuenta con bloques atencionales, residuales y redes neuronales que tienen la misma estructura de los codificadores. Sin embargo, tienen un bloque atencional de enmascaramiento y un bloque residual adicionales. Luego viene una capa lineal que, junto con la capa softmax, permite generar una a una las palabras de la secuencia de salida. Veamos entonces cómo funciona paso a paso la decodificación. (Sotaquirá, 2020)

Decodificación: bloque atencional con enmascaramiento

La traducción comienza con la palabra clave “inicio”, la cual es codificada con el embedding y posicionalmente. Al ingresar al primer decodificador es procesada por el bloque atencional de enmascaramiento. Este bloque es prácticamente idéntico al bloque atencional visto anteriormente: codifica la relación entre diferentes elementos de la secuencia de salida, usando los queries, keys y values vistos en la Figura 12.

Pero con una diferencia importante: como se está generando cada palabra de manera secuencial, una a una, el decodificador debe prestar atención únicamente a la palabra generada actualmente y a las anteriores, no a las futuras. Por ejemplo, si en la secuencia traducida nos

ubicamos en la palabra “la”, el decodificador debería tener acceso a esta palabra y a “amo”, pero no a palabras que aparecerán posteriormente en la secuencia (“comida” e “italiana”): (Sotaquirá, 2020)



Figura 24. Momento del enmascaramiento
Fuente: (Sotaquirá, 2020)

Así que para evitar esto se agrega un bloque que enmascara, es decir que simplemente hace cero, las palabras a las que durante la decodificación no se debe prestar atención:



Figura 25. El resultado del enmascaramiento

Fuente: (Sotaquirá, 2020)

Al igual que con el codificador, en este caso también se emplean múltiples bloques atencionales para detectar relaciones a diferentes niveles.

Decodificación: el bloque atencional

Todos los bloques residuales, así como la red neuronal de este decodificador funcionan de forma idéntica a como ocurría en los codificadores. Así que nos enfocaremos ahora en el bloque atencional que en este caso tiene la misma estructura, pero un funcionamiento ligeramente diferente al del codificador. Este bloque enfoca su atención tanto en la secuencia original como en la de salida y para ello toma la salida del codificador y las lleva a las redes “queries” y “keys”, mientras que el nodo “values” usa como entrada el dato proveniente del bloque residual anterior: (Sotaquirá, 2020)

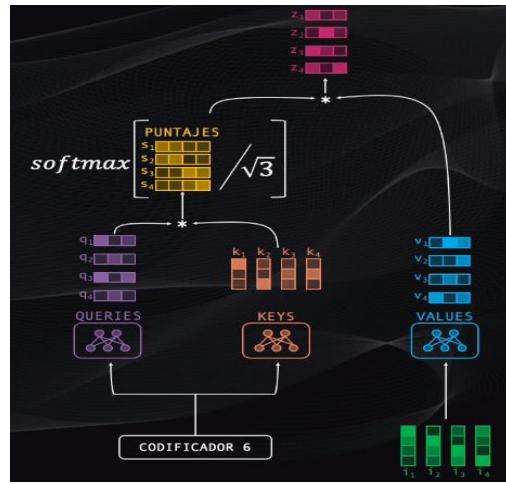


Figura 26. El bloque atencional del decodificador
Fuente: (Sotaquirá, 2020)

Es de esta manera como el codificador le indica al decodificador a qué elementos debe prestar más atención al momento de generar la secuencia de salida. De nuevo, se usan múltiples bloques atencionales de manera simultánea para codificar asociaciones a diferentes niveles.

Decodificación: múltiples decodificadores y etapas de salida

Este bloque se replica un total de seis veces, y al final genera un vector con cantidades numéricas, y lo único que falta es convertirlo, en una palabra:



Figura 27. Múltiples decodificadores para encontrar relaciones entre palabras y grupos de palabras a diferentes niveles
Fuente: (Sotaquirá, 2020)

Para eso se usa en primer lugar la capa lineal, que es simplemente una red neuronal que toma el vector producido por el decodificador y lo transforma en un vector mucho más grande. Por ejemplo, si el traductor aprende 10000 palabras (es decir el tamaño del vocabulario), entonces el vector de salida de la capa lineal tendrá precisamente 10000 elementos. La capa softmax toma cada elemento de este vector y lo convierte en una probabilidad, todas con valores positivos entre 0 y 1. La posición con la probabilidad más alta será seleccionada y la palabra asociada con dicha posición será precisamente la salida del modelo en ese instante de tiempo:

(Sotaquirá, 2020)

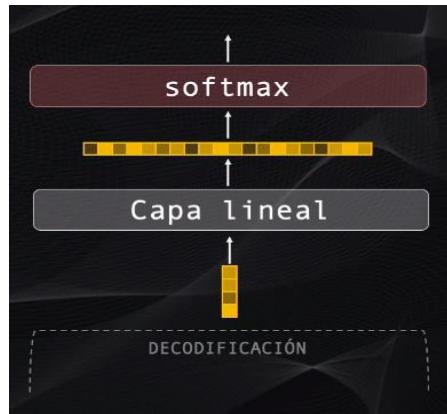


Figura 28. Salida del bloque de decodificación
Fuente: (Sotaquirá, 2020)

Y el proceso se repite hasta generar la totalidad de la secuencia de salida.

3.2.6 Dataset

Un dataset es un conjunto de datos organizados de forma sistemática, que se utiliza generalmente para entrenar modelos de aprendizaje automático. Estos conjuntos de datos pueden contener una variedad de tipos de información, como textos, imágenes, audios, etc. El objetivo principal de un dataset es proporcionar suficiente información al modelo para que aprenda patrones y pueda hacer predicciones o tomar decisiones. (Solis, 2023)

En contexto de la arquitectura Transformer, como la utilizada en modelos como GPT (Generative Pre-trained Transformer), un dataset se utiliza para entrenar el modelo en tareas específicas. La arquitectura Transformer se basa en mecanismos de atención multi-cabeza que permiten al modelo capturar relaciones entre diferentes partes de la entrada, lo que la hace especialmente efectiva para tareas de procesamiento de lenguaje natural (NLP). (Ashish Vaswani, 2017)

A continuación, se explica cómo funciona un dataset en la arquitectura Transformer:

1. Preprocesamiento del Dataset: Antes de alimentar el dataset al modelo Transformer, es necesario realizar un preprocesamiento. Esto puede incluir tokenización (dividir el texto en palabras o subpalabras), normalización (convertir todo a minúsculas, eliminar signos de puntuación, etc.), y otras transformaciones necesarias para preparar los datos para el modelo. (Sotaquirá, 2023)
2. División del Dataset: El dataset suele dividirse en conjuntos de entrenamiento, validación y prueba. El conjunto de entrenamiento se utiliza para entrenar el modelo, el conjunto de validación se utiliza para ajustar los hiperparámetros y prevenir el sobreajuste, y el conjunto de prueba se utiliza para evaluar el rendimiento final del modelo. (Sotaquirá, 2023)
3. Entrenamiento del Modelo: Una vez que el dataset está preparado y dividido, se alimenta al modelo Transformer durante el proceso de entrenamiento. Durante este proceso, el modelo aprende a mapear las entradas del dataset a las salidas deseadas (por ejemplo, predecir la siguiente palabra en una frase). (SANTOS, 2022)
4. Aprendizaje de Características: El modelo Transformer utiliza mecanismos de atención para aprender características importantes de los datos. Esto le permite capturar relaciones a largo plazo en el texto y comprender el contexto en el que se encuentra cada palabra o token. (SANTOS, 2022)
5. Evaluación del Modelo: Una vez que el modelo ha sido entrenado, se evalúa su rendimiento utilizando el conjunto de prueba. Esto permite determinar cómo de bien el modelo generaliza a datos no vistos durante el entrenamiento. (SANTOS, 2022)

3.2.6.1 Recolección de Datos para el Dataset

En el proceso de recolección de datos para la creación de nuestro dataset, empleamos una variedad de generadores y recursos. La elección de estas herramientas se basó en la necesidad de obtener datos de alta calidad y diversidad que representaran de manera precisa el dominio o problema que queríamos abordar. (López, 2019)

Algunos de los generadores que se utilizó son:

- **CSS Grid Generator** consiste en un generador de rejillas o tablas con el número de filas y columnas que tú elijas. El generador es muy simple. Número de columnas, número de filas, margen entre columnas y filas, asignar un nombre a filas y columnas... Tras pulsar en Mostrar Código obtendremos el código CSS que deberemos procesar para agregar a nuestro dataset. (López, 2019)
- **Ultimate CSS Gradient Generator** se utilizó para generar estilos CSS con, degradados de colores para fondos o menús. Además de las plantillas ya creadas puedes combinar varios colores y generar el degradado en la orientación que elijas (vertical, horizontal, diagonal, radial) y ocupan el tamaño que necesites. Además, puedes usar la tonalidad, la saturación y el brillo, así como elegir un formato de colores entre HEX, RGB o HSL. Luego podremos seleccionar el código y copiarlo. Este generador de degradado CSS es compatible con la práctica totalidad de navegadores web modernos: IE, Edge, Firefox, Chrome, Safari, Opera, iOS, Android, por lo que fue un generador que nos brindó estilos CSS con el diferenciador del degradado de color para nuestro dataset. (López, 2019)
- **CSS Code Generators** En el caso de CSS Code Generators no nos encontraremos con un generador CSS sino con once, cada uno para una tarea concreta, como crear una caja con sombreado, configurar una imagen de fondo, generar columnas y tablas, marcos y bordes, estilos de letra y sombreado de texto, degradados de color, etc. Solo tenemos que elegir el generador en cuestión y cambiar los elementos de los menús para obtener un resultado. Además, según el generador se ofrece una galería de ejemplos. Luego veremos el resultado en formato HTML, CSS y tal y como se verá en tu sitio web una vez publicado. (López, 2019)

- **The Ultimate CSS Generator** nos permite crear código relacionado con animaciones, fondos, bordes y marcos, colores, filtros para imágenes, textos, listas... Elegimos el generador que queremos, que nos indicará si es compatible con los navegadores modernos, y luego solo tenemos que retocar los elementos para ver cómo cambia la vista previa. Debajo de la vista previa veremos el código que podemos copiar pulsando en el botón correspondiente. (López, 2019)
- **EnjoyCSS** tiene los generadores habituales relacionados con botones, textos, marcos, fondos, sombreados, transiciones, patrones... Aunque cada generador tiene sus propias características, en general funcionan con una lista de opciones en forma de menú con valores que debemos cambiar para configurar el código CSS. En todo momento veremos el resultado final, y cuando esté todo en orden, bastará con pulsar en Get the code para obtener el código fuente CSS. (López, 2019)

Además, también optamos por obtener código de GitHub, ya que en esta plataforma podemos encontrar código de buena calidad listo para ser utilizado en nuestro dataset. Algunos repositorios de GitHub de referencia podrían incluir:

- Repositorio de Bootstrap: <https://github.com/twbs/bootstrap>
- Repositorio de Normalize.css: <https://github.com/necolas/normalize.css>
- Repositorio de Materialize CSS: <https://github.com/Dogfalo/materialize>
- Repositorio de CSS Frameworks: <https://github.com/awesome-css-group/awesome-css>

3.2.6.2 Datasets de Entrenamiento, Prueba, Evaluacion

Tipos de Conjuntos de Datos de Entrenamiento

Los conjuntos de datos de entrenamiento (train), validación (validation) y prueba (test) son esenciales en el desarrollo y evaluación de modelos de aprendizaje automático y aprendizaje profundo. Cada uno tiene un propósito específico que ayuda a garantizar que el modelo sea capaz de generalizar bien a datos no vistos y a evitar problemas como el sobreajuste.

A continuación, explicaremos cada uno de manera específica:

1. Conjunto de Datos de Entrenamiento (TRAIN):

Propósito: El conjunto de datos de entrenamiento se utiliza para entrenar el modelo. Durante el entrenamiento, el modelo ajusta sus parámetros utilizando este conjunto de datos para aprender a hacer predicciones o realizar tareas específicas. (aprendemachinelearning, 2020)

Características:

- Tamaño: Suele ser el conjunto de datos más grande.
- Variedad: Debe ser representativo de los datos que el modelo encontrará en la práctica.
- Etiquetas: Normalmente incluye las etiquetas o respuestas correctas que el modelo debe aprender a predecir.

Importancia: Un buen conjunto de datos de entrenamiento es crucial para que el modelo aprenda patrones relevantes y construya una representación interna adecuada de los datos. (aprendemachinelearning, 2020)

2. Conjunto de Datos de Validación (VALIDATION):

Propósito: El conjunto de datos de validación se utiliza para ajustar los hiperparámetros del modelo y para evaluar su rendimiento durante el entrenamiento. (aprendemachinelearning, 2020)

Características:

- Tamaño: Suele ser más pequeño que el conjunto de datos de entrenamiento.

- Variedad: Debe ser similar al conjunto de datos de entrenamiento, pero independiente de él.
- Etiquetas: Normalmente incluye las etiquetas o respuestas correctas para evaluar el rendimiento del modelo.

Importancia: Permite ajustar los hiperparámetros del modelo (como la tasa de aprendizaje (learning rate), el tamaño del lote (batch size), etc.) y evitar el sobreajuste (overfitting) al proporcionar una evaluación independiente del rendimiento del modelo durante el entrenamiento. (aprendemachinelearning, 2020)

3. Conjunto de Datos de Prueba (TEST):

Propósito: El conjunto de datos de prueba se utiliza para evaluar el rendimiento final del modelo después de que ha sido entrenado y validado. (aprendemachinelearning, 2020)

Características:

- Tamaño: Suele ser similar al conjunto de datos de validación.
- Variedad: Debe ser similar al conjunto de datos de entrenamiento y validación, pero independiente de ellos.
- Etiquetas: Normalmente incluye las etiquetas o respuestas correctas para evaluar el rendimiento del modelo de manera objetiva.

Importancia: Proporciona una evaluación final e imparcial del rendimiento del modelo en datos no vistos, lo que ayuda a estimar cómo se comportará el modelo en la práctica real. (aprendemachinelearning, 2020)

Estos conjuntos de datos o datasets juegan roles críticos en el desarrollo de modelos de aprendizaje automático y aprendizaje profundo, ayudando a garantizar que los modelos sean precisos, generalicen bien a nuevos datos y sean útiles en aplicaciones del mundo real.

3.2.7 Tokenizacion (Tokenization)

La tokenización es un proceso fundamental en el campo de la Inteligencia Artificial, especialmente en el procesamiento del lenguaje natural (NLP, por sus siglas en inglés). Se refiere al proceso de dividir una cadena de texto en unidades más pequeñas llamadas "tokens". Un token puede ser tan pequeño como un carácter o tan grande como una palabra o incluso una frase completa. (Kosar, 2022)

¿Para qué sirve la tokenización?

- 1. Preprocesamiento de Texto:** Antes de entrenar un modelo de NLP, es necesario convertir el texto en una forma que la máquina pueda entender y procesar. La tokenización es el primer paso en este preprocesamiento. (Kosar, 2022)
- 2. Análisis de Texto:** La tokenización permite realizar análisis estadísticos sobre el texto, como contar la frecuencia de palabras, identificar n-gramas, etc. (Kosar, 2022)
- 3. Algoritmos de Machine Learning:** Los modelos de aprendizaje automático, especialmente los modelos de lenguaje como BERT o GPT, requieren el texto tokenizado como entrada. (Kosar, 2022)

Técnicas de Tokenización

1. Tokenización Basada en Espacios (Whitespace Tokenization):

- Descripción: Divide el texto en tokens basándose en los espacios entre palabras.
- Ejemplo: "Hola, mundo" se tokeniza en ["Hola,", "mundo"].

2. Tokenización Basada en Palabras (Word Tokenization):

- Descripción: Divide el texto en palabras individuales.
- Ejemplo: "La casa es grande." se tokeniza en ["La", "casa", "es", "grande", "."].
- Herramientas: NLTK, spaCy.

3. Tokenización Basada en Caracteres (Character Tokenization):

- Descripción: Divide el texto en caracteres individuales.
- Ejemplo: "Casa" se tokeniza en ["C", "a", "s", "a"].

4. Tokenización con Subword Units:

- Descripción: Divide el texto en subunidades más pequeñas que las palabras, como prefijos, sufijos o raíces.
- Ejemplo: "Corriendo" se tokeniza en ["Cor", "riendo"].
- Herramientas: Byte-Pair Encoding (BPE), WordPiece, SentencePiece.

5. Tokenización Basada en Expresiones Regulares:

- Descripción: Usa patrones definidos por expresiones regulares para dividir el texto en tokens.
- Ejemplo: Divide el texto basándose en patrones como números, símbolos, etc.

6. Tokenización con Aprendizaje Automático:

- Descripción: Entrena un modelo para aprender las reglas de tokenización basadas en un corpus de texto.
- Herramientas: Aprendizaje supervisado, redes neuronales.

3.2.8 Incrustacion (Embedding)

Embedding: Un embedding en el contexto de procesamiento del lenguaje natural y modelos de aprendizaje automático, como los Transformers, es una representación vectorial de una palabra, token o entidad en un espacio dimensional. Estas representaciones vectoriales capturan el significado semántico y la relación con otras palabras o tokens en el contexto del lenguaje. (Kosar, 2023)

Importancia de Entender el Embedding

Conexión entre Tokenización y Embedding: Después de tokenizar el texto y obtener una secuencia de IDs de tokens, estos IDs se convierten en embeddings para ser procesados por el modelo. (Kosar, 2023)

Representación Semántica: Los embeddings capturan el significado semántico de las palabras o tokens, permitiendo que el modelo entienda y procese el texto de manera más efectiva.

¿Qué son los Embeddings?

Vectorización de Tokens: Los embeddings son vectores numéricos que representan cada token en un espacio dimensional. Por ejemplo, la palabra "programar" podría tener un embedding como [0.2, 0.5, -0.3].

Aprendizaje durante el Entrenamiento: Durante el entrenamiento, los embeddings se ajustan para capturar las relaciones semánticas y sintácticas entre los tokens, optimizando así el rendimiento del modelo en tareas específicas. (Kosar, 2023)

Integración en la Arquitectura Transformer

Entrada del Modelo: Los embeddings se utilizan como entrada para las capas del modelo Transformer, donde se lleva a cabo el procesamiento y las operaciones necesarias para la tarea específica, como generación de texto, clasificación, o en nuestro caso generación de código CSS.

Contextualización: En modelos como GPT-2 XL, los embeddings son contextuales, lo que significa que el embedding de un token puede variar según el contexto en el que aparece, mejorando así la comprensión del modelo sobre el lenguaje natural. (Kosar, 2023)

3.2.8.1 Tokenizacion VS Incrustacion

La tokenización y la incrustación (embedding) son dos conceptos fundamentales en el procesamiento de lenguaje natural (NLP) y el aprendizaje automático.

A continuación, se explicará las diferencias entre ellos:

- La entrada se tokeniza, los tokens luego se incrustan
- Las incrustaciones (Embeddings) de texto de salida se clasifican nuevamente en tokens, que luego se pueden decodificar en texto.
- La tokenización convierte un texto en una lista de números enteros.
- La incrustación (Embedding) convierte la lista de números enteros en una lista de vectores (lista de incrustaciones)
- La información posicional sobre cada token se agrega a las incrustaciones mediante codificaciones posicionales (positional encodings) o incrustaciones.

Cómo se Relacionan:

De Tokenización al Embeddings: Primero, se tokeniza el texto para obtener una secuencia de IDs de tokens. Luego, estos IDs de tokens se pasan al modelo para obtener los embeddings correspondientes. Ejemplo: Si la frase "Me gusta programar en Python" se tokeniza en [10, 45, 123, 7], estos IDs se utilizan para obtener los embeddings de cada token mediante el modelo. (Rathinapandi, 2023)

De Embeddings (Incrustaciones) al Procesamiento del Modelo: Una vez obtenidos los embeddings, se utilizan como entrada para las capas del modelo Transformer. El modelo realiza operaciones matriciales y funciones de activación para procesar estos embeddings y generar salidas, como la predicción de la siguiente palabra. (Rathinapandi, 2023)

A continuación, se muestra una breve descripción de este proceso en la Figura 29

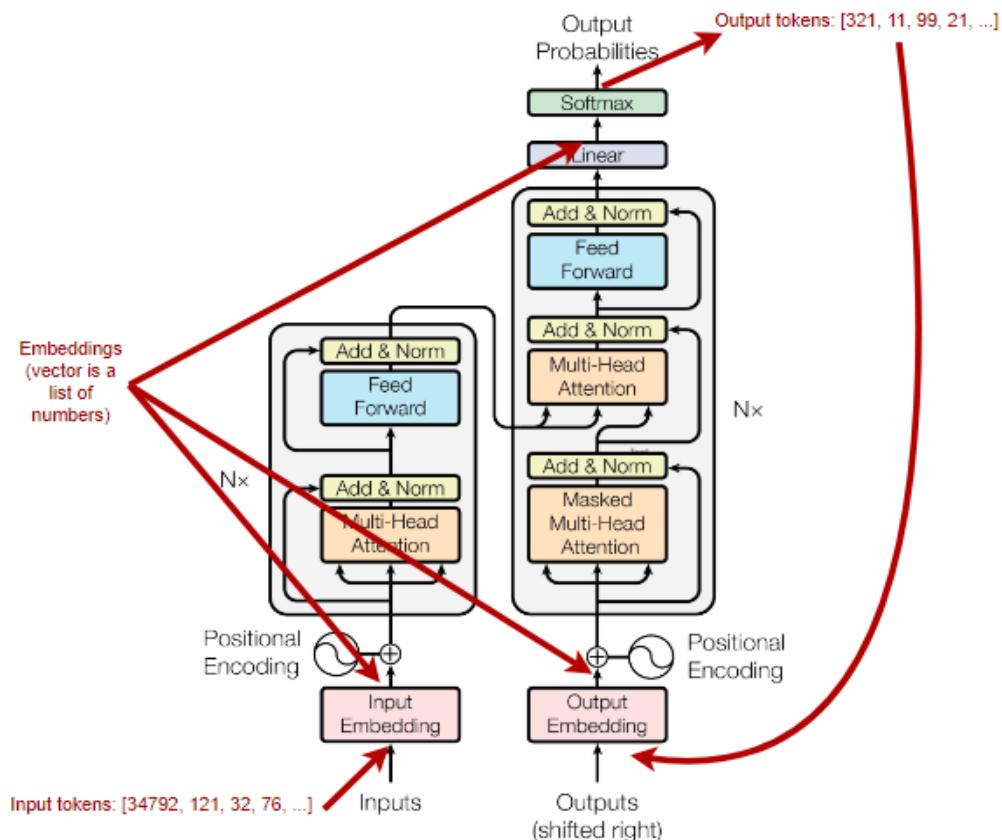


Figura 29. Tokenizacion, Incrustacion y Procesamiento del Modelo
Fuente: Elaboración Propia

3.2.9 Fine-Tuning

El Fine-Tuning o ajuste fino permite tomar un modelo entrenado que realiza bien una determinada tarea y aprovechar todo su conocimiento para resolver una nueva tarea específica; aunque, por supuesto, con ciertas reglas.

Se trata de un proceso en el que se realiza un “ajuste fino” de algunas capas de la red neuronal para obtener las salidas deseadas. Es decir, se ajustan ligeramente ciertas representaciones del modelo preentrenado para que este resulte más relevante en el problema que se desea resolver. Así, se evita definir la estructura de la red neuronal y realizar el entrenamiento desde cero. (Torres, 2023)

3.2.9.1 Origen del Fine-Tuning en NLP

En el campo del NLP, el uso de modelos pre-entrenados como Word2Vec, GloVe y, más recientemente, modelos Transformer como BERT, GPT y otros, ha llevado a la adopción del "fine-tuning" como una técnica efectiva para adaptar estos modelos a tareas específicas o dominios particulares. (Torres, 2023)

3.2.9.2 Transfer Learning (Aprendizaje por Transferencia)

El "transfer learning" o aprendizaje por transferencia es una técnica en el aprendizaje automático y el aprendizaje profundo que consiste en utilizar conocimientos adquiridos al entrenar un modelo en una tarea específica (tarea fuente) para mejorar el rendimiento en una tarea relacionada pero diferente (tarea objetivo). Esta técnica se ha vuelto muy popular y efectiva en diferentes campos del aprendizaje automático y especialmente en el procesamiento de lenguaje natural (NLP), visión por computadora y otras áreas de aprendizaje automático. (UNIR, 2023)

3.2.9.3 Fine Tuning vs Transfer Learning (Aprendizaje por Transferencia)

Es preciso diferenciar lo que es el Fine-Tuning del Transfer Learning (aprendizaje por transferencia): aunque los dos son enfoques de entrenamiento de redes con algunos datos y ambos se basan en el conocimiento existente, la realidad es que poseen grandes diferencias. (Kumar, 2024)

En el Transfer Learning (aprendizaje por transferencia), se toma un modelo entrenado en una tarea para reutilizarlo en la resolución de otra, pero congelando los parámetros del modelo existente. El proceso es el siguiente: (Kumar, 2024)

- Se carga el modelo entrenado y se congelan las capas preentrenadas para evitar la pérdida de información.
- Se añaden nuevas capas entrenables sobre las congeladas, que se entranan con otro conjunto de datos.

En el Fine Tuning (ajuste fino), en cambio, se toman los parámetros de la red existente para entrenarlos aún más y así realice la segunda tarea. Básicamente, se adapta la estructura del modelo y se entrena. Para ello, el procedimiento es este: (Kumar, 2024)

- Al modelo existente se le eliminan y agregan capas necesarias para la resolución de la nueva tarea.
- En la nueva estructura de modelo se congelan solo aquellas capas que provienen de la red original, cuyo conocimiento se desea conservar para el nuevo entrenamiento.
- Se procede a entrenar el modelo con los nuevos datos para la nueva tarea. Solo se actualizan los pesos de las capas nuevas.

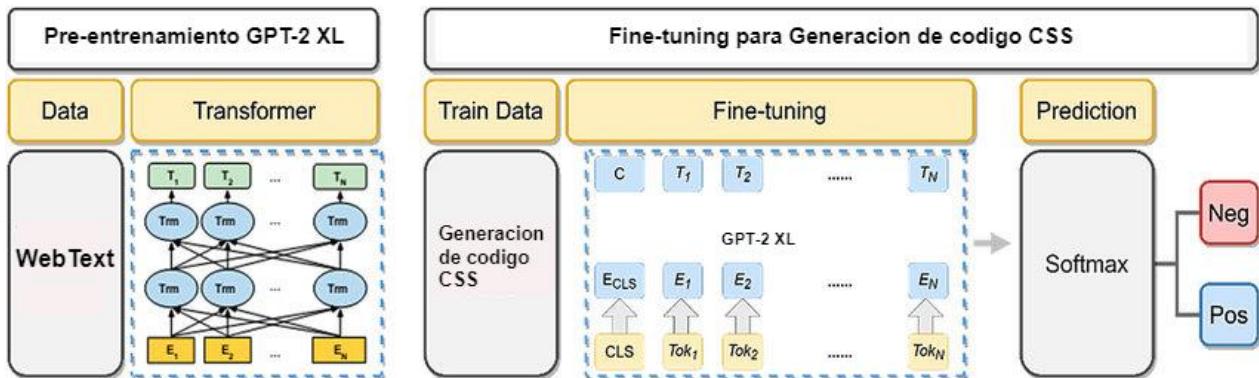


Figura 30. Fine Tuning Aplicado al Transformer para la Generacion de codigo CSS
Fuente: Elaboración Propia

3.3. Marco Teórico de Ingeniería

3.3.1 Metodología de desarrollo

La programación extrema (XP) es una metodología ágil de gestión de proyectos que se centra en la velocidad y la simplicidad con ciclos de desarrollo cortos. Esta metodología se basa en 5 valores, 5 reglas y 12 prácticas de programación. Si bien tiene una estructura rígida, el resultado de estos sprints altamente centrados y las integraciones continuas buscan dar como resultado un producto de mayor calidad.

3.3.1.1 ¿Qué es la programación extrema (XP)?

La programación extrema es una metodología ágil de gestión de proyectos que se centra en la velocidad y la simplicidad con ciclos de desarrollo cortos y con menos documentación. La estructura del proceso está determinada por 5 valores fundamentales, 5 reglas y 12 prácticas de XP. (Raeburn, 2022)

Al igual que otras metodologías ágiles, la programación extrema es un método de desarrollo de software dividido en sprints de trabajo. Los marcos ágiles siguen un proceso iterativo, en el que se completa y revisa el marco al final de cada sprint, refinándolo para adaptarlo a los requisitos cambiantes y alcanzar la eficiencia máxima. Al igual que otros métodos ágiles, el diseño de la programación extrema permite a los desarrolladores responder a las solicitudes de los clientes, adaptarse y realizar cambios en tiempo real. Sin embargo, la programación extrema es mucho más disciplinada; realiza revisiones de código frecuentes y pruebas unitarias para realizar cambios rápidamente. Además, es muy creativa y colaborativa, ya que promueve el trabajo en equipo durante todas las etapas de desarrollo. (Raeburn, 2022)

3.3.1.2 Ciclo de vida de la programación extrema (XP)

El ciclo de vida de XP fomenta la integración continua, ya que requiere que los miembros del equipo trabajen casi constantemente, cada hora o todos los días. Sin embargo, el ciclo de vida completo se estructura de la siguiente manera:

- Extraer trabajos sin finalizar de las historias de usuarios
- Priorizar los elementos más importantes

- Comenzar con la planificación iterativa
- Incorporar un plan realista
- Mantener una comunicación constante con todas las partes interesadas y empoderar al equipo
- Presentar el trabajo
- Recibir comentarios
- Regresar a la etapa de planificación iterativa y repetir si es necesario.

3.3.1.3 ¿Cuáles son las 12 prácticas de la programación extrema?

Para perfeccionar aún más el proceso, la programación extrema también implementa un conjunto de 12 prácticas a lo largo del proceso. Se basan en el Manifiesto ágil, pero se adaptan a las necesidades de la programación extrema. (Raeburn, 2022)

- 1. El juego de planificación:** La planificación XP se usa para guiar el trabajo. Los resultados de la planificación deben ser los objetivos que pretendas alcanzar, los plazos previstos y los pasos a seguir. (Raeburn, 2022)
- 2. Pruebas de clientes:** Cuando finalices una función nueva, el cliente desarrollará una prueba de aceptación para determinar si has cumplido con la historia de usuario original. (Raeburn, 2022)
- 3. Pequeñas entregas:** La programación extrema realiza entregas pequeñas y periódicas para obtener información importante durante todo el proceso. A menudo, las entregas se envían directamente a los clientes, aunque también pueden enviarse internamente. (Raeburn, 2022)
- 4. Diseño simple:** El sistema XP está diseñado para ser simple, producirá solo lo necesario y nada más. (Raeburn, 2022)
- 5. Programación en parejas:** Toda la programación la realizan simultáneamente dos desarrolladores que se sientan físicamente uno al lado del otro. No hay trabajo individual en la programación extrema. (Raeburn, 2022)
- 6. Desarrollo guiado por pruebas (TDD):** Debido a que la programación extrema se basa en los comentarios, se requieren pruebas exhaustivas. A través de ciclos cortos,

los programadores realizan pruebas automatizadas para luego reaccionar de inmediato. (Raeburn, 2022)

7. **Refactorización:** Aquí es donde se deberá prestar especial atención a los detalles más finos del código base, para eliminar los duplicados y asegurarse de que el código sea coherente. De esta manera obtendrás diseños simples y de alta calidad. (Raeburn, 2022)
8. **Propiedad colectiva:** Cualquier par de desarrolladores puede modificar el código en cualquier momento, independientemente de que lo hayan desarrollado o no. En la programación extrema, la codificación se realiza en equipo, y el trabajo de todos se lleva a cabo según los estándares colectivos más altos. (Raeburn, 2022)
9. **Integración continua:** Los equipos de XP no esperan a que se completen las iteraciones, sino que se integran constantemente. A menudo, un equipo de XP se integrará varias veces al día. (Raeburn, 2022)
10. **Ritmo de trabajo sostenible:** La intensidad de los trabajos de XP requiere que se establezca un ritmo de trabajo sostenible. Los equipos deben determinar cuánto trabajo pueden producir a este ritmo por día y por semana, y usarlo para establecer plazos de trabajo. (Raeburn, 2022)
11. **Metáfora:** La metáfora es, literalmente, una metáfora. Se decide en equipo y se usa un lenguaje para expresar cómo debe funcionar el equipo. Por ejemplo, somos hormigas trabajando en colectivo para construir el hormiguero. (Raeburn, 2022)
12. **Estándares de codificación:** Los equipos de XP siguen un estándar. De la misma manera que un grupo de escritores necesita adoptar el tono de una marca para que parezca que siempre está escribiendo una misma persona, los desarrolladores de XP deben codificar de la misma manera unificada para que parezca que el código esté escrito por un solo desarrollador. (Raeburn, 2022)

3.3.2 Modelo Pre Entrenado

El uso de un modelo pre-entrenado tiene importantes ventajas. Reduce los costos de computación, la huella de carbono, y permite utilizar modelos de última generación sin tener que entrenar uno desde cero.

Openia-community en Hugging Face proporciona acceso a modelos pre-entrenados en una amplia gama de tareas. Cuando se utiliza un modelo pre-entrenado, lo entrenas con un dataset específico para tu tarea. Esto se conoce como fine-tuning, una técnica de entrenamiento que permite adaptar un modelo pre-entrenado a una tarea específica. En el contexto del aprendizaje automático y el procesamiento de lenguaje natural, el fine-tuning se ha convertido en una práctica común y efectiva para mejorar el rendimiento de modelos en tareas relacionadas pero diferentes a la tarea original para la cual fueron entrenados.

Para seleccionar un modelo pre-entrenado se tomó en cuenta diferentes métricas y arquitecturas, considerando también los usos previstos y limitaciones, donde también se consideró métricas relevantes como la cantidad de parámetros (parameters), capas del modelo (layers), capas ocultas (hidden), la cantidad de cabezas (heads) dada la aplicación de la arquitectura Transformer en el modelo pre-entrenado. También se consideró los datasets usados para el entrenamiento del modelo seleccionado (GPT-2 XL), en donde se investigó y analizó diferentes modelos utilizados en el área de NLP (natural language processing) que cuentan con la implementación de una arquitectura de Transformers.

3.3.2.1 GPT-2 XL:

El Modelo pre-entrenado seleccionado considerando todo lo anteriormente establecido es “GPT-2 XL” a continuación se explicará las diferentes características del modelo “GPT-2 XL”.

Dado que las métricas que mencionamos anteriormente fueron un factor relevante a considerar para la selección de nuestro modelo pre-entrenado, a continuación, se dará una breve explicación de las métricas tomadas en consideración con la idea de posteriormente profundizar conceptualmente en las métricas y en su funcionamiento y aplicación en este proyecto. Estas son las siguientes:

3.3.2.2 Parámetros (Parameters):

En el contexto de estos modelos IA como GPT-2 XL, los parámetros son los pesos asociados con las conexiones entre las neuronas en una red neuronal profunda.

Los parámetros en un modelo de lenguaje como GPT (Generative Pre-trained Transformer) son típicamente tensores multidimensionales de números reales que se ajustan durante el entrenamiento para que el modelo pueda aprender a generar texto coherente y relevante basado en los datos de entrada. Estos parámetros capturan la distribución estadística de los datos de entrenamiento y se utilizan para calcular las probabilidades de las palabras siguientes en una secuencia de texto dada la historia previa. (HuggingFace, 2020)

En una red neuronal profunda, los parámetros son típicamente tensores multidimensionales que representan los pesos y sesgos de las conexiones entre las neuronas en las diferentes capas de la red. Estos tensores pueden ser ajustados durante el entrenamiento mediante algoritmos de optimización para minimizar alguna función de pérdida y mejorar el rendimiento del modelo. (HuggingFace, 2020)

A continuación, en el siguiente gráfico se muestra la comparativa de parámetros ('parameters') de GPT-2 XL con respecto a otros modelos disponibles en Hugging Face.

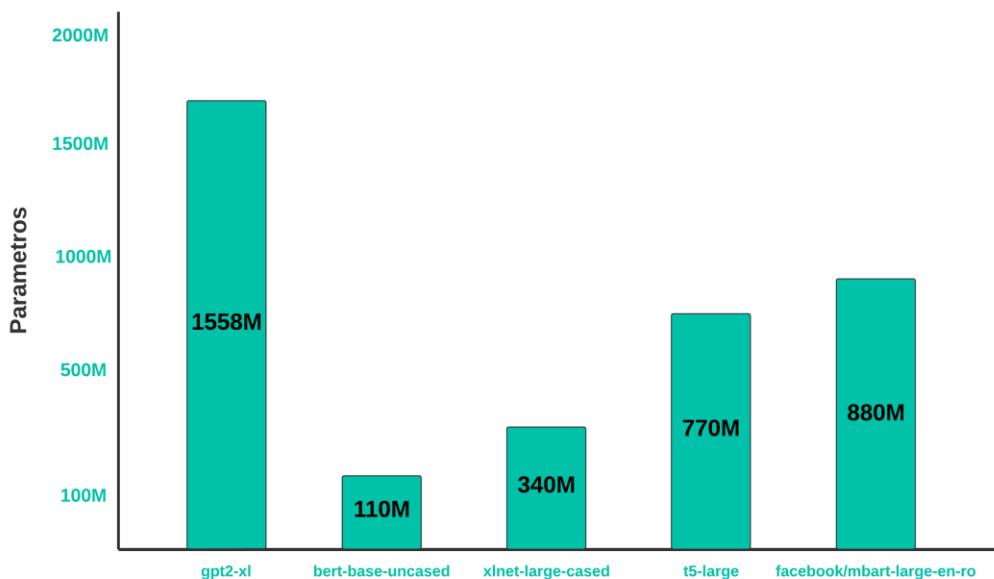


Figura 31. Grafico de modelos pre-entrenados de Hugging Face según sus parametros
Fuente: Elaboración Propia

3.3.2.3 Cabezas (Heads):

En una arquitectura basada en transformers, como la utilizada en modelos como GPT (Generative Pre-trained Transformer) o BERT (Bidirectional Encoder Representations from Transformers), el término "cabezas" se refiere a subdivisiones de atención dentro de una capa de atención multi-cabeza. (Doshi, 2021)

Para entender mejor qué son las cabezas y para qué sirven, primero revisaremos el concepto de atención en una red Transformer:

- **Atención:**

En una arquitectura transformer, la atención es un mecanismo que permite a la red enfocarse en partes específicas de una secuencia de entrada mientras procesa otras partes simultáneamente. Esto es útil para comprender mejor las relaciones entre las diferentes partes de una secuencia, ya que permite que la red preste más "atención" a algunas partes que a otras. (Doshi, 2021)

- **Capa de atención multi-cabeza:**

En una arquitectura transformer, como se utiliza en GPT o BERT, una capa de atención multi-cabeza consta de múltiples cabezas de atención que funcionan de manera paralela y se combinan en última instancia para formar una salida combinada. Cada cabeza de atención opera independientemente y se enfoca en diferentes aspectos de la secuencia de entrada. (Doshi, 2021)

- **Cabezas:**

Las "cabezas" son subdivisiones de atención dentro de una capa de atención multi-cabeza. Cada cabeza de atención tiene sus propios pesos (parámetros) que determinan cómo se realiza la atención. Estas cabezas permiten que el modelo aprenda diferentes representaciones y relaciones entre los tokens de entrada. (Doshi, 2021)

- **Utilidad:**

La utilización de múltiples cabezas de atención en una capa multi-cabeza permite al modelo capturar diferentes patrones y relaciones en los datos de entrada, mejorando así su

capacidad para comprender y generar secuencias de manera efectiva. Las cabezas de atención permiten que el modelo atienda diferentes aspectos de la entrada simultáneamente y, por lo tanto, capturar diferentes tipos de información útil en la secuencia. (HuggingFace, 2020) (paperswithcode, 2023)

A continuación, en el siguiente gráfico se muestra la comparativa de cabezas o cabeceras ('heads) de GPT-2 XL con respecto a otros modelos disponibles en Hugging Face.

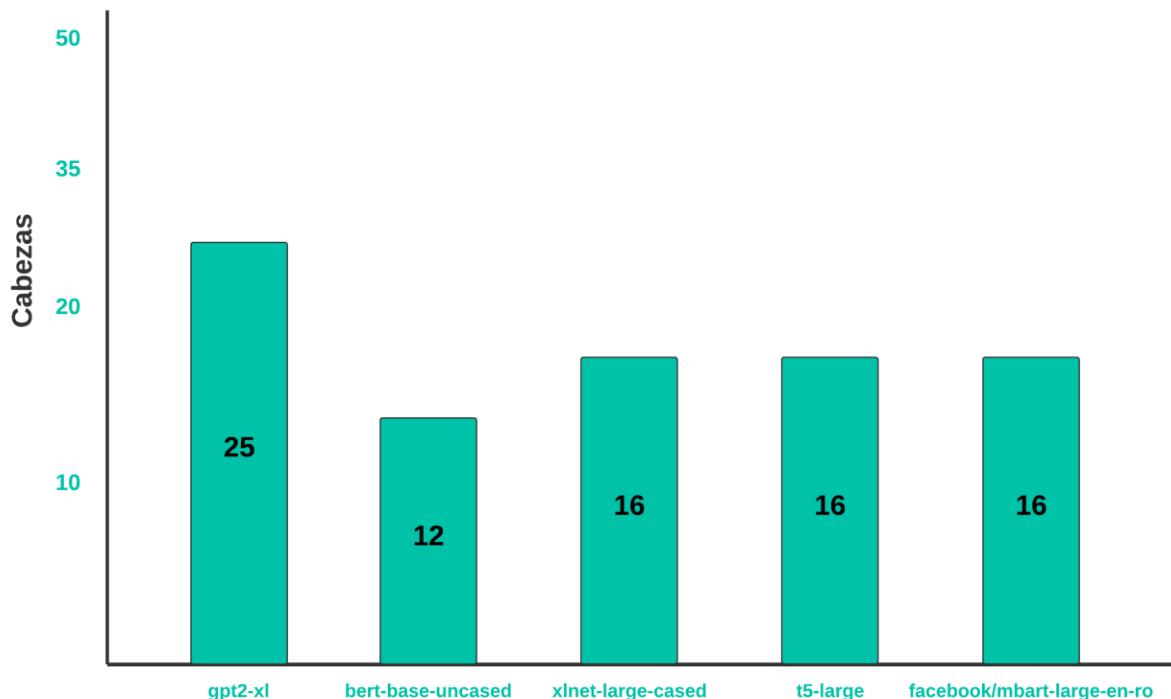


Figura 32. Grafico de modelos pre-entrenados de Hugging Face según sus Cabezas
Fuente: Elaboración Propia

3.3.2.4 Conjunto de datos (Datasets):

El dataset utilizado para el entrenamiento del modelo GPT-2 XL fue WebText pesa 40 GB de textos, pero no se ha hecho público. Aunque se ha publicado la lista de los 1000 dominios principales presentes en WebText.

Este modelo se entrenó (y se evaluó con) WebText, un conjunto de datos que consta del contenido de texto de 45 millones de enlaces publicados por usuarios de la red social "Reddit".

WebText está compuesto de datos derivados de enlaces salientes de Reddit y no consta de datos tomados directamente del propio Reddit. Antes de generar el conjunto de datos, se utilizó una lista de bloqueo para asegurarse de no tomar muestras de una variedad de subreddits que contienen contenido sexual explícito u ofensivo. (paperswithcode, 2023)

3.3.2.5 Accesibilidad al Modelo y otras consideraciones:

En cuanto a la accesibilidad al modelo GPT-2 XL, se encontró mucha información relevante en Hugging Face, junto con una gran cantidad de modelos, conjuntos de datos, espacios y documentaciones disponibles para probar diferentes modelos pre-entrenados. Esta información adicionalmente cuenta con guías y tutoriales de implementación de modelos pre-entrenados.

3.3.2.6 Mecanismos de Atención y Representaciones Distribuidas en GPT-2 XL:

Embeddings de Palabras: Cada palabra en el vocabulario se representa mediante un vector (embedding) de números reales de alta dimensión, capturando su significado y contexto en el lenguaje.

Mecanismos de Atención: Los mecanismos de atención permiten al modelo asignar diferentes pesos o atención a diferentes partes de la entrada, permitiendo capturar relaciones y dependencias entre palabras y secuencias de manera efectiva.

Función de Pérdida y Optimización: Durante el entrenamiento, el modelo se optimiza para minimizar una función de pérdida, que mide la diferencia entre las predicciones del modelo y las etiquetas reales en un conjunto de datos de entrenamiento. Esta optimización ajusta los parámetros del modelo para mejorar su capacidad para generar texto coherente y relevante.

Capacidad del Modelo: GPT-2 tiene una gran cantidad de parámetros (millones o incluso miles de millones), lo que le permite capturar una amplia gama de patrones y estructuras en el lenguaje natural, desde palabras simples hasta estructuras gramaticales complejas.

3.3.2.7 Detalles del Modelo GPT-2 XL:

Descripción del modelo: GPT-2 XL es la versión de parámetros 1.61B de GPT-2, un modelo de lenguaje basado en transformers creado y lanzado por OpenAI. El modelo

es un modelo previamente entrenado en el idioma inglés que utiliza un objetivo de modelado de lenguaje causal (CLM). (Hugging Face, 2024)

Desarrollado por: OpenAI.

Tipo de modelo: modelo de lenguaje basado en transformers

Idioma(s): inglés

Licencia: Licencia MIT modificada

Modelos relacionados: GPT-2, GPT-Medium y GPT-Large

Recursos para más información:

- Trabajo de investigación
- Publicación de blog de OpenAI
- Repositorio de GitHub
- Tarjeta modelo OpenAI para GPT-2
- Publicación de blog sobre el lanzamiento de OpenAI
- GPT-2 1.61B

- **Uso:**

Uso directo: Los principales usuarios previstos de estos modelos son investigadores y profesionales de la IA. Principalmente imaginamos que los investigadores utilizarán estos modelos de lenguaje para comprender mejor los comportamientos, capacidades, sesgos y limitaciones de los modelos de lenguaje generativo a gran escala. (Hugging Face, 2024)

Uso posterior:

A continuación, se muestran algunos casos de uso secundarios que creemos que son probables:

- Asistencia de escritura: asistencia gramatical, autocompletado (para prosa o código normal)

- Escritura creativa y arte: explorar la generación de textos creativos y ficticios; ayudando a la creación de poesía y otras artes literarias.
- Entretenimiento: Creación de juegos, chat bots y generaciones divertidas.

Mal uso y uso fuera de alcance: Debido a que los modelos de lenguaje a gran escala como GPT-2 no distinguen la realidad de la ficción, no admitimos casos de uso que requieran que el texto generado sea verdadero. Además, los modelos de lenguaje como GPT-2 reflejan los sesgos inherentes a los sistemas en los que fueron entrenados, por lo que no recomendamos que se implementen en sistemas que interactúan con humanos a menos que quienes los implementen primero realicen un estudio de los sesgos relevantes para el uso previsto. (Hugging Face, 2024)

No encontramos diferencias estadísticamente significativas en las sondas de sesgo de género, raza y religión entre 774M y 1.61B, lo que implica que todas las versiones de GPT-2 deben abordarse con niveles similares de precaución en casos de uso que son sensibles a sesgos en torno a atributos humanos. (Hugging Face, 2024)

- **Riesgos, limitaciones y sesgos:**

Sesgos: Importantes investigaciones han explorado problemas de sesgo y equidad con los modelos lingüísticos.

Los datos de entrenamiento utilizados para este modelo no se han publicado como un conjunto de datos que se pueda explorar. Sabemos que contiene una gran cantidad de contenido de Internet sin filtrar, que está lejos de ser neutral. Las predicciones generadas por el modelo pueden incluir estereotipos perturbadores y dañinos entre las clases protegidas; características de identidad; y grupos sensibles, sociales y ocupacionales. (Hugging Face, 2024)

Dado que en nuestro caso en particular no ocuparemos ni generaremos ningún tipo de descripción humana estos sesgos no aplican para la generación de código CSS.

- **Riesgos y limitaciones:** Cuando se lanzó el modelo este contaba con 1,5 mil millones de parámetros, GPT-2 se puede ajustar para evitar un mal uso. Desde el Centro sobre Terrorismo, Extremismo y Contraterrorismo (CTEC) del Instituto Middlebury de

Estudios Internacionales descubrieron que los grupos extremistas pueden usar GPT-2 para uso indebido, específicamente ajustando los modelos GPT-2 en cuatro posiciones ideológicas: supremacía blanca, Marxismo, islamismo yihadista y anarquismo. El CTEC demostró que es posible crear modelos que puedan generar propaganda sintética para estas ideologías. También muestran que, a pesar de tener una baja precisión de detección en resultados sintéticos, los métodos de detección basados en ML pueden dar a los expertos sospechas razonables de que un actor está generando texto sintético. (Hugging Face, 2024)

- **Datos de Entrenamiento:** El equipo de OpenAI quería entrenar este modelo en un corpus lo más grande posible. Para construirlo, eliminaron todas las páginas web de los enlaces salientes en Reddit que recibieron al menos 3 karma. Tenga en cuenta que todas las páginas de Wikipedia se eliminaron de este conjunto de datos, por lo que el modelo no se entrenó en ninguna parte de Wikipedia. El conjunto de datos resultante (llamado WebText) pesa 40 GB de textos, pero no se ha hecho público. Puede encontrar una lista de los 1000 dominios principales presentes en WebText. (Hugging Face, 2024)
- **Procedimiento de entrenamiento:** El modelo fue entrenado en un corpus muy grande de datos en inglés de forma auto supervisada. Esto significa que fue entrenado previamente solo con los textos sin procesar, sin que ningún ser humano los etiquete de ninguna manera (razón por la cual puede usar una gran cantidad de datos disponibles públicamente) con un proceso automático para generar entradas y etiquetas a partir de esos textos.

Más precisamente, fue entrenado para adivinar la siguiente palabra en oraciones. Más precisamente, las entradas son secuencias de texto continuo de cierta longitud y los objetivos son la misma secuencia, desplazada una ficha (palabra o fragmento de palabra) hacia la derecha. (Hugging Face, 2024)

El modelo utiliza internamente un mecanismo de máscara para garantizar que las predicciones para el token “*i*” solo utilicen las entradas de los tokens “*I*” futuros, “*i*” pero no los futuros.

De esta manera, el modelo aprende una representación interna del idioma inglés que luego puede usarse para extraer características útiles para tareas posteriores. Los textos

se tokenizan utilizando una versión a nivel de bytes de Byte Pair Encoding (BPE) (para caracteres Unicode) y un tamaño de vocabulario de 50.257. Las entradas son secuencias de 1024 tokens consecutivos. (Hugging Face, 2024)

- **Datos de prueba, factores y métricas:** Dado que el modelo opera a nivel de bytes y no requiere preprocessamiento ni tokenización con pérdidas, podemos evaluarlo en cualquier punto de referencia de modelo de lenguaje.

Los resultados de los conjuntos de datos de modelado del lenguaje comúnmente se informan en una cantidad que es una versión escalada o exponencial de la probabilidad logarítmica negativa promedio por unidad de predicción canónica (generalmente un carácter, un byte o una palabra).

Evaluamos la misma cantidad calculando la probabilidad logarítmica de un conjunto de datos según un LM de WebText y dividiéndola por el número de unidades canónicas. Para muchos de estos conjuntos de datos, los LM de WebText se probarían significativamente fuera de distribución, teniendo que predecir texto agresivamente estandarizado, artefactos de tokenización como puntuación y contracciones desconectadas, oraciones mezcladas e incluso la cadena, que es extremadamente rara en WebText, que ocurre solo 26 veces en 40 mil millones de bytes.

De esta manera se muestra los resultados principales... utilizando destokenizadores reversibles que eliminan la mayor cantidad posible de estos artefactos de tokenización/preprocesamiento. Dado que estos destokenizadores son invertibles, aún podemos calcular la probabilidad logarítmica de un conjunto de datos y podemos considerarlos como una forma simple de adaptación de dominio. (Hugging Face, 2024)

- **Resultados:**

Rendimiento del modelo GPT-2 XL en tareas de NLP

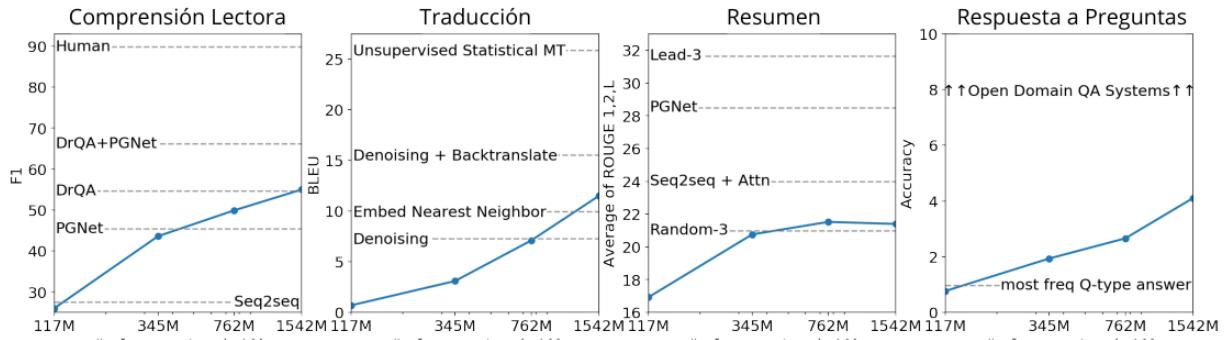


Figura 33. Rendimiento del Modelo GPT-2 XL en Tareas NLP de Comprension Lectora, Traducion, Resumen y Respuesta a Preguntas.

Fuente: (Language Models are Unsupervised Multitask Learners, 2019)

El modelo logra los siguientes resultados sin ningún fine-tuning.

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	IBW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	21.8
117M	35.13	45.99	87.65	83.4	29.41	65.85	1.16	1.17	37.50	75.20
345M	15.60	55.48	92.35	87.1	22.76	47.33	1.01	1.06	26.37	55.72
762M	10.87	60.12	93.45	88.0	19.93	40.31	0.97	1.02	22.05	44.575
1542M	8.63	63.24	93.30	89.05	18.34	35.76	0.93	0.98	17.48	42.16

Figura 34. Resultados del Modelo GPT-2 XL

Fuente: (Language Models are Unsupervised Multitask Learners, 2019)

3.3.3 Pytorch

PyTorch es una biblioteca de aprendizaje automático de código abierto desarrollada principalmente por el equipo de investigación de inteligencia artificial de Facebook (Facebook AI Research, FAIR). Ofrece herramientas y librerías para construir y entrenar modelos de aprendizaje profundo de manera eficiente. (Bourke, 2024)

Características Principales de PyTorch:

1. Tensores Dinámicos:

PyTorch utiliza tensores dinámicos, lo que significa que puedes modificar los tensores (como matrices multidimensionales) en tiempo de ejecución, similar a cómo lo harías con los arrays

de NumPy. Esto proporciona flexibilidad y facilidad para construir y depurar modelos. (Bourke, 2024)

2. Autograd:

PyTorch incluye una biblioteca de diferenciación automática (Autograd) que permite calcular automáticamente gradientes para optimizar modelos durante el entrenamiento. Esto simplifica la implementación de algoritmos de aprendizaje automático y aprendizaje profundo. (Bourke, 2024)

3. APIs de Alto Nivel:

PyTorch ofrece APIs de alto nivel que facilitan la construcción de modelos complejos utilizando componentes predefinidos y módulos como `nn.Module`, `nn.Sequential`, entre otros. (Bourke, 2024)

4. Integración con Hardware Acelerado:

PyTorch proporciona integraciones con hardware acelerado como GPUs y TPUs para acelerar el entrenamiento y la inferencia de modelos, lo que permite procesar grandes cantidades de datos de manera más rápida. (Bourke, 2024)

5. Comunidad Activa y Ecosistema:

PyTorch cuenta con una comunidad activa de desarrolladores y una amplia gama de bibliotecas y herramientas adicionales que facilitan tareas como la visualización, el procesamiento de datos y la implementación de modelos avanzados.

Aplicaciones de PyTorch:

Investigación en Aprendizaje Automático y Aprendizaje Profundo:

PyTorch es ampliamente utilizado en la investigación académica y la industria para desarrollar y experimentar con nuevos algoritmos y modelos de aprendizaje automático y aprendizaje profundo. (pytorch.org, 2024)

Desarrollo de Aplicaciones de Inteligencia Artificial:

PyTorch se utiliza para desarrollar aplicaciones y sistemas de inteligencia artificial en una amplia variedad de dominios, como visión por computadora, procesamiento de lenguaje natural, reconocimiento de voz, entre otros. (pytorch.org, 2024)

Educación y Enseñanza:

PyTorch es una herramienta popular en la educación y la enseñanza de aprendizaje automático y aprendizaje profundo debido a su facilidad de uso, su flexibilidad y su amplio soporte educativo. (pytorch.org, 2024)

3.3.4 NextJs para el desarrollo web

Next.js es un marco de trabajo de desarrollo web de código abierto basado en React que se utiliza para crear aplicaciones web modernas y escalables. Tiene varias funcionalidades y beneficios que incluyen:

- Renderización del lado del servidor (SSR) y generación de sitios estáticos (SSG): Next.js permite renderizar las páginas en el servidor antes de enviarlas al navegador del usuario, lo que mejora el rendimiento y la SEO. También admite la generación de sitios estáticos, lo que significa que puede generar HTML estático para sus páginas web en tiempo de compilación, lo que ofrece un mejor rendimiento y seguridad. (nextjs.org, 2024)
- Enrutamiento basado en archivos: Next.js proporciona un sistema de enrutamiento basado en la estructura de archivos del proyecto, lo que facilita la navegación entre páginas. (nextjs.org, 2024)
- Pre-renderización bajo demanda (SSG dinámico): Además de la generación de sitios estáticos en tiempo de compilación, Next.js también admite la pre-renderización bajo demanda, lo que significa que puede generar páginas estáticas en el servidor en tiempo de ejecución según la solicitud del usuario. (nextjs.org, 2024)

- (nextjs.org, 2024) API incorporado: Next.js proporciona una API incorporada que permite crear API RESTful o GraphQL dentro de la misma aplicación Next.js.
- Recarga en caliente (Hot Module Replacement): Next.js admite la recarga en caliente, lo que significa que los cambios en el código se reflejan automáticamente en el navegador sin necesidad de recargar la página. (nextjs.org, 2024)
- Optimización de imágenes: Next.js ofrece optimización automática de imágenes, lo que ayuda a mejorar el rendimiento de la aplicación al reducir el tamaño de las imágenes sin comprometer la calidad. (nextjs.org, 2024)

3.3.5 Paradigma de Programación

La programación orientada a componentes (Component-Oriented Programming, COP) es un paradigma de programación que se enfoca en la construcción de sistemas de software a través de la composición de componentes reutilizables. En lugar de escribir grandes bloques de código desde cero, los programadores crean sistemas utilizando componentes predefinidos que encapsulan funcionalidades específicas. (ecured.cu, 2022)

Los componentes en la programación orientada a componentes son unidades independientes de software que pueden ser ensambladas para formar aplicaciones más grandes. Estos componentes pueden ser tanto simples como complejos y pueden incluir desde simples funciones hasta módulos completos con interfaces definidas. (ecured.cu, 2022)

Algunas características importantes de la programación orientada a componentes incluyen:

- Reutilización: Los componentes son diseñados para ser reutilizables en diferentes contextos, lo que puede reducir el tiempo y esfuerzo requerido para desarrollar nuevas aplicaciones.
- Modularidad: Los componentes son módulos independientes que pueden ser desarrollados, probados y mantenidos de forma separada, lo que facilita la administración del código y la colaboración entre equipos de desarrollo.

- Interoperabilidad: Los componentes pueden ser interoperables, lo que significa que pueden comunicarse y cooperar entre sí a través de interfaces bien definidas, lo que permite la integración de diferentes tecnologías y sistemas.
- Encapsulamiento: Los componentes encapsulan su funcionalidad y ocultan los detalles de implementación interna, lo que promueve la modularidad y reduce la complejidad del sistema.

Algunos ejemplos de tecnologías y lenguajes de programación que admiten la programación orientada a componentes incluyen Java (con su tecnología de JavaBeans), .NET Framework (con Windows Forms y WPF), así como tecnologías de frontend como React.js y Angular.js.

3.3.6 Colab

Es un servicio gratuito de nube alojado por Google para fomentar la investigación sobre Aprendizaje de Máquina e inteligencia Artificial y sirve como herramienta de creación de notebooks basado en celdas que permite el uso de Markdown y Python, algunas de las principales características del Colab son: (Google, 2024)

- Dado que se ejecuta en una máquina de Google, no es necesario realizar ninguna configuración.
- Google proporciona acceso gratuito a las GPU (Lo que es útil para el Entrenamiento de modelos de IA).
- Es fácil de compartir, como cualquier archivo en el drive. (Google, 2024)

3.3.7 Visual Studio Code

Visual Studio Code es un potente editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Y que posee una amplia cantidad de plugins, que permite extender su funcionalidad. El editor incluye (VS Code, 2023):

- Incluye soporte para depuración
- Control de Git integrado
- Resaltado de sintaxis

- Finalización de código inteligente.
- Fragmentos de código.
- Refactorización de código.
- Personalización.
- Atajos

3.3.8 MongoDB

MongoDB es un sistema de gestión de bases de datos (DBMS) NoSQL, es decir, una base de datos no relacional. Se caracteriza por su capacidad para almacenar y manejar grandes volúmenes de datos de manera eficiente y escalable. Algunas características clave de MongoDB son:

- Estructura de datos flexible: A diferencia de las bases de datos relacionales, MongoDB no requiere un esquema fijo para los datos. Esto significa que puedes almacenar documentos en MongoDB sin tener que definir previamente la estructura de los datos.
- Documentos JSON-like: MongoDB almacena los datos en documentos BSON (Binary JSON), que son estructuras de datos similares a JSON (JavaScript Object Notation). Estos documentos pueden contener campos de diferentes tipos de datos y anidarse de manera flexible.
- Escalabilidad horizontal: MongoDB es altamente escalable y distribuible. Puede distribuir los datos a través de múltiples servidores y escalarse horizontalmente para manejar grandes volúmenes de tráfico y almacenamiento.
- Consultas complejas: MongoDB admite consultas complejas utilizando su lenguaje de consulta basado en JSON llamado Query Language. También proporciona índices para mejorar el rendimiento de las consultas.
- Alta disponibilidad y tolerancia a fallos: MongoDB proporciona características integradas para garantizar la alta disponibilidad de los datos y la tolerancia a fallos, como la replicación automática y la fragmentación (sharding). (MongoDB, 2023)

3.3.9 Postman

Postman es una herramienta de colaboración para el desarrollo de API que facilita la creación, prueba, documentación y uso compartido de API. Algunas de sus principales características son:

- Interfaz de usuario amigable: Postman ofrece una interfaz de usuario intuitiva que permite a los desarrolladores interactuar con API de manera eficiente.
 - Creación y prueba de solicitudes HTTP: Los usuarios pueden crear y enviar diferentes tipos de solicitudes HTTP, como GET, POST, PUT, DELETE, etc., para probar el funcionamiento de sus API.
 - Organización de solicitudes en colecciones: Postman permite organizar las solicitudes en colecciones, lo que facilita la gestión y la ejecución de grupos de solicitudes relacionadas.
 - Entornos y variables: Los desarrolladores pueden definir entornos y variables para personalizar y parametrizar las solicitudes, lo que facilita la configuración de diferentes entornos, como desarrollo, prueba y producción.
 - Automatización de pruebas: Postman permite crear y ejecutar pruebas automatizadas para verificar el comportamiento de las API y asegurar la calidad del software.
 - Generación de documentación: Postman puede generar documentación automáticamente a partir de las solicitudes y respuestas de la API, lo que facilita la comprensión y el uso de la API por parte de otros desarrolladores.
 - Colaboración y uso compartido: Los usuarios pueden colaborar en equipos y compartir colecciones, entornos y pruebas con otros miembros del equipo o con la comunidad.
- (Postman, 2024)

CAPÍTULO IV

METODOLOGÍA APLICADA AL PROYECTO

4.1. Metodología de la Investigación

4.1.1 Metodología General

Para la realización del proyecto se emplearán métodos empíricos para la obtención y recolección de información, que permitan determinar las opiniones y necesidades de los desarrolladores de software. Posteriormente, mediante la aplicación del método Inductivo, en la etapa de análisis en el desarrollo del proyecto, se conseguirá estructurar y modelar los diferentes comportamientos existentes en el ámbito estudiado, al paradigma orientado a componentes. Y por ende poder construir un modelo de inteligencia artificial que cumpla con los requerimientos. Finalmente, mediante el método deductivo, se podrá inferir el comportamiento durante la etapa de diseño en el desarrollo del sistema, de acuerdo a lo que se ha definido previamente en otras etapas.

4.1.2 Métodos, Técnicas y Herramientas

Los diferentes métodos, técnicas y herramientas que se van a emplear en el desarrollo del proyecto que van a permitir obtener los requerimientos deseados para el diseño óptimo del modelo de inteligencia artificial son los siguientes:

Observación

Mediante este método se recopilará la información de las características más necesarias y que se utilizan con frecuencia en diferentes plataformas de estilos CSS, y que van a permitir definir los lineamientos a seguir durante el desarrollo de la herramienta.

Técnica de análisis y procesamiento

Se empleará la técnica de análisis y procesamiento para la obtención de información necesaria para el entrenamiento de un modelo generador de código CSS, el cual consistirá en un set de

entrenamiento y otro de prueba que se procesará con todas sus propiedades y características requeridas para su correcto funcionamiento.

Método de entrenamiento

Se empleará el método de entrenamiento para el modelo de inteligencia artificial de forma iterativa donde se ajustará y probará el modelo con un set de datos de prueba y otro de entrenamiento hasta lograr lo resultado deseados.

4.2. Metodología de Ingeniería

Para el desarrollo del proyecto se planea seguir la metodología de desarrollo XP de ingeniería de software con un paradigma de programación orientada a componentes.

El presente proyecto seguirá el paradigma de programación orientada a componentes, empleando la metodología ágil, Extreme Programming (XP), el cuál posee un ciclo de vida iterativo e incremental. Así mismo el desarrollo del sistema se realizará en varias plataformas, una aplicación web enfocada en el uso del modelo IA y un modelo generador de código CSS que serán utilizados por los desarrolladores de software.

La aplicación web, se encontrará desarrollada empleando la librería “ReactJs”, utilizando Javascript y CSS y la cual estará conectada a un modelo de Inteligencia Artificial que será desarrollado con la librería “Pytorch” utilizando Python y JavaScript, por lo tanto, se puede implementar el modelo basado en componentes para la creación de la interfaz de usuario. Del mismo modo, con Java y Spring se desarrollará un servicio web API Rest, que permita la conexión del lado del servidor y exponga mi modelo IA con la aplicación web desarrollada anteriormente. Finalmente, exponiendo mi modelo IA con una API se tendrá acceso al modelo para ser consumido por los desarrolladores para su implementación en proyectos open source.

4.2.1 Desarrollo del Proyecto

Para el desarrollo del proyecto se plantea seguir la metodología de desarrollo XP, la cual permite obtener entregables completos de cada módulo o subsistema que conforma el producto final. Por lo tanto, para la realización del presente proyecto se ha determinado realizar 8 iteraciones, detalladas en la Tabla 1:

Nº de Iteracion	FASE	PRODUCTO
1	EXPLORACION	Analisis Preliminar del Proyecto
2	PLANIFICACION	Definicion de la Arquitectura del Sistema.
3	ITERACIONES	Dataset de Prueba y Entrenamiento
4		Modelo de Inteligencia Artificial
5		API de servicios web para Modelo
6		Sistema Generador de CSS
7	PRUEBAS	Ajuste del Modelo Mantenimiento y Pruebas
8	REVISION	Revision Final del Modelo

Tabla 1. Desarrollo del Proyecto

Fuente: Elaboración Propia

4.2.2 Planificación del Proyecto

Como Planificación se presenta el siguiente cronograma y se estima que el tiempo requerido para el desarrollo del proyecto será de 20 semanas, también se utilizará el Diagrama de Gantt para organizar y ejecutar el cronograma de manera eficiente a continuación mostrado.

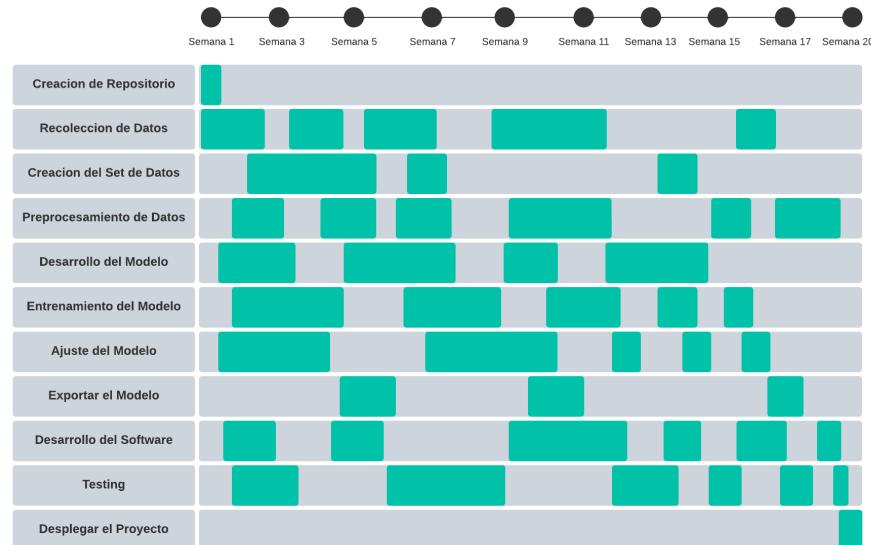


Figura 35. Diagrama de Gantt

Fuente: Elaboración Propia

CAPÍTULO V

INGENIERÍA DEL PROYECTO

5.1. Fase de Exploración: Análisis Preliminar del Proyecto

5.1.1 Proceso de Requerimientos

Para determinar los diferentes requerimientos para el presente proyecto, en primera instancia se realiza una investigación y análisis de los estilos y técnicas que en la actualidad utilizan los desarrolladores frontend con la tecnología CSS.

Del mismo modo, se pudo obtener muchos otros requerimientos mediante el proceso observación y revisión bibliográfica sobre plataformas y modelos LLM de inteligencia artificial para generación de texto y código. Por lo tanto, como primer paso, y para una mejor organización en lo referente al análisis y obtención de requerimiento es necesario identificar a los actores de los diferentes casos de uso para determinar los requerimientos funcionales y no funcionales.

5.1.1.1 Identificación de Actores

De acuerdo al estudio realizado en Internet en foros y pappers científicos, por medio de observación, revisión documental, revisión bibliográfica y estadísticas, se ha logrado identificar a los siguientes usuarios:

El Administrador, Desarrollador Web, Desarrollador Frontend

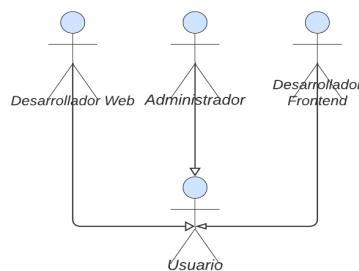


Figura 36. Diagrama de Actores
Fuente: Elaboración Propia

5.1.1.2 Descripción de Actores

Los actores identificados anteriormente, son aquellos que se encargan de acceder al sistema ya sea en su versión Web como su versión API, los roles identificados son: Administrador, Desarrollador Web, Desarrollador Frontend.

Nombre del Actor:	Administrador
Descripción:	Es aquel que tiene acceso total a la aplicación Web para poder realizar cualquier operación, añadir y quitar a otros usuarios y registros
Notas:	Este actor puede realizar las siguientes acciones en la aplicación Web: <ul style="list-style-type: none">▪ Registrar, Editar o Eliminar Usuarios.▪ Añadir nuevos administradores.▪ Revisar el funcionamiento correcto del Modelo▪ Administrar la API▪ Auditorias.

Tabla 2. Descripción del Administrador

Fuente: Elaboración Propia

Nombre del Actor:	Desarrollador Web
Descripción:	Es aquel que tiene acceso a la API para usar los servicios y también aplicación Web para generación de código Css
Notas:	Este actor puede realizar las siguientes acciones en la aplicación Web y en la API: <ul style="list-style-type: none">▪ Puede acceder a la API.▪ Puede acceder a la aplicación Web.▪ Puede enviar peticiones a la API directamente.▪ Puede recibir respuestas de la API directamente.▪ Puede generar código CSS con la aplicación Web.▪ Puede copiar el código CSS generado de la aplicación Web.

Tabla 3. Descripción de Desarrollador Web

Fuente: Elaboración Propia

Nombre del Actor:	Desarrollador Frontend
Descripción:	Es aquel que tiene acceso a la aplicación Web para generación de código Css
Notas:	<p>Este actor puede realizar las siguientes acciones en la aplicación Web:</p> <ul style="list-style-type: none"> ▪ Puede acceder a la aplicación Web. ▪ Puede enviar peticiones a la API desde la Aplicación Web. ▪ Puede recibir respuestas de la API desde la Aplicación Web. ▪ Puede generar código CSS con la aplicación Web. ▪ Puede copiar el código CSS generado de la aplicación Web.

Tabla 4. Descripción de Desarrollador Frontend
Fuente: Elaboración Propia

5.1.1.3 Requerimientos Funcionales

Los requerimientos funcionales del sistema se definen como todas aquellas funcionalidades del sistema deseadas por el usuario. Estos requerimientos permiten tener la idea clara de que es lo que debe hacer el software y como se lo debe desarrollar.

La obtención de requerimientos funcionales del sistema para el presente proyecto, está basada en la necesidad que tienen los Desarrolladores Frontend y Desarrolladores Web para poder mejorar su desempeño al desarrollar con la tecnología CSS. Esta información fue obtenida por medio de internet y del proceso de observación. Por lo tanto, se puede deducir que el presente sistema debe tener las siguientes características.

R.1. Validación de acceso al sistema.

R.1.1. Al sistema solo podrán acceder usuarios que hayan creado su cuenta (Administrador, Desarrolladores Frontend y Desarrolladores Web).

R.1.2. Para acceder al sistema debe ingresar cuenta de usuario y contraseña.

R.2. Registro de Información.

R.2.1. Permitir la introducción de la información personal de los Desarrolladores Frontend y Desarrolladores Web.

R.2.2. Permitir la escritura de código HTML en la plataforma.

R.3. Visualización de Información.

R.3.1. Permitir al Programador pueda ver el historial código CSS anteriormente generado.

R.3.2. Permitir al Programador pueda ver la vista generada en base a su código fuente HTML y CSS.

R.3.3. Permitir al Programador pueda ver diversas vistas generadas para que pueda seleccionar.

R.3.4. Permitir al Programador pueda ver su código fuente HTML.

R.4. Interacción Funcional con el Modelo

R.4.1. Permitir al Programador pueda comunicarse con el Modelo mediante el uso de la API.

R.4.2. Permitir al Programador puede enviar peticiones y recibir respuesta de la API.

R.4.3. Permitir al Programador pueda usar la API para aplicaciones de terceros mediante créditos limitados.

R.5. Interacción con el Modelo mediante la Interfaz de Usuario

R.5.1. Permitir al Programador ingresar su código fuente HTML para generar código CSS.

R.5.2. Permitir al Programador editar en tiempo real su código fuente introducido (HTML).

R.5.3. Permitir al Programador copiar el código CSS generado.

R.5.4. Permitir al Programador copiar las vistas generadas por la combinación de CSS y HTML.

R.5.5. Permitir al Programador copiar las vistas del historial de vistas anteriormente generadas por la combinación de CSS y HTML.

R.5.6. Permitir al Programador editar en tiempo real el código fuente generado (CSS).

5.1.1.4 Requerimientos no Funcionales

La principal finalidad de la elaboración de un modelo de generación de código CSS es de contar con un modelo versátil y accesible en cualquier momento y en cualquier lugar. Para que Desarrolladores Web y Desarrolladores Frontend puedan utilizar el modelo de inteligencia artificial en todo momento.

Para ello se plantea que el modelo posee las siguientes características, las cuales se constituyen como requerimientos no funcionales:

Fiabilidad

El modelo debe ser fiable ante fallos, como, por ejemplo, una falla en la generación de código con errores sintácticos.

Disponibilidad

El modelo debe estar disponible en cualquier momento para los usuarios

Funcionalidad

Asegurar que el producto funcione tal como se ha especificado en los requerimientos.

Rendimiento y Escalabilidad

El software cumple su funcionalidad a cabalidad y debe además poder ser ampliado en un futuro.

Usabilidad

Debe ser usable para todos usuarios.

Mantenibilidad

Capacidad del software que permite que una determinada modificación o inclusión de un nuevo módulo sea implementada.

5.2. Fase de Elaboración: Definición de la Arquitectura del Sistema

5.2.1 Diagrama de Casos de Uso General

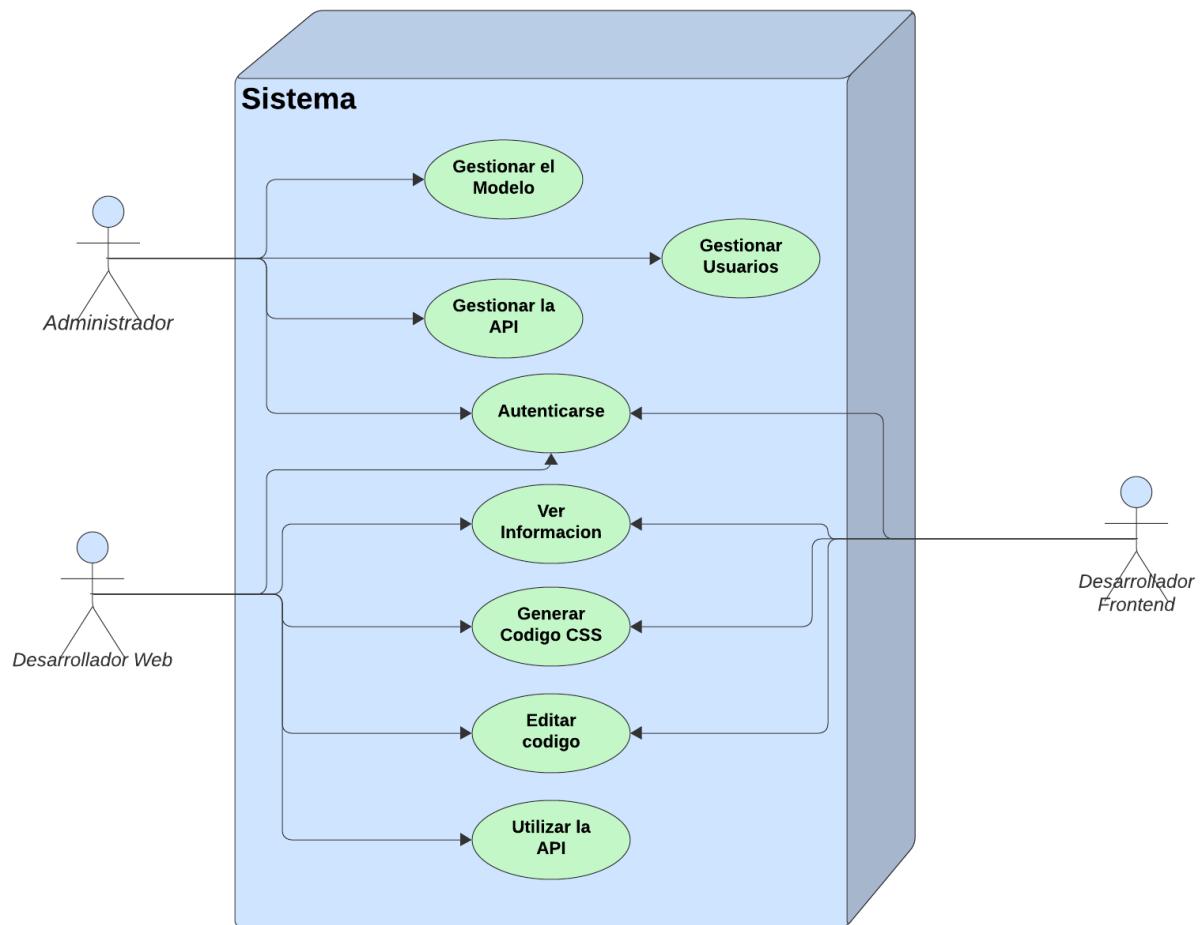


Figura 37. Diagrama General de Casos de Uso
Fuente: Elaboración Propia

5.2.2 Diagrama de Casos de Uso Clasificación por prioridad

- Diagrama de Casos de Uso del Usuario

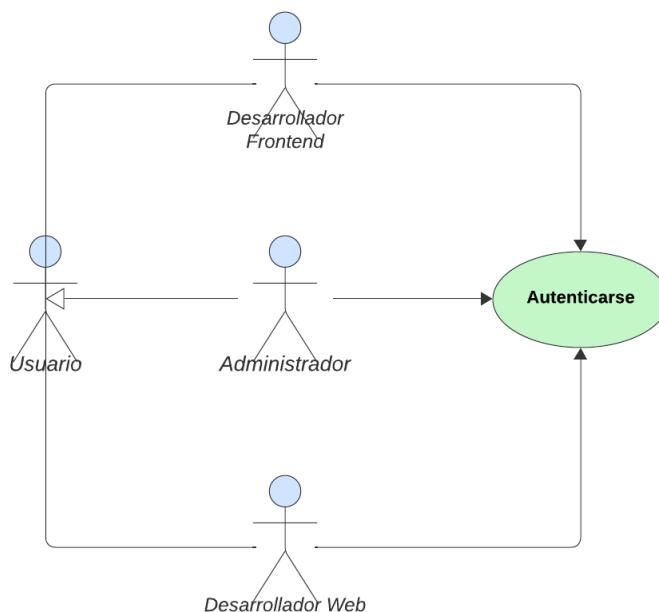


Figura 38. Diagrama de Casos de Uso del Usuario
Fuente: Elaboración Propia

- Diagrama de Casos de Uso del Administrador

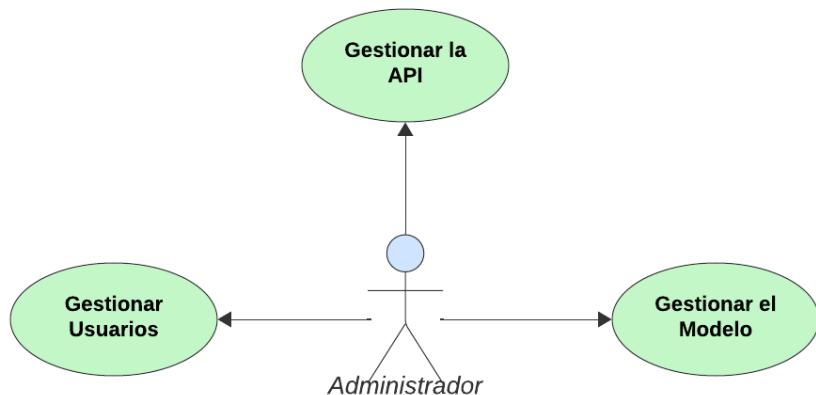


Figura 39. Diagrama de Casos de Uso del Administrador
Fuente: Elaboración Propia

5.2.3 Diagrama de Casos de Uso Clasificados por funcionalidad

- Diagrama de Casos de Uso de Generación de código CSS

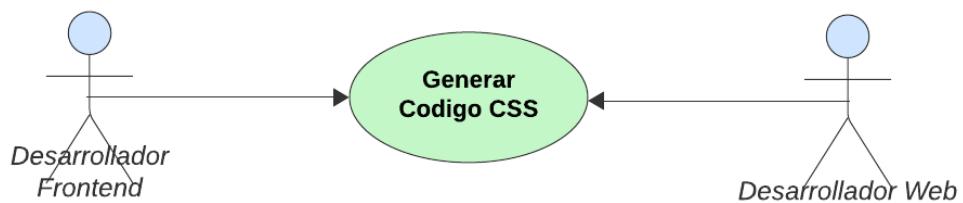


Figura 40. Diagrama de Casos de Uso de Generacion de codigo CSS

Fuente: Elaboración Propia

- Diagrama de Casos de Uso de Edición de Código en Tiempo Real



Figura 41. Diagrama de Casos de Uso de Edicion de Código en Tiempo Real

Fuente: Elaboración Propia

5.2.4 Descripción de los Casos de Uso

Caso de Uso 1:	Gestionar el Modelo
Actores:	Administrador
Tipo:	Primario
Descripción:	El caso de uso realiza la gestión, mantenimiento y pruebas del modelo de Inteligencia Artificial de Generación de código CSS, verificando el correcto funcionamiento del modelo y de la generación de código que realiza.
Caso de Uso 2:	Gestionar Usuario
Actores:	Administrador
Tipo:	Primario
Descripción:	Registro de los diferentes usuarios que tengan acceso al Sistema Web y también a la API que consulta al Modelo, gestión de créditos de los usuarios para el uso de la API.
Caso de Uso 3:	Gestionar API
Actores:	Administrador
Tipo:	Primario
Descripción:	Gestión, mantenimiento y actualización de nuevas funcionalidades de los servicios de la API que consume el modelo de Inteligencia Artificial de Generación de código CSS.
Caso de Uso 4:	Autenticación
Actores:	Administrador, Desarrollador Web, Desarrollador Frontend
Tipo:	Secundario
Descripción:	El caso de uso realiza la validación de usuarios. Se activa cuando un usuario intenta identificarse ante el sistema para poder acceder a las distintas funcionalidades.
Caso de Uso 5:	Ver Información
Actores:	Desarrollador Web, Desarrollador Frontend
Tipo:	Primario

Descripción:	El modelo permitirá a los Desarrolladores Web y Desarrolladores Frontend ver el código CSS generado y un historial de código generado con anterioridad, también ver la vista resultante de la implementación del código HTML y el código CSS generado.
Caso de Uso 6:	Generar Código CSS
Actores:	Desarrollador Web, Desarrollador Frontend
Tipo:	Primario
Descripción:	El modelo permitirá a los Desarrolladores Web y Desarrolladores Frontend generar código CSS en base a etiquetas HTML en tiempo real y copiar dicho código.
Caso de Uso 7:	Editar Código
Actores:	Desarrollador Web, Desarrollador Frontend
Tipo:	Primario
Descripción:	El Sistema Web permitirá a los Desarrolladores Web y Desarrolladores Frontend editar el código HTML y el código CSS generado en tiempo real para editar la vista resultante.
Caso de Uso 8:	Utilizar la API
Actores:	Desarrollador Web
Tipo:	Primario
Descripción:	El caso de uso permitirá a los Desarrolladores Web y otros perfiles de ingenieros de software utilizar los servicios de la API para consumir el Modelo, así como para utilizar la API en aplicaciones de terceros.

Tabla 5. Descripción de los Casos de Uso

Fuente: Elaboración Propia

5.2.5 Proceso de Análisis y Diseño

El proceso de análisis y diseño pretende obtener una descripción general y detallada del modelo, y tiene como objetivo principal satisfacer los requerimientos definidos por el usuario. Del mismo modo servirá de base para posibles nuevos diseños del modelo. Esta etapa se caracteriza por refinar los requerimientos, y del mismo modo considerar los riesgos que involucra el desarrollo del modelo.

5.2.6 Análisis de Riesgos

Se considera riesgo a todo lo que pueda afectar negativamente al proyecto, dentro de determinados límites. Será un riesgo todo lo que afecte a acontecimientos futuros que implique cambio o implique elección o incertidumbre. Entre los posibles riesgos encontrados y a los que se expone el presente proyecto, se encuentran:

Planificación equivocada.

- Riesgo: Alto.
- Descripción del Riesgo: La planificación del proyecto no cumple los objetivos o el alcance definido en los requerimientos.
- Impacto: Retraso y confusión durante el desarrollo del sistema.
- Indicadores: El proyecto no cumple con lo propuesto en los plazos acordados.
- Estrategia de Mitigación: Supervisar constantemente los objetivos y alcances del proyecto.
- Plan de Contingencia: Replantear el alcance y los objetivos del proyecto en base al tiempo disponible y con una planificación acorde.

Insuficiente conocimiento de algunas herramientas de desarrollo de Software.

- Riesgo: Medio.
- Descripción del Riesgo: Tener escaso dominio de algunas herramientas para el desarrollo de software.
- Impacto: Retraso en el avance de algunas tareas del proyecto.
- Indicadores: Falta de tiempo, retraso en las entregas e iteraciones.

- Estrategia de Mitigación: Destinar tiempo adicional para la profundización y el aprendizaje de las herramientas necesarias para el desarrollo del proyecto.
- Plan de Contingencia: Buscar herramientas alternativas y de migración rápida en la que el desarrollo pueda realizarse con la velocidad deseada.

Cambios repentinos de requerimientos.

- Riesgo: Alto.
- Descripción del Riesgo: Un cambio en requerimientos implica realizar nuevamente el análisis y diseño del requerimiento, por lo que significaría un retraso en los entregables.
- Impacto: Realizar nuevamente el análisis de los requerimientos, y posiblemente haber realizado trabajo que debe ser cambiado.
- Estrategia de Mitigación: Realizar un análisis profundo de los requerimientos que esté abierto a algunos cambios imprevistos.
- Plan de Contingencia: Establecer mecanismos de análisis de requerimientos periódicos que permitan identificar a tiempo un posible cambio de requerimiento.

Proyecto no completado en el plazo establecido

- Riesgo: Alto.
- Descripción del Riesgo: Por diversas causas es probable no completar las tareas previstas en el plazo establecido al inicio del proyecto.
- Impacto: El proyecto no fue concluido a tiempo.
- Indicadores: Existe un retraso en las tareas en cada iteración del proceso unificado.
- Estrategia de Mitigación: Hacer un mayor seguimiento a las tareas para tratar de cumplirlas según el cronograma definido.
- Plan de Contingencia: Reducir a la funcionalidad mínima para completar los objetivos y alcance de proyecto.

Escatimar en el control de calidad

- Riesgo: Alto.
- Descripción del Riesgo: El proyecto no cuenta con las suficientes pruebas unitarias o de sistemas que garanticen y validen la calidad del sistema.
- Impacto: El funcionamiento del sistema puede tener fallas.
- Indicadores: No se realiza pruebas al haber finalizado la implementación de algún subsistema.
- Estrategia de Mitigación: Se debe realizar pruebas de calidad del sistema durante todas las fases de desarrollo.
- Plan de Contingencia: Se debe realizar pruebas exhaustivas durante el desarrollo del sistema, para asegurar en mayor medida la calidad del mismo.

Entrenamiento insuficiente de Modelo

- Riesgo: Alto
- Descripción del Riesgo: El Modelo de inteligencia artificial no cuenta con diversos recursos para el correcto entrenamiento.
- Impacto: El funcionamiento del Modelo puede generar código incorrecto
- Indicadores: Un set de datos insuficiente, una incorrecta configuración de hiperparámetros y la nula implementación de una función de perdida para evaluar las predicciones del modelo.
- Estrategia de Mitigación: Se debe ajustar constantemente el entrenamiento del modelo de inteligencia artificial hasta cumplir con los requerimientos de generación de código.
- Plan de Contingencia: Se debe validar y evaluar el Modelo y la generación de código CSS en cada fase del entrenamiento cumpliendo estándares de calidad.

5.2.7 Diagramas de Frontera del Modelo

5.2.7.1 Diagrama Frontera del Administrador

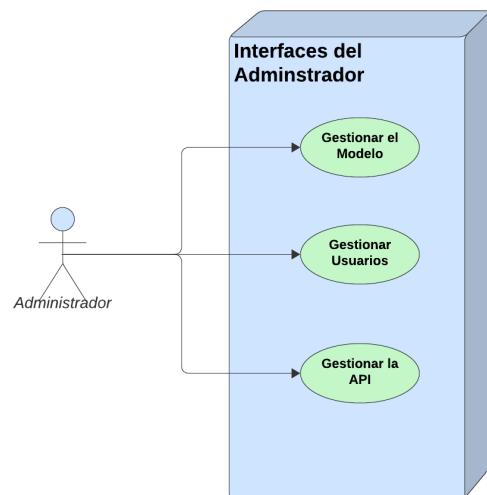


Figura 42. Diagrama Frontera del Administrador

Fuente: Elaboración Propia

5.2.7.2 Diagrama Frontera del Desarrollador Web

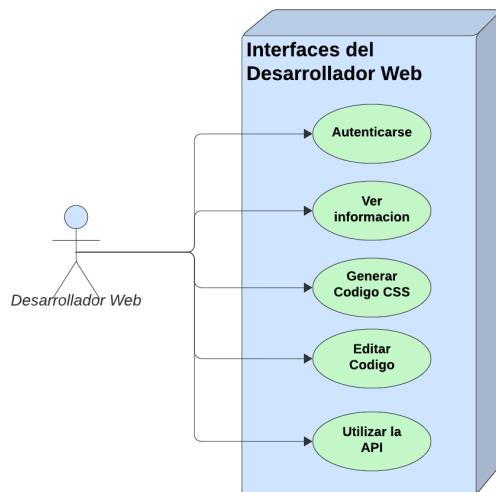


Figura 43. Diagrama Frontera del Desarrollador Web

Fuente: Elaboración Propia

5.2.7.3 Diagrama Frontera del Desarrollador Frontend

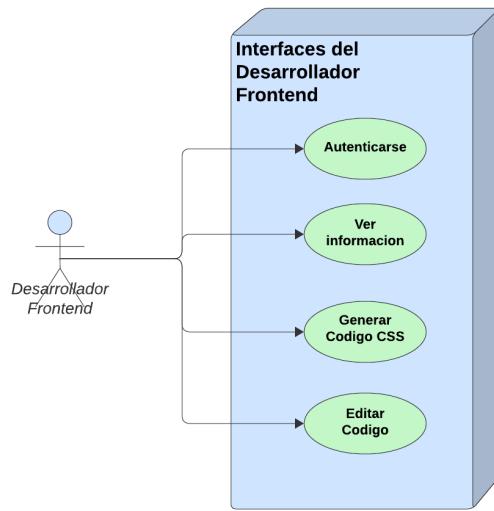


Figura 44. Diagrama Frontera del Desarrollador Frontend
Fuente: Elaboración Propia

5.2.8 Diagrama de Componentes del Sistema

Un diagrama de componentes sirve para representar la estructura interna de un sistema de software en términos de los componentes que lo componen y las relaciones entre ellos. Este tipo de diagrama es útil para entender la arquitectura de un sistema, mostrando cómo los diferentes componentes del software se organizan y trabajan juntos para lograr los objetivos del sistema.

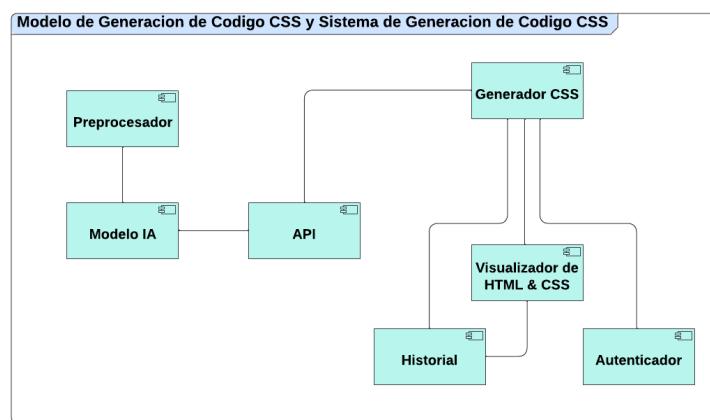


Figura 45. Diagrama de Componentes del Sistema
Fuente: Elaboración Propia

❖ Componente Modelo IA

Realiza la generación de Código CSS en base al entrenamiento por la interacción directa con el Componente de Preprocesador.

❖ Componente Preprocesador

Realiza el preprocesamiento de datos para el entrenamiento del Modelo IA.

❖ Componente API

Realiza la comunicación del Modelo IA con terceros e interactúa con Componente Generador CSS.

❖ Componente Generador CSS

Realiza y hace posible la interacción de los usuarios con el Modelo IA permitiéndole la generación de código CSS en base a etiquetas HTML, e interactúa con los componentes del Visualizador de HTML & CSS, el Componente de Historial y el Componente Autenticador.

❖ Componente Visualizador de HTML & CSS

Realiza la previsualización del código HTML & CSS, como la edición en tiempo real, también nos permite copiar el código y nos brinda la vista resultante del código CSS generado y el código HTML en tiempo real, además este componente interactúa con el componente del Historial.

❖ Componente del Historial

El componente nos permite ver todas las vistas generadas anteriormente, como resultado del código CSS generado por el modelo y el código HTML.

❖ Componente Autenticador

Realiza el proceso de acceso del usuario en el Sistema Generador de código CSS.

5.3. Fase de Desarrollo

5.3.1 Datasets de Entrenamiento, Validación y Prueba

Durante la fase inicial del desarrollo del Proyecto, comenzamos con la creación y gestión de un conjunto de datos que denominaremos como "Dataset" o "set de Datos". Para cumplir con este propósito, necesitamos desarrollar tres conjuntos de datos diferentes que nos servirán para diferentes propósitos: un dataset de entrenamiento, un dataset de validacion y un dataset de prueba.

5.3.1.1 Diagrama de Flujo

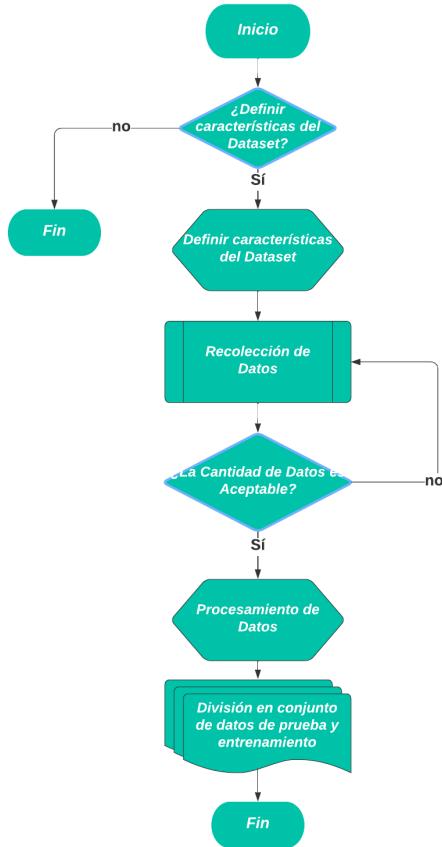


Figura 46. Diagrama de Flujo del Desarrollo del Dataset de Entrenamiento, Validacion, Prueba
Fuente: Elaboración Propria

5.3.1.2 Características del Dataset

Los datasets que se desarrollaran deben tener ciertas características específicas para garantizar la eficiencia del entrenamiento del modelo. A continuación, se muestra algunas de las características que se consideraron para la recolección de datos de nuestro dataset:

- Etiquetas HTML y sus atributos: El dataset debe inferir etiquetas HTML comunes junto con sus atributos asociados. Esto permitirá al modelo aprender la relación entre las etiquetas HTML y los estilos CSS correspondientes.
- Estilos CSS asociados: Cada etiqueta HTML en el dataset debe estar acompañada de los estilos CSS asociados. Estos estilos podrían incluir propiedades como color, tamaño de fuente, márgenes, rellenos, etc.
- Variabilidad y diversidad: Es importante que el dataset contenga una amplia variedad de etiquetas HTML con diferentes atributos y estilos CSS asociados. Esto ayudará al modelo a aprender patrones generales y a generalizar mejor en datos nuevos.
- Datos de entrada y salida emparejados: El dataset debe tener datos emparejados de entrada y salida. La entrada serían las etiquetas HTML y sus atributos, mientras que la salida serían los estilos CSS correspondientes.
- Datos etiquetados de forma adecuada: Cada ejemplo en el dataset debe estar correctamente etiquetado para que el modelo pueda aprender de manera supervisada. Esto significa que cada entrada debe estar asociada con la salida esperada.
- Datos de calidad: Es crucial que los datos en el dataset estén limpios y sean de alta calidad. Esto significa que no deben contener errores o anomalías que puedan afectar negativamente el entrenamiento del modelo.
- Tamaño del dataset adecuado: El dataset debe ser lo suficientemente grande como para proporcionar suficientes ejemplos de entrenamiento y validación. Un tamaño de dataset adecuado puede ayudar a prevenir el sobreajuste y mejorar el rendimiento del modelo.

Para realizar la correcta creación del Dataset se optó por tomar en cuenta la calidad y cantidad de los datos junto con la variabilidad y diversidad de los mismos para obtener un Dataset competente para el entrenamiento del modelo.

5.3.1.3 Recolección de Datos

El proceso de recolección de datos se a bordo de tal manera que se pueda recolectar una gran cantidad de datos, el Dataset precisa de código CSS principalmente como de código HTML.

Para esto se utilizó diversos generadores CSS automatizados que generan estilos estándar, bajo las reglas de la sintaxis del código CSS, también se utilizó varios repositorios de código abierto.

Algunos de los generadores que se utilizó son:

- **CSS Grid Generator**



Figura 47. CSS Grid Generator

Fuente: CSS Grid Generator

- **Ultimate CSS Gradient Generator**

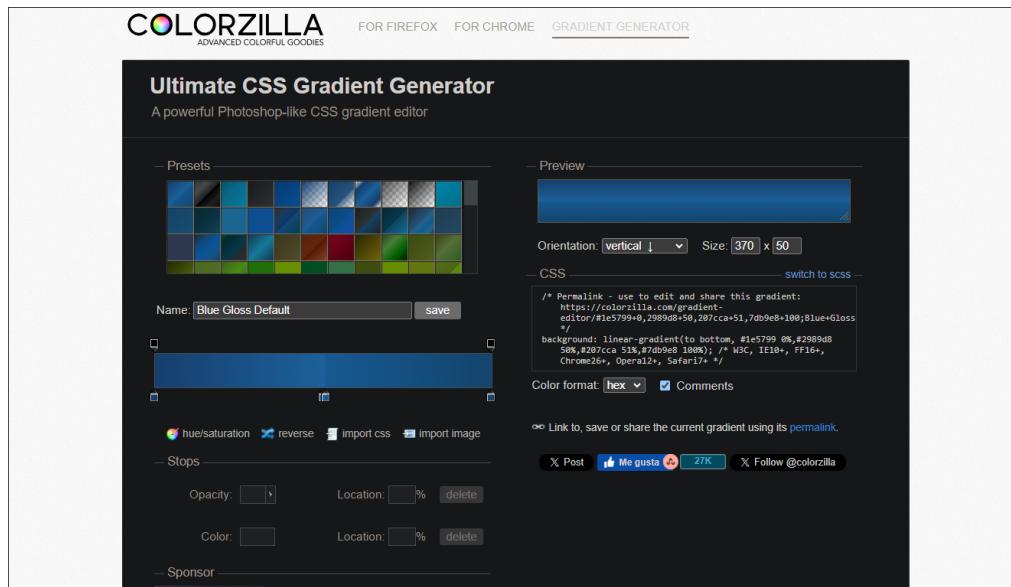


Figura 48. Ultimate CSS Gradient Generator
Fuente: Ultimate CSS Gradient Generator

- **CSS Code Generators**

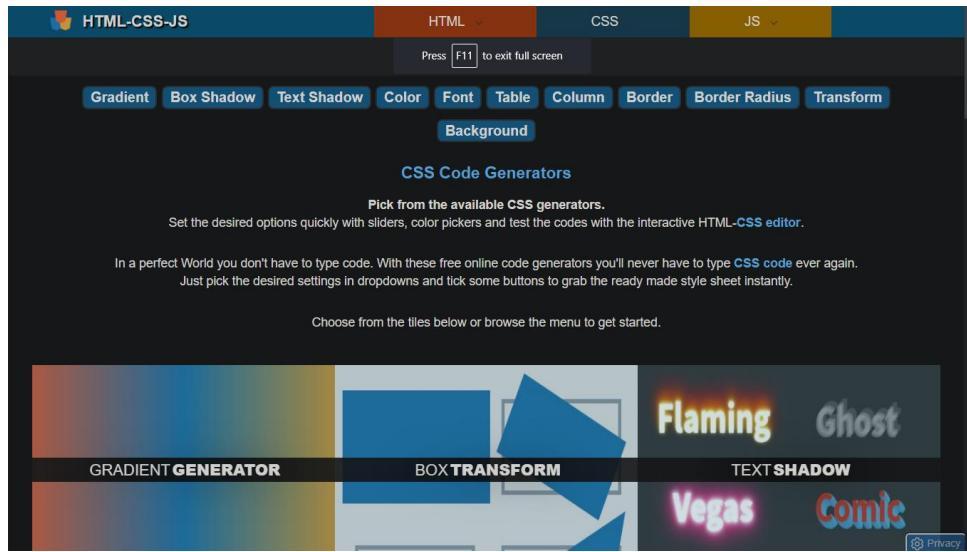


Figura 49. CSS CodeGenerators
Fuente: CSS Code Generators

- **The Ultimate CSS Generator**

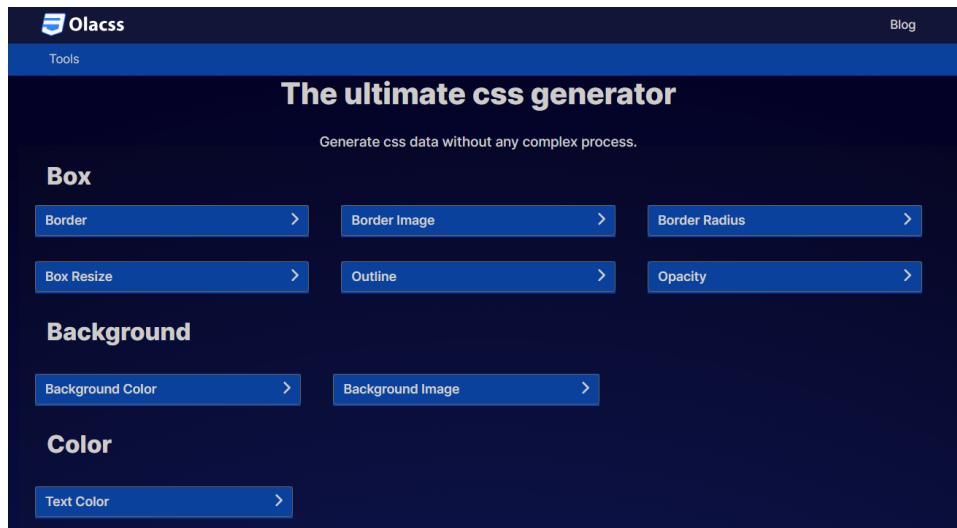


Figura 50. The Ultimate CSS Generator

Fuente: The Ultimate CSS Generator

- **EnjoyCSS**

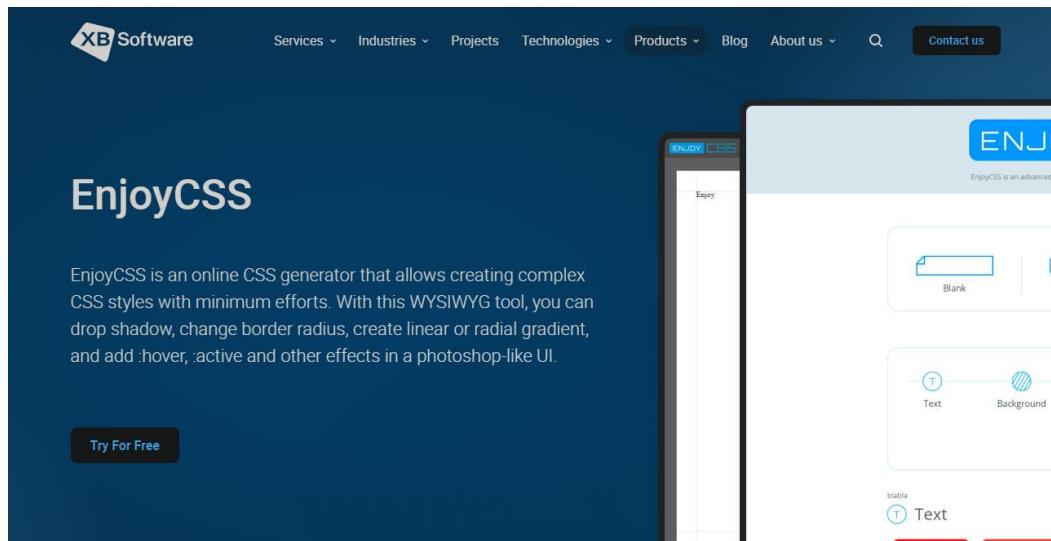
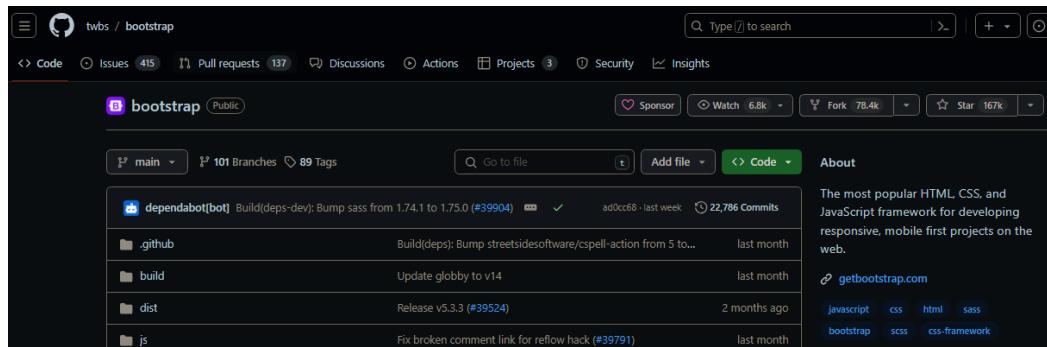


Figura 51. EnjoyCSS

Fuente: EnjoyCSS

Algunos repositorios de GitHub de referencia son:

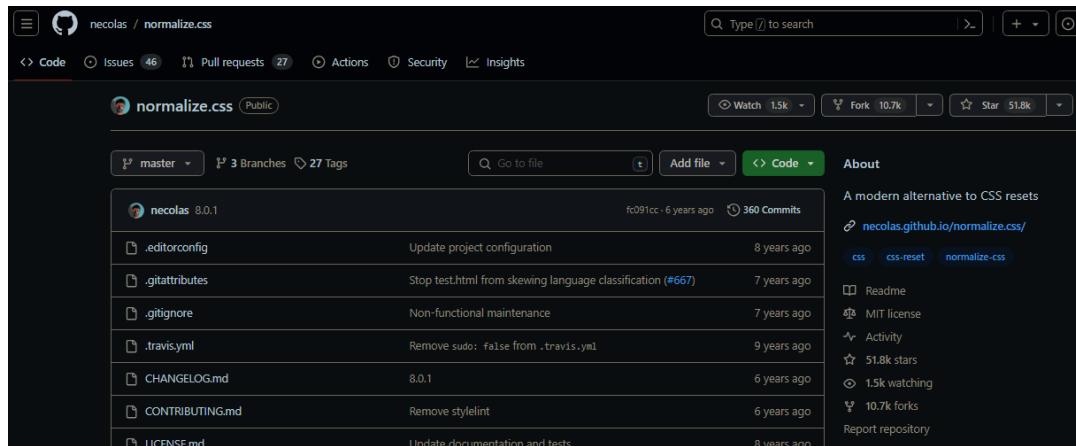
- Repositorio de Bootstrap: <https://github.com/twbs/bootstrap>



The screenshot shows the GitHub repository page for 'bootstrap' by 'twbs'. The repository has 101 branches and 89 tags. The 'About' section describes it as 'The most popular HTML, CSS, and JavaScript framework for developing responsive, mobile first projects on the web.' It includes links to 'getbootstrap.com' and categories like 'Javascript', 'css', 'html', 'sass', 'bootstrap', 'scss', and 'css-framework'. The commit history shows recent activity from 'dependabot[bot]' and other contributors.

Figura 52. Repositorio de Bootstrap
Fuente: Repositorio de Bootstrap

- Repositorio de Normalize.css: <https://github.com/necolas/normalize.css>



The screenshot shows the GitHub repository page for 'normalize.css' by 'necolas'. The repository has 3 branches and 27 tags. The 'About' section describes it as 'A modern alternative to CSS resets' and provides a link to 'necolas.github.io/normalize.css'. It includes categories like 'css', 'css-reset', and 'normalize-css'. The commit history shows contributions from 'necolas' and others over several years.

Figura 53. Repositorio de Normalize.css
Fuente: Repositorio de Normalize.css

- Repositorio de Materialize CSS: <https://github.com/Dogfalo/materialize>

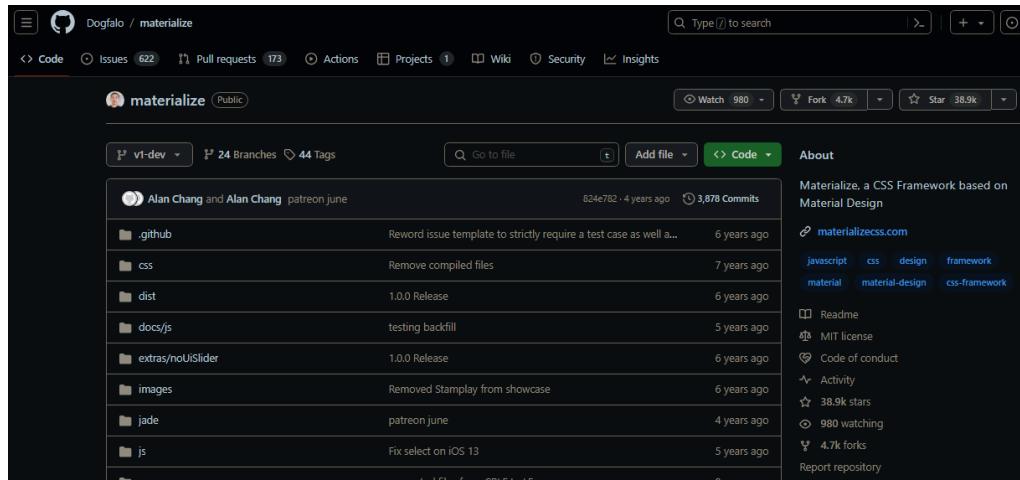


Figura 54. Repositorio de Materialize CSS

Fuente: Repositorio de Materialize CSS

- Repositorio de Awesome CSS: <https://github.com/awesome-css-group/awesome-css>

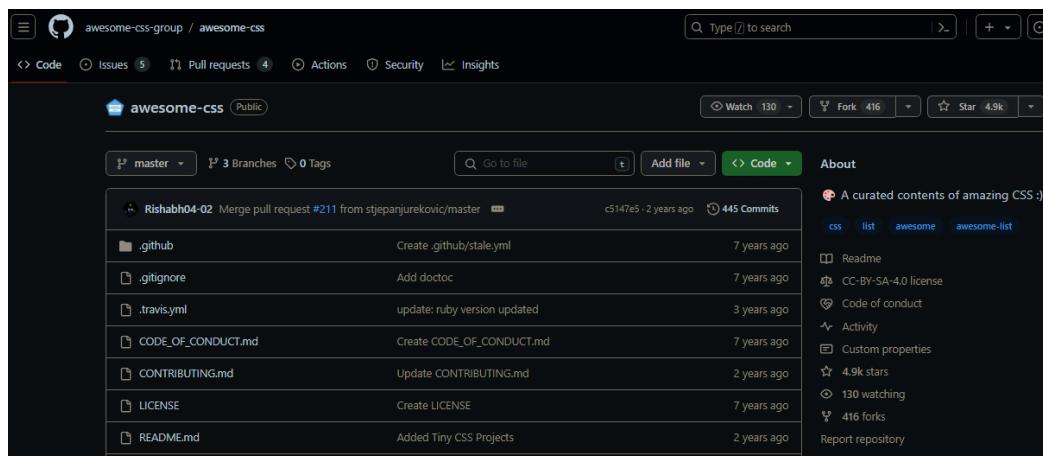


Figura 55. Repositorio de Awesome CSS

Fuente: Repositorio de Awesome CSS

5.3.1.4 Procesamiento de Datos

El procesamiento de datos es un proceso iterativo y que puede requerir ajustes a medida que se avanza en el análisis y se comprenden mejor los datos y las necesidades del proyecto.

Para nuestro caso específico para que el modelo pueda reconocer patrones optamos por etiquetar los datos en dos tipos diferentes que a su vez están relacionados mediante objetos en un archivo JSON en donde se implementó un array o lista llamada “styles” que contiene objetos que a su vez contienen dos etiquetas las cuales son “style” (para el estilo o código CSS) y “selector” (para definir el selector de la etiqueta HTML a la que se aplicara los estilos) ambos están relacionados, al proporcionar al modelo ejemplos de entrada (en el selector) con sus correspondientes salidas deseadas (en styles), el modelo entenderá su relación implícita a la hora de realizar el entrenamiento mediante el aprendizaje supervisado, por esta razón es que se utilizó la técnica de etiquetado para el procesamiento de los datos en donde nuestras etiquetas son “style” y “selector”.

La calidad y precisión de las etiquetas son críticas para el éxito de un modelo de aprendizaje automático, por lo que es importante prestar mucha atención al proceso de etiquetado y, en caso de etiquetación manual, proporcionar capacitación y directrices claras a los anotadores para asegurar la coherencia y la precisión de las etiquetas.

A continuación, se muestra como está organizado el dataset resultante listo para el entrenamiento del modelo en la siguiente figura:

```
[{"styles": [ { "style" : "body { overflow:hidden ; background:rgb(25,35,125) } ", "selector" : "body" }, { ... }, { ... }, { ... }, { ... }, { "style" : "div.drop-container { position:absolute ; top:0 ; right:0 ; bottom:0 ; left:0 ; margin:auto ; height:200px ; width:200px } ", "selector" : "div.drop-container" } ]}
```

Figura 56. Ejemplo de Dataset

Fuente: Elaboración Propia

5.3.1.5 División en conjunto de datos de entrenamiento, validación y prueba

Para el entrenamiento de nuestro Modelo Generador de Código CSS, es crucial estructurar adecuadamente el dataset. Para ello, dividiremos el conjunto de datos en tres archivos de tipo JSON:

- STYLES.json (Dataset de Entrenamiento - Train): Este archivo contendrá la mayoría de los datos que utilizaremos para entrenar nuestro modelo. Es fundamental que este conjunto de datos sea lo suficientemente grande y diverso para que el modelo aprenda patrones y características relevantes del código CSS. En este archivo, incluiremos ejemplos de estilos CSS con sus respectivas propiedades y etiquetas HTML.
- VALIDATION.json (Dataset de Validación): Este archivo se utilizará para ajustar y validar el rendimiento del modelo durante el entrenamiento. Extraeremos una pequeña porción del dataset original y la utilizaremos para evaluar cómo se está desempeñando el modelo en tareas específicas sin sobreajustar el entrenamiento. Es crucial que los datos en este archivo sean representativos pero distintos de los datos de entrenamiento para una validación efectiva.
- TEST.json (Dataset de Prueba): Este archivo servirá para evaluar el rendimiento final del modelo una vez que haya sido entrenado y validado. No debe haber datos duplicados con los archivos de entrenamiento y validación para garantizar una evaluación imparcial. Los resultados obtenidos con este conjunto de datos nos darán una visión clara de la capacidad del modelo para generalizar a nuevos datos no vistos previamente.

5.3.2 Modelo de Inteligencia Artificial

Esta es la fase más importante del proyecto, el desarrollo de un nuevo Modelo de Inteligencia Artificial Generador de código CSS sin ningún contexto más que etiquetas o código HTML. Esta fase de desarrollo del modelo consta de diferentes partes y técnicas a seguir, las cuales se explicarán a lo largo del apartado.

5.3.2.1 Diagrama de Flujo

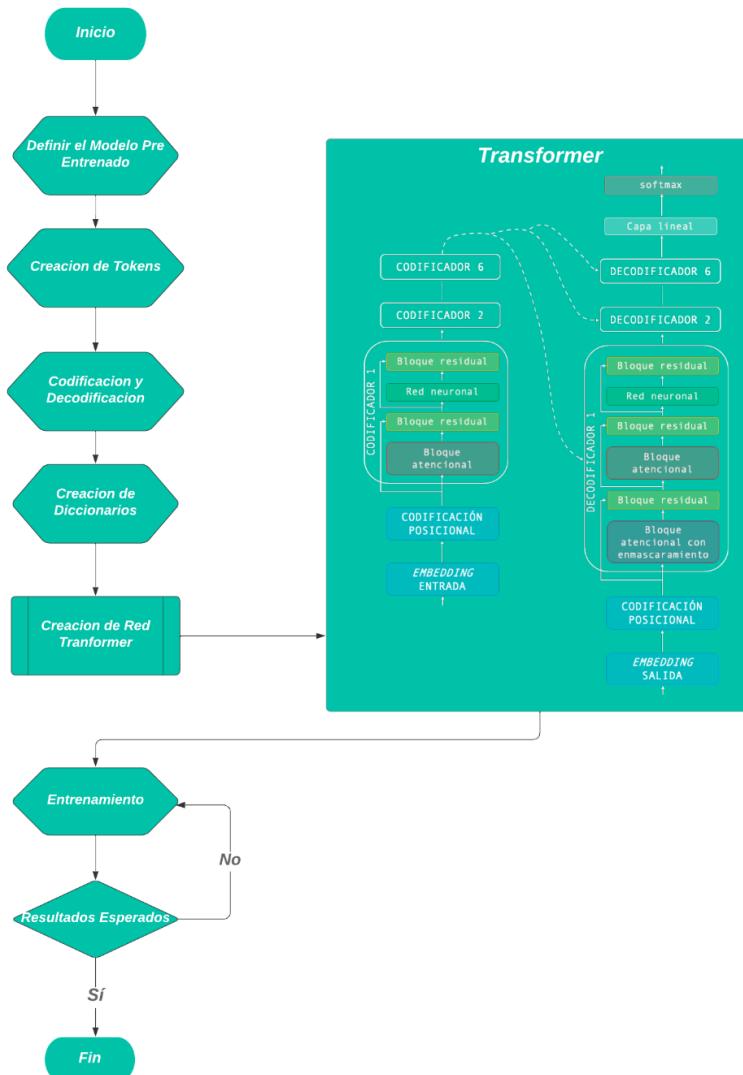


Figura 57. Diagrama de Flujo del Desarrollo del Modelo IA
Fuente: Elaboración Propia

5.3.2.2 Implementación del Modelo

La implementación del Modelo pre-entrenado GPT-2 XL se realizó utilizando la librería de Hugging Face Transformers una librería que proporciona APIs para descargar y entrenar modelos pre-entrenados de última generación. Los modelos que contiene la librería Transformers se pueden utilizar en diferentes modalidades, tales como:

- Texto: clasificación de texto, extracción de información, respuesta a preguntas, resumir, traducción y generación de texto en más de 100 idiomas.
- Imágenes: clasificación de imágenes, detección de objetos y segmentación.
- Audio: reconocimiento de voz y clasificación de audio.
- Multimodal: respuesta a preguntas en tablas, reconocimiento óptico de caracteres, extracción de información de documentos escaneados, clasificación de videos y respuesta visual a preguntas.

La biblioteca admite una integración entre tres de las bibliotecas de deep learning más populares: PyTorch, TensorFlow y JAX. Permite entrenar un modelo con tres líneas de código en un framework y cárgarlo para inferencia con otro.

Cada arquitectura de Transformers se define en un módulo de Python independiente para que se puedan personalizar fácilmente para investigación y experimentos.

Adicionalmente Hugging Face nos brinda una documentación para implementar el modelo pre-entrenado GPT-2 XL esta es una de las características importantes a la hora de seleccionar el modelo pre-entrenado, por su disponibilidad y documentación para la utilización del modelo.

Preparando el Dataset

Anteriormente se explicó de manera general cómo preparar el dataset para el entrenamiento de un modelo pre-entrenado. A continuación, profundizaremos en el proceso y la aplicación de las técnicas y el código necesario para el preprocesamiento del dataset. Posteriormente, implementaremos el modelo pre-entrenado y tendremos todo listo para aplicar la técnica de Fine-Tuning.

Entonces inicialmente importamos la función `load_dataset()` desde modulo `datasets` de la biblioteca `datasets` , es una función conveniente proporcionada por la biblioteca `datasets` que facilita la carga de conjuntos de datos comunes utilizados en el procesamiento del lenguaje natural (NLP) y la visión por computadora. Al utilizar `load_dataset()`, puedes cargar rápidamente conjuntos de datos populares sin tener que preocuparte por descargarlos, descomprimirlos o formatearlos manualmente, también es muy útil para la creación de un “custom dataset” (conjunto de datos personalizado), se refiere a un conjunto de datos que ha sido creado específicamente para una tarea o aplicación particular, en lugar de utilizar conjuntos de datos estándar o preexistentes. Este tipo de conjunto de datos se adapta a las necesidades específicas de un proyecto o investigación y puede incluir datos recolectados, etiquetados o procesados de manera personalizada.

Por qué utilizar `load_dataset()`:

Facilidad de uso: `load_dataset` simplifica el proceso de carga de datos, especialmente para usuarios que no están familiarizados con la estructura y el formato de los conjuntos de datos.

Conjuntos de datos preprocesados: La función `load_dataset` proporciona conjuntos de datos que ya están preprocesados y listos para usar en tareas de aprendizaje automático y aprendizaje profundo.

Soporte para múltiples conjuntos de datos: `datasets` ofrece una amplia variedad de conjuntos de datos, incluyendo conjuntos de datos estándar utilizados en NLP como SQuAD, IMDb, Wikipedia, entre otros. La función `load_dataset` te permite acceder fácilmente a estos conjuntos de datos.

Aplicación de load_dataset():

```
▶ from datasets import load_dataset

data_files = {
    "train": "/content/Generador-Css/dataset/STYLES.json",
    "validation": "/content/Generador-Css/dataset/VALIDATION.json",
    "test": "/content/Generador-Css/dataset/TEST.json"
}

dataset = load_dataset("json", data_files=data_files, field='styles')
```

Figura 58. Creacion del Dataset

Fuente: Elaboración Propia

En el desarrollo e implementación de modelos de Inteligencia Artificial, hay una tendencia a usar una variedad de herramientas con un nivel de abstracción y complejidad alto. Por esta razón, se considera oportuno explicar de manera detallada cada componente, librería o función utilizada en el desarrollo del modelo de generación de código CSS.

En la figura 45 se puede apreciar el código que hace posible la creación del dataset pero expliquemos por bloques el funcionamiento y el porqué de la implementación.

Como ya se explicó en la primera línea de código se tiene la importación de la función `load_dataset()` a continuación se puede ver la creación de un diccionario llamado `data_files` que contiene rutas a archivos de datos para diferentes divisiones de un conjunto de datos (entrenamiento, validación y prueba). A continuación, se explicará cada parte y su significado:

```
data_files = {
    "train": "/content/Generador-Css/dataset/STYLES.json",
    "validation": "/content/Generador-Css/dataset/VALIDATION.json",
    "test": "/content/Generador-Css/dataset/TEST.json"
}
```

Figura 59. Creacion de Diccionario de rutas de los diferentes Datasets preprocesados

Fuente: Elaboración Propia

data_files: Es un diccionario que almacena las rutas de los archivos de datos para la creacion de divisiones del conjunto de datos o dataset.

"train": "/content/Generador-Css/dataset/STYLES.json": Esta línea especifica la ruta al archivo de datos de entrenamiento. El archivo STYLES.json en la carpeta /content/Generador-Css/dataset/ contiene los datos que se utilizarán para entrenar el modelo.

"validation": "/content/Generador-Css/dataset/VALIDATION.json": Esta línea especifica la ruta al archivo de datos de validación. El archivo VALIDATION.json en la misma carpeta contiene los datos que se utilizarán para evaluar el modelo durante el entrenamiento para evitar el sobreajuste.

"test": "/content/Generador-Css/dataset/TEST.json": Esta línea especifica la ruta al archivo de datos de prueba. El archivo TEST.json se utiliza para evaluar el rendimiento final del modelo después de haber sido entrenado y validado.

Todos los archivos de tipo JSON tienen la misma estructura que se definió en el apartado anterior sobre los dataset de entrenamiento, validación y prueba.

Posteriormente unificamos nuestros archivos tipo JSON que contienen nuestros datasets de entrenamiento, validación y prueba utilizando la función antes mencionada `load_dataset()`

```
dataset = load_dataset("json", data_files=data_files, field='styles')
```

Figura 60. Creacion de Dataset de Entrenamiento

Fuente: Elaboración Propia

En la figura 47 se puede apreciar el uso de la función `load_dataset()` que se utiliza para cargar conjuntos de datos utilizando la biblioteca “datasets” de Hugging Face. en donde se introducen tres diferentes parámetros que explicaremos a continuación:

Parámetros:

"json":

Descripción: Especifica el formato del conjunto de datos que se va a cargar.

Valor: En este caso, indicamos que los archivos de datos están en formato JSON.

data_files=data_files:

Descripción: Especifica el diccionario que contiene las rutas de los archivos de datos para diferentes divisiones (entrenamiento, validación y prueba).

Valor: data_files es el diccionario que se definió anteriormente con las rutas de los archivos JSON para cada división del conjunto de datos.

field='styles':

Descripción: Especifica el campo o la clave del diccionario dentro de los archivos JSON que contiene los datos que se van a cargar.

Valor: En este caso, se implementó dentro de cada archivo JSON un campo llamado styles que contiene los datos que se desea cargar.

Implementación de Modelo Pre-entrenado GPT-2 XL usando GPT2Model

GPT2Model es un modelo de lenguaje basado en la arquitectura Transformer desarrollado por OpenAI. Es una versión mejorada y más grande de su predecesor, GPT (Generative Pre-trained Transformer), diseñado para generar texto de alta calidad y comprensible, incluso en tareas complejas de lenguaje natural.

El propósito principal de GPT2Model es generar texto coherente y relevante en una variedad de tareas de procesamiento del lenguaje natural (NLP), como generación de texto, traducción automática, resumen de texto, análisis de sentimientos, entre otros. También puede ser utilizado como base para ajustar fino (fine-tuning) en tareas específicas, utilizando conjuntos de datos más pequeños para adaptar el modelo a una tarea particular.

La eficacia de GPT-2 se debe en gran parte a su arquitectura Transformer y su capacidad para aprender representaciones distribuidas de palabras y secuencias de texto.

```
from transformers import GPT2Model  
model = GPT2Model.from_pretrained('gpt2-xl')
```

Figura 61. Cargando el Modelo GPT-2 XL

Fuente: Elaboración Propia

Se puede apreciar en la figura 48 que el siguiente bloque de código la importa el Modelo GPT-2 y carga una instancia pre-entrenada de “gpt2-xl”.

Abordaremos el significado de manera más explícita con la siguiente explicación.

from transformers import GPT2Model: Este código importa la clase GPT2Model del módulo transformers de la biblioteca Hugging Face Transformers. La biblioteca Transformers proporciona implementaciones de modelos de lenguaje pre-entrenados, como GPT-2, BERT, T5, entre otros.

GPT2Model.from_pretrained('gpt2-xl'): GPT2Model.from_pretrained() es un método estático de la clase GPT2Model que se utiliza para cargar una instancia pre-entrenada del modelo GPT-2.

'gpt2-xl': Es una cadena que especifica el tamaño o la variante del modelo que quieras cargar. En este caso, 'gpt2-xl' se refiere a la versión extra grande de GPT-2 (XL), que tiene más parámetros y es más potente que las versiones más pequeñas.

El método from_pretrained descarga automáticamente los pesos pre-entrenados y la configuración del modelo desde un repositorio en línea y crea una nueva instancia de GPT2Model con esos pesos y configuración.

Funcionamiento Interno:

Descarga de Pesos Pre-Entrenados: El método `from_pretrained()` descarga automáticamente los pesos pre-entrenados del modelo GPT-2 (XL) desde un repositorio en línea. Estos pesos son los que fueron aprendidos durante el entrenamiento del modelo en una gran cantidad de datos textuales.

Creación de la Instancia del Modelo: Una vez descargados los pesos, el método `from_pretrained` crea una nueva instancia de GPT2Model con esos pesos y la configuración correspondiente. Esta instancia ya está lista para ser utilizada para inferencia, generación de texto, o ajuste fino (fine tuning) en tareas específicas.

Utilización del Modelo: La variable `model` ahora contiene una instancia del modelo GPT-2 (XL) pre-entrenado, que puede ser utilizada para generar texto, realizar inferencias en tareas de procesamiento del lenguaje natural, o ser ajustado fino (fine-tuning) en tareas específicas

utilizando conjuntos de datos adicionales que para este caso sería el dataset de estilos CSS que con anterioridad preprocesamos y cargamos listo para ser utilizado para el entrenamiento del modelo.

5.3.2.3 Tokenización e Incrustación

La elección de la técnica de tokenización depende del tipo de texto y del objetivo del análisis o del modelo. En muchos casos, se combinan varias técnicas para obtener un mejor resultado. Por ejemplo, en NLP avanzado, se utilizan técnicas como la tokenización con subword units para capturar tanto la semántica de las palabras como su estructura morfológica.

Sin embargo, el modelo pre-entrenado GPT-2 XL incluye un tokenizador (GPT2Tokenizer) proporcionado por Hugging Face. Este tokenizador nos permite realizar la tokenización de los datos de entrada que conforman nuestro dataset. Una vez que hemos tokenizado nuestro dataset, podemos pasar estos tokens al modelo para llevar a cabo el entrenamiento.

A continuación, explicaremos el código utilizado para la tokenización de nuestro dataset en la siguiente figura.

```
[ ] from transformers import GPT2Tokenizer
tokenizer = GPT2Tokenizer.from_pretrained('gpt2-xl')
def tokenize_function(examples):
    return tokenizer(examples["style"], padding="max_length", truncation=True)
tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

Figura 62. Cargando el Tokenizador GPT2Tokenizer para la Tokenización del Dataset

Fuente: Elaboración Propia

from transformers import GPT2Tokenizer: Importa la clase GPT2Tokenizer de la biblioteca transformers de Hugging Face. Esta clase se utiliza para tokenizar texto usando el tokenizador pre-entrenado de GPT-2.

tokenizer = GPT2Tokenizer.from_pretrained('gpt2-xl'): Carga el tokenizador pre-entrenado GPT-2 XL utilizando el método from_pretrained. El parámetro 'gpt2-xl' especifica que queremos usar la versión "extra large" del modelo GPT-2.

def tokenize_function(examples): Define una función llamada tokenize_function que toma un argumento examples. Esta función se utilizará más tarde para aplicar la tokenización a los ejemplos del dataset.

return tokenizer(examples["style"], padding="max_length", truncation=True): Dentro de la función tokenize_function, se utiliza el tokenizador para tokenizar los ejemplos del dataset.

- **examples["style"]:** Selecciona la columna "style" de los ejemplos del dataset. Se asume que cada ejemplo tiene una columna "style" que contiene el texto que queremos tokenizar.
- **padding="max_length":** Añade padding a los tokens para que todos tengan la misma longitud. La longitud se establece en el máximo de tokens encontrados en el batch.
 - Ejemplo: Si el tokenizador convierte una frase en 20 tokens, pero nuestro modelo GPT-2 XL espera siempre secuencias de 50 tokens, entonces se añadirán 30 tokens de padding al final de la secuencia para alcanzar la longitud deseada.
- **truncation=True:** Trunca los tokens si exceden el límite máximo de longitud establecido por el modelo. En este caso, coincide con la longitud máxima de tokens después del padding.
 - Ejemplo: Si tenemos una frase que se tokeniza en 60 tokens, pero nuestro modelo GPT-2 XL solo puede manejar secuencias de hasta 50 tokens, entonces se eliminarán 10 tokens del final de la secuencia para que tenga la longitud adecuada.

tokenized_datasets = dataset.map(tokenize_function, batched=True): Aplica la función tokenize_function a cada ejemplo del dataset.

- **dataset:** Hace referencia a nuestro objeto dataset que contiene los datos a tokenizar.
- **map:** Método que aplica la función tokenize_function a cada dato del dataset.
- **batched=True:** Indica que la tokenización se realiza en batches (lotes) de datos, lo cual es más eficiente y rápido que procesar cada ejemplo individualmente.

5.3.2.4 Fine-Tuning

Para nuestro caso específico, dado que hemos seleccionado el modelo GPT-2 XL, que genera texto en inglés, aplicaremos fine-tuning para que nuestro modelo pre-entrenado se especialice en la tarea de generación de código CSS. Esta técnica de fine-tuning nos permitirá adaptar el modelo a las particularidades y especificidades del lenguaje y la estructura del código CSS, mejorando así su capacidad para generar código coherente y válido.

Aplicaremos el fine-tuning en el momento exacto después de haber cargado el modelo pre-entrenado GPT-2 XL, pero antes de comenzar con la generación de texto o la inferencia en la tarea objetivo. Es importante destacar que el fine-tuning debe realizarse cuidadosamente, ajustando los hiperparámetros y monitorizando el rendimiento del modelo durante el entrenamiento para evitar el sobreajuste y garantizar que el modelo se especialice de manera efectiva en la tarea de generación de código CSS.

Con este enfoque de fine-tuning, nuestro objetivo es aprovechar el conocimiento previamente adquirido por el modelo GPT-2 XL en la generación de texto en inglés y adaptarlo específicamente a la tarea de generación de código CSS.

Creación de Sub datasets

Creamos un subconjunto más pequeño del conjunto de datos completo para realizar ajustes y reducir el tiempo necesario:

```
[ ] small_train_dataset = tokenized_datasets["train"].shuffle(seed=42).select(range(1000))
small_eval_dataset = tokenized_datasets["test"].shuffle(seed=42).select(range(1000))
```

Figura 63. Creacion de Sub Datasets

Fuente: Elaboración Propia

small_train_dataset = tokenized_datasets["train"].shuffle(seed=42).select(range(1000)):

tokenized_datasets["train"]: Aquí estamos accediendo al conjunto de datos de entrenamiento ("train") desde el diccionario tokenized_datasets. Se asume que tokenized_datasets es un diccionario que contiene conjuntos de datos tokenizados.

.shuffle(seed=42): Esta parte del código está barajando el conjunto de datos. El método shuffle reordena aleatoriamente los ejemplos del conjunto de datos. Se especifica seed=42 para asegurar que el orden sea reproducible (es decir, cada vez que ejecutes el código con el mismo valor de semilla, obtendrás el mismo orden aleatorio).

.select(range(1000)): Este método selecciona los primeros 1000 ejemplos del conjunto de datos barajado. Utiliza range(1000) para crear una lista de números del 0 al 999 y luego selecciona esos índices del conjunto de datos.

Por lo tanto, small_train_dataset es un nuevo conjunto de datos que consiste en los primeros 1000 ejemplos del conjunto de datos de entrenamiento original, pero en un orden aleatorio.

small_eval_dataset = tokenized_datasets["test"].shuffle(seed=42).select(range(1000)):

tokenized_datasets["test"]: Aquí estamos accediendo al conjunto de datos de evaluación ("test") desde el diccionario tokenized_datasets.

.shuffle(seed=42): Al igual que antes, estamos barajando el conjunto de datos de evaluación.

.select(range(1000)): También estamos seleccionando los primeros 1000 ejemplos del conjunto de datos de evaluación barajado. Así, small_eval_dataset es un nuevo conjunto de datos que consiste en los primeros 1000 ejemplos del conjunto de datos de evaluación original, pero en un orden aleatorio.

Evaluación

El entrenador no evalúa automáticamente el rendimiento del modelo durante el entrenamiento. Se Debe pasar a Trainer una función para calcular e informar métricas. La biblioteca Evaluar proporciona una función de precisión simple que puede cargar con la función evaluate.load():

```
import evaluate
metric = evaluate.load("accuracy")
```

Figura 64. Creacion de la Metrica de Evaluacion
Fuente: Elaboración Propia

import evaluate: En esta línea, estamos importando un módulo llamado evaluate, que contiene funciones relacionadas con la evaluación del modelo.

metric = evaluate.load("accuracy"): Aquí, estamos llamando a una función load del módulo evaluate con el argumento "accuracy". evaluate.load("accuracy"): La función load es una función definida que carga y devuelve una métrica específica basada en la cadena de texto proporcionada como argumento. En este caso, "accuracy" indica que queremos cargar la métrica de precisión. metric = evaluate.load("accuracy"): El resultado de llamar a evaluate.load("accuracy") se asigna a la variable metric. Por lo tanto, metric contendrá la métrica de precisión que se ha cargado.

Antes de realizar el fine tuning se implementó la función compute_metrics para calcular la precisión de sus predicciones. Antes de pasar sus predicciones para calcular, se debe convertir las predicciones a logits (todos los modelos de Transformers devuelven logits):

```
[ ] def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)
```

Figura 65. Creacion de Funcion para computar las Metricas
Fuente: Elaboración Propia

logits, labels = eval_pred: En esta línea, estamos desempaquetando eval_pred en dos variables: logits y labels.

- logits: Es un tensor que contiene las predicciones del modelo antes de aplicar la función de activación. Es decir, son valores continuos que representan la "confianza" del modelo en cada clase.
- labels: Es un tensor que contiene las etiquetas reales del conjunto de datos.

predictions = np.argmax(logits, axis=-1): Aquí estamos utilizando np.argmax de NumPy para encontrar el índice del valor máximo en logits a lo largo del último eje (axis=-1). Esto nos da las predicciones del modelo en forma de índices de clases.

- predictions: Es un tensor que contiene las clases predichas por el modelo para cada ejemplo.

return metric.compute(predictions=predictions, references=labels): Finalmente, estamos llamando a una función compute de nuestra función metric anteriormente explicada. Esta función toma dos argumentos:

predictions: Las clases predichas por el modelo.

references: Las etiquetas reales del conjunto de datos. La función compute calcule y devuelva algunas métricas (como precisión, recall, F1-score, etc.) basadas en las predicciones y las etiquetas reales.

Trainer (Entrenador)

Transformers proporciona una clase de Trainer (Entrenador) optimizada para el entrenamiento Modelos de Transformers, lo que facilita comenzar a entrenar sin escribir manualmente su propio ciclo de entrenamiento. La API de Trainer admite una amplia gama de opciones y funciones de capacitación, como registro, acumulación de gradientes y precisión mixta.

Se cargo el modelo y especifico la cantidad de etiquetas esperadas.

Para monitorear las métricas de evaluación durante el ajuste fino (fine tuning), especifique el parámetro `evaluation_strategy` en sus argumentos de entrenamiento para informar la métrica de evaluación al final de cada época:

```
from transformers import TrainingArguments
training_args = TrainingArguments(output_dir="test_trainer", evaluation_strategy="epoch")
```

Figura 66. Implementación de TrainingArguments

Fuente: Elaboración Propia

```
from transformers import TrainingArguments:
```

En esta línea, estamos importando dos clases del paquete transformers:

TrainingArguments: Esta clase se utiliza para definir y almacenar los argumentos de entrenamiento para un modelo. Esto incluye cosas como la ruta del directorio de salida, la estrategia de evaluación, el número de épocas, etc.

```
training_args = TrainingArguments  
(output_dir="test_trainer",evaluation_strategy="epoch"):
```

Aquí estamos creando una instancia de TrainingArguments llamada training_args con dos argumentos:

- output_dir="test_trainer": Especifica el directorio donde se guardarán los archivos de salida del entrenamiento, como los checkpoints del modelo y los registros de entrenamiento. En este caso, el directorio se llamará "test_trainer".
- evaluation_strategy="epoch": Define la estrategia de evaluación durante el entrenamiento. En este caso, la evaluación se realizará al final de cada época ("epoch"). Esto significa que después de cada época de entrenamiento, el modelo se evaluará en un conjunto de datos de evaluación para medir su rendimiento.

Posteriormente creamos nuestro Trainer

```
from transformers import Trainer  
trainer = Trainer(  
    model=model,  
    args=training_args,  
    train_dataset=small_train_dataset,  
    eval_dataset=small_eval_dataset,  
    compute_metrics=compute_metrics,  
)
```

Figura 67. Implementacion del Trainer
Fuente: Elaboración Propia

from transformers import Trainer: Aquí estamos importando la clase Trainer del paquete transformers. Como se mencionó anteriormente, Trainer es responsable de manejar el proceso de entrenamiento de un modelo con Hugging Face Transformers en nuestro caso el modelo GPT-2 XL.

A continuación, se crea una instancia de Trainer llamada trainer y configurándola con varios argumentos:

model=model: Aquí pasamos el modelo GPT-2 XL que se va a entrenar. Se asume que model es un modelo predefinido o personalizado que se ha definido previamente.

args=training_args: Le pasamos los argumentos de entrenamiento definidos en training_args. Estos argumentos controlan el comportamiento del entrenamiento, como la ruta del directorio de salida y la estrategia de evaluación que anteriormente se definió.

train_dataset=small_train_dataset: Especificamos el conjunto de datos de entrenamiento que se utilizará para entrenar el modelo. small_train_dataset es el conjunto de datos de entrenamiento reducido que hemos creado anteriormente.

eval_dataset=small_eval_dataset: Especificamos el conjunto de datos de evaluación que se utilizará para evaluar el modelo durante el entrenamiento. small_eval_dataset es el conjunto de datos de evaluación reducido que hemos creado anteriormente.

compute_metrics=compute_metrics: Pasamos la función compute_metrics que hemos definido anteriormente. Esta función se utilizará para calcular métricas de evaluación (como precisión, recall, etc.) durante el entrenamiento y la evaluación del modelo.

5.3.2.5 Entrenamiento

Después de haber preparado y configurado adecuadamente el modelo utilizando el Trainer, el siguiente paso es iniciar el entrenamiento del modelo. Durante este proceso de entrenamiento, aplicamos el fine-tuning utilizando el nuevo conjunto de datos específico para adaptar el modelo pre-entrenado a la tarea o problema que estamos abordando que es la generación de código CSS.

Este proceso de fine-tuning nos permite aprovechar las características generales del lenguaje que el modelo ha aprendido durante el pre-entrenamiento, mientras lo adaptamos y afinamos para que se desempeñe de manera óptima en la nueva tarea o conjunto de datos.

```
trainer.train()
```

Figura 68. Entrenamiento del Modelo

Fuente: Elaboración Propia

Esta línea invoca el método train() en el objeto trainer, que es una instancia de la clase Trainer del paquete transformers de Hugging Face.

Cuando se llama al método train(), el Trainer comenzará el proceso de entrenamiento del modelo que se ha configurado previamente. A continuación se detalla lo que hace este método:

Inicialización:

Antes de comenzar el entrenamiento, el Trainer inicializa todos los componentes necesarios, como el modelo, el optimizador, el planificador de tasa de aprendizaje, etc., basándose en la configuración proporcionada.

Entrenamiento: Durante el entrenamiento, el Trainer recorre el conjunto de datos de entrenamiento (small_train_dataset en este caso) en lotes y entrena el modelo. Para cada lote, el proceso de entrenamiento incluye:

- Propagación hacia adelante (Forward Propagation): El modelo calcula las predicciones para el lote actual.
- Cálculo de la pérdida (Loss Calculation): Se calcula la pérdida entre las predicciones del modelo y las etiquetas reales del lote.
- Propagación hacia atrás (Backpropagation): Se calculan los gradientes de la pérdida respecto a los parámetros del modelo.

- Actualización de los parámetros: Se actualizan los parámetros del modelo utilizando un algoritmo de optimización (como el descenso del gradiente estocástico) para minimizar la pérdida.
- Evaluación: Según la configuración de `evaluation_strategy` en `training_args`, después de cada época, el Trainer evaluará el modelo en el conjunto de datos de evaluación (`small_eval_dataset` en este caso) utilizando la función `compute_metrics` para calcular y mostrar las métricas de evaluación, como la precisión, el recall, etc.
- Guardado de checkpoints: Además, dada la configuración de `TrainingArguments`, el Trainer puede guardar checkpoints del modelo y otros archivos de salida en el directorio especificado en `output_dir`.
- Finalización: Una vez que se completa el número especificado de épocas o pasos de entrenamiento, el método `train()` finaliza y devuelve el control al programa principal.

Se realizaron n cantidad de experimentos o entrenamientos variando los hiperparámetros para lograr la mayor precisión (accuracy) y también para lograr la aproximación de la precisión de entrenamiento con la precisión de validación (que es la precisión obtenida con el dataset de validación), de esta manera evitando el sobreajuste (overfitting). Una vez concluidos los experimentos o entrenamientos, se espera que el modelo alcance la mayor precisión posible.

A continuación, se presenta un conjunto de cinco experimentos realizados con información relevante sobre el entrenamiento:

Experimento 1

Se realizó el entrenamiento con los siguientes hiperparámetros :

- `output_dir="test_trainer"`
 - Utilidad: Al finalizar el entrenamiento, el modelo y otros archivos relacionados se guardarán en este directorio.
- `evaluation_strategy="epoch"`
 - Utilidad: Se evaluará el modelo al final de cada época (epoch).

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	1.025352	0.547000
2	No log	0.998141	0.555000
3	No log	0.990429	0.595000

```
TrainOutput(global_step=375, training_loss=0.989107177734375, metrics={'train_runtime': 0.994, 'total_flos': 789354427392000.0, 'train_loss': 0.989107177734375, 'epoch': 3.0})
```

Figura 69. Resultados del Entrenamiento del Modelo con 3 épocas

Fuente: Elaboración Propia

Resultados del Entrenamiento

El resultado del entrenamiento de un modelo generalmente se registra en forma de métricas y pérdidas durante cada época (o iteración completa a través del conjunto de datos). Estas métricas y pérdidas nos brindan información valiosa sobre el rendimiento y la calidad del modelo durante el entrenamiento y la validación. A continuación, se explica qué significan las métricas:

Columnas de Resultados:

Epoch (Época): Representa la época actual durante el entrenamiento. Cada época se refiere a una iteración completa a través del conjunto de datos de entrenamiento.

Training Loss (Pérdida de Entrenamiento): Es el valor medio de la función de pérdida calculada en el conjunto de datos de entrenamiento durante la época actual. Esta métrica indica cuán bien o mal está el modelo ajustando los pesos durante el entrenamiento. En este caso, aparece "No log", lo que indica que no se ha registrado la pérdida de entrenamiento o que hubo un problema durante el entrenamiento.

Validation Loss (Pérdida de Validación): Es el valor medio de la función de pérdida calculada en el conjunto de datos de validación durante la época actual. Esta métrica proporciona una estimación del rendimiento del modelo en datos no vistos durante el

entrenamiento. Un valor menor de pérdida de validación generalmente indica un mejor rendimiento del modelo, pero es importante tener en cuenta que una pérdida muy baja en validación puede ser señal de sobreajuste.

Accuracy (Exactitud): Es la proporción de predicciones correctas realizadas por el modelo en el conjunto de datos de entrenamiento durante la época actual. Esta métrica se calcula como el número de predicciones correctas dividido por el número total de predicciones. Una precisión más alta indica un mejor rendimiento del modelo en la tarea específica para la que se está entrenando.

Validation Accuracy (Exactitud de Validación): Es la proporción de predicciones correctas realizadas por el modelo en el conjunto de datos de validación durante la época actual. Esta métrica se calcula como el número de predicciones correctas dividido por el número total de predicciones. Una precisión más alta indica un mejor rendimiento del modelo en la tarea específica para la que se está entrenando.

Interpretación de los Resultados con 3 épocas

Epoch 1:

- No se registró la pérdida de entrenamiento.
- La pérdida de validación es 1.025352.
- La precisión es 0.547000.
- La precisión de validación es 0.446000.

Epoch 2:

- No se registró la pérdida de entrenamiento.
- La pérdida de validación es 0.998141.
- La precisión ha aumentado ligeramente a 0.555000.
- La precisión de validación ha aumentado ligeramente a 0.447000.

Epoch 3:

- No se registró la pérdida de entrenamiento.
- La pérdida de validación ha disminuido a 0.990429.
- La precisión ha aumentado significativamente a 0.595000.
- La precisión de validación ha aumentado significativamente a 0.445000.

Análisis

- La pérdida de validación está disminuyendo con cada época, lo que generalmente indica que el modelo está mejorando su capacidad para generalizar a partir de los datos de entrenamiento al conjunto de validación.
- La precisión también está aumentando con cada época, lo cual es una señal positiva del rendimiento del modelo. Sin embargo, se debe observar si este aumento en la precisión es consistente a medida que se continúa el entrenamiento.

Experimento 2

Se realizo el entrenamiento con los siguientes hiperparametros :

- output_dir="test_trainer"
 - Utilidad: Al finalizar el entrenamiento, el modelo y otros archivos relacionados se guardarán en este directorio.
- evaluation_strategy="epoch"
 - Utilidad: Se evaluará el modelo al final de cada época (epoch).
- num_train_epochs=10
 - Utilidad: Se especifica el número de épocas, en este caso 10

Interpretación de los Resultados con 10 épocas

trainer.train()				
Epoch	Training Loss	Validation Loss	Accuracy	
1	No log	1.215096	0.445000	[1250/1250 22:02, Epoch 10/10]
2	No log	1.128125	0.503000	
3	No log	1.205466	0.522000	
4	0.953800	1.400184	0.515000	
5	0.953800	1.478618	0.549000	
6	0.953800	2.014113	0.561000	
7	0.953800	2.433239	0.540000	
8	0.152700	2.626811	0.544000	
9	0.152700	2.685140	0.553000	
10	0.152700	2.788081	0.537000	

Figura 70. Resultados del Entrenamiento del Modelo con 10 épocas

Fuente: Elaboración Propia

Épocas 1-3:

- La pérdida de entrenamiento no se registró (No log), lo cual puede ser debido a un problema en el registro de la métrica o a un problema en el modelo.
- La pérdida de validación muestra una tendencia a aumentar ligeramente después de la primera época, mientras que la precisión aumenta gradualmente.

Épocas 4-6:

- La pérdida de entrenamiento comienza a disminuir desde la cuarta época, lo que indica que el modelo está mejorando su capacidad para ajustarse a los datos de entrenamiento.
- Sin embargo, la pérdida de validación comienza a aumentar, lo que podría ser una señal de sobreajuste (overfitting). La precisión sigue mejorando, pero a un ritmo más lento.

Épocas 7-10:

- La pérdida de entrenamiento sigue disminuyendo, lo que indica un buen ajuste a los datos de entrenamiento.

- La pérdida de validación continúa aumentando, confirmando la sospecha de sobreajuste (overfitting).
- La precisión muestra variaciones, pero no muestra una mejora significativa después de la séptima época.

Recomendaciones:

Sobreajuste: Los síntomas de sobreajuste son evidentes al entrenar con 10 épocas, ya que la pérdida de validación comienza a aumentar mientras que la pérdida de entrenamiento sigue disminuyendo. Es importante considerar técnicas como la regularización, la reducción de la complejidad del modelo o el uso de más datos para combatir el sobreajuste.

Mejorar del modelo: Se debe intentar ajustar los hiperparámetros, como la tasa de aprendizaje, el tamaño del lote o la arquitectura del modelo, para mejorar el rendimiento.

Monitoreo: Se debe seguir monitoreando las métricas de pérdida y precisión durante el entrenamiento para evaluar la efectividad de las mejoras implementadas.

Experimento 3

Se realizó el entrenamiento con los siguientes hiperparámetros :

- output_dir="test_trainer"
 - Utilidad: Al finalizar el entrenamiento, el modelo y otros archivos relacionados se guardarán en este directorio.
- evaluation_strategy="epoch"
 - Utilidad: Se evaluó el modelo al final de cada época (epoch).
- num_train_epochs=10
 - Utilidad: Se especificó el número de épocas, en este caso 10
- learning_rate=5e-5
 - Utilidad: Es la tasa de aprendizaje que determina el tamaño de los ajustes que se realizan durante el entrenamiento.
- per_device_train_batch_size=16
 - Utilidad: Es el número de ejemplos de entrenamiento que se procesan en una iteración.

- weight_decay=0.01
 - Utilidad: Se agrego una penalización a los pesos grandes del modelo durante el entrenamiento para prevenir el sobreajuste.
- load_best_model_at_end=True
 - Utilidad: Se determino que se debe cargar el mejor modelo (según la métrica metric_for_best_model) al final del entrenamiento.
- metric_for_best_model="accuracy"
 - Utilidad: Se indico que la métrica de precisión (accuracy) se utilizará para evaluar y seleccionar el mejor modelo.
- greater_is_better=True,
 - Utilidad: Se estableció que un valor mayor de la métrica especificada en metric_for_best_model es mejor. Por lo tanto, dado que se definió metric_for_best_model="accuracy", un valor más alto de precisión es considerado mejor. Si greater_is_better=False, entonces un valor menor de la métrica sería mejor.
- save_strategy="epoch"
 - Utilidad: Se indico que se guardará un checkpoint del modelo al final de cada época (epoch) durante el entrenamiento.

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	1.112796	0.511000
2	No log	1.034644	0.548000
3	No log	1.083542	0.566000
4	No log	1.411227	0.527000
5	No log	1.487625	0.563000
6	No log	1.484118	0.592000
7	No log	1.790912	0.609000
8	0.519900	2.032171	0.607000
9	0.519900	2.196414	0.584000
10	0.519900	2.151977	0.604000

Figura 71. Resultados del Entrenamiento del Modelo con 10 épocas y ajustes adicionales de hiperparametros
Fuente: Elaboración Propia

Interpretación de los Resultados

Épocas 1 a 3:

- Pérdida de Validación: La pérdida de validación disminuye ligeramente de 1.112796 a 1.083542 en la época 3.
- Precisión: La precisión aumenta de 51.1% en la época 1 a 56.6% en la época 3. Durante estas primeras épocas, el modelo muestra una ligera mejora tanto en la pérdida de validación como en la precisión.
- Esto sugiere que el modelo está aprendiendo a partir de los datos de entrenamiento, aunque la mejora es modesta.

Épocas 4 a 6:

- Pérdida de Validación: La pérdida de validación aumenta significativamente de 1.411227 en la época 4 a 1.484118 en la época 6.
- Precisión: La precisión fluctúa entre 52.7% y 59.2%.

- En estas épocas, la pérdida de validación comienza a aumentar, lo que es una señal de que el modelo podría estar empezando a sobreajustarse. Aunque la precisión mejora ligeramente en la época 6, el aumento continuado de la pérdida de validación es preocupante y sugiere que el modelo no está generalizando bien a nuevos datos.

Épocas 7 a 10:

- Pérdida de Validación: La pérdida de validación sigue aumentando hasta alcanzar valores altos entre 1.790912 y 2.196414.
- Precisión: La precisión se mantiene entre 58.4% y 60.9%.
- En estas últimas épocas, la pérdida de validación y la precisión muestran signos claros de sobreajuste. Aunque la precisión mejora ligeramente, la alta pérdida de validación indica que el modelo no está generalizando bien a nuevos datos, lo que confirma el sobreajuste del modelo.

Recomendaciones:

- Combatir el sobreajuste: Implementar técnicas en los entrenamientos posteriores para combatir el sobreajuste, como regularización o dropout. Estas técnicas ayudarán al modelo a generalizar mejor a nuevos datos y a evitar memorizar el conjunto de entrenamiento.
- Monitoreo Continuo: Continuar monitoreando la pérdida de validación y la precisión durante el entrenamiento. Si la pérdida de validación comienza a aumentar mientras la pérdida de entrenamiento se mantiene baja, es una señal clara de sobreajuste.

Experimento 4

Se realizó el entrenamiento con los mismos hiperparámetros del Experimento 3 pero con 35 épocas:

- num_train_epochs=35
 - Utilidad: Se especificó el número de épocas, en este caso 10

Interpretación de los Resultados con 35 épocas

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	2.224345	0.540000
2	No log	2.226825	0.553000
3	No log	2.214316	0.536000
4	No log	2.640922	0.521000
5	No log	2.511788	0.540000
6	No log	2.663109	0.534000
7	No log	2.666532	0.572000
8	0.184400	2.854013	0.569000
9	0.184400	3.288095	0.530000
10	0.184400	3.404044	0.517000
11	0.184400	3.070557	0.553000
12	0.184400	3.233344	0.558000
13	0.184400	3.575033	0.537000
14	0.184400	3.351683	0.562000
15	0.184400	3.249551	0.570000
16	0.035700	3.576901	0.554000
17	0.035700	3.606007	0.567000
18	0.035700	3.656398	0.569000
19	0.035700	3.668078	0.565000
20	0.035700	3.965503	0.553000
21	0.035700	3.954703	0.551000

Figura 72. Resultados del Entrenamiento del Modelo hasta la época 21

Fuente: Elaboración Propia

22	0.035700	4.179539	0.536000
23	0.035700	3.960376	0.555000
24	0.003300	4.339951	0.535000
25	0.003300	3.938530	0.557000
26	0.003300	3.857939	0.562000
27	0.003300	4.001281	0.565000
28	0.003300	3.894458	0.567000
29	0.003300	3.903498	0.563000
30	0.003300	3.814002	0.571000
31	0.003300	3.904545	0.571000
32	0.008700	3.927652	0.570000
33	0.008700	3.936891	0.570000
34	0.008700	3.947489	0.571000
35	0.008700	3.957203	0.570000

Figura 73. Resultados del Entrenamiento del Modelo hasta la epoca 35

Fuente: Elaboración Propia

Interpretación de los Resultados

Épocas 1-10

- Training Loss: Las pérdidas no se registran, lo que puede indicar un problema en la configuración del modelo o del código.
- Validation Loss: En general, la pérdida varía entre 2.2 y 2.6, lo cual es relativamente alto.
- Accuracy: La precisión varía entre 0.517 y 0.572, no es muy alta pero tampoco muy baja.

Épocas 11-20

- Training Loss: Se estabiliza en un valor más bajo.
- Validation Loss: Aumenta significativamente después de la época 16, lo cual es una señal de sobreajuste (overfitting).
- Accuracy: Varía entre 0.537 y 0.570, con cierta mejoría inicial pero luego se estabiliza.

Épocas 21-30

- Training Loss: Se mantiene en un valor muy bajo, lo que puede indicar sobreajuste.
- Validation Loss: Aumenta aún más, lo que confirma el sobreajuste.
- Accuracy: Se mantiene entre 0.551 y 0.571, sin grandes mejoras.

Épocas 31-35

- Training Loss: Se mantiene estable, pero muy bajo, lo que sugiere sobreajuste.
- Validation Loss: La pérdida sigue aumentando, lo que es una señal clara de sobreajuste.
- Accuracy: Se mantiene estable entre 0.563 y 0.571.

Experimento 5

Se realizó el entrenamiento con los mismos hiperparámetros del Experimento 5 pero con 100 épocas:

- num_train_epochs=100
 - Utilidad: Se especificó el número de épocas, en este caso 100

Interpretación de los Resultados con 100 épocas

Épocas 1-25

- Las pérdidas de entrenamiento no se registran, lo que puede indicar un problema en la configuración del modelo o del código.
- En general, la pérdida de validación varía entre 2.2 y 2.9, lo cual es relativamente alto.
- La precisión varía entre 0.517 y 0.572, no es muy alta pero tampoco muy baja.

Épocas 25-50

- Se estabiliza en un valor más bajo de la pérdida de entrenamiento.
- La pérdida de validación aumenta significativamente después de la época 26, lo cual es una señal de sobreajuste (overfitting).
- La precisión varía entre 0.547 y 0.580, con cierta mejoría inicial pero luego se estabiliza.

Épocas 50-75

- La pérdida de entrenamiento se mantiene en un valor muy bajo, lo que puede indicar sobreajuste.
- La pérdida de validación aumenta aún más, lo que confirma el sobreajuste.
- La precisión se mantiene entre 0.551 y 0.591, sin grandes mejoras.

Épocas 75-100

- La pérdida de entrenamiento se mantiene estable, pero muy bajo, lo que sugiere sobreajuste.
- La pérdida de validación sigue aumentando, lo que es una señal clara de sobreajuste.
- La precisión se mantiene estable entre 0.563 y 0.591.

Como conclusiones iniciales de los cinco experimentos, se puede notar claramente que el modelo se ha estancado en su crecimiento de precisión. La razón detrás de este estancamiento parece ser el sobreajuste (overfitting) que se ha ido generando a lo largo del entrenamiento. Por consiguiente, fue necesario detener el entrenamiento para realizar ajustes adicionales al modelo con el fin de mitigar el sobreajuste y permitir que la métrica de precisión supere el umbral del 0.750. Este nivel de precisión es crucial para asegurar un rendimiento óptimo del modelo.

Es importante destacar que el sobreajuste puede ocurrir cuando el modelo se ajusta demasiado a los datos de entrenamiento específicos, perdiendo así la capacidad de generalizar correctamente a nuevos datos. Para abordar este problema, se implementaron estrategias como la regularización y la validación cruzada durante el proceso de entrenamiento.

Además, se observó que ciertas características de los datos podrían haber contribuido al sobreajuste, como la presencia de datos desbalanceados o la falta de representación de algunas clases. Esto sugiere la necesidad de un preprocesamiento más exhaustivo de los datos y posiblemente la inclusión de técnicas de aumento de datos para mejorar la diversidad y la representación de la muestra.

En futuros experimentos, se planea explorar en mayor profundidad estas cuestiones y probar diferentes arquitecturas de modelos para encontrar la configuración óptima que permita un crecimiento constante en la precisión sin caer en el sobreajuste. Además, se considerará la posibilidad de utilizar técnicas avanzadas de optimización y ajuste de hiperparámetros para mejorar aún más el desempeño del modelo.

5.3.2.6 Resultados de rendimiento del Modelo

Como resultados del rendimiento del Modelo tenemos que para mitigar el sobreajuste (overfitting), se aplicaron diferentes técnicas como Dropout, Regularización por ruido en la entrada, Data Augmentation.

En donde el modelo mejoró su rendimiento a la hora de entrenar, mitigando así el sobreajuste. A continuación, mostramos las gráficas de resultados obtenidos por el modelo considerando las métricas que son:

- Perdida de Validación
- Perdida de Entrenamiento
- Precisión de Validación
- Precisión de Entrenamiento

Inicialmente dado que se tiene un sobreajuste se muestran las gráficas con los resultados de Métricas de Precisión y Perdida confirmando el sobreajuste.

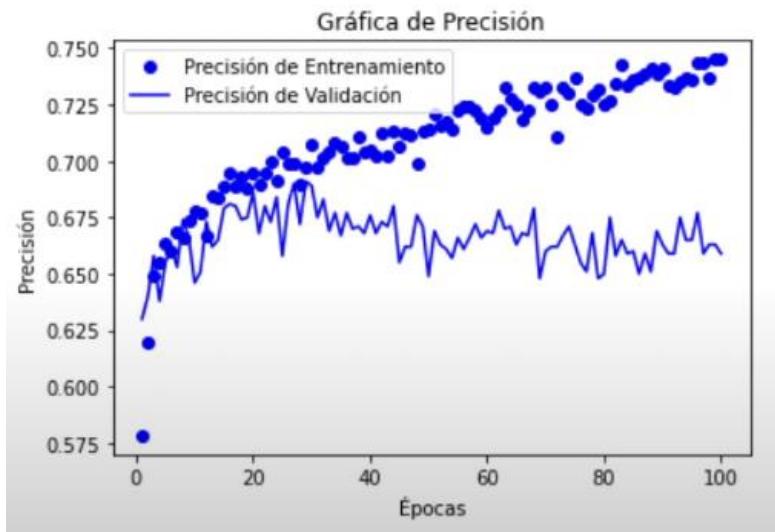


Figura 74. Resultados del Entrenamiento del Modelo con Overfitting Alto Metrica Precision
Fuente: Elaboración Propia

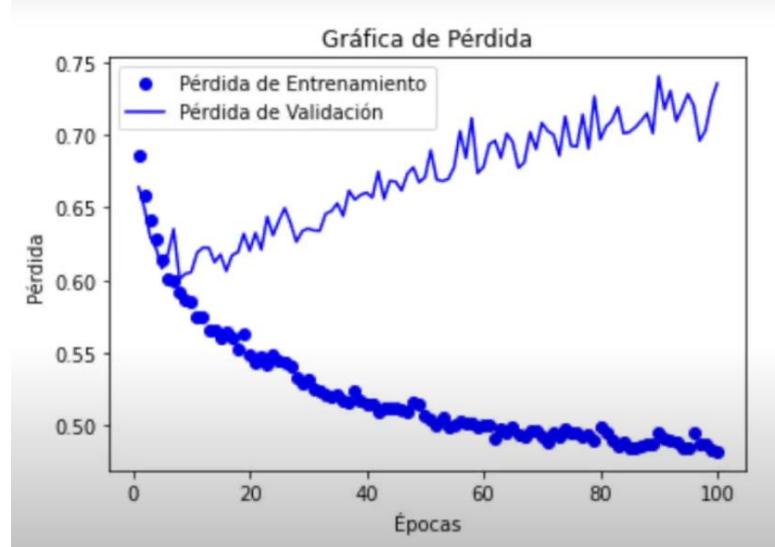


Figura 75. Resultados del Entrenamiento del Modelo con Overfitting Alto Metrica Perdida
Fuente: Elaboración Propia

Posteriormente se aplicó al modelo Data Augmentation, Dropout para mitigar el sobreajuste y estos fueron los resultados:



Figura 76. Resultados del Entrenamiento del Modelo con Overfitting Medio Metrica Perdida
Fuente: Elaboración Propia

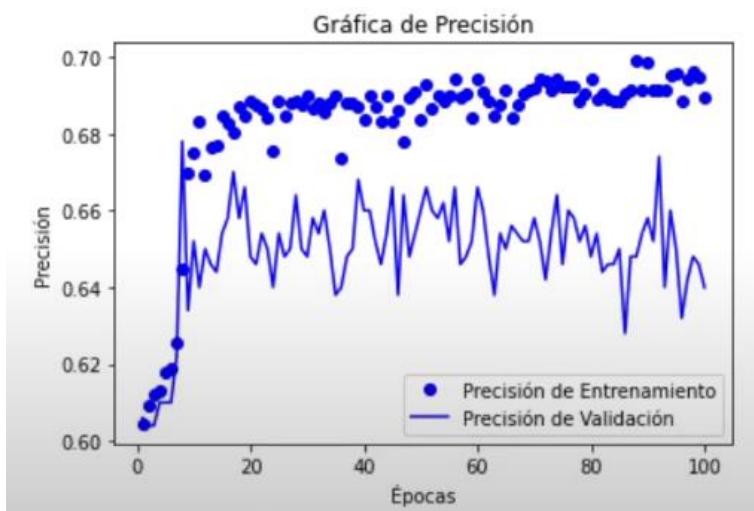


Figura 77. Resultados del Entrenamiento del Modelo con Overfitting Medio Metrica Precision
Fuente: Elaboración Propia

Se puede apreciar en los gráficos que el sobreajuste se redujo de manera exitosa, pero aún se puede reducir mucho más para esto aplicamos la técnica de Regularización por ruido en la entrada y variamos aún más los Datos de validación y tambien el Dropout con esto logramos mitigar aún más el sobreajuste y estos son los resultados:

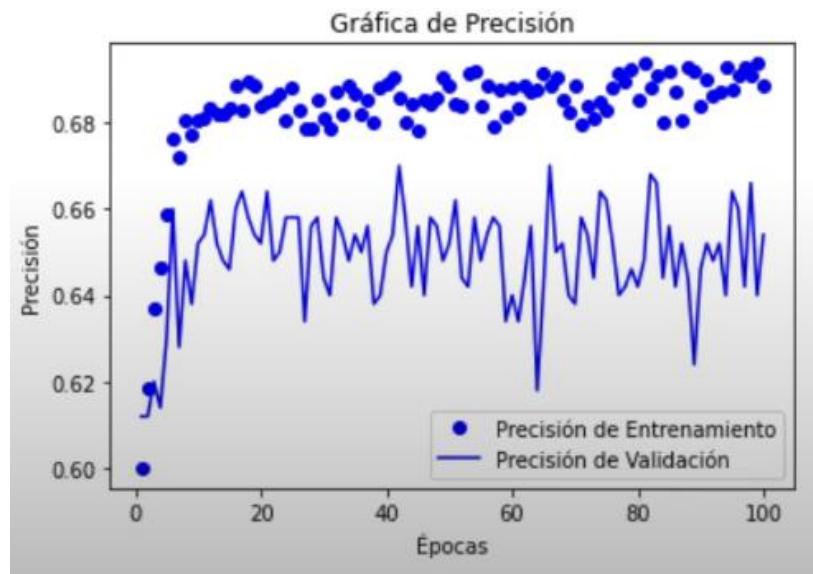


Figura 78. Resultados del Entrenamiento del Modelo con Overfitting Bajo Metrica Precision
Fuente: Elaboración Propia

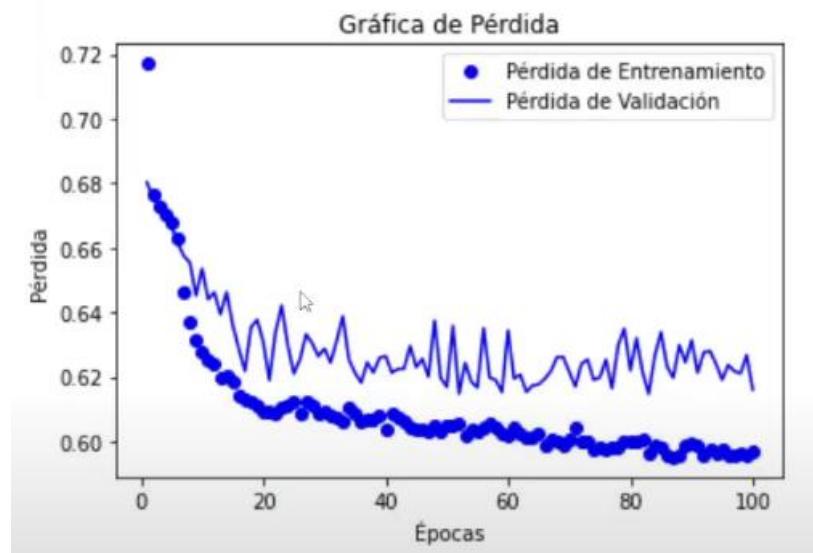


Figura 79. Resultados del Entrenamiento del Modelo con Overfitting Bajo Metrica Perdida
Fuente: Elaboración Propia

Se puede apreciar en los gráficos que se logró mitigar el sobreajuste de manera exitosa. Ahora solo queda continuar con el entrenamiento y mejorar el modelo. También es importante dejar claro que siempre se puede mejorar un modelo y reducir aún más el sobreajuste. Dado que la creación de un modelo de inteligencia artificial es un proceso iterativo y escalable, el entrenamiento puede continuar mejorando, así como el rendimiento. Como respuesta a la solución del proyecto planteado, se optó por obtener resultados de generación eficientes para la generación de código CSS, y estos resultados se pueden ver en las siguientes gráficas:

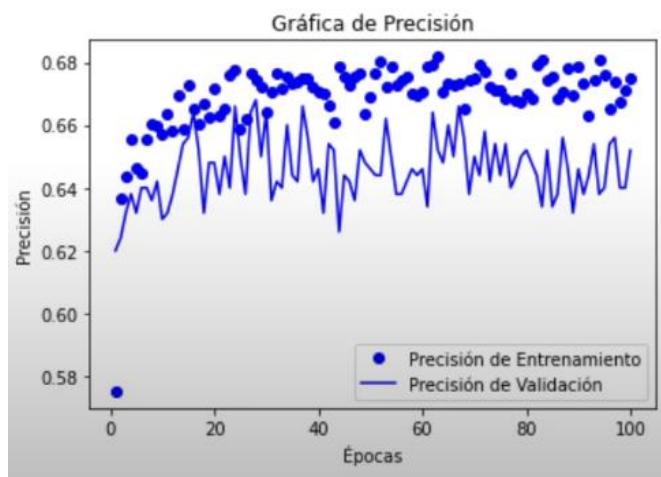


Figura 80. Resultados del Entrenamiento del Modelo Finales Metrica Precision
Fuente: Elaboración Propia

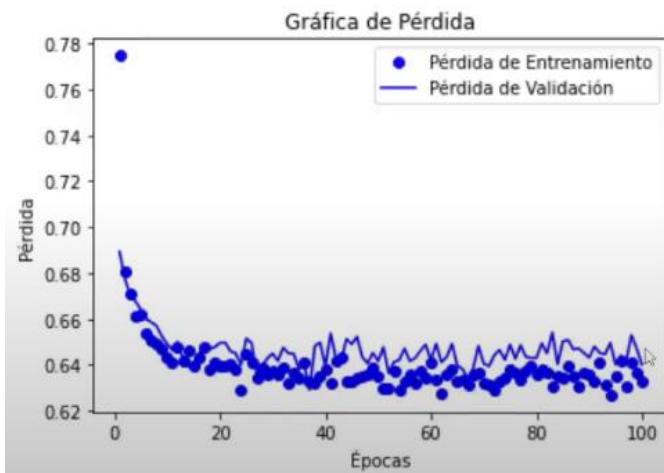


Figura 81. Resultados del Entrenamiento del Modelo Finales Metrica Perdida
Fuente: Elaboración Propia

5.3.2.7 Conclusiones con respecto a la Implementación del Modelo

En conclusión, al examinar los cinco experimentos iniciales que se realizaron modificando diferentes hiperparámetros, podemos concluir que para que el modelo generador de código CSS sea óptimo, es necesario realizar experimentos de forma iterativa y ajustar constantemente los hiperparámetros. Este enfoque de experimentación y análisis es muy habitual en el campo de la inteligencia artificial.

Es esencial ajustar los hiperparámetros para que las métricas del modelo cumplan con el rendimiento esperado. Dado que este proceso es iterativo, hasta alcanzar los resultados esperados por el modelo. La optimización de los hiperparámetros es un proceso clave que requiere paciencia y experimentación para encontrar la configuración óptima que maximice la precisión y el rendimiento del modelo en la tarea específica de generación de código CSS.

También se pudo mitigar de manera exitosa el sobreajuste, lo cual representa un avance significativo en la mejora del modelo. Este logro nos proporciona una mayor confianza en la capacidad del modelo para generalizar correctamente a nuevos datos, lo que es fundamental para la generación de código CSS. Al abordar eficazmente el sobreajuste, hemos aumentado la capacidad del modelo para capturar patrones subyacentes en los datos sin perder su capacidad de generalización.

Este éxito en la mitigación del sobreajuste también resalta la importancia de implementar estrategias efectivas durante el proceso de entrenamiento del modelo. Entre estas estrategias se incluyen técnicas de regularización, dropout y ajuste cuidadoso de hiperparámetros. Además, la comprensión y la evaluación continuas del desempeño del modelo son fundamentales para identificar y abordar cualquier problema de sobreajuste a medida que surja.

Es importante destacar que, si bien hemos logrado mitigar el sobreajuste de manera exitosa en este punto, el proceso de mejora del modelo es continuo. Siempre hay margen para la optimización adicional y la reducción aún mayor del sobreajuste. La naturaleza iterativa y escalable del desarrollo de modelos de inteligencia artificial nos permite continuar refinando y mejorando el rendimiento del modelo con el tiempo.

5.3.3 API de servicios web para el Modelo

Esta API permitirá al desarrollador de software, enviar etiquetas HTML mediante una petición POST al modelo generador de código CSS. Una vez que el modelo recibe estas etiquetas, las analiza e infiere los estilos apropiados para cada una de ellas, generando así código CSS correspondiente a las etiquetas proporcionadas. El objetivo principal es facilitar la creación de estilos CSS basados en el contenido HTML, automatizando el proceso de diseño y personalización de interfaces web.

El flujo de funcionamiento de esta API es sencillo: el usuario envía una solicitud POST con el contenido HTML de las etiquetas que desea estilizar. La API recibe esta solicitud y extrae el contenido HTML, que posteriormente es enviado al modelo generador de código CSS. El modelo, tras analizar el contenido HTML, genera el código CSS necesario para estilizar las etiquetas de acuerdo a los estilos inferidos.

Esta aproximación ofrece una forma eficiente y automatizada de generar estilos CSS, especialmente útil para diseñadores y desarrolladores web que buscan optimizar su flujo de trabajo. Al utilizar esta API, es posible obtener rápidamente estilos CSS personalizados sin necesidad de escribir manualmente el código, lo que puede ahorrar tiempo y reducir posibles errores. Además, esta automatización permite adaptar fácilmente los estilos a diferentes diseños y layouts, mejorando la flexibilidad y adaptabilidad en el desarrollo de proyectos web.

La API actúa como un puente entre el contenido HTML y el código CSS, facilitando la generación de estilos de forma automática y eficiente. A medida que se envían más etiquetas HTML al modelo, se pueden obtener estilos CSS más precisos y personalizados, adaptándose mejor a las necesidades específicas de cada proyecto web.

A continuación, se muestra una breve descripción del endpoint principal de la API, el cual será utilizado por ingenieros de software, desarrolladores frontend y cualquier otro desarrollador o usuario que desee acceder al modelo directamente mediante la API.

Descripción de los Endpoints POST /generate-css:

Descripción: Este endpoint se recibe datos en formato JSON con el contenido HTML de las etiquetas para las cuales se generará el código CSS.

Parámetros:

- html_content: El contenido HTML de las etiquetas para las cuales se generará el CSS.

Respuesta:

- css: El código CSS generado por el modelo basado en el contenido HTML proporcionado.

Ejemplo de Petición:

```
{  
    "html_content": "<h1 class=\"title\">Title</h1>"  
}
```

Modelo generador de código CSS basado en etiquetas HTML:

- Creamos una función generate_css_from_html que toma un parámetro html_content (el contenido HTML de las etiquetas).
- Esta función genera un bloque de código CSS básico utilizando el contenido HTML proporcionado.
- Endpoint POST /generate-css:
 - Definimos una ruta /generate-css que acepta solicitudes POST.
 - Dentro de este endpoint, validamos los datos recibidos en formato JSON.
 - Llamamos a la función generate_css_from_html con el contenido HTML proporcionado.
 - Devolvemos el código CSS generado en formato JSON como respuesta.
- Manejo de errores: Si falta el campo html_content en la solicitud, devolvemos un error con un mensaje adecuado y un código de estado 400 (Bad Request).

5.3.4 Sistema Generador de CSS

Una vez que el Modelo ha sido entrenado cumpliendo con la precisión esperada, y se implementó una API para enviar peticiones al Modelo para generar código CSS, se desarrolló un sistema generador de código CSS que utiliza la API y el Modelo generador de código CSS.

Se implemento diversos componentes en el Sistema generador de CSS los cuales son:

1. Barra de navegación vertical
2. Visualizador
3. Inicio de sesión
4. Historial

5.3.4.1 Estructura de Archivos de Sistema Generador de Código CSS

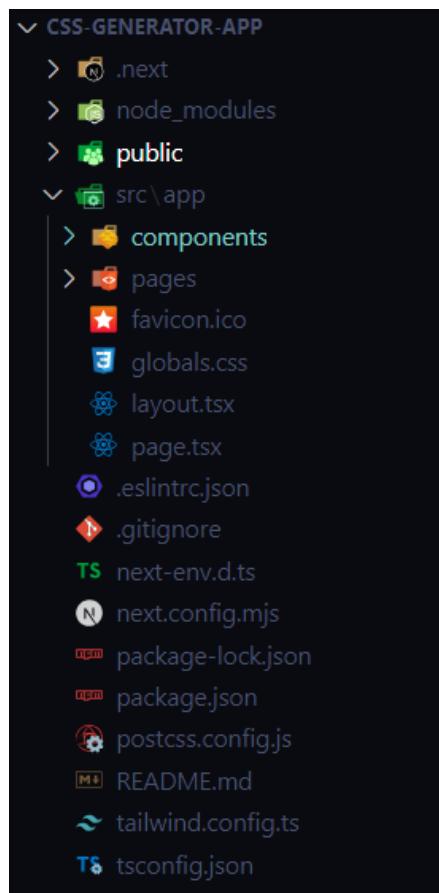


Figura 82. Estructura de archivos general del Sistema Generador de código CSS
Fuente: Elaboración Propia

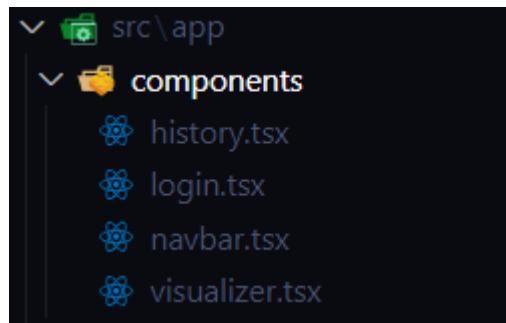


Figura 83. Componentes del Sistema Generador de código CSS

Fuente: Elaboración Propia

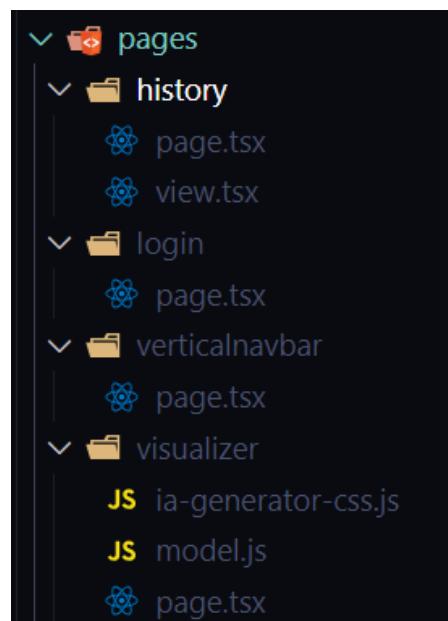


Figura 84. Páginas del Sistema Generador de código CSS

Fuente: Elaboración Propia

5.3.4.1 Barra de navegación vertical

El componente de la barra de navegación vertical se implementó para acceder a los diferentes servicios o componentes del sistema generador de CSS. Estos servicios o componentes incluyen:

- Visualizador
- Historial
- Inicio de Sesión

5.3.4.2 Visualizador

El componente del visualizador está conformado por tres subcomponentes principales. Cada uno tiene una función específica que contribuye al funcionamiento integral del visualizador:

- Editor de Código HTML: Este subcomponente permite a los usuarios introducir y editar el código HTML de manera intuitiva. Proporciona herramientas de resaltado de sintaxis y autocompletado para facilitar la escritura y corrección del código.
- Editor de Código CSS: Este subcomponente se encarga de mostrar y editar el código CSS generado por el modelo. Ofrece características avanzadas como previsualización en tiempo real y sugerencias inteligentes para mejorar la eficiencia en la edición del estilo.
- Vista Previa del Resultado: Este último subcomponente muestra la vista resultante del código HTML introducido por el usuario combinado con el código CSS generado por el modelo de inteligencia artificial. Proporciona una representación visual instantánea de cómo se verá el diseño final, permitiendo al usuario realizar ajustes y optimizaciones de manera efectiva.

El visualizador actúa como una herramienta integral para desarrolladores y diseñadores, ofreciendo una interfaz amigable y eficiente para trabajar con código HTML y CSS.

Gracias a la interacción fluida entre los tres subcomponentes, los usuarios pueden experimentar con diferentes combinaciones de código y ver los resultados al instante, lo que acelera el proceso de diseño y desarrollo web.

El Editor de Código HTML y el Editor de Código CSS proporcionan un entorno de trabajo familiar para aquellos familiarizados con la codificación web, mientras que la Vista Previa del Resultado ofrece una representación visual inmediata, lo que facilita la comprensión y la depuración del código.

Además, el uso de un modelo de inteligencia artificial para generar el código CSS añade un nivel adicional de automatización y optimización al proceso. Este modelo puede aprender y adaptarse a patrones y estilos comunes, generando código CSS más eficiente y optimizado, lo que permite a los usuarios centrarse más en la creatividad y menos en la codificación manual.

5.3.4.3 Inicio de sesión

El componente de inicio de sesión se ha implementado con el propósito principal de permitir que el usuario inicie sesión y, posteriormente, pueda generar código CSS desde el sistema.

Todo el código CSS generado se guarda automáticamente en un historial específico, lo que permite al usuario acceder tanto al código generado como a sus resultados en cualquier momento posterior.

Este componente resulta fundamental para garantizar una experiencia fluida y personalizada para los usuarios del sistema. Al iniciar sesión, los usuarios pueden acceder a todas las funcionalidades ofrecidas por la plataforma, incluida la capacidad de generar y guardar código CSS de manera eficiente.

Además de simplemente permitir el acceso al sistema, el componente de inicio de sesión también garantiza la seguridad de la información del usuario, ya que solo aquellos con credenciales válidas pueden acceder a sus datos y al historial de código generado.

El componente de inicio de sesión está integrado con Google y GitHub para facilitar el inicio de sesión correspondiente. Esta integración permite a los usuarios utilizar sus cuentas de Google o GitHub para iniciar sesión de manera rápida y segura en nuestra plataforma de generación de código CSS.

La integración con estas dos plataformas líderes proporciona a los usuarios opciones adicionales para acceder a nuestros servicios, lo que aumenta la comodidad y la flexibilidad. Además, al utilizar las credenciales de Google o GitHub, los usuarios pueden confiar en la seguridad y la autenticación de estas plataformas reconocidas.

Esta funcionalidad es especialmente útil para aquellos usuarios que ya tienen cuentas en Google o GitHub, ya que les permite acceder a nuestra plataforma sin la necesidad de crear y recordar otra contraseña. Simplificando así el proceso de inicio de sesión y mejorando la experiencia del usuario en general.

5.3.4.4 Historial

El componente de historial es una función esencial que permite al usuario almacenar todo el código CSS generado por el modelo de inteligencia artificial. Además del código CSS, también se guarda en el historial el código HTML correspondiente y la vista resultante generada por el usuario con la ayuda del modelo.

Este componente desempeña un papel crucial al facilitar el seguimiento y la gestión de todo el trabajo realizado por el usuario. Al almacenar tanto el código CSS como el HTML asociado, el historial proporciona una referencia completa de los proyectos en curso y pasados.

5.3.5 API de servicios web para el Sistema Generador de CSS

También se desarrolló una API para la creación, actualización, eliminación y obtención de Vistas y Usuarios. Esto permite a los usuarios revisar su trabajo anterior, realizar comparaciones y ajustes, y recuperar fácilmente cualquier código necesario para proyectos futuros. Además, al incluir el inicio de sesión, podemos gestionar a los usuarios y sus históricos de Vistas generadas por el resultado del código HTML y CSS.

Se implementaron dos CRUD's en la API para Sistema generador de CSS los cuales son:

1. Vistas
2. Usuario

5.3.5.1 Estructura de Archivos de la API del Sistema Generador de Código CSS

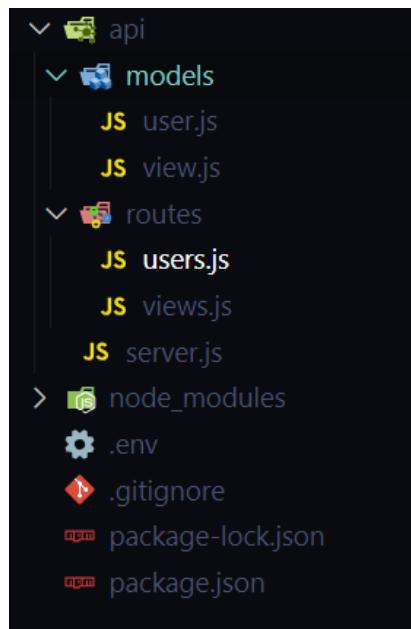


Figura 85. Estructura de archivos general de la API para el Sistema Generador de código CSS

Fuente: Elaboración Propia

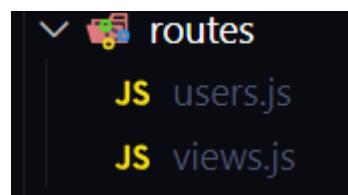


Figura 86. Rutas de la API para Sistema Generador de código CSS

Fuente: Elaboración Propia

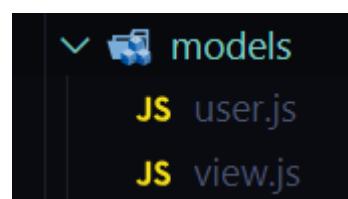


Figura 87. Modelos de la API para Sistema Generador de código CSS

Fuente: Elaboración Propia

5.3.5.2 CRUD de Usuarios

Descripción de los Endpoints:

Descripción: En el CRUD para el usuario se crearon cuatro endpoints que reciben datos en formato JSON los cuales son:

- POST api/user/
- GET api/users/
- PATCH api/user/:id
- DELETE api/user/:id

A continuación, se muestran los servicios en las siguientes capturas de pantalla.

- POST api/user/

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:3000/user/
- Body (JSON):**

```
1 {  
2   "email": "h0921db72035@gmail.com",  
3   "password": "wqf0957"  
4 }
```
- Response Status:** 201 Created
- Response Time:** 58 ms
- Response Size:** 368 B

Figura 88. Petición POST de usuarios de la API

Fuente: Elaboración Propia

- GET api/users/

The screenshot shows the Postman interface with a GET request to `http://localhost:3000/users/`. The response body is a JSON array containing three user objects:

```

1 [
2   {
3     "_id": "6632a2f85539cdcf649f6173",
4     "email": "mouredev@gmail.com",
5     "password": "hackermen33",
6     "__v": 0
7   },
8   {
9     "_id": "6632a3195539cdcf649f6175",
10    "email": "haal@gmail.com",
11    "password": "cityr234",
12    "__v": 0
13   },
14   {
15     "_id": "6632a4127e180e9fc8a0137",
16     "email": "h0921db72035@gmail.com",
17     "password": "wqf0957",
18     "__v": 0
19   }
]

```

Figura 89. Petición GET de usuarios de la API

Fuente: Elaboración Propia

- PATCH api/user/:id

The screenshot shows the Postman interface with a PATCH request to `http://localhost:3000/user/6632a4127e180e9fc8a0137`. The request body is a JSON object:

```

1 {
2   "email": "alanTuringCambridge@gmail.com",
3   "password": "ai&math"
4 }

```

The response body is a JSON object with the updated user information:

```

1 {
2   "_id": "6632a4127e180e9fc8a0137",
3   "email": "alanTuringCambridge@gmail.com",
4   "password": "ai&math",
5   "__v": 0
6 }

```

Figura 90. Petición PATCH de usuarios de la API

Fuente: Elaboración Propia

- DELETE api/user/:id

The screenshot shows a Postman interface with the following details:

- Method:** DELETE
- URL:** http://localhost:3000/users/6632a3195539cdcf649f6175
- Body:** JSON (Pretty) - The response body is:


```

1  {
2   |   "message": "Deleted User"
3 }
```
- Status:** 200 OK
- Time:** 31 ms
- Size:** 293 B

Figura 91. Petición DELETE de usuarios de la API

Fuente: Elaboración Propia

5.3.5.3 CRUD de Vistas

Descripción de los Endpoints:

Descripción: En el CRUD para las vistas se crearon cuatro endpoints que reciben datos en formato JSON, estos datos son el código HTML, CSS, etc los cuales son:

- POST api/view/
- GET api/views/
- PATCH api/view/:id
- DELETE api/view/:id

A continuación, se muestran los servicios en las siguientes capturas de pantalla.

- POST api/view/

The screenshot shows a Postman interface with a POST request to `http://localhost:3000/view/`. The request body is a JSON object:

```

1 {
2   "user": "alexth@gmail.com",
3   "codeCss": ".button{",
4   "codeHtml": "<button>hello</button>"
5 }

```

The response status is `201 Created` with a response body:

```

1 {
2   "user": "alexth@gmail.com",
3   "codeCss": ".button{",
4   "codeHtml": "<button>hello</button>",
5   "_id": "6632bbe73ba9f0dd26b4f65e",
6   "date": "2024-05-01T22:02:15.372Z",
7   "__v": 0
8 }

```

Figura 92. Petición POST de vistas de la API

Fuente: Elaboración Propia

- GET api/views/

The screenshot shows a Postman interface with a GET request to `http://localhost:3000/views/`. The response status is `200 OK` with a response body:

```

114 [
115   {
116     "_id": "66319253858077f85ae45218",
117     "user": "alanTuring@gmail.com",
118     "codeCss": "\nbutton{\n  background-color: #28a745; \n  color: #ffffff; \n  border: none; \n  padding: 12px 24px; \n  font-size: 16px; \n  font-family: Arial, sans-serif; \n  letter-spacing: 1px; \n  font-weight: bold; \n  cursor: pointer; \n  border-radius: 4px; \n  transition: background-color 0.3s ease; \n}\nbutton:hover {\n  background-color: #64580D; \n}",
119     "codeHtml": "<button>",
120     "date": "2024-05-01T00:52:35.889Z",
121     "__v": 0
122   },
123   {
124     "_id": "6631925e858077f85ae4521c",
125     "user": "alanTuring@gmail.com",
126     "codeCss": "\nbutton{\n  background-color: #28a745; \n  color: #ffffff; \n  border: none; \n  padding: 12px 24px; \n  font-size: 16px; \n  font-family: Arial, sans-serif; \n  letter-spacing: 1px; \n  font-weight: bold; \n  cursor: pointer; \n  border-radius: 4px; \n  transition: background-color 0.3s ease; \n}\nbutton:hover {\n  background-color: #64580D; \n}"
127   }
128 ]

```

Figura 93. Petición GET de vistas de la API

Fuente: Elaboración Propia

- PATCH api/view/:id

The screenshot shows a Postman request for a PATCH operation. The URL is `http://localhost:3000/views/663192b1858077f85ae4522e`. The request body is a JSON object:

```

1  {
2   ... "user": "change@gmail.com",
3   ... "codeCss": ".button{ margin}",
4   ... "codeHtml": "<button>yared</button>"
5 }

```

The response status is 200 OK, time: 64 ms, size: 435 B.

Figura 94. Petición PATCH de vistas de la API
Fuente: Elaboración Propia

- DELETE api/view/:id

The screenshot shows a Postman request for a DELETE operation. The URL is `http://localhost:3000/views/663192b1858077f85ae4522e`. The response status is 200 OK, time: 44 ms, size: 293 B. The response body is a JSON object:

```

1  {
2   ... "message": "Deleted View"
3 }

```

Figura 95. Petición DELETE de vistas de la API
Fuente: Elaboración Propia

5.4. Fase de Pruebas

5.4.1 Plan de Pruebas

Las pruebas son una parte fundamental en el proceso de desarrollo de software, ya que permite asegurar la calidad y el funcionamiento correcto del producto a entregar. El plan de pruebas propuesto en el presente proyecto se detalla en la siguiente tabla:

ACTIVIDAD	TÉCNICA	RESPONSABLE	CASO DE PRUEBA
Pruebas de Unidad	Caja Negra Caja Blanca	Desarrollador	Registro de Información Componente Visualizador Componente de Historial
Pruebas de Componentes o Integración	Verificación de Interfaces.	Desarrollador	Prueba de la funcionalidad de las interfaces de los diferentes casos
Pruebas de Regresión o Sistema	Verificación total del sistema.	Desarrollador	Todos los módulos en conjunto.
Pruebas de Aceptación	Validación de Requerimientos por parte del usuario.	Usuario	Todos los módulos en conjunto

Tabla 6. Plan de Pruebas

Fuente: Elaboración Propia



Figura 96. Diagrama de Validación del Plan de Pruebas
 Fuente: ISTQB – Testing a través del ciclo de vida del Software (2016)

Pruebas de Unidad: Las pruebas de unidad verifican las unidades funcionales más pequeñas que componen un módulo, por lo tanto, este tipo de pruebas se realizan a nivel de código, es decir que se centran en pruebas a clases y funciones, asegurando su funcionamiento adecuado. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

Pruebas de Caja Blanca: El método de Caja Blanca, es el cuál mira el código y estructura del producto o unidad que se va a probar, y emplea ese conocimiento en la realización de determinadas pruebas.

Pruebas de Caja Negra: Es el método en el cual el elemento es estudiado únicamente desde el punto de vista de las entradas que recibe y las salidas que produce, sin tener en cuenta el funcionamiento interno. Estas pruebas se realizan desde la interfaz gráfica.

Uno de los Casos de Prueba por Unidad para el Sistema de Generador de código CSS, se detalla en la siguiente tabla:

CASO DE PRUEBA: PRUEBAS EN EL INICIO DE SESIÓN DE USUARIOS	
DATOS GENERALES	
	Tipo de Prueba: Prueba de Unidad
	Técnica de Prueba: Caja Blanca, Caja Negra
	Numero de Case de Prueba: 1
Objetivo	Verificar que el inicio de sesión de un nuevo usuario, funcione correctamente de acuerdo a lo especificado en los requisitos.
Precondición	El usuario debe haberse autenticado correctamente.
Entrada	Información del Usuario
Salida	La lista de Usuario se actualiza con el nuevo inicio de sesión introducido.
Descripción del Problema	Existen Campos que deben introducirse de manera obligatoria, pero existen campos vacíos en algunos inicios de sesiones.
Solución	Realizar validaciones en todos los campos del inicio de sesión de usuarios
Conclusión	Cada vez que se comete un error a la hora de llenar algún campo en el inicio de sesión (correo electrónico, contraseña y campos vacíos). El sistema oportunamente muestra un mensaje de error al usuario.

Tabla 7. Prueba de unidad: Inicio de Sesión de Usuario

Fuente: Elaboración Propia

Prueba de Integración o Componentes: Una prueba de integración es aquella que se encarga de probar los métodos o funcionalidad de un componente cuando se encuentra trabajando con otros componentes de manera combinada. Se suelen ejecutar después de las pruebas unitarias.

Prueba de Regresión o Sistema: Una prueba de sistema o regresión se encarga de verificar el funcionamiento del producto completo, es decir que se trata de una prueba general de todo el software.

OBJETIVO DE LA PRUEBA		PROBAR QUE EL USUARIO PUEDA VER LA VISTA RESULTANTE DEL CÓDIGO HTML Y CSS EN EL VISUALIZADOR
Técnica		Generar al menos 10 Estilos CSS introduciendo código HTML Para que se pueda ver la vista resultante en el Visualizador.
Criterios de realización		Verificar que un usuario pueda introducir y editar código HTML y CSS correctamente en tiempo real y observar la vista resultante.
Consideraciones Especiales		Ninguna

Tabla 8. Prueba de Integracion: Visualizador
Fuente: Elaboracion Propia

Pruebas de Aceptación: El producto completo es probado por los usuarios finales o los representantes para determinar la preparación para el despliegue. Esta prueba permite probar los resultados de salida del sistema ante diferentes entradas, con la prueba de caja negra se intenta encontrar errores como ser: errores de interfaz, errores de rendimiento, errores de acceso a la BD.

CASO DE USO: GESTIONAR HISTORIAL				
Resultado: P = Paso, F = Fallo				
Nro.	PRUEBA	RESULTADOS	P	F
1	Generar nuevo Estilo CSS	Se Genero exitosamente	OK	-
2	Listar Historial	Todos los Estilos CSS fueron listados	OK	-
3	Modificar Estilo CSS	Se modiflico el Estilo CSS exitosamente	OK	-
4	Eliminar Estilo CSS	Se elimino correctamente el Estilo CSS	OK	-

Tabla 9. Prueba de Aceptacion: Caso de Uso : Gestionar Historial
Fuente: Elaboracion Propia

Según las pruebas realizadas, se puede concluir que el porcentaje de error es mínimo, por lo tanto, aceptable.

5.4.2 Cronograma de Ejecución

Tomando en cuenta la metodología XP (eXtreme Programming), que se centra en una serie de prácticas de desarrollo de software específicas, las cuales presentaran un incremento en el producto al finalizar las mismas. Para ello, se consideró la planificación del proyecto según lo establecido en la **Figura 27: Diagrama de Gantt**

CAPÍTULO VI

ANÁLISIS DE RESULTADOS

6.1. Presentación de Resultados

6.1.1 Validaciones Orientadas al Cumplimiento de Objetivos

La validación del cumplimiento de los objetivos planteados en el CAPITULO I: INTRODUCCIÓN, se lo realizará a través de capturas de pantallas del sistema, que permitirán demostrar que los objetivos se han cumplido de manera adecuada.

- Desarrollar una barra de navegación para navegar por el sistema hacia el Visualizador, Historial, Inicio de Sesión.

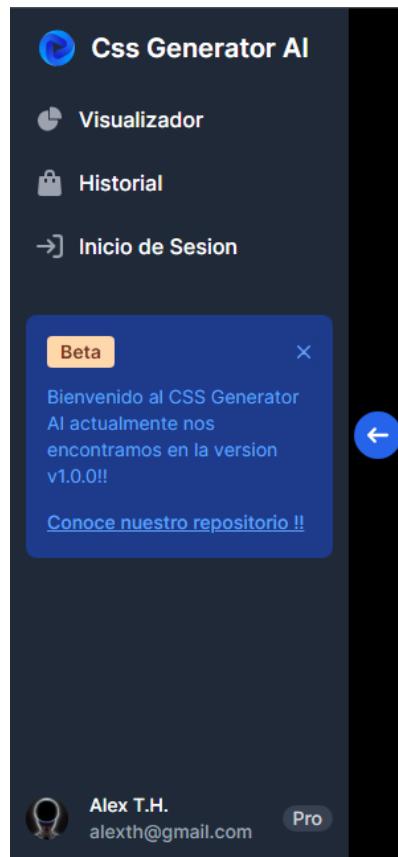


Figura 97. Barra de Navegacion
Fuente: Elaboracion Propia

- Desarrollar el Modelo de Generador de Código CSS para la aplicación web, que permita al usuario mediante el visualizador generar código CSS y visualizar su resultado.

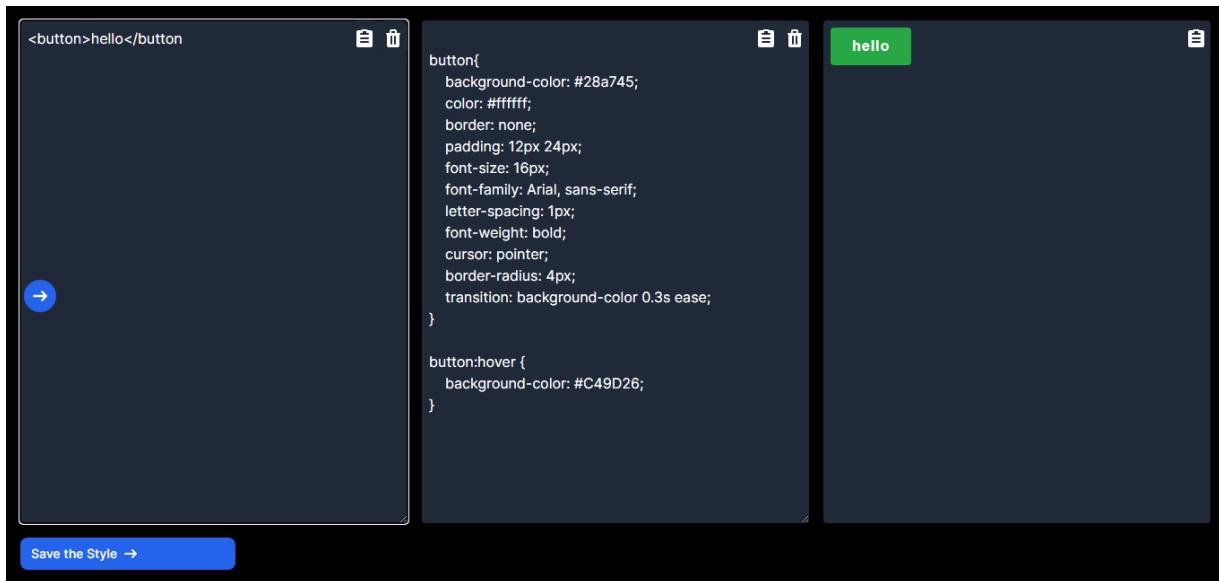


Figura 98. Visualizador de código HTML, CSS y Vista resultante

Fuente: Elaboración Propia

- Desarrollar un inicio de sesión para que el usuario pueda hacer login en el sistema generador de código CSS.

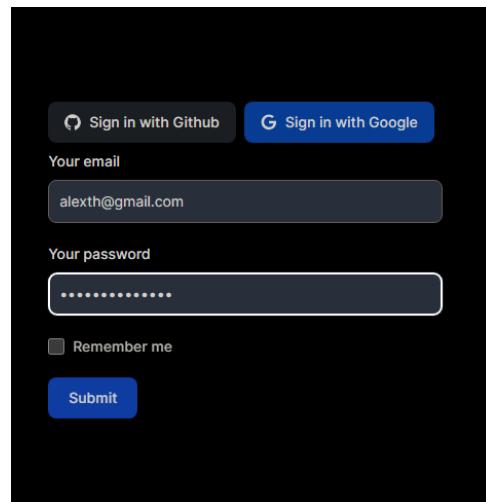


Figura 99. Inicio de Sesión

Fuente: Elaboración Propia

- Desarrollar un historial de estilos CSS generados para que el usuario pueda acceder al contenido anteriormente generado.

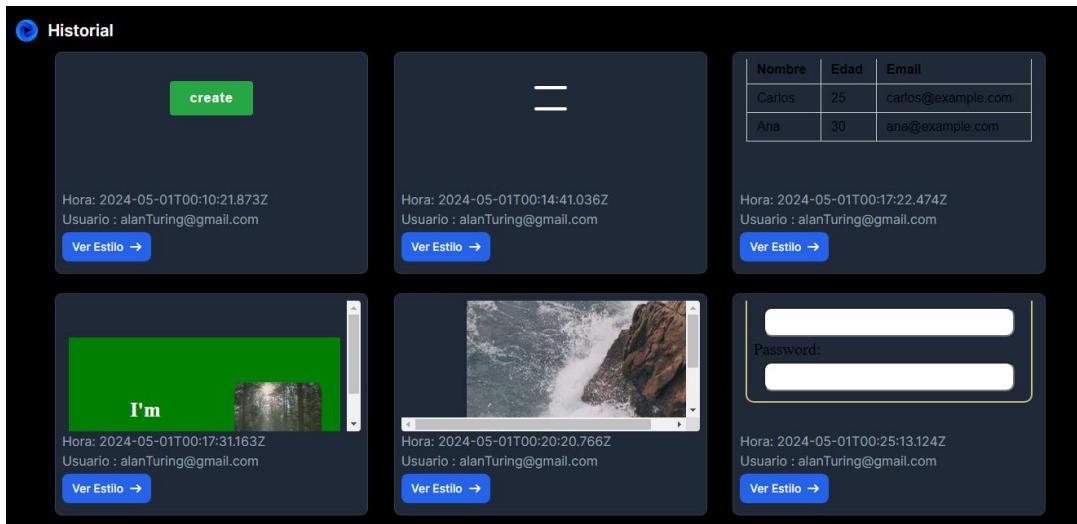


Figura 100. Historial de Estilos Generados
Fuente: Elaboracion Propria

6.2. Plan de Puesta en Marcha

La puesta en marcha, se refiere a las acciones requeridas para que el producto de software pueda ser utilizado por los usuarios finales. Por lo tanto, se tiene que verificar que el software funcione correctamente y responda adecuadamente a las especificaciones.

6.2.1 Hardware y Software

Se describen las características mínimas que deben poseer los dispositivos finales donde se utilizará el sistema.

6.2.1.1 Hardware

COMPUTADORA	
DESCRIPCIÓN	REQUISITOS MÍNIMOS
RAM	1 GB o superior
DISCO DURO	100 GB o superior
TARJETA DE VIDEO	128 MB o superior
INTERNET	2 Mbps o superior

Tabla 10. Requisitos mínimos de Hardware

Fuente: Elaboracion Propia

6.2.1.2 Software

COMPUTADORA	
SISTEMA OPERATIVO	Microsoft Windows 10 o versiones superiores. Todas las Distribuciones de Linux. MacOS
NAVEGADOR WEB	Mozilla, Google Chrome, Edge ó Safari

Tabla 11. Requisitos mínimos de Software

Fuente: Elaboracion Propia

6.2.2 Capacitación de Usuarios

La capacitación de los usuarios del sistema tendrá como objetivo principal, lo citado en los siguientes puntos:

- Capacitar a los usuarios del sistema para que puedan iniciar sesión.
- Capacitar a los usuarios del sistema para que puedan generar código CSS adecuadamente.
- Capacitar a los usuarios para que puedan utilizar la información relacionada al código generado, historial y sus vistas correspondientes.

Por lo tanto, para poder cumplir los objetivos mencionados, se ha considerado las siguientes actividades:

- Elaboración del manual de uso del sistema.
- Reuniones grupales con los usuarios del sistema, con el fin de que puedan conocer y explorar la funcionalidad del sistema.
- Elaboración y publicación de videos tutoriales interactivos sobre el uso del sistema.

6.3. Costos

6.3.1 Costos de Desarrollo y Esfuerzo

El modelo de estimación de costos COCOMO II permite medir el costo de desarrollo de un software a través de la estimación de Puntos Objetos, el cual es un conteo de pantallas, informes y módulos de lenguajes de tercera generación desarrolladas en la aplicación. Por lo tanto, el costo total calculado del Producto entregable es aproximadamente 1661,4 \$ calculados mediante el modelo de estimación COCOMO II, el cálculo del producto se encuentra detallado en el ANEXO A del presente documento.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- Mediante un exhaustivo análisis bibliográfico, se ha obtenido un profundo conocimiento del procesamiento del lenguaje natural, lo que ha sentado las bases teóricas necesarias para el desarrollo del modelo de inteligencia artificial basado en el procesamiento del lenguaje natural para la generación de código CSS.
- Se ha diseñado un algoritmo de generación de código CSS que aprovecha los principios del procesamiento natural para traducir de manera inteligente las etiquetas HTML introducidas por el usuario en estilos CSS coherentes y semánticamente significativos.
- El algoritmo implementado ha demostrado su capacidad para reducir de manera efectiva la cantidad de código repetitivo y las líneas de código CSS generadas, lo que mejora la legibilidad, mantenibilidad y eficiencia del código resultante.
- Se ha desarrollado un prototipo de aplicación web que permite a los usuarios introducir etiquetas HTML y obtener una previsualización en tiempo real de las vistas generadas, lo que facilita el proceso de diseño y desarrollo de páginas web.
- La aplicación web cuenta con una interfaz intuitiva y fácil de usar, que permite a los usuarios interactuar de manera sencilla con el sistema y visualizar los cambios en tiempo real, sin la necesidad de conocimientos profundos en CSS.
- El proyecto ha validado con éxito la viabilidad y eficacia del enfoque propuesto para la generación de código CSS basado en el procesamiento natural, demostrando su potencial para mejorar la productividad y la calidad en el desarrollo web.
- A pesar de los logros alcanzados, existen oportunidades de mejora y expansión del proyecto, como la optimización del modelo de generación de código CSS, la incorporación de funcionalidades adicionales en la aplicación web y la exploración de nuevas aplicaciones del procesamiento del lenguaje natural en el desarrollo de software. Estas áreas representan posibles direcciones para futuras investigaciones y desarrollos.

RECOMENDACIONES

- Se recomienda mejorar el entrenamiento del modelo generador de código CSS con el fin de que pueda generalizar e inferir mejor los datos recibidos. Asimismo, se sugiere optimizar su rendimiento y eficiencia, lo que permitiría empaquetar y exportar el modelo para que otros ingenieros de software puedan utilizarlo en sus proyectos personales, ampliando así su alcance y utilidad en la comunidad de desarrollo.
- Se recomienda que las versiones futuras del software generador de código CSS permitan que, al momento de generar código CSS en la aplicación web, no solo se produzca una única alternativa de estilos, sino más bien una variedad de alternativas para estilos aplicables a la etiqueta HTML proporcionada como entrada. Esto no solo enriquecería la experiencia del usuario al ofrecer opciones más diversas, sino que también aumentaría la flexibilidad y la adaptabilidad del software a diferentes necesidades y preferencias de diseño. Además, facilitaría a los desarrolladores la selección del estilo más adecuado para sus proyectos.
- Se recomienda que en futuras versiones de la API para el software generador de código CSS se añadan más servicios para la creación posterior de nuevas funcionalidades.
- Se debe establecer políticas de uso y acceso del modelo generador de código CSS y a la aplicación web de generación de código CSS, para que no se use la tecnología de forma perjudicial.
- Se debe proteger los datos de autenticación de los usuarios de tal modo que se pueda proteger la privacidad de la información de los mismos.
- Se recomienda, replicar la iniciativa del presente proyecto en las diferentes tecnologías existentes y actuales en el momento en que se lee este proyecto.
- Se recomienda experimentar y probar diferentes modelos pre-entrenados del estado del arte actual para aplicar diversas técnicas en la expansión o creación de modelos generadores de código.

REFERENCIAS BIBLIOGRÁFICAS Y BIBLIOGRAFÍA

REFERENCIAS BIBLIOGRÁFICAS

aprendemachinelearning. 2020. aprendemachinelearning. *aprendemachinelearning*. [En línea] aprendemachinelearning, 2 de 3 de 2020. [Citado el: 20 de 4 de 2024.] <https://www.aprendemachinelearning.com/sets-de-entrenamiento-test-validacion-cruzada/>.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. 2017. Attention Is All You Need. *arxiv*. Cornell University, 2017, Vols. 15 pages, 5 figures, 1706.03762.

Author, Guest. 2020. rockcontent. *rockcontent*. [En línea] © 2013-2021 Rock Content, 12 de Abril de 2020. [Citado el: 24 de Abril de 2023.] <https://rockcontent.com/es/blog/bootstrap/>.

aws. 2022. amazon.com. *amazon.com*. [En línea] AWS, 16 de 12 de 2022. [Citado el: 3 de 9 de 2023.] <https://aws.amazon.com/es/what-is/neural-network/#:~:text=Una%20red%20neuronal%20es%20un,lo%20hace%20el%20cerebro%20hu mano..>

Bootstrap. 2023. getbootstrap. *getbootstrap*. [En línea] Bootstrap, 15 de Enero de 2023. [Citado el: 24 de Abril de 2023.] <https://getbootstrap.com/>.

Bourke, Daniel. 2024. learnpytorch. *learnpytorch.io*. [En línea] learnpytorch.io, 16 de 1 de 2024. [Citado el: 13 de 4 de 2024.] <https://www.learnpytorch.io/>.

BuiltWith. 2023. BuiltWith. *BuiltWith*. [En línea] BuiltWith® Pty Ltd, 24 de Octubre de 2023. [Citado el: 23 de Abril de 2023.] <https://builtwith.com>.

datascientest. 2023. datascientest.com. *datascientest*. [En línea] DataScientest, 10 de 8 de 2023. [Citado el: 6 de 6 de 2023.] <https://datascientest.com/es/inteligencia-artificial-definicion>.

Doshi, Ketan. 2021. towardsdatascience. *towardsdatascience.com*. [En línea] towardsdatascience.com, 17 de 1 de 2021. [Citado el: 12 de 1 de 2024.] <https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853>.

ecured.cu. 2022. ecured. *ecured.cu*. [En línea] ecured.cu, 2022. [Citado el: 12 de 3 de 2024.] https://www.ecured.cu/Programaci%C3%B3n_orientada_a_componentes.

Galvan, Pedro. 2021. sg.com.mx. *sg.com.mx*. [En línea] Software Guru, 15 de 10 de 2021. [Citado el: 4 de 9 de 2023.] <https://sg.com.mx/revista/56/inteligencia-artificial-desarrollo-software>.

Google. 2024. <https://research.google.com>. *https://research.google.com*. [En línea] google research, 5 de 3 de 2024. [Citado el: 15 de 4 de 2024.] <https://research.google.com/collaboratory/intl/es/faq.html>. 234.

Hugging Face. 2024. *huggingface. huggingface.co.* [En línea] *huggingface.co*, 2024. [Citado el: 14 de 3 de 2024.] <https://huggingface.co/openai-community/gpt2-xl>.

HuggingFace. 2020. *huggingface. huggingface.co.* [En línea] *huggingface.co*, 2020. [Citado el: 16 de 3 de 2024.] https://huggingface.co/transformers/v3.0.2/pretrained_models.html.

IBM. 2023. *ibm.com. [ibm.com](https://www.ibm.com/mx-es/topics/artificial-intelligence).* [En línea] IBM, 15 de 9 de 2023. [Citado el: 4 de 5 de 2024.] <https://www.ibm.com/mx-es/topics/artificial-intelligence>. 234.

IDC. 2023. IDC. *IDC.* [En línea] IDC Corporate, 2023. [Citado el: 23 de 8 de 2023.] <https://www.idc.com/getdoc.jsp?containerId=prUS49670122>.

Kosar, Vaclav. 2022. *vaclavkosar.com. [vaclavkosar.com](https://vaclavkosar.com/ml/Tokenization-in-Machine-Learning-Explained).* [En línea] *vaclavkosar.com*, 16 de 9 de 2022. [Citado el: 19 de 4 de 2024.] <https://vaclavkosar.com/ml/Tokenization-in-Machine-Learning-Explained>.

—. 2023. *vaclavkosar.com. [vaclavkosar.com](https://vaclavkosar.com/ml/Embeddings-in-Machine-Learning-Explained).* [En línea] *vaclavkosar.com*, 11 de 9 de 2023. [Citado el: 20 de 4 de 2024.] <https://vaclavkosar.com/ml/Embeddings-in-Machine-Learning-Explained>.

Kumar, Ajitesh. 2024. *vitalflux. [vitalflux.com](https://vitalflux.com/transfer-learning-vs-fine-tuning-differences/).* [En línea] *vitalflux.com*, 23 de 1 de 2024. [Citado el: 18 de 4 de 2024.] <https://vitalflux.com/transfer-learning-vs-fine-tuning-differences/>.

Language Models are Unsupervised Multitask Learners. Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei. 2019. 2019.

López, José María. 2019. *blogthinkbig. [blogthinkbig](https://blogthinkbig.com/generadores-css-escribir-codigo).* [En línea] *blogthinkbig*, 2019. [Citado el: 16 de 3 de 2024.] <https://blogthinkbig.com/generadores-css-escribir-codigo>.

—. 2022. <https://hipertextual.com>. *Hipertextual.* [En línea] Hipertextual SL, 18 de 6 de 2022. [Citado el: 4 de 3 de 2024.] <https://hipertextual.com/2022/06/movimiento-no-code-sin-codigo-234234>.

marketing-branding. 2023. *marketing-branding. [marketing-branding](https://www.marketing-branding.com/chat-gpt-inteligencia-artificial-que-es-seo-friendly/).* [En línea] Marketing Branding Blog, 12 de Febrero de 2023. [Citado el: 13 de Junio de 2023.] <https://www.marketing-branding.com/chat-gpt-inteligencia-artificial-que-es-seo-friendly/>. 234.

Matich, Damián Jorge. 2001.

https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/mati-ch-redesneuronales.pdf.

https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/mati-ch-redesneuronales.pdf. [En línea] Universidad Tecnológica Nacional – Facultad Regional Rosario, 20 de 3 de 2001. [Citado el: 4 de 5 de 2024.]

https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/mati-ch-redesneuronales.pdf.

mdn. 2023. <https://developer.mozilla.org>. *MDN Web Docs*. [En línea] Mozilla, 13 de 11 de 2023. [Citado el: 4 de 3 de 2024.]

https://developer.mozilla.org/es/docs/Glossary/CSS_preprocessor. 34r234.

MongoDB. 2023. <https://www.mongodb.com>. <https://www.mongodb.com>. [En línea]

<https://www.mongodb.com>, 2023. [Citado el: 12 de 5 de 2024.]

<https://www.mongodb.com/docs/>. 23.

Nation, SlashData's Developer. 2023. Developer Nation. *Developer Nation*. [En línea]

Developer Nation, 2023. [Citado el: 23 de 8 de 2023.]

<https://www.developernation.net/developer-reports/de20>.

nextjs.org. 2024. nextjs. *nextjs.org*. [En línea] nextjs.org, 16 de 2 de 2024. [Citado el: 20 de 4 de 2024.] <https://nextjs.org/docs>.

Nvidia. 2023. <https://la.blogs.nvidia.com>. <https://la.blogs.nvidia.com>. [En línea] Nvidia, 22 de 6 de 2023. [Citado el: 23 de 8 de 2023.] <https://la.blogs.nvidia.com/2023/06/22/que-son-los-modelos-de-base/>.

paperswithcode. 2023. paperswithcode. *paperswithcode.com*. [En línea] paperswithcode.com, 2023. [Citado el: 15 de 3 de 2024.] <https://paperswithcode.com/dataset/webtext>.

Peiró, Rosario. 2020. economipedia.com. *economipedia.com*. [En línea] Economipedia, 1 de 8 de 2020. [Citado el: 5 de 9 de 2023.] <https://economipedia.com/definiciones/lenguaje-css.html#:~:text=El%20lenguaje%20CSS%20es%20un,el%20formato%20de%20p%C3%A1ginas%20web..>

Postman. 2024. <https://learning.postman.com>. <https://learning.postman.com>. [En línea] <https://learning.postman.com>, 2024. [Citado el: 12 de 4 de 2024.] <https://learning.postman.com/docs/introduction/overview/>. 56.

pytorch.org. 2024. pytorch. *pytorch.org*. [En línea] pytorch.org, 15 de 3 de 2024. [Citado el: 18 de 4 de 2024.] <https://pytorch.org/docs/stable/index.html>.

Raeburn, Alicia. 2022. Asana. *Asana*. [En línea] Asana Inc, 28 de 11 de 2022. [Citado el: 4 de 3 de 2024.] <https://asana.com/es/resources/extreme-programming-xp>. 23423.

Rathinapandi, Geoffrey. 2023. medium. *medium.com*. [En línea] medium.com, 30 de 10 de 2023. [Citado el: 22 de 4 de 2024.] <https://geoffrey-geofe.medium.com/tokenization-vs-embedding-understanding-the-differences-and-their-importance-in-nlp-b62718b5964a>.

SANTOS, PALOMA RECUERO DE LOS. 2022. Telefonica Tech. *Telefonica Tech*. [En línea] Telefonica Tech, 24 de 1 de 2022. [Citado el: 5 de 2 de 2024.] <https://telefonicatech.com/blog/datos-entrenamiento-vs-datos-de-test>.

Solis, Diego Caceres. 2023. openwebinars.net. *openwebinars*. [En línea] openwebinars.net, 1 de 5 de 2023. [Citado el: 15 de 4 de 2024.] <https://openwebinars.net/blog/datasets-que-son-y-como-acceder-a-ellos/>.

Sotaquirá, Miguel. 2020. codificandobits. *codificandobits*. [En línea] © 2023 Codificando Bits, 30 de Junio de 2020. [Citado el: 24 de Abril de 2023.]
<https://www.codificandobits.com/blog/redes-transformer/#la-red-transformer>.

—. 2023. www.codificandobits.com. *codificandobits*. [En línea] codificandobits, 16 de 6 de 2023. [Citado el: 2 de 2 de 2024.] <https://www.codificandobits.com/blog/sets-entrenamiento-validacion-y-prueba/>.

StackOverflow. 2023. StackOverflow. *StackOverflow*. [En línea] Stackoverflow, 2023. [Citado el: 23 de 8 de 2023.] <https://stackoverflow.com/>.

Statcounter. 2023. Statcounter. *Statcounter*. [En línea] Statcounter, 2023. [Citado el: 23 de 8 de 2023.] <https://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-202208-202208-bar>.

Torres, Moisés Barrios. 2023. pangeanic. *blog.pangeanic.com*. [En línea] [blog.pangeanic.com](https://blog.pangeanic.com/es/que-es-fine-tuning), 18 de 4 de 2023. [Citado el: 25 de 4 de 2024.]
<https://blog.pangeanic.com/es/que-es-fine-tuning>.

UNIR. 2023. unir. *www.unir.net*. [En línea] www.unir.net, 2 de 11 de 2023. [Citado el: 7 de 4 de 2024.] <https://www.unir.net/ingenieria/revista/transfer-learning>.

VS Code. 2023. <https://code.visualstudio.com>. *https://code.visualstudio.com*. [En línea] Microsoft, 10 de 2023. [Citado el: 10 de 5 de 2024.] <https://code.visualstudio.com/docs>. 43.

BIBLIOGRAFÍA

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. arXiv preprint arXiv:1706.03762, 2017.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever. Language Models are Unsupervised Multitask Learners. OpenAI blog, 2019.
- Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122v2, 2017.
- Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.
- Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. arXiv preprint arXiv:1610.10099v2, 2017.
- Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130, 2017.
- GIRALDO, O.P. (2007). Ingeniería de Requisitos. Volumen, 13.
- SOMERVILLE, IAN. Ingeniería de Software. 7ma Edición España-Madrid 2005
- GÓMEZ A. LÓPEZ M. MIGANI S. OTAZÚ A. COCOMO, Un modelo de estimación de proyectos de Software. (2018). Página 32.
- Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. arXiv preprint arXiv:1511.06114, 2015.
- Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attentionbased neural machine translation. arXiv preprint arXiv:1508.04025, 2015.

GLOSARIO

GLOSARIO DE TÉRMINOS

CSS: Cascade Style Sheets, en español «Hojas de estilo en cascada», es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML o XHTML

HTML: HyperText Markup Language ('lenguaje de marcas de hipertexto'), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros.

UML: El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento.

Framework: Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

Codificación: Se llama codificación a la transformación de la formulación de un mensaje a través de las reglas o normas de un código o lenguaje predeterminado. **Interfaz:** es un término que procede del vocablo inglés interface. En informática, esta noción sirve para señalar a la conexión que se da de manera física y a nivel de utilidad entre dispositivos o sistemas.

Inteligencia Artificial (IA): Campo de la informática que se centra en el desarrollo de sistemas y máquinas capaces de realizar tareas que requieren inteligencia humana.

Modelo de Inteligencia Artificial: Una representación computacional o matemática de un sistema o proceso que utiliza algoritmos y datos para resolver problemas o tomar decisiones.

Transformer: Una arquitectura de modelo de aprendizaje automático basada en atención que se utiliza principalmente en tareas de procesamiento de lenguaje natural y que ha demostrado ser eficaz en una variedad de aplicaciones.

Machine Learning: (Aprendizaje Automático) Subcampo de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender patrones a partir de datos y mejorar su rendimiento con la experiencia, sin ser explícitamente programadas.

Deep Learning: (Aprendizaje Profundo) Subcampo del aprendizaje automático que se centra en el uso de redes neuronales profundas, que son modelos con múltiples capas ocultas, para aprender patrones complejos y realizar tareas sofisticadas como reconocimiento de imágenes, procesamiento de lenguaje natural, entre otros.

Natural Language Processing (NLP): (Procesamiento de Lenguaje Natural) Campo de la inteligencia artificial que se centra en la interacción entre las computadoras y el lenguaje humano, permitiendo a las máquinas entender, interpretar y generar lenguaje humano de manera natural.

Fine Tuning: (Ajuste Fino) Proceso de ajuste de un modelo de aprendizaje automático preentrenado en un conjunto de datos específico para mejorar su rendimiento en una tarea particular.

Transfer Learning: (Transferencia de Aprendizaje) En aprendizaje automático, es el proceso de transferir el conocimiento adquirido al entrenar un modelo en una tarea a otra tarea relacionada, generalmente utilizando un modelo preentrenado como punto de partida.

Redes Neuronales: Modelos computacionales inspirados en el funcionamiento del cerebro humano, compuestos por neuronas artificiales interconectadas que se utilizan en diversos campos del aprendizaje automático y la inteligencia artificial.

Layers: Capas en una red neuronal, que son unidades estructurales que procesan y transforman los datos a medida que pasan a través de la red.

Heads: (Cabezas) Componentes de una capa de atención en modelos de atención múltiple, que se utilizan para calcular diferentes aspectos de la atención en paralelo.

Multi-Head Attention: (Atención Multicabezal) Una forma de atención en modelos de Transformer donde se calculan múltiples funciones de atención simultáneamente, lo que permite al modelo enfocarse en diferentes partes de la entrada de manera más efectiva.

Attention: (Atención) Mecanismo utilizado en redes neuronales que asigna ponderaciones a diferentes partes de una secuencia de entrada para resaltar la importancia relativa de cada elemento en la tarea que se está realizando.

Tokenization: (Tokenización) Proceso de dividir un texto en unidades más pequeñas, llamadas tokens, como palabras, caracteres o subpalabras, para su posterior procesamiento.

Embedding: (Incrustación) Representación numérica de un elemento, como una palabra o un documento, en un espacio vectorial de baja dimensión, que captura su significado semántico.

Parameters: (Parámetros) Variables ajustables en un modelo de aprendizaje automático que se optimizan durante el entrenamiento para minimizar una función de pérdida.

Dataset: (Conjunto de Datos) Colección de datos estructurados que se utiliza para entrenar, validar o probar modelos de aprendizaje automático.

GPT - Generative Pre-trained Transformer: (Transformer Preentrenado Generativo) Un modelo de lenguaje basado en Transformer preentrenado que puede generar texto de manera autónoma.

GPT-2 XL - Generative Pre-trained Transformer 2 Extra Large: Una versión específica de GPT-2 con mayor capacidad y número de parámetros.

Training Dataset: (Conjunto de Datos de Entrenamiento) Subconjunto de datos utilizado para entrenar un modelo de aprendizaje automático.

Validation Dataset: (Conjunto de Datos de Validación) Subconjunto de datos utilizado para ajustar los hiperparámetros de un modelo y evaluar su rendimiento durante el entrenamiento.

Test Dataset: (Conjunto de Datos de Prueba) Subconjunto de datos utilizado para evaluar el rendimiento final de un modelo después de haber sido entrenado y validado.

LLM: (Lenguaje de Modelado de Largo) Modelo de lenguaje diseñado para generar texto coherente y de alta calidad basado en contextos largos.

PyTorch: Una biblioteca de aprendizaje profundo de código abierto para Python, desarrollada principalmente por Facebook's AI Research lab (FAIR), que proporciona herramientas para la creación y entrenamiento de modelos de aprendizaje automático.

NextJS: Un framework de desarrollo web de código abierto basado en React que permite la creación de aplicaciones web y sitios estáticos de manera eficiente.

Programación orientada a componentes: Paradigma de programación que se centra en la construcción de sistemas de software a partir de componentes reutilizables e independientes.

Colab: (Colaboratory) Plataforma en línea de Google que permite escribir y ejecutar código de Python en un entorno de notebook interactivo, especialmente diseñado para el aprendizaje automático.

Visual Studio Code: Un editor de código fuente desarrollado por Microsoft que ofrece características avanzadas de edición, depuración y extensibilidad para una amplia variedad de lenguajes de programación.

MongoDB: Una base de datos NoSQL de código abierto, orientada a documentos, que utiliza un modelo de datos flexible y almacenamiento tipo JSON para la gestión de datos.

Postman: Una plataforma de colaboración para el desarrollo de APIs que permite a los desarrolladores diseñar, probar y depurar APIs de manera eficiente.

API: (Interfaz de Programación de Aplicaciones) Conjunto de reglas y protocolos que permiten a diferentes componentes de software comunicarse entre sí.

CRUD: (Create, Read, Update, Delete) Operaciones básicas para la gestión de datos en bases de datos, que incluyen la creación, lectura, actualización y eliminación de registros.

HuggingFace: Una empresa y comunidad de código abierto conocida por sus contribuciones en el campo del procesamiento del lenguaje natural (NLP), incluyendo bibliotecas de aprendizaje automático y modelos preentrenados.

Python: Un lenguaje de programación de alto nivel, interpretado y de propósito general, conocido por su sintaxis clara y legible. Python es ampliamente utilizado en desarrollo web, análisis de datos, inteligencia artificial, entre otros campos.

React: Una biblioteca de JavaScript de código abierto desarrollada por Facebook que se utiliza para construir interfaces de usuario interactivas y reactivas en aplicaciones web. React utiliza un enfoque basado en componentes para la construcción de interfaces de usuario.

ANEXOS

ANEXO A: ESTIMACIÓN DEL COSTO DEL SOFTWARE DESARROLLADO

Una vez definido el nivel de complejidad de las mismas (simple, medio y alto), se podrá obtener un valor total, que permitirá realizar el cálculo final de esfuerzo. Se puede observar en la siguiente Tabla. con la cantidad de puntos objeto asignados a cada módulo.

Nombre de Pantalla	N de Vistas	Complejidad	Valor	TOTAL
Barra de navegación	3	Simple	1	10
Inicio de sesión	5	Medio	2	13
Historial	5	Complejo	2	20
Visualizador	10	Complejo	3	30
			Puntos Objeto	73

Tabla 12. Calculo de Puntos Objeto

Fuente: Elaboracion Propia

Calculo Puntos Objeto del Sistema

%REUSE: Porcentaje de pantallas y módulos reutilizables es igual a 65%

$$NOP = \frac{(Puntos\ Objeto) * (100 - \%REUSE)}{100}$$

$$NOP = \frac{73 * (100 - 65)}{100} = 25,6$$

Cálculo del valor Persona – Mes estimado (Esfuerzo)

PROD: Experiencia y capacidad del desarrollador (Bajo, 7)

$$PM = \frac{NOP}{PROD} = \frac{25,6}{7} = 3,65$$

Esfuerzo

El valor del esfuerzo en la realización de la aplicación es:

$$PM = 3.65$$

Costo de Desarrollo Sueldo:

El sueldo promedio de un programador junior en Bolivia es similar a 3000 Bs.

$$Costo Prod = Sueldo * PM$$

$$Costo Prod = \frac{3000 \text{ Bs}}{PM} * 3,65 \text{ PM} = 10950 \text{ Bs.}$$

Costo de Aplicación de la solución

Los siguientes gastos se efectuaron durante el desarrollo del proyecto presentado:

ITEM	CANTIDAD	PRECIO/UNIDAD (Bs.)	TOTAL
Papel bond Tamaño Carta	1	30	30
Conexión a Internet Fibra Óptica	1	250	250
Teléfono móvil gama Baja	1	400	400
		Total (Bs.)	680
		Total (\$us.)	98,38

Tabla 13. Costos de material de escritorio, equipos y conectividad

Fuente: Elaboracion Propia

$$Costo Total = Costos Producto + Total Costos Hardware$$

$$Costo Total = 10950 + 680 = 11630 \text{ Bs.}$$

$$Costo Total = 11630 \text{ Bs.}$$

$$Costo Total = 1661,4 \text{ $us}$$

Por lo tanto, el costo total del Producto entregable es aproximadamente 1661,4 \$ calculados mediante el modelo de estimación COCOMO II.

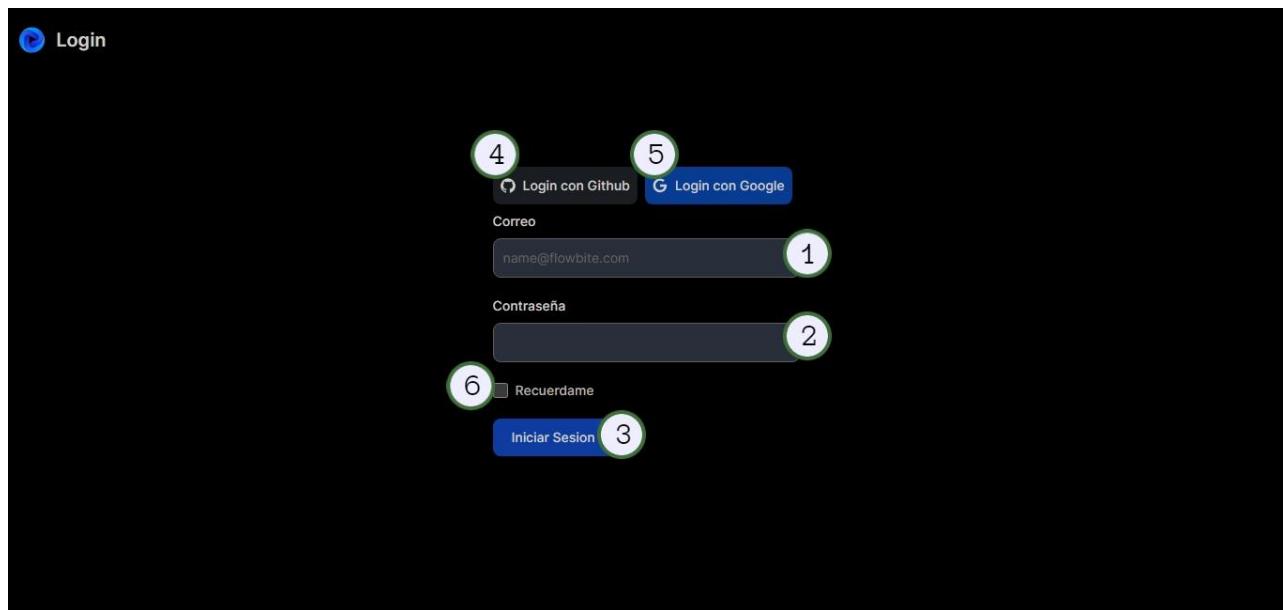
ANEXO B: MANUAL DE USUARIO

El Sistema de Generación de código CSS posee una aplicación web para generación de código CSS en tiempo real.

El presente manual detalla el funcionamiento de la aplicación web.

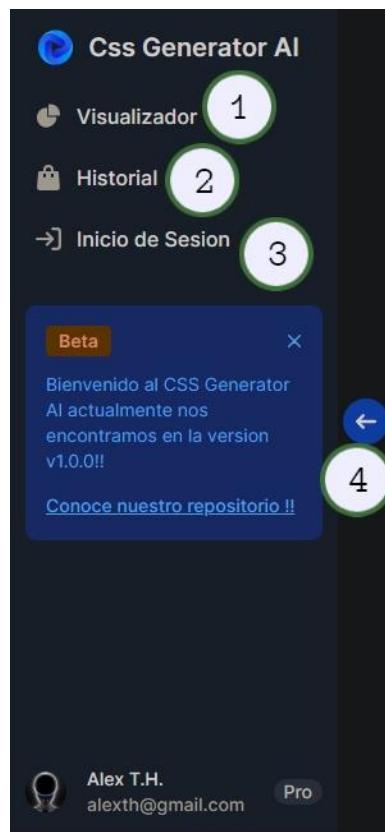
APLICACIÓN WEB

Pantalla de Inicio de Sesión



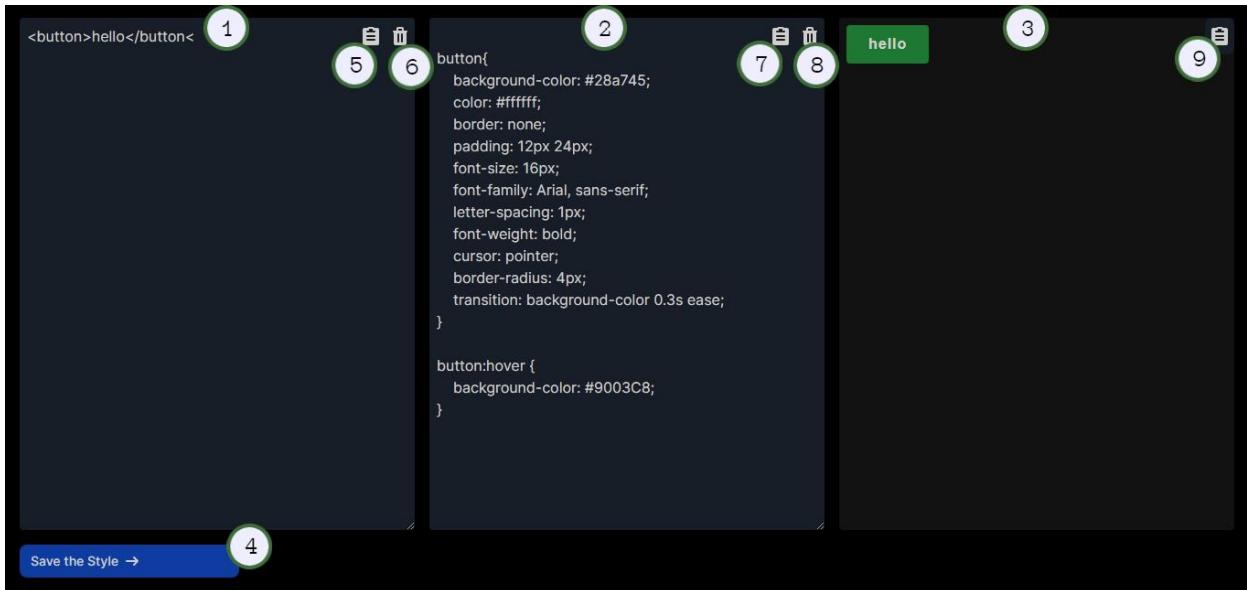
1. Introducir Correo
2. Introducir Contraseña
3. Presionar en el botón “Iniciar Sesión” para acceder al Sistema
4. Presionar en el botón “Login con Github” para acceder al Sistema
5. Presionar en el botón “Login con Google” para acceder al Sistema
6. Presionar en el checkbox “Recuérdame” para se guarde la sesión.

Pantalla de Barra de Navegación



1. Presionar en el Link “Visualizador” para acceder al Visualizador
2. Presionar en el Link “Historial” para acceder al Historial
3. Presionar en el Link “Inicio de Sesión” para acceder al Inicio de Sesión
4. Presionar el botón “Flecha” para ocultar y mostrar la Barra de Navegación

Pantalla de Visualizador



1. Escribir código HTML en el TextArea para genera código CSS.
2. TextArea para generar y editar el código CSS.
3. IFrame con Vista resultante del código HTML y CSS.
4. Presionar el botón “Save the Style” para guardar el código HTML y CSS en el Historial.
5. Presionar el botón “Copiar” para copiar el código HTML del TextArea.
6. Presionar el botón “Eliminar” para eliminar el código HTML del TextArea.
7. Presionar el botón “Copiar” para copiar el código CSS del TextArea.
8. Presionar el botón “Eliminar” para eliminar el código CSS del TextArea.
9. Presionar el botón “Copiar” para copiar el código HTML y CSS del IFrame.

Pantalla de Historial

The screenshot shows a dark-themed application interface titled "Historial". At the top left is a "create" button. Below it, there are five items listed:

- 1**: A card with a green header containing the word "create". It displays the creation time and user information: "Hora: 2024-05-01T00:10:21.873Z" and "Usuario : alanTuring@gmail.com". A "Copiar Estilo" button with a copy icon is at the bottom right.
- 2**: A card with a green header containing the letters "I'm". It displays the creation time and user information: "Hora: 2024-05-01T00:17:31.163Z" and "Usuario : alanTuring@gmail.com". A "Copiar Estilo" button with a copy icon is at the bottom right.
- A card showing a small thumbnail image of a waterfall.
- A card showing a small thumbnail image of a table with data.
- A card showing a table with three columns: "Nombre", "Edad", and "Email". The data rows are: "Carlos" (25, carlos@example.com), "Ana" (30, ana@example.com).

1. Tarjeta de vista de estilos e información del Usuario y Hora de Creación del Estilo.
2. Presionar el botón “Copiar Estilo” para Copiar el Código HTML y CSS generados.