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 PARA GENERACIÓN DE CÓDIGO CSS**

Universitario: Alex Tumiri Huanca

Docente: Ing. Oswaldo Velásquez Aroni

Sucre, - de 2024

ÍNDICE DE CONTENIDO

[CAPÍTULO I 1](#_Toc164273169)

[INTRODUCCIÓN 1](#_Toc164273170)

[1.1 Antecedentes 1](#_Toc164273171)

[1.2 Situación Problemática 3](#_Toc164273172)

[1.3 Problema Central 5](#_Toc164273173)

[1.4 Abordaje de la Solución 5](#_Toc164273174)

[1.5 Objetivos 7](#_Toc164273175)

[**1.5.1 Objetivo General** 7](#_Toc164273176)

[**1.5.2 Objetivos Específicos** 7](#_Toc164273177)

[1.6 Justificación 8](#_Toc164273178)

[**1.6.1 Justificación Tecnológica** 8](#_Toc164273179)

[CAPÍTULO II 9](#_Toc164273180)

[MARCO CONTEXTUAL 9](#_Toc164273181)

[2.1. Análisis de la Situación Actual 9](#_Toc164273182)

[**2.1.1 Antecedentes Generales** 9](#_Toc164273183)

[**2.1.2 Descripción de Tipos de Desarrolladores de Software** 9](#_Toc164273184)

[CAPÍTULO III 1](#_Toc164273185)

[FUNDAMENTO TEÓRICO 1](#_Toc164273186)

[3.1. Antecedente Teórico 1](#_Toc164273187)

[**3.1.1 Impacto de la Inteligencia Artificial en el Desarrollo de Software** 1](#_Toc164273188)

[**3.1.2 Modelos Generadores de textos** 2](#_Toc164273189)

[**3.1.3 Movimiento no Code** 4](#_Toc164273190)

[3.2. Marco Teórico del contexto 5](#_Toc164273191)

[**3.2.1 Lenguaje Css** 5](#_Toc164273192)

[**3.2.2 La Inteligencia Artificial y sus áreas de aplicación** 7](#_Toc164273193)

[**3.2.3 Redes Neuronales** 9](#_Toc164273194)

* [**¿Cuáles son los tipos de redes neuronales?** 14](#_Toc164273195)

[**3.2.4 Transformers** 15](#_Toc164273196)

[**3.2.5 Red Transformer** 16](#_Toc164273197)

[3.3. Marco Teórico de Ingeniería 29](#_Toc164273198)

[**3.3.1 Metodología de desarrollo** 29](#_Toc164273199)

* [**¿Qué es la programación extrema (XP)?** 29](#_Toc164273200)
* [**Ciclo de vida de la programación extrema (XP)** 30](#_Toc164273201)
* [**¿Cuáles son las 12 prácticas de la programación extrema?** 30](#_Toc164273202)

[**3.3.2 Pytorch** 32](#_Toc164273203)

[**3.3.3 React para el desarrollo web** 34](#_Toc164273204)

[**3.3.4 Paradigma de Programación** 34](#_Toc164273205)

[**3.3.5 Colab** 35](#_Toc164273206)

[**3.3.6 Visual Studio Code** 36](#_Toc164273207)

[CAPÍTULO IV 37](#_Toc164273208)

[METODOLOGÍA APLICADA AL PROYECTO 37](#_Toc164273209)

[4.1. Metodología de la Investigación 37](#_Toc164273210)

[**4.1.1 Metodología General** 37](#_Toc164273211)

[**4.1.2 Métodos, Técnicas y Herramientas** 37](#_Toc164273212)

[4.2. Metodología de Ingeniería 38](#_Toc164273213)

[**4.2.1 Desarrollo del Proyecto** 38](#_Toc164273214)

[**4.2.2 Planificación del Proyecto** 39](#_Toc164273215)

[CAPÍTULO V 40](#_Toc164273216)

[INGENIERÍA DEL PROYECTO 40](#_Toc164273217)

[5.1. Fase de Exploración: Análisis Preliminar del Proyecto 40](#_Toc164273218)

[**5.1.1 Proceso de Requerimientos** 40](#_Toc164273219)

* [**5.1.1.1 Identificación de Actores** 40](#_Toc164273220)
* [**5.1.1.2 Descripción de Actores** 41](#_Toc164273221)
* [**5.1.1.3 Requerimientos Funcionales** 42](#_Toc164273222)
* [**5.1.1.4 Requerimientos no Funcionales** 44](#_Toc164273223)

[5.2. Fase de Elaboración: Definición de la Arquitectura del Sistema 45](#_Toc164273224)

[**5.2.1 Diagrama de Casos de Uso General** 45](#_Toc164273225)

[**5.2.2 Diagrama de Casos de Uso Clasificación por prioridad** 46](#_Toc164273226)

[**5.2.3 Diagrama de Casos de Uso Clasificados por funcionalidad** 47](#_Toc164273227)

[**5.2.4 Descripción de los Casos de Uso** 48](#_Toc164273228)

[**5.2.5 Proceso de Análisis y Diseño** 50](#_Toc164273229)

[**5.2.6 Análisis de Riesgos** 50](#_Toc164273230)

[**5.2.7 Diagramas de Frontera del Modelo** 53](#_Toc164273231)

* [**5.2.7.1 Diagrama Frontera del Administrador** 53](#_Toc164273232)
* [**5.2.7.2 Diagrama Frontera del Desarrollador Web** 53](#_Toc164273233)
* [**5.2.7.3 Diagrama Frontera del Desarrollador Frontend** 54](#_Toc164273234)

[**5.2.8 Diagrama de Componentes del Sistema** 54](#_Toc164273235)

[5.3. Fase de Desarrollo 56](#_Toc164273236)

[**5.3.1 Datasets de Entrenamiento, Validación y Prueba** 56](#_Toc164273237)

* [**5.3.1.1 Diagrama de Flujo** 56](#_Toc164273238)
* [**5.3.1.2 Características del Dataset** 57](#_Toc164273239)
* [**5.3.1.3 Recolección de Datos** 58](#_Toc164273240)
* [**5.3.1.4 Procesamiento de Datos** 59](#_Toc164273241)
* [**5.3.1.5 División en conjunto de datos de entrenamiento, validación y prueba** 60](#_Toc164273242)

[**5.3.2 Modelo de Inteligencia Artificial** 63](#_Toc164273243)

* [**5.3.2.1 Diagrama de Flujo** 63](#_Toc164273244)
* [**5.3.2.2 Modelo Pre-Entrenado** 64](#_Toc164273245)
* [**5.3.2.3 Implementación del Modelo** 71](#_Toc164273246)
* [**5.3.2.4** **Tokenización e Incrustación** 79](#_Toc164273247)
* [**5.3.2.5** **Fine-Tuning** 85](#_Toc164273248)
* [**5.3.2.7 Red Transformer** 91](#_Toc164273249)
* [**5.3.2.8 Entrenamiento** 91](#_Toc164273250)

[**5.3.3 API de servicios web para Modelo** 92](#_Toc164273251)

[**5.3.4 Sistema Generador de CSS** 92](#_Toc164273252)

[5.4. Fase de Pruebas 92](#_Toc164273253)

[**5.4.1 Ajuste del Modelo** 92](#_Toc164273254)

[**5.4.2 Mantenimiento y Pruebas** 92](#_Toc164273255)

[BIBLIOGRAFÍA 93](#_Toc164273256)

**ÍNDICE DE FIGURAS**

[Figura 1. Arquitectura del software de generación css 7](#_Toc137561678)

[Figura 2. Diagrama de gantt 9](#_Toc137561679)

[Figura 3. Diagrama de casos de uso de un model de inteligencia artificial 15](#_Toc137561680)

[Figura 4. Diagrama de proceso de desarrollo de chat gpt 16](#_Toc137561681)

[Figura 5. Diagrama de bloques general de una red transformer 25](#_Toc137561682)

[Figura 6. Embedding de entrada de la red transformer 26](#_Toc137561683)

[Figura 7. Codificador de posición de la red transformer 26](#_Toc137561684)

[Figura 8. El bloque de codificación de la red transformer 27](#_Toc137561685)

[Figura 9. Detalle del primer codificador en la red transformer 28](#_Toc137561686)

[Figura 10. La relación entre cada palabra de la frase 'i love italian food' 28](#_Toc137561687)

[Figura 11. Obtención de los 'queries', 'keys' y 'values' 29](#_Toc137561688)

[Figura 12. Comparación de 'queries' y 'keys' y obtención de los puntajes 29](#_Toc137561689)

[Figura 13. Aplicación de la función 'softmax' a la matriz de puntajes 30](#_Toc137561690)

[Figura 14. La salida del bloque atencional 31](#_Toc137561691)

[Figura 15. Los múltiples bloques atencionales permiten detectar asociaciones entre palabras y grupos de palabras a diferentes niveles 32](#_Toc137561692)

[Figura 16. Los elementos del bloque residual 32](#_Toc137561693)

[Figura 17. La etapa de salida del codificador: una red neuronal seguida de un bloque residual 33](#_Toc137561694)

[Figura 18. Salida resultante del primer codificador 34](#_Toc137561695)

[Figura 19. Elementos del decodificador 34](#_Toc137561696)

[Figura 20. Momento del enmascaramiento 36](#_Toc137561697)

[Figura 21. El resultado del enmascaramiento 36](#_Toc137561698)

[Figura 22. El bloque atencional del decodificador 37](#_Toc137561699)

[Figura 23. Múltiples decodificadores para encontrar relaciones entre palabras y grupos de palabras a diferentes niveles 38](#_Toc137561700)

[Figura 24. Salida del bloque de decodificación 38](#_Toc137561701)

# **CAPÍTULO I**

# **INTRODUCCIÓN**

# **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 inabarcable e intangible de documentos, imágenes y protocolos que componen la telaraña de información que crece a pasos de gigante.

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 a 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 he 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.

# **Situación Problemática**

* Actualmente existe poca productividad por parte de los ingenieros de software a la hora de desarrollar interfaces de usuario, diseño y estilos con el lenguaje de diseño gráfico CSS. Debido a la complejidad de la implementación de un diseño web efectivo y la gran cantidad de líneas de código que esto puede requerir, muchos desarrolladores pueden enfrentarse a un proceso de desarrollo lento y tedioso este proceso consiste en 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.
* En algunas situaciones, el desarrollador de software puede tener la capacidad y conocimiento técnico para desarrollar interfaces de muy buena calidad con CSS, pero resulta que carece de habilidades de diseño necesarias para crear una interfaz de usuario atractiva y funcional. El conocimiento en creación de interfaces y la habilidad de diseño son esenciales para crear una experiencia de usuario efectiva y agradable, y esto a menudo requiere de una combinación de habilidades técnicas y creativas. Si un desarrollador de software carece de estas habilidades, puede dar como resultado en una interfaz de usuario poco atractiva, difícil de usar y menos efectiva.
* 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 tienen una curva de aprendizaje compleja 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.

Cada herramienta de CSS también tiene sus propios pros y contras, y la elección de la herramienta adecuada dependerá de las necesidades y objetivos específicos del proyecto. A continuación, se presentan algunos de los pros y contras de las herramientas de CSS más populares:

* Bootstrap es una de las herramientas de CSS más populares y ampliamente utilizadas, lo que significa que hay una gran comunidad de desarrolladores y recursos disponibles en línea. Bootstrap es fácil de usar y tiene una curva de aprendizaje corta, lo que lo hace ideal para proyectos de desarrollo web rápidos. Además, Bootstrap ofrece una gran cantidad de componentes, como botones, formularios y carruseles, que se pueden implementar fácilmente en cualquier proyecto. Debido a su popularidad, Bootstrap a menudo se utiliza para crear sitios web que se ven muy similares. Además, Bootstrap tiene una gran cantidad de clases de estilo, lo que puede aumentar el tamaño de los archivos CSS y ralentizar el rendimiento de la página.
* Por otro lado, herramientas como Tailwind CSS, Semantic UI son altamente personalizables y modulares, lo que permite a los desarrolladores crear diseños únicos y específicos para sus proyectos son muy fáciles de usar, pero pueden ser menos conocidas que otras herramientas de CSS, lo que puede dificultar la búsqueda de recursos y apoyo en línea. Además, al igual que con otras herramientas de CSS, aumentar la cantidad de clases de estilo en el código, lo que puede dificultar la lectura y el mantenimiento.

El desarrollo de software con la tecnología CSS como con otras tecnologías tiende a ser repetitivo, poco intuitivo, ineficiente y en su mayoría reduce la productividad del 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 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 lo parámetros se procesan de forma serial, a continuación, un gráfico de la arquitectura Transformer”. (Sotaquirá, 2020).

Los módulos de codificación y decodificación, también se utiliza las técnicas de embedding y los bloques atencionales toda esta arquitectura es muy 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. (Sotaquirá, 2020)



**Figura 1.** Arquitectu**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.

# **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.

# **CAPÍTULO II**

# **MARCO CONTEXTUAL**

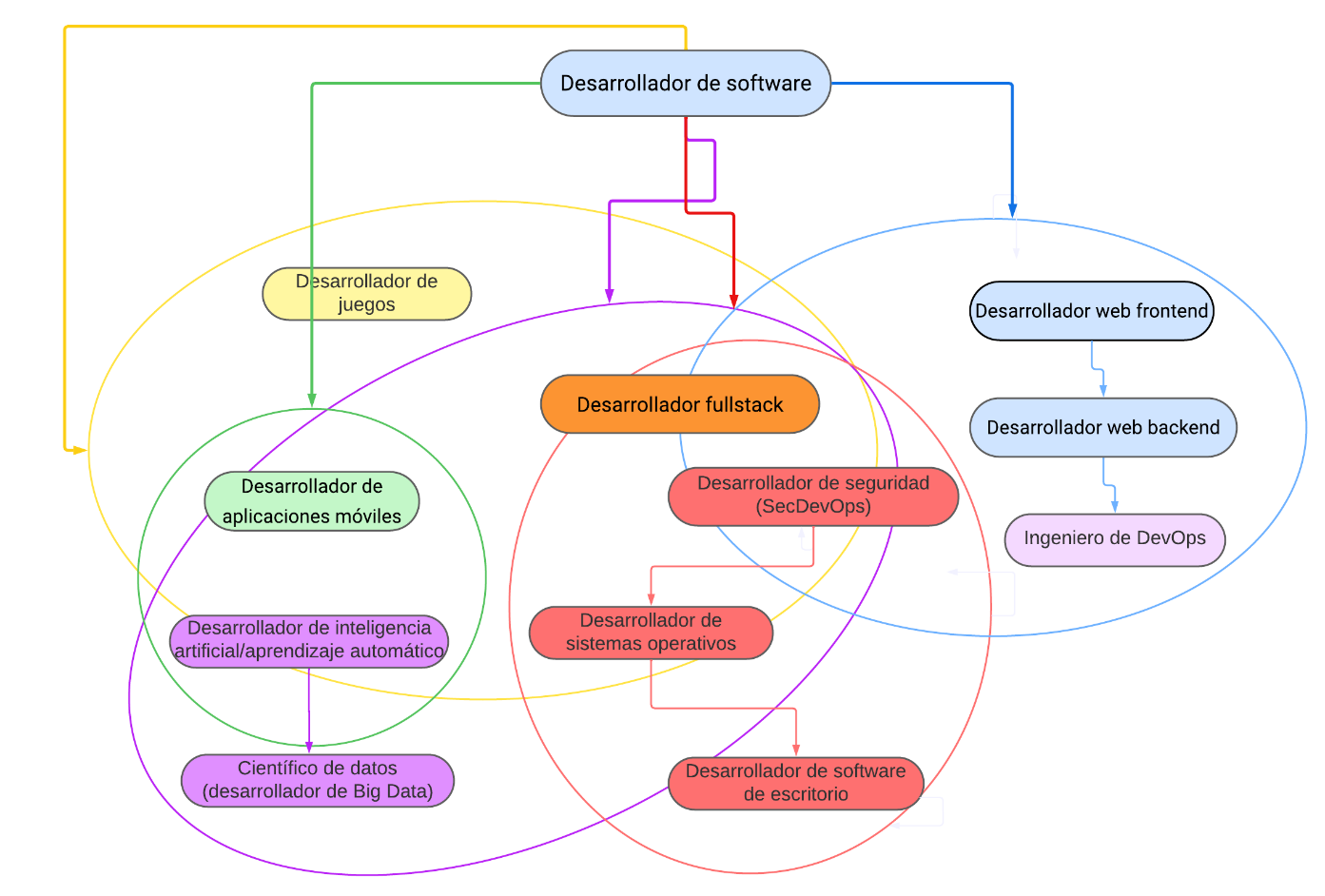
# **2.1. Análisis de la Situación Actual**

## **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.

## **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.** Digrama de Tipos de Desarrolladores

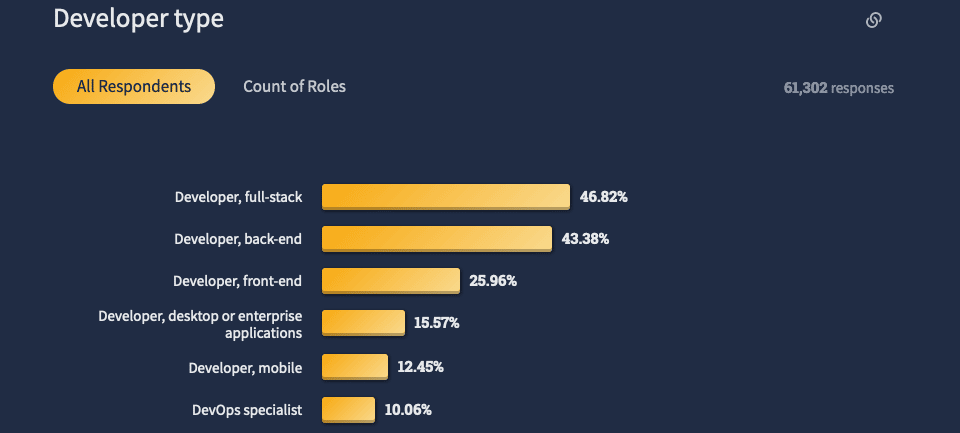
Fuente: Elaboración Propia

**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.

**¿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.

**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.

**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.

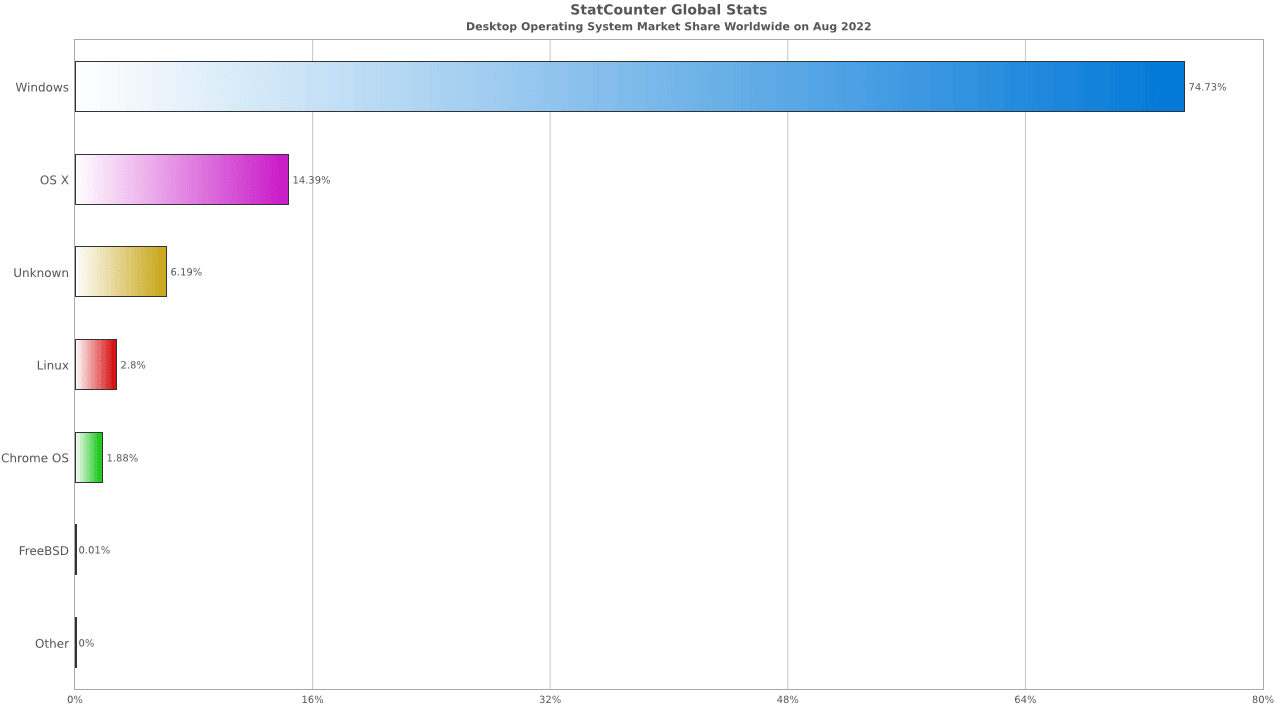


**Figura 3.** Principales tipos de desarrolladores en 2022.

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.

**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.



**Figura 4.** Cuota de mercado de los sistemas operativos de escritorio, 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:

**Procesamiento de lenguaje natural (NLP):** Los Transformers son ampliamente utilizados en aplicaciones de NLP, como sistemas de traducción automática, resumen de texto, generación de texto, análisis de sentimientos, reconocimiento de entidades nombradas, extracción de información, chatbots y asistentes virtuales. Los transformers, en particular los modelos de lenguaje como GPT (Generative Pre-trained Transformer), han logrado avances significativos en estas tareas.

**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.

**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.

**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.

**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.

**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.



**Figura 3.** Diagrama de Casos de uso de un Modelo de Inteligencia Artificial

Fuente: (Nvidia, 2023)

# **CAPÍTULO III**

# **FUNDAMENTO TEÓRICO**

# **3.1. Antecedente Teórico**

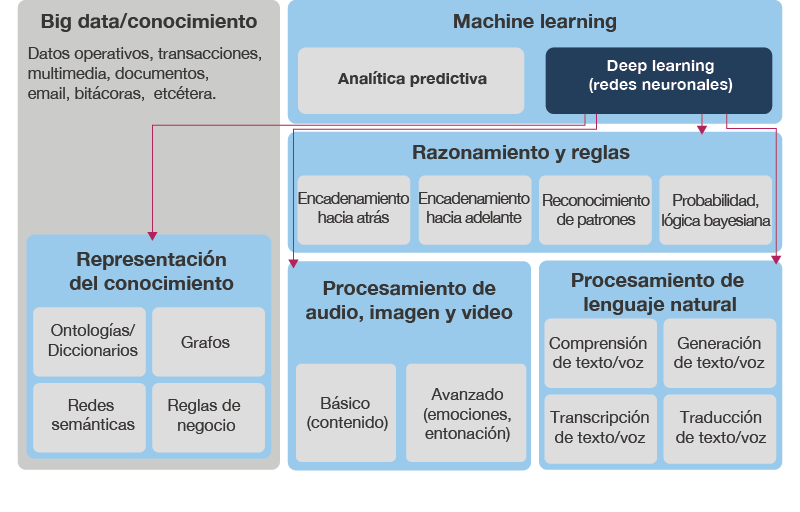
El desarrollo de software se refiere al proceso de crear, diseñar, programar, probar y mantener programas de computadora, aplicaciones móviles o sistemas informáticos. Es un proceso sistemático que involucra diversas etapas y actividades para construir software funcional y de calidad.

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 5.** 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 4.** Diagrama de Proceso de Desarrollo de Chat GPT

Fuente: (marketing-branding, 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 sigue 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 puzle. 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, Herramients y Frameworks de CSS que mejoran la eficiencia en el desarrollo, el diseño responsive, la reutilización de componentes, 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)

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)

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 syntax (en-US) única del preprocesador. Existen varios preprocesadores CSS de los cuales escoger, sin embargo, la mayoría de preprocesadores CSS añadiran algunas características que no existen en CSS puro, como variable, mixins, 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**

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)

**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 y una neurona de la capa está asociada con un peso.

Matemáticamente: Si representamos las activaciones de las neuronas en la capa como un vector y los pesos asociados con las conexiones hacia la capa como una matriz , entonces el cálculo de las activaciones en la capa se realiza mediante una operación de multiplicación de matrices:

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.

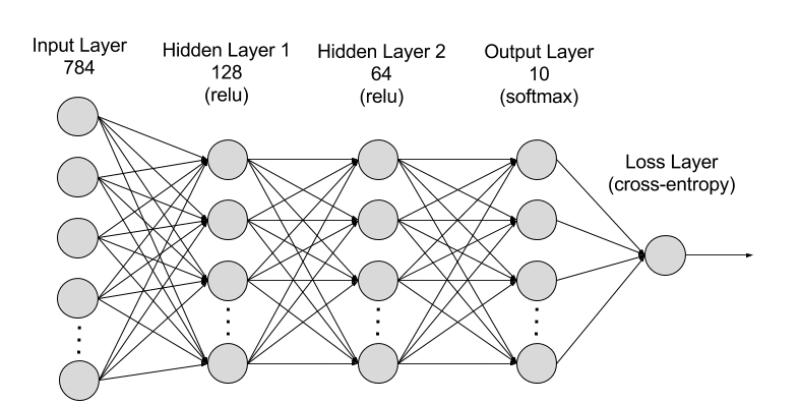
**Sesgos (Biases):**

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.

Matemáticamente: Si representamos los sesgos en la capa como un vector , entonces el cálculo de las activaciones en la capa se modifica agregando el sesgo:

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

**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 6.** Arquitectura de una Red Neuronal Simple

Fuente: (aws, 2022)

### **¿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**

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 entrenan 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 7.** 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

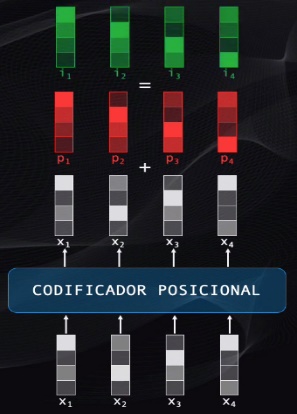
Primero está el bloque embedding, que es simplemente un algoritmo que convierte el texto en una serie de vectores, o tokens, es decir en una representación numérica que puede ser “comprendida” por la red.



**Figura 8.** Embedding de entrada de la Red Transformer

Fuente: (Sotaquirá, 2020)

#### El codificador de posición



**Figura 9.** 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)

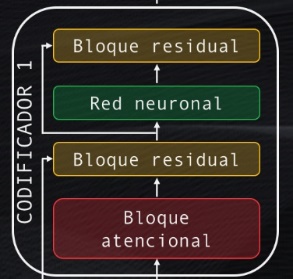
#### Codificación y el bloque atencional



**Figura 10.** 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 11.** 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 12.** 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 13.** 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 13.** 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 14.** 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 15.** 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 16.** Los múltiples bloques atencionales permiten detectar asociaciones entre palabras y grupos de palabras a diferentes niveles

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 17.** 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 18.** 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 19.** 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 20.** 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 21.** 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 22.** 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 23.** 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 24.** 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 25.** Salida del bloque de decodificación

Fuente: (Sotaquirá, 2020)

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

# **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.

### **¿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)

### **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.

### **¿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 pretendes 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 es critores 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 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.

**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.

1. **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.

1. **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.

1. **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.

1. **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.

**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.

**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.

## **3.3.3 React para el desarrollo web**

React.js, comúnmente conocido como React, es una biblioteca de JavaScript de código abierto desarrollada por Facebook. Se utiliza principalmente en el desarrollo web para la creación de interfaces de usuario interactivas y eficientes. Algunas de las ventajas que ofrece React para aplicar en tu proyecto incluyen:

* Componentización: React utiliza un enfoque basado en componentes, lo que permite dividir la interfaz de usuario en piezas independientes y reutilizables. Esto facilita la creación, mantenimiento y escalabilidad del código.
* Virtual DOM: React utiliza un modelo de DOM virtual que optimiza las actualizaciones en el navegador, minimizando las manipulaciones directas en el DOM y mejorando el rendimiento de la aplicación.
* JSX: React utiliza JSX, una extensión de JavaScript que permite escribir código HTML dentro de archivos JavaScript. Esto facilita la creación de componentes con una sintaxis más intuitiva y legible.
* Unidireccionalidad de datos: React sigue el principio de unidireccionalidad de datos, lo que significa que los datos fluyen en una sola dirección dentro de la aplicación. Esto facilita el seguimiento de los cambios y la depuración del código.
* Ecosistema y comunidad activa: React cuenta con un ecosistema robusto de herramientas y bibliotecas complementarias, como React Router para el enrutamiento, Redux para la gestión del estado y Material-UI para componentes de interfaz de usuario predefinidos. Además, tiene una comunidad activa que proporciona soporte, documentación y recursos educativos.
* Reutilización de código: La arquitectura basada en componentes de React facilita la reutilización de código, lo que puede ahorrar tiempo y esfuerzo en el desarrollo de nuevas características y aplicaciones.

## **3.3.4 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.

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.

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.

## **3.3.5 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:

* 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.

## **3.3.6 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:

* Incluye soporte para depuración
* Control de Git integrado
* Resaltado de sintaxis
* Finalización de código inteligente.
* Fragmentos de código.
* Refactorización de código.
* Personalización.
* Atajos

# **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 aplicacion wev 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 Figura 26:

|  |  |  |  |
| --- | --- | --- | --- |
| N° de Iteracion | FASE | PRODUCTO | |
| 1 | **EXPLORACION** | | **Analisis Preliminar del Proyecto** |
| 2 | **PLANIFICACION** | | **Definicion de la Arquitectura del Sistema.** |
| 3 |  | | **Dataset de Prueba y Entrenamiento** |
| 4 | **ITERACIONES** | | **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** |

**Figura 26.** 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 27.** 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**

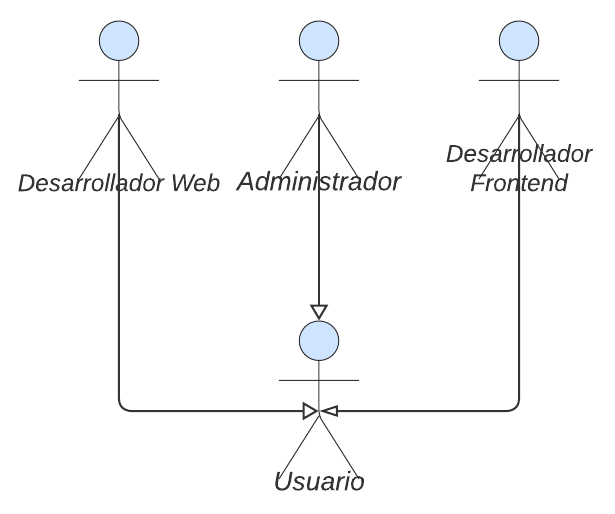
Para determinar los diferentes requerimientos para el presente proyecto, en primera instancia se realiza una investigación y análisis de los estilos y técnicas que en la actualidad utilizan los desarrolladores frontend con la tecnología CSS.

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 28.** 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:  ▪ Registrar, Editar o Eliminar Usuarios.  ▪ Añadir nuevos administradores.  ▪ Revisar el funcionamiento correcto del Modelo  ▪ Administrar la API  ▪ Auditorias. |

**Figura 29.** Descripcion 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:  ▪ 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. |

**Figura 30.** Descripcion 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: | Este actor puede realizar las siguientes acciones en la aplicación Web:  ▪ 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. |

**Figura 30.** Descripcion 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 él 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**

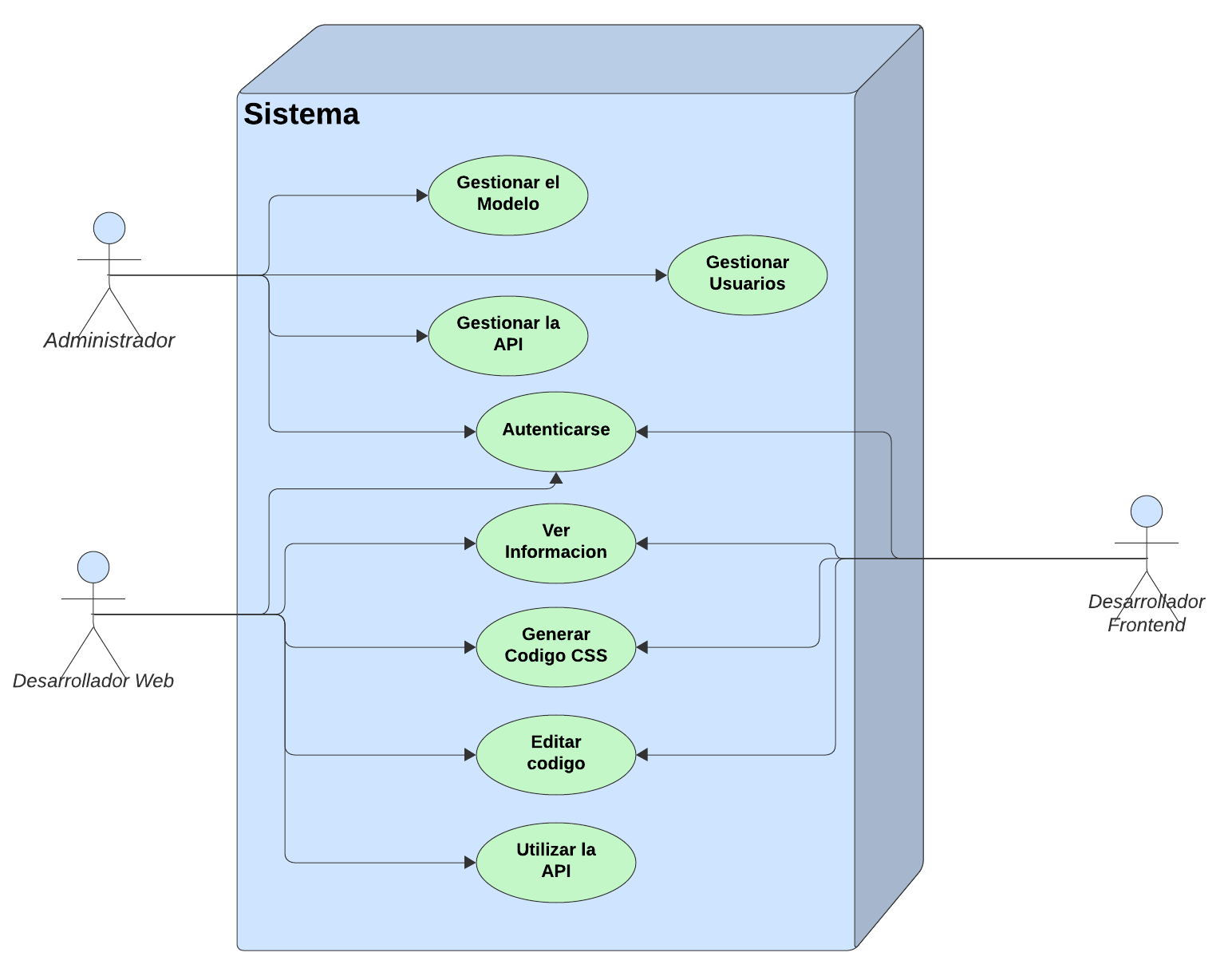
Debe ser usable para todos usuarios.

**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**

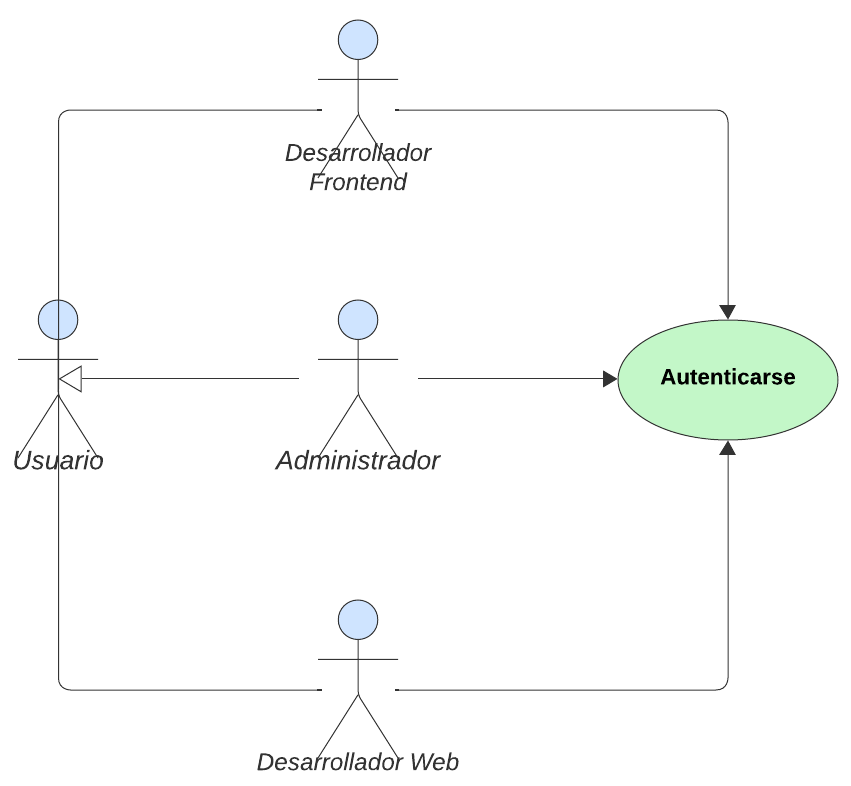


**Figura 31.** Diagrama General de Casos de Uso

Fuente: Elaboración Propia

## **5.2.2 Diagrama de Casos de Uso Clasificación por prioridad**

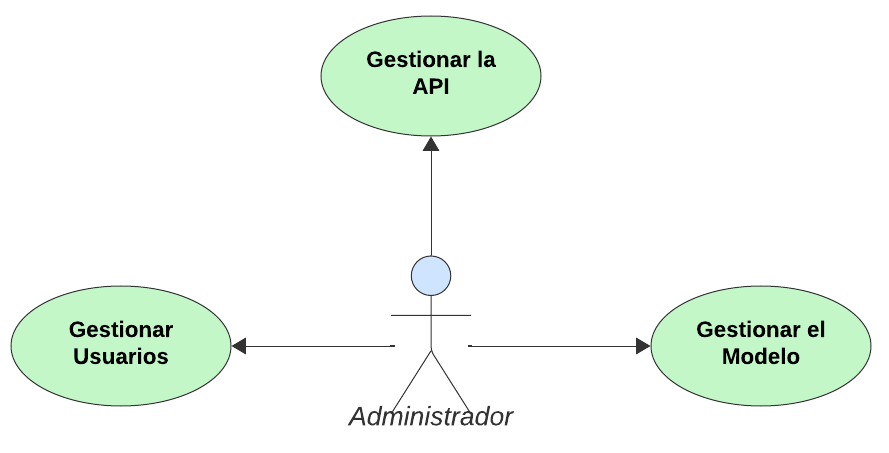
* Diagrama de Casos de Uso del Usuario



**Figura 32.** Diagrama de Casos de Uso del Usuario

Fuente: Elaboración Propia

* Diagrama de Casos de Uso del Administrador

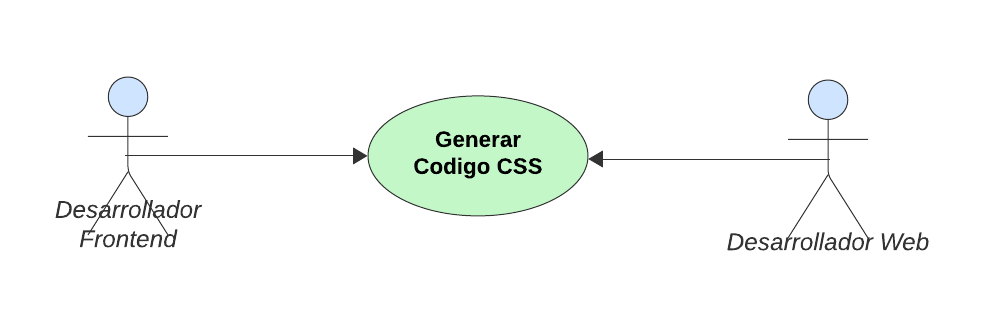


**Figura 33.** Diagrama de Casos de Uso del Administrador

Fuente: Elaboración Propia

## **5.2.3 Diagrama de Casos de Uso Clasificados por funcionalidad**

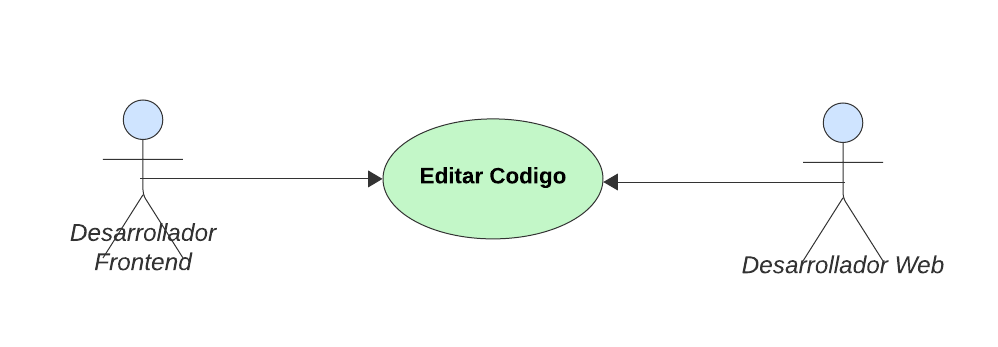
* Diagrama de Casos de Uso de Generación de código CSS



**Figura 34.** 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 35.** Diagrama de Casos de Uso de Edicion de Codigo 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 codigo 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. |

**Figura 36.** Descripcion 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**

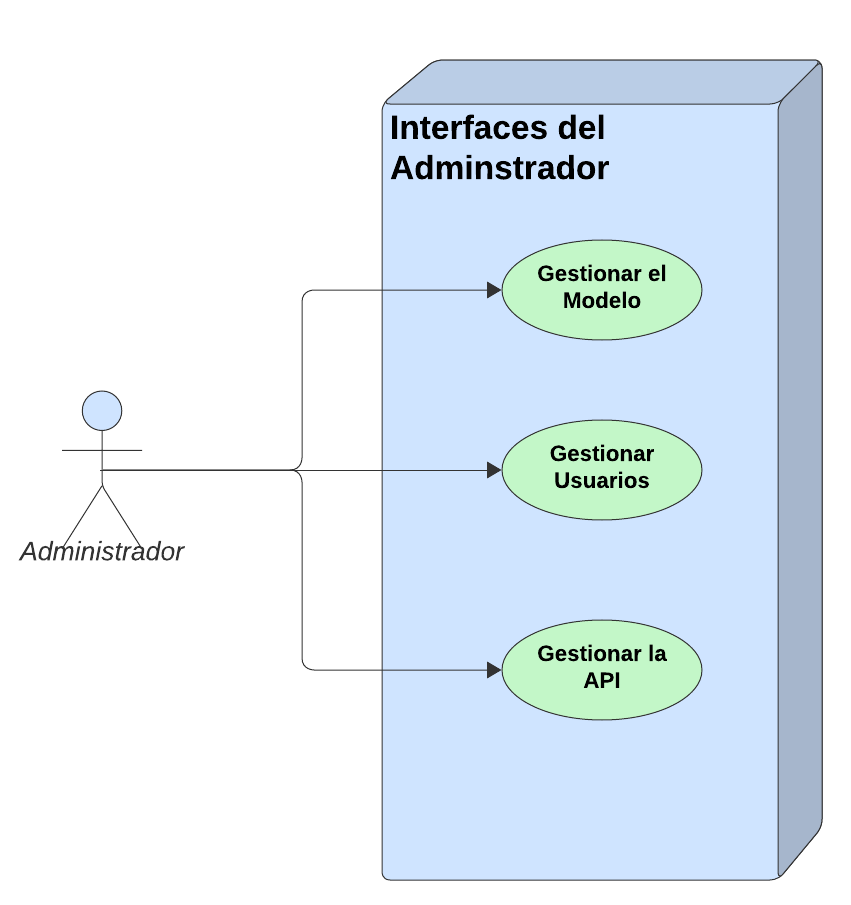
* Riesgo: Alto.
* Descripción del Riesgo: El proyecto no cuenta con las suficientes pruebas unitarias o de sistemas que garanticen y validen la calidad del sistema.
* Impacto: El funcionamiento del sistema puede tener fallas.
* Indicadores: No se realiza pruebas al haber finalizado la implementación 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**

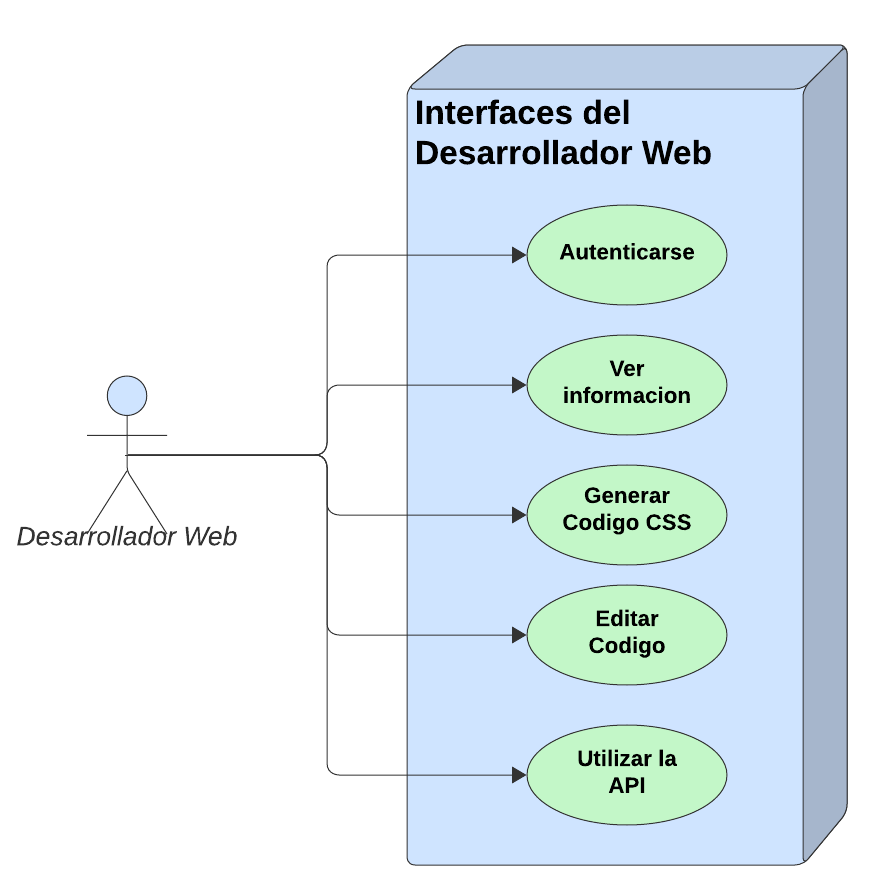
### **5.2.7.1 Diagrama Frontera del Administrador**

****

**Figura 37.** Diagrama Frontera del Admintrador

Fuente: Elaboración Propia

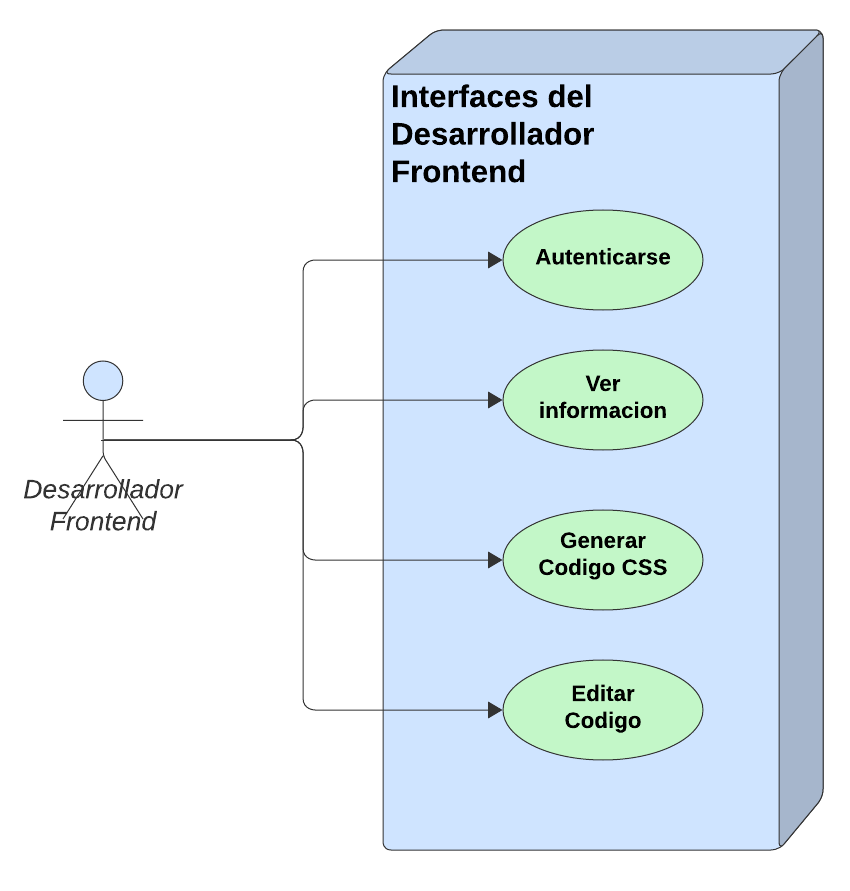
### **5.2.7.2 Diagrama Frontera del Desarrollador Web**



**Figura 38.** Diagrama Frontera del Desarrollador Web

Fuente: Elaboración Propia

### **5.2.7.3 Diagrama Frontera del Desarrollador Frontend**

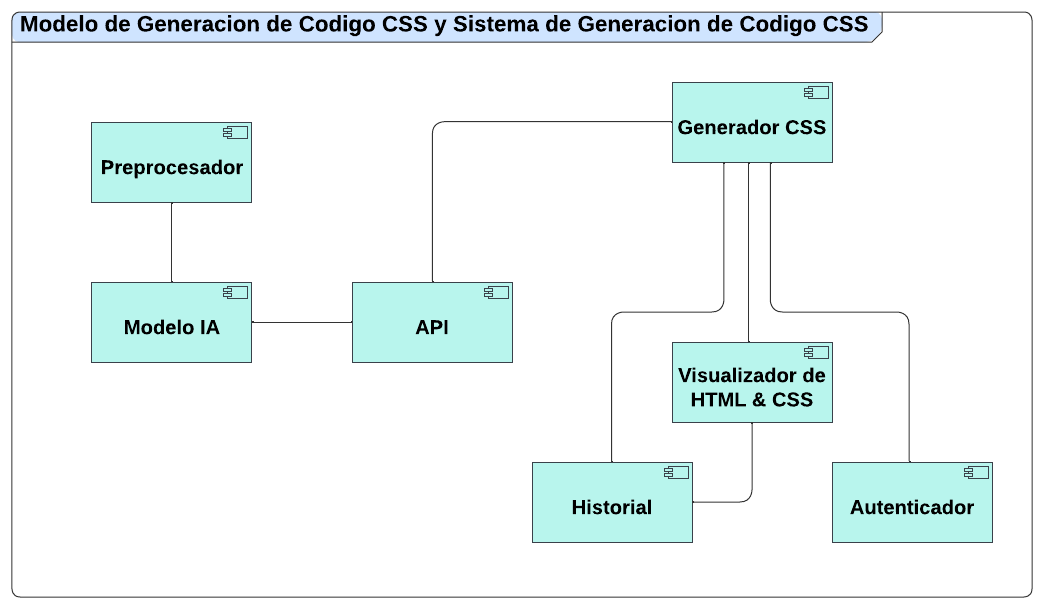


**Figura 39.** Diagrama Frontera del Desarrollador Frontend

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 40.** 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

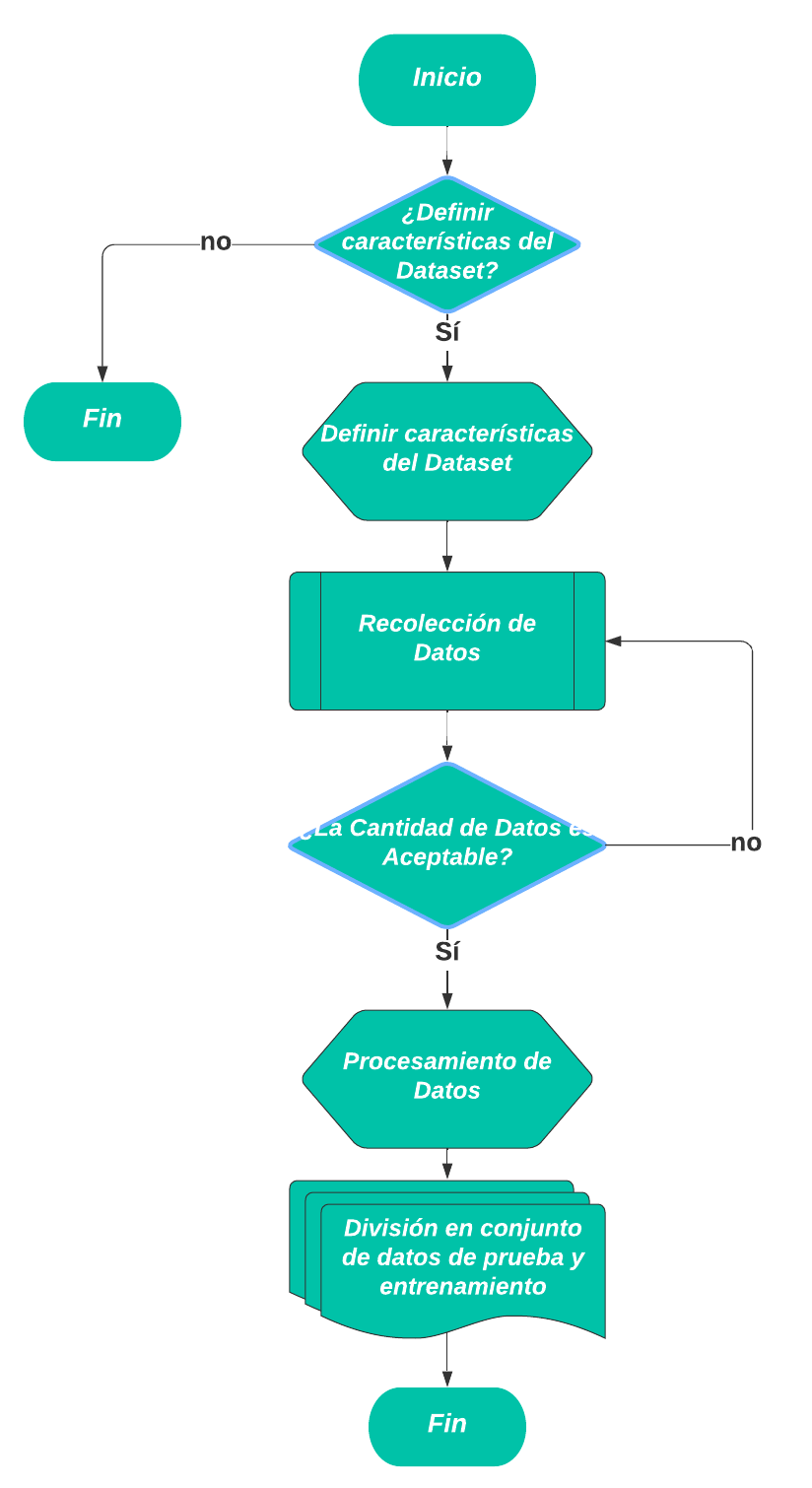
Realiza el proceso de acceso del usuario en el Sistema Generador de código CSS.

# **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 41.** Diagrama de Flujo del Desarrollo del Dataset de Entrenamiento, Validacion, Prueba

Fuente: Elaboración Propia

### **5.3.1.2 Características del Dataset**

Los datasets anteriormente mencionados deben tener ciertas características específicas para garantizar la eficiencia del entrenamiento del modelo. A continuación, se muestra algunas de las características consideradas para la recolección de Datos:

* Etiquetas HTML y sus atributos: El dataset deben 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 de buena calidad que contengan las mejores prácticas, 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 cumpliendo así con la calidad de los datos.

Algunos de los generadores que se utilizó son:

* **CSS Grid Generator** consiste en un práctico 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 pegar en nuestro sitio web para obtener la tabla.
* **Ultimate CSS Gradient Generator** para generar degradados de colores para fondos o menús, Ultimate CSS Gradient Generator te lo pone fácil. 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 de jugar con los colores, combinando hasta seis diferentes, 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…
* **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.
* **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.
* **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.

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](https://github.com/necolas/normalize.css%20)
* Repositorio de Materialize CSS: <https://github.com/Dogfalo/materialize>
* Repositorio de CSS Frameworks: <https://github.com/awesome-css-group/awesome-css>
* Repositorio de CSS Snippets: <https://github.com/30-seconds/30-seconds-of-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:



**Figura 42.** Ejemplo de Dataset

Fuente: Elaboración Propia

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

La división en conjuntos de datos de entrenamiento, validación y prueba es una práctica fundamental en el aprendizaje automático. Consiste en separar el conjunto de datos disponible en tres partes.

Esta división ayuda a evaluar la capacidad del modelo para generalizar datos no vistos. Se suele asignar la mayoría de los datos al conjunto de entrenamiento y una porción más pequeña al conjunto de validación y de prueba, manteniendo una proporción adecuada para obtener resultados confiables. Es crucial para el desarrollo y validación efectivo de modelos de aprendizaje automático.

Para el entrenamiento de nuestro Modelo Generador de Código CSS dividiremos el dataset en tres archivos de tipo JSON llamados “STYLES.json”, “VALIDATION.json” y “TEST.json” los cuales cumplirán los roles de dataset de entrenamiento (train), dataset de validación (validation) y dataset de prueba (test) que explicaremos a continuación.

**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.

**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.

**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.

**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.

**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.

**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.

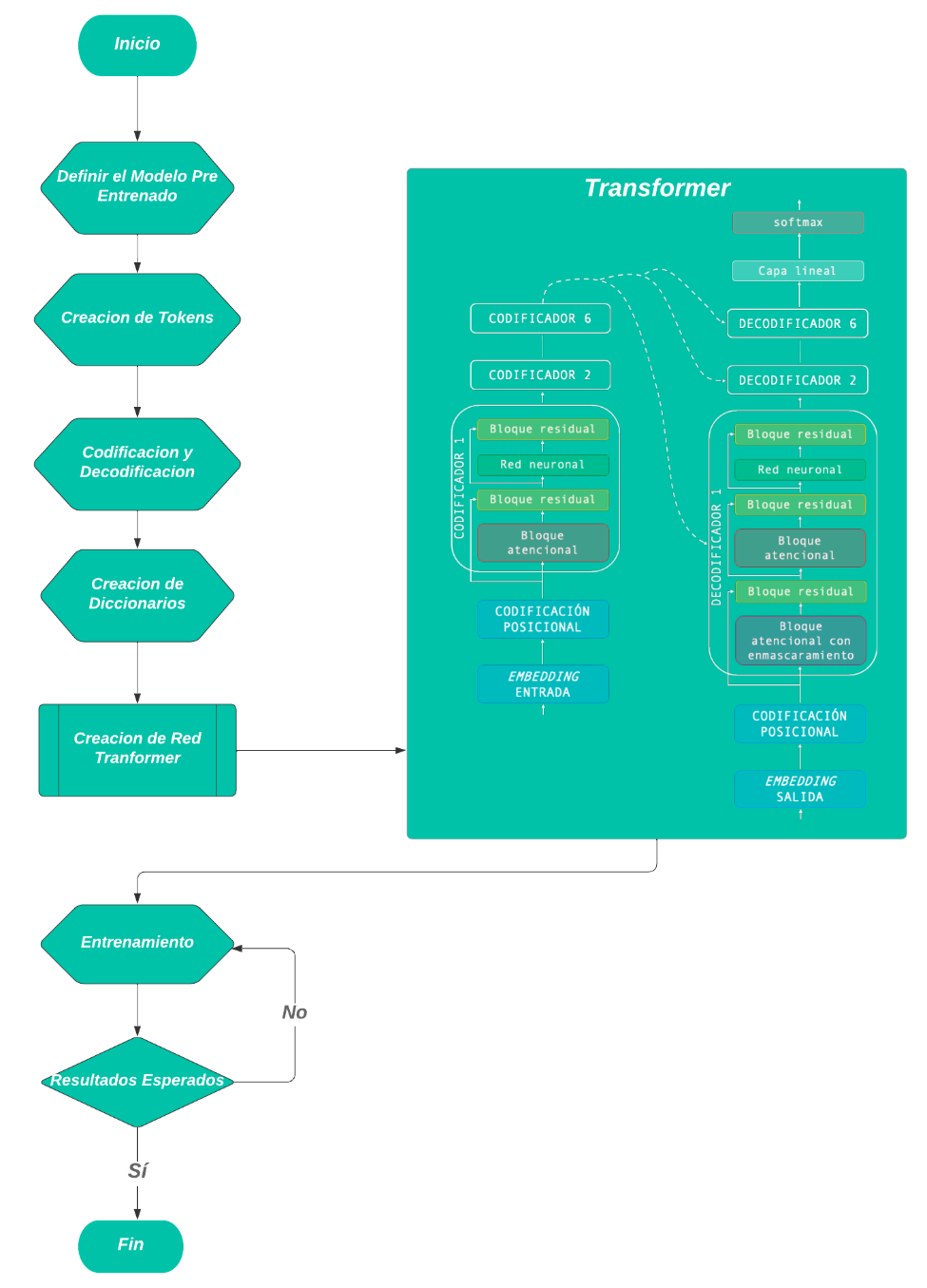
**Importancia:** Proporciona una evaluación final e imparcial del rendimiento del modelo en datos no vistos, lo que ayuda a estimar cómo se comportará el modelo en la práctica real.

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.

## **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 43.** Diagrama de Flujo del Desarrollo del Modelo IA

Fuente: Elaboración Propia

### **5.3.2.2 Modelo Pre-Entrenado**

El uso de un modelo pre-entrenado tiene importantes ventajas. Reduce los costos de computación, la huella de carbono, y te 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 utilizas un modelo pre-entrenado, lo entrenas con un dataset específico para tu tarea. Esto se conoce como fine-tuning, una técnica de entrenamiento increíblemente poderosa.

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. Tambien se consideró los datasets usados para el entrenamiento del modelo seleccionado (GPT-2 XL), en donde se investigó y analizo diferentes modelos utilizados en el área de NLP (natural language processing) que cuentan con la implementación de una arquitectura de Transformers.

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:

**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.

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.

**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.

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.

* **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.

* **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.

* **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.

**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.

**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.

**Detalles del Modelo:**

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).

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
* **Usos:**

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.

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.

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.

* **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.

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. Desdo 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.

* **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.
* **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.

El modelo utiliza internamente un mecanismo de máscara para garantizar que las predicciones para el token *“i”* solo utilicen las entradas de los tokens *“1”* 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.

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

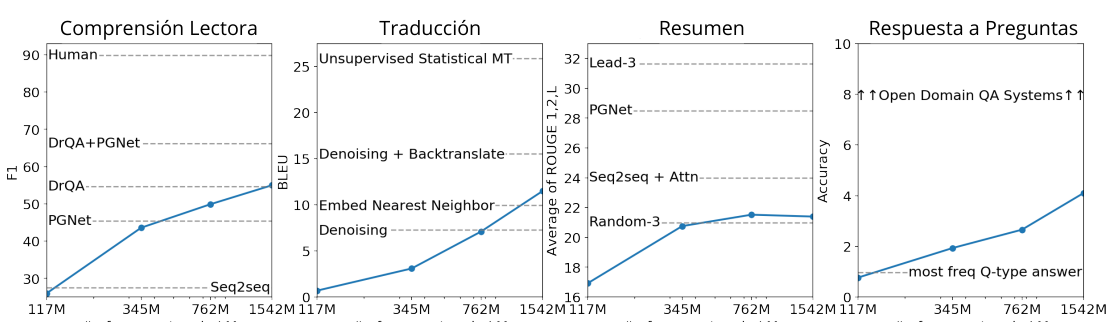
Los resultados de los conjuntos de datos de modelado del lenguaje comúnmente se informan en una cantidad que es una versión escalada o exponencial de la probabilidad logarítmica negativa promedio por unidad de predicción canónica (generalmente un carácter, un byte o una palabra).

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.

* **Resultados:**

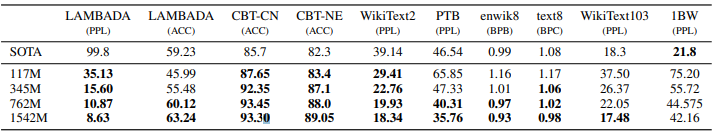
Rendimiento del modelo GPT-2 XL en tareas de NLP



**Figura 43.** Rendimiento del Modelo GPT-2 XL en Tareas NLP de Comprension Lectora, Traduccion, Resumen y Respuesta a Preguntas.

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

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



**Figura 44.** Resultados del Modelo GPT-2 XL

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

### **5.3.2.3 Implementación del Modelo**

La implementación del Modelo pre-entrenado GPT-2 XL se realizó gracias a la ayuda de la librería de Hugging Face Transformers una librería que proporciona APIs para descargar y entrenar fácilmente modelos pre-entrenados de última generación. El uso de modelos pre-entrenados puede reducir los costos de cómputo, la huella de carbono y ahorrar tiempo al entrenar un modelo desde cero. 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 perfecta entre tres de las bibliotecas de deep learning más populares: PyTorch, TensorFlow y JAX. Permite entrenar tu 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 de calidad 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.

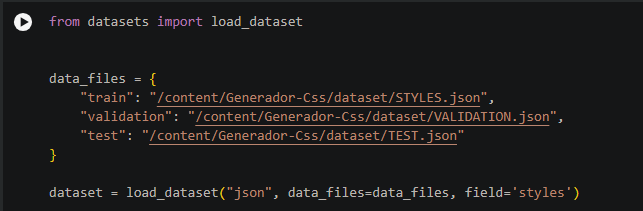
**Por qué utilizar load\_dataset():**

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():



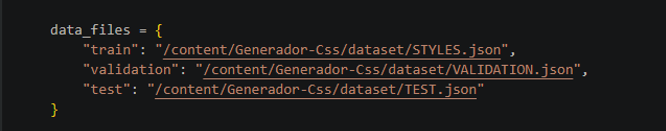
**Figura 45.** Creacion del Dataset

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:



**Figura 46.** 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 utilizados la función antes mencionada load\_dataset()



**Figura 47.** Creacion de Dataset de Entrenamiento

Fuente: Elaboración Propia

En la figura 47 se puede apreciar el uso de la funcion 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 definio 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.

**Cargamos nuestro 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.

**Características de GPT-2:**

Modelo Pre-entrenado: GPT-2 se entrena inicialmente en una amplia gama de textos de la web para capturar una comprensión general del lenguaje natural.

Arquitectura Transformer: Utiliza la arquitectura Transformer, que permite procesar y generar texto de manera eficiente a través de mecanismos de atención.

Generación de Texto de Alta Calidad: GPT-2 es conocido por generar texto coherente, relevante y de alta calidad en una variedad de tareas de NLP.

Flexibilidad: Puede ser adaptado y ajustado fino (fine-tuning) para tareas específicas utilizando conjuntos de datos más pequeños.

Tamaño Variado: GPT-2 está disponible en diferentes tamaños, desde pequeños modelos con menos parámetros hasta modelos grandes con cientos de millones de parámetros, que ofrecen diferentes niveles de capacidad y rendimiento.

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.

**Mecanismos de Atención y Representaciones Distribuidas:**

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.

****

**Figura 48.** 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 quieres 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.

### **Tokenización e Incrustación**

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.

**¿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.
2. Análisis de Texto: La tokenización permite realizar análisis estadísticos sobre el texto, como contar la frecuencia de palabras, identificar n-gramas, etc.
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.

**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.

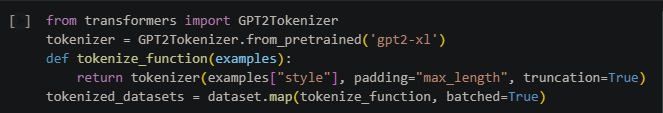
6. Tokenización con Aprendizaje Automático:

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

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 tokenizacion de nuestro dataset en la siguiente figura.



**Figura 49.** Cargando el Tokenizador GPT2Tokenizer para la Tokenizacion 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.

Entendiendo solo el funcionamiento de la Tokenizacion del dataset no es suficiente debido a que al momento de implementar nuestro dataset tokenizado a nuestro modelo internamente en el entrenamiento existe un proceso relevante el cual creo pertinente explicar para profundizar mas con respecto a la arquitectura transformer este proceso es el Embedding o Incrustacion.

**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.

**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.

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.

**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.

**Tokenizacion vs Embedding (Incrustaciones):**

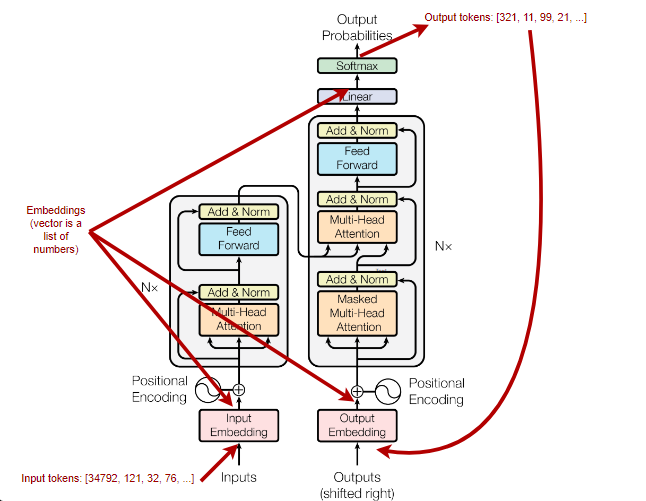
* la entrada se tokeniza , los tokens luego se incrustan
* Las incrustaciones (Embeddings) de texto de salida se clasifican nuevamente en tokens, que luego se pueden decodificar en texto.
* La tokenización convierte un texto en una lista de números enteros.
* La incrustación (Embedding) convierte la 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.

**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.

A continuación, se muestra una breve descripción de este proceso en la Figura 50



**Figura 50.** Tokenizacion, Incrustacion y Procesamiento del Modelo

Fuente: Elaboración Propia

### **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.

**Fine Tuning vs Transfer Learning (Aprendizaje por Transferencia)**

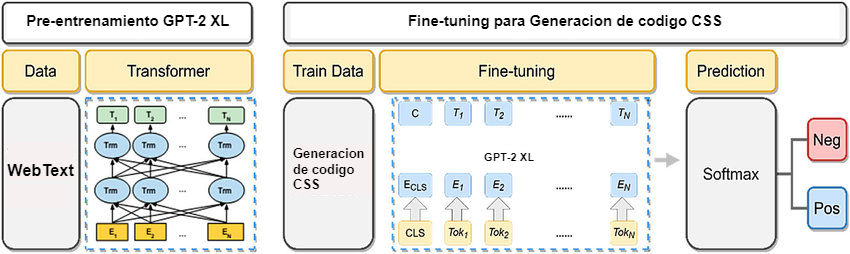
**Es preciso diferenciar lo que es el Fine-Tuning del Transfer Learning (aprendizaje por transferencia):** aunque los dos son enfoques de entrenamiento de redes con algunos datos y ambos se basan en el conocimiento existente, la realidad es que poseen grandes diferencias.

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:

* 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 entrenan 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:

* 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.

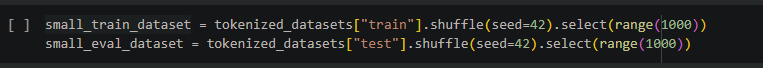


**Figura 51.** Diagrama de Fine Tuning Aplicado a Transformer para la Generacion de codigo CSS

Fuente: Elaboración Propia

**Creación de Sub datasets**

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



**Figura 52.** 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():



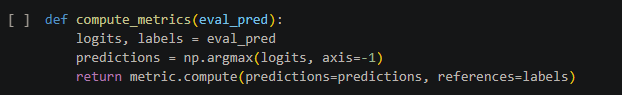
**Figura 53.** Creacion de la Metrica de Evaluacion

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):



**Figura 54.** Creacion de Sub Datasets

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 funcion 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 devuelve 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:



**Figura 55.** Implementacion de TrainingArguments

Fuente: Elaboración Propia

**from transformers import TrainingArguments:**

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

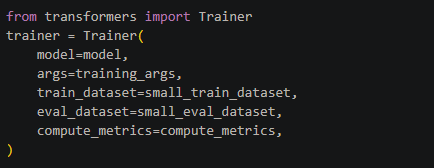
TrainingArguments: Esta clase se utiliza para definir y almacenar los argumentos de entrenamiento para un modelo. Esto incluye cosas como la ruta del directorio de salida, la estrategia de evaluación, el número de épocas, etc.

**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



**Figura 56.** Implementacaion 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 definio.

**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.

### **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.



**Figura 57.** Entrenamiento del Modelo

Fuente: Elaboración Propia

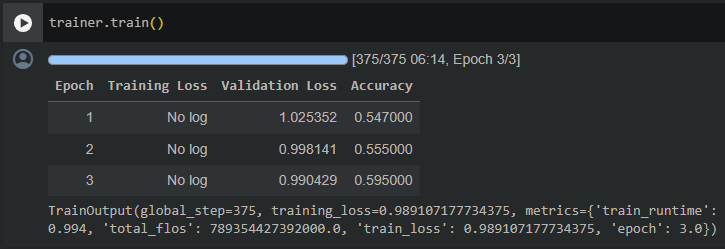
Esta línea invoca el método train() en el objeto trainer, que es una instancia de la clase Trainer del paquete transformers de Hugging Face.

Cuando se llama al método train(), el Trainer comenzará el proceso de entrenamiento del modelo que se ha configurado previamente. Aquí te detallo 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.



**Figura 58.** Resultados del Entrenamiento del Modelo con 3 epocas

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 podría indicar 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 o Precisión):** 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 epocas**

Epoch 1: No se registró la pérdida de entrenamiento. La pérdida de validación es 1.025352, y la precisión es 0.547000.

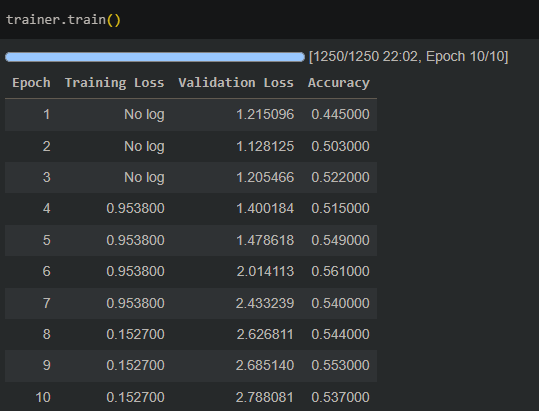
Epoch 2: No se registró la pérdida de entrenamiento. La pérdida de validación es 0.998141, y la precisión ha aumentado ligeramente a 0.555000.

Epoch 3: No se registró la pérdida de entrenamiento. La pérdida de validación ha disminuido a 0.990429, y la precisión ha aumentado significativamente a 0.595000.

**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.

**Interpretación de los Resultados con 10 epocas**



**Figura 59.** Resultados del Entrenamiento del Modelo con 10 epocas

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 epocas, 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.

**Conclusiones con respecto a la Implementación del Modelo**

En conclusión, al examinar los dos experimentos iniciales que se realizaron modificando únicamente el hiperparámetro del número de épocas a 3 y 10, 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, se profundizará en la fase de pruebas de este proyecto, 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.

## **API de servicios web para 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 recibe datos en formato JSON con el contenido HTML de las etiquetas para las cuales se generará el código CSS. Claro, aquí tienes el texto corregido y extendido:

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 Peticion:

{

"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**

En Proceso

# **5.4. Fase de Pruebas**

## **5.4.1 Ajuste del Modelo**

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

## **5.4.2 Mantenimiento y Pruebas**

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

# **BIBLIOGRAFÍA**

**Author, Guest. 2020.** rockcontent. *rockcontent.* [Online] © 2013-2021 Rock Content, Abril 12, 2020. [Cited: Abril 24, 2023.] https://rockcontent.com/es/blog/bootstrap/.

**aws. 2022.** amazon.com. *amazon.com.* [Online] AWS, 12 16, 2022. [Cited: 9 3, 2023.] https://aws.amazon.com/es/what-is/neural-network/#:~:text=Una%20red%20neuronal%20es%20un,lo%20hace%20el%20cerebro%20humano..

**Bootstrap. 2023.** getbootstrap. *getbootstrap.* [Online] Bootstrap, Enero 15, 2023. [Cited: Abril 24, 2023.] https://getbootstrap.com/.

**BuiltWith. 2019.** BuiltWith. *BuiltWith.* [Online] BuiltWith® Pty Ltd, Octubre 24, 2019. [Cited: Abril 23, 2023.] https://builtwith.com.

**datascientest. 2023.** datascientest.com. *datascientest.* [Online] DataScientest, 8 10, 2023. [Cited: 6 6, 2023.] https://datascientest.com/es/inteligencia-artificial-definicion.

**Galvan, Pedro. 2021.** sg.com.mx. *sg.com.mx.* [Online] Software Guru, 10 15, 2021. [Cited: 9 4, 2023.] https://sg.com.mx/revista/56/inteligencia-artificial-desarrollo-software.

**IDC. 2023.** IDC. *IDC.* [Online] IDC Corporate, 2023. [Cited: 8 23, 2023.] https://www.idc.com/getdoc.jsp?containerId=prUS49670122.

**marketing-branding. 2023.** marketing-branding. *marketing-branding.* [Online] Marketing Branding Blog, Febrero 12, 2023. [Cited: Junio 13, 2023.] https://www.marketing-branding.com/chat-gpt-inteligencia-artificial-que-es-seo-friendly/. 234.

**Nation, SlashData’s Developer. 2023.** Developer Nation. *Developer Nation.* [Online] Developer Nation, 2023. [Cited: 8 23, 2023.] https://www.developernation.net/developer-reports/de20.

**Nvidia. 2023.** https://la.blogs.nvidia.com. *https://la.blogs.nvidia.com.* [Online] Nvidia, 6 22, 2023. [Cited: 8 23, 2023.] https://la.blogs.nvidia.com/2023/06/22/que-son-los-modelos-de-base/.

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

**Sotaquirá, Miguel. 2020.** codificandobits. *codificandobits.* [Online] © 2023 Codificando Bits, Junio 30, 2020. [Cited: Abril 24, 2023.] https://www.codificandobits.com/blog/redes-transformer/#la-red-transformer.

**StackOverflow. 2023.** StackOverflow. *StackOverflow.* [Online] Stackoverflow, 2023. [Cited: 8 23, 2023.] https://stackoverflow.com/.

**Statcounter. 2023.** Statcounter. *Statcounter.* [Online] Statcounter, 2023. [Cited: 8 23, 2023.] https://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-202208-202208-bar.