



Universidad de Cuenca

Facultad de Ingeniería

Carrera de Ingeniería de Sistemas

Generación de Código a partir de requisitos basados en voz. Un análisis multicriterio de herramientas de generación de código basadas en Inteligencia artificial

Trabajo de titulación previo a la obtención del título de Ingeniero de Sistemas

Autor:

Cristian Ricardo Méndez Espinoza

Director:

Luis Otto Parra González

ORCID:  0000-0003-3004-1025

Tutor:

María Fernanda Granda Juca

ORCID:  0000-0002-5125-8234

Cuenca, Ecuador
2024-10-03

Resumen

La presente investigación titulada: “Generación de Código a partir de requisitos basados en voz. Un análisis multicriterio de herramientas generadoras de código basadas en Inteligencia artificial” se centra en seleccionar, evaluar y comparar herramientas de inteligencia artificial aplicadas a un estudio de caso sobre el registro y manejo de mascotas en la ciudad de Cuenca, para la generación de código a través de prompts basados en voz, donde se aplican patrones en las entradas proporcionadas con el fin de obtener resultados más completos. Se realiza una investigación exhaustiva de los tipos de herramientas existentes, además de entrevistas hechas a profesionales del área de software donde se analizan un conjunto de 10 herramientas; se toman como criterios de selección que sean generadoras de código, que la herramienta acepte entradas por voz de forma dictada o clonada y licencia de uso libre. Como resultado se obtienen 4 herramientas IA que cumplen con estos requisitos: ChatGPT, Bing, Gemini y Replit a las cuales se aplica una revisión de casos de uso mediante las pruebas hechas de forma práctica con los prompts elaborados, realizando también un análisis multicriterio basado en características de Interfaz, precisión de voz, facilidad de uso, funciones y compatibilidad mediante procesos de cálculo manuales y con el respaldo de la herramienta RStudio. Finalmente se provee una guía de estas 4 herramientas resultantes.

Palabras clave del autor: inteligencia artificial, generación de código, comandos de voz, multicriterio, patrones para prompts



El contenido de esta obra corresponde al derecho de expresión de los autores y no compromete el pensamiento institucional de la Universidad de Cuenca ni desata su responsabilidad frente a terceros. Los autores asumen la responsabilidad por la propiedad intelectual y los derechos de autor.

Repositorio Institucional: <https://dspace.ucuenca.edu.ec/>

Abstract

This research entitled: “Code Generation from Voice-Based Requirements. A multi-criteria analysis of artificial Intelligence-based code generating tools” focuses on selecting, evaluating, and comparing artificial intelligence tools applied to a case study on the registration and management of pets in the city of Cuenca, for the generation of code through voice-based prompts, where they are applied patterns in the inputs provided to obtain more complete results. An exhaustive investigation of the types of existing tools is carried out, in addition to interviews with professionals in the software area where a set of 10 tools are analyzed; The selection criteria are that they generate code, that the tool accepts voice input dictated or cloned, and that it has a free use license. As a result, 4 AI tools are obtained that meet these requirements: ChatGPT, Bing, Gemini and Replit, to which a review of use cases is applied through tests carried out nearly with the prepared prompts, also performing a multi-criteria analysis based on Interface features, voice accuracy, ease of use, functions and compatibility through manual calculation processes and supported by the RStudio tool. Finally, a guide to these 4 resulting tools is provided.

Author Keywords: artificial intelligence, code generation, voice commands, multicriteria, prompt patterns



The content of this work corresponds to the right of expression of the authors and does not compromise the institutional thinking of the University of Cuenca, nor does it release its responsibility before third parties. The authors assume responsibility for the intellectual property and copyrights.

Institutional Repository: <https://dspace.ucuenca.edu.ec/>

Índice de contenido

Resumen.....	2
Abstract.....	3
Índice de contenido.....	3
Dedicatorias	9
Agradecimientos	10
Capítulo 1: Introducción	11
1.1 Antecedentes de investigación.....	11
1.1.1 Contextualización del desarrollo de software	11
1.1.2 Importancia de la generación de código	11
1.2 Problemática y justificación de la investigación.....	12
1.3 Preguntas de investigación	13
1.4 Objetivos.....	14
1.4.1 Objetivo General	14
1.4.2 Objetivos Específicos	14
Capítulo 2: Marco Teórico.....	15
2.1 Generación de código automatizada.....	15
2.1.1 Proceso de generación de código automatizada	15
2.1.2 Ventajas de la generación de código automatizada.....	17
2.2 Herramientas de inteligencia artificial para generación de código	18
2.2.1 Desarrollo de la inteligencia artificial	19
2.2.2 Técnicas de generación de código.....	21
2.2.3 Generación de código a partir de prompts	23
2.2.4 Aplicación práctica de las herramientas de IA de generación de código	25
2.3 Reconocimiento de voz en las IA	29
2.3.1 Proceso de reconocimiento de voz	30
2.3.2 Beneficios y dificultades del reconocimiento de voz	30
2.4 Aspectos de generación de código por IA a partir de requisitos basados en voz	32
2.4.1 Automatización.....	32
2.4.2 Ingeniería	32
2.4.3 Licencias de las herramientas IA	33
2.4.4 Regulaciones legales sobre el uso de la IA	34

2.4.5	Toma de decisiones multicriterio para software (Decision making)	34
Capítulo 3: Diseño del caso de estudio para la evolución de rendimiento		44
3.1	Características del software a desarrollar del caso de estudio.	44
3.2	Diseño de investigación	46
3.3	Recolección de datos	48
3.4	Población y muestra	48
3.5	Criterios para el análisis de herramientas	49
3.6	Análisis cualitativo de resultados	50
3.7	Resultados y discusión	51
3.7.1	Resultados de la entrevista	51
3.7.2	Preselección de herramientas IA	59
3.8	Resultados de pruebas	78
3.8.1	ChatGPT	79
3.8.2	Bard (Gemini)	90
3.8.3	Bing Chat	98
3.8.4	Replit GhostWriter	105
3.9	Análisis e Interpretación de resultados	109
3.10	Matriz de multicriterio	165
3.11	Aplicación método TOPSIS	166
Capítulo 4: Guía de aplicación de herramientas		173
4.1	Guía de uso de las herramientas utilizadas	173
4.1.1	Uso de ChatGPT para generar código por reconocimiento de voz	173
4.1.2	Uso de Gemini para generar código por reconocimiento de voz	175
4.1.3	Uso de Bing Chat para generar código por reconocimiento de voz	176
4.1.4	Uso de Replit GhostWriter para generar código por reconocimiento de voz	178
4.1.5	Problemas encontrados e indicaciones para solucionarlos	179
Capítulo 5: Conclusiones y recomendaciones		181
5.1	Conclusiones	181
5.2	Recomendaciones	183
Referencias		184
Anexos		187

Índice de figuras

Figura 1. Proceso de generación automática de código.....	17
Figura 2. Gráfico de estadísticas de género sobre los participantes de la entrevista.....	51
Figura 3. Gráfico sobre sectores donde laboran los entrevistados.....	52
Figura 4. Estadísticas del nivel de experiencia en el área de estudio de los participantes de la entrevista.	52
Figura 5. Estadísticas sobre los tipos de software usados por los participantes de la entrevista.	52
Figura 6. Estadísticas sobre el uso de herramientas IA	53
Figura 7. Gráfico resumen de respuestas dadas por los entrevistados sobre requisitos dados por usuarios y colocados como prompt de entrada en la herramienta de IA.....	54
Figura 8. Criterios a tomar en cuenta para el uso de una herramienta con IA.	56
Figura 9. Gráfico indicativo sobre experiencias negativas con el uso de herramientas de IA en los entrevistados.	56
Figura 10. Logo GitHub Copilot	59
Figura 11. Interfaz GitHub Copilot	60
Figura 12. Logo Replit GhostWriter	61
Figura 13. Interfaz Replit GhostWriter	62
Figura 14. Logo ChatGPT	63
Figura 15. Interfaz ChatGPT	63
Figura 16. Logo AI2sql.....	64
Figura 17. Interfaz AI2sql.....	65
Figura 18. Logo Spellbox	66
Figura 19. Interfaz Spellbox.....	66
Figura 20. Logo Amazon CodeWhisperer.....	67
Figura 21. Interfaz Amazon CodeWhisperer.....	68
Figura 22. Logo Tabnine	69
Figura 23. Interfaz Tabnine	69
Figura 24. Logo Bing Chat.....	70
Figura 25. Interfaz Bing Chat	71
Figura 26. Logo Gemini	72
Figura 27. Interfaz Gemini	72
Figura 28. Logo AlphaCode	73
Figura 29. Interfaz AlphaCode	74
Figura 30. Talk to ChatGPT (Extensión para entrada por voz en ChatGPT).....	79
Figura 31. Talk to Mia (Extensión para entrada por voz en ChatGPT).....	80
Figura 32. Resultados en ChatGPT sobre el primer prompt.....	80
Figura 33. Contenido HTML generado con el segundo prompt en ChatGPT.	81
Figura 34. Vista previa del HTML generado por ChatGPT resultado del segundo prompt.	82
Figura 35. Contenido CSS generado por ChatGPT, aplicado al HTML generado inicialmente.	83
Figura 36. Prueba CSS del primer prompt en ChatGPT.....	83
Figura 37. JavaScript del segundo prompt generado por ChatGPT.....	84

Figura 38. Registro exitoso de una mascota utilizando el código generado por ChatGPT.	85
Figura 39. Búsqueda de una mascota en el aplicativo generado por ChatGPT.	86
Figura 40. Códigos generados por ChatGPT integrados a Visual Studio Code.	86
Figura 41. Aplicativo generado con ChatGPT con respecto a la información requerida del tercer prompt.	87
Figura 42. Código generado del tercer prompt por ChatGPT, integrado a Visual Studio Code.	88
Figura 43. Prueba del segundo prompt en ingles dentro de la herramienta ChatGPT.	89
Figura 44. Función reconocimiento de voz en Gemini.	90
Figura 45. Código generado en Gemini con el primer prompt.	91
Figura 46. Resultado del HTML generado por Gemini para el segundo prompt.	91
Figura 47. Código HTML para el registro de mascotas con Gemini.	92
Figura 48. Código CSS generado por Gemini para el aplicativo.	93
Figura 49. Resultado de aplicar el CSS al aplicativo generado por Gemini.	93
Figura 50. JavaScript generado con Gemini, integrado a Visual Studio Code.	94
Figura 51. Práctica HTML del segundo prompt en Gemini.	94
Figura 52. HTML del tercer prompt en Gemini.	95
Figura 53. Prueba HTML del tercer prompt en Gemini.	96
Figura 54. Ejemplo de generación de código en inglés utilizando Gemini.	97
Figura 55. HTML del primer prompt en Bing.	99
Figura 56. Prueba HTML del primer prompt en Bing	99
Figura 57. Interfaz de registro de mascotas con Bing.	100
Figura 58. HTML del segundo prompt en Bing.	101
Figura 59. Prueba HTML del segundo prompt en Bing.	101
Figura 60. JavaScript del segundo prompt en Bing, integrado a Visual Studio Code.	102
Figura 61. HTML del tercer prompt en Bing.	102
Figura 62. Pruebas en inglés realizadas con la herramienta Bing.	104
Figura 63. Interfaz de usuario generada con prompts en ingles dentro de Bing Chat.	104
Figura 64. HTML del primer prompt en Replit.	105
Figura 65. Prueba CSS y JavaScript del primer prompt en Replit.	106
Figura 66. Segundo prompt en Replit.	106
Figura 67. Tercer prompt en Replit.	107
Figura 68. Resultados (Errores) en pruebas posteriores en Replit Ghostwriter.	107
Figura 69. Resultados de la evaluación	124
Figura 70. Análisis Topsis realizado utilizando la herramienta R Studio	172

Índice de tablas

Tabla 1. Ventajas y desventajas del reconocimiento de voz	31
Tabla 2. Comparación de antecedentes investigativos	41
Tabla 3. Ficha informativa de GitHub Copilot	60
Tabla 4. Ficha informativa de Replit GhostWriter	62
Tabla 5. Ficha informativa de ChatGPT	64
Tabla 6. Ficha informativa de AI2sql.....	65
Tabla 7. Ficha informativa de Spellbox.....	67
Tabla 8. Ficha informativa de Amazon CodeWhisperer	68
Tabla 9. Ficha informativa de Tabnine.....	70
Tabla 10. Ficha informativa de Bing Chat.....	71
Tabla 11. Ficha informativa de Gemini	73
Tabla 12. Ficha informativa de AlphaCode	74
Tabla 13. Análisis multicriterio con base a características técnicas	75
Tabla 14. Análisis multicriterio con base a características de usabilidad	76
Tabla 15. Resultados del análisis multicriterio	76
Tabla 16. Formato de evaluación de características para las herramientas IA seleccionadas en el estudio.....	78
Tabla 17. Resultados de la evaluación a ChatGPT	89
Tabla 18. Resultados de la evaluación a Bard (Gemini).....	98
Tabla 19. Resultados de la evaluación a Bing Chat	104
Tabla 20. Resultados de la evaluación a Replit Ghostwriter	108
Tabla 21. Puntuación multicriterio.....	165
Tabla 22. Ponderación multicriterio (Método TOPSIS).....	167
Tabla 23. Aplicación Método TOPSIS - Paso 1	168
Tabla 24. Aplicación Método TOPSIS - Paso 2	168
Tabla 25. Aplicación Método TOPSIS - Paso 3	169
Tabla 26. Aplicación Método TOPSIS - Paso 4	169
Tabla 27. Aplicación Método TOPSIS - Paso 4 (1).....	170
Tabla 28. Aplicación Método TOPSIS - Paso 5	170
Tabla 29. Aplicación Método TOPSIS - Paso 5 (1).....	170
Tabla 30. Aplicación Método TOPSIS - Paso 6	171

Dedicatorias

Dedico este trabajo de titulación de manera especial a mi madre, Hirma Espinoza, quien ha sido mi fuente de inspiración para superarme cada día. A mi padre, Ramiro Méndez, y a toda mi familia, gracias a su apoyo moral y económico, me han permitido alcanzar este logro tan significativo en mi vida profesional.

Dedico esta tesis también a mis tías Baniní y Petita, por estar siempre conmigo desde que tengo uso de razón y guiar mis pasos en lo espiritual, por ayudarme a crecer con buenos principios, enseñanzas, valores con los que he podido salir adelante en mi vida personal y estudiantil.

Agradecimientos

Agradezco a Dios por guiar mi camino a lo largo de estos años de esfuerzo y dedicación. A toda mi familia, por su apoyo constante e incondicional. A mis amigos y compañeros de la universidad por su motivación y acompañamiento, haciendo de este recorrido un viaje de aprendizaje, risas, entre otras actividades que han ayudado a formarme como persona.

Extiendo mi gratitud a mis tutores que han dado seguimiento constante a este proyecto, trabajando de la mano conmigo para cumplir con los objetivos planteados. Su orientación y conocimiento han sido fundamentales para alcanzar este logro.

Capítulo 1: Introducción

1.1 Antecedentes de investigación

1.1.1 Contextualización del desarrollo de software

El desarrollo de software ha ido tomando gran importancia en la era contemporánea, la demanda de sistemas informáticos cada vez más complejos y sofisticados es constante, y se debe sobre todo a la gran suma de recursos económicos que se destina para su desarrollo, el acelerado avance científico, la competencia entre compañías fabricantes y la sociedad de consumo. Durante el año 2021 el mercado de software llegó a los 565.000 millones de dólares, mientras que las predicciones hasta el año 2028 señalan que habría un crecimiento de aproximadamente el 12%, cifras que demuestran la jerarquía de este sector (Castro Rojas & Espitia Peña, 2022).

Dentro de este sector, la proliferación de herramientas basadas en inteligencia artificial (IA) ha revolucionado el enfoque tradicional hacia la generación de código, ofreciendo soluciones innovadoras que pueden interpretar y traducir requisitos de manera eficiente. Esta evolución tecnológica no solo facilita la implementación de sistemas más avanzados y adaptables, sino que también abre nuevas posibilidades en términos de velocidad y precisión en el desarrollo de software.

En este contexto, la generación de código a partir de requisitos basados en voz es una disciplina prometedora que busca optimizar el proceso de desarrollo, al tiempo que reduce la carga de trabajo y el tiempo, en relación con la programación manual; sin embargo, mientras que la generación de código promete una mayor accesibilidad y productividad, su implementación efectiva depende mayormente de la elección de herramientas adecuadas; por lo que además de la importancia de los desarrolladores de software también están los analistas e investigadores que dan luz sobre la efectividad de cada opción que se encuentra en el mercado.

1.1.2 Importancia de la generación de código

Uno de los avances más significativos en el área del desarrollo de software es la generación de código, que a su vez contribuye a este avance, debido a que permite automatizar gran parte del proceso de codificación, reduciendo la carga cognitiva del programador y liberándolo para centrarse en aspectos más creativos y de diseño eliminando la necesidad de codificación manual, lo que aumenta la productividad de los equipos de desarrollo, acelerando el tiempo de comercialización de los “productos y servicios” de software.

Las herramientas de generación de código basadas en inteligencia artificial (IA) proporcionan una interpretación precisa y coherente de los requisitos, reduciendo así la probabilidad de errores humanos y asegurando una mayor consistencia en el código producido. Mientras que el margen de error de una IA es inferior al 3%, el de una persona puede llegar hasta el 5% (Estrada et al., 2022).

Otro aspecto relevante es la accesibilidad que aporta la IA a personas con diferentes niveles de habilidad técnica, fomentando la diversidad en el campo de la tecnología, lo que es importante para impulsar la “innovación y el progreso” en la industria del software, mejorando cada vez más la forma como se desarrolla, la eficiencia, la precisión y la accesibilidad del proceso de desarrollo.

1.2 Problemática y justificación de la investigación

El problema de investigación gira sobre la poca información existente de herramientas IA adaptadas específicamente para la generación automática de código a partir de requisitos de voz, así como en la necesidad de comparar y analizar estas herramientas de manera profunda utilizando métodos científicos, ya que, debido a la novedad de estas tecnologías en el mercado y a la falta de garantías sobre su eficacia y fiabilidad varias de estas herramientas pueden no funcionar adecuadamente o pueden no cumplir con los estándares de calidad necesarios para su uso en entornos de desarrollo de software.

Además, cuando se dispone de herramientas de calidad, su costo puede ser elevado, lo que limita el acceso de muchas organizaciones y profesionales a estas tecnologías avanzadas, por lo tanto, es importante realizar una investigación exhaustiva para identificar, comparar y analizar las herramientas de IA disponibles para la generación de código. La investigación se enfoca en identificar opciones accesibles en términos de costo, ya sea código abierto, de bajo costo o gratuitas, asegurando que las herramientas recomendadas sean asequibles para una amplia gama de profesionales y organizaciones.

En este sentido, un análisis de las distintas herramientas de generación de código mediante comandos de voz es un paso fundamental para evaluar las funciones de cada una, y realizar una comparación sobre su uso, fiabilidad y otros factores relevantes, para crear una referencia sólida que ayude a los desarrolladores a entender las diversas herramientas disponibles, mediante una guía que ofrezca información detallada sobre las características, capacidades, ventajas y desventajas de cada una, permitiendo a los profesionales del desarrollo de software comprender mejor qué opciones se adaptan mejor a sus necesidades específicas y objetivos de un proyecto.

De igual manera, la investigación emplea métodos científicos para evaluar y comparar las herramientas definiendo criterios de evaluación claros y la aplicación de pruebas y análisis eficaces para medir el rendimiento, la eficacia y la calidad de cada una de ellas, y al hacerlo, se proporciona una evaluación imparcial que permitirá a los desarrolladores tomar decisiones informadas y basadas en datos.

Al proporcionar una evaluación ecuánime, la investigación permite a los desarrolladores tomar decisiones informadas sobre qué tecnologías son más pertinentes a adoptar, lo que minimiza la incertidumbre y el riesgo asociados con la selección de herramientas, lo que llevará a una implementación más eficiente y efectiva de soluciones tecnológicas en el desarrollo de software.

Además de evaluar la calidad y el rendimiento de cada herramienta, la investigación también se centra en identificar opciones más accesibles en términos de costo, lo que garantiza que las herramientas recomendadas no solo cumplan con los estándares, sino que también sean asequibles para una amplia gama de profesionales y organizaciones en el campo del desarrollo de software.

1.3 Preguntas de investigación

La investigación llevada a cabo en este trabajo de titulación, busca recopilar información sobre las fortalezas y debilidades de cada herramienta en diversos contextos y situaciones de uso. Además, pretende identificar los factores que influyen en la precisión de la conversión de voz a texto, así como destacar las limitaciones en la generación de código optimizado y en la gestión de las complejidades sintácticas y semánticas del lenguaje de programación empleado.

Con base a esto, las preguntas de investigación planteadas son las siguientes:

- ¿Qué herramientas de IA se pueden utilizar para obtener código fuente por medio de prompts de voz dictados por una persona?
- ¿Qué métodos se utilizaron para seleccionar las herramientas IA con base a requisitos de licencia gratuita, generación de código y dictado por voz?
- ¿Qué métodos se utilizaron para evaluar las herramientas generadoras de código?
- ¿Cuáles son los resultados más significativos obtenidos en las entrevistas efectuadas a los profesionales de software?

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar un análisis multicriterio de herramientas de generación de código basadas en Inteligencia artificial, con el fin de proporcionar una guía para la selección y aplicación efectiva de estas tecnologías en el desarrollo de aplicaciones a partir de requisitos basados en voz.

1.4.2 Objetivos Específicos

- Aplicar técnicas de toma de decisiones basadas en multicriterio para seleccionar un conjunto representativo de herramientas de inteligencia artificial utilizadas para la generación de código a partir de requisitos basados en voz.
- Evaluar el rendimiento a través de caso de estudio denominado “Software para el registro, gestión y tenencia responsable de mascotas en la ciudad de Cuenca”, en las que cada herramienta sea analizada en términos de precisión, eficiencia y calidad del código generado, con el propósito de establecer una clasificación.
- Elaborar una guía metodológica de aplicación de cada herramienta seleccionada, para la generación de código.

Capítulo 2: Marco Teórico

2.1 Generación de código automatizada

Para dar contexto a la investigación, es necesario explicar que el código es “un conjunto de instrucciones escritas en un lenguaje de programación” que un procesador puede entender y ejecutar para realizar una tarea específica. La generación de código es el proceso de producir automáticamente “instrucciones escritas en un lenguaje de programación” específico para el desarrollo de software, que tiene su origen en la primera mitad del siglo XX donde los programadores escribían código de forma manual (Agrawal et al., 2020).

Bajo este paradigma, la generación de código ha tenido una evolución con el paso de las décadas, siendo al principio desarrollado con el lenguaje ensamblador, que requería un profundo conocimiento de la arquitectura de la computadora y una gestión meticulosa de la memoria y los registros del sistema; hasta la introducción de lenguajes de programación de alto nivel como FORTRAN, COBOL y Lisp, que convierten el código escrito en un código máquina comprensible.

Por su parte la generación de código automatizada apareció en la década de los 60 y es el proceso en el que se utiliza software para crear automáticamente fragmentos de código de programación, en lugar de escribirlos de forma manual, utilizando algoritmos y reglas predefinidas que permiten generar código de forma eficiente y rápida, aprovechando patrones comunes, y conocimientos previos; cuya finalidad es evitar tareas repetitivas.

Este proceso puede aplicarse en diferentes niveles de abstracción, desde la generación de código de bajo nivel, como instrucciones de máquina o código ensamblador, hasta la generación de código de alto nivel, como código fuente en lenguajes de programación de propósito general que se utilizan en lenguajes de programación como Python, Java, C++, entre otras (Agrawal et al., 2020).

En la actualidad existen varias técnicas y enfoques para la generación de código automatizada, que van desde simples plantillas o macros hasta sistemas más complejos basados en IA y aprendizaje automático, que pueden variar en su nivel de automatización y flexibilidad, así como en su capacidad para adaptarse a diferentes contextos y requisitos de desarrollo de software.

2.1.1 Proceso de generación de código automatizada

El proceso de generación de código comienza con la identificación del modelo de origen (*ver Figura 1*), donde se crea un modelo conceptual que sirve como base, se pueden hacer ajustes

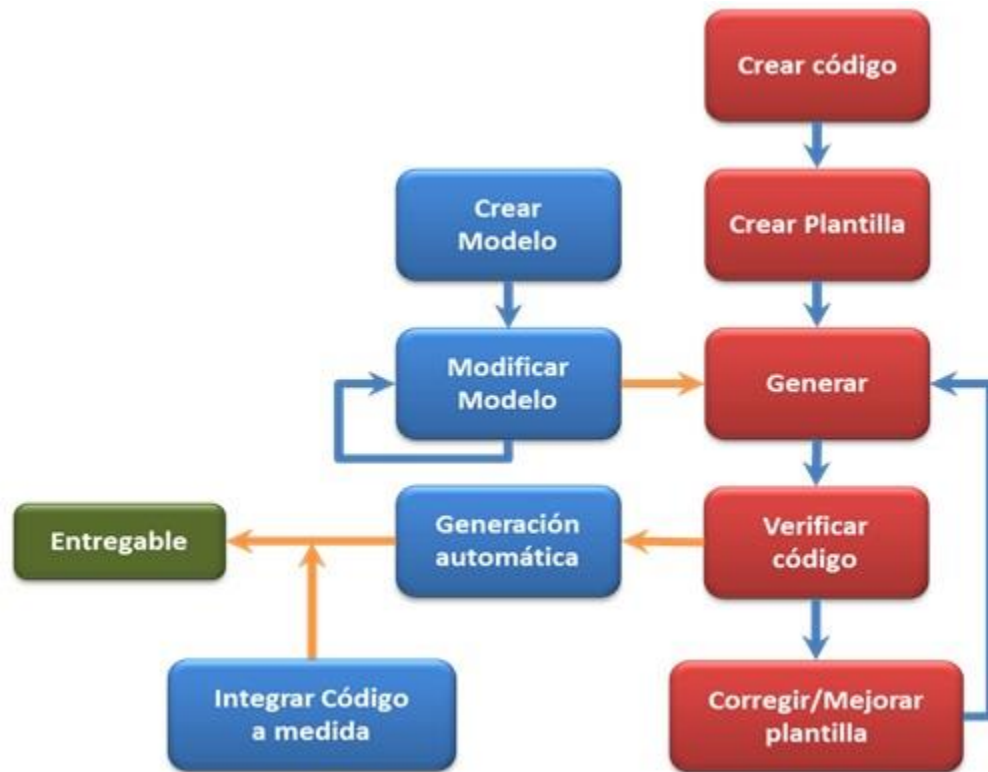
al modelo inicial si es necesario, se comienza a escribir el código basado en el modelo, desarrollando plantillas para automatizar la generación, especialmente para patrones repetitivos, a partir de las cuales se genera el código automáticamente, verificándose para asegurar su correcto funcionamiento y cumplimiento de requisitos; si se detectan errores, se ajustan las plantillas, y una vez corregido, el proceso puede ser automatizado utilizando herramientas de software especializadas (Ruiz, 2015).

Es importante señalar que, después de tener la plantilla, se utiliza un motor de generación para producir el código final, motor que toma como entrada el modelo de origen y la plantilla, y mediante una serie de reglas y algoritmos, rellena los espacios reservados de la plantilla con datos concretos, generando así el código completo. En este paso, es común el uso de comandos o scripts que automatizan el proceso de generación (Ronanki et al., 2023).

Una vez generado el código, se lleva a cabo un proceso de verificación para el cumplimiento de los requisitos y verificación de errores. Esta verificación puede incluir pruebas automáticas, revisiones manuales o análisis estático del código. Si se encuentran errores o deficiencias, se realizan las correcciones necesarias (Ronanki et al., 2023).

Finalmente, el código generado y verificado se entrega como el producto final del proceso de generación. Este código puede ser utilizado directamente en el desarrollo de software o integrado en sistemas más complejos. La automatización en la generación de código acelera el desarrollo de software al disminuir la necesidad de trabajo manual y al aumentar la coherencia y la calidad del código producido (Ronanki et al., 2023).

Figura 1. Proceso de generación automática de código



Fuente: <https://itblogsogeti.com/2015/02/10/desarrollo-software-david-ruiz-sogeti/>

2.1.2 Ventajas de la generación de código automatizada

(Carbonell-García et al., 2023) indican que la generación de código automatizada tiene varias ventajas, entre las más relevantes se encuentran:

- Incremento de la productividad, permitiendo a desarrolladores generar rápidamente grandes cantidades de código, reduciendo el tiempo y el esfuerzo requeridos para implementar funcionalidades específicas.
- Reducción de errores, al automatizar parte del proceso de codificación, se minimiza el riesgo de errores humanos y se mejora la calidad y consistencia del código generado.
- Reutilización de código, ya que puede facilitar la reutilización de componentes de software existentes, lo que ayuda a acelerar el desarrollo y reduce la duplicación de esfuerzos.
- Adaptabilidad, pues pueden adaptarse fácilmente a cambios en los requisitos o especificaciones del proyecto, lo que permite una mayor flexibilidad y agilidad en el desarrollo de software.

A pesar de las ventajas que tiene, la generación de código automatizada también presenta inconvenientes, siendo uno de ellos, la complejidad en la configuración y personalización de las herramientas que dispone, que, si bien son eficientes una vez configuradas correctamente, el proceso de definir reglas, parámetros y transformaciones puede ser complicado y requerir un conocimiento técnico avanzado.

Otro desafío es la necesidad de mantener actualizadas las reglas y plantillas, para adaptarse a cambios en los requisitos del software o en las tecnologías utilizadas a medida que evoluciona el proyecto y se introducen nuevas funcionalidades o cambios, es necesario ajustar y modificar las reglas de generación para garantizar que el código generado siga siendo válido y efectivo, lo cual requiere de esfuerzo adicional y supervisión constante por parte de los desarrolladores (Manjarrés-Betancur y Echeverri-Torres, 2020).

En algunos casos la generación de código automatizada puede generar código redundante o no óptimo, es posible que no siempre logren optimizar completamente el código generado y esto puede resultar en la producción de código innecesariamente largo, complejo o ineficiente, lo que puede dificultar la comprensión, mantenimiento y depuración del software resultante.

2.2 Herramientas de inteligencia artificial para generación de código

Las herramientas IA para generación de código son sistemas que utilizan técnicas de dicha rama informática, entre ellos el “aprendizaje automático y el procesamiento de lenguaje natural”, para automatizar o asistir en el proceso de creación de código, ya sea de forma completa o partes específicas, a partir de diferentes tipos de entradas, como descripciones en lenguaje natural, ejemplos de entrada y salida, o incluso imágenes (Lau y Guo, 2023).

En términos más sencillos, estas herramientas son como asistentes virtuales que pueden ayudar a los programadores a escribir código más rápido y eficiente, utilizando modelos de lenguaje natural (PLN), aprendizaje automático o redes neuronales artificiales. Los modelos de lenguaje natural son capaces de comprender las intenciones del usuario expresadas en lenguaje humano y transformarlas en código fuente. Por su parte, los algoritmos de aprendizaje automático pueden analizar grandes volúmenes de código existente para identificar patrones y reglas, que posteriormente se utilizan para generar nuevo código (Lau y Guo, 2023).

Por su parte, las redes neuronales, son un tipo de modelo de aprendizaje automático, utilizadas en la generación de código para aprender la estructura y la sintaxis del código a partir de ejemplos, pues algunas herramientas permiten a los usuarios proporcionar ejemplos de entrada

y salida para la funcionalidad que desean implementar, y luego utilizan técnicas de IA para generar automáticamente el código necesario para producir esos resultados.

Estas herramientas de IA para la generación de código pueden ser útiles en diversas situaciones y proyectos, desde el desarrollo rápido de prototipos hasta la automatización de tareas repetitivas. No obstante, es fundamental recordar que estas herramientas están en continuo desarrollo y, aunque no pueden sustituir completamente la intervención humana y la experiencia de los programadores, los avances recientes son significativos (Carbonell-García et al., 2023).

2.2.1 Desarrollo de la inteligencia artificial

La inteligencia artificial se define como la capacidad de un sistema mecánico para recibir, procesar y aplicar conocimientos y habilidades adquiridos. Inicialmente, su objetivo era asistir o sustituir tareas humanas, lo que llevó a la creación de tecnología informática y avances en la navegación y la industria (Berryhill et al., 2020).

Los antecedentes de la IA se remontan a partir del siglo XV, donde se inicia una revolución científica y tecnológica que cambia la comprensión del mundo y se originan numerosos inventos que transformaron la sociedad industrial en postindustrial, mientras que, en la segunda mitad del siglo XX, a través de la cuarta revolución científica y tecnológica, se produjo un desarrollo significativo de la actividad intelectual, incluida la introducción de la inteligencia artificial en las nuevas tecnologías (Berryhill et al., 2020).

A partir de 1980, la IA tuvo un resurgimiento significativo después del período de escepticismo en la década de 1970, durante este tiempo, hubo avances importantes en varias áreas de la IA, impulsados por el aumento de la capacidad tecnológica y el interés renovado en la aplicación práctica, donde se realizaron investigaciones en áreas como el razonamiento basado en casos, la visión informática y los sistemas expertos, con un enfoque en el desarrollo de aplicaciones comerciales que introdujeron nuevas técnicas de aprendizaje automático, como las redes neuronales multicapa y los algoritmos genéticos, permitiendo el procesamiento de datos más complejos y la resolución de problemas (Erazo-Luzuriaga et al., 2023).

Durante la década de 1990, la IA se consolidó aún más con la introducción de sistemas de recomendación, reconocimiento de voz y visión por computadora en aplicaciones comerciales y de consumo, desarrollando algoritmos más sofisticados de aprendizaje automático, entre ellos las “máquinas de vectores de soporte y las redes neuronales convolucionales”, que permitieron avances en áreas relacionadas a la IA entre ellas el procesamiento de lenguaje natural; que

impulsó además la inversión en IA y la integración de tecnologías para el manejo de productos y servicios.

A principios del siglo XXI, la IA experimentó un crecimiento exponencial, impulsado por avances en el procesamiento de una gran cantidad de datos y el desarrollo de algoritmos de aprendizaje profundo, en áreas como la conducción autónoma, la robótica y los sistemas de recomendación personalizada, provocando un cambio en la forma de vida, continuando como objeto de una investigación activa y desarrollo en campos como la medicina, logística y la educación (Becker et al., 2023).

En términos de aplicaciones, la inteligencia artificial tiene numerosas ventajas, como el diagnóstico de enfermedades, el apoyo en el ámbito jurídico, el análisis de un gran volumen de datos y la eficiencia en diversas industrias; sin embargo, también plantea amenazas, como el desempleo masivo, la pérdida de control sobre la IA y el desarrollo de conflictos socioeconómicos y religiosos.

(Corvalán, 2018) menciona que hasta la actualidad la IA se divide en 5 tipos, que son:

- a. **Generativa**, es como una especie de creatividad artificial, a la cual puedes dar órdenes, datos o imágenes y pedir que genere información o un producto a partir de esta; es decir, es una IA que funciona basándose en lo que ha aprendido de la información anterior.
- b. **Fuerte**, es aquella capaz de pensar y resolver problemas de manera muy parecida a como lo haría un humano, ya que puede entender el contexto, mediante la experiencia y adaptación a nuevas situaciones de forma independiente.
- c. **Explicable**, se refiere a sistemas de IA que pueden explicar de manera comprensible cómo llegan a sus decisiones o conclusiones; esto significa que, cuando una IA realiza una acción o proporciona un resultado, puede explicar de forma transparente los procesos y factores que la llevaron a esa conclusión.
- d. **Amigable**, está diseñada para trabajar junto a los humanos de manera segura y colaborativa, pues, no solo realiza tareas, sino que también considera las necesidades de las personas con las que interactúa.
- e. **Multimodal**, es el avance de la IA, que puede entender diferentes las formas indicadas en los otros tipos. No solo puede entender palabras, sino también imágenes, sonidos e incluso gestos y es capaz de procesar información de múltiples fuentes para comprender el mundo que la rodea de una manera más completa.

2.2.2 Técnicas de generación de código

a) Generación de código basada en plantillas

La generación de código basada en plantillas es una técnica comúnmente utilizada en el desarrollo de software que implica la creación de plantillas predefinidas que contienen fragmentos de código con espacios reservados para datos específicos, que actúan como moldes del código final que se generará, donde en lugar de escribir manualmente todo el código desde cero, los desarrolladores pueden utilizar estas plantillas como base y luego rellenar los espacios reservados con valores concretos según sea necesario, lo que permite generar código de manera más rápida y consistente, ya que los patrones y estructuras comunes pueden ser reutilizados en múltiples partes del proyecto (Adriano y Guapi, 2023).

Esta técnica es bastante útil en situaciones donde se necesita generar código repetitivo, que se denomina como “boilerplate”, donde se realiza creación de clases, métodos o interfaces con una estructura estándar. Además, puede facilitar la adopción de mejores prácticas de desarrollo al proporcionar plantillas que siguen convenciones de codificación y estilos de arquitectura específicos (Adriano y Guapi, 2023).

Para utilizar esta técnica, los desarrolladores pueden emplear herramientas específicas de generación de código o simplemente crear sus propias plantillas utilizando un lenguaje de marcado. Las plantillas pueden contener “placeholders” o marcadores de posición que indican dónde se insertarán los datos concretos durante el proceso de generación, que pueden ser simples etiquetas de texto o pueden incluir lógica más compleja para generar código dinámicamente.

b) Generación de código mediante modelado

La generación de código mediante modelado es una técnica en la que se utiliza un modelo abstracto para describir la estructura y el comportamiento del sistema que se desea implementar y que puede ser representado en diferentes formas, como diagramas UML (Lenguaje Unificado de Modelado), modelos de datos, diagramas de flujo, entre otros. Una vez que se ha creado este modelo abstracto, se utiliza un proceso de transformación para traducir el modelo en código concreto en uno o varios lenguajes de programación (Larrosa et al., 2019).

El proceso de transformación implica la definición de reglas y transformaciones que especifican cómo se debe traducir cada elemento del modelo en código real. Estas reglas pueden incluir la asignación de clases a archivos de código fuente, la traducción de relaciones entre clases en

relaciones de dependencia en el código, la generación de métodos y funciones a partir de operaciones definidas en el modelo, entre otros aspectos (Larrosa et al., 2019).

En este sentido, esta técnica permite mantener una separación clara entre la especificación del sistema y su implementación, lo que facilita la comprensión y la modificación del sistema a lo largo del tiempo. Además, al utilizar un modelo abstracto como punto de partida, los desarrolladores pueden centrar su atención en la lógica del negocio y los aspectos funcionales del sistema, sin preocuparse demasiado por los detalles de implementación.

Esta técnica es especialmente útil en el desarrollo de software basado en modelos (Model-Driven Development, MDD), donde el modelo abstracto sirve como fuente de verdad para la implementación del sistema. Además, la generación de código mediante modelado permite la reutilización de patrones de diseño y componentes de software a través de la creación de bibliotecas de modelos y transformaciones reutilizables.

c) Generación de código a partir de ejemplos

La generación de código a partir de ejemplos es la técnica que se basa en la idea de aprender de ejemplos concretos para generar código automáticamente, en lugar de escribir manualmente el código, los desarrolladores proporcionan ejemplos de entrada y salida que representan el comportamiento deseado del sistema, que pueden ser casos de prueba, datos de entrenamiento o simplemente pares de entrada y salida que ilustran la funcionalidad requerida (Franganillo, 2023).

Una vez que se proporcionan los ejemplos, se utilizan técnicas de aprendizaje automático o procesamiento de lenguaje natural para procesar estos ejemplos, que tiene como objetivo identificar patrones, reglas y correlaciones entre los datos de entrada y salida que puedan ser utilizados para generar automáticamente el código correspondiente (Franganillo, 2023).

En este sentido, si se desea crear una función que calcule el doble de un número, se podrían proporcionar ejemplos como (2, 4), (3, 6) y (5, 10), donde el primer número representa la entrada y el segundo número representa la salida esperada, por lo que, utilizando estos ejemplos, un algoritmo de aprendizaje automático podría inferir que la función simplemente multiplica la entrada por dos y generar automáticamente el código correspondiente.

Entre las ventajas más importantes de la técnica descrita está el permitir a los desarrolladores expresar sus intenciones de forma intuitiva y fácilmente entendible, sin necesidad de conocimientos especializados en programación. Además, esta técnica puede acelerar el proceso

de desarrollo al hacer automáticas las tareas que suelen ser repetitivas y rutinarias. También puede mejorar la calidad del código generado al minimizar el riesgo de errores humanos.

Sin embargo, este tipo de generación de código también presenta desafíos y limitaciones, como que, puede ser difícil capturar todos los casos de uso posibles con ejemplos limitados, lo que puede llevar a la generación de código incompleto o incorrecto, y, la calidad del código generado puede depender en el mayor de los casos de la calidad de los ejemplos proporcionados y la capacidad del algoritmo de aprendizaje automático para generalizar a partir de estos ejemplos.

d) Generación de código basada en reglas y patrones

Como forma generación de código la técnica basada en reglas y patrones implica la definición de reglas y patrones que describen cómo debe ser generado el código en función de ciertas condiciones o características del sistema, que pueden ser realizados manualmente o pueden ser aprendidos automáticamente a partir de ejemplos existentes de código mediante “técnicas de minería de datos o aprendizaje automático” (Tejera et al., 2020).

Es decir, se establecen reglas que especifican qué código debe generarse en función de ciertas condiciones o criterios, que, a su vez, están basadas en convenciones de codificación, patrones de diseño, requisitos del sistema o cualquier otro factor relevante para la generación del código. Además de las reglas, se establecen patrones que describen estructuras comunes o fragmentos de código reutilizables en diversas partes del sistema, incluyendo plantillas de código, macros, funciones de utilidad u otros recursos útiles para la generación de código. Una vez definidas las reglas y los patrones, se emplea un generador de código o un motor de plantillas para aplicar estos elementos y generar automáticamente el código correspondiente. Este proceso puede ser totalmente automatizado o puede necesitar la intervención manual de los desarrolladores para personalizar o ajustar el código generado según sea necesario.

Esta técnica puede mejorar la productividad al automatizar tareas repetitivas y rutinarias de desarrollo de software, manteniendo la consistencia y la calidad del código generado al aplicar convenciones de codificación y patrones de diseño establecidos. Además, facilita la reutilización de código al proporcionar un conjunto de reglas y patrones que pueden ser aplicados en diferentes partes del sistema.

2.2.3 Generación de código a partir de prompts

Se pueden encontrar gran variedad de formas para indicarle a una herramienta de IA el contenido del código que se espera como resultado luego de haber colocado la instrucción, es así que

investigaciones como (White et al., 2023) donde mencionan patrones como Output Automator que son capaces de generar software en ChatGPT mediante una entrada completa y concreta, se indican además aspectos como la declaración del alcance conversacional inicial que se tiene que dar como contexto al aplicativo, adicional a esto con ayuda de frases como “de ahora en adelante”, “durante el resto de la conversación” con las que se puede iniciar la instrucción a proveer en la herramienta, se consigue preparar de una forma más completa los resultados que se espera obtener durante el proceso de interacción. Entre los patrones mencionados según la circunstancia específica del usuario en la herramienta generadora de código, se encuentra el patrón generador de API con el que se puede tener una especificación de “interfaz de programación de aplicaciones”, que a pesar de estar orientado a especificaciones de API REST, se puede también aplicar a sistemas CRUD que impliquen trabajar de forma conjunta con las funcionalidades de interfaces y diseño con la parte lógica.

En el estudio realizado por (De Vito et al., 2023) se habla de ECHO como un enfoque para mejorar la calidad de casos de uso dentro de modelos de lenguaje grandes (LLM: large language model), donde las indicaciones son diseñadas de forma adecuada para interactuar con ChatGPT, se menciona criterios de importancia como la corrección para que el caso de uso pueda adecuarse con precisión según el comportamiento que se espera que tenga el sistema, el orden correcto de los pasos y la atomicidad en el proceso, otro criterio es lo exhaustivo para cubrir escenarios posibles, alternativos y de error. El caso de uso debe seguir una secuencia lógica de eventos, afinar en los detalles del requerimiento y el uso de términos coherentes. Se afirma también que la efectividad de los resultados de un LLM se encuentra directamente relacionada con la calidad de indicaciones dadas por los usuarios, como ejemplo se tiene el caso de que una persona brinde varias versiones de una misma indicación, que además pueden tener el mismo significado pero que la herramienta de IA lo puede interpretar de manera distinta; esto debido a un contexto manejado por el LLM donde se emplean varias oraciones con una misma intención.

(De Vito et al., 2023) menciona sobre el enfoque ECHO y sus dos fases para generación de modelos de calidad basados en casos de uso, una es la ingeniería del método de co-incitacion donde es importante indicar de manera inicial el rol que va a tener la herramienta IA, con el uso de frases: “Actúa como un Ingeniero de Software”, de este modo, la IA asumirá aquel rol, esto con el fin de obtener una respuesta completa y detallada para cada criterio a colocar. La división de los casos de uso ayudará a que la herramienta genere código más completo y sugerencias que permitan establecer mejoras conforme se avanza con la interacción, se menciona también

que en el caso de la herramienta ChatGPT se tiene limitaciones en el contexto de conversación con un número máximo de tokens de 4096 que incluyen entradas y salidas.

2.2.4 Aplicación práctica de las herramientas de IA de generación de código

El desarrollo tecnológico ha transformado radicalmente la sociedad, proporcionando numerosos beneficios en todos los ámbitos, siendo una de las áreas en las que este progreso ha sido notable el desarrollo de software, donde las herramientas IA están cada vez más evolucionando los métodos que usan para crear y mantener el código, donde este tipo de herramientas ofrecen una serie de ventajas, incluyendo la generación automática, la cual, agiliza el proceso, reduce los errores cometidos por humanos y mejora a su vez la calidad del producto final (Celi-Parraga et al., 2021).

Así mismo, en la generación de código, las IA han demostrado ser bastante útiles en la creación de aplicaciones complejas y en la adaptación a diferentes contextos y tecnologías, pues pueden analizar grandes conjuntos de datos y aprender patrones complejos para generar código que se adapte a las necesidades específicas del proyecto y facilitar la colaboración entre equipos de desarrollo, al brindar un marco común para la escritura y la estructuración del código (Gamboa, 2019).

Por consiguiente, la aplicación práctica que aportan las IA a la generación automatizada de código, se puede señalar en los siguientes aspectos:

- **Automatización del desarrollo de software**, es un proceso que “implica el uso de herramientas y técnicas” para reducir la intervención manual en todas las “etapas del ciclo de vida del desarrollo de software”, que abarca desde la planificación y el diseño hasta la implementación, las pruebas y el despliegue de aplicaciones; teniendo como objetivo mejorar la eficiencia, reducir los errores humanos, acelerar el tiempo de entrega de los proyectos y mejorar la calidad del software resultante (Celi-Parraga et al., 2021).

En este ámbito, las IA han facilitado la creación de código al automatizar la generación de secciones repetitivas o rutinarias. Utilizando herramientas de generación de código basadas en plantillas, reglas y patrones, los desarrolladores pueden producir clases, métodos y estructuras de datos de forma más rápida y consistente. Esto libera tiempo para que los programadores se concentren en tareas más complejas y creativas, como el diseño de la arquitectura del software y la resolución de problemas.

Además, “la automatización facilita la integración continua y la entrega continua (CI/CD, por sus siglas en inglés)”, procesos fundamentales en el desarrollo de software moderno; y debido a que estos procesos conllevan la automatización de la compilación, las pruebas y el despliegue de código en entornos de desarrollo, pruebas y producción, automatizar estas tareas, acelera el ciclo de desarrollo, permitiendo la entrega frecuente y confiable de nuevas características y correcciones de errores.

- **Aumento de la productividad**, considerando que la productividad tiene que ver con la capacidad de obtener más resultados en menos tiempo y con menos recursos, en el área de desarrollo de software, el aumento de la productividad es fundamental para satisfacer las demandas del mercado, cumplir con plazos ajustados y mejorar la competitividad de una organización, manteniendo al mismo tiempo la calidad del trabajo realizado, las IA otorgan resultados directos de la aplicación de diversas estrategias y técnicas que optimizan el proceso de creación de aplicaciones y productos (Gamboa, 2019).

Una de las principales formas en que se logra el aumento de la productividad es a través de la “automatización de tareas repetitivas y rutinarias”, como, por ejemplo, la generación automática de código, la ejecución de pruebas automáticas, la integración y la entrega continua (CI/CD), entre otros procesos automatizados, que son parte de acciones que cumplen ciertas herramientas IA.

Además, el aumento de la productividad, gracias a las herramientas IA, se logra mediante la adopción de metodologías ágiles y enfoques colaborativos de trabajo; estas metodologías, promueven la iteración rápida, la retroalimentación continua y una adaptación flexible ante los cambios, lo que permite a los equipos de desarrollo dar respuesta efectiva a las necesidades cambiantes del mercado y del cliente; del mismo modo, el trabajo en equipo y la colaboración activa entre los miembros del equipo ayudan a compartir conocimientos, resolver problemas más rápidamente y promover un ambiente de trabajo positivo y motivador.

Otro factor que contribuye al incremento de la productividad y que implica herramientas IA, es la adopción de tecnologías que simplifican y agilizan el desarrollo de software, como el uso de entornos de desarrollo integrados (IDE), plataformas de gestión de proyectos, sistemas de control de versiones y herramientas de colaboración en línea, mismas que proporcionan funcionalidades avanzadas, como la escritura de código asistida, la depuración integrada, la gestión automatizada de cambios y la comunicación en tiempo real, mejorando la eficiencia y la calidad del trabajo realizado.

- **Mejora de la calidad del código**, es un aspecto esencial en el desarrollo de software, ya que causa un impacto de forma directa en la fiabilidad y el rendimiento del producto, ya que este cumple con los estándares de diseño, estructura y funcionamiento, así como su capacidad para ser entendido, modificado y ampliado por otros desarrolladores en el futuro, pero que también implica adoptar prácticas y técnicas que promuevan la claridad, la consistencia, la simplicidad y la eficiencia del mismo (Llanos et al., 2021).

Entre las formas más efectivas de mejorar la calidad del código es seguir buenas prácticas de codificación y aplicar patrones de diseño probados, siguiendo convenciones de codificación estándar, como nombrar variables y funciones de manera descriptiva, mantener una estructura lógica y coherente en el código, y evitar la duplicación innecesaria de código.

Otra tarea importante para mejorar la calidad del código es realizar pruebas exhaustivas y continuas a lo largo del proceso de desarrollo, como la implementación de pruebas unitarias, de integración y de aceptación automatizadas, que ayudan a identificar y corregir errores de manera temprana en el ciclo de desarrollo, garantizando que el código funcione correctamente y cumpla con los requisitos especificados (Llanos et al., 2021).

Además, al mejorar la calidad del código se logra mediante la adopción de prácticas de refactorización regular, que se refiere a mejorar o reestructurar código que ya existe evitando cambiar la forma de comportamiento, con el objetivo de hacerlo más claro, legible, mantenible y eficiente, y que ayuda a eliminar la deuda técnica, reducir la complejidad del código y mantenerlo actualizado con buenas prácticas y mediante estándares propios de la industria.

- **Adaptación a diferentes contextos y tecnologías**, es importante debido a la diversidad de plataformas, frameworks y requisitos que existen en la actualidad, y que se relaciona a la capacidad de los desarrolladores para ajustarse y trabajar eficazmente en entornos diversos, ya sea en términos de infraestructura de hardware, sistemas operativos, dispositivos móviles, web, nube, entre otros, así como en el uso de diferentes tecnologías y lenguajes de programación.

En ese sentido el desarrollo con IA, mediante el uso de tecnologías y herramientas versátiles y compatibles con múltiples plataformas y contextos es más sencillo; algunos frameworks y lenguajes de programación, como JavaScript, Python o Java, son ampliamente utilizados debido a su capacidad para trabajar en una variedad de entornos y contextos. Además, el uso de herramientas de desarrollo multiplataforma, como Visual Studio Code, Eclipse o IntelliJ IDEA, permite a los desarrolladores trabajar en diferentes sistemas operativos y entornos de desarrollo sin problemas.

Otra forma de adaptación es mediante la adopción de metodologías y prácticas de desarrollo flexibles y adaptables, lo que ayuda a trabajar de manera eficiente y colaborativa en entornos complejos y dinámicos, adaptándose rápidamente a los cambios y requisitos del mercado. La automatización de procesos de desarrollo, pruebas y despliegue también es fundamental para garantizar una adaptación ágil y eficaz a diferentes contextos y tecnologías.

Hay que tomar en cuenta que, la formación continua y la actualización de conocimientos son clave para adaptarse a los avances tecnológicos y a los cambios en el panorama de desarrollo de software; estar al tanto de las últimas tendencias, herramientas y tecnologías, y estar dispuestos a aprender y experimentar con nuevas tecnologías es necesario para satisfacer las demandas del mercado y los requisitos de un proyecto (Figuerola y Postigio, 2023).

- **Facilidad de aprendizaje**, disponer de herramientas, recursos y procesos que fomenten las formas de comunicación, como se intercambian los conocimientos y colaborando activamente dentro de los equipos de desarrollo, este es un aspecto en el que aportan las IA, al disponer de información bastante amplia.

Igualmente, el usar herramientas, tales como los sistemas de gestión de proyectos, plataformas de comunicación y repositorios de código, facilita la colaboración remota y la comunicación en tiempo real entre los miembros de un equipo. Estas herramientas permiten a los desarrolladores compartir ideas, revisar el trabajo de otros, colaborar en proyectos y mantener un registro de los cambios realizados en el código, lo que promueve la transparencia y la cohesión en el equipo.

Otro aspecto importante es la promoción de un ambiente de aprendizaje continuo, donde los desarrolladores puedan adquirir nuevas habilidades, explorar nuevas tecnologías y compartir su conocimiento con otros. Esto puede lograrse mediante la organización de sesiones de formación y talleres internos, participando en conferencias, incluyendo también los eventos de la industria, y la asignación de tiempo para la investigación y experimentación en proyectos personales.

Además, la creación de una cultura de retroalimentación constructiva y apertura a la crítica ayuda a los desarrolladores a aprender de sus errores, mejorar continuamente su trabajo y crecer profesionalmente. Al fomentar un ambiente donde se valoren la colaboración, las formas de aprendizaje y crecimiento, las empresas pueden construir equipos de desarrollo altamente eficientes, motivados y capacitados para enfrentar los desafíos del desarrollo de software en un entorno con altos niveles de complejidad y competitividad.

2.3 Reconocimiento de voz en las IA

El reconocimiento de voz permite comprender y procesar el habla humana dentro de un procesador, que conlleva un proceso de convertir las señales de audio de voz texto escrito o comandos entendibles por una computadora; lo que supone varias etapas, partiendo de la captura del audio, el procesamiento del sonido, y la conversión del habla en texto utilizando algoritmos y modelos de reconocimiento (Yao, 2021).

Al día de hoy, esta tecnología se utiliza en varias aplicaciones y dispositivos, como sistemas de asistencia virtual en teléfonos inteligentes, sistemas de dictado en software de procesamiento de texto, sistemas de control de voz en automóviles y en la accesibilidad para personas con discapacidad, automatización de tareas en entornos industriales entre otros usos, que la han convertido en una herramienta perfecta para el desarrollo tecnológico.

De hecho, el reconocimiento de voz ha avanzado tanto en las últimas décadas, a través del desarrollo de algoritmos de inteligencia artificial y el incremento en la capacidad de procesamiento de las computadoras, que se ha transformado en un asistente personal dentro de los hogares y para llevar a cabo acciones cotidianas en equipos de tecnología, únicamente por orden de voz del usuario (Yao, 2021).

La precisión en el reconocimiento de voz ha ido mejorando de forma significativa a lo largo de los años, gracias a la integración de algoritmos de aprendizaje profundo y conjuntos de datos masivos, haciendo que los sistemas comprendan una amplia gama de acentos, dialectos y patrones de habla, lo que resulta en una transcripción más precisa de las palabras habladas. Sin embargo, aunque estos sistemas han alcanzado niveles impresionantes de precisión, aún pueden surgir situaciones en las que se produzcan errores.

Además de la precisión en el reconocimiento, los sistemas de reconocimiento de voz han avanzado en el procesamiento natural del lenguaje (PNL), lo que les permite comprender el significado detrás del conjunto de palabras y un contexto establecido. Esto ha dado lugar al desarrollo de asistentes de voz inteligentes capaces de mantener conversaciones significativas con los usuarios y realizar tareas basadas en comandos de voz. Los métodos de PNL han permitido a los sistemas entender el contexto, identificar objetos y responder a las solicitudes de manera más intuitiva, lo que ha mejorado la experiencia del usuario y ampliado las posibilidades de interacción con la IA.

2.3.1 Proceso de reconocimiento de voz

El proceso de reconocimiento de voz se da mediante tres fases principales, que inician una vez se ha emitido la orden vocal; y estas son:

- a) Adquisición de señales, es el primer paso, que captura las formas de onda de sonido utilizando un micrófono, en una señal de audio que contiene la voz humana que se desea reconocer, donde el micrófono convierte el sonido formado por ondas en señales eléctricas que luego pueden ser procesadas por una computadora u otro dispositivo electrónico.
- b) Preprocesamiento, una vez capturada la señal de voz, se lleva a cabo un preprocesamiento para mejorar la calidad del sonido y eliminar posibles interferencias, como el ruido de fondo o el eco, realizando la eliminación de ruido mediante filtros digitales, la cancelación de eco para eliminar reverberaciones no deseadas y la normalización del volumen para asegurar que la señal esté dentro de un rango adecuado para su procesamiento posterior.
- c) Extracción de características, sigue luego del preprocesamiento y en ella se extraen características relevantes de la señal de voz, identificando información útil en la señal procesada, como características de frecuencia, duración y amplitud, en la que se utilizan técnicas de análisis espectral para identificar picos de frecuencia que corresponden a las vocalizaciones en el habla humana, características fundamentales para distinguir entre diferentes sonidos y reconocer patrones que corresponden a palabras y frases específicas.

2.3.2 Beneficios y dificultades del reconocimiento de voz

Los comandos de voz ofrecen beneficios que son significativos en términos de eficiencia y comodidad, principalmente la rapidez con la que se pueden ejecutar tareas utilizando la voz, lo que ahorra tiempo y esfuerzo al evitar la necesidad de escribir o interactuar manualmente con dispositivos, lo que es útil en situaciones en las que especialmente las manos están ocupadas o cuando se necesita una respuesta rápida, como en la conducción o en entornos de trabajo activos.

Además, los comandos de voz permiten la multitarea de manera más fluida, ya que los usuarios pueden realizar acciones mientras mantienen su enfoque en otras tareas, ya que, un usuario puede enviar mensajes de texto mientras camina o realizar búsquedas en internet mientras realiza una tarea doméstica, mejorando la productividad al permitir que las personas realicen múltiples acciones simultáneamente sin interrumpir su flujo de trabajo (Aleman et al., 2019).

Otro beneficio importante es el aprendizaje automático integrado en los sistemas de reconocimiento de voz, debido a que sus diseños están hechos para adaptarse y mejorar con el

tiempo; los algoritmos utilizan el aprendizaje automático para ajustar sus modelos basados en el análisis de datos de voz y texto, lo que resulta en una mayor precisión y comprensión con el uso continuado, adaptándose a las preferencias individuales y al estilo de habla de los usuarios (Alemán et al., 2019).

Sin embargo, a pesar de estos beneficios, los comandos de voz también enfrentan una serie de complicaciones y desafíos, siendo uno de sus principales problemas que no están libres de errores y pueden malinterpretar los comandos, especialmente en entornos ruidosos o con acentos difíciles de entender, generando frustraciones por parte del usuario y a la necesidad de repetir los comandos, lo que contrarresta la eficiencia que se supone que deben proporcionar (Figuerola y Postigio, 2023).

Además, el ruido de fondo y otras interferencias pueden afectar negativamente la precisión del reconocimiento de voz, lo que dificulta aún más la comunicación efectiva con los dispositivos, muchas veces devolviendo una respuesta equivocada. En entornos ruidosos, como en la calle o en lugares concurridos, los sistemas de voz pueden tener dificultades para distinguir entre la voz del usuario y el ruido ambiental.

Otro de los aspectos importantes que se deben considerar es la seguridad y la privacidad, de datos, que se han tornado frágiles con la evolución tecnológica, y los comandos de voz son herramientas que representa un riesgo potencial para la seguridad si se utilizan en entornos públicos o compartidos, ya que otras personas pueden escuchar los comandos y acceder a información sensible o realizar acciones no deseadas. Además, existe la preocupación de que los dispositivos de voz puedan ser vulnerables a ataques cibernéticos o piratería, lo que compromete la privacidad y sobre todo la seguridad de los usuarios (Tomalá et al., 2023).

Tabla 1. *Ventajas y desventajas del reconocimiento de voz*

VENTAJAS Y DESVENTAJAS DEL RECONOCIMIENTO DE VOZ	
Ventajas	Desventajas
Facilidad de uso y rapidez	Precisión invariable, pueden generar errores
Mejorar la accesibilidad	Incompatibilidad con algunos idiomas o dialectos
Es una herramienta multitarea	Es frágil ante el ruido ambiental

Posee un algoritmo de aprendizaje automático	Existen riesgos en la privacidad y la seguridad de datos
Mayor eficiencia y productividad	Tiene dependencia de conexión a red de internet

2.4 Aspectos de generación de código por IA a partir de requisitos basados en voz

2.4.1 Automatización

La automatización para la generación de código de IA por comando de voz implica algoritmos y técnicas de aprendizaje automático que crean automáticamente segmentos de código, funciones completas o incluso aplicaciones enteras sin la intervención directa de los programadores. Para esta tarea las herramientas de IA utilizan modelos de aprendizaje automático entrenados previamente con grandes cantidades de datos de código fuente, lo que les permite entender patrones comunes, estructuras y convenciones de codificación. A través de esta comprensión, las herramientas pueden generar código nuevo o completar código existente de manera inteligente, adaptándose a las necesidades específicas (Tuomi, 2018).

Una de las técnicas comunes utilizadas en la automatización de generación de código es el modelado de lenguaje natural (PNL), que permite a las herramientas interpretar y procesar el lenguaje humano para transformarlo en instrucciones de programación. Además, también pueden utilizar técnicas de generación basada en reglas, donde se aplican reglas predefinidas y patrones de codificación para producir código coherente y de alta calidad, entre ellas, prácticas de codificación recomendadas, convenciones de nomenclatura y estructuras de diseño comunes.

2.4.2 Ingeniería

El modelo de ingeniería generalmente utilizada en tareas generativas, es el PNL que se emplea para convertir una descripción en lenguaje natural de entrada en un programa ejecutable, cuyo proceso comienza con el codificador, que toma la descripción del lenguaje natural como entrada y la convierte en una representación vectorial de longitud fija. Esta representación vectorial, también llamada vector de contexto, captura la información esencial de la entrada y la codifica en una forma que la red neuronal pueda procesar.

Dicho codificador utiliza técnicas de aprendizaje profundo, entre ellas las redes neuronales recurrentes (RNN: recurrent neural network) o también están las redes neuronales

convolucionales (CNN: convolutional neural network), para aprender y representar de manera efectiva la información contenida en la descripción del lenguaje natural, y una vez que la entrada ha sido codificada en un vector de contexto por el codificador, este vector se pasa al decodificador (Tuomi, 2018).

El decodificador es responsable de tomar el vector de contexto y generar una salida de programa ejecutable que cumpla con la descripción proporcionada en el lenguaje natural, mediante el uso de técnicas de decodificación por atención o la decodificación basada en reglas, para convertir la representación vectorial en código fuente legible por la máquina. El decodificador, al igual que el codificador, puede estar compuesto por redes neuronales recurrentes o cualquier otra arquitectura adecuada para la tarea específica de generación de código.

2.4.3 Licencias de las herramientas IA

Las licencias y el uso de las IA en general pueden variar dependiendo del proveedor y del tipo de herramienta, las licencias de software para este tipo de herramientas suelen regirse por modelos comerciales estándar, como licencias de código abierto, licencias propietarias o modelos de suscripción.

No obstante, las licencias de código abierto son comunes en el ámbito de la IA y la generación de código, y permiten a los usuarios acceder al código fuente, modificarlo y redistribuirlo de acuerdo con los términos de la licencia específica, lo que es beneficioso para la comunidad de desarrollo, ya que fomenta la colaboración, la transparencia y la innovación compartida. Varias herramientas populares que a menudo se utilizan en la generación de código por IA, están disponibles bajo licencias de código abierto.

Por otro lado, algunas herramientas de generación de código por IA pueden ofrecer licencias propietarias, que restringen el acceso al código fuente y pueden requerir el pago de una tarifa de licencia para su uso; licencias que suelen proporcionar características adicionales, soporte técnico y garantías de calidad por parte del proveedor, pero pueden limitar la flexibilidad y la personalización en comparación con las licencias de código abierto; se debe revisar detenidamente los términos y condiciones de estas licencias para comprender los derechos y restricciones asociados con el uso de cada herramienta.

De igual manera, algunas herramientas pueden seguir un modelo de suscripción, donde los usuarios tienen que pagar una tarifa recurrente para acceder a la respectiva plataforma y tener acceso a sus características, que puede ser conveniente para aquellos que desean acceso

continuo a actualizaciones, nuevas funcionalidades y soporte técnico, pero puede requerir un compromiso financiero a largo plazo, lo que es adecuado para empresas, pero no tanto para desarrolladores principiantes o estudiantes.

2.4.4 Regulaciones legales sobre el uso de la IA

En el caso de Ecuador, hasta el momento no existe una ley específica que regule el desarrollo, distribución y uso de herramientas de IA. Sin embargo, en 2021 la UNESCO adoptó la Recomendación sobre la Ética de la Inteligencia Artificial, que sugiere parámetros de utilización de esta tecnología e insta a los estados miembros a desarrollar sus propias normativas que regulen la IA, siempre que sean respetados los derechos humanos y ambientales (UNESCO, 2021).

En ausencia de una legislación específica, a nivel nacional se rige a lo que prohíben o permiten la constitución y otras normas, y en principios que sean éticos permitiendo llegar a un uso responsable de las IAs, como la adopción de enfoques basados en derechos humanos, como la transparencia, la equidad, entre otros factores que intervienen en el proceso de toma de decisiones relacionado con la tecnología.

En este sentido, varios países han adoptado leyes y regulaciones para regular su normativa en aspectos relacionadas con la IA, como la privacidad de datos, la responsabilidad civil, la discriminación algorítmica y la seguridad cibernética; introduciendo en diferentes códigos de ley estos elementos.

Por lo tanto, es importante tomar en cuenta que, a pesar de no existir una ley específica sobre IA, leyes y regulaciones existentes en Ecuador, como las leyes de protección de datos personales, la legislación de responsabilidad civil y las leyes laborales, pueden aplicarse a la IA en contextos específicos.

2.4.5 Toma de decisiones multicriterio para software (Decision making)

La toma de decisiones multicriterio para software es una herramienta que se utiliza para abordar situaciones en las cuales se deben evaluar múltiples factores al seleccionar una opción entre varias alternativas, en este caso para elegir un software, en la cual, metodológicamente se tiene que considerar una serie de criterios relevantes, como pueden ser sus funciones, costo, facilidad de uso, soporte técnico, seguridad, entre otros (Bravo et al., 2021).

Este proceso de toma de decisiones pretende dar una estructura sistemática al proceso de elección con la finalidad de comparar diferentes opciones de software, mediante técnicas como análisis de valoración ponderada, análisis jerárquico de procesos, análisis de compromiso o también llamado de Pareto, TOPSIS y análisis de preferencia discreta, entre los más conocidos, para asignar pesos a los criterios y comparar las alternativas de acuerdo a estos criterios (Bravo et al., 2021).

Como una de las herramientas más efectivas para el *Decision making*, la técnica para el “orden de preferencia por similitud con la solución ideal” (TOPSIS, por sus siglas en inglés) es un método que surgió en la década de 1980 y se ha utilizado ampliamente para evaluar alternativas en función de múltiples criterios, identificando la mejor alternativa al seleccionar la que tenga “la distancia más corta desde la solución ideal y la mayor distancia desde la solución ideal negativa” (Bravo et al., 2021).

Para entender mejor cómo funciona TOPSIS, considere que se trata de tomar una decisión sobre qué teléfono móvil comprar entre cinco opciones diferentes, para ello, se evalúa cada teléfono en función de criterios como la cantidad de RAM, la memoria interna, las dimensiones de la pantalla, la batería y su duración y el precio; este medio proporciona un marco estructurado para asignar rangos a cada opción, teniendo en cuenta los pesos y el impacto de los factores dados.

A parte de las ponderaciones, TOPSIS permite calcular el impacto de cada factor, es decir, si tiene un impacto positivo o negativo en la decisión, es decir, si se quiere maximizar la duración de la batería, pero minimizar el precio del teléfono, se asignaría un peso positivo al criterio de duración de la batería y un peso negativo al precio, lo que ayuda a reflejar adecuadamente las preferencias de usuario y los objetivos al evaluar las alternativas.

En la elección de software de IA para la generación de código, TOPSIS ayuda a examinar factores como la eficacia, la precisión del modelo, la flexibilidad y la facilidad de integración, ya que aplicando esta técnica se puede identificar una solución que sea la que más se ajuste a las necesidades específicas de un proyecto, considerando de manera adecuada todos los criterios relevantes, lo que conduce a una elección informada y óptima del software.

2.5 Desarrollo dirigido por funciones

Feature-driven development (FDD) o desarrollo dirigido por funciones (Riady et al., 2022) es un método de desarrollo de software ágil donde se priorizan dos etapas: la inicial que se trata de determinar las características a implementar dentro de un software, la siguiente y final consiste

en la respuesta a los cambios de características por parte del cliente, donde el desarrollador debe actuar de manera inmediata ante la petición de los clientes para ir cambiando el software resultante con base a los requisitos inicialmente establecidos. Los procesos se realizan de forma iterativa incremental, al final de estos pasos se entrega un documento final con la versión del software elaborado. Los pasos que sigue el método son los siguientes:

- Desarrollar el modelo general.
- Construir la lista de características.
- Planificar por característica.
- Diseñar por característica.
- Construir por característica.

2.6 Antecedentes Investigativos

La investigación realizada por (Huang et al., 2024) titulada "Generative Software Engineering" estudia los avances en el campo de la ingeniería de software impulsados por el desarrollo rápido de técnicas de aprendizaje profundo, la mejora en la capacidad computacional y la disponibilidad de conjuntos de datos de entrenamiento, identificando la importancia de los modelos pre-entrenados y los Modelos de Lenguaje de Gran Tamaño (LLM: large language model), como BERT, Transformer y ChatGPT, en la mejora de las capacidades lingüísticas y su aplicación en la ingeniería de software.

La investigación subraya la importancia de las tareas generativas en la ingeniería de software, donde los modelos pre-entrenados y LLMs muestran habilidades en representación del lenguaje y comprensión contextual; capacidades que permiten a los modelos adaptarse a tareas generativas mediante ajuste fino, transferencia de conocimiento y técnicas de ingeniería de prompts.

El estudio proporciona una revisión exhaustiva de las tareas generativas en ingeniería de software, categorizándolas según "el ciclo de vida del desarrollo de software". Además, se resumen los modelos pre-entrenados y LLMs avanzados utilizados, los conjuntos de datos y las métricas de evaluación. Se identifican fortalezas, debilidades y brechas en los enfoques existentes, y se proponen posibles direcciones de investigación.

En los resultados se pueden identificar algunas limitaciones en las revisiones literarias existentes, como la falta de análisis exhaustivo del proceso de desarrollo de modelos pre-entrenados a

LLMs, la falta de distinción en los tipos de tareas o actividades de ingeniería de software y la ausencia de análisis sistemático.

De igual manera en La investigación realizada por (Feng et al., 2020), titulada "CodeBERT: A Pre-Trained Model for Programming and Natural Languages" presenta CodeBERT como un modelo pre-entrenado bimodal para lenguajes de programación y lenguaje natural que aprende representaciones de propósito general que soportan aplicaciones NL-PL como búsqueda de código en lenguaje natural, generación de documentación de código, etc.

Este modelo se desarrolla con una arquitectura neural basada en Transformer y se entrena con una función objetivo híbrida que incorpora la tarea de detección de tokens reemplazados, que consiste en detectar alternativas viables muestreadas de generadores, facilitando utilizar tanto datos bimodales de pares NL-PL como datos unimodales, donde los primeros proporcionan tokens de entrada para el entrenamiento del modelo mientras que los últimos ayudan a aprender mejores generadores.

El modelo se evalúa en dos aplicaciones NL-PL mediante la sintonización fina de los parámetros del modelo, y los resultados muestran que la herramienta logra un rendimiento óptimo tanto en la búsqueda de código en lenguaje natural como en la generación de documentación de código. Además, para investigar qué tipo de conocimiento se aprende en CodeBERT, se construye un conjunto de datos para la sonda NL-PL y se evalúa en un entorno de configuración de cero disparos donde los parámetros de los modelos pre-entrenados están fijos, y según los resultados supera a modelos pre-entrenados anteriores en la sonda NL-PL.

Esta investigación, presenta CodeBERT como el primer modelo pre-entrenado bimodal grande para lenguaje natural y lenguaje de programación, y se identifican varias direcciones potenciales para investigaciones futuras, como mejorar los generadores con evidencia bimodal o arquitecturas neuronales más complicadas, incorporar árboles sintácticos abstractos (AST, por su siglas en inglés) en el paso de pre-entrenamiento y aplicar CodeBERT a más tareas relacionadas con NL-PL y a una gama más amplia de lenguajes de programación.

Por su parte, dentro de la investigación presentada por (Chen et al., 2021), titulada "Evaluating Large Language Models Trained on Code", que trata sobre introducir Codex, un modelo de lenguaje GPT fine-tuned en código público disponible en GitHub, y estudia sus capacidades para escribir código en Python, considerando que Codex, en su versión de producción, alimenta GitHub Copilot.

En el conjunto de evaluación Humaneval, diseñado para medir la corrección funcional en la síntesis de programas a partir de docstrings, Codex resuelve el 28.8% de los problemas, mientras que GPT-3 resuelve el 0% y GPT-J resuelve el 11.4%. Además, los resultados indican que muestrear repetidamente del modelo es una estrategia efectiva para producir soluciones de trabajo para problemas difíciles; utilizando este método, se resuelven el 70.2% de los problemas con 100 muestras por problema.

En ese sentido, el trabajo demuestra que es posible entrenar modelos de lenguaje grandes para producir código funcionalmente correcto a partir de docstrings de lenguaje natural y la importancia de entrenar los modelos en conjuntos de datos más similares al conjunto de evaluación y de generar múltiples muestras para mejorar el rendimiento del modelo; generando además un debate sobre las implicaciones de desplegar tecnologías de generación de código poderosas, incluyendo aspectos de seguridad, economía y limitaciones del modelo que indican espacio para mejoras futuras.

En la investigación llevada a cabo por (Sun et al., 2022), titulada "Investigating Explainability of Generative AI for Code through Scenario-based Design", la cual tiene como objeto de estudio el significado de que un modelo de IA generativa sea explicativo; tomando en cuenta que, aunque la disciplina emergente de la IA explicativa (XAI, por sus siglas en inglés) ha avanzado en ayudar a las personas a comprender modelos discriminatorios, se ha prestado menos atención a los modelos generativos que producen artefactos como salida en lugar de decisiones. A través del diseño basado en escenarios y enfoques de diseño XAI impulsados por preguntas, se exploran las necesidades de conocer de los usuarios para la IA generativa dentro de tres casos de uso: pasar de lenguaje natural a código fuente, traducir y autocompletar código; realizando 9 muestras con 43 ingenieros de software, donde se utilizan ejemplos reales de modelos de IA generativa de vanguardia para obtener las necesidades de explicabilidad de los usuarios, y proponiendo 4 tipos de características XAI para la IA generativa para código, donde también se recopilan ideas adicionales de diseño de los participantes.

En la discusión, se destaca la necesidad de que los usuarios formen un modelo mental claro sobre el alcance, las características y las limitaciones de los espacios de entrada y salida, así como una vista global de cómo el modelo genera salidas a partir de las entradas. Se señala que las necesidades de explicabilidad de los participantes se centraron abrumadoramente en las explicaciones de entrada, salida y globales, lo que sugiere una discrepancia entre lo que necesitan los usuarios de las aplicaciones de IA generativa y el enfoque actual de la comunidad

técnica en explicaciones para el aprendizaje de representaciones y visualización de representaciones.

Asimismo, se discuten las implicaciones del diseño para la IA generativa para código, destacando la recepción positiva de explicaciones en lenguaje natural y las interacciones con la IA generativa, observando que las necesidades de explicabilidad y las necesidades de los usuarios en general pueden diferir según el nivel de habilidades de programación; resaltando la utilidad de la IA generativa para código, especialmente en cuanto a si los usuarios deben entender cómo funciona la IA y ajustar sus entradas para obtener resultados más óptimos.

Los resultados del estudio destacan la importancia de adoptar enfoques humanizados, participativos y basados en preguntas para el diseño de IA explicativa, especialmente en dominios novedosos como la ingeniería de software; recomendado tomar en cuenta sus logros para futuros trabajos en IA y HCI, para una mejor colaboración humano-IA en la ingeniería de software, y fomentar de esta manera enfoques más centrados en el humano para impulsar el desarrollo técnico de la IA.

Otro estudio, que se considera, es la investigación realizada por (Cámara et al., 2023), cuyo título "On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML", trata sobre estudiar las capacidades y limitación actuales de los modelos de IA para generación de código, específicamente enfocados en tareas de modelado de software; repasando los modelos de lenguaje grande (LLMs), utilizados por Copilot y ChatGPT, y como se espera que estos revolucionen el desarrollo de software, realizado un análisis del estado actual en relación con el modelado de software.

Para su estudio, los autores interactuaron individualmente con ChatGPT para comprender sus capacidades de modelado y características principales; sin un enfoque netamente sistemático, sino únicamente exploraron diversas ideas y realizaron interacciones con ChatGPT para comprender sus capacidades, redactando prompts, pidiendo a ChatGPT que creará modelos de diferentes tamaños y que creará modelos.

Los hallazgos de la investigación muestran que, en contraste con la generación de código, el rendimiento de la versión actual de ChatGPT para el modelado de software es limitado, con varias deficiencias sintácticas y semánticas, falta de consistencia en las respuestas y problemas de escalabilidad. Sin embargo, en lugar de descartar estos modelos, los autores sugieren trabajar para mejorar las habilidades de modelado de ChatGPT y otros LLMs, con el objetivo de construir un futuro en el que estos asistentes jueguen un papel destacado en el modelado.

Las conclusiones sugieren que los asistentes colaboran en gran medida con el desarrollo de modelos, la generación de pruebas basadas en modelos y la educación en modelado. Además, se plantea la aparición de nuevos roles en ingeniería de software, como ingenieros de solicitudes, y la necesidad de optar por métodos de enseñanza de modelado para incorporar los LLMs.

Por su parte, el estudio "¿Perfection Not Required? Human-AI Partnerships in Code Translation" realizado por (Weisz et al., 2021) realiza una investigación sobre la traducción de código mediante técnicas de inteligencia artificial generativa, específicamente el uso de modelos de traducción neuronal no supervisados (NMT), teniendo como objetivo general investigar hasta qué punto los ingenieros de software tolerarían imperfecciones en la traducción de código generada por estos modelos, así como explorar formas de detectar y corregir dichos errores.

En la investigación, utilizando un enfoque de escenario de diseño, se llevaron a cabo entrevistas con 11 ingenieros de software para comprender sus reacciones al uso de un modelo NMT en el contexto de la modernización de aplicaciones, centrándose en la tarea de traducir código de un lenguaje de programación a otro. Este enfoque permitió discutir la utilidad y deseabilidad de trabajar con un sistema de IA imperfecto, así como establecer la aceptación de las salidas del sistema y explorar oportunidades futuras para la IA generativa en la modernización de aplicaciones.

Los resultados obtenidos revelan cómo características de la interfaz de usuario, como resaltado de confianza (que resalta las líneas de código, indicando su fiabilidad, si se tiene que cambiar o si pueden ser mejoradas) y traducciones alternativas, ayudan a los ingenieros de software a trabajar con modelos NMT generativos y comprenderlos mejor, pues, los participantes mostraron disposición para aceptar la ayuda de un asistente de IA imperfecto y consideraron que una herramienta que pudiera realizar la traducción de código sería útil para agilizar su trabajo.

Se identificó la necesidad de desarrollar IA de generación de código explicativa para ayudar a comprender los modelos generativos y generar explicaciones específicas sobre por qué se generó un artefacto dado. Además, se destacó la importancia de explorar diferentes patrones de interacción humano IA y capacidades más allá de la traducción de código para mejorar la colaboración efectiva entre humanos y modelos generativos.

Finalmente, el estudio titulado "A Categorical Archive of ChatGPT Failures" realizado por (Borji, 2023) se enfoca en identificar y analizar las limitaciones y fallas del modelo de lenguaje ChatGPT, considerando que, a pesar de las capacidades de esta herramienta en ciertas tareas, se debe

ver la necesidad de mejoras adicionales en áreas como solución de problemas matemáticos, el razonamiento, la reducción de sesgos, entre otros.

El estudio identifica once categorías de fallas, que incluyen razonamiento, errores factuales, matemáticas, generación de código, entre otros; discutiendo los riesgos, limitaciones e implicaciones sociales de ChatGPT, con el objetivo de ayudar a investigadores y desarrolladores a mejorar futuros modelos de lenguaje y chatbots.

Además, la investigación plantea como hipótesis sobre si ChatGPT y otros modelos de lenguaje comprenden verdaderamente lo que generan o simplemente memorizan patrones, por lo que se debe evaluar su capacidad para razonar con sentido común y generar soluciones creativas para problemas novedosos, así como la importancia de mejorar la capacidad del modelo para indicar su nivel de confianza en las respuestas.

De igual forma, se aborda la preocupación ética y social sobre el uso de modelos como ChatGPT, incluyendo la presencia de sesgos en los datos de entrenamiento, la posibilidad de propagación de desinformación y la necesidad de hacer que los sistemas de IA sean explicables para comprender el razonamiento detrás de sus decisiones.

Tabla 2. Comparación de antecedentes investigativos

Tabla comparativa de Antecedentes investigativos					
Autores	Tema	Enfoque de investigación	Tipo de IA	Método de análisis	Entrada de prompts
(Huang et al., 2024)	Generative Software Engineering	Eficiencia en ingeniería de software y automatización de procesos	De licencia gratuita y basados en proyectos de investigación académica	Revisión de literatura y análisis de datos bibliográficos	No se realizaron pruebas
(Feng et al., 2020)	CodeBERT: A Pre-Trained Model for Programming and Natural Languages	Precisión y eficacia en el procesamiento de lenguajes de programación y naturales	De código abierto, Proyecto de los autores	Método práctico basado en la arquitectura bidireccional multicapa de Transformer	Procesamiento de texto y código

(Chen et al., 2021)	Evaluating Large Language Models Trained on Code	Eficiencia y efectividad en el entrenamiento de modelos de lenguaje grandes	Licencia de uso específica de OpenAI	Pruebas unitarias y framework de evaluación para medir el rendimiento del modelo en la generación de funciones a partir de docstrings	Procesamiento de texto y modelos de cadenas de documentación (docstrings)
(Sun et al., 2022)	Investigating Explainability of Generative AI for Code through Scenario-based Design	Explicabilidad de la IA generativa en la generación de código a través del diseño basado en escenarios	Licencia privativa (TransCoder y Copilot)	Enfoque experimental basado en escenarios	Interacción de participantes por texto, modelos, voz.
(Cámara et al., 2023)	On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML	Capacidad de modelado de ChatGPT en tareas de modelado UML	De licencia gratuita en GPT 3.5	Multicriterio en aspectos como corrección, completitud, cobertura de conceptos de modelado, expresividad, capacidades de traducción entre lenguajes y	Procesamiento de texto y código

				dominio del problema	
(Weisz et al., 2021)	Perfection Not Required? Human-AI Partnerships in Code Translation"	Análisis de cómo interactuarían con un modelo de traducción automática neuronal (NMT) en el contexto de la modernización de aplicaciones.	Uso de modelo de traducción automática neuronal de código abierto	Análisis temático para identificar patrones en los datos recopilados y agruparlos en temas relevantes	Escenarios de diseño
(Borji, 2023)	A Categorical Archive of ChatGPT Failures"	Se centra en analizar las deficiencias de ChatGPT en diferentes áreas de razonamiento y comprensión, categorizando sus fallos en once áreas específicas	Licencia gratuita en ChatGPT 3.5	Análisis mediante de observación	Procesamiento de texto

Capítulo 3: Diseño del caso de estudio para la evolución de rendimiento

Como se ha comentado en este trabajo, la proliferación de la IA se ha posicionado como una herramienta poderosa para automatizar y optimizar diversos procesos, incluyendo la generación de código. Mediante este capítulo se procura evaluar el rendimiento de diferentes herramientas de IA mediante un caso de estudio práctico, denominado "Software para el registro, gestión y tenencia responsable de mascotas en la ciudad de Cuenca". Este caso de estudio proporciona un escenario realista para analizar la precisión, eficiencia y calidad del código generado por las IAs para el desarrollo de una página web.

Principalmente, con este capítulo, se busca establecer una clasificación de las herramientas de IA evaluadas, basándose en su desempeño en términos de precisión en la interpretación de los requisitos, eficiencia en la generación del código y la calidad del mismo. Para llevar a cabo esta evaluación, se ha seguido la metodología propuesta en el artículo "*Guidelines for Conducting and Reporting Case Study Research in Software Engineering*" de los autores (Runeson & Höst, 2009). Este artículo ofrece un marco detallado y riguroso para la realización de estudios de caso en la ingeniería de software, asegurando que los resultados obtenidos sean tanto válidos como reproducibles.

La evaluación de las herramientas de IA no solo proporcionará una visión comparativa de su rendimiento, sino que también destacará las ventajas y limitaciones de cada una en un contexto de desarrollo, y con lo cual, identificar las herramientas más adecuadas para proyectos similares. A continuación, se describirá en detalle la metodología utilizada para el estudio de caso, se presentarán los resultados obtenidos y se discutirán las implicaciones de estos resultados en el contexto del desarrollo de software asistido por IA.

3.1 Características del software a desarrollar del caso de estudio.

El caso de estudio por el cual se evaluará el rendimiento de la herramientas IA con respecto a la generación de código, comprende un software para el registro, gestión y tenencia responsable de mascotas en la ciudad de Cuenca. Estos requisitos se implementarán mediante la creación de una página web, que contempla las siguientes características:

- **Formulario de Registro de Mascotas.** Campos de Entrada donde se permita ingresar datos específicos sobre la mascota, tales como:
 - Especie: Selección del tipo de mascota (perro, gato, etc.).
 - Género: Selección del género de la mascota (macho, hembra).
 - Raza: Campo para especificar la raza de la mascota.

- Fecha de Nacimiento: Selección de la fecha de nacimiento.
- Color: Campo para describir el color de la mascota.
- Peso: Campo para ingresar el peso de la mascota.
- Foto: Carga de una foto de la mascota.
- Código de Chip: Campo para el número de microchip de la mascota.
- **Información del Propietario.** Campos de entrada que incluya secciones para capturar los datos del propietario de la mascota, como:
 - Nombre: Nombre completo del propietario.
 - Edad: Edad del propietario.
 - Dirección: Dirección de residencia del propietario.
 - Número de Celular: Teléfono móvil para contacto.
 - Correo Electrónico: Dirección de email del propietario.
 - Número de Persona de Referencia: Un contacto de referencia adicional.
- **Información Médica.** Campos de Entrada con área dedicada para capturar detalles médicos de la mascota, incluyendo:
 - Médico Tratante: Nombre del veterinario responsable.
 - Código de Ficha: Número de identificación de la ficha médica.
 - Vacunas: Información sobre las vacunas administradas.
 - Desparasitaciones: Detalles sobre desparasitaciones realizadas.
 - Tratamientos: Registro de tratamientos médicos aplicados.
- **Funcionalidad de Búsqueda.** Área donde se permita buscar a la mascota.
 - Búsqueda por Nombre, Número de Microchip o Nombre del Propietario: Permite buscar una mascota específica utilizando su nombre, el número de microchip o el nombre del propietario.
 - Resultados: Muestra una lista de mascotas que coinciden con los criterios de búsqueda, presentando detalles básicos como nombre, especie, y código de microchip.
- **Generación de Informes Poblacionales.** Parte estadísticas donde poder generar informes que muestran estadísticas sobre la población de mascotas registradas, tales como:
 - Distribución por Especie: Número de mascotas por especie (perros, gatos, etc.).
 - Distribución por Género: Número de mascotas masculinas y femeninas.
 - Distribución por Raza: Conteo de mascotas según su raza.

- Informe Visual: Presenta los resultados de las estadísticas en un formato visual en la página, facilitando la comprensión de la información.
- **Diseño de la Página.**
 - Interfaz minimalista: Un diseño de interfaz de usuario limpia y moderna, utilizando HTML y CSS para estilos básicos y un diseño responsivo.
 - Facilidad de Navegación: La página debe estar estructurada para ser intuitiva, facilitando a los usuarios la entrada y búsqueda de información.
- **Optimización y Manejo de Datos.**
 - Código JavaScript Eficiente: Uso de JavaScript para manejar la lógica de registro, búsqueda y generación de informes, asegurando un rendimiento óptimo.
 - Validación Básica: Implementación de validación de datos en los formularios para asegurar la integridad de la información ingresada.

Las características a contemplar y desarrollar son 7, para cubrir los requerimientos del software del caso de estudio mencionado y por el cual serán evaluadas las herramientas IAs determinadas. Se parte de la idea de la generación de código holística, que significa que las herramientas IAs seleccionadas, deberán proporcionar el mejor código posible considerando las 7 características planteadas, logrando así obtener en la medida de lo posible, un código íntegro.

3.2 Diseño de investigación

En la presente investigación el diseño es experimental, mismo que se utiliza para examinar y analizar las relaciones de causa y efecto entre variables, el objetivo es estudiar las características de las herramientas de IA por comandos de voz para la generación de código, que se llevará a cabo mediante la realización de pruebas en un caso de estudio que involucra el desarrollo de un software que permita el registro, gestión y tenencia responsable de mascotas en la ciudad de Cuenca, específicamente a cargo de la Comisión de Gestión Ambiental del GAD Municipal.

Los prompts que se van a utilizar para introducir en cada una de las herramientas seleccionadas han sido estructurados con los patrones de inicialización y modelado revisados en las investigaciones de (White et al., 2023) y (De Vito et al., 2023), quedando como resultado los descritos a continuación:

- a) Para generar el código de inicialización: *"Actúa como un ingeniero de software. Escribe el código para inicializar el sistema de identificación y registro de mascotas, agregando la configuración inicial, como la conexión a la base de datos, manejo de inicio de sesión,*

validaciones, manejo de errores y la carga de bibliotecas necesarias, para una aplicación de registro y consulta de mascotas, que sea óptimo, con un diseño simple y moderno”.

- b) Para generar el código detallado: *“De ahora en adelante, genera código para el mismo software en HTML, CSS y JavaScript con las siguientes especificaciones: se desea contar con el ingreso de datos de mascotas tales como especie, genero, raza, fecha de nacimiento, color, peso, foto, código del chip; que se incluyan también datos del propietario como nombre, edad, dirección, número de celular, correo electrónico, número de persona de referencia; se debe incluir también la información médica como Médico tratante, código de ficha, vacunas, desparasitaciones, tratamientos; se desea también la búsqueda de mascotas en el sistema; que permita la búsqueda de mascotas por su nombre, número de microchip o nombre del propietario; que el código generado sea óptimo, con un diseño simple y moderno”.*
- c) Para obtener el código de generación de informes poblacionales: *“Genera el código para crear informes poblacionales basados en los datos de las mascotas registradas, estadísticas sobre la población de mascotas por especie, ubicación geográfica, por género y raza. Que sea optimo, con un diseño simple y moderno”.*

El primer prompt se centra en generar el código de inicialización con una orden bastante general, únicamente solicitando un diseño óptimo, simple y moderno, pero deja cierto margen para la interpretación en términos de qué apartados se cargarán y cómo se configurará la conexión a la base de datos; mientras que en el segundo prompt, la orden es más específica, incluyendo campos como datos de las mascotas, propietario e información médica, y se requiere funcionalidad de búsqueda; esperando que el código generado se pueda ampliar con relación al anterior.

El último prompt se enfoca en generar el código como una ampliación o funcionalidad adicional a lo que ya se obtiene, ya sea a partir del primero o segundo, que tendrá como finalidad en el software la creación de informes poblacionales basados en los datos de mascotas registradas, por la necesidad de estadísticas sobre la población de mascotas por diferentes categorías; para verificar como actúa cada herramienta IA ampliando un código de forma secuencial.

El diseño experimental implica la manipulación controlada de la variable independiente, es decir las herramientas IA, para observar el efecto que tienen sobre la variable dependiente, o sea el software mencionado. En este estudio, se diseñarán pruebas específicas que permitan evaluar cómo cada herramienta afecta la generación de código en el contexto de dicho software, y como influyen en las características, la eficiencia, la precisión y la facilidad de uso.

3.3 Recolección de datos

En el proceso de selección de herramientas IA, se emplearán dos métodos distintos para la recolección de datos; en primer lugar, se llevarán a cabo entrevistas con profesionales en el desarrollo de software a través de la herramienta Google Forms disponible en el enlace: <https://forms.gle/BwN4jY7GHLmYyrNC8>, las mismas que permitirán obtener información directa y detallada sobre las experiencias, opiniones y recomendaciones de expertos en el campo; según las herramientas que han utilizado previamente, su experiencia con ellas, las características que consideran más importantes y cualquier consejo o sugerencia que puedan ofrecer para la selección adecuada. En el anexo 2 se visualiza la guía de entrevista desarrollada, que permitirá seleccionar las opciones más adecuadas de IA para generación de código.

En segundo lugar, se realiza una revisión exhaustiva de la bibliografía relacionada con cada una de las herramientas; como artículos científicos, libros, informes técnicos y otros recursos relevantes que proporcionen información detallada sobre las características, capacidades, ventajas y limitaciones de cada una de las herramientas que recomienden procedentes de la entrevista realizada.

Una vez obtenidos los datos se realizará una selección de 4 herramientas con las características más óptimas mediante la selección multicriterio, y se realizará un análisis mediante las pruebas indicadas en el diseño de investigación experimental, pruebas que permitirán evaluar y comparar directamente el rendimiento, la eficacia y la idoneidad de cada herramienta; es decir de acuerdo a cada uno de los factores de la herramienta MCDM, TOPSIS.

3.4 Población y muestra

Dado que en la actualidad existen numerosas herramientas IA para generación de código en el mercado, la población sería el conjunto completo de estas herramientas, sin restricciones; dichas herramientas pueden variar en términos de funcionalidades, características, licencias, facilidad de uso y otros aspectos relevantes para su evaluación.

La muestra, por otro lado, es un subconjunto representativo de la población que se selecciona para participar en el estudio; este caso, se propone realizar un levantamiento de información sobre las herramientas de IA para la generación de código por comando de voz, se preseleccionarán 10 herramientas para su análisis inicial.

Posteriormente, se llevará a cabo un proceso de selección y evaluación más detallado de estas 10 herramientas preseleccionadas. Según sus funciones, características y otros criterios relevantes, se elegirán las 4 mejores herramientas para realizar el análisis experimental indicado.

3.5 Criterios para el análisis de herramientas

Los criterios de análisis para las herramientas seleccionadas serán los siguientes;

- a) **Interfaz** de la herramienta, que juega un papel fundamental en la experiencia del usuario, pues si la interfaz es intuitiva y fácil de usar facilita la interacción y el uso de la herramienta, lo que puede mejorar la productividad y la eficiencia del desarrollo de software.
- b) Otro aspecto importante a considerar son las **funciones** que dispone la plataforma; una herramienta robusta debería ofrecer una amplia variedad de funcionalidades integradas que faciliten el proceso de generación de código y optimicen el flujo de trabajo del desarrollador.
- c) El tipo de **licencias** y los **costos** asociados son factores clave a evaluar, ya que pueden influir en la viabilidad económica de la herramienta para la organización, si la herramienta ofrece licencias gratuitas o de código abierto, así como las opciones de licenciamiento y los costos asociados para el uso comercial.
- d) El **soporte que brinda el proveedor** garantiza la disponibilidad de asistencia técnica y recursos de ayuda en caso de problemas o preguntas, si el proveedor ofrece un buen soporte puede ayudar a resolver problemas de manera rápida y eficiente, minimizando el tiempo de inactividad y optimizando la productividad del equipo de desarrollo.
- e) La **seguridad** que brinda la herramienta es otro aspecto crítico a tener en cuenta, especialmente en entornos donde se manejan datos sensibles o confidenciales.
- f) El nivel de **precisión en la interpretación de comandos de voz** es esencial para garantizar la efectividad y utilidad de la herramienta, ya que esta va a reducir significativamente la necesidad de correcciones manuales y mejorar la eficiencia del proceso de desarrollo.
- g) La **eficiencia y velocidad** con la que la herramienta puede generar código se debe considerar, ya que se puede acelerar el desarrollo de software y mejorar la productividad del equipo.
- h) La **adaptabilidad** de cada herramienta a múltiples lenguajes de programación, frameworks y paradigmas de desarrollo es crucial para su versatilidad y utilidad en diferentes entornos y proyectos. Una herramienta que ofrece compatibilidad con una amplia gama de tecnologías puede adaptarse mejor a las necesidades cambiantes del desarrollo de software.

- i) La **compatibilidad** con sistemas operativos y plataformas es un factor importante a considerar, especialmente en entornos donde se utilizan diferentes sistemas y tecnologías. Una herramienta que es compatible con una variedad de sistemas operativos y plataformas puede ofrecer una mayor flexibilidad.

Estos criterios propuestos para las herramientas seleccionadas son diversos y abarcan aspectos clave para evaluar la efectividad de un producto o herramienta en el contexto de un estudio; sin embargo, para esta investigación, se seleccionarán únicamente aquellos que sean más adecuados según los factores que se pretende analizar y los resultados obtenidos de las entrevistas con profesionales en el uso de IA para generación de código y desarrollo de software, pues, si bien todos los criterios son importantes, no todos serán relevantes para los objetivos planteados; priorizando los criterios que se alineen para evaluar la eficacia y la idoneidad de las herramientas de IA para generación de código y desarrollo de software por reconocimiento de voz.

3.6 Análisis cualitativo de resultados

El análisis de los resultados se presentará de forma secuencial, empleando una guía que contiene la descripción de los criterios de análisis utilizados y presentando los hallazgos para cada herramienta que se ha analizado, con lo que proporciona una visión general de los criterios de análisis seleccionados, destacando la importancia de cada uno y su relevancia para la evaluación de las herramientas, con una explicación de cada criterio y el puntaje obtenido por cada una de las herramientas IA.

Para las pruebas de funcionalidad del código generado, se utilizará una herramienta llamada Visual Studio Code, que es un editor de código, además de ser una IDE, que significa un entorno de desarrollo integrado, tiene funciones que permiten reproducir el código y hacer pruebas para desarrolladores. Esta herramienta se utiliza para verificar el funcionamiento del código que genere cada uno de las herramientas que se van a analizar en este estudio.

Luego, se presentarán los resultados obtenidos para cada herramienta, comenzando con las características generales y detallando el desempeño de la herramienta en relación con cada uno de los criterios de análisis. Para finalizar la presentación de los resultados se hará una evaluación general de cada herramienta, resaltando sus fortalezas y debilidades y proporcionando recomendaciones para su uso en diferentes contextos y escenarios de desarrollo de software.

3.7 Resultados y discusión

3.7.1 Resultados de la entrevista

Entre los resultados más relevantes realizados a los profesionales en el área de desarrollo de software se obtiene en primer lugar los datos de la población como estadísticas de género, profesión, años de experiencia en el área de desarrollo de software, tal como se muestra a continuación:

Estadísticas demográficas de los participantes

Se pueden observar en las siguientes ilustraciones los datos demográficos de los participantes de la entrevista realizada.

Figura 2. Gráfico de estadísticas de género sobre los participantes de la entrevista.

A efectos de llevar estadísticas de los participantes en esta encuesta, por favor indique su género:

34 respuestas

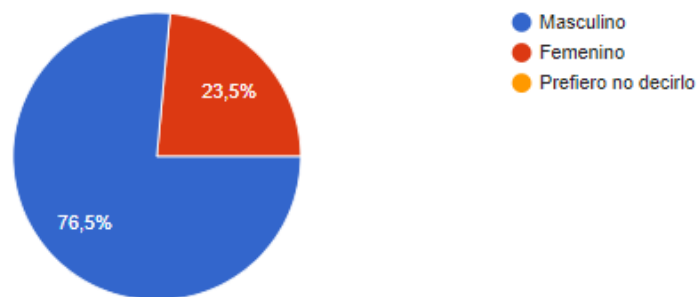
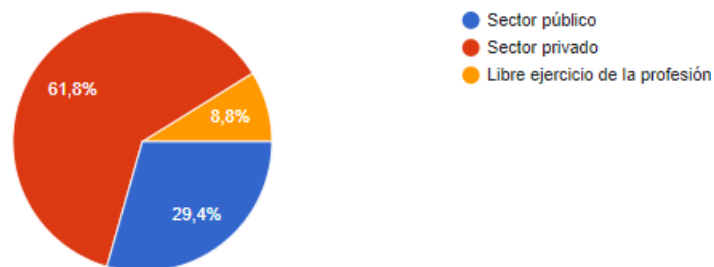


Figura 3. Gráfico sobre sectores donde laboran los entrevistados.

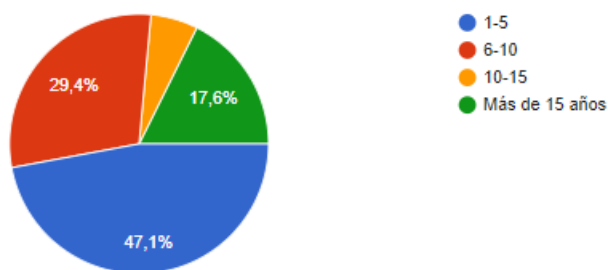
¿Usted trabaja en el sector público o privado, o en el libre ejercicio de la profesión?

34 respuestas

**Figura 4.** Estadísticas del nivel de experiencia en el área de estudio de los participantes de la entrevista.

¿Cuántos años de experiencia tiene en desarrollo de software?

34 respuestas

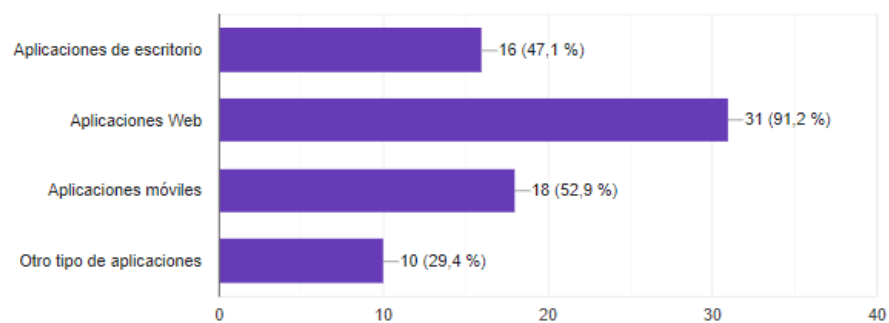


Entre los tipos de software más desarrollados por los participantes, se destacan las aplicaciones web tal como se observa a continuación:

Figura 5. Estadísticas sobre los tipos de software usados por los participantes de la entrevista.

Marque los tipos de software que usted desarrolla:

34 respuestas



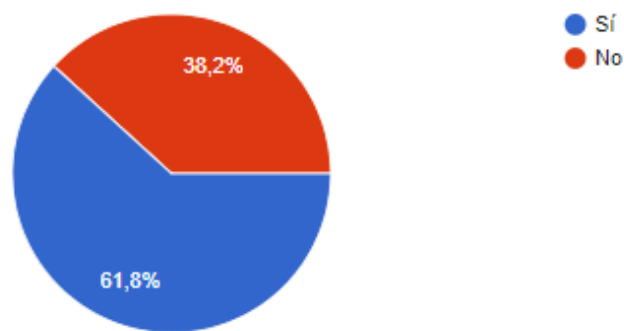
Estadísticas de análisis sobre herramientas IA utilizadas por los participantes

A continuación, se presenta uno de los gráficos, obteniendo como resultado de las entrevistas sobre el uso de herramientas de inteligencia artificial por parte de los participantes, donde la mayoría afirma haber usado herramientas IA para la generación de código fuente para sus propósitos:

Figura 6. Estadísticas sobre el uso de herramientas IA

¿Usted ha utilizado herramientas de IA para generar código fuente?

34 respuestas



Pregunta: ¿Cuál es su experiencia específica en el uso de herramientas de IA para la generación de código? Escriba un párrafo

Los participantes de la entrevista afirman que algunos de los usos dados a las herramientas de IA son para refactorización de código, mejoras de calidad en el mismo, ampliar su conocimiento en los lenguajes y frameworks, como ayuda también para encontrar errores en un código ya implementado anteriormente el mismo que puede ser ingresado como prompt, generación de funciones para complementar código ya desarrollado por la persona, búsqueda de documentación, en bussines intelligence para la generación de cubos de datos a través de funciones de extracción de datos, consultas de ejemplos de código, generación de código de utilería (getters, setters, objetos json), generación de boilerplate, mejoras y optimizaciones de código.

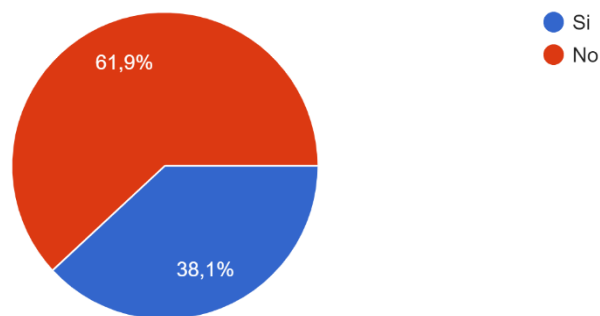
Pregunta: ¿La información que ha ingresado como datos de entrada a la herramienta de generación de código está basada en los requisitos previamente definidos con los usuarios?

Los entrevistados han indicado en la mayoría de los casos que la información que ingresan a la herramienta de IA no tiene nada que ver con la información tratada con los usuarios con anterioridad. Esto se puede observar en la siguiente Figura:

Figura 7. Gráfico resumen de respuestas dadas por los entrevistados sobre requisitos dados por usuarios y colocados como prompt de entrada en la herramienta de IA.

¿La información que ha ingresado como datos de entrada a la herramienta de generación de código está basada en los requisitos previamente definidos con los usuarios?

21 respuestas



Pregunta: ¿Podría mencionar algunas herramientas de IA que haya utilizado o conozca para la generación de código? Escriba un párrafo.

Con base a su conocimiento, los participantes entrevistados aportaron con un compendio de herramientas, y, considerando que en varias entrevistas se mencionan las mismas, se obtiene como respuesta las siguientes herramientas:

- ChatGPT
- Tabnine
- Gemini
- Merlin
- Copilot
- OpenAI Code
- BlackBox
- Llama
- Bart
- You.com

- Claude
- Codeio
- Bing

Pregunta: ¿Cuáles considera que son las principales ventajas de las herramientas de IA para generación de código? (Señale ventajas por cada herramienta que conozca) Escriba un párrafo.

Entre las ventajas colocadas por los participantes de la entrevista se pudieron obtener en resumen las siguientes:

- A pesar de no brindar modelos 100% acertados, sirven de ayuda para ir mejorando código ya generado.
- Optimización de tiempos para la programación tomando el código ya generado para no empezar un desarrollo desde 0. Para soluciones complejas permite perfeccionar el código y adaptarlo a casos reales.
- Permiten realizar búsqueda rápida de documentación.
- Generación de estructuras de código con base a buenas prácticas.
- Completitud de pruebas unitarias de forma rápida bajo formatos personalizados.
- Alternativas de solución a problemas presentados en código desarrollado por una persona, ofreciendo visualización de cada una antes de implementarla.
- Algunas herramientas como BlackBox permiten generar código con imagen, lo cual ayuda a que sea mucho más práctico para el desarrollo enfocado al front-end.
- Copilot tiene como ventaja que extiende el contexto de lo que se va escribiendo, autocompleta los textos, ChatGPT ágil para la generación de código en general. Chat GPT Vision para generación de código a partir de imágenes. Claude para la generación de código mucho más actualizado.
- Con la herramienta BlackBox se tiene como ventajas disminuir el tiempo de desarrollo y la calidad de código que se obtiene.
- Bing tiene la ventaja de generar imágenes con base a un texto proporcionado.

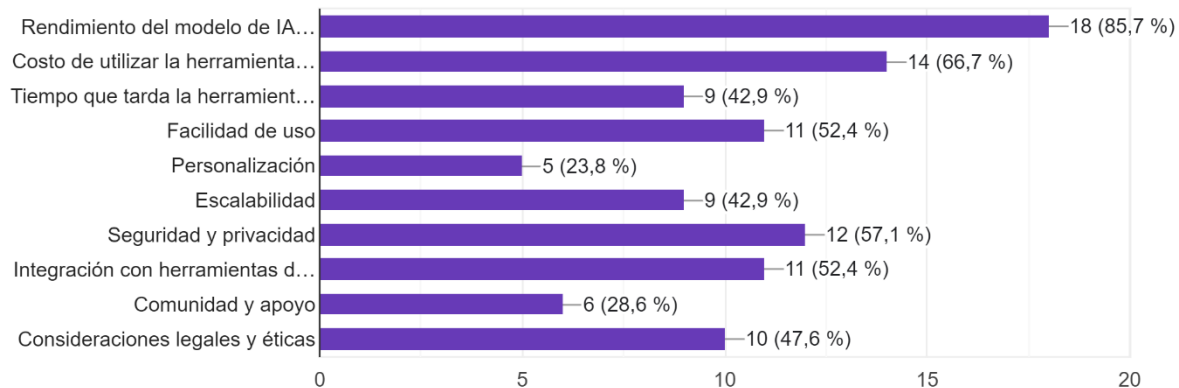
Pregunta: Seleccione cinco criterios que considera usted deben tomarse en cuenta para seleccionar una herramienta de IA:

Se muestra a continuación un gráfico resumen sobre la importancia en los criterios que se debe tomar en cuenta para seleccionar una herramienta para la generación de código con IA, donde

se puede observar que el rendimiento, la facilidad de uso, la seguridad son las características más importantes señaladas por los participantes de la entrevista:

Figura 8. Criterios a tomar en cuenta para el uso de una herramienta con IA.

Seleccione cinco criterios que considera usted deben tomarse en cuenta para seleccionar una herramienta de IA:
21 respuestas

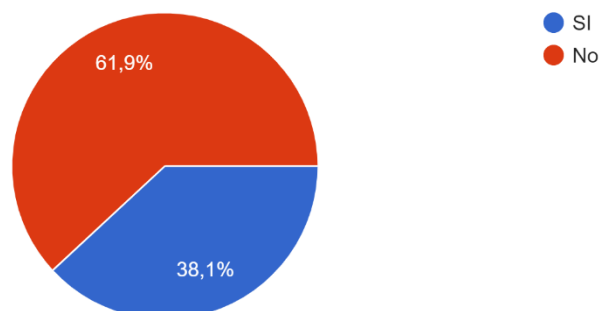


Pregunta: ¿Ha tenido alguna experiencia negativa o algún desafío con alguna de estas herramientas?

En el siguiente gráfico se puede observar que la mayoría de participantes no han tenido experiencias negativas con el uso de las herramientas para generación de código con IA:

Figura 9. Gráfico indicativo sobre experiencias negativas con el uso de herramientas de IA en los entrevistados.

¿Ha tenido alguna experiencia negativa o algún desafío con alguna de estas herramientas?
21 respuestas



Pregunta: Indique cuáles han sido las experiencias negativas que ha tenido en el uso de alguna herramienta de generación de código mediante IA. Escriba un párrafo.

Entre experiencias negativas destacadas por los participantes de la entrevista se detallan las siguientes:

- Repetitividad en las respuestas, falta de precisión y errores comunes.
- Generación de código no funcional para aplicaciones complejas, refactorización necesaria.
- Las herramientas IA no consideran calidad de código o mejores prácticas, optimizaciones, dando respuestas muy sencillas.
- Referencias a código obsoleto, que no funcionan con versiones actualizadas del lenguaje o framework utilizado. Es necesaria la intervención del usuario para corregir.
- Cuando la herramienta no entiende de forma correcta la indicación colocada, lo que requiere escribir de forma repetitiva los prompts.
- Generación de código ineficiente, cambio en el requisito de lo que se está indicando.

Pregunta: Indique cuáles han sido los desafíos con los cuales se han enfrentado durante el uso de alguna herramienta de generación de código mediante IA. Escriba un párrafo.

Los profesionales han indicado como desafíos desde su perspectiva personal los siguientes:

- Ajuste y personalización de la herramienta.
- Construcción de un prompt de calidad.
- Generación de código considerado como correcto.
- Solicitud del código como tal y no alternativas que evitan el uso del código como es el caso de librerías.
- Información desactualizada que suelen brindar las herramientas, respondiendo muchas de las veces con recursos obsoletos.
- Resultados imprecisos o incorrectos.
- Facilitar en concreto lo que se requiere, sin resolver el problema propuesto de forma inicial.

Pregunta: ¿Cuáles son sus recomendaciones para aquellos que estén por utilizar herramientas de IA para la generación de código? Escriba un párrafo.

Lo detallado a continuación corresponde a las respuestas brindadas por los usuarios como recomendación sobre el uso de las herramientas de IA para la generación de código:

- Generar prompts significativos, específicos siendo lo más detallados posibles.
- No utilizarlas para realizar todo el código, sino como un apoyo a procesos largos o difíciles de entender, ya que en ocasiones puede generar código basura.
- Usar para buscar documentación, nunca código completo.
- Utilizar sentencias específicas y el máximo de contexto posible para que la herramienta pueda llegar a generar el código que se necesita.
- Es importante evitar brindar datos personales o privados de la institución, incluyendo el código propio, para así evitar problemas de difusión no intencional de información sensible.
- Son muy útiles para acelerar el proceso de desarrollo y mejora de la productividad.
- Es obligatorio para los desarrolladores.
- Ser claros en las inquietudes planteadas.
- Se debe entender bien el lenguaje de programación para entender el código que la IA genera y poder acoplarlo a código ya escrito permitiendo las modificaciones en caso de ser necesario, identificando también si se trata de un código óptimo.
- Tener cuidado en buscar respuestas perfectas a través de una serie de preguntas. Generalmente es preferible tomar una respuesta incompleta y ajustarlas a las necesidades del momento.
- Es una guía, pero no reemplaza el factor humano de saber dónde, cómo y para qué son esos fragmentos de código. Es un apoyo para el programador y se debe utilizar con coherencia y también estudiar el tema de prompts para pulir las órdenes.
- Probar y validar.
- Consultar de una manera correcta que es lo que se necesita, en inglés si es posible, usar como fuente de ayuda no para generar el código final.
- Usarlo únicamente como herramienta auxiliar, la IA no siempre resuelve los problemas por si sola.
- Son opciones presentes en la actualidad que se pueden aprovechar para mejores resultados en el trabajo.
- Analizar bien los requisitos del usuario y probar para ver si es lo que se quiere obtener.
- Realizar pruebas de rendimiento.
- Analizar las soluciones obtenidas y dar información adicional para que pueda generar código más fiable y reutilizable.

Comentarios adicionales:

- Las IA para desarrollo se están perfeccionando, pero para soluciones de desarrollo tipo ERP o sistemas de alta complejidad resultan ser aún demasiado básicas.
- La IA puede ser un gran aliado con conocimientos previos de lo que se desea desarrollar.

3.7.2 Preselección de herramientas IA

Los resultados obtenidos de las entrevistas proporcionan la información necesaria para realizar una preselección de 10 herramientas IA para la generación de código, de acuerdo a los datos, derivados de las experiencias y opiniones de profesionales en el campo del desarrollo de software, y el análisis de los criterios considerados, como la precisión, la integración con entornos de desarrollo existentes, la facilidad de uso y la adaptabilidad a diferentes contextos de desarrollo, con lo que se ha logrado establecer una lista preliminar de herramientas que presentan un potencial para cubrir las necesidades de los desarrolladores.

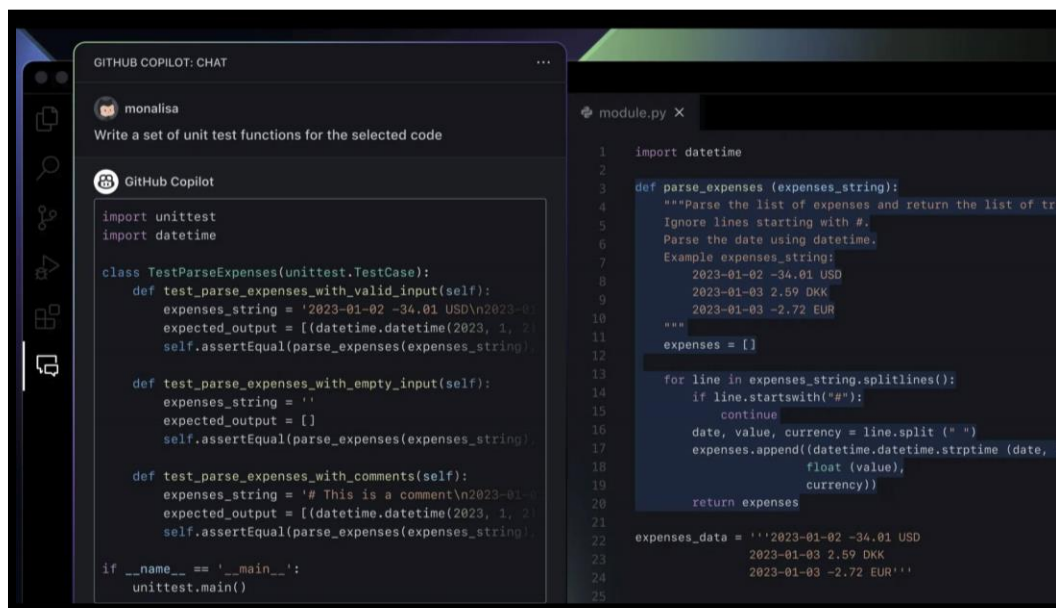
Estas herramientas preseleccionadas constituyen la base y punto de partida para el análisis experimental posterior, pues brinda pautas sobre la cual llevar a cabo pruebas y evaluaciones más detalladas. El contar con respaldo de las opiniones y experiencias de profesionales calificado, permite tener confianza en que las herramientas seleccionadas tienen la características y funciones necesarias que aporte al desarrollo de código.

I. GitHub Copilot

Figura 10. *Logo GitHub Copilot*



Figura 11. Interfaz GitHub Copilot



Reseña: GitHub Copilot es una herramienta de asistencia de código de inteligencia artificial desarrollada conjuntamente por GitHub y OpenAI, que puede generar automáticamente fragmentos de código de alta calidad, información contextual, etc. A través del procesamiento del lenguaje natural y la tecnología de aprendizaje automático, puede generar código automáticamente analizando el código, los comentarios y la información contextual escrita por los programadores (Torres, 2021).

Tabla 3. Ficha informativa de GitHub Copilot

Ficha informativa de GitHub Copilot	
Fecha de lanzamiento	Octubre - 2021
Tipo de programa	Generación de código
Desarrollador	GitHub, OpenAI
Tipo de licencia	Licencia privativa
Compatibilidad de lenguaje	Multilenguaje (Python – HTML - JavaScript – TypeScript – Ruby – Go)

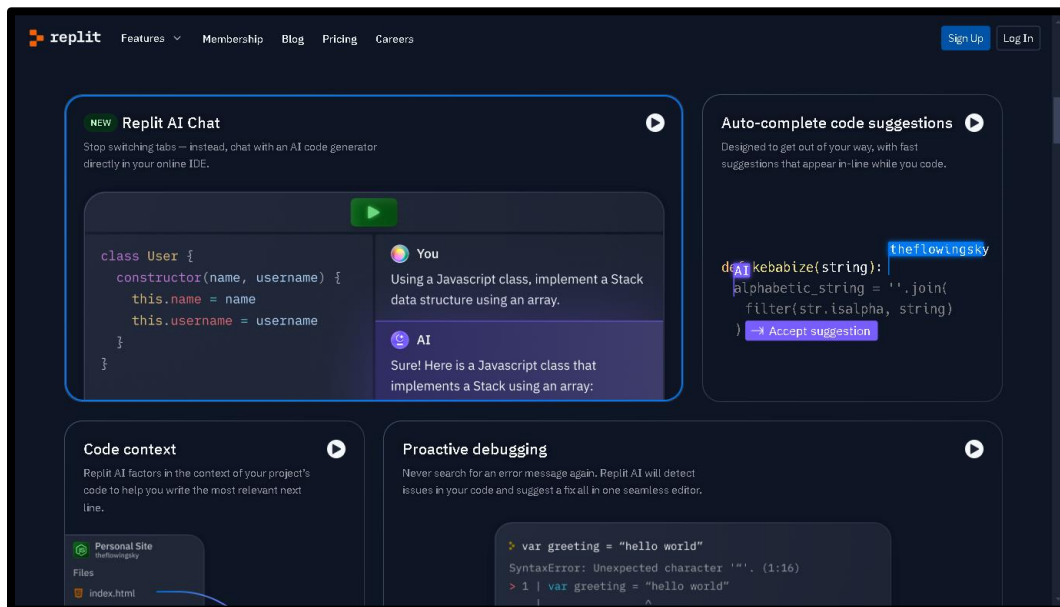
Reconocimiento de voz	Integrado
Características	Dispone de comando de voz en Copilot
	Copilot sugerirá automáticamente algunos posibles fragmentos de código según el contexto y las reglas de sintaxis
	Utiliza tecnología avanzada de procesamiento de lenguaje natural y algoritmos de aprendizaje automático

II. Replit GhostWriter

Figura 12. Logo Replit GhostWriter



Figura 13. Interfaz Replit GhostWriter



Reseña: Ghostwriter es una herramienta sencilla que sirve completar el código automáticamente y responder a consultas en lenguaje natural, etc., que actúa como un asistente fantasma detrás de escena, brindando a los desarrolladores un soporte y facilitándoles la escritura de código complejo. El bot de Ghostwriter comprende la intención de los desarrolladores y genera fragmentos de código según sus necesidades.

Tabla 4. Ficha informativa de Replit GhostWriter

Ficha informativa de Replit GhostWriter	
Fecha de lanzamiento	Octubre - 2022
Tipo de programa	Asistente de generación de código
Desarrollador	Replit
Tipo de licencia	Gratuita
Compatibilidad de lenguaje	Multilenguaje
Reconocimiento de voz	Esta en desarrollo

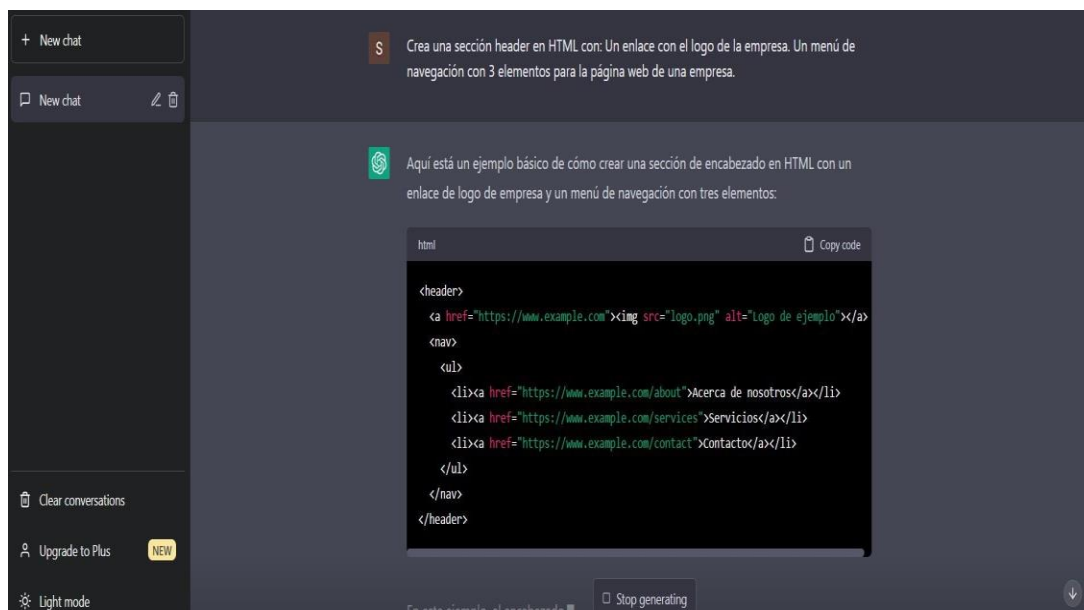
Características	Dispone de chatbot dentro de la misma ventana
	Admite prácticamente todos los lenguajes de programación
	Proporcionar sugerencias de depuración y reparación de errores en tiempo real

III. ChatGPT

Figura 14. Logo ChatGPT



Figura 15. Interfaz ChatGPT



Reseña: ChatGPT es una herramienta IA, que puede responder a preguntas sobre cualquier contenido planteado por los usuarios, y sus respuestas son bastante similares a las respuestas humanas, que sirve también para depurar código, escribir descripciones de código, convertir un lenguaje de programación a otro o realizar operaciones repetitivas estandarizadas de programación (Sisilitsyn & Osadchyi, 2023).

Tabla 5. *Ficha informativa de ChatGPT*

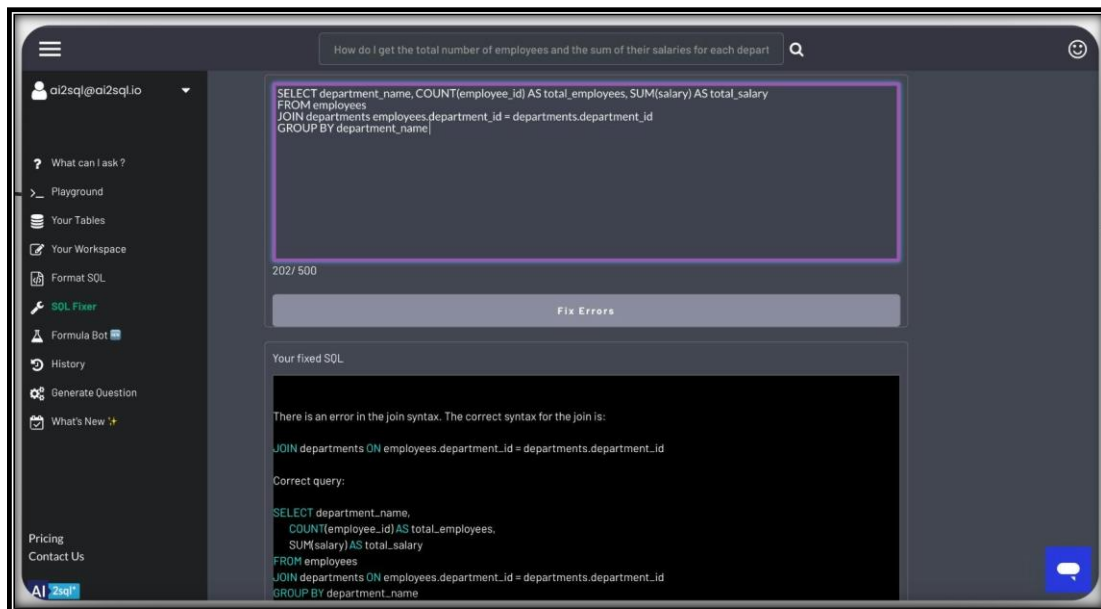
Ficha informativa de ChatGPT	
Fecha de lanzamiento	Noviembre - 2022
Tipo de programa	Chatbot – Generación de código
Desarrollador	OpenAI
Tipo de licencia	Licencia privativa
Compatibilidad de lenguaje	Multilenguaje (Principalmente HTML – JavaScript – CSS y Python)
Reconocimiento de voz	Se integra con complemento
Características	Recibe indicaciones mediante comando de voz clonada
	Funciona como un chat generativo previamente entrenado que aprovecha el procesamiento del lenguaje natural
	Se alimenta de información de fuentes de internet y posee aprendizaje automático

IV. AI2sql

Figura 16. *Logo AI2sql*



Figura 17. Interfaz AI2sql



Reseña: AI2SQL es una plataforma que permite a los usuarios crear modelos de aprendizaje automático personalizados e implementarlos en una base de datos SQL, donde los usuarios pueden integrar fácilmente el aprendizaje automático en aplicaciones y sistemas existentes mediante la creación de modelos utilizando una variedad de componentes prediseñados y luego implementándolos en una base de datos SQL.

Tabla 6. Ficha informativa de AI2sql

Ficha informativa de AI2sql	
Fecha de lanzamiento	Octubre - 2021
Tipo de programa	Generación de código SQL
Desarrollador	OpenAI
Tipo de licencia	Licencia de pago
Compatibilidad de lenguaje	SQL, MySQL, SQL Server y PostgreSQL
Reconocimiento de voz	Se puede integrar con complemento

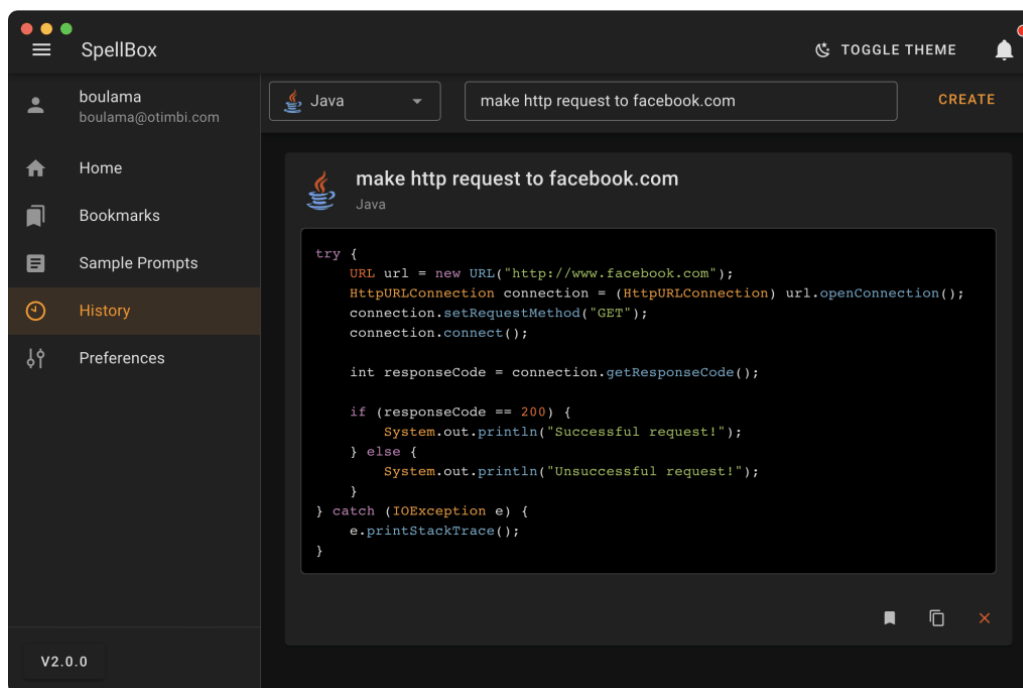
Características	Contiene una función de embellecimiento de código que lo hace más legible
	Su algoritmo integra características efectivas de optimización y prevención de errores
	Transforma las consultas de lenguaje natural en código SQL

V. Spellbox

Figura 18. Logo Spellbox



Figura 19. Interfaz Spellbox



Reseña: SpellBox es una aplicación de codificación de IA que proporciona asistentes de codificación para programadores para resolver cualquier problema de programación o ingeniería al instante, generando código multiplataforma basado en requisitos de texto.

Tabla 7. Ficha informativa de Spellbox

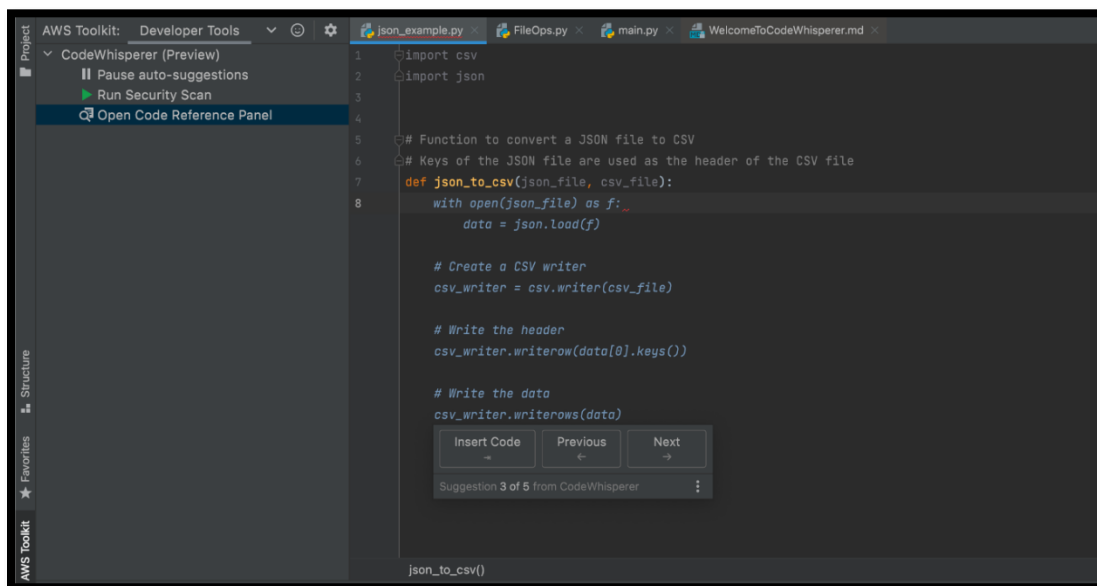
Ficha informativa de Spellbox	
Fecha de lanzamiento	Enero - 2023
Tipo de programa	Asistente de programación
Desarrollador	Spellbox
Tipo de licencia	Licencia de pago
Compatibilidad de lenguaje	Python – HTML - JavaScript – CSS – Java - Rust
Reconocimiento de voz	No disponible
Características	Sus principales atributos es que evita errores de sintaxis, realiza depuración y búsqueda efectiva
	Guarda fragmentos de código que son fáciles de localizar
	Tiene función de explicación de código lo que facilita el trabajo a desarrolladores principiantes

VI. Amazon CodeWhisperer

Figura 20. Logo Amazon CodeWhisperer



Figura 21. Interfaz Amazon CodeWhisperer



Reseña: Amazon CodeWhisperer es una IA de la compañía Amazon que funciona como asistente de generación código, que funciona con base a modelos y plantillas que se seleccionan de manera sencilla y que se puede modificar de acuerdo a los requerimientos del usuario, y que además utiliza tecnología de aprendizaje automático y, cuando se habilita en varios IDE, genera automáticamente código a partir de comentarios.

Tabla 8. Ficha informativa de Amazon CodeWhisperer

Ficha informativa de Amazon CodeWhisperer	
Fecha de lanzamiento	Junio - 2022
Tipo de programa	Generación de código
Desarrollador	AWS
Tipo de licencia	Licencia privativa
Compatibilidad de lenguaje	Multilenguaje (Python, JavaScript, C#, Go, Rust, PHP, Ruby, Kotlin, C, C++, SQL, Scala, HCL)
Reconocimiento de voz	No disponible

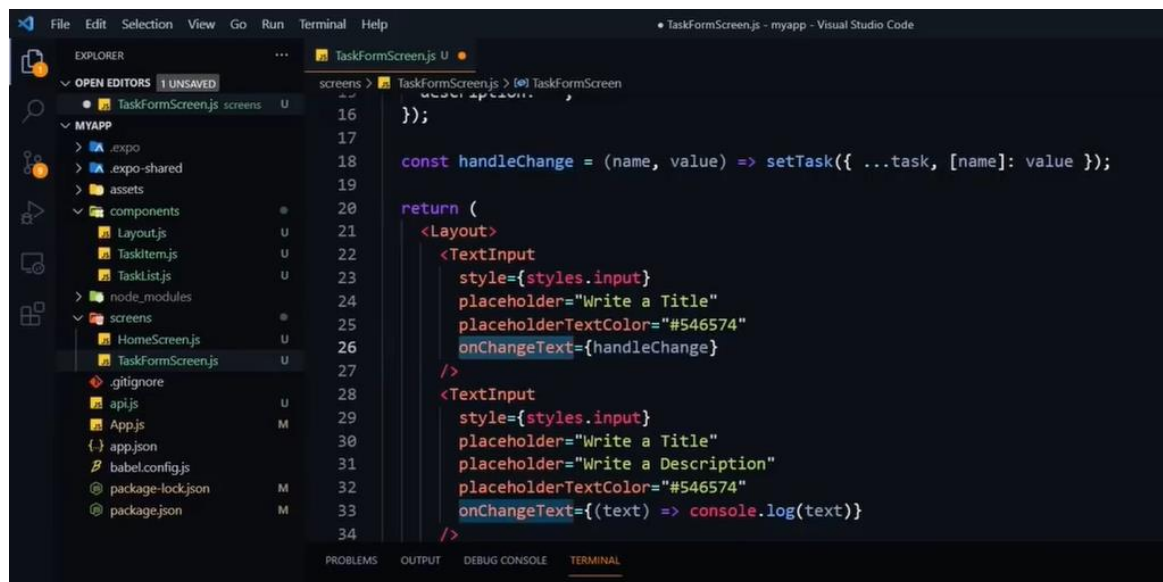
Características	Dispone de la función de escaneo de seguridad
	Filtra sugerencias de código sesgadas y mejorar la precisión del código.
	Verifica continuamente el código y brinda recomendaciones gramaticalmente correctas que se sintetizan en función de su estilo de codificación y nombres de variables.

VII. Tabnine

Figura 22. Logo Tabnine



Figura 23. Interfaz Tabnine



Reseña: TabNine es un complemento de codificación de autocompletado, que a diferencia de los complementos de autocompletado ordinario, utiliza el aprendizaje profundo para ayudar a completar el código.

Tabla 9. *Ficha informativa de Tabnine*

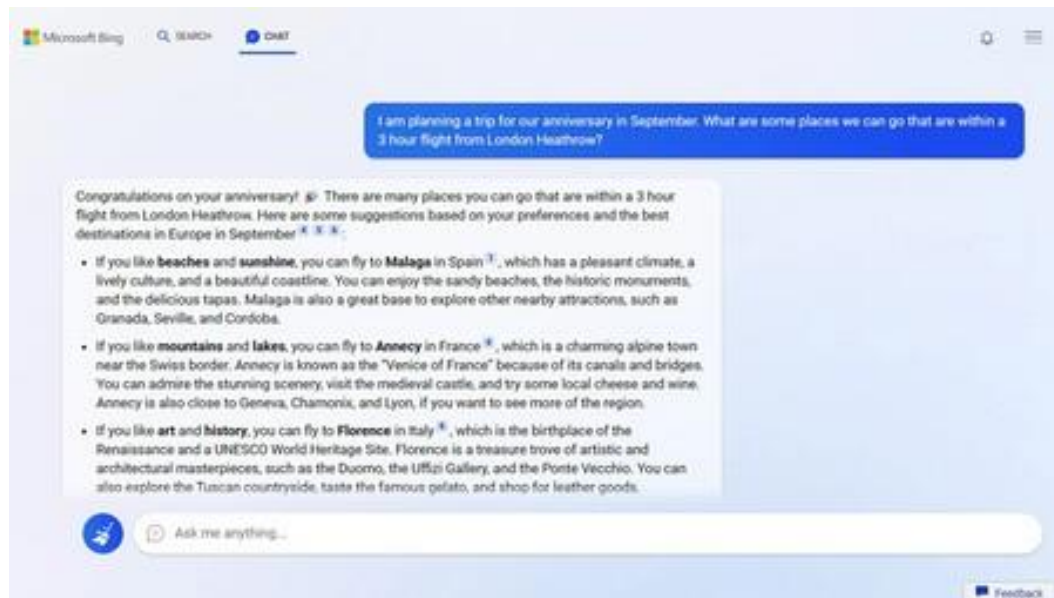
Ficha informativa de Tabnine	
Fecha de lanzamiento	Octubre - 2018
Tipo de programa	Asistente de programación y generación de código
Desarrollador	Tabnine
Tipo de licencia	Licencia privativa
Compatibilidad de lenguaje	Multilenguaje (Python, JavaScript, Java, C++, Go, TypeScript, SQL, HTML, CSS)
Reconocimiento de voz	En desarrollo
Características	Su actualización y soporte es automatizado
	Utiliza el aprendizaje profundo para aumentar la velocidad y precisión
	Es compatible con casi todos los idiomas e IDE

VIII. Bing Chat

Figura 24. *Logo Bing Chat*



Figura 25. Interfaz Bing Chat



Reseña: Bing Chat es un buscador y asistente de generación de código basado en IA perteneciente a la compañía Microsoft que, ayuda con cualquier trabajo de codificación, desde generar código hasta explicar la lógica detrás de él, desde desarrollar un sitio web de WordPress hasta crear una aplicación e incluso refactorizar su código existente.

Tabla 10. Ficha informativa de Bing Chat

Ficha informativa de Bing Chat	
Fecha de lanzamiento	Julio - 2022
Tipo de programa	Multifunción (Chatbot, Diseño, Generación de código)
Desarrollador	Microsoft
Tipo de licencia	Licencia gratuita
Compatibilidad de lenguaje	Multilenguaje (Python, HTML, CSS, JavaScript – Java - C++ - Go - TypeScript)
Reconocimiento de voz	Integrado

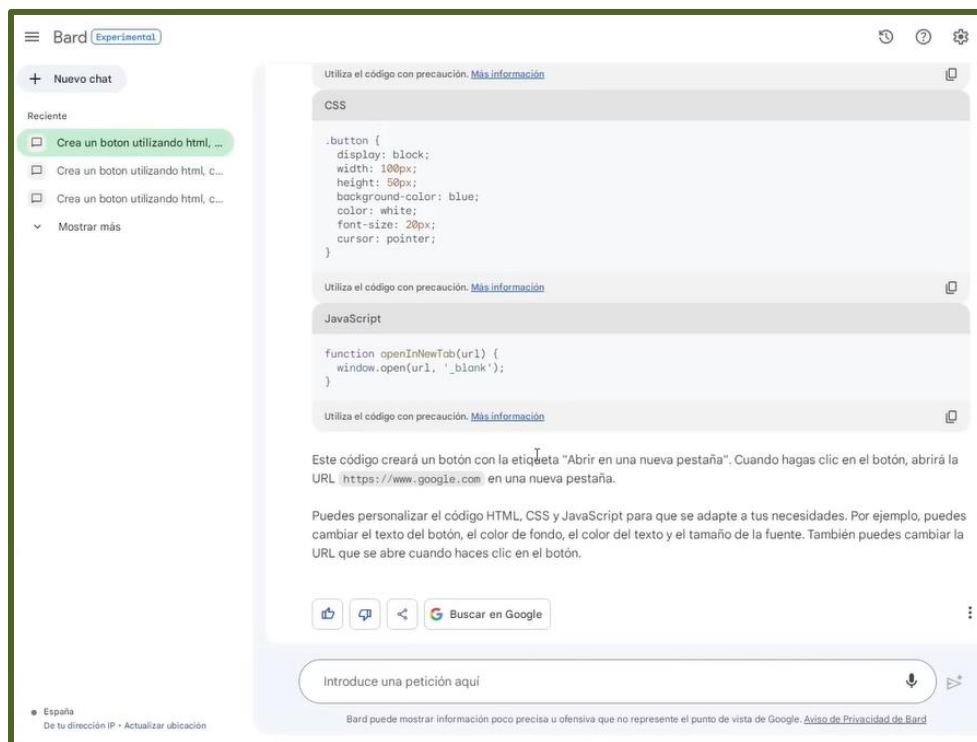
Características	Su función primordial se centra en la sintaxis
	Se caracteriza por la velocidad debido a sus funciones de Chat y Workbooks
	La privacidad, seguridad y almacenamiento son sus principales atributos

IX. Gemini (Bard)

Figura 26. Logo Gemini



Figura 27. Interfaz Gemini



Reseña: Google Bard o Gemini es una herramienta IA de codificar, predecir, hacer referencia, experimentar y crear imágenes al mismo tiempo que se integra con otros programas por lo que permiten agregar columnas adicionales a las hojas y compartirlas fácilmente.

Tabla 11. *Ficha informativa de Gemini*

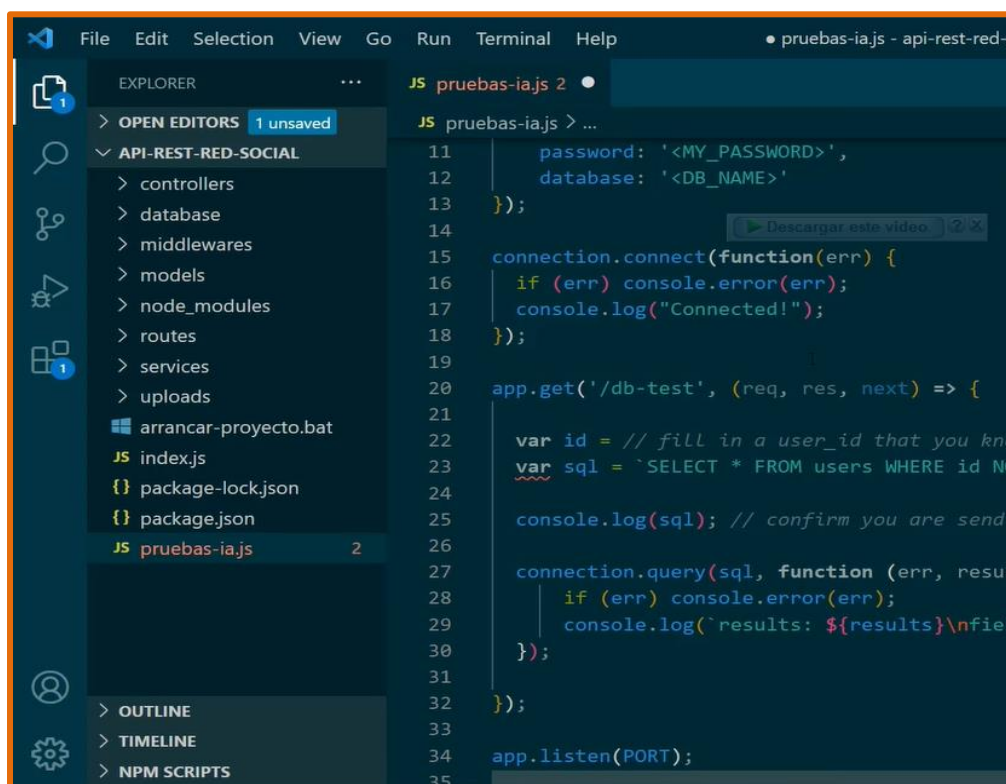
Ficha informativa de Gemini (Bard)	
Fecha de lanzamiento	Abril - 2023
Tipo de programa	Chatbot - Generación de código
Desarrollador	Google
Tipo de licencia	Gratuita
Compatibilidad de lenguaje	Multilenguaje (Python, HTML, CSS, JavaScript – Java - C++ - Go - TypeScript)
Reconocimiento de voz	Integrado
Características	Está diseñado para redactar, generar ideas y proyectos, y desarrollar código
	Utiliza procesamiento del lenguaje natural (NLP) y aprendizaje automático (ML) para simular conversaciones humanas.
	Integra la herramienta de Google de control de voz

X. AlphaCode

Figura 28. *Logo AlphaCode*



Figura 29. Interfaz AlphaCode



Reseña: AlphaCode es una herramienta que utiliza un modelo de lenguaje para generar código a partir de una gran cantidad de muestras de programas, las filtra de acuerdo con los resultados de la ejecución, obtiene los resultados de las pruebas de muestra de la descripción del problema y luego agrupa las muestras restantes para obtener una pequeña cantidad de envíos finales para evaluación.

Tabla 12. Ficha informativa de AlphaCode

Ficha informativa de AlphaCode	
Fecha de lanzamiento	Febrero – 2022
Tipo de programa	Generación de código
Desarrollador	DeepMind
Tipo de licencia	Licencia privativa
Compatibilidad de lenguaje	Python – JavaScript – TypeScript – Ruby – Go

Reconocimiento de voz	No disponible
Características	Se basa en el modelo Transformer
	Utiliza un protocolo muy específico al resolver el problema de escribir código, y este protocolo determina la canalización del sistema
	Se basa en el código de Github y otras herramientas

Análisis de resultados de la población de investigación

De las 10 herramientas que se ha tomado como población de investigación, se realiza un análisis para determinar cuáles son las IA más óptimas para el análisis multicriterio y pruebas de acuerdo a los parámetros del estudio; por tal motivo se realizan dos tablas, en la primera se evalúan las herramientas según las características técnicas, y la segunda se evalúan las características de usabilidad, teniendo un total de las 10 características más importantes que debe tener una IA.

Tabla 13. *Análisis multicriterio con base a características técnicas*

IA	Interfaz	Funciones	Licencias	Soporte	Seguridad
Copilot	X	X		X	X
Replit					
GhostWriter	X	X	X	X	X
ChatGPT	X	X	X	X	X
AI2sql	X	X		X	X
Spellbox	X	X		X	X
Amazon CW	X	X		X	X
Tabnine	X	X		X	X
Bing Chat	X	X	X	X	X
Gemini	X	X	X	X	X
AlphaCode	X	X		X	X

Tabla 14. *Análisis multicriterio con base a características de usabilidad*

IA	Precisión De Voz	Facilidad De Uso	Eficiencia Y Velocidad	Adaptabilidad	Compatibilidad
Copilot	X	X	X	X	X
Replit GhostWriter	X		X	X	X
ChatGPT	X	X	X	X	X
AI2sql	X	X	X	X	X
Spellbox		X	X	X	X
Amazon CW		X	X	X	X
Tabnine		X	X	X	X
Bing Chat	X	X	X	X	X
Gemini	X	X	X	X	X
AlphaCode		X	X	X	X

Con las dos tablas donde se refleja la evaluación de las características de la IA, se presenta la siguiente tabla con los resultados del análisis multicriterio.

Tabla 15. *Resultados del análisis multicriterio*

IA	Total
Copilot	9
Replit GhostWriter	9
ChatGPT	10
AI2sql	9
Spellbox	8
Amazon CW	8
Tabnine	8
Bing Chat	10
Gemini	10
AlphaCode	8

Requisitos a considerar para la selección de las herramientas:

1. Herramientas generadoras de código (Todas cumplen).
2. Licencia Libre (Licencias).
3. Disponibilidad de Voz (Precisión de voz).

ChatGPT, aunque su enfoque principal es responder preguntas y generar texto, demuestra ser útil para recibir indicaciones mediante comandos de voz clonados, lo que facilita la interacción para la generación de código; conjuntamente con sus funciones de compresión, llega a ser casi como un razonamiento humano, permite que las instrucciones que se emiten, se generen de la forma en que se espera.

Gemini (Bard), los datos recopilados muestran que esta herramienta utiliza técnicas avanzadas de NLP, por lo cual se adapta mejor los comandos de voz y generar código de manera más precisa; así como también el uso de algoritmos ML hace que esta herramienta mejore continuamente su capacidad de adaptabilidad; y su compatibilidad con varios lenguajes de programación, es algo interesante de someter a pruebas.

Bing Chat, herramienta que no solo proporciona sugerencias de código basadas en el contexto y las reglas de sintaxis, sino que también dispone de comandos de voz en Microsoft, lo que la hace especialmente útil para la generación de código por comandos basados en voz, que es el propósito de la investigación.

Replit Ghostwriter, A diferencia de lo que se ha señalado en ChatGPT o Gemini, esta herramienta no es un chatbot, sino que está diseñada específicamente para programación, por lo que la misma ventana hace que se tenga una interacción más fluida para la generación de código, incluso no es específicamente mediante comandos de voz, sino usando texto.

Estas cuatro herramientas se ajustan a los criterios indicados de acuerdo a sus capacidades y su relación con el propósito de la investigación, principalmente por que han demostrado en la práctica ser valiosas en el ámbito de la generación de código, además, tienen integrado (y se pueden integrar) comandos basados en voz, debido a su capacidad para proporcionar sugerencias precisas y útiles, así como su facilidad de interacción.

Además de que todas ofrecen funcionalidades avanzadas en el procesamiento de lenguaje natural, lo que les permite comprender y responder de manera efectiva, son compatibles con varios tipos de lenguaje de programación, haciendo que exista facilidad para el uso de prompts, ya sea para un programador novato o experto. Todas estas herramientas cuentan con interfaces intuitivas y amigables, bastante similares unas de otras.

En conjunto, ChatGPT, Gemini, Bing y Replit son las cuatro herramientas IA que resultan más adecuadas para ser aprovechadas en el desarrollo del caso de estudio propuesto para este trabajo, que es, el software de manejo ambiental para la tenencia responsable de mascotas en

la ciudad de Cuenca. Estas herramientas facilitan el proceso de desarrollo al proporcionar sugerencias de código, comprender el lenguaje natural, generar respuestas automatizadas y ofrecer asistencia en tiempo real.

Por otro lado, las herramientas restantes, tomadas como población, no se han seleccionado para el análisis por diversas razones; en el caso de GitHub Copilot (a pesar de obtener una de las puntuaciones más altas) y AlphaCode, a la fecha en que se realiza este trabajo, cuentan con una licencia privativa, lo que limita su accesibilidad para ciertos usuarios, y la investigación se enfoca en código abierto. En el caso de Spellbox y AI2sql, aunque ofrece funciones relevantes, su licencia de pago lo hace una barrera adicional para su utilización, y por lo tanto su exclusión del estudio. Por otro lado, Amazon CodeWhisperer y Tabnine, aunque disponen de licencia gratuita, no cuentan con reconocimiento de voz como característica, lo cual es un requisito específico de la presente investigación.

3.8 Resultados de pruebas

Para las pruebas a realizar se toma como ejemplo el proceso de evaluación de modelos presente en la investigación de (De Vito et al., 2023), donde se manejan con base a criterios de evaluación de los casos de uso de una herramienta de software colocando parámetros de evaluación como número de cambios realizados para obtener el software deseado, esfuerzos, entre otros; teniendo en cuenta también el desarrollo de software ágil sobre Feature-driven development (Riady et al., 2022) donde se seleccionan las funciones principales que debe tener el software generado por las herramientas IA tomando en cuenta el enfoque principal del caso de estudio, junto con su correcto funcionamiento.

Esto mediante los siguientes parámetros de evaluación:

Tabla 16. Formato de evaluación de características para las herramientas IA seleccionadas en el estudio.

	Eficiencia	Precisión	Calidad de código
Análisis			
Ponderación	1-5	1-5	1-5

Los campos anteriormente colocados se describen a detalle a continuación:

- **Eficiencia:** La capacidad de generar resultados de manera rápida, coherente y satisfactoria.

- Precisión: Que tan exacto y apropiado es el resultado generado, en consecuencia, al prompt dictado.
- Calidad de código generado: El código generado sea cumpla con la funcionalidad requerida.

Con los parámetros de evaluación determinados, se pondrá a prueba a continuación las herramientas IA seleccionadas, mediante el caso de estudio, en el que se sugiere la elaboración de un software, en este caso, página web; esta página web debe permitir: el registro de los datos de las mascotas; la consulta de aquellos datos ingresados; y, la generación de informes sobre las mascotas registradas. Todo esto con el fin de determinar cuál herramienta IA se adapta y cumple con mayor satisfacción el objetivo planteado.

3.8.1 ChatGPT

Como se ha explicado anteriormente la IA ChatGPT, requiere de otra herramienta para introducir órdenes por comandos de voz para generación de código; lo que se realizó inicialmente a través de la extensión de Google llamada Talk-to-ChatGPT (Figura 30), que se requiere instalar en nuestra computadora, para interactuar de manera natural con la herramienta, sin necesidad de escribir los comandos manualmente, y generar código simplemente utilizando nuestra voz. Sin embargo, por cuestiones de falta de soporte en esta extensión, quedó fuera de uso y en la versión 3.5 de ChatGPT por lo que es factible utilizar otra extensión disponible también en la tienda de extensiones de Google Chrome llamada Talk to Mia que es con la que se va a trabajar para las pruebas (Figura 31).

Figura 30. Talk to ChatGPT (Extensión para entrada por voz en ChatGPT)

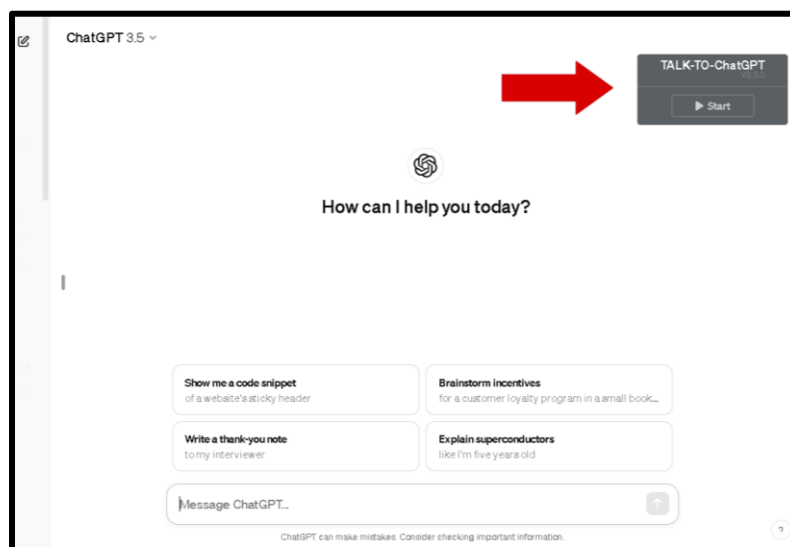
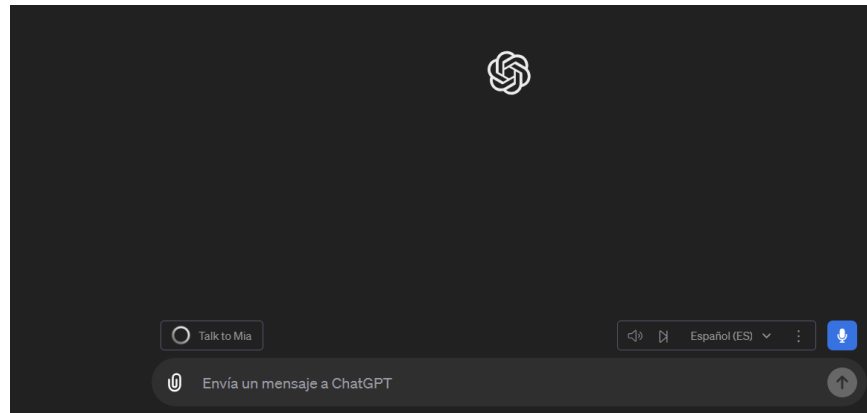
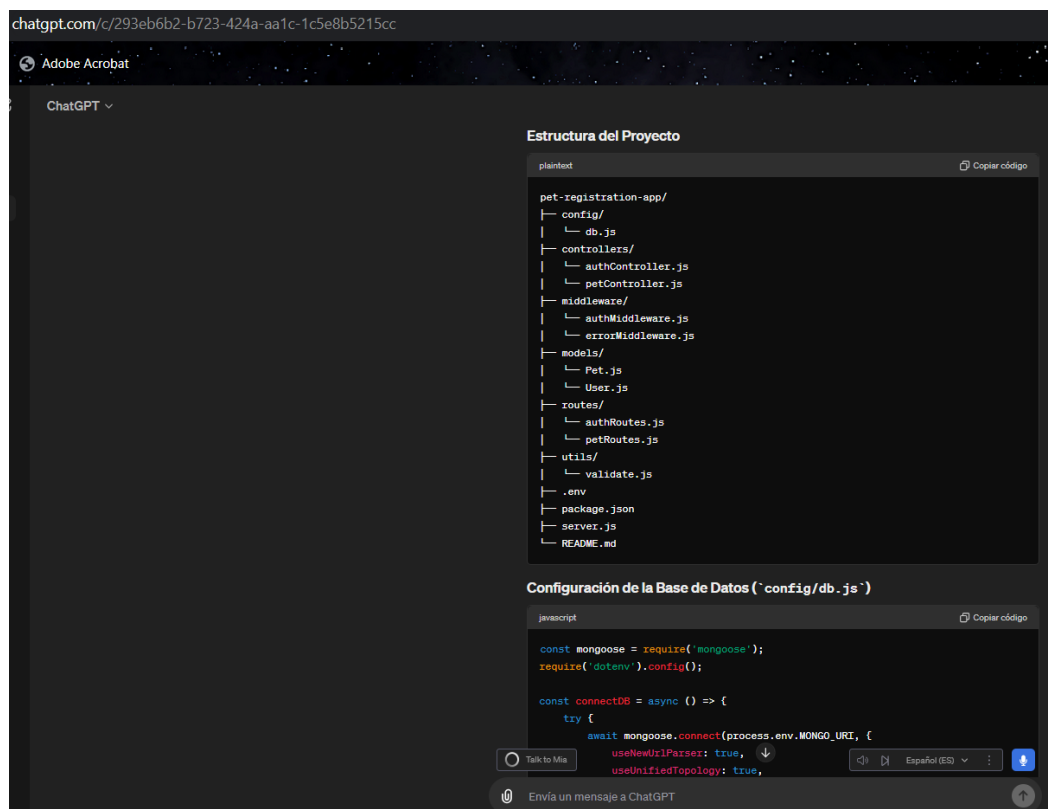


Figura 31. *Talk to Mia (Extensión para entrada por voz en ChatGPT)*



Una vez ingresado el **primer prompt**, propuesto dentro de la metodología, para la escritura de código tanto en HTML, CSS y JavaScript, se ha obtenido una estructura básica de la página web, con una estructura de proyecto bien definida y los procesos de configuración de la base de datos de tal forma que el usuario pueda realizar una conexión y uso inicial de forma rápida, en la Figura 32 se puede observar parte de los resultados obtenidos con este prompt dentro de ChatGPT.

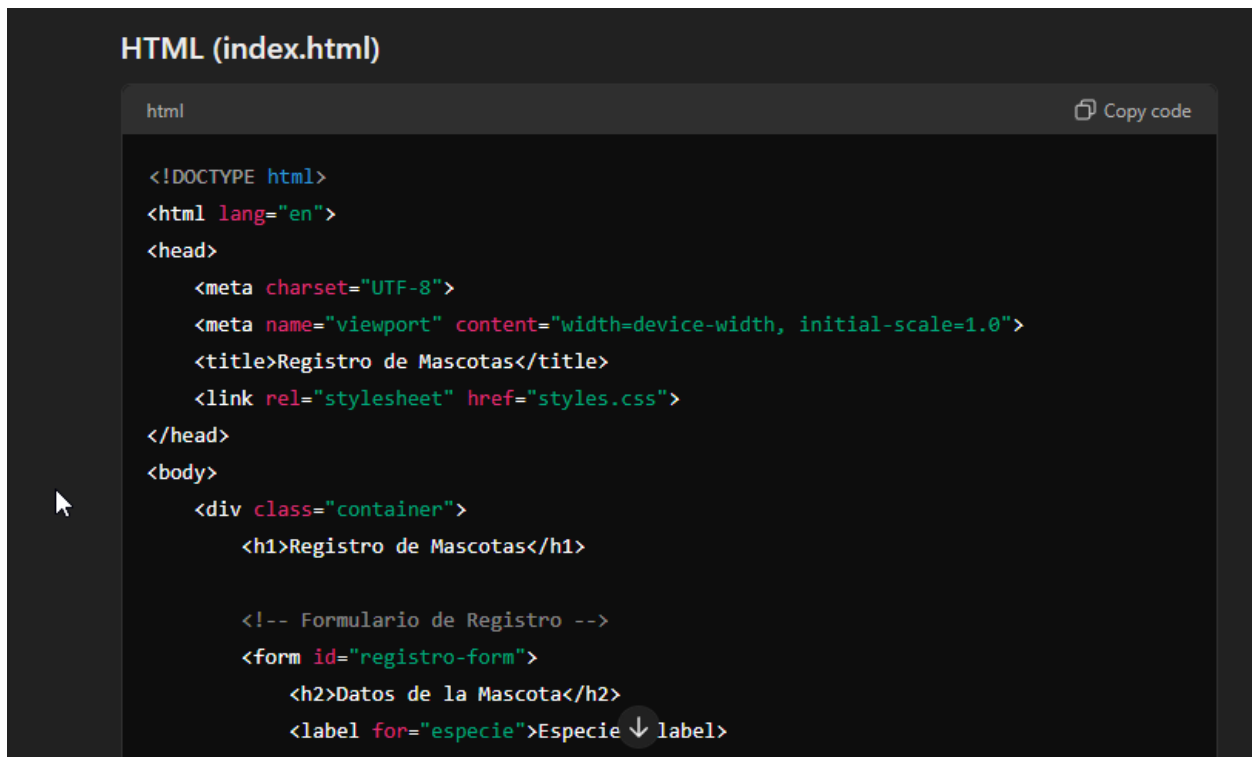
Figura 32. *Resultados en ChatGPT sobre el primer prompt*



Es importante destacar que el primer prompt es para inicializar el contexto de la conversación con la herramienta IA, con el fin de que para el dictado del siguiente prompt, se puede obtener de forma rápida un resultado concreto y funcional con base a la solicitud y requisitos exactos que se indican, ya que en este caso se debe tomar en cuenta que la medición de tiempo y esfuerzo por parte del programador, está en tener un clúster en este caso con MongoDB Atlas para realizar las configuraciones necesarias en cuanto a la conexión con la base de datos, para luego tener el código funcional de forma local y con esto agregar o quitar las funciones o elementos que se considere. Se coloca evidencias del contenido de cada archivo, estructura del proyecto inicial propuesto por ChatGPT en la sección de anexos.

Se procede con el dictado del segundo prompt, a través del cual se especifica a ChatGPT que se requieren únicamente 3 archivos, HTML, CSS y JavaScript con el fin de realizar las pruebas funcionales dentro de la herramienta Visual Studio Code. Se muestra parte del contenido del código generado en HTML en el caso de segundo prompt en la Figura 33.

Figura 33. Contenido HTML generado con el segundo prompt en ChatGPT.



```
HTML (index.html)

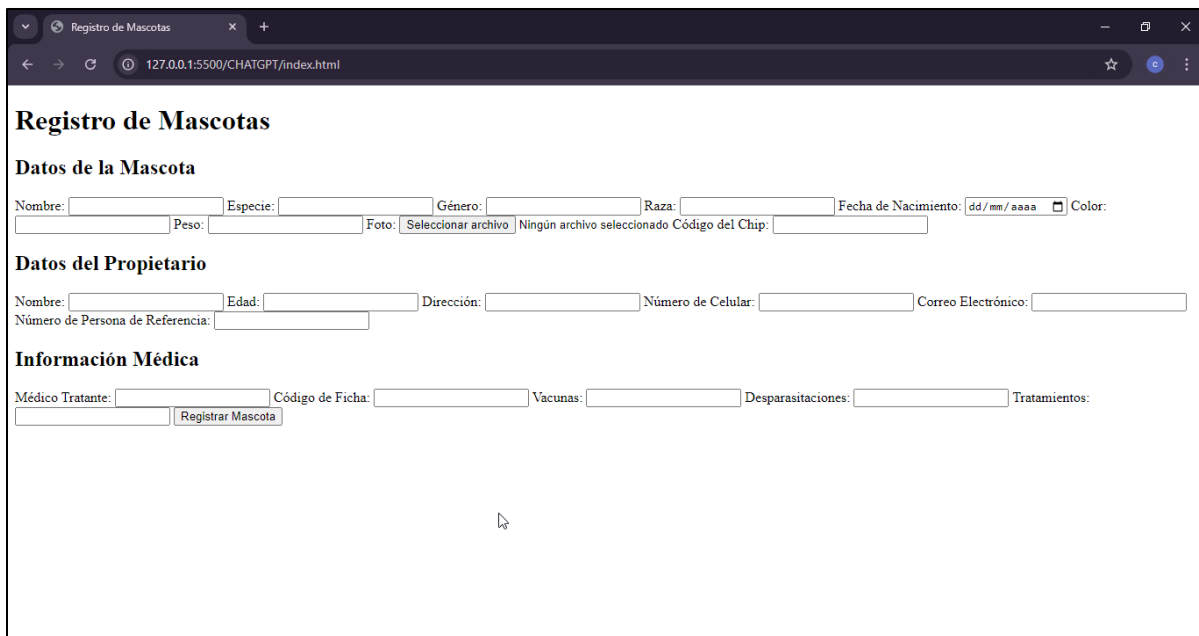
html Copy code

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registro de Mascotas</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Registro de Mascotas</h1>

    <!-- Formulario de Registro -->
    <form id="registro-form">
      <h2>Datos de la Mascota</h2>
      <label for="especie">Especie ↓ label>
```

En el proceso práctico, pegando el código HTML en Visual Studio Code, ha dado como resultado básicamente tres segmentos de un software; primero el título de programa, junto con el área para registrar la mascota; el siguiente segmento que consta de apartados para registrar al propietario de la mascota; y, el tercer segmento, la información médica, para cerrar con un botón que permite registrar a la mascota, una vez incluidos los datos pertinentes. Se incluye evidencia de este proceso en la Figura 34.

Figura 34. Vista previa del HTML generado por ChatGPT resultado del segundo prompt.



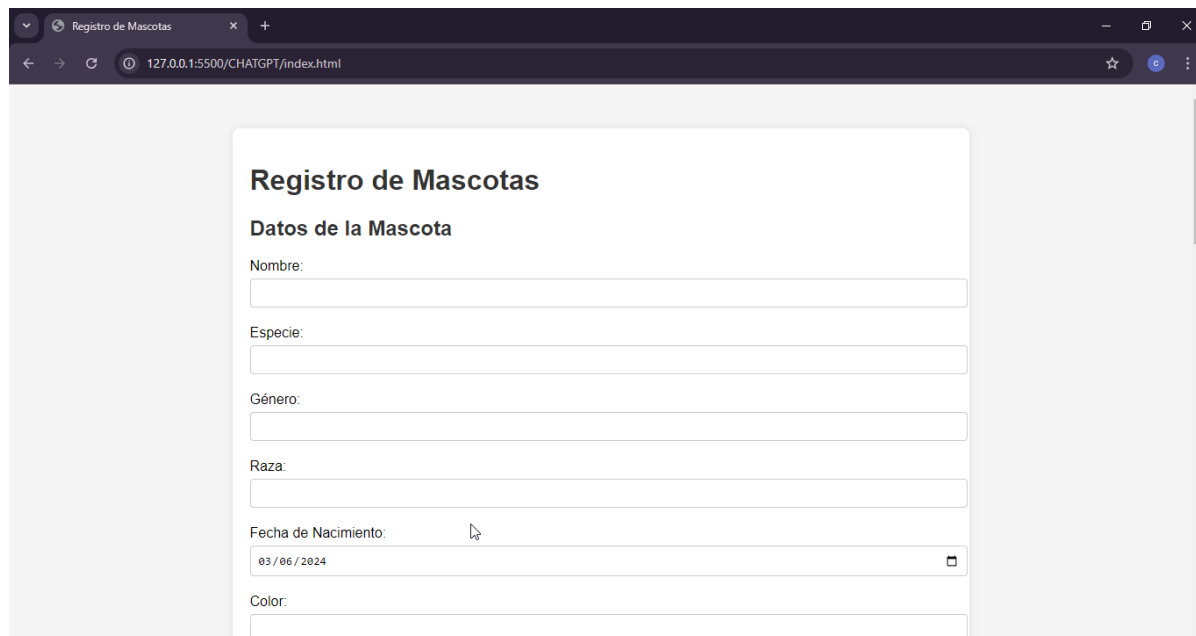
The screenshot shows a web browser window with the title 'Registro de Mascotas'. The address bar displays '127.0.0.1:5500/CHATGPT/index.html'. The form is organized into three main sections:

- Datos de la Mascota**: Includes input fields for Nombre, Especie, Género, Raza, Fecha de Nacimiento (with a date picker), Color, and a Foto field with a file selection button. There is also a field for Código del Chip.
- Datos del Propietario**: Includes input fields for Nombre, Edad, Dirección, Número de Celular, and Correo Electrónico. There is also a field for Número de Persona de Referencia.
- Información Médica**: Includes input fields for Médico Tratante, Código de Ficha, Vacunas, Desparasitaciones, and Tratamientos. A 'Registrar Mascota' button is located at the bottom of this section.

Por su parte, la escritura de código CSS proporciona estilos visuales para mejorar la apariencia y la utilización del usuario en la aplicación, definiendo reglas de estilo para diferentes elementos generados anteriormente en HTML, como el cuerpo de la página, el formulario, los campos de entrada, los botones y los mensajes de respuesta.

Además, se han especificado estilos para resaltar los campos de entrada y los botones, con colores de fondo y texto que hacen que los elementos sean claramente distinguibles y más llamativos; en tanto que para los mensajes de respuesta se han estilizado para indicar visualmente el éxito o el fracaso de las operaciones de registro mediante colores de fondo y texto para diferenciar entre mensajes de éxito y mensajes de error. En la Figura 35 se puede visualizar el resultado de aplicar el CSS generado por ChatGPT al HTML mostrado anteriormente.

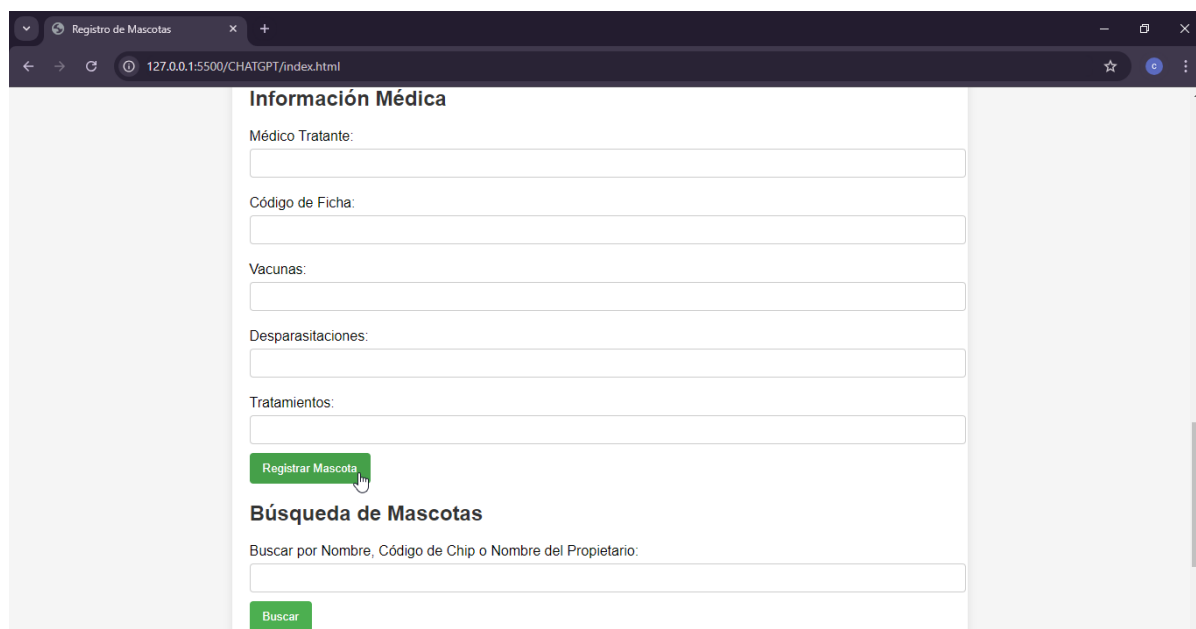
Figura 35. Contenido CSS generado por ChatGPT, aplicado al HTML generado inicialmente.



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5500/CHATGPT/index.html'. The page content is a form titled 'Registro de Mascotas' (Pet Registration). Below the title is a section 'Datos de la Mascota' (Pet Data) containing several input fields: 'Nombre:' (Name), 'Especie:' (Species), 'Género:' (Gender), 'Raza:' (Breed), 'Fecha de Nacimiento:' (Birth Date) with a date picker showing '03/06/2024', and 'Color:' (Color). The form is styled with a clean, modern look, featuring a light gray background and white input fields with thin borders.

En la herramienta de prueba los cambios que se ha dado mediante el código CSS se pueden evidenciar en los títulos de cada entrada, que se ven más notorias, así como también en las barras de ingreso y registro.

Figura 36. Prueba CSS del primer prompt en ChatGPT.

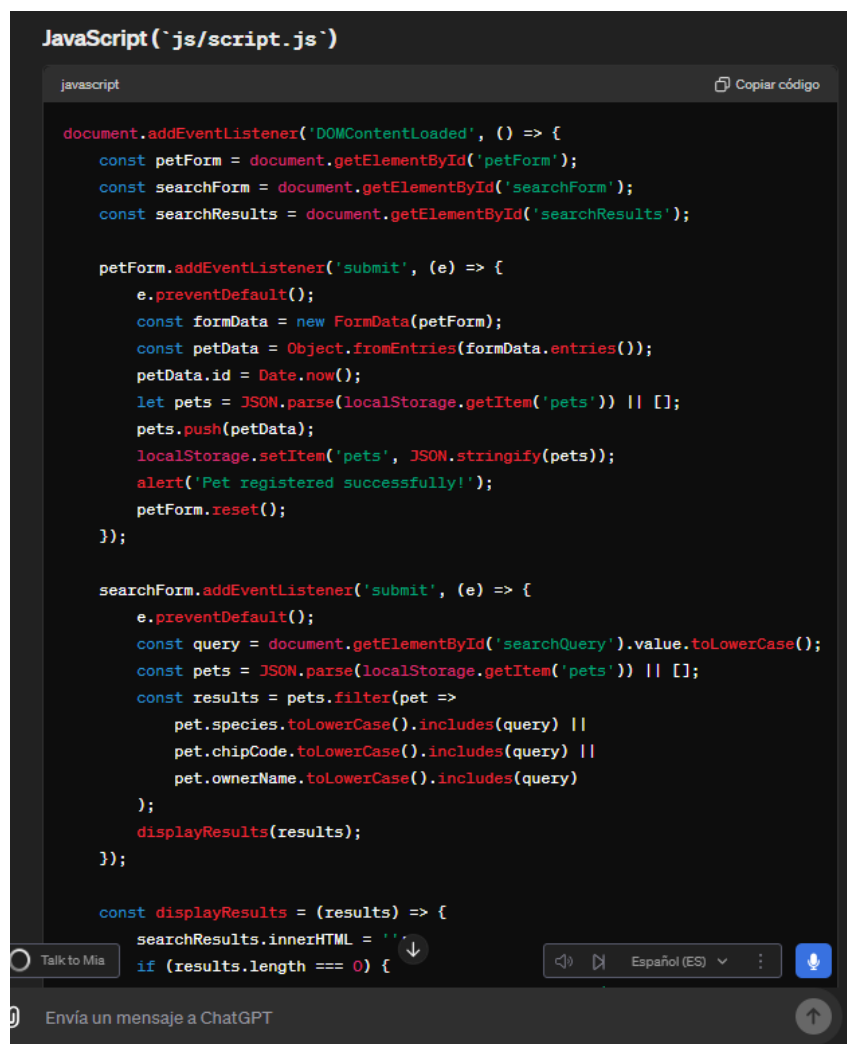


The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5500/CHATGPT/index.html'. The page content is a form titled 'Información Médica' (Medical Information). Below the title are several input fields: 'Médico Tratante:' (Treating Doctor), 'Código de Ficha:' (Record Code), 'Vacunas:' (Vaccines), 'Desparasitaciones:' (Deworming), and 'Tratamientos:' (Treatments). Below these fields is a green button labeled 'Registrar Mascota' (Register Pet). Below the button is a section titled 'Búsqueda de Mascotas' (Pet Search) with a text input field labeled 'Buscar por Nombre, Código de Chip o Nombre del Propietario:' (Search by Name, Chip Code or Owner Name) and a green button labeled 'Buscar' (Search). The form is styled with a clean, modern look, featuring a light gray background and white input fields with thin borders.

Para JavaScript la escritura de código desarrollado aporta funcionalidad interactiva a la aplicación, tomando elementos de la biblioteca jQuery, para simplificar algunas operaciones y definiendo el envío para el formulario de registro, es decir, cuando el usuario completa el formulario y lo envía, el evento de envío se activa, y se capturan los valores introducidos por el usuario, en este caso ChatGPT ha generado código JavaScript capaz de mantenerse funcional mediante el LocalStorage del navegador.

La petición se envía mediante formularios, teniendo el respectivo submit en el form de registro de mascotas, se maneja el proceso de almacenamiento local de la información en la base de datos, haciendo que la operación se realice con éxito, para que el servidor devuelva un mensaje de confirmación, que se muestra al usuario; o en caso contrario si se produce un error, se muestre el mensaje de error correspondiente.

Figura 37. JavaScript del segundo prompt generado por ChatGPT.



```
JavaScript (`js/script.js`)

javascript Copiar código

document.addEventListener('DOMContentLoaded', () => {
  const petForm = document.getElementById('petForm');
  const searchForm = document.getElementById('searchForm');
  const searchResults = document.getElementById('searchResults');

  petForm.addEventListener('submit', (e) => {
    e.preventDefault();
    const formData = new FormData(petForm);
    const petData = Object.fromEntries(formData.entries());
    petData.id = Date.now();
    let pets = JSON.parse(localStorage.getItem('pets')) || [];
    pets.push(petData);
    localStorage.setItem('pets', JSON.stringify(pets));
    alert('Pet registered successfully!');
    petForm.reset();
  });

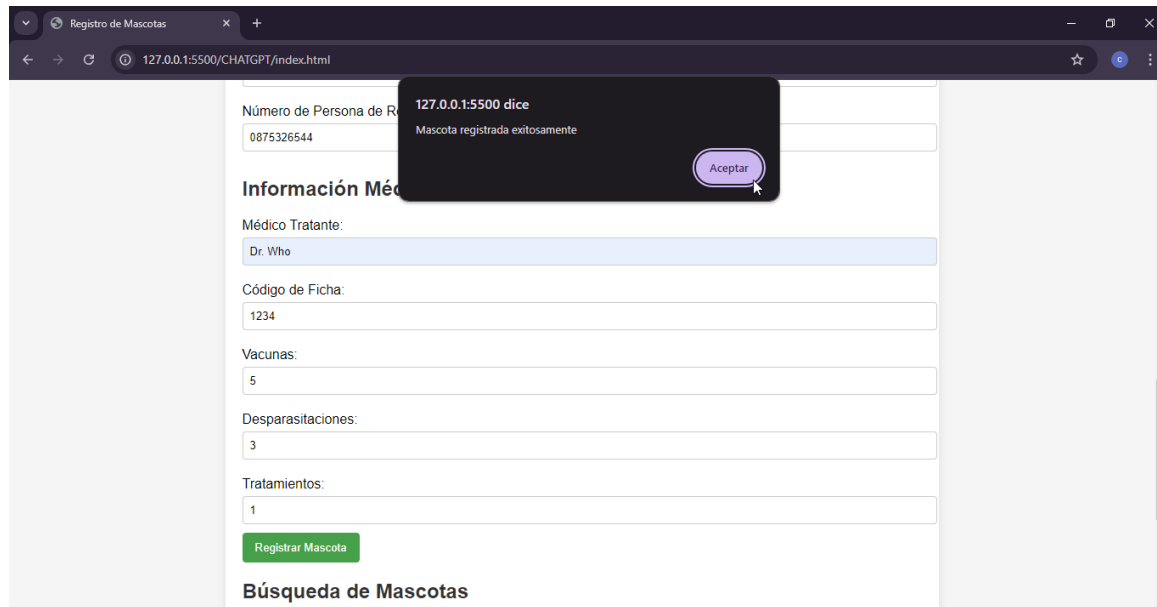
  searchForm.addEventListener('submit', (e) => {
    e.preventDefault();
    const query = document.getElementById('searchQuery').value.toLowerCase();
    const pets = JSON.parse(localStorage.getItem('pets')) || [];
    const results = pets.filter(pet =>
      pet.species.toLowerCase().includes(query) ||
      pet.chipCode.toLowerCase().includes(query) ||
      pet.ownerName.toLowerCase().includes(query)
    );
    displayResults(results);
  });

  const displayResults = (results) => {
    searchResults.innerHTML = '';
    if (results.length === 0) {

```

En el proceso práctico se verifica que, tras la incorporación del código a la herramienta Visual Studio Code, la aplicación permite ingresar datos de forma efectiva, teniendo también interacción dinámica en la parte del selector de fechas para el campo de fecha de nacimiento de la mascota. En la Figura 38 se puede visualizar un registro exitoso de una mascota dentro del aplicativo que se encuentra ya funcional gracias al LocalStorage. Inclusive se ha creado un campo que permite subir una fotografía de la mascota desde nuestro computador.

Figura 38. Registro exitoso de una mascota utilizando el código generado por ChatGPT.



En función al registro exitoso anterior, se tiene ingresada una mascota en el sistema con sus datos correspondientes, para comprobar esto se procede a utilizar la segunda función del sistema que es la consulta de la mascota por nombre, microchip o propietario. Tal como se muestra en la Figura 39, se realiza la búsqueda de la mascota con nombre Teo, donde se puede observar que en efecto la mascota existe en el sistema y se muestran algunos de los datos referentes a la misma.

Figura 39. *Búsqueda de una mascota en el aplicativo generado por ChatGPT.*

Registrar Mascota

Búsqueda de Mascotas

Buscar por Nombre, Código de Chip o Nombre del Propietario:

Buscar

Raul (665)

Mascota: Teo

Especie: Gato

Genero: Macho

Raza: Persa

Color: Naranja

Informes Poblacionales

Generar Informe

Como se puede apreciar, los resultados obtenidos de la búsqueda arrojan información importante de identificación, el nombre del propietario, detalles de la mascota, como su nombre, genero, raza, color, especie; funcionando correctamente. El código apreciado desde el entorno Visual Studio Code se puede observar en la Figura 40.

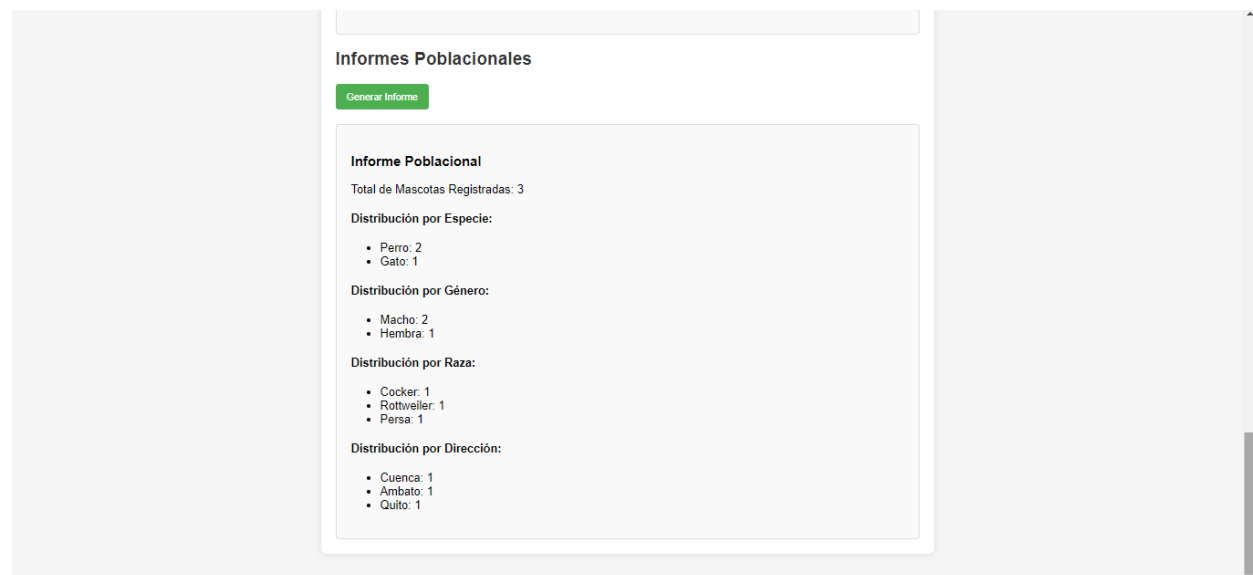
Figura 40. *Códigos generados por ChatGPT integrados a Visual Studio Code.*

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Registro de Mascotas</title>
7   <link rel="stylesheet" href="styles.css">
8 </head>
9 <body>
10  <div class="container">
11    <h1>Registro de Mascotas</h1>
12
13    <!-- Formulario de Registro -->
14    <form id="registro-form">
15      <h2>Datos de la Mascota</h2>
16      <label for="especie">Especie:</label>
17      <input type="text" id="especie" name="especie" required>
18
19      <label for="genero">Género:</label>
20      <input type="text" id="genero" name="genero" required>
21
22      <label for="raza">Raza:</label>
23      <input type="text" id="raza" name="raza" required>
24
25      <label for="fechaNacimiento">Fecha de Nacimiento:</label>
26      <input type="date" id="fechaNacimiento" name="fechaNacimiento" required>
27
28      <label for="color">Color:</label>
29      <input type="text" id="color" name="color" required>
30
31      <label for="peso">Peso:</label>
32      <input type="number" id="peso" name="peso" step="0.01" required>
```

El código HTML se ha enriquecido con una sección de información médica para ingresar detalles sobre el médico tratante, el código de ficha, las vacunas, las desparasitaciones y los tratamientos de la mascota, que es importante para mantener un registro detallado de la salud y el bienestar de las mascotas, lo que facilita su seguimiento. Otra nueva funcionalidad es la inclusión de una sección de búsqueda de mascotas en el sistema que permite a los usuarios buscar mascotas por su nombre, número de chip o nombre del propietario, proporcionando una manera de acceder a la información almacenada en la base de datos. El software se divide en 4 apartados, que son datos de la mascota, datos de propietarios, datos médicos y búsqueda; así como también dentro de cada apartado se observa diferentes campos para ingresar información completa, sobre cada uno de los aspectos de manera precisa.

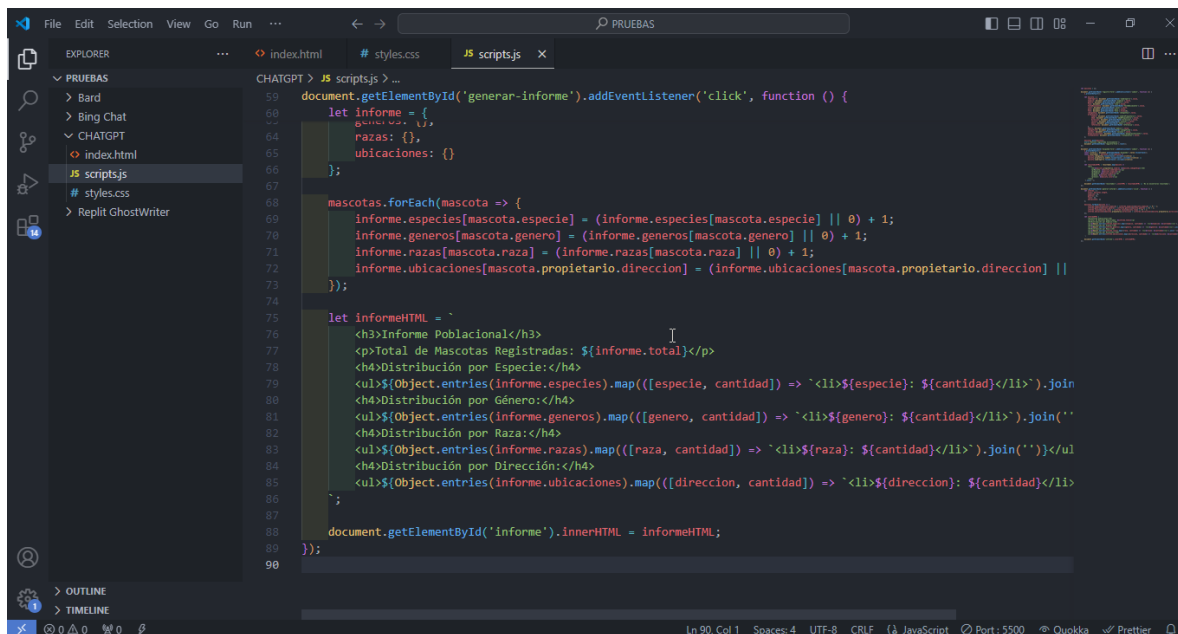
Finalmente, solicitando código a ChatGPT, mediante el **tercer prompt**, se han creado secciones dentro del mismo código anterior debido al contexto que se llevaba dentro de la conversación, con identificadores que muestran las estadísticas poblacionales mediante la opción adicional que ahora forma parte del aplicativo, tal como se muestra en la Figura 41.

Figura 41. *Aplicativo generado con ChatGPT con respecto a la información requerida del tercer prompt.*



Se muestran satisfactoriamente los datos poblacionales de información ingresada anteriormente para el registro de mascotas en el aplicativo, se observa en la Figura 42 los datos pertinentes con respecto a cuantas mascotas hay registradas, que tanto de cada especie, genero, raza y dirección.

Figura 42. Código generado del tercer prompt por ChatGPT, integrado a Visual Studio Code.



Uno de los puntos importantes dentro de las pruebas realizadas en cada una de las herramientas seleccionadas para el análisis práctico ha sido la repetición de cada uno de los procesos; es decir utilizando los mismos prompts, se ha ingresado por comandos de voz para comprobar su efectividad y confirmar la consistencia de los resultados.

Los resultados en ChatGPT al realizar pruebas adicionales con los mismos tres prompts, revelaron resultados prácticamente idénticos a los obtenidos en la primera prueba y no se observaron cambios significativos en la precisión, la eficiencia o la calidad del código generado, lo que sugiere una estabilidad en el rendimiento de esta IA; además que comprobando al ingresar el código generado en Visual Studio Code, la muestra de software también es la misma.

Pruebas en ChatGPT con prompts en inglés

Al realizar pruebas con ChatGPT utilizando los prompts en inglés, los resultados son bastante similares a los obtenidos con los prompts en español, lo que también se confirma al reproducir el código generado en Visual Studio Code, donde se puede verificar que el código resultante sigue siendo coherente y funcional.

La efectividad de los resultados independientemente del idioma de los prompts es un claro indicio de que esta herramienta es capaz de manejar eficazmente tanto el español como el inglés en el proceso de generación de código, y además indica la fiabilidad de ChatGPT en la interpretación

y ejecución de las solicitudes mediante la voz, lo que es una característica de versatilidad en diferentes contextos. Resultado de la ejecución de este prompt se muestra en la Figura 43.

Figura 43. Prueba del segundo prompt en inglés dentro de la herramienta ChatGPT.

Pet Registration System

Register a Pet

Pet Information

Species:

Gender:

Male

Breed:

Date of Birth:

dd/mm/yyyy

Resultados de la evaluación con base a las pruebas realizadas

Tabla 17. Resultados de la evaluación a ChatGPT

	Eficiencia	Precisión	Calidad de código
Análisis	La herramienta entregó resultados velozmente y acorde a lo requerido por voz.	Los resultados entregados en la estructura HTML fueron correctos, el CSS, para la rapidez con la que fue generada, se complementó bien y el resultado JS no tuvo errores, además incluyó la funcionalidad de LocalStorage.	El código generado no tuvo mayores inconvenientes al momento de ejecutarlos, integrándose holísticamente la estructura HTML, con CSS y JS, funcionando exitosamente.
Ponderación	5	5	5

3.8.2 Bard (Gemini)

Para el análisis de funciones de Bard se debe indicar que es un Chatbot perteneciente a la compañía Google, que, a partir del 24 de febrero de 2024 ha sido renombrado como Gemini, cuya característica más relevante para la emisión de ordenes por voz es su diseño integrado con funcionalidad de micrófono, lo que permite la entrada de manera natural y facilita la interacción de comandos.

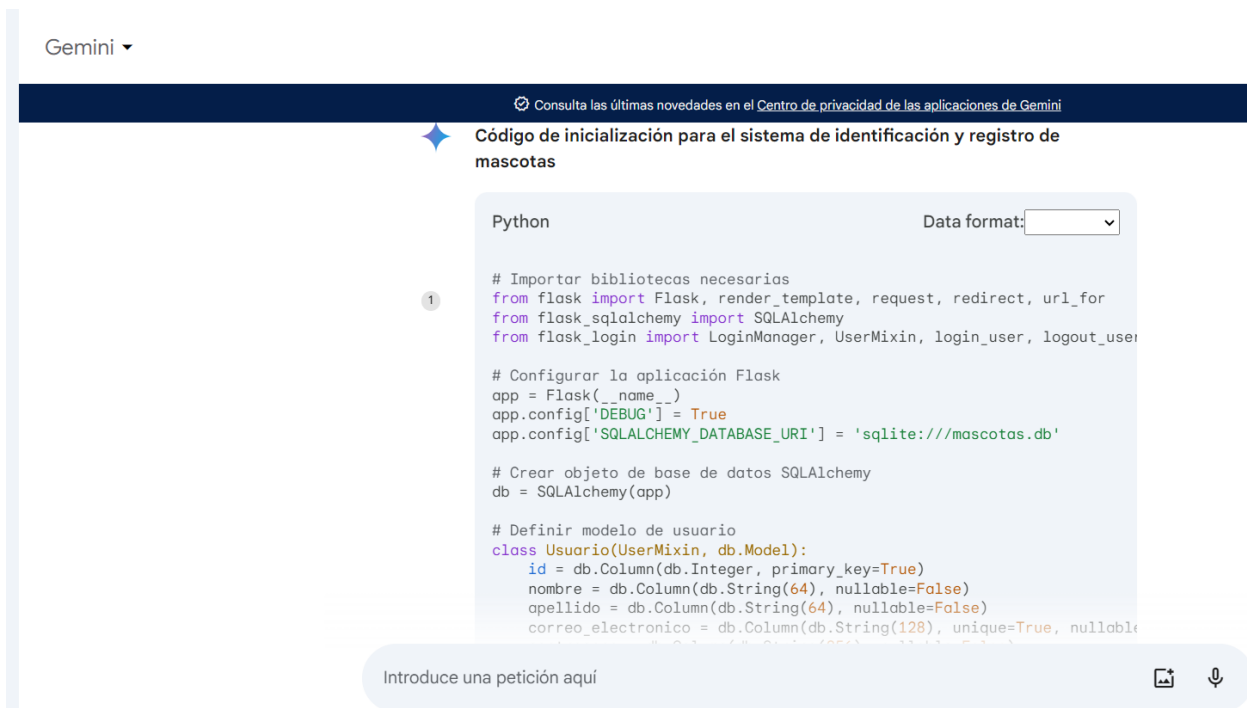
Entonces, la IA, ahora denominada Gemini, tiene integrado el comando de voz, aunque contrario a lo que sucedió en ChatGPT con Talk-to-Mia si bien la herramienta recibe los comandos por audio de voz, no los devuelve de la misma forma, sino que únicamente generar el código o la información solicitada de forma escrita.

Figura 44. Función reconocimiento de voz en Gemini.



Al introducir el **primer prompt**, en Gemini, se ha generado código en Python utilizando las librerías de Flask para la conexión con la base de datos de SQLite, se pudo observar que define los modelos de cada clase y además se maneja con un inicio de sesión de usuario para el acceso a las funciones de registro y consulta de mascotas. Parte del código generado se puede observar en la Figura 45. Cabe aclarar que este código generado no va a ser sobre el que se van a realizar las pruebas, ya que únicamente se utiliza este prompt para iniciar con el contexto de la conversación dentro de esta herramienta IA.

Figura 45. Código generado en Gemini con el primer prompt.



Ingresando el código generado en Gemini al software de demostración se puede verificar que efectivamente se despliegan una serie de campos, como que permiten el registro por nombre, especie, raza, edad, género, foto, propietario, dirección, teléfono y correo electrónico; no obstante se evidencia un error que es los espacios entre campos, que están casi superpuestos y a pesar de Visual Studio Code permite realizar cambios en el código y corregir este aspecto, se mantiene el código original debido a que se está evidenciando los resultados que da esta herramienta IA, tal como se puede ver en la Figura 46. El código generado para el HTML se lo puede evidenciar en la Figura 47.

Figura 46. Resultado del HTML generado por Gemini para el segundo prompt.

Registrar Mascota

Nombre:

Especie:

Raza:

Edad (años):

Género:

Foto:

Propietario:

Dirección:

Teléfono:

Correo electrónico:

Figura 47. Código HTML para el registro de mascotas con Gemini.

```

HTML

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registro Completo de Mascotas</title>
  <link rel="stylesheet" href="css/estilo.css">
</head>
<body>
  <header>
    <h1>Registro Completo de Mascotas</h1>
  </header>

  <main>
    <section class="seccion-formulario">
      <h2>Registrar Mascota</h2>
      <form id="frmRegistrarMascota">
        <h2>Datos de la Mascota</h2>
        <div class="grupo-datos">
          <label for="txtNombreMascota">Nombre:</label>
          <input type="text" id="txtNombreMascota" name="nombreMa
        </div>

        <h2>Datos del Propietario</h2>

        <div class="grupo-datos">
          <label for="txtNombrePropietario">Nombre:</label>
          <input type="text" id="txtNombrePropietario" name="nombrePropietario"
        </div>

        <div class="grupo-datos">
          <label for="txtEdadPropietario">Edad:</label>
          <input type="number" id="txtEdadPropietario" name="edadPropietario"
        </div>

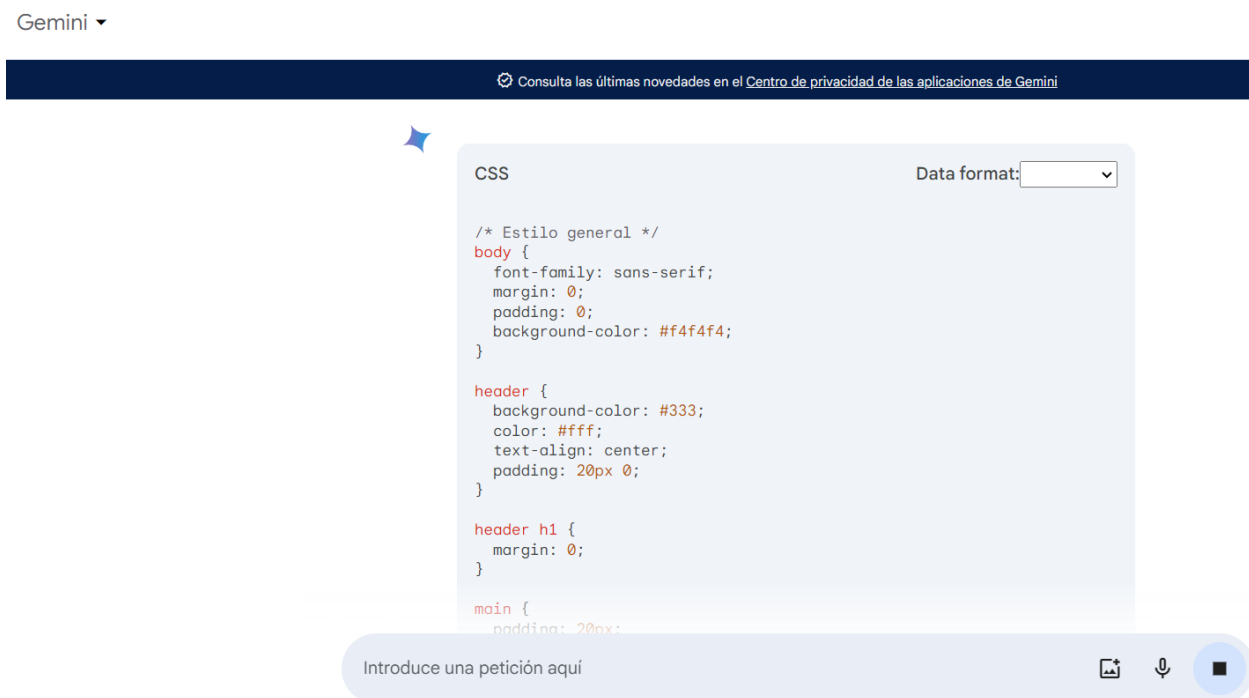
        <div class="grupo-datos">
          <label for="txtDireccionPropietario">Dirección:</label>
          <input type="text" id="txtDireccionPropietario" name="direccionPropietario"
        </div>

        <div class="grupo-datos">
          <label for="txtTelefonoPropietario">Teléfono:</label>
          <input type="tel" id="txtTelefonoPropietario" name="telefonoPropietario"
        </div>

        <div class="grupo-datos">
          <label for="txtCorreoPropietario">Correo electrónico:</label>
          <input type="email" id="txtCorreoPropietario" name="correoPropietario"
        </div>
      </form>
    </section>
  </main>
</body>
</html>

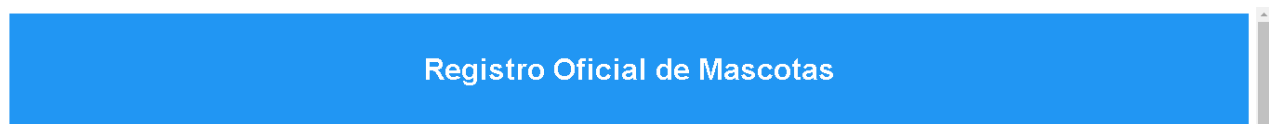
```

Figura 48. Código CSS generado por Gemini para el aplicativo.



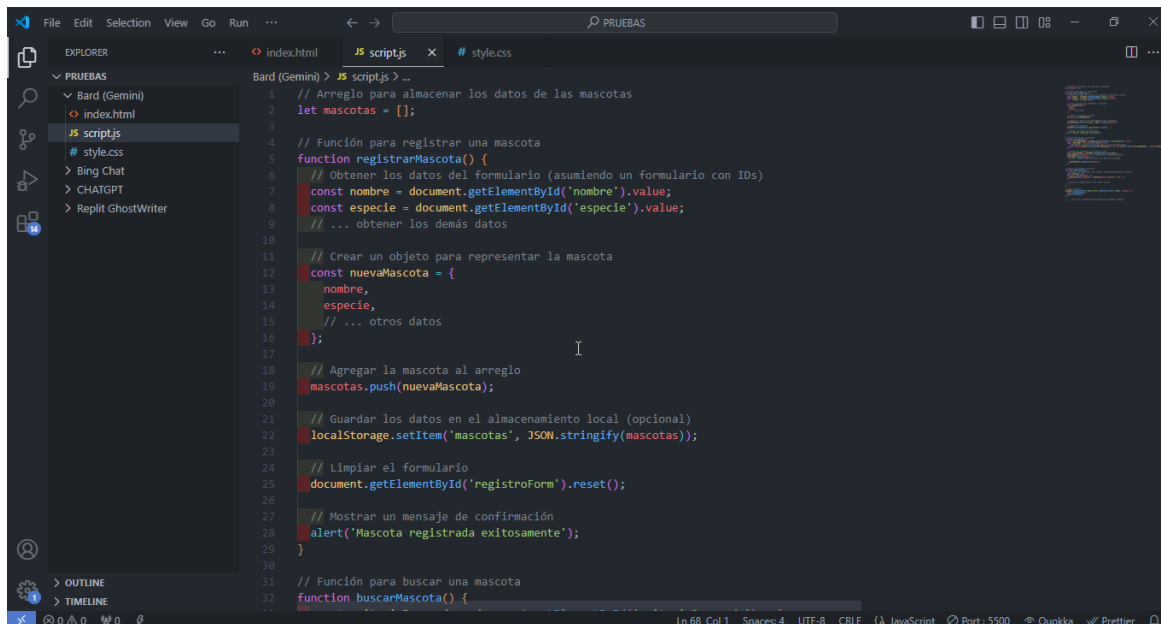
En cuanto al código CSS generado con este prompt, se puede evidenciar en la Figura 49 que es reducido y se enfoca en el desarrollo de código para el diseño de encabezado y que división de secciones se hace visible; se puede evidenciar un color de fondo azul (#2196F3) para el encabezado y un color de texto blanco (#fff), como la separación de bordes y división de campos, las cuales, como se ha mencionado, se pueden editar para tener una mayor o menor separación.

Figura 49. Resultado de aplicar el CSS al aplicativo generado por Gemini.



Mediante el código JavaScript se agrega funcionalidad interactiva al software, con el que se pretende controlar el flujo de datos entre el usuario y la base de datos, integrando el LocalStorage, que simula una base de datos y llama a un arreglo donde se almacenarían la información de las mascotas. Esto se puede ver en la Figura 50.

Figura 50. JavaScript generado con Gemini, integrado a Visual Studio Code.



```
1 // Arreglo para almacenar los datos de las mascotas
2 let mascotas = [];
3
4 // Función para registrar una mascota
5 function registrarMascota() {
6   // Obtener los datos del formulario (asumiendo un formulario con IDs)
7   const nombre = document.getElementById('nombre').value;
8   const especie = document.getElementById('especie').value;
9   // ... obtener los demás datos
10
11   // Crear un objeto para representar la mascota
12   const nuevaMascota = {
13     nombre,
14     especie,
15     // ... otros datos
16   };
17
18   // Agregar la mascota al arreglo
19   mascotas.push(nuevaMascota);
20
21   // Guardar los datos en el almacenamiento local (opcional)
22   localStorage.setItem('mascotas', JSON.stringify(mascotas));
23
24   // Limpiar el formulario
25   document.getElementById('registroForm').reset();
26
27   // Mostrar un mensaje de confirmación
28   alert('Mascota registrada exitosamente');
29 }
30
31 // Función para buscar una mascota
32 function buscarMascota() {
```

La funcionalidad que se reconoce en el código generado, captura el evento submit del formulario de registro, extrae la información de los campos del formulario y se crea un nuevo objeto mascota, para aumentar y actualizar la tabla, dejando limpio el formulario tras el ingreso.

Al emitir la orden para el **segundo prompt** el código HTML se ha actualizado para ampliar sus funcionalidades y mejorar su organización que se muestran en la escritura como secciones y grupos de datos, e inclusive devolviendo código para el color y diseño de cada sección dentro de este lenguaje. Ingresando el código a Visual Studio Code, es notoria que se ha agregado una sección de información del propietario de la mascota, incluyendo campos para el número de persona de referencia, edad, dirección, teléfono y correo electrónico.

Figura 51. Práctica HTML del segundo prompt en Gemini.

Datos de la Mascota

Nombre:

Especie:

Raza:

Género:

Fecha de Nacimiento:

Color:

Peso (kg):

Foto: Ninguno archivo selec.

Código Microchip:

Datos del Propietario

Nombre:

Edad:

Dirección:

Teléfono:

Correo electrónico:

Al introducir el **tercer prompt**, se evidencia que Gemini genera código de forma muy sencilla, únicamente centrada a la petición, lo cual se puede considerar ventajoso, pero también limitante, pues si bien habla de precisión, también habla de una herramienta IA que no tiene función de autoaprendizaje más extensa.

Para la creación de informes, el código muestra una función de <select> que permite desplegar varias opciones para mostrar dichos informes de acuerdo a los solicitado, y que puede ser de uso simplificado para el usuario; además se muestra un elemento div con el id que albergará el contenido del informe generado dinámicamente.

Figura 52. HTML del tercer prompt en Gemini.

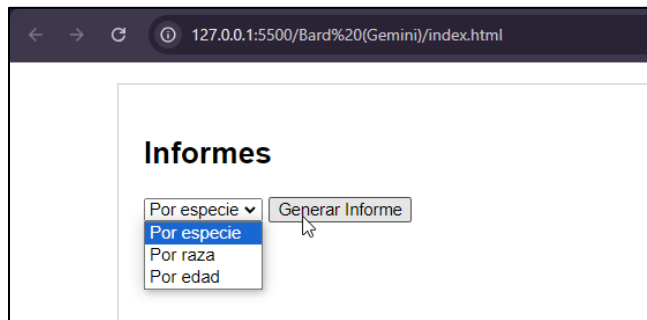
HTML

```
<h2>Población de Mascotas por Especie</h2>

<table>
  <tr>
    <th>Especie</th>
    <th>Cantidad</th>
  </tr>
  <tr>
    <td>Perro</td>
    <td>120</td>
  </tr>
  <tr>
    <td>Gato</td>
    <td>80</td>
  </tr>
  <tr>
    <td>Otro</td>
    <td>30</td>
  </tr>
</table>
```

Este HTML representado simula una interfaz básica para la generación de informes poblacionales de mascotas en Visual Studio Code, que permite al usuario seleccionar el tipo de informe deseado (población por especie, raza, o por edad) y generar el informe con un simple clic en un botón. El contenido del informe, como una tabla de población por especie, se muestra en un área específica.

Figura 53. Prueba HTML del tercer prompt en Gemini.



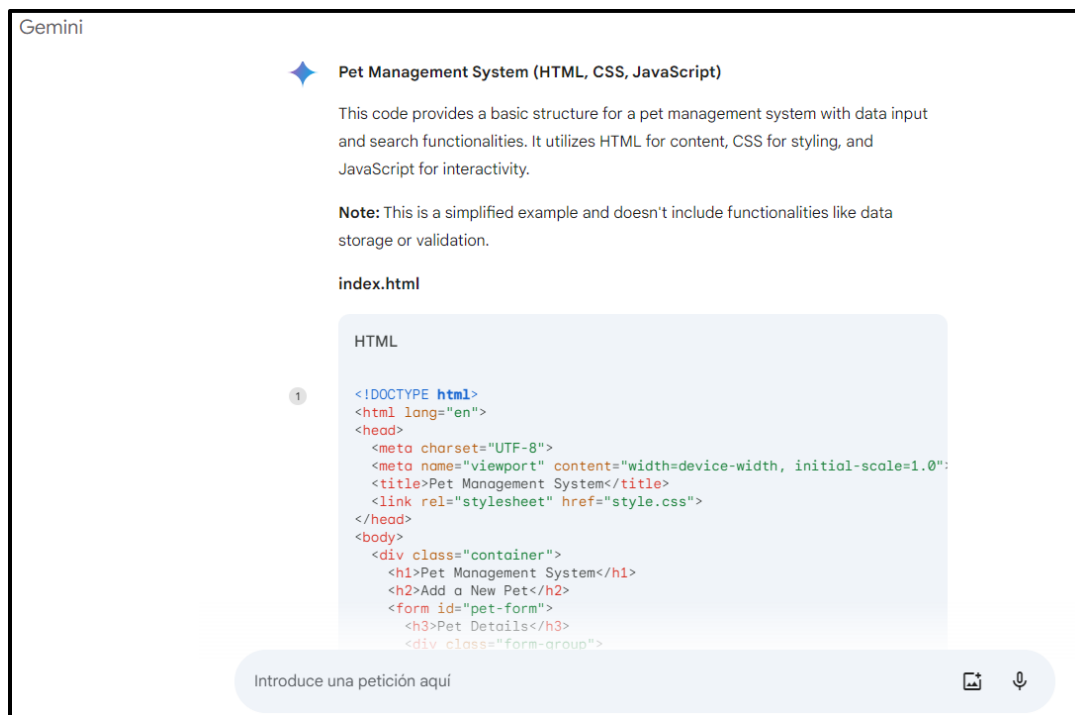
Las pruebas adicionales realizadas con los mismos tres prompts en Gemini mostraron algunas variaciones en los resultados, la herramienta generó código HTML en tres partes, cada una correspondiente a un elemento específico de la interfaz de usuario, se observaron cambios sutiles en la estructura y organización del código en comparación con las pruebas anteriores. Sin embargo, se mantuvo la característica de generar código con una división clara en secciones, facilitando la comprensión y la edición del mismo.

Así mismo en algunas ocasiones, el código generado es más extenso en pruebas diferentes, mientras que en otras ocasiones era un poco más reducido; lo que ocasionaba que pierda funcionalidad, ya que no seguía una línea de comprensión con el código generado con anterioridad, llegando incluso a cambiarlo completamente, lo cual, si el usuario no tiene conocimientos de programación, le resultara complicado generar un código funcional.

Además, se notó que el comando de voz de Gemini es bastante sensible, lo que puede generar transcripciones inexactas cuando existe ruido o falta de claridad en las palabras pronunciadas en la emisión de ordenes por voz. Esta sensibilidad del micrófono se ha traducido en errores de transcripción y, por lo tanto, en la generación de código con errores. Además de esto es necesario repetirlo varias ocasiones que devuelva código completo, o interactuar más tiempo con la herramienta con el fin de tener el código funcional.

Pruebas en Gemini con prompts en inglés

Figura 54. Ejemplo de generación de código en inglés utilizando Gemini.



En las pruebas realizadas con Gemini utilizando prompts, se han encontrado algunos detalles importantes que vale la pena mencionar, la mayoría de las veces, al solicitar la generación de código en HTML, CSS y JavaScript, la IA solo produce código HTML, lo cual es una limitación en la capacidad de la IA para comprender completamente las instrucciones dadas.

No obstante, un aspecto positivo en esta herramienta, es que al dar las ordenes de voz en inglés la comunicación se vuelve más interactiva durante el proceso de generar código, ya que no se obtiene únicamente el código solicitado, sino que la IA también explica, de forma general, lo que ha hecho y ofrece algunas recomendaciones para mejorar o ampliar el código.

Resultados de la evaluación con base a las pruebas realizadas**Tabla 18.** Resultados de la evaluación a Bard (Gemini)

	Eficiencia	Precisión	Calidad de código
Análisis	La herramienta entregó resultados con rapidez, sin embargo, presenta errores en la comprensión de lo dictado por voz, generando pequeños errores.	Los resultados entregados fueron apropiados, con el detalle que, en cada nuevo requerimiento, iba adaptando (agregando o quitando) el código, reflejando limitaciones y carencias en autoaprendizaje.	El código generado al momento de su ejecución no representó fallas como tal, aunque solo podía registrar las mascotas por nombre y especie, únicamente.
Ponderación	3	3	2

3.8.3 Bing Chat

Desde un principio, Bing tiene diferencias bastante palpables a comparación de ChatGPT y Gemini, ya que es una herramienta IA centrada específicamente en la búsqueda desde bases datos, por lo que su configuración y funciones están diseñadas para satisfacer las necesidades específicas de este tipo de usuarios.

Una de las diferencias clave es que Bing Chat, desde el momento del registro se simplifica, y solicita a los usuarios que seleccionen su área de interés con los que les gustaría trabajar, aspecto que permite una experiencia más personalizada y adaptada a las necesidades y preferencias individuales de cada usuario.

En lo que se refiere a generación de código, esta herramienta trata las entradas de cada usuario como solicitudes que se pueden responder desde workbooks, lo que significa que ofrece la posibilidad de utilizar prompts predefinidos para iniciar un proyecto de desarrollo que no sea tan específico, brindando un punto de partida para una tarea, y haciendo desde el inicio una experiencia óptima. Los comandos de voz funcionan de manera similar a Gemini, es decir, su interfaz ya trae un micrófono incorporado.

Al introducir el primer prompt en Bing Chat, se observa que el código generado es bastante simple y limitado, y no despliega funciones propias de IA siendo más detallada, el código generado se limitaba a lo básico solicitado, ofreciendo únicamente el nombre y un campo de entrada para registrar datos, lo cual puede ser insuficiente para satisfacer las necesidades de un proyecto de desarrollo web más complejo, o para un principiante en desarrollo.

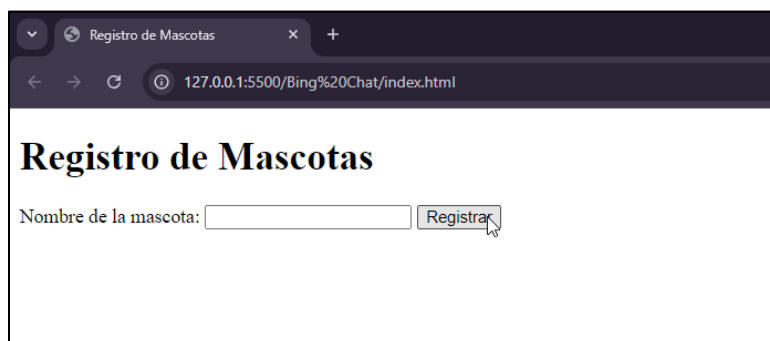
Figura 55. HTML del primer prompt en Bing.

```
HTML

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registro de Mascotas</title>
  <link rel="stylesheet" href="css/styles.css">
</head>
<body>
  <h1>Registro de Mascotas</h1>
  <form id="registro-form">
    <label for="nombre">Nombre de la mascota:</label>
    <input type="text" id="nombre" required>
    <!-- Agrega más campos según tus necesidades (raza, edad, etc.) -->
    <button type="submit">Registrar</button>
  </form>

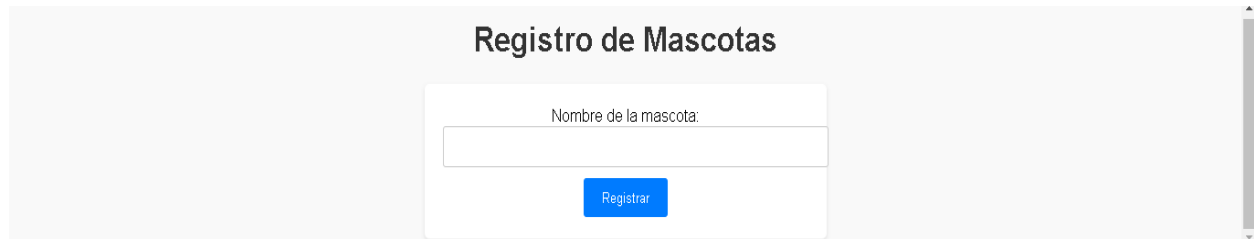
  <script src="js/main.js"></script>
```

Al copiar el código generado por Bing Chat en Visual Studio Code, se confirmó que la salida era efectivamente limitada, ya que generó un único campo, que sirve para ingresar el nombre de la mascota y un botón de registro; no obstante, cabe señalar, que la rapidez de reproducción del código es adecuada, y su apego con un prompt, que talvez resulta incompleto, también es parte de las limitaciones que se han ocasionado.

Figura 56. Prueba HTML del primer prompt en Bing

Por su parte, el código CSS, es funcional con lo que corresponde, pues aporta diseño al interfaz producido con HTML, dando un aspecto preferencial de ubicación, al centro, acertado con la separación de campos, y con el tamaño y colores de letras y formas.

Figura 57. *Interfaz de registro de mascotas con Bing.*

The image shows a web interface for registering a pet. At the top, the title "Registro de Mascotas" is centered in a bold, dark font. Below the title is a light gray rectangular box containing a form. Inside the box, there is a text input field with the placeholder text "Nombre de la mascota:". Below the input field is a blue button with the word "Registrar" in white text. The entire form is centered on a light gray background.

De forma similar sucede con el código JavaScript, que se escribe para una buena comunicación entre el usuario, el formulario y la base de datos, asegurando que los datos se almacenen correctamente y que el usuario reciba una respuesta adecuada; por lo que, a más de permitir el registro, también disminuye los errores que se pueden presentar dentro del software.

Al introducir el **segundo prompt** en Bing Chat, se generó un código reducido, aunque el prompt código solicita secciones para verificar ciertos aspectos, como la existencia de campos para nombre, edad, raza, etc., sobre la mascota, se observó que solo se generaba un campo en cada sección en lugar de varios apartados.

Este resultado demuestra que esta herramienta puede requerir de prompts más concretos, e inclusive otra forma de introducir las solicitudes para desarrollar una funcionalidad más amplia y completa, se puede decir que Bing no está cumpliendo con los requisitos mínimos del software que se intenta reproducir.

Figura 58. HTML del segundo prompt en Bing.

```
HTML

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registro de Mascotas</title>
  <link rel="stylesheet" href="css/styles.css">
</head>
<body>
  <h1>Registro de Mascotas</h1>
  <form id="registro-form">
    <!-- Datos de la mascota -->
    <label for="nombre">Nombre de la mascota:</label>
    <input type="text" id="nombre" required>
    <!-- Agrega más campos según tus necesidades (especie, género, raza, etc.) -->
    <!-- Datos del propietario -->
    <label for="propietario">Nombre del propietario:</label>
    <input type="text" id="propietario" required>
    <!-- Agrega más campos según tus necesidades (edad, dirección, teléfono, correo, etc.) -->

    <!-- Información médica -->
    <label for="medico">Médico tratante:</label>
    <input type="text" id="medico">
    <!-- Agrega más campos según tus necesidades (código de ficha, vacunas, desparasitaciones, etc.) -->
  </form>
</body>
```

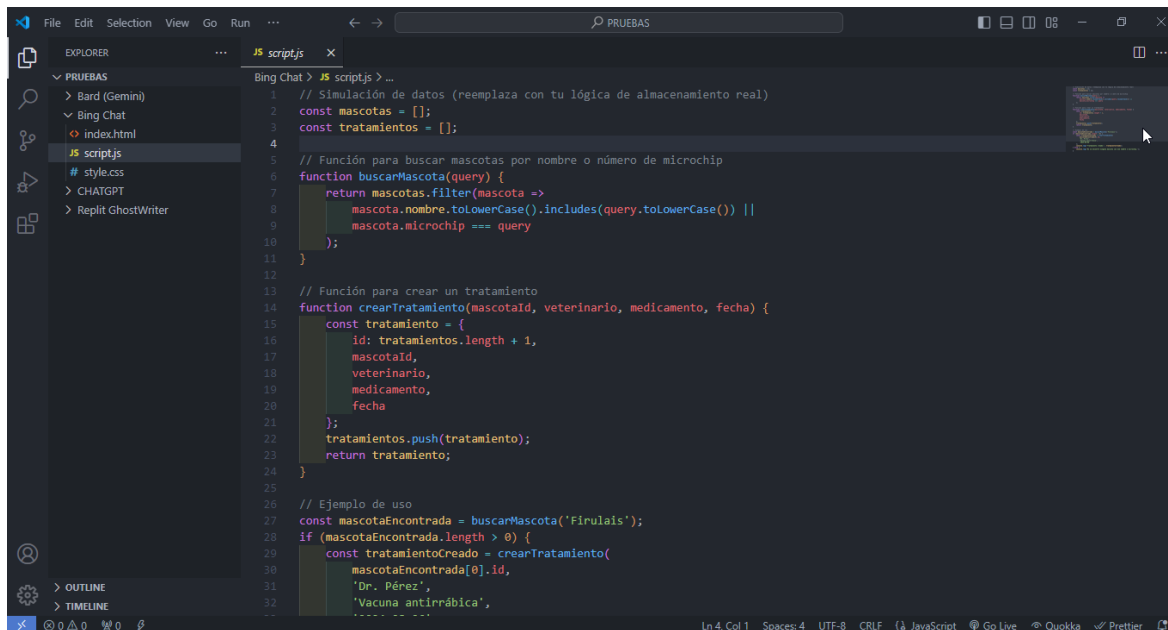
Figura 59. Prueba HTML del segundo prompt en Bing.

Registro de Mascotas

Nombre de la mascota: Nombre del propietario: Médico tratante:

Con el segundo prompt, se agragaron dos campos más: nombre del propietario y medico tratante; Bing chat presenta codigos sencillos y limitados, pero a justa medida, funcionales para lo especifico que comprende de lo dictado.

Figura 60. JavaScript del segundo prompt en Bing, integrado a Visual Studio Code.



El tercer prompt en Bing Chat, no ha generado campos de entrada, sino únicamente títulos o secciones, por lo tanto, sería necesario introducir entradas de registro ya con un software utilizable para comprobar la funcionalidad de las estadísticas poblacionales según lo requerido.

Figura 61. HTML del tercer prompt en Bing.



En las pruebas posteriores realizadas con Bing Chat, se observaron también algunas diferencias en comparación con las primeras, y esto se debe a que Bing se destaca por su enfoque específico en la búsqueda desde bases de datos, lo que se refleja en su configuración y funciones diseñadas para satisfacer necesidades particulares, y trata las entradas como solicitudes que pueden responderse desde workbooks, y por ese motivo, hay veces que no ofrece los tres tipos de lenguaje solicitado.

El código generado por Bing Chat en la primera prueba fue simple pero efectivo, ofreciendo lo solicitado, como el nombre y un campo de entrada para registrar datos y al introducir este código en Visual Studio Code, se confirmó su diseño; pero en la segunda prueba y otras pruebas realizadas, el código generado no cumplió completamente con los requisitos del solicitados en el prompt. Además, en algunas pruebas, a pesar de solicitar código en HTML, CSS y JavaScript, la IA generó solo en HTML, y en algunas ocasiones también en CSS, pero no en JavaScript, lo que demuestra limitaciones en su capacidad para cumplir con las solicitudes de manera completa.

Pruebas en Bing con prompts en inglés

Al realizar pruebas en idioma inglés en Bing las diferencias comparación con las pruebas anteriores en español, al igual que en Gemini, se ha notado que, a pesar de indicar que la solicitud sea para generar código en HTML, CSS y JavaScript, la IA solo genera código en HTML, siendo necesario repetir o precisar la orden para obtener el código en los otros lenguajes.

En tanto que, también Bing muestra una interacción más efectiva, incluso mejor que Gemini, pues ofrece recomendaciones más precisas, ya que además de proporcionar el código solicitado, la IA también menciona recomendaciones adicionales, como considerar el uso de un marco como React para el frontend y Node.js con Express para el backend, sugerencias que pueden ser útiles para desarrollar funciones más complejas.

Figura 62. Pruebas en inglés realizadas con la herramienta Bing.**Figura 63.** Interfaz de usuario generada con prompts en inglés dentro de Bing Chat.**Pet Data Input**

Pet Name:

Owner Name:

Submit

Search Pets

Search by pet name, microchip, or owner's name

Search

Resultados de la evaluación con base a las pruebas realizadas**Tabla 19.** Resultados de la evaluación a Bing Chat

	Eficiencia	Precisión	Calidad de código
Análisis	La herramienta entregó resultados muy limitados, ya que funciona más como un buscador inteligente, lo cual entrega resultados simples.	Los resultados entregados no fueron precisos per se, carecieron en gran parte de lo requerido, entregando, incluso, mayormente, no pudo generar código JS.	El código generado fue muy limitado, y si funcionó, pero no cumplía más de la mitad de los requerimientos solicitados.
Ponderación	2	1	1

3.8.4 Replit GhostWriter

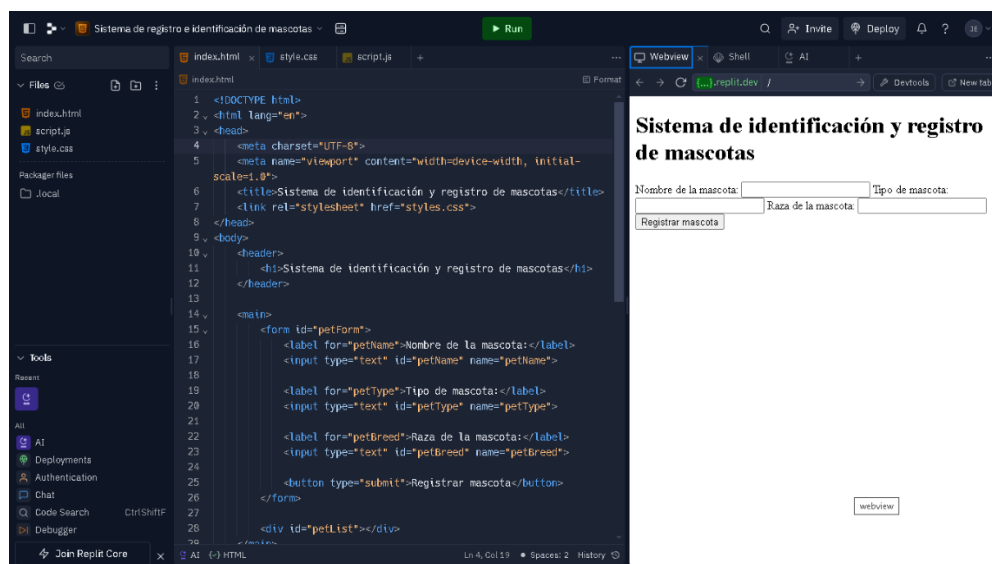
Lo que diferencia a Replit GhostWriter de ChatGPT, Gemini y Bing es su funcionalidad, debido a que no es un chatbot en sí mismo, sino más bien una herramienta IA creada específicamente para la generación de código y la programación, por lo que su configuración y funciones están diseñadas para tareas de desarrolladores y programadores.

Una de las diferencias clave es que Replit GhostWriter, desde el momento del registro, solicita a los usuarios que seleccionen su nivel de conocimiento en programación y el tipo de lenguaje con el que les gustaría trabajar, lo que da una experiencia más personalizada y adaptada a las preferencias de cada usuario.

Otro factor, es que esta herramienta trata las entradas de los usuarios como proyectos, lo que significa que ofrece la posibilidad de utilizar plantillas y modelos predefinidos para iniciar un proyecto de desarrollo web, inclusive proyectos creados por otros usuarios a los cuales se les puede modificar para un proyecto propio. También, un elemento relevante de este web es que tiene integrado el interfaz de reproducción del código por lo que no es necesario exportarlo a Visual Studio Code, sino que únicamente presionar el botón de *run*, para que se genere automáticamente el webview.

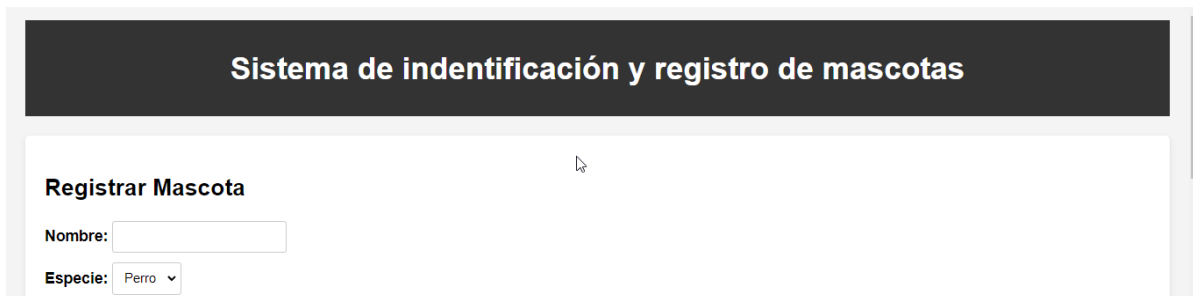
En ese sentido, mediante el **primer prompt**, se obtiene un código base que la IA permite ir modificando de acuerdo a las necesidades del usuario; pero ya con el ingreso de orden por comando de voz se genera una base de programación, como que en la vista previa contiene tres secciones para generar una entrada, con su mando de registro, una vez escritos los datos.

Figura 64. HTML del primer prompt en Replit.



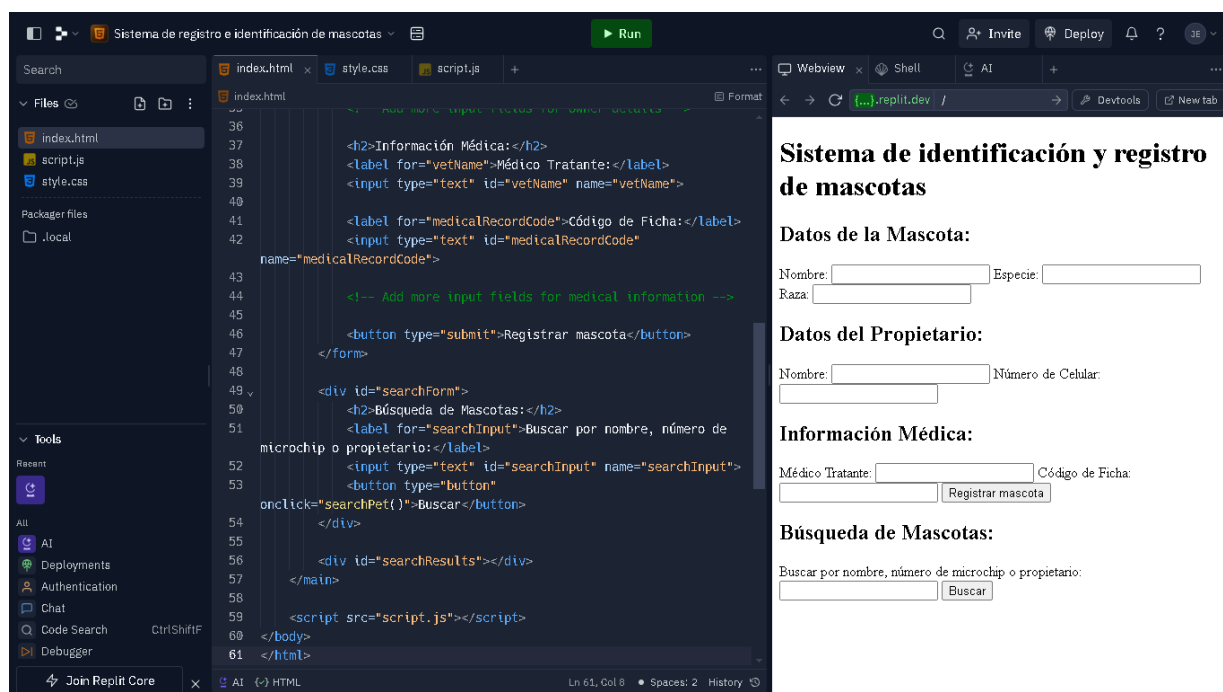
Cuando al HTML se le agrega el código CSS y JavaScript, se revela una mejora del diseño que se traduce en una ficha más ordenada, atractiva y funcional.

Figura 65. Prueba CSS y JavaScript del primer prompt en Replit.



Para el **segundo prompt** el código se amplía a secciones como venía sucediendo con las herramientas anteriores, y de igual forma en la vista previa se genera secciones campos y detalles que con las funciones que presenta esta IA puede transformarse en un software usable para el propósito que se pretende.

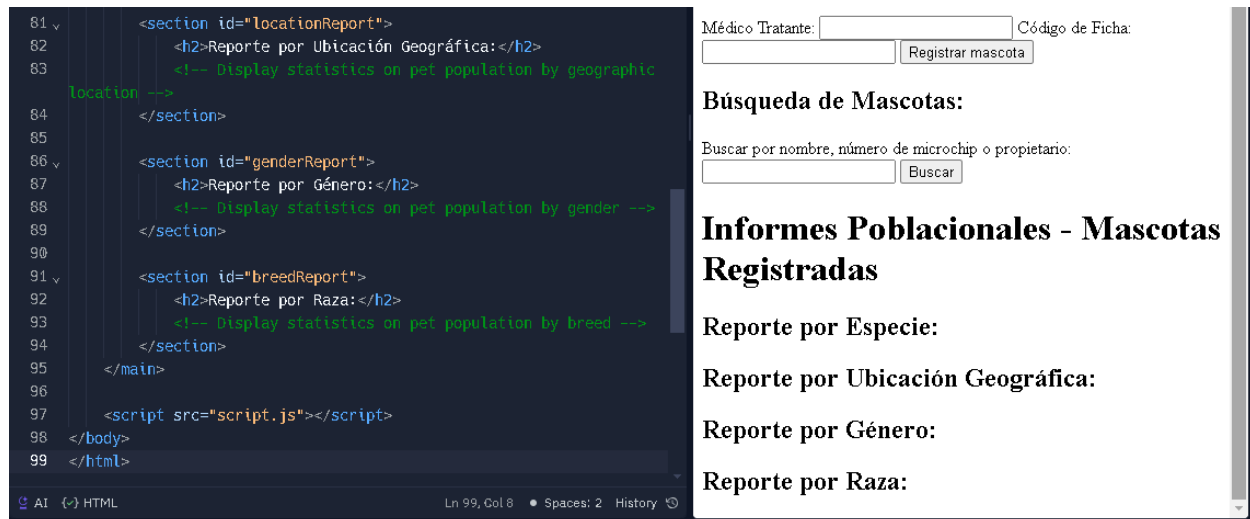
Figura 66. Segundo prompt en Replit.



Con el **tercer prompt**, el código muestra una estructura muy diferenciada, que es una característica de esta IA, y que facilita el trabajo, y que con el **run** da origen a 4 apartados de

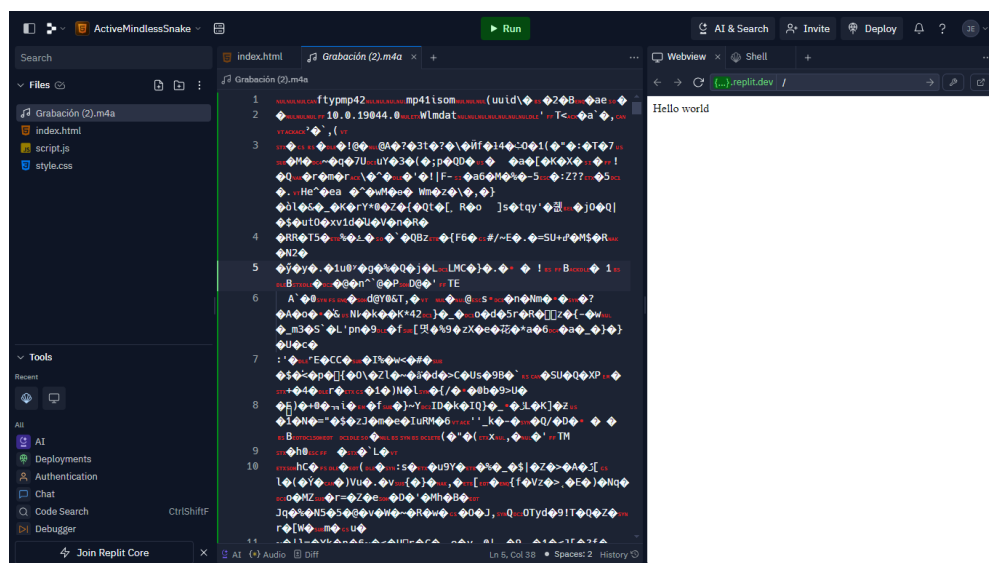
informe que son presentados al mismo tiempo, y que pueden permitir actualizarse al momento de generar nuevas entradas.

Figura 67. Tercer prompt en Replit.



Con Replit se identificó un problema con el comando de voz, pues a diferencia de otras herramientas, donde los comandos de voz funcionan de manera fluida, en esta se encontró que el reconocimiento de voz clonada, por donde se ingresa la ordenes, no siempre es preciso, lo que puede resultar en la generación de errores y en la falta de reconocimiento de algunas órdenes; problema que limita la funcionalidad de la herramienta y dificulta su uso para esta investigación; pues cuando funciona, el código generado es óptimo.

Figura 68. Resultados (Errores) en pruebas posteriores en Replit Ghostwriter.



Pruebas en Replit Ghostwriter con prompts en inglés

Las pruebas que se hicieron en Replit Ghostwriter con los comandos de voz en inglés no salieron como esperábamos; de la misma manera que con los prompts en español hubo algunos problemas con el reconocimiento de voz que todavía está en desarrollo, por ahora, los resultados obtenidos no reflejan completamente el potencial de esta herramienta; podemos considerar que cuando el reconocimiento de voz está integrado en esta IA, seguramente se convertirá en una de las más óptimas del mercado para la generación de código.

Resultados de la evaluación con base a las pruebas realizadas

Tabla 20. Resultados de la evaluación a Replit Ghostwriter

	Eficiencia	Precisión	Calidad de código
Análisis	La herramienta entregó resultados eficientemente, con un detalle que en ocasiones no comprendía correctamente lo dictado por voz, generando errores.	Siendo una IA orientada a programadores, los resultados fueron apropiados y se generaron cumpliendo los requerimientos solicitados.	El código generado al momento de ejecución, funciona de manera adecuada, teniendo en cuenta que el código está dirigido como complemento para usuarios que sepan programación.
Ponderación	2	4	4

3.9 Análisis e Interpretación de resultados

Para efectuar el análisis, primero se debe tomar en cuenta un ejemplo del código funcional y completo que cumple con los requerimientos y características planteadas, para de esta manera, establecer una interpretación de los resultados ya expuestos en el apartado 3.8 *Resultados de pruebas*.

La estructura HTML debe incluir las características planteadas que básicamente son los formularios para los registros pertinentes y las áreas para búsqueda y presentación de resultados de la búsqueda.

Código HTML:

```
<!DOCTYPE html>

<html lang="es">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Registro de Mascotas</title>

  <link rel="stylesheet" href="styles.css">

</head>

<body>

  <header>

    <h1>Registro de Mascotas</h1>

  </header>

  <main>

    <section id="registro">

      <h2>Registrar Mascota</h2>

      <form id="form-mascota">

        <div>

          <label for="nombre">Nombre:</label>
```

```
<input type="text" id="nombre" name="nombre" required>
</div>
<div>
  <label for="especie">Especie:</label>
  <select id="especie" name="especie" required>
    <option value="Perro">Perro</option>
    <option value="Gato">Gato</option>
    <option value="Otro">Otro</option>
  </select>
</div>
<div>
  <label for="genero">Género:</label>
  <select id="genero" name="genero" required>
    <option value="Macho">Macho</option>
    <option value="Hembra">Hembra</option>
  </select>
</div>
<div>
  <label for="raza">Raza:</label>
  <input type="text" id="raza" name="raza" required>
</div>
<div>
  <label for="fechaNacimiento">Fecha de Nacimiento:</label>
  <input type="date" id="fechaNacimiento" name="fechaNacimiento" required>
</div>
```

```
<div>

  <label for="color">Color:</label>

  <input type="text" id="color" name="color" required>

</div>

<div>

  <label for="peso">Peso (Kg):</label>

  <input type="number" id="peso" name="peso" required>

</div>

<div>

  <label for="foto">Foto:</label>

  <input type="file" id="foto" name="foto" accept="image/*">

</div>

<div>

  <label for="chip">Código del Chip:</label>

  <input type="text" id="chip" name="chip" required>

</div>

<h2>Datos del Propietario</h2>

<div>

  <label for="nombrePropietario">Nombre:</label>

  <input type="text" id="nombrePropietario" name="nombrePropietario" required>

</div>

<div>

  <label for="edadPropietario">Edad:</label>

  <input type="number" id="edadPropietario" name="edadPropietario" required>

</div>
```

```
<div>

  <label for="direccionPropietario">Dirección:</label>

  <input type="text" id="direccionPropietario" name="direccionPropietario" required>

</div>

<div>

  <label for="celularPropietario">Celular:</label>

  <input type="tel" id="celularPropietario" name="celularPropietario" required>

</div>

<div>

  <label for="correoPropietario">Correo Electrónico:</label>

  <input type="email" id="correoPropietario" name="correoPropietario" required>

</div>

<div>

  <label for="referenciaPropietario">Persona de Referencia:</label>

  <input type="text" id="referenciaPropietario" name="referenciaPropietario"
required>

</div>

<h2>Información Médica</h2>

<div>

  <label for="medico">Médico Tratante:</label>

  <input type="text" id="medico" name="medico" required>

</div>

<div>

  <label for="ficha">Código de Ficha:</label>

  <input type="text" id="ficha" name="ficha" required>

</div>
```



```
<div>

  <label for="vacunas">Vacunas:</label>

  <textarea id="vacunas" name="vacunas" rows="4" placeholder="Lista de
vacunas"></textarea>

</div>

<div>

  <label for="desparasitaciones">Desparasitaciones:</label>

  <textarea id="desparasitaciones" name="desparasitaciones" rows="4"
placeholder="Lista de desparasitaciones"></textarea>

</div>

<div>

  <label for="tratamientos">Tratamientos:</label>

  <textarea id="tratamientos" name="tratamientos" rows="4" placeholder="Lista de
tratamientos"></textarea>

</div>

<button type="submit">Registrar Mascota</button>

</form>

</section>

<section id="busqueda">

  <h2>Buscar Mascota</h2>

  <form id="form-busqueda">

    <div>

      <label for="busquedaInput">Buscar por nombre, chip o propietario:</label>

      <input type="text" id="busquedaInput" name="busquedaInput" required>

    </div>

    <button type="submit">Buscar</button>

  </form>

</section>
```

```
</form>

<div id="resultadosBusqueda">

</div>

</section>

<section id="informes">

  <h2>Informes Poblacionales</h2>

  <button id="btn-reporteEspecie">Especies</button>

  <button id="btn-reporteDireccion">Direcciones</button>

  <button id="btn-reporteGenero">Género</button>

  <button id="btn-reporteRaza">Raza</button>

  <div id="resultadosInformes">

    </div>

  </section>

</main>

<script src="script.js"></script>

</body>

</html>
```

Con esta estructura, se agregan los estilos CSS, que aportan en mejorar la apariencia, donde se busca un diseño minimalista, limpio y fácil de comprender para el usuario.

Código CSS:

```
body {

  font-family: sans-serif;

  margin: 0;

  padding: 20px;

  background-color: #f4f4f4;

}
```

```
header {  
    background-color: #333;  
    color: #fff;  
    text-align: center;  
    padding: 10px 0;  
}  
  
main {  
    display: flex;  
    flex-direction: column;  
    gap: 20px;  
    margin-top: 20px;  
}  
  
section {  
    background-color: #fff;  
    padding: 20px;  
    border-radius: 5px;  
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);  
}  
  
form {  
    display: flex;  
    flex-direction: column;  
    gap: 10px;  
}  
  
label {  
    font-weight: bold;
```

```
}  
  
input, select, textarea {  
    padding: 8px;  
    border: 1px solid #ccc;  
    border-radius: 3px;  
}  
  
button {  
    padding: 10px 15px;  
    background-color: #4CAF50;  
    color: #fff;  
    border: none;  
    border-radius: 3px;  
    cursor: pointer;  
}  
  
#resultadosBusqueda, #resultadosInformes {  
    margin-top: 10px;  
}  
  
#resultadosBusqueda ul, #resultadosInformes ul {  
    list-style: none;  
    padding: 0;  
}  
  
#resultadosBusqueda li, #resultadosInformes li {  
    margin-bottom: 5px;  
    border: 1px solid #ccc;  
    padding: 10px;
```

```
border-radius: 3px;  
background-color: #f9f9f9;  
}
```

Finalmente, para que su funcionalidad este completa, se contempla el código JavaScript que haga funcionar la página.

Código JS:

```
const formMascota = document.getElementById("form-mascota");  
const formBusqueda = document.getElementById("form-busqueda");  
const resultadosBusqueda = document.getElementById("resultadosBusqueda");  
const btnReporteEspecie = document.getElementById("btn-reporteEspecie");  
const btnReporteDireccion = document.getElementById("btn-reporteDireccion");  
const btnReporteGenero = document.getElementById("btn-reporteGenero");  
const btnReporteRaza = document.getElementById("btn-reporteRaza");  
const resultadosInformes = document.getElementById("resultadosInformes");  
let mascotas = [];  
function guardarMascota(event) {  
    event.preventDefault();  
    const nombre = document.getElementById("nombre").value;  
    const especie = document.getElementById("especie").value;  
    const genero = document.getElementById("genero").value;  
    const raza = document.getElementById("raza").value;  
    const fechaNacimiento = document.getElementById("fechaNacimiento").value;  
    const color = document.getElementById("color").value;  
    const peso = document.getElementById("peso").value;  
    const foto = document.getElementById("foto").files[0]; // Obtener el archivo de imagen  
    const chip = document.getElementById("chip").value;
```

```
const nombrePropietario = document.getElementById("nombrePropietario").value;  
const edadPropietario = document.getElementById("edadPropietario").value;  
const direccionPropietario = document.getElementById("direccionPropietario").value;  
const celularPropietario = document.getElementById("celularPropietario").value;  
const correoPropietario = document.getElementById("correoPropietario").value;  
const referenciaPropietario = document.getElementById("referenciaPropietario").value;
```

```
const medico = document.getElementById("medico").value;  
const ficha = document.getElementById("ficha").value;  
const vacunas = document.getElementById("vacunas").value;  
const desparasitaciones = document.getElementById("desparasitaciones").value;  
const tratamientos = document.getElementById("tratamientos").value;
```

```
const mascota = {  
  nombre: nombre,  
  especie: especie,  
  genero: genero,  
  raza: raza,  
  fechaNacimiento: fechaNacimiento,  
  color: color,  
  peso: peso,  
  foto: foto,  
  chip: chip,  
  propietario: {
```

```
        nombre: nombrePropietario,
        edad: edadPropietario,
        direccion: direccionPropietario,
        celular: celularPropietario,
        correo: correoPropietario,
        referencia: referenciaPropietario
    },
    informacionMedica: {
        medico: medico,
        ficha: ficha,
        vacunas: vacunas,
        desparasitaciones: desparasitaciones,
        tratamientos: tratamientos
    }
};

mascotas.push(mascota);

formMascota.reset();

alert("Mascota registrada correctamente!");
}

function buscarMascota(event) {
    event.preventDefault();
```

```
const busquedaInput = document.getElementById("busquedaInput").value;

const resultados = [];

mascotas.forEach(mascota => {

    if (mascota.nombre.toLowerCase().includes(busquedaInput.toLowerCase()) ||

        mascota.chip.toLowerCase().includes(busquedaInput.toLowerCase()) ||

        mascota.propietario.nombre.toLowerCase().includes(busquedaInput.toLowerCase())) {

        resultados.push(mascota);

    }

});

mostrarResultadosBusqueda(resultados);

}

function mostrarResultadosBusqueda(resultados) {

    resultadosBusqueda.innerHTML = "";

    if (resultados.length > 0) {

        const lista = document.createElement("ul");

        resultados.forEach(mascota => {

            const item = document.createElement("li");

            item.innerHTML = `

                <h3>${mascota.nombre}</h3>

                <p>Especie: ${mascota.especie}</p>

                <p>Raza: ${mascota.raza}</p>

            `;

            lista.appendChild(item);

        });

        resultadosBusqueda.appendChild(lista);

    }

}
```



```
        <p>Chip: ${mascota.chip}</p>

        <p>Propietario: ${mascota.propietario.nombre}</p>
    `;

    lista.appendChild(item);

});

resultadosBusqueda.appendChild(lista);

} else {

    resultadosBusqueda.innerHTML = "<p>No se encontraron resultados.</p>";

}

}

function generarInforme(tipoInforme) {

    const resultados = [];

    mascotas.forEach(mascota => {

        switch (tipoInforme) {

            case "especie":

                if (!resultados.includes(mascota.especie)) {

                    resultados.push(mascota.especie);

                }

                break;

            case "direccion":

                if (!resultados.includes(mascota.propietario.direccion)) {

                    resultados.push(mascota.propietario.direccion);

                }

            }
```

```
        break;

    case "genero":

        if (!resultados.includes(mascota.genero)) {

            resultados.push(mascota.genero);

        }

        break;

    case "raza":

        if (!resultados.includes(mascota.raza)) {

            resultados.push(mascota.raza);

        }

        break;

    }

});

mostrarResultadosInformes(resultados, tipoInforme);

}

function mostrarResultadosInformes(resultados, tipoInforme) {

    resultadosInformes.innerHTML = "";

    if (resultados.length > 0) {

        const lista = document.createElement("ul");

        resultados.forEach(resultado => {

            const item = document.createElement("li");

            item.innerHTML = `
```

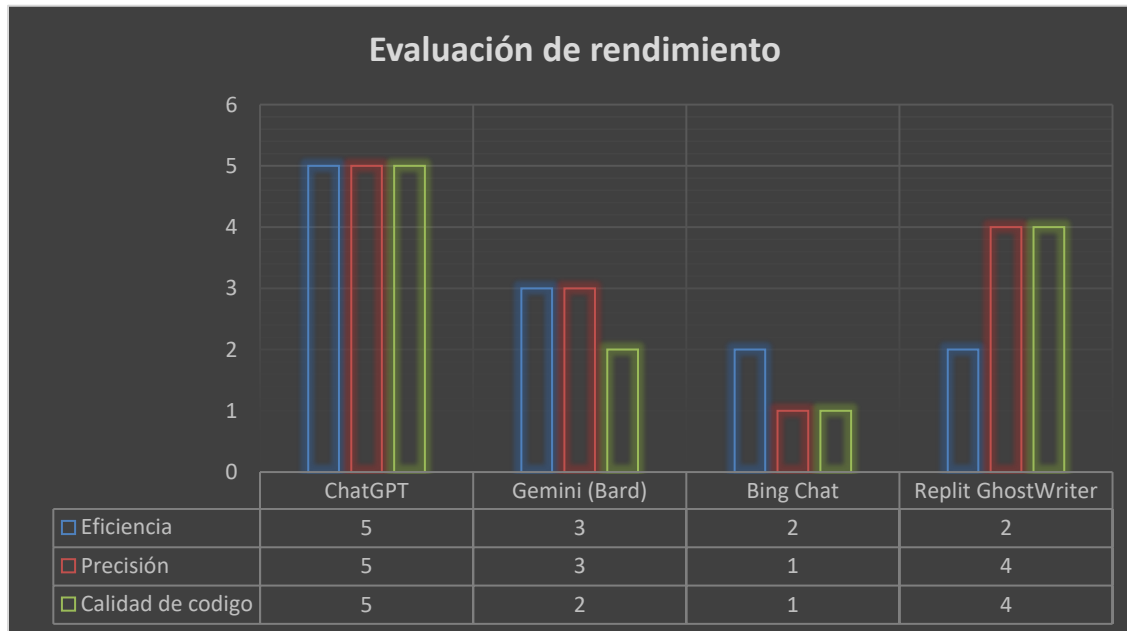
```
        <p>${resultado}</p>
    `;
    lista.appendChild(item);
  });
  resultadosInformes.appendChild(lista);
} else {
  resultadosInformes.innerHTML = "<p>No hay datos disponibles para este informe.</p>";
}
}
```



```
formMascota.addEventListener("submit", guardarMascota);
formBusqueda.addEventListener("submit", buscarMascota);

btnReporteEspecie.addEventListener("click", () => generarInforme("especie"));
btnReporteDireccion.addEventListener("click", () => generarInforme("direccion"));
btnReporteGenero.addEventListener("click", () => generarInforme("genero"));
btnReporteRaza.addEventListener("click", () => generarInforme("raza"));
```

Con una versión del código funcional y los resultados ya expuestos de las herramientas IAs. Partiendo de que la investigación se basa en una metodología de comparación multicriterio, el proceso de toma de decisiones comprende el análisis de resultados mediante una tabla de *desicion making* donde cada columna muestra criterios clave para evaluar las herramientas, se podrá obtener una visión completa de las fortalezas y debilidades de cada herramienta, teniendo con esto un análisis final asertivo y preciso.

Figura 69. Resultados de la evaluación

Primeramente, sobre la interfaz, ChatGPT es principalmente textual, lo que proporciona una experiencia simple y directa, de fácil comunicación con el sistema, y que no está adornada por elementos innecesarios; sin embargo, al haberse integrado un recurso interno para el reconocimiento de voz, se torna un tanto tedioso; no obstante, es un elemento que se requiere añadir y configurar una única vez. En cuanto a la generación de código, como se pudo observar, ChatGPT fue la que entregó el mejor código posible, funcional íntegramente, cumpliendo con las características planteadas, entregando resultados satisfactorios:

Código HTML:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Registro de Mascotas</title>

  <link rel="stylesheet" href="styles.css">
```

```
</head>

<body>

  <div class="container">

    <h1>Registro de Mascotas</h1>

    <!-- Formulario de Registro -->

    <form id="registro-form">

      <h2>Datos de la Mascota</h2>

      <label for="nombreperro">Nombre:</label>

      <input type="text" id="nombreperro" name="nombreperro" required>

      <label for="especie">Especie:</label>

      <input type="text" id="especie" name="especie" required>

      <label for="genero">Género:</label>

      <input type="text" id="genero" name="genero" required>

      <label for="raza">Raza:</label>

      <input type="text" id="raza" name="raza" required>

      <label for="fechaNacimiento">Fecha de Nacimiento:</label>

      <input type="date" id="fechaNacimiento" name="fechaNacimiento" required>

      <label for="color">Color:</label>

      <input type="text" id="color" name="color" required>
```

<label for="peso">Peso:</label>

<input type="number" id="peso" name="peso" step="0.01" required>

<label for="foto">Foto:</label>

<input type="file" id="foto" name="foto" accept="image/*">

<label for="codigoChip">Código del Chip:</label>

<input type="text" id="codigoChip" name="codigoChip" required>

<h2>Datos del Propietario</h2>

<label for="nombrePropietario">Nombre:</label>

<input type="text" id="nombrePropietario" name="nombrePropietario" required>

<label for="edadPropietario">Edad:</label>

<input type="number" id="edadPropietario" name="edadPropietario" required>

<label for="direccion">Dirección:</label>

<input type="text" id="direccion" name="direccion" required>

<label for="celular">Número de Celular:</label>

<input type="tel" id="celular" name="celular" required>

<label for="email">Correo Electrónico:</label>

<input type="email" id="email" name="email" required>

<label for="referencia">Número de Persona de Referencia:</label>

<input type="tel" id="referencia" name="referencia" required>

<h2>Información Médica</h2>

<label for="medico">Médico Tratante:</label>

<input type="text" id="medico" name="medico" required>

<label for="codigoFicha">Código de Ficha:</label>

<input type="text" id="codigoFicha" name="codigoFicha" required>

<label for="vacunas">Vacunas:</label>

<input type="text" id="vacunas" name="vacunas" required>

<label for="desparasitaciones">Desparasitaciones:</label>

<input type="text" id="desparasitaciones" name="desparasitaciones" required>

<label for="tratamientos">Tratamientos:</label>

<input type="text" id="tratamientos" name="tratamientos" required>

<button type="submit">Registrar Mascota</button>

</form>

<!-- Búsqueda de Mascotas -->

<h2>Búsqueda de Mascotas</h2>

```
<form id="busqueda-form">

    <label for="busqueda">Buscar por Nombre, Código de Chip o Nombre del
Propietario:</label>

    <input type="text" id="busqueda" name="busqueda" required>

    <button type="submit">Buscar</button>

</form>


<!-- Resultados de la Búsqueda -->

<div id="resultados"></div>


<!-- Informes Poblacionales -->

<h2>Informes Poblacionales</h2>

<button id="generar-informe">Generar Informe</button>

<div id="informe"></div>


</div>

<div class="button-fin">

    <a href="/index.html" class="buttonf">VOLVER INICIO</a>

</div>

<script src="scripts.js"></script>


</body>

</html>
```

Código CSS:


```
body {  
    font-family: Arial, sans-serif;  
    background-color: #f5f5f5;  
    margin: 0;  
    padding: 0;  
}  
  
.container {  
    width: 80%;  
    max-width: 800px;  
    margin: 50px auto;  
    padding: 20px;  
    background-color: #fff;  
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
    border-radius: 8px;  
}  
  
h1, h2 {  
    color: #333;  
}  
  
form {  
    margin-bottom: 20px;  
}
```

```
label {  
    display: block;  
    margin: 10px 0 5px;  
}  
  
input[type="text"], input[type="date"], input[type="number"], input[type="file"], input[type="tel"],  
input[type="email"] {  
    width: 100%;  
    padding: 8px;  
    margin-bottom: 10px;  
    border: 1px solid #ccc;  
    border-radius: 4px;  
}  
  
button {  
    display: inline-block;  
    padding: 10px 15px;  
    background-color: #4CAF50;  
    color: #fff;  
    border: none;  
    border-radius: 4px;  
    cursor: pointer;  
}  
  
button:hover {  
    background-color: #45a049;
```

```
}
```

```
#resultados, #informe {  
    margin-top: 20px;  
    background-color: #f9f9f9;  
    padding: 20px;  
    border: 1px solid #ddd;  
    border-radius: 4px;  
}
```

Código JS:

```
let mascotas = [];
```

```
document.getElementById('registro-form').addEventListener('submit', function (e) {  
    e.preventDefault();
```

```
let mascota = {  
    nombreperro: document.getElementById('nombreperro').value,  
    especie: document.getElementById('especie').value,  
    genero: document.getElementById('genero').value,  
    raza: document.getElementById('raza').value,  
    fechaNacimiento: document.getElementById('fechaNacimiento').value,  
    color: document.getElementById('color').value,  
    peso: document.getElementById('peso').value,  
    foto: document.getElementById('foto').files[0],  
    codigoChip: document.getElementById('codigoChip').value,
```

```
    propietario: {  
      nombre: document.getElementById('nombrePropietario').value,  
      edad: document.getElementById('edadPropietario').value,  
      direccion: document.getElementById('direccion').value,  
      celular: document.getElementById('celular').value,  
      email: document.getElementById('email').value,  
      referencia: document.getElementById('referencia').value,  
    },  
    medico: document.getElementById('medico').value,  
    codigoFicha: document.getElementById('codigoFicha').value,  
    vacunas: document.getElementById('vacunas').value,  
    desparasitaciones: document.getElementById('desparasitaciones').value,  
    tratamientos: document.getElementById('tratamientos').value,  
  };  
  
  mascotas.push(mascota);  
  
  alert('Mascota registrada exitosamente');  
  
  document.getElementById('registro-form').reset();  
});  
  
document.getElementById('busqueda-form').addEventListener('submit', function (e) {  
  e.preventDefault();  
  
  const criterio = document.getElementById('busqueda').value.toLowerCase();  
  
  const resultados = mascotas.filter(mascota =>  
    mascota.codigoChip.toLowerCase().includes(criterio) ||
```

```
    mascota.propietario.nombre.toLowerCase().includes(criterio) ||  
    mascota.nombreperro.toLowerCase().includes(criterio)  
);
```

```
let resultadosHTML = resultados.map(mascota => `  
    <div>  
        <h3>${mascota.propietario.nombre} (${mascota.codigoChip})</h3>  
        <p>Mascota: ${mascota.nombreperro}</p>  
        <p>Especie: ${mascota.especie}</p>  
        <p>Genero: ${mascota.genero}</p>  
        <p>Raza: ${mascota.raza}</p>  
        <p>Color: ${mascota.color}</p>  
    </div>  
`).join("");
```

```
document.getElementById('resultados').innerHTML = resultadosHTML || 'No se encontraron  
resultados';  
});
```

```
document.getElementById('generar-informe').addEventListener('click', function () {  
    let informe = {  
        total: mascotas.length,  
        especies: {},  
        generos: {},  
        razas: {},  
        ubicaciones: {}  
    };  
});
```

```
};
```

```

mascotas.forEach(mascota => {
    informe.especies[mascota.especie] = (informe.especies[mascota.especie] || 0) + 1;
    informe.generos[mascota.genero] = (informe.generos[mascota.genero] || 0) + 1;
    informe.razas[mascota.raza] = (informe.razas[mascota.raza] || 0) + 1;
    informe.ubicaciones[mascota.propietario.direccion] =
(informe.ubicaciones[mascota.propietario.direccion] || 0) + 1;
});

```

```

let informeHTML = `
    <h3>Informe Poblacional</h3>
    <p>Total de Mascotas Registradas: ${informe.total}</p>
    <h4>Distribución por Especie:</h4>
    <ul>${Object.entries(informe.especies).map(([especie, cantidad]) => `<li>${especie}:
    ${cantidad}</li>`).join("")}</ul>
    <h4>Distribución por Género:</h4>
    <ul>${Object.entries(informe.generos).map(([genero, cantidad]) => `<li>${genero}:
    ${cantidad}</li>`).join("")}</ul>
    <h4>Distribución por Raza:</h4>
    <ul>${Object.entries(informe.razas).map(([raza, cantidad]) => `<li>${raza}:
    ${cantidad}</li>`).join("")}</ul>
    <h4>Distribución por Dirección:</h4>
    <ul>${Object.entries(informe.ubicaciones).map(([direccion, cantidad]) => `<li>${direccion}:
    ${cantidad}</li>`).join("")}</ul>
`;

```

```
document.getElementById('informe').innerHTML = informeHTML;  
});
```

El código entregado por ChatGPT, es funcional al 100%, cumpliendo con los requerimientos y características, y claro, partiendo como una solida base en caso de personalizar al gusto del desarrollador.

La herramienta de IA Gemini, al ser también principalmente textual, la interacción de manera fluida y directa, con la ventaja de que su micrófono incorporado hace más natural el uso de comandos por voz. Gemini funciona mas como un asistente propiamente, en donde el desarrollador debe contar con conocimientos de programación y simplemente solicitar ideas y código aislado a Gemini, ya que por si sola, no genera un código integro funcional, el código proporcionado por Gemini por si solo, no funciona al 100%.

Código HMTL:

```
<!DOCTYPE html>  
  
<html>  
  
<head>  
  
  <title>Registro de Mascotas</title>  
  
  <link rel="stylesheet" href="style.css">  
  
</head>  
  
<body>  
  
  <h1>Registro de Mascotas</h1>  
  
  
  
  <div id="registro">  
  
    <form id="registroForm">  
  
      <label for="nombre">Nombre:</label>  
  
      <input type="text" id="nombre" required>  
  
      <label for="especie">Especie:</label>  
  
      <input type="text" id="especie" required>
```

```
<button type="submit">Registrar</button>

</form>

</div>
```

```
<div id="busqueda">

  <label for="criterioBusqueda">Buscar por:</label>

  <input type="text" id="criterioBusqueda">

  <button onclick="buscarMascota()">Buscar</button>

</div>
```

```
<div id="resultados">

</div>
```

```
<div id="informes">

  <h2>Informes</h2>

  <select id="tipoInforme">

    <option value="especie">Por especie</option>

    <option value="raza">Por raza</option>

    <option value="edad">Por edad</option>

  </select>

  <button onclick="generarInforme()">Generar Informe</button>

  <canvas id="myChart"></canvas>

</div>
```



```
<script src="script.js"></script>

</body>

</html>
```

Este código proporcionado por Gemini sirve como una base ejemplo, ya que para que funcione, se debe completar el formulario de mascotas con las demás entradas, y además integrar los otros formularios restantes, como lo son el registro del propietario y el apartado médico.

Código CSS:

```
body {

    font-family: Arial, sans-serif;

    margin: 20px;

    display: flex;

    flex-direction: column;

    align-items: center;

}


form {

    width: 400px;

    padding: 20px;

    border: 1px solid #ccc;

    border-radius: 5px;

}


label {

    font-weight: bold;

}
```

```
#resultados {  
    margin-top: 20px;  
    border: 1px solid #ccc;  
    padding: 10px;  
}  
  
/* Estilos para la sección de informes */  
#informes {  
    margin-top: 20px;  
    padding: 20px;  
    border: 1px solid #ccc;  
}  
  
canvas {  
    width: 100%;  
    height: 400px;  
}
```

En cuanto al código CSS, es bastante limitado y no se integra con la estructura incompleta proporcionada anteriormente.

Código JS:

```
// Arreglo para almacenar los datos de las mascotas  
let mascotas = [];  
  
// Función para registrar una mascota  
function registrarMascota() {  
    // Obtener los datos del formulario (asumiendo un formulario con IDs)  
    const nombre = document.getElementById('nombre').value;
```

```
const especie = document.getElementById('especie').value;

// ... obtener los demás datos

// Crear un objeto para representar la mascota
const nuevaMascota = {
  nombre,
  especie,
  // ... otros datos
};

// Event listeners
document.getElementById('registroForm').addEventListener('submit', (event) => {
  event.preventDefault();
  registrarMascota();
});

// ... otros event listeners para los botones de búsqueda e informes
// ... (resto de tu código)

function generarInforme() {
  const tipoInforme = document.getElementById('tipoInforme').value;
  const datos = obtenerDatosParaInforme(tipoInforme); // Función para filtrar los datos

  // Crear un gráfico con Chart.js
  const ctx = document.getElementById('myChart').getContext('2d');
  const myChart = new Chart(ctx, {
```

```
type: 'bar', // Puedes cambiar el tipo de gráfico (bar, line, pie, etc.)

data: {

  labels: datos.labels,

  datasets: [{

    label: tipoInforme,

    data: datos.data,

    backgroundColor: 'rgba(54, 162, 235, 0.2)',

    borderColor: 'rgba(54, 162, 235, 1)',

    borderWidth: 1

  ]

},

options: {

  scales: {

    y: {

      beginAtZero: true

    }

  }

}

});

}
```

El código generado por Gemini no es funcional íntegramente, el usuario debe tener conocimientos en programación para poder integrar los extractos de código que genera Gemini, al propio código que debería tener el desarrollador.

Bing Chat es otra herramienta que posee una interfaz basada en texto que puede integrarse con otros servicios de Microsoft, lo que permite ampliarse para incluir ciertos botones, menús desplegables, incluyendo entradas de voz. En cuanto a la generación de código, Bing es bastante limitado, funcional mas como un consulta inteligente, donde busca recursos en la web y la

presenta de una manera facilitada. Ofrece un código funcional, pero el usuario deber ser un desarrollador con conocimientos, ya que el código generado no es funcional de manera íntegra, no funciona por sí solo, como en el caso de ChatGPT.

Código HTML:

```
<!DOCTYPE html>

<html lang="es">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Registro de Mascotas</title>

  <link rel="stylesheet" href="stylex.css">

</head>

<body>

  <h1>Registro de Mascotas</h1>

  <form id="petForm">

    <!-- Datos de la mascota -->

    <label for="petName">Nombre de la mascota:</label>

    <input type="text" id="petName" required>

    <!-- Agrega más campos para especie, género, raza, etc. -->

    <!-- Datos del propietario -->

    <label for="ownerName">Nombre del propietario:</label>

    <input type="text" id="ownerName" required>

    <!-- Agrega más campos para edad, dirección, teléfono, etc. -->
```

```
<!-- Información médica -->

<label for="vetName">Médico tratante:</label>

<input type="text" id="vetName">

<!-- Agrega más campos para código de ficha, vacunas, etc. -->

<!-- Foto de la mascota

<label for="petPhoto">Foto de la mascota:</label>

<input type="file" id="petPhoto">-->

<!-- Botón para enviar el formulario -->

<button class="submitBtn" type="submit">Registrar </button>

</form>

<!-- JavaScript para la búsqueda y estadísticas -->

<script src="script.js"></script>

</body>

</html>
```

Como se puede apreciar, el código generado por Bing, es limitado, aunque funciona muy bien como base, en donde el desarrollador debe completar el código, que en este caso, son ampliar los datos de entrada de los formularios, tanto del registro de mascotas, del propietario y la parte médica.

Código CSS:

```
/* Estilos generales */

body {

    font-family: Arial, sans-serif;

    background-color: #f9f9f9;

    margin: 0;

    padding: 0;

}
```

```
.app {  
    max-width: 600px;  
    margin: 0 auto;  
    padding: 20px;  
    background-color: #ffffff;  
    border-radius: 8px;  
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}
```

```
/* Estilos para la tabla */
```

```
table {  
    width: 100%;  
    border-collapse: collapse;  
    margin-top: 20px;  
}
```

```
th, td {  
    padding: 10px;  
    text-align: left;  
    border-bottom: 1px solid #ddd;  
}
```

```
th {  
    background-color: #f5f5f5;
```

```
}
```

```
/* Estilos para el campo de búsqueda */
```

```
#searchInput {  
    width: 100%;  
    padding: 10px;  
    margin-bottom: 20px;  
    border: 1px solid #ccc;  
    border-radius: 4px;  
}
```

```
/* Estilos para el botón de envío */
```

```
.submitBtn {  
    display: block;  
    margin-left: auto;  
    padding: 15px 30px;  
    border: none;  
    background-color: purple;  
    color: white;  
    border-radius: 6px;  
    cursor: pointer;  
    font-size: 16px;  
    margin-top: 30px;  
}
```



```
.submitBtn:hover {  
  background-color: #9867C5;  
  transform: translateY(-2px);  
}
```

El código CSS presentado contempla solo estilos generales, lo cual lo vuelve incompleto, pero funciona como una base para que un desarrollador con conocimientos.

Código JS:

```
// Simulación de datos (reemplaza con tu lógica de almacenamiento real)
```

```
const mascotas = [];
```

```
const tratamientos = [];
```

```
// Función para buscar mascotas por nombre o número de microchip
```

```
function buscarMascota(query) {  
  return mascotas.filter(mascota =>  
    mascota.nombre.toLowerCase().includes(query.toLowerCase()) ||  
    mascota.microchip === query  
  );  
}
```

```
// Función para crear un tratamiento
```

```
function crearTratamiento(mascotaId, veterinario, medicamento, fecha) {  
  const tratamiento = {  
    id: tratamientos.length + 1,  
    mascotaId,  
    veterinario,  
    medicamento,  
    fecha
```

```
        fecha
    };

    tratamientos.push(tratamiento);

    return tratamiento;
}

// Ejemplo de uso

const mascotaEncontrada = buscarMascota('Firulais');

if (mascotaEncontrada.length > 0) {
    const tratamientoCreado = crearTratamiento(
        mascotaEncontrada[0].id,
        'Dr. Pérez',
        'Vacuna antirrábica',
        '2024-08-06'
    );

    console.log('Tratamiento creado:', tratamientoCreado);
} else {
    console.log('No se encontró ninguna mascota con ese nombre o microchip.');
```

El código generado es bastante limitado, y la mejor manera de aprovechar la herramienta Bing Chat, sería con consultas de código aislado, ya que no cuenta con la capacidad de asociar los códigos entre sí, carece de la capacidad de recordar el código generado anteriormente y asociarlo con un nuevo requerimiento.

Replit GhostWriter por su parte es más rica en funcionalidades en comparación con las otras herramientas, que centra netamente en la programación, resultando más complicada al momento de utilizar. El código generado por esta herramienta si es funcional de manera íntegra, siendo su desventaja simplemente la precisión de comprender lo dictado por voz, lo cual lo vuelve muy

tedioso y prácticamente aumentando su dificultad, convirtiéndola en una herramienta muy buena en cuando a su motor de generación, pero muy mala en cuanto a el dictado por voz. Sin embargo, después ardua paciencia, si se obtiene un código muy completo y bastante funcional íntegramente.

Código HTML:

```
<!DOCTYPE html>

<html lang="es">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Registro de Mascotas</title>

  <link rel="stylesheet" href="styles.css">

</head>

<body>

  <header>

    <h1>Registro de Mascotas</h1>

  </header>

  <main>

    <section id="registro">

      <h2>Registrar Mascota</h2>

      <form id="form-mascota">

        <!-- Datos de la Mascota -->

        <div>

          <label for="nombre">Nombre:</label>

          <input type="text" id="nombre" name="nombre" required>
```

```
</div>

<div>

  <label for="especie">Especie:</label>

  <select id="especie" name="especie" required>

    <option value="Perro">Perro</option>

    <option value="Gato">Gato</option>

    <option value="Otro">Otro</option>

  </select>

</div>

<div>

  <label for="genero">Género:</label>

  <select id="genero" name="genero" required>

    <option value="Macho">Macho</option>

    <option value="Hembra">Hembra</option>

  </select>

</div>

<div>

  <label for="raza">Raza:</label>

  <input type="text" id="raza" name="raza" required>

</div>

<div>

  <label for="fechaNacimiento">Fecha de Nacimiento:</label>

  <input type="date" id="fechaNacimiento" name="fechaNacimiento" required>

</div>

<div>
```

```
<label for="color">Color:</label>

<input type="text" id="color" name="color" required>

</div>

<div>

  <label for="peso">Peso (Kg):</label>

  <input type="number" id="peso" name="peso" required>

</div>

<div>

  <label for="foto">Foto:</label>

  <input type="file" id="foto" name="foto" accept="image/*">

</div>

<div>

  <label for="chip">Código del Chip:</label>

  <input type="text" id="chip" name="chip" required>

</div>

<!-- Datos del Propietario -->

<h2>Datos del Propietario</h2>

<div>

  <label for="nombrePropietario">Nombre:</label>

  <input type="text" id="nombrePropietario" name="nombrePropietario" required>

</div>

<div>

  <label for="edadPropietario">Edad:</label>

  <input type="number" id="edadPropietario" name="edadPropietario" required>
```

```
</div>

<div>

    <label for="direccionPropietario">Dirección:</label>

    <input type="text" id="direccionPropietario" name="direccionPropietario" required>

</div>

<div>

    <label for="celularPropietario">Celular:</label>

    <input type="tel" id="celularPropietario" name="celularPropietario" required>

</div>

<div>

    <label for="correoPropietario">Correo Electrónico:</label>

    <input type="email" id="correoPropietario" name="correoPropietario" required>

</div>

<div>

    <label for="referenciaPropietario">Persona de Referencia:</label>

    <input type="text" id="referenciaPropietario" name="referenciaPropietario"
required>

</div>

<!-- Información Médica -->

<h2>Información Médica</h2>

<div>

    <label for="medico">Médico Tratante:</label>

    <input type="text" id="medico" name="medico" required>

</div>

<div>
```

```
<label for="ficha">Código de Ficha:</label>

<input type="text" id="ficha" name="ficha" required>

</div>

<div>

  <label for="vacunas">Vacunas:</label>

  <textarea id="vacunas" name="vacunas" rows="4" placeholder="Lista de
vacunas"></textarea>

</div>

<div>

  <label for="desparasitaciones">Desparasitaciones:</label>

  <textarea id="desparasitaciones" name="desparasitaciones" rows="4"
placeholder="Lista de desparasitaciones"></textarea>

</div>

<div>

  <label for="tratamientos">Tratamientos:</label>

  <textarea id="tratamientos" name="tratamientos" rows="4" placeholder="Lista de
tratamientos"></textarea>

</div>

<button type="submit">Registrar Mascota</button>

</form>

</section>

<section id="busqueda">

  <h2>Buscar Mascota</h2>

  <form id="form-busqueda">
```

```
<div>

  <label for="busquedaInput">Buscar por nombre, chip o propietario:</label>

  <input type="text" id="busquedaInput" name="busquedaInput" required>

</div>

<button type="submit">Buscar</button>

</form>

<div id="resultadosBusqueda">

  <!-- Resultados de la búsqueda -->

</div>

</section>


<section id="informes">

  <h2>Informes Poblacionales</h2>

  <button id="btn-reporteEspecie">Especies</button>

  <button id="btn-reporteDireccion">Direcciones</button>

  <button id="btn-reporteGenero">Género</button>

  <button id="btn-reporteRaza">Raza</button>

  <div id="resultadosInformes">

    <!-- Resultados de los informes -->

  </div>

</section>


</main>


<script src="script.js"></script>
```



```
</body>
```

```
</html>
```

El código HTML presentado es funcional y cumple con los requerimientos.

Código CSS:

```
body {  
    font-family: sans-serif;  
    margin: 0;  
    padding: 20px;  
    background-color: #f4f4f4;  
}
```

```
header {  
    background-color: #333;  
    color: #fff;  
    text-align: center;  
    padding: 10px 0;  
}
```

```
main {  
    display: flex;  
    flex-direction: column;  
    gap: 20px;  
    margin-top: 20px;  
}
```

```
section {  
    background-color: #fff;  
    padding: 20px;  
    border-radius: 5px;  
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);  
}
```

```
form {  
    display: flex;  
    flex-direction: column;  
    gap: 10px;  
}
```

```
label {  
    font-weight: bold;  
}
```

```
input, select, textarea {  
    padding: 8px;  
    border: 1px solid #ccc;  
    border-radius: 3px;  
}
```

```
button {
```

```
padding: 10px 15px;
background-color: #4CAF50;
color: #fff;
border: none;
border-radius: 3px;
cursor: pointer;
}

#resultadosBusqueda, #resultadosInformes {
    margin-top: 10px;
}

#resultadosBusqueda ul, #resultadosInformes ul {
    list-style: none;
    padding: 0;
}

#resultadosBusqueda li, #resultadosInformes li {
    margin-bottom: 5px;
    border: 1px solid #ccc;
    padding: 10px;
    border-radius: 3px;
    background-color: #f9f9f9;
}
```

El código CSS presentado se integra exitosamente con el código HTML generado anteriormente.

Código JS:

```
// Array para almacenar las mascotas registradas
```

```
let mascotas = [];
```

```
// Función para guardar una nueva mascota
```

```
function guardarMascota(event) {
```

```
    event.preventDefault(); // Prevenir el envío del formulario por defecto
```

```
    // Obtener los datos del formulario
```

```
    const nombre = document.getElementById("nombre").value;
```

```
    const especie = document.getElementById("especie").value;
```

```
    const genero = document.getElementById("genero").value;
```

```
    const raza = document.getElementById("raza").value;
```

```
    const fechaNacimiento = document.getElementById("fechaNacimiento").value;
```

```
    const color = document.getElementById("color").value;
```

```
    const peso = document.getElementById("peso").value;
```

```
    const foto = document.getElementById("foto").files[0]; // Obtener el archivo de imagen
```

```
    const chip = document.getElementById("chip").value;
```

```
    // Obtener los datos del propietario
```

```
    const nombrePropietario = document.getElementById("nombrePropietario").value;
```

```
    const edadPropietario = document.getElementById("edadPropietario").value;
```

```
    const direccionPropietario = document.getElementById("direccionPropietario").value;
```

```
    const celularPropietario = document.getElementById("celularPropietario").value;
```

```
    const correoPropietario = document.getElementById("correoPropietario").value;
```

```
const referenciaPropietario = document.getElementById("referenciaPropietario").value;
```

```
// Obtener la información médica
```

```
const medico = document.getElementById("medico").value;
```

```
const ficha = document.getElementById("ficha").value;
```

```
const vacunas = document.getElementById("vacunas").value;
```

```
const desparasitaciones = document.getElementById("desparasitaciones").value;
```

```
const tratamientos = document.getElementById("tratamientos").value;
```

```
// Crear un objeto mascota con los datos
```

```
const mascota = {
```

```
    nombre: nombre,
```

```
    especie: especie,
```

```
    genero: genero,
```

```
    raza: raza,
```

```
    fechaNacimiento: fechaNacimiento,
```

```
    color: color,
```

```
    peso: peso,
```

```
    foto: foto, // Almacena la referencia al archivo de imagen
```

```
    chip: chip,
```

```
    propietario: {
```

```
        nombre: nombrePropietario,
```

```
        edad: edadPropietario,
```

```
        direccion: direccionPropietario,
```

```
        celular: celularPropietario,
```

```
        correo: correoPropietario,
        referencia: referenciaPropietario
    },
    informacionMedica: {
        medico: medico,
        ficha: ficha,
        vacunas: vacunas,
        desparasitaciones: desparasitaciones,
        tratamientos: tratamientos
    }
};

// Agregar la mascota al array
mascotas.push(mascota);

// Limpiar el formulario
formMascota.reset();

// Mostrar un mensaje de éxito
alert("Mascota registrada correctamente!");
}

// Función para buscar mascotas
function buscarMascota(event) {
    event.preventDefault();
```

```
const busquedaInput = document.getElementById("busquedaInput").value;

const resultados = [];

// Buscar por nombre, chip o propietario
mascotas.forEach(mascota => {

  if (mascota.nombre.toLowerCase().includes(busquedaInput.toLowerCase()) ||

    mascota.chip.toLowerCase().includes(busquedaInput.toLowerCase()) ||

    mascota.propietario.nombre.toLowerCase().includes(busquedaInput.toLowerCase())) {

    resultados.push(mascota);

  }

});

// Mostrar los resultados de la búsqueda
mostrarResultadosBusqueda(resultados);

}

// Función para mostrar los resultados de la búsqueda
function mostrarResultadosBusqueda(resultados) {

  resultadosBusqueda.innerHTML = ""; // Limpiar resultados anteriores

  if (resultados.length > 0) {

    const lista = document.createElement("ul");

    resultados.forEach(mascota => {

      const item = document.createElement("li");
```

```
    item.innerHTML = `
        <h3>${mascota.nombre}</h3>
        <p>Especie: ${mascota.especie}</p>
        <p>Raza: ${mascota.raza}</p>
        <p>Chip: ${mascota.chip}</p>
        <p>Propietario: ${mascota.propietario.nombre}</p>
    `;
    lista.appendChild(item);
});
resultadosBusqueda.appendChild(lista);
} else {
    resultadosBusqueda.innerHTML = "<p>No se encontraron resultados.</p>";
}
}
```

// Función para generar informes poblacionales

```
function generarInforme(tipoInforme) {
    const resultados = [];

    // Obtener datos para el informe
    mascotas.forEach(mascota => {
        switch (tipoInforme) {
            case "especie":
                if (!resultados.includes(mascota.especie)) {
                    resultados.push(mascota.especie);
                }
            }
        }
    });
}
```



```
        }  
        break;  
    case "direccion":  
        if (!resultados.includes(mascota.propietario.direccion)) {  
            resultados.push(mascota.propietario.direccion);  
        }  
        break;  
    case "genero":  
        if (!resultados.includes(mascota.genero)) {  
            resultados.push(mascota.genero);  
        }  
        break;  
    case "raza":  
        if (!resultados.includes(mascota.raza)) {  
            resultados.push(mascota.raza);  
        }  
        break;  
    }  
});  
  
    // Mostrar los resultados del informe  
    mostrarResultadosInformes(resultados, tipoInforme);  
}
```

// Función para mostrar los resultados de los informes

```
function mostrarResultadosInformes(resultados, tipoInforme) {  
    resultadosInformes.innerHTML = ""; // Limpiar resultados anteriores  
  
    if (resultados.length > 0) {  
        const lista = document.createElement("ul");  
        resultados.forEach(resultado => {  
            const item = document.createElement("li");  
            item.innerHTML = `  
                <p>${resultado}</p>  
            `;  
            lista.appendChild(item);  
        });  
        resultadosInformes.appendChild(lista);  
    } else {  
        resultadosInformes.innerHTML = "<p>No hay datos disponibles para este informe.</p>";  
    }  
}  
  
// Agregar eventos a los formularios  
formMascota.addEventListener("submit", guardarMascota);  
formBusqueda.addEventListener("submit", buscarMascota);  
  
// Agregar eventos a los botones de informes  
btnReporteEspecie.addEventListener("click", () => generarInforme("especie"));  
btnReporteDireccion.addEventListener("click", () => generarInforme("direccion"));
```

```
btnReporteGenero.addEventListener("click", () => generarInforme("genero"));
```

```
btnReporteRaza.addEventListener("click", () => generarInforme("raza"));
```

El código JS se integra a la perfección con los códigos entregados anteriormente, generando así un código integro y funcional, sin necesidad de que el usuario necesariamente se vea obligado a agregarle algo adicional para que funcione.

En el apartado de la precisión de voz, se identificaron algunos aspectos relevantes; en ChatGPT, la extensión Talk-to-ChatGPT demostró ser muy precisa, con apenas unos pocos errores en los términos durante los tres prompts analizados, demostrando un alto nivel de exactitud; en Gemini existieron algunos desafíos, debido a la configuración sensible del micrófono a menudo replicaba otros términos diferentes a los indicados durante el ingreso de órdenes habladas.

En este criterio, Bing Chat fue el más efectivo, prácticamente no se registraron errores de transcripción durante el proceso, mostrando precisión y fiabilidad en la interpretación de los comandos vocales; a diferencia de Replit GhostWriter presentó donde dado que la IA es una función de escritura de texto es adicional en esta herramienta, hubo que agregarle voz clonada para el reconocimiento y resultaba en varios errores de ortografía y terminología; y la precisión en este formato ha influido de manera negativa en la generación de código.

La utilización para las tres primeras herramientas es prácticamente igual, sencillas y de fácil manejo, lo que permite a los usuarios interactuar con estas herramientas de manera intuitiva y sin dificultades independientemente de su nivel de habilidad técnica, aunque, las respuestas de ChatGPT son más integra, mientras que Gemini y Bing chat, pueden en cada respuesta, generar un código nuevo incompatible con el anterior. Por otro lado, Replit GhostWriter se distingue de las otras herramientas y requiere un mínimo de conocimientos básicos para su manejo, por lo cual para principiantes resulta complejo su uso, pero una vez que los usuarios se familiarizan con la plataforma puede ser una buena herramienta para el desarrollo y programación.

Sobre las funciones, de igual forma, ChatGPT y Gemini comparten un desempeño muy similar, centrando en la creación de conversaciones nuevas, la capacidad de escribir solicitudes (en este caso, por voz) y la posibilidad de corregir en caso de no recibir una respuesta adecuada; y aunque Bing comparte también tiene estas funciones, ofrece características dentro de la misma página. para aspectos como recetas, planificación, diseño y entrenamiento físico, lo que la hace multifuncional.

En el caso de Replit GhostWriter, al estar centrado en la programación, sus funciones se enfocan en aspectos de la generación de código, como búsqueda de proyectos, modelos y plantillas; clasificación de proyectos que el usuario va realizando, y otras funciones relacionadas con el desarrollo y la programación de software.

La compatibilidad de lenguajes de programación demostró que todas las herramientas ofrecen soporte para una amplia variedad de lenguajes, pero Replit GhostWriter se destaca por su enfoque en la adaptabilidad, ya que, a diferencia de las otras herramientas, permite trabajar en ficheros separados para cada lenguaje, lo que facilita la organización y el desarrollo de proyectos que requieren el uso de múltiples lenguajes.

En las pruebas posteriores realizadas con las mismas herramientas, se observó, en general, que los resultados fueron prácticamente consistentes con los obtenidos inicialmente. A pesar de repetir los mismos prompts y condiciones de prueba, ChatGPT se destacó por encima de las otras tres; Gemini si entrega buenos resultados, pero para un desarrollador con conocimientos previos que sepa integrar los resultados, lo mismo sucede con Bing; Replit las inconsistencias que tiene son con la transcripción, se dificulta bastante la comprensión de los comandos de voz, y eso puede resultar frustrante, sin embargo, cuando logra entender correctamente, ofrece resultados bastante buenos. En el anexo 8, se encuentra detallado el despliegue y acceso de estos códigos generados por cada herramienta IAs utilizada (ChatGPT, Bard (Gemini), Bing Chat y Replit GhostWriter), en las cuales, se puede visualizar e interactuar con las páginas, con base a los códigos generados tal y como están presentadas en este apartado.

3.10 Matriz de multicriterio

Tabla 21. Puntuación multicriterio

HERRAMIENTA	INTERFAZ	PRECISIÓN DE VOZ	FACILIDAD DE USO	FUNCIONES	COMPATIBILIDAD DE LENGUAJES
ChatGPT	Simple, Textual, Directa Puntaje: 7	Óptima, Exactitud, Sin errores Puntaje: 8	Intuitivo, Sencillo, Sin dificultades Puntaje: 8	Versátil, Conversacional, Auto-corrección Puntaje: 8	Amplia, Variedad, Múltiples lenguajes Puntaje: 7
Gemini (Bard)	Fluida, Textual, Natural Puntaje: 7	Algunos errores, Sensible, Replicaba otros términos Puntaje: 6	Sencilla, Flexible, Accesible Puntaje: 8	Variada, Multifuncional, Conversacional Puntaje: 6	Amplia, Variedad, Múltiples lenguajes Puntaje: 7
Bing Chat	Integrada, Textual, Flexible Puntaje: 7	Óptima, Fiable, pocos errores de transcripción Puntaje: 7	Intuitivo, Sencillo, Sin complicaciones Puntaje: 7	Multifuncionales, Interactiva, Específica Puntaje: 8	Amplia Variedad, Complementaria Puntaje: 6
Replit GhostWriter	Complejo, Especializado, Funcional Puntaje: 6	Baja, Errores, fallas gramaticales Puntaje: 5	Requiere conocimientos, Complejo, Desafiante Puntaje: 6	Amplias, Programación, requiere conocimientos Puntaje: 8	Adaptativa, Prácticamente soporta todos los lenguajes Puntaje: 8

La puntuación dada en cada herramienta en cada uno de los aspectos, se ha desarrollado a criterio del investigador, no obstante, se sustenta en factores analizados en las pruebas y una medida sobre 10; en el interfaz se ha evaluado la simplicidad, textualidad y dirección; donde se otorga puntaje alto si el interfaz es simple y directo, facilitando la interacción con el sistema; mientras que la puntuación se va reduciendo si son fluidas y textuales, pero tienen algunas limitaciones en cuanto a sensibilidad y confusión al identificar funciones.

La precisión de voz se califica la capacidad de cada herramienta para interpretar y procesar comandos de voz con exactitud, óptima precisión y falta de errores; mientras que los errores en la transcripción de voz y reconocimiento de términos, se refleja en puntuación inferior en comparación con otras herramientas.

Igualmente, en la facilidad de uso se observa lo intuitivo y simple en la utilización, es decir que si han sido diseñadas para ser accesibles y sin complicaciones. Por su parte, funciones corresponde a la variedad y utilidad de funciones ofrecidas por una herramienta; si incluyen conversaciones, corrección y capacidades multifuncionales, alcanzan un puntaje alto, aunque requiera conocimientos técnicos para aprovechar al máximo sus funciones.

En el último criterio, compatibilidad de lenguajes, se evalúa la capacidad de cada herramienta soportar los múltiples lenguajes de programación existentes; pues esa función las hace adaptables a diferentes necesidades de desarrollo de software y muestra su capacidad para trabajar en diferentes proyectos y necesidades del usuario.

3.11 Aplicación método TOPSIS

Conforme a las variables de esta investigación, se asignó una puntuación de importancia a cada uno de los criterios de acuerdo a su relevancia para el tema de estudio, estas puntuaciones se basan en la consideración de cómo cada criterio impacta en el objetivo general y en la importancia relativa que se le otorga en el contexto de la generación de código; por lo tanto, la puntuación asignada a cada criterio es la siguiente:

Interfaz: 0.1 puntos

Precisión de voz: 0.3 puntos

Facilidad de uso: 0.2 puntos

Funciones: 0.3 puntos

Compatibilidad de lenguajes: 0.1 puntos

Estas puntuaciones reflejan la prioridad dada a cada criterio en función de su contribución al desarrollo de código óptimo para su aplicación en el desarrollo de software, y son los valores de referencia que se utiliza para calcular cual es o son las herramientas que han sido más óptimas en el proceso realizado anteriormente.

Tanto la puntuación, como la ponderación de cada uno de los criterios queda de la siguiente manera:

Tabla 22. Ponderación multicriterio (Método TOPSIS)

Ponderación					
Herramientas	Interfaz	Precisión de voz	Facilidad de uso	Funciones	Compatibilidad
ChatGPT	7	8	8	8	7
Gemini	7	6	8	6	7
Bing Chat	7	7	7	8	6
Replit GhostWriter	6	5	6	8	8

Los pesos dado en la ponderación para cada criterio se da de acuerdo a varios factores que corresponde a esta investigación en específico; se asignan mayores pesos a los criterios que están directamente relacionados con los objetivos del estudio, es decir, la precisión de voz y la facilidad de uso se consideran más importantes para evaluar la efectividad de las herramientas en el contexto de la generación de código a partir de requisitos basados en voz.

También, se ponderan más los criterios que tienen un impacto significativo en la calidad del código generado y en la eficiencia del proceso de desarrollo, como, la capacidad de las herramientas para ofrecer funciones avanzadas y la compatibilidad con diversos lenguajes de programación pueden influir en la productividad y versatilidad de los desarrolladores.

Otro factor que se considera y se ha tenido en cuenta son las opiniones y recomendaciones de expertos a los cuales se ha entrevistado en el campo de la IA y la generación de código, así como las experiencias y preferencias de los usuarios que se ha obtenido como parte de la investigación bibliográfica; garantizando que los pesos asignados sean adecuados y representativos.

Ahora bien, para aplicar el método TOPSIS, introducimos los valores en una hoja de Excel, y se procede a realizar el proceso que consta de los siguientes pasos:

Paso 1: Se eleva los valores al cuadrado, se suman todos los valores de cada criterio, y se saca la raíz cuadrada de cada uno de los totales.

Tabla 23. Aplicación Método TOPSIS - Paso 1

Herramientas	Interfaz	Precisión de voz	Facilidad de uso	Funciones	Compatibilidad
ChatGPT	49	64	64	64	49
Gemini	49	36	64	36	49
Bing Chat	49	49	49	64	36
Replit GhostWriter	36	25	36	64	64
TOTAL	183	174	213	228	198
RAÍZ CUADRADA	13,53	13,19	14,59	15,10	14,07

Paso 2: Para la normalización de valores se divide los valores de cada criterio para la raíz cuadrada obtenida de la tabla anterior.

Tabla 24. Aplicación Método TOPSIS - Paso 2

Herramientas	Interfaz	Precisión de voz	Facilidad de uso	Funciones	Compatibilidad
ChatGPT	0,51745	0,60648	0,54815	0,52981	0,49747
Gemini	0,51745	0,45486	0,54815	0,39736	0,49747
Bing Chat	0,51745	0,53067	0,47963	0,52981	0,42640
Replit GhostWriter	0,44353	0,37905	0,41111	0,52981	0,56854

Paso 3: Se multiplica los valores de cada criterio obtenidos en la tabla por los valores de ponderación de importancia de cada criterio, e identificamos el valor más alto y el valor más bajo para cada columna.

Tabla 25. Aplicación Método TOPSIS - Paso 3

Herramientas	Interfaz	Precisión de voz	Facilidad de uso	Funciones	Compatibilidad
ChatGPT	0,0517	0,1819	0,1096	0,1589	0,0497
Gemini	0,0517	0,1365	0,1096	0,1192	0,0497
Bing Chat	0,0517	0,1592	0,0959	0,1589	0,0426
Replit GhostWriter	0,0444	0,1137	0,0822	0,1589	0,0569
A*	0,0517	0,1819	0,1096	0,1589	0,0569
A-	0,0444	0,1137	0,0822	0,1192	0,0426

Paso 4: Se obtiene la distancia a la solución ideal positiva, mediante la potencia al cuadrado de la resta de cada criterio menos el valor más alto; y la distancia a la solución ideal negativa mediante la potencia al cuadrado de la resta de cada criterio menos el valor más bajo.

Fórmula

Distancia a la solución ideal positiva = $(C - A^*)^2$

Tabla 26. Aplicación Método TOPSIS - Paso 4

Herramientas	Interfaz	Precisión de voz	Facilidad de uso	Funciones	Compatibilidad
ChatGPT	0,0000000	0,0000000	0,0000000	0,0000000	0,0000505
Gemini	0,0000000	0,0020690	0,0000000	0,0015789	0,0000505
Bing Chat	0,0000000	0,0005172	0,0001878	0,0000000	0,0002020
Replit GhostWriter	0,0000546	0,0046552	0,0007512	0,0000000	0,0000000

Fórmula

Distancia a la solución ideal negativa = $(C - A)^2$

Tabla 27. Aplicación Método TOPSIS - Paso 4 (1)

Herramientas	Interfaz	Precisión de voz	Facilidad de uso	Funciones	Compatibilidad
ChatGPT	0,00005464	0,004655	0,007511	0,001578	0,0000505
Gemini	0,00005464	0,0005172	0,007511	0,0000	0,0000505
Bing Chat	0,00005464	0,0002069	0,0001877	0,001578	0,0000
Replit GhostWriter	0,0000	0,0000	0,0000	0,001578	0,0002020

Paso 5: Se suma las distancias de solución ideal positiva para cada una de las herramientas, es decir de forma horizontal y se saca la raíz cuadrada de los valores totales; y de igual manera para la tabla de distancias de solución ideal negativa.

Tabla 28. Aplicación Método TOPSIS - Paso 5

Herramientas	Total Distancia A*	Raíz Cuadrada Distancia A*
ChatGPT	0,0000505	0,007106691
Gemini	0,0036984	0,060814619
Bing Chat	0,0009071	0,030117354
Replit GhostWriter	0,0054610	0,073898518

Tabla 29. Aplicación Método TOPSIS - Paso 5 (1)

Herramientas	Total Distancia A-	Raíz Cuadrada Distancia A-
ChatGPT	0,0070904	0,08420477
Gemini	0,0013736	0,037061637

Bing Chat	0,0038904	0,062372679
Replit GhostWriter	0,0017810	0,042201511

Paso 6: Para calcular la proximidad a la solución ideal, se suma la raíz cuadrada de las distancias de solución ideal positiva más las distancias de solución ideal negativa, y su resultado se divide para cada raíz cuadrada de la distancia ideal negativa.

Formula

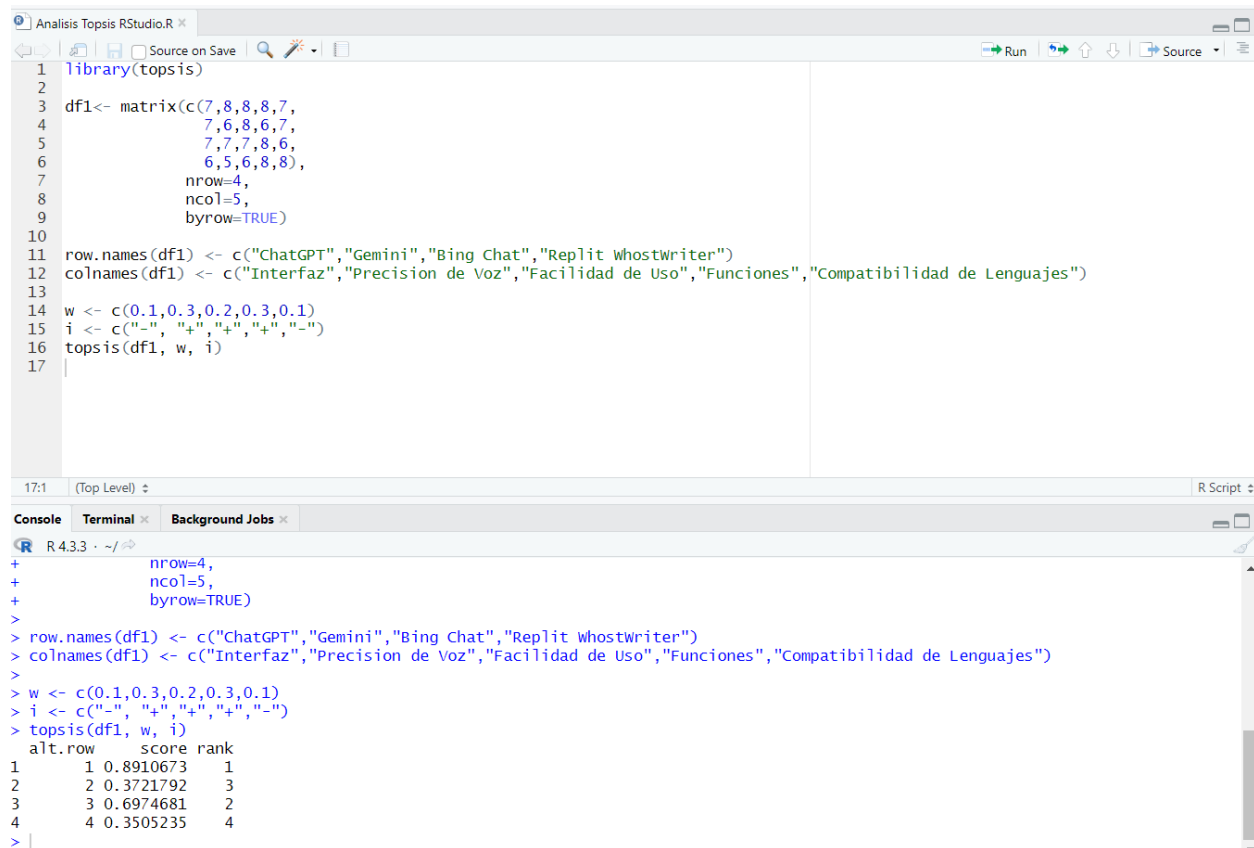
$$\text{Proximidad a la solución ideal } (P_i) = \frac{\sqrt{A^-}}{\sum \sqrt{A^+} + \sum \sqrt{A^-}}$$

Tabla 30. Aplicación Método TOPSIS - Paso 6

Herramientas	Suma Raíz Cuadrada Distancia A* y A-	Aproximación a la solución ideal	Resultados Final
ChatGPT	0,091311461	0,922170881	0,922170881 (IA más óptima)
Gemini	0,097876257	0,378658098	
Bing Chat	0,092490033	0,674371898	
Replit GhostWriter	0,116100029	0,363492686	0,363492686 (IA menos óptima)

Las herramientas IA más óptimas para la generación de código por reconocimiento de voz se pueden verificar de acuerdo a los valores más altos obtenidos de la aproximación a la solución ideal. Es decir, se ha comprobado que ChatGPT es la más óptima, seguido de Bing Chat, en tercer lugar, Gemini, y finalmente como la menos efectiva se identifica a Replit GhostWriter.

Se complementaron los cálculos con el análisis utilizando la herramienta R Studio, donde por falta de precisión los resultados no son exactos, pero el resultado final en cuanto a la jerarquía entre la mejor y peor herramienta comparado con el obtenido en el proceso manual detallado anteriormente, no cambia.

Figura 70. Análisis Topsis realizado utilizando la herramienta R Studio

The screenshot shows the R Studio interface with a script editor and a console. The script editor contains the following R code:

```
1 library(topsis)
2
3 df1<- matrix(c(7,8,8,8,7,
4               7,6,8,6,7,
5               7,7,7,8,6,
6               6,5,6,8,8),
7             nrow=4,
8             ncol=5,
9             byrow=TRUE)
10
11 row.names(df1) <- c("ChatGPT","Gemini","Bing Chat","Replit WhostWriter")
12 colnames(df1) <- c("Interfaz","Precision de Voz","Facilidad de Uso","Funciones","Compatibilidad de Lenguajes")
13
14 w <- c(0.1,0.3,0.2,0.3,0.1)
15 i <- c("-", "+", "+", "+", "-")
16 topsis(df1, w, i)
17
```

The console shows the execution of the code and the resulting Topsis analysis output:

```
R 4.3.3 ~ /
+      nrow=4,
+      ncol=5,
+      byrow=TRUE)
> row.names(df1) <- c("ChatGPT","Gemini","Bing Chat","Replit WhostWriter")
> colnames(df1) <- c("Interfaz","Precision de Voz","Facilidad de Uso","Funciones","Compatibilidad de Lenguajes")
>
> w <- c(0.1,0.3,0.2,0.3,0.1)
> i <- c("-", "+", "+", "+", "-")
> topsis(df1, w, i)
  alt.row  score rank
1      1 0.8910673   1
2      2 0.3721792   3
3      3 0.6974681   2
4      4 0.3505235   4
```

Capítulo 4: Guía de aplicación de herramientas

4.1 Guía de uso de las herramientas utilizadas

La guía de uso tiene la intención de compartir un modelo efectivo que va más allá de su valoración multicriterio, permitiendo obtener provecho a los usuarios desde la experiencia obtenida en esta investigación. Dado que a pesar que en actualidad la tecnología y la generación de código por voz está ganando relevancia, no existen información abundante, clara y precisa sobre su uso.

Además, considerando que el comando por voz, a pesar de ser un elemento enriquecedor, no está integrado de manera óptima en la mayor parte de herramientas, esta guía busca facilitar el uso de las herramientas mencionadas, ya sea acoplando software externo u obteniendo provecho del existente.

4.1.1 Uso de ChatGPT para generar código por reconocimiento de voz

a. Descripción general de la IA

ChatGPT es una herramienta de inteligencia artificial desarrollada por OpenAI que utiliza la arquitectura GPT (Generative Pre-trained Transformer) para generar respuestas de texto coherentes cuya principal función es la generación de texto para mantener conversaciones y responder preguntas, hacer cálculos, traducir idiomas, o que también se puede utilizar para la generación de código de desarrollo de software.

Para el desarrollo de código la IA se basa en modelos de lenguaje entrenados con grandes cantidades de datos textuales que le permite comprender el contexto y generar respuestas que imitan el estilo y la estructura del lenguaje humano, debido a que utiliza un enfoque de aprendizaje automático para mejorar continuamente su capacidad de comprensión y generación de texto.

La interfaz de esta herramienta es principalmente textual, lo que proporciona una experiencia simple y directa para los usuarios, ya sea de forma escrita o a través de la entrada de voz, los usuarios pueden expresar sus requisitos de codificación de manera natural, y ChatGPT generará código en función de estos comandos verbales. Sin embargo, para utilizar la entrada de voz en ChatGPT, los usuarios deben integrar herramientas externas como Talk-to-ChatGPT, que permite la entrada de voz en la plataforma.

b. Incorporación de la extensión Talk-to-Mia

Se ha encontrado en Talk-to-Mia, una solución la introducción de prompts por voz en ChatGPT, ya que precisamente está diseñada para integrarse en el navegador web, haciendo que los usuarios puedan dictar sus requisitos de codificación en voz alta y recibir respuesta, agilizando el proceso con una interacción natural y fluida con el sistema con la eliminación de la necesidad de escribir manualmente una orden.

La extensión Talk-to-Mia es una opción accesible y fácil que traduce instrucciones habladas y que para instalarse requiere de entrar en las extensiones de Google y añadirla al navegador web, activarla, realizar configuraciones básicas y comenzar a hablar los requisitos en voz alta. En desarrollo de software, la extensión procesará la entrada de voz y generará código correspondiente en el área de texto.

Es decir, los pasos para el uso de esta extensión en ChatGPT, son:

- Abrir el navegador web y entrar a la tienda de extensiones.
- Localiza la extensión Talk-to-Mia y agregar al navegador para instalarla.
- En la barra de extensiones del navegador configurar la extensión, para el idioma de entrada y salida.
- La primera vez que se utilice la extensión, autorizar el acceso al micrófono del dispositivo para aceptar la entrada de voz.
- Una vez configurada, la extensión estará lista para recibir comandos de voz para la generación de código.

a) Recomendaciones para generación de código en ChatGPT

Una vez en la plataforma de ChatGPT es importante asegurarse que el reconocimiento de voz de Talk-to-Mia, contenga la configuración más acertada para que utilizando la función de reconocimiento de voz, proporcionar las instrucciones, siendo claro y conciso al expresar las ordenes en voz alta; también verificar que exista un ambiente que no tenga ruidos o interferencias que puedan afectar el proceso.

Una vez que se ha proporcionado los requisitos, la plataforma generará automáticamente el código correspondiente, al que corresponde realizar de lo generado para asegurar de que cumpla con las necesidades del desarrollador, aunque como Talk-to-Mia devuelve la respuesta por audio, se puede escuchar de esta forma, y se es necesario, consultar a la IA si el código generado requiere de mejora, realiza ajustes o correcciones.

Seguidamente, una vez se esté satisfecho con el código generado, se puede copiar o guardar el proyecto con el nombre que considere el usuario, para que cuando se requiera generar código adicional o realizar modificaciones, se repita o amplie el proceso anterior, ya que a medida que nos familiarizamos con la plataforma y sus capacidades se podrá aprovechar al máximo esta herramienta en el desarrollo de software.

4.1.2 Uso de Gemini para generar código por reconocimiento de voz

a. Características y funcionalidad de la IA

Gemini, anteriormente conocido como Google Bard, es un Chatbot desarrollado por Google que ofrece una amplia gama de funciones, que incluye la capacidad de respuesta a partir de instrucciones por voz, debido a que contiene técnicas avanzadas de procesamiento del lenguaje natural y aprendizaje automático para comprender y ejecutar comandos de voz de manera precisa y eficiente.

Además, puede realizar una variedad de tareas, como responder preguntas, realizar búsquedas en la web, cálculos, proporcionar recomendaciones y mucho más; siendo una de sus principales funciones la interacción a través de comandos de voz, lo que la hace especialmente útil para tareas que requieren entrada de voz, como la generación de código.

De igual manera, entre estas características se destaca su capacidad para comprender el contexto de las conversaciones y adaptarse a las necesidades del usuario, lo que significa que puede interpretar comandos complejos y generar código en función de instrucciones detalladas que se le proporcionen.

b. Instrucciones para utilizar la función ordenes por voz

Al tener, Gemini la función de órdenes por voz incorporada dentro de su software, el procedimiento es simple, primero se debe acceder a la plataforma a través de un navegador web mediante el uso de una cuenta de registro, que es gratuita, y, una vez dentro, encontrar la opción para activar el reconocimiento de voz, a la derecha de la barra de escritura para ordenes; esta función puede estar ubicada en la barra de herramientas o en el menú principal de la plataforma, para que activado el reconocimiento de voz, puedes comenzar a pronunciar las órdenes de manera natural.

El procesamiento de instrucciones de Gemini proporciona respuestas a consultas o ejecución de acciones específicas, que puede incluir también la generación de código, dando opción a que, si el usuario no está satisfecho con la respuesta, puedes ajustar las instrucciones y la IA se

adaptará para brindar una mejor respuesta. La función de órdenes por voz en Gemini está diseñada para facilitar la interacción con la plataforma, permitiéndote realizar tareas de manera eficiente y sin necesidad de utilizar el teclado o el mouse.

c. Recomendaciones para generar código mediante comandos de voz en Gemini

De acuerdo a la experiencia obtenida en esta investigación, para garantizar un rendimiento óptimo al generar código mediante comandos de voz en Gemini, se recomienda seguir algunas acciones clave; primero, es importante hablar claramente y de manera articulada al pronunciar los comandos, esto ayudará a que el sistema reconozca con precisión las instrucciones y genere el código de manera efectiva; además, evitar ruidos externos que puedan interferir.

Otra recomendación es utilizar prompts simples y directos, en lugar de dar instrucciones complejas en una sola frase, es preferible dividir las partes más pequeñas y concretas, ya que se facilita así que la IA comprenda mejor y el resultado entregado sea más preciso.

Además, es útil revisar las respuestas proporcionadas por las herramientas para comprobar de que se ajusten a las necesidades. En caso de que la respuesta no sea la esperada, se puede intentar reformular las instrucciones y/o proporcionar más detalles para obtener un resultado más acertado.

Por otro lado, aunque el interfaz de Gemini es de uso fácil, inclusive para una persona que incluso no tenga experiencia previa, no obstante, es importante practicar y familiarizarse con cada una de las funciones de Gemini incluido el uso de comando de voz. Ya que, lógicamente mientras más se utilice más se conocerá capacidades y facilitara la formulación de instrucciones y en la interpretación de las respuestas generadas por la plataforma.

4.1.3 Uso de Bing Chat para generar código por reconocimiento de voz

a. Descripción y funciones de la herramienta

Bing Chat es una herramienta creada por Microsoft, que ofrece una serie de características y funciones, diseñadas para facilitar la interacción con el sistema y ser un asistente virtual de inteligencia artificial eficiente; entre sus principales ventajas, destaca que proporciona un entorno intuitivo y de fácil acceso, permitiendo a los usuarios expresar sus requerimientos a través de comandos de voz y recibir respuestas precisas y útiles sobre una variedad de temas.

Otra de las principales funciones de esta IA es su interfaz texto o voz, que permite a los usuarios comunicarse de manera clara y directa con el sistema, facilitando la formulación de instrucciones

y la comprensión de las respuestas generadas por la plataforma. Además, Bing Chat ofrece la posibilidad de integrarse con otros servicios de Microsoft, lo que amplía su funcionalidad y le permite ofrecer características adicionales, como la creación de botones, menús desplegables y otras interfaces de su compañía desarrolladora.

Además, destaca la capacidad de la herramienta para interpretar comandos de voz de manera precisa y confiable, esto gracias a su sistema de reconocimiento de voz que ya viene incluido en sus funciones, por lo que la plataforma puede entender las instrucciones de los usuarios con una alta tasa de efectividad, y que hace que en desarrollo de software el código generado sea útil para un proyecto.

b. Indicaciones para mejorar la precisión y efectividad al utilizar comandos de voz

De las cuatro herramientas estudiadas con base al multicriterio al utilizar comandos de voz para generar código, Bing Chat y Gemini resultan bastante similares, por lo que igual que la herramienta anterior es importante considerar algunas recomendaciones prácticas que pueden ayudar a optimizar el uso de esta IA; entre las principales se puede mencionar que:

- Se debe utilizar Bing Chat en un entorno tranquilo y libre de ruidos externos, ya que cualquier ruido de fondo puede interferir con el reconocimiento de voz y dificultar la precisión de las instrucciones.
- Dar comandos por voz, asegurándose de hablar de manera clara y pausada, ni demasiado rápido o murmurado, ya que puede entorpecer que el sistema interprete las instrucciones.
- Introducir las instrucciones con frases cortas y simples, hará que se reduzcan los errores de interpretación.
- Repetir y/o reformular de manera diferente las instrucciones; pues las practicas han demostrado que encontrando los prompts adecuados esta IA llega a generar una respuesta más optima.
- Verifica la transcripción de la IA para comprobar una buena interpretación.
- Tomarse el tiempo para practicar y experimentar con la herramienta para mejorar la experiencia de uso.

4.1.4 Uso de Replit GhostWriter para generar código por reconocimiento de voz

a. Definición, características y enfoque en programación

De la muestra seleccionada para la investigación Replit GhostWriter, se puede acotar que es una herramienta diferente a la otras, y a pesar de que es una IA que se enfoca específicamente en la generación de código y la programación, hace de otras de sus características ajena a lo visto en las anteriores, ya que esta creada para ofrecer a los usuarios una forma innovadora de crear y desarrollar software de manera rápida y eficiente.

La capacidad de esta herramienta para generar código de forma autónoma se debe a que utiliza algoritmos de procesamiento de lenguaje natural para comprender el contexto y las necesidades del proyecto en función de las instrucciones y requisitos proporcionados por el usuario. Además de generar código completamente nuevo, también proporciona asistencia en la escritura de código, ya que sugiere fragmentos de código que ayuda a completar automáticamente líneas de código y ofrecer correcciones y mejoras basadas en las prácticas recomendadas y el contexto que presenta un proyecto.

Otra característica destacada de esta IA es su capacidad para adaptarse y mejorar continuamente su rendimiento, ya que sus algoritmos de aprendizaje automático son muy avanzados, haciendo del feedback con los usuarios un procedimiento único en el desarrollo, aprendiendo de los errores y actualizando sus modelos continuamente para ofrecer resultados precisos y útiles con el tiempo.

En su manejo, se ha identificado que está diseñado para ser compatible con una amplia variedad de lenguajes de programación, donde se puede encontrar los más reconocidos en el mercado de la programación, como Python, JavaScript, Java, C++, entre otros, lo que permite a los usuarios utilizar la herramienta en una amplia gama de proyectos y contextos de desarrollo.

Semejante característica es que está integrado directamente en el entorno de desarrollo en línea de Replit, lo que facilita el acceso a proyectos que sirven como modelo de uso por parte de los usuarios, en donde se puede compartir proyectos con otros usuarios e incluso trabajar de forma mancomunada. El principal obstáculo encontrado en esta IA es la necesidad de conocimientos básicos que debe tener un usuario sobre programación, lo que implica que la herramienta no es para todo tipo de beneficiario.

b. Función de comando por voz en Replit GhostWriter

Para acceder a la función de comando por voz en Replit GhostWriter, se pueden emplear dos métodos distintos, cada uno diferente, con su procedimiento individual; el primero trata de la integración de voz clonada y la entrada de audio directamente en la plataforma, para una vez dentro del entorno de desarrollo, el usuario puede activar esta función mediante la configuración y reproducción correspondiente.

Aquí, el proceso implica grabar un archivo de audio que contenga los prompts o requisitos específicos para el código que se desea generar, donde la IA va a interpretar los comandos y generar el código correspondiente, basándose en las órdenes dadas en dicho archivo de audio.

La segunda opción, surge de la misma empresa Replit, la cual ofrece una aplicación llamada Talk to create, diseñada para facilitar la comunicación con la IA mediante comandos de voz, sin embargo, esta aplicación está en periodo de prueba, por lo que algunas funciones podrían estar sujetas a cambios o limitaciones.

Para utilizarla, los usuarios deben buscarla en la plataforma en línea, y luego instalarla en su dispositivo; una vez instalada, basta con activar la función de reconocimiento de voz y comenzar a dar los comandos para generar código inmediato; en esta opción lo que corresponde es esperar y estar al tanto de cuando la compañía lance la versión final, para conocer sobre la disponibilidad y el funcionamiento de dicha aplicación.

4.1.5 Problemas encontrados e indicaciones para solucionarlos

Durante la investigación práctica se pudo evidenciar varias dificultades en la generación de código por comandos de voz utilizando herramientas como ChatGPT, Gemini, Bing y Replit, que pueden causar problemas al inicio de un proyecto de desarrollo y programación. Entre los principales problemas encontrados, se puede enumerar los siguientes, para los que se recomienda algunas y posibles soluciones:

- Dificultad para reconocer comandos de voz correctamente, es decir fallas en la precisión de introducción de prompts; que para solucionarle es importante asegurarse de hablar claramente y en un entorno con poco ruido o probar ajustar la sensibilidad del micrófono en las configuraciones de la herramienta para adaptarlo mejor a las condiciones de grabación; además sería recomendable utilizar un hardware de micrófono externo al dispositivo que cumpla con ciertos aspectos de eliminación de ruido.

- Generación de código incorrecto o incompleto, por lo que es importante verificar que los comandos de voz utilizados sean claros, proporcionando detalles suficientes para que la herramienta comprenda correctamente lo que se requiere, que también incluye no emitir información innecesaria. Además, revisar los prompts y las respuestas generadas para corregir cualquier error o falta de información.
- Falta de compatibilidad con ciertos lenguajes de programación o características específicas, en el caso de Replit GhostWriter, explorar la posibilidad de trabajar con diferentes archivos y configuraciones para ajustarse mejor a las necesidades del proyecto, mientras que, en otras herramientas, como ChatGPT o Gemini, es importante familiarizarse con las capacidades y limitaciones de la herramienta y ajustar los comandos de voz en consecuencia.
- Interfaz poco intuitiva y dificultad para navegar por la herramienta, para la que además de identificar anticipadamente que, si la IA cumple con los requerimientos del proyecto, hay que tomarse el tiempo para explorar las funciones y características de la herramienta, y utilizar recursos como tutoriales o documentación para comprender mejor cómo aprovechar al máximo sus capacidades.
- Tiempo de respuesta lento o intermitente, verificar la conexión a internet y la estabilidad del servidor de la herramienta utilizada. Además, considerar la posibilidad de actualizar la herramienta a la última versión disponible para obtener mejoras de rendimiento y correcciones de errores.

Capítulo 5: Conclusiones y recomendaciones

5.1 Conclusiones

Objetivo general: El objetivo general planteado en este trabajo de titulación es: Desarrollar un análisis multicriterio de herramientas de generación de código basadas en Inteligencia artificial, con el fin de proporcionar una guía para la selección y aplicación efectiva de estas tecnologías en el desarrollo de aplicaciones a partir de requisitos basados en voz.

Conclusiones del objetivo general:

- El desarrollo de un análisis multicriterio para herramientas de generación de código basadas en inteligencia artificial ha permitido una evaluación exhaustiva y equilibrada de diferentes tecnologías disponibles en el mercado. Este análisis se centró en criterios fundamentales como la precisión en la interpretación de requisitos basados en voz, la eficiencia en la generación de código y la calidad del código producido.
- La investigación culminó en la creación de una guía detallada para la selección y aplicación efectiva de herramientas de generación de código basadas en inteligencia artificial. Esta guía incluye recomendaciones claras y prácticas sobre cómo evaluar y elegir las herramientas más adecuadas según los requisitos específicos de cada proyecto.

Objetivos Específicos

Objetivo específico 1: Aplicar técnicas de toma de decisiones basadas en multicriterio para seleccionar un conjunto representativo de herramientas de inteligencia artificial utilizadas para la generación de código a partir de requisitos basados en voz.

Conclusiones del objetivo específico 1:

- La aplicación de técnicas de toma de decisiones basadas en multicriterio demostró ser eficaz para seleccionar un conjunto representativo de herramientas de inteligencia artificial destinadas a la generación de código a partir de requisitos basados en voz.
- Estas técnicas permitieron una evaluación equilibrada, objetiva y fundamentada, garantizando que las herramientas seleccionadas fueran adecuadas para distintos contextos y necesidades de desarrollo de software.
- Este enfoque sistemático ayudó a identificar herramientas que no solo sobresalían en precisión, eficiencia y calidad del código generado, sino también en su capacidad de adaptación a diferentes escenarios de uso y requisitos de proyecto.

Objetivo específico 2: Evaluar el rendimiento a través de caso de estudio denominado “Software para el registro, gestión y tenencia responsable de mascotas en la ciudad de Cuenca”, en las que cada herramienta sea analizada en términos de precisión, eficiencia y calidad del código generado, con el propósito de establecer una clasificación.

Conclusiones del objetivo específico 2:

- El caso de estudio "Software para el registro, gestión y tenencia responsable de mascotas en la ciudad de Cuenca" proporcionó un escenario realista y aplicable para evaluar el rendimiento de las herramientas seleccionadas. A través de esta evaluación, se pudo establecer una clasificación clara basada en la precisión, eficiencia y calidad del código generado.
- Los resultados mostraron variaciones significativas entre las herramientas en términos de su capacidad para interpretar requisitos basados en voz y generar código optimizado. Algunas herramientas destacaron en precisión, mientras que otras mostraron superioridad en eficiencia o calidad del código, siendo la herramienta ChatGPT la más destacada, y resaltando la importancia de elegir la herramienta adecuada según las necesidades específicas del proyecto.
- Se pudo evidenciar que el software generado puede ayudar a enfocarse a los programadores en la correcta estructuración del proyecto, en cuando a componentes como middleware, controladores, interfaces gráficas y conexiones con bases de datos externas; sin embargo por motivos de pruebas de código funcional se optó por generar los 3 archivos (HTML, CSS y JavaScript) necesarios para validar la ejecución de los casos de uso mediante una base de datos local del navegador, ayudando con esto a realizar los análisis correspondientes

Objetivo específico 3: Elaborar una guía metodológica de aplicación de cada herramienta seleccionada, para la generación de código.

Conclusiones del objetivo específico 3:

- Se desarrolló una guía metodológica detallada que describe la aplicación efectiva de cada herramienta seleccionada para la generación de código. Esta guía incluye pasos detallados para la configuración y uso de las herramientas, así como recomendaciones basadas en las evaluaciones realizadas.
- La guía proporciona instrucciones claras sobre cómo maximizar el rendimiento de cada herramienta, incluyendo la integración de requisitos basados en voz y la generación de

código de alta calidad. También se incluyen consejos sobre cómo solucionar errores identificados durante la evaluación.

Englobando las conclusiones, se obtiene que: el análisis multicriterio aplicado en esta tesis ha demostrado ser una estrategia efectiva para seleccionar herramientas de generación de código basadas en IA, evaluando su rendimiento en un caso de estudio práctico y desarrollando una guía metodológica para su aplicación. Estas contribuciones ofrecen una base sólida para la selección y uso eficiente de tecnologías de IA en el desarrollo de aplicaciones a partir de requisitos basados en voz, apoyando así la innovación y la mejora continua en el campo del desarrollo de software.

5.2 Recomendaciones

- Es crucial adoptar metodologías de toma de decisiones multicriterio en otros procesos de selección de herramientas tecnológicas para garantizar una evaluación equilibrada y objetiva, además, las herramientas tecnológicas evolucionan rápidamente, por lo que es recomendable realizar reevaluaciones periódicas utilizando las técnicas multicriterio para asegurar que las herramientas seleccionadas sigan siendo las más adecuadas.
- Se recomienda probar las herramientas en diferentes escenarios y casos de estudio ayudará a entender mejor sus fortalezas y debilidades en diversos contextos, lo que facilitará la selección de la herramienta más adecuada para cada proyecto específico, también es recomendable documentar los resultados de las evaluaciones para fomentar el conocimiento y la mejora continua.
- Se recomienda que las guías deban ser revisadas y actualizadas periódicamente para reflejar las nuevas funcionalidades y mejoras de las herramientas, así como las mejores prácticas emergentes en el campo, junto con la integración de las guías metodológicas como parte integral del proceso de desarrollo de software, asegurando que se sigan de manera consistente y eficiente.
- Se recomienda integrar prácticas de desarrollo ágil en el uso de herramientas de IA para garantizar que los equipos puedan responder rápidamente a cambios y mantener un enfoque iterativo en la mejora continua del código y las funcionalidades desarrolladas.

Referencias

- Adriano, W. G., & Guapi Cuji, D. A. (2023). Influencia del código abierto y su simplicidad en el desarrollo de sistemas web académicos. *Revista Científica y Tecnológica UPSE*, 10(1), 10–18. <https://doi.org/10.26423/rctu.v10i1.730>
- Agrawal, S., Suryawanshi, S., Arsude, V., Maid, N., & Kawarkhe, M. (2020). Factors Involved in Artificial Intelligence-based Automated HTML Code Generation Tool. *Proceedings of the 2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing, ICSIDEMPC 2020*. <https://doi.org/10.1109/ICSIDEMPC49020.2020.9299609>
- Alemán Espinoza, J. W., Ruiz Vargas, M. R., & Aguilar Juárez, E. A. (2019). Retrospectiva al Desarrollo de Chatbots y Procesamiento del Lenguaje Natural. *Realidad y Reflexión*. <https://doi.org/10.5377/ryr.v48i0.7095>
- Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). Programming Is Hard - or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. *SIGCSE 2023 - Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, 1. <https://doi.org/10.1145/3545945.3569759>
- Berryhill, J., Heang, K. K., Clogher, R., & McBride, K. (2020). Hola, Mundo: La Inteligencia Artificial y su uso en el Sector Público. In *OCDE* (Vol. 11, Issue 22).
- Borji, A. (2023). *A Categorical Archive of ChatGPT Failures*. <http://arxiv.org/abs/2302.03494>
- Bravo Amarante, E., Kiran Schulz, R., Romero Romero, O., López Bastida, E. J., Güereca, L. P., Bravo Amarante, E., Kiran Schulz, R., Romero Romero, O., López Bastida, E. J., & Güereca, L. P. (2021). Análisis de decisiones multicriterio en la integración de herramientas de la economía ecológica. *Revista Universidad y Sociedad*, 13(4).
- Cámara, J., Troya, J., Burgueño, L., & Vallecillo, A. (2023). On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML. *Software and Systems Modeling*, 22(3), 781–793. <https://doi.org/10.1007/s10270-023-01105-5>
- Carbonell-García, C. E., Burgos-Goicochea, S., Calderón-de-los-Ríos, D. O., & Paredes-Fernández, O. W. (2023). La Inteligencia Artificial en el contexto de la formación educativa. *EPISTEME KOINONIA*, 6(12). <https://doi.org/10.35381/e.k.v6i12.2547>
- Castro Rojas, L. F., & Espitia Peña, E. (2022). Estado de progreso en el desarrollo de proyectos software apoyado por Semat y QFD. *Ingeniare. Revista Chilena de Ingeniería*, 30(4). <https://doi.org/10.4067/s0718-33052022000400719>
- Celi-Parraga, R. J., Varela-Tapia, E. A., Acosta-Guzmán, I. L., & Montaña-Pulzara, N. R. (2021). Técnicas de procesamiento de lenguaje natural en la inteligencia artificial conversacional textual. *AlfaPublicaciones*, 3(4.1), 40–52. <https://doi.org/10.33262/ap.v3i4.1.123>
- Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., ... Zaremba, W. (2021). *Evaluating Large Language Models Trained on Code*. <http://arxiv.org/abs/2107.03374>
- Corvalán, J. G. (2018). Inteligencia artificial: Retos, desafíos y oportunidades - Prometea: La primera inteligencia artificial de Latinoamérica al servicio de la Justicia. In *Revista de Investigações Constitucionais* (Vol. 5, Issue 1). <https://doi.org/10.5380/rinc.v5i1.55334>

- De Vito, G., Palomba, F., Gravino, C., Di Martino, S., & Ferrucci, F. (2023). ECHO: An Approach to Enhance Use Case Quality Exploiting Large Language Models. *Proceedings - 2023 49th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2023*, 53–60. <https://doi.org/10.1109/SEAA60479.2023.00017>
- Erazo-Luzuriaga, A. F., Ramos-Secaira, F. M., Galarza-Sánchez, P. C., & Boné-Andrade, M. F. (2023). La inteligencia artificial aplicada a la optimización de programas informáticos. *Journal of Economic and Social Science Research*, 3(1). <https://doi.org/10.55813/gaeal/jessr/v3/n1/61>
- Estrada, F., Loor, H., & Viteri, L. (2022). Reemplazo de personal humano por inteligencia artificial: ventajas y desventajas. *Revista Investigación y Negocios*, 15(25).
- Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. (2020). *CodeBERT: A Pre-Trained Model for Programming and Natural Languages*. <http://arxiv.org/abs/2002.08155>
- Figueroa, S. M., & Postigio, M. C. (2023). Mejorando la práctica educativa con inteligencia artificial y NLP (Procesamiento de Lenguaje Natural): Una herramienta valiosa para docentes y estudiantes. *Revista Nuestro NOA*, 0(16).
- Franganillo, J. (2023). La inteligencia artificial generativa y su impacto en la creación de contenidos mediáticos. *Methados Revista de Ciencias Sociales*, 11(2). <https://doi.org/10.17502/mrcs.v11i2.710>
- Gamboa, E. D. (2019). Prototipo De Un Chatbot Para Compras Online Utilizando Bot Framework. *Universidad Internacional de La Rioja*.
- Huang, Y., Chen, Y., Chen, X., Chen, J., Peng, R., Tang, Z., Huang, J., Xu, F., & Zheng, Z. (2024). *Generative Software Engineering*. <http://arxiv.org/abs/2403.02583>
- Larrosa Uribe, D., Verón Marcos, S., Fernández Oliva, P., & Delgado Dapena, M. D. (2019). Ejecución automática de pruebas en entornos empresariales de producción de software. *Informática y Sistemas: Revista de Tecnologías de La Informática y Las Comunicaciones*, 3(1). <https://doi.org/10.33936/isrtic.v3i1.1585>
- Lau, S., & Guo, P. (2023). From “Ban It Till We Understand It” to “Resistance is Futile”: How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. *ICER 2023 - Proceedings of the 2023 ACM Conference on International Computing Education Research V.1*. <https://doi.org/10.1145/3568813.3600138>
- Llanos Mosquera, J. M., Hidalgo Suarez, C. G., & Bucheli Guerrero, V. A. (2021). Una revisión sistemática sobre aula invertida y aprendizaje colaborativo apoyados en inteligencia artificial para el aprendizaje de programación. *Tecnura*, 25(69). <https://doi.org/10.14483/22487638.16934>
- Manjarrés-Betancur, R. A., & Echeverri-Torres, M. M. (2020). Asistente virtual académico utilizando tecnologías cognitivas de procesamiento de lenguaje natural. *Revista Politécnica*, 16(31). <https://doi.org/10.33571/rpolitec.v16n31a7>
- Runeson, P., Höst, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* vol. 14, 131–164 (2009). <https://doi.org/10.1007/s10664-008-9102-8>

- Riady, S. R., Sofi, K., Shadiq, J., & Arifin, R. W. (2022). Selection of Feature Driven Development (FDD) Model in Agile Method for Developing Information System of Mosque Management. *Journal of Computer Networks, Architecture and High-Performance Computing*, 4(2), 127–136. <https://doi.org/10.47709/cnahpc.v4i2.1469>
- Ronanki, K., Cabrero-Daniel, B., Horkoff, J., & Berger, C. (2023). RE-centric Recommendations for the Development of Trustworthy(er) Autonomous Systems. *ACM International Conference Proceeding Series*. <https://doi.org/10.1145/3597512.3599697>
- Ruiz, D. (2015, February 10). *Generación de código – Claves del éxito*. Artículos y Tendencias Sobre Soluciones Tecnológicas IT: Blog.
- Sisilitsyn, Y. O., & Osadchy, V. V. (2023). USING CHATGPT IN DISTANCE LEARNING FOR BEGINNERS IN PROGRAMMING. *INFORMATION TECHNOLOGIES AND LEARNING TOOLS*, 97(5).
- Sun, J., Liao, Q. V., Muller, M., Agarwal, M., Houde, S., Talamadupula, K., & Weisz, J. D. (2022). Investigating Explainability of Generative AI for Code through Scenario-based Design. *International Conference on Intelligent User Interfaces, Proceedings IUI*, 212–228. <https://doi.org/10.1145/3490099.3511119>
- Tejera-Martínez, F., Aguilera, D., & Vélchez-González, J. M. (2020). Lenguajes de programación y desarrollo de competencias clave. Revisión sistemática. *Revista Electrónica de Investigación Educativa*, 22. <https://doi.org/10.24320/redie.2020.22.e27.2869>
- Tomalá De La Cruz, M. A., Mascaró Benites, E. M., Carrasco Cachinelli, C. G., & Aroni Caicedo, E. V. (2023). Incidencias de la inteligencia artificial en la educación. *RECIMUNDO*, 7(2). [https://doi.org/10.26820/recimundo/7.\(2\).jun.2023.238-251](https://doi.org/10.26820/recimundo/7.(2).jun.2023.238-251)
- Tuomi, I. (2018). The Impact of Artificial Intelligence on Learning, Teaching, and Education. *JRC Publications Repository*.
- UNESCO. (2021). *Recomendación sobre la ética de la inteligencia artificial*.
- Weisz, J. D., Muller, M., Houde, S., Richards, J., Ross, S. I., Martinez, F., Agarwal, M., & Talamadupula, K. (2021). Perfection Not Required? Human-AI Partnerships in Code Translation. *International Conference on Intelligent User Interfaces, Proceedings IUI*, 402–412. <https://doi.org/10.1145/3397481.3450656>
- White, J., Hays, S., Fu, Q., Spencer-Smith, J., & Schmidt, D. C. (2023). *ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design*. <http://arxiv.org/abs/2303.07839>
- Yao, Q. (2021). The Application of Artificial Intelligence Voice Recognition on Helping Elders Use Mobile Phones More Easily. *Journal of Physics: Conference Series*, 1824(1). <https://doi.org/10.1088/1742-6596/1824/1/012007>

Anexos

Anexo 1. Guía de entrevista

GUÍA DE ENTREVISTA
Generación de código fuente con Inteligencia Artificial
Esta entrevista trata del tema relacionado con la generación de código mediante Inteligencia Artificial (IA) como una técnica de levantamiento de datos y está relacionado con el trabajo de investigación denominado "Generación de Código a partir de requisitos basados en voz. Un análisis multicriterio de herramientas de generación de código basadas en Inteligencia artificial" de la Universidad de Cuenca. Gracias anticipadas por su colaboración.
SECCIÓN 1
A efectos de llevar estadísticas de los participantes en esta encuesta, por favor indique su género:
☐ Masculino ☐ Femenino ☐ Prefiero no decirlo
Indique el título de tercer nivel que usted tiene:

Indique el título de cuarto nivel que usted tiene (Ninguno=no tiene título de 4to. nivel):

¿Usted trabaja en el sector público o privado, o en el libre ejercicio de la profesión?
☐ Sector público ☐ Sector privado ☐ Libre ejercicio de la profesión
¿Cuántos años de experiencia tiene en desarrollo de software?
☐ 1-5

O 6-10

O 10-15

O Más de 15 años

Marque los tipos de software que usted desarrolla:

☐ **Aplicaciones de escritorio**

☐ **Aplicaciones Web**

☐ **Aplicaciones móviles**

☐ **Otro tipo de aplicaciones**

¿Usted ha utilizado herramientas de IA para generar código fuente?

O Si

O No

Preguntas relacionadas con el uso de herramientas de generación de código con IA

¿Cuál es su experiencia específica en el uso de herramientas de IA para la generación de código?

Escriba un párrafo.

¿La información que ha ingresado como datos de entrada a la herramienta de generación de código está basada en los requisitos previamente definidos con los usuarios?

O Si

O No

¿Podría mencionar algunas herramientas de IA que haya utilizado o conozca para la generación de código? Escriba un párrafo.

¿Cuáles considera que son las principales ventajas de las herramientas de IA para generación de código? (Señale ventajas por cada herramienta que conozca) Escriba un párrafo.

Seleccione cinco criterios que considera usted deben tomarse en cuenta para seleccionar una herramienta de IA:
☐ Rendimiento del modelo de IA usado (que afecta la precisión del código)
☐ Costo de utilizar la herramienta (monetario + recurso computacional)
☐ Tiempo que tarda la herramienta en generar código
☐ Facilidad de uso
☐ Personalización
☐ Escalabilidad
☐ Seguridad y privacidad
☐ Integración con herramientas de desarrollo existentes
☐ Comunidad y apoyo
☐ Consideraciones legales y éticas
¿Ha tenido alguna experiencia negativa o algún desafío con alguna de estas herramientas?
O Si
O No
En relación a sus experiencias o desafíos presentados
Indique cuáles han sido las experiencias negativas que ha tenido en el uso de alguna herramienta de generación de código mediante IA. Escriba un párrafo.
Indique cuáles han sido los desafíos con los cuales se han enfrentado durante el uso de alguna herramienta de generación de código mediante IA. Escriba un párrafo.
Recomendaciones y comentarios finales

¿Cuáles son sus recomendaciones para aquellos que estén por utilizar herramientas de IA para la generación de código? Escriba un párrafo.

Comentarios adicionales

Muchas gracias por sus respuestas

Anexo 2. Entrevista en el Formulario de Google

Entrevista - Generación de código

docs.google.com/forms/d/e/1FAIpQLSdfyzUyJBp7fi49EemkjhUqjWjMcLSB-_VkwLRaaVRwol_KdQ/viewform

ASUS E-Service U eVirtual UC SGA Facebook WhatsApp Google Drive CitarReferencias-ISO Overleaf - Latex YouTube Fotos Discord Multi

Entrevista - Generación de código fuente con Inteligencia Artificial

Esta entrevista trata del tema relacionado con la generación de código mediante Inteligencia Artificial (IA) como una técnica de levantamiento de datos y está relacionado con el trabajo de investigación denominado "Generación de Código a partir de requisitos basados en voz. Un análisis multicriterio de herramientas de generación de código basadas en Inteligencia artificial" de la Universidad de Cuenca.

Gracias anticipadas por su colaboración.

* Indica que la pregunta es obligatoria

Correo electrónico *

☐ Registrar mi dirección de correo electrónico en mi respuesta

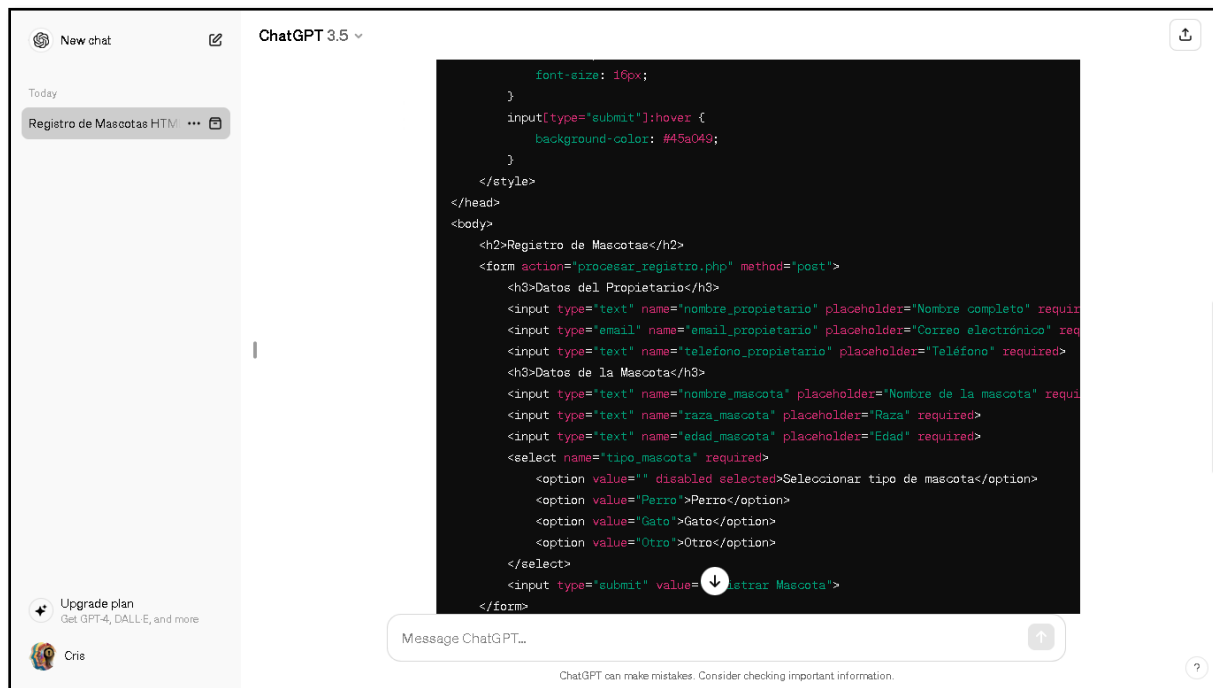
A efectos de llevar estadísticas de los participantes en esta encuesta, por favor * indique su género:

☐ Masculino

☐ Femenino

☐ Prefiero no decirlo

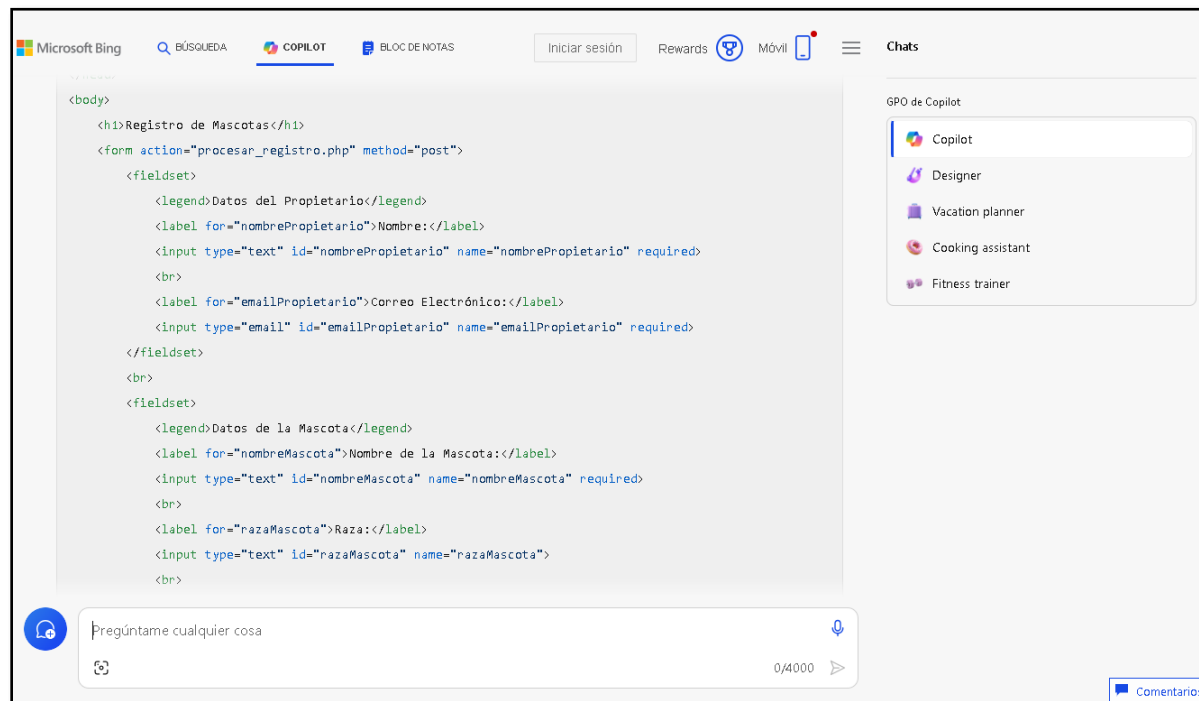
Anexo 3. Interfaz de ChatGPT



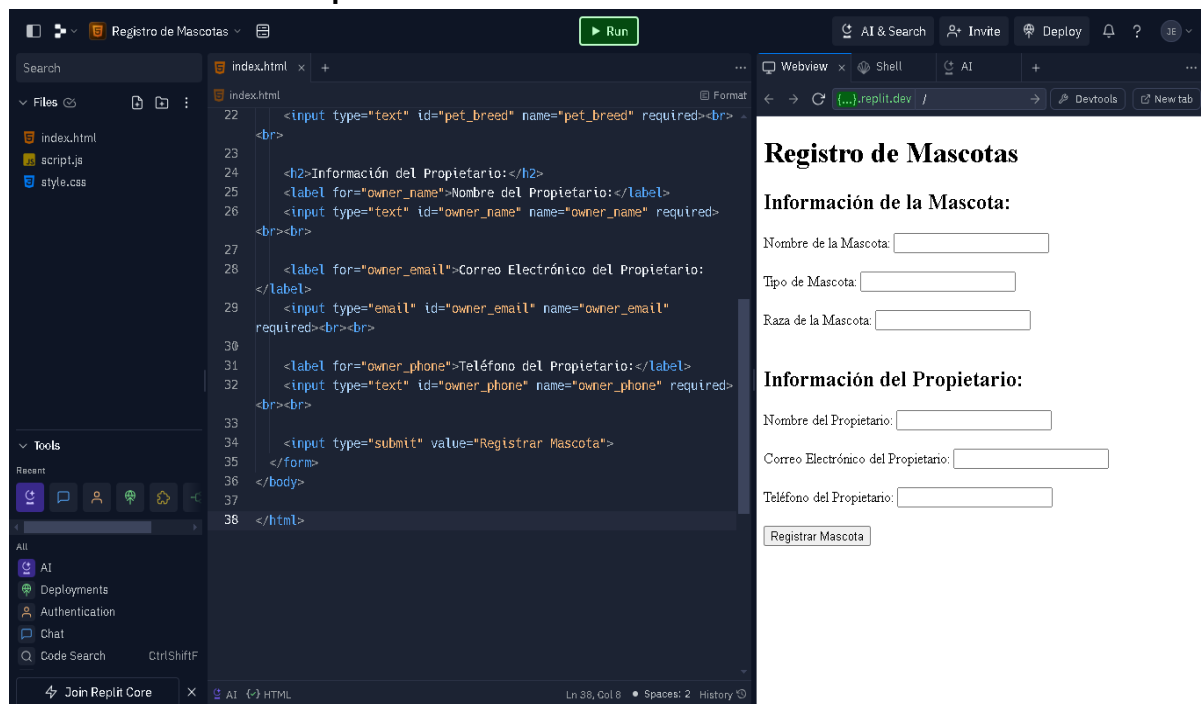
Anexo 4. Interfaz de Gemini



Anexo 5. Interfaz de Bing Chat



Anexo 6. Interfaz de Replit GhostWriter



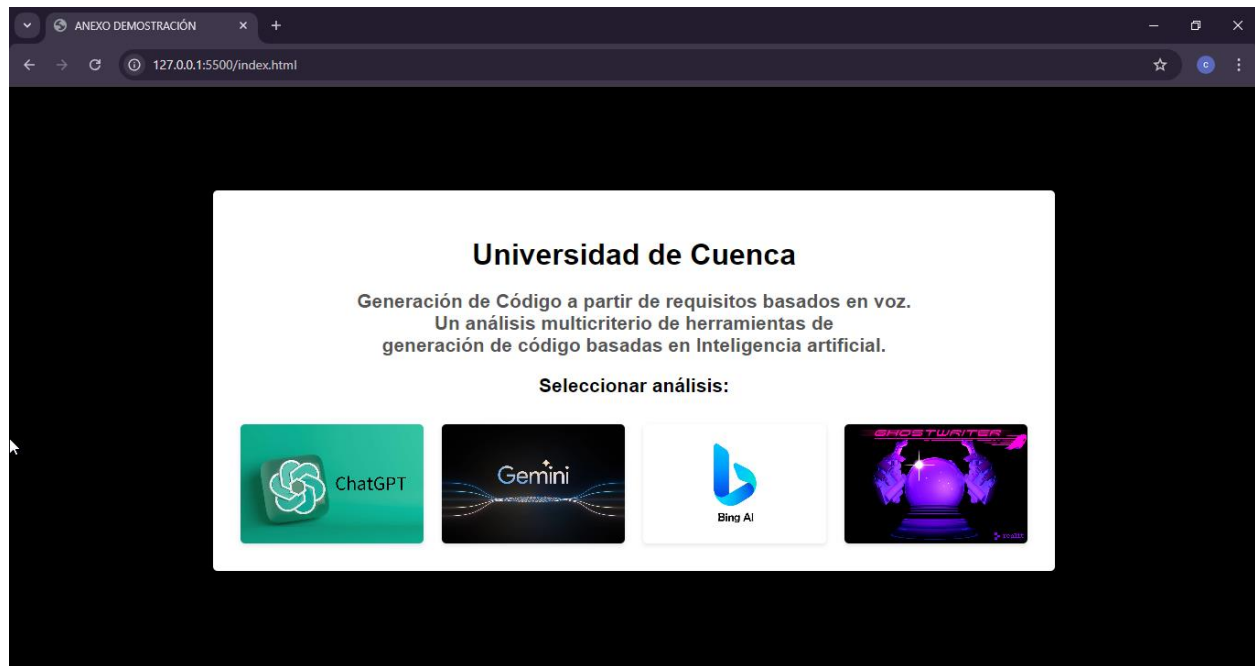
Anexo 7. Método TOPSIS en Excel

ArchivoInicioInsertarDisposición de páginaFórmulasDatosRevisarVistaProgramadorAyuda¿Qué desea hacer?

128

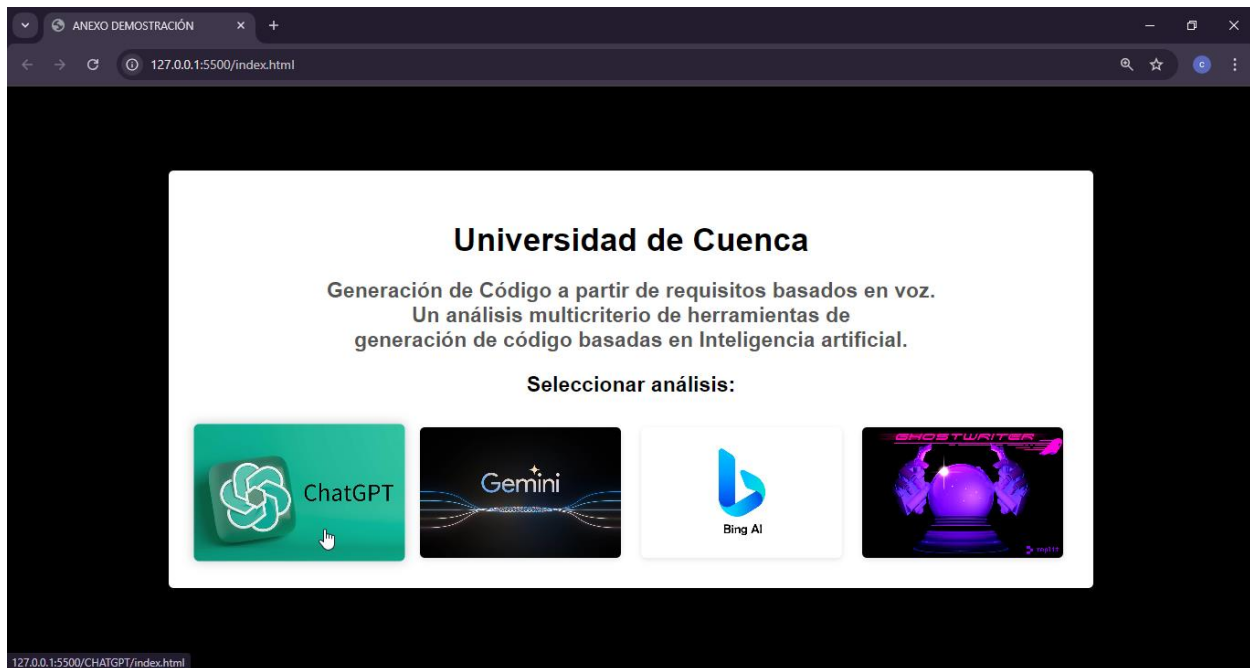
</

Anexo 8. Alojamiento de los resultados generados por las herramientas IAs seleccionadas, con respecto al objetivo 2 (caso de estudio planteado) de la tesis, para que se pueda apreciar su respectivo funcionamiento.

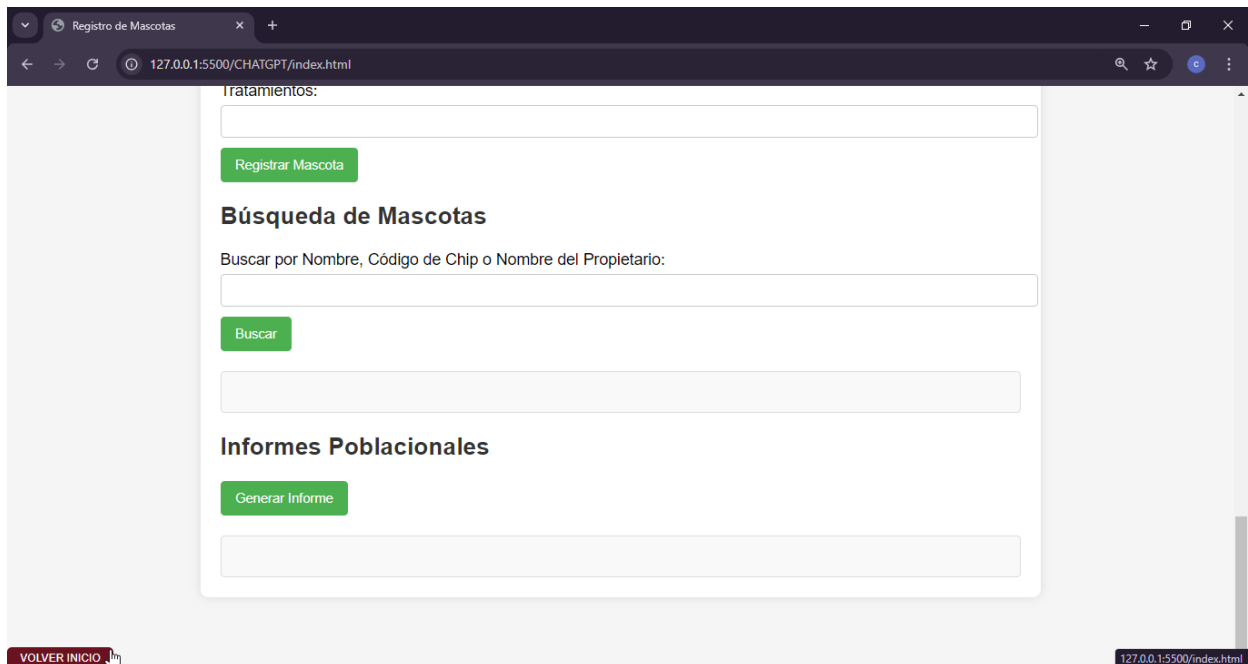


Para la presentación de los resultados generados por las herramientas IAs seleccionadas en este estudio, se elaboró una página estilo menú, que sirve para tener organizados los resultados y su apreciación no se torne tediosa. El pequeño menú consta de 4 opciones, que representan las herramientas IAs determinadas y utilizadas en este estudio: ChatGPT, Gemini (Bard); Bing Chat y Replit GhostWriter.

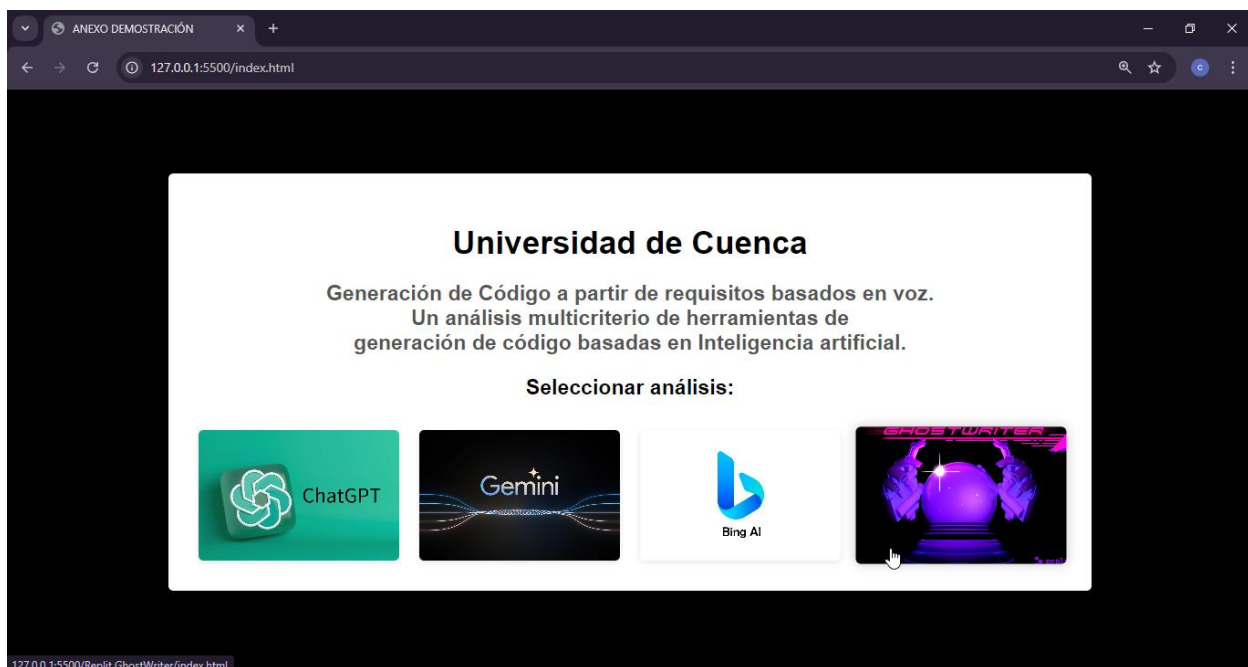
Al seleccionar cualquiera de las opciones presentadas, se redirigirá a los resultados (códigos) pertinentes generados y presentados en esta presente tesis.



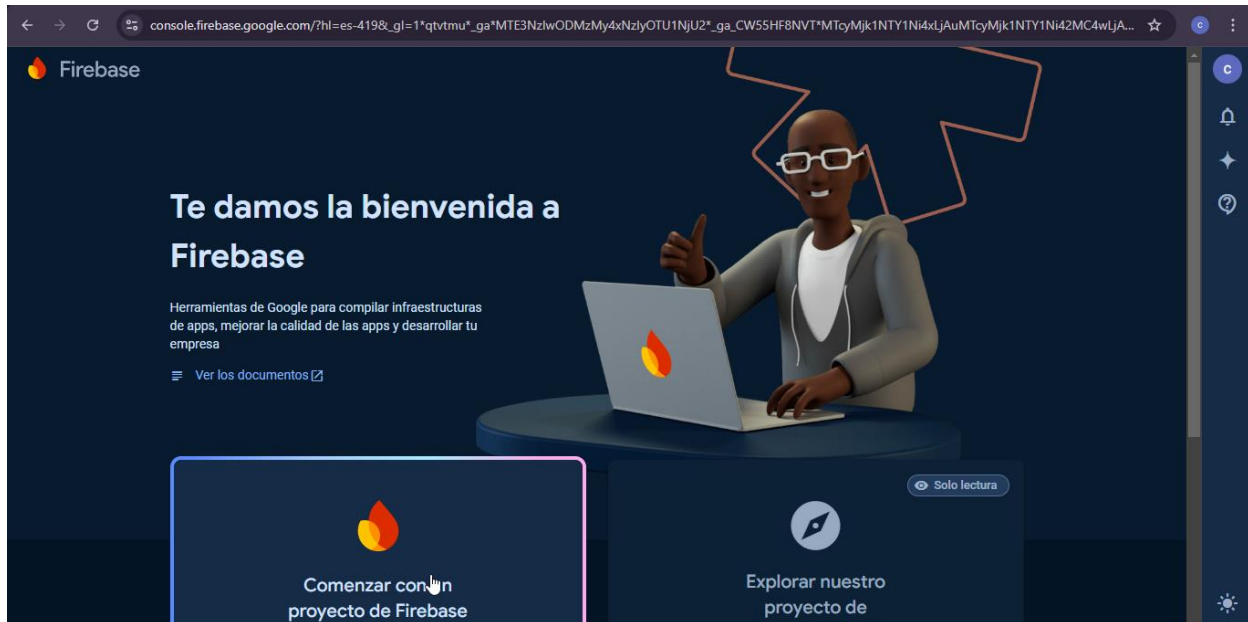
Al seleccionar, por ejemplo, la opción ChatGPT, se podrá visualizar y utilizar la página generada por la herramienta mencionada, con respecto al caso de estudio planteado en el objetivo 2.



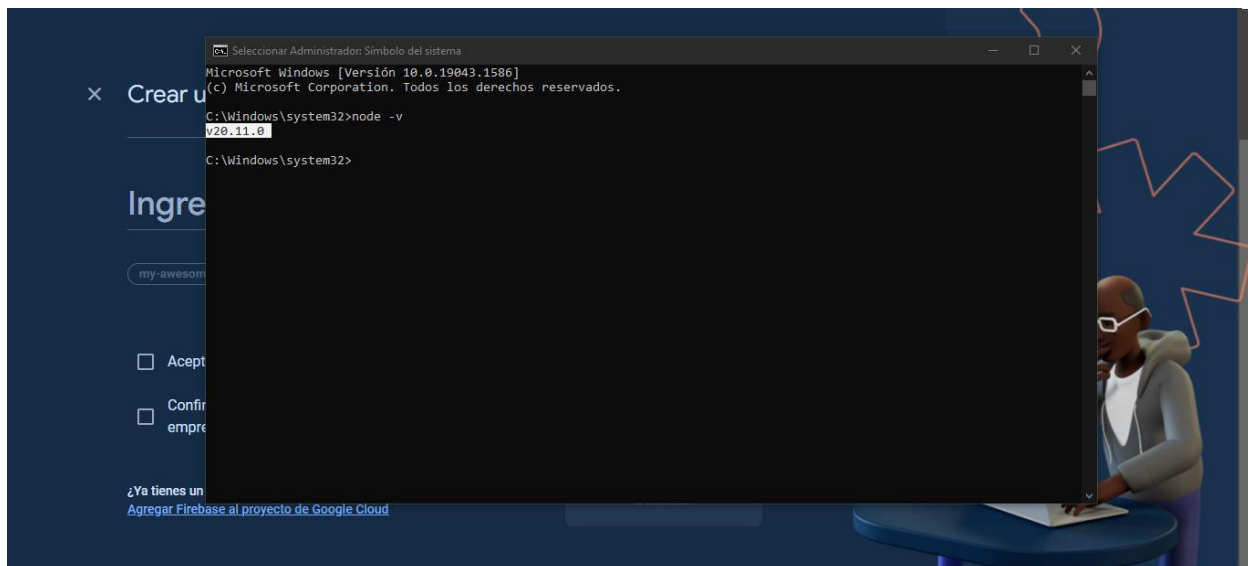
En la parte inferior, en todas las páginas de demostración, se encuentra la opción de volver al menú inicial, para poder seleccionar otra opción y así, ir las revisando todas sin dificultad.



Para alojar esta demostración y que pueda ser accedida por cualquier persona, se utilizó la herramienta proporcionada por Google: Firebase. Esta herramienta permite obtener un alojamiento y hosting gratuito.



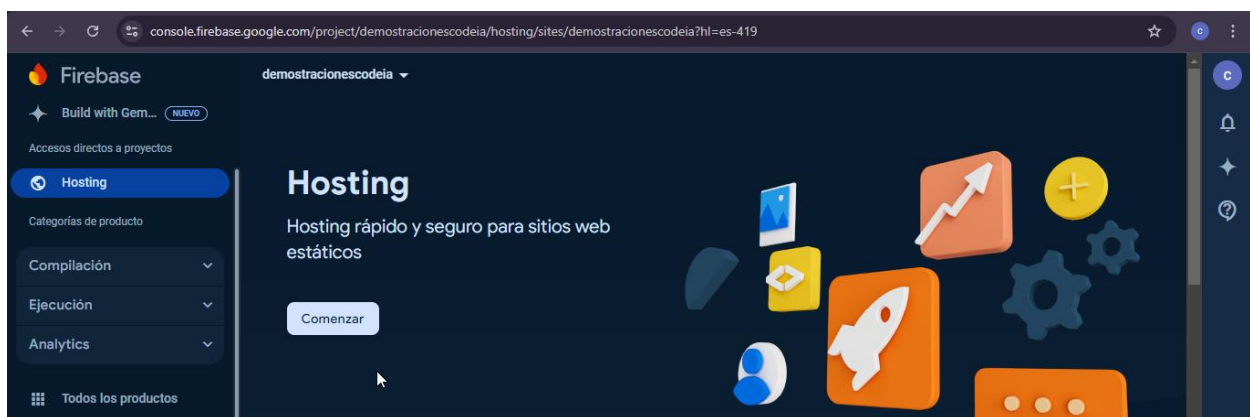
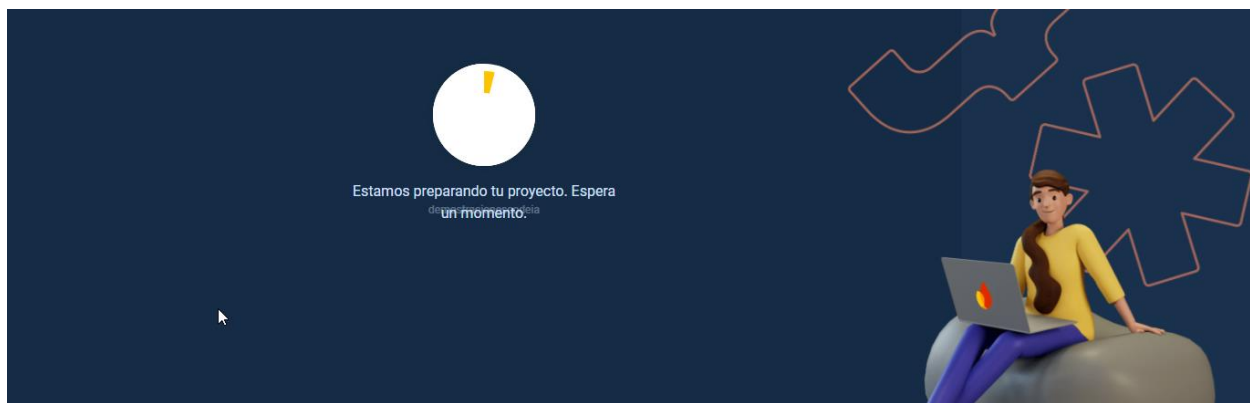
Para poner en funcionamiento y hacer el despliegue en Firebase es relativamente sencillo. Primeramente se necesita tener instalado en el computador el entorno Node.js

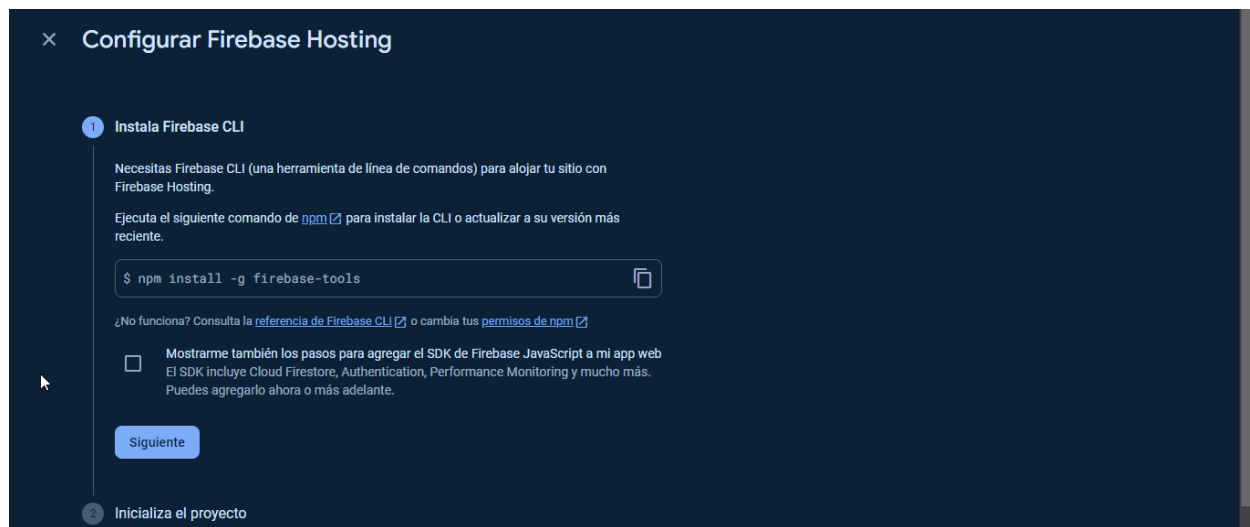


Continuamos asignando un nombre al proyecto en Firebase, que también, ese mismo nombre servirá para dominio:

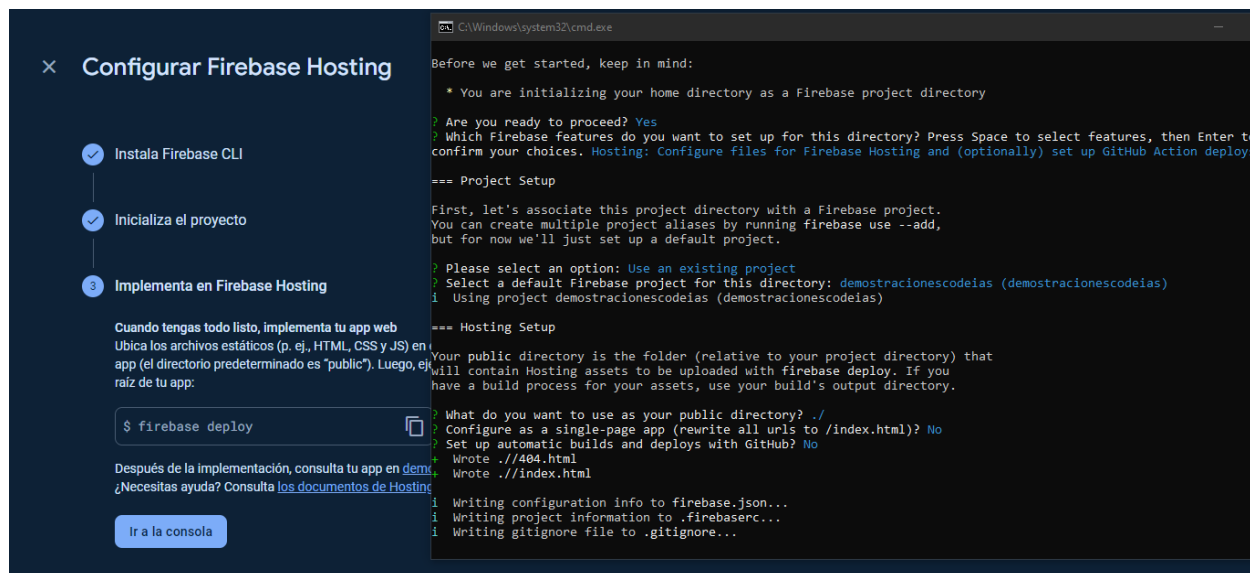


Una vez asignado el nombre, procedemos a crear el proyecto y configurar el despliegue.

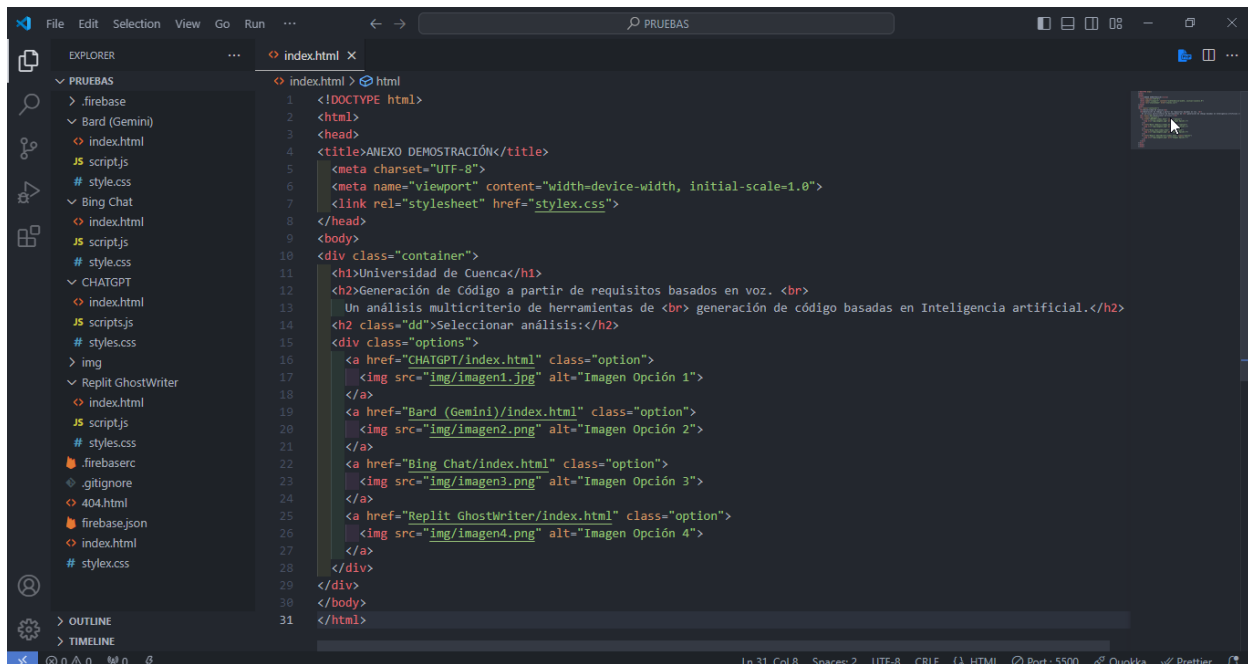




Se procede a copiar y pegar los códigos proporcionados por Firebase a nuestro CMD, para que el propio Firebase haga sus configuraciones pertinentes:

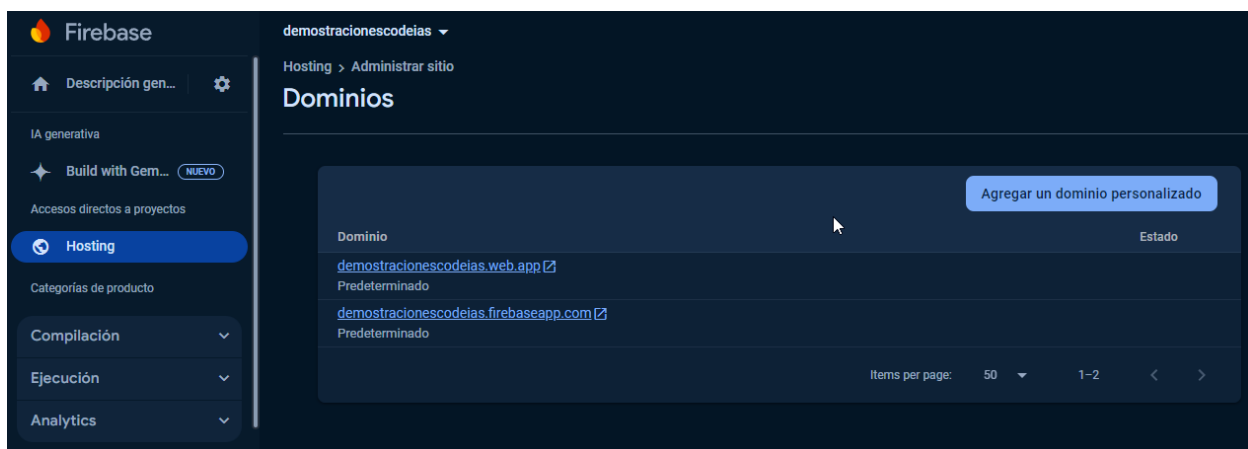


En el transcurso, Firebase hará unas preguntas sobre el directorio del proyecto, y si permitimos que se rescriba la página principal (index.html), a la cual se responde con la letra: N.

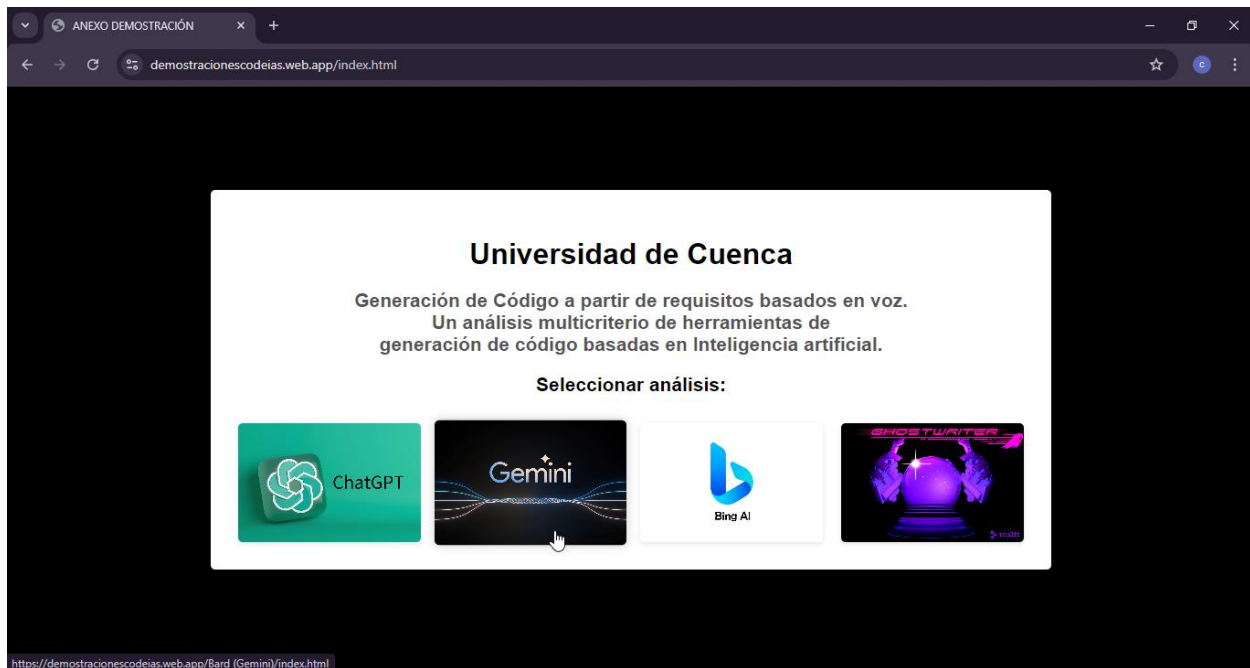


```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>ANEXO DEMOSTRACIÓN</title>
5 <meta charset="UTF-8">
6 <meta name="viewport" content="width=device-width, initial-scale=1.0">
7 <link rel="stylesheet" href="stylesheet.css">
8 </head>
9 <body>
10 <div class="container">
11 <h1>Universidad de Cuenca</h1>
12 <h2>Generación de Código a partir de requisitos basados en voz. <br>
13 Un análisis multicriterio de herramientas de <br> generación de código basadas en Inteligencia artificial.</h2>
14 <h2 class="dd">Seleccionar análisis:</h2>
15 <div class="options">
16 <a href="CHATGPT/index.html" class="option">
17 
18 </a>
19 <a href="Bard (Gemini)/index.html" class="option">
20 
21 </a>
22 <a href="Bing Chat/index.html" class="option">
23 
24 </a>
25 <a href="Replit GhostWriter/index.html" class="option">
26 
27 </a>
28 </div>
29 </div>
30 </body>
31 </html>
```

Es importante tener todos los archivos organizados dentro de una carpeta, para que Firebase pueda hacer sus formatos pertinentes y se pueda lograr el alojamiento.



Completado el alojamiento, Firebase nos proporciona dos dominios de acceso, permitiéndonos agregar un dominio personalizado, en caso de contar con uno.



Para acceder a revisar y testear los resultados presentados de la generación de código a partir de comandos de voz, por medio de herramientas de inteligencia artificial, ingresar a cualquiera de los siguientes dominios:

<https://demostracionescodeias.web.app>

<https://demostracionescodeias.firebaseio.com>

Anexo 9. Captura de una parte del artículo académico realizado en base al trabajo de titulación.

Exploring generative AI tools to obtain source code from voice-based requirements specification

Cristian Méndez-Espinoza
Department of Computer Science
University of Cuenca
Cuenca, Ecuador
ricardo.mendez@ucuenca.edu.ec

Otto Parra
Department of Computer Science
University of Cuenca
Cuenca, Ecuador
otto.parra@ucuenca.edu.ec

María Fernanda Granda
Department of Computer Science
University of Cuenca
Cuenca, Ecuador
fernanda.granda@ucuenca.edu.ec

Abstract—The software development lifecycle starts with the requirements elicitation considering the end-users (stakeholders) as first-class citizens. In this context, there are many methods to obtain the software requirements to develop an application, one is the user interview. However, the next step is processing the interview results to design the application required by the end-users. Then, software developers begin the implementation using any programming language. The time required to obtain a visible result of the implementation is sometimes several days (or weeks). Therefore, if the end-user can verify the result of applying the requirements to obtain the software in a short time is very important because it is possible to check and give suggestions to improve the obtained results. To obtain the first version of the software we can take the audio of the interview as input in a generative AI tool to obtain the “first version” of the software. The audio is converted into text using natural language processing and then such text is given as input to the generative AI tool to obtain the “first version” of the software. In this context, we selected four generative AI tools by applying multi-criteria decision-making (specifically, TOPSIS), then by using these AI tools we generated source code taking as input the aforementioned voice-based requirements. We evaluate the results obtained with experts and software developers in a software product and we get results that could help the end-users to check the compliance of the requirements in a short time.

Keywords—voice-based requirements, generative AI tool, software development life cycle

I. INTRODUCTION

The software development lifecycle (SDLC) consists of

software engineers. So, early on, end-users should be able to review the software product under development and thus verify compliance with the requirements throughout the development process. Therefore, this software product verification in a “first version” will permit saving resources (time, money, etc.) during the software development.

We have two possibilities to obtain a “first version” of a software product: (i) using agile methods for software development [3] and writing the source code according to the requirements specifications based on feature-driven development [4], and (ii) using artificial intelligence (AI) tools to generate source code of software products. In this paper, we consider the second option to generate source code of software products based on feature-driven development and voice-based requirements specification.

Artificial Intelligence has its origins in the 40s of the last century [5], however, in recent years it has had an increase in its development, especially in applications of long language models (LLM) which have been implemented in AI tools that allow us to perform almost any activity in today's society. Currently, it is a more emerging topic in the field of software engineering, therefore, we are interested in applying it to obtain early versions of software products.

In the related literature, there are several solutions to apply generative AI to generate the source code of a software product. We consider those AI tools to satisfy the following features such as: it has an open-source license, it supports voice as input based on requirements specifications, among others