UNIVERSIDAD DON BOSCO

Facultad de Ingeniería

Escuela de Computación

Investigación Aplicada 1:

Copilot en Visual Studio Code

Asignatura:

DPS G02T

Docente

Ing. Alexander Siguenza

Integrantes

Cartagena Rivera	José Ángel	CR190362
Chaves Flores	Jeanluca	CF190725
Mejía Ortiz	Karla Lissette	MO190663
Sánchez Mangandi	Ángel Guillermo	SM192656

Índice

1. Historia y Desarrollo de GitHub Copilot

a. Orígenes y evolución de GitHub Copilot

GitHub Copilot es un asistente de programación basado en inteligencia artificial que fue lanzado en junio de 2021. Su desarrollo se originó en la necesidad de herramientas que optimicen el proceso de escritura de código, permitiendo a los programadores ser más eficientes y creativos. En un entorno donde la demanda de software sigue creciendo, la idea de un asistente que pueda sugerir líneas de código y funciones completas se volvió esencial.

La concepción de Copilot se basa en el modelo de lenguaje GPT-3 de OpenAI, desarrollado utilizando una vasta cantidad de datos, incluyendo código fuente de repositorios públicos. Este enfoque permitió a Copilot no solo entender la sintaxis de varios lenguajes de programación, sino también captar patrones, estilos y mejores prácticas en la escritura de código. A lo largo de su evolución, Copilot ha sido reconocido como una herramienta innovadora que ayuda a los desarrolladores a reducir el tiempo de codificación y a mejorar la calidad del software.

b. Colaboración entre GitHub y OpenAI

La colaboración entre GitHub y OpenAI ha sido fundamental para el desarrollo de Copilot. GitHub, una plataforma de desarrollo colaborativo líder en la industria, se asoció con OpenAI, una organización de investigación en inteligencia artificial, para integrar las capacidades avanzadas de generación de lenguaje de Codex en su plataforma.

OpenAI aportó su experiencia en modelos de lenguaje y aprendizaje automático, mientras que GitHub proporcionó el entorno y los datos necesarios para entrenar y refinar estos modelos en el contexto del desarrollo de software. Esta sinergia permitió crear una herramienta robusta que entiende el contexto del código y puede sugerir fragmentos útiles, completaciones y soluciones a problemas de programación.

Beneficios de la colaboración:

La colaboración entre GitHub y OpenAI ha producido una herramienta que no solo asiste a los desarrolladores en la escritura de código, sino que también tiene el potencial de cambiar la educación en programación y el desarrollo de software en general. Algunas de las principales ventajas de GitHub Copilot incluyen:

- ✓ **Aumento de la productividad**: Los desarrolladores pueden escribir código más rápidamente con sugerencias automáticas y completaciones contextuales.
- ✓ **Reducción de errores**: Copilot ayuda a identificar y corregir errores comunes en el código, mejorando la calidad general del software.

- ✓ **Innovación continua**: La herramienta sigue evolucionando con retroalimentación constante de los usuarios, lo que asegura que se mantenga relevante y efectiva en un campo en rápida evolución.

c. Versiones y mejoras desde su lanzamiento inicial

Estas mejoras han sido impulsadas por la retroalimentación continua de los usuarios y los avances en la tecnología de inteligencia artificial, consolidando a GitHub Copilot como una herramienta esencial para los desarrolladores de software en todo el mundo.

Desde su lanzamiento en 2021, GitHub Copilot ha experimentado varias actualizaciones y mejoras significativas:

Julio 2021: Poco después de su lanzamiento, Copilot recibió comentarios extensivos de los usuarios, lo que permitió mejorar su precisión y reducir errores comunes. Se realizaron ajustes para aumentar la relevancia de las sugerencias y la fluidez en la integración con el entorno de desarrollo integrado (IDE).

Octubre 2021: Se introdujo una actualización importante que mejoró la capacidad de Copilot para comprender el contexto más amplio del código, resultando en sugerencias más relevantes y útiles para los desarrolladores. Esta actualización también incluyó mejoras en la interfaz de usuario, facilitando la interacción y personalización de las sugerencias.

Marzo 2022: Copilot expandió su soporte para más lenguajes de programación, incluyendo Ruby, Java y PHP, mejorando su utilidad para una mayor cantidad de desarrolladores. También se optimizó la integración con diferentes IDEs, como Visual Studio Code y JetBrains, para ofrecer una experiencia más cohesiva y eficiente.

Enero 2023: Con una versión optimizada de Codex, Copilot pudo ofrecer sugerencias más rápidas y precisas, además de introducir la capacidad de aprender de las correcciones y preferencias de los usuarios. Esta versión también incorporó mejoras en la personalización, permitiendo a los desarrolladores ajustar las configuraciones para alinearlas mejor con sus estilos y necesidades de codificación.

Marzo 2024: La última actualización incluye soporte mejorado para lenguajes de programación emergentes y la incorporación de aprendizaje continuo basado en los proyectos en los que se utiliza Copilot. Esto permite que Copilot se adapte de manera más efectiva a los cambios en las tecnologías y metodologías de desarrollo, ofreciendo siempre las sugerencias más actualizadas y pertinentes.

2. Funcionamiento Técnico de GitHub Copilot

a. Arquitectura y modelo de inteligencia artificial detrás de Copilot

GitHub utiliza un modelo llamado Codex, variante de GPT-3, es un modelo de variante natural desarrollado por OpenAI, que está programado y entrenado específicamente para entender y generar código en diferentes lenguajes de programación, esto para ayudar a los creadores de código a desarrollar aplicaciones de manera mas eficiente. Esto se traduce en una arquitectura de redes neuronales eficaz en el procesamiento de lenguaje natural, es decir, comprende entradas como formas de comunicación, idiomas, códigos, imágenes, entre otras formas de expresión.

b. Algoritmos y técnicas de aprendizaje profundo utilizados

Codex fue entrenado en base a una gran cantidad de datos de código fuente, ya sea por repositorios publicados en GitHub, entradas básicas de varios lenguajes, grandes cantidades de texto en internet, o entrenamiento manual, permitiéndole a Codex comprender patrones y estructuras en el ámbito de la programación. Esto con el fin de que Codex presentara respuestas a tareas específicamente relacionadas con la programación como completar fragmentos de código, sugerir funciones, y ayudar a resolver errores, siempre manteniendo una calidad optima en el código generado.

c. Cómo Copilot se integra con Visual Studio Code a nivel técnico

La integración de GitHub Copilot con Visual Studio Code (VS Code) se realiza a través de una extensión específica que proporciona funcionalidades de autocompletado y sugerencias de código dentro del editor. A nivel técnico, la integración se maneja de la siguiente manera:

Extensión de VS Code: GitHub Copilot se implementa como una extensión que se instala en Visual Studio Code. Esta extensión actúa como un puente entre el entorno de desarrollo y el modelo de inteligencia artificial, enviando fragmentos de código y solicitudes de sugerencias al servidor de Copilot y recibiendo las respuestas correspondientes.

API de GitHub Copilot: La extensión se comunica con el servidor de GitHub Copilot mediante una API que permite el intercambio de datos entre el editor y el modelo de lenguaje. La API maneja las solicitudes de autocompletado y las respuestas generadas por el modelo, integrando estas sugerencias en el flujo de trabajo del desarrollador.

Interfaz de Usuario: La extensión de Copilot incluye una interfaz de usuario dentro de VS Code que muestra las sugerencias de código generadas por el modelo. Esta interfaz está diseñada para ser intuitiva y se integra con las características estándar del editor, como el resaltado de sintaxis y la navegación del código.

Procesamiento en Tiempo Real: A medida que el desarrollador escribe código en VS Code, la extensión envía información en tiempo real al servidor de Copilot. El modelo procesa esta información y devuelve sugerencias que se muestran en el editor en forma de autocompletado y recomendaciones contextuales.

3. Impacto en la Productividad del Desarrollador

a. Estudios de caso y análisis comparativos pre y post-integración de Copilot

En términos de productividad, le evolución de GitHub Copilot ha ido mejorando exponencialmente, ya que antes de la normalización de la Inteligencia artificial, el tiempo dedicado a la realización del proyecto se dividía básicamente en la búsqueda de información, el funcionamiento de cada línea de código, prueba y error, interrupciones por terceros, además que la consistencia y la calidad de la realización de los proyectos era variable ya que dependía de la experiencia del desarrollador, la buena implementación del lenguaje utilizado y su compatibilidad con las distintas plataformas. Ahora gracias a esta herramienta, la búsqueda se simplifica en sugerencias brindadas por Copilot, el autocompletado de líneas e incluso bloques de código, la gran compatibilidad con distintos lenguajes y los frameworks adecuados al tipo de trabajo realizado en el proyecto, que a pesar de las sugerencias de Copilot, es esencial una revisión humana continua para asegurar la calidad y seguridad del código, pero es incluso más cómodo que realizar procesos repetitivos desde cero y una revisión manual del proyecto intensiva.

b. Métodos de medición de productividad en el desarrollo de software

La medición de la productividad en el desarrollo de software puede ser un desafío, dado que no existe una métrica única que capture todos los aspectos del trabajo de un desarrollador. A continuación, se presentan algunos métodos comunes para evaluar la productividad en este contexto:

Análisis de Líneas de Código: Uno de los métodos tradicionales para medir la productividad es contar las líneas de código (LOC) escritas. Sin embargo, esta métrica puede ser engañosa, ya que no considera la calidad del código ni el valor real de las funcionalidades implementadas. Una línea de código puede ser muy simple o extremadamente compleja.

Tiempo de Resolución de Problemas: El tiempo que un desarrollador tarda en resolver un problema o bug puede ser un indicador de productividad. Sin embargo, esta métrica debe ser interpretada con cuidado, ya que la complejidad del problema y la experiencia del desarrollador pueden influir en el tiempo requerido.

Retroalimentación de Revisiones de Código: Las revisiones de código por parte de pares pueden proporcionar una evaluación cualitativa de la productividad. Las observaciones y comentarios de otros desarrolladores sobre la calidad y la eficacia del código pueden ofrecer una perspectiva valiosa sobre el desempeño y la productividad.

c. Percepciones y experiencias de desarrolladores que utilizan Copilot

- "Hay que entender que Copilot no es capaz de escribir por sí mismo el código que muestra cuando se arroja una consulta, sino es el hecho que se basa en el contexto del código que generamos, los nombres de funciones y los comentarios, para producir sugerencias que pudieran sernos de utilidad y nos ahorraran tiempo, sobre todo en las tareas más frecuentes y anodinas. Que incluso a veces para funciones más complejas, se necesita hacer una descripción de lo que debe recibir y devolver, para en muchos casos, Copilot nos ofrecerá una solución bastante funcional, que aun así será interesante revisar. Por otro lado, Como ha aprendido de programadores reales, en ocasiones genera un código tan sintético que, aún funcionando, puede resultar altamente ilegible y poco mantenible y, en algunas ocasiones, puede incluso contener operaciones bastante inseguras para asegurar dicho funcionamiento." -Pablo Huet, Experto FrontEnd.
- "Copilot es capaz de predecir el mensaje de registro que quiere añadir basándose a las condiciones del bloque. También destaca cómo la herramienta **sugiere un nombre adecuado para una función sin que no se haya escrito ni una línea de código** de ella, solamente a partir de un comentario que escribas. Además, ha encontrado mucha utilidad a la hora de **escribir código repetitivo**"-Juan Font, Desarrollador.
- "La herramienta **no va a reemplazar a los programadores** ni mucho menos, pero sí que va a quitarles mucho trabajo y muchas búsquedas en Stackoverflow, un portal online en el que muchos programadores resuelven dudas."-Pablo Serrano, ingeniero especializado en sistemas.

4. Ventajas de Utilizar GitHub Copilot

a. Casos de uso específicos donde Copilot ha demostrado ser beneficioso

Desarrollo de Código Rápido y Eficiente: En proyectos donde el tiempo es esencial, GitHub Copilot ha permitido a los desarrolladores escribir código de manera más rápida al proporcionar sugerencias contextuales y autocompletar fragmentos de código. Por ejemplo, en el desarrollo de una aplicación web, Copilot puede sugerir automáticamente funciones y clases basadas en el contexto del código existente, acelerando el proceso de codificación y reduciendo el tiempo de desarrollo.

Refactorización de Código: En el proceso de refactorización, donde se busca mejorar la estructura del código sin cambiar su funcionalidad, Copilot ha asistido en la identificación de patrones comunes y en la aplicación de mejoras en el código.

Por ejemplo, en un proyecto de refactorización de una base de código heredada, Copilot ayudó a simplificar y optimizar el código, lo que resultó en una mayor mantenibilidad y eficiencia.

b. Mejora en la eficiencia de escritura de código

GitHub Copilot esta establecida como la herramienta por excelencia para ayudar a los programadores a escribir código más rápido, funcionando como compañero que sugiere líneas de código mientras escribes en tiempo real, por lo que puede ser una herramienta esencial para los desarrolladores al acelerar el proceso de creación de software; a la vez qu puede generar comentarios explicativos para ayudar a comprender mejor la funcionalidad y que relevancia tiene para el proyecto, todo esto en lenguaje natural.

c. Reducción del tiempo en tareas repetitivas y comunes

GitHub Copilot tiene la funcionalidad de predecir líneas o bloques de códigos basado en el código que se ha escrito anteriormente, siendo particularmente útil cuando se está trabajando en estructuras con código repetitivo, como bucles, plantillas de funciones o declaraciones condicionales. Además, como Copilot comprende patrones, este es capaz de arrojar sugerencias al código que se esta escribiendo en tiempo real, esto capaz de hacerlo en distintos lenguajes de programación.

d. Asistencia en la resolución de problemas complejos y depuración

Una de las funcionalidades mas destacadas con las que cuenta GitHub Copilot es que puede testear el código que hayamos generado, ya sea por sugerencia de Copilot o por cuenta propia, analizándolo en busca de errores y su funcionamiento dentro del proyecto, el comportamiento de las funciones, clases o métodos que se estén desarrollando, todas estas sugerencias pueden incluir la estructura básica de las pruebas para verificar el comportamiento esperado. Es fundamental asegurarse de que las pruebas sean completas y proporcionen una cobertura completa del código. Una de las mejores prácticas en la resolución de problemas complejos es la escritura de pruebas unitarias que validen la funcionalidad del código. Copilot puede sugerir y generar pruebas unitarias basadas en el código que ya has escrito, lo que te permite detectar errores y asegurar la correcta funcionalidad del código.

5. Limitaciones y Desafíos de GitHub Copilot

a. Dependencia excesiva y su impacto en el desarrollo de habilidades

Actualmente, Copilot se ha convertido en una herramienta sumamente utilizada tanto en desarrolladores novatos adentrándose en el mundo de la programación, como por desarrolladores ya experimentados en su campo; la herramienta con la que se aborda los desafíos buscan aprovechar al máximo las capacidades de la IA, es con la que hay que tener en cuenta que esta en una etapa temprana de desarrollo y que no puede reemplazar por completo el juicio y la creatividad de la mente humana, pese a todo esto la herramienta tiene un uso desmedido e indiferente ante cualquier problemática que se presente, claro que todo esto en el ámbito de la programación, ante cualquier situación en la que se tenga dificultades se consulta a la herramienta, creando así una dependencia excesiva de la IA en el desarrollo de software, lo que puede llevar a la pérdida de habilidades y conocimientos humanos esenciales.

b. Problemas de seguridad y privacidad de datos

Las limitaciones para tomar en cuenta al momento de integrar GitHub Copilot en nuestro proyecto, es que esta herramienta se ha entrenado de una gran cantidad de código de repositorios públicos, y que en ocasiones puede sugerir fragmentos de código que son muy similares a los existentes en estos repositorios. Esto plantea preocupaciones sobre la reutilización de código con licencias específicas o con derechos de autor, siendo que este no proporciona la atribución necesaria del código sugerido, ya que puede venir de una fuente con licencia que incluso necesite mención. El uso de IA en el desarrollo de software plantea preocupaciones sobre la privacidad y la seguridad de los datos, especialmente cuando se trata de aprender de bases de código existentes. Es fundamental garantizar que las soluciones de IA protejan la privacidad y la seguridad de los datos, y que cumplan con las regulaciones aplicables.

c. Calidad y relevancia de las sugerencias proporcionadas por Copilot

Es de saber que las sugerencias que arroja GitHub Copilot no es sustituto de la revisión de código o el juicio humano por lo tanto se de debe tener cuidado al momento de tomar en cuenta el código proporcionado, haciendo una revisión exhaustiva y teniendo conocimiento previo sobre lo que consultamos dentro de la herramienta; ya que en situaciones complejas, Copilot puede proponer soluciones no muy adecuadas con la consulta, especialmente si se trata en el ámbito empresarial o requisitos específicos que no están bien documentados en el código o no pudiéndolos procesar de manera eficiente, esto puede ocurrir cuando se usa un lenguaje que no esté bien fundamentado en la base de datos de Copilot puede tener dificultades para interpretar el resultado y arrojar sugerencias inciertas.

Otra cosa para tener en cuenta es que al momento de destacar en creatividad o donde el diseño es clave, Copilot puede no ser la mejor opción para consultar, ya que normalmente este arrojará comandos genéricos o diseños simples, esto puede influir en el pensamiento critico del desarrollador o hacer que se desvíe de su enfoque original, que incluso puede sobrecargar de código innecesario el proyecto.

d. Reacciones y críticas de la comunidad de desarrolladores

Las soluciones de IA pueden estar sesgadas debido a la información y los datos con los que han sido entrenadas. Los desarrolladores deben ser conscientes de este riesgo y trabajar para minimizar el sesgo y la discriminación en sus aplicaciones de IA. Los desarrolladores aceptan las propuestas de Copilot en torno a un 27% del tiempo, pero incluso cuando genera basura, es más cómodo aceptarlo y corregirlo manualmente que tener que escribirlo todo desde cero. Según algunos desarrolladores, este tipo de herramientas se encuentran aún en la fase de ayuda a la conducción, ni mucho menos en la de piloto automático, pero eso implica que les permite incrementar el tiempo que pasan sin tener que salir de su zona de concentración para salir a buscar algún detalle específico, o que sea especialmente bueno a la hora de corregir errores básicos y sencillos.

6. Comparación con Otras Herramientas Similares

a. Evaluación de otras herramientas de asistencia al desarrollo basadas en IA

A continuación, se presenta una evaluación de algunas herramientas destacadas que, al igual que GitHub Copilot, utilizan IA para asistir en el proceso de desarrollo:

TabNine: Es una herramienta de autocompletado basada en IA que ofrece soporte para una amplia gama de lenguajes de programación y editores. Utiliza un modelo de lenguaje profundo entrenado en un corpus extenso de código fuente para predecir y sugerir líneas de código, facilitando una escritura más rápida y precisa. La principal ventaja de TabNine es su capacidad de integrarse con varios IDEs y editores, proporcionando una experiencia consistente en diferentes entornos de desarrollo.

Kite: Es otra herramienta popular de autocompletado y asistencia en el código que emplea modelos de IA para mejorar la productividad del desarrollador. Ofrece sugerencias en tiempo real y puede aprender de la base de código del usuario para proporcionar recomendaciones más relevantes. Kite también incluye una función de documentación integrada que ayuda a entender mejor las bibliotecas y funciones utilizadas en el código.

Codex (de OpenAI): Es el modelo de lenguaje que impulsa GitHub Copilot, también se utiliza en otras herramientas de desarrollo. Su capacidad para comprender y generar código en

múltiples lenguajes lo convierte en una herramienta versátil para diversos entornos de desarrollo. Codex se destaca por su habilidad para interpretar y completar fragmentos de código complejos, así como por su integración con plataformas como Visual Studio Code a través de extensiones específicas.

b. Comparación de características, rendimiento y aceptación en la comunidad

A continuación, se presenta una comparación de las características clave, el rendimiento y la aceptación en la comunidad para Copilot y sus competidores:

Características	Copilot	TabNine	Kite	Codex
Lenguajes Soportados	Múltiples lenguajes, incluyendo Python, JavaScript, TypeScript, entre otros.	Múltiples lenguajes, con soporte adicional para lenguajes específicos mediante plugins.	Múltiples lenguajes, con énfasis en Python y JavaScript.	Múltiples lenguajes, incluyendo soporte para lenguajes complejos.
Integración IDE	Visual Studio Code, entre otros.	VS Code, Sublime Text, IntelliJ IDEA, entre otros.	VS Code, Atom, Sublime Text, entre otros.	Visual Studio Code, entre otros.
Capacidad de Autocompletado	Sugerencias contextuales avanzadas, generación de código y completado de fragmentos.	Autocompletado basado en el código fuente y sugerencias contextuales.	Autocompletado en tiempo real y sugerencias basadas en el contexto del código.	Sugerencias avanzadas y generación de código contextual.
Rendimiento	Alto, con integración fluida en la mayoría de los IDEs soportados.	Alto, aunque puede depender de la configuración del sistema y el IDE.	Sugerencias avanzadas y generación de código contextual.	Muy alto, con capacidad para entender y generar código en tiempo real.
Aceptación en la Comunidad	Altamente aceptado, especialmente en comunidades de desarrollo activas en GitHub.	Popular entre desarrolladores que buscan una solución de autocompletado gratuita.	Popular en la comunidad de desarrolladores que priorizan la integración con editores de texto.	Muy aceptado debido a su origen en OpenAI y su rendimiento avanzado.

c. Ventajas y desventajas relativas de Copilot frente a sus competidores.

Al comparar GitHub Copilot con otras herramientas, se pueden identificar las siguientes ventajas y desventajas:

Herramienta	Ventajas	Desventajas
Copilot	Integración con GitHub, generación avanzada de código, soporte de múltiples lenguajes.	Costo asociado para versiones avanzadas, posible dependencia del contexto.
TabNine	Gratuito, soporte para múltiples IDEs, autocompletado basado en el código existente.	Funcionalidades avanzadas requieren una suscripción de pago.
Kite	Autocompletado en tiempo real, soporte para varios editores de texto.	Menor profundidad en algunas sugerencias comparado con Copilot.
Codex	Avanzada capacidad de comprensión y generación de código soporta múltiples lenguajes.	Principalmente integrado con VS Code, puede tener un coste elevado.

7. Requerimientos de Hardware y Software

a. Especificaciones técnicas necesarias para un rendimiento óptimo

Para asegurar un rendimiento óptimo de Copilot en Visual Studio Code, se recomiendan las siguientes especificaciones mínimas:

Procesador: Intel Core i5 o AMD Ryzen 5 (o equivalente) como mínimo. Para una experiencia más fluida, especialmente cuando se trabaja con proyectos grandes o complejos, un procesador Intel Core i7 o AMD Ryzen 7 es preferible.

Memoria RAM: 8 GB de RAM como mínimo. Se recomienda 16 GB para proyectos de mayor envergadura o cuando se ejecutan múltiples extensiones y herramientas simultáneamente.

Espacio en Disco: 1 GB de espacio libre en disco para la instalación de Visual Studio Code y GitHub Copilot. Sin embargo, es aconsejable disponer de un espacio adicional para proyectos y dependencias.

Sistema Operativo: Windows 10 o superior, macOS 10.14 (Mojave) o superior, o una distribución de Linux compatible (como Ubuntu 18.04 o superior).

b. Comparación de rendimiento en diferentes configuraciones de hardware sistemas operativos

El rendimiento de GitHub Copilot puede variar dependiendo de la configuración del hardware y el sistema operativo:

Windows 10 vs. Windows 11: Los usuarios de Windows 11 pueden experimentar una ligera mejora en el rendimiento debido a las optimizaciones de la nueva versión del sistema operativo y sus características de gestión de recursos.

macOS: En macOS, el rendimiento puede ser muy similar al de Windows, aunque los modelos más nuevos de Mac con Apple Silicon (M1 o M2) pueden ofrecer tiempos de respuesta más rápidos y una mejor eficiencia energética.

Linux: La experiencia en Linux puede variar dependiendo de la distribución y la configuración del entorno de escritorio. En general, los sistemas con hardware más reciente y configuraciones optimizadas pueden ofrecer un rendimiento comparable al de Windows y macOS.

c. Consideraciones sobre el consumo de recursos y la eficiencia

GitHub Copilot está diseñado para ser eficiente en términos de uso de recursos, pero hay algunas consideraciones a tener en cuenta:

Uso de CPU y RAM: Aunque GitHub Copilot no debería consumir una cantidad significativa de CPU o RAM, la extensión puede utilizar recursos adicionales dependiendo del tamaño del proyecto y de la cantidad de código que se está generando. En general, una configuración más potente permitirá una respuesta más rápida y menos interrupciones en el flujo de trabajo.

Red y Conectividad: GitHub Copilot requiere una conexión a Internet estable para comunicarse con los servidores de OpenAI y procesar las solicitudes de completado de código. Las interrupciones en la conectividad pueden afectar la velocidad y la precisión de las sugerencias.

Optimización del Entorno de Desarrollo: Para minimizar el consumo de recursos, es recomendable cerrar otras aplicaciones y procesos que no sean necesarios mientras se utiliza Visual Studio Code. Además, mantener Visual Studio Code y las extensiones actualizadas puede mejorar el rendimiento general.