



# IAs para Programar: Maximizando la Eficiencia

Descubre cómo las IAs como GitHub Copilot y Tabnine están transformando el desarrollo de software. Estas herramientas automatizan tareas, detectan errores y ofrecen sugerencias inteligentes.

## Integrantes:

- Giancarlos Hernaiz
- Andrés Gutiérrez
- Annuar Abouharb

# Selección de la Herramienta de IA

## Evaluación de necesidades

Identificar qué aspectos del desarrollo se quieren optimizar (Autocompletado, Depuración, Refactorización).

1

2

3

## Instalación y configuración

Descargar e integrar la IA en el IDE o entorno de desarrollo preferido

## Comparación de opciones

Analizar herramientas de IA según compatibilidad, costos y funcionalidades.



# Adaptación al Flujo de Trabajo

1

## Familiarización

Prueba las funciones básicas de la IA.

2

## Personalización

Ajusta preferencias (estilo de código, lenguajes).

3

## Pruebas Controladas

Evalúa el rendimiento en proyectos pequeños.

# Uso Cotidiano en Desarrollo



## Sugerencias

Acepta, edita o rechaza propuestas de código.



## Contextualización

Proporciona comentarios para mejorar sugerencias.



## Colaboración

Comparte configuraciones para uniformizar estilos.



# Evaluación y Optimización

## Monitoreo

Identifica si la IA ahorra tiempo.



## Retroalimentación

Reporta falsos positivos/negativos.

## Ajustes

Refina preferencias según la experiencia.

# Manejo de Desafíos

## Mitigar dependencia

Validar manualmente código de IA, evitar sesgo de automatización

## Privacidad y seguridad

Desactivar envío de código sensible a la nube, revisar políticas de datos

# Capacitación y Mejora Continua

## Aprendizaje activo

Documentar casos de éxito/fracaso, participar en comunidades

## Explorar nuevas funcionalidades

Probar integraciones avanzadas de IA

Factores	GitHub Copilot	Tabnine	V0 (Vercel)	Codeium	BlackBox AI
<b>Factores que cumple</b>	Autocompletado contextual avanzado. Soporte para múltiples lenguajes (Python, JS, etc.). Integración nativa con VS Code y JetBrains. Aprendizaje basado en código abierto. Sugerencias en tiempo real.	Autocompletado rápido y preciso. Funciona offline (modelos locales). Soporta más de 20 lenguajes. Personalización básica. Bajo consumo de recursos.	Generación rápida de componentes web (React, Next.js). Integración con Vercel/Next.js. Diseño orientado a prototipado. Uso de IA para UI/UX. Comandos naturales en inglés.	Autocompletado gratuito y multilenguaje. Soporta IDEs populares (VS Code, JetBrains). Aprendizaje contextual. Baja latencia en sugerencias. Opciones de colaboración.	Enfoque en generación de código desde descripciones naturales. Soporta Python, JS, Go. Integración con VS Code. Ideal para tareas repetitivas. Interfaz minimalista.
<b>Factores deseables que podría mejorar</b>	Mayor transparencia en origen de sugerencias. Mejor soporte para arquitecturas complejas. Reducción de sesgos en código generado. Modo offline limitado.	Soporte para proyectos grandes con múltiples archivos. Más personalización de estilos. Explicación clara del uso de datos. Compatibilidad con más IDEs.	Limitado al ecosistema JavaScript/Next.js. Poca flexibilidad para backends complejos. Documentación escasa. Sin modo offline.	Mejor comprensión de contextos complejos. Soporte para más frameworks. Mayor transparencia en modelos de IA. Optimizar el consumo de RAM.	Poca precisión en lenguajes menos populares. Sin integración con otros IDEs. Sugerencias genéricas en código crítico. No soporta colaboración en equipo.

# Aspectos Clave del Modelo de IA

- 1 Autocompletado Inteligente
- 2 Personalización y Configuración
- 3 Análisis de Código en Tiempo Real
- 4 Eficiencia y Rendimiento
- 5 Aprendizaje Continuo
- 6 Seguridad y Privacidad
- 7 Integración con Herramientas de Desarrollo
- 8 Soporte para Proyectos Complejos
- 9 Colaboración en Equipo
- 10 Interfaz de Usuario Intuitiva
- 11 Documentación y Explicaciones
- 12 Retroalimentación y Mejora Continua

# IA para programar

## Autocompletado Inteligente

Predice líneas de código completas usando modelos entrenados en código público. Ideal para aprender sintaxis de nuevos lenguajes o frameworks sin consultar documentación constantemente.

## Generación de Código Contextual (Code Completion)

Sugiere código en tiempo real basado en el contexto del proyecto. Reduce tiempo en tareas repetitivas (Ej: crear funciones CRUD) y mejora la productividad en prototipos.

### Integrantes:

Giancarlos Hernaiz 30.747.059

Andrés Gutierrez 30.662.790

Annuar Abouharb 30.350.953

## Integración con IDEs y Editores

Funciona dentro de entornos como VSCode, PyCharm o JetBrains sin configuración compleja, incluso ya viniendo por defecto con otros nuevos editores como Cursor, Trae y Windsurf. Elimina la necesidad de cambiar de ventana, manteniendo el flujo de trabajo concentrado.

## Asistencia en Frameworks y Librerías

Guía en el uso de herramientas específicas (React, TensorFlow, etc.) con ejemplos integrados. Reduce la curva de aprendizaje en tecnologías complejas.

## Recomendaciones de Mejores Prácticas

Sugiere estándares de código y patrones de diseño según el contexto. Ayuda a adoptar convenciones profesionales (Ej: Principios SOLID, Clean Code, etc.) desde etapas tempranas.

## Análisis de Código en Tiempo Real

Detecta errores lógicos, vulnerabilidades o código ineficiente. Funciona como un "tutor automatizado" que explica bugs y ofrece soluciones específicas.

# Especificación de requerimientos funcionales y no funcionales

## Requerimientos Funcionales

- **Generación de código:** Generación de funciones completas con  $\geq 90\%$  precisión
- **Análisis de código:** Detección de vulnerabilidades OWASP Top 10 con  $\leq 5\%$  falsos positivos
- **Personalización:** Permitir configurar de estilo de código por proyecto
- **Autocompletado:** Implementación de sugerencias de código en tiempo real para acelerar el desarrollo
- **Integración:** Soporte para integración con entornos de desarrollo integrados (IDE) comunes

## Requerimientos No Funcionales

- **Rendimiento:** Latencia  $\leq 800\text{ms}$  en proyectos  $\leq 50.000$  líneas
- **Seguridad:** Cifrado AES-256 para código en la nube
- **Compatibilidad:** Soporte para los lenguajes de programación más usados (Python, JavaScript, Java, etc.).
- **Escalabilidad:** Estructura para soportar múltiples usuarios concurrentes sin degradación del rendimiento
- **Disponibilidad:** Tiempo de actividad garantizados al 99.9%



# Accesibilidad

- 1 Cumplimiento de estándares de accesibilidad web WCAG 2.0
- 2 Opciones de navegación alternativas para personas con discapacidades visuales o motoras
- 3 Soporte para lectores de pantalla y teclados especializados
- 4 Modos adaptativos con explicaciones paso a paso

# Requisitos Éticos Según la UNESCO

**1** **Respeto a Derechos Humanos:**

Código libre de discriminación, violencia y violaciones.

**3** **Transparencia Radical:**

Documentación pública de datos, limitaciones y decisiones.

**5** **Control Humano:**

Revisión obligatoria antes de ejecutar sugerencias IA.

**2** **Inclusión y Equidad:**

Datos diversos y acceso asequible globalmente.

**4** **Sostenibilidad Ambiental:**

Optimización para reducción de huella de carbono.



# Validación y Criterios de Aceptación



## Técnicas de Verificación

- Pruebas técnicas
  - Benchmarking de rendimiento y pruebas de estrés
- Validación con usuarios
  - Pruebas con equipos de desarrollo y sesiones de usabilidad
- Auditorías
  - Revisión de seguridad del código y evaluación de ética



## Criterios de Aceptación

- **Funcionalidad:** 85% sugerencias útiles (con una tolerancia de  $\pm 5\%$ )
- **Rendimiento:** 95% respuestas en menos de 1 segundo
- **Accesibilidad:** 100% cumplimiento WCAG AA
- **Ética:** Sin violaciones éticas



## Documentación Final

- Especificación de Requisitos de Software (SRS)
- Plan de Pruebas detallado
- Manual de Implementación Ética