

PAIR PROGRAMMING

IA

Sebastian Andres Rodriguez

Juan Diego Arias





TABLA DE CONTENIDOS

• Motivación y contexto	01
• ¿Qué es pair programming clásico?	02
• ¿Qué es pair programming con IA?	03
• Características clave del enfoque	04
• Historia y evolución	05
• Ventajas del pair programming con IA	06
• Desventajas y retos	07
• Casos de uso	08
• casos reales de industria	09
• Mercado laboral y Stack ampliado con IA	10
• Evolución de capacidades de los modelos	11
• Vibe coding vs pair programming con IA	12
• Problemas del Vibe Coding	13



MOTIVACIÓN Y CONTEXTO

Empresas colombianas ya exigen habilidades en IA para contratar, según estudio de Deel

El bajo compromiso de los colaboradores y las restricciones presupuestarias, no dejan avanzar en esta tecnología.

- Empresas colombianas ya exigen habilidades en IA.
- IA empieza a ser parte del “stack” mínimo de un desarrollador.
- ¿Se volvió una necesidad aprender a usar las herramientas que ofrece la IA generativa?

The image shows a job listing for a "Software Developer Semi Senior" position at TW Group. The listing includes the following details:

- Location: Remoto
- Experience: Semi Senior
- Type: Full time
- Field: Programación
- Salary: Bruto \$1600 - 1900 USD/mes
- Postulations: 114 postulaciones
- Response Time: Responde entre 1 y 9 días
- Last Check: Revisado por última vez hoy

Below the details are buttons for "Postular" (Apply) and "Compartir" (Share), along with links to share via Email, LinkedIn, WhatsApp, and Imagen. A note indicates that the application requires Spanish.

Buscamos un/a Ingeniero/a de Desarrollo de Software Semi Senior para construir soluciones B2B de alto impacto en proyectos críticos de infraestructura y gestión documental empresarial.

Desarrollarás plataformas para importantes empresas donde la disponibilidad, seguridad y datos en tiempo real son fundamentales. Integrarás múltiples APIs externas, automatizarás procesos críticos, gestionarás documentación en Google Workspace, implementarás capacidades de IA en productos reales, y soportarás decisiones estratégicas de nuestros clientes.

Stack moderno: Laravel, Vue, MySQL, Azure/GCP, Docker, Git, Jira + IA aplicada al desarrollo (Cursor, Claude Code) e integrada en soluciones (Gemini, OpenAI, Claude).

QUÉ ES PAIR PROGRAMMING CLÁSICO?

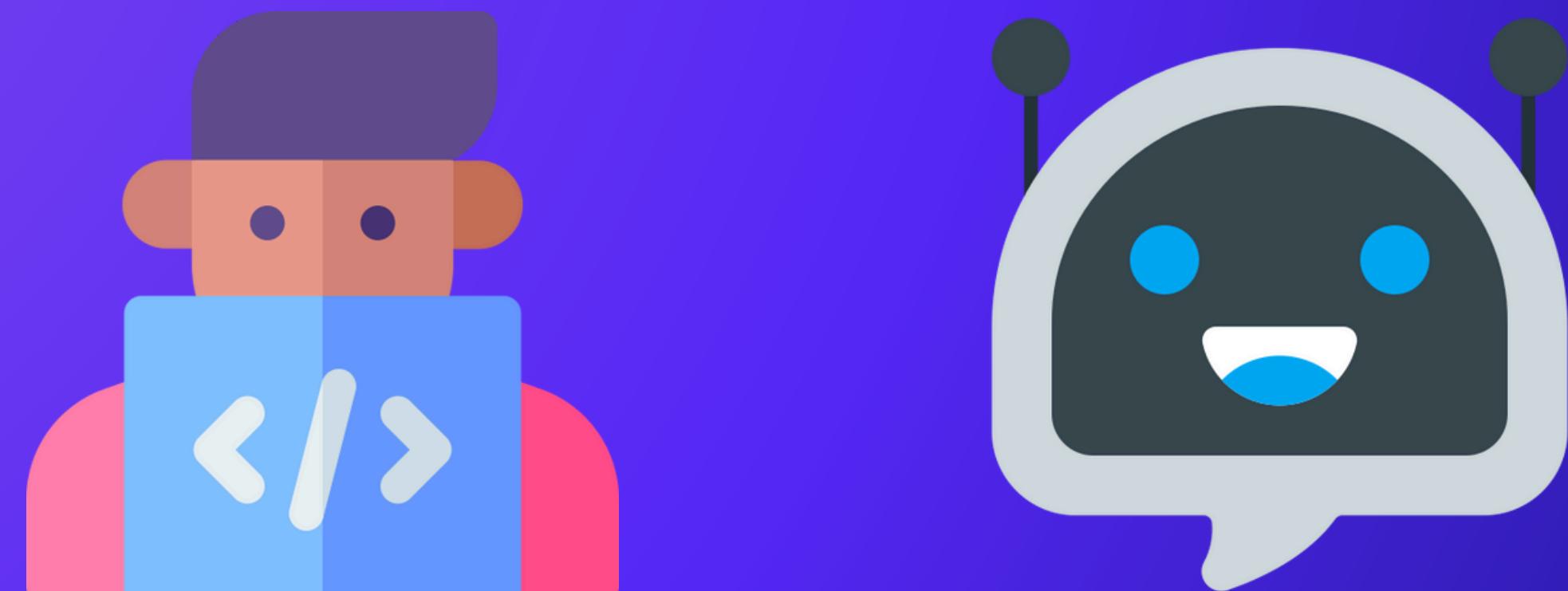


conductor
escribe el código.

Feedback continuo!

Navegante
revisa y piensa en la
estrategia.

QUÉ ES PAIR PROGRAMMING CON IA?



Colaboración iterativa!

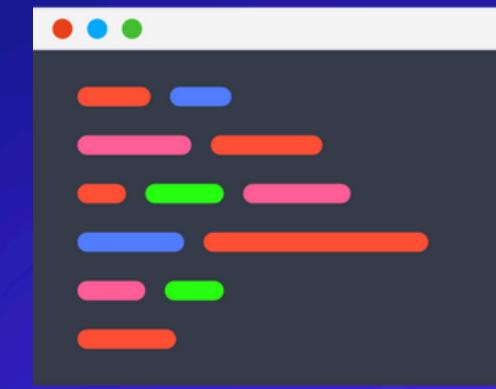
Desarrollador
Describe intención,
revisa, decide

Asistente de IA
Sugiere código, explica,
propone tests

CARACTERÍSTICAS CLAVE



Sugerencias contextuales en tiempo real.



Integración en el IDE



Prompt engineering



Colaboración iterativa y feedback inmediato

MODELO

LLMs que
reciben texto y
devuelven el
texto (ej GPT)



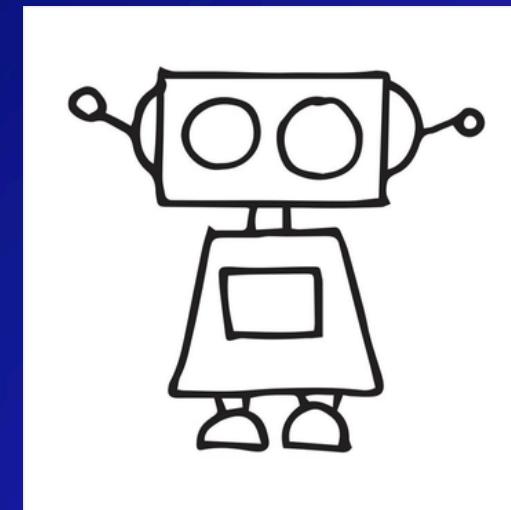
HERRAMIENTA

Es la integracion con
el IDE y consume los
modelos (ej Copilot)



MODOS

Agente



Puede editar el código,
archivos, siguiendo una orden

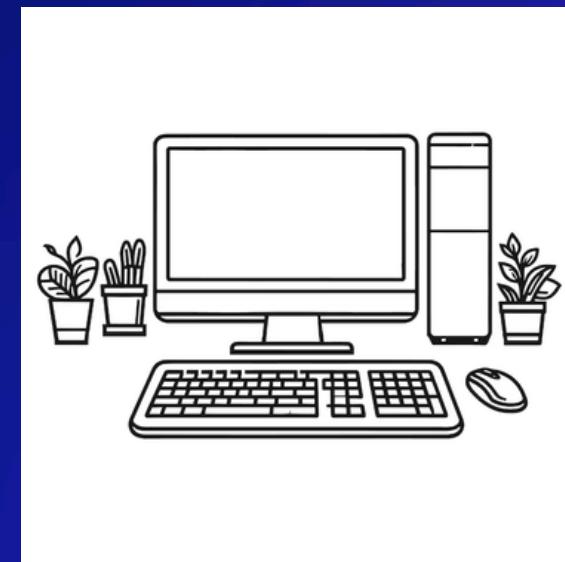
Pregunta



Puede consultar y leer
respondiendo preguntas

ENTORNOS

Local



Esta consulta los archivos del
proyecto de tu IDE

Remoto (Repositorio)



La herramienta trabaja desde
un repositorio haciendo PR

HISTORIA Y EVOLUCION



- 2018: TabNine – primer completador con redes neuronales.**
- 2021: GitHub Copilot – “AI pair programmer”.**
- 2022: CodeWhisperer, Codeium.**
- 2022–23: ChatGPT.**
- 2023–24: Claude 3, editores AI-native (Cursor).**
- 2024–25: modelos de razonamiento (o1, o3), GPT-5.**



VENTAJAS

INCREMENTO DE VELOCIDAD Y PRODUCTIVIDAD

El beneficio mas inmediato es la aceleración en la escritura de código, ahorrando tiempo especialmente en tareas repetitivas

APRENDIZAJE ACELERADO

La IA generativa ofrece recursos personalizados y retroalimentación inmediata, adaptando el contenido al ritmo y necesidades de cada estudiante.

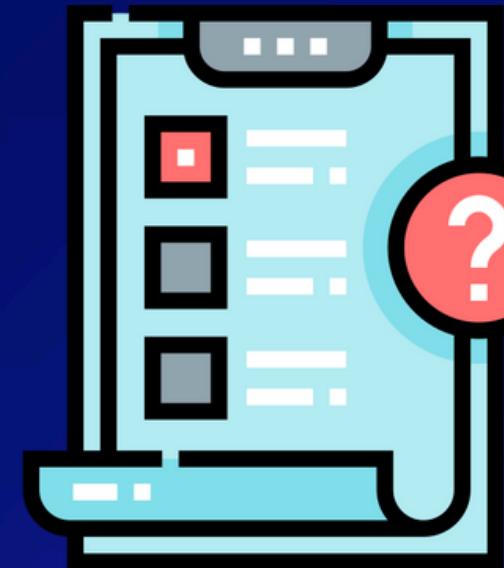
ENFOQUE EN TAREAS DE MAYOR NIVEL = ALIVIO DEL TRABAJO TEDIOSO

Delegar a la IA la escritura de código repetitivo o “mecánico” permite que el desarrollador humano concentre mas su tiempo y energía en aspectos de alto nivel, creativos o críticos del sistema .

DISPONIBILIDAD PERMANENTE

A diferencia de un colega humano, la IA esta disponible las 24 horas, todos los días.

DESVENTAJAS Y RETOS



DEPENDENCIA EXCESIVA

Existe el riesgo de que los desarrolladores se acostumbren a aceptar sugerencias de la IA sin un análisis critico profundo.

FALSOS POSITIVOS

En la practica, se han observado situaciones donde la IA sugiere con mucha seguridad una solución que resulta ser incorrecta.

RIESGOS DE PRIVACIDAD Y SEGURIDAD DEL CODIGO

Muchas herramientas de IA envían fragmentos de código a servidores en la nube, lo que genera preocupaciones sobre la confidencialidad del código propietario.

NECESIDAD DE APRENDER A USAR LA IA

Irónicamente, aunque la IA simplifica muchas tareas, para utilizarla bien hay que adquirir nuevas habilidades.

CASOS DE USO

CÓDIGO REPETITIVO Y BOILERPLATE

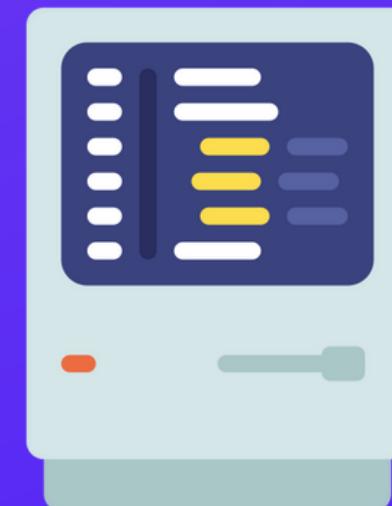


- Genera controladores, DTOs, repositorios a partir de unos pocos ejemplos.
- Mantiene consistencia de estilo y nombres en todo el proyecto.



PRUEBAS UNITARIAS Y CASOS DE PRUEBA

- Propone tests a partir de la firma y el cuerpo de una función.
- Sugiere casos borde que muchas veces se olvidarían (0, negativos, nulos).



REFACTORIZACIÓN Y MODERNIZACIÓN DE CÓDIGO LEGACY

- Sugiere patrones para migrar código antiguo a frameworks modernos.
- Repite un mismo patrón de refactor en múltiples módulos.
- Ayuda a identificar funciones duplicadas o APIs obsoletas.

CASOS REALES DE INDUSTRIA

NARANJA X (FINTECH)



- Desarrolladores reportan ahorro de 2-3 horas de trabajo al día con Copilot.
- ~50 % menos tiempo para resolver ciertos problemas técnicos.
- Permite manejar más proyectos en paralelo sin sacrificar calidad.

FUTURE PROCESSING (EMPRESA DE SOFTWARE)



- +34 % de velocidad en código nuevo y +38 % en pruebas unitarias.
- 96 % de devs dicen que Copilot acelera su trabajo diario.
- Migración Angular→React ~40 % más rápida de lo estimado.

BANCO GALICIA (SECTOR BANCARIO)



- Usan Copilot para implementar una funcionalidad regulatoria en tiempo récord.
- Equipo logra entregar cambios críticos entre viernes y lunes*.
- La propia empresa reconoce que sin IA "habría sido muy improbable" llegar a tiempo.

EL VIBE CODING Y SU MERCADO LABORAL

Jonathan Lin · 2º
Overengineering & Vibe Coding Cleanup Specialist
Territorio Federal de Kuala Lumpur, Malasia
1 mil seguidores · Denis Lin es contacto en común

Hamza Gulraiz · 2º
Vibe coding cleanup specialist | React Native Developer | Mobile Innovation Enthusiast
Lahore
Ofrece servicios: Desarrollo de aplicaciones móviles, Desarrollo web, Desarrollo de iOS, Pruebas de software, Desarrollo de Android

Nishant Gohel · 2º
Vibe coding cleanup specialist
Bengaluru
22 mil seguidores · Mani Kiran Nalabolu (He/Him) y Muhammad Farooq Khalid son contactos en común

Wilmer Mauricio Herrera Renteria · 2º
Vibe Coding Fixer | Software Architect | MSc in AI
Quito
Anterior: SugarCRM Developer Specialist en Casabaca S.A.
Ofrece servicios: Desarrollo web

Victor Istrati · 2º
Senior Web Engineer | Vibe Coding Cleanup Specialist
Chisinau, Moldavia
Actual: Vibe Coding Cleanup Specialist en Freelance
Victoria Negara es contacto en común

AI Vibe Coder
Prime Works LLC · Colombia
Hace 1 semana · 170 solicitudes

Descubre a quién ha contratado Prime Works LLC para este puesto

Solicitar **Guardar**

You will be working for a US-based technology company leading the development of an Ecommerce platforms that supports independent retailers in North America as they adopt and embrace new components of their digital marketing solutions. We build and maintain thousands of retailer websites that must run reliably, efficiently, and effectively.

We are looking for a collaborator who can execute fast, iterate with us, and help turn ideas into usable products—apps, browser extensions, internal dashboards, micro-SaaS tools, etc.

We only want the best and if you are looking to work with a fast growing and stable American company then this position is for you.

Responsibilities:

- Convert product ideas + user stories into working prototypes quickly.
- Build small full-stack apps, internal tools, browser extensions, and lightweight APIs.
- Work collaboratively in short feedback loops (quick calls when needed, async most of the time).
- Keep code organized so projects can scale (no spaghetti, minimal tech debt).
- Pay attention to security, privacy, and separation of concerns.

PRINCIPIOS SOLID VS. PAIR PROGRAMMING CON IA

Principio SOLID	Impacto	Relación con pair programming con IA
Responsabilidad Única (SRP)	Alto	La IA puede sugerir clases o funciones demasiado grandes si el prompt es ambiguo. El desarrollador debe guiarla para mantener responsabilidades claras y dividir el código en unidades coherentes.
Abierto/Cerrado (OCP)	Medio-Alto	Los copilotos facilitan extender comportamientos mediante nuevas clases o estrategias, pero también pueden inducir a modificar código existente de forma rápida. Se requiere disciplina para preferir extensiones sobre cambios destructivos.
Sustitución de Liskov (LSP)	Medio	La IA puede generar jerarquías de herencia sin respetar completamente las precondiciones o invariantes. El par humano debe revisar que las subclases mantengan contratos válidos y que los tests cubran estos casos.
Segregación de Interfaces (ISP)	Medio	Al generar interfaces a partir de descripciones en lenguaje natural, la IA tiende a agrupar muchas operaciones juntas. Un buen uso del pair programming con IA implica refinar esas interfaces hacia versiones más específicas y cohesivas.
Inversión de Dependencias (DIP)	Alto	Los asistentes favorecen patrones de inyección de dependencias y abstracciones reutilizables cuando se les guía correctamente, pero también pueden acoplar directamente a implementaciones concretas si el desarrollador no lo supervisa.

ATRIBUTOS DE CALIDAD VS. PAIR PROGRAMMING CON IA

Atributo de calidad	Impacto	Efecto del pair programming con IA
Mantenibilidad	Alto	La IA puede homogenizar estilos y extraer funciones reutilizables, facilitando refactorizaciones frecuentes. Sin embargo, si se acepta código sin entenderlo, se incrementa la deuda técnica y se dificulta el mantenimiento futuro.
Rendimiento	Medio	Los modelos no siempre optimizan por eficiencia; pueden proponer soluciones correctas pero subóptimas. Es rol del humano identificar cuellos de botella y orientar a la IA a usar estructuras y algoritmos más eficientes.
Seguridad	Alto	Los asistentes pueden sugerir código inseguro (por ejemplo, consultas SQL sin sanitizar o validación insuficiente). En pair programming, el desarrollador debe revisar este aspecto con especial atención, complementando con análisis estático y pruebas de seguridad.
Usabilidad del código (legibilidad, claridad)	Alto	Cuando se les guía con buenos prompts, los asistentes tienden a generar nombres consistentes, comentarios y documentación básica, mejorando la legibilidad. Mal usados, pueden introducir código verbose o confuso.
Confiabilidad	Medio-Alto	La generación de pruebas unitarias y casos borde incrementa la probabilidad de detectar errores temprano. No obstante, la confiabilidad final depende de la integración de esas pruebas en el ciclo CI/CD y de la revisión humana del oráculo de cada test.

ANÁLISIS DE TÁCTICAS VS. PAIR PROGRAMMING CON IA

Táctica arquitectónica	Impacto	Relación con el tema
Encapsulación y separación de responsabilidades	Alto	El uso de IA para refactorizar facilita extraer componentes y servicios con límites más claros; el desarrollador puede pedir explícitamente “extraer este módulo” o “separar responsabilidades”, acelerando la encapsulación.
Introducción de capas y puntos de extensión	Medio-Alto	La IA ayuda a generar adaptadores, interfaces y capas de servicio, reforzando tácticas de modifiabilidad. Sin embargo, requiere que el arquitecto exprese estos puntos de extensión en los prompts y en la estructura inicial.
Tácticas de testabilidad (inyección de dependencias, logs, aserciones)	Alto	Los asistentes pueden insertar pruebas, mocks y trazas en puntos clave del sistema, incrementando la observabilidad y la facilidad de prueba, sobre todo cuando se combinan con pipelines automatizados.
Tácticas de rendimiento (caching, uso eficiente de recursos)	Medio	La IA puede sugerir patrones de caching comunes, pero no tiene visibilidad directa del entorno de producción. Es necesario complementar con métricas y perfiles de rendimiento reales para decidir qué tácticas aplicar.
Tácticas de seguridad (validación, autenticación, manejo de errores)	Medio-Alto	Los modelos conocen bibliotecas y buenas prácticas de seguridad, pero pueden omitir controles en contextos específicos. El par humano debe validar que las tácticas propuestas sean consistentes con las políticas de seguridad de la organización.

ANÁLISIS DE PATRONES VS. PAIR PROGRAMMING CON IA

Patrón	Impacto	Relación con el tema
Patrones de diseño (Strategy, Factory, Adapter, Observer, etc.)	Alto	La IA reconoce muchos de estos patrones y puede generar implementaciones estándar a partir de descripciones (“usar Strategy para seleccionar algoritmo de pago”). El riesgo es aplicarlos mecánicamente sin analizar si el patrón realmente es necesario.
Arquitectura en capas (Layered)	Medio-Alto	Los copilotos facilitan separar controladores, servicios y repositorios repitiendo el mismo esqueleto de capa. Esto refuerza la claridad de responsabilidades, siempre que el equipo mantenga la disciplina de no “saltar capas” en las integraciones.
Arquitecturas orientadas a servicios / microservicios	Medio	La IA puede generar esqueletos de servicios, APIs y contratos entre microservicios. Sin embargo, las decisiones de particionado, límites de contexto y manejo de datos distribuidos siguen siendo tareas críticas del arquitecto humano.
Pipes and Filters / Data pipeline	Medio	Los asistentes son útiles para encadenar transformaciones de datos (por ejemplo, pipelines de ETL o procesamiento de logs). Aun así, se requiere un diseño explícito del flujo y de los puntos de observabilidad.
Patrones de observabilidad (logging centralizado, métricas, tracing)	Alto	Con buenos prompts, la IA puede insertar puntos de log, métricas y trazas de forma sistemática en el código, facilitando la instrumentación necesaria para monitorear sistemas complejos.

ANÁLISIS DE MERCADO LABORAL VS. PAIR PROGRAMMING CON IA

Perfil del mercado	Demanda de IA	Implicaciones del pair programming con IA
Desarrollador junior	Alta	Se espera que domine al menos un copiloto de código y pueda integrarlo en su flujo diario. El pair programming con IA se vuelve parte de su “stack mínimo”, además de los lenguajes y frameworks básicos.
Desarrollador mid-level / senior	Muy alta	Debe combinar destreza técnica con capacidad de supervisar y corregir el código generado por la IA, guiando a perfiles junior y participando en decisiones de arquitectura donde la IA es un componente activo.
Arquitecto/a de software	Muy alta	Se espera que defina lineamientos de uso de IA, seleccione herramientas y modelos, y diseñe procesos de pair programming con IA que respeten los atributos de calidad del sistema y las políticas de la organización.
Orquestador de IA / Prompt engineer	Alta y creciente	Su rol gira explícitamente en torno a diseñar prompts, flujos de herramientas y evaluaciones de calidad para interacciones humano–IA. El pair programming con IA es su contexto natural de trabajo.
Especialista en plataformas de IA (MLOps / LLMOps)	Alta	Debe ofrecer entornos seguros y eficientes para que los equipos de desarrollo puedan usar modelos de IA como “pares” de programación, cuidando aspectos de coste, seguridad, latencia y cumplimiento normativo.

MERCADO LABORAL

IA YA ES PARTE DEL STACK, NO SOLO “HABILIDAD EXTRA”

Desarrollador Cursor IA

SCOFT TECHNOLOGY

Córdoba, Cordoba, Argentina

Tecnología, Sistemas y Telecomunicaciones/Tecnologías de la Información

Full-time Híbrido

Postularse

Hace 15 días

Descripción del puesto

¡Subite a la nave de **SCOFT Technology!** 🚀🚀

Queremos que vengas a desafiar y sumar tu talento a una empresa en pleno crecimiento. Nos encontramos en la búsqueda de un Desarrollador Cursor IA. Participaras **de forma práctica en el desarrollo** para acelerar entregas, elevar la calidad del código y modelar estándares técnicos del equipo.

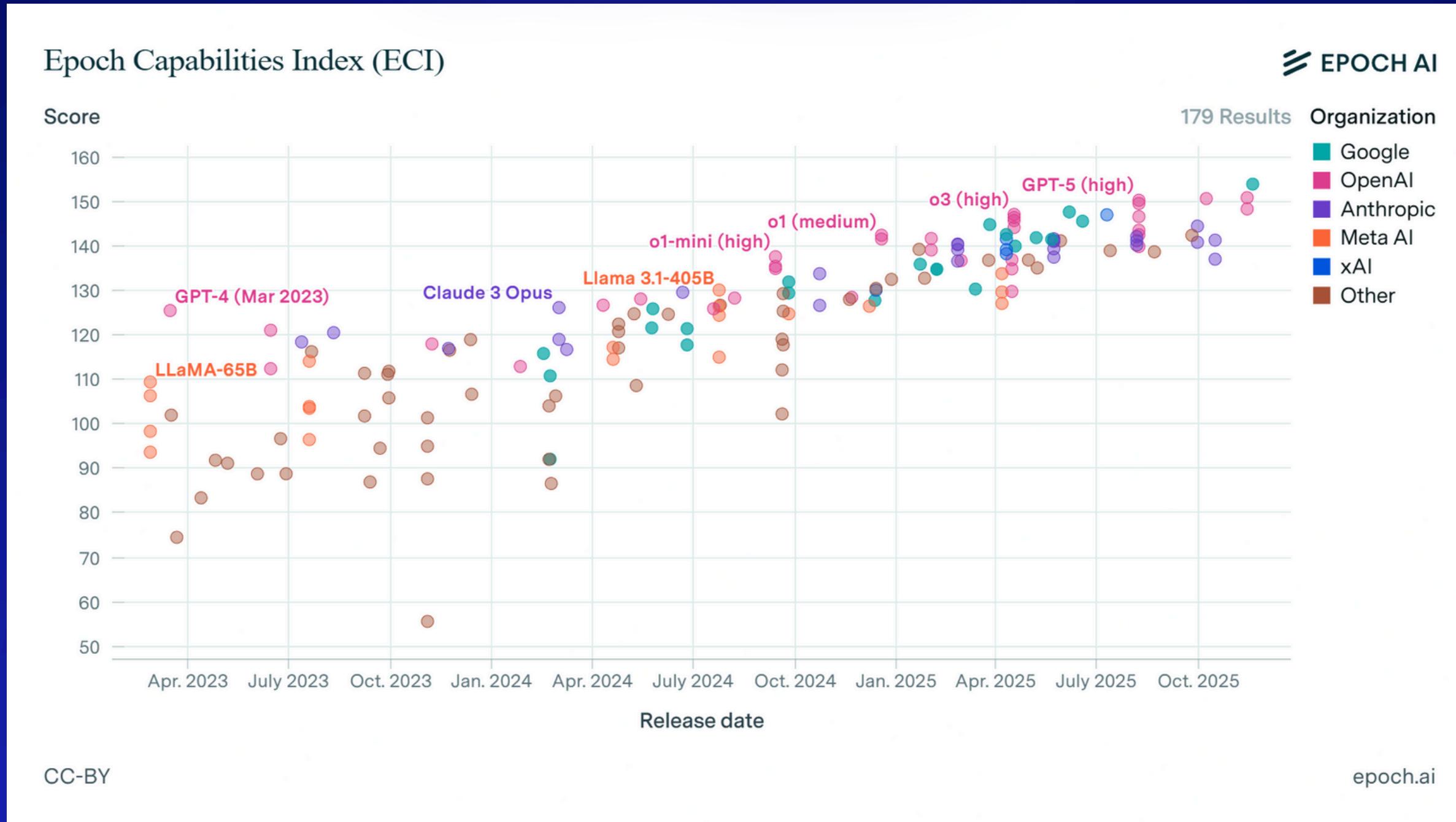
Nosotros

Somos un equipo de profesionales en **Desarrollos Ágil de Software** de última generación. Nuestros servicios son la

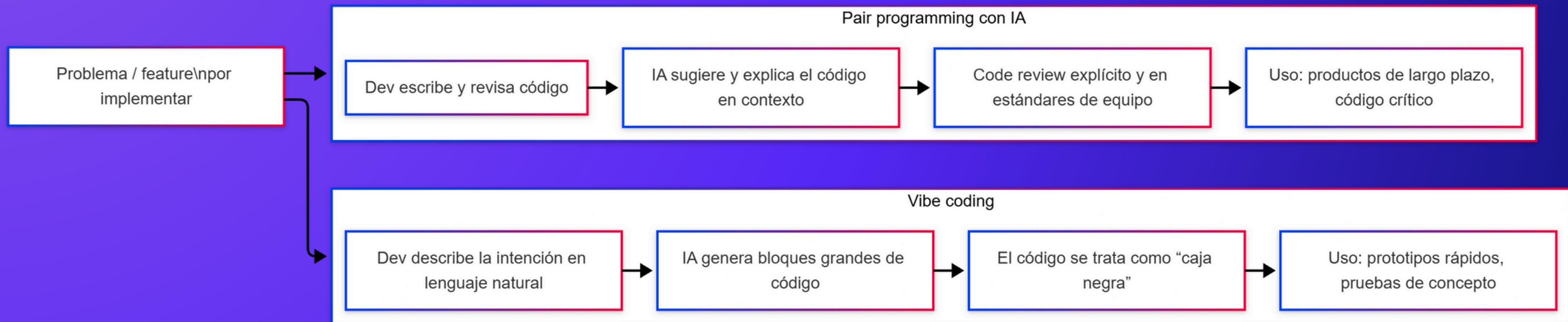
B (BASICA), M (MEDIA) Y A (AVANZADA).

Cuadro I MATRIZ DE PERFILES Y DEMANDA DE HABILIDADES RELACIONADAS CON IA EN EQUIPOS DE DESARROLLO				
Perfil	Alfabetización	Pair prog. con IA	Orquestación de IA	Gobernanza / Ética
Desarrollador junior	B	M	B	B
Desarrollador mid-level	M	M	M	M
Desarrollador senior / TL	M	A	A	M
Arquitecto / Plataformas IA	M	A	A	A

EVOLUCIÓN DE CAPACIDADES DE LOS MODELOS



VIBE CODING VS PAIR PROGRAMMING CON IA



PERO.. ¿QUE ES VIBE CODING?

ES PROGRAMAR DE FORMA MUY FLUIDA E INTUITIVA, DEJANDO QUE LAS IDEAS SALGAN RÁPIDO SIN PREOCUPARSE DEMASIADO AL INICIO POR LA ESTRUCTURA PERFECTA, LAS BUENAS PRÁCTICAS O LA ARQUITECTURA, PARA LUEGO IR PULIENDO EL CÓDIGO

PROBLEMAS DEL VIBE CODING

MANTENIBILIDAD Y LEGIBILIDAD

Código generado en sesiones de vibe coding tiende a ser difícil de mantener, ya que suele carecer de una arquitectura explícita y de documentación sistemática

SEGURIDAD Y CUMPLIMIENTO

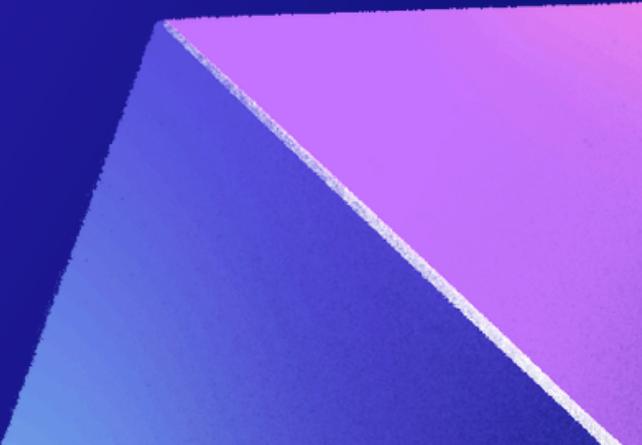
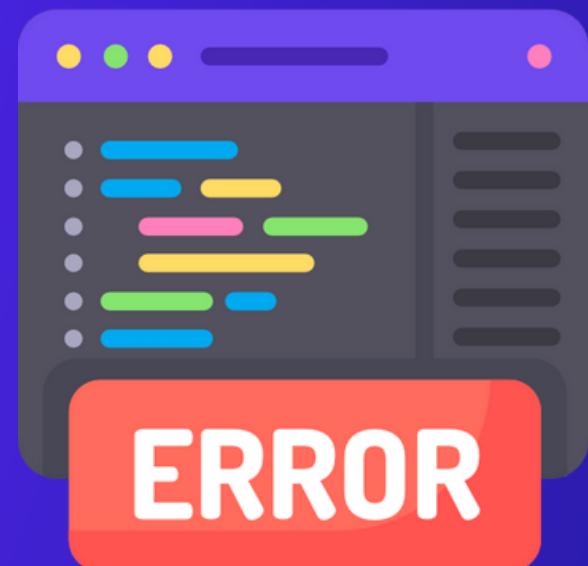
Aceptar código “por vibra” sin revisión rigurosa puede introducir vulnerabilidades graves, uso incorrecto de librerías criptográficas, manejo inseguro de datos sensibles, etc.

GOBERNANZA Y RESPONSABILIDAD

Si el equipo no entiende el código que ejecuta, se difumina la responsabilidad ante fallos o incidentes.

EROSION DE HABILIDADES FUNDAMENTALES

Si los desarrolladores se acostumbran a no leer ni comprender el código, existe el riesgo de perder (o nunca desarrollar) competencias básicas en algoritmos, estructuras de datos y diseño de software.



COSTOS ENTRE PAIR PROGRAMMING CON IA Y VIBE CODING

Dimensión de costo	Pair programming con IA	Vibe coding
Tokens y cómputo de IA	Consumo moderado y granular; prompts más pequeños, centrados en el archivo actual.	Consumo alto; prompts largos, respuestas masivas y posibles agentes que recorren todo el repositorio.
Tiempo humano inicial	Mayor tiempo de revisión, diseño y pruebas por funcionalidad.	Menor tiempo para llegar a “algo que funciona” (prototipo).
Deuda técnica y retrabajo futuro	Menor; el código tiende a ser más aliñeados con la arquitectura y principios de diseño.	Mayor; riesgo de reescrituras, bugs ocultos y refactorizaciones complejas.
Costo total en proyectos de largo plazo	Generalmente menor: más caro al inicio, pero más barato en mantenimiento.	Generalmente mayor: barato al iniciar, caro al sostener y escalar.
Adecuación al contexto	Ideal para productos críticos, regulados o de vida larga.	Adecuado para prototipos, experimentos y validación rápida de ideas.

CONCLUSIONES Y LECCIONES APRENDIDAS

PAIR PROGRAMMING CON IA NO ES SOLO USAR UN AUTOCOMPLETADO MÁS AVANZADO, SINO CAMBIAR CÓMO SE ORGANIZA Y DISEÑA EL DESARROLLO.

TRATAR A LA IA COMO UNA AUTÉNTICA PAREJA DE PROGRAMACIÓN, CON ROLES CLAROS Y FEEDBACK CONSTANTE, PUEDE MEJORAR PRODUCTIVIDAD, CALIDAD DEL CÓDIGO Y APRENDIZAJE. PERO ESTOS BENEFICIOS SOLO APARECEN CUANDO HAY DISEÑO, REVISIÓN Y VALIDACIÓN HUMANA.

ACELERA EL INICIO DE UN PROYECTO, PERO NO REEMPLAZA LA DISCIPLINA DE INGENIERÍA NECESARIA PARA MANTENERLO EN EL TIEMPO.

EL MERCADO LABORAL VALORA CADA VEZ MÁS A QUIENES COMBINAN BUENA BASE TÉCNICA CON CAPACIDAD DE ORQUESTAR SISTEMAS DE IA, EVALUAR SUS SALIDAS CRÍTICAMENTE Y COMUNICAR DECISIONES ASISTIDAS POR IA.

MUCHAS VECES, UN ENFOQUE UN POCO MÁS LENTO PERO MÁS GUIADO (PAIR PROGRAMMING CON IA) RESULTA MÁS BARATO A MEDIO PLAZO QUE GENERAR MUCHO CÓDIGO RÁPIDO Y LUEGO TENER QUE REESCRIBIR O REFACTORIZAR.

EL PAIR PROGRAMMING CON IA Y EL VIBE CODING NO SON ENFOQUES EXCLUYENTES, SINO HERRAMIENTAS COMPLEMENTARIAS DENTRO DE UN MISMO ECOSISTEMA DE DESARROLLO ASISTIDO POR IA.

DEMOSTRACION



GRACIAS!

