# Vibe coding

En los últimos años, el desarrollo de software ha comenzado a experimentar un cambio profundo gracias al avance de las inteligencias artificiales generativas. En ese contexto surge la idea de Vibe Coding, un enfoque que propone una nueva forma de crear aplicaciones, no desde el código tradicional, sino a través de prompts: instrucciones escritas en lenguaje natural que se comunican con una IA para generar partes o incluso sistemas completos de software. Vibe Coding no es una metodología formal aún, pero representa una tendencia creciente: programar sin necesidad de ser programador. La idea central es simple pero potente. En lugar de escribir líneas de código en un editor, el desarrollador,o incluso una persona sin experiencia técnica, describe lo que quiere lograr, como si estuviera explicándoselo a un colega. Por ejemplo: “Quiero una aplicación móvil que permita registrar los gastos diarios y que muestre un gráfico de resumen mensual”. Esa descripción se convierte en un prompt que una IA como ChatGPT, GitHub Copilot o herramientas similares interpreta para generar automáticamente el código correspondiente. En algunos casos, incluso puede crear toda una estructura base del proyecto, incluyendo pantallas, lógica y datos simulados.

Este enfoque tiene múltiples ventajas. Primero, democratiza la programación: muchas personas que antes no podían crear software por falta de conocimientos técnicos ahora tienen una vía de acceso más directa. Además, acelera el proceso de desarrollo, especialmente en las primeras etapas, como cuando se quiere hacer un prototipo rápido para validar una idea. También promueve la creatividad, porque permite centrarse en el “qué” en lugar del “cómo”, al menos en una primera instancia. En educación, puede ser una herramienta poderosa para que los estudiantes se concentren en la lógica y el diseño de soluciones, sin quedar atrapados por la frustración que muchas veces genera la sintaxis de los lenguajes de programación.

Obviamente, no todo es perfecto. Usar Vibe Coding requiere cierta comprensión básica del desarrollo para poder revisar y corregir el código generado. Las IAs aún cometen errores o generan soluciones que funcionan, pero que no son del todo eficientes o seguras. También existe el riesgo de que las personas se vuelvan dependientes de estas herramientas sin comprender lo que hay detrás. Pero como ocurre con cualquier tecnología emergente, estos desafíos se pueden abordar con una combinación de criterio humano, formación crítica y buenas prácticas.

Hoy en día, hay muchas plataformas que facilitan esta forma de trabajo de forma genérica: ChatGPT con funciones de codificación, GitHub Copilot integrado a Visual Studio Code, Replit con su asistente Ghostwriter, y herramientas emergentes como Cursor.sh o Builder.io, que permiten crear sitios web enteros a partir de descripciones en lenguaje natural. Algunas soluciones más avanzadas como LangChain incluso permiten construir sistemas conversacionales o apps con lógica compleja sin escribir todo el código desde cero.

No obstante, esta investigación se centrará en herramientas que permitan generar programas en los lenguajes Python y C++, específicamente orientados al desarrollo con programación concurrente.

Al mismo tiempo, se hará foco en herramientas de inteligencia artificial que permitan generar aplicaciones desarrolladas con el framework Android Studio, con el objetivo de crear soluciones para smartphones y smartwatches que funcionen bajo el sistema operativo Android. En este sentido, se trabajará con los lenguajes Java y Kotlin. Se opta por Android Studio en lugar de otras alternativas (como React Native, JavaScript, TypeScript, Flutter, entre otras), debido a que permite generar código nativo, lo que resulta en un funcionamiento más eficiente y un mejor aprovechamiento de los recursos del sistema operativo Android. En contraste, las soluciones multiplataforma mencionadas funcionan como una capa de abstracción adicional, lo cual puede impactar negativamente en el rendimiento.

Por otra parte, se planifica utilizar herramientas de Vibe Coding para la creación de aplicaciones embebidas utilizando placas de desarrollo ESP32. Para este caso, se empleará el framework de Arduino, a través de su lenguaje Wiring, descartando el uso del entorno ESP-IDF. Esta elección responde a la mayor practicidad del entorno Arduino para el desarrollo de proyectos de IoT en contextos educativos. Además, los programas creados para el ESP32 utilizarán el sistema operativo de tiempo real FreeRTOS, lo cual permitirá desarrollar y contrastar aplicaciones embebidas que utilizan sistema operativo con aquellas que no lo requieren.

En resumen, la investigación se enfocará en herramientas de Vibe Coding que permitan desarrollar programas con las siguientes características:

* Aplicaciones concurrentes en Python y C++.
* Aplicaciones móviles para Android, desarrolladas en Android Studio y compatibles con smartphones y smartwatches.
* Aplicaciones para sistemas embebidos con ESP32, utilizando el framework de Arduino junto a FreeRTOS.
* Por este motivo, se seleccionarán herramientas de Vibe Coding que se ajusten a estos objetivos.

# Generación de Prompts para Vibe coding

En el marco de la programación asistida por inteligencia artificial, Vibe Coding se consolida como una metodología centrada en la creación de software mediante lenguaje natural, es decir, a través de prompts. A diferencia de las formas tradicionales de programar, en este enfoque los desarrolladores, o incluso personas sin formación técnica, se comunican directamente con modelos de lenguaje para generar código, estructuras de proyecto, interfaces gráficas y lógica funcional. Por eso, la manera en que se redactan los prompts no es un detalle menor: es el corazón del proceso.

Un prompt en Vibe Coding no es simplemente un pedido suelto. Es una instrucción cuidadosamente pensada que debe transmitir la intención del usuario de forma clara, específica y contextualizada. Lo primero que hay que entender es que el modelo de IA no "lee la mente": trabaja con texto, y cuanto mejor esté formulado ese texto, más útil y preciso será el código que produzca.

En este sentido, los prompts más eficaces comparten ciertas características. En primer lugar, deben ser claros y directos. Un ejemplo poco eficaz sería: “hacé una app de tareas”. En cambio, un prompt útil podría ser: “Quiero una aplicación móvil que permita al usuario agregar, editar y eliminar tareas. Debe tener una lista ordenada por fecha, y cada tarea debe tener un título, una descripción y un botón para marcar como completada”. Este nivel de detalle permite que la IA entienda mejor qué se espera y produzca resultados más ajustados. Otro aspecto clave es el uso del lenguaje técnico cuando sea necesario. Aunque Vibe Coding permite trabajar con descripciones en lenguaje natural, en algunos casos conviene incluir referencias específicas a tecnologías, frameworks o convenciones. Por ejemplo: “Generá un backend en Python con Flask que exponga una API RESTful con operaciones CRUD para una base de datos SQLite”. Al incluir los términos adecuados, se evita ambigüedad y se orienta al modelo hacia la solución deseada. Además, la construcción de prompts puede beneficiarse del enfoque iterativo. Es decir, no se espera que el primer prompt genere el programa final perfecto. Muchas veces se empieza con una base —como un esqueleto de la aplicación— y luego se envían nuevos prompts para refinar, extender o corregir lo que la IA produjo. Esto se asemeja a un diálogo entre humano y asistente: “Agregá validación para que el campo de email no quede vacío”, “Traducí todos los textos al español”, “Mostrame los datos en una tabla con paginación”, etc. También se recomienda dividir el pedido en pasos cuando el proyecto es complejo. En lugar de enviar un prompt muy extenso y general, es preferible avanzar por partes, como si se tratara de un proceso de diseño incremental. Por ejemplo: primero el modelo de datos, luego la interfaz, después las rutas, y así sucesivamente. Esto ayuda a mantener el control sobre el código generado y facilita la revisión. Otra buena práctica es anticipar errores comunes o restricciones desde el comienzo. Incluir condiciones como “Evitá usar librerías externas”, “Usá solo funciones nativas de JavaScript”, o “Asegurate de que sea compatible con Android 10 en adelante” permite evitar malentendidos y reduce el trabajo posterior de corrección manual. Finalmente, una dimensión más avanzada del diseño de prompts tiene que ver con el uso de ejemplos y contraejemplos, técnica conocida como few-shot prompting. Por ejemplo, si queremos que la IA use cierto estilo de codificación o arquitectura, podemos incluir fragmentos de código como guía. Del mismo modo, podemos decir: “No uses clases, prefiero funciones puras”, o “Usá arquitectura MVC, como en este ejemplo”, y adjuntar un modelo base.

Plantillas para crear prompts de vibe coding

En base a lo anteriormente mencionado, a continuación, se describe diferentes plantillas a tener cuenta a la hora de desarrollar Prompts para crear programas a través de vibecoding.

# Plantillas generales

**Plantilla Básica**

Quiero un programa en [lenguaje] que haga lo siguiente:

* [Funcionalidad principal]
* [Otras funciones o detalles importantes]
* [Restricciones o condiciones opcionales]

**Plantilla en base a requisitos**

Escribí un programa en [lenguaje] que cumpla los siguientes requisitos:

* [Requisito funcional 1]
* [Requisito funcional 2]
* [Requisito técnico o restricción]
* Además, [otro comportamiento deseado].

**Plantillas para app móviles**

Generá una aplicación en [tecnología] que tenga:

* Una interfaz para [interacción del usuario]
* Conexión a [API o base de datos]
* Funcionalidad de [función clave]
* El diseño debe ser [simple/profesional/minimalista], y compatible con [restricción si hay].

**Plantillas para módulos o funciones**

* Escribí una función en [lenguaje] que:
  + Tome como entrada [parámetros]
  + Realice [acción deseada]
  + Devuelva [resultado esperado]
* No debe usar [restricciones, si hay].

Ejemplos de uso de las plantillas probadas en algunas herramientas de vibe coding:

Ejemplo 1

* **LENGUAJE DE PROGRAMACION**:
  + kotlin en Android studio
* **PLATAFORMA:**
  + Crear una aplicación para Wear OS 5
* **FUNCIONALIDAD A IMPLEMENTAR:**
  + Generar una notificación en pantalla completa que, automáticamente y sin intervención del usuario, abra una Activity que muestre contenido dinámico.
* **REQUISITOS ESPECIFICOS:**
  + La Activity abierta por la notificación debe mostrar datos dinámicos en tiempo real
  + Utilizar el patron MVVM para su construcción
* **RESTRICCIONES:** 
  + La notificación **no debe mostrarse** si la pantalla del smartwatch está bloqueada.

Ejemplo 2

* **LENGUAJE DE PROGRAMACION**:
  + kotlin en Android studio
* **PLATAFORMA:**
  + Crear una aplicación para Wear OS 5 en el smartwtach
  + Crear una aplicación para smartphone con versiones de Android superiores a Android 10.
* **FUNCIONALIDAD A IMPLEMENTAR:**
  + Desarrollar una aplicación que permita enviar un dato desde un smartphone a un smartwatch cuando el usuario presione un botón.
* **REQUISITOS ESPECIFICOS:**
  + La aplicación del smartphone debe incluir un campo de texto y un botón, de manera que el usuario pueda ingresar y enviar un mensaje de texto.
  + El smartwatch debe mostrar el mensaje recibido a través de un campo de texto visible en pantalla.
  + Utilizar el patron MVVM para su construccion
* **RESTRICCIONES:**
  + La comunicación entre dispositivos debe implementarse utilizando la clase WearableListenerService.
  + o Las interfaces gráficas deben diseñarse con XML en el smartphone y con Jetpack Compose en el smartwatch..

En base a esto se ha definido de forma preliminar una plantilla de prueba para crear aplicaciones Android que conecten con ESP32 a través de IoT

**Plantilla embebido Android más general**

Quiero crear un sistema IoT que conecte un ESP32 con una app Android.

* El ESP32 debe: [listar sensores, actuadores, tipo de conexión]
* La app Android debe: [mostrar datos, controlar algo, almacenar info]
* La comunicación debe ser por: [Bluetooth, HTTP, MQTT, Firebase]
* Usar en la app: [Kotlin, Room, Jetpack, etc.]
* Usar en el ESP32: [FreeRTOS, WiFi.h, PubSubClient, Patron de máquina de estados, etc.]

# Herramientas probadas hasta la fecha para trabajar en el proyecto de investigación.

Hasta la fecha se han probado las siguientes herramientas de vibe coding

* Gpt:
* Github copilot
* Geminis
* Cursor
* Windsurf

# Análisis de resultados de pruebas realizadas con las herramientas mencionadas

**Herramientas web utilizadas.**

# Gpt:

Se han probado diferentes prompts para generar programas en Android Studio con Kotlin, utilizando el modelo 4.0 de OpenAI. Los resultados obtenidos han sido diversos. Esto se debe a que, para problemas básicos, los resultados suelen ser satisfactorios, pero en algunos casos es necesario refinar las respuestas corrigiendo la salida generada por la inteligencia artificial.

No obstante, para problemas más avanzados, muchas veces resultó necesario realizar sucesivas iteraciones de refinamiento, ya que al ejecutar el código generado, este no funcionaba de manera correcta.

Una particularidad es que ChatGPT genera el código fuente de algunos archivos del proyecto de Kotlin directamente en el navegador web, por lo que el programador debe copiar y pegar el código en el proyecto a compilar en Android Studio. Es importante destacar que, en ocasiones, no se generan todos los archivos del proyecto, faltando algunos archivos de configuración. Por ejemplo, el archivo build.gradle, que el usuario debe configurar manualmente.

A su vez, también se han probado prompts básicos para la programación del ESP32, obteniendo buenos resultados, ya que las respuestas necesitaron ser refinadas en muy pocas ocasiones..

Enlace :<https://chatgpt.com/>

# Gemini:

Al igual que con ChatGPT, se probaron diferentes prompts para generar programas en Android Studio utilizando Kotlin y Java. En este caso, se evaluaron los modelos 2.5 Flash y 2.5 Pro. Los resultados obtenidos con esta herramienta fueron de menor grado de satisfacción en comparación con ChatGPT, ya que fue necesario refinar en mayor medida las respuestas generadas, e incluso, en algunos casos, completarlas con los resultados proporcionados por otra herramienta de inteligencia artificial.

Esta herramienta también se probó a través de la versión web, por lo que el programador debió copiar y pegar manualmente los archivos en los proyectos de Android Studio para poder compilarlos.

En esta herramienta IA también se han probado prompts básicos para programar los embebidos de ESP32, teniendo resultados aceptables, ya que algunas veces hubo que refinar las respuestas.

Enlace: <https://gemini.google.com/app?hl=es>

# Github Copilot:

Esta herramienta web es provista a través del sitio GitHub. Consiste en una plataforma de vibecoding que utiliza diferentes modelos especializados en programación. Entre los modelos disponibles de forma gratuita se encuentran Azure OpenAI GPT-4.0 y GPT-4.1, Gemini Flash 2.0 y Claude 3.5 Sonnet. Estos modelos están basados en Codex.

Se ha probado esta herramienta utilizando prompts para generar programas tanto básicos como complejos en Android Studio, con los lenguajes Kotlin y Java. Los resultados obtenidos varían según el modelo seleccionado, aunque en la mayoría de los casos fue necesario aplicar sucesivos refinamientos hasta alcanzar una solución adecuada.

Si bien esta herramienta puede utilizarse directamente desde la web, también existen varios plugins que permiten su integración en distintos entornos de desarrollo (IDE), como Visual Studio Code y Android Studio.

Además, esta herramienta se ha utilizado para generar programas básicos destinados a funcionar con el microcontrolador ESP32, obteniendo buenos resultados.

Enlace : <https://github.com/copilot>

**Herramientas de escritorio para vibe coding**

# Cursor:

Cursor puede utilizarse para crear aplicaciones en varios lenguajes de programación, entre los que se encuentran Python, C++ y Kotlin. Se presenta como una aplicación tipo IDE que puede ser descargada en la computadora, y puede utilizarse complementariamente con otros frameworks. En este caso, se realizaron pruebas en conjunto con Android Studio. Este programa genera el código completo de un proyecto de Android Studio, incluyendo sus archivos de configuración, dejando todo listo para ser ejecutado en un smartphone. Los resultados obtenidos fueron satisfactorios para prompts simples, en los que no fue necesario realizar demasiados refinamientos en el código generado.No obstante, para proyectos complejos (por ejemplo, aquellos que involucran múltiples dispositivos, como un smartphone y un smartwatch), en algunos casos se presentaron errores en archivos de configuración como build.gradle o el AndroidManifest.xml. Por esta razón, fue necesario realizar múltiples refinamientos hasta obtener un proyecto funcional.

Uno de los beneficios que presenta esta herramienta es la posibilidad de copiar los errores mostrados por el compilador y solicitárselos a la inteligencia artificial para que los corrija. Esta, automáticamente, corrige el código y lo deja preparado para ser compilado nuevamente en Android Studio, con la validación final del programador.

Enlace: <https://www.cursor.com/>

# Windsurf:

Al igual que la herramienta anterior, esta plataforma de vibecoding puede descargarse en la computadora como un entorno de desarrollo (IDE), basado en Visual Studio Code. No obstante, en algunos editores también puede utilizarse como un plugin. Esta herramienta permite generar código en distintos lenguajes de programación, como Python, C++, Kotlin y Java. Al igual que Cursor, trabaja de manera complementaria con el framework de Android Studio: primero se genera el proyecto base en Android Studio y luego WindSurf lo completa en función de nuestras especificaciones, utilizando modelos de lenguaje (LLM). En base a esto, se probaron diferentes prompts. Para especificaciones simples, WindSurf generó correctamente el proyecto completo. Sin embargo, en algunos casos se presentaron incompatibilidades entre dependencias, lo que obligó a refinar las respuestas generadas por la IA, y en muchos casos fue necesaria la corrección manual del código para que el proyecto funcionara correctamente

Enlace: <https://windsurf.com/>