IAF
INTELIGENCIA ARTIFICIAL
FUNDACIÓN

print (“

PROGRAMACIÓN CON IA ”)

NIVEL I

PENSAR COMO PROGRAMADOR

MATERIAL DE REPASO

Start | Nivel II | Nivel III

FUNDAMENTOS

Nuestro objetivo: pensar como programadores

Programar es una forma de resolver problemas, no una lista de comandos para memorizar.

EN PALABRAS SIMPLES

Primero comprendo la necesidad. Luego ordeno una estrategia, la convierto en instrucciones y pruebo si el resultado cumple el objetivo. El código es una parte del proceso, no el proceso completo.

Descomposición Dividir un problema en partes manejables.

FUNDAMENTOS

Un algoritmo organiza una solución

Una computadora necesita pasos claros, ordenados y suficientemente precisos.

La programación en la vida cotidiana

Antes del código aparece una idea fundamental: el algoritmo.

ALGORITMO Pasos ordenados y precisos para resolver un problema o alcanzar un resultado.

01 Preparar
Reunir los elementos.

02 Ordenar
Preparar la yerba.

03 Condición
Si hirvió, dejar enfriar.

04 Repetir
Cebar y comprobar.

También importan las condiciones, las repeticiones y el criterio para comprobar el resultado.

06

EJEMPLO DEL MATE

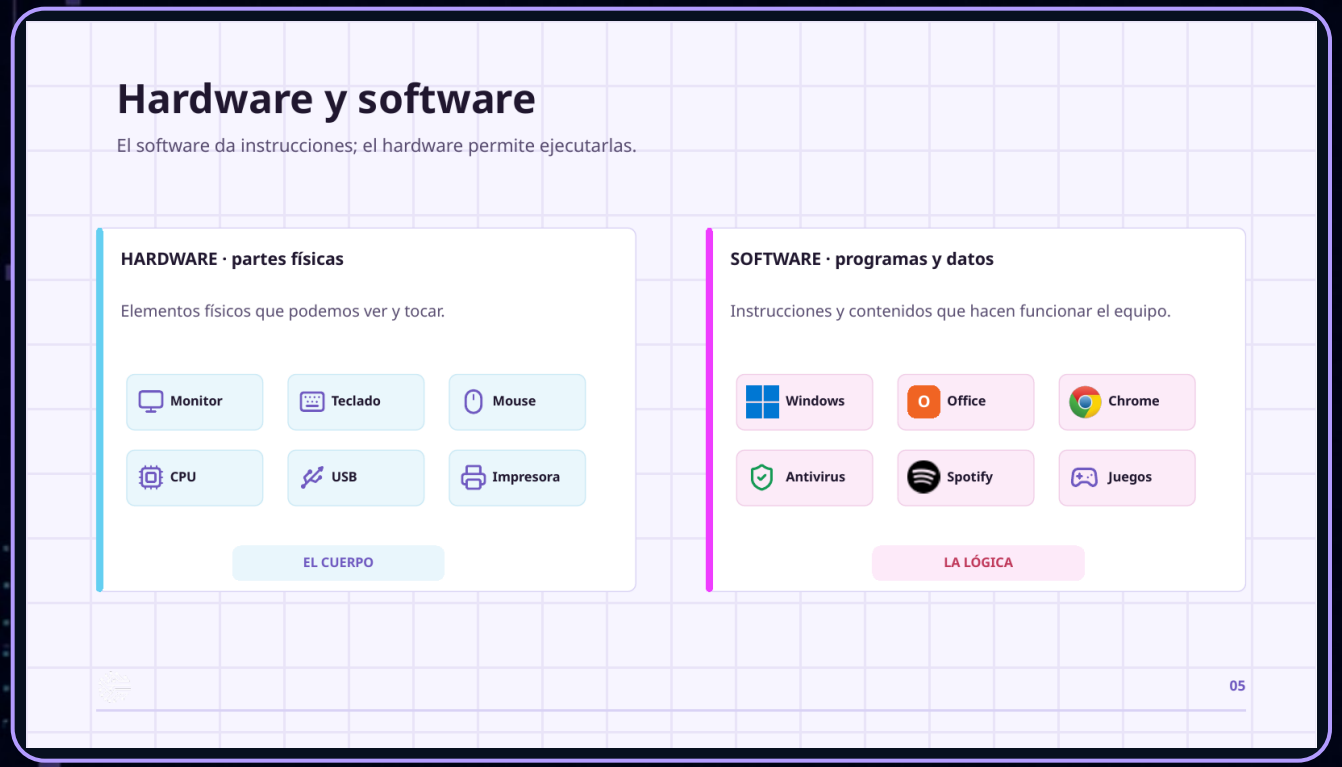
Para una persona, “preparar un mate” parece una sola acción. Para una máquina debemos separar insumos, orden, condiciones y repeticiones. Si falta agua o la temperatura no es adecuada, el algoritmo debe contemplarlo.

Condición Regla que determina qué camino seguir.

FUNDAMENTOS

Hardware y software: dos capas inseparables

El equipo físico ejecuta; los programas organizan lo que debe hacer.



EJEMPLO

Al tocar un botón, la pantalla detecta el gesto -hardware- y la aplicación interpreta la acción -software-. Sin instrucciones, el equipo no sabe qué resultado producir.

Hardware	Componentes físicos que ejecutan acciones.
Software	Instrucciones y datos que organizan esas acciones.

FUNDAMENTOS

El lenguaje binario: la base de las máquinas

Los sistemas electrónicos representan información mediante estados que solemos expresar como 0 y 1.

¿POR QUÉ DOS VALORES?

Un circuito puede distinguir con fiabilidad entre dos estados, como encendido y apagado. Esos estados se representan como 1 y 0. Combinados, permiten codificar números, letras, imágenes, sonidos e instrucciones.

01001000 01001111 01001100 01000001

H O L A

No escribimos aplicaciones completas en binario: usamos lenguajes más cercanos a nuestra forma de pensar.

CLAVE

El binario explica cómo representa datos una máquina. Los lenguajes de programación funcionan como una capa de traducción entre nuestra intención y esas operaciones internas.

HISTORIA

El telar de Jacquard: instrucciones antes de las computadoras

Una innovación textil anticipó la idea de cambiar resultados mediante instrucciones almacenadas.

El telar de Jacquard

Tarjetas perforadas: instrucciones que cambiaban el patrón sin reconstruir la máquina.

1801–1804

Telar de Jacquard · Wikimedia Commons

Del patrón a la instrucción

El mecanismo se presentó desde 1801 y fue patentado en 1804. Las tarjetas indicaban qué hilos levantar: cambiar la tarjeta cambiaba el resultado.

TARJETA

Patrón codificado

MÁQUINA

Lee instrucciones

TELA

Produce el resultado

07

CONTEXTO HISTÓRICO

Entre 1801 y 1804, las tarjetas perforadas permitieron cambiar el patrón del tejido sin reconstruir la máquina. La información estaba separada del mecanismo: al cambiar las tarjetas, cambiaba el resultado.

Tarjeta perforada Soporte físico que codificaba instrucciones.

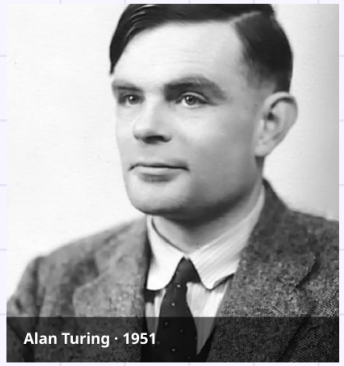
HISTORIA

Alan Turing y la idea de una máquina universal

Una misma máquina puede resolver tareas distintas cuando recibe procedimientos diferentes.

Alan Turing y la computación moderna

Una máquina universal podía representar y ejecutar distintos procedimientos.



Alan Turing · 1951

Una misma máquina, muchos programas

El comportamiento cambia cuando cambian las instrucciones, sin reconstruir físicamente el dispositivo.

Idea central

Separar la máquina de las instrucciones abrió el camino al software moderno y a la programación de propósito general.

Fotografía: Wikimedia Commons

08

CONTEXTO HISTÓRICO

Turing ayudó a formalizar que un procedimiento preciso puede ejecutarse paso a paso. Una misma máquina podría realizar tareas diferentes si recibe instrucciones diferentes. Esta idea sostiene gran parte de la computación moderna.

Máquina universal Modelo capaz de ejecutar procedimientos diferentes.

HISTORIA

Cuando “computadora” era una ocupación humana

Antes de ser una máquina, “computadora” era el nombre de una ocupación especializada.

Cuando las computadoras eran personas

Durante décadas, “computador/a” fue el nombre de un trabajo humano.

Computadoras humanas de NACA · 1949

Personas que calculaban

Equipos —con una participación decisiva de mujeres— resolvían tablas, trayectorias y operaciones científicas de forma manual.

Máquinas que automatizan

La programación trasladó procedimientos humanos a sistemas capaces de ejecutarlos con velocidad y repetición.

↗ NASA on The Commons · Wikimedia Commons

09

CONTEXTO HISTÓRICO

Antes de identificar a una máquina, la palabra describía a personas que realizaban cálculos. Muchas mujeres trabajaron en astronomía, aeronáutica y defensa. La tecnología electrónica se construyó sobre conocimiento y trabajo humano previo.

Cálculo humano Trabajo especializado realizado antes de la automatización electrónica.

HISTORIA

ENIAC: computación electrónica a gran escala

La computación electrónica convirtió procesos físicos y lógicos en una nueva escala de cálculo.

ENIAC: computación electrónica a gran escala

Una computadora electrónica digital de propósito general.

ENIAC · presentada en 1946

Una computadora electrónica de propósito general

Ocupaba una sala completa y se configuraba conectando cables y paneles. Programar era todavía una tarea física y especializada.

Evolución

De reconfigurar hardware pasamos a escribir instrucciones reutilizables en lenguajes cada vez más cercanos a las personas.

Fotografía: Wikimedia Commons

10

CONTEXTO HISTÓRICO

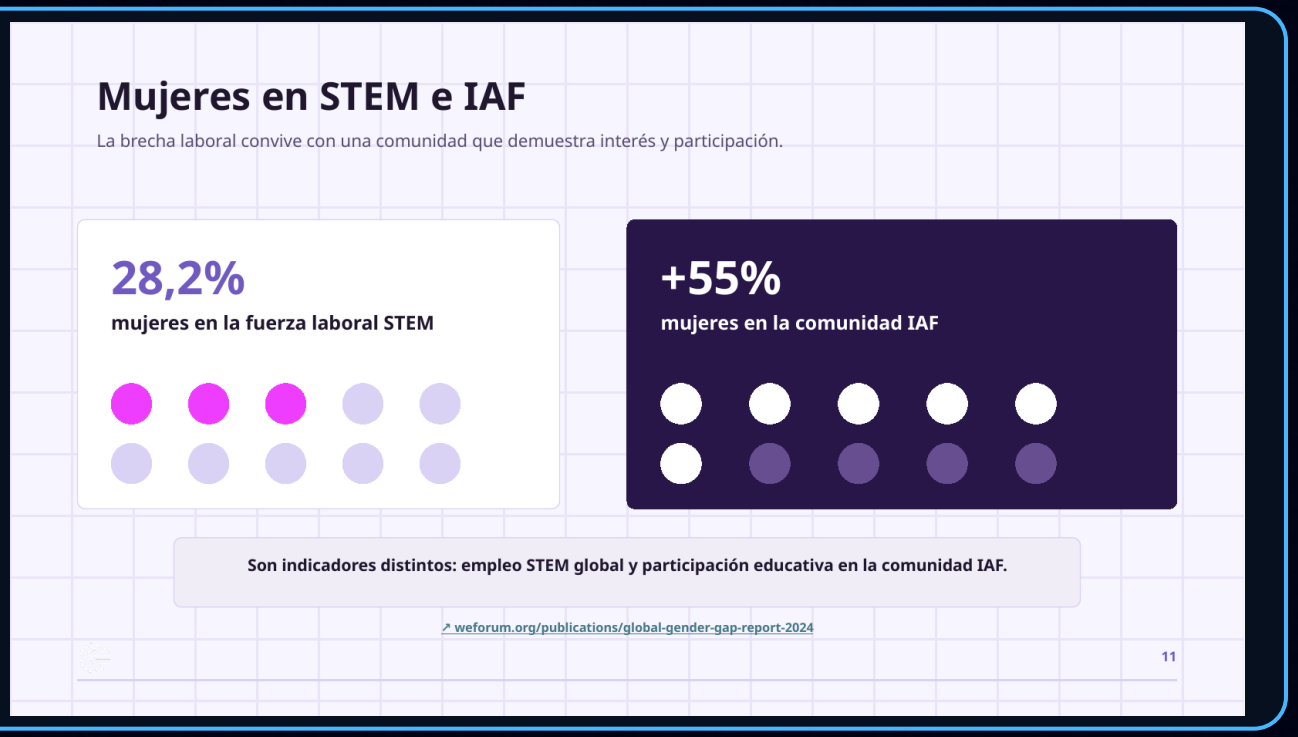
ENIAC fue una computadora electrónica digital de propósito general. Sus programadoras configuraban paneles y conexiones físicas. Esto muestra que el software no siempre fue un archivo: primero también fue una organización lógica implementada sobre la máquina.

Programación física Configuración de paneles, cables y conexiones.

HISTORIA

Mujeres en STEM: interés, acceso y oportunidades

El desafío no es solo despertar interés: también es transformar formación en oportunidades.



CONTEXTO HISTÓRICO

La participación laboral de las mujeres en áreas STEM sigue siendo desigual. Al mismo tiempo, en la comunidad IAF más del 55% de las personas interesadas son mujeres. El reto de gestión es transformar interés y formación en capacidades, proyectos y oportunidades.

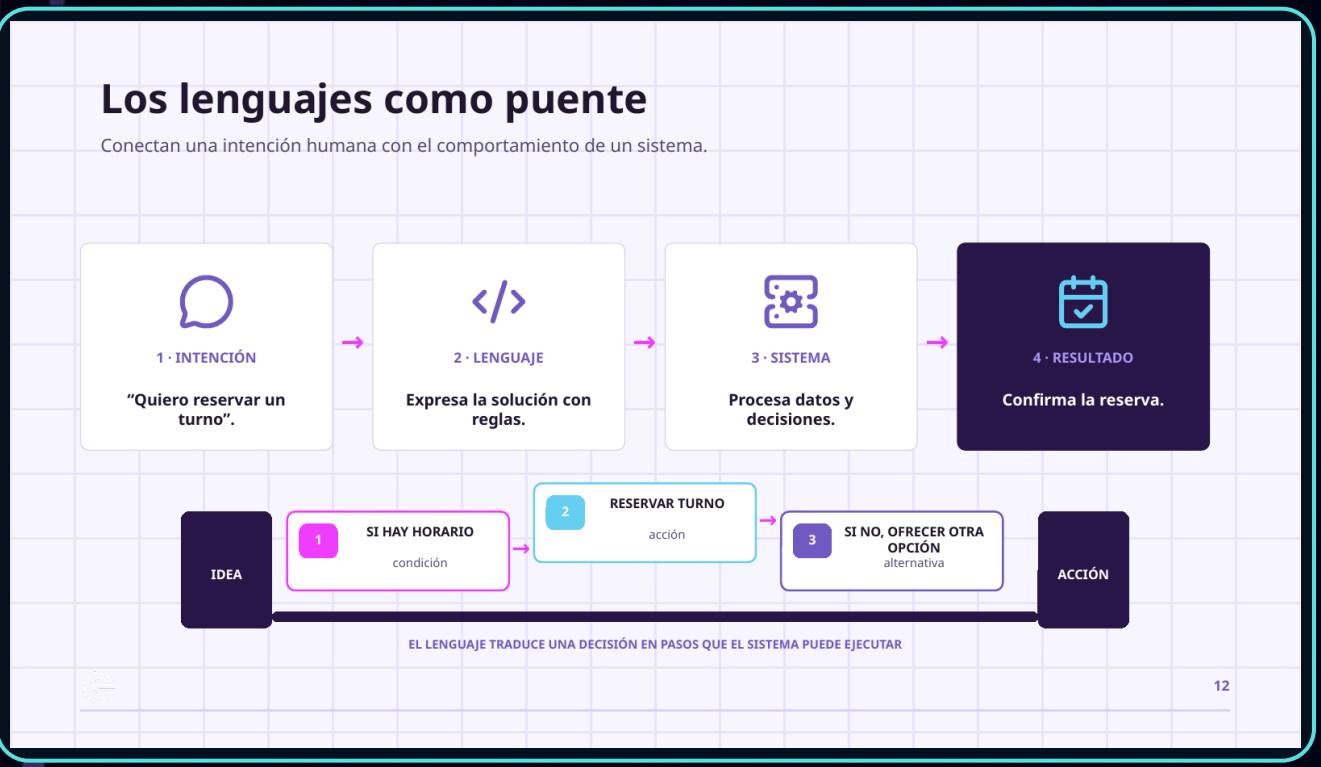
STEM	Ciencia, tecnología, ingeniería y matemática.

Fuente: Global Gender Gap Report 2024

DESARROLLO WEB

Los lenguajes funcionan como un puente

Conectan una intención humana con el comportamiento comprobable de un sistema.



RECORRIDO

Expreso una necesidad, la escribo en un lenguaje con reglas, el sistema la procesa y obtengo un resultado. Si el resultado no coincide con la intención, reviso el puente: instrucciones, datos o condiciones.

Sintaxis Reglas para escribir código válido.

DESARROLLO WEB

C y C++: evolución y vigencia

Los lenguajes históricos siguen presentes en sistemas y productos actuales.

De C a C++: evolución y vigencia

Lenguajes históricos siguen presentes en productos que usamos todos los días.

C · Dennis Ritchie

Linux · Git · SQLite · sistemas operativos y herramientas de bajo nivel.

C++ · Bjarne Stroustrup

Unreal Engine · Fortnite · Photoshop · Chrome y videojuegos de alto rendimiento.

Linux · Git · SQLite · Unreal · Fortnite · Photoshop · Chrome

Ejemplos con componentes o herramientas desarrolladas en C/C++; no necesariamente utilizan un único lenguaje.

↗ Retratos: Wikimedia Commons

13

DIFERENCIA ORIENTATIVA

C se destaca por su cercania al funcionamiento del sistema y su eficiencia. C++ surgio como una extension que agrego herramientas para organizar programas mas complejos. Ambos se usan cuando el rendimiento y el control importan.

Multiparadigma Permite organizar soluciones con distintos enfoques.

Rendimiento Uso eficiente de memoria y procesamiento.

Lenguajes frecuentes y sus usos

No existe un lenguaje “mejor” para todo: la elección depende del problema y del entorno.

MAPA RÁPIDO

JavaScript aporta interacción web; Python se usa en automatización, datos e IA; SQL consulta bases de datos; Java aparece en aplicaciones empresariales; C# se utiliza en aplicaciones empresariales, apps y videojuegos; C y C++ se usan en sistemas, motores y software de alto rendimiento.

Ecosistema	Herramientas, bibliotecas y comunidad de un lenguaje.
Biblioteca	Código reutilizable para resolver tareas frecuentes.

Tecnologías principales del desarrollo web

El navegador combina estructura, apariencia y comportamiento; el servidor procesa otras responsabilidades.

ROLES DIFERENTES

HTML es un lenguaje de marcado que estructura el contenido. CSS es un lenguaje de estilos que define su presentación. JavaScript sí es un lenguaje de programación y aporta lógica e interacción. Python suele utilizarse en backend, automatización, datos e IA.

HTML · W3Schools CSS · W3Schools JavaScript · W3Schools

Python · W3Schools

DESARROLLO WEB

Del código a una sección real

Las tres capas se combinan para producir una experiencia que podemos ver y utilizar.

LECTURA VISUAL

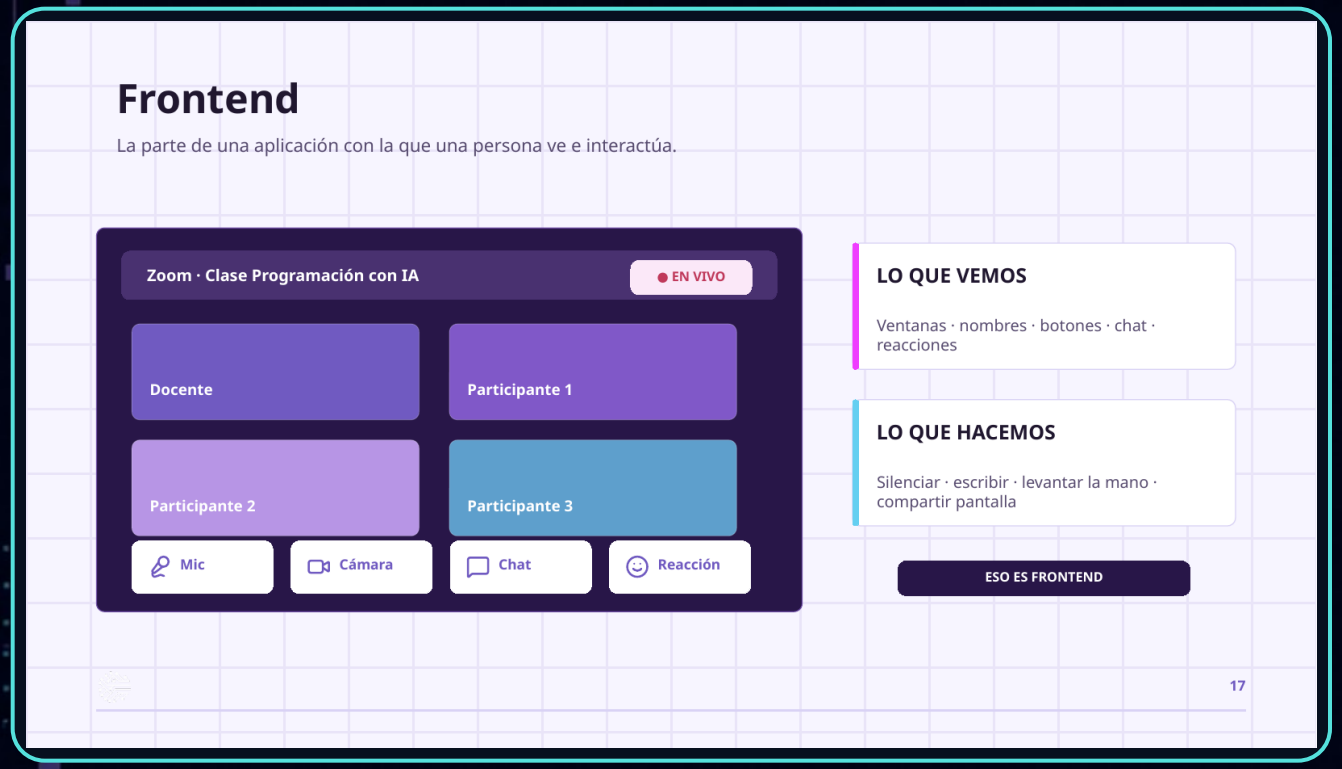
HTML crea el título y el botón. CSS aplica colores, espacios y jerarquía. JavaScript escucha el clic y genera una respuesta. Separar estas responsabilidades facilita encontrar qué parte debemos modificar.

DOM	Representación de la página que JavaScript puede consultar y modificar.
Selector	Referencia para aplicar estilos o comportamiento a un elemento.

DESARROLLO WEB

Frontend: la experiencia visible

Es la parte de la aplicación con la que una persona ve, navega e interactúa.



EJEMPLO

En una plataforma de cursos, el frontend muestra salas, participantes, botones y mensajes. Puede validar datos para mejorar la experiencia, pero las verificaciones sensibles deben repetirse en el backend: el código del navegador puede manipularse.

Responsive	Diseño que se adapta a distintos tamaños de pantalla.
UX	Decisiones que mejoran la experiencia de uso.

DESARROLLO WEB

Backend: procesos detrás de la interfaz

Recibe solicitudes, aplica reglas, consulta datos y prepara respuestas.

EJEMPLO

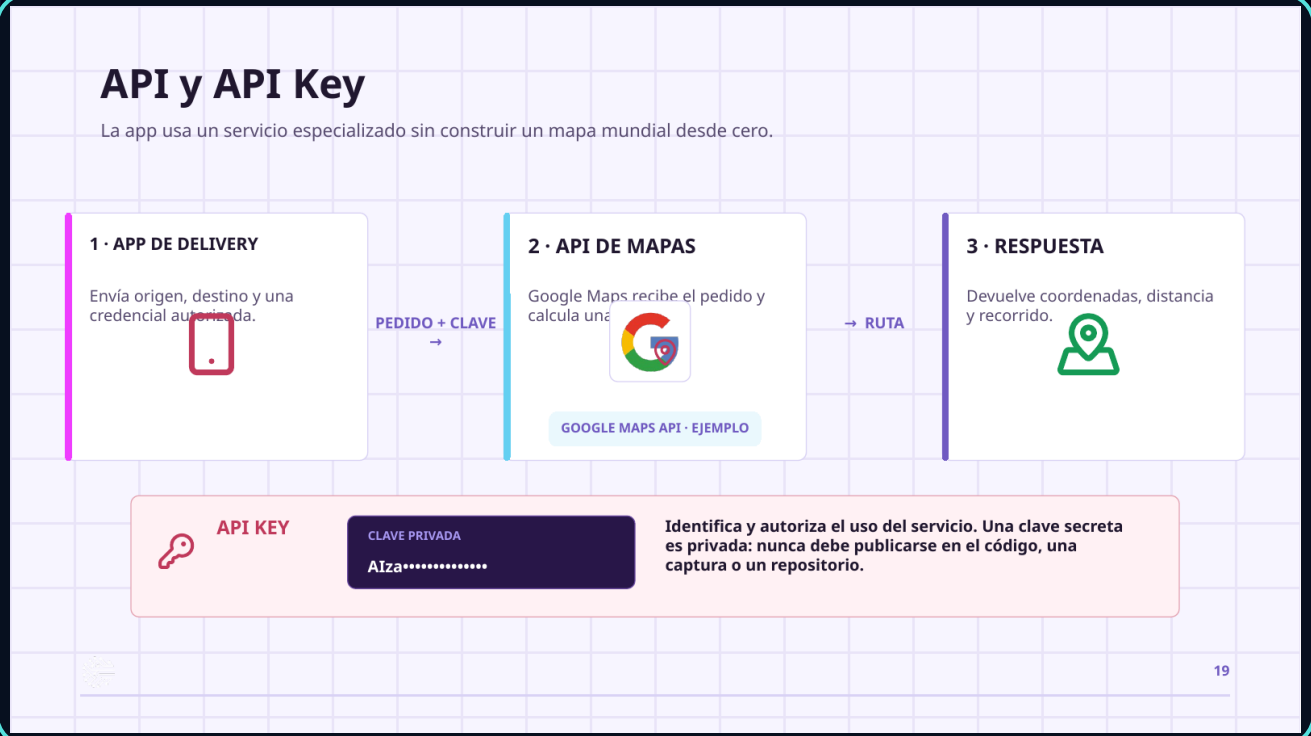
Cuando una persona inicia sesión, el backend comprueba sus datos y permisos. Cuando envía un pedido, registra la operación y calcula el resultado. La interfaz no necesita mostrar todos esos procesos.

Regla de negocio	Condición propia del funcionamiento de una aplicación.
Base de datos	Sistema para almacenar, consultar y actualizar datos.

DESARROLLO WEB

API y API Key con Google Maps

Una aplicación puede utilizar un servicio especializado sin construirlo desde cero.



EJEMPLO SIMPLE

Una app de delivery solicita a Google Maps el cálculo de una ruta mediante operaciones definidas por su API y recibe distancia y tiempo. La API Key identifica el proyecto que hace la solicitud: debe protegerse con restricciones de dominio, API, cuota y permisos.

API	Interfaz con reglas y operaciones para intercambiar datos o usar funciones.
API Key	Identificador del proyecto que requiere restricciones y controles.

DESARROLLO WEB

Cómo viaja una acción por una web

Un flujo frecuente conecta la interfaz, la API, el backend y los datos; no todas las acciones recorren todas las capas.

EJEMPLO DE RESERVA

1) La persona confirma desde el frontend. 2) El frontend envía una solicitud a la API expuesta por el backend. 3) El backend verifica reglas. 4) Consulta o guarda datos. 5) La API devuelve la respuesta y el frontend muestra “Reserva confirmada”.

Persistencia Guardar información para recuperarla después.

Respuesta HTTP Respuesta estandarizada que devuelve el servidor.

IA

Programación tradicional y asistida por IA

Dos enfoques complementarios: cambian las herramientas, pero no la responsabilidad profesional.

CON FUNDAMENTOS

La persona diseña la solución, escribe o adapta el código, lo prueba, depura y documenta. Trabajar sin IA puede ser útil para aprender una base, comprender una parte crítica o resolver tareas donde el contexto no debe compartirse.

CON ASISTENCIA DE IA

La IA puede proponer estructuras, explicar fragmentos, generar borradores y localizar errores. La persona sigue definiendo requisitos, revisando cada cambio, ejecutando pruebas y protegiendo datos, claves y dependencias.

RECOMENDACIÓN

Combiná fundamentos y asistencia: pedí cambios pequeños, compará el resultado con el objetivo, leé el código y probá los casos principales. Incorporá solo aquello que puedas comprender, verificar y mantener.

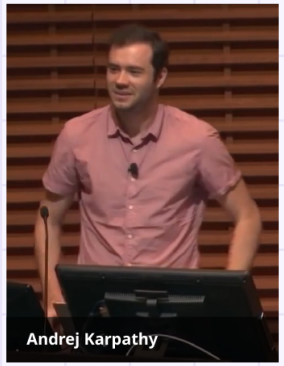
IA

Andrej Karpathy y el origen del Vibe Coding

El término fue presentado de manera informal el 2 de febrero de 2025.

Vibe Coding: origen y significado

Andrej Karpathy introdujo el término el 2 de febrero de 2025.



Andrej Karpathy @karpathy

SÍNTESIS FIEL EN ESPAÑOL

Una nueva forma de programar consiste en describir lo que queremos en lenguaje natural y dejar que la IA genere o modifique el código. En el sentido original, Karpathy también reconocía que a veces aceptaba cambios sin revisar demasiado cómo estaban construidos.

LENGUAJE NATURAL → CÓDIGO

POCA REVISIÓN DEL CÓDIGO

Publicación original · 02/02/2025

EN ESTE CURSO

La IA genera; la persona define, prueba, revisa, corrige y valida antes de publicar.

↗ x.com/karpathy/status/1886192184808149383

22

CONTEXTO

Karpathy es investigador, ingeniero y educador especializado en IA y redes neuronales. Fue integrante fundador de OpenAI y dirigió el área de IA de Tesla. Describió una práctica informal de crear software conversando con modelos y guiándose principalmente por el resultado.

Ver publicacion original de Karpathy

Vibe Coding	Práctica informal de generar software mediante diálogo con IA.
Desarrollo asistido	Uso de IA con revisión, pruebas y responsabilidad humana.

IA

Qué significa Vibe Coding en este curso

Tomamos la velocidad de la conversación, pero agregamos comprensión, prueba y responsabilidad.

DEFINICIÓN INFORMAL

Vibe Coding describe una práctica de construir software conversando con IA y avanzando por resultados, a veces sin revisar en profundidad el código generado.

NUESTRO ENFOQUE RESPONSABLE

Usamos desarrollo asistido por IA: no aceptamos el resultado a ciegas. Leemos la estructura, probamos funciones, cuidamos datos y claves, detectamos errores y decidimos qué cambios conservar.

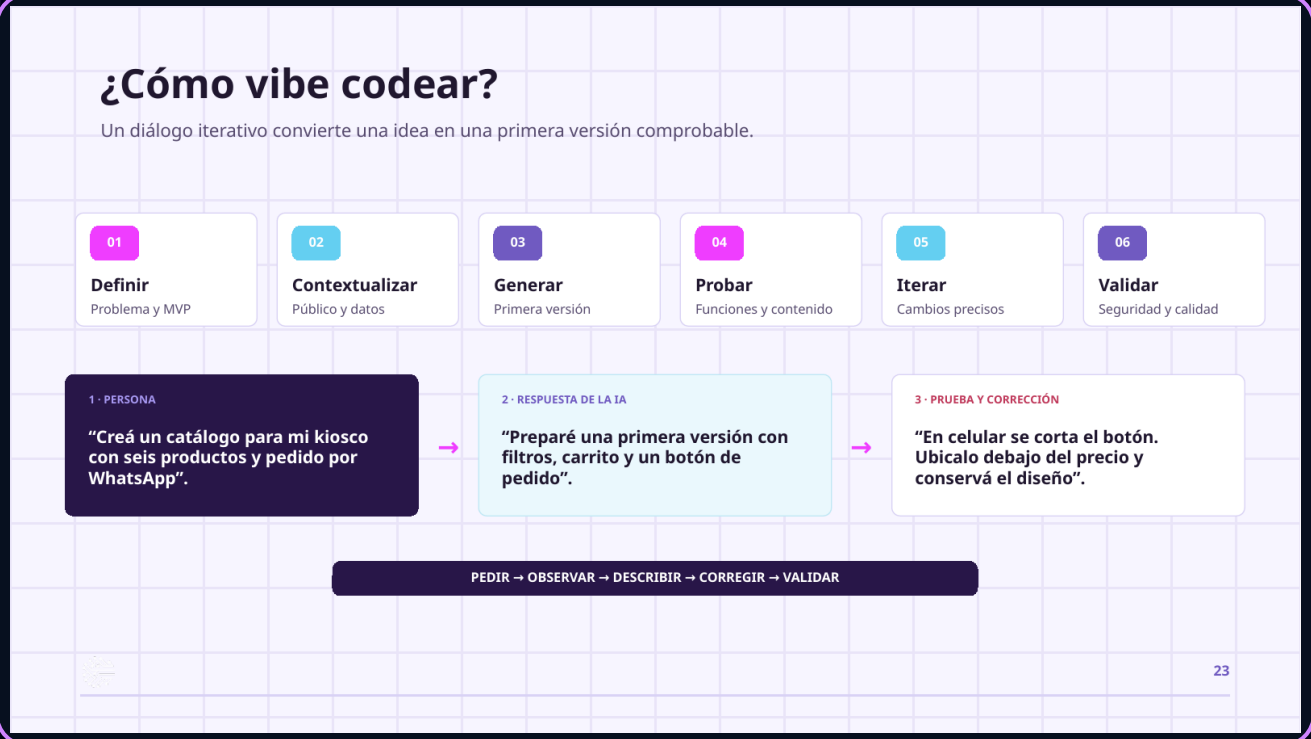
OBJETIVO DEL CURSO

Que cada participante pueda pasar de una idea concreta a una primera solución funcional asistida por IA, manteniendo el control del proyecto y pudiendo explicar sus decisiones.

IA

Cómo Vibe Codear con criterio

Un diálogo iterativo convierte una idea en una primera versión comprobable.



CICLO RECOMENDADO

Definir el problema → aportar contexto → pedir una primera versión → probarla → describir una corrección precisa → volver a validar. Trabajamos en ciclos cortos para saber qué cambio produjo cada resultado.

Iterar	Generar, probar, corregir y volver a comprobar.
Contexto	Información que reduce ambigüedades en el pedido.

IA

Plantilla para realizar un pedido claro

Cuanto menos deba adivinar el asistente, más alineada será la primera versión.

ROL

Actuá como desarrollador frontend y diseñador UX/UI.

OBJETIVO

Creá una web para mostrar productos y recibir consultas.

PÚBLICO

Personas adultas y jóvenes, desde celular y PC.

CONTENIDO

Categorías, productos, precios y promociones.

FUNCIONES

Catálogo y consulta por WhatsApp.

LÍMITES

Un solo HTML, responsive y con comentarios.

Copiala y reemplazá cada ejemplo por la información de tu propio proyecto.

IA

Cómo pedir una corrección útil

“No funciona” aporta poca información. Una corrección efectiva compara lo actual con lo esperado.

PEDIDO IMPRECISO

“Arreglalo, está mal.” No indica dónde ocurre el problema, qué resultado se esperaba ni qué partes deben conservarse.

PEDIDO PRECISO

“En celular, el menú queda abierto y tapa el título. Quiero que se cierre al elegir una opción. No cambies colores ni contenido.”

FÓRMULA

Contexto + comportamiento actual + resultado esperado + límites del cambio.

PROYECTO

Cómo leer código sin memorizarlo

Primero reconocemos bloques y relaciones; después profundizamos en cada símbolo.

```
<!-- BOTON DE PEDIDO -->

<button class="consultar">

  Consultar por WhatsApp

</button>
```

LECTURA LÍNEA POR LÍNEA

El comentario identifica la sección. <button> crea el elemento. class="consultar" le asigna un nombre reutilizable para CSS o JavaScript. El texto central es lo que ve la persona. </button> cierra el elemento.

Etiqueta	Marca que define un elemento HTML.
Clase CSS	Nombre reutilizable para aplicar estilos o seleccionar elementos.

PROYECTO

Comentarios: notas internas dentro del código

No se presentan como contenido de la página ni se ejecutan, pero pueden verse al inspeccionar el archivo.

HTML

```
<!-- BARRA DE NAVEGACIÓN -->
<nav>...</nav>
```

CSS

```
/* ESTILOS DE LA NAVEGACIÓN */
.nav { display: flex; }
```

JS

```
// ABRIR MENÚ EN CELULAR
menu.classList.toggle("abierto");

/*
  ABRIR O CERRAR EL MENÚ
  EN DISPOSITIVOS MÓVILES
*/
```

QUÉ ES UN COMENTARIO

Es una anotación no ejecutable escrita para orientar a quien lee el código. Puede verse al inspeccionar el archivo: nunca debe contener claves, contraseñas ni datos privados.

PARA QUÉ SIRVE

Permite identificar secciones, justificar decisiones y pedir cambios precisos: “modificá solo BARRA DE NAVEGACIÓN y conservá el resto”.

SINTAXIS

HTML: <!-- --> · CSS: /* */ · JavaScript: // para una línea o /* */ para un bloque.

PROYECTO

Comprobar y mejorar una solución

Aplicado al juego del QR y al Kiosko vistos en clase.

1 · OBSERVAR

¿Qué ocurre?

2 · UBICAR

¿Dónde ocurre?

3 · COMPARAR

¿Qué esperaba?

4 · CORREGIR

¿Qué cambio preciso?

5 · REPETIR

¿Sigue funcionando?

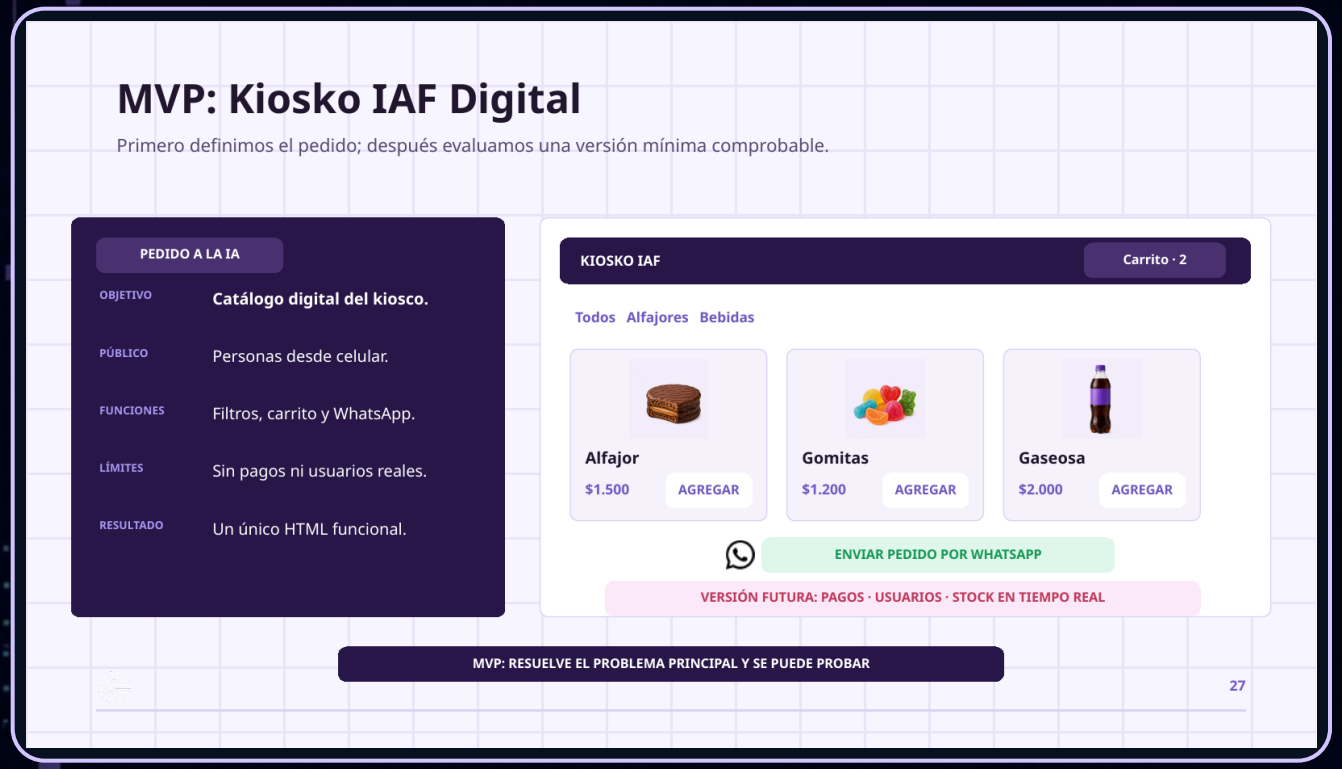
APLICACIÓN VISTA EN CLASE

Como en el juego del QR y el Kiosko: si el botón funciona en computadora pero queda fuera de pantalla en celular, la corrección debe limitarse al diseño responsive y luego probar ambos tamaños.

PROYECTO

MVP: Kiosko IAF Digital

Primero validamos la hipótesis central con una versión simple y útil; después decidimos si conviene ampliar.



HIPÓTESIS A VALIDAR

Catálogo básico + productos + precios + consulta por WhatsApp. Esta versión permite observar si las personas comprenden la oferta y muestran intención real de iniciar un pedido.

MEJORAS POSTERIORES

Filtros, carrito, usuarios, pagos, stock y panel administrador quedan fuera de la primera versión. Se priorizan después, según la evidencia obtenida con usuarios reales.

PROYECTO

Cómo construir un MVP paso a paso

MVP significa Producto Mínimo Viable: mínimo en alcance, viable en utilidad y profesional en ejecución.

1 · PROBLEMA

Necesito mostrar productos y recibir consultas.

2 · ALCANCE

Defino qué entra y qué queda para después.

3 · PEDIDO

Explico público, contenido, estética y límites.

4 · GENERAR

La IA crea una primera versión funcional.

5 · PROBAR

Una persona completa el recorrido principal.

6 · APRENDER

Uso la evidencia para decidir la próxima mejora.

PROYECTO

Checklist antes de publicar

La IA acelera la construcción; la validación sigue siendo una responsabilidad humana.

- [ ] **La acción principal se puede completar.**
- [ ] **La propuesta se entiende sin explicación oral.**
- [ ] **Textos, precios y contactos son correctos.**
- [ ] **Botones y enlaces hacen lo que prometen.**
- [ ] **La web se adapta a celular y computadora.**
- [ ] **No hay claves ni datos privados expuestos.**
- [ ] **Los cambios no rompieron funciones anteriores.**
- [ ] **Otra persona pudo probar el recorrido.**

REPASO

Actividad: convertí una idea en un proyecto

La consigna prepara la materia prima para el Nivel II.

EJEMPLO RESUELTO

“Quiero una web para un emprendimiento de pastelería, dirigida a personas del barrio, que permita conocer productos y consultar por WhatsApp.”

PROBLEMA

¿Qué necesidad aborda?

PÚBLICO

¿Quién la usaría?

ACCIÓN

¿Qué debe poder hacer?

MVP

¿Cuál es la versión más pequeña útil?

REPASO

Cuatro ideas para llevarse

La programación combina pensamiento, comunicación, prueba y responsabilidad.

SÍNTESIS

Pensar: comprender el problema. Expresar: convertir la intención en instrucciones. Comprobar: probar el resultado. Mejorar: iterar sin perder el objetivo. La persona conserva las decisiones en todo el recorrido.

REPASO

Glosario I: fundamentos y web

Definiciones breves para volver a consultar durante la práctica.

ALGORITMO	Secuencia precisa y finita de pasos.
DESCOMPOSICIÓN	División de un problema en partes.
SINTAXIS	Reglas para escribir código válido.
BINARIO	Representación mediante dos estados: 0 y 1.
FRONTEND	Interfaz visible con la que se interactúa.
BACKEND	Procesos y reglas ejecutados del lado del servidor.
BASE DE DATOS	Sistema para almacenar, consultar y actualizar datos.
RESPONSIVE	Diseño adaptable a distintos tamaños de pantalla.
DOM	Representación de la página que JavaScript puede modificar.
PERSISTENCIA	Conservación de datos para recuperarlos después.

REPASO

Glosario II: comunicación, IA y producto

Conceptos que aparecen al crear, integrar y validar una solución.

API	Interfaz con reglas y operaciones definidas.
API KEY	Identificador de proyecto que debe restringirse.
VIBE CODING	Práctica informal de generar software con IA.
PROMPT / PEDIDO	Instrucción y contexto dados al asistente.
ITERAR	Probar, corregir y volver a comprobar.
VALIDAR	Obtener evidencia sobre una hipótesis.
MVP	Versión simple para probar valor con usuarios.
DEPURAR	Localizar, comprender y corregir errores.

REPASO

Fuentes y recursos para ampliar

Los enlaces están incorporados como vínculos funcionales dentro del PDF.

Historia de la computación · Computer History Museum
Abrir recurso · Ctrl+clic

Alan Turing · Encyclopaedia Britannica
Abrir recurso · Ctrl+clic

ENIAC · University of Pennsylvania
Abrir recurso · Ctrl+clic

Global Gender Gap Report 2024 · WEF
Abrir recurso · Ctrl+clic

Publicación original de Andrej Karpathy
Abrir recurso · Ctrl+clic

Documentación de Google Maps Platform
Abrir recurso · Ctrl+clic

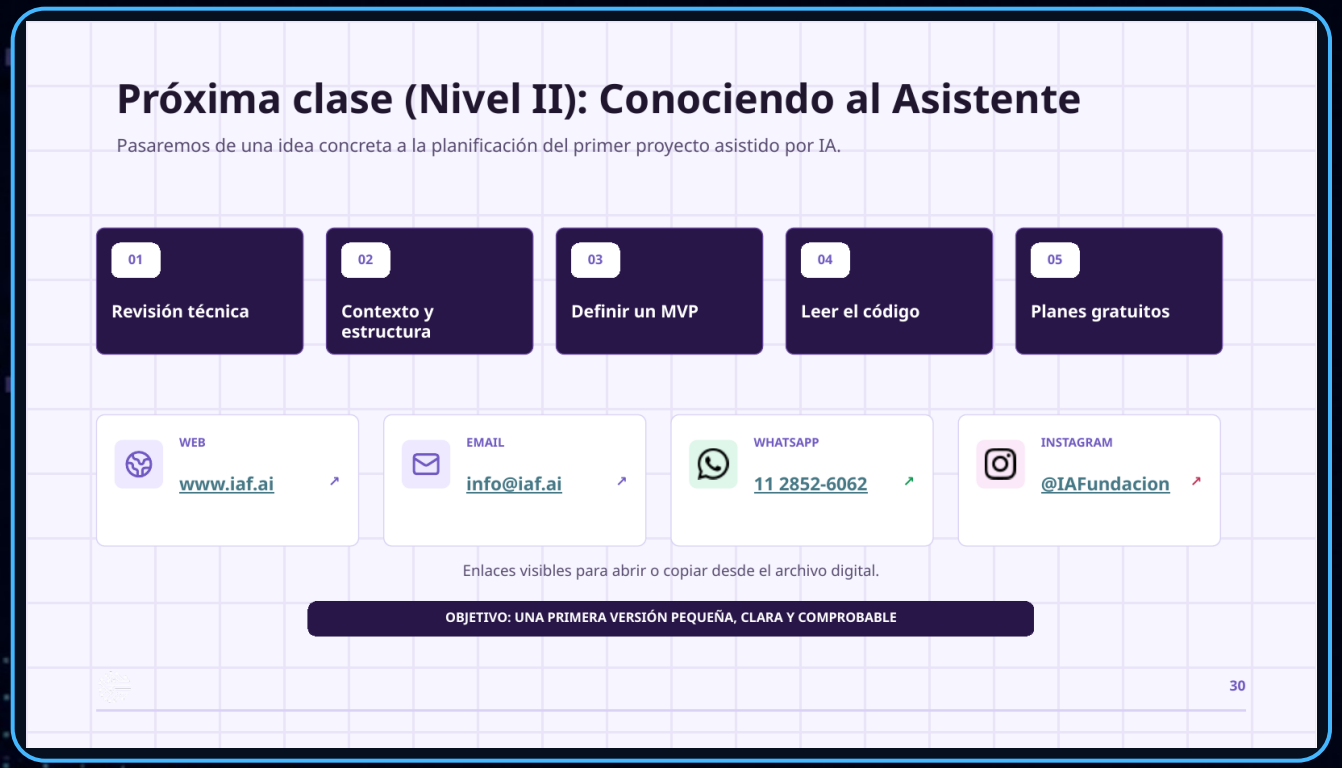
W3Schools · HTML, CSS, JavaScript y Python
Abrir recurso · Ctrl+clic

Fundación de Inteligencia Artificial
Abrir recurso · Ctrl+clic

En PDF, el visor decide cómo abre cada vínculo. Usá Ctrl+clic o clic derecho → “Abrir vínculo en una pestaña nueva” para mantener abierto el material.

REPASO

Próxima etapa: del concepto al primer proyecto



LO QUE SIGUE

Repaso de HTML, CSS, JavaScript y Python; contexto para el asistente; definición del MVP; lectura de comentarios y secciones; corrección de errores; herramientas accesibles para comenzar.

CREANDO FUTURO JUNTOS

www.iaf.ai

@IAFundacion

info@iaf.ai

WhatsApp IAF