

CONCEPTO 2 DE 4

Tipos de aplicaciones

Móviles, de escritorio y web — dónde vive el código y quién lo ejecuta

LAS TRES FAMILIAS

**Móviles**

Se instalan en el teléfono. El código corre en iOS o Android.

Ej: Instagram, WhatsApp, TikTok

**Escritorio**

Se instalan en tu computadora. El código corre en Windows, Mac o Linux.

Ej: Word, Spotify, Chrome

**Web**

No se instalan. Abrís el navegador, escribís una URL y el código llega desde un servidor.

Ej: Gmail, MercadoLibre, Notion

DENTRO DE LAS APPS MÓVILES: NATIVAS, HÍBRIDAS Y PWAS

No todas las apps móviles son iguales. La distinción más importante está en cómo está escrito el código:

TIPO**Nativa**

Escrita específicamente para una plataforma. Una versión en Swift/Objective-C para iOS, otra en Kotlin/Java para Android. Máximo rendimiento y acceso completo al hardware.

Ej: La app de iOS de Instagram no es el mismo código que la de Android

TIPO**Híbrida (Cross-platform)**

Un solo código base que genera versiones para iOS y Android. Más eficiente de desarrollar y mantener. Puede haber diferencias menores de rendimiento.

Ej: Apps hechas con React Native o Flutter — un código, dos plataformas

TIPO**PWA**

Progressive Web App — es una web que se comporta como una app. Se puede "instalar" desde el navegador, funcionar offline, y recibir notificaciones push. No pasa por la App Store.

Ej: Twitter Lite, Starbucks, Pinterest

PARA PENSAR**¿Por qué importa?**

Cada enfoque impacta en el pipeline DevOps. Una app nativa necesita builds separados. Una PWA se despliega como una web más. Una híbrida comparte buena parte del proceso.

LA PREGUNTA CLAVE: ¿DÓNDE CORRE EL CÓDIGO?

TIPO	¿DÓNDE CORRE?	¿CÓMO SE ACTUALIZA?	¿NECESITA INTERNET?
Móvil nativa	En el dispositivo (iOS/Android)	Nueva versión en la App Store / Google Play	Solo para datos y sincronización
Escritorio	En tu computadora	Actualizás el instalador, el update automático	Solo para datos
PWA	En el navegador, con cache local	El servidor publica la nueva versión — se detecta automáticamente	Funciona parcialmente offline
Web	En el navegador (código viene del servidor en cada visita)	Automático — el servidor ya tiene la nueva versión	Sí, siempre

LO QUE DISTINGUE A LAS APPS WEB

En una app móvil o de escritorio, **la interfaz está instalada en el dispositivo**. El servidor solo responde consultas. En una app web, el servidor también entrega la interfaz en cada visita — el navegador recibe HTML, CSS y JavaScript frescos, los ejecuta localmente, y el usuario ve la pantalla.

Eso tiene implicancias concretas:

Sin instalación

El usuario no baja nada. Entrás a la URL y la app está. Eso reduce fricción, pero también significa que cada visita depende del servidor.

Deploy instantáneo

Cuando el equipo sube una nueva versión, todos los usuarios la ven de inmediato. No esperan que el usuario acepte un update. No hay fragmentación de versiones.

CDN para los archivos estáticos

Los archivos JS, CSS e imágenes suelen servirse desde un CDN — servidores distribuidos geográficamente. Alguien en Uruguay recibe esos archivos del servidor más cercano, no del datacenter original.

Frontend como microservicio

En arquitecturas avanzadas, el frontend también corre en contenedores — en Kubernetes, por ejemplo. El "frontend web" deja de ser solo un S3 y pasa a ser un servicio más en el cluster.

EL BACKEND: SIEMPRE PRESENTE, SIN IMPORTAR EL FRONTEND

Hay algo que todas las apps modernas tienen en común: **un backend que tiene que estar siempre encendido**. Instagram se instala en tu teléfono, pero se conecta a servidores del otro lado del mundo para cargar tus fotos. WhatsApp vive en tu celular, pero tus mensajes pasan por servidores de Meta.

El tipo de app define la interfaz, no la necesidad de infraestructura.

```

|
|
API REST – el contrato común
|
|
[ Backend – lógica de negocio ]
|
|
[ Base de Datos ]

```

Todos los clientes — iOS, Android, web, escritorio — hablan con el **mismo backend** a través de la misma API. El backend no sabe ni le importa si quien lo llama es un navegador, una app móvil, u otro servicio. Eso es lo que hace que las APIs sean tan poderosas: separan la interfaz de la lógica.

LA MISMA LLAMADA DESDE CUALQUIER CLIENTE

El backend no sabe ni le importa quién lo llama. Una app iOS, una web en el navegador, y un script de Python usan exactamente el mismo mecanismo para comunicarse con él. El código cambia de lenguaje, pero el mensaje HTTP es idéntico:

```
// Web – JavaScript en el navegador
const res = await fetch('https://api.mibanco.com/cuenta/saldo', {
  headers: { Authorization: `Bearer ${token}` }
})
const data = await res.json()
console.log(data.saldo) // → 47350

// Mobile – React Native (mismo código, distinta pantalla)
const res = await fetch('https://api.mibanco.com/cuenta/saldo', {
  headers: { Authorization: `Bearer ${token}` }
})
const data = await res.json()
setText(`Saldo: ${data.saldo}`) // actualiza componente nativo

# Script Python – mismo endpoint, misma respuesta
import requests
res = requests.get(
    'https://api.mibanco.com/cuenta/saldo',
    headers={'Authorization': f'Bearer {token}'})
print(res.json()['saldo']) # → 47350

```

Los tres llaman al mismo backend, reciben la misma respuesta JSON y el resultado es el mismo número. Lo único que cambia es cómo cada cliente *muestra* ese número. El backend no tiene que saber si está hablando con Chrome, con un iPhone o con un script de automatización.

Esto es lo que hace poderosa a una buena API: es el contrato que permite que equipos distintos construyan interfaces distintas sobre la misma lógica de negocio, sin coordinarse en cada cambio.

¿POR QUÉ IMPORTA ESTO PARA EL EQUIPO DE INFRAESTRUCTURA?

Que el backend tenga que estar siempre arriba no alcanza. Tiene que:



Siempre encendido

24/7 — una caída afecta a todos los frontends al mismo tiempo



Escalable

El Mundial empieza y de golpe hay 10x más tráfico del normal



Seguro

Protegido contra ataques — es el punto de entrada a todos los datos

Ese es uno de los trabajos de un DevOps: asegurarse de que los servidores estén arriba, escalados y seguros — sin importar si la app que los usa es web, móvil o de escritorio. En AWS vamos a ver los servicios que hacen exactamente eso.

¿SON EXCLUYENTES LOS TIPOS DE APP?

No. **Spotify** tiene app de escritorio, app móvil (nativa en iOS y Android) Y versión web — las tres apuntan al mismo backend. Lo que cambia es la interfaz y dónde corre el código del lado del cliente. La lógica de negocio, la base de datos, la autenticación — todo eso es compartido.

Eso también significa que un bug en el backend afecta a todos los frontends al mismo tiempo. Y que mejorar el rendimiento del backend mejora la experiencia en todos los dispositivos.