

CONCEPTO 3 DE 4

Anatomía de una app web

Frontend, Backend, Base de datos y API — las piezas que forman cualquier app moderna

EL MODELO MENTAL: EL CAJERO AUTOMÁTICO

Cuando usás un cajero interactuás con una pantalla, botones y un teclado — eso es el **frontend**. Detrás, un servidor del banco verifica quién sos y aplica las reglas — eso es el **backend**. Tu saldo, tus movimientos y tu cuenta viven en algún lugar aunque todo se apague — eso es la **base de datos**. Y la forma en que el cajero le habla al servidor del banco sigue un contrato definido — eso es la **API**.

LAS TRES CAPAS PRINCIPALES



Frontend

Todo lo que ves e interactuás. Corre en el navegador del usuario — no en el servidor.

HTML — estructura

CSS — estilo

JavaScript — interactividad



Backend

El servidor que procesa los pedidos, aplica la lógica de negocio y responde. Nunca lo ves directamente.

Python

Node.js

Java

Go



Base de datos

Almacenamiento persistente. Si apagás el servidor, los datos siguen ahí. El backend la consulta para leer y escribir.

PostgreSQL

MySQL

MongoDB

DynamoDB

FRONTEND EN DETALLE: HTML, CSS Y JAVASCRIPT

HTML define qué elementos existen en la página. **CSS** define cómo se ven. **JavaScript** define cómo se comportan. Los tres siempre juntos:

```
<!-- HTML: estructura -->
<button id="btn-saldo">Consultar saldo</button>

/* CSS: estilo */
#btn-saldo { background: #6D28D9; color: white; padding: 12px 24px; }
```

```
// JavaScript: comportamiento
document.getElementById('btn-saldo').addEventListener('click', () => {
  // llama a la API del backend
  fetch('api.mibanco.com/cuenta/saldo')
})
```

El frontend corre completamente en el dispositivo del usuario. Podés abrir las DevTools del navegador (F12) y ver el HTML, CSS y JS de cualquier sitio que visites.

BACKEND EN DETALLE: LÓGICA DE NEGOCIO

La lógica de negocio son las reglas de la app. Tienen que vivir en el backend — no en el frontend — porque el usuario podría manipular el código del frontend desde las DevTools.

- Verificar que el usuario esté autenticado antes de mostrar datos sensibles
- No permitir comprar más unidades de las que hay en stock
- Calcular el precio final con impuestos, descuentos y costos de envío
- Enviar notificaciones cuando se realiza una compra

BASE DE DATOS: SQL VS NOSQL

SQL (relacional)

Datos en tablas con columnas fijas. Las tablas se relacionan entre sí. Muy buenas cuando los datos tienen estructura clara y relaciones.

PostgreSQL, MySQL, SQLite

NoSQL

Documentos, clave-valor o grafos. Estructura flexible. Ideal para datos que cambian de forma, o cuando necesitás escalar horizontalmente.

MongoDB, Redis, DynamoDB (AWS)

Una consulta SQL típica luce así:

```
-- Traer el saldo de la cuenta número 1234
SELECT saldo, titular FROM cuentas
WHERE id = 1234
  AND activa = true;
```

API: EL CONTRATO ENTRE SISTEMAS

DEFINICIÓN

API (Application Programming Interface) — el conjunto de operaciones que un backend pone a disposición para que otros sistemas lo usen. Define qué podés pedir, cómo pedirlo, y qué vas a recibir a cambio.

La mayoría de las APIs web modernas siguen el estilo **REST** y devuelven datos en formato **JSON** — un formato de texto estructurado que tanto humanos como máquinas pueden leer:

```
// Respuesta de GET api.mibanco.com/cuenta/1234
{
  "titular": "Micaela Marrero",
  "saldo": 47350,
  "moneda": "UYU",
  "ultima_operacion": "2024-01-10T14:22:00Z"
}
```

SISTEMA QUE LLAMA	LLAMA A	PARA QUÉ
Sitio con "Sign in con Google"	API de Google	Verificar identidad. El sitio nunca ve tu contraseña de Google.
App del clima	API meteorológica	"¿Qué temperatura hay en Montevideo ahora?" La app no tiene satélites propios.
Tienda online	API de MercadoPago	Procesar el cobro. La tienda nunca toca tu tarjeta directamente.

¿CÓMO SABE EL FRONTEND DÓNDE ESTÁ EL BACKEND?

El frontend tiene guardada en su configuración la dirección del backend — igual a como agendás un contacto en el celular. No memorizás el número, buscás el nombre y el teléfono sabe a qué número llamar.

En el mundo del desarrollo esa "agenda de contactos" se llama **variables de entorno**. Son pares clave-valor que viven fuera del código, en archivos de configuración como `.env` :

```
# archivo .env del frontend
API_URL=https://api.mibanco.com
TIMEOUT_MS=5000
```

¿Por qué no poner la dirección hardcodeada en el código? Porque la dirección del backend cambia según el ambiente: en tu computadora es `localhost:3000` , en staging es `api-staging.mibanco.com` , en producción es `api.mibanco.com` . Las variables de entorno te permiten tener el mismo código y solo cambiar la configuración según dónde estás corriendo.