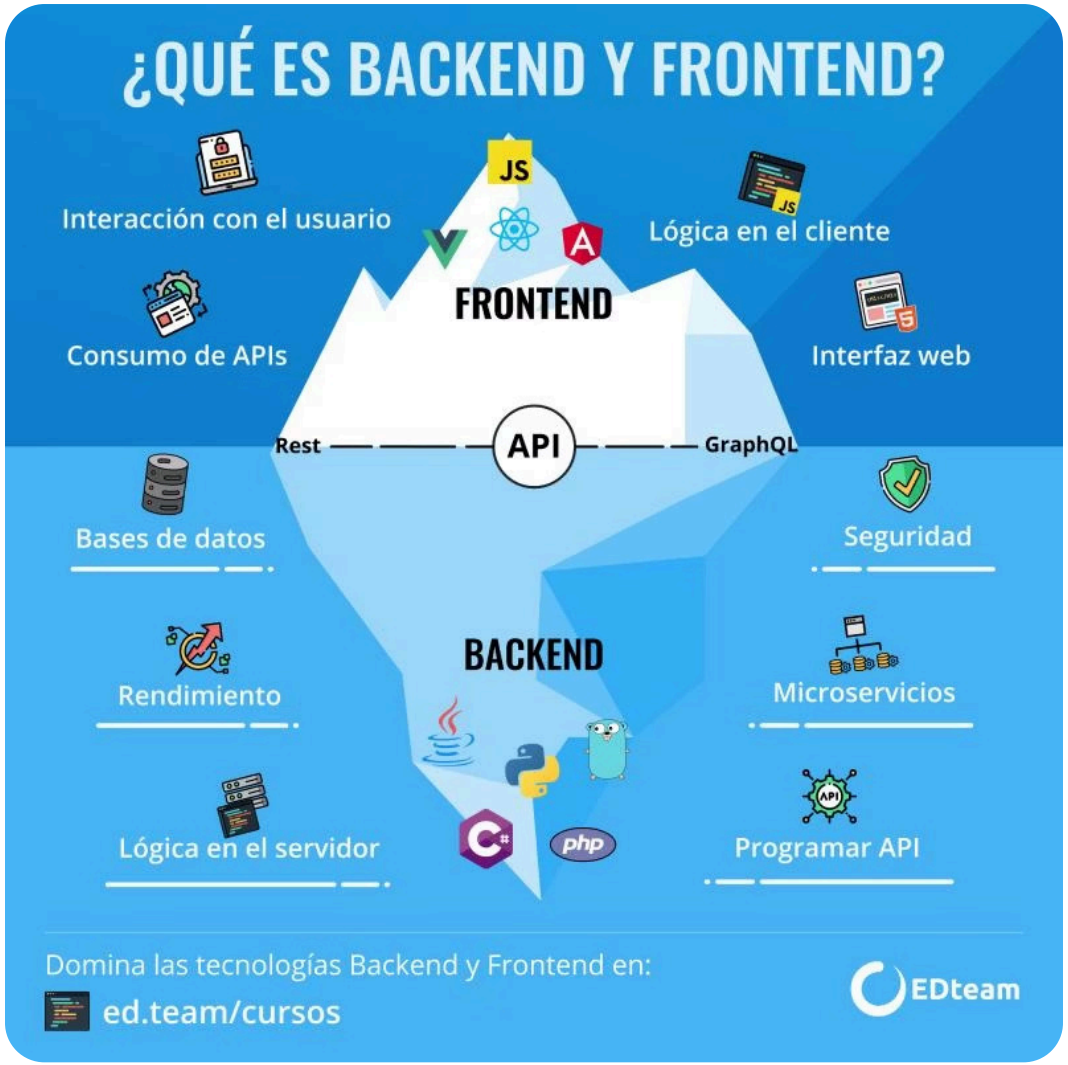
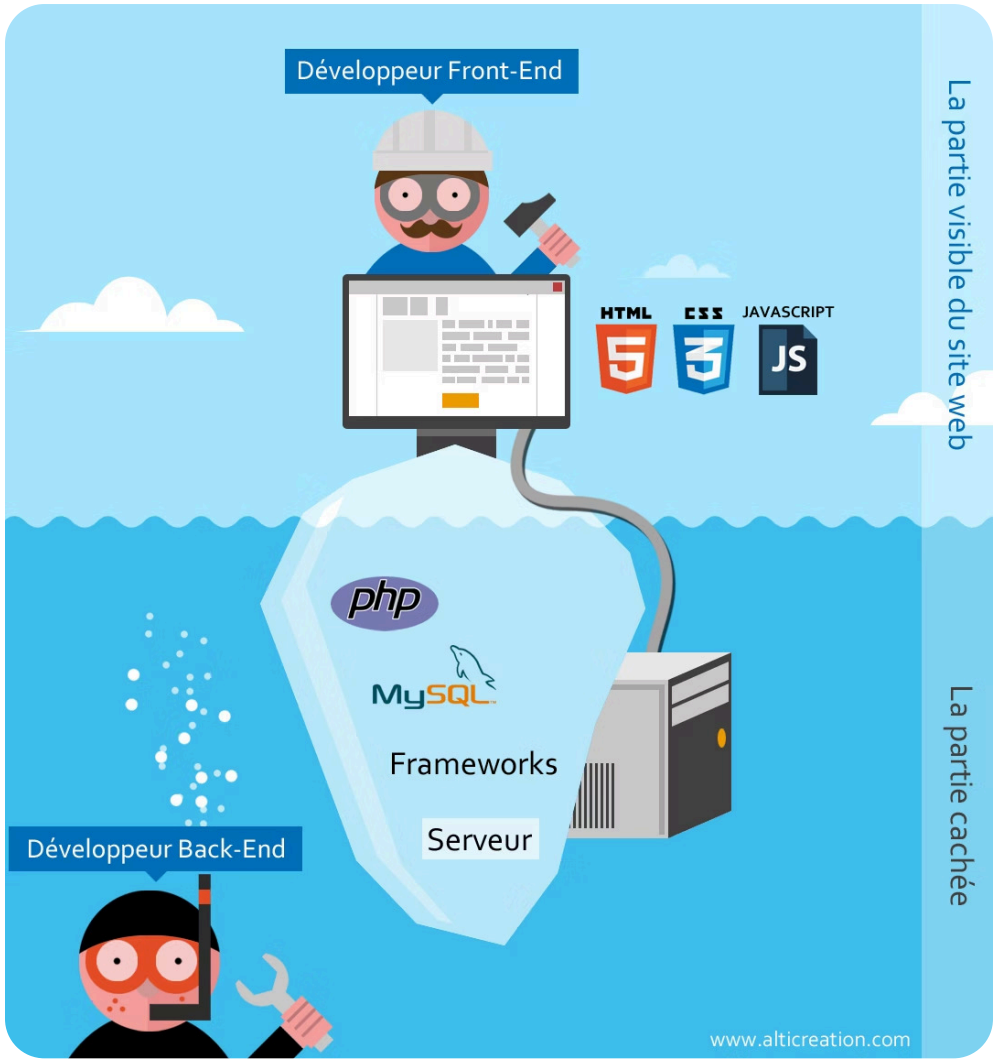
Formación profesional: Especialización en Programador Web

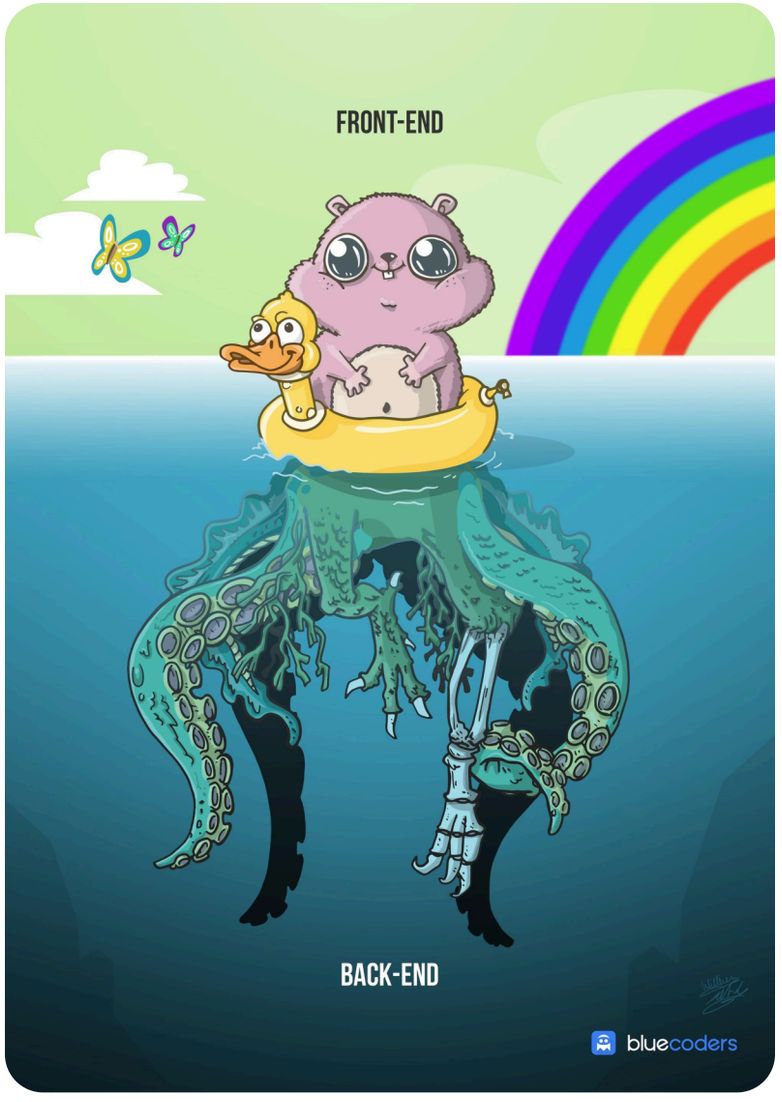
PROF: KATHERINE MOLINA

El Ecosistema Digital

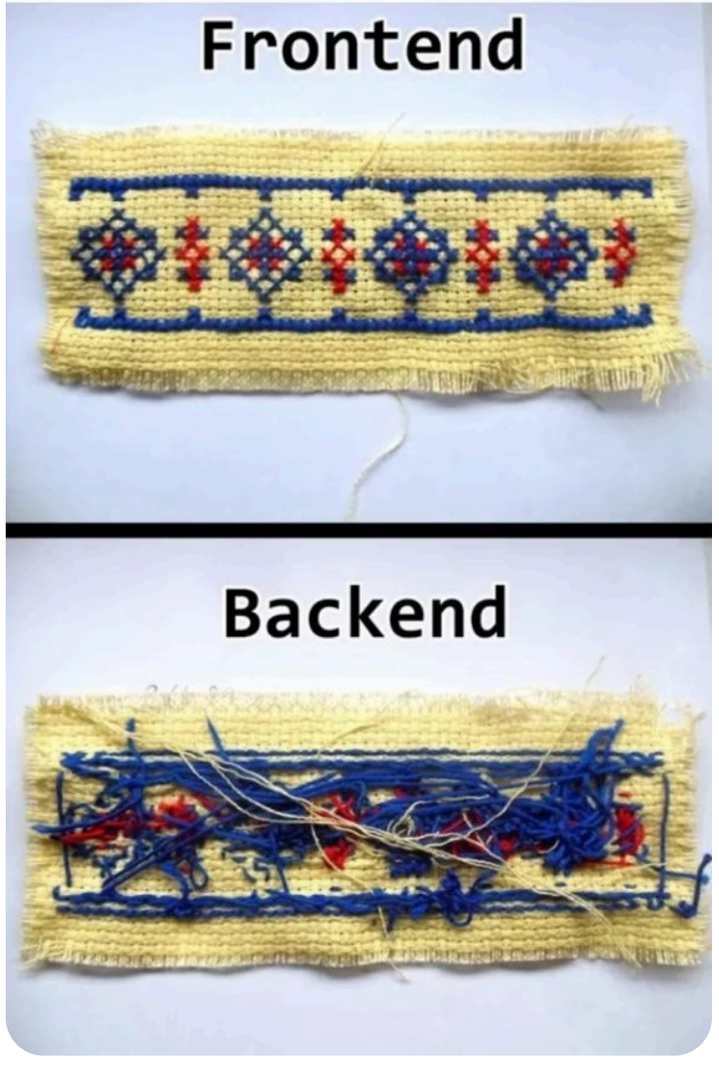
La Web es un iceberg: el usuario solo ve e interactua con lo que ve en las pantallas, en su telefono, computadora, tablet o televisor, pero bajo la superficie ocurre la verdadera magia. Cada clic, cada búsqueda y cada interacción activa un sistema complejo de capas que trabajan en conjunto para entregar resultados en **milisegundos**.







Front-end vs Back-end



🏧 Frontend

El cajero automático o la app del banco. La interfaz amigable donde el usuario interactúa directamente para ver su saldo, transferir dinero o pagar servicios.

🔐 Backend

El sistema bancario que valida tu PIN, verifica fondos disponibles y procesa operaciones de forma segura. Todo ocurre detrás de escena, invisible para el usuario.

💰 Base de Datos

Donde están almacenadas las cuentas, saldos, movimientos y transacciones. El repositorio central que garantiza que tu dinero está seguro y disponible cuando lo necesitas.

Analogía: El Restaurante

Otra forma sencilla de entender la arquitectura web es compararla con el funcionamiento de un restaurante.

🍽️ Frontend

El comedor del restaurante. Lo que el cliente ve y con lo que interactúa: el menú, la decoración, los meseros, la experiencia visual. Es la interfaz amigable donde el usuario hace su pedido.

👨‍🍳 Backend

La cocina del restaurante. Donde ocurre toda la magia: los chefs procesan los pedidos, validan ingredientes disponibles, preparan los platos siguiendo recetas (lógica de negocio) y aseguran que todo sea seguro e higiénico.

📦 Base de Datos

La despensa del restaurante. Donde se almacenan todos los ingredientes, inventario y recursos. Sin una despensa bien organizada, la cocina no puede funcionar ni el restaurante puede servir a sus clientes.

Frontend: Lo que el usuario Ve

El frontend es todo aquello con lo que el usuario interactúa directamente en el navegador: botones, formularios, imágenes, animaciones y la estructura visual de la página. Es la capa más visible y la primera impresión de cualquier aplicación web.

HTML

La estructura y el esqueleto. Define los elementos de la página: títulos, párrafos, imágenes, enlaces y formularios. Es el esqueleto sobre el que se construye todo lo demás.

CSS

El diseño y la estética visual. Controla colores, tipografías, espaciado, animaciones y la apariencia general. Es lo que transforma el esqueleto en una experiencia visual atractiva.

JavaScript

El motor de interactividad. Hace que la página responda al usuario: valida formularios, carga contenido dinámico, anima elementos y comunica con el servidor sin recargar la página.

HTML ¿Qué hay en la página?

Centro de Formación Profesional

Programador Web

Clase 03/06

Esta página está hecha **solo con HTML**. No tiene CSS ni JavaScript. En el celular se ve simple y sin diseño, pero ya tiene estructura.

HTML es el esqueleto: define *qué* hay, no cómo se ve ni cómo se mueve.

← Volver al inicio

Documentación HTML (MDN)

Sección 1: ¿Qué aprenderemos?

Etiquetas: h1, h2, p, a, img y div.

Organizar contenido en bloques con divs.

Sección 2: Imagen

Imagen con la etiqueta img (ancho 100% del celular):

HTML: Estructura Básica

El lenguaje HTML utiliza etiquetas (tags) para estructurar el contenido de una página web. Cada etiqueta tiene un propósito específico, como definir encabezados, párrafos, enlaces o imágenes.

```
<!DOCTYPE html>
<html>
<head>
  <title>Mi Primera Página</title>
</head>
<body>
  <h1>¡Hola, Mundo!</h1>
  <p>Esta es una página web muy simple.</p>
</body>
</html>
```

Este fragmento de código define la estructura esencial de cualquier página HTML, declarando el tipo de documento, el idioma, el título en la pestaña del navegador y un encabezado junto con un párrafo visible en el cuerpo de la página.

CSS ¿Cómo se ve?

CSS: Estructura Básica

CSS utiliza reglas para definir cómo se presentan los elementos HTML. Cada regla consiste en un **selector** que apunta a uno o más elementos HTML, y un bloque de **declaraciones** que especifica las propiedades (como color o tamaño de fuente) y sus valores.

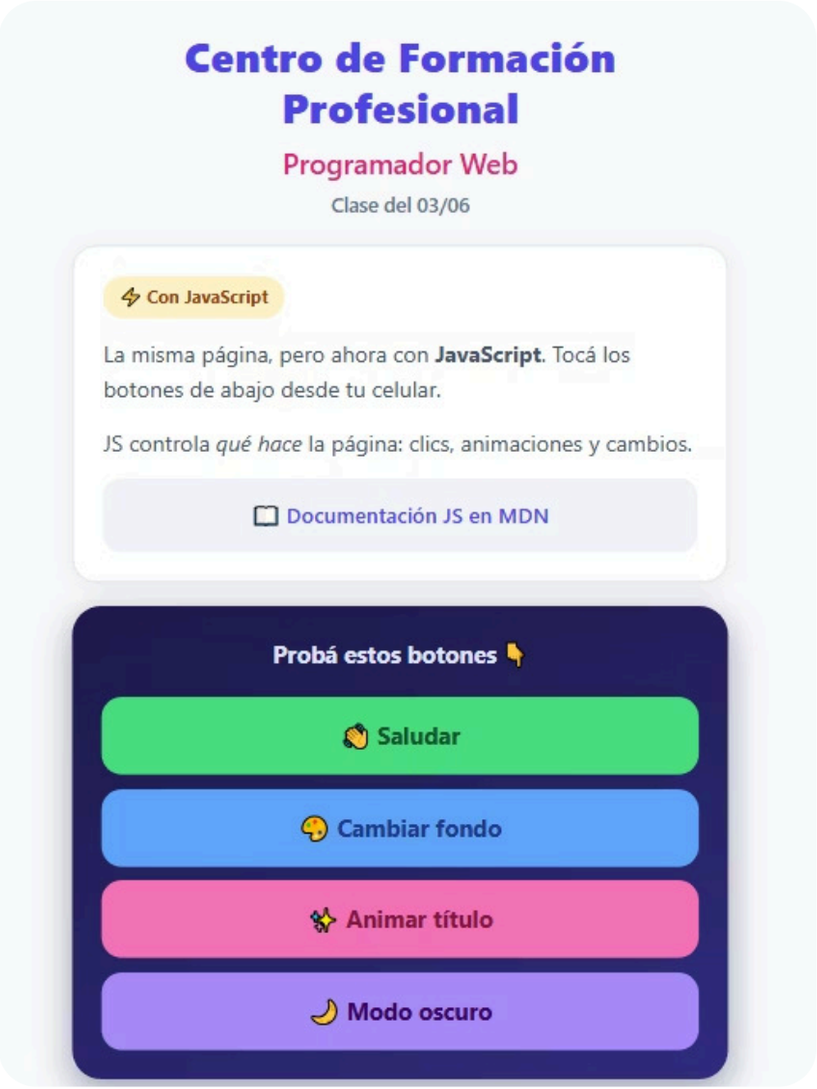
```
/* Este es un comentario en CSS */
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; //establece la fuente del texto.
    background-color: #f8f9fa; //establece un fondo gris muy claro (#f8f9fa).
    color: #343a40; //establece el color principal del texto en un gris oscuro (#343a40).
}

h1 {
    color: #007bff;
    text-align: center; //centra el título horizontalmente
    margin-bottom: 20px; //deja un espacio de 20 píxeles debajo del título.
}

p {
    font-size: 16px; //tamaño de letra de 16 píxeles.
    line-height: 1.6; //aumenta el espacio entre líneas para facilitar la lectura.
    padding: 0 15px; //agrega un espacio interno de 15 píxeles a izquierda y derecha.
}
```

En este ejemplo, se establecen estilos globales para el cuerpo de la página (`body`), se personalizan los encabezados principales (`h1`) y se dan atributos de tipografía y espaciado a los párrafos (`p`), mostrando cómo CSS da vida visual al contenido.

JavaScript ¿Qué puede hacer? ¿Qué hace la pagina cuando el usuario interactúa?



JavaScript: ¡Haciendo la Web Interactiva!

JavaScript es el lenguaje que da vida a las páginas web, permitiendo que respondan a las acciones del usuario y realicen tareas dinámicas sin necesidad de recargar la página. Con JavaScript, podemos crear experiencias ricas y fluidas, desde formularios interactivos hasta animaciones complejas y actualizaciones de contenido en tiempo real.

```
// Obtener el botón y el párrafo por su ID
const botonCambiar = document.getElementById('miBoton');
const parrafo = document.getElementById('miParrafo');

// Añadir un "escuchador de eventos" al botón
botonCambiar.addEventListener('click', function() {
    // Cambiar el texto del párrafo al hacer clic
    parrafo.textContent = '¡El texto ha cambiado gracias a JavaScript!';
    // También podrías mostrar una alerta
    alert('¡Interacción exitosa!');
});

// Nota: En un documento HTML, necesitarías un botón
// <button id="miBoton">Haz clic aquí</button>
// y un párrafo <p id="miParrafo">Texto original</p>
// para que este código funcione.
```

Este fragmento de JavaScript ilustra cómo un script puede detectar el clic de un usuario en un botón (identificado como `miBoton`) y, en respuesta, modificar el contenido de un párrafo (`miParrafo`) y mostrar una alerta. Es un ejemplo sencillo, pero fundamental para entender cómo JavaScript añade interactividad y dinamismo a una página.

Backend: La lógica oculta

El backend es el cerebro de la aplicación. Opera completamente fuera de la vista del usuario, pero es responsable de que todo funcione correctamente, de forma segura y eficiente.

⚡ Procesamiento de Datos

El servidor recibe información, aplica reglas de negocio, realiza cálculos y toma decisiones lógicas antes de devolver una respuesta al cliente.

🔐 Autenticación y Sesiones

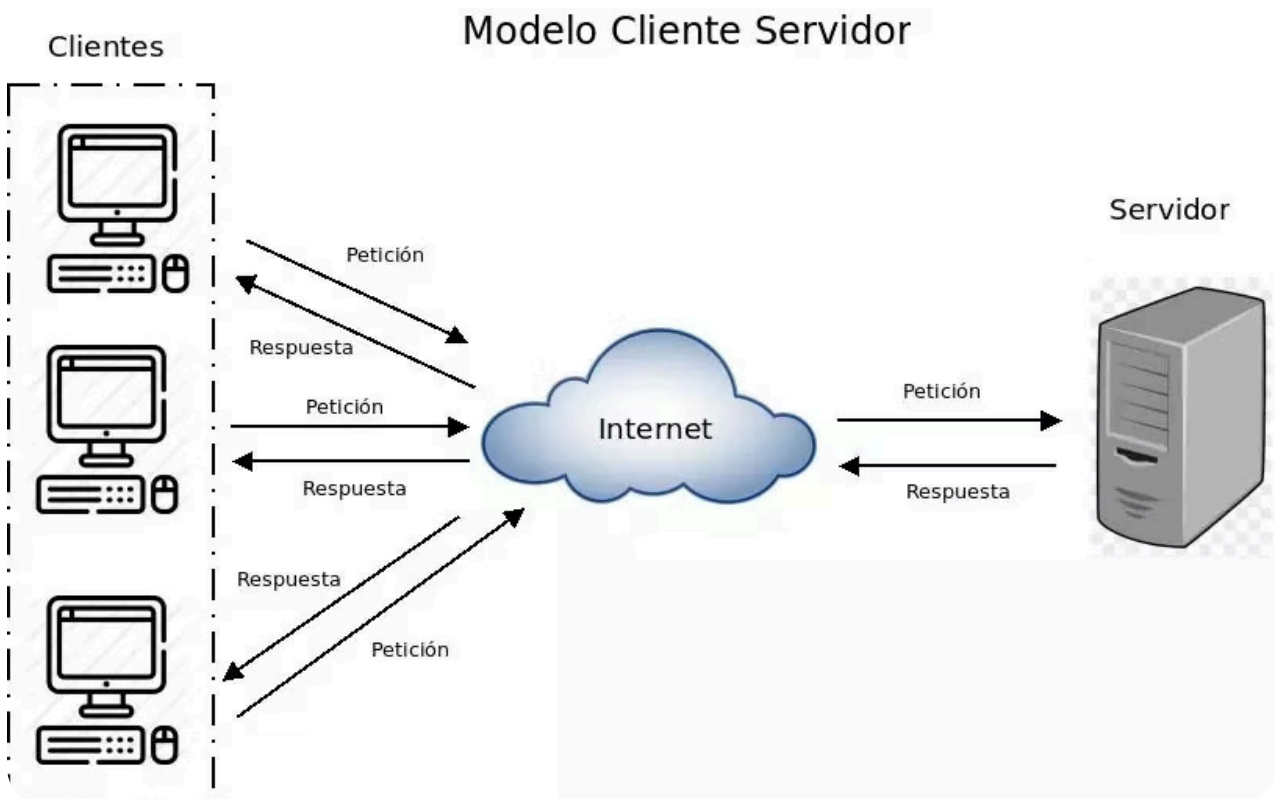
Verifica identidades de usuarios, gestiona inicios de sesión, controla permisos y protege el acceso a recursos sensibles de la aplicación.

🛡️ Seguridad

Protege la aplicación contra ataques, valida datos de entrada, cifra información sensible y asegura que las comunicaciones entre cliente y servidor sean confidenciales.

Comunicación: El Puente Cliente-Servidor

Cada vez que interactúas con una página web, ocurre una conversación instantánea entre tu navegador y un servidor remoto. Este ciclo de **Request/Response** es el fundamento de toda la Web moderna.



Este ciclo ocurre en milisegundos. El protocolo **HTTP** es el lenguaje común que usan el cliente y el servidor para entenderse, y cada interacción en la web —desde cargar una imagen hasta enviar un formulario— sigue este patrón.

Base de Datos: El repositorio de la información

Una base de datos es un sistema organizado para registrar, conservar y consultar datos estructurados de manera eficiente. Sin ella, ninguna aplicación web podría recordar usuarios, productos, pedidos o cualquier información entre sesiones.

BD vs SGBD

La **Base de Datos** es el repositorio de información. El **SGBD** (Sistema de Gestión de Bases de Datos) es el software que la administra: MySQL, PostgreSQL, MongoDB, entre otros.

Modelo Relacional

Organiza los datos en **tablas con relaciones** entre sí, permitiendo consultas masivas eficientes mediante el lenguaje SQL. Es el modelo más utilizado en aplicaciones web.

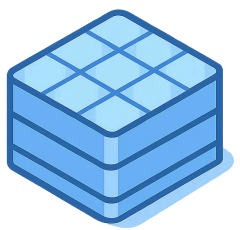
¿Por qué importa?

Sin una base de datos bien diseñada, una aplicación web no puede escalar, mantener consistencia ni garantizar que la información esté disponible cuando se necesita.

- Almacenamiento persistente de datos
- Consultas rápidas y organizadas
- Integridad y seguridad de la información
- Soporte para millones de usuarios

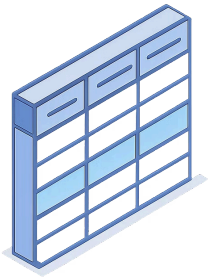
Conceptos Fundamentales de Bases de Datos

Una base de datos organiza la información de manera lógica y eficiente. Entender sus componentes básicos es clave para comprender cómo funciona el backend y cómo se gestionan los datos en cualquier aplicación web.



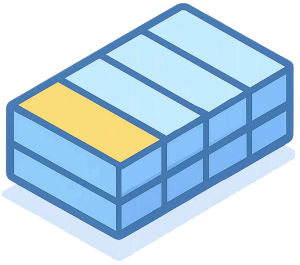
Tabla

Estructura principal. Un conjunto de datos organizados en filas y columnas, similar a una hoja de cálculo. Cada tabla representa una entidad (ej. "Usuarios", "Productos").



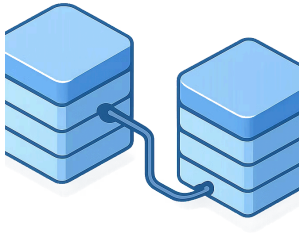
Fila (Registro)

Entrada individual. Cada fila en una tabla representa un único registro o instancia de la entidad. Por ejemplo, en una tabla de "Usuarios", cada fila sería un usuario específico.



Columna (Campo)

Atributo o propiedad. Cada columna define un tipo específico de dato o atributo de la entidad. En una tabla de "Usuarios", las columnas podrían ser "Nombre", "Email", "Contraseña".



Relación

Conexiones entre tablas. Permiten vincular datos entre diferentes tablas, formando una red de información. Por ejemplo, una tabla de "Pedidos" puede estar relacionada con la tabla de "Usuarios" y "Productos".

SQL: El Lenguaje Universal de las Bases de Datos

SQL (Structured Query Language) es el estándar para interactuar con bases de datos relacionales, permitiendo a las aplicaciones almacenar, manipular y recuperar datos de forma eficiente.

¿Qué es SQL?

El lenguaje estándar. Es un lenguaje declarativo diseñado para gestionar y consultar datos almacenados en sistemas de gestión de bases de datos relacionales (SGBDR).

¿Cuándo se inventó?

Orígenes en los 70s. Fue desarrollado en IBM por Donald D. Chamberlin y Raymond F. Boyce a principios de la década de 1970, inicialmente llamado SEQUEL (Structured English Query Language).

¿Por qué se creó?

Necesidad de estandarización. Surgió para ofrecer una forma consistente y poderosa de acceder y manipular grandes volúmenes de datos en sistemas complejos, facilitando la integración y el análisis.

Dato Curioso

El debate de la pronunciación. Aunque sus creadores lo llamaban "SEQUEL", su nombre fue acortado a SQL por motivos legales. Hoy en día, muchas personas lo pronuncian "sequel", mientras otras dicen "esse-cue-ell".

Comandos Básicos de SQL

Aquí tienes una muestra de los comandos más utilizados para interactuar con tus datos:

SELECT

```
SELECT nombre, email FROM Usuarios WHERE edad > 30;
```

Recupera datos de una o más tablas en la base de datos, filtrando según condiciones.

INSERT

```
INSERT INTO Productos (nombre, precio) VALUES ('Laptop', 1200.00);
```

Añade nuevas filas de datos a una tabla existente.

UPDATE

```
UPDATE Clientes SET ciudad = 'Madrid' WHERE id = 101;
```

Modifica datos existentes en una tabla, basándose en criterios específicos.

DELETE

```
DELETE FROM Pedidos WHERE fecha < '2023-01-01';
```

Elimina filas de datos de una tabla.

Ejemplo:

Código	*Nombre*	*Posición*	*Salario*
1	Edgardo Trujillo	Gerente	19000
2	Lidimarie Fonsi	Empleada	12000
3	Jean Piaget	Empleado	13500
4	Jerome Bruner	Empleado	14000

Diagrama de una base de datos de un restaurante

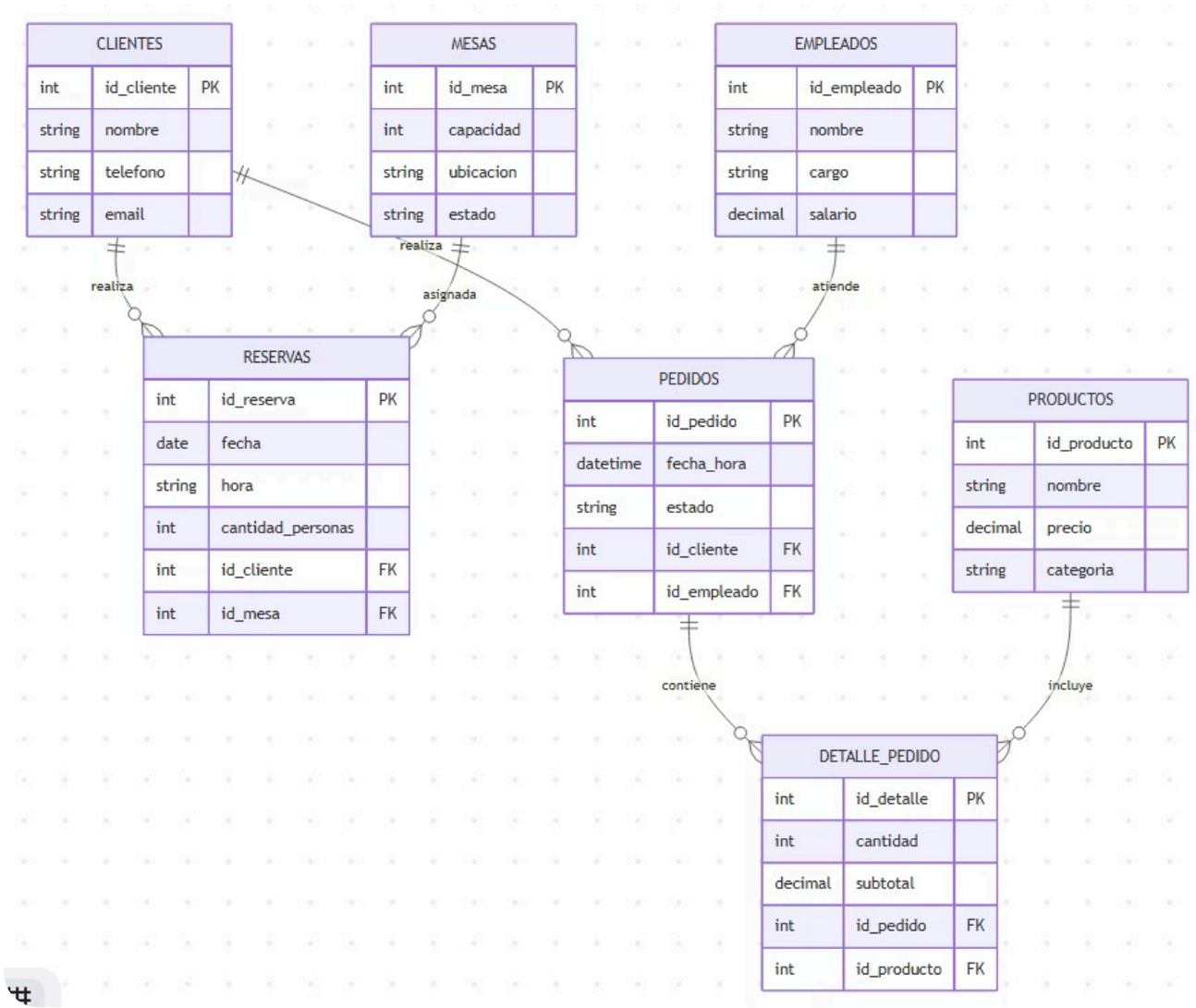


Diagrama de una base de datos de un cine

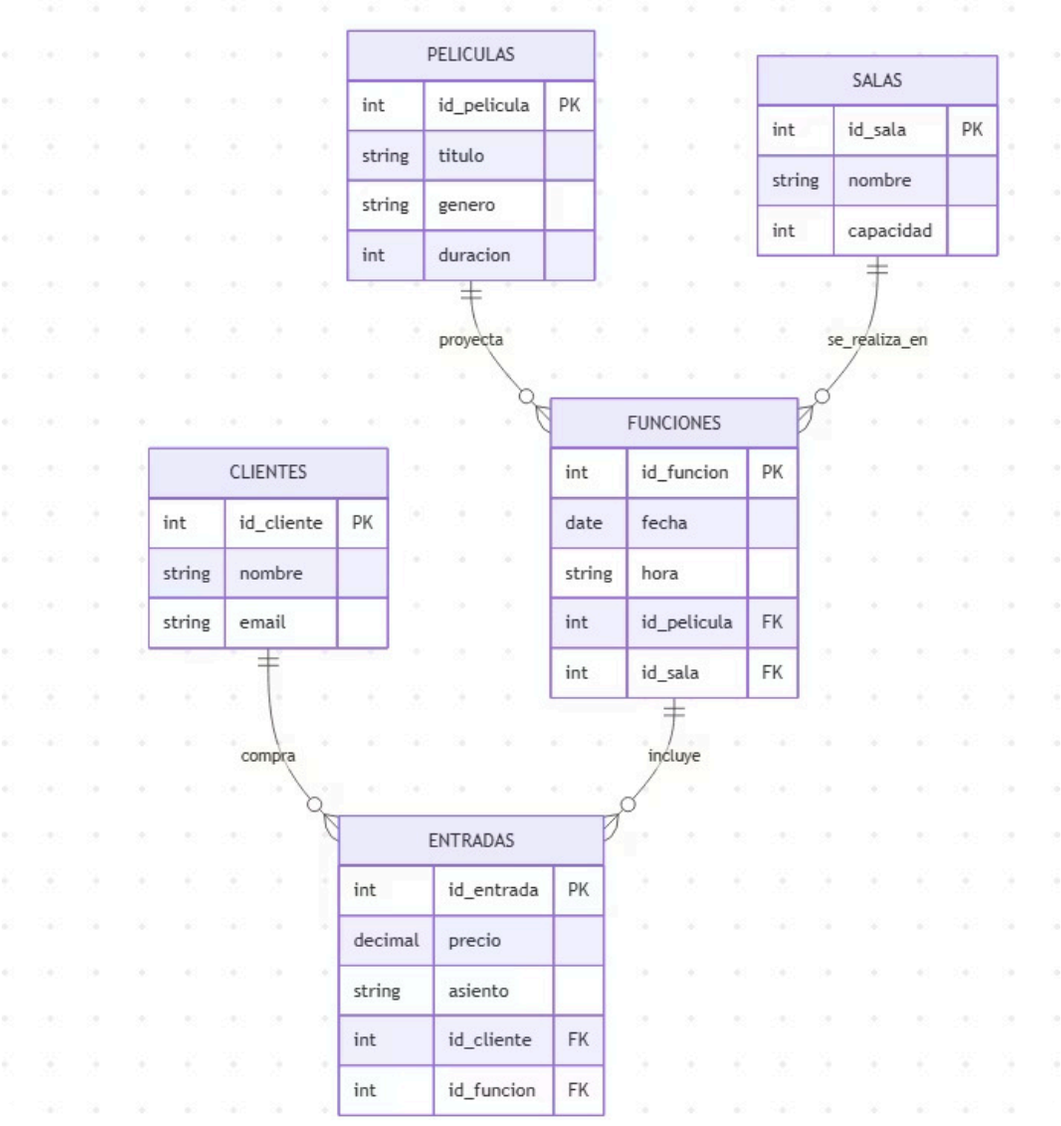


Diagrama de una base de datos de una tienda online

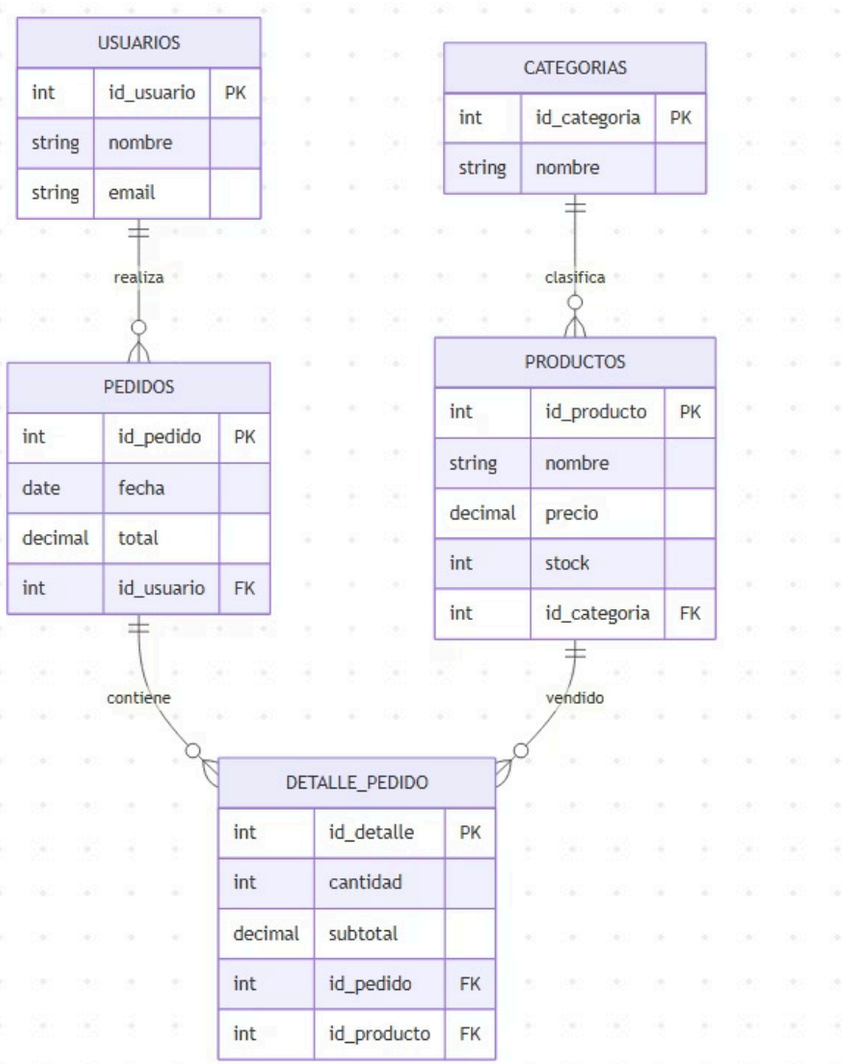
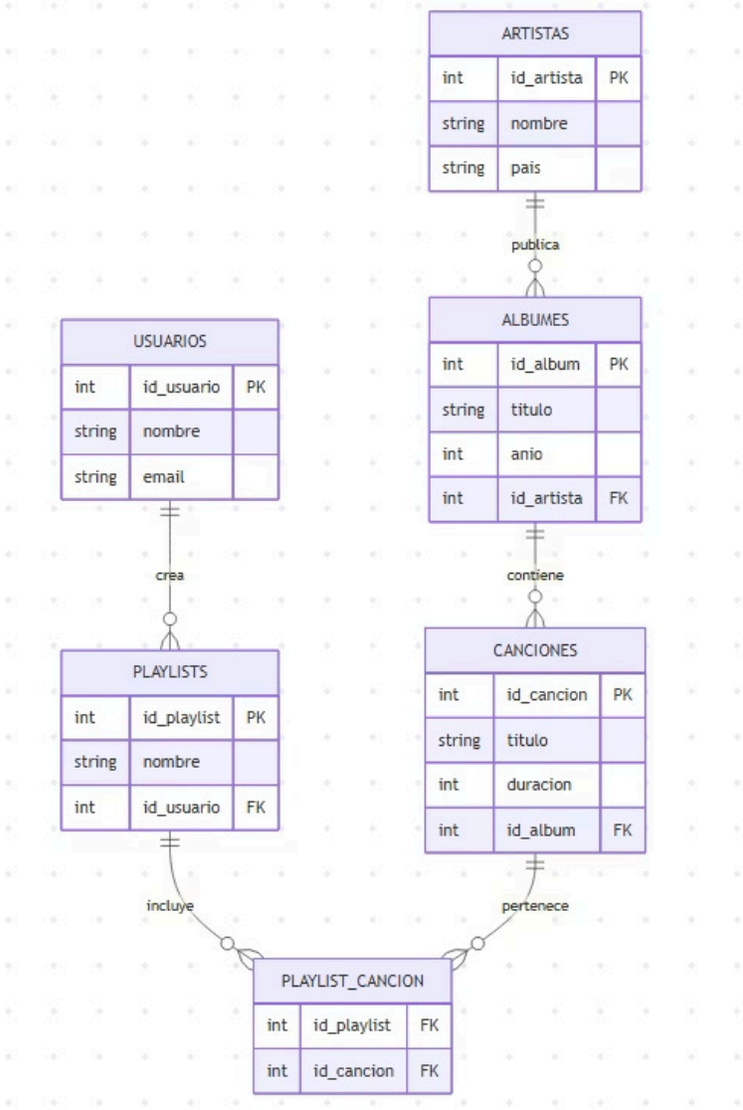


Diagrama de una base de datos de Spotify



El flujo de trabajo completo

Cada acción del usuario desencadena una cadena de eventos que recorre las tres capas de la arquitectura web en cuestión de milisegundos. Entender este flujo es clave para pensar como programador.

1. El Usuario Actúa

Hace clic en un botón o envía un formulario desde el **Frontend**.

2. El Backend Procesa

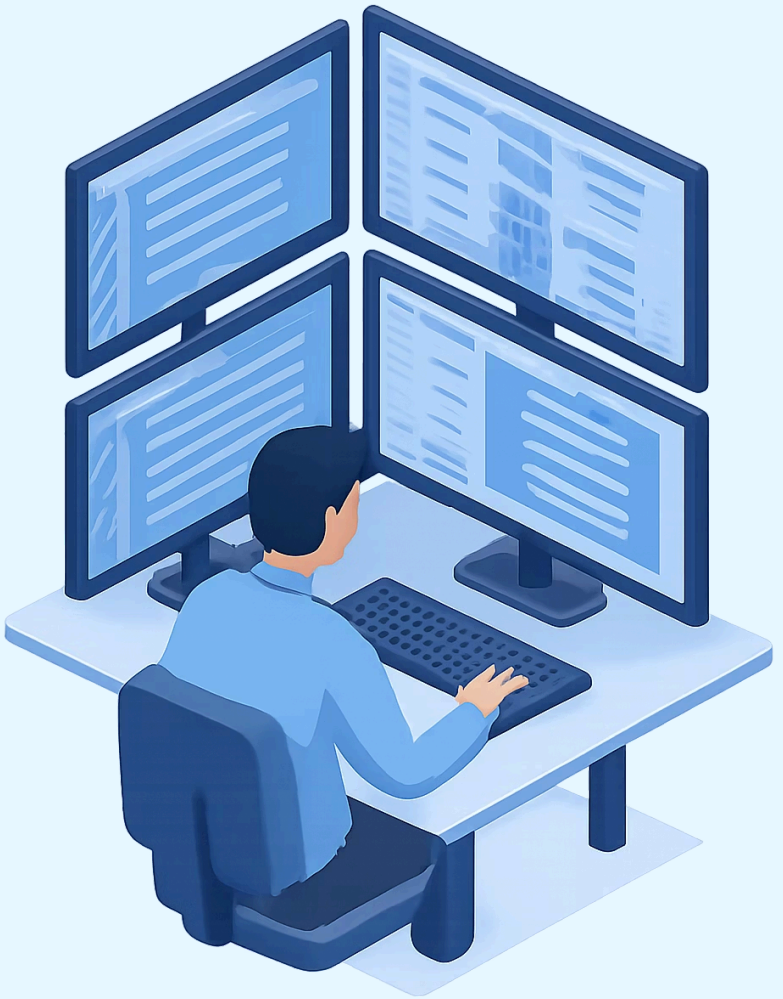
El servidor recibe la petición, valida datos y aplica la **lógica de negocio**.

3. La BD Responde

Se guarda o recupera información de la **Base de Datos** de forma estructurada.

4. Resultado en Pantalla

El **Frontend** renderiza la respuesta actualizada para el usuario.



La Evolución del Desarrollador

Un programador web completo no solo escribe código: comprende cómo se conectan todas las piezas del sistema.

🌐 Desarrollo Full-Stack

Comprender la infraestructura completa —frontend, backend y base de datos— permite crear productos escalables, eficientes y robustos. El desarrollador full-stack puede tomar decisiones técnicas informadas en cualquier capa del sistema.

🏗️ Arquitectura de Tres Capas

La separación en capas (presentación, lógica de negocio y datos) es el estándar de la industria. Facilita el mantenimiento, la escalabilidad y el trabajo en equipo, ya que cada capa tiene responsabilidades claras y definidas.

¿Cuál es el Rol del Programador?

El programador web es mucho más que alguien que escribe código. Es un **resolutor de problemas** que usa la tecnología como herramienta para crear soluciones reales.

Analizar

Comprender el problema antes de escribir una sola línea de código. Definir requerimientos, identificar usuarios y mapear el flujo de la aplicación.

Programar

Escribir código limpio, eficiente y mantenible siguiendo buenas prácticas, estándares y patrones de la industria.

Diseñar

Planificar la arquitectura, elegir las tecnologías adecuadas y definir cómo se organizarán los datos, la lógica y la interfaz del usuario.

Probar y Mejorar

Validar que el sistema funciona correctamente, detectar errores, optimizar el rendimiento y iterar basándose en el feedback real.

Tu Camino Hacia el Código

La programación web no es solo aprender lenguajes y herramientas. Es desarrollar una **mentalidad de resolución de problemas** y entender el sistema completo antes de construirlo.

01

Entender la Arquitectura

Conocer cómo funciona el ecosistema web: frontend, backend y base de datos trabajando juntos.

02

Conocer el Rol del Programador

Internalizar que programar es analizar, diseñar, construir y mejorar — no solo escribir código.

03

Analizar y Diseñar

Antes de programar, siempre hay que entender el problema, planificar la solución y definir la estructura.

04

Programar con Propósito

Escribir código que resuelva problemas reales, con calidad, claridad y una visión arquitectónica sólida.

Recuerda: Todo gran desarrollador comenzó exactamente donde estás tú ahora. La clave es la curiosidad, la práctica constante y nunca dejar de aprender 😊