


ARQUITECTURA DE UNA APLICACIÓN WEB — INTERACCIÓN ENTRE FRONTEND Y API



Solorzano Rodas, Julio Andres
[COMPANY NAME] [Company address]

¿Qué es la arquitectura por capas en una aplicación web?

La **arquitectura por capas** es una forma de organizar una aplicación dividiéndola en diferentes niveles o “capas”, cada una con una función específica.

El objetivo es **separar responsabilidades**, facilitar el **mantenimiento**, la **escalabilidad** y permitir que distintas partes se desarrollen o actualicen sin afectar a las demás.

Las tres capas principales en una aplicación web moderna son:

1. Capa de presentación (Frontend)

- Es la **interfaz de usuario**, lo que el usuario ve y con lo que interactúa.
- Está construida con **HTML, CSS y JavaScript**, y puede usar frameworks como **React, Vue, Angular, Svelte**, etc.
- Su función es mostrar la información proveniente del backend y capturar las acciones del usuario (clics, formularios, búsquedas, etc.).

Ejemplo:

Cuando un usuario abre un sitio de compras, el frontend muestra el catálogo de productos y permite añadir artículos al carrito.

2. Capa lógica / aplicación (Backend o API)

- Es el **cerebro de la aplicación**, donde se procesan las reglas de negocio.
- Se encarga de recibir las solicitudes del frontend, procesarlas y devolver las respuestas.
- Está desarrollada con lenguajes como **Node.js, Python (Django/Flask), Java (Spring Boot), PHP (Laravel), .NET**, etc.
- Expone **endpoints** (rutas) a los que el frontend puede acceder mediante **peticiones HTTP**.

Ejemplo:

El backend tiene un endpoint `/api/productos` que, al ser consultado, obtiene los productos desde la base de datos y los devuelve en formato JSON.

3. Capa de base de datos

- Es donde se **almacenan los datos** permanentes de la aplicación: usuarios, pedidos, productos, mensajes, etc.
 - Puede ser una base **relacional** (MySQL, PostgreSQL) o **no relacional** (MongoDB, Firebase).
 - El backend se comunica con la base de datos mediante **consultas (queries)** para leer, insertar, modificar o eliminar datos.
-

Comunicación entre Frontend y API

1. El **Frontend** realiza una **petición HTTP (request)** al servidor.
2. Esa petición se dirige a un **endpoint** del **API**, por ejemplo:
GET <https://api.tienda.com/productos>
3. El **Backend** procesa la solicitud, consulta la base de datos y genera una **respuesta (response)** en formato **JSON**.
4. El **Frontend** recibe ese JSON y actualiza la interfaz mostrando los datos al usuario.

Conceptos clave:

- **HTTP (HyperText Transfer Protocol):** protocolo que define cómo se envían y reciben los datos.
- **Request:** la solicitud que el frontend hace al servidor (puede incluir parámetros, headers, cuerpo, etc.).
- **Response:** la respuesta que el servidor envía de vuelta al frontend.
- **Endpoint:** la dirección (URL) específica dentro del API donde se atiende una petición.
- **JSON (JavaScript Object Notation):** formato de texto ligero usado para intercambiar datos entre frontend y backend.


Diagrama del flujo completo

 Usuario


↓

 Frontend (Interfaz web / app)


↓ [HTTP Request: GET /api/productos]

 API / Backend (Lógica de negocio)


↓ [Consulta SQL o a NoSQL]

 Base de datos (Datos almacenados)

↑ [Resultados devueltos]

 API / Backend (Procesa y responde)

↑ [HTTP Response con JSON]

 Frontend (Muestra los datos al usuario)

Ejemplos reales de comunicación Frontend → API

Ejemplo 1: Login de usuario

- El usuario introduce su correo y contraseña.
- El frontend envía una **petición POST** a /api/login con el cuerpo:

```
{ "email": "usuario@correo.com", "password": "123456" }
```
- El backend verifica las credenciales en la base de datos y responde:

```
{ "token": "abc123", "nombre": "Juan Pérez" }
```
- El frontend guarda el token y redirige al panel del usuario.

Ejemplo 2: Cargar lista de productos

- El frontend hace una **petición GET** a /api/productos.
- El backend consulta la base de datos y responde:

```
[  
  { "id": 1, "nombre": "Laptop", "precio": 1200 },  
  { "id": 2, "nombre": "Teclado", "precio": 45 }  
]
```

- El frontend muestra los productos en una cuadrícula o lista.

Separar el **Frontend** del **Backend** permite que cada parte evolucione de forma independiente, usando tecnologías y equipos distintos. Esto mejora la **escalabilidad**, ya que se pueden optimizar o desplegar por separado según la demanda. Además, facilita el **mantenimiento y las pruebas**, reduce dependencias entre capas y promueve una arquitectura más limpia y modular.