











Desarrollo Web

Línea de Tiempo del Desarrollo Web

Década de 1990

-  1989: Creación de la World Wide Web por Tim Berners-Lee
-  1990: Primer sitio web
-  1993: Primeros navegadores gráficos
-  1995: JavaScript (Netscape)
-  1998: Google fundado

Década de 2000

-  2004: Web 2.0 comienza
-  2005: YouTube lanzado
-  2007: Primero iPhone
-  2009: Node.js creado
-  2013: React introducido

Arquitecturas de Desarrollo Web

1. Arquitectura Cliente-Servidor

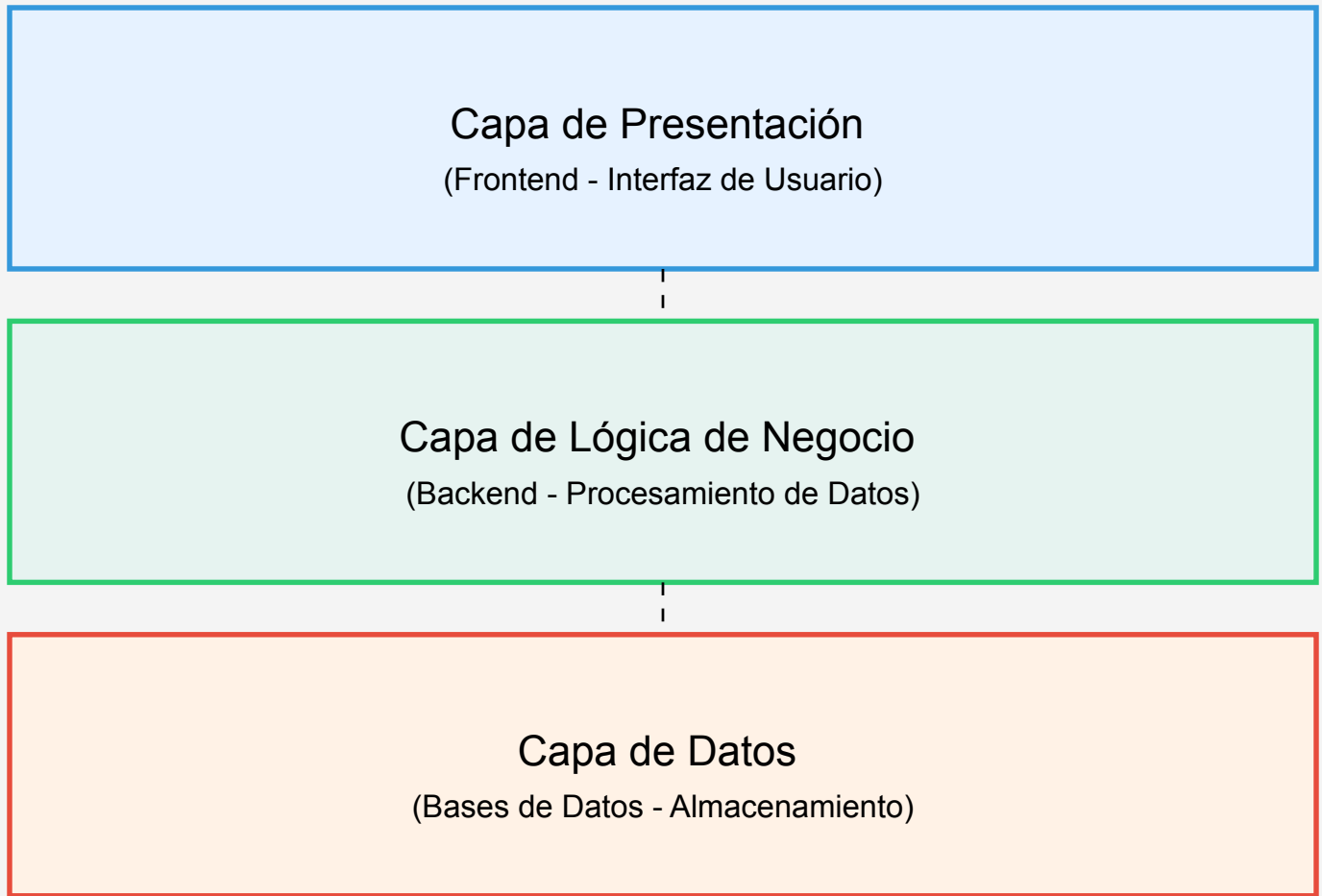


Explicación Detallada de la Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo computacional donde:

- **Cliente:**
 - Dispositivo o aplicación que solicita servicios o recursos
 - Típicamente un navegador web en desarrollo web
 - Responsable de renderizar la interfaz de usuario
 - Envía solicitudes al servidor
 - Procesa y muestra la respuesta recibida
- **Servidor:**
 - Equipo o sistema que proporciona servicios o recursos
 - Procesa solicitudes del cliente
 - Maneja lógica de negocio
 - Gestiona acceso a bases de datos
 - Genera respuestas para el cliente

2. Arquitectura de 3 Capas



Explicación Detallada de la Arquitectura de 3 Capas

La arquitectura de 3 capas divide la aplicación en componentes separados:

1. Capa de Presentación (Frontend):

- Interfaz de usuario visible para el usuario final
- Tecnologías: HTML, CSS, JavaScript
- Frameworks: React, Vue, Angular
- Responsable de la experiencia de usuario
- Envía solicitudes a la capa de negocio
- Renderiza datos recibidos

2. Capa de Lógica de Negocio (Backend):

- Procesa la lógica de la aplicación
- Valida datos de entrada
- Implementa reglas de negocio
- Lenguajes: Python, Java, Node.js, PHP
- Frameworks: Express, Django, Spring Boot
- Comunica la capa de presentación con la de datos
- Gestiona autenticación y autorización

3. Capa de Datos:

- Almacena y gestiona información
- Bases de datos relacionales: MySQL, PostgreSQL
- Bases de datos no relacionales: MongoDB, Firebase
- Gestiona consultas y transacciones
- Asegura integridad de datos
- Implementa mecanismos de seguridad

Ventajas de la Arquitectura de 3 Capas

- Mayor modularidad y separación de responsabilidades
- Facilita el mantenimiento y escalabilidad
- Permite desarrollo independiente de cada capa
- Mejora la reutilización de código
- Simplifica la actualización de componentes

Evolución de Arquitecturas Web

Arquitectura Monolítica

Toda la aplicación en un solo bloque

Arquitectura Cliente-Servidor

Separación básica de cliente y servidor

Arquitectura de 3 Capas

Presentación, Lógica, Datos

Microservicios

Servicios independientes y desacoplados

Serverless

Ejecución bajo demanda en la nube


JAMStack


JavaScript, APIs, Markup

Frameworks Frontend Modernos

React

Librería de Facebook para UI


 Componentes reutilizables


 Virtual DOM


 Ecosistema robusto

Vue.js

Framework progresivo

 Curva de aprendizaje suave


 Flexible y ligero


 Componentes simples

Angular

Plataforma de Google

 Completo y robusto


 TypeScript nativo


 Herramientas integradas

Svelte

Compilador de UI

 Alto rendimiento


 Código más ligero


 Menor overhead

Frameworks Backend Modernos

Node.js

JavaScript en servidor


 No bloqueante


 npm (gestor paquetes)

 Multiplataforma

Express.js

Minimalist web framework

 Routing simple


 Middleware potente


 Altamente extensible

Nest.js

Framework Node.js escalable


 Arquitectura modular


 TypeScript nativo


 Inspirado en Angular

Django (Python)

Framework baterías incluidas

 Seguridad integrada

 ORM potente

 Desarrollo rápido

Seguridad Web



Tokens JWT



OWASP Top 10: Principales vulnerabilidades



Autenticación Sin Contraseña: WebAuthn



Headers de Seguridad: CORS, CSP



Protección DDoS: Mitigación de ataques

Herramientas y Automatización



CI/CD: Integración continua



Testing Automatizado: Jest, Cypress



Containerización: Docker, Kubernetes



Infraestructura como Código: Terraform

Servicios Cloud Más Utilizados

AWS EC2

Máquinas virtuales escalables

AWS Lambda

Computación serverless

AWS S3

Almacenamiento de objetos

AWS RDS

Bases de datos relacionales

AWS DynamoDB

Base de datos NoSQL

AWS API Gateway

Gestión de APIs

Herramientas Esenciales de Desarrollo

Git

Control de versiones

GitHub

Repositorios y colaboración

Docker

Contenedores

Kubernetes

Orquestación de
contenedores

VS Code

Editor de código

Postman

Prueba de APIs

Metodologías de Desarrollo



Scrum



Desarrollo Ágil



DevOps



Test-Driven Development (TDD)



Continuous Integration/Continuous Deployment (CI/CD)

SSR y SCG con Next.js

Explorando cómo Next.js soluciona problemas de SEO y mejora el rendimiento de las aplicaciones web.

¿Qué es SSR (Server-Side Rendering)?

SSR es un enfoque en el que la página web se genera en el servidor y se envía al cliente como HTML ya renderizado. Esto permite que los motores de búsqueda indexen el contenido de la página fácilmente, mejorando así el SEO.

Con SSR en Next.js, las páginas se renderizan en el servidor antes de ser enviadas al navegador, lo que mejora la velocidad de carga y facilita el rastreo por parte de los motores de búsqueda.

¿Qué es SCG (Static Site Generation)?

SCG es un enfoque en el que las páginas se generan en el momento de la compilación y se sirven como archivos estáticos. Esto también mejora el SEO, ya que las páginas ya están listas para ser indexadas por los motores de búsqueda.

En Next.js, SCG permite crear sitios web ultrarrápidos con contenido estático, ideal para páginas que no necesitan actualización constante de datos en tiempo real.

Cómo Next.js resuelve el problema de SEO

Generación de contenido estático: Next.js permite generar páginas estáticas, lo que facilita que los motores de búsqueda indexen el contenido sin depender de JavaScript.

Pre-renderizado: Next.js ofrece tanto SSR como SCG, lo que significa que las páginas pueden ser renderizadas antes de ser enviadas al cliente, asegurando que el contenido esté disponible para los motores de búsqueda.

Optimización automática: Next.js maneja automáticamente la optimización de las imágenes, el código y la carga de recursos, lo que mejora el rendimiento y la experiencia de usuario, factores importantes para el SEO.

Mejor rendimiento: La carga rápida de páginas, que es posible tanto con SSR como con SCG, es un factor clave para una buena puntuación de SEO, ya que Google prioriza las páginas rápidas.

Ventajas de SSR y SCG en Next.js

Mejora del SEO: Al renderizar las páginas del lado del servidor o generar contenido estático, Next.js asegura que los motores de búsqueda puedan rastrear y entender el contenido de las páginas fácilmente.

Mejora en el tiempo de carga: Tanto SSR como SCG permiten tiempos de carga rápidos, lo cual es crucial para la experiencia del usuario y el SEO.

Flexibilidad: Next.js permite elegir entre SSR y SCG según las necesidades del proyecto, ofreciendo flexibilidad para distintas situaciones.

El Futuro del Desarrollo Web

Aprendizaje Continuo es la Clave 🚀

La tecnología evoluciona constantemente. Mantente actualizado, sé curioso y nunca dejes de aprender.