

UNIVERSIDAD TECNOLÓGICA DE TEHUACÁN

S - 2. Arquitectura de información y navegación accesible

Nombre del alumno:

Danna Paola Tobón Mosqueda

Docente:

José Miguel Carrera Pacheco

ING EN DESARROLLO Y GESTIÓN DE SOFTWARE

TEHUACÁN, PUEBLA, ENERO-ABRIL 2026

Diferencia entre Página Web vs. Aplicación Web

Aunque ambas corren en un navegador, sus propósitos y arquitecturas son radicalmente distintos.

Característica	Página Web (Sitio Web)	Aplicación Web (Web App)
Objetivo Principal	Informar (Consumo de contenido).	Realizar tareas (Interacción y creación).
Interacción	Unidireccional (El usuario lee/ve).	Bidireccional (El usuario manipula datos).
Complejidad	Baja (HTML/CSS, poco JS).	Alta (Lógica de negocio compleja, bases de datos).
Estado	Generalmente "Stateless" (no guarda sesión compleja).	Mantiene estado (Sesiones, carritos, perfiles).
Ejemplos	Blogs, Portafolios, Landing Pages, Sitios de noticias.	Gmail, Trello, Facebook, Banca en línea.

Ejemplos Reales de Aplicaciones Web Profesionales

Figma: Una herramienta de diseño vectorial compleja que corre en el navegador. Reemplazó software de escritorio instalable.

Airbnb: Gestiona reservas, pagos, mensajería en tiempo real, mapas y perfiles de usuario. Conecta oferta y demanda (Marketplace).

Slack/Discord (Versión Web): Comunicación en tiempo real usando WebSockets. No requiere refrescar la página para recibir mensajes.

¿Qué problemas se resuelven con software?

En la industria, el software es una herramienta para eliminar fricción. Generalmente se resuelven problemas en estas categorías:

Ineficiencia / Automatización: Tareas manuales repetitivas que consumen tiempo humano

Conectividad / Comunicación: Barreras geográficas o de tiempo

Acceso a Información / Transparencia: Datos que están ocultos o son difíciles de encontrar

Gestión de Transacciones: Seguridad y confianza en intercambios de dinero o bienes

Arquitectura General de Aplicaciones Web

Para construir una base sólida y escalable, se deben entender las tres capas principales:

A. Frontend (El Cliente)

Es lo que el usuario ve y toca. En una aplicación moderna, el Frontend no solo "muestra cosas", sino que gestiona lógica de presentación.

Tecnologías: React, Vue, Angular (Frameworks), HTML5, CSS3.

Responsabilidad: UX/UI, accesibilidad, llamadas al servidor (API requests)

B. Backend (El Servidor)

El cerebro de la operación. El usuario nunca lo ve, pero sin esto, la aplicación no funciona.

Lógica de Negocio: Reglas del sistema (ej. "No permitir comprar si no hay saldo").

Base de Datos: Donde persiste la información (SQL como PostgreSQL, o NoSQL como MongoDB).

API (Application Programming Interface): El puente que permite al Frontend pedirle datos al Backend.

C. Infraestructura / Entornos

Dónde vive el código. No es suficiente con que funcione en "mi computadora".

Entorno de Desarrollo: Tu computadora local.

Entorno de Staging (Pruebas): Un espejo del real para probar antes de lanzar.

Entorno de Producción: Lo que usan los clientes reales.

CI/CD (Integración y Despliegue Continuo): Automatización para pasar el código de tu PC a producción sin romper nada.

Análisis de Plataformas Similares (Benchmarking)

Ejemplo A: Zendesk / Jira Service Management

Son los líderes mundiales en gestión de tickets para empresas.

Lo bueno: Flujos de trabajo potentes (Ej: "Si el reporte es de 'Infraestructura', manda correo automático a Mantenimiento").

Lo malo: Son demasiado complejos, caros y se sienten "corporativos". No están diseñados para que un alumno reporte algo en 30 segundos desde su celular en el pasillo.

Ejemplo B: Aplicaciones de Reporte Ciudadano (Ej. "SeeClickFix")

Apps que usan los ayuntamientos para que ciudadanos reporten baches o luminarias rotas.

Lo bueno: Uso excelente de geolocalización y subida de fotos.

Lo malo: Generalmente la información es pública.

El riesgo para el proyecto: Un reporte de "acoso" o "mal trato de un profesor" NO puede tratarse igual que una "silla rota".

Lección: se necesitan Niveles de Privacidad.