S -1. Definición del problema y arranque profesional

Universidad Tecnológica de Tehuacán

Grado y Grupo:

7 B

Materia:

DWP

Alumno:

Kevin Rojas Martínez

Investigación individual

1. Diferencia entre Página Web y Aplicación Web

Aunque a menudo usamos ambos términos de manera indistinta, existe una línea técnica que los separa. La principal diferencia radica en la **interactividad**: mientras que una página web está diseñada para ser leída (consumo de información), una aplicación web está diseñada para ser usada (realización de tareas).

Comparativa: Página Web vs. Aplicación Web

Característica	Página / Sitio Web	Aplicación Web (Web App)
Objetivo principal	Informar y mostrar contenido.	Realizar funciones o tareas específicas.
Interacción	Pasiva (leer, ver imágenes, navegar).	Activa (crear, editar, borrar datos).
Contenido	Mayormente estático (igual para todos).	Dinámico (cambia según el usuario).
Autenticación	Rara vez requiere inicio de sesión.	Generalmente requiere usuario y contraseña.
Complejidad	Baja; fácil de desarrollar y mantener.	Alta; requiere bases de datos y lógica compleja.

1. Página Web (o Sitio Web)

Es como un folleto digital. Su función es presentar información de manera clara y estructurada. El usuario navega a través de enlaces para consumir el contenido, pero no puede modificar lo que ve en la pantalla ni procesar datos complejos.

- Ejemplos: Blogs personales, páginas de noticias (como un periódico digital), portafolios de artistas o sitios corporativos de "quiénes somos".
- Tecnologías: Principalmente HTML y CSS.

2. Aplicación Web

Es un software que se ejecuta en el navegador. Aquí, el usuario interactúa directamente con la lógica del sistema. Las aplicaciones web suelen estar conectadas a una base de datos y permiten que la interfaz cambie en tiempo real según lo que el usuario haga (por ejemplo, publicar un estado o arrastrar una tarea).

- Ejemplos: Gmail, Google Docs, Netflix, Facebook o plataformas de banca en línea.

- Tecnologías: Requiere lenguajes de programación más avanzados como JavaScript (React, Angular), Python, PHP o Node.js.

2. Ejemplos reales de aplicaciones web profesionales

- **Productividad y Colaboración (SaaS)**

Estas son herramientas donde el navegador se convierte en una oficina completa.

Google Docs / Office 365: No solo lees un documento; lo creas, lo editas simultáneamente con otros y el sistema guarda versiones automáticamente en la nube.

Notion: Una aplicación "todo en uno" que combina bases de datos, calendarios y wikis. Su complejidad técnica permite personalizar toda la interfaz según tus necesidades.

Figma: Es una aplicación web de diseño gráfico profesional. Permite que varios diseñadores muevan elementos en un lienzo infinito al mismo tiempo, algo que antes solo hacían programas instalados en la PC.

- **Gestión Empresarial y CRM**

Utilizadas por empresas para organizar sus ventas, clientes y finanzas.

Salesforce: El estándar de la industria para CRM (Customer Relationship Management). Permite rastrear cada interacción con un cliente, generar reportes de ventas y automatizar correos.

Trello / Monday.com: Aplicaciones de gestión de proyectos que usan tableros dinámicos. Puedes arrastrar tareas, asignar responsables y adjuntar archivos que se sincronizan al instante para todo el equipo.

HubSpot: Una plataforma compleja que gestiona marketing, ventas y servicio al cliente desde un solo panel de control web.

- **Entretenimiento y Consumo de Datos**

Aunque parecen simples "catálogos", su tecnología de procesamiento es masiva.

Netflix: Es una aplicación web avanzada que utiliza algoritmos de recomendación en tiempo real y ajusta la calidad del video automáticamente según tu conexión a internet.

Spotify Web Player: Te permite gestionar listas de reproducción, buscar entre millones de canciones y controlar la reproducción en otros dispositivos desde el navegador.

3. Qué tipo de problemas se resuelven con software

El software se utiliza fundamentalmente para resolver problemas de eficiencia y manejo de información. En primer lugar, soluciona la dificultad de procesar grandes volúmenes de datos que serían inmanejables para el cerebro humano. Mediante algoritmos, el software puede realizar cálculos matemáticos complejos, analizar tendencias en mercados financieros o predecir patrones climáticos en cuestión de milisegundos, garantizando una precisión que elimina el error manual.

En el ámbito de la organización, el software resuelve problemas de logística y gestión. Permite que las empresas controlen inventarios masivos, coordinen agendas de equipos globales y almacenen registros históricos que pueden ser consultados al instante. Sin estas herramientas, la administración de una multinacional o de un sistema de salud público colapsaría ante la acumulación de archivos físicos y la lentitud de los procesos burocráticos tradicionales.

Otro pilar importante es la solución a los problemas de distancia y comunicación. El software de red permite la colaboración en tiempo real, rompiendo las barreras geográficas. Esto facilita desde la educación remota hasta el control de maquinaria industrial desde otro continente. Básicamente, transforma el espacio físico en un entorno digital donde el intercambio de ideas y recursos es instantáneo y constante.

Finalmente, el software ataca el problema de la repetitividad. Automatiza tareas monótonas como el envío de correos masivos, la validación de transacciones bancarias o el respaldo de archivos liberando a las personas para que se enfoquen en actividades creativas o estratégicas. En resumen, donde existe un proceso que puede ser definido por reglas lógicas, el software interviene para hacerlo más rápido, seguro y escalable.

4. Arquitectura general de aplicaciones web (frontend, backend, entornos)

La arquitectura de una aplicación web es la estructura que define cómo interactúan los diferentes componentes para entregar una experiencia funcional al usuario. Se divide principalmente en dos grandes capas que se comunican entre sí, además de los entornos donde el código cobra vida.

El Frontend: La capa de presentación

El frontend es todo lo que el usuario ve y con lo que interactúa directamente en su navegador. Su función principal es convertir el diseño y los datos en una interfaz visualmente atractiva y funcional. Se construye utilizando HTML para la estructura, CSS para el estilo y el diseño visual, y JavaScript para la interactividad. En aplicaciones profesionales, se suelen usar librerías o frameworks como React, Vue o Angular para gestionar interfaces complejas de manera eficiente.

Backend: La capa de acceso a datos

El backend es el motor que corre en el servidor y que el usuario nunca ve. Su responsabilidad es procesar las reglas de negocio, gestionar la seguridad, autenticar usuarios y comunicarse con la base de datos. Cuando realizas una acción en el frontend (como iniciar sesión), este envía una solicitud al backend; el servidor procesa la información y devuelve una respuesta. Los lenguajes más comunes aquí son Python, Node.js, Java o PHP, y se apoya en bases de datos como PostgreSQL o MongoDB para almacenar la información de forma permanente.

La comunicación y el API

Para que estas dos capas trabajen juntas, necesitan un lenguaje común. Generalmente, esto se logra a través de una API (interfaz de programación de aplicaciones), que actúa como un puente. El frontend solicita información mediante protocolos como HTTP, y el backend responde enviando datos, usualmente en un formato ligero llamado JSON. Esta separación permite que, por ejemplo, puedas cambiar el diseño del frontend sin tener que reconstruir toda la lógica del servidor.

Entornos de desarrollo y despliegue

El software no se lanza directamente al público en cuanto se escribe una línea de código. Se maneja a través de diferentes "entornos" para garantizar la estabilidad. El entorno de Desarrollo (Local) es donde los programadores escriben y prueban el código en sus propias computadoras. Luego pasa a Staging (o QA), un servidor que imita exactamente las condiciones reales para realizar pruebas de errores. Finalmente, el código llega a Producción, que es el entorno real donde los usuarios finales acceden a la aplicación.

5. **Analizar 2 plataformas reales similares a tu idea**

1. Z-City (Anteriormente CityCop / Reporte Ciudadano)

Aunque existen varias versiones locales, estas plataformas se basan en la participación ciudadana para reportar incidentes en un mapa en tiempo real.

- Similitud con tu idea: Utiliza un sistema de geolocalización donde los usuarios marcan un punto en el mapa y seleccionan el tipo de problema (en tu caso sería "Fuga de agua" o "Corte de suministro").

- Fortaleza: La validación comunitaria. Si varios usuarios en la misma zona reportan que "llegó el agua", el sistema confirma la alerta automáticamente para todos los vecinos.

- Qué puedes aprender: Estas apps suelen fallar si la interfaz es lenta. Para tu aplicación web, la clave sería un botón de "Reportar estado" muy simple que no requiera más de dos clics.

2. Waze (Modelo de Alertas de Tráfico)

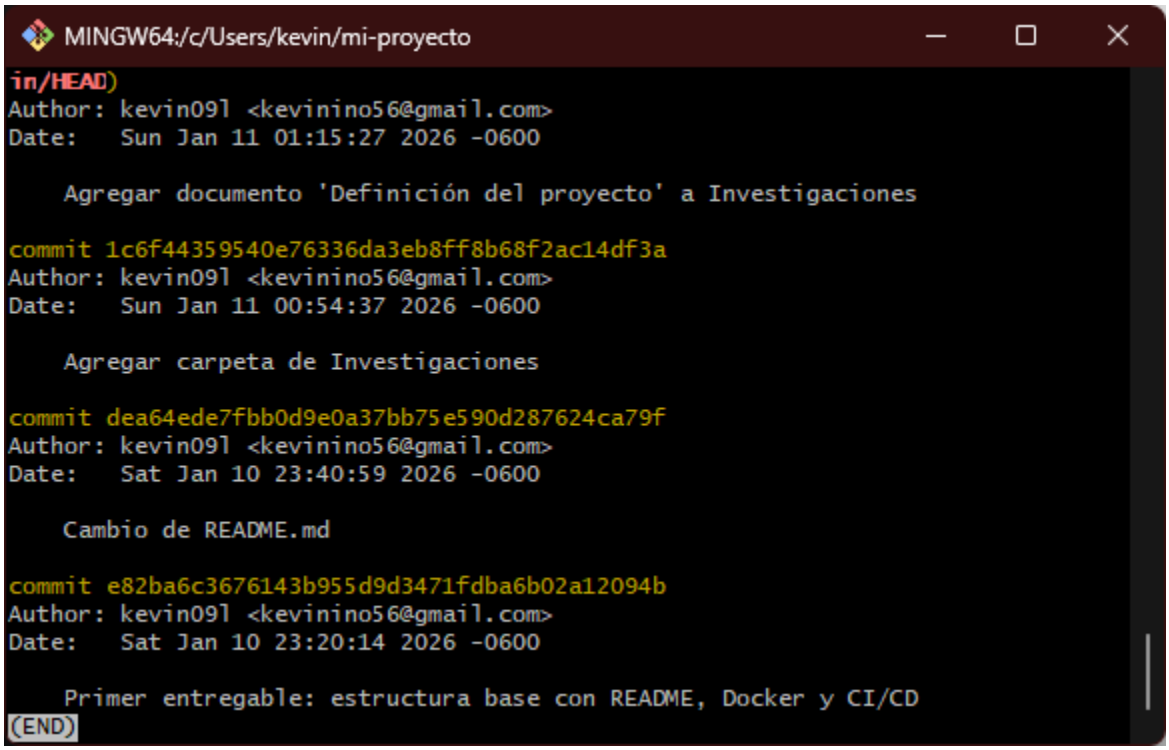
Aunque es para transporte, Waze es el mejor ejemplo de software que resuelve problemas de "escasez" (en este caso, de espacio vial) mediante reportes en vivo.

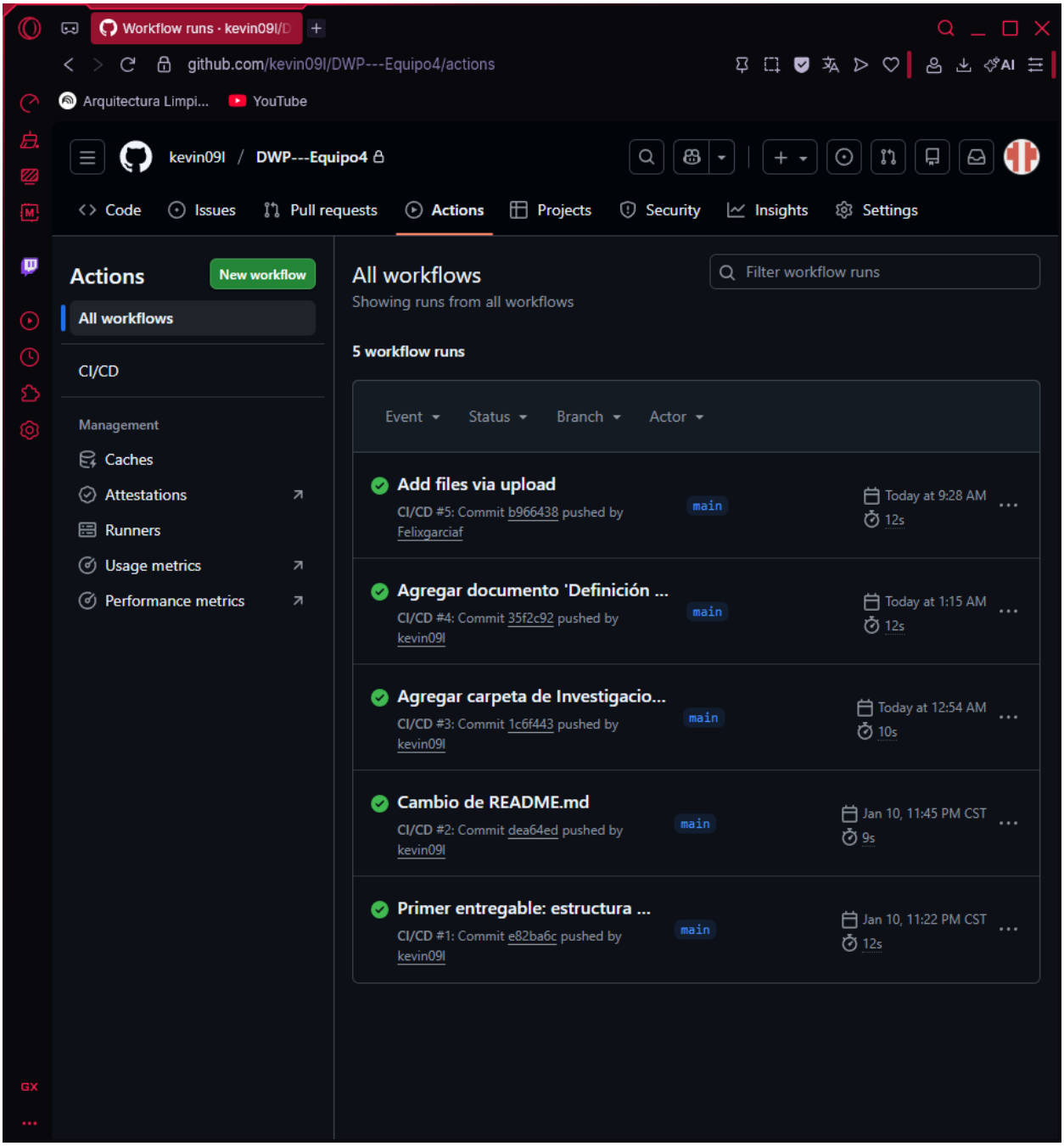
- Similitud con tu idea: El sistema de alertas predictivas. Si Waze detecta que un coche se detiene, asume que hay tráfico. Tu app podría detectar que si 10 personas de una colonia abren la app a las 7:00 AM para reportar "Sin agua", se debe disparar una alerta masiva de "Corte en progreso".

- Fortaleza: El backend procesa miles de datos para dar una respuesta simple al usuario: "Toma esta ruta". En tu app, el backend procesaría los reportes de fugas para generar un "Mapa de Presión" o "Estado del Suministro" por colonias.

- Qué puedes aprender: La importancia de las notificaciones push. Una aplicación web profesional de este tipo necesita usar Service Workers para que el usuario reciba el aviso de "¡Ya hay agua!" incluso si no tiene abierta la página en ese momento.

6. commits propios




Workflow runs · kevin09l/D
github.com/kevin09l/DWP---Equipo4/actions
Arquitectura Limpi...
YouTube
kevin09l / DWP---Equipo4
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
Actions
New workflow
All workflows
CI/CD
Management
Caches
Attestations
Runners
Usage metrics
Performance metrics
All workflows
Showing runs from all workflows
Filter workflow runs
5 workflow runs
Event
Status
Branch
Actor
Add files via upload
CI/CD #5: Commit b966438 pushed by Felixgarciaf
main
Today at 9:28 AM
12s
Agregar documento 'Definición ...
CI/CD #4: Commit 35f2c92 pushed by kevin09l
main
Today at 1:15 AM
12s
Agregar carpeta de Investigacio...
CI/CD #3: Commit 1c6f443 pushed by kevin09l
main
Today at 12:54 AM
10s
Cambio de README.md
CI/CD #2: Commit dea64ed pushed by kevin09l
main
Jan 10, 11:45 PM CST
9s
Primer entregable: estructura ...
CI/CD #1: Commit e82ba6c pushed by kevin09l
main
Jan 10, 11:22 PM CST
12s