Secretaría de Educación

UTyP DIRECCIÓN GENERAL DE UNIVERSIDADES TECNOLÓGICAS y POLITÉCNICAS

UTTEHUACÁN Universidad Tecnológica de Tehuacán

UNIVERSIDAD TECNOLÓGICA DE TEHUACÁN

PROGRAMA EDUCATIVO

TECNOLOGÍAS DE LA INFORMACIÓN INGENIERÍA EN DESARROLLO Y GESTIÓN DE SOFTWARE

ALUMNO

ARTURO CASTAÑEDA SERRANO

ASIGNATURA

DESARROLLO WEB PROFESIONAL

ACTIVIDAD

INVESTIGACIÓN INDIVIDUAL

DOCENTE

JOSÉ MIGUEL CARRERA PACHECO

Investigación de conceptos

Página web

Una página web es una “ventana de Internet” que funciona como un documento único con información y contenido alojado en un servidor, el cual se visualiza a través del navegador cuando se visita un sitio web, su propósito es comunicar datos, ofrecer servicios o mostrar productos en línea, convirtiéndose en la unidad básica de la experiencia digital, y suele formar parte de un sitio web compuesto por varias páginas relacionadas bajo un mismo dominio que representan a una organización, empresa o persona.

Aplicación web

Una aplicación web es un tipo de software que se ejecuta directamente en el navegador, sin necesidad de instalación en el dispositivo del usuario. Su propósito es permitir el acceso a funcionalidades complejas de manera sencilla y segura, facilitando la comunicación entre empresas y clientes en cualquier momento. Ejemplos comunes incluyen los carros de compra en sitios de comercio electrónico, la mensajería instantánea, los sistemas de banca en línea y los servicios de correo electrónico.

Estas aplicaciones destacan por su accesibilidad, ya que pueden utilizarse desde distintos navegadores y dispositivos. Además, su desarrollo resulta más eficiente y económico, pues una misma versión funciona en múltiples plataformas sin necesidad de crear iteraciones específicas. Para los usuarios, la experiencia es simple: no requieren descargas ni mantenimiento local, y las actualizaciones de seguridad se aplican automáticamente, reduciendo riesgos y garantizando que siempre estén al día.

En cuanto a su funcionamiento, las aplicaciones web se basan en una arquitectura cliente-servidor. El navegador ejecuta los scripts del lado del cliente para mostrar la interfaz y permitir la interacción, mientras que el servidor procesa las solicitudes y gestiona los datos. Esta dinámica posibilita acciones como leer contenido, enviar formularios o realizar transacciones en línea.

Diferencia entre página web y aplicación web

Aunque tanto las páginas web como las aplicaciones web se acceden desde un navegador, su propósito y funcionamiento son distintos. Un sitio web está diseñado principalmente para mostrar información de manera estática o con una interacción mínima, los usuarios suelen leer, navegar o consumir contenidos como textos, imágenes o videos. Ejemplos comunes son blogs, portafolios o páginas corporativas, donde la interacción se limita a formularios simples o botones de navegación.

Por otro lado, una aplicación web es un software en línea que permite al usuario realizar tareas específicas de forma dinámica e interactiva. A diferencia de un sitio web, aquí el usuario no solo consume información, sino que también introduce, modifica y gestiona datos en tiempo real. Plataformas como la banca en línea, redes sociales o herramientas de

gestión de proyectos son claros ejemplos, ya que ofrecen funcionalidades avanzadas que requieren procesamiento en el servidor y conexión con bases de datos.

La diferencia también se refleja en la tecnología y el desarrollo. Mientras que un sitio web puede construirse con HTML, CSS y algún JavaScript básico, una aplicación web necesita marcos más complejos como React o Angular, además de lenguajes de back-end y sistemas de autenticación. Asimismo, el mantenimiento de un sitio web suele centrarse en actualizar contenidos y plugins, mientras que el de una aplicación web implica supervisar seguridad, rendimiento y compatibilidad con APIs y bases de datos.

Ejemplos de aplicaciones web

Spotify

Spotify soluciona varios problemas fundamentales en la manera en que las personas acceden, disfrutan y descubren música. En primer lugar, elimina la necesidad de comprar discos físicos o archivos individuales, ofreciendo un catálogo inmenso de más de 100 millones de canciones disponibles al instante desde cualquier dispositivo. También aborda el reto de la personalización y descubrimiento, gracias a algoritmos de recomendación y listas de reproducción inteligentes, ayuda a los usuarios a encontrar nueva música y podcasts que se ajustan a sus gustos. Spotify soluciona el desafío de la distribución masiva con baja latencia. Su arquitectura está diseñada para manejar millones de reproducciones simultáneas, garantizando que la música se escuche sin interrupciones y con tiempos de carga mínimos.

Arquitectura general

El frontend de Spotify está diseñado para ofrecer una experiencia fluida, rápida y atractiva en múltiples dispositivos, integrando tecnologías modernas que permiten la interacción directa con los servicios del backend y garantizan consistencia en todas las plataformas.

Spotify cuenta con aplicaciones para móvil, escritorio y web, todas construidas con interfaces gráficas intuitivas que facilitan la navegación, la búsqueda de canciones, la gestión de playlists y la interacción social. En el caso de la aplicación web, se emplean tecnologías como React.js y frameworks de JavaScript que permiten renderizado dinámico y componentes reutilizables, lo que asegura un rendimiento óptimo y una experiencia uniforme. Para las aplicaciones móviles, se utilizan frameworks nativos como Swift en iOS y Kotlin/Java en Android, además de librerías multiplataforma que permiten mantener coherencia en diseño y funcionalidad.

El backend de Spotify está diseñado como un sistema distribuido de microservicios que permite manejar millones de usuarios simultáneamente y ofrecer música en tiempo real con baja latencia. Cada componente cumple una función específica y se integra mediante APIs y protocolos seguros, lo que asegura escalabilidad y resiliencia.

Este se apoya en balanceadores de carga que distribuyen las solicitudes de los clientes hacia distintos servidores, evitando cuellos de botella y garantizando disponibilidad. Los

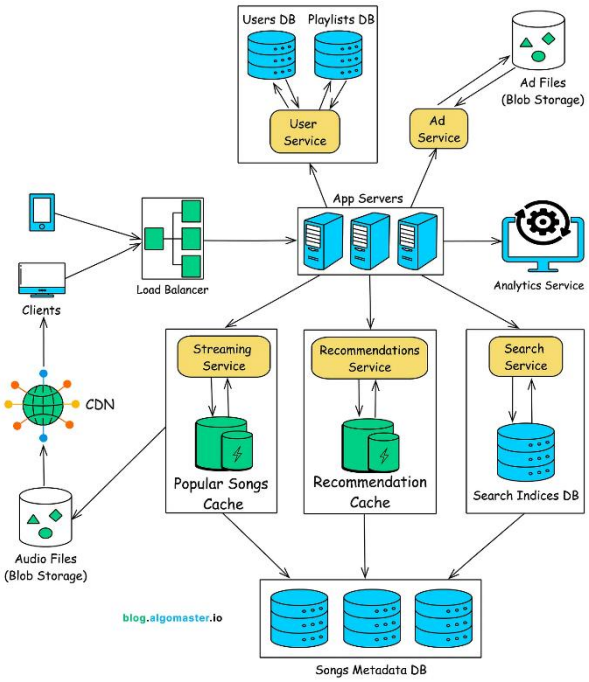
servidores de aplicación reciben las peticiones y las redirigen al microservicio correspondiente. Estos microservicios están escritos principalmente en Java, Python y C++, y se comunican entre sí mediante gRPC y REST APIs, lo que facilita la interoperabilidad y el mantenimiento.

Entre los servicios más relevantes se encuentran:

- Streaming Service, que gestiona la entrega de archivos de audio desde sistemas de almacenamiento distribuido y redes CDN, optimizando la calidad según el ancho de banda del usuario.
- Search Service, que utiliza motores como Elasticsearch para indexar millones de canciones y permitir búsquedas rápidas y precisas.
- Recommendation Service, que aplica algoritmos de filtrado colaborativo, machine learning y análisis de metadatos para generar recomendaciones personalizadas.
- User Service, encargado de manejar perfiles, playlists y preferencias, apoyado en bases de datos relacionales y NoSQL.
- Ad Service, que administra la inserción dinámica de anuncios en la experiencia de usuarios gratuitos.

En cuanto al almacenamiento, Spotify combina diferentes tecnologías: bases de datos relacionales como PostgreSQL para datos estructurados, sistemas NoSQL como Cassandra para información distribuida y dinámica, almacenamiento en la nube tipo Amazon S3 para archivos de audio y anuncios, y Redis como caché para acelerar el acceso a contenido popular.

La infraestructura se ejecuta sobre entornos en la nube con despliegues automatizados mediante Kubernetes y Docker, lo que permite escalar servicios de forma dinámica. Además, se emplean técnicas de sharding y replicación para garantizar consistencia y disponibilidad global. La seguridad se asegura con OAuth 2.0 para autenticación, cifrado TLS en tránsito y cifrado en reposo para datos sensibles.



Zoom

Zoom soluciona uno de los grandes retos de la comunicación moderna: conectar a personas de manera inmediata y confiable sin importar la distancia. Su diseño está pensado para ofrecer llamadas uno a uno, reuniones grupales y conferencias masivas con audio, video y opción de compartir pantalla, lo que elimina las barreras físicas y permite la colaboración en tiempo real. Además, brinda la posibilidad de grabar las sesiones para consultarlas después, lo que resuelve la necesidad de conservar información y facilita el aprendizaje o la revisión de acuerdos.

Arquitectura general

En el frontend, el cliente de Zoom es la puerta de entrada para los usuarios. Está disponible en aplicaciones móviles, de escritorio y navegadores, ofreciendo una interfaz sencilla para iniciar o unirse a reuniones. Este cliente maneja la captura de video, audio y pantalla, y se conecta con los servicios centrales mediante protocolos como WebRTC y WebSockets. Además, se apoya en mecanismos como HTTP tunneling para atravesar firewalls y proxies, asegurando que los usuarios puedan conectarse incluso en redes restringidas.

En primer lugar, Zoom emplea WebSocket Handlers y Managers para mantener conexiones persistentes entre clientes y servidores, lo que permite la comunicación bidireccional en tiempo real. Estos se despliegan detrás de balanceadores de carga y utilizan algoritmos de enrutamiento eficientes para distribuir tráfico y garantizar tolerancia a fallos.

El servicio de señalización coordina la comunicación entre usuarios, gestionando el inicio, la terminación y el estado de las llamadas. Se integra con el User Service, que almacena datos de usuarios en bases distribuidas y aplica mecanismos de caché para reducir la carga y mejorar el rendimiento.

Para la conexión directa entre clientes, Zoom utiliza servidores STUN que ayudan a descubrir direcciones IP públicas y establecer conexiones peer-to-peer. Cuando estas conexiones fallan, entra en acción el TURN Server, que actúa como intermediario para retransmitir tráfico, desplegado en ubicaciones geográficas distribuidas para minimizar la latencia.

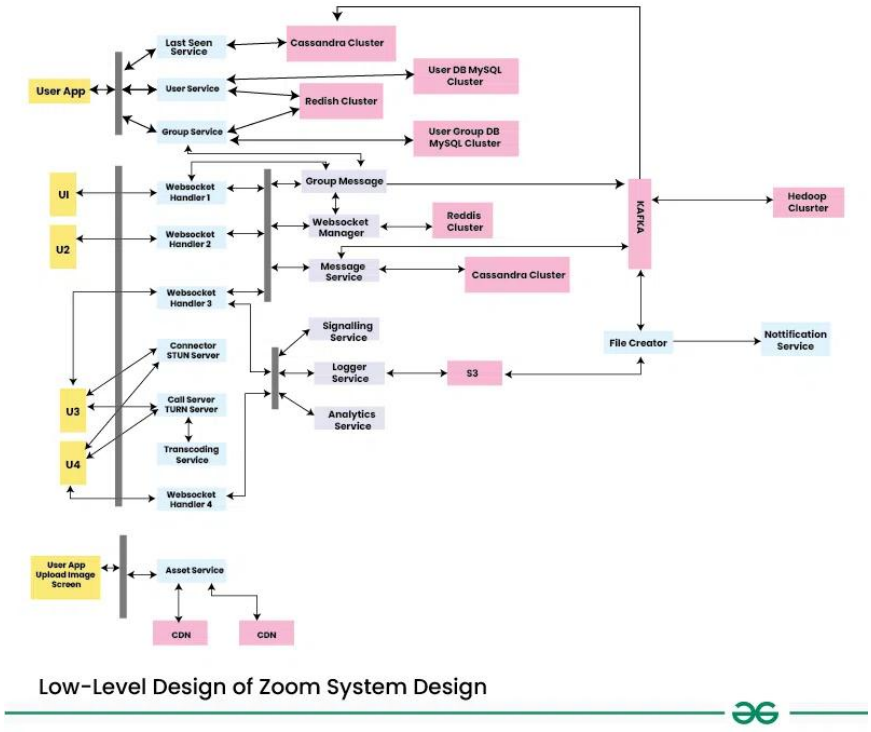
El transporte de video y audio se realiza principalmente sobre UDP, ya que este protocolo sacrifica fiabilidad en favor de velocidad, lo cual es preferible en videoconferencias donde es mejor perder algunos fotogramas que ralentizar la transmisión. En cambio, para datos críticos como autenticación o mensajería, se utiliza TCP.

Los Call Servers y transcodificadores son responsables de manejar grandes grupos de usuarios. Estos agregan y convierten flujos de video/audio a diferentes resoluciones y formatos, adaptándose al ancho de banda y capacidades de cada dispositivo. Se apoyan en CDNs para distribuir contenido de manera eficiente y cercana a los usuarios.

En cuanto al almacenamiento y procesamiento, Zoom utiliza infraestructura en la nube como AWS para hospedar metadatos de reuniones, aplicaciones web y tráfico en tiempo real, mientras que Oracle Cloud se emplea especialmente para usuarios educativos. Las

bases de datos están organizadas en tablas distribuidas para usuarios, reuniones y grabaciones.

Además, el backend incluye Kafka para registrar eventos como cambios de ancho de banda durante las llamadas, lo que permite análisis en tiempo real y ajustes dinámicos de calidad. También cuenta con microservicios especializados: gestión de usuarios, programación de reuniones, streaming de video, chat, grabación y notificaciones, todos diseñados para escalar horizontalmente y comunicarse mediante APIs internas.



Low-Level Design of Zoom System Design

Facebook

Facebook, como sistema de diseño a gran escala, está pensado para resolver múltiples necesidades de los usuarios y desafíos técnicos que surgen al conectar a miles de millones de personas en todo el mundo. En esencia, lo que soluciona es la posibilidad de comunicación, interacción social y acceso a información en tiempo real, pero detrás de ello hay una arquitectura compleja que atiende problemas de seguridad, rendimiento y escalabilidad.

Las interacciones sociales son facilitadas para la comunicación mediante publicaciones, comentarios, reacciones y mensajes en tiempo real, resolviendo la necesidad de interacción inmediata.

Permite que cada persona personalice su espacio digital con información personal, intereses y preferencias, resolviendo la necesidad de identidad en línea. Mediante notificaciones se puede mantener a los usuarios informados de manera constante sobre lo que ocurre en la red social.

Arquitectura general

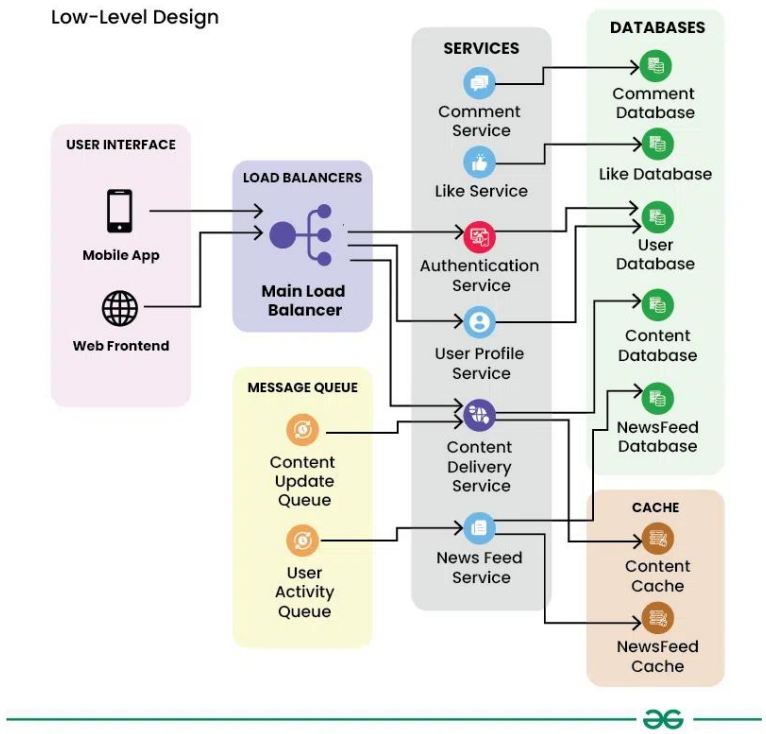
El frontend de Facebook está construido para ofrecer una experiencia rápida, dinámica y consistente en múltiples dispositivos, apoyándose en tecnologías modernas que permiten escalar la interfaz a millones de usuarios. La base está formada por HTML, CSS y JavaScript, que definen la estructura, el estilo y la interacción de la plataforma. Sobre esta base, Facebook utiliza React.js, una librería creada por la propia compañía, que facilita el desarrollo de componentes reutilizables y altamente interactivos, optimizando la renderización y reduciendo los tiempos de carga. Además, también es importante señalar que Facebook emplea PHP en su frontend, especialmente en la generación dinámica de páginas y en la integración con sus sistemas internos.

Para sus aplicaciones móviles, Facebook emplea React Native, que permite desarrollar aplicaciones nativas para iOS y Android utilizando JavaScript y componentes similares a los del entorno web.

El backend de Facebook está diseñado para soportar un volumen masivo de usuarios y datos, y se compone de diversas tecnologías que trabajan de manera integrada para garantizar rendimiento, escalabilidad y disponibilidad. En primer lugar, la plataforma utiliza bases de datos relacionales como MySQL, que han sido altamente modificadas para manejar la enorme cantidad de información estructurada, como perfiles y relaciones entre usuarios. Para datos no estructurados y distribuidos, emplea sistemas NoSQL como Cassandra y HBase, que permiten almacenar publicaciones, comentarios y mensajes de manera eficiente en múltiples servidores.

La capa de caché es crítica para la velocidad de respuesta, aquí entran en juego Memcached y Redis, que almacenan temporalmente datos de uso frecuente, reduciendo la carga sobre las bases de datos principales y acelerando la entrega de contenido. En cuanto a la comunicación interna entre servicios, Facebook utiliza Apache Kafka como sistema de mensajería distribuida, lo que permite manejar flujos de datos en tiempo real, como notificaciones y actualizaciones de estado. El backend está soportado por una infraestructura de servidores distribuidos y sistemas de replicación, que aseguran redundancia y tolerancia a fallos. Todo esto se complementa con sistemas de monitoreo y seguridad, que permiten detectar problemas en tiempo real y proteger la información de los usuarios.

El balanceo de carga se realiza con herramientas como Nginx, que distribuyen el tráfico entre miles de servidores para evitar cuellos de botella y garantizar que los usuarios siempre tengan acceso rápido. Además, para la entrega de contenido multimedia (imágenes, videos), se apoya en una CDN propia, que acerca los archivos a los usuarios desde servidores geográficamente cercanos.



Plataformas similares a mi idea

La idea que se me ocurrió fue el de hacer una aplicación web en la que las personas pudieran dar visibilidad a los problemas que se encuentran en su entorno, por ejemplo, baches en las calles, fugas de agua, drenajes tapados, falta de iluminación pública, etc. Con el fin de dar visibilidad a aquellos problemas con los que constantemente uno está conviviendo, además de generar “presión” para que esos fueran resueltos lo más antes posibles.

Fixa min gata

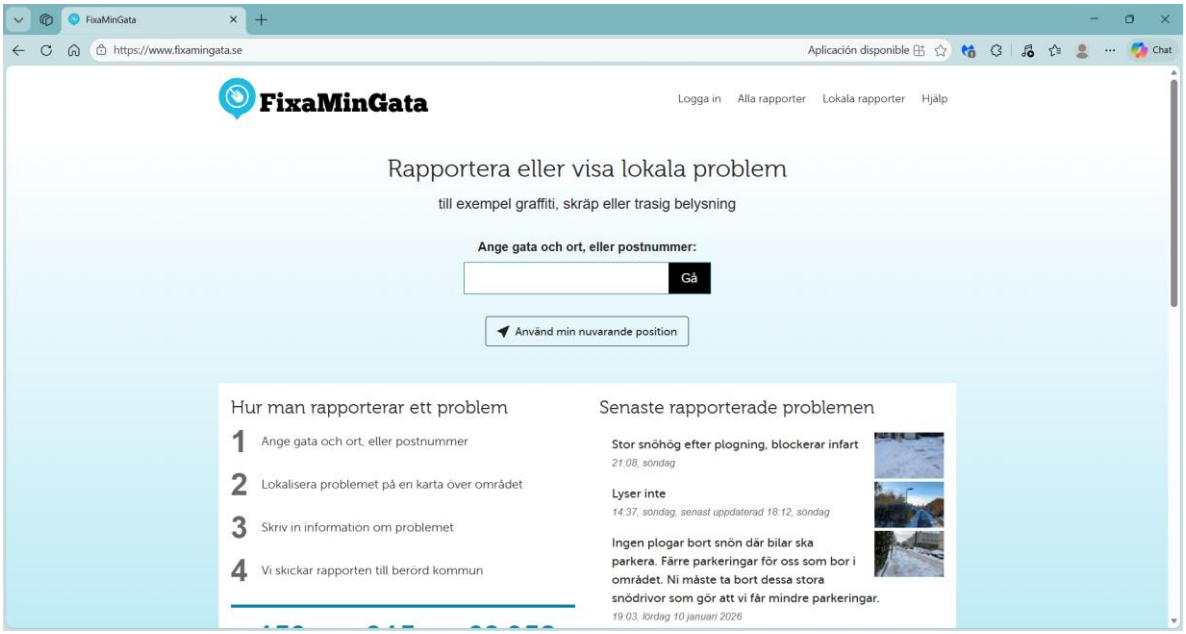
URL: [FixaMinGata]

Fixa min gata (en sueco, "Arregla mi calle") es una plataforma digital de participación ciudadana diseñada para facilitar la comunicación entre los residentes de Suecia y sus administraciones locales. Su propósito principal es servir como un canal centralizado donde los ciudadanos pueden informar sobre problemas en la infraestructura pública, tales como baches en el pavimento, farolas fundidas, grafitis, acumulación de basura o señales de tráfico dañadas.

El funcionamiento de la plataforma es intuitivo y se basa en la geolocalización. El usuario comienza introduciendo un código postal o una dirección, lo que despliega un mapa interactivo de la zona. Sobre este mapa, el informante marca el punto exacto de la

incidencia, añade una descripción detallada del problema y tiene la opción de adjuntar fotografías para aportar mayor claridad. Una vez enviada la denuncia, el sistema identifica automáticamente a qué municipio o autoridad competente le corresponde la gestión y le remite el informe por correo electrónico o a través de integraciones directas en sus sistemas de gestión interna.

Entre sus herramientas técnicas más destacadas se encuentra el uso de OpenStreetMap para la cartografía, lo que garantiza una base de datos geográfica abierta y colaborativa. Además, la plataforma utiliza el estándar Open311, una API abierta que permite la interoperabilidad entre los informes ciudadanos y los sistemas de despacho y seguimiento de incidencias que utilizan los ayuntamientos suecos.



072 - Atención Ciudadana de Nuevo Laredo

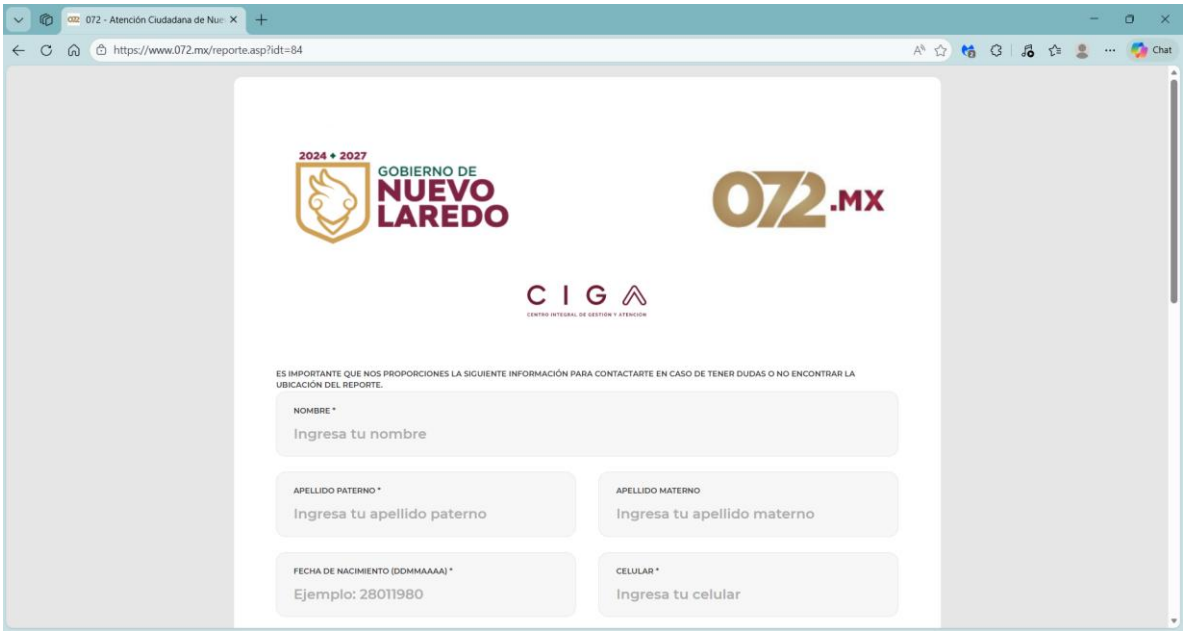
URL: [072 - Atención Ciudadana de Nuevo Laredo]

El portal 072.mx es una plataforma tecnológica de atención ciudadana utilizada por diversos gobiernos municipales en México (principalmente en el estado de Tamaulipas, como Nuevo Laredo, Matamoros y Ciudad Victoria) para gestionar reportes sobre servicios públicos e infraestructura urbana. Su utilidad fundamental es servir como un puente directo y digital entre el ciudadano y la administración local, permitiendo que cualquier persona denuncie desperfectos en la vía pública sin necesidad de acudir físicamente a las oficinas gubernamentales. El enlace que contiene el id (idt=84) corresponde al área de Bacheo, una de las demandas más comunes en la gestión urbana.

En cuanto a su funcionamiento, el sitio opera bajo un modelo de ticket de servicio o folio. El proceso comienza cuando el usuario accede al formulario y proporciona información de contacto básica, necesaria para recibir actualizaciones sobre el estatus de su petición.

Posteriormente, el sistema solicita una descripción detallada del problema y, lo más importante, la ubicación exacta del incidente, la cual puede complementarse con referencias visuales. Una vez enviado el formulario, el sistema genera automáticamente un número de folio único, el cual permite al ciudadano dar seguimiento a través del mismo portal o vía telefónica para verificar si la cuadrilla correspondiente ya atendió el reporte.

El uso de la extensión .asp (Active Server Pages) indica que el sitio corre sobre una arquitectura de servidores de Microsoft, diseñada para manejar bases de datos dinámicas en tiempo real. Para la gestión operativa, el portal está integrado con el Centro Integral de Gestión y Atención (CIGA), que actúa como el "cerebro" logístico que recibe la información digital y la canaliza a las dependencias operativas. Además, el sistema suele emplear APIs de cartografía para la validación de direcciones y cuenta con módulos de notificación automatizada vía correo electrónico para mantener informado al usuario.



Fuentes de información

¿Qué es una página web? | *Lenovo México*. (s. f.). https://www.lenovo.com/mx/es/glosario/pagina-web/

Amazon Web Services. (s. f.). *¿Qué es una aplicación web?* Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/web-application/

Boada, D. (2025, 18 diciembre). *¿Cuál es la diferencia entre una aplicación web y un sitio web?* Hostinger Tutoriales. https://www.hostinger.com/es/tutoriales/diferencias-entre-aplicacion-web-y-sitio-web

Singh, A. P. (2024, 2 octubre). Design Spotify - System Design interview. *AlgoMaster Newsletter*. https://blog.algomaster.io/p/design-spotify-system-design-interview

GeeksforGeeks. (2025, 23 julio). *Design Spotify Premium | System Design*. GeeksforGeeks. https://www.geeksforgeeks.org/system-design/design-spotify-premium-system-design/

GeeksforGeeks. (2025b, julio 23). *Designing Zoom | System Design*. GeeksforGeeks. https://www.geeksforgeeks.org/system-design/designing-zoom-system-design/

GeeksforGeeks. (2025a, julio 23). *Design Facebook | System Design*. GeeksforGeeks. https://www.geeksforgeeks.org/system-design/design-facebook-system-design/