INSTITUTO POLITÉCNICO NACIONAL

''La técnica al servicio de la patria''

UNIDAD PROFESIONAL INTERDISCIPLINARIA
DE INGENIERIA Y CIENCIAS SOCIALES
ADMINISTRATIVAS

ASIGNATURA:

PROGRAMACIÓN WEB

PROFESOR:

Francisco Javier Bueno Vásquez

ALUMNO:

Saldaña Enriquez Carlos Abraham

SECUENCIA: 5NV50

Investigación 1er Parcial

Introducción a las Aplicaciones Web

Fecha de entrega: 22 de marzo 2024

Índice:

Introducción:

Las aplicaciones web han transformado la manera en que nos relacionamos con el entorno digital, brindando desde páginas básicas de información hasta sistemas avanzados de compras en línea. Las aplicaciones se dividen en cuatro grupos: las estáticas presentan información fija, las dinámicas crean contenido interactivo y personalizado, el comercio electrónico facilita la compra y venta en línea, y los gestores de contenidos permiten gestionar y publicar contenido de manera sencilla.

Para entender cómo operan estas aplicaciones, es importante conocer la arquitectura cliente/servidor (C/S), un modelo esencial en el que las labores se dividen entre los servidores que proveen recursos o servicios y los clientes que solicitan dichos servicios. Este modelo sirve como la fundación para el desarrollo y funcionamiento de las aplicaciones web.

Las aplicaciones tienen la flexibilidad de distribuir sus funciones principales - presentación, lógica de negocio y gestión de datos - de diversas formas, dando lugar a tres modelos predominantes: Presentación Distribuida, en el que la interfaz de usuario está en el cliente mientras que los datos y la lógica de negocio se manejan en el servidor; Aplicación Distribuida, que implica una mayor interacción entre cliente y servidor, con más funciones distribuidas en el lado del cliente; y Datos Distribuidos, donde la gestión de datos se dispersa en múltiples servidores o localizaciones.
Finalmente, los patrones de arquitectura web ofrecen soluciones estandarizadas a problemas comunes en el diseño y desarrollo de aplicaciones web. Estos patrones no solo promueven la reutilización de código, sino que también ayudan a estructurar aplicaciones de manera eficiente, facilitando su mantenimiento y escalabilidad.

Este panorama general subraya la importancia de comprender los diferentes tipos de aplicaciones web, la arquitectura cliente/servidor y los modelos de distribución de funciones, así como los patrones de arquitectura web, para el desarrollo efectivo de soluciones digitales en la era actual.

1. Aplicaciones Web

Una aplicación web es un tipo de software que se ejecuta en un navegador web y que, en lugar de estar instalada en un dispositivo, se accede a través de una red, como Internet.

Las aplicaciones web pueden ser muy variadas en función de su finalidad, y pueden incluir desde aplicaciones simples, como un formulario de contacto en un sitio web, hasta aplicaciones más complejas, como una red social; dependiendo las necesidades de tu empresa.

Una de las ventajas de las aplicaciones web es que se pueden acceder desde cualquier dispositivo con conexión a Internet y navegador web, lo que las hace muy convenientes para los usuarios.

a. Aplicación Web Estática

Una aplicación web estática es un sitio web compuesto por páginas HTML fijas que se entregan exactamente como están almacenadas, sin ningún tipo de procesamiento o generación dinámica de contenido en el servidor en el momento de la solicitud. Cada usuario recibe el mismo contenido. Estas aplicaciones se contrastan con las aplicaciones web dinámicas, donde el contenido de las páginas puede cambiar y adaptarse en tiempo real según las interacciones del usuario, las solicitudes del servidor o entradas de datos.

Esto quiere decir este tipo de web muestra poca información y no suele cambiar mucho. Son sitios donde presentan información a los visitantes sin permitir ninguna interactuación con el contenido, en ocasiones incluyen imágenes o vídeos, sin embargo, tiene que ser lo más simple posibles para diferenciarse del resto. Las aplicaciones web estáticas son los sitios diseñados para presentar información a los visitantes sin permitir que interactúen con el contenido más allá de su lectura. Se caracteriza por lo siguiente:

- Se suelen desarrollar en HTML y CSS y puede utilizarse algo de JavaScript. Pueden presentar contenido digital con movimiento, como vídeos, audio, banners, o GIF animados.
- Cambiar los contenidos también es complicado, se necesita modificar el HTML (recodificar la página) y actualizarlo en el servidor.
- Se suelen emplear para ofrecer información concisa y permanente.

Sin embargo, su limitación principal es que no pueden proporcionar una experiencia de usuario personalizada o interacciones dinámicas, lo que puede limitar su utilidad en algunos casos de uso.

b. Aplicación Web Dinámica

Una aplicación web dinámica es un sitio web que genera contenido en tiempo real en respuesta a las acciones o solicitudes del usuario. A diferencia de las aplicaciones web estáticas, que sirven el mismo contenido a todos los usuarios, las aplicaciones dinámicas pueden personalizar el contenido para cada usuario y permitir interacciones más complejas. Esto se logra a través de la ejecución de código del lado del servidor (utilizando tecnologías como Node.js, PHP, Python con Django o Flask, etc.) o del lado del cliente (con JavaScript y frameworks como React, Vue.js, Angular, etc.), y a menudo requiere una base de datos para almacenar la información.

Esta es mucho más completa a nivel técnico. A diferencia de la estática, esta sí que permite al usuario una experiencia interactiva y personalizada, además pueden actualizar la información de la página en tiempo real según las acciones del usuario. Son sitios en la red que no se mantienen estáticos a lo largo del tiempo y que, por el contrario, están en un proceso constante de actualización que hace que haya diferentes contenidos cada vez que se visitan. Estas aplicaciones pueden ser realmente simples y únicamente modifican una parte del sitio tras cada actualización o cambiar completamente cuando alguien las visita. Las aplicaciones web dinámicas presentan los siguientes rasgos:

- Utilizan bases de datos para cargar la información y los contenidos se actualizan cada vez que el usuario accede a la aplicación.
- La actualización de los contenidos es sencilla, la mayoría se administra mediante un CMS. No se requiere acudir al servidor.
- Permiten implementar numerosas funcionalidades, como foros o bases de datos.
- Admite muchas posibilidades de diseño y presentación.
- Hay interacción en ellas. El usuario puede realizar cambios.

Una de las principales ventajas de las aplicaciones web dinámicas es su capacidad para proporcionar una experiencia de usuario personalizada y adaptada a las necesidades del usuario.

c. Comercio Electrónico

El comercio electrónico, también conocido como e-commerce, se refiere a la compra y venta de bienes o servicios utilizando internet, así como la transferencia de dinero y datos para ejecutar estas transacciones. Se ha convertido en una de las formas más importantes de comercio debido a su conveniencia y accesibilidad, permitiendo a los consumidores adquirir productos de cualquier parte del mundo.

Las aplicaciones web de comercio electrónico son una de las más populares en el mercado digital de hoy. Esto se debe a que cada vez son más las empresa y personas que prefieren el uso de plataformas digitales para comercializar sus mercancías. Pero el que algo sea popular no significa que sea simple. Su desarrollo es más complejo que el de las anteriores, ya que debe permitir la realización de transacciones online a través de los distintos métodos de pago (tarjetas de crédito y débito, PayPal, etc.). Un sitio web de comercio electrónico debe estar optimizado para muchas cosas: mostrar un producto, describirlo y exponer sus características principales; permitir que se añadan o quiten productos; administrar el pago de los clientes; generar facturas electrónicas y hasta contar con un buen diseño y herramientas de contacto como formularios y chat en vivo. Por otro lado, un buen sitio web de comercio electrónico debe ser al mismo tiempo una aplicación web dinámica. El comercio electrónico presenta los siguientes rasgos:

- Se debe crear un panel de gestión para el administrador
- Las empresas utilizan plataformas de comercio electrónico para promocionar sus productos o servicios a través de diversas estrategias de marketing digital, como el SEO, publicidad en redes sociales y email marketing.
- Incluye la gestión de devoluciones, atención al cliente y mantenimiento de productos, ofreciendo una plataforma para que los clientes soliciten y reciban soporte después de la compra.
- Las tiendas en línea están disponibles las 24 horas del día, lo que permite a los consumidores comprar en cualquier momento y lugar.
- Las plataformas de e-commerce pueden ofrecer experiencias de compra personalizadas basadas en el comportamiento y preferencias del usuario, mejorando la satisfacción del cliente.

Para que un sitio web de comercio electrónico sea bueno ha de ser al mismo tiempo una aplicación web, con el objetivo de mantener siempre actualizados los productos en cuanto a la disponibilidad de su mercancía, los costes y la información de consulta.

La ventaja que ofrece las tiendas virtuales es que permiten a los usuarios comprar productos desde cualquier lugar y en cualquier momento. Además, las tiendas virtuales pueden llegar a un público más amplio que las tiendas físicas, por lo que las ventas aumentan.

d. Aplicación Web con Gestor de Contenidos

Una aplicación web con gestor de contenidos (CMS, por sus siglas en inglés, Content Management System) es una plataforma de software que permite crear, editar, gestionar y publicar contenido digital de manera fácil y sin necesidad de conocimientos profundos en programación o diseño web. Los CMS son utilizados para administrar el contenido de sitios web y aplicaciones web, facilitando tareas como la actualización de textos, imágenes, videos, y cualquier otro tipo de contenido digital.

Una aplicación web con gestor de contenidos (CMS) es un tipo de aplicación web que permite a los usuarios crear y gestionar el contenido de un sitio web de forma fácil y eficiente, sin necesidad de tener conocimientos técnicos avanzados en programación, ya que proporcionan una interfaz de usuario intuitiva que permite a los usuarios crear, publicar, editar y eliminar contenido, así como gestionar la estructura y el diseño del sitio web. Los sistemas de gestión de contenidos o CMS (Content Management System) permiten a los usuarios administradores crear y gestionar el contenido de la aplicación web de forma sencilla. Es la opción más recomendable cuando el contenido de la aplicación deba ser actualizado continuamente. Como tal, estas aplicaciones funcionan como intermediarios entre tú y tu sitio web. Por ello están optimizadas con todas las herramientas necesarias para que tus contenidos estén actualizados y en la mejor condición para ser publicados en tu sitio. Tres de los CMS más habituales son:

- WordPress. Es el más conocido y extendido de todos. Destaca por ser más asequible para principiantes que otras opciones
- Joomla!. Gestor de contenidos de software libre con largo recorrido como WordPress. Proporciona gran flexibilidad, aunque es más recomendable para desarrolladores o diseñadores web con experiencia
- Drupal. Otro CMS de código abierto. Su arquitectura modular permite mucha personalización. Se emplea sobre todo para la construcción de plataformas de social publishing y comunidades.

Una de las principales ventajas de es que el usuario puede ahorrar tiempo y costes en actualizar y mantener su sitio web regularmente, dado que los CMS pueden ser personalizados y adaptados a las necesidades particulares de cada sitio web.

e. Cuadro Comparativo

Característica	Aplicación Web Estática	Aplicación Web Dinámica	Comercio Electrónico	Aplicación Web con Gestor de Contenidos
Definición	Sitios fijos sin interacción dinámica o actualización de contenido en tiempo real.	Sitios que permiten la interacción y actualización de contenido en tiempo real.	Sitios dinámicos diseñados específicamente para la compra y venta de productos o servicios en línea.	Sitios dinámicos que utilizan un sistema (CMS) para gestionar y publicar contenido de manera fácil y estructurada.
Interacción del usuario	Limitada o nula.	Alta, personalizable según las acciones del usuario.	Alta, incluye transacciones, cuentas de usuario, valoraciones, etc.	Alta, permite a los usuarios interactuar con el contenido (comentar, valorar, compartir).
Gestión de contenido	Estático, no cambia a menos que se edite el código fuente.	Dinámico, el contenido puede cambiar sin necesidad de modificar el código directamente.	Dinámico, con funcionalidades específicas para el manejo de inventarios, pedidos, pagos, etc.	Dinámico, facilitado a través de una interfaz de usuario del CMS para la gestión de contenido sin conocimientos técnicos avanzados.
Ejemplo	Sitio web corporativo simple sin actualizaciones frecuentes.	Blog con contenido actualizado regularmente.	Tienda en línea como Amazon.	Sitio web de noticias que utiliza WordPress como CMS.
Personalización y escalabilidad	Baja. Necesidad de editar el código para cambios.	Alta. Facilita la adaptación a las necesidades del usuario.	Alta. Requiere adaptaciones constantes para mejorar la experiencia del usuario y optimizar las ventas.	Alta. El CMS permite ampliar o modificar el sitio de manera sencilla.
Mantenimiento y actualización	Bajo. No requiere actualizaciones constantes a menos que se desee cambiar el contenido.	Moderado a alto. Dependiendo de la complejidad de las interacciones y la frecuencia de actualización de contenido.	Alto. Necesita actualizaciones constantes para inventario, ofertas, seguridad, etc.	Moderado. Los CMS facilitan las actualizaciones, aunque el contenido debe ser actualizado regularmente.

2. Arquitectura cliente/servidor (C/S)

Para este apartado vamos a utilizar como ejemplo el proceso de ver un video en YouTube, detallando cómo se lleva a cabo esta acción siguiendo la arquitectura Cliente/Servidor (C/S).

Paso 1: El Cliente Hace una Petición

- Usuario: Abre su navegador web y visita el sitio web de YouTube.
- Acción: Escribe "MrBeast" en la barra de búsqueda y presiona Enter.
- Tecnología involucrada: El navegador (cliente) envía una petición HTTP al servidor de YouTube solicitando los videos relacionados con MrBeast.

Paso 2: El Servidor Procesa la Petición

- Servidor: El servidor de YouTube recibe la petición.
- Acción: Busca en su base de datos los videos que coinciden con la consulta "MrBeast".
- Tecnología involucrada: El servidor utiliza algoritmos de búsqueda para encontrar y seleccionar los videos más relevantes basándose en la consulta. Luego, genera una página de resultados en HTML que incluye una lista de estos videos, con miniaturas y descripciones.

Paso 3: El Servidor Envía una Respuesta

- Servidor: Una vez que la página de resultados está lista, el servidor la envía de vuelta al navegador del cliente.
- Acción: El servidor responde con un código de estado HTTP (ejemplo: 200 OK) y el contenido HTML de la página de resultados.
- Tecnología involucrada: Además del HTML, el servidor envía CSS para el diseño de la página y JavaScript para funcionalidades interactivas, como la reproducción de videos o la carga dinámica de comentarios.

Paso 4: El Cliente Muestra la Respuesta

- Usuario: Recibe la página de resultados en su navegador.
- Acción: Elige un video de la lista de resultados y hace clic en él.
- Tecnología involucrada: El navegador interpreta el HTML, CSS, y JavaScript para mostrar la página de resultados. Al hacer clic en un video, el navegador solicita el contenido del video al servidor.

Paso 5: Reproducción del Video

- Servidor: Recibe la petición del video específico y comienza a transmitir el video al cliente.

- Acción: El servidor envía el video en un formato adecuado para ser reproducido en el navegador.
- Tecnología involucrada: Se utiliza streaming de video, lo que permite al usuario comenzar a ver el video casi de inmediato, sin necesidad de descargarlo completamente primero.

Resumen de la Interacción C/S en este Ejemplo:

1. **Petición de Búsqueda:** El cliente (navegador web) envía una petición al servidor de YouTube buscando videos sobre MrBeast.
2. **Respuesta con Resultados:** El servidor procesa la petición, buscando en su base de datos y enviando una página de resultados al cliente.
3. **Selección y Petición de Video:** El usuario selecciona un video, y el cliente envía otra petición al servidor solicitando el video específico.
4. **Transmisión de Video:** El servidor responde enviando el video al cliente, permitiendo su reproducción en tiempo real.

3. Funciones básicas de una aplicación (presentación, negocio y datos) entre el cliente y el servidor

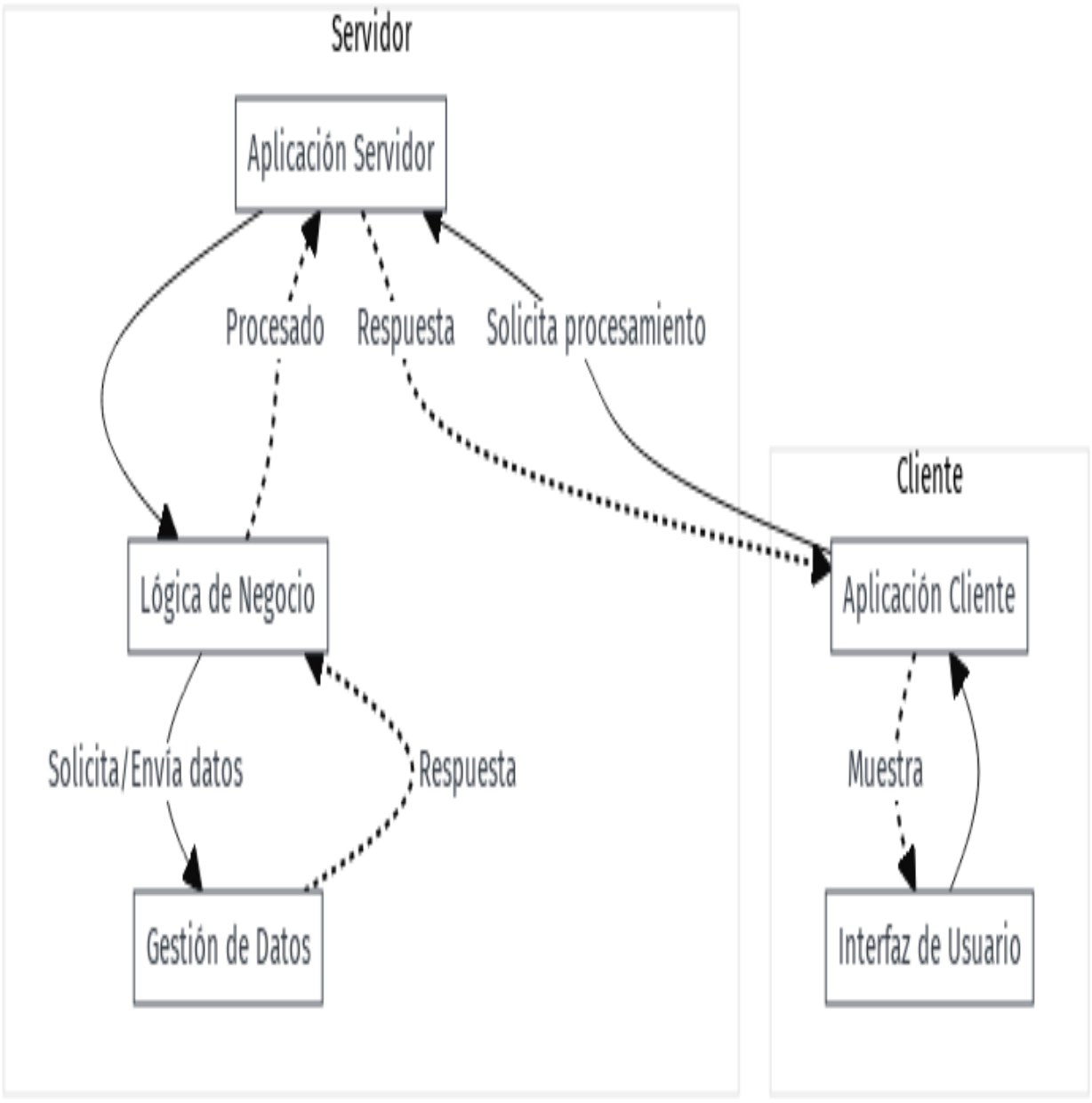
Cuando se desarrollan aplicaciones, especialmente en el entorno web moderno, uno de los aspectos clave a considerar es cómo se distribuyen y gestionan las tres funciones fundamentales de una aplicación: la presentación, la lógica de negocio y la gestión de datos. La forma en que estas funciones se reparten entre el cliente (usualmente el navegador web o la aplicación móvil del usuario) y el servidor puede tener un impacto significativo en el rendimiento, la escalabilidad, la seguridad y la experiencia general del usuario. Existen principalmente tres modelos de distribución que definen esta relación entre cliente y servidor: Presentación Distribuida, Aplicación Distribuida y Datos Distribuidos.

Cada modelo ofrece un enfoque diferente en cuanto a dónde se sitúan el procesamiento y almacenamiento de los datos, cómo se manejan las interacciones del usuario y cómo se ejecutan las operaciones de la aplicación. La elección entre estos modelos depende de múltiples factores, incluyendo los objetivos específicos de la aplicación, las expectativas de los usuarios, las limitaciones de infraestructura y los requisitos de seguridad.

Cada uno de estos modelos tiene sus ventajas y desventajas, y la elección entre ellos depende de los requisitos específicos de la aplicación, como la necesidad de rendimiento, escalabilidad, disponibilidad, y la experiencia de usuario deseada. En la práctica, muchas aplicaciones modernas combinan aspectos de estos tres modelos para crear soluciones híbridas que se ajustan mejor a sus necesidades.

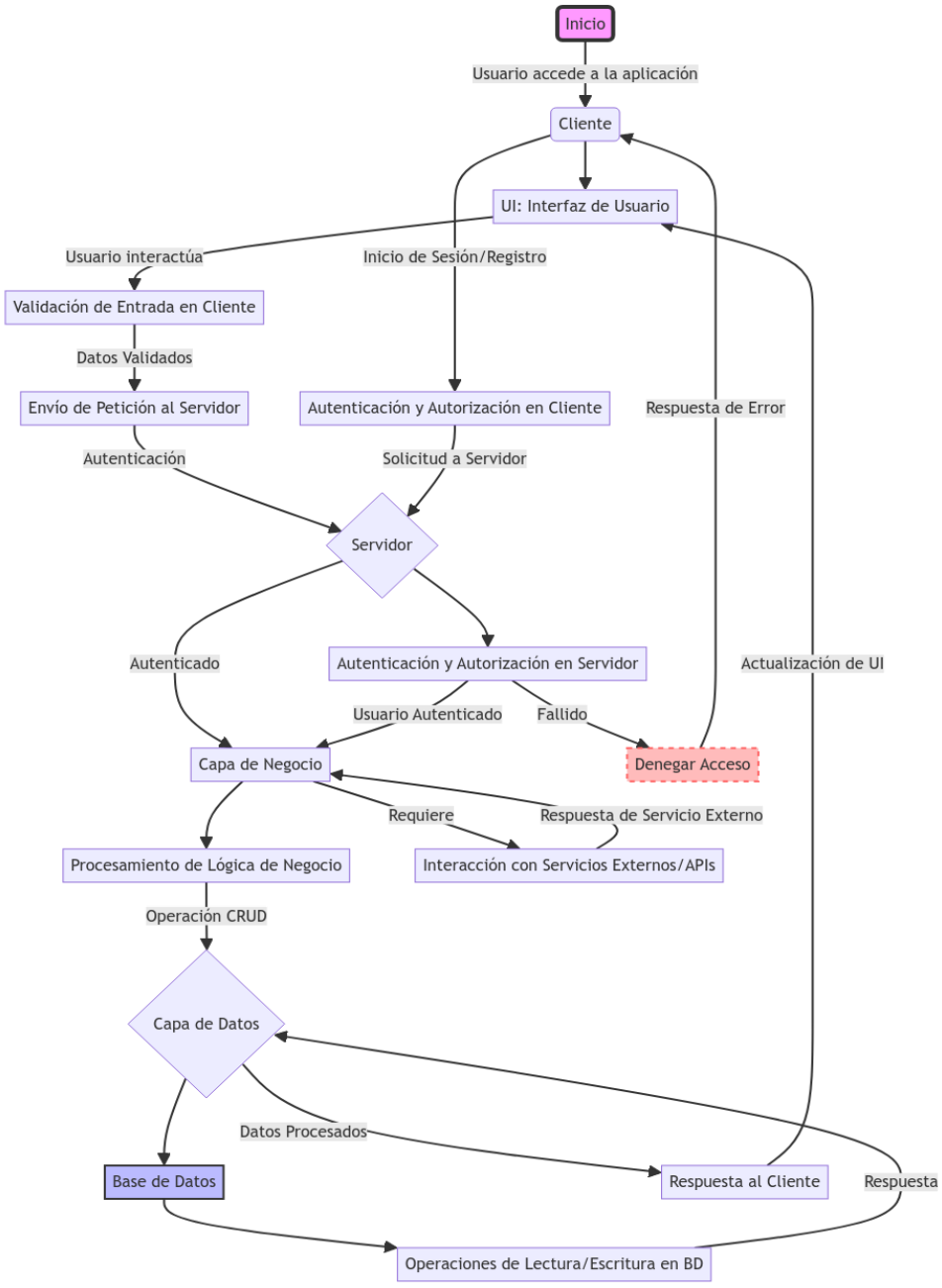
a. Presentación Distribuida

En un escenario de Presentación Distribuida, la información y los recursos se distribuyen entre diferentes sistemas y ubicaciones. El usuario interactúa con la aplicación a través de su dispositivo cliente, la lógica de negocio se ejecuta en el servidor de aplicaciones, y los datos se almacenan y se recuperan del servidor de datos.



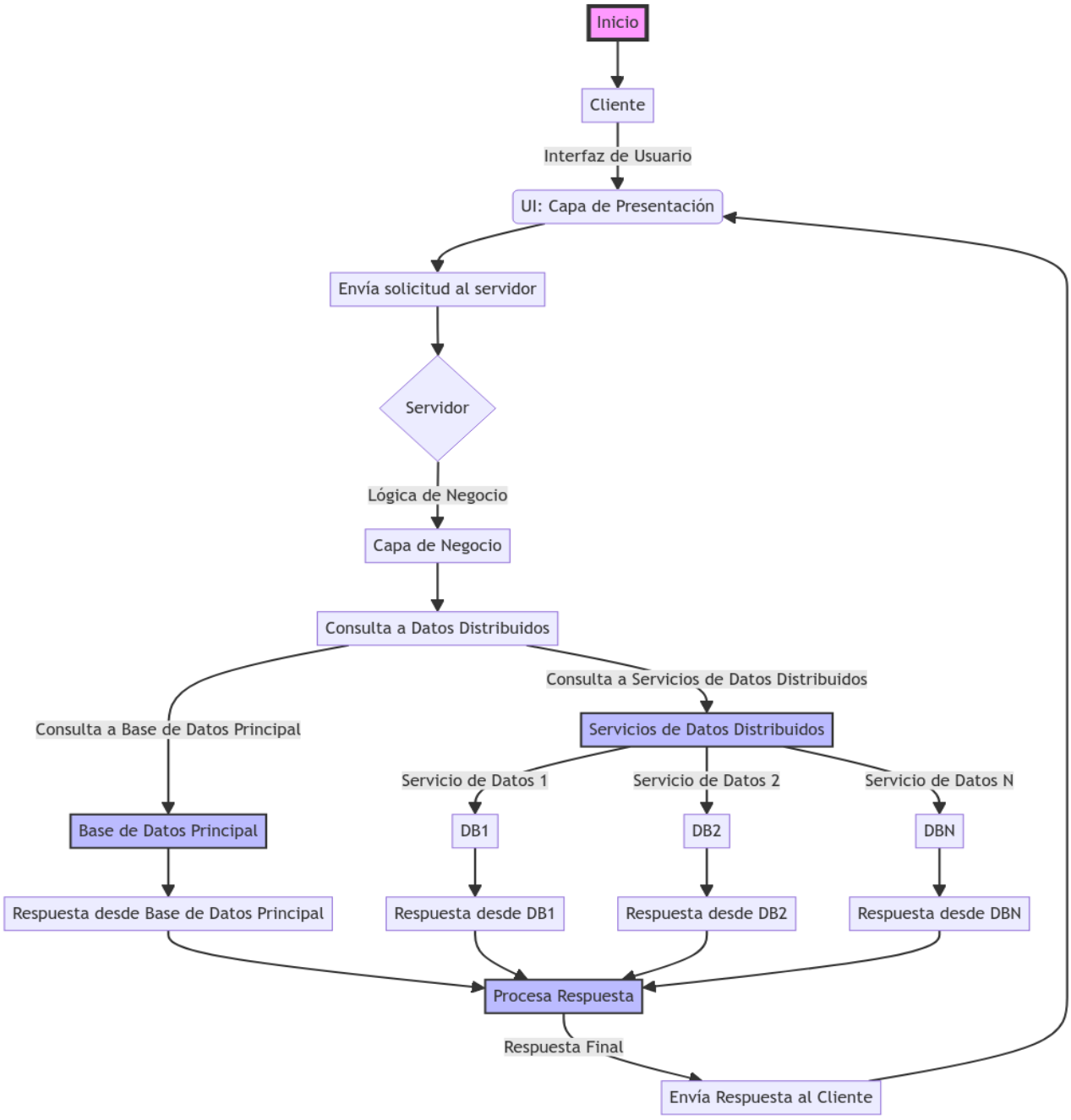
b. Aplicación Distribuida

El diagrama muestra el flujo de trabajo en una aplicación distribuida, comenzando con la interacción del usuario en el cliente (como una página web o una app móvil), pasando por la validación de entrada y la autenticación. Después, se envían las solicitudes al servidor, donde se procesan en la lógica de negocio y se interactúa con la base de datos o servicios externos según sea necesario. Finalmente, el resultado se devuelve al cliente para actualizar la interfaz de usuario. Este proceso incluye pasos de seguridad como la autenticación y la autorización para asegurar que solo los usuarios permitidos puedan realizar ciertas acciones.



c. Datos Distribuidos.

El diagrama ilustra el funcionamiento de una aplicación con datos distribuidos, enfocándose en cómo las solicitudes del usuario, desde la interfaz en el cliente, pasan por el servidor donde se procesan. La lógica de negocio en el servidor determina qué datos se necesitan, consultando tanto una base de datos principal como múltiples servicios de datos distribuidos. Estos servicios pueden estar en diferentes ubicaciones o servidores, permitiendo que la aplicación acceda a una amplia gama de datos. Las respuestas de estas diversas fuentes se combinan y procesan para enviar finalmente una respuesta actualizada al cliente, reflejando el resultado de su solicitud inicial.



4. Los patrones de arquitectura Web

Los patrones de arquitectura web son esquemas o plantillas reutilizables diseñados para resolver problemas comunes en el desarrollo de aplicaciones web. Estos patrones proporcionan una estructura organizada que ayuda a los desarrolladores a construir aplicaciones web eficientes, escalables y fáciles de mantener. Al seguir estos patrones, los equipos pueden acelerar el proceso de desarrollo, mejorar la calidad del código y facilitar la colaboración entre los miembros del equipo.

Los patrones de arquitectura no son soluciones terminadas que se pueden implementar directamente en el código; en cambio, son guías o mejores prácticas que describen cómo abordar ciertos problemas de diseño de software. Incluyen estrategias para estructurar el código, organizar los componentes de la aplicación, gestionar el flujo de datos, y cómo los diferentes elementos del sistema interactúan entre sí.

Los patrones de arquitectura web son fundamentales para el desarrollo de aplicaciones eficientes, escalables y fáciles de mantener. Algunos de los patrones más utilizados en la actualidad:

Modelo-Vista-Controlador (MVC)
Separa la aplicación en tres componentes principales: el modelo (que maneja la lógica y los datos), la vista (que maneja la presentación y la interfaz de usuario), y el controlador (que actúa como un intermediario entre el modelo y la vista). Es ampliamente usado en aplicaciones web para organizar el código de manera eficiente.

Ventaja: Separación de preocupaciones. Facilita el desarrollo, mantenimiento y prueba de aplicaciones al separar la lógica de negocio, la interfaz de usuario y la entrada/salida.

Desventaja: Complejidad en aplicaciones grandes. A medida que la aplicación crece, el manejo entre modelos, vistas y controladores puede volverse complejo y difícil de gestionar.

Ejemplo: Ruby on Rails, un framework de desarrollo web escrito en Ruby, es un ejemplo clásico de la implementación del patrón MVC. Este framework estructura las aplicaciones web dividiendo las responsabilidades entre los modelos (lógica de negocio y acceso a datos), las vistas (interfaz de usuario) y los controladores (lógica de aplicación), lo que facilita el desarrollo y mantenimiento.

Modelo-Vista-Presentador (MVP)
Similar al MVC, pero con una diferencia clave en el manejo de la lógica de presentación. En MVP, el Presentador tiene más responsabilidades que el Controlador en MVC, y trabaja más estrechamente con la Vista para actualizar la interfaz de usuario.

Ventaja: Mejora la comunicación entre la interfaz de usuario y la lógica de negocio al introducir el Presentador como mediador, facilitando el testeo de ambas partes por separado.

Desventaja: Aunque el presentador ayuda a separar la lógica de negocio de la UI, puede crear un acoplamiento fuerte entre la vista y el presentador, lo que dificulta su reutilización.

Ejemplo: Android apps. Aunque Android no impone un patrón de arquitectura específico, MVP ha sido popular entre los desarrolladores de Android para estructurar aplicaciones. En este contexto, el Presentador actúa como intermediario entre la Vista (actividades y fragmentos de la UI) y el Modelo (datos y lógica de negocio), gestionando la comunicación entre ellos.

Modelo-Vista-ViewModel (MVVM)
Introduce el componente ViewModel, que es responsable de exponer los datos del Modelo de una manera que sea fácil de manejar por la Vista. Es especialmente popular en aplicaciones que utilizan enlaces de datos, como las construidas con frameworks como Angular o Vue.js.

Ventaja: Enlace de datos bidireccional. Permite una sincronización automática entre la vista y el modelo de datos, lo que reduce el código de pegamento necesario y facilita la gestión de estados de la UI.

Desventaja: Curva de aprendizaje y complejidad. Implementar correctamente el patrón MVVM y el enlace de datos puede ser complejo y requiere una curva de aprendizaje, especialmente en proyectos grandes.

Ejemplo: Microsoft's .NET Framework. Particularmente, WPF (Windows Presentation Foundation) y Silverlight tecnologías donde MVVM ha sido ampliamente adoptado. Este patrón facilita el enlace de datos entre la vista y los datos de modelo a través del ViewModel, lo cual es especialmente útil en aplicaciones ricas en datos donde la actualización dinámica de la UI es frecuente.

Arquitectura Basada en Servicios (SOA)
Promueve el desarrollo de aplicaciones compuestas por servicios independientes que se comunican entre sí, típicamente a través de la web o llamadas a API. Esta arquitectura es útil para construir sistemas escalables y flexibles.

Ventaja: Reutilización y composición de servicios. Fomenta el desarrollo de servicios independientes que pueden ser combinados de diversas maneras para crear aplicaciones complejas, lo que mejora la flexibilidad y la escalabilidad**.**

Desventaja: Sobrecarga y rendimiento. La comunicación entre servicios puede introducir latencia y sobrecarga de rendimiento, especialmente si los servicios están distribuidos en diferentes redes o sistemas.

Ejemplo: Amazon Web Services (AWS). Amazon ha construido una vasta suite de servicios basada en la arquitectura SOA, permitiendo a los desarrolladores utilizar y combinar estos servicios de forma modular para construir aplicaciones escalables y flexibles en la nube.

Arquitectura Orientada a Microservicios
Similar a SOA pero con un enfoque en servicios más pequeños y altamente desacoplados que realizan una única función. Favorece la escalabilidad horizontal y facilita la implementación continua y la entrega continua.

Ventaja: Desacoplamiento y escalabilidad. Permite a los equipos desarrollar, desplegar y escalar partes de una aplicación de forma independiente, lo que puede conducir a ciclos de desarrollo más rápidos y a una mayor resiliencia.

Desventaja: Desacoplamiento y escalabilidad. Permite a los equipos desarrollar, desplegar y escalar partes de una aplicación de forma independiente, lo que puede conducir a ciclos de desarrollo más rápidos y a una mayor resiliencia.

Ejemplo: Netflix. Es uno de los ejemplos más citados de una aplicación que utiliza microservicios. Netflix evolucionó de una aplicación monolítica a una arquitectura basada en microservicios para mejorar la escalabilidad y la resiliencia de su plataforma de streaming global.

Arquitectura de Aplicaciones de Página Única (SPA)
Las SPA cargan una sola página web y actualizan dinámicamente el contenido sin necesidad de recargar la página completa. Utiliza intensamente JavaScript y frameworks como React, Angular o Vue.js para crear experiencias de usuario fluidas.

Ventaja: Experiencia de usuario fluida. Al minimizar las recargas de la página, las SPA pueden ofrecer una experiencia más cercana a la de una aplicación de escritorio, con tiempos de respuesta rápidos y transiciones suaves.

Desventaja: SEO y rendimiento inicial. Las SPA pueden enfrentar dificultades con la optimización de motores de búsqueda (SEO) y pueden tener tiempos de carga iniciales más largos debido a la carga del framework y el código de la aplicación**.**

Ejemplo: Gmail. Google implementó Gmail como una SPA para proporcionar una experiencia de usuario fluida y dinámica similar a una aplicación de escritorio. Esto permite a los usuarios interactuar con sus correos electrónicos de manera eficiente sin recargas de página completas.

Arquitectura de Aplicaciones Serverless
Permite a los desarrolladores construir y ejecutar aplicaciones y servicios sin preocuparse por la gestión del servidor. Los proveedores de la nube automáticamente aprovisionan, escalan y gestionan la infraestructura necesaria para ejecutar el código.

Ventaja: Eficiencia de costos y escalabilidad automática. Reduce la necesidad de administrar la infraestructura subyacente, ya que los recursos se autoescalanan y se paga solo por el tiempo de ejecución del código, no por el tiempo de inactividad del servidor.

Desventaja: Limitaciones del proveedor y arranque en frío. Las aplicaciones serverless pueden enfrentarse a limitaciones específicas del proveedor de la nube y problemas de latencia debido al arranque en frío de funciones inactivas.

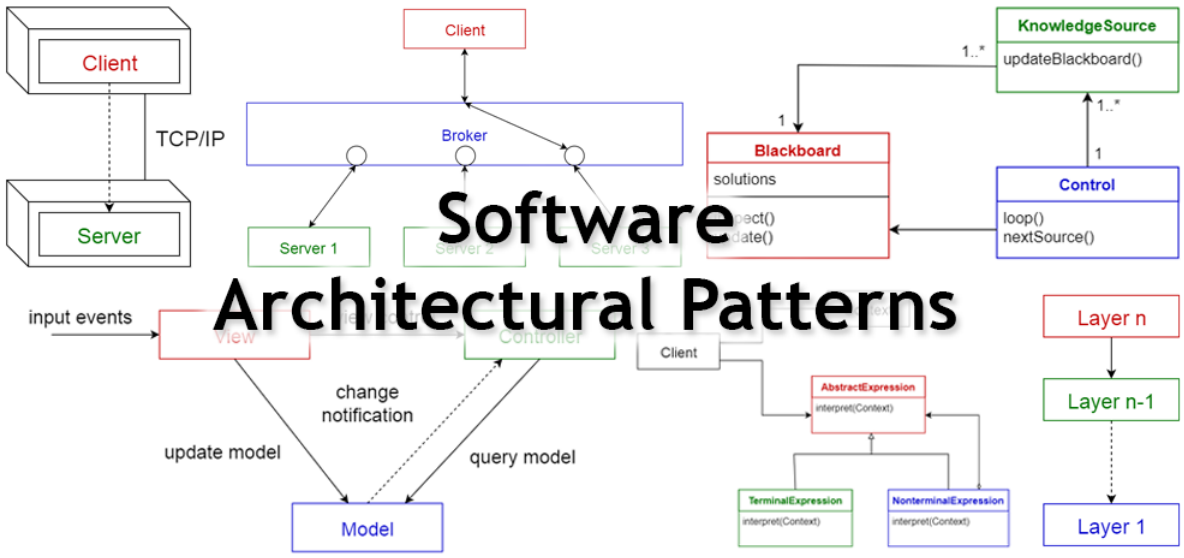
Ejemplo: AWS Lambda. Lambda permite a los desarrolladores ejecutar código en respuesta a eventos sin provisionar ni gestionar servidores, lo que es un ejemplo claro de cómo se puede implementar una arquitectura serverless para crear aplicaciones eficientes y escalables.

Arquitectura Orientada a Eventos
Se centra en la producción, detección y consumo de eventos o mensajes. Es ideal para aplicaciones que requieren integración en tiempo real entre sistemas desacoplados o para construir sistemas reactivos.

Ventaja: Reactividad y desacoplamiento. Permite que los componentes del sistema reaccionen a eventos o cambios de estado de forma desacoplada, lo que facilita la construcción de sistemas reactivos y escalables**.**

Desventaja: Complejidad de gestión de eventos. El manejo de eventos, especialmente en sistemas a gran escala, puede volverse complejo debido al seguimiento del flujo de eventos y la garantía de la entrega de mensajes.

Ejemplo: Apache Kafka. Aunque Kafka es más un producto que una aplicación específica, su uso en sistemas distribuidos para manejar flujos de datos y eventos en tiempo real ilustra cómo se puede implementar una arquitectura orientada a eventos para lograr alta escalabilidad y rendimiento.



Conclusión:

En conclusión, el universo de las aplicaciones web se extiende que va desde las simples páginas estáticas hasta sofisticadas plataformas de comercio electrónico y sistemas de gestión de contenidos, destacando la gran capacidad de la web para adaptarse a diversas necesidades de usuarios y empresas. Estas aplicaciones, fundamentadas en la arquitectura cliente/servidor, demuestran la importancia de una distribución eficaz de las tareas entre clientes y servidores para optimizar el rendimiento, la seguridad y la experiencia del usuario.

La diferenciación entre aplicaciones estáticas, dinámicas, de comercio electrónico y con gestor de contenidos refleja la evolución de las expectativas de los usuarios y las tecnologías web, impulsando un desarrollo continuo hacia soluciones más interactivas, personalizadas y accesibles. Los modelos de Presentación Distribuida, Aplicación Distribuida y Datos Distribuidos ofrecen estrategias específicas para la asignación de las funciones de presentación, negocio y datos, lo que permite a los diseñadores y desarrolladores optimizar recursos y mejorar la interacción con el usuario.

Los patrones de arquitectura web representan una herramienta invaluable en este contexto, proporcionando soluciones probadas a problemas comunes y fomentando la reutilización de código, lo cual facilita la creación de aplicaciones robustas, escalables y fáciles de mantener. Estos patrones no solo contribuyen a la eficiencia del desarrollo, sino que también aseguran que las aplicaciones puedan adaptarse a los cambios tecnológicos y a las demandas del mercado con mayor agilidad.

Por tanto, la comprensión profunda de los tipos de aplicaciones web, la arquitectura, los modelos de distribución de funciones y los patrones de diseño son esenciales para cualquiera en el campo del desarrollo web, ya sea para la creación de nuevas aplicaciones o para la mejora de las existentes. Mirando hacia el futuro, la innovación continua en estas áreas será clave para aprovechar al máximo las oportunidades que ofrece la tecnología web, enfrentando los desafíos emergentes y satisfaciendo las expectativas de los usuarios.

Referencias:

RIDE. Rev. Iberoam. Investig. Desarro. Educ vol.8 no.15 Guadalajara jul./dic. 2017

https://doi.org/10.23913/ride.v8i15.323

Pastor Pérez, J. (2013). Estudio y clasificación de tipos de aplicaciones Web y determinación de atributos de usabilidad más relevantes.

http://hdl.handle.net/10251/32839.

3c Tecnología: glosas de innovación aplicadas a la pyme, ISSN-e 2254-4143, Vol. 7, Nº. 3, 2018, págs. 28-49

https://dialnet.unirioja.es/servlet/articulo?codigo=6551743

Ttito, J., Phele, J. (2021). Desarrollo del sistema Intranet utilizando Cliente Servidor para optimizar los servicios en el centro de salud Mazuko, 2018 [Tesis, Universidad Nacional Amazónica de Madre de Dios].

http://hdl.handle.net/20.500.14070/725

LUJÁN MORA, Sergio. Programación en Internet: clientes web. Alicante : Editorial Club Universitario, 2001. ISBN 978-84-8454-118-9, 224 p.

http://hdl.handle.net/10045/16994

Valverde Giromé, F. (2010). OOWS 2.0: UN MÉTODO DE INGENIERÍA WEB DIRIGIDO POR MODELOS PARA LA PRODUCCIÓN DE APLICACIONES WEB 2.0 [Tesis doctoral no publicada]. Universitat Politècnica de València.

https://doi.org/10.4995/Thesis/10251/8977