ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Tema 1.3:

Estructura de una Aplicación Web (AW).

Introducción

La estructura de una aplicación web se refiere a los componentes y la organización que permiten su correcto funcionamiento. Comprender estos elementos es fundamental para desarrollar aplicaciones web eficientes y escalables.

Objetivo:

Conocer y entender los elementos básicos que componen una aplicación web y su estructura organizativa para facilitar el desarrollo y mantenimiento de aplicaciones web robustas.

Conceptos Importantes.

Cliente-Servidor.

Un cliente es un dispositivo que necesita tener acceso a ciertos servicios para poder acceder a un servidor que brinda servicios ante las peticiones que realiza a los ordenadores, los cuales se ven conectados mediante una red Lituma-Sarmiento, A. F., & Vizñay-Durán, J. K. (2023).

Conceptos Importantes.

FrontEnd (Lado del Cliente).

El desarrollo del frontend desempeña un papel fundamental en la construcción de aplicaciones o sistemas web, ya que proporciona una experiencia agradable y navegación óptima al visitar una página. Adicional a ello, su relevancia se resume en su capacidad para no solo presentar una interfaz visual atractiva, sino también mejorar la accesibilidad y el rendimiento de la página. Galarza Jiménez, K. A. (2023).

Conceptos Importantes.

Servidor (Backend).

El backend se refiere a las partes de una aplicación informática o del código de un programa que permiten su funcionamiento y a las que no puede acceder un usuario. La mayoría de los datos y la sintaxis de funcionamiento se almacenan y se accede a ellos en el back end de un sistema informático. Normalmente, el código se compone de uno o varios lenguajes de programación. El backend también se denomina capa de acceso a los datos del software o del hardware e incluye cualquier funcionalidad a la que sea necesario acceder y navegar por medios digitales (Vallejo Cano, 2022).

.

Conceptos Importantes.

Base de Datos.

Según Continental University of Florida, un Sistema de Gestión de Bases de Datos o SGBD (también conocido en inglés como Data Base Management System o DBMS) es un software que brinda la posibilidad de administrar y gestionar una base de datos, es decir permite utilizar, configurar, extraer y recuperar información almacenada, con el objetivo de que los usuarios accedan a dicha información a través de una interfaz desde donde puedan interactuar mediante la lectura, creación, eliminación y/o actualización de datos.

.

Conceptos Importantes.

API (Interfaz de Programación de Aplicaciones).

Una Interfaz de Programación de Aplicaciones (API) es un conjunto de funciones, procedimientos o métodos creados para ser consumidos o utilizados por otro software de forma segura. Una API facilita la comunicación entre aplicaciones en formato estándar para el intercambio de datos.

Las APIs pueden habilitar el acceso a sus recursos a aplicaciones de terceros de forma controlada y segura Lara Méndez, M. A. (2021).

Conceptos Importantes.

Middleware.

Según Amazon Web Services (2024), un Middleware es un software con el que las diferentes aplicaciones se comunican entre sí. Brinda funcionalidad para conectar las aplicaciones de manera inteligente y eficiente, de forma que se pueda innovar más rápido. El middleware actúa como un puente entre tecnologías, herramientas y bases de datos diversas para que pueda integrarse sin dificultad en un único sistema. Este sistema único provee un servicio unificado a sus usuarios. Por ejemplo, una aplicación frontend de Windows envía y recibe datos desde un servidor backend de Linux, pero los usuarios de la aplicación no están al tanto de la diferencia.

Ejemplo.

Imaginemos la creación de una aplicación de Tienda Virtual en Línea, para ello empezamos por la etapa de Ingeniería de Requisitos.

Nos enfocaremos en las etapas de:

- **Análisis de Requisitos.**
- **Diseño de la Aplicación.**

Análisis de Requisitos.

Identificación de Actores

Administrador: Gestiona productos, categorías, usuarios y órdenes.

Cliente: Navega, añade productos al carrito, realiza compras y consulta sus pedidos.

Sistema de Pago: Procesa transacciones de pago de manera segura.

Análisis de Requisitos.

Requiitos Funcionales.

Gestión de Productos: El administrador puede añadir, editar y aliminar productos y categorías.

Navegación de Productos: Los clientes pueden buscar y filtrar productos por categoría, precio y popularidad.

Carrito de Compras: Los clientes pueden añadir productos al carrito, modificar cantidades y eliminar productos.

Proceso de Pago: Los clientes pueden realizar pagos seguros a través de una pasarela de pago.

Gestión de Pedidos: Los clientes pueden ver el estado d sus pedidos y el historial de compras.

Análisis de Requisitos.

Requisitos Funcionales.

Autenticación y Autorización: Usuarios deben registrarse e iniciar sesión para realizar compras y gestionar productos.

Notificaciones: Envío de correos electrónicos para confirmaciones de pedidos y notificaciones importantes.

Análisis de Requisitos.

Requisitos No Funcionales.

Seguridad: Protección de datos sensibles, uso de HTTPS y autenticación segura.

Escalabilidad: La aplicación debe manejar un creciente número de usuarios y transacciones sin afectar el rendimiento.

Usabilidad: La interfaz debe ser intuitiva y fácil de usar tanto en dispositivos móviles como en ordenadores.

Rendimiento: Respuestas rápidas a las solicitudes de los usuarios, con tiempos de carga mínimos.

Mantenibilidad: Código limpio y bien documentado para facilitar futuras actualizaciones y mantenimiento.

Diseño de la Aplicación.

Arquitectura de la Aplicación

Frontend (Cliente).

Componentes:

- **Catálogo de Productos:** Muestra lista de productos.
- **Carrito de Compras:** Visualiza y gestiona los productos seleccionados para la compra.
- **Formulario de Pago:** Gestiona la entrada de datos de pago y dirección.
- **Historial de Pedidos:** Muestra las órdenes pasadas y su estado.

Diseño de la Aplicación.

Arquitectura de la Aplicación

Backend (Servidor).

- **Framework:** Node.js con Express.js
- **API RESTful:**
  - **Endpoints:**
    - **/products:** CRUD de productos
    - **/categories:** CRUD de categorías.
    - **/users:** Registro y autenticación de usuarios.
    - **/orders:** Gestión de órdenes de compra
    - **/cart:** Operacoines sobre el carrito de compras.
- **Autenticación:** Implementación de JWT (Json Web Tokens) para la autenticación y autorización de usuarios.

Diseño de la Aplicación.

Arquitectura de la Aplicación

Base de Datos

- **Tipo:** NoSQL, utilizando MongoDB
- Colecciones:
    - **Users:** Información de los usuarios (nombre, correo, contraseña encriptada).
    - **Products:** Información de los productos (nombre, descripción, precio, categoría, stock)
    - **Categories:** Información de las categorías de productos.
    - **Orders:** Detalles de los pedidos realizados (usuario, productos, estado, fecha)
    - **Cart:** Información temporal de los carritos de compra.

Diseño de la Aplicación.

Arquitectura de la Aplicación

Integraciones

- **Pasarela de Pago:** Integración con servicios como Stripe o PayPal para procesar pagos.
- **Correo Electrónico:** Integración con servicios de correo electrónico para notificaciones y confirmaciones (por ejemplo, SendGrid)

Diseño de la Aplicación.

Diagramas

- **Diagrama de Componentes:**
- **Diagrama de Flujo de Datos:**

Diseño de la Aplicación.

Mockups de Interfaces

- Pantalla de Catálogo de Productos
- Pantalla de Carrito de Compras
- Pantalla de Pago

Diseño de la Aplicación.

Consideraciones Técnicas

- **Seguridad:** Uso de HTTPS, almacenamiento seguro de contraseñas (bcrypt), protección contra ataques XSS y CSRF.
- **Rendimiento:** Uso de técnicas de lazy loading, cacheo de datos y optimización de consultas a la base de datos.
- **Despliegue:** Uso de Docker para la contenedorización de la aplicación y despliegue en plataformas como AWS, Heroku o Digital Ocean.

Tarea

Analizar y Diseñar una Aplicación Web

En equipos de 3 estudiantes, realizar las siguientes actividades:

1. Seleccionar una aplicación web existente (por ejemplo, una red social, un sistema de gestión, una tienda en línea).
2. Describir los elementos que componen la aplicación web seleccionada.
3. Elaborar un diagrama de estructura de la aplicación, identificando los componentes frontend, backend, base de datos, API y middleware.
4. Presentar un informe en pdf con el análisis y el diagrama elaborado.

Recursos

- **Diseñar una Aplicación con Frontend, Backend y Data Base**
- **Porque no podemos conectar el Frontend directamente a la Base de Datos**
- **Cómo comenzar a DISEÑAR una WEB o APP**

Conclusiones

Comprender la estructura de una aplicación web es esencial para el desarrollo de sistemas eficientes y mantenibles. Identificar y organizar correctamente los componentes frontend, backend, base de datos, API y middleware facilita el desarrollo colaborativo y la escalabilidad de la aplicación.

Recomendaciones

- Utilizar frameworks y bibliotecas populares para asegurar la eficiencia y la comunidad de soporte.
- Seguir buenas prácticas de seguridad, especialmente en la comunicación entre frontend y backend.
- Documentar la estructura de la aplicación para facilitar el mantenimiento y futuras actualizaciones.

Bibliografía

Amazon Web Services. (2024). ¿Qué es el middleware?. Recuperado de aquí

Continental University of Florida. (2023, febrero 16). Sistema de gestión de base de datos: todo lo que debes saber. CUF Digital. Recuperado de aquí.

Galarza Jiménez, K. A. (2023). Desarrollo de sistema de gestión de venta de licores para la licorería “La Nenita”: Desarrollo de un Frontend. Recuperado de aquí

Lara Méndez, M. A. (2021). Modelo de una interfaz de programación de aplicaciones REST utilizando GO basado en normas y principios de seguridad de la información y aplicaciones web. Escuela Superior Politécnica de Chimborazo. Riobamba, Recuperado de aquí

Lituma-Sarmiento, A. F., & Vizñay-Durán, J. K. (2023). Análisis y Diseño de una propuesta de sistema integral de Gestión Empresarial basado en una arquitectura Cliente-Servidor. MQRInvestigar. Recuperado de aquí

Bibliografía

Vallejo Cano, K. L. (2022). Estudio comparativo de las tecnologías para el desarrollo del back-end “nodejs” y “php”. [Tesis de grado, Universidad de Ejemplo]. Repositorio Digital de la Universidad de Ejemplo. Recuperado de aquí