EJERCICIO 5: ARQUITECTURA DEL BACKEND

1. Tecnologías a utilizar:

Node.js: es un lenguaje de programación muy utilizado para el desarrollo de aplicaciones web, y cuenta con una amplia variedad de librerías y frameworks para el desarrollo de backend.

Express: es un framework de Node.js que simplifica el desarrollo de aplicaciones web al proporcionar un conjunto de herramientas para el manejo de rutas y templates.

MongoDB: es una base de datos NoSQL que se adapta muy bien a aplicaciones web y permite almacenar grandes cantidades de datos de forma escalable y flexible.

Mongoose: es una librería de Node.js que proporciona una capa de abstracción para interactuar con MongoDB desde JavaScript.

JWT: es una tecnología de autenticación y autorización basada en tokens que permite proteger las rutas y endpoints de la aplicación de acceso no autorizado.

2. Organización de archivos:

Es recomendable organizar los archivos del backend en diferentes carpetas para facilitar su mantenimiento y escalabilidad. Algunas de las carpetas que podrían utilizarse son:

config: para archivos de configuración de la aplicación.

controllers: para definir las funciones que manejan las peticiones de los clientes.

models: para definir los modelos de datos que se utilizan en la aplicación.

routes: para definir las rutas de la aplicación.

middleware: para definir middleware que se ejecutan antes o después de las funciones de los controladores.

utils: para definir funciones o herramientas que se utilizan en diferentes partes de la aplicación.

3. Patrones de diseño:

Es recomendable utilizar patrones de diseño en el desarrollo del backend para mejorar la calidad del código, el modularidad y la escalabilidad de la aplicación. Algunos patrones de diseño que podrían utilizarse son:

MVC (Modelo-Vista-Controlador): para separar la lógica de la aplicación en diferentes capas y facilitar su mantenimiento y escalabilidad.

Singleton: para garantizar que solo haya una instancia de una clase en la aplicación.

EJERCICIO 6: NOMENCLATURA

Bases de Datos:

Nombra las bases de datos con un nombre descriptivo y fácil de recordar.

Utiliza minúsculas para los nombres de las bases de datos y evita usar espacios.

Utiliza subrayados para separar palabras si es necesario.

Variables:

Nombra las variables de forma descriptiva y significativa.

Utiliza nombres que reflejen el propósito de la variable.

Utiliza camelCase para nombrar las variables.

Evita utilizar nombres de variables abreviados o genéricos, como "x", "y" o "data".

Funciones:

Nombra las funciones de forma descriptiva y significativa.

Utiliza verbos en infinitivo para nombrar las funciones.

Utiliza camelCase para nombrar las funciones.

Utiliza nombres de funciones que reflejen el propósito de la función y las acciones que realiza.

Clases:

Nombra las clases con nombres en singular y en mayúscula la primera letra de cada palabra.

Utiliza nombres que reflejen el propósito de la clase.

Utiliza CamelCase para nombrar las clases.

Git:

Utiliza nombres de ramas de git descriptivos y significativos.

Utiliza nombres que reflejen el propósito de la rama.

Utiliza guiones para separar las palabras en el nombre de la rama.

Utiliza minúsculas para los nombres de las ramas.

Otros:

Evita utilizar nombres genéricos, como "temp" o "dummy".

Utiliza nombres que sean fáciles de recordar y de entender.

Evita utilizar números y caracteres especiales en los nombres.

Utiliza nombres que sean lo suficientemente descriptivos para que otros desarrolladores puedan entender su propósito.

Al seguir estas directrices, podemos crear un código y una documentación más claros, consistentes y fáciles de entender, lo que facilitará la colaboración en el equipo de desarrollo y mejorará la eficiencia en nuestros proyectos.