Arquitectura del BACKEND

ARBEY GIOVANNY CHICA GIL

MEDLLÍN

ANTIOQUIA

COLOMBIA

ENERO – 2024

**INTRODUCCIÓN**  
  
  
Desarrollo de arquitectura para el desarrollo de software desde el lado del Backend estableciendo parámetros sobre cómo o porque elegir una tecnología u otra para implementar en el desarrollo.

[TECNOLOGIAS: 3](#_Toc231479931)

[ESTRUCTURA Y/O ORGANIZACIÓN DE ARCHIVOS: 4](#_Toc790393035)

[PATRONES DE DISEÑO: 4](#_Toc1992888332)

[SEGURIDAD 5](#_Toc1854108154)

[ESCALABILIDAD 5](#_Toc652993422)

Arquitectura del Backend

## **TECNOLOGIAS**:

1. **Lenguaje de Programación**:

Elegir uno sostenible, con comunidad y escalable para mantener el proyecto en el tiempo.

para el ejemplo: C#

1. **Framework**:

Elegir uno que nos permita desarrollar de forma rápida y escalable.

que sea seguro, confiable y que cuente con comunidad amplia comunidad de desarrolladores, para posibles errores/bugs.

para el ejemplo: .NET Core

1. **Base de Datos**:

Elegir una base de datos que nos permita almacenar la información de forma segura y escalable.

para un comercio electrónico, se la mejor opción podría ser una base de datos relacional.

para el ejemplo: SQL Server

1. **ORM**:

Elegir un ORM que nos permita mapear la base de datos con el lenguaje de programación,

para poder realizar las consultas de forma más sencilla y escalable. y que sea rápido para ahorrar tiempo de desarrollo.

para el ejemplo: Entity Framework Core

1. **Autenticación**:

implementar un sistema de autenticación que nos permita validar los usuarios que se conectan a la aplicación.

que permita mantener la seguridad de la información y que sea escalable.

para el ejemplo: JWT = Json Web Token

1. **API REST**:

diseñar y desarrollar una API REST que nos permita comunicar el Frontend con el Backend.

que permita mantener la comunicación entre los dos sistemas priorizando la seguridad apoyando la escalabilidad con microservicios.

para el ejemplo: ASP.NET Core Web API

## **ESTRUCTURA Y/O ORGANIZACIÓN DE ARCHIVOS:**

1. **Organización**:

Dividir el proyecto por MODULOS y/o ENTIDADES, para mantener un orden y escalabilidad.

para el ejemplo: USUARIOS - PRODUCTOS - PEDIDOS - ETC.

1. **Capas**:

que el proyecto use capaz es una buena opción para mantener un orden entre los módulos y secciones del proyecto.

siendo uno muy popular y usado en proyectos de gran escala, el patrón MVC (Modelo Vista Controlador).

## **PATRONES DE DISEÑO**:

1. **Patrón MVC**:

Modelo Vista Controlador, es un patrón de diseño que nos permite separar la lógica de negocio de la interfaz de usuario.

1. **Inyección de Dependencias**:

es un patrón de diseño que nos permite inyectar las dependencias de una clase en otra, para poder usarlas sin necesidad de instanciarlas.

de igual manera que permita facilitar las pruebas unitarias y la reducción de acoplamiento entre clases.

1. **Repositorio**:

es un patrón de diseño que nos permite separar la logica de negocio de la logica de acceso a datos.

usando el encapsulamiento de las clases para poder acceder a los datos de forma más sencilla.

## **SEGURIDAD**

1. **HTTPS:**

el enrutamiento HTTP permite la conexión entre el Frontend y el Backend, de forma que garantice la seguridad de la información.

1. **Validaciones**:

validar los datos ingresados y la forma en que estos son ingresados para evitar posibles ataques de inyección de código y/o "SQL Injection".

1. **Autenticación CSRF/XSS**:

permiten obtener seguridad ante posibles ataques CSRF mediante solicitudes y falsificación de solicitudes entre sitios.

## **ESCALABILIDAD**

1. **Balanceo de cargas**:

es una técnica que nos permite distribuir la carga de trabajo entre varios servidores, para evitar sobrecargas y caídas del sistema.

1. **Escalabilidad Horizonta**l:

es una técnica que nos permite aumentar la capacidad de procesamiento de un sistema, aumentando el número de servidores,

para que pueda manejar un aumento en la carga de trabajo al agregar más instancias del servidor.

1. **Caché**:

usar un sistema cache para que optimice el rendimiento y tiempo de respuesta del sistema, al almacenar datos que se usan con frecuencia.

1. **Microservicios**:

permite dividir el sistema en servicios independientes, que se comunican entre sí, para que puedan ser escalables y mantenibles.