



ANEP



UTU

DIRECCIÓN GENERAL
DE EDUCACIÓN
TÉCNICO PROFESIONAL



ESI
Escuela Superior de Informática

Documentación Programación

ZENET

Integrantes: Damian Suffo, Gaston Ferron, Jorge Gallero.



Índice

Índice.....	2
Arquitectura usada.....	2
Facilidad de desarrollo.....	2
Facilidad de mantenimiento.....	3
Facilidad de escalado.....	3
Backend.....	3
¿Qué es?.....	3
API'S.....	4
Concepto de api.....	4
Api de almacén.....	4
Api de seguimiento.....	4
Api de autenticación.....	4
Estándar REST.....	5
"Representational State Transfer".....	5
Backoffice de administración.....	6
Aplicación de almacen.....	6
Aplicación de chofer.....	7
Aplicación de Seguimiento.....	7

Arquitectura usada

El patrón MVC, o Modelo-Vista-Controlador, es un patrón de arquitectura de software que separa la lógica de la interfaz de usuario de la lógica de negocio. Esto hace que las aplicaciones sean más fáciles de desarrollar, mantener y escalar.

En el MVC, la lógica de la interfaz de usuario se encuentra en la vista. La vista es responsable de mostrar la información al usuario y de responder a sus interacciones. La lógica de negocio se encuentra en el modelo. El modelo es responsable de almacenar los datos y de realizar las operaciones sobre ellos. El controlador es el encargado de conectar la vista con el modelo.

Esta separación de responsabilidades hace que las aplicaciones MVC sean más fáciles de mantener. Si hay algún problema con la interfaz de usuario, solo es necesario

modificar la vista. Si hay algún problema con la lógica de negocio, solo es necesario modificar el modelo.

El MVC también facilita el escalado de aplicaciones. Si la aplicación necesita ser ejecutada en un entorno con un gran número de usuarios, se puede escalar el modelo o el controlador. Esto no afecta a la vista, que puede seguir siendo utilizada sin cambios.

Facilidad de desarrollo

El MVC separa la lógica de la interfaz de usuario de la lógica de negocio. Esto hace que sea más fácil desarrollar aplicaciones complejas porque los desarrolladores pueden concentrarse en una parte del código a la vez.

Por ejemplo, un desarrollador puede trabajar en la lógica de la interfaz de usuario sin tener que preocuparse por la lógica de negocio. Esto hace que el desarrollo sea más eficiente y reduce el riesgo de errores.

Facilidad de mantenimiento

El MVC hace que sea más fácil mantener las aplicaciones porque las diferentes partes del código están bien separadas. Esto hace que sea más fácil identificar y corregir los errores.

Por ejemplo, si hay un error en la lógica de la interfaz de usuario, solo se tiene que modificar el código de la interfaz de usuario. Esto hace que el mantenimiento sea más rápido y sencillo.

Facilidad de escalado

El MVC hace que sea más fácil escalar las aplicaciones porque las diferentes partes del código pueden ser ejecutadas en diferentes servidores. Esto permite que el sistema se adapte a un mayor volumen de tráfico.

Por ejemplo, si el sistema necesita procesar más pedidos, se pueden añadir más servidores para ejecutar la lógica de negocio. Esto permite que el sistema se escale de forma horizontal para satisfacer la demanda de los usuarios.

En conclusión, el MVC es una buena elección para nuestros proyectos porque ayuda a hacer que nuestros proyectos sean más fáciles de desarrollar, mantener y escalar a futuro.

Backend

¿Qué es?

El backend es la parte de un sistema que se encarga de procesar los datos y proporcionar los servicios necesarios para la interfaz del usuario. En otras palabras, el backend es lo que ocurre en segundo plano cuando interactuamos con una aplicación o un sitio web.

En Zennet, usamos el backend para gestionar la lógica de negocio de nuestros proyectos (En este caso ZTracking, el sistema de QuickCarry). La lógica de negocio es el conjunto de reglas que definen cómo funciona un sistema. Por ejemplo, la lógica de negocio de un sistema de e-commerce podría definir cómo se procesan los pedidos, cómo se calculan los precios o cómo se gestiona el inventario y los posibles envíos si los hay.

API'S

Concepto de api

Una API es un conjunto de definiciones y protocolos que permiten que dos sistemas se comuniquen entre sí. En otras palabras, una API es una forma de que un sistema "hable" con otro sistema.

En Zennet, usamos las APIs para exponer los servicios del backend a otros sistemas. Por ejemplo, podríamos usar una API para exponer los servicios de un sistema de e-commerce a una plataforma de comercio electrónico. Esto permitiría que la plataforma de e-commerce realizará pedidos o envíos, gestionará el inventario o recupera información sobre los clientes.

Api de almacén

- API de Almacén: Esta API se encarga de gestionar la

Información relativa al almacén, los productos y los lotes.

Esta API debe implementar el estándar REST, y está restringida a la aplicación de los funcionarios del almacén.

Api de seguimiento

- API de tránsito o seguimiento: Esta API se encarga de gestionar la información relativa al tránsito de los productos, los camiones, el trayecto y sus estados. Esta API implementa el estándar REST. La utilizan tanto la aplicación de camioneros como la aplicación de tránsito como origen de información, por lo que debe contemplar diferentes niveles de acceso a la información y su manipulación.

Api de autenticación

- API de autenticación: Esta aplicación es una API que se encarga de gestionar la autenticación de cada API, así como el alcance de permisos de los mismos. Debe proveer un token de autenticación para realizar las tareas correspondientes en cada aplicación, así como proveer el alcance de permisos para cada usuario.

Estándar REST

“Representational State Transfer”

Es un conjunto de restricciones para definir como la arquitectura de un sistema se debe comportar.

Sus reglas o principios son:

- Cliente (Frontend)-servidor (Backend): El cliente y el servidor son entidades independientes que se comunican entre sí.
- Recursos: Los datos se modelan como recursos que pueden ser accedidos a través de URLs.
- Estado de representación: El estado del recurso se representa en el cuerpo de la respuesta, también debe devolver una respuesta http.
- Cacheo: Los recursos pueden ser almacenados en caché para mejorar el rendimiento (Aplica en aplicaciones o sistemas webs).

Ejemplo de un servicio REST

Un ejemplo de un servicio REST es una API que expone la información sobre los productos de un catálogo. El servicio podría definir los siguientes recursos:

- `/productos`: Lista de todos los productos.
- `/productos/{id}`: Información sobre un producto específico.

El servicio podría utilizar los siguientes métodos HTTP para acceder a los recursos:

- GET: Obtener información sobre un recurso.
- POST: Crear un nuevo recurso.
- PUT: Actualizar un recurso existente.
- DELETE: Eliminar un recurso.
- PATCH: Actualiza parcialmente un recurso existente.

Backoffice de administración

- Backoffice de Administración: Esta aplicación se encarga de gestionar todo el funcionamiento del sistema, como manipulación de usuarios, almacenes, así como datos de las otras aplicaciones como camiones, lotes, paquetes, trayectos y diferentes status. usa el MVC como arquitectura y c# .net.

Aplicación de almacén

La aplicación de almacén hecha en C# Windows Forms es una aplicación que permite a los usuarios realizar operaciones de almacenamiento, como crear, actualizar, eliminar y consultar productos. La aplicación consume dos APIs: la API de auth y la API de almacén.

La API de auth proporciona servicios para autenticar y autorizar a los usuarios. La aplicación utiliza la API de auth para autenticar a los usuarios antes de permitirles acceder a las operaciones de almacenamiento.

La API de almacén proporciona servicios para acceder a los datos de almacenamiento. La aplicación utiliza la API de almacén para realizar operaciones de almacenamiento, como crear, actualizar, eliminar y consultar productos, lotes, envíos etc.

La aplicación cuenta con 3 partes, frontend (Todo lo visual, botones, inputs etc) La segunda parte, las requests (son las peticiones a las apis, la forma en la cual se comunican los diferentes sistemas y se trae información de la base de datos) y la tercera parte es el módulo de traducción del sistema.

El sistema de traducción funciona con archivos de recursos los cuales tienen cargados ya los mensajes con un identificador el cual se establecieron en una clase ya creada llamada "Messages" con su respectivo controlador o manager llamado "LanguageManager" el mismo está vinculado a un evento creado en el main form. A su vez los form's hijos también "escuchan" o ven ese cambio para que también puedan saber cuando el usuario selecciona un idioma u otro.

Aplicación de chofer

La aplicación de choferes hecha en C# Windows Forms es una aplicación que permite a los choferes realizar operaciones relacionadas con su trabajo, como crear, actualizar, eliminar y consultar viajes. La aplicación consume dos APIs: la API de auth y la API de viajes.

La API de auth proporciona servicios para autenticar y autorizar a los usuarios. La aplicación utiliza la API de auth para autenticar a los choferes antes de permitirles acceder a las operaciones de viajes.

La API de viajes proporciona servicios para acceder a los datos de viajes. La aplicación utiliza la API de viajes para realizar operaciones de viajes, como crear, actualizar, eliminar y consultar viajes.

Esta aplicación cuenta con el mismo código o módulo de traducción que la aplicación de almacen.

Aplicación de Seguimiento

La aplicación de seguimiento hecha en C# Windows Forms es una aplicación que permite a los clientes realizar un seguimiento de sus pedidos. La aplicación consume una API: la API de seguimiento.

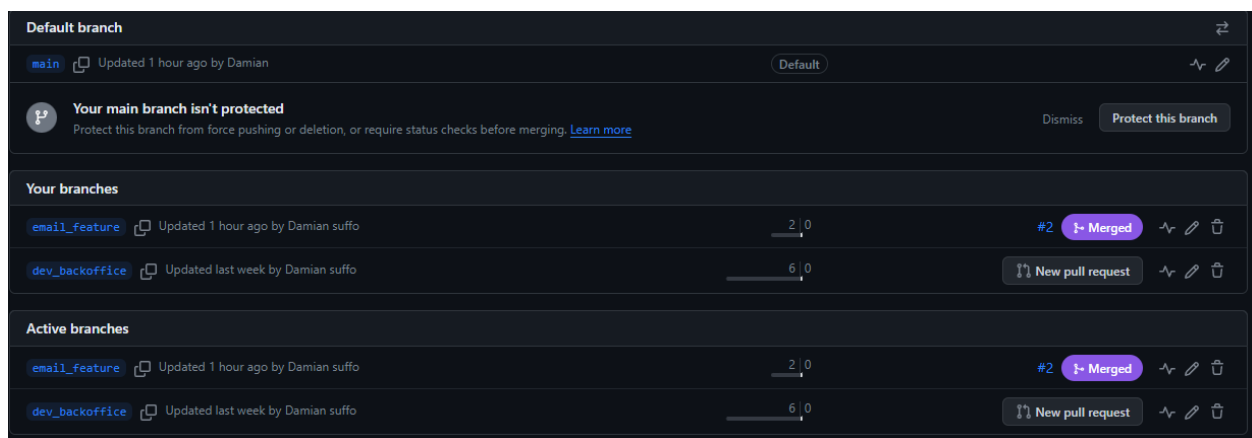
La API de viajes proporciona servicios para acceder a los datos de viajes. La aplicación utiliza la API de viajes para realizar operaciones de seguimiento, como consultar el estado de un pedido.

Uso de Git y Github:

Se utilizó git como software de control de versión y github para alojarlas.

En cada proyecto se crearon las ramas principales (main) rama de desarrollo(development) y cada feature o módulo nuevo con su respectiva rama o branch.


BackOffice:



Aplicación de almacén:

Your branches				
map-feature	Updated last week by damian	2 0	New pull request	🔖 ✎ 🗑
api-communication	Updated last week by damian	2 0	#2 Merged	🔖 ✎ 🗑
development	Updated last month by damian	17 0	New pull request	🔖 ✎ 🗑
storehouseForm	Updated last month by damian	17 0	New pull request	🔖 ✎ 🗑
visual-enhancements	Updated last month by damian	17 0	New pull request	🔖 ✎ 🗑
View more of your branches >				
Active branches				
map-feature	Updated last week by damian	2 0	New pull request	🔖 ✎ 🗑
api-communication	Updated last week by damian	2 0	#2 Merged	🔖 ✎ 🗑
visual-enhancements	Updated last month by damian	17 0	New pull request	🔖 ✎ 🗑
storehouseForm	Updated last month by damian	17 0	New pull request	🔖 ✎ 🗑
development	Updated last month by damian	17 0	New pull request	🔖 ✎ 🗑

Aplicación de camionero:

Default branch				
main	Updated last month by damian	Default		🔖 ✎
<div>  Your main branch isn't protected Protect this branch from force pushing or deletion, or require status checks before merging. Learn more <div>Dismiss</div> <div>Protect this branch</div> </div>				
Your branches				
truckerForms	Updated 3 weeks ago by damian	0 27	New pull request	🔖 ✎ 🗑
translate_feature	Updated last month by damian	0 22	New pull request	🔖 ✎ 🗑
development	Updated last month by damian	0 2	New pull request	🔖 ✎ 🗑
loginForm	Updated last month by damian	0 1	New pull request	🔖 ✎ 🗑
Active branches				
truckerForms	Updated 3 weeks ago by damian	0 27	New pull request	🔖 ✎ 🗑
translate_feature	Updated last month by damian	0 22	New pull request	🔖 ✎ 🗑
development	Updated last month by damian	0 2	New pull request	🔖 ✎ 🗑
loginForm	Updated last month by damian	0 1	New pull request	🔖 ✎ 🗑

Aplicación de Seguimiento:

Default branch

`main` Updated 5 days ago by Damian Default

Your main branch isn't protected
Protect this branch from force pushing or deletion, or require status checks before merging. [Learn more](#) Dismiss Protect this branch

Your branches

<code>MyOrders_feature</code> Updated 4 days ago by damian	0 12	New pull request		
<code>development</code> Updated 5 days ago by Damian	0 5	New pull request		
<code>main_form</code> Updated 5 days ago by damian	0 4	#1 Merged		

Active branches

<code>MyOrders_feature</code> Updated 4 days ago by damian	0 12	New pull request		
<code>development</code> Updated 5 days ago by Damian	0 5	New pull request		
<code>main_form</code> Updated 5 days ago by damian	0 4	#1 Merged		

Api almacén:

Default branch

`main` Updated 4 months ago by Agus Client Default

Your branches

<code>api_almacen_development</code> Updated last week by damian suffo	0 56	New pull request		
<code>development</code> Updated 4 months ago by Agus Client	0 0	New pull request		

Active branches

<code>api_almacen_development</code> Updated last week by damian suffo	0 56	New pull request		
--	--------	------------------	--	--

Stale branches

<code>development</code> Updated 4 months ago by Agus Client	0 0	New pull request		
--	-------	------------------	--	--

Api seguimiento:

Default branch

main

Updated last month by damian

Default

Your main branch isn't protected

Protect this branch from force pushing or deletion, or require status checks before merging. [Learn more](#)

Dismiss

Protect this branch

Your branches

development	Updated last month by damian	0 12	New pull request			
controllers~feature	Updated last month by damian	0 11	New pull request			
models~features	Updated last month by damian	0 5	New pull request			

Active branches

development	Updated last month by damian	0 12	New pull request			
controllers~feature	Updated last month by damian	0 11	New pull request			
models~features	Updated last month by damian	0 5	New pull request			

Api de autenticación:

Default branch

main

Updated last month by Damian Suffo

Default

Your main branch isn't protected

Protect this branch from force pushing or deletion, or require status checks before merging. [Learn more](#)

Dismiss

Protect this branch

Your branches

refactorization	Updated 2 weeks ago by Damian Suffo	0 1	New pull request			
development	Updated last month by Damian Suffo	0 0	New pull request			

Active branches

refactorization	Updated 2 weeks ago by Damian Suffo	0 1	New pull request			
development	Updated last month by Damian Suffo	0 0	New pull request			