

PROYECTO:

# DESARROLLO DE UN SISTEMA WEB ECOMMERCE CON FUNCIONALIDADES BÁSICAS, UTILIZANDO TECNOLOGÍAS LIBRES BASADAS C#, .NET CORE 8.0, FRONTEND ANGULAR 17.

Autor:

Ing. Francisco Higuera

MAYO 2024

# INDICE

Contenido

[DESARROLLO DE UN SISTEMA WEB ECOMMERCE CON FUNCIONALIDADES BÁSICAS, UTILIZANDO TECNOLOGÍAS LIBRES BASADAS C#, .NET CORE 8.0, FRONTEND ANGULAR 17. 1](#_Toc166326677)

[INDICE 2](#_Toc166326678)

[INTRODUCCIÓN 3](#_Toc166326679)

[OBJETIVOS: 4](#_Toc166326680)

[Objetivo General: 4](#_Toc166326681)

[Objetivos Específicos: 4](#_Toc166326682)

[REQUERIMIENTOS: 4](#_Toc166326683)

[Requerimientos funcionales: 4](#_Toc166326684)

[Extra: 5](#_Toc166326685)

[Requerimientos técnicos: 5](#_Toc166326686)

[Pude elegir una de las dos opciones 5](#_Toc166326687)

[1. FUNDAMENTACIÓN TEÓRICA 5](#_Toc166326688)

[1.1 Marco Teórico 5](#_Toc166326689)

[1.1.1 Desarrollo web 5](#_Toc166326690)

[1.1.2 Angular 17 6](#_Toc166326691)

[1.1.3 SQL Server 6](#_Toc166326692)

[1.1.4 TypeScript 6](#_Toc166326693)

[1.1.5 Requerimientos funcionales 7](#_Toc166326694)

[1.1.5.1 Ingresar/Almacenar/Gestionar: 7](#_Toc166326695)

[1.1.5.2 Generar/Proporcionar: 7](#_Toc166326696)

[1.1.6 Requerimientos no funcionales: 7](#_Toc166326697)

[1.2.- Definición de términos básicos: 8](#_Toc166326698)

[2. HERRAMIENTAS Y MÉTODOS UTILIZADOS EN EL SISTEMA 10](#_Toc166326699)

[CONCLUSIONES 15](#_Toc166326700)

[WEBGRAFÍA. 16](#_Toc166326701)

# INTRODUCCIÓN

Este proyecto tiene como propósito el diseño y desarrollo de un sistema web encargado gestionar el almacenamiento de una entidad llamada Sistema Ecommerce, para la empresa Viamatica.

En el primer apartado inicialmente se presentarán los objetivos trazados de manera general y específicos, seguidos de los requerimientos establecidos por la empresa Viamatica para el desarrollo del sistema planteado.

En este orden de ideas, en el punto número uno se describe la fundamentación teórica del proyecto, desglosado en sustentos teóricos como lo son las definiciones de términos básicos utilizados y que a su vez facilitaran el desarrollo del proyecto para cumplir el objetivo planteado.

Seguidamente se presenta en el segundo punto las herramientas y métodos utilizados para el desarrollo del sistema, donde se incluye el software seleccionado, el desarrollo tanto de backend como de frontend y el despliegue al servidor de prueba, detallando así paso a paso la ejecución del sistema.

En este sentido, en el presente proyecto se describen los fundamentos y metodologías de desarrollo, plataformas, servidor, lenguaje de programación y desarrollo del lado del servidor y cliente en ambiente Web, así como la bases de datos relacionales y sistemas de gestión de bases de datos utilizados como herramientas, aunado a las opciones y tecnologías, todo ello con el propósito de presentar una mejor reseña sobre el proceso del sistema desarrollado.

# OBJETIVOS:

### Objetivo General:

Desarrollar un sistema web con funcionalidades básicas, utilizando tecnologías libres basadas en C#, .Net Core 8.0, Frontend Angular 17.

### Objetivos Específicos:

* Analizar los requerimientos y herramientas para el desarrollo del sistema.
* Desarrollar el backend.
* Desarrollar el frontend.

# REQUERIMIENTOS:

En este apartado se muestran los requerimientos establecidos por la empresa Viamatica para el desarrollo del Sistema web con funciones básicas.

Se solicita que un usuario pueda crear una cuenta de usuario y realizar compras vía web a través de la aplicación.

### Requerimientos funcionales:

* Diseñar un API Restful (CRUD completo).

### Extra:

* Agregar autenticación al API con JWT
* Hacer pruebas unitarias.

### Requerimientos técnicos:

* Base de datos con sus respectivas tablas
* Un login y registro de usuarios (clientes)
* Un listado de categorías que a su vez contengan productos
* Se pueda seleccionar el producto para añadirlo al carrito
* Opción de Compra, lo del carrito generar una factura
* Se debe hacer la BD, el back (Net Core), y el front (Angular), adicional tomar en cuenta los puntos que se calificaran
* Funcionamiento
* Flujo correcto
* Código limpio (estructurado, parametrizaciones)
* Uso correcto de componentes

### Pude elegir una de las dos opciones

1. Puede crear una carpeta en POSTMAN para pegarle a todos los endpoints creados y luego compartirla conmigo.
2. Puede hacer un HTML en Vue sencillo para hacer los formularios de creación de usuario, de login, listar e interactuar con los SISTEMA ECOMMERCEs, listar los usuarios si el ROLE se lo permite.

# FUNDAMENTACIÓN TEÓRICA

# Marco Teórico

### Desarrollo web

El desarrollo de Aplicaciones Web ha ido creciendo a lo largo de los años superando la inclinación al desarrollo de aplicaciones de escritorio. Uno de los motivos principales es la facilidad de acceso a una aplicación web combinada con la simplicidad de actualización de esta. El desarrollo web está basado en una arquitectura cliente-servidor, en la cual el cliente es el que empieza la comunicación mediante un navegador web realizando peticiones hacia el servidor. El servidor es el encargado de responder a dichas peticiones del cliente para procesar la información recibida y mostrar las páginas solicitadas

### Angular 17

Angular es una plataforma de desarrollo para la creación de aplicaciones web de una sola página. Está escrito en TypeScript y es conocido por su enfoque en componentes, lo que permite a los desarrolladores dividir su interfaz de usuario en partes manejables y reutilizables. Angular también ofrece características como enlace de datos bidireccional e inyección de dependencias, lo que reduce la cantidad de código que los desarrolladores tienen que escribir y ayuda a evitar errores. Además, Angular tiene una interfaz de línea de comandos potente que facilita la creación y el despliegue de aplicaciones, así como herramientas para formularios, enrutamiento, internacionalización y accesibilidad. También es compatible con varias bibliotecas de componentes para ayudar a los desarrolladores a crear interfaces de usuario ricas y atractivas.

### SQL Server

SQL Server es un sistema gestor de base de datos relacionales (SGBD) desarrollado por Microsoft1. Se caracteriza por su rendimiento líder en el sector, seguridad y alta disponibilidad2. SQL Server utiliza el lenguaje de base de datos Transact-SQL, que amplía el estándar del lenguaje básico SQL, aportándole nuevas y más potentes características como el uso de Triggers y procedimientos almacenados1. Además, ofrece capacidades avanzadas para el almacenamiento, la gestión y el análisis de datos, permitiendo a los usuarios crear bases de datos, tablas y consultas para recuperar información específica de un gran conjunto de datos.

### TypeScript

TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft1. Es un superconjunto de JavaScript, lo que significa que todos los programas JavaScript existentes son programas TypeScript válidos2. TypeScript está diseñado para facilitar a los desarrolladores la escritura de aplicaciones a gran escala, proporcionando una forma eficiente y eficaz para escribir código2. Ofrece características como la comprobación estática opcional de tipos, clases y módulos, que ayudan a los desarrolladores a escribir código limpio y fácil de mantener2. Gracias a su sintaxis intuitiva y a una codificación familiar al estilo JavaScript, TypeScript es una gran opción para los desarrolladores que desean crear aplicaciones robustas.

### Requerimientos funcionales

Un sistema gestor de información debe poder contar con los siguientes requerimientos funcionales:

### Ingresar/Almacenar/Gestionar:

Información de usuarios, categorías, productos y órdenes de compra.

### Generar/Proporcionar:

Consultas de documentación: técnica, contractual, de equipos, planos, manuales, catálogos, facturas, procedimientos, hojas de datos y otras en caso de requerirse.

### Requerimientos no funcionales:

Un sistema debe poder contar con los siguientes requerimientos no funcionales:

* + - * La Interfaz dinámica, simple y fácil de usar.
      * Debe Administrar y asignar permisos para diferentes tipos de usuarios.
      * Debe ser rápido y eficiente.
      * Debe ser multi-usuario.
      * La ejecución del sistema gestor de información podrá realizarse a nivel local (PC), interno (INTRANET) y externo (INTERNET).
      * Personalización de consultas de acuerdo a los requerimientos del usuario surgidas durante el proceso de desarrollo del proyecto.
      * El sistema debe contar con las seguridades respectivas y restricciones de acceso a usuarios no autorizados de modo que tenga integridad y confiabilidad de los datos.
        + Respaldo periódico y automatizado de datos
        + Un sistema debe posibilitar la verificación de un historial de edición de información.
        + Debe tener un manual de gestión/edición de información del sistema.
        + Debe garantizar que la información ingresada sea correcta y coherente

# 1.2.- Definición de términos básicos:

**Api Rest:** Es una interfaz de programación de aplicaciones (API o API web) que se ajusta a los límites de la arquitectura REST y permite la interacción con los servicios web de RESTful. El informático Roy Fielding es el creador de la transferencia de estado representacional (REST).

**Bases de datos relacionales:** Una base de datos relacional es una recopilación de elementos de datos con relaciones predefinidas entre ellos. Estos elementos se organizan como un conjunto de tablas con columnas y filas. Las tablas se utilizan para guardar información sobre los objetos que se van a representar en la base de datos.

**Bases de datos no relacionales:** Son un sistema de almacenamiento de información que se caracteriza por no usar el lenguaje SQL para las consultas. Esto no significa que no puedan usar el lenguaje SQL, pero no lo hacen como herramienta de consulta, sino como apoyo. Por ello también se les suele llamar NoSQL o «no solo SQL».

Otra de sus principales características es que no trabajan con estructuras definidas. Es decir, los datos no se almacenan en tablas**,** y la información tampoco se organiza en registros o campos.

**Frameworks:** Es un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida para la organización y desarrollo de software.

**Javascript:** Es un lenguaje de programación ligero, interpretado, o compilado justo-a- tiempo (just-in-time) con funciones de primera clase. Si bien es más conocido como un lenguaje de scripting (secuencias de comandos) para páginas web, y es usado en muchos entornos fuera del navegador, tal como Node.js, Apache CouchDB y Adobe Acrobat JavaScript es un lenguaje de programación basada en prototipos,

multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo programación funcional).

**ORM:** Un ORM es un modelo de programación que permite mapear las estructuras de una base de datos relacional (SQL Server, Oracle, MySQL, etc.), en adelante RDBMS (Relational Database Management System), sobre una estructura lógica de entidades con el objeto de simplificar y acelerar el desarrollo de nuestras aplicaciones.

**SQL:** SQL o lenguaje de consulta estructurada es la interfaz principal utilizada para comunicarse con bases de datos relacionales. SQL se convirtió en un estándar del Instituto Nacional Estadounidense de Estándares (ANSI) en 1986. Todos los motores de bases de datos relacionales populares admiten el SQL de ANSI estándar. Algunos de estos motores también tienen extensión al SQL de ANSI para admitir funcionalidades específicas de ese motor. SQL se utiliza para agregar, actualizar o eliminar filas de datos, por lo que se recuperan subconjuntos de datos para aplicaciones de análisis y procesamiento de transacciones, y para administrar todos los aspectos de la base de datos.

## HERRAMIENTAS Y MÉTODOS UTILIZADOS EN EL SISTEMA

#### Software seleccionado

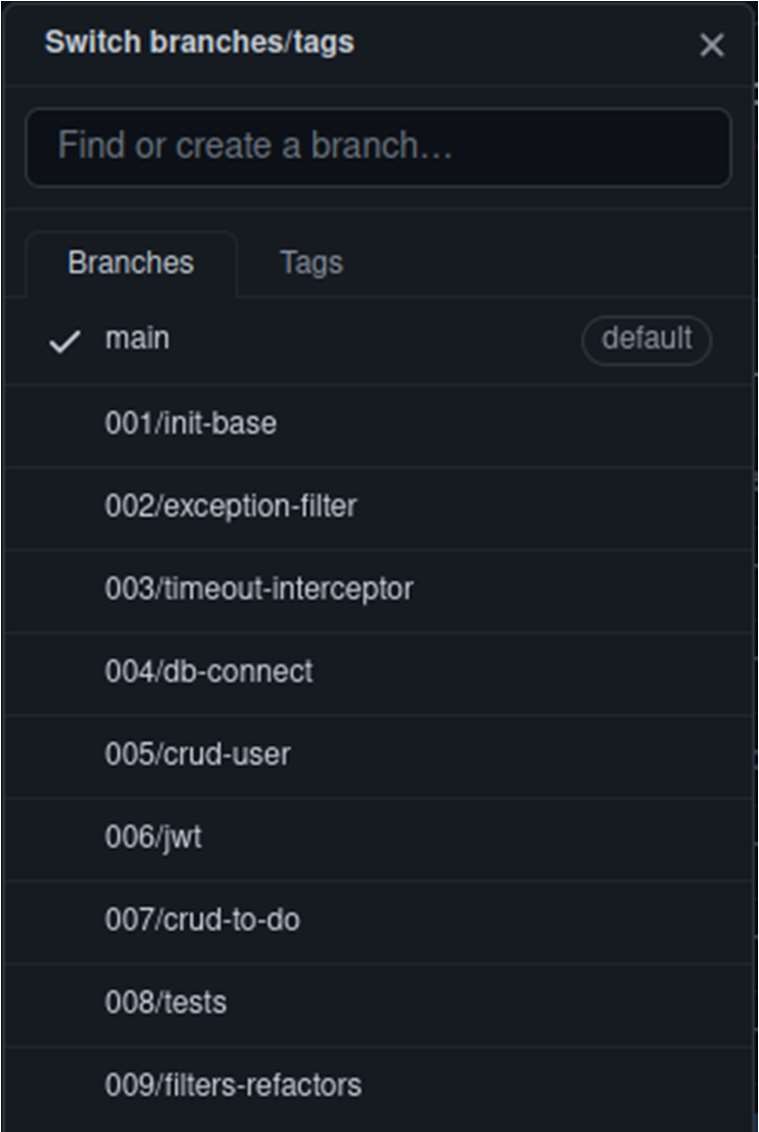
Para el desarrollo del backend de la aplicación Sistema Ecommerce se seleccionó como lenguaje de programación C# versión 16.13.0, framework .NET CORE y como servidor de base de datos sql server.

En cuanto al frontend se utilizó Angular versión 17, y para el versionado tanto para el frontend como para el backend se utilizó git.

#### Desarrollo del Backend

* + 1. **Actividades**

Para el desarrollo del backend se realizó por una serie de tareas, cuya evolución se registró en el git a través de ramas, el repositorio es: https://github.com/higueraf/Sistema Ecommerce- nestjs.



#### - Base inicial:

Se instaló primeramente nestjs-cli con el comando:

# npm i -g @nestjs/cli

Seguido, se creó el proyecto con el andamiaje por defecto que distribuye nestjs:

# nest new project-name

#### - Implementación Excepciones Globales:

Se incluye la funcionalidad de manejar las excepciones de manera global, definiendo el filtro en una carpeta dentro del proyecto src/common/filters en el archivo http- exception.filter.ts, y luego se agrega de manera global en el archivo main.ts. Esto permite que la aplicación no se interrumpa por errores e informe el error en mensajes a través de la respuesta según la petición.

#### Implementación de time-out interceptor:

Este interceptor permite evaluar si la respuesta de una petición está tardando mucho tiempo, devolviendo mensaje de error en la respuesta según el tiempo establecido. Esto se estableció en un archivo en la carpeta /src/common/interceptors en el archivo timeout.interceptor.ts., al igual que las excepciones globales, se incluyó el llamado al interceptor en el archivo main.ts.

#### Conexión a bases de datos:

Para conectar e interactuar la aplicación con la base de datos postgresql se utilizó el ORM TypeORM 0.3.7 y la librería pg 8.7.3, para dicha conexión se incluyó el módulo del ORM en el archivo de src/app.modulo.ts en la sección imports:

TypeOrmModule.forRoot({ type: 'postgres',

host: process.env.DB\_HOST,

port: parseInt(process.env.DB\_PORT, 10), username: process.env.DB\_USER, password: process.env.DB\_PASSWORD,

database: process.env.DB\_DB,

entities: [ dirname + '/../api/\*\*/\*.entity{.ts,.js}'], autoLoadEntities: true,

synchronize: true,

}),

Cabe destacar que esta tarea se incluyó la lectura del archivo de variables de ambiente “.env”, con la librería dotenv 16.0.1.

#### CRUD de Usuarios:

Se creó una carpeta /src/api/users para incluir los archivos referentes al CRUD de usuarios que incluye: user.entity.ts donde se declara la entidad de la base de datos según el ORM, users.controller.ts donde se distribuye la lógica y se define los endpoints, users.module.ts donde se declara el módulo de usuario y se importan otros módulos que se utilizan en el CRUD, el servicio users.service.ts donde se encuentra la lógica de interacción con la base datos, y por último en la carpeta DTO se encuentra dos archivos de objetos para incluir usuarios y para responder en el login.

#### Implementación JWT

Se incluye una carpeta /src/api/auth/ donde se colocaron los archivos referentes al control acceso de usuarios y generación de JWT.

#### CRUD Sistema Ecommerce:

Se creó una carpeta /src/api/Sistema Ecommerce para incluir los archivos referentes al CRUD de To- Do que incluye: Sistema Ecommerce.entity.ts donde se declara la entidad de la base de datos según el ORM, Sistema Ecommerce.controller.ts donde se distribuye la lógica y se define los endpoints, users.module.ts donde se declara el módulo de usuario y se importan otros módulos que se utilizan en el CRUD, el servicio users.service.ts donde se encuentra la lógica de interacción con la base datos, y por último en la carpeta DTO se encuentra un archivo de objetos para incluir Sistema Ecommerce.

#### Tests Unitarios:

Los tests permiten evaluar el funcionamiento de los procesos tanto de los CRUD de los

usuarios y Sistema Ecommerce, así como de la autenticación, y se realizan a través de los archivos:

/src/api/users/users.controller.spec.ts

/src/api/users/users.service.spec.ts

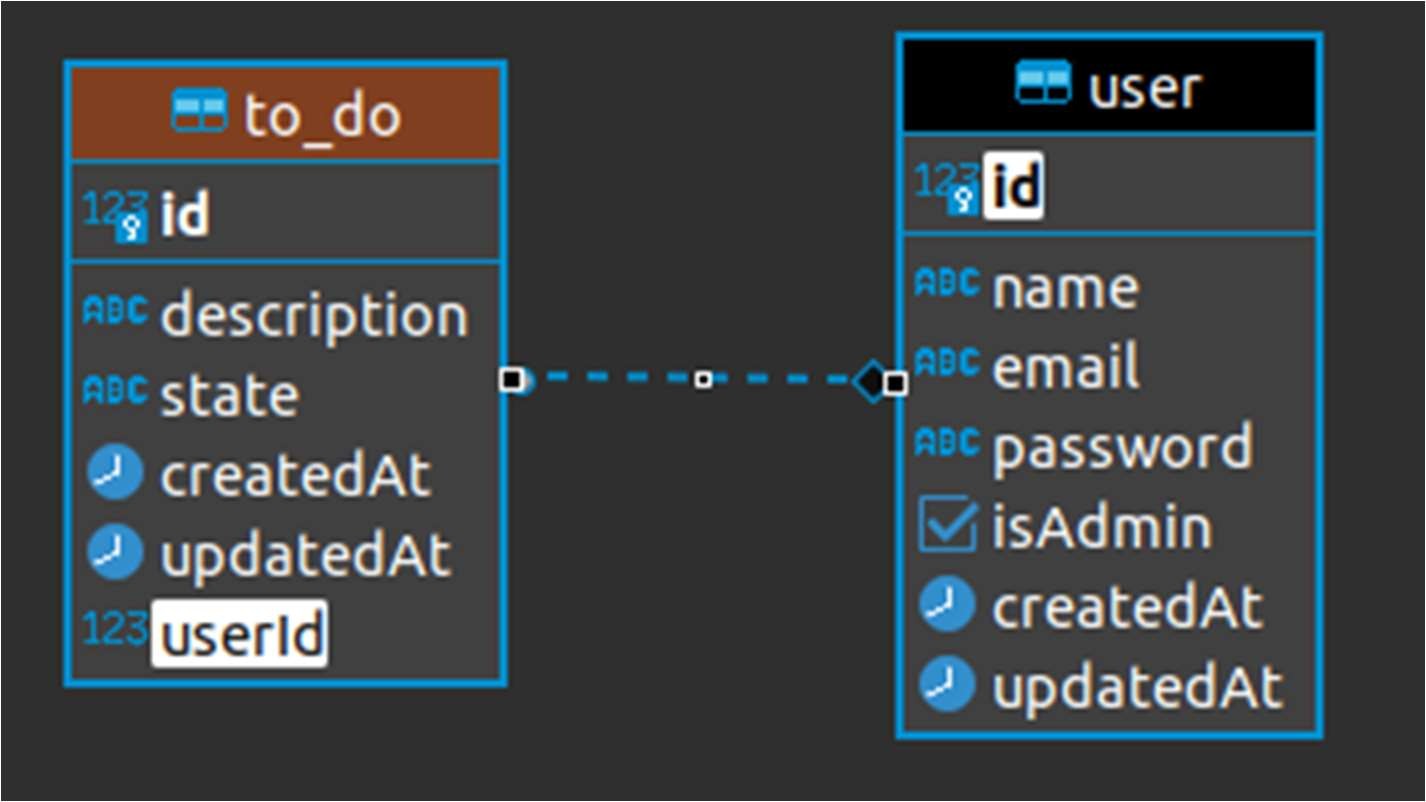
/src/api/Sistema Ecommerce/Sistema Ecommerce.controller.spec.ts

/src/api/Sistema Ecommerce/Sistema Ecommerce.service.spec.ts

#### Filtros y refactorización:

En este apartado se incluyen filtros y ajustes a códigos.

#### Modelo de Base de Datos:



* + 1. **Endpoints:**

Control de acceso o login

curl --location --request POST '[http://Sistema Ecommerce-](http://to-do-/) vuejs.franciscohiguera.me/api/v1/auth/login' \

--header 'Content-Type: application/json' \

--data-raw '{

"email": "[higueraf@Sistema Ecommerce.com](mailto:higueraf@to-do.com)", "password": "12345"

}'

Registro de usuario

curl --location --request POST '[http://Sistema Ecommerce-](http://to-do-/) vuejs.franciscohiguera.me/api/v1/auth/register' \

--header 'Content-Type: application/json' \

--data-raw '{ "name":"[higueraf@Sistema Ecommerce.com](mailto:higueraf@to-do.com)", "email": "[higueraf@Sistema Ecommerce.com](mailto:higueraf@to-do.com)", "password": "12345", "isAdmin": true

}'

CRUD de Usuarios:

Listar usuarios

curl --location --request GET '[http://Sistema Ecommerce-](http://to-do-/) vuejs.franciscohiguera.me/api/v1/users' \

--header 'Authorization: Bearer TOKEN'

Consultar un usuario

curl --location --request GET '[http://Sistema Ecommerce-](http://to-do-/) vuejs.franciscohiguera.me/api/v1/users/2' \

--header 'Authorization: Bearer TOKEN'

Crear usuario

curl --location --request POST '[http://Sistema Ecommerce-](http://to-do-/) vuejs.franciscohiguera.me/api/v1/users' \

--header 'Authorization: Bearer TOKEN' \

--header 'Content-Type: application/json' \

--data-raw '{

"name":"higueraf",

"email": "[higueraf@Sistema Ecommerce.com](mailto:higueraf@to-do.com)", "password": "12345", "isAdmin": true

}'

Actualizar Usuario:

curl --location --request PUT '[http://Sistema Ecommerce-](http://to-do-/) vuejs.franciscohiguera.me/api/v1/users/3' \

--header 'Authorization: Bearer TOKEN' \

--header 'Content-Type: application/json' \

--data-raw '{ "name":"[adminmodidddficado@admin.com](mailto:adminmodidddficado@admin.com)", "email": "[admin@admin.com](mailto:admin@admin.com)",

"password": "12345", "isAdmin": false

}'

Eliminar Usuario:

curl --location --request DELETE '[http://Sistema Ecommerce-](http://to-do-/) vuejs.franciscohiguera.me/api/v1/users/3' \

--header 'Authorization: Bearer TOKEN' \

--data-raw '' CRUD Sistema Ecommerce:

Listar Sistema Ecommerce

curl --location --request GET '[http://Sistema Ecommerce-vuejs.franciscohiguera.me/api/v1/to-](http://to-do-vuejs.franciscohiguera.me/api/v1/to-) do' \

--header 'Authorization: Bearer TOKEN'

Consultar un Sistema Ecommerce

curl --location --request GET '[http://Sistema Ecommerce-vuejs.franciscohiguera.me/api/Sistema Ecommerce/5'](http://to-do-vuejs.franciscohiguera.me/api/to-do/5%27)

\

--header 'Authorization: Bearer TOKEN'

Crear un Sistema Ecommerce

curl --location --request POST '[http://Sistema Ecommerce-vuejs.franciscohiguera.me/api/v1/to-](http://to-do-vuejs.franciscohiguera.me/api/v1/to-) do' \

--header 'Authorization: Bearer TOKEN' \

--header 'Content-Type: application/json' \

--data-raw '{

"description":"description Sistema Ecommerce example description", "state": "ACTIVE"

}'

Actualizar un Sistema Ecommerce

curl --location --request PUT '[http://Sistema Ecommerce-vuejs.franciscohiguera.me/api/v1/to-](http://to-do-vuejs.franciscohiguera.me/api/v1/to-) do/2' \

--header 'Authorization: Bearer TOKEN' \

--header 'Content-Type: application/json' \

--data-raw '{

"description":"description Sistema Ecommerce example description modify", "state": "ACTIVE"

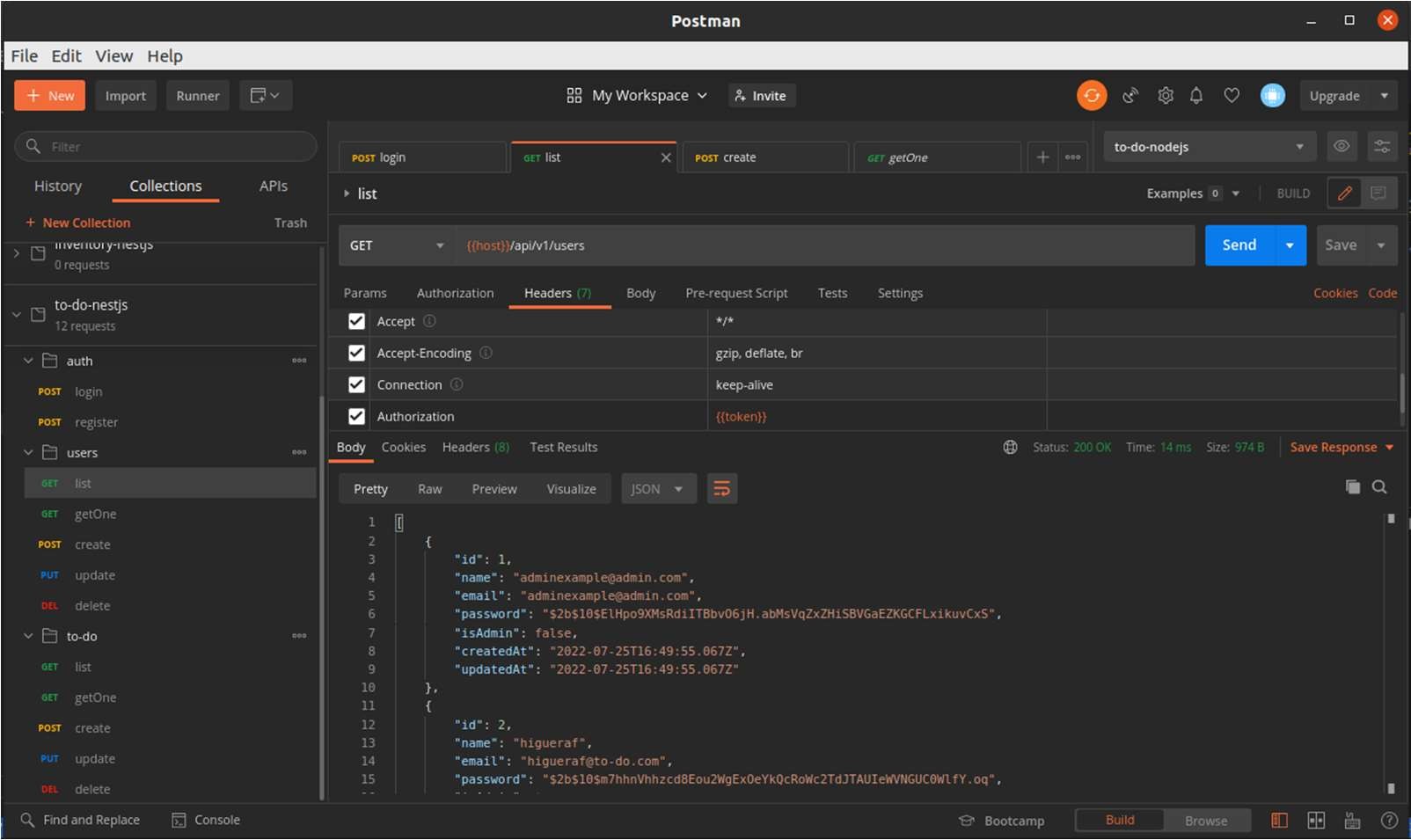
}'

Borrar un Sistema Ecommerce

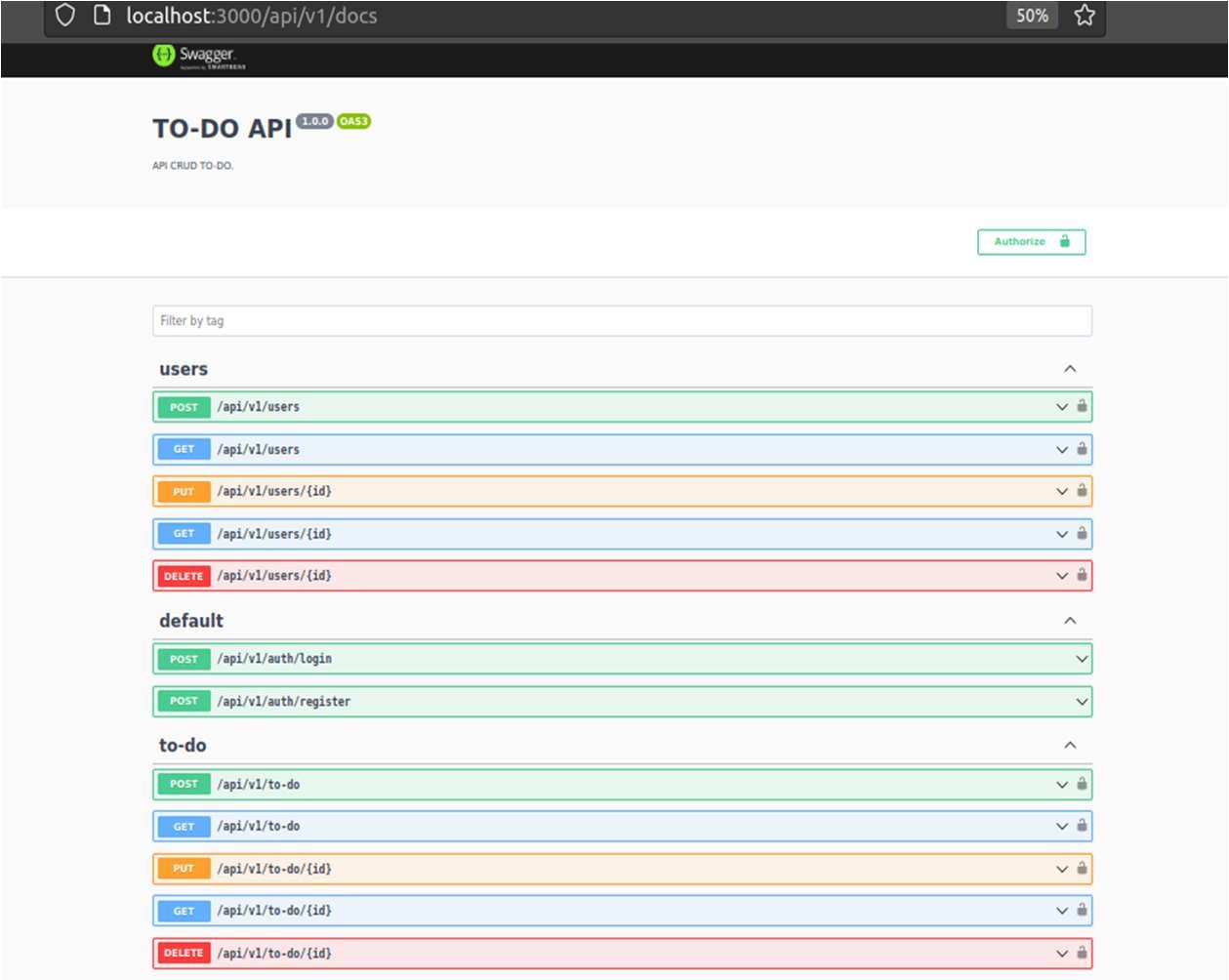
curl --location --request DELETE '[http://Sistema Ecommerce-](http://to-do-/) vuejs.franciscohiguera.me/api/v1/Sistema Ecommerce/8' \

--header 'Authorization: Bearer TOKEN' \

--data-raw ''

Documentación Postman

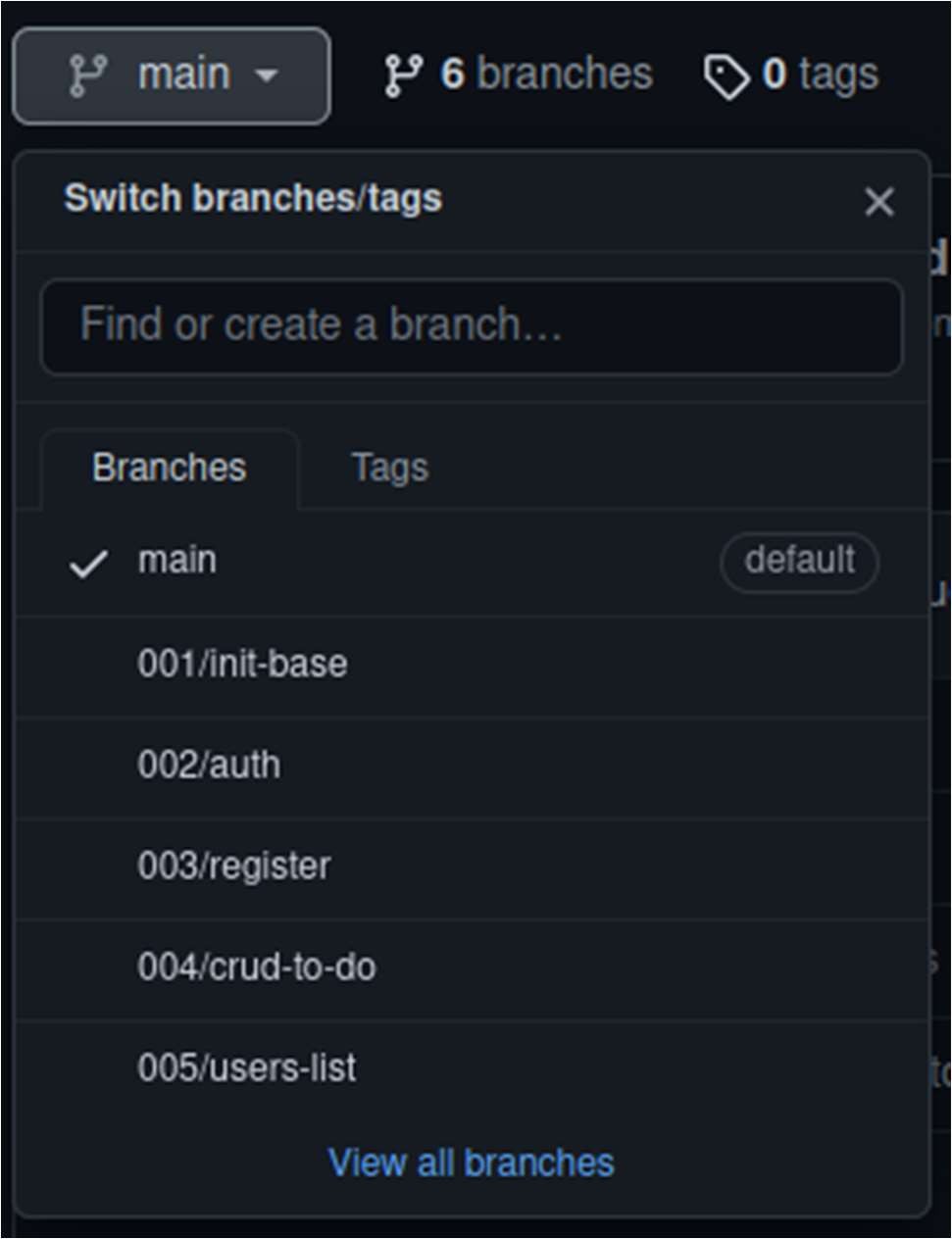
Documentación de endpoints a través de swagger



#### Desarrollo del Frontend

**2.3.1. Actividades:**

Para el desarrollo del frontend se realizó igualmente por una serie de tareas, cuya evolución se registró en el git a través de ramas, el repositorio es: https://github.com/higueraf/Sistema Ecommerce-vuejs.



#### Base inicial:

Se creó el proyecto con el andamiaje por defecto que distribuye nestjs: # npm init vue@latest

Inicia solicitando los datos del proyecto y preferencias:

Project name: … Sistema Ecommerce-vuej

✔ Add TypeScript? … No / Yes

✔ Add JSX Support? … No / Yes

✔ Add Vue Router for Single Page Application development? … No / Yes

✔ Add Pinia for state management? … No / Yes

✔ Add Vitest for Unit testing? … No / Yes

✔ Add Cypress for both Unit and End-to-End testing? … No / Yes

✔ Add ESLint for code quality? … No / Yes

✔ Add Prettier for code formatting? … No / Yes

Scaffolding project in ./<your-project-name>... Done.

#### Autorización:

Se realizó maquetación del formulario de login utilizando las librerías material-design- icons 3.0.1, materialize-css 1.0.0-rc.2, también se incluyó la lógica de conexión contra la api a través de la librería axios 0.27.2

#### Registro de usuario:

Maquetación del formulario de registro de usuario y se incluyó la lógica de conexión hacia la api contra el endpoint de registro de usuario.

#### Crud Sistema Ecommerce:

Maquetación de la pantalla de lista de Sistema Ecommerce, se incluyeron modales para el registro, modificación y eliminación de Sistema Ecommerce, también se incluyó la lógica de conexión hacia la api contra los endpoints CRUD de la aplicación Sistema Ecommerce-nestjs.

#### Lista de Usuarios:

Maquetación de la pantalla de lista de usuarios, que se muestra solo a los usuarios administradores, a la vez incluyó la lógica de conexión contra la api contra el endpoint de listar los usuarios.

Pantalla de Inicio-login:

Registro de Usuario:

Home o Pantalla principal

Lista de Sistema Ecommerce

Modal Incluir Sistema Ecommerce

Modal para Modificar Sistema Ecommerce

Lista de Usuarios

# CONCLUSIONES

Durante el desarrollo de la aplicación, determinar los objetivos del sistema es una norma básica en un proyecto, estos objetivos permiten establecer los alcances máximos y evitan que aparezcan nuevos requerimientos que no estén contemplados desde un inicio.

Las herramientas y tecnologías de uso libre (software libre) utilizadas en este proyecto de desarrollo facilitó la solución a los problemas que se presentaron durante la etapa del desarrollo, esto se debe a que podemos encontrar una gran cantidad de información debidamente documentadas por los expertos de las mismas, permitiendo compartir la experiencia a través de un intercambio de información que enriquece más aun nuestra base de conocimientos adquiridos.

El desarrollo e implementación del aplicativo Web representa para cualquier empresa una herramienta solida tanto de control, ordenamiento, almacenamiento y consulta de archivos, lo que permite tener mejor control de calidad en la documentación recabada de un determinado proyecto.

El uso de aplicativos Web permite agilizar la preparación y elaboración de un producto final siempre y cuando se tenga clara la estructura del contenido y se empiecen a receptar los archivos técnicos digitales finales. Gracias al análisis y desarrollo del sistema Ecommerce y a las pruebas realizadas, se logró verificar el cumplimiento de los requerimientos.

La combinación y el aprovechamiento de las características de los frameworks, Vue.js y nestjs utilizados en el front-end y en el back-end respectivamente, ha sido de gran ayuda para lograr implantar un sistema web rápido, brindando una buena experiencia de usuario.

# WEBGRAFÍA.

Camacho, S. 2015. Desarrollo de una plataforma web para el sistemade gestión de la información de proyectos de fiscalización realizados por la empresa tecnie, accesible local y remotamente.

chrome- extension://efaidnbmnnnibpcajpcglclefindmkaj/https://bibdigital.epn.edu.ec/bitstrea m/15000/10786/1/CD-6318.pdf

Maldonado, J. 2016. Desarrollo e implementación de un sistema web de seguimiento y evaluación de las prácticas pre-profesionales para la facultad de ingeniería escuela civil de la PUCE**.**

<http://repositorio.puce.edu.ec/handle/22000/12562>

Peña, A. 2022. Sistema web aplicando vue.js y laravel para la gestión de comercialización de productos en la farmacia FARMARED'S N° 257 https://repositorio.uta.edu.ec/handle/123456789/34832

https://medium.com/swlh/deploy-nest-js-app-with-postgres-in-vps-e1ce4abd2cad

https://thucnc.medium.com/deploy-a-vuejs-web-app-with-nginx-on-ubuntu-18-04- f93860219030

https://[www.appsyoda.com/blog/deploying-vuejs-app-using-nginx/](http://www.appsyoda.com/blog/deploying-vuejs-app-using-nginx/) https://[www.redhat.com/es/topics/api/what-is-a-rest-api](http://www.redhat.com/es/topics/api/what-is-a-rest-api)

https://aws.amazon.com/es/relational- database/#:~:text=Una%20base%20de%20datos%20relacional%20es%20una%2 0recopilaci%C3%B3n%20de%20elementos,en%20la%20base%20de%20datos.

https://ayudaleyprotecciondatos.es/bases-de-datos/no-relacional/

https://[www.edix.com/es/instituto/framework/#:~:text=Un%20framework%20es%20](http://www.edix.com/es/instituto/framework/#%3A~%3Atext%3DUn%20framework%20es%20) un%20esquema,organizaci%C3%B3n%20y%20desarrollo%20de%20software.

https://developer.mozilla.org/es/docs/Web/JavaScript

https://ualmtorres.github.io/SeminarioNestJS/#:~:text=NestJS%20es%20un%20fra mework%20para,aplicaciones%20m%C3%A1s%20f%C3%A1ciles%20de%20mant ener.

https://es.wikipedia.org/wiki/PostgreSQL

https://snipcart.com/blog/learn-vanilla-javascript-before-using-js-frameworks https://platzi.com/contributions/que-es-vanilla-js-o-javascript-puro/ <https://www2.deloitte.com/es/es/pages/technology/articles/que-es-orm.html>

<https://www.step2gen.com/blogs/key-highlights-of-angular-17-what-you-need-to-know>

https://sqlearning.com/es/introduccion-sql-server/introduccion-sql-server/