Documentación Proyecto Veterinaria

Juan Pablo Adams Parra
Isac Cortes Buitrago
Emmanuel Bolivar Marin
Juan Jose Estrada Velez

Politécnico Jaime Isaza Cadavid.

Ingeniería Informática
Taller De Lenguajes De Programación

Tabla de contenido

Introducción

En el mundo contemporáneo, la integración de la tecnología desempeña un papel esencial en la optimización y eficiencia de diversos sectores, y la atención veterinaria no es la excepción. En este contexto, se presenta un proyecto web innovador diseñado para revolucionar la gestión interna de una clínica veterinaria, utilizando tecnologías de vanguardia como Angular, Node.js y MySQL, con la aplicación del sólido patrón de arquitectura Modelo Vista Controlador (MVC). Este proyecto no solo busca modernizar los procesos administrativos de la veterinaria, sino también mejorar la calidad de atención al cliente y la eficacia en el cuidado de las mascotas.

El enfoque principal del sistema se basa en la gestión integral de roles, adaptando la plataforma a las necesidades específicas de distintos actores dentro de la clínica. Desde veterinarios que requieren acceso a historias clínicas y datos médicos, hasta vendedores encargados de gestionar productos y facturas, el sistema asigna responsabilidades de manera eficiente. La inclusión de roles como administrador y dueño permite una supervisión completa del sistema y una gestión estratégica de la información.

Además de la asignación de roles, el proyecto aborda la administración de datos cruciales para el funcionamiento de la veterinaria. Desde la gestión de personas, incluyendo clientes y personal interno, hasta el seguimiento detallado de mascotas y sus historias clínicas, el sistema proporciona una plataforma completa para el almacenamiento y acceso rápido a información relevante. La gestión integrada de facturas y productos contribuye aún más a la eficiencia operativa, permitiendo un control

preciso sobre las transacciones comerciales y el inventario de la clínica. En resumen, este proyecto web representa una solución integral que fusiona la tecnología avanzada con la atención veterinaria, marcando un hito en la evolución de la gestión clínica para el bienestar de las mascotas y la comodidad de los propietarios.

Manual del programador

Arquitectura del sistema

La arquitectura utilizada es La arquitectura Modelo-Vista-Controlador (MVC) es un patrón de diseño ampliamente utilizado en el desarrollo de software que separa la lógica de la aplicación en tres componentes principales: Modelo, Vista y Controlador. Esta separación facilita la modularidad, el mantenimiento y la escalabilidad de las aplicaciones. Aquí se describe cada componente de la arquitectura MVC:

Modelo (Model):

- **Responsabilidad:** El Modelo representa la lógica de la aplicación y la manipulación de datos. Es responsable de acceder a la base de datos, procesar la información y realizar las operaciones de lógica empresarial.
- **Características:**

- Contiene la representación de los datos y las reglas de negocio.
- Puede enviar notificaciones de cambios a las Vistas.
- No tiene conocimiento de las Vistas o Controladores.

Vista (View):

- **Responsabilidad:** La Vista es responsable de la presentación de la información al usuario y de la interfaz de usuario. Muestra los datos provenientes del Modelo y reacciona a las acciones del usuario, enviando eventos al Controlador.
- **Características:**
  - Presenta la información de manera visual.
  - Recibe datos del Modelo y los muestra al usuario.
  - Puede enviar eventos al Controlador.

Controlador (Controller):

- **Responsabilidad:** El Controlador actúa como intermediario entre el Modelo y la Vista. Recibe eventos de entrada del usuario desde la Vista, procesa la lógica de la aplicación y actualiza el Modelo en consecuencia. También actualiza la Vista para reflejar los cambios en el Modelo.
- **Características:**
  - Maneja eventos de entrada del usuario desde la Vista.
  - Actualiza el Modelo en respuesta a acciones del usuario.
  - Actualiza la Vista para reflejar cambios en el Modelo.

Flujo de Trabajo:

1. El usuario interactúa con la Vista, generando eventos (clics, entradas, etc.).
2. La Vista envía estos eventos al Controlador.
3. El Controlador procesa los eventos y actualiza el Modelo según sea necesario.
4. El Modelo, al ser actualizado, notifica a la Vista.
5. La Vista recupera datos actualizados del Modelo y se actualiza visualmente.
6. El flujo de interacción continua según las acciones del usuario.

Ventajas de MVC:

- **Separación de Responsabilidades:** La separación clara de las responsabilidades facilita el mantenimiento y la escalabilidad del código.
- **Reutilización de Código:** Los componentes son independientes y pueden ser reutilizados en diferentes partes de la aplicación.
- **Facilita el Desarrollo en Equipo:** Diferentes desarrolladores pueden trabajar en paralelo en Modelos, Vistas y Controladores sin interferencias significativas.

Configuración del Entorno de desarrollo

Requisitos de Software:

- HTML
- CSS
- Javascript
- Typescript
- Node.js
- Express
- Angular
- MySQL

Configuración del Entorno:

- **En consola, ubicado en la carpeta veterinariaFrontend:**

Ejecutar "npm install" Para instalar las dependencias necesarias para el proyecto Angular.

Ejecuar "ng serve --o" o "ng serve" para ejecutar en modo desarrollador el proyecto (Es necesario tener intalado @angular/cli de manera global).

En caso de no tener @angular/cli instalado, Ejecuar "npm run start" o "npm start" para ejecutar en modo desarrollador el proyecto.

- **En consola, ubicado en la carpeta veterinariaBackendjs:**

Ejecutar "npm install" Para instalar las dependencias necesarias para el proyecto Express y MySql.

Ejecutar "npm run dev" para ejecutar el proyecto en modo desarrollador mediante la herramienta nodemon o "npm start" para inicar el proyecto.

Para La Base de Datos, en la carptera veterinariaBackendjs/sql se encuentra en script de la creacion de la base de datos.

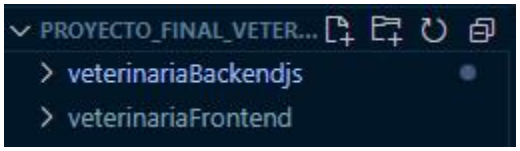
Por defecto, El proyecto está utilizando el puerto localhoost:3306 para la ejecución de la base de datos, puerto iniciado por defecto al instalar MySql con la configuración base.

Para los llamados a la base de datos, se está utiliando por defecto el usuario: "root" con una password: "1234".

Para la Ejeucion del Proyecto, es necesario tener los 3 servicios mencionados anteriormente activos simultáneamente.

Estructura del Código Fuente:

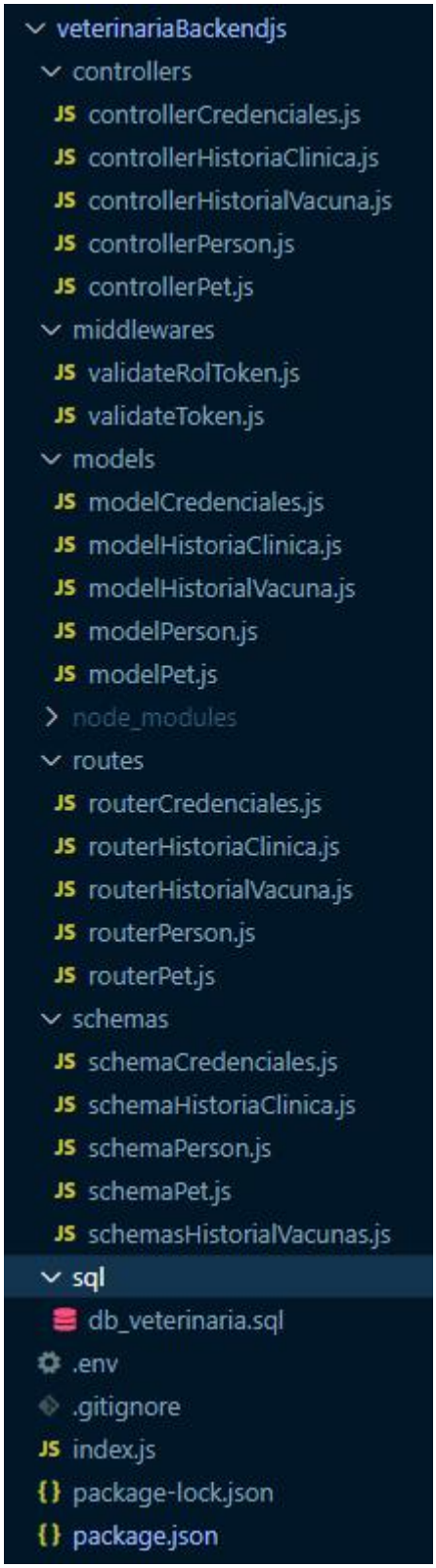
El proyecto se divide en 2 carpetas principales, como se muestra en la imagen siguiente:



La primera de ellas representa el Back-end de la aplicación y la segunda el Front-end.

Back-end

Dentro del directorio Back-end se incluyen las siguientes carpetas:


veterinariaBackendjs
controllers
controllerCredenciales.js
controllerHistoriaClinica.js
controllerHistorialVacuna.js
controllerPerson.js
controllerPet.js
middlewares
validateRolToken.js
validateToken.js
models
modelCredenciales.js
modelHistoriaClinica.js
modelHistorialVacuna.js
modelPerson.js
modelPet.js
node_modules
routes
routerCredenciales.js
routerHistoriaClinica.js
routerHistorialVacuna.js
routerPerson.js
routerPet.js
schemas
schemaCredenciales.js
schemaHistoriaClinica.js
schemaPerson.js
schemaPet.js
schemasHistorialVacunas.js
sql
db_veterinaria.sql
.env
.gitignore
index.js
package-lock.json
package.json

De dichas carpetas se destaca:

- **Controllers:** contiene los controladores para cada modelo de la aplicación.
- **middlewares:** contiene archivos de creación y validación de credenciales
- **models:** contiene los archivos CRUD para cada tabla de la base de datos.
- **node_modules:** contiene los paquetes instalados de node.
- **routes:** contiene los rutas, por medio de las cuales se realiza la comunicación con los controladores.
- **schemas:** contiene las validaciones pertinentes de los datos que son llevados al modelo.
- **sql:** contiene el script creación de la base de datos

Front-end

Dentro del directorio Front-end se incluyen las siguientes carpetas:


veterinariaFrontend
.angular
.vscode
node_modules
src
app
components
carrusel
footer
formulario-mascota
formulario-persona
navbar
guards
login.guard.ts
interfaces
persona.ts
pet.ts
user.ts
pages
about-us
home
login
mascota
persona
services
credenciales.service.ts
mascota.service.ts
persona.service.ts
utils
add-token.interceptor.ts
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.config.ts
app.routes.ts
assets
favicon.ico
index.html
main.ts
styles.css