Instituto de Formación Técnica Superior (I.F.T.S) N° 29

Tecnicatura Superior en Desarrollo de Software

Proyecto Segunda Parte

“Proyecto y Prácticas Formativas”

Espacio Curricular: Desarrollo de Sistemas Web (Back End)

Comisión: 2A

Año: 2025

Profesor: Emir Eliezer Garcia Ontiveros

Alumnos: MAXIMILIANO ALEM
PABLO M. IGLESIAS
EDUARDO JOSE TORCELLO
FABIO NATANAEL MORA

Sistema de Gestión de Clínica

Información del Proyecto

Nombre del Proyecto: Sistema de Gestión de Clínica

Versión: 2.0.0

Fecha: Septiembre 2025

Materia: Desarrollo de Sistemas Web (Back End) - 2° A

Carrera: Tecnicatura Superior en Desarrollo de Software

Introducción

Este proyecto corresponde al segundo parcial de la materia Desarrollo Web Backend. Tiene como objetivo principal extender la aplicación desarrollada en el primer parcial, integrando una base de datos MongoDB y aplicando las buenas prácticas de desarrollo vistas hasta la fecha. El proyecto consiste en un sistema de gestión de clínica médica que permite administrar pacientes, médicos, usuarios y turnos, con autenticación de roles y vistas dinámicas renderizadas con Pug.

Integrantes y Responsabilidades

Nombre	Rol Principal	Responsabilidades Específicas
Pablo M. Iglesias	**Líder de Proyecto Backend Developer**	Integración con MongoDB, controladores, middlewares y conexión
Maximiliano Alem	**Frontend & Documentación**	Vistas Pug, autenticación, documentación y pruebas

Repositorio del Proyecto: https://github.com/PMIglesias/BSWB-proyecto-backend-final

Objetivos del Proyecto

Objetivos específicos

- Mejorar la aplicación del primer parcial con **Node.js** y **Express**.
- Integrar **MongoDB** utilizando **Mongoose** como ODM.
- Implementar un sistema de **roles y autenticación** basado en sesiones.
- Aplicar **asincronía**, **POO** y **modularidad** en controladores, modelos y rutas.
- Seguir buenas prácticas de desarrollo backend y documentación.

Objetivos generales

- Desarrollar software por encargo.
- Integrar equipos de trabajo para el desarrollo colaborativo.
- Asumir roles especializados dentro del proyecto.
- Desempeñarse de forma autónoma en sistemas de baja complejidad.

Tecnologías utilizadas

Tecnología	**Uso principal**
Node.js	Entorno de ejecución backend
Express.js	Framework para el servidor
MongoDB + Mongoose	Base de datos y modelado de datos
Pug	Motor de plantillas para vistas
dotenv	Manejo de variables de entorno
bcryptjs	Encriptación de contraseñas
express-session	Manejo de sesiones y autenticación
CSS modular + JS	Interactividad y diseño (modo oscuro, búsqueda, modales)

Funcionalidades principales

- Gestión de Pacientes: CRUD completo (crear, listar, editar, eliminar).
- Gestión de Turnos: Asignación de turnos a pacientes, búsqueda y confirmación.
- Gestión de Médicos: Control de médicos con sus datos y especialidades.
- Gestión de Usuarios: CRUD con roles (admin, recepcionista, medico).
- Autenticación: Login y logout con sesiones.
- Control de Acceso: Middlewares que validan rol y sesión.
- Modo Oscuro: Persistente mediante localStorage.

Rutas principales (API REST)

Método	Ruta	Descripción
GET	/pacientes	Lista todos los pacientes
POST	/pacientes	Crea un nuevo paciente
PUT	/pacientes/:id	Actualiza un paciente
DELETE	/pacientes/:id	Elimina un paciente
GET	/turnos	Lista todos los turnos
POST	/turnos	Asigna un nuevo turno
GET	/auth/login	Renderiza la vista de login
POST	/auth/login	Inicia sesión
POST	/auth/logout	Cierra sesión

Estructura del proyecto

El proyecto sigue la estructura MVC con carpetas separadas para controladores, modelos, rutas, middlewares, vistas y archivos públicos. La base de datos se gestiona desde el archivo db.js, y las rutas están centralizadas en el enrutador principal. A continuación se presenta la estructura principal:

```
├─ data
│  └─ pacientes.json
├─ public
│  ├─ css
│  │  └─ styles.css
│  └─ js
│     ├─ asignarTurno.js
│     ├─ pacientes.js
│     └─ theme.js
└─ src
  ├─ app.js
  ├─ config
  │  └─ db.js
  ├─ controllers
  │  ├─ auth.controller.js
  │  ├─ medico.controller.js
  │  ├─ pacientes.controller.js
  │  ├─ turnos.controller.js
  │  └─ usuario.controller.js
  ├─ middlewares
  │  ├─ auth.js
  │  ├─ logger.js
  │  └─ validarPaciente.js
  ├─ models
  │  ├─ Medico.js
  │  ├─ Paciente.js
  │  ├─ Turno.js
  │  ├─ Usuario.js
  │  └─ paciente.model.js
  ├─ routes
  │  ├─ auth.routes.js
  │  ├─ index.js
  │  ├─ medico.routes.js
  │  ├─ pacientes.routes.js
  │  ├─ turnos.routes.js
  │  └─ usuario.routes.js
  ├─ scrips
  │  └─ seed.js
  ├─ utils
  │  └─ fileManager.js
  └─ views
    ├─ asignarTurno.pug
    ├─ error.pug
    ├─ index.pug
    ├─ layout.pug
    ├─ login.pug
    └─ pacientes.pug
```

Bibliografía y recursos

Documentación oficial de Node.js , Express y MongoDB .

Mongoose Docs

MDN Web Docs - Promesas, Fetch API y Async/Await

Tutoriales de Fazt, y freeCodeCamp.

Conclusión

Esta segunda entrega consolida la aplicación en un entorno real con base de datos MongoDB, autenticación de usuarios y estructura profesional bajo el patrón MVC.

Se aplicaron conceptos avanzados de asincronía, middleware, modularización y POO, alcanzando una aplicación funcional, segura y escalable.