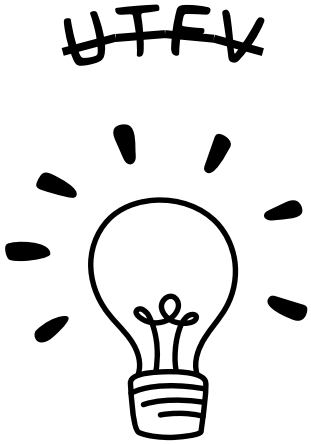

FRAMEWORK Y MONGO DB

Docente:
Silvia Guadalupe Bernal Fuentes

Alumnos:

Serrano Vázquez Jonatan Manuel

Mejía Villafranca Brandon Uriel

Jaramillo Martínez Donovan Alexis

Sánchez Gómez José Daniel

Contenido

1.2. Descripción General

¿Qué problema resuelve la aplicación?

La refaccionaria “Jony” maneja inventario de piezas automotrices, ventas y clientes de forma manual o desorganizada. Esto genera pérdidas, errores de stock y mala atención al cliente.

¿Quiénes son los usuarios objetivo?

- Dueño y empleados de la refaccionaria.
- Clientes que consultan disponibilidad de productos.
- Administradores encargados del stock y ventas.

1.3. Funcionalidades Principales

1. **Gestión de inventario:** agregar, editar y eliminar productos.
2. **Registro de ventas:** guardar ventas por fecha, productos y cliente.
3. **Búsqueda y consulta de productos disponibles:** por nombre, categoría o marca.

1.4. Tecnologías a Utilizar

- **Framework backend:** Express.js
- **Base de datos:** MongoDB (usando MongoDB Atlas)
- **ORM:** Mongoose
- **Otras herramientas:**
    - Postman (para pruebas API)
    - Dotenv (para configuración segura)
    - Nodemon (en desarrollo)
    - Git y GitHub (control de versiones)

1.5. Diagrama de Estructura (Colecciones en MongoDB)

```
Productos
 - nombre
 - marca
 - categoría
 - precio
 - cantidad_disponible
 - descripción

Ventas
 - fecha
 - productos (array con referencias a Productos + cantidad vendida)
 - total
 - cliente (nombre opcional)

Usuarios (opcional si hay login)
 - nombre
 - rol (admin, empleado)
 - email
 - contraseña
```

1.6. Justificación

¿Por qué estas tecnologías?

- **Express.js** permite crear una API de manera rápida y estructurada.
- **MongoDB** es ideal para manejar productos con diferentes características (categorías, marcas, cantidades) sin necesidad de esquemas fijos.
- **Mongoose** facilita el modelado y las validaciones.
- Con **Postman** se prueban rutas fácilmente, y **Git** asegura un historial de cambios claro.

2. Desarrollo de la Aplicación (Fase de Implementación)

2.1. Configuración del Entorno

- npm init -y
- Instalar paquetes:

```
npm install express mongoose dotenv cors
npm install --save-dev nodemon
```

Estructura recomendada:

```
/controllers
/models
/routes
/config
index.js
.env
```

Conexión con MongoDB en /config/db.js y .env:

```
MONGO_URI=mongodb+srv://<usuario>:<password>@<cluster>.mongodb.net/refajony
```

2.2. Modelado de Datos

Modelo de Producto (models/Producto.js):

```
const mongoose = require('mongoose');

const ProductoSchema = new mongoose.Schema({
  nombre: String,
  marca: String,
  categoría: String,
  precio: Number,
  cantidad_disponible: Number,
  descripción: String
});

module.exports = mongoose.model('Producto', ProductoSchema);
```

2.3. Implementación de Funcionalidades

Rutas básicas:

- GET /productos
- POST /productos
- PUT /productos/:id
- DELETE /productos/:id
- POST /ventas
- GET /ventas

Controladores separados, estructura MVC.

2.4. Pruebas y Validaciones

- Usa **Postman** para probar todas las rutas.
- Valida que no se vendan productos con stock insuficiente.
- Muestra mensajes de error claros (por ejemplo, “Producto no encontrado”).

2.5. Interfaz de Usuario (opcional)

Opcionalmente puedes hacer una interfaz sencilla para:

- Ver productos disponibles.
- Agregar productos.
- Registrar ventas.

Tecnologías recomendadas: HTML/CSS + JS o React.