# \*\*Desarrollo Full Stack de un sitio web E-Commerce: SimpleClothing\*\*

## \*\*Información del Proyecto\*\*

| \*\*Nombre:\*\* | \*\*Matrícula:\*\* |

|-----------------------------------|-----------------------------------|

| Ricardo Arriaga Ortegón | AL03110119 |

| Dana Elena Zertuche Castro | AL02991408 |

| \*\*Materia:\*\* | \*\*Maestro:\*\* |

| Desarrollo Full Stack | Juan Manuel Cepeda Barragán |

| \*\*Tarea:\*\* | \*\*Fecha:\*\* |

| Avance de Proyecto | 31/01/2025 |

---

## \*\*Descripción General\*\*

El proyecto \*\*SimpleClothing\*\* es un sitio web de comercio electrónico desarrollado para una marca mexicana de ropa urbana unisex. El objetivo principal es proporcionar una plataforma en línea donde los usuarios puedan explorar productos, realizar compras y mantenerse informados sobre las novedades de la marca a través de un blog. El sitio está diseñado con un enfoque minimalista y colores corporativos (negro y amarillo), garantizando una experiencia de usuario atractiva y funcional.

---

## \*\*Configuración del Proyecto\*\*

### \*\*1. Inicialización del Proyecto\*\*

1. \*\*Iniciar un nuevo proyecto con `npm init`\*\*:

```bash

npm init -y

```

2. \*\*Instalar dependencias necesarias\*\*:

```bash

npm install express jsonwebtoken bcryptjs cors dotenv mongoose

```

3. \*\*Configurar variables de entorno\*\*:

Crear un archivo `.env` en la raíz del proyecto con las siguientes variables:

```plaintext

PORT=5000

MONGO\_URI=mongodb+srv://<usuario>:<contraseña>@cluster0.mongodb.net/simpleclothing?retryWrites=true&w=majority

JWT\_SECRET=miclaveultrasecreta

```

4. \*\*Estructura del Proyecto\*\*:

```

SimpleClothing/

├── backend/

│ ├── models/

│ ├── controllers/

│ ├── routes/

│ ├── middleware/

│ ├── config/

│ └── server.js

├── frontend/

│ ├── public/

│ │ ├── index.html

│ │ ├── styles/

│ │ └── scripts/

│ └── images/

├── .env

├── .gitignore

└── README.md

```

---

## \*\*Backend (API RESTful con Express.js)\*\*

### \*\*1. Creación del Servidor Básico\*\*

1. \*\*Archivo `server.js`\*\*:

```javascript

const express = require("express");

const mongoose = require("mongoose");

const dotenv = require("dotenv");

const cors = require("cors");

dotenv.config();

const app = express();

const PORT = process.env.PORT || 5000;

// Middleware

app.use(cors());

app.use(express.json());

// Conexión a MongoDB

mongoose

.connect(process.env.MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log("Conectado a MongoDB"))

.catch((err) => console.error("Error conectando a MongoDB:", err));

// Rutas

app.use("/api/productos", require("./routes/productoRoutes"));

app.use("/api/auth", require("./routes/authRoutes"));

// Middleware de manejo de errores

app.use((err, req, res, next) => {

res.status(500).json({ message: err.message });

});

app.listen(PORT, () => console.log(`Servidor corriendo en http://localhost:${PORT}`));

```

### \*\*2. Operaciones CRUD\*\*

1. \*\*Modelo de Producto (`models/Producto.js`)\*\*:

```javascript

const mongoose = require("mongoose");

const ProductoSchema = new mongoose.Schema({

nombre: { type: String, required: true },

descripcion: { type: String, required: true },

precio: { type: Number, required: true },

imagenes: [{ type: String }],

tallas: [{ type: String }],

categoria: { type: String, required: true },

stock: { type: Number, default: 0 },

});

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

```

2. \*\*Controlador de Productos (`controllers/productoController.js`)\*\*:

```javascript

const Producto = require("../models/Producto");

exports.crearProducto = async (req, res) => {

try {

const producto = new Producto(req.body);

await producto.save();

res.status(201).json(producto);

} catch (err) {

res.status(500).json({ message: "Error al crear el producto" });

}

};

exports.obtenerProductos = async (req, res) => {

try {

const productos = await Producto.find();

res.status(200).json(productos);

} catch (err) {

res.status(500).json({ message: "Error al obtener los productos" });

}

};

```

3. \*\*Rutas de Productos (`routes/productoRoutes.js`)\*\*:

```javascript

const express = require("express");

const productoController = require("../controllers/productoController");

const router = express.Router();

router.post("/", productoController.crearProducto);

router.get("/", productoController.obtenerProductos);

module.exports = router;

```

### \*\*3. Autenticación con JWT\*\*

1. \*\*Modelo de Usuario (`models/Usuario.js`)\*\*:

```javascript

const mongoose = require("mongoose");

const bcrypt = require("bcryptjs");

const UsuarioSchema = new mongoose.Schema({

nombre: { type: String, required: true },

email: { type: String, required: true, unique: true },

password: { type: String, required: true },

});

UsuarioSchema.pre("save", async function (next) {

if (this.isModified("password")) {

this.password = await bcrypt.hash(this.password, 10);

}

next();

});

module.exports = mongoose.model("Usuario", UsuarioSchema);

```

2. \*\*Controlador de Autenticación (`controllers/authController.js`)\*\*:

```javascript

const Usuario = require("../models/Usuario");

const jwt = require("jsonwebtoken");

exports.registrarUsuario = async (req, res) => {

try {

const usuario = new Usuario(req.body);

await usuario.save();

res.status(201).json({ message: "Usuario registrado exitosamente" });

} catch (err) {

res.status(500).json({ message: "Error al registrar el usuario" });

}

};

exports.iniciarSesion = async (req, res) => {

try {

const { email, password } = req.body;

const usuario = await Usuario.findOne({ email });

if (!usuario || !(await usuario.comparePassword(password))) {

return res.status(401).json({ message: "Credenciales inválidas" });

}

const token = jwt.sign({ id: usuario.\_id }, process.env.JWT\_SECRET, { expiresIn: "1h" });

res.status(200).json({ token });

} catch (err) {

res.status(500).json({ message: "Error al iniciar sesión" });

}

};

```

3. \*\*Rutas de Autenticación (`routes/authRoutes.js`)\*\*:

```javascript

const express = require("express");

const authController = require("../controllers/authController");

const router = express.Router();

router.post("/registro", authController.registrarUsuario);

router.post("/login", authController.iniciarSesion);

module.exports = router;

```

---

## \*\*Frontend (HTML, CSS, JavaScript)\*\*

### \*\*1. Página de Inicio (`index.html`)\*\*:

```html

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>SimpleClothing - Inicio</title>

<link rel="stylesheet" href="styles/global.css">

<link rel="stylesheet" href="styles/home.css">

</head>

<body>

<header>

<h1>SimpleClothing</h1>

<nav>

<a href="index.html">Inicio</a>

<a href="productos.html">Productos</a>

<a href="login.html">Iniciar Sesión</a>

</nav>

</header>

<main>

<section id="destacados">

<h2>Productos Destacados</h2>

<div id="productos-container"></div>

</section>

</main>

<script src="scripts/home.js"></script>

</body>

</html>

```

### \*\*2. JavaScript para Cargar Productos (`scripts/home.js`)\*\*:

```javascript

document.addEventListener("DOMContentLoaded", async () => {

try {

const response = await fetch("/api/productos");

const productos = await response.json();

const container = document.getElementById("productos-container");

productos.forEach((producto) => {

const div = document.createElement("div");

div.innerHTML = `<h3>${producto.nombre}</h3><p>${producto.descripcion}</p>`;

container.appendChild(div);

});

} catch (err) {

console.error("Error cargando productos:", err);

}

});

```

---

## \*\*Despliegue Preliminar\*\*

1. \*\*Despliegue Local\*\*:

- Ejecutar el servidor:

```bash

npm start

```

- Acceder a `http://localhost:5000`.

2. \*\*Despliegue en la Nube (Render)\*\*:

- Subir el backend a Render y configurar la base de datos en MongoDB Atlas.

- Desplegar el frontend como un sitio estático.

---

## \*\*Documentación Técnica\*\*

### \*\*Endpoints de la API\*\*

- \*\*Productos\*\*:

- `GET /api/productos`: Obtener todos los productos.

- `POST /api/productos`: Crear un nuevo producto.

- \*\*Autenticación\*\*:

- `POST /api/auth/registro`: Registrar un nuevo usuario.

- `POST /api/auth/login`: Iniciar sesión.

---

## \*\*Entregables\*\*

1. \*\*Código Fuente\*\*:

- Repositorio en GitHub: [Enlace al Repositorio](#)

2. \*\*Documentación Técnica\*\*:

- Explicación de la estructura del proyecto y endpoints de la API.

3. \*\*Video Demostrativo\*\*:

- [Enlace al Video](#)

4. \*\*URL del Despliegue Preliminar\*\*:

- [Enlace al Despliegue](#)

---

## \*\*Conclusión\*\*

El proyecto \*\*SimpleClothing\*\* proporciona una base sólida para un sitio web de comercio electrónico funcional. Con las mejoras y optimizaciones futuras, puede convertirse en una plataforma robusta y eficiente para la compra de ropa en línea.

---