**Bootcamp Fullstack**

**Informe Proyecto Final**

**PROFESOR:**

JULIAN LATORRE

**INTEGRANTES:**

GODOY VALENCIA JUAN DAVID

CASTRO VALENCIA CAROLINA

**TALENTO TECH**

**BOGOTÁ D.C.**

**2024**

**Introducción**

Este proyecto fue desarrollado como parte de un aprendizaje grupal en el programa de desarrollo web Full Stack. El Task Manager es una aplicación que permite a los usuarios gestionar sus tareas a través de un CRUD (Crear, Leer, Actualizar y Eliminar). Se integraron tecnologías front-end (HTML, CSS, y JavaScript) y back-end (Node.js con JSON como formato de intercambio de datos y una base de datos en MySQL desplegada en Railway).

El repositorio del código fuente fue subido a GitHub, y se documentaron todos los avances, así como los desafíos enfrentados durante el desarrollo y despliegue.

**Propósito del proyecto:**

Desarrollar una aplicación web usando HTML, CSS, JS junto con el framework ExpressJS de NodeJS que permita a los usuarios gestionar tareas de forma eficiente. Con un sistema de autenticación, los usuarios pueden registrarse e iniciar sesión de forma segura. Una vez autenticados, los usuarios pueden crear, editar y eliminar tareas. La interfaz intuitiva muestra las diferentes tareas facilitando la organización personal.

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **Tecnologías Utilizadas**

1. **Front-End**:
   * HTML y CSS para la estructura y estilo del sitio.
   * JavaScript para la interacción dinámica.
2. **Back-End**:
   * Node.js con Express para el servidor.
   * JSON para el intercambio de datos.
3. **Base de Datos**:
   * MySQL para almacenamiento.
   * Railway para despliegue de base de datos.
4. **Control de Versiones**:
   * GitHub como repositorio central
5. **Despliegue:**
   * Render para despliegue del proyecto principal

**Proceso de Desarrollo**

1. **Planificación**:
   * Definición de funcionalidades esenciales: Crear, Leer, Actualizar y Eliminar tareas.
   * Diseño de la base de datos MySQL2.
   * Organización del trabajo en equipo, asignando roles específicos.
2. **Implementación**:
   * Desarrollo del front-end para la interfaz de usuario.
   * Configuración del back-end con Node.js y conexión a la base de datos.
   * Uso de Railway para desplegar la base de datos.
3. **Desafíos**:
   * Problemas de conexión entre Railway y el servidor local.
   * Ajustes necesarios en las consultas SQL.
   * Configuración del entorno de producción en Render para el despliegue.

**Soluciones**:

* + Investigación en documentación oficial.
  + GitHub como repositorio central.

**Tabla de verificación de funcionalidades**

| **Funcionalidad** | **Descripción** | **Estado** | **Revisado** |
| --- | --- | --- | --- |
| **Crear Tarea** | Permite al usuario añadir una nueva tarea al sistema. | ✅ | [ ] |
| **Leer Tareas** | Muestra las tareas existentes en la base de datos. | ✅ | [ ] |
| **Actualizar Tarea** | Posibilita editar detalles de una tarea existente. | ✅ | [ ] |
| **Eliminar Tarea** | Permite eliminar una tarea seleccionada. | ✅ | [ ] |
| **Validación de Formularios** | Comprueba campos obligatorios antes de enviar el formulario. | ✅ | [ ] |
| **Diseño Responsivo** | La interfaz se adapta a dispositivos móviles y de escritorio. | ✅ | [ ] |
| **Despliegue del Back-End** | Correcto despliegue del servidor en Render. | ✅ | [ ] |
| **Conexión a la Base de Datos** | Comunicación estable con la base de datos MySQL en Railway. | ✅ | [ ] |
| **Documentación en GitHub** | Incluye README detallado y enlaces a recursos del proyecto. | ✅ | [ ] |
| **Pruebas de Usuario** | Verificación de que todas las funcionalidades operan correctamente. | ✅ | [ ] |

**DOCUMENTACIÓN**

**Estructura del Proyecto**

.gitignore

backend/

.env

config/

db.js

controllers/

authController.js

taskController.js

middlewares/

authMiddleware.js

routes/

authRoutes.js

taskRoutes.js

server.js

frontend/

index.html

login.html

main.js

register.html

styles.css

package.json

1. **Backend**

El backend está construido con Node.js y Express. Se encarga de manejar las solicitudes HTTP, la autenticación de usuarios y la interacción con la base de datos MySQL.

**Archivos y Directorios**

config/db.js: Configuración de la conexión a la base de datos MySQL.

controllers/authController.js: Controlador para el registro e inicio de sesión de usuarios.

controllers/taskController.js: Controlador para la gestión de tareas (crear, obtener, actualizar y eliminar).

middlewares/authMiddleware.js: Middleware para la autenticación de usuarios mediante JWT.

routes/authRoutes.js: Rutas para el registro e inicio de sesión de usuarios.

routes/taskRoutes.js: Rutas para la gestión de tareas.

server.js: Archivo principal del servidor Express.

**Configuración de la Base de Datos**

El archivo db.js configura la conexión a la base de datos MySQL utilizando las variables de entorno definidas en el archivo .env.

**Controladores**

AuthController: Maneja el registro e inicio de sesión de usuarios.

register: Registra un nuevo usuario.

login: Inicia sesión de un usuario existente.

TaskController: Maneja la gestión de tareas.

getTasks: Obtiene todas las tareas de un usuario.

getTaskById: Obtiene una tarea por su ID.

createTask: Crea una nueva tarea.

updateTask: Actualiza una tarea existente.

deleteTask: Elimina una tarea por su ID.

**Middlewares**

AuthMiddleware: Verifica la autenticación del usuario mediante JWT.

authenticate: Middleware para autenticar usuarios.

**Rutas**

AuthRoutes: Define las rutas para el registro e inicio de sesión de usuarios.

/register: Ruta para registrar un nuevo usuario.

/login: Ruta para iniciar sesión.

TaskRoutes: Define las rutas para la gestión de tareas.

/: Ruta para obtener todas las tareas y crear una nueva tarea.

/:id: Ruta para obtener, actualizar y eliminar una tarea por su ID.

**Servidor**

El archivo server.js configura y arranca el servidor Express, sirviendo los archivos estáticos del frontend y definiendo las rutas de la API.

1. **Frontend**

El frontend está construido con HTML, CSS y JavaScript. Se encarga de la interfaz de usuario y de interactuar con el backend mediante solicitudes HTTP.

**Archivos**

index.html: Página principal de la aplicación.

login.html: Página de inicio de sesión.

register.html: Página de registro.

styles.css: Estilos CSS para la aplicación.

main.js: Lógica de la aplicación en JavaScript.

**Funcionalidades**

Registro de Usuarios: Permite a los usuarios registrarse en la aplicación.

register-form: Maneja el formulario de registro.

Inicio de Sesión: Permite a los usuarios iniciar sesión en la aplicación.

login-form: Maneja el formulario de inicio de sesión.

Gestión de Tareas: Permite a los usuarios crear, actualizar y eliminar tareas.

fetchTasks: Obtiene y muestra las tareas del usuario.

add-task: Maneja la creación de nuevas tareas.

update-task: Maneja la actualización de tareas existentes.

delete-task: Maneja la eliminación de tareas.

**Instalación y Ejecución**

1. Clona el repositorio.
2. Instala las dependencias del backend:

cd backend

npm install

1. Configura las variables de entorno en el archivo .env en el directorio backend.
2. Inicia el servidor:

npm start

1. Abre la dirección en la que se está ejecutando el servidor.

**Dependencias**

Las dependencias del proyecto están listadas en el archivo package.json.

bcrypt: Para el hash de contraseñas.

body-parser: Para parsear el cuerpo de las solicitudes HTTP.

cors: Para habilitar CORS.

dotenv: Para manejar variables de entorno.

express: Framework de servidor web.

jsonwebtoken: Para la autenticación mediante JWT.

mysql2: Cliente MySQL para Node.js.

**Links útiles:**  
 Link del proyecto: <https://tasks-manager-js.onrender.com/>  
 Repositorio del código: <https://github.com/juandavidg123/tasks-manager-js>