Manual Técnico del Proyecto: Módulo de Gestión de Empleados

1. Introducción

El presente manual técnico documenta el desarrollo y despliegue del Módulo de Gestión de Empleados, un aplicativo basado en el stack tecnológico MEAN (MongoDB, Express, Angular, Node.js). La aplicación permite gestionar empleados mediante operaciones CRUD (Crear, Leer, Actualizar, Eliminar), interactuando con una base de datos MongoDB.

2. Arquitectura del Sistema

2.1 Diagrama de Arquitectura

Descripción: La aplicación está dividida en dos partes principales:

Frontend: Desarrollado con Angular, maneja la interfaz de usuario y la interacción con el backend.

Backend: Desarrollado con Node.js y Express, se encarga de la lógica de negocio y la comunicación con la base de datos MongoDB.

Diagrama: (Incluir un diagrama que muestre la interacción entre Angular, Express, Node.js, y MongoDB).

2.2 Modelo de Datos

Descripción: El modelo de datos de empleados utilizado en MongoDB es:

_id: string; // Identificador único generado automáticamente por MongoDB.

name: string; // Nombre del empleado.

position: string; // Cargo o posición del empleado en la empresa.

office: string; // Oficina o ubicación del empleado.

salary: number; // Salario del empleado.

3. Backend

3.1 Descripción General

Lenguaje de programación: JavaScript.

Framework utilizado: Express.js.

Base de datos: MongoDB.

Configuración del servidor: El backend se inicia con el comando node index.js, y se pueden realizar cambios dinámicos durante el desarrollo utilizando nodemon para reiniciar el servidor automáticamente.

3.2 Estructura del Proyecto

Controllers: Contiene la lógica de negocio para manejar las operaciones CRUD. El archivo empleado.controller.js es el encargado de gestionar estas operaciones.

Models: Define la estructura de los datos en MongoDB. El archivo empleado.js establece el esquema del empleado en la base de datos.

Routes: Define las rutas de la API que mapean las operaciones CRUD a las funciones del controlador. Estas rutas están en empleado.routes.js.

Archivos adicionales:

database.js: Configuración de la conexión con la base de datos MongoDB.

index.js: Punto de entrada del backend donde se inicializa el servidor.

3.3 Instalación y Configuración

Requisitos previos:

Node.js (versión recomendada 14.x o superior).

MongoDB (versión recomendada 5.x o superior).

Instrucciones de instalación:

Clonar el repositorio del backend.

Instalar las dependencias utilizando npm install.

Configurar las variables de entorno en un archivo .env, incluyendo la URI de MongoDB.

Ejecutar npm start o npm run dev para iniciar el servidor.

3.4 Endpoints de la API

GET /api/empleados: Obtener la lista de empleados.

POST /api/empleados: Crear un nuevo empleado.

PUT /api/empleados/:id: Actualizar un empleado existente.

DELETE /api/empleados/:id: Eliminar un empleado.

3.5 Conexión a la Base de Datos

Configuración: La conexión con MongoDB se maneja en database.js. La URI de la base de datos se establece a través de una variable de entorno en el archivo .env. Es importante asegurarse de que la base de datos esté correctamente configurada y accesible.

4. Frontend

4.1 Descripción General

- Lenguaje de programación: TypeScript.
- Framework/Biblioteca utilizada: Angular.
- Estructura del Proyecto:
- src/app/: Contiene todos los componentes y servicios de Angular.
- angular.json: Archivo de configuración del proyecto Angular.

4.2 Instalación y Configuración

- Requisitos previos:
- Node.js (versión recomendada 14.x o superior).
- Angular CLI (versión recomendada 14.x o superior).
- Instrucciones de instalación:
- Clonar el repositorio del frontend.
- Instalar las dependencias utilizando npm install.
- Configurar el archivo de entorno .env con la URL de la API del backend.
- Ejecutar npm start para iniciar la aplicación frontend.

4.3 Estructura del Proyecto

Componentes principales: Los componentes clave manejan la vista de empleados, permitiendo la interacción del usuario con el sistema. Por ejemplo, el componente EmpleadoComponent lista los empleados y maneja las operaciones CRUD.

Servicios: Un servicio Angular (EmpleadoService) se encarga de realizar las llamadas a la API del backend para manejar los datos de empleados.

4.4 Conexión con el Backend

Descripción: Los componentes de Angular interactúan con el backend utilizando servicios HTTP para enviar y recibir datos. Esto se realiza a través del servicio EmpleadoService, que se comunica con los endpoints de la API.

Flujo de Datos: El servicio se suscribe a las respuestas HTTP del backend y actualiza el estado de la aplicación en el frontend.

5. Pruebas

5.1 Pruebas Backend

Descripción: Se realizaron pruebas unitarias e integración en el backend utilizando herramientas como Postman e Insomnia para validar la funcionalidad de los endpoints y la correcta interacción con la base de datos.

Metodología: Se probaron todos los endpoints de la API para asegurar que las operaciones CRUD se realizan correctamente.

5.2 Pruebas Frontend

Descripción: Las pruebas en el frontend se realizaron utilizando Jasmine y Karma, que son las herramientas estándar en Angular para validar la funcionalidad de los componentes.

Enfoque: Se realizaron pruebas de unidad y pruebas end-to-end (E2E) para asegurar que cada componente se comporta según lo esperado.

6. Despliegue

6.1 Backend

Descripción: El backend se puede desplegar en un servidor de producción como Heroku, AWS, o en un entorno local. Es importante asegurarse de que todas las configuraciones, especialmente las variables de entorno y la conexión a la base de datos, estén correctamente establecidas.

Pasos:

Configurar las variables de entorno en el servidor.

Asegurarse de que MongoDB esté disponible y accesible desde el servidor.

Ejecutar npm start para iniciar la aplicación en producción.

6.2 Frontend

Descripción: El frontend puede ser desplegado en servicios como Netlify o Vercel, que son adecuados para aplicaciones estáticas.

Pasos:

Ejecutar ng build --prod para compilar la aplicación en modo producción.

Subir los archivos compilados a la plataforma de despliegue.

Configurar la base de datos para que apunte a la API del backend en producción.

7. Conclusiones

Resumen: El Módulo de Gestión de Empleados permite gestionar eficientemente los datos de los empleados a través de una interfaz amigable y un backend robusto. La arquitectura MEAN facilita el desarrollo escalable y el mantenimiento de la aplicación.

Automatización y Eficiencia: La implementación de un gestor de empleados permite automatizar las tareas administrativas relacionadas con la gestión del personal, reduciendo significativamente el tiempo y esfuerzo necesarios para realizar operaciones como la actualización de registros, seguimiento de información laboral y cálculo de salarios.

Centralización de Datos: Utilizando una base de datos NoSQL como MongoDB, se logra una centralización eficiente de la información de los empleados, permitiendo un acceso rápido y organizado a los datos desde cualquier lugar y en cualquier momento, lo que facilita la toma de decisiones en tiempo real.

Flexibilidad y Escalabilidad: El uso de MongoDB como base de datos permite manejar estructuras de datos flexibles, lo que es ideal para una aplicación de gestión de empleados donde los requisitos pueden evolucionar con el tiempo. Además, la arquitectura del sistema basada en MEAN es fácilmente escalable para soportar un mayor volumen de datos y usuarios.

Desarrollo Full Stack con JavaScript: El uso del stack MEAN permite a los desarrolladores trabajar con un solo lenguaje de programación (JavaScript) en todo el proyecto, desde el frontend (Angular) hasta el backend (Node.js y Express), lo que simplifica el proceso de desarrollo y facilita la integración entre los distintos componentes del sistema.

Interfaz de Usuario Dinámica y Responsiva: Angular, como framework de frontend, ofrece una interfaz de usuario dinámica y responsiva que mejora la experiencia del usuario. Esto es crucial en un gestor de empleados, ya que permite que la administración del personal se realice de manera intuitiva y eficiente desde cualquier dispositivo.

Manejo Eficiente de Datos en Tiempo Real: Gracias a la arquitectura basada en Node.js, el sistema es capaz de manejar múltiples solicitudes simultáneamente de manera eficiente, permitiendo la actualización en tiempo real de los datos de empleados y mejorando la interacción entre el usuario y el sistema.

Modularidad y Mantenibilidad: La estructura modular del proyecto, con la separación clara entre componentes, servicios y rutas, facilita el mantenimiento y la actualización del sistema, lo que es esencial para garantizar la sostenibilidad del proyecto a largo plazo.

Seguridad y Control de Acceso: Utilizando tecnologías como Express.js en el backend, se puede implementar fácilmente la autenticación y autorización, asegurando que solo los usuarios autorizados puedan acceder o modificar los datos sensibles de los empleados, lo que es vital para la protección de la información personal.

Facilidad de Despliegue y Escalabilidad en la Nube: La arquitectura basada en MEAN permite un despliegue sencillo en plataformas cloud como AWS, Heroku o servicios de frontend como Netlify y Vercel, garantizando que la aplicación sea accesible desde cualquier lugar y pueda escalarse según las necesidades del negocio.

Preparación para el Futuro: La elección del stack MEAN asegura que el proyecto esté alineado con las tecnologías modernas y emergentes, permitiendo que el gestor de empleados esté preparado para futuras integraciones con otras herramientas y servicios, como sistemas de inteligencia artificial o análisis de datos.

Mejoras Futuras: Se podría integrar autenticación de usuarios para mejorar la seguridad y realizar optimizaciones en el rendimiento de la aplicación.