Universidad Provincial del Sudoeste

Trabajo Final de Desarrollo de Aplicaciones Web 2

Task Manager

Desarrollado por Pablo Uriel Gomez y Matias Bussetti

6 de diciembre de 2023

Índice

1 Introducción

El proyecto de **Task Manager** es una aplicación moderna de gestión de tareas que proporciona a los usuarios una solución eficiente para organizar sus responsabilidades diarias. Desarrollado con tecnologías como Vue.js, TypeScript, Tailwind CSS y MongoDB, esta aplicación web ofrece una experiencia completa que abarca desde la gestión básica de tareas hasta funciones avanzadas como filtrado, ordenamiento y marcado especial de tareas.

2 Objetivos del Proyecto

2.1 Gestión de Tareas

- Creación, edición y eliminación de tareas.
- Marcado de tareas como completadas o favoritas.

2.2 Filtrado y Ordenamiento

- Filtrado inteligente de tareas completadas y favoritas.
- Ordenamiento dinámico por fecha, título y otras categorías relevantes.

2.3 Autenticación de Usuarios

- Registro seguro de nuevos usuarios.
- Inicio de sesión para acceder a las funcionalidades personalizadas.

2.4 Desarrollo Modular

- Estructura organizada en módulos claros para una fácil mantenibilidad.
- Uso eficiente de componentes Vue.js para la creación de interfaces de usuario reutilizables.

2.5 Tecnologías Modernas

- Adopción de Vue 2 y TypeScript para el desarrollo frontend.
- Integración de Tailwind CSS para una interfaz de usuario atractiva y altamente personalizable.
- Implementación de MongoDB como base de datos para un almacenamiento eficiente y escalable.
- Uso de Docker para facilitar la implementación y la portabilidad.

3 Estructura del Proyecto

3.1 Frontend

La carpeta `frontend` alberga la interfaz de usuario principal de la aplicación. Aquí, los elementos clave incluyen:

- `src/assets`: Contiene archivos estáticos, como el principal archivo de estilos (`main.css`).
- `src/components`: Organiza componentes reutilizables para una interfaz modular y mantenible.
- `src/icons`: Incluye componentes Vue específicos para íconos personalizados.
- `src/router`: Configuración de rutas mediante Vue Router para una navegación eficiente.
- `src/services`: Lógica para interactuar con el backend, utilizando Axios para solicitudes HTTP.
- `src/views`: Define las vistas principales de la aplicación, como `Login`, `Registro` y `Tareas`.
- `App.vue`: Archivo principal que define la estructura general de la aplicación.

3.2 Backend

En la carpeta `backend`, se encuentra la lógica del servidor y la conexión a la base de datos. Aquí, destacamos:

- `models`: Define el esquema de MongoDB para las tareas almacenadas.
- `controllers`: Contiene la lógica que manipula las tareas.
- `routes`: Define las rutas de la API, incluyendo una para la construcción del frontend.
- `utils`: Incluye utilidades como funciones y un Logger personalizado.
- `app.js`: Archivo principal que configura y ejecuta el servidor backend.
- `dockerfile.dev`: Configuración de Docker para facilitar el desarrollo.
- `.env`: Archivo que contiene variables de entorno sensibles.

4 Tecnologías Utilizadas

4.1 Frontend

- **Vue 2**: Framework progresivo para la construcción de interfaces de usuario.
- **TypeScript**: Extiende JavaScript al agregar tipos estáticos opcionales.
- **Tailwind CSS**: Framework de utilidades de estilos altamente personalizable.

- **Axios**: Cliente HTTP para realizar solicitudes al backend.
- **Firebase**: Herramientas y servicios en la nube para el desarrollo de aplicaciones.
- **PostCSS**: Procesador de CSS que permite utilizar plugins para mejorar y optimizar el código.

4.2 Backend

- **Node.js**: Entorno de ejecución para JavaScript en el servidor.
- **Express**: Framework web para construir aplicaciones robustas en Node.js.
- **MongoDB**: Base de datos NoSQL para el almacenamiento eficiente y escalable de datos.
- **Redis**: Sistema de almacenamiento en caché para mejorar el rendimiento.
- **Docker**: Plataforma que facilita el desarrollo, la implementación y la ejecución de aplicaciones.

5 Dependencias

5.1 Frontend

```
{
  "dependencies": {
    "@vue/composition-api": "^1.4.6",
    "axios": "^1.6.2",
    "firebase": "^10.6.0",
    "luxon": "^3.4.4",
    "vue": "^2.6.14",
    "vue-router": "^3.5.3"
  },
  "devDependencies": {
    "typescript": "~4.7.3",
    "tailwindcss": "^3.3.5",
    "vite": "^2.8.4",
    // Otras dependencias...
  }
}
```

5.2 Backend

```
{
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "ioredis": "^5.3.2",
```

```
    "mongoose": "^8.0.1",
    "nodemon": "^3.0.1"
    // Otras dependencias...
  }
}
```

6 Instalación y Configuración

Sigue estos pasos para instalar y configurar la aplicación en tu entorno local:

1. **Clonar el Repositorio:**
   `git clone https://github.com/PabloUGomez/Proyecto-Final.git`
2. **Instalar Dependencias Frontend:**
   `cd frontend`
   `npm install`
3. **Instalar Dependencias Backend:**
   `cd backend`
   `npm install`
4. **Configuración de Variables de Entorno:**
   Asegúrate de configurar adecuadamente las variables de entorno en el archivo `.env` del backend.
5. **Ejecutar la Aplicación Frontend:**
   `cd frontend`
   `npm run dev`
6. **Ejecutar la Aplicación Backend:**
   `cd backend`
   `npm run start`
7. **Acceder a la Aplicación:**
   Visita `http://localhost:8000` en tu navegador.

7 Uso del Proyecto

7.1 Registro e Inicio de Sesión:

Los usuarios deben registrarse o iniciar sesión para acceder a las funcionalidades de la aplicación.

7.2 Gestión de Tareas:

- Crear, editar y borrar tareas de manera intuitiva.
- Marcar tareas como completadas y favoritas según sea necesario.

7.3 Filtrado y Ordenamiento:

- Utilizar opciones avanzadas de filtrado para visualizar tareas específicas.
- Ordenar tareas de acuerdo a criterios como fecha y título.

7.4 Interfaz de Usuario Amigable:

Navegar por la aplicación de manera fácil y cómoda gracias a una interfaz bien diseñada.

8 Ejemplos de Uso

Imagina a un usuario que busca organizar sus tareas diarias de manera eficiente. Podría seguir estos pasos:

8.1 Creación de Tarea:

1. Crea una nueva tarea con el título "Terminar entrega de la universidad".
2. Asigna la categoría .[Ed]ucaciónτ proporciona una descripción detallada.

8.2 Marcado Especial:

1. Marca la tarea como favorita si consideras que es de alta prioridad.
2. Después de completar la tarea, márcala como Çompletada".

8.3 Filtrado y Ordenamiento:

1. Utiliza funciones avanzadas para filtrar tareas completadas o favoritas.
2. Ordena las tareas por fecha de creación o alfabéticamente según sea necesario.

8.4 Organización Eficiente:

Utiliza la aplicación como una herramienta centralizada para mantenerse organizado.

9 Backend: Análisis Profundo

9.1 Descripción

El *Server Side* consiste en una aplicación API Restful utilizando Node.js como intérprete para ejecutar JavaScript como lenguaje de programación. En este, se utiliza el patrón Modelo-Vista-Controlador para presentar los recursos a los consumidores. Se utilizó la Express como librería que provee una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características. A continuación, se describirá el uso de estas:

9.2 Configuración Previa:

El desarrollo del Server Side ha sido pensado para depender solo de **MongoDB** y **Redis**. Por lo que se deberá tener previamente estos instalados de forma local o en otro host. En la **carpeta *backend*** ubicada en el proyecto, se puede encontrar el archivo con nombre ***.env***. En este se podrán hacer los cambios necesarios que requieran los distintos entornos de desarrollo. En esta se podrán encontrar las siguientes variables de entorno:

- Puerto del servidor
- Host y Puerto de Mongo
- Host de Redis

9.3 Rutas, Endpoints y Middlewares:

Express nos brinda características simples para realizar implementaciones complejas. En estas se pueden encontrar las rutas y los middlewares.

9.3.1 Middleware

El sistema solo contiene un middleware utilizado por las rutas de tareas. Este middleware cumple la función de comprobar que exista en el encabezado la propiedad ***auth***. La propiedad ***auth*** se utilizará como método para separar los usuarios, unos de otros. Ya que el sistema no contiene un módulo de autenticación.

9.3.2 Rutas y Endpoints

El sistema posee 6 rutas que corresponden al controlador `taskController`. Estas representan los siguientes *Endpoints*:

/api/tasks

MÉTODO /ruta *función()*: Descripción

GET / *index()*: Retorna un arreglo con las tareas del usuario

POST / *store()*: Almacena una tarea

PUT /:id *update()*: Actualiza una tarea

PUT /:id/favorita *setFavorite()*: Establece el valor de favorito de una tarea

PUT /:id/completada *setComplete()*: Establece el valor de completada de una tarea

DELETE /:id *delete()*: Borra una tarea

10 Docker

10.1 Archivos

Dentro del proyecto se puede encontrar la carpeta ***/docker***, la cual contiene los siguientes archivos:

- ***docker-compose.yml***: Este archivo se utiliza con Docker Compose para definir y configurar servicios, redes y volúmenes para tu aplicación, permitiendo la ejecución de múltiples contenedores de manera coordinada.
- ***/js***
    - ***Dockerfile.dev***: Un archivo Dockerfile específico para el desarrollo de aplicaciones Node.js.
- ***/nginx***
    - ***Dockerfile***: Archivo Dockerfile para la imagen de Nginx, un servidor web y proxy inverso.
    - ***nginx.conf***: Archivo de configuración de Nginx.
- ***/redis***
    - ***Dockerfile.dev***: Archivo Dockerfile específico para el desarrollo de aplicaciones que utilizan Redis.
    - ***redis.conf***: Archivo de configuración de Redis.

11 Versionado

`https://github.com/PabloUGomez/Proyecto-Final`