Imatia
Laura Pérez López

Task Manager

29 de septiembre del 2023

DEFINICIONES Y ESPECIFICACIÓN DE REQUERIMIENTOS

Definición del problema

El problema a tratar es la creación de una aplicación que permita la gestión de tareas personales desarrollando tanto la API que conforme el backend de la aplicación como la web que la consume. Para ello, en el backend se ha implementado en String Boot mientras que el apartado visual se ha desarrollado con Angular, utilizando el módulo externo Material.

Funcionalidades implementadas

El usuario podrá realizar las siguientes acciones desde la página web.

- Crear una nueva tarea.
- Establecer el estado de la tarea como “Por realizar”, “En proceso” o “Finalizada”.
- Eliminar una tarea.
- Modificar una tarea.
- Visualizar las tareas creadas, ordenadas según su fecha de vencimiento, en diferentes columnas.
- Visualización de los detalles de una tarea concreta.
- Paginación de cada una de las columnas en las que se encuentran las diferentes tareas.
- Soporte multi-idioma [Inglés - Español].

ARQUITECTURA DEL SISTEMA

Backend

Se trata de una implementación en Java, una API Rest implementada con Spring Boot, utilizando la base de datos H2.

Dentro de la base de datos, tendremos una tabla donde guardaremos todos los datos necesarios para poder generar un objeto Task. Los parámetros guardados en base de datos son los siguientes:

- **id**: Identificador único asociado a cada tarea.
- **date**: Fecha de finalización de la tarea.
- **name**: Nombre de la tarea.
- **status**: Estado de la tarea. Puede tomar los valores “To do”, “In progress”, “Done”.
- **type**: Tipo de tarea. Puede tomar los valores "Task", "Epic", "Incidence", "Research".
- **priority**: Prioridad de la tarea. Puede tomar los valores "Low", "Medium", "High", "Let him cook".
- **description**: Descripción detallada de la tarea.

La implementación de la API Rest se divide en cuatro clases principales:

- **Task**: Objeto tarea, el cual está construido por todos los datos almacenados en la base de datos.
- **Task Controller**: Controlador encargado de realizar las peticiones a la base de datos. Será a donde realicemos las peticiones desde el front, por lo que en ella estableceremos los enlaces a los que haremos las mismas.
- **Task Repository**: Repositorio asociado al objeto Task.
- **Task Not Found Exception**: Excepción personalizada que nos informará que no se encuentra la Task que se está buscando en la base de datos.

Frontend

El apartado gráfico de la aplicación se encuentra íntegramente realizado en angular, utilizando el módulo Material. Todas las aplicación se basa en cinco componentes:

- **main**: Esqueleto principal de la aplicación. Incluye la barra de navegación así como el botón para soporte multi-idioma. A partir de este componente se realiza todo el control del idioma de la aplicación así como la navegación entre los distintos componentes que se van a definir a continuación.
- **trello**: Vista de las diferentes tareas. Nos permite ver ordenadas, visualmente, las diferentes tareas en sus correspondientes columnas y mover las tarjetas de tareas entre ellas mediante un evento “drag and drop”. Asimismo, se incluye en cada columna un elemento de paginación, el cual nos permite navegar entre las diferentes tareas de esa columna. También podemos acceder a los detalles de cada tarea haciendo click en el nombre de la tarea.

- **view Task**: Vista de los detalles de las diferentes tareas. Se habilitarán botones para modificar y eliminar la tarea en cuestión.
- **add Task**: Añadir una tarea a memoria. Validará los datos introducidos y realizará una petición a backend para guardarla.

DESARROLLO DEL PROYECTO

Decisiones tomadas durante el desarrollo

- Para no aumentar la complejidad, se ha decidido guardar en el backend los arrays de tipos, prioridades y estados de una tarea. La forma óptima o posible ampliación sería crear una entidad para cada uno, haciendo así que la inclusión de alguno de ellos sea más sencilla.
- Para no sobrecargar las peticiones a back y que sean más rápidas, se ha realizado una función setStatus() para únicamente realizar una modificación en la tarea.
- Existe delay visual a la hora de insertar las tareas en la nueva columna, a causa de la implementación de la reordenación de las tareas para mostrarlas por orden de fecha.
- Se ha intentado crear un componente de cada columna trello para optimizar código. Esto no ha sido posible a causa de problemas a la hora de linkear las listas. Por ello, se ha tenido que triplicar el código para que la paginación sea independiente de cada columna.