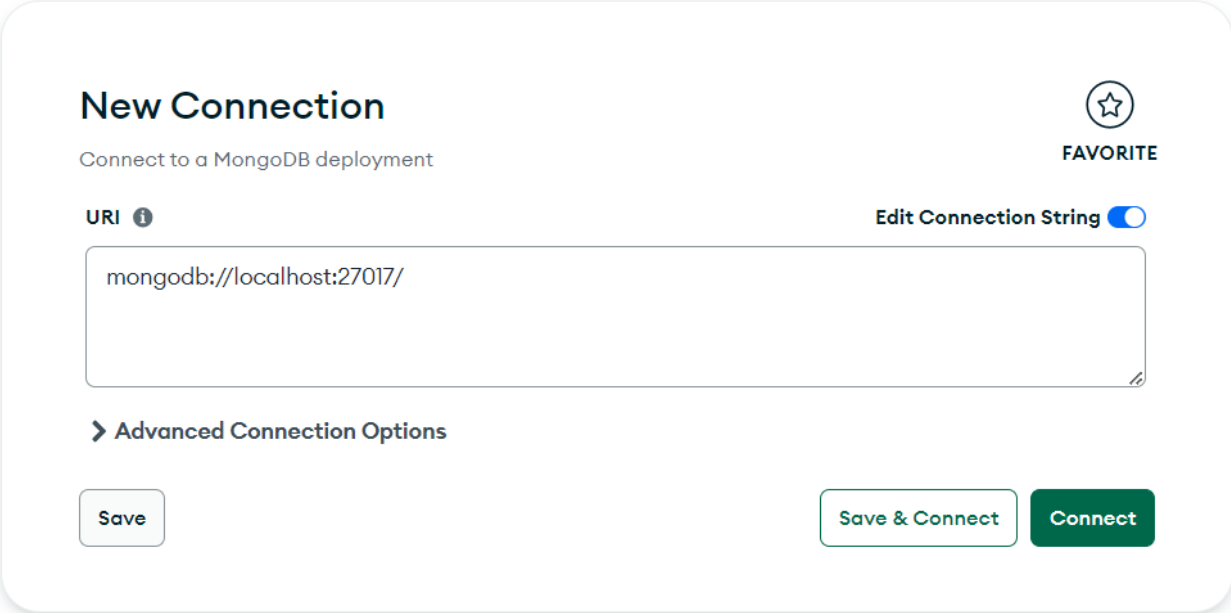
To-Do List

¿Cómo ejecutar el Proyecto localmente?

- Clonar el repositorio de GitHb, en consola poner el siguiente comando **git clone** https://github.com/pilig07/To-do-list.git
- Teniendo el proyecto clonado, buscamos en nuestro explorador de archivos la ubicación del proyecto y lo abrimos en Visual Code puede ser desde consola con **code .** o desde Visual Code explorar archivos.
- Teniendo abierto el IDLE abriremos 2 terminales dentro de nuestra misma carpeta:
    - La primera terminar tiene que ir al directorio “backend (server)” usamos el comando **cd backend (server)**
    - En esta primera terminar damos **npm install** para descargar las dependencias del proyecto
    - Para la segunda terminar mismo procedimiento solo que ahora el direcotrio es “frontend”
- Dentro de la terminal del backend el comando para correr es **npm run dev** y para el frontend es **npm start**
    - Nota importante: Para que el servidor funcione debemos tener instaldo MongoDB y además tenerlo corriendo con la conexión default



Teniendo esto el programa ya puede ser completamente instalado y utilizado sin problema alguno.

Arquitectura del proyecto

Nuestro directorio principal se llama “To-do list” de ahí partimos en 2 carpetas que separan el backend y el frontend. Para la carpeta del backend utiliza Node.js y Express.js como framework y lenguaje de programación, a su vez se utilizaron las dependencias de:

- Cors: permite que los recursos del servidor sean accesibles desde diferentes dominios
- Express: Framework web para Node.js que facilita la creación de aplicaciones web y APIs
- JsonWebtoken: autenticación y autorización segura de usuarios en aplicaciones web
- Mongoose: modelar los datos de la aplicación y facilita las interacciones con la base de datos MongoDB.
- Nodemon: Herramienta que ayuda a desarrollar aplicaciones Node.js reiniciando automáticamente el servidor cuando se detectan cambios en los archivos del proyecto.

Dentro del directorio backend tenemos index.js que es nuestro motor del servidor donde se encuentran todas las APIs creadas para permitir la manipulación de datos con nuestra base de datos. Además, tenemos la carpeta model con la definición de las colecciones de usuarios y tareas para la inserción de instancias en la base de datos.

Para la parte del frontend tenemos la creación de la parte del cliente para nuestro sistema utilizando como base React, además de las dependencias:

- React-Dom: montar y actualizar componentes de React en el navegador.
- React-Icons: Biblioteca de iconos para React
- React-router-dom: manejar la navegación y el enrutamiento en aplicaciones React.

De ahí tenemos la carpeta src, la cual se divide en varios directorios el primero se llama autenticación el cual lleva las vistas de Login y Registro de usuarios. La carpeta styles alberga las diferentes hojas de estilos utilizada para las interfaces.

En la carpeta Task tenemos la vista principal del sistema donde se despliegan las tareas, y dentro de esta carpeta una llamada componentes la cual almacena la acción principal de nuestro sistema que es añadir tareas, esto con la intención de poder aplicar este estilo ya

sea para añadir usuarios, otros objetos o cualquier cosa que quiera ser montada como una lista a futuro en el sistema.

Y por ultimo App.js donde tenemos la raíz y compilación total de nuestro proyecto, es decir manejamos las rutas, para que el usuario pueda moverse por la aplicación.

Decisiones técnicas

Separación del Backend y Frontend:

- Motivo: Mantener una arquitectura limpia y modular que facilite el desarrollo, mantenimiento y escalabilidad del proyecto. La separación permite trabajar de manera independiente en el backend y el frontend, mejorar la organización del código y permitir la implementación de pruebas y despliegues separados.
- Beneficios: Permite que los equipos de desarrollo trabajen en paralelo, facilita el cambio de tecnología en una parte del sistema sin afectar la otra, y mejora la seguridad al tener una separación clara entre la lógica del servidor y la interfaz de usuario.

No uso de SASS:

- Motivo: Limitaciones técnicas debido a problemas de compilación con Python en mi maquina personal.
- Alternativa: Se optó por utilizar CSS3 estándar para estilos. Aunque SASS ofrece características avanzadas, CSS3 es suficientemente poderoso para este proyecto y no depende de herramientas adicionales que podrían fallar en el entorno de desarrollo.

Elección de Dependencias:

- Nodemon: Herramienta que reinicia automáticamente el servidor de Node.js al detectar cambios en el código, mejorando la productividad durante el desarrollo.

Frontend:

- React-Icons: Colección de iconos populares para mejorar la interfaz de usuario.

Estas decisiones fueron tomadas para asegurar que el proyecto se desarrollara de manera eficiente, segura y escalable, utilizando herramientas y tecnologías robustas y bien soportadas en la industria. Además considerando que no es un trabajo tan robusto y que era para desarrollarse en un corto periodo de tiempo.

Mejoras

Para la parte de desarrollo sería bueno implementar:

- Autenticación en 2 pasos: para aumentar la seguridad del sistema, hacer más eficiente el inicio de sesión para los usuarios, un mejor trafico de datos.
- Recuperación de contraseña: siempre hay que tener en cuenta que los usuarios no son perfectos y necesitas de ayudas y soluciones dentro del sistema y esta situación es muy común que suceda.
- Mejora de interfaz de usuario: utilizando recursos como Tailwind o Bootstrap, sass no lo recomiendo porque es demasiado pesado y consume infinidad de recursos y si lo que queremos es eficiencia y rápida respuesta no sería la solución.
- Integración con calendario: Permitir a los usuarios ver y gestionar sus tareas en un formato de calendario, facilitando la planificación y organización, además de hacer más útil nuestra aplicación.

La planificación para el desarrollo seria dependiendo de los requisitos del proyecto y la escalabilidad de este a futuro, no es posible repartir tareas de funcionalidades que ni son reales ni el cliente necesita. Partiendo de ahí sería medir con la escala de Fibonacci el esfuerzo que amerita cada tarea para llegar y cumplir los objetos y necesidades del proyecto.