Documentación de notes-app

Aplicación desarrollada en un stack MERN

Creación del servidor

Paso 1: Inicio del Proyecto

Creamos la carpeta notes_app_sps y dentro de ella creamos la carpeta servidor, abrimos la consola e inicializamos el proyecto Node.js con npm init.

Paso 2: Instalación de dependencias

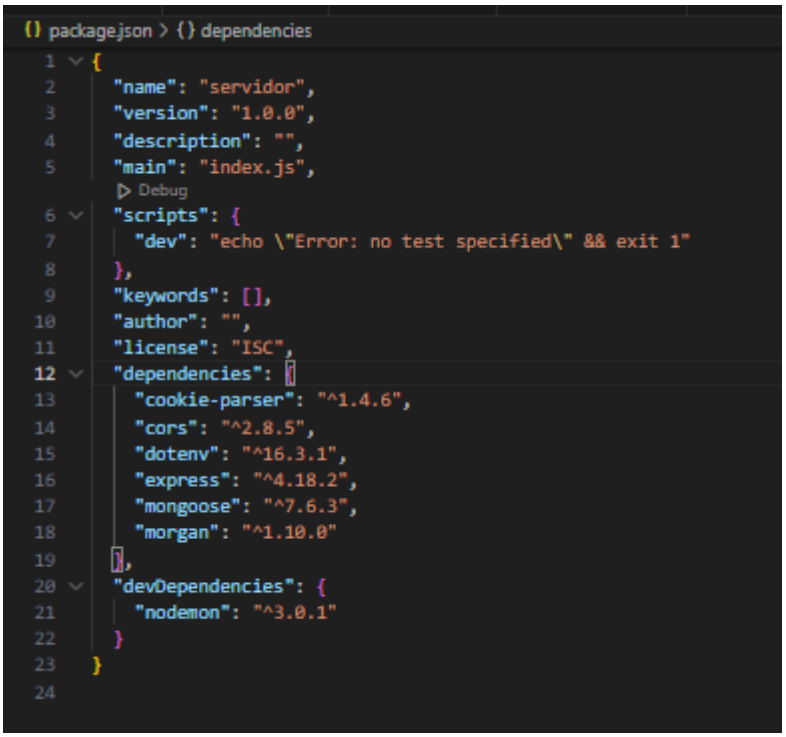
```
    "cookie-parser": "^1.4.6",
    "cors": "^2.8.5",
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "mongoose": "^7.6.3",
    "morgan": "^1.10.0"
```

Instalaremos las siguientes dependencias con el comando
“npm install cookie-parser cors dotenv express mongoose morgan”

y también instalaremos nodemon como dependencia de desarrollo usando

“npm install nodemon -D”

nos deberá quedar un package.json asi

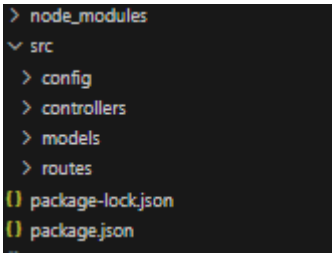


Paso 3: configuración Inicial
En el mismo package json haremos dos cambios, añadimos al archivo
"type": "module",

y en la parte de scripts añadimos el script
"dev": "nodemon src/app.js"

Paso 4: Creación de carpetas
Crearemos la carpeta src, y dentro de ella las carpetas, config, models, controllers y routes

nos debe quedar la estructura del proyecto así:



Paso 4: Conexión con mongodb
Crearemos un archivo llamado variables.env en el cual pondremos la url de la base de datos

MONGO_DB_URL=mongodb+srv://auth:1234@stlbackend.tns0hoa.mongodb.net/notesapp

Dentro de la carpeta config se crea un archivo llamado db.js en el ponemos toda la lógica necesaria para realizar una conexión con la base de datos.

Paso 5: Configuración de express y mongoDB
Iniciaremos creando el archivo app.js dentro de la carpeta src, en ese archivo se creara la lógica principal de la aplicación, como la conexiòn con express, la conexión con las rutas, y el uso de Cors, y Morgan, para iniciar el servidor lo correremos en el puerto 3000.

Cors lo usamos para evitar el error cuando, el cliente de react consuma el servidor, en el deberemos poner el url de donde esta alojado el cliente.

Morgan lo usamos para que nos imprima en consola las diferentes peticiones http que le llegan al servidor.

Paso 6: Creación de los modelos
Nos vamos a la carpeta models y creamos el archivo user.js y note.js en ellos crearemos el modelo de mongodb

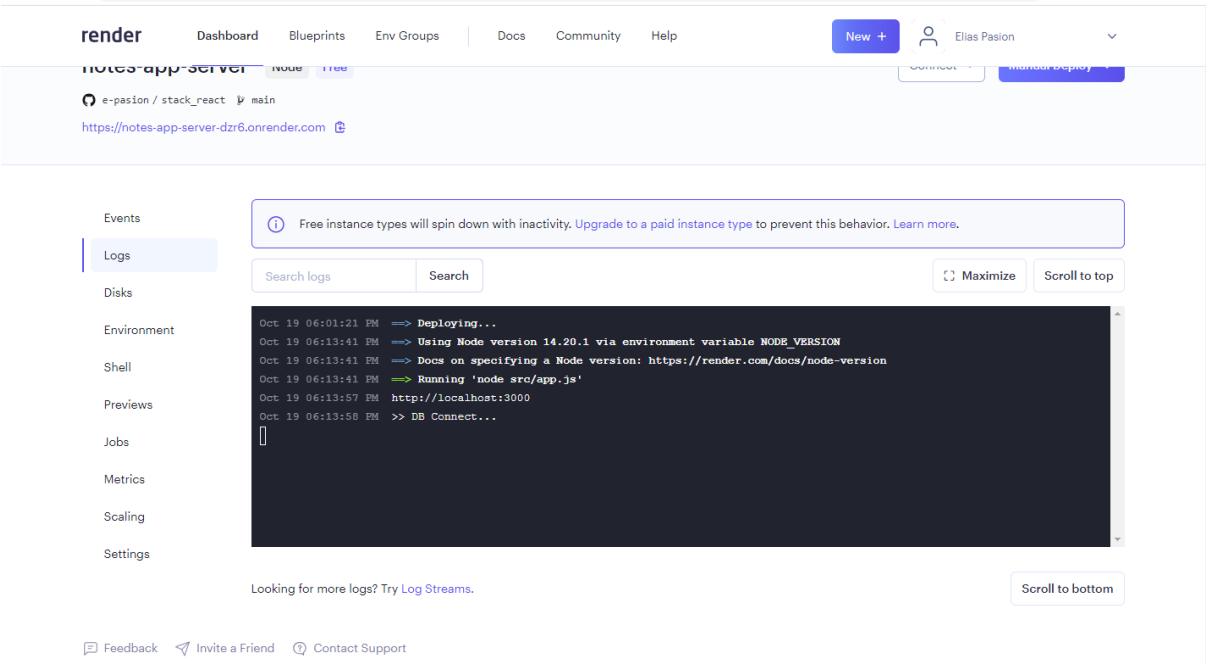
En la de user únicamente irá el nombre del usuario

En la de note, irá el título, el contenido, la fecha que por defecto será ahora y tendrá una relación con user, para que cada nota tenga un usuario

Paso 7: Creación de los controller
Crearemos en la carpeta controller los archivos user.controller.js y note.controller.js en los cuales se usará la misma lógica, se realiza un método para obtener todo, un método para obtener un solo elemento usando el ID, un método para crear, un método para actualizar usando el ID y un método para borrar usando el ID, esto mismo tanto para user como para note.

Paso 8: Creación de las rutas
Crearemos en la carpeta routes los archivos user.routes.js y note.routes.js en los cuales definiremos las rutas necesarias y asociamos a cada ruta a su respectivo método del controlador, el proceso es el mismo tanto en user como en note.

Paso 9: desplegar el servidor
El servidor quedará subido en render, para subirlo hay que entrar en la página y darle acceso al repositorio, automáticamente nos dara un link para acceder a el



en nuestro caso el link sera:
https://notes-app-server-dzr6.onrender.com/

Documentación cliente

Paso 1: instalación inicial

Instalaremos react con vite usando el siguiente comando
npm create vite@latest notes_app_sps -- --template react

luego instalaremos tailwind usando
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p

y también react-router-dom, react hook form y axios
npm install react-router-dom axios react-hook-form

Paso 2: Crear Pages
Dentro de src creamos la carpeta Pages
Donde crearemos los siguientes archivos
NotePage.jsx
NoteFormPage.jsx
UserPage.jsx
En cada uno creamos el diseño según lo definido en la práctica

también crearemos la carpeta components para añadir un navbar

Paso 3: Creación de rutas
Iremos a app.jsx y crearemos las rutas de la siguiente forma

```jsx
function App() {

  return (
    <>
    <BrowserRouter>
      <Navbar/>
      <Routes>
            <Route path='/' element={<NotePage/>}></Route>
            <Route path='/notes' element={<NoteFormPage/>}></Route>
            <Route path='/users' element={<UserPage/>}></Route>
      </Routes>
      </BrowserRouter>
    </>
  )
}

export default App
```

Paso 4: Crear conexión con el backend
Dentro de src crearemos la carpeta api donde crearemos el archivo axios.js y también crearemos note.js y user.js

```
import axios from 'axios'

const instance= axios.create({
    baseURL: 'https://notes-app-server-dzr6.onrender.com/api_notes_app/',
    withCredentials:true
})
export default instance
```

```
import axios from "./axios";


export const getNotesRequest= () => axios.get(`/notes`)
export const getNoteRequest=id => axios.get(`/notes/${id}`)
export const createNoteRequest=note => axios.post(`/notes`,note)
export const updateNoteRequest=(note) => axios.put(`/notes/${note._id}`,note)
export const deleteNoteRequest=id => axios.delete(`/notes/${id}`)
```

Paso 5 editar los componentes
Cada componente se editó con lo requerido en el documento,
en el de users se añadió la funcionalidad de crear y listar usuarios
en el task form la función de listar usuarios, traer una nota,crear y editar notas
y en el task la función de listar notas y borrarlas

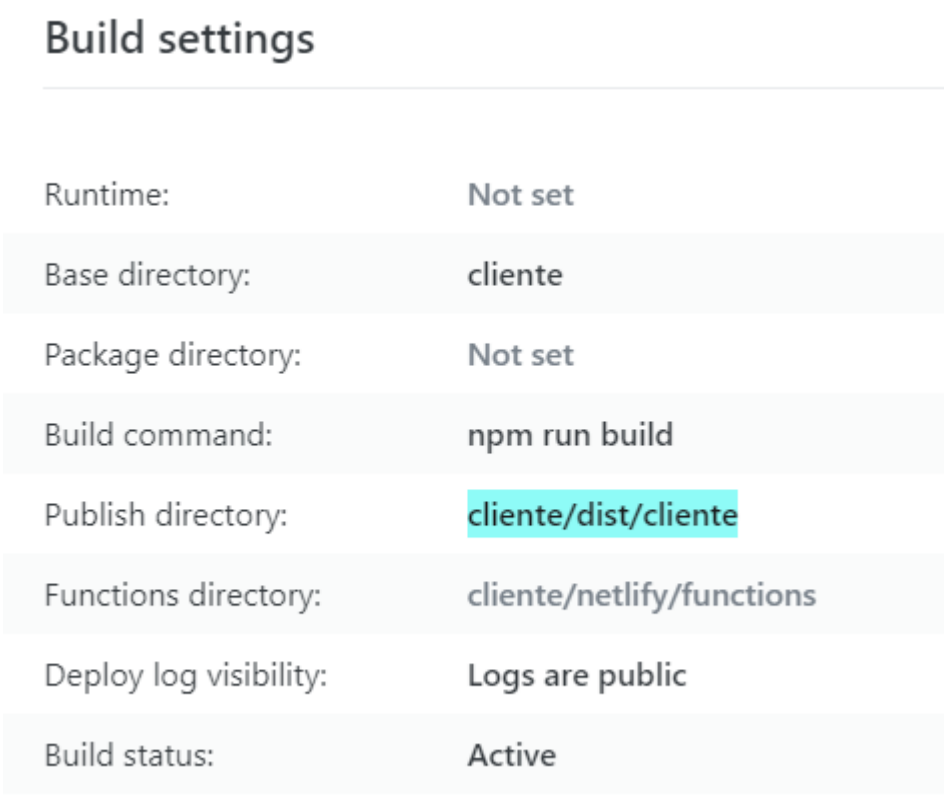
solo hubo una dificultad y fue al momento de convertir la fecha estandar de mongo db a lo requerido en las tarjetas como 1 week ago o in 3 years, para lo cual se creó el siguiente algoritmo que transforma la fecha

```
function calculateTime(date) {
    const currentDate = new Date();
    const targetDate = new Date(date);
    const differenceInMilliseconds = currentDate - targetDate;
    let seconds = Math.floor(differenceInMilliseconds / 1000);
    const negative = (seconds<0)?true:false;
    seconds=Math.abs(seconds);
    let time=0;
    let sufix;

    if (seconds < 60) {
        time=seconds;
        sufix=`second${(time>1)?'s':''}`
    } else if (seconds < 3600) {
            time=seconds/60
            sufix=`minute${(time>1)?'s':''}`
      } else if (seconds < 86400) {
            time=seconds/3600
            sufix=`hour${(time>1)?'s':''}`
        } else if (seconds < 604800) {
            time=seconds/86400
            sufix=`day${(time>1)?'s':''}`
      } else if (seconds < 2419200) {
            time=seconds/604800
            sufix=`week${(time>1)?'s':''}`
      }  else if (seconds < 29030400) {
        time=seconds/2419200
        sufix=`month${(time>1)?'s':''}`
      } else {
        time=seconds/29030400
        sufix=`year${(time>1)?'s':''}`
      }

    if(negative) return `in ${time.toFixed(1)} ${sufix}`
    return `${time.toFixed(1)} ${sufix} ago`;
  }
```

Paso 5 despliegue en netlify
lo primero que haremos será correr el comando npm run build para generar la carpeta que será desplegada en el sitio web

luego vamos a https://app.netlify.com/

donde traeremos el repositorio de nuestro github y ponemos las siguientes opciones



y ya con eso tendremos el servidor del cliente funcionando en el siguiente link

https://heartfelt-shortbread-5effc2.netlify.app/