Memoria Examen Parcial 3 Enero – Mario Merino Zapata

Backend en puerto 5000

Frontend en puerto 5173

URLs de la aplicación

URL del frontend:

https://ing-web-parcial3.vercel.app

ing-web-parcial3.vercel.app

Tecnologías utilizadas

- **Node.js**

Runtime de JavaScript

- **Express.js**

Express.js es un framework de desarrollo de JavaScripts, utilizado para desarrollar el backend de nuestra aplicación.

- **MongoDB Atlas**

Servicio en la nube utilizado para gestionar y alojar la base de datos MongoDB, garantizando escalabilidad, seguridad y disponibilidad global.

- **Docker**

Para el despliegue de los microservicios y la base de datos local.

- **Vue.js**

Framework de JavaScript progresivo para construir interfaces de usuario dinámicas y basadas en componentes.

- **Cors**

Es un mecanismo de seguridad que controla las solicitudes HTTP que se realizan desde un script que está fuera del servidor. De forma que si accedes a otro dominio desde un sitio web, ese nuevo dominio no puede realizar solicitudes a tu cuenta sin autorización.

- **Tailwind CSS**

Tailwind CSS es un framework de utilidades para CSS que permite construir diseños modernos y personalizados directamente en el HTML, acelerando el desarrollo frontend

- **Vercel**

Plataforma utilizada para el despliegue del frontend, optimizada para aplicaciones web modernas y rápidas.

- **Railway**

Plataforma empleada para alojar los microservicios del backend

Algunas de las librerías importantes utilizadas

- **Nodemon**

Empleada para depurar más fácilmente el código, ya que relanza la aplicación cada vez que guardas un fichero.

- **dotenv**

Usada para manejar variables de entorno, como la URI de conexión a la base de datos.

- **Mongoose**

Utilizada para acceder a nuestra base de datos de MongoDB.

- **Axios**

Utilizada para realizar solicitudes HTTP entre microservicios y comunicar distintas API de forma eficiente.

- **Easymde**

Un editor de Markdown sencillo y fácil de usar que permite a los usuarios escribir y dar formato a texto de manera intuitiva.

- **marked**

Intérprete de markdown.

- **Multer**

Es un middleware de Node.js que se encarga de manejar las subidas de archivos por medio de formularios. Esto la hace especialmente indicada para subir imágenes.

- **Streamifier**

Transforma un buffer en un flujo legible, lo que facilita el procesamiento de grandes cantidades de datos. De nuevo, especialmente útil para manejar imágenes.

- **Cloudinary**

Es una plataforma que permite almacenar imágenes y vídeos en la nube. Como muchas otras, tiene una API a la que se accede mediante la biblioteca del mismo nombre.

Acesso a la base de datos con mongo

La base de datos utilizada en el proyecto es MongoDB Atlas, una solución en la nube que ofrece alta disponibilidad y seguridad. La conexión a la base de datos se realiza mediante el siguiente connection string, que puede ser configurado en el entorno del proyecto:

mongodb+srv://admin:admin@frontend-cluster.imsr6.mongodb.net/?retryWrites=true&w=majority&appName=frontend-cluster

Este connection string permite gestionar la base de datos de forma remota y segura, manteniendo los datos centralizados en un entorno escalable.

Arquitectura y esquema de navegación

Despliegue en local

Seguimos los siguientes pasos:

1. Clonar el proyecto del repositorio GitHub correspondiente, o bien descomprimir el zip de la entrega con el código fuente

2. Hay que crear un archivo .env en el directorio backend/ que contenga las mismas variables de entorno que .env.example

[Aunque ya lo he adjuntado a la entrega, lo copio aquí también para mayor claridad]:

MONGODB_URI = "mongodb+srv://admin:admin@frontend-cluster.imsr6.mongodb.net/?retryWrites=true&w=majority&appName=frontend-cluster"

CLOUD_NAME = "dygacvfuj"

CLOUDINARY_API_KEY = "592149721219436"

CLOUDINARY_API_SECRET = "2t3Qggi7emirX6CXAYtO8HRJqLY"

CLOUDINARY_URL= "cloudinary://592149721219436:2t3Qggi7emirX6CXAYtO8HRJqLY@dygacvfuj"

3. Ya que estamos en el directorio backend/, hacemos 'npm install' y 'npm run dev'. Esto iniciará nuestro backend en el puerto 5000

4. Volvemos ahora al root y hacemos lo mismo, pero esta vez accediendo previamente al directorio frontend: 'npm install' y 'npm run dev' (será necesaria otra terminal para no terminar el proceso del backend).

5. El frontend estará operativo en el puerto por defecto de vue, http://localhost:5173/

Otro despliegue en local posible por si unos pasos no se entienden:

1. Asegurarse de que Docker Desktop (y Docker engine) se encuentren instalados y en funcionamiento, así como Node.js.
2. Descomprimir el zip de la entrega con el código fuente, o bien clonar el proyecto del repositorio GitHub correspondiente.

   https://github.com/MarioMerZap/IngWebParcial3.git

3. Hacer cd al root del proyecto, (normalmente al clonarlo, hacer

cd IngWebParcial3/

siendo este el nombre de la carpeta del proyecto). En caso de haber descomprimido el zip, podemos seleccionar la carpeta > click derecho > open in terminal

4. Una vez estamos en la root del proyecto, debemos copiar el contenido de '.env.example' a un archivo al que debemos llamar '.env' si este no existe. En caso de que la terminal usada lo permita, podemos hacer:

   cp .env.example .env

Habrá que completar las variables de entorno, sustituyendo los valores de las claves de cloudinary y resend por sus valores reales.

(Para facilitar el proceso, se adjunta a la entrega un .env completo que se puede usar directamente, con todas las variables de entorno configuradas correctamente)

este .env será el que utilice nuestro docker-compose para las variables de entorno.

5. Una vez tenemos nuestro .env, simplemente ejecutamos nuestros contenedores con

docker compose up -d

[para detenerlos, docker compose down]

esto pulleará la imagen de una instancia de mongoDB oficial desde dockerhub, y construirá los contenedores del backend de nuestra aplicación.

6. La aplicación estará en funcionamiento en los puertos definidos en el .env

7. Para ejecutar el frontend, nos vamos a la carpeta "frontend/" y hacemos lo mismo, copiamos el .env.example a un archivo .env (en este caso no hace falta modificarlo, puesto que estamos usando URLs locales que ya se encuentran definidas correctamente, se adjunta también el .env por conveniencia).

8. Ejecutar npm install y npm run dev

9. Acceder al frontend de la aplicación con ***http://localhost:5173***

Modelos del backend:

En el backend solo he creado mensajes el cual permitirá guardar en la base de datos los siguientes datos:

de: { type: *String*, required: true }, // ID del usuario que envía

para: { type: *String*, required: true }, // ID del destinatario

asunto: { type: *String*, required: true },

contenido: { type: *String*, required: true },

adjunto: { type: *String*, default: null }, // URL de la imagen adjunta

token: { type: *String*, required: true }, // Token del usuario autenticado

stamp: { type: *Date*, default: *Date*.now }

El motivo de esto es que solo he creado una entidad en el backend y cambiado cosas, que por desgracia creo que han llevado a romper otras funcionalidades.

Lo que hace mi programa sería entrar a una bandeja de correos donde puedes seleccionar uno y responderlo llevándote al crear mensaje correspondiente o puedes pulsar el botón de enviar mensaje y crear uno nuevo.

Para el login usaré Google y para acceder a el deberá ir a el enlace/login ya que no me dio tiempo a añadir un botón.

Aclaraciones sobre posibles problemas:

Me ha dado momentáneamente un problema algo raro donde al iniciar sesión con oauth la url me decía ui mismatch pero al darle atrás y volver a intentarlo se solucionó.

Me da problema al enviar el mensaje pues necesita imagen pero me da problema con el cloudinary, problema que no me dio en el anterior parcial pero en este sí porque no he podido revisar el backend, fuera de eso la bandeja de entrada aparece llena de correos vacíos pues no he podido crear un seeding pero como tal se ven claras las funciones