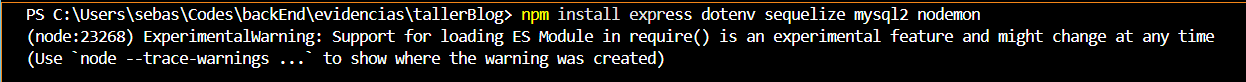
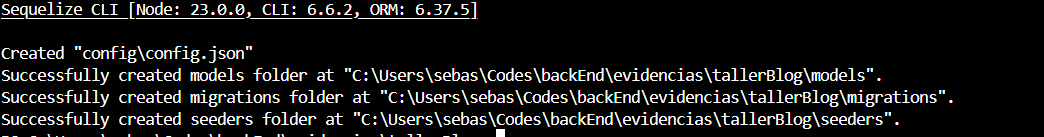
1. Iniciamos el proyecto y configuramos las dependencias ejecutando el comando: `npm install express dotenv sequelize mysql2 nodemon`.



1. Iniciamos Sequelize-cli



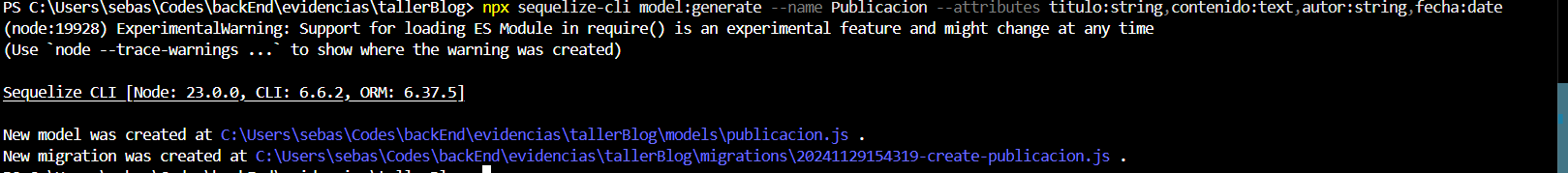
1. Crear base de datos



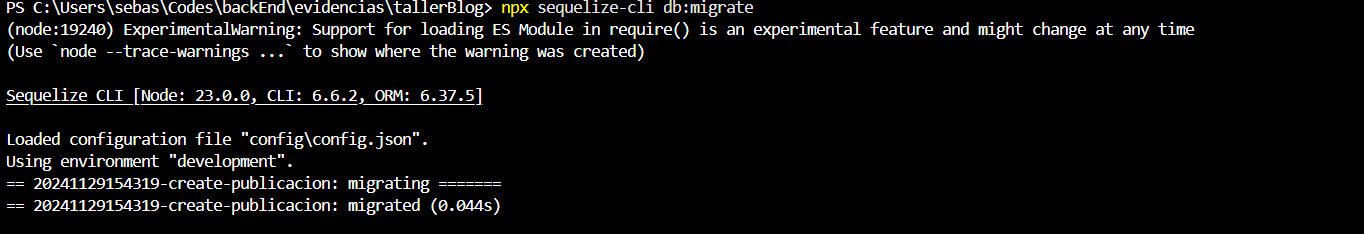
1. Establecemos nuestras credenciales en el archivo `config.json`.



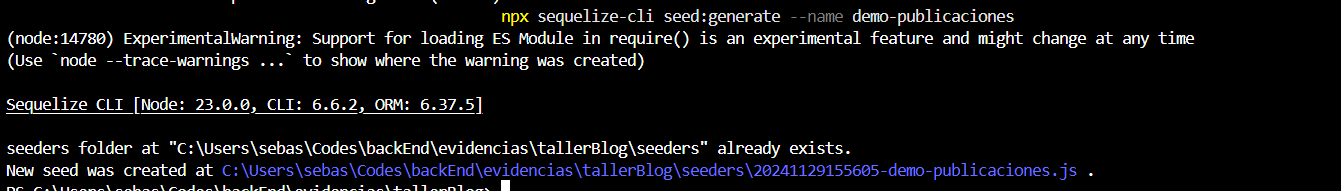
1. Generamos el modelo de la tabla para nuestra base de datos SQL utilizando el comando: `npx sequelize-cli model:generate --name Publicacion --attributes titulo:string,contenido:text,autor:string,fecha:date`.



1. Transferimos la estructura de la tabla a nuestra base de datos ejecutando el comando: `npx sequelize-cli db:migrate`.



1. Generamos un seed para datos de ejemplo utilizando el comando: `npx sequelize-cli seed:generate --name demo-publicaciones`.



1. Editamos los campos en el archivo del seeder para definir los datos que se enviarán a la base de datos.

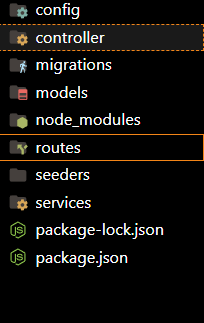


1. Cargamos los seeds en la base de datos ejecutando el comando:  
   npx sequelize-cli db:seed:all.

Texto

Descripción generada automáticamente

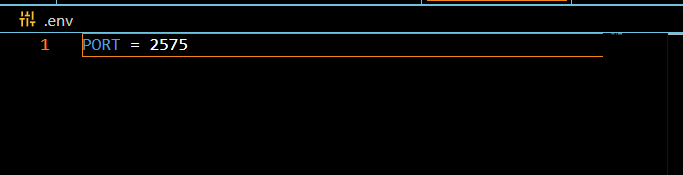
1. Organizamos el proyecto creando las carpetas: controller, routers y services.



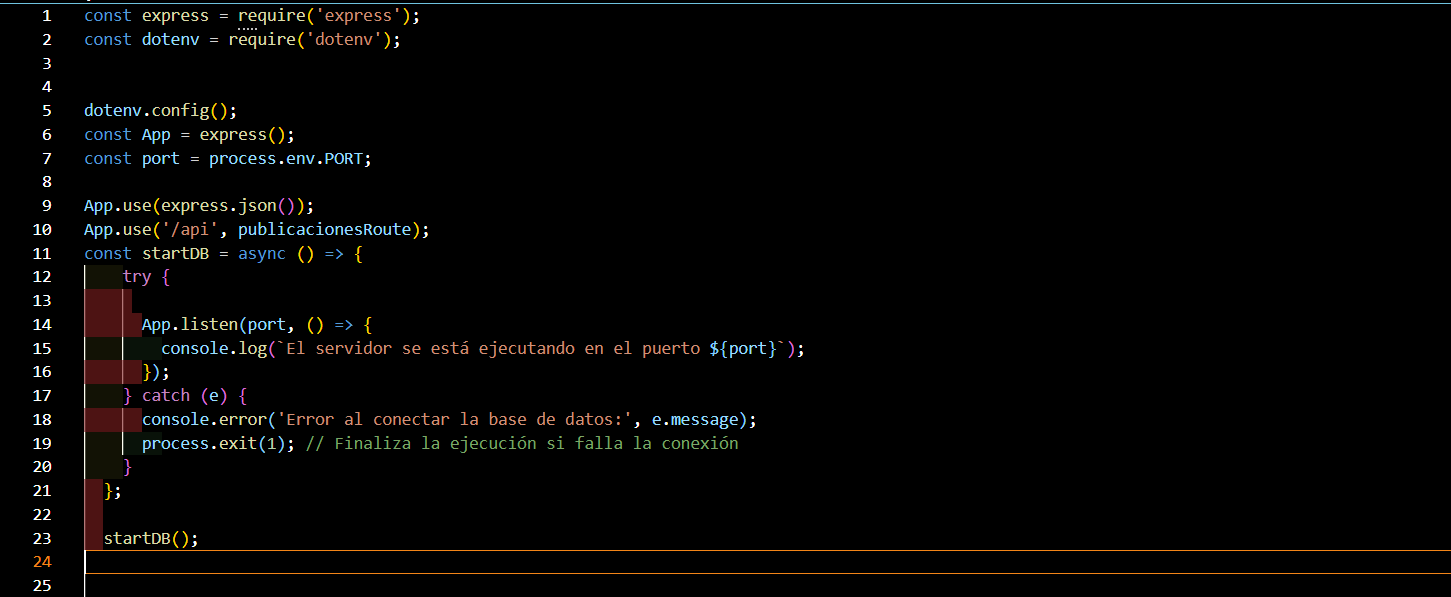
1. Configuramos el proyecto creando el archivo .env para las variables de entorno y el archivo principal de la aplicación, index.js.



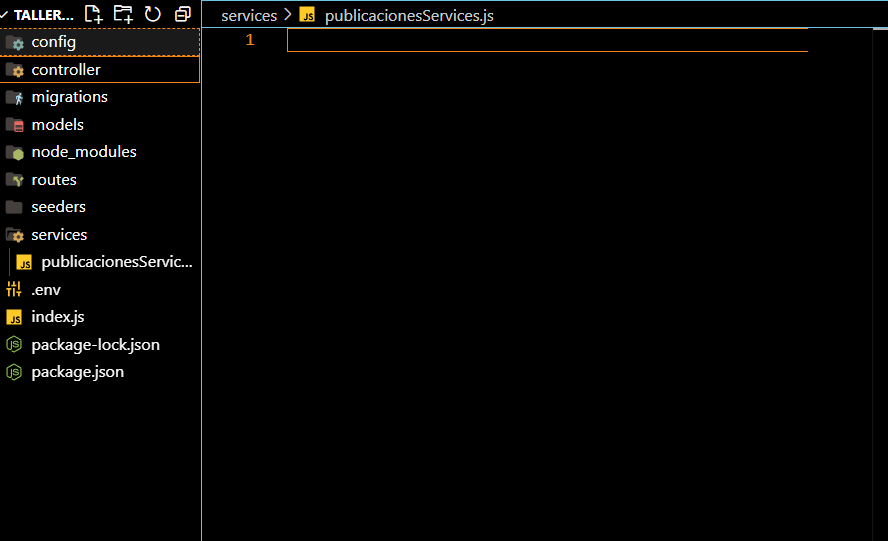
1. Establecemos las variables necesarias en el archivo .env para configurar el entorno de la aplicación.



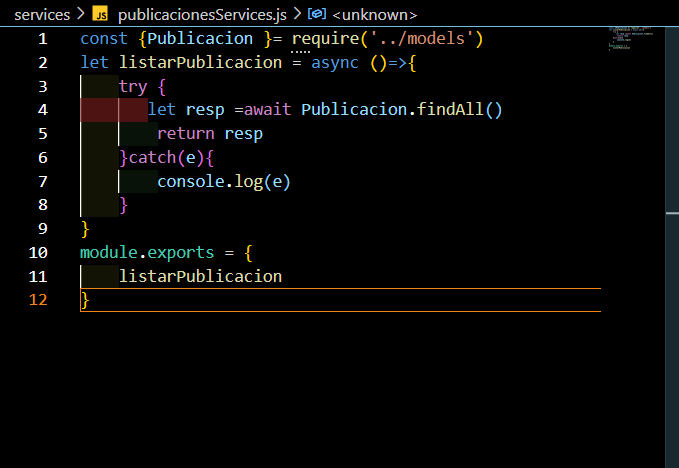
1. Configuramos el archivo principal index.js para iniciar la aplicación y definir su comportamiento básico. Esto incluirá la configuración de Express, la conexión a la base de datos, y la integración de los controladores y rutas.



1. Creamos el archivo publicacionesService.js en la carpeta services para gestionar la lógica de negocio de las publicaciones, como la creación, obtención, actualización y eliminación de las mismas.



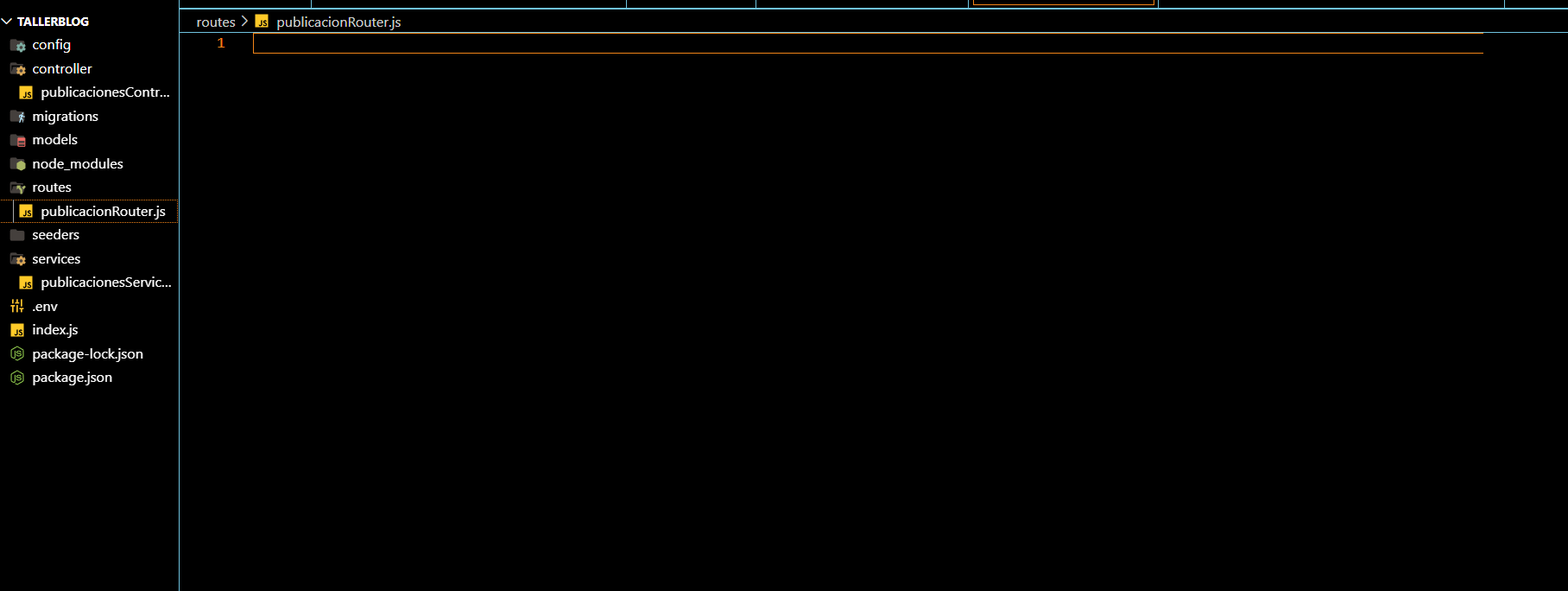
1. Añadimos la lógica al archivo para recuperar todas las publicaciones almacenadas en la base de datos y exportamos la función listarPublicacion.



1. Creamos el archivo publicacionesController.js en la carpeta controller para gestionar las solicitudes relacionadas con las publicaciones. Este archivo se encargará de interactuar con el servicio de publicaciones y devolver las respuestas correspondientes.
2. Importamos la función listarPublicacion desde publicacionesService y añadimos la lógica al archivo para manejar las solicitudes que devuelvan las publicaciones guardadas en la base de datos. Luego, exportamos la función listarPublicaciones.



1. Creamos el archivo publicacionesRouter.js en la carpeta routers para definir las rutas relacionadas con las publicaciones, gestionando las solicitudes HTTP y redirigiéndolas al controlador adecuado.



1. Importamos la función listarPublicaciones desde la ruta controllers/publicacionesController y exportamos el router.



1. Importamos publicacionesRoute desde la ruta ./routes/publicacionRoute y luego agregamos las rutas para las consultas en Postman utilizando:  
   app.use('/api', publicacionesRoute).



1. Luego, ejecutamos el servidor con el comando:  
   npm run dev.

Texto

Descripción generada automáticamente

1. [Finalmente, realizamos la consulta en Postman utilizando la URL:](http://localhost:2575/api/publicacion)http://localhost:2575/api/publicacion.

