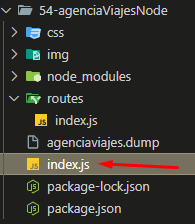
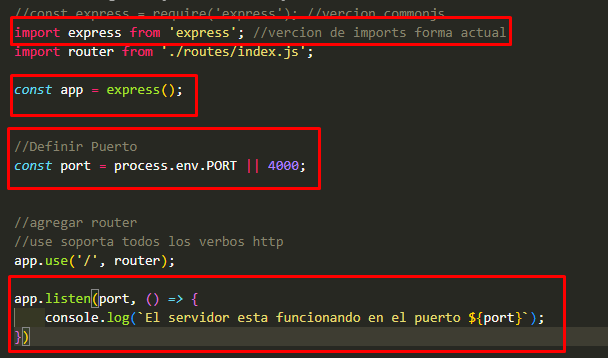
1. Instalar node
2. Instalar dependencias



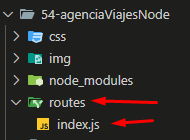
1. Crear archivo index.js y realizar su configuración

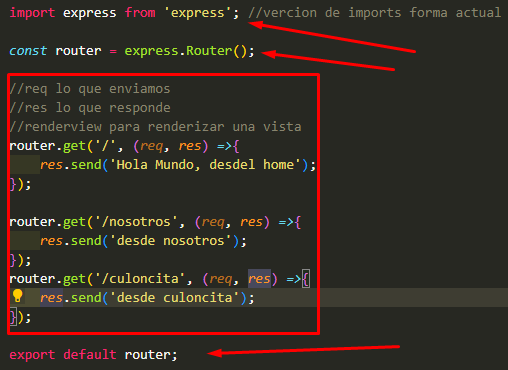


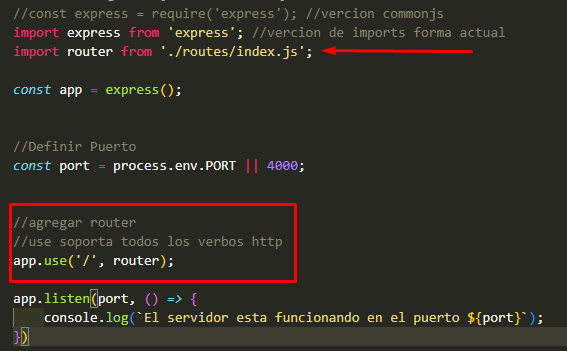
1. Configurar el index.js



1. Generar el routing, una carpeta con las rutas y crearle un archivo index.js



1. Importar express y generar el routing y exportar,
2. Importar y hacer uso del archivo importado

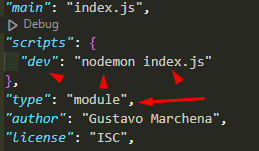


Nota: cuando se utiliza import es necesario realizar una configuración de modulos en el package.json, si no este no funciona, cuando utilizamos commonjs como se muestra en la imagen y esta comentado no es necesario la configuración.

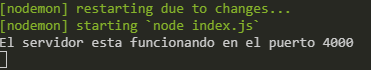
1. Para ejecutar el proyecto tenemos que tener instalado un watcher que este mirando por los cambios este es un paquete que se llama nodemon



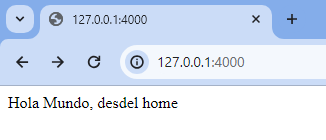
1. Realizamos la configuración de modulos, y en la ejecución del script dev colocamos el archivo que se ejecutara

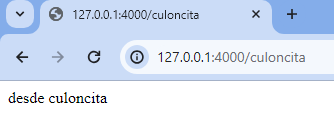


1. Tecleamos en consola npm run dev y se estaría corriendo el watcher.



Lo que veriamos en el navegador





//Template Engines

motores de plantilla

Angular, React o vue reemplazan estos template engines en una aplicacion

En node y Expess se utiliza Pug, EJS, HBS, React 'no es la libreria de react'

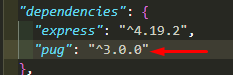
un template engine se instala via npm

npm install pug

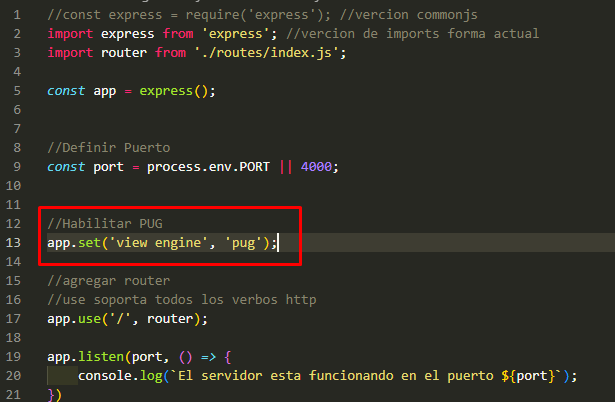
y se habilita en el archivo principal

1. Instalar pug

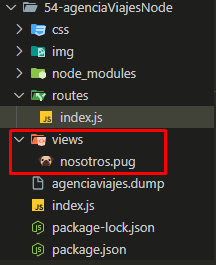
npm i pug



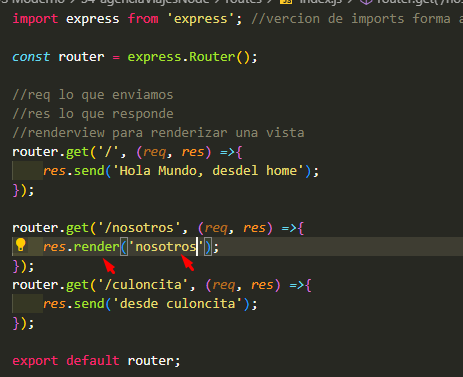
1. Habilitar Pug en el archivo index.js de la raíz donde esta toda la configuración



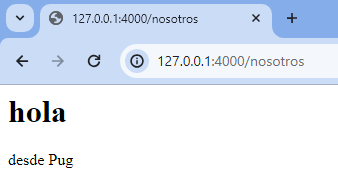
1. Creamos la vista en una carpeta view y el archivo que contendrá el html



1. Hacemos el llamado de la vista a renderizar en el routing, donde nosotros en el res.render(nosotros) es el nombre del template



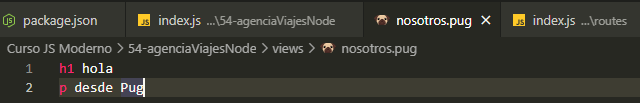
1. La salida



Nota: el motor de plantillas de pug escribe un html mas corto

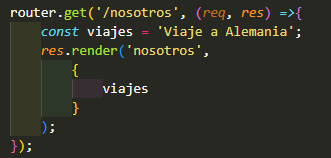
H1 sin abrir ni cerrar etiqueda

P sin abrir ni cerrar etiqueta

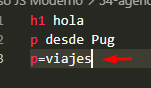


Pasando valores asi la vista

1. Después del nombre de la vista pasamos un objeto, podemos hacer uso de objet literal enjansmen



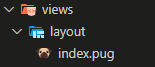
En pug a la etiqueta html se le asigna el valor



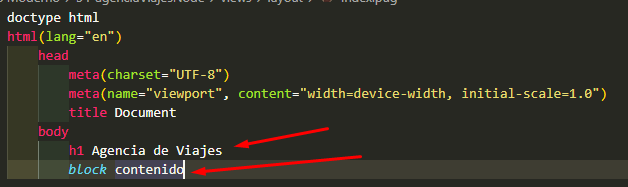
La otra forma de pasar una variable es sin el igual p #{viajes}

Pagina maestra o template principal, es un template genérico que contendrá una estructura y solo la llamamos para que esta tenga lo general, es como en laravel cuando aplicamos yiel o en Django.

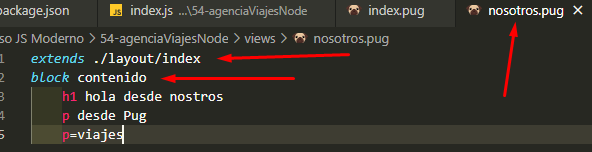
1. Crearmos la carpeta la carpeta layout dentro de views y creamos un archivo index.pug



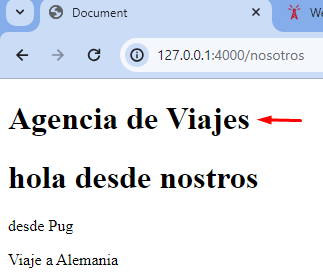
El conteniod del archivo será la estructura html la etique que aparecerá en todos los archivos y un bloque



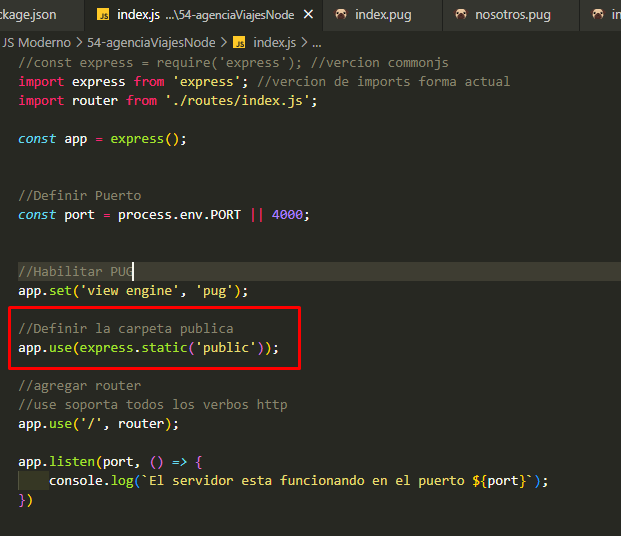
1. Ahora hay que mandar a llamar dicho archivo en los archivos que mostraran los datos.

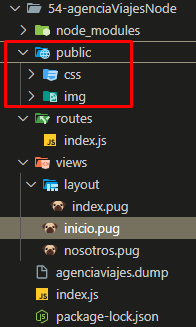


Agencia de Viajes es el texto que viene desde la plantilla

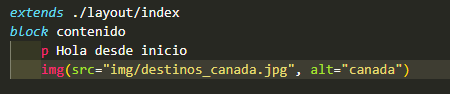


Definir carpeta publica para hacer uso de archivos staticos

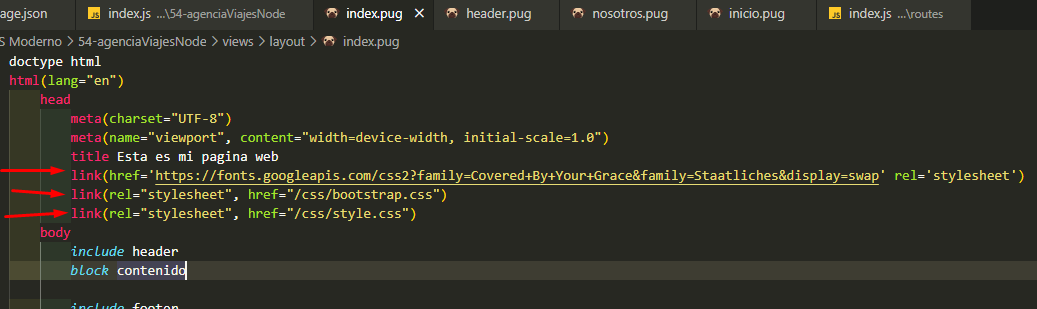




Como llamar a los archivos



Agregando CSS externo y agregando fuente de Google fonts



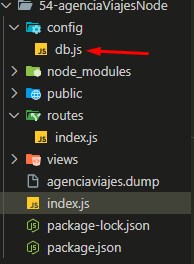
include importa archivos completos

block solamente ciertas bloques de codigo que tu definas.

Conexión a una base de datos descargar

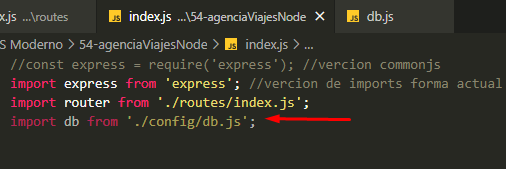
npm i mysql2@2.2.5 [sequelize@6.3.5](mailto:sequelize@6.3.5)

1. Generar la conexión





Importamos el archivo de configuración para hacer uso de la conexión

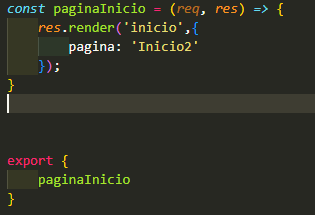


Agregando Controller

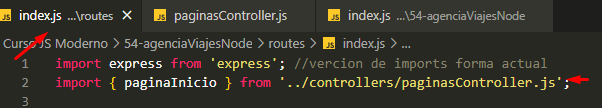
1. Crear carpeta de controllers



De la parte que le habíamos pasado que renderizaba el .pug ese código lo pasamos en el controlador



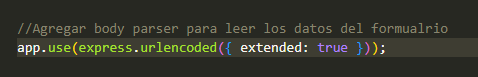
Importamos el controlador



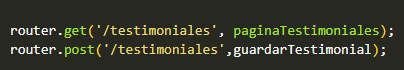
Pasamos el controlador



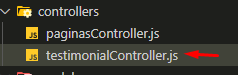
Activar urlencoded para recibir datos via post



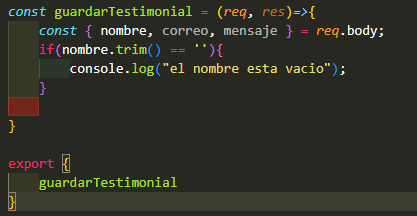
Crear la url via post



Crear carpeta y archivo que gestionara el controlador

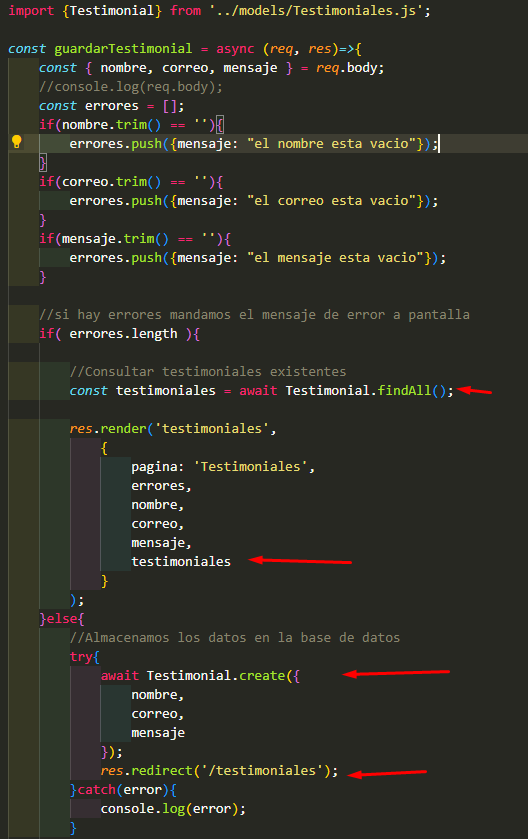


Comenzar con la validación

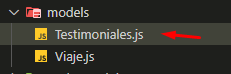


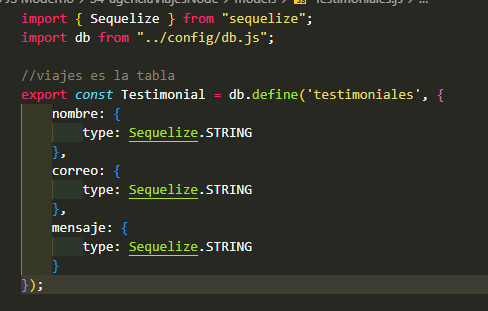
En express existe una librería que se llama express valiator pero para este caso de aprender es mejor hacerlo a manita

Aquí vemos el código completo de la consulta a la base de datos, en una verificamos si hubo error mostramos los testimoniales que existan en la base de datos, en la otra creamos los datos y redireccionamos a la vista

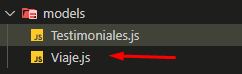


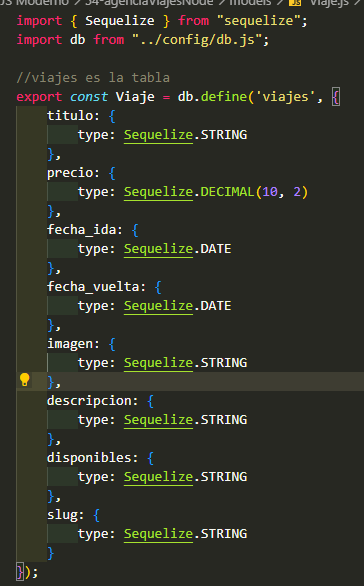
El modelo para guardar los datos





Asi como este modelo también se creo un modelo para los viajes, cuando se crean, y para realizar select y estos nos traigan dichos datos.

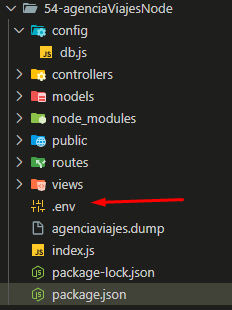




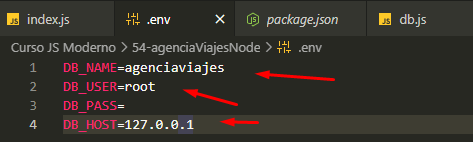
//dotenv

libreria para poner las variables de entorno

creamos el archivo .env

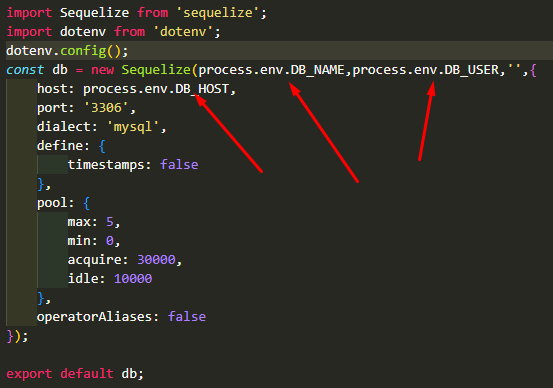


Pasamos los datos de conexión al archivo

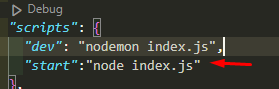


Llamamos a los valores, primero importamos la librería, luego dotenv.config();

Y por ultimo pasamos los datos



Configurar deploy con node, se crea ese start para que el servidor inicie con node y no con nodemon



Deploy

