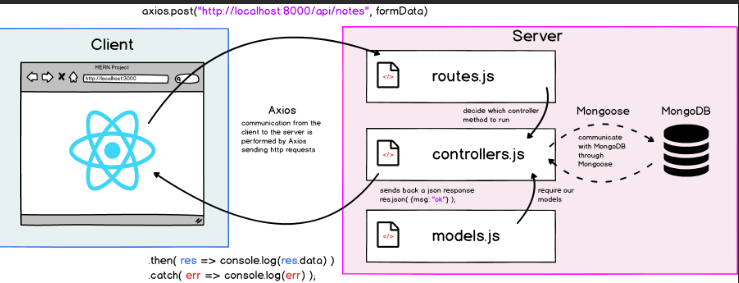
Hasta este momento, hemos estado separando por completo nuestro front-end y backend. Cuando usamos create-react-app, usamos el comando npm run startpara servir nuestra aplicación React. Sin embargo, queremos incluir React dentro de nuestro proyecto Express.

La forma en que haremos esto es creando nuestro proyecto React para que cree los archivos HTML, CSS y Javascript que necesitamos para nuestra SPA. Entonces, podemos señalarlo dentro de nuestro proyecto Express. En las siguientes lecciones, aprenderemos cómo configurar nuestro proyecto Full Stack MERN y vincular todas las tecnologías.

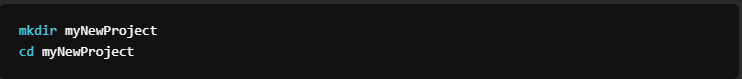
Nuestras aplicaciones MERN utilizarán la siguiente arquitectura:



Vamos a utilizar Axios en nuestro proyecto React para comunicarnos con nuestro servidor. Es importante recordar que a React no le importa el lenguaje que se ejecuta en nuestro servidor. React simplemente hará peticiones a nuestro servidor y obtendrá respuestas.

Full Stack MERN

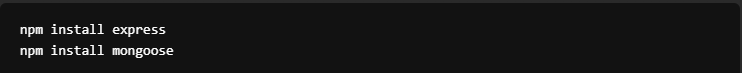
Pasemos directamente a la creación de nuestro proyecto MERN de estructura completa. Primero, crea una nueva carpeta llamada "myNewProject" y cden ella.



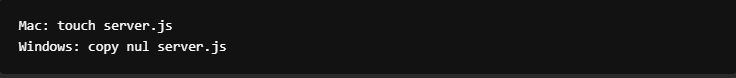
A continuación, crea un nuevo proyecto a través de:



Esto creará el package.json para nuestro servidor. Luego tendremos que instalar nuestras dependencias:



Después, a través del terminal o la interfaz de usuario, crea un nuevo archivo llamado server.js.



Luego, dentro de server.js agrega el siguiente código:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Creemos nuestra estructura de proyecto modularizada haciendo una carpeta llamada "servidor" y luego creemos cuatro carpetas más dentro de esa llamada "config", "controllers", "models" y "routes".

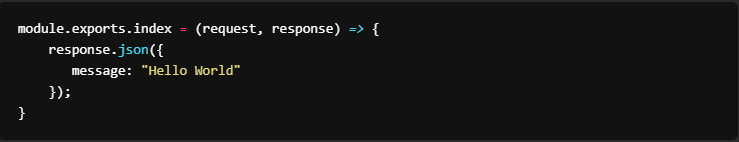
Así es como creamos la estructura del proyecto para nuestro backend. Ahora, creemos nuestro proyecto React vía create-react-app. Como React se usa para el código del lado del cliente, podemos llamar a nuestro proyecto "client". Asegúrate de estar en el mismo nivel de carpeta que el archivo "server.js".



Ahora que has construido tu proyecto React, ejecutarás dos servidores diferentes: tu servidor React front-end con recarga en vivo y tu servidor Express.

Ahora que tenemos la configuración básica, creemos el resto de nuestro proyecto en forma de una aplicación MERN de estructura completa "Hello World". Agreguemos un controlador dentro de la carpeta 'controladores', llamado person.controller.js.

controllers / person.controller.js



A continuación, en la carpeta de "routes", creemos el archivoperson.routes.js.

routes / person.routes.js

Texto

Descripción generada automáticamente

Ahora, tenemos una ruta que termina en 'api' que simplemente devolverá un objeto con un mensaje igual a "Hello World". Enlacemos esto en nuestro server.js:

server.js

Texto

Descripción generada automáticamente

Ahora, cuando visitemos 'localhost: 8000 / api', obtendremos una respuesta JSON de {message: "Hello World"}.

Tenemos nuestro back-end configurado, pero vamos a mostrar esto en nuestro front-end.

Comencemos ahora a configurar nuestro proyecto React. Cambia los directorios a tu proyecto React, llamado client, y ejecuta lo siguiente:



Estamos instalando axios para que podamos hacer una solicitud fácilmente a nuestro backend. Dentro de la carpetasrc, crea un nuevo archivo llamado Main.js. En el archivo Main.js, haremos una llamada API y mostraremos nuestro mensaje. Usaremos el ganchouseEffect para hacer la llamada a la API y guardar el mensaje en el estado.

cliente / src / Main.js

Texto

Descripción generada automáticamente

Estamos configurando el mensaje predeterminado como "Loading...". A continuación, en tu archivo App.js, deberás insertar el componente Main de la siguiente manera:

Texto

Descripción generada automáticamente con confianza baja

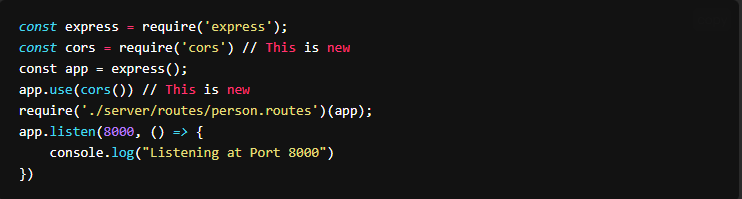
En este punto, puedes iniciar tu servidor vía nodemon server.jsy tu aplicación React vía npm run starten dos consolas diferentes. Una vez que tengas ambos en ejecución, deberías poder visitar 'localhost: 3000'. Una vez que lo visites, terminarás viendo "Loading..." como tu mensaje. ¿Porqué es eso? Esto se debe a que estamos haciendo una solicitud a nuestro servidor desde un origen diferente. Entonces, detengamos el servidor express e instalemos un paquete adicional dentro de tu servidor.

Cors



Esto instalará la capacidad de realizar solicitudes de origen cruzado. Ahora, necesitaremos cambiar nuestro server.jscomo sigue:

server.js



Ahora podemos realizar solicitudes de origen cruzado en nuestro proyecto. Cuando actualices tu aplicación React, deberías ver brevemente "Loading ..." como el mensaje, que luego será reemplazado por "Hola Mundo". ¡Felicidades! Has creado tu primera aplicación MERN. A continuación, comenzaremos a vincular nuestra base de datos.

Crear (Parte I)

Para iniciar una aplicación MERN CRUD de estructura completa, necesitaremos configurar nuestra base de datos. Agreguemos un archivo mongoose.config.js en nuestra carpeta server / config:

config/mongoose.config.js

Texto

Descripción generada automáticamente

Podemos ver que estamos haciendo referencia a nuestra carpeta de modelos. Comencemos a crear un software de gestión de relaciones con el cliente y creemos un nuevo modelo Person,.

models/person.model.js

Texto

Descripción generada automáticamente

A continuación, necesitamos importar nuestro archivo de configuración de mongoose para que encienda el servidor y nuestro nuevo archivo person.routes, podemos manejar una solicitud posterior para crear una persona. Sin embargo, para procesar las solicitudes POST, primero debemos agregar un par de declaraciones app.use con algún middleware express.

Nuestro server.js ahora tendrá algunas líneas de código más:

server.js

Texto

Descripción generada automáticamente

Ahora, en nuestro controlador, podemos agregar un nuevo método para manejar la creación de nuestra persona:

controllers/person.controller.js

Texto

Descripción generada automáticamente

Ahora, vamos a actualizar nuestras rutas:

Texto

Descripción generada automáticamente

Al agregar esta línea, ahora podemos crear personas enviando una solicitud de publicación a localhost:8000/api/persons.

Veamos el lado React de las cosas.

Crear (Parte II)

Vamos a vincular la creación a nuestro proyecto React . Dentro de la carpetasrc, agreguemos dos carpetas más: viewsy components. Luego, pasemos Main.jsa la carpeta views. Podemos cambiar nuestro Main.jspara que tenga un formulario. Dentro de nuestra carpeta components, creemos un nuevo archivo llamado PersonForm.js. Este componente será el formulario al que podemos agregar una persona y realizar una solicitud a nuestra API.

components/PersonForm.js

Texto

Descripción generada automáticamente

Ahora, tenemos un formulario funcional (aunque no muy interesante).

Solo necesitamos actualizar nuestro proyecto React para que aparezca en el DOM. Cambia tu Main.js a lo siguiente:

views/Main.js

Texto

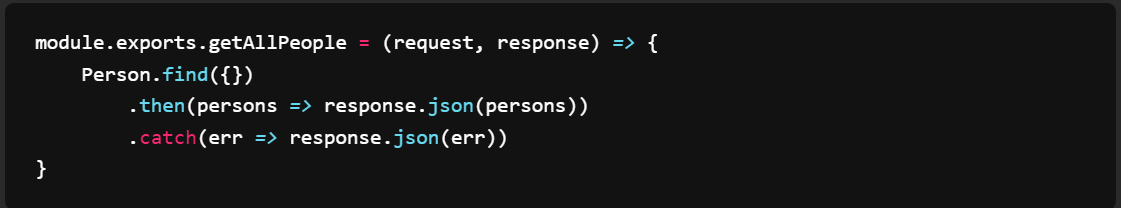
Descripción generada automáticamente

Ahora puedes probar tu formulario para ver si está funcionando. Deberías poder ver la respuesta en la consola.

Hasta ahora tenemos la capacidad de crear personas. Sin embargo, ¿cómo sabemos si estas personas están realmente en nuestra base de datos? Podríamos verificarlo manualmente. O bien, podríamos hacer una solicitud a nuestra API que recuperará todas las personas en nuestra base de datos por nosotros.

Agrega una función a tu archivo de controlador que se verá así:

server/controllers/person.controller.js



y agrega esto a nuestras rutas:

Texto

Descripción generada automáticamente

Puedes probar esto con Postman. Ahora podemos volver a nuestro front-end React. Creamos un nuevo componente llamado PersonList.js.

components/PersonList.js

Texto

Descripción generada automáticamente

Solo estamos enviando nuestras personas a views/Main.js

Texto

Descripción generada automáticamente

Estamos cargando a todas las personas en nuestra página principal, y solo enviamos la lista de personas una vez que recibimos una respuesta de la API. A continuación, necesitaremos crear una nueva vista que pueda mostrarnos los detalles de una persona en particular. Sólo tenemos su nombre y apellido, por lo que no será necesariamente tan interesante. Sin embargo, veamos cómo funciona.

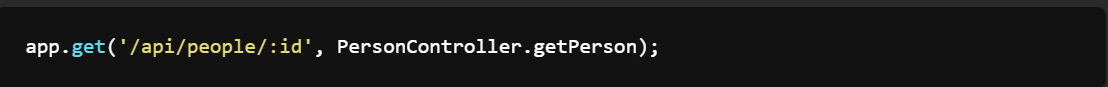
En nuestro servidor, necesitamos otro método en nuestro controlador:

server/controllers/person.controller.js

Texto

Descripción generada automáticamente

Y agregar otra ruta:



De vuelta a React. Necesitaremos crear un nuevo archivo llamado Detail.jsen nuestra carpeta de vistas:

views/Detail.js

Texto

Descripción generada automáticamente

Dado que ahora tenemos enrutamiento al front-end, tendremos que volver a trabajar nuestro archivo src/App.js en nuestro proyecto React:

Primero, ejecuta npm install react-router-dom@5.3.0dentro de tu proyecto React.

App.js

Texto

Descripción generada automáticamente

Y ahora deberíamos tener un proyecto simple que recupere nuestros datos de nuestra API. La página principal pasa a la ruta http://localhost:3000/people

Nota: Actualmente necesitaremos actualizar nuestra página para que aparezca la nueva persona. Refactorizaremos nuestro código en Advanced MERN para lograr esto.