Create the server

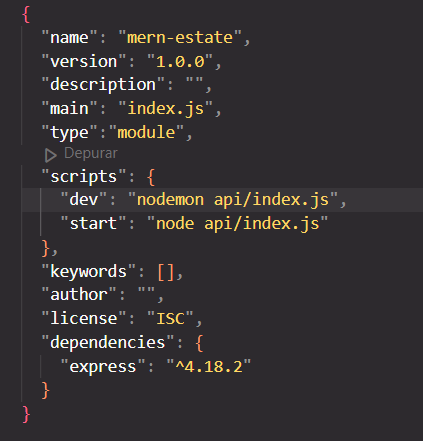
create a folder named api into the root

in the the terminal and in the root :

npm init -y

A screenshot of a computer

Description automatically generated





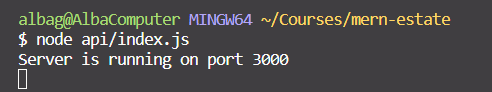
Now we need to create index.js file inside api folder

install express

npm i express

A computer screen shot of a program code

Description automatically generated



mv .git ../ This move git repository to the root directory

como bloquear cellular por robo

A screenshot of a phone

Description automatically generated

A screenshot of a computer

Description automatically generated

class 7: connect to the database

open two terminal one for the api and another for the client and run them:



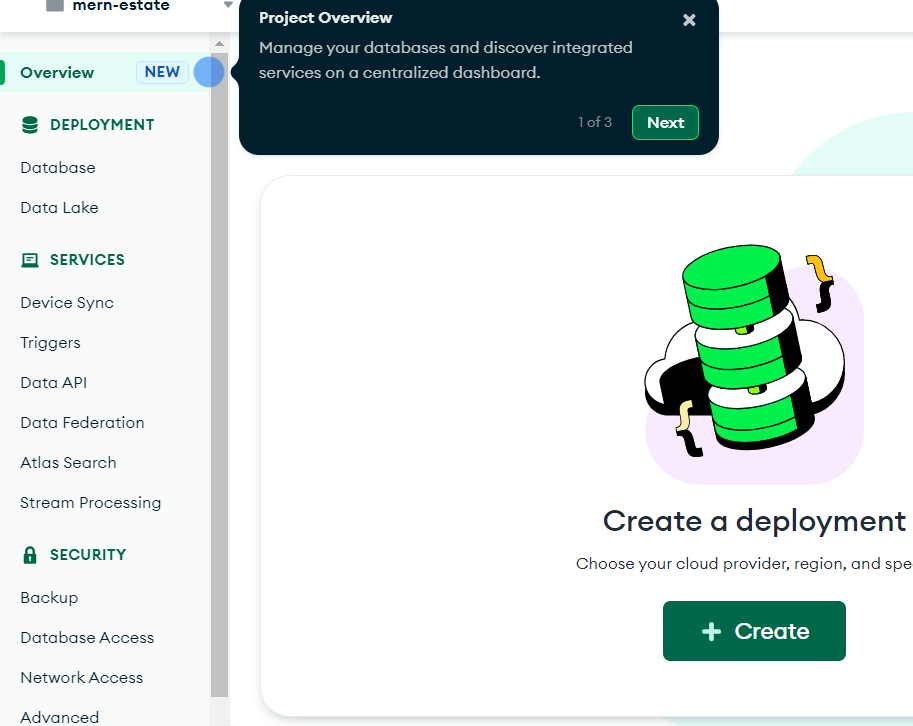
npm i mongoose

in the chrome sigin mongo atlas and create a new project:

A screenshot of a web page

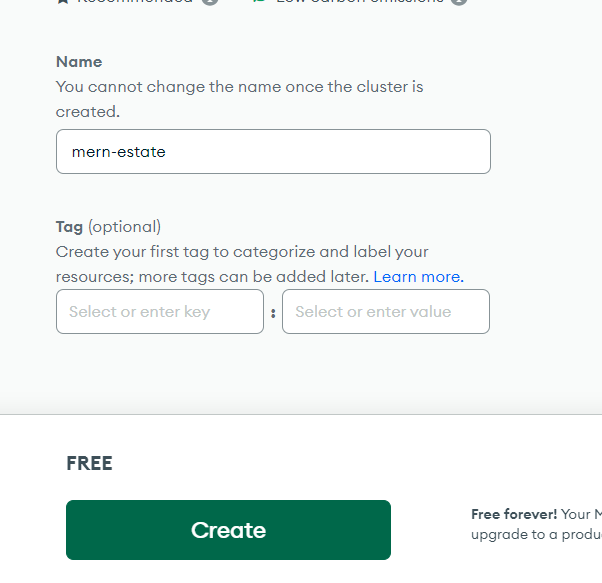
Description automatically generated

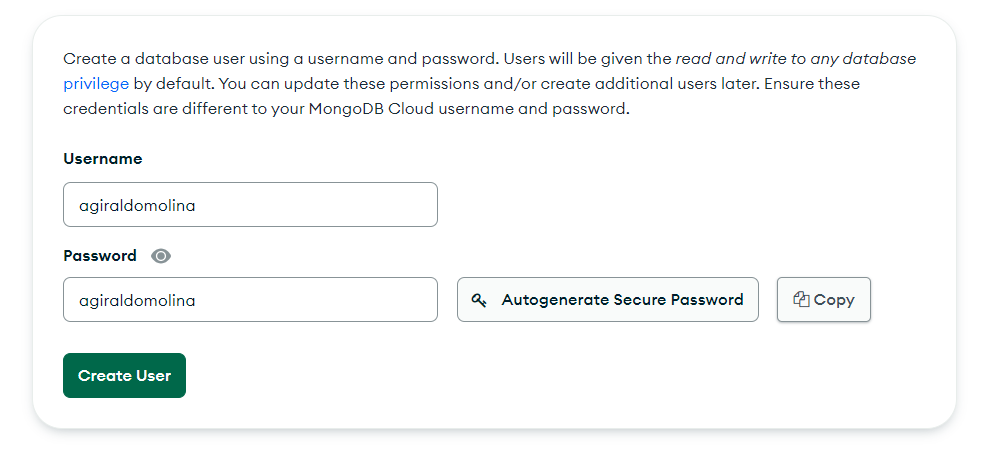






next build a database





A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated



finish and close

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

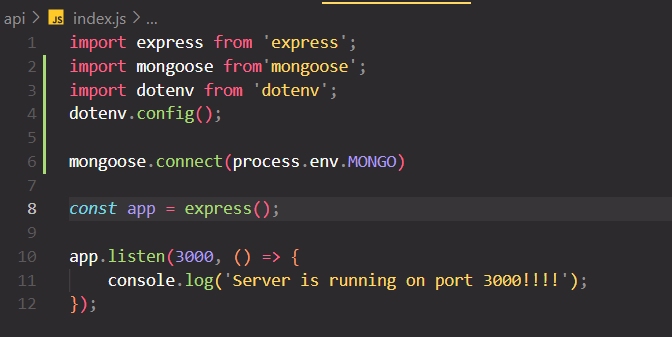
A screenshot of a phone

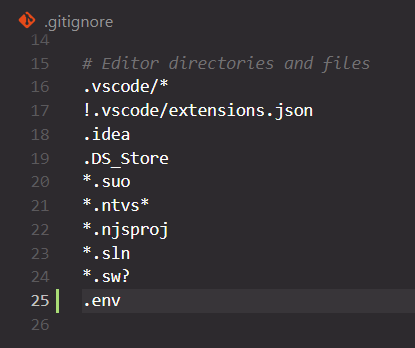
Description automatically generated

create .env file and create a variable:

MONGO="mongodb+srv://agiraldomolina:agiraldomolina@mern-estate.h6or5ga.mongodb.net/mern-estate?retryWrites=true&w=majority"

>npm install dotenv





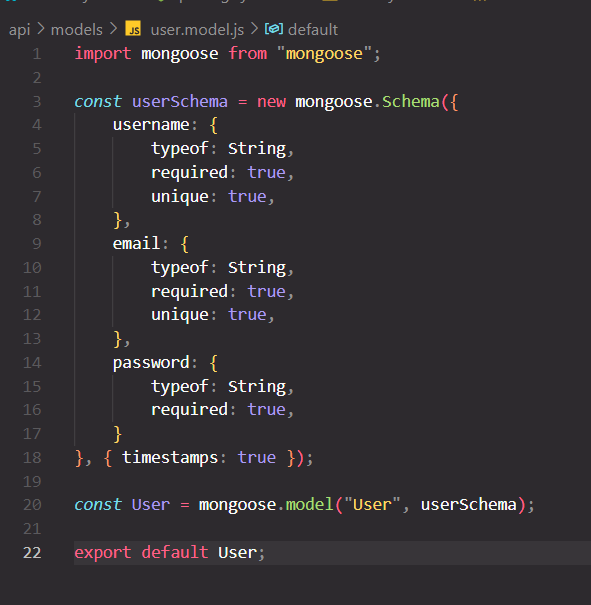


CLASS 8 Create a user model

A screenshot of a computer

Description automatically generated

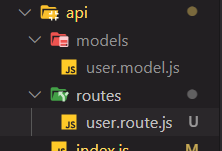
\*\*\*\*alt +shift+down arrow\*\*\*\* copy twice a blok of code



CLASS 9 Create a test API route

**request is the data that we get from the client side**

**response is the data we sent back from the server side**



A screenshot of a computer

Description automatically generated

 A screen shot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

each time we call /api/user index will call ‘/test’ from userRouter which calls test function which is defined in the controller file

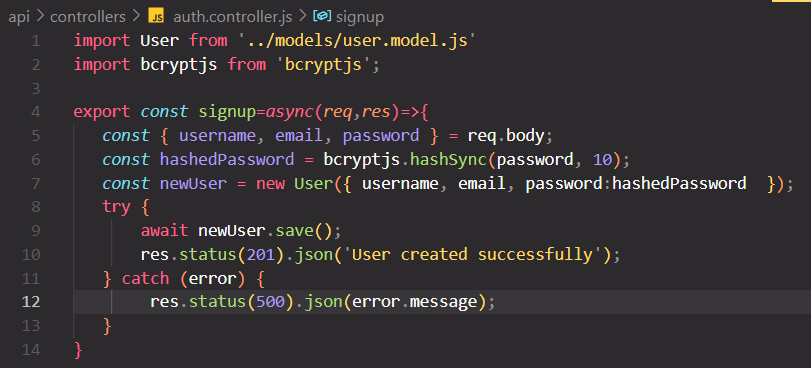
CLASS 10 : Create sign up API route

A screenshot of a computer

Description automatically generated



A screen shot of a computer screen

Description automatically generated 

A screen shot of a computer program

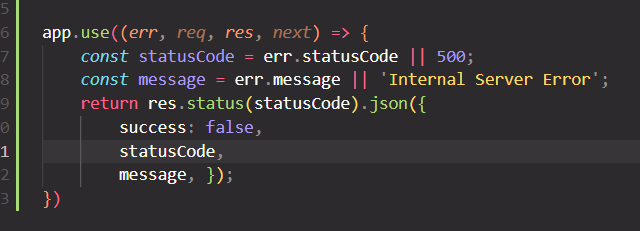
Description automatically generated



npm i bcryptjs

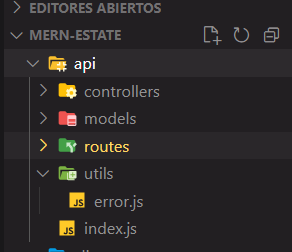
CLASS 11 : Create middleware and function to handle errors

in the api/index.js





Create a new folder and a new file:

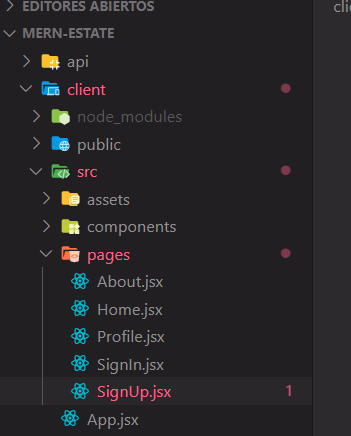


A screen shot of a computer program

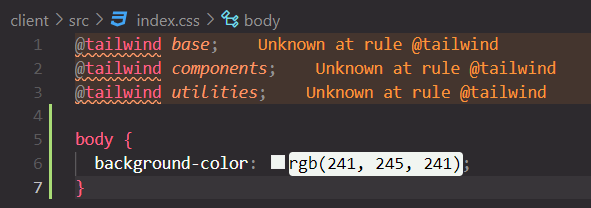
Description automatically generated

CLASS 12:Commplete sign up page UI

Beowse to SignUp.jsx



To have same color in all views:





texto grande, centrado, con negrita y un margen de 7

|  |
| --- |
| import { Link } from 'react-router-dom'  export default *function* SignUp() {    return (      <div *className*='p-3 max-w-lg mx-auto'>        <h1 *className*='text-3xl text-center font-semibold my-7'>Sign Up</h1>        <form *className*='flex flex-col gap-4'>          <input *type*="text" *placeholder*='username' *className*='border p-3 rounded-lg' *id*='username' />          <input *type*="text" *placeholder*='email' *className*='border p-3 rounded-lg' *id*='email' />          <input *type*="text" *placeholder*='password' *className*='border p-3 rounded-lg' *id*='password' />          <button *className*='bg-slate-700 text-white p-3 rounded-lg uppercase hover:opacity-95 disabled:opacity-80'>Sign Up</button>        </form>        <div *className*='flex gap-2 mt-5'>          <p>Have an account?</p>          <Link *to*={'/sign-in'}>            <span *className*='text-blue-700'>Sing In</span>          </Link>        </div>      </div>    )  } |

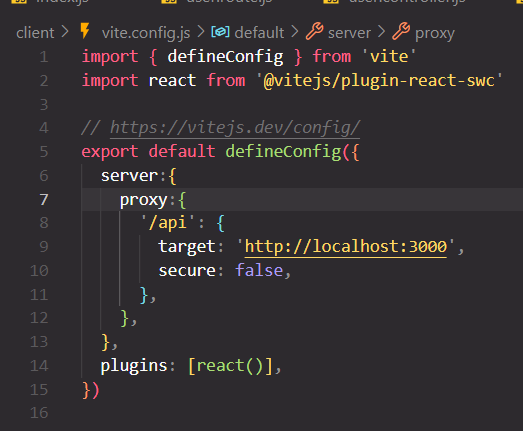
CLASS 13: Complete sign up page functionality

We are going to add functionality to this page and connect it to the database

so we want to add an unchanged event listenet to the inputs

all information is stored in a piece of state called formData

Create a proxi:



it allows the client side charge the data in the api side

and this the new code for sigup.jsx:

import { useState } from 'react'

import { Link, useNavigate } from 'react-router-dom'

export default *function* SignUp() {

*const* [formData, setFormData] = useState({})

*const* [error, setError] = useState(null);

*const*[loading, setLoading] = useState(false);

*const* navigate = useNavigate();

*const* handleChange = (*e*) => {

    setFormData({

      ...formData, [e.target.id]: e.target.value

     });

    };

*const* handleSubmit = *async*(*e*) => {

    e.preventDefault()

    try {

      setLoading(true)

*const* res = await fetch('/api/auth/signup', {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify(formData),

      });

*const* data = await res.json();

      if (data.success=== false) {

        setError(data.message)

        setLoading(false)

        return

      }

      setLoading(false)

      setError(null)

      navigate('/sign-in')

    } catch (error) {

      setLoading(false);

      setError(error.message);

    }

  };

  return (

    <div *className*='p-3 max-w-lg mx-auto'>

      <h1 *className*='text-3xl text-center font-semibold my-7'>Sign Up</h1>

      <form *onSubmit*={handleSubmit} *className*='flex flex-col gap-4'>

        <input *type*="text" *placeholder*='username' *className*='border p-3 rounded-lg' *id*='username' *onChange*={handleChange}/>

        <input *type*="text" *placeholder*='email' *className*='border p-3 rounded-lg' *id*='email' *onChange*={handleChange}/>

        <input *type*="text" *placeholder*='password' *className*='border p-3 rounded-lg' *id*='password' *onChange*={handleChange}/>

        <button *disabled*={loading} *className*='bg-slate-700 text-white p-3 rounded-lg uppercase hover:opacity-95 disabled:opacity-80'>{loading?'Loading...':'Sign Up'}</button>

      </form>

      <div *className*='flex gap-2 mt-5'>

        <p>Have an account?</p>

        <Link *to*={'/sign-in'}>

          <span *className*='text-blue-700'>Sing In</span>

        </Link>

      </div>

      {error && <p *className*='text-red-500'>{error}</p>}

    </div>

  )

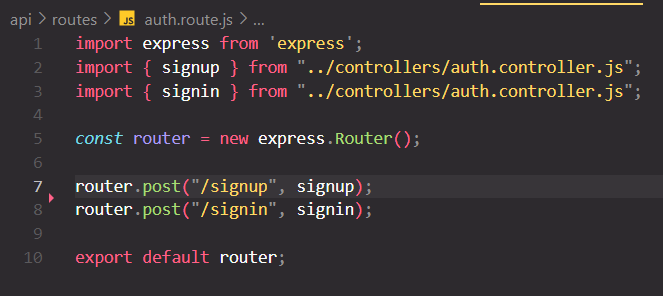
}

CLASS 14: Create singn in API route

we need to install jsonwentoken library:

npm install jsonwebtoken

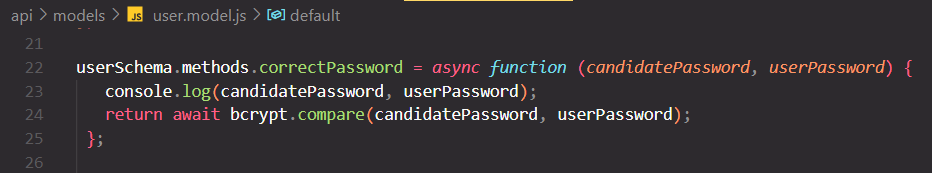
inside auth.route.js we create a new router with the post method which is going to be for sign in and it call the signin function



signin function was created in the auth.controller.js file:



and I create the correctPassword function in the user.model js file:



and we need to import:



Testing with thunder:

A screenshot of a computer program

Description automatically generated

CLASS 15 Complete sig in page functionality

This is the in the signin.jsx file:

import { useState } from 'react'

import { Link, useNavigate } from 'react-router-dom'

import { FaEye, FaEyeSlash } from 'react-icons/fa';

export default *function* SignIn() {

*const* [showPassword, setShowPassword] = useState(false);

*const* PasswordIcon = showPassword? FaEye : FaEyeSlash;

*const* [formData, setFormData] = useState({})

*const* [error, setError] = useState(null);

*const*[loading, setLoading] = useState(false);

*const* navigate = useNavigate();

*const* handleChange = (*e*) => {

    setFormData({

      ...formData, [e.target.id]: e.target.value

     });

    };

*const* handleSubmit = *async*(*e*) => {

    e.preventDefault()

    try {

      setLoading(true)

*const* res = await fetch('/api/auth/signin', {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

        },

        body: JSON.stringify(formData),

      });

*const* data = await res.json();

      if (data.success=== false) {

        setError(data.message)

        setLoading(false)

        return

      }

      setLoading(false)

      setError(null)

      navigate('/')

    } catch (error) {

      setLoading(false);

      setError(error.message);

    }

  };

  return (

    <div *className*='p-3 max-w-lg mx-auto'>

      <h1 *className*='text-3xl text-center font-semibold my-7'>Sign In</h1>

    <form *onSubmit*={handleSubmit} *className*='flex flex-col gap-4'>

      <input *type*="text" *placeholder*='email' *className*='border p-3 rounded-lg' *id*='email' *onChange*={handleChange}/>

      <div *className*='bg-white p-3 border rounded-lg flex items-center'>

        <input

*type*={showPassword? 'text' : 'password'}

*placeholder*='password'

*className*='flex-grow bg-transparent focus:outline-none w-24 sm:w-64'

*id*='password'

*onChange*={handleChange}

        />

        <button *type*="button" *onClick*={() => setShowPassword(!showPassword)} *className*='focus:outline-none text-gray-500'>

          <PasswordIcon *className*='text-slate-600'/>

        </button>

      </div>

      <button *disabled*={loading} *className*='bg-slate-700 text-white p-3 rounded-lg uppercase hover:opacity-95 disabled:opacity-80'>{loading?'Loading...':'Sign In'}</button>

    </form>

      <div *className*='flex gap-2 mt-5'>

        <p>Dont have an account?</p>

        <Link *to*={'/sign-up'}>

          <span *className*='text-blue-700'>Sing Up</span>

        </Link>

      </div>

      {error && <p *className*='text-red-500'>{error}</p>}

    </div>

  )

}

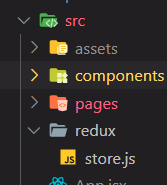
Note that the fetch methos retrieve data from the API side

CLASS 16: Add redux toolkit

in the cliente side:

npm install @reduxjs/toolkit react-redux

create a new folder and a new file:



and in the file store.js copy next code:

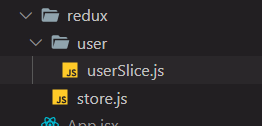
A screen shot of a computer program

Description automatically generated

in main.js file:



create a new folder and a new file:



and write this code in the userSlice.js file:

import {createSlice} from "@reduxjs/toolkit";

*const* initialState = {

    currentUser: null,

    error: null,

    loading: false

};

*const* userSlice = createSlice({

    name: "user",

    initialState,

    reducers: {

        signInStart: (*state*) => {

            state.loading = true;

        },

        signInSuccess: (*state*, *action*) => {

            state.loading = false;

            state.currentUser = action.payload;

            state.error = null;

        },

        signInFailure: (*state*, *action*) => {

            state.loading = false;

            state.error = action.payload;

        }

    }

});

export *const* { signInStart, signInSuccess, signInFailure } = userSlice.actions;

export default userSlice.reducer;

in the store.js file:

