Application Restaurant

NodeJS



{

  "name": "backend-prisma-mysql-restaurant",

  "version": "1.0.0",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "build": "prisma generate && next build",

    "postinstall": "prisma generate"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "dependencies": {

    "@prisma/client": "^5.12.1",

    "cors": "^2.8.5",

    "express": "^4.18.2"

  },

  "devDependencies": {

    "prisma": "^5.12.1"

  },

  "description": ""

}



generator client {

provider = "prisma-client-js"

}

datasource db {

provider = "mysql"

url = env("DATABASE\_URL")

}

model categorie {

id Int @id @default(autoincrement())

desCat String? @db.VarChar(30)

imageCat String? @db.VarChar(250)

products products[]

}

model cmd {

id Int @id @default(autoincrement())

codeEmp Int

dateCmd String @db.VarChar(50)

prixTotalCmd Float

tableNumber Int?

status Int

nomClient String @db.VarChar(50)

depuis String @db.VarChar(50)

verifiedByAdmin String @db.VarChar(50)

employe employe @relation(fields: [codeEmp], references: [id], onUpdate: Restrict, map: "cmd\_ibfk\_1")

prodcmd prodcmd[]

@@index([codeEmp], map: "codeEmp")

}

model employe {

id Int @id @default(autoincrement())

matriculeEmp String @db.VarChar(10)

nomPrenom String? @db.VarChar(40)

adresse String? @db.VarChar(40)

contact String? @db.VarChar(30)

tel String? @db.VarChar(15)

fonction String? @db.VarChar(20)

login String? @db.VarChar(10)

motPasse String? @db.VarChar(10)

caisse Int? @default(1)

cmd cmd[]

}

model ligne\_tickets {

id Int @id @default(autoincrement())

qte Decimal @db.Decimal(5, 0)

prix Int

produit\_id Int

ticket\_id Int

products products @relation(fields: [produit\_id], references: [id], onUpdate: Restrict, map: "ligne\_tickets\_ibfk\_1")

tickets tickets @relation(fields: [ticket\_id], references: [id], onUpdate: Restrict, map: "ligne\_tickets\_ibfk\_2")

@@index([produit\_id], map: "ligne\_tickets\_ibfk\_1")

@@index([ticket\_id], map: "ligne\_tickets\_ibfk\_2")

}

model prodcmd {

id Int @id @default(autoincrement())

codeProd Int

codeCmd Int

qte Int

prix Float @default(0)

tauxTVA Float @default(0)

cmd cmd @relation(fields: [codeCmd], references: [id], onUpdate: Restrict, map: "prodcmd\_ibfk\_1")

products products @relation(fields: [codeProd], references: [id], onUpdate: Restrict, map: "prodcmd\_ibfk\_2")

@@index([codeCmd], map: "codeCmd")

@@index([codeProd], map: "codeProd")

}

model products {

id Int @id @default(autoincrement())

title String? @db.VarChar(17)

image String? @db.VarChar(102)

previewDescription String? @db.VarChar(112)

detailDescription String? @db.VarChar(250)

price Decimal @db.Decimal(4, 0)

qtestock Int

codeCategorie Int

ligne\_tickets ligne\_tickets[]

prodcmd prodcmd[]

categorie categorie @relation(fields: [codeCategorie], references: [id], onUpdate: Restrict, map: "products\_ibfk\_1")

@@index([codeCategorie], map: "codeCategorie")

}

model tickets {

id Int @id @default(autoincrement())

numero Int

dateticket DateTime @db.Date

client String @db.VarChar(50)

ligne\_tickets ligne\_tickets[]

}



const express = require('express');

const { PrismaClient } = require('@prisma/client')

const prisma = new PrismaClient()

const router = express.Router();

//Ajout d'un product

router.post('/', async (req, res, )=> {

 const {

  title   ,

  image   ,

  previewDescription ,

  detailDescription ,

  price           ,

  qtestock        ,

  codeCategorie

} =req.body

    try {

        const product = await prisma.products.create({

            data: {

              title :  title,

              image :  image,

              previewDescription :previewDescription,

              detailDescription :detailDescription,

              price      : price    ,

              qtestock   : qtestock    ,

              codeCategorie :codeCategorie

            }

        })

        res.json(product)

      } catch (error) {

        res.status(500).json({

          message: "Something went wrong", error : error

        })

      }

});

// afficher la liste des products.

router.get('/', async (req, res, )=> {

    try {

        // products est le nom du model précisé dans prisma.schema

        const products = await prisma.products.findMany({

            include: {

                categorie: true,

                     },

          })

            res.json(products)

        } catch (error) {

         res.status(500).json({

        message: "Something went wrong", error : error

       })

       }

});

// afficher un product.

router.get('/:id', async (req, res, )=> {

    const { id } = req.params

    try {

        const product = await prisma.products.findUnique({

            where: {

                id: Number(id),

            }

        })

       res.json(product)

      } catch (error) {

        res.status(500).json({

          message: "Something went wrong", error : error

        })

      }

});

// modifier un product

router.put('/:id', async (req, res)=> {

  const {

    title   ,

    image   ,

    previewDescription ,

    detailDescription ,

    price           ,

    qtestock        ,

    codeCategorie

  } =req.body

    const id  = req.params.id;

    try {

     const product = await prisma.products.update({

          data: {

              title :  title,

              image :  image,

              previewDescription :previewDescription,

              detailDescription :detailDescription,

              price      : price    ,

              qtestock   : qtestock    ,

              codeCategorie :codeCategorie

            },

            where: { id: Number(id)},

        })

    res.json(product);

    } catch (error) {

    res.status(404).json({ message: error.message });

    }

});

// Supprimer un product

router.delete('/:id', async (req, res)=> {

    const  id  = req.params.id;

    try {

     await prisma.products.delete({

        where: { id: Number(id) },

    })

    res.json({ message: "product "+ id +" deleted successfully." });

} catch (error) {

    res.status(404).json({ message: error.message });

    }

});

module.exports = router;



const express=require('express');

const cors=require('cors')

const app = express()

//BodyParser Middleware

app.use(express.json());

app.use(cors())

// requête

app.get("/",(req,res)=>{

    res.send("bonjour");

    });

// Appel de routes

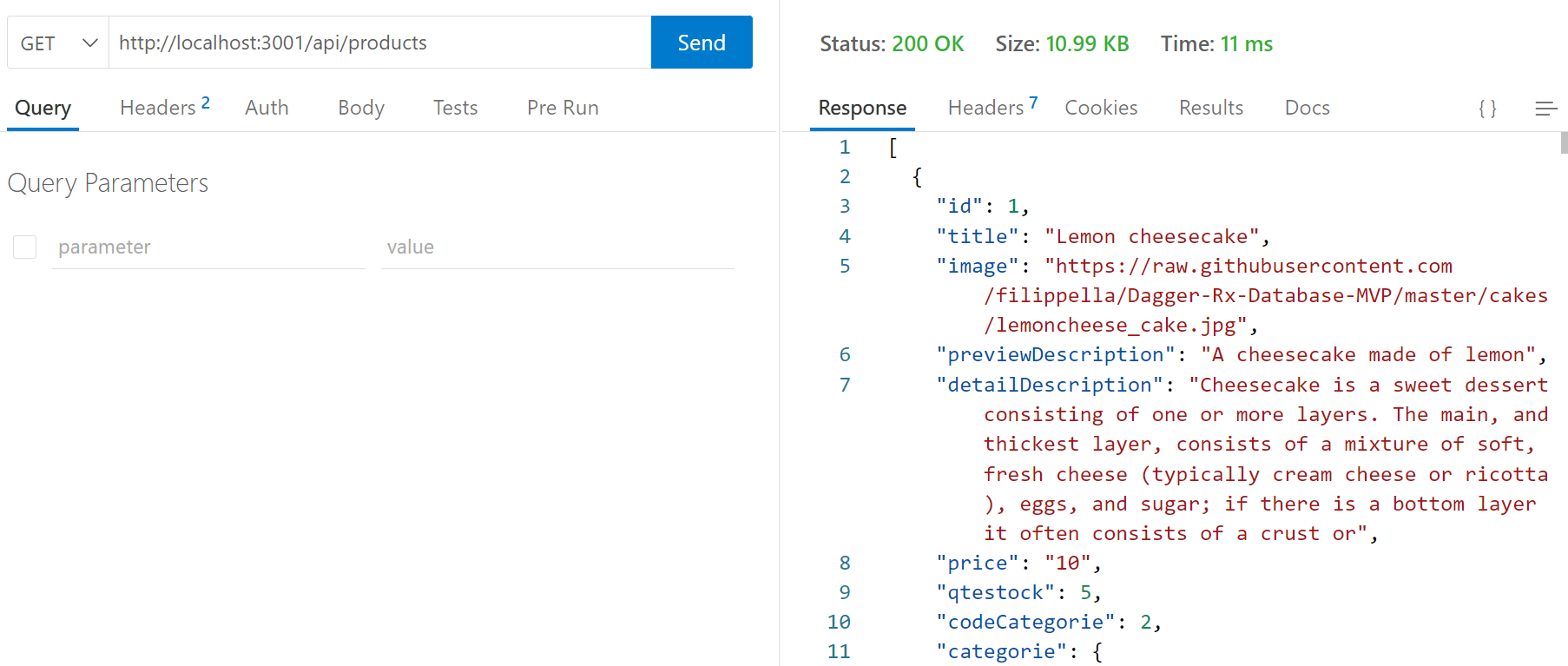
const productsRouter =require("./routes/products.route")

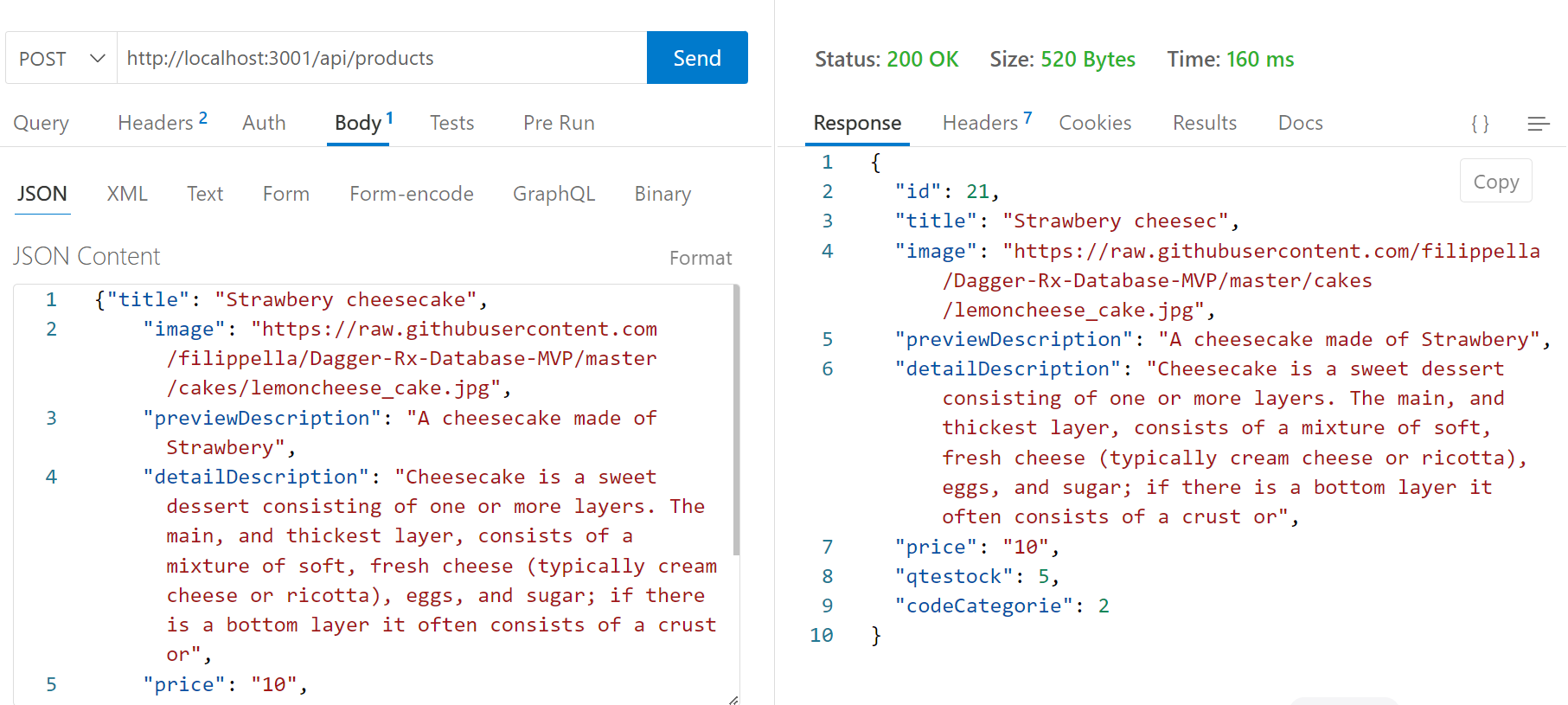
app.use('/api/products', productsRouter);

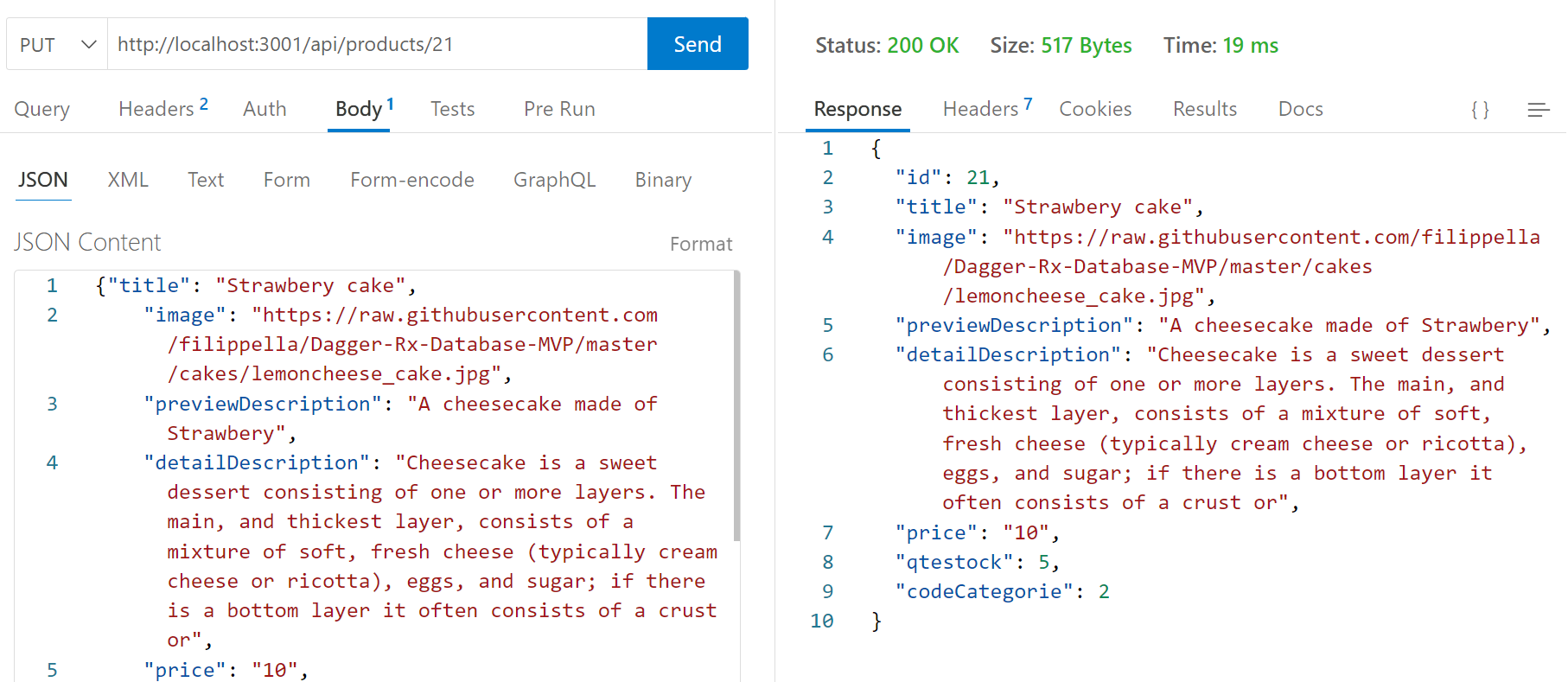
const PORT = process.env.PORT || 3001

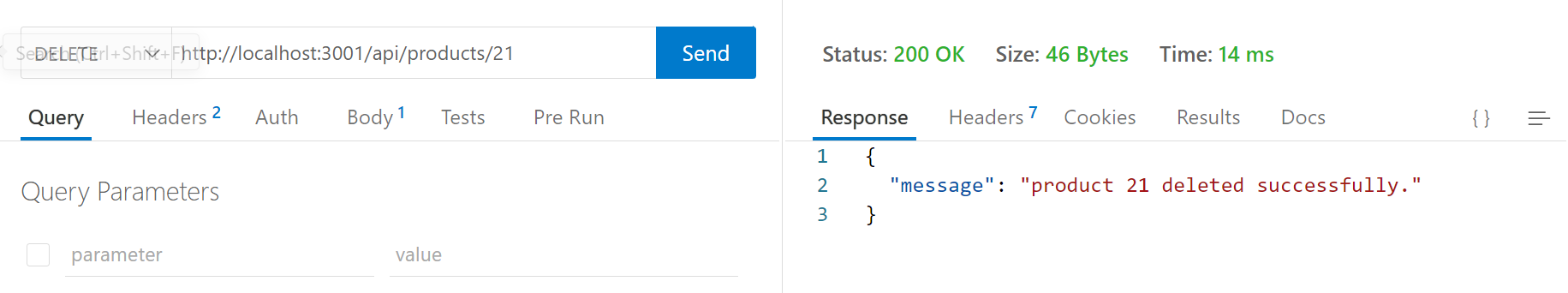
app.listen(PORT, () => console.log(`Server is running on port ${PORT}`))

module.exports = app;

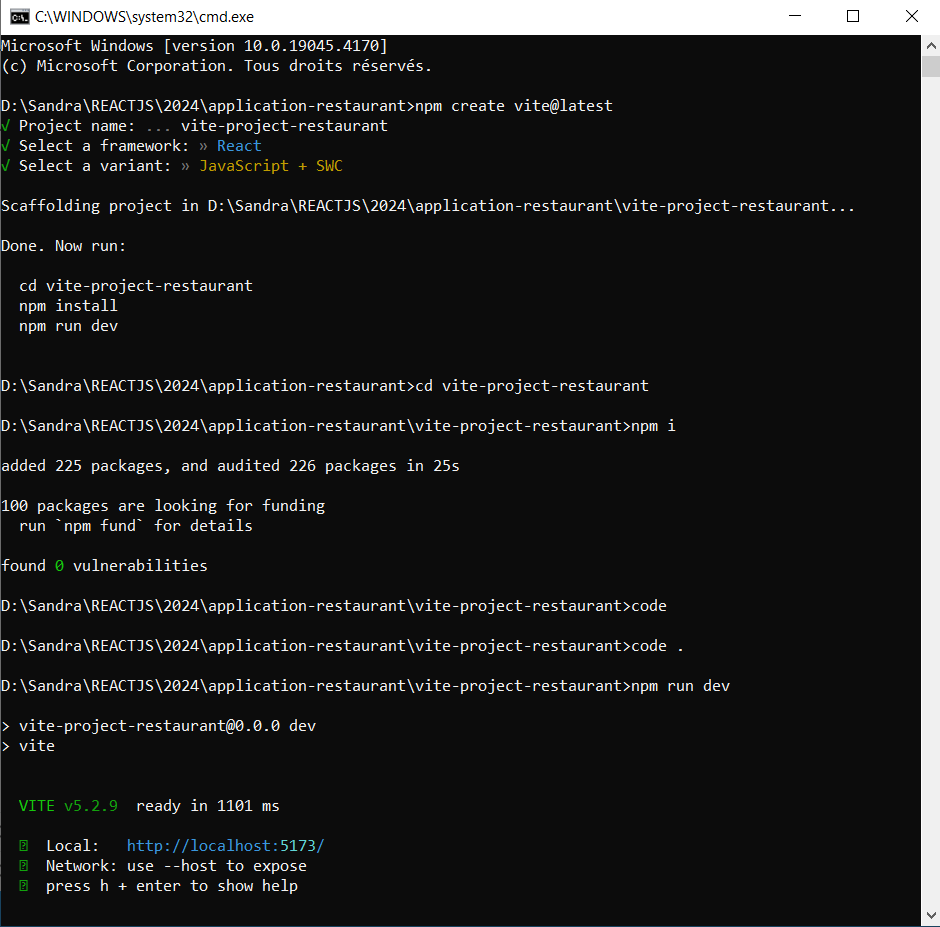








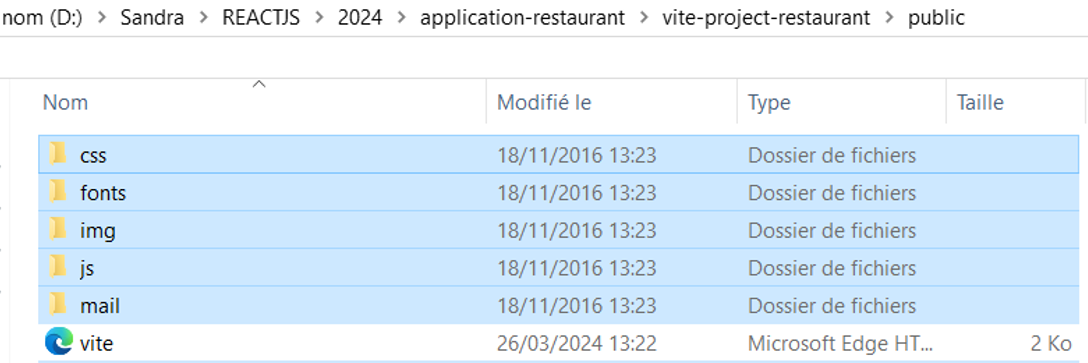
React JS



Télécharger template

<https://www.free-css.com/free-css-templates/page231/touche>

Déplacer



index.html : le header : copier son contenu puis l’ajouter à index.html du projet react

Aussi les appels des script.js

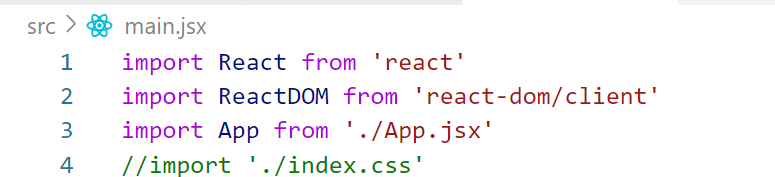
Le body de l’index.html : le convertir en jsx puis le mettre dans un composant

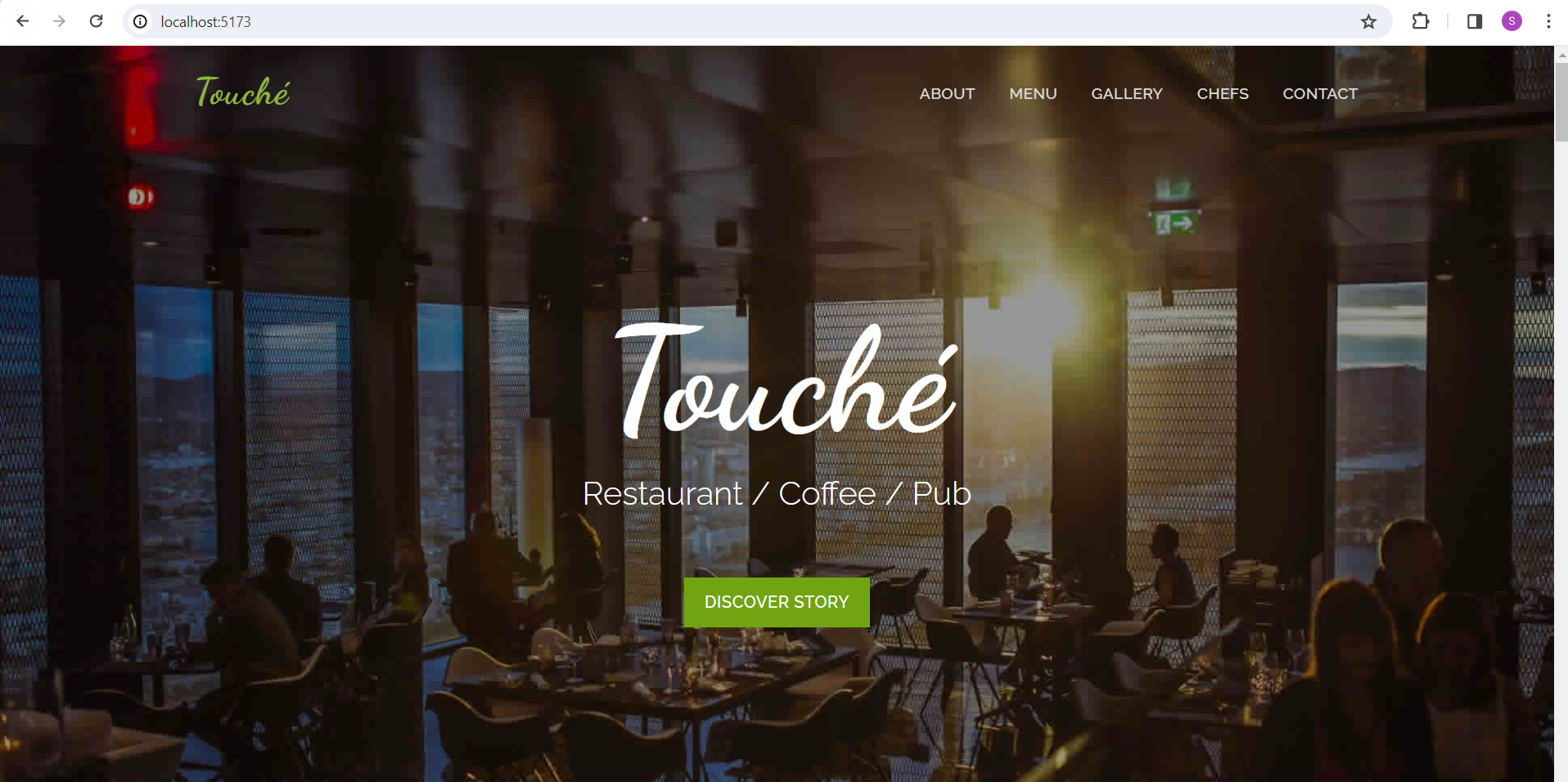
<https://transform.tools/html-to-jsx>



Faire appel à ce composant par la suite

Puis supprimer l’import de l’index.css dans main





npm i axios



import axios from 'axios'

export default axios.create({

baseURL:"http://localhost:3001/api/"

})



import Api from "../axios/Api";

const PRODUCT\_API="products"

export const fetchproducts=async()=> {

return await Api.get(PRODUCT\_API);

}

export const fetchproductById=async(productId)=> {

    return await Api.get(PRODUCT\_API + '/' + productId);

    }

export const deleteproduct=async(productId) =>{

    return await Api.delete(PRODUCT\_API + '/' + productId);

    }

export const addproduct=async(product)=> {

    return await Api.post(PRODUCT\_API, product);

    }

export const updateproduct=(product) =>{

    return Api.put(PRODUCT\_API + '/' + product.\_id, product);

    }



import React,{useState,useEffect} from 'react'

import {fetchproducts} from '../../services/productsService'

import AffProducts from './affProducts';

const ListProducts = () => {

    const [products, setProducts] = useState("");

    useEffect(() => {

        fetchproducts().then((response) =>setProducts(response.data))

                        .catch((error) =>console.log(error))

      }, []);

  return (

   <AffProducts products={products} />

  )

}

export default ListProducts





import ListProducts from './components/products/listProducts'

….

 <ul className="nav navbar-nav navbar-right">

          <li>

          <a href="#orders" className="page-scroll">

              Orders

            </a>

</li>

<li>

            <a href="#about" className="page-scroll">

              About

            </a>

          </li>

…..

  {/\* Contact Section \*/}

  <div id="contact" className="text-center">

    <div className="container">

      <div className="section-title text-center">

        <h2>Contact Form</h2>

        <hr />

        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit duis sed.</p>

      </div>

      <div className="col-md-10 col-md-offset-1">

        <form name="sentMessage" id="contactForm" noValidate="">

          <div className="row">

            <div className="col-md-6">

              <div className="form-group">

                <input

                  type="text"

                  id="name"

                  className="form-control"

                  placeholder="Name"

                  required="required"

                />

                <p className="help-block text-danger" />

              </div>

            </div>

            <div className="col-md-6">

              <div className="form-group">

                <input

                  type="email"

                  id="email"

                  className="form-control"

                  placeholder="Email"

                  required="required"

                />

                <p className="help-block text-danger" />

              </div>

            </div>

          </div>

          <div className="form-group">

            <textarea

              name="message"

              id="message"

              className="form-control"

              rows={4}

              placeholder="Message"

              required=""

              defaultValue={""}

            />

            <p className="help-block text-danger" />

          </div>

          <div id="success" />

          <button type="submit" className="btn btn-custom btn-lg">

            Send Message

          </button>

        </form>

      </div>

    </div>

  </div>

  {/\* Order Section \*/}

  <div id="orders" className="text-center">

<h2>Orders</h2>

<div className="row">

        <div className="portfolio-items">

 <ListProducts />

       </div>

 </div>

  </div>