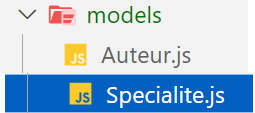
Atelier 4 NextJS : CRUD livres

# Les Models

On suppose que le Model Auteur est déjà créé (voir atelier 2)

Sous /src/models créez un fichier nommé Specialite.js



import mongoose from "mongoose"

var specialiteSchema = mongoose.Schema({

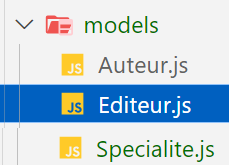
    nomspecialite: String

})

const Specialite = mongoose.models.Specialite ||mongoose.model('Specialite', specialiteSchema);

export default Specialite

Sous /src/models créez un fichier nommé Editeur.js



import mongoose from "mongoose"

var editeurSchema = mongoose.Schema({

    maisonedit: String,

    siteweb: String,

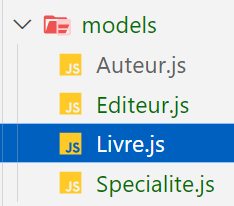
    email: String

})

const Editeur = mongoose.models.Editeur ||mongoose.model('Editeur', editeurSchema);

export default Editeur

Sous /src/models créez un fichier nommé Livre.js



import mongoose from "mongoose"

import Editeur from './Editeur'

import Specialite from './Specialite'

import Auteur from './Auteur'

var livreSchema = mongoose.Schema({

    isbn: String,

    titre: {type: String, required: true},

    annedition: Number,

    prix: Number,

    qtestock: Number,

    couverture: String,

    specialite: {

        type: mongoose.Schema.Types.ObjectId,

        ref: Specialite,

    },

    maised: {

        type: mongoose.Schema.Types.ObjectId,

        ref: Editeur

    },

    auteurs: [{

        type: mongoose.Schema.Types.ObjectId,

        ref: Auteur

    }]

})

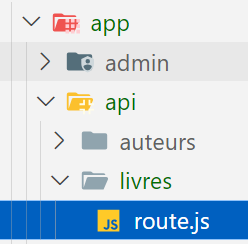
const Livre = mongoose.models.Livre ||mongoose.model('Livre', livreSchema);

export default Livre

# API Routes

## Livre

Définir les points de terminaison de l'API, dans notre cas, app/api/livres/route.js



import { HttpStatusCode } from 'axios';

import connectDB from '@/lib/connectDB';

import Livre from '@/models/Livre';

import {  NextResponse } from 'next/server';

export async function POST(req) {

    try {

        await connectDB();

        const body = await req.json();

        if (body.titre) {

            const livre = await Livre.create(body);

            return NextResponse.json(

                { livre, message: 'Your book has been created' },

                { status: HttpStatusCode.Created },

            );

        }

        return NextResponse.json({ message: 'Title is missing' }, { status: HttpStatusCode.BadRequest });

    } catch (error) {

        return NextResponse.json({ message: error }, { status: HttpStatusCode.BadRequest });

    }

}

export async function GET() {

    try {

        await connectDB();

        const livres = await Livre.find({}, null, {sort: {'\_id': -1}})**.populate('auteurs').populate('specialite').populate('maised')**

        return NextResponse.json(livres );

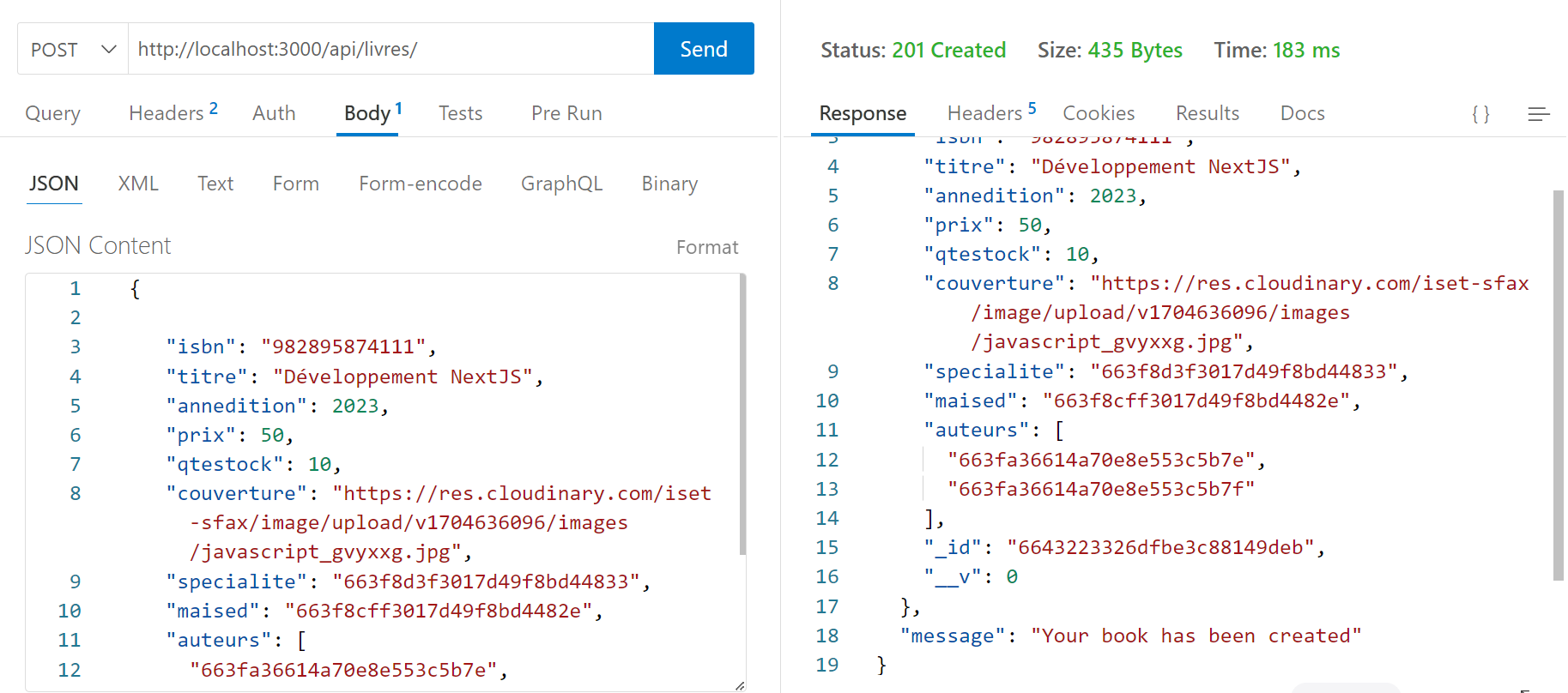
    } catch (error) {

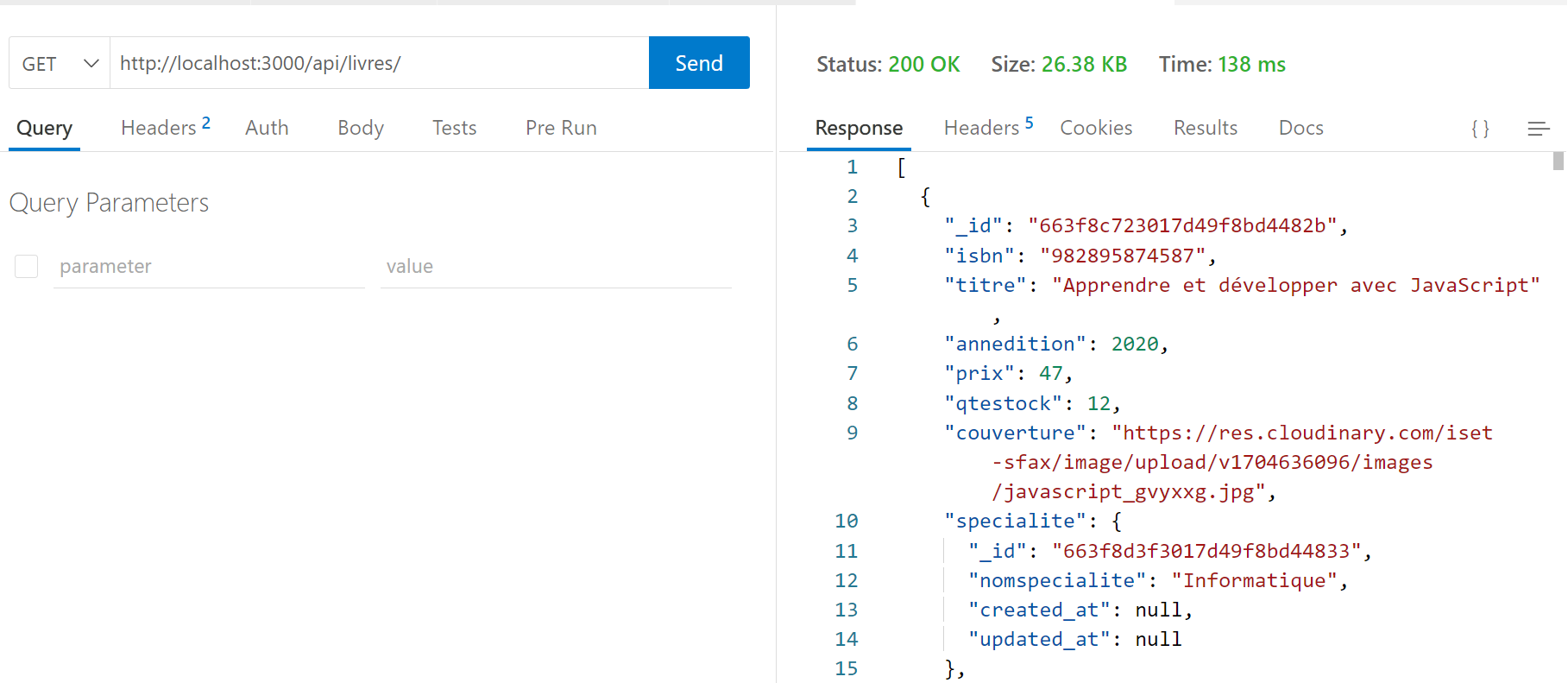
        return NextResponse.json({ error : error });

    }

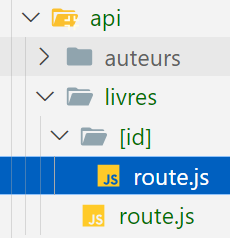
}

Test Api





Dans le répertoire app/api/livres créer le dossier [id] ayant le fichier route.js



import { HttpStatusCode } from 'axios';

import connectDB from '@/lib/connectDB';

import Livre from '@/models/Livre';

import {  NextResponse } from 'next/server';

export async function GET(\_, { params }) {

    try {

        await connectDB();

        const livre = await Livre.findById(params.id).populate('auteurs').populate('specialite').populate('maised')

        if (livre) {

            return NextResponse.json(livre);

        }

        return NextResponse.json({ message: `Livre ${params.id} not found` }, { status: HttpStatusCode.NotFound });

    } catch (error) {

        return NextResponse.json({ message: error }, { status: HttpStatusCode.BadRequest });

    }

}

export async function PUT(req, { params }) {

    try {

        await connectDB();

        const livre = await Livre.findById(params.id);

        if (livre) {

            const body= await req.json();

            if (body.titre) {

                livre.titre = body.titre;

            }

            if (body.isbn) {

                livre.isbn = body.isbn;

            }

            if (body.annedition) {

                livre.annedition = body.annedition;

            }

            if (body.prix) {

                livre.prix = body.prix;

            }

            if (body.qtestock) {

                livre.qtestock = body.qtestock;

            }

            if (body.couverture) {

                livre.couverture = body.couverture;

            }

            if (body.specialite) {

                livre.specialite = body.specialite;

            }

            if (body.maised) {

                livre.maised = body.maised;

            }

            if (body.auteurs) {

                livre.auteurs = body.auteurs;

            }

            livre.save();

            return NextResponse.json({ livre });

        }

        return NextResponse.json({ message: `Livre ${params.id} not found` }, { status: HttpStatusCode.NotFound });

    } catch (error) {

        return NextResponse.json({ message: error }, { status: HttpStatusCode.BadRequest });

    }

}

export async function DELETE(\_, { params }) {

    try {

        await connectDB();

        const livre = await Livre.findById(params.id);

        if (livre) {

            await Livre.findByIdAndDelete(livre.\_id);

            return NextResponse.json({ message: `Book ${params.id} has been deleted` });

        }

        return NextResponse.json({ message: `Book ${params.id} not found` }, { status: HttpStatusCode.NotFound });

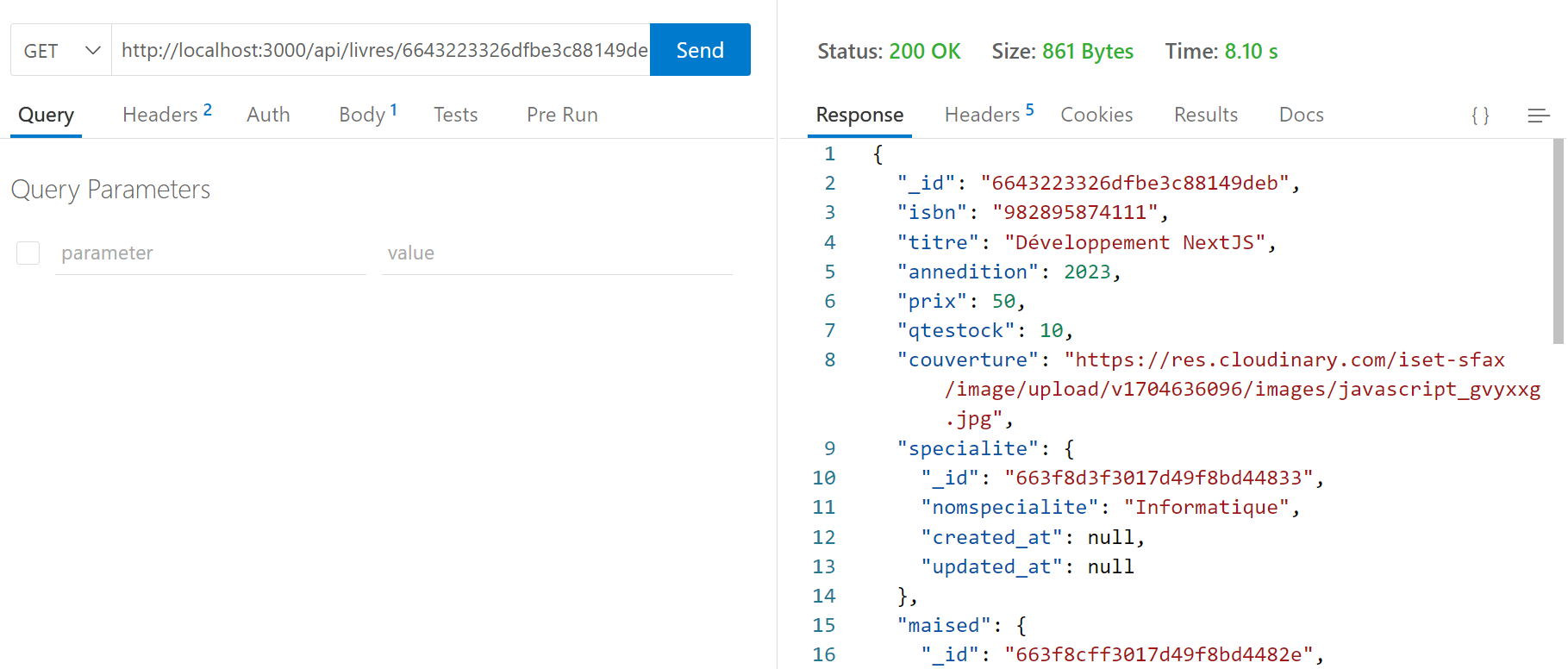
    } catch (error) {

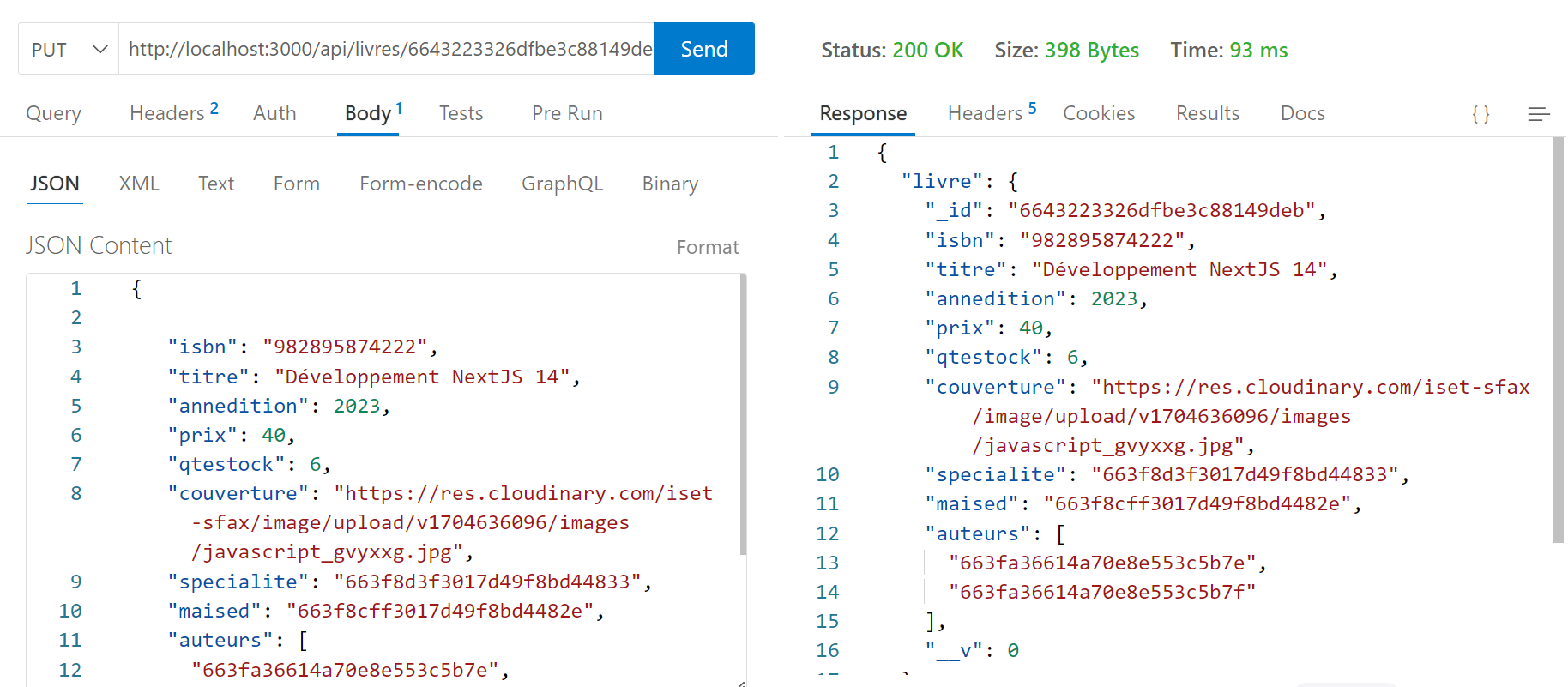
        return NextResponse.json({ message: error }, { status: HttpStatusCode.BadRequest });

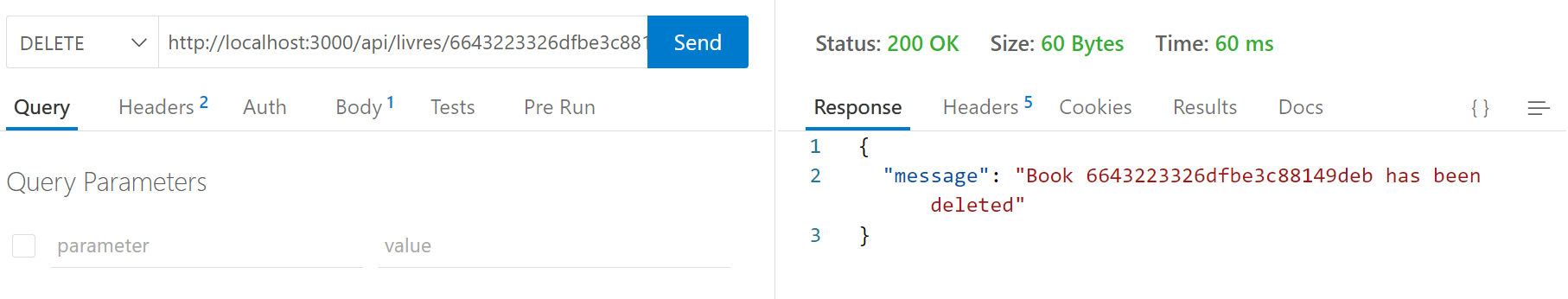
    }

}

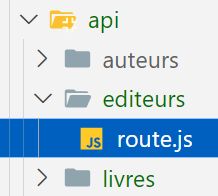
Test Api







## Editeur



import connectDB from '@/lib/connectDB';

import Editeur from '@/models/Editeur';

import {  NextResponse } from 'next/server';

export async function GET() {

    try {

        await connectDB();

        const editeurs = await Editeur.find({}, null, {sort: {'\_id': -1}});

        return NextResponse.json(editeurs );

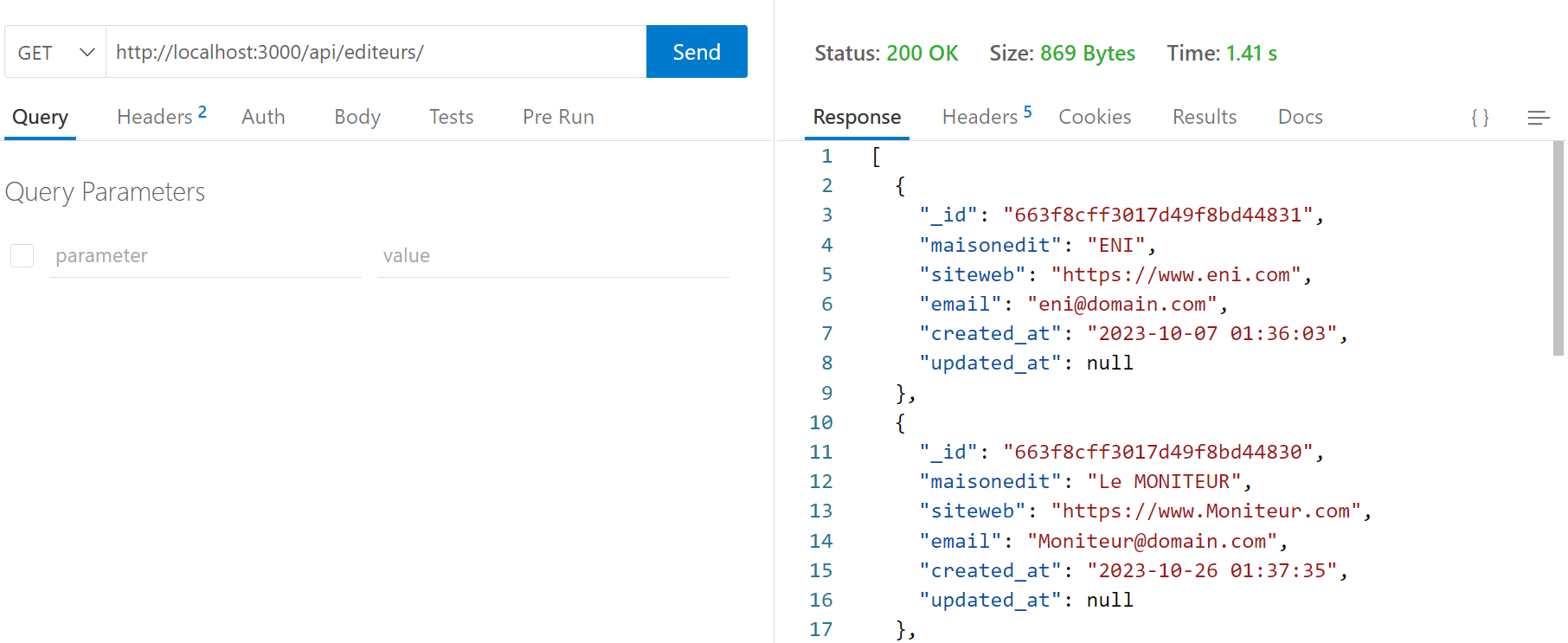
    } catch (error) {

        return NextResponse.json({ error });

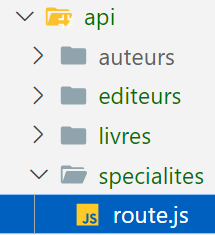
    }

}

Test Api



## Spécialité



import connectDB from '@/lib/connectDB';

import Specialite from '@/models/Specialite';

import {  NextResponse } from 'next/server';

export async function GET() {

    try {

        await connectDB();

        const specialites = await Specialite.find({}, null, {sort: {'\_id': -1}});

        return NextResponse.json(specialites );

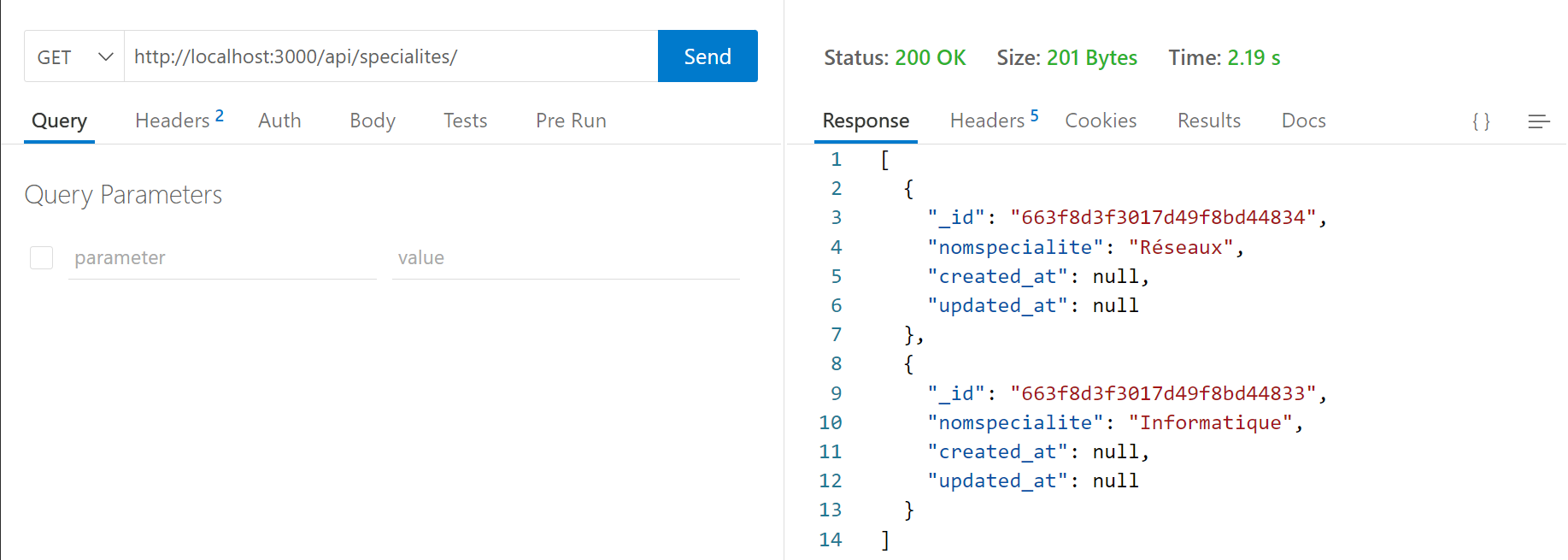
    } catch (error) {

        return NextResponse.json({ error });

    }

}

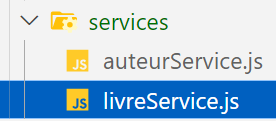
Test Api



# Services

## Livre

Sous /src/services créer livreService.js



const LIVRE\_API="/livres/"

export const fetchLivres=async()=> {

const res = await fetch(process.env.NEXT\_PUBLIC\_API\_URL+LIVRE\_API, { cache: 'no-store' })

const response = await res.json();

return response;

}

export const fetchLivreById=async(livreId)=> {

    const res = await fetch(process.env.NEXT\_PUBLIC\_API\_URL+LIVRE\_API+`${livreId}`, { cache: 'no-store' },{

        method: 'GET'

    });

    const response = await res.json();

     return response;

}

export const deleteLivre=async(livreId) =>{

    const res = await fetch(process.env.NEXT\_PUBLIC\_API\_URL+LIVRE\_API+`${livreId}`,{

        method: 'DELETE'

    });

    const response = await res.json();

    return response;

}

export const addLivre=async(livre)=> {

    const res = await fetch(process.env.NEXT\_PUBLIC\_API\_URL+LIVRE\_API, {

        method: 'POST',

        headers: {

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

        },

        body: JSON.stringify(livre),

    });

    const response = await res.json();

    return response;

}

export const editLivre=async(livre) =>{

    const res = await fetch(process.env.NEXT\_PUBLIC\_API\_URL+LIVRE\_API+`${livre.\_id}`, {

        method: 'PUT',

        headers: {

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

        },

        body: JSON.stringify(livre),

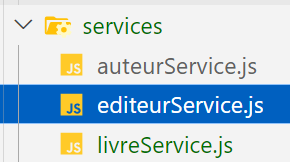
    });

    const response = await res.json();

    return response;

}

## Editeur



const EDITEUR\_API="/editeurs/"

export const fetchEditeurs=async()=> {

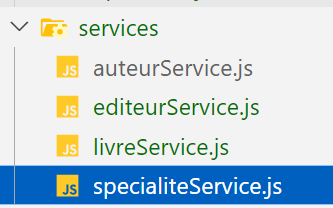
const res = await fetch(process.env.NEXT\_PUBLIC\_API\_URL+EDITEUR\_API, { cache: 'no-store' })

const response = await res.json();

return response;

}

## Spécialité



const SPECIALITE\_API="/specialites/"

export const fetchSpecialites=async()=> {

const res = await fetch(process.env.NEXT\_PUBLIC\_API\_URL+SPECIALITE\_API, { cache: 'no-store' })

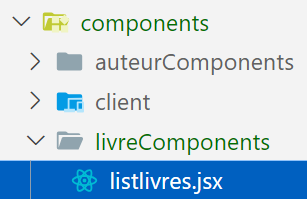
const response = await res.json();

return response;

}

# Affichage des livres

Sous /src/components/ créer le dossier livreComponents puis créer le fichier listlivres.jsx



"use client"

import  { useMemo } from 'react';

import { MaterialReactTable } from 'material-react-table';

import { Box } from '@mui/material';

const Listlivres = ({livres}) => {

  const columns = useMemo(

    () => [

        {

            accessorKey: 'couverture',

            header: 'COUVERTURE',

            Cell: ({ cell}) => (

            <Box

            sx={{

            display: 'flex',

            alignItems: 'center',

            gap: '1rem',

            }}

            >

            <img

                      src={cell.getValue()}

                      alt="couverture"

                      height="50"

                      width="50"

                    />

            </Box>),

            },

    {

    accessorKey: 'titre',

    header: 'TITRE',

    size: 100,

    },

    {

    accessorKey: 'isbn',

    header: 'ISBN',

    size: 50,

    },

    {

    accessorKey: 'annedition',

    header: 'ANNEE',

    size: 50,

    },

    {

    accessorKey: 'prix',

    header: 'PRIX',

    size: 50,

    },

    {

    accessorKey: 'qtestock',

    header: 'QTE',

    size: 50,

    },

    {

        accessorFn: (originalRow) => originalRow.auteurs.map((aut,i)=>{

          return <div key={i}>{aut.nomauteur}</div>

          }),

        id: 'aut.\_id',

        header: 'AUTEURS',

         },

     {

        accessorKey: 'specialite.nomspecialite',

        header: 'SPECIALITE',

        size: 50,

        },

    {

        accessorKey: 'maised.maisonedit',

        header: 'EDITEUR',

        size: 50,

            },

    ],

    [livres],

    );

return (

<div>

<MaterialReactTable columns={columns} data={livres} />;

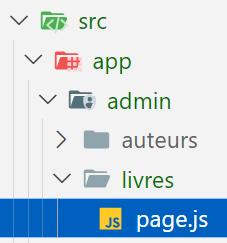
</div>

  )

}

export default Listlivres

On va créer un dossier appelé livres sous /src/app/admin ayant page.js



import Listlivres from '@/components/livreComponents/listlivres';

import {fetchLivres} from "@/services/livreService"

const getLivres=async()=>{

const data=await fetchLivres()

return data;

}

const LivrePage = async() =>{

    const livres=await getLivres()

  return (

   <div>

      <Listlivres livres={livres}/>

    </div>

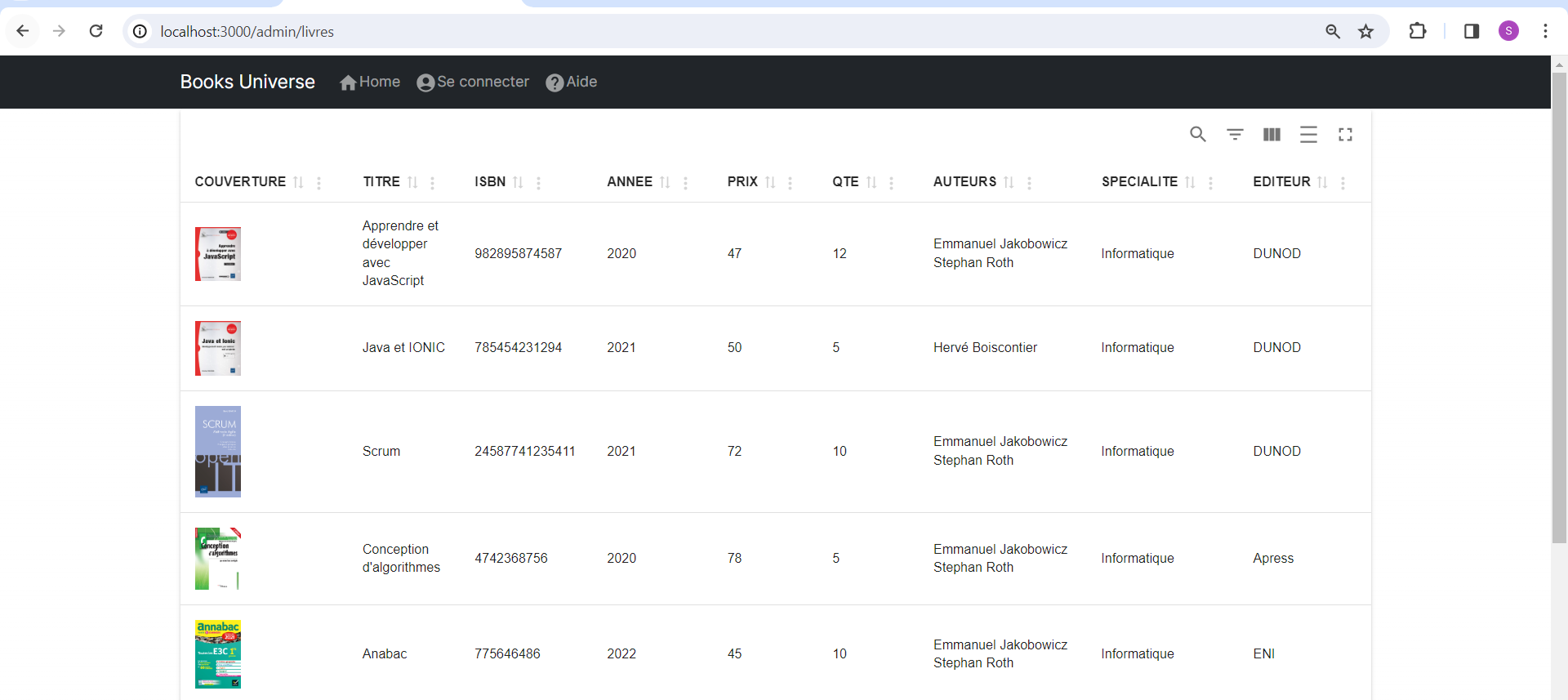
  )

}

export default LivrePage

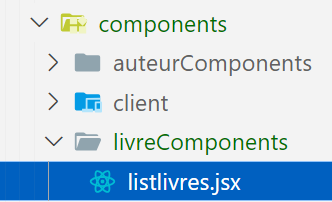
Test dans le navigateur

<http://localhost:3000/admin/livres>



# Suppression des livres

On va ajouter l’action de suppression, modifier le contenu de listlivres.jsx



"use client"

import  { useMemo } from 'react';

import { MaterialReactTable } from 'material-react-table';

import { Box } from '@mui/material';

import Button from 'react-bootstrap/Button';

import {deleteLivre} from "@/services/livreService"

import { useRouter } from "next/navigation";

import DeleteForeverIcon from '@mui/icons-material/DeleteForever';

const Listlivres = ({livres}) => {

    const router = useRouter();

    const deletelivre=(id)=>{

      if(window.confirm("supprimer Livre O/N")) {

        deleteLivre(id)

        .then((res)=>{ console.log(res)

         router.refresh()

        })

        .catch(error=>{

            console.log(error)

           })

  }

        }

  const columns = useMemo(

    () => [

        {

            accessorKey: 'couverture',

            header: 'COUVERTURE',

            Cell: ({ cell}) => (

            <Box

            sx={{

            display: 'flex',

            alignItems: 'center',

            gap: '1rem',

            }}

            >

            <img

                      src={cell.getValue()}

                      alt="couverture"

                      height="50"

                      width="50"

                    />

            </Box>),

            },

    {

    accessorKey: 'titre',

    header: 'TITRE',

    size: 100,

    },

    {

    accessorKey: 'isbn',

    header: 'ISBN',

    size: 50,

    },

    {

    accessorKey: 'annedition',

    header: 'ANNEE',

    size: 100,

    },

    {

    accessorKey: 'prix',

    header: 'PRIX',

    size: 50,

    },

    {

    accessorKey: 'qtestock',

    header: 'QTE',

    size: 50,

    },

    {

        accessorFn: (originalRow) => originalRow.auteurs.map((aut,i)=>{

            return <div key={i}>{aut.nomauteur}</div>

          }),

        id: 'aut.\_id',

        header: 'AUTEURS',

         },

     {

        accessorKey: 'specialite.nomspecialite',

        header: 'SPECIALITE',

        size: 50,

        },

    {

        accessorKey: 'maised.maisonedit',

        header: 'EDITEUR',

        size: 50,

            },

            {

                accessorKey: '\_id',

                header: 'actions',

                size: 50,

                Cell: ({ cell, row }) => (

                <div >

                <Button

                onClick={(e) => {

                deletelivre(cell.row.original.\_id,e);

                }}

                variant="danger"

                size="md"

                className="text-danger btn-link delete"

                >

                <DeleteForeverIcon />

                </Button>

                </div>

                ),

                },

    ],

    [livres],

    );

return (

<div>

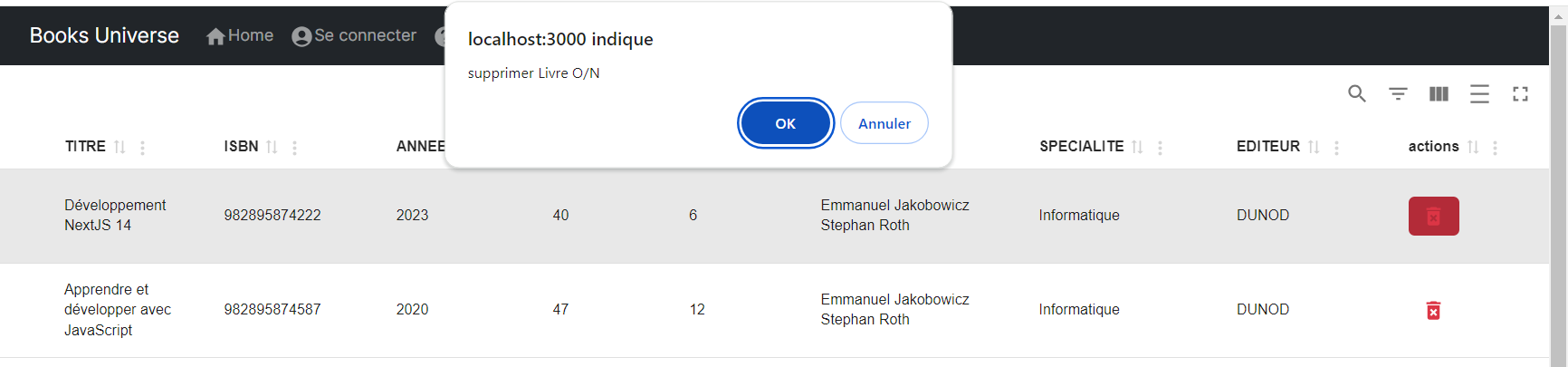
<MaterialReactTable columns={columns} data={livres} />;

</div>

  )

}

export default Listlivres



# Ajout de livres

On va ajouter le formulaire d’ajout.

Tout d’abord modifier listlivres.jsx pour ajouter le bouton de lien vers le formulaire.

"use client"

import  { useMemo } from 'react';

import { MaterialReactTable } from 'material-react-table';

import { Box } from '@mui/material';

import Button from 'react-bootstrap/Button';

import {deleteLivre} from "@/services/livreService"

import { useRouter } from "next/navigation";

import DeleteForeverIcon from '@mui/icons-material/DeleteForever';

import Link from 'next/link';

import AddCircleOutlineIcon from '@mui/icons-material/AddCircleOutline';

const Listlivres = ({livres}) => {

    const router = useRouter();

    const deletelivre=(id)=>{

      if(window.confirm("supprimer Livre O/N")) {

        deleteLivre(id)

        .then((res)=>{ console.log(res)

         router.refresh()

        })

        .catch(error=>{

            console.log(error)

           })

  }

        }

  const columns = useMemo(

    () => [

        {

            accessorKey: 'couverture',

            header: 'COUVERTURE',

            Cell: ({ cell}) => (

            <Box

            sx={{

            display: 'flex',

            alignItems: 'center',

            gap: '1rem',

            }}

            >

            <img

                      src={cell.getValue()}

                      alt="couverture"

                      height="50"

                      width="50"

                    />

            </Box>),

            },

    {

    accessorKey: 'titre',

    header: 'TITRE',

    size: 100,

    },

    {

    accessorKey: 'isbn',

    header: 'ISBN',

    size: 50,

    },

    {

    accessorKey: 'annedition',

    header: 'ANNEE',

    size: 100,

    },

    {

    accessorKey: 'prix',

    header: 'PRIX',

    size: 50,

    },

    {

    accessorKey: 'qtestock',

    header: 'QTE',

    size: 50,

    },

    {

        accessorFn: (originalRow) => originalRow.auteurs.map((aut,i)=>{

            return <div key={i}>{aut.nomauteur}</div>

          }),

        id: 'aut.\_id',

        header: 'AUTEURS',

         },

     {

        accessorKey: 'specialite.nomspecialite',

        header: 'SPECIALITE',

        size: 50,

        },

    {

        accessorKey: 'maised.maisonedit',

        header: 'EDITEUR',

        size: 50,

            },

            {

                accessorKey: '\_id',

                header: 'actions',

                size: 50,

                Cell: ({ cell, row }) => (

                <div >

                <Button

                onClick={(e) => {

                deletelivre(cell.row.original.\_id,e);

                }}

                variant="danger"

                size="md"

                className="text-danger btn-link delete"

                >

                <DeleteForeverIcon />

                </Button>

                </div>

                ),

                },

    ],

    [livres],

    );

return (

<div>

<Button

variant='dark'

size="sm"

>

<Link

href="/admin/livres/newLivre"

style={{

  textDecoration: 'none',

  color: 'pink',

  fontSize: 14,

}}

>

<AddCircleOutlineIcon/> Nouveau

</Link>

</Button>

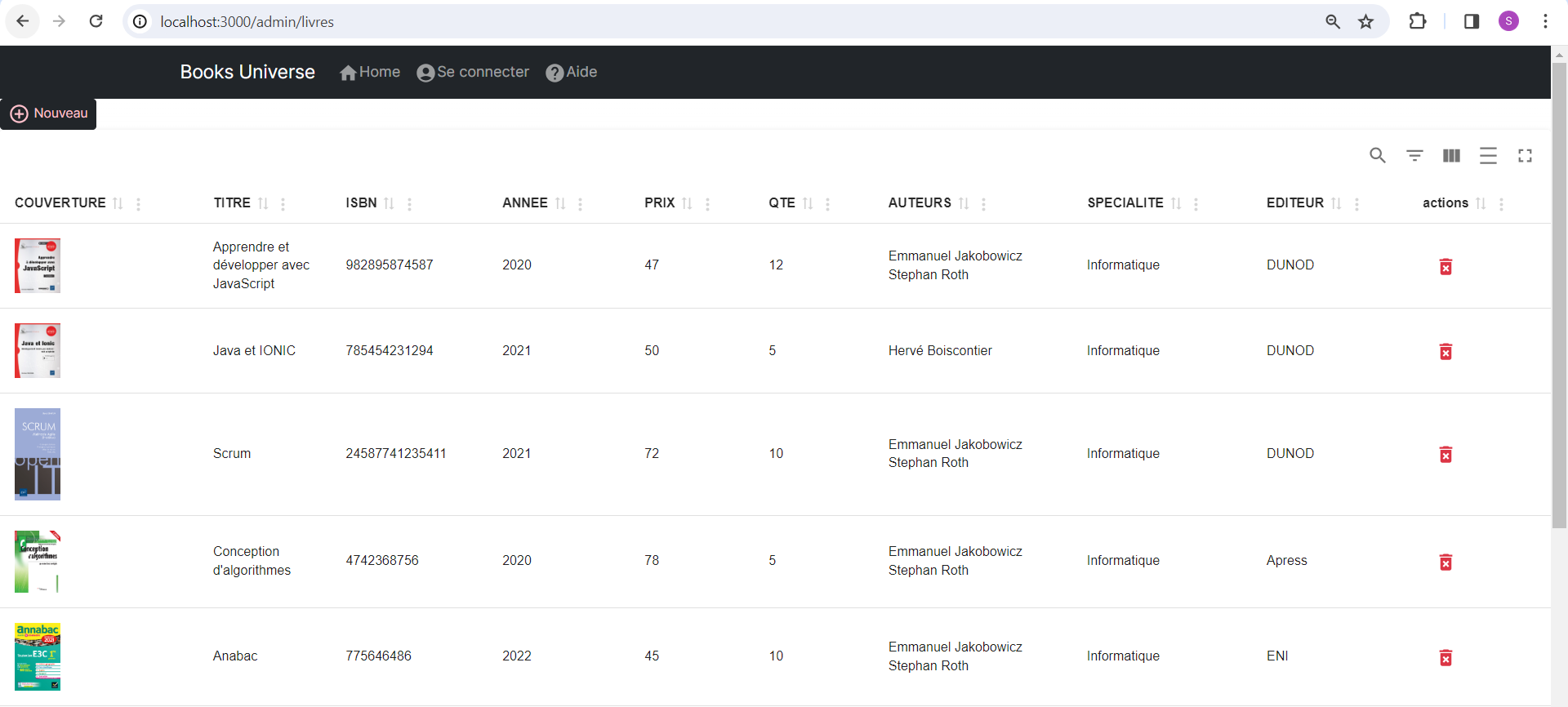
<MaterialReactTable columns={columns} data={livres} />;

</div>

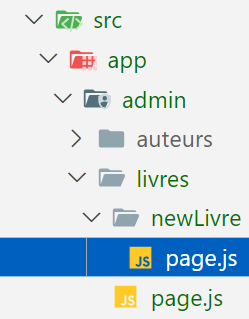
  )

}

export default Listlivres



Créer sous /src/app/admin/livres/ le dossier newLivre incluant page.js



import NewLivre from "@/components/livreComponents/newLivreComponent";

import { fetchEditeurs } from "@/services/editeurService";

import { fetchSpecialites } from "@/services/specialiteService";

import { fetchAuteurs } from "@/services/auteurService";

const getEditeurs=async()=>{

    const data=await fetchEditeurs()

    return data;

}

const getSpecialites=async()=>{

    const data=await fetchSpecialites()

    return data;

}

const getAuteurs=async()=>{

    const data=await fetchAuteurs()

    return data;

}

const NewLivrePage = async() => {

    const editeurs=await getEditeurs()

    const specialites=await getSpecialites()

    const auteurs=await getAuteurs()

  return (

    <div>

     <NewLivre LesEditeurs={editeurs} lesSpecialites={specialites} lesAuteurs={auteurs} />

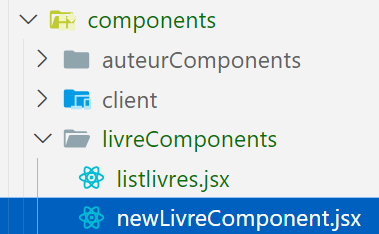
    </div>

  )

}

export default NewLivrePage

Par la suite, créer sous /src/components/livreComponents le fichier newLivreComponent.jsx



"use client"

import React, { useState } from 'react';

import Button from 'react-bootstrap/Button';

import Form from 'react-bootstrap/Form';

import Col from 'react-bootstrap/Col';

import InputGroup from 'react-bootstrap/InputGroup';

import Row from 'react-bootstrap/Row';

import { useRouter } from "next/navigation";

import { addLivre } from '@/services/livreService';

import TextField from '@mui/material/TextField';

import MenuItem from '@mui/material/MenuItem';

const NewLivre = ({LesEditeurs,lesSpecialites,lesAuteurs}) => {

    const router = useRouter();

const [isbn, setIsbn] = useState("");

const [titre, setTitre] = useState("");

const [annedition, setAnnedition] = useState("");

const [prix, setPrix] = useState("");

const [qtestock, setQtestock] = useState("");

const [couverture, setCouverture] = useState("");

const [specialite, setSpecialite] = useState("");

const [maised, setMaised] = useState("");

const [auteurs, setAuteurs] = useState([]);

const [validated, setValidated] = useState(false);

const handleSubmit = (e) => {

e.preventDefault();

const form = e.currentTarget;

if (form.checkValidity() === true) {

const newLivre = {

    isbn,

    titre,

    annedition,

    prix,

    qtestock,

    couverture,

    specialite,

    maised,

    auteurs

};

//faire le add dans la BD

addLivre(newLivre)

.then(res => {

  router.push('/admin/livres')

    router.refresh()

})

.catch(error=>{

alert("Erreur ! Insertion non effectuée")

})

}

setValidated(true);

}

const handleReset = () => {

    setIsbn("")

    setTitre("")

    setAnnedition("")

    setPrix("")

    setQtestock("")

    setCouverture("")

    setSpecialite("")

    setMaised("")

    setAuteurs("")

}

const toggleAuteurs = (option) => {

    if (auteurs.includes(option)) {

        setAuteurs(

            auteurs.filter((item) =>

                item !== option));

    } else {

        setAuteurs(

            [...auteurs, option]);

    }

};

return (

<div>

 <Form noValidate validated={validated} onSubmit={handleSubmit}>

 <h2>Ajout Livre</h2>

<div className="container w-100 d-flex justify-content-center">

<div>

<div className='form mt-3'>

<Row className="mb-2">

<Form.Group as={Col} md="6" >

<Form.Label >ISBN \*</Form.Label>

<Form.Control

required

type="text"

placeholder="ISBN Livre"

value={isbn}

onChange={(e)=>setIsbn(e.target.value)}

/>

<Form.Control.Feedback type="invalid">

Saisir ISBN Livre

</Form.Control.Feedback>

</Form.Group>

<Form.Group as={Col} md="6" >

<Form.Label >Titre \*</Form.Label>

<Form.Control

required

type="text"

placeholder="Titre Livre"

value={titre}

onChange={(e)=>setTitre(e.target.value)}

/>

<Form.Control.Feedback type="invalid">

Saisir titre Livre

</Form.Control.Feedback>

</Form.Group>

<Form.Group as={Col} md="6">

<Form.Label>Année \*</Form.Label>

<Form.Control

required

type="number"

placeholder="année édition"

value={annedition}

onChange={(e)=>setAnnedition(e.target.value)}

/>

<Form.Control.Feedback type="invalid">

Saisir Année

</Form.Control.Feedback>

</Form.Group>

<Form.Group className="col-md-6">

<Form.Label>Prix \*</Form.Label>

<InputGroup hasValidation>

<Form.Control

type="number"

required

placeholder="Prix"

value={prix}

onChange={(e)=>setPrix(e.target.value)}

/>

<Form.Control.Feedback type="invalid">

Prix Incorrect

</Form.Control.Feedback>

</InputGroup>

</Form.Group>

</Row>

<Row className="mb-2">

<Form.Group className="col-md-6">

<Form.Label>Quantité \*</Form.Label>

<InputGroup hasValidation>

<Form.Control

type="number"

required

placeholder="Qté"

value={qtestock}

onChange={(e)=>setQtestock(e.target.value)}

/>

<Form.Control.Feedback type="invalid">

Qté Incorrecte

</Form.Control.Feedback>

</InputGroup>

</Form.Group>

<Form.Group className="col-md-6">

<Form.Label>Couverture \*</Form.Label>

<InputGroup hasValidation>

<Form.Control

type="text"

required

placeholder="Couverture"

value={couverture}

onChange={(e)=>setCouverture(e.target.value)}

/>

<Form.Control.Feedback type="invalid">

Couverture Incorrecte

</Form.Control.Feedback>

</InputGroup>

</Form.Group>

</Row>

<Row className="mb-2">

<Form.Group className="col-md-6">

<Form.Label>Spécialité \*</Form.Label>

<InputGroup hasValidation>

<Form.Control

required

as="select"

type="select"

value={specialite}

onChange={(e)=>setSpecialite(e.target.value)}

>

<option></option>

{lesSpecialites && lesSpecialites.map((spec)=><option key={spec.\_id}

value={spec.\_id}>{spec.nomspecialite}</option>

)}

</Form.Control>

<Form.Control.Feedback type="invalid">

Spécialité Incorrecte

</Form.Control.Feedback>

</InputGroup>

</Form.Group>

<Form.Group className="col-md-6">

<Form.Label>Edition \*</Form.Label>

<InputGroup hasValidation>

<Form.Control

required

as="select"

type="select"

value={maised}

onChange={(e)=>setMaised(e.target.value)}

>

<option></option>

{LesEditeurs && LesEditeurs.map((ed)=><option key={ed.\_id}

value={ed.\_id}>{ed.maisonedit}</option>

)}

</Form.Control>

<Form.Control.Feedback type="invalid">

Edition Incorrecte

</Form.Control.Feedback>

</InputGroup>

</Form.Group>

</Row>

<Row className="mb-2">

<Form.Group className="col-md-6">

<Form.Label>Auteurs \*</Form.Label>

<InputGroup hasValidation>

<TextField

                   name="auteurs"

                   variant="outlined"

                   select

                   SelectProps={{multiple: true}}

                   value={auteurs}

                   helperText="Sélectionner des auteurs"

                   onChange={(event)=>setAuteurs(event.target.value)}

                    >

               {

               lesAuteurs ?

               lesAuteurs.map((aut)=>

                    <MenuItem  key={aut.\_id} value={aut.\_id}>{aut.nomauteur}</MenuItem>

               )

               :null

               }

             </TextField>

             <Form.Control.Feedback type="invalid">

Auteurs Incorrecters

</Form.Control.Feedback>

</InputGroup>

</Form.Group>

</Row>

</div>

</div>

</div>

<center>

<Button type="submit">Enregistrer</Button>

<Button type="button" className="btn btn-warning"

onClick={()=>handleReset()}>Annuler</Button>

</center>

</Form>

</div>

);

};

export default NewLivre