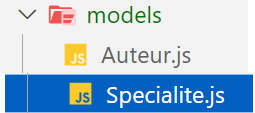
Atelier 4 NextJS : CRUD livres

# 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({

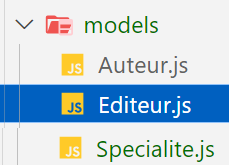
    nomspecialite: String

})

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,

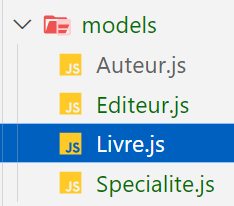
    email: String

})

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

    }]

})

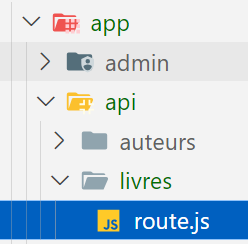
const Livre = mongoose.models.Livre ||mongoose.model('Livre', livreSchema);

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 Livre from '@/models/Livre';

import {  NextResponse } from 'next/server';

import connectDB from '@/lib/connectDB';

export async function POST(req) {

    try {

        const body = await req.json();

        const newLivre = new Livre(body)

        const livre = await newLivre.save();

            return NextResponse.json(

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

                { status: HttpStatusCode.Created },

            );

  } 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 );

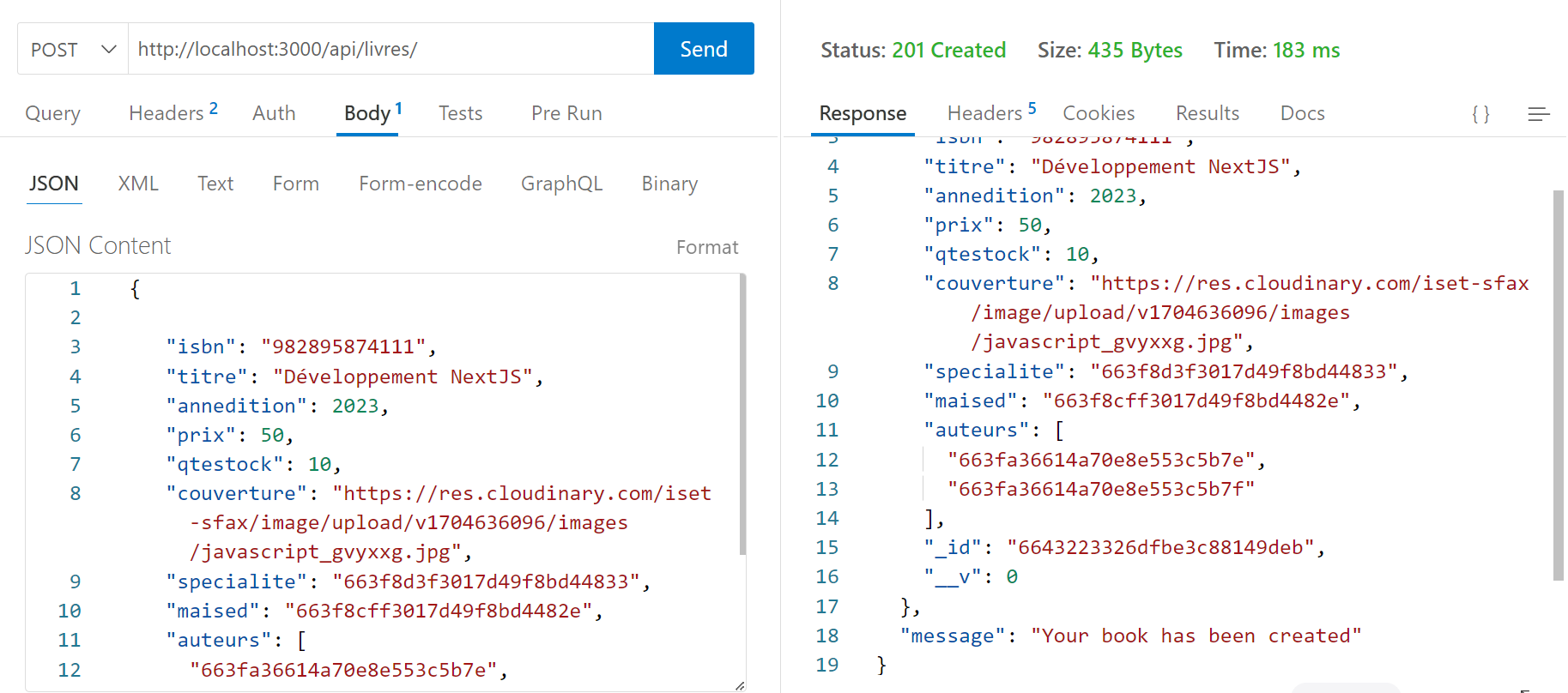
    } catch (error) {

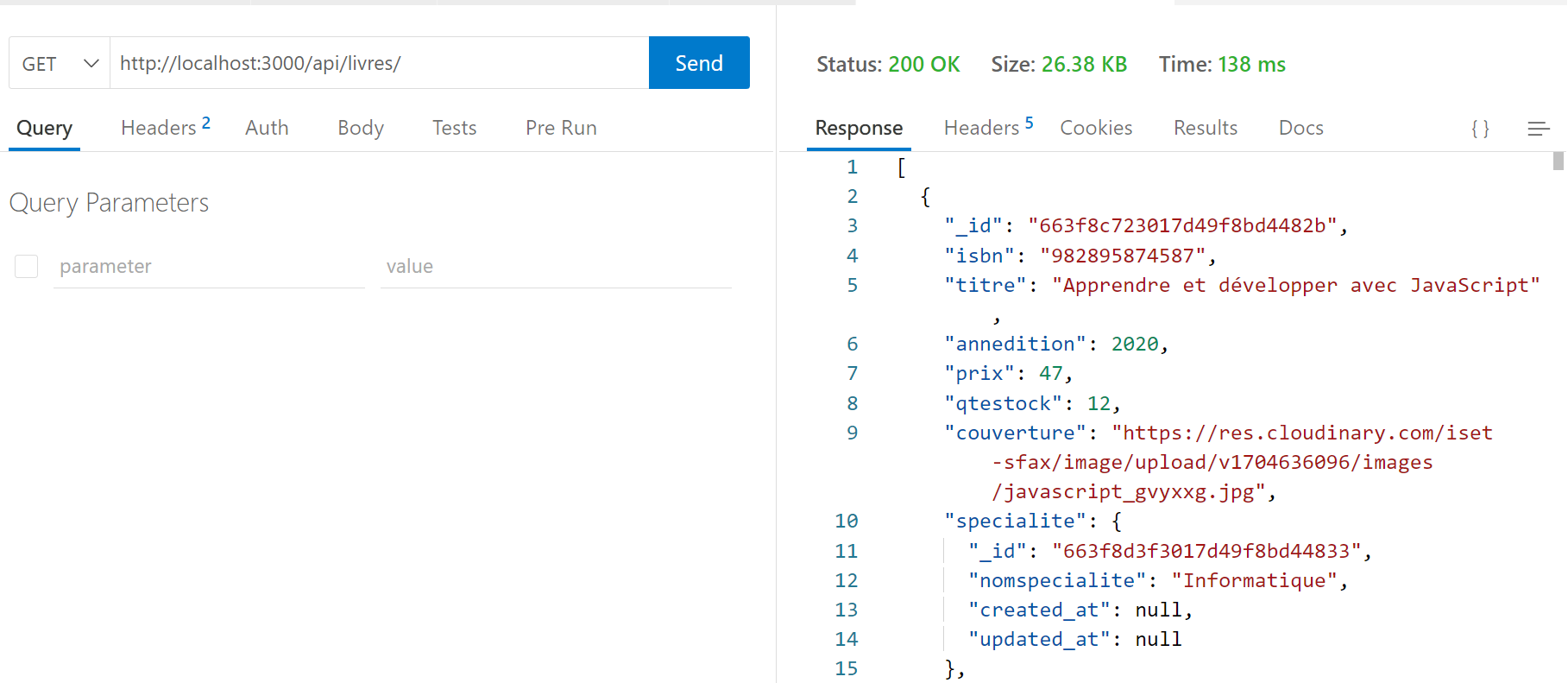
        return NextResponse.json({ error : error });

    }

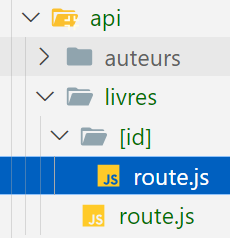
}

Test Api





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



import { HttpStatusCode } from 'axios';

import Livre from '@/models/Livre';

import {  NextResponse } from 'next/server';

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

    try {

        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 {

        const body= await req.json();

        const livre = await Livre.findByIdAndUpdate(

            params.id,

            { $set: body },

          { new: true }

        );

            return NextResponse.json({ livre });

        } catch (error) {

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

    }

}

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

    try {

            await Livre.findByIdAndDelete(params.id);

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

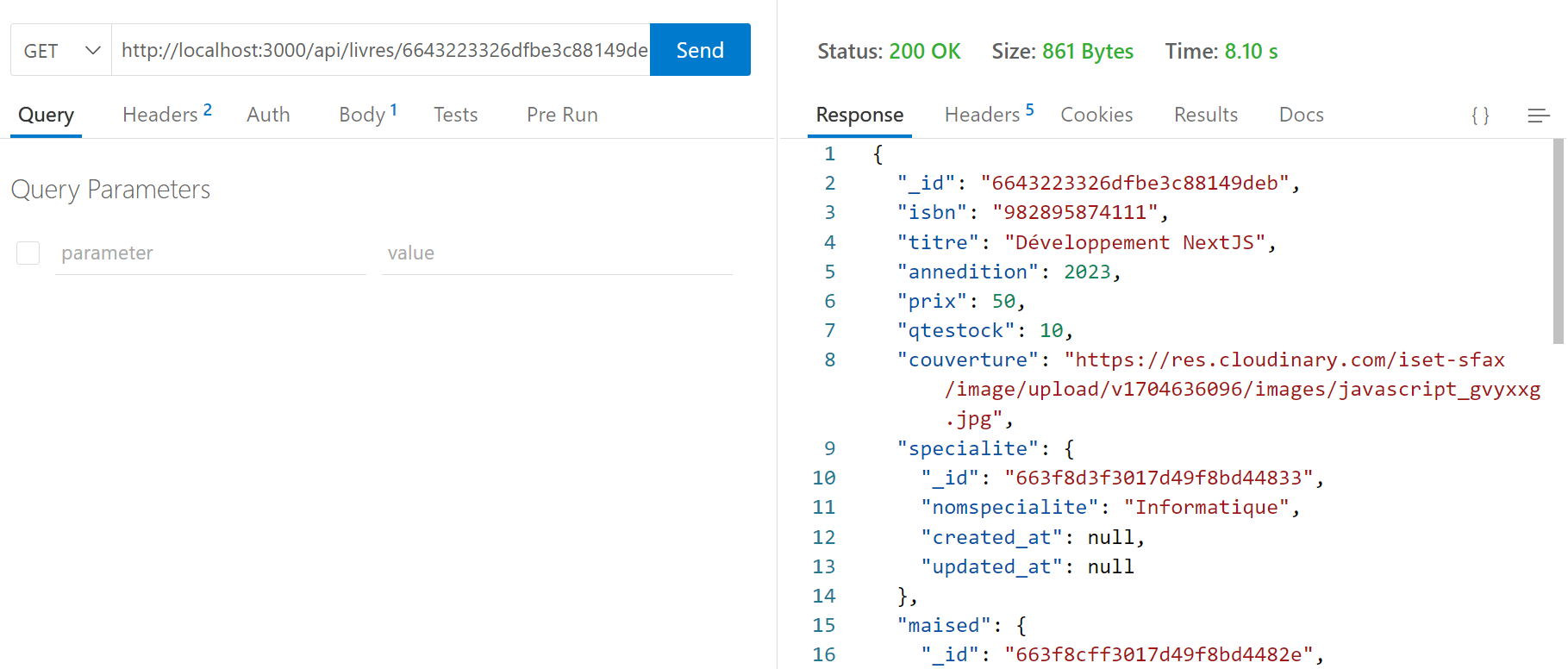
    } catch (error) {

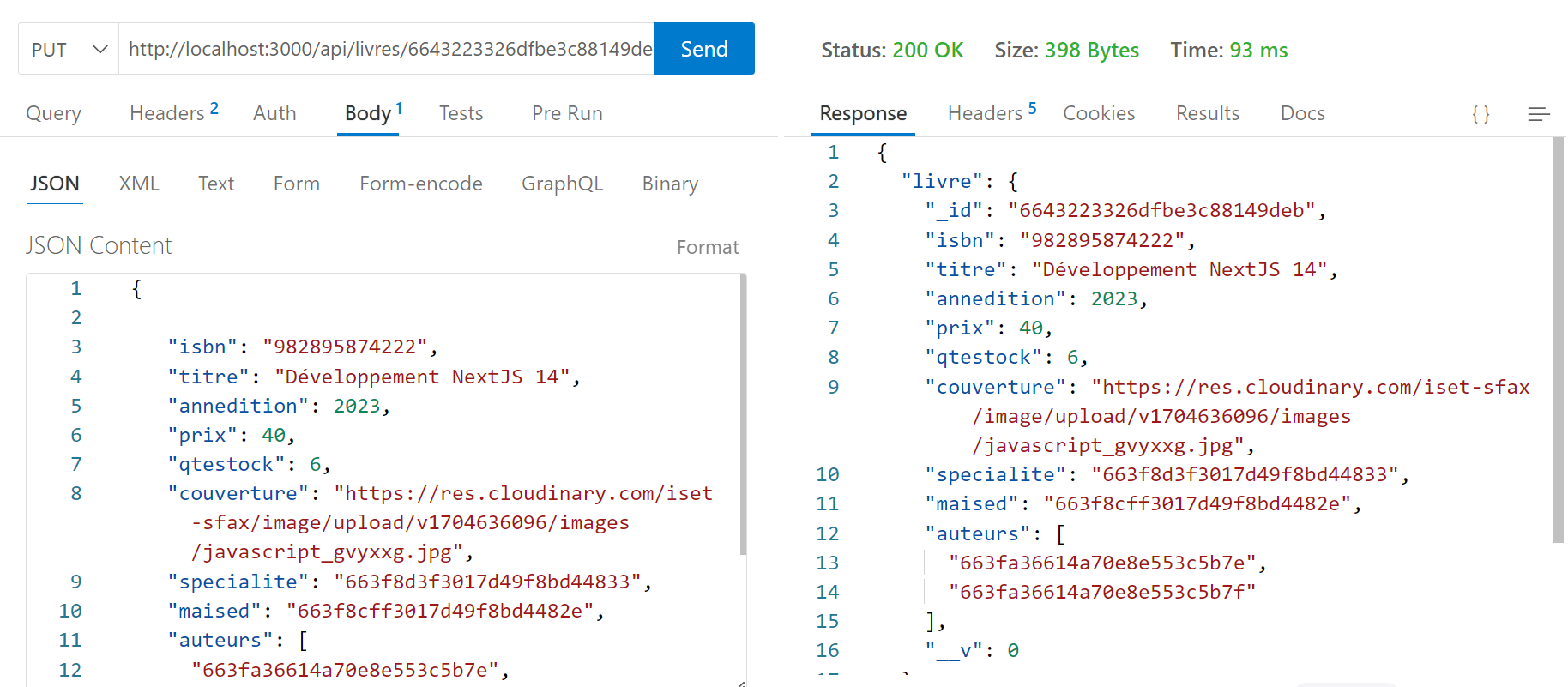
        return NextResponse.json({ message: error }, { status: HttpStatusCode.BadRequest });

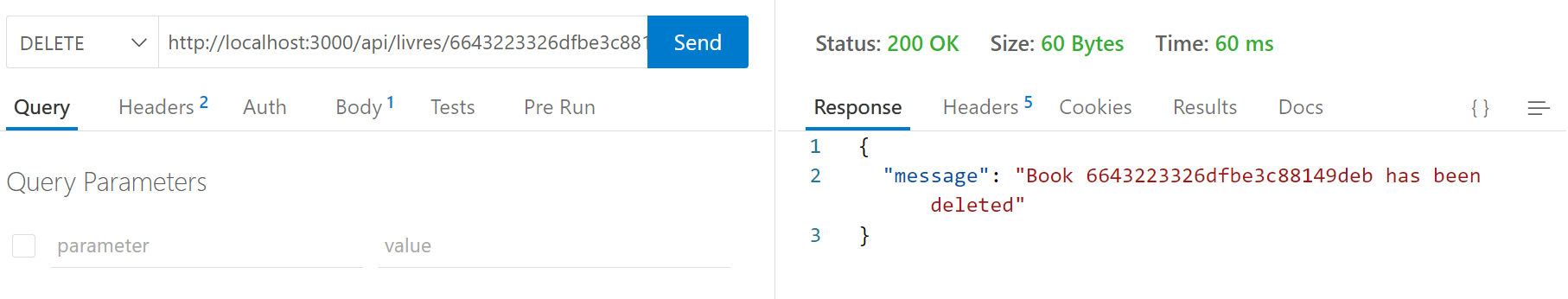
    }

}

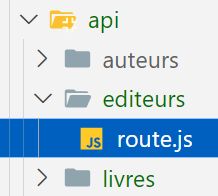
Test Api







## Editeur



import Editeur from '@/models/Editeur';

import {  NextResponse } from 'next/server';

import connectDB from '@/lib/connectDB';

export async function GET() {

    try {

        await connectDB();

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

        return NextResponse.json(editeurs );

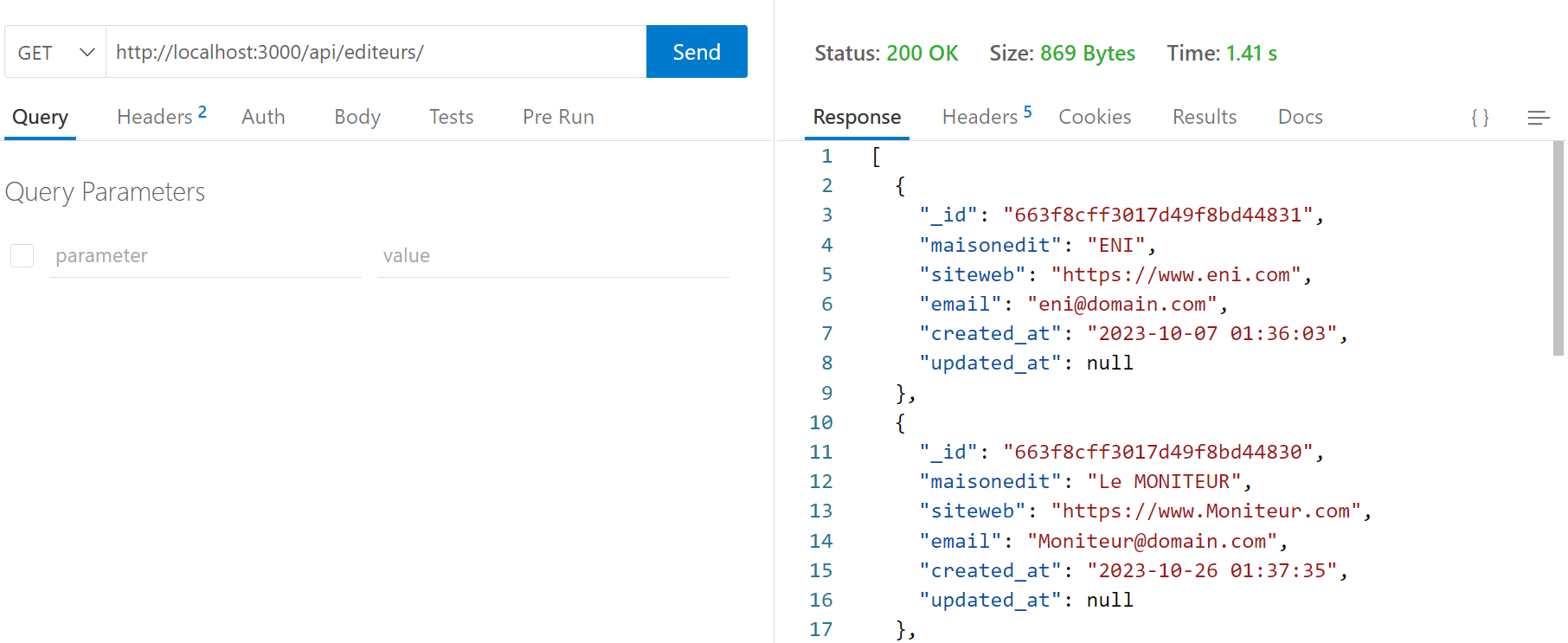
    } catch (error) {

        return NextResponse.json({ error });

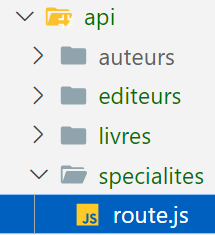
    }

}

Test Api



## Spécialité



import Specialite from '@/models/Specialite';

import {  NextResponse } from 'next/server';

import connectDB from '@/lib/connectDB';

export async function GET() {

    try {

        await connectDB();

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

        return NextResponse.json(specialites );

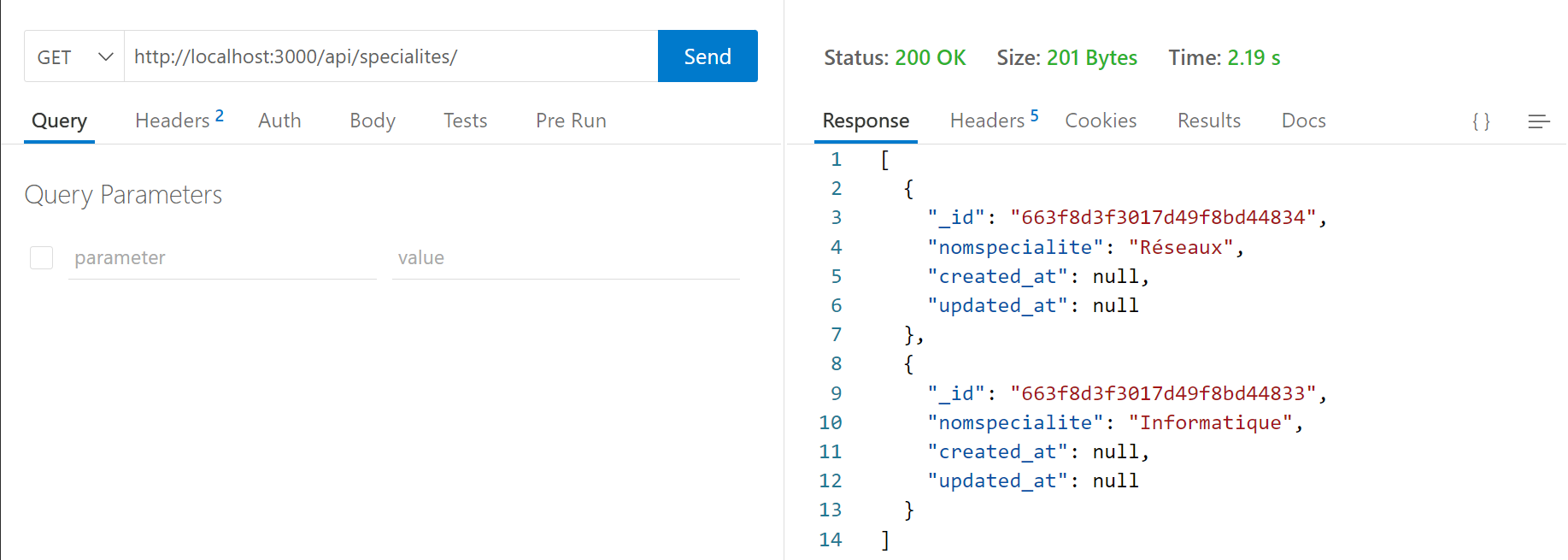
    } catch (error) {

        return NextResponse.json({ error });

    }

}

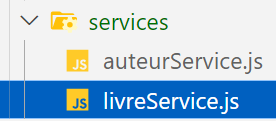
Test Api



# 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),

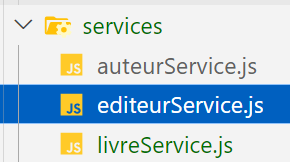
    });

    const response = await res.json();

    return response;

}

## Editeur



const EDITEUR\_API="/editeurs/"

export const fetchEditeurs=async()=> {

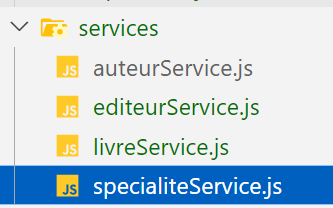
const res = await fetch(process.env.NEXT\_PUBLIC\_API\_URL+EDITEUR\_API, { cache: 'no-store' })

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' })

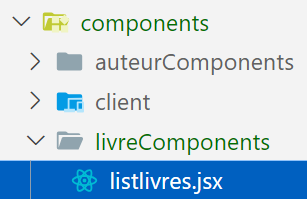
const response = await res.json();

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} />;

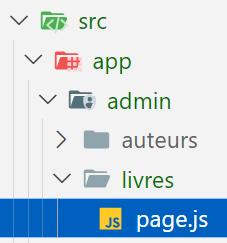
</div>

  )

}

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>

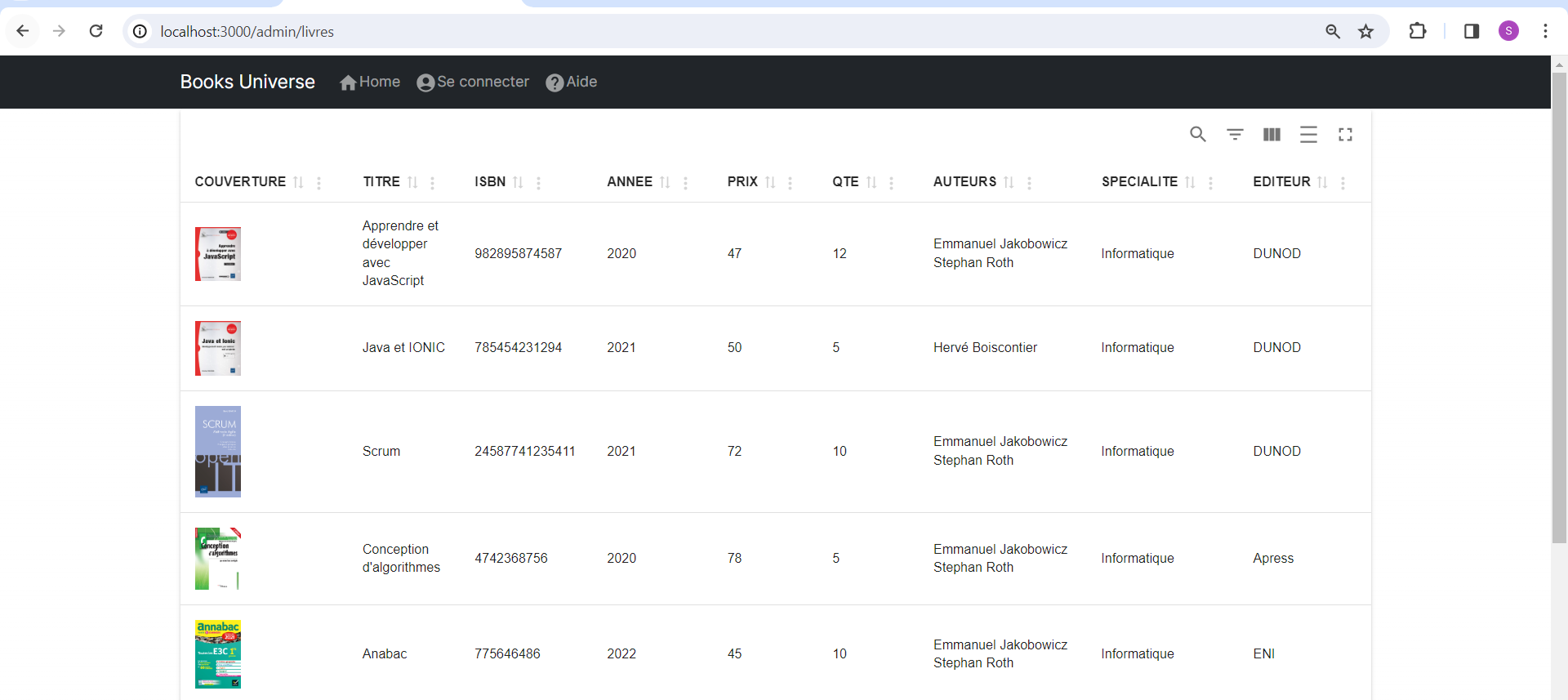
  )

}

export default LivrePage

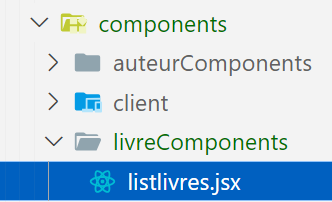
Test dans le navigateur

<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>

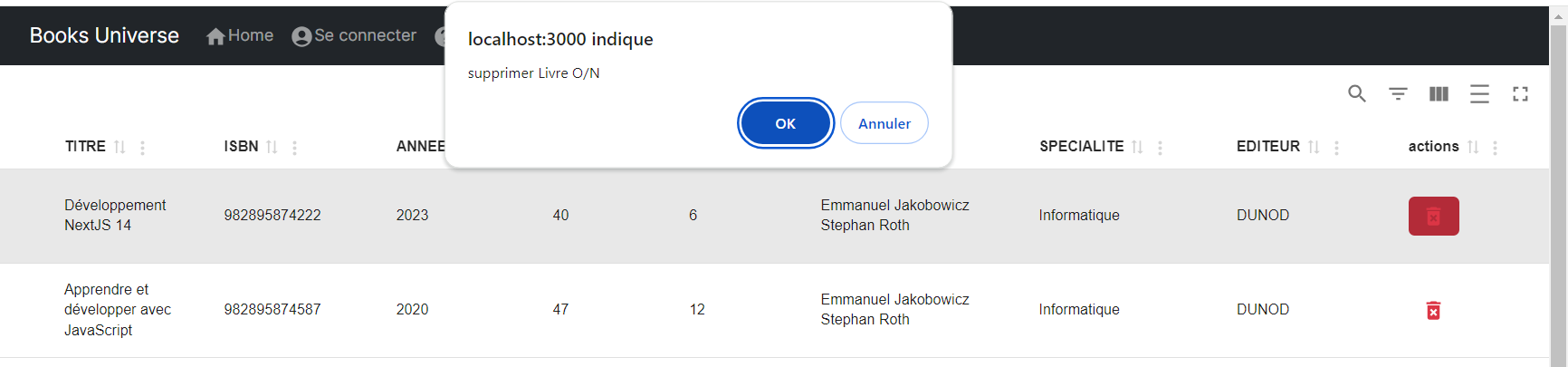
<MaterialReactTable columns={columns} data={livres} />;

</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>

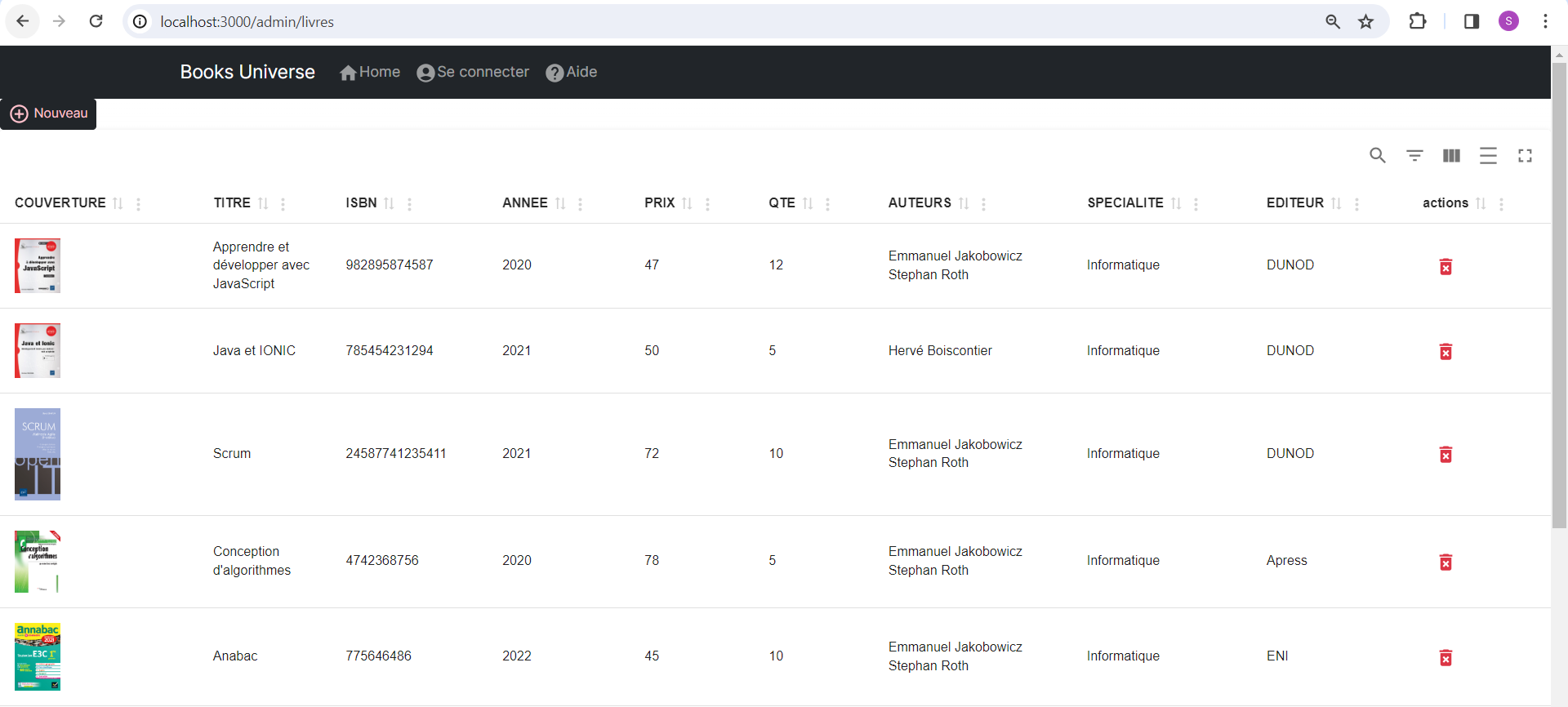
<MaterialReactTable columns={columns} data={livres} />;

</div>

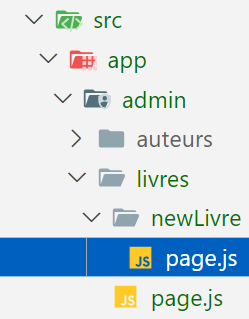
  )

}

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} />

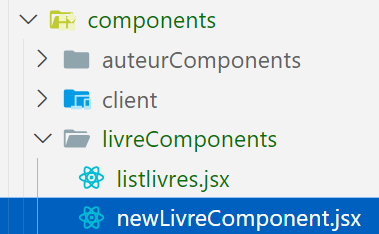
    </div>

  )

}

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(**[]**)

}

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>

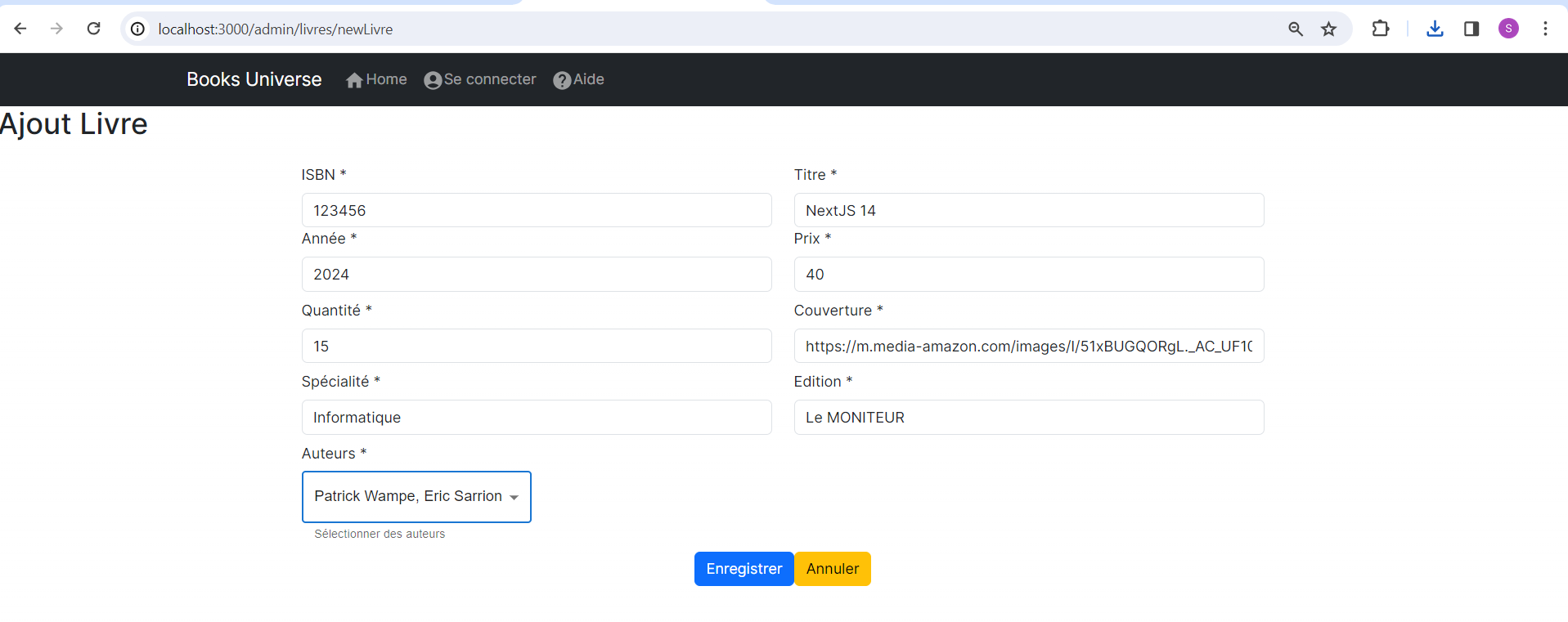
</Form>

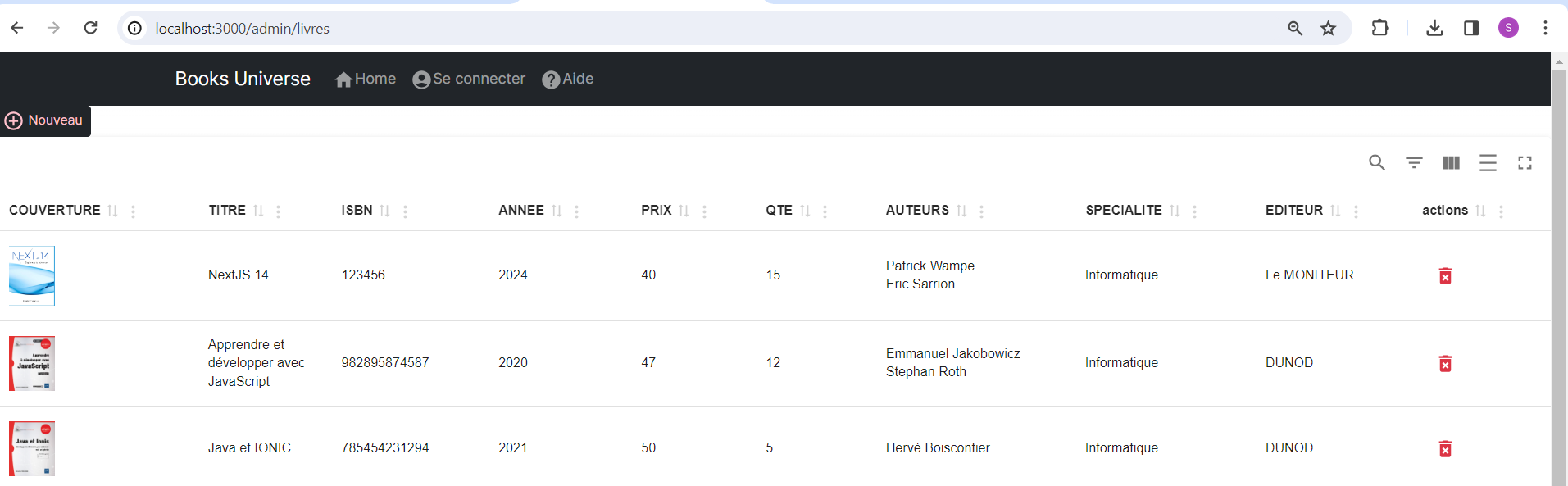
</div>

);

};

export default NewLivre





# Modification des livres

Maintenant, on va procéder à la création du formulaire de la modification.

D’abord, modifier listlivres pour ajouter le bouton Link vers la page edit.



"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';

import EditOutlinedIcon from '@mui/icons-material/EditOutlined';

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

                  size="md"

                  className="text-primary btn-link edit"

                  >

                    <Link href={`/admin/livres/updateLivre/${cell.row.original.\_id}`}>

                    <EditOutlinedIcon/>

                </Link>

                </Button>

                <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} />;

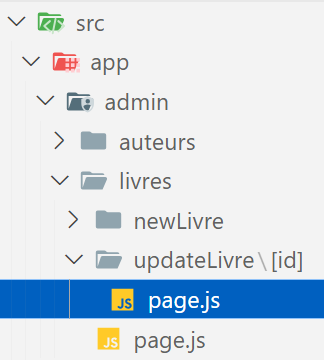
</div>

  )

}

export default Listlivres

Ensuite, sous /src/app/admin/livres créer le dossier updateLivre puis le dossier **[id]** dans lequel créer page.js



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

import UpdateLivre from '@/components/livreComponents/updateLivreComponent';

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

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

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

const getlivre=async(id)=>{

  const data=await fetchLivreById(id)

  return data;

}

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 LivreUpdatePage = async({params}) => {

  const livre = await getlivre(params.id)

  const editeurs=await getEditeurs()

  const specialites=await getSpecialites()

  const auteurs=await getAuteurs()

  return (

    <div>

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

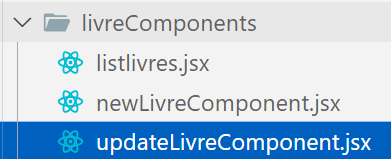
    </div>

  )

}

export default LivreUpdatePage

Ensuite, créer le composant updateLivreComponent.jsx sous /src/components/livreComponents



"use client"

import React, { useState,useEffect } 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 { editLivre } from '@/services/livreService';

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

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

const UpdateLivre = ({livre,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);

useEffect(() => {

    setIsbn(livre.isbn)

    setTitre(livre.titre)

    setAnnedition(livre.annedition)

    setPrix(livre.prix)

    setQtestock(livre.qtestock)

    setCouverture(livre.couverture)

    setSpecialite(livre.specialite.\_id);

    setMaised(livre.maised.\_id);

    //cas liste auteurs

    const tabAut=[]

    livre.auteurs.map((item) =>{

     tabAut.push(item.\_id)

    })

    setAuteurs(tabAut)

 }, [livre]);

const handleSubmit = (e) => {

e.preventDefault();

const form = e.currentTarget;

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

const newLivre = {

**\_id:livre.\_id,**

    isbn,

    titre,

    annedition,

    prix,

    qtestock,

    couverture,

    specialite,

    maised,

    auteurs

};

//faire le update dans la BD

editLivre(newLivre)

.then(res => {

  router.push('/admin/livres')

    router.refresh()

})

.catch(error=>{

alert("Erreur ! Modification non effectuée")

})

}

setValidated(true);

}

const handleReset = () => {

router.push('/admin/livres')

router.refresh()

}}

return (

<div>

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

 <h2>Modification 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>

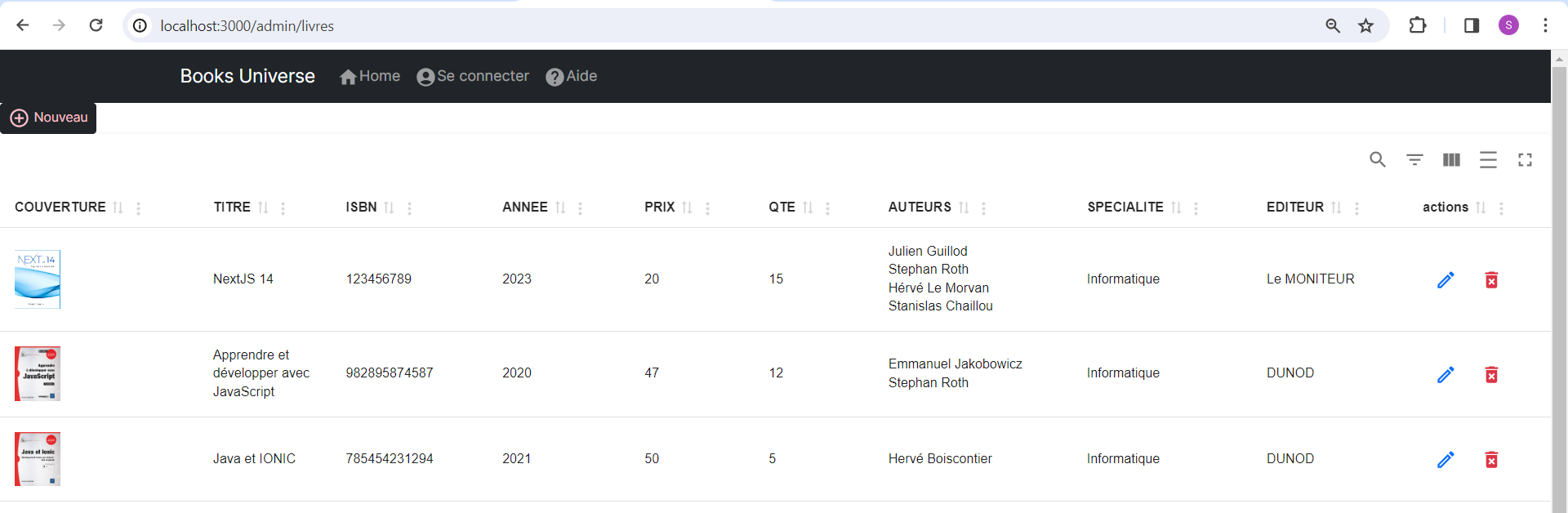
</Form>

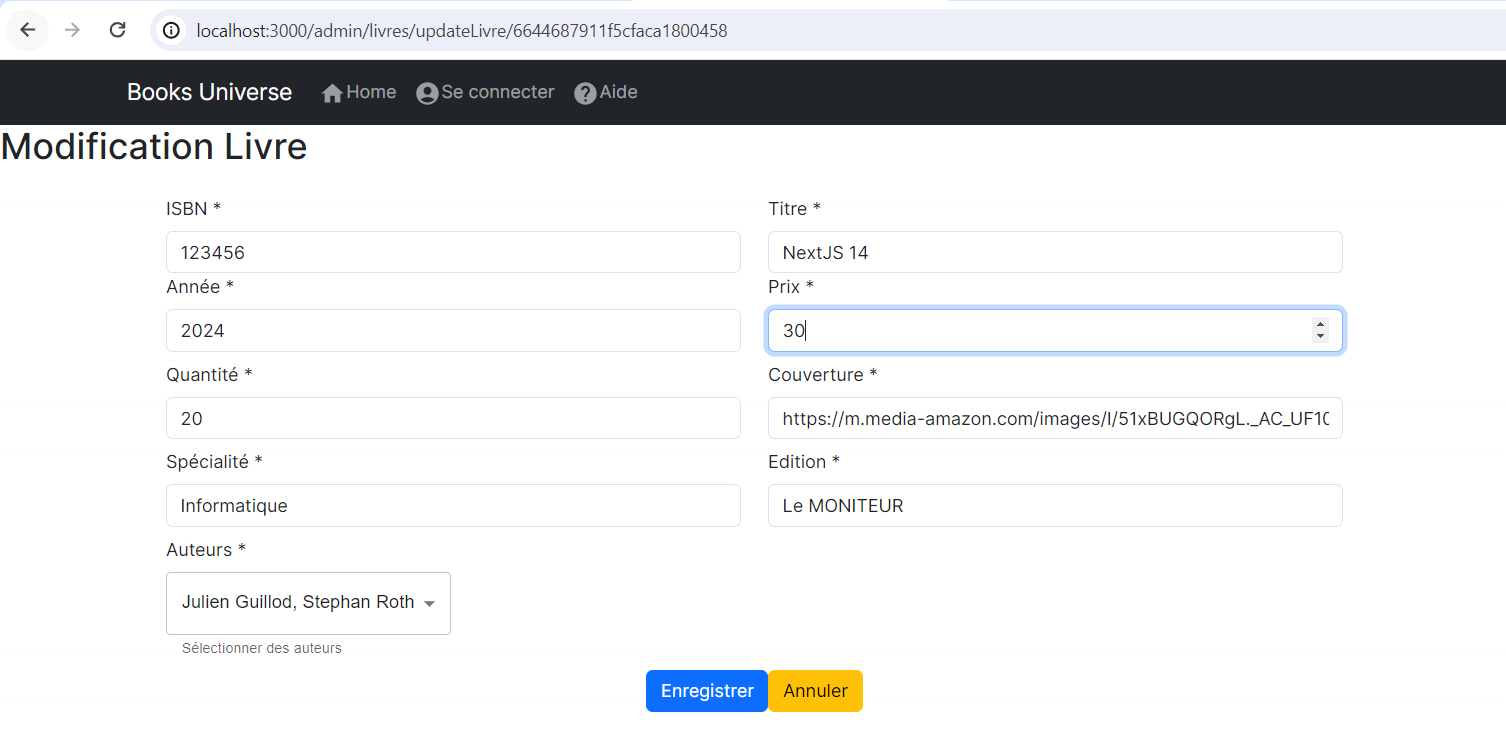
</div>

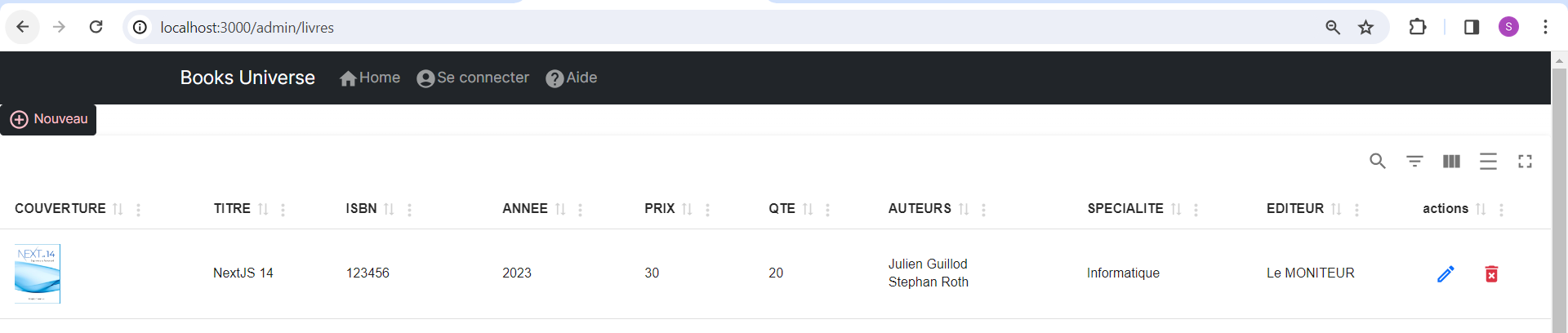
);

};

export default UpdateLivre







# Ajout de livre avec Upload image avec Cloudinary et Filepond

Maintenant on va intégrer l’upload vers Cloudinary. Il s’agit d’une API d'images et de vidéos pour des expériences visuelles puissantes. Vous pourrez y stocker, transformer, optimiser et diffuser tous vos actifs multimédias avec des API, des widgets ou une interface utilisateur faciles à utiliser.

La progression de l’upload sera effectuée avec Filepond qui est une bibliothèque JavaScript qui peut télécharger tout ce que vous lui lancez, optimise les images pour des téléchargements plus rapides et offre une expérience utilisateur exceptionnelle, accessible et fluide.

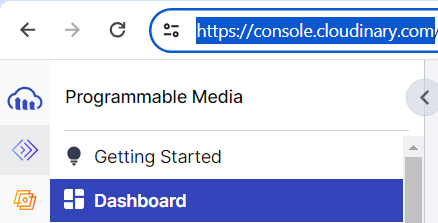
Tout d’abord, il faut créer un compte dans le site Cloudinary en suivant cette URL :

<https://cloudinary.com/>

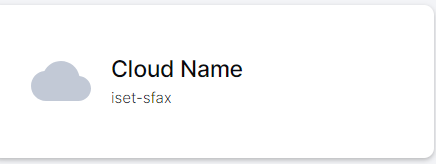
Une fois le compte est créé, accédez à la console https://console.cloudinary.com/

Nous avons 2 données précises à noter qu’on va en avoir besoin par la suite dans le code.

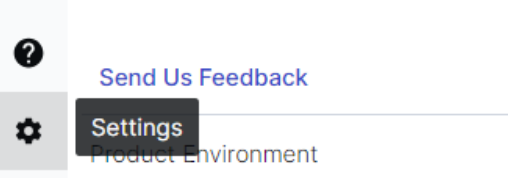
D’abord dans la rubrique Dashboard



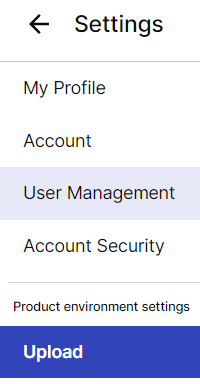
Copiez la valeur du cloudName. Exemple :



Par la suite, accédez à Settings (en bas à gauche)



Puis Upload (Menu à gauche)



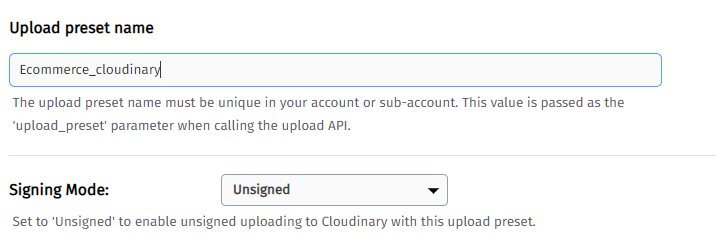
Puis en bas de la page, on retrouvera, « Upload presets »,

Cliquer sur

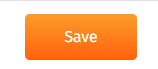


Puis informer le name

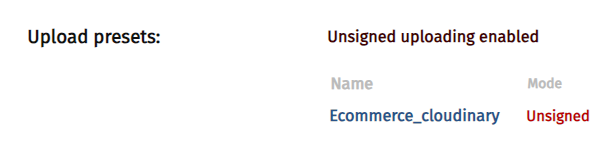
Aussi il faut choisir Unsigned



Puis valider



Il faut aussi copier le nom. Exemple :



Ensuite, dans le projet, installer les packages suivants :

npm i filepond react-filepond filepond-plugin-file-encode filepond-plugin-file-validate-type filepond-plugin-image-exif-orientation filepond-plugin-image-preview filepond-plugin-pdf-preview

Modifier le contenu de NewProduct.jsx

**N.B.** : Veuillez changer les valeurs ci-dessous dans le code par les paramètres de votre compte Cloudinary.



"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';

import axios from 'axios';

import { FilePond,registerPlugin } from 'react-filepond'

import 'filepond/dist/filepond.min.css';

import FilePondPluginImageExifOrientation from 'filepond-plugin-image-exif-orientation'

import FilePondPluginImagePreview from 'filepond-plugin-image-preview'

import 'filepond-plugin-image-preview/dist/filepond-plugin-image-preview.css'

registerPlugin(FilePondPluginImageExifOrientation, FilePondPluginImagePreview)

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

const [files, setFiles] = useState([]);

    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(**[]**)

setFiles(**[]**)

}

const serverOptions = () => { console.log('server pond');

  return {

    process: (fieldName, file, metadata, load, error, progress, abort) => {

        console.log(file)

      const data = new FormData();

      data.append('file', file);

      data.append('upload\_preset', 'Ecommerce\_cloudinary');

      data.append('cloud\_name', 'iset-sfax');

      data.append('public\_id', file.name);

      axios.post('https://api.cloudinary.com/v1\_1/iset-sfax/image/upload', data)

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

        .then((data) => {

          console.log(data);

         setCouverture(data.url) ;

          load(data);

        })

        .catch((error) => {

          console.error('Error uploading file:', error);

          error('Upload failed');

          abort();

        });

    },

  };

};

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>

<div style={{ width: "80%", margin: "auto", padding: "1%" }}>

     <FilePond

                   files={files}

                   acceptedFileTypes="image/\*"

                   onupdatefiles={setFiles}

                   allowMultiple={false}

                   server={serverOptions()}

                   name="file"

          />

    </div>

</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>

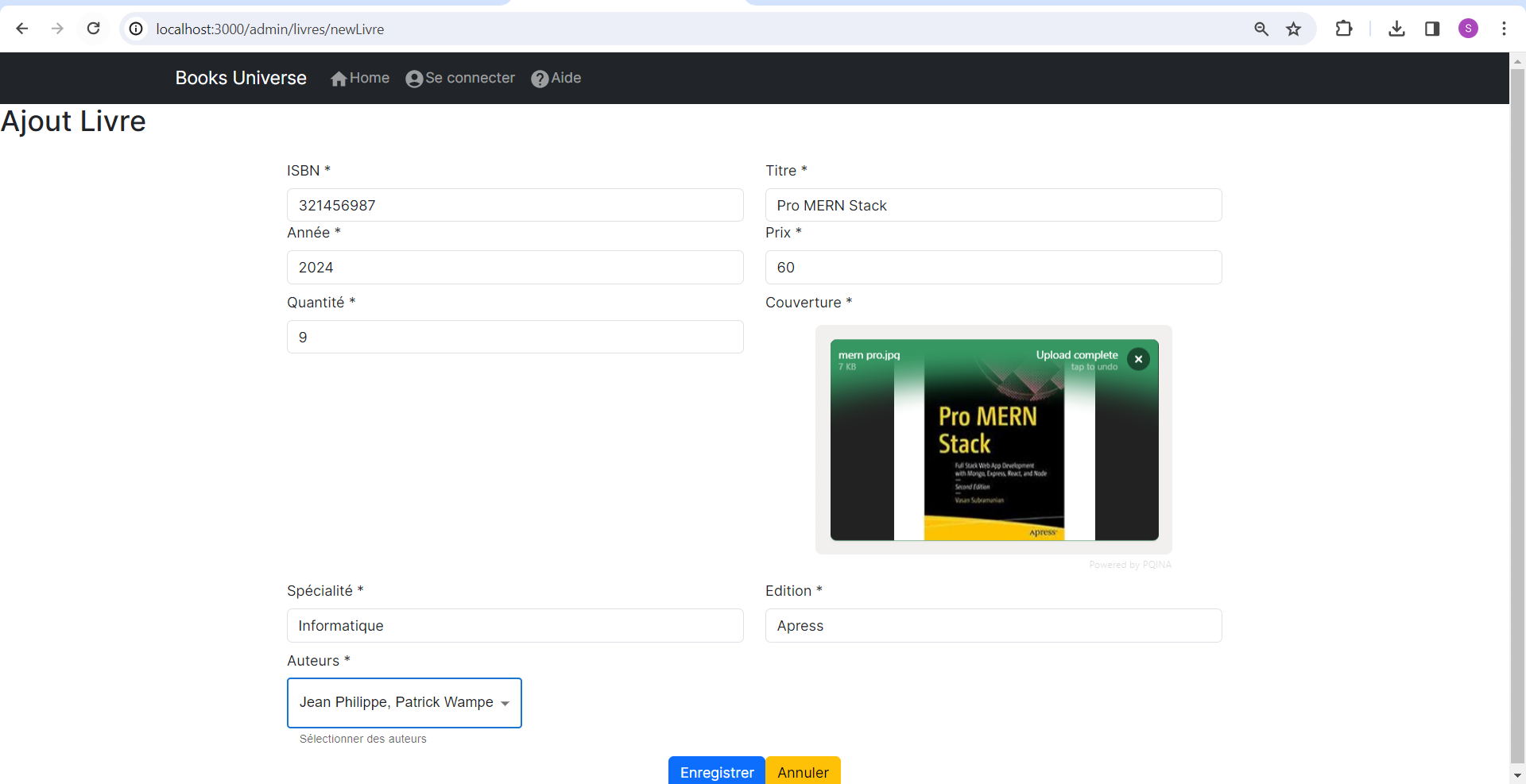
</Form>

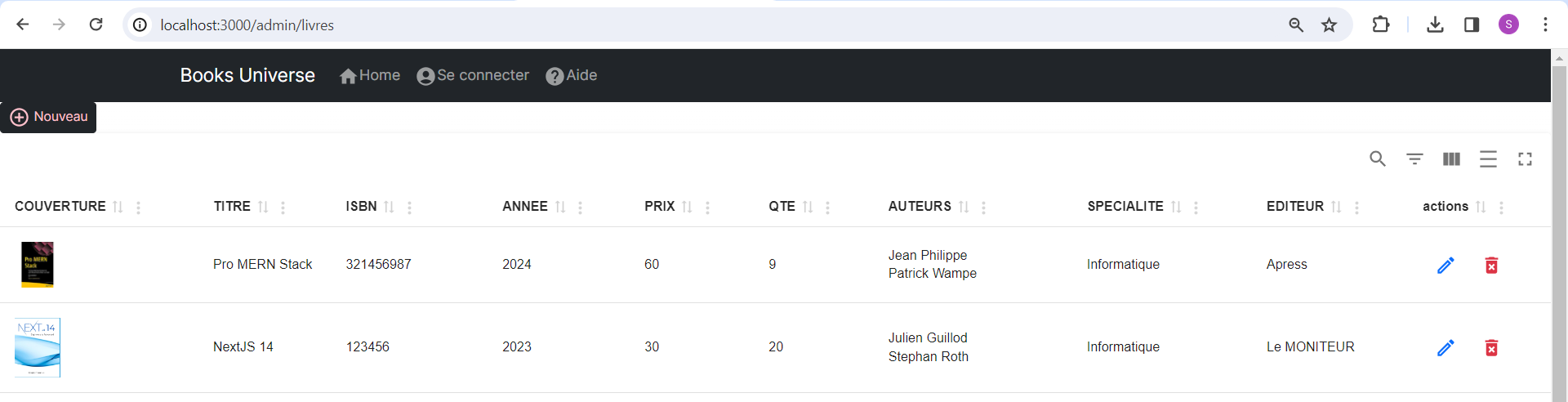
</div>

);

};

export default NewLivre





# Modification livre avec Upload image avec Cloudinary et Filepond



"use client"

import React, { useState,useEffect } 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 { editLivre } from '@/services/livreService';

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

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

import axios from 'axios';

import { FilePond,registerPlugin } from 'react-filepond'

import 'filepond/dist/filepond.min.css';

import FilePondPluginImageExifOrientation from 'filepond-plugin-image-exif-orientation'

import FilePondPluginImagePreview from 'filepond-plugin-image-preview'

import 'filepond-plugin-image-preview/dist/filepond-plugin-image-preview.css'

registerPlugin(FilePondPluginImageExifOrientation, FilePondPluginImagePreview)

const UpdateLivre = ({livre,LesEditeurs,lesSpecialites,lesAuteurs}) => {

const [files, setFiles] = useState([]);

    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);

useEffect(() => {

    setIsbn(livre.isbn)

    setTitre(livre.titre)

    setAnnedition(livre.annedition)

    setPrix(livre.prix)

    setQtestock(livre.qtestock)

    setCouverture(livre.couverture)

    setSpecialite(livre.specialite.\_id);

    setMaised(livre.maised.\_id);

    //cas liste auteurs

    const tabAut=[]

    livre.auteurs.map((item) =>{

     tabAut.push(item.\_id)

    })

    setAuteurs(tabAut)

    //cas Filepond

    setFiles( [

        {

          source: livre.couverture,

          options: { type: 'local' }

        }

        ])

 }, [livre]);

const handleSubmit = (e) => {

e.preventDefault();

const form = e.currentTarget;

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

const newLivre = {

    \_id:livre.\_id,

    isbn,

    titre,

    annedition,

    prix,

    qtestock,

    couverture,

    specialite,

    maised,

    auteurs

};

//faire le update dans la BD

editLivre(newLivre)

.then(res => {

  router.push('/admin/livres')

    router.refresh()

})

.catch(error=>{

alert("Erreur ! Modification non effectuée")

})

}

setValidated(true);

}

const handleReset = () => {

    router.push('/admin/livres')

    router.refresh()

}

const serverOptions = () => { console.log('server pond');

return {

  load: (source, load, error, progress, abort, headers) => {

      var myRequest = new Request(source);

      fetch(myRequest).then(function(response) {

        response.blob().then(function(myBlob) {

          load(myBlob);

        });

      });

    },

  process: (fieldName, file, metadata, load, error, progress, abort) => {

      console.log(file)

    const data = new FormData();

    data.append('file', file);

    data.append('upload\_preset', 'Ecommerce\_cloudinary');

    data.append('cloud\_name', 'iset-sfax');

    data.append('public\_id', file.name);

    axios.post('https://api.cloudinary.com/v1\_1/iset-sfax/image/upload', data)

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

      .then((data) => {

        console.log(data);

       setCouverture(data.url) ;

        load(data);

      })

      .catch((error) => {

        console.error('Error uploading file:', error);

        error('Upload failed');

        abort();

      });

  },

};

};

return (

<div>

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

 <h2>Modification 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>

<div style={{ width: "80%", margin: "auto", padding: "1%" }}>

     <FilePond

                   files={files}

                   acceptedFileTypes="image/\*"

                   onupdatefiles={setFiles}

                   allowMultiple={false}

                   server={serverOptions()}

                   name="file"

          />

    </div>

</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 UpdateLivre

