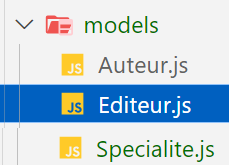
Atelier 5 NextJS : CRUD éditeurs avec Modal

# Model

On suppose que le model est déjà créé (voir l’atelier 4) /src/models/Editeur.js



import mongoose from "mongoose"

var editeurSchema = mongoose.Schema({

    maisonedit: String,

    siteweb: String,

    email: String

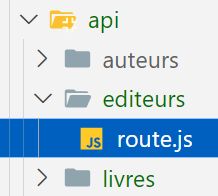
})

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

export default Editeur

# API Routes

Modifiez le contenu de /src/app/api/editeurs/route.js



import Editeur from '@/models/Editeur';

import {  NextResponse } from 'next/server';

import { HttpStatusCode } from 'axios';

  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 });

    }

}

export async function POST(req) {

    try {

        const body = await req.json();

        const newEditeur = new Editeur(body)

        const editeur = await newEditeur.save();

        return NextResponse.json(

                { editeur, message: 'Your editor has been created' },

                { status: HttpStatusCode.Created },

            );

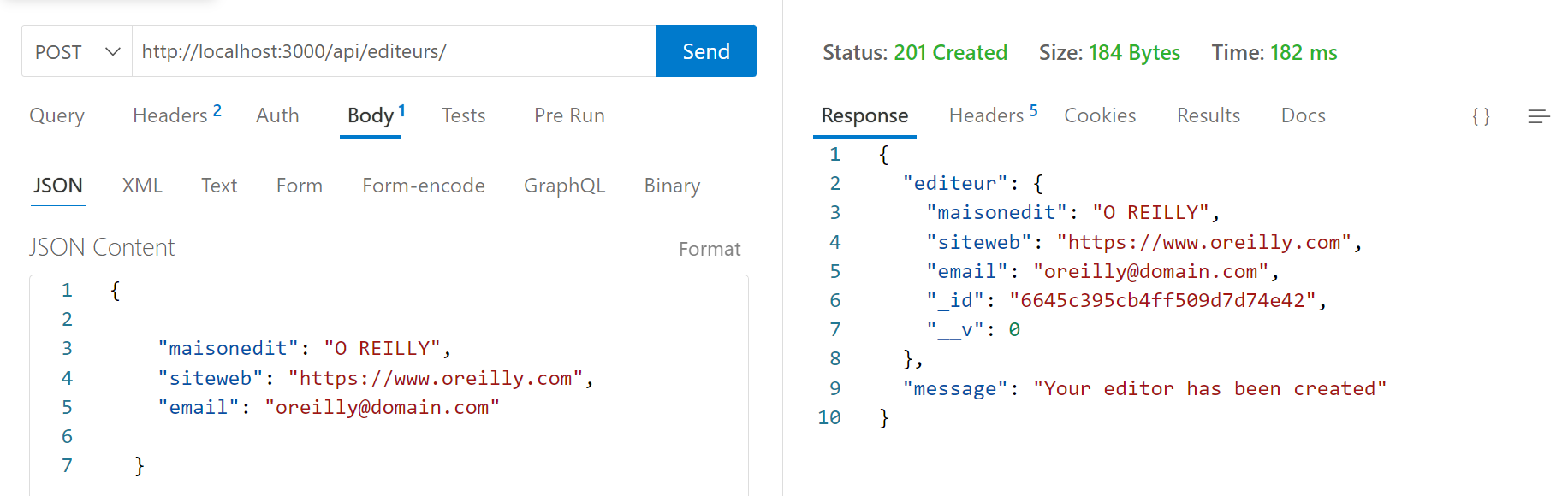
    } catch (error) {

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

    }

}

Test Api



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

import { HttpStatusCode } from 'axios';

import Editeur from '@/models/Editeur';

import {  NextResponse } from 'next/server';

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

    try {

        const editeur = await Editeur.findById(params.id);

        if (editeur) {

            return NextResponse.json(editeur);

        }

        return NextResponse.json({ message: `Editor ${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 editeur = await Editeur.findByIdAndUpdate(

            params.id,

            { $set: body },

          { new: true }

        );

            return NextResponse.json({ editeur });

         } catch (error) {

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

    }

}

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

    try {

        await Editeur.findByIdAndDelete(params.id);

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

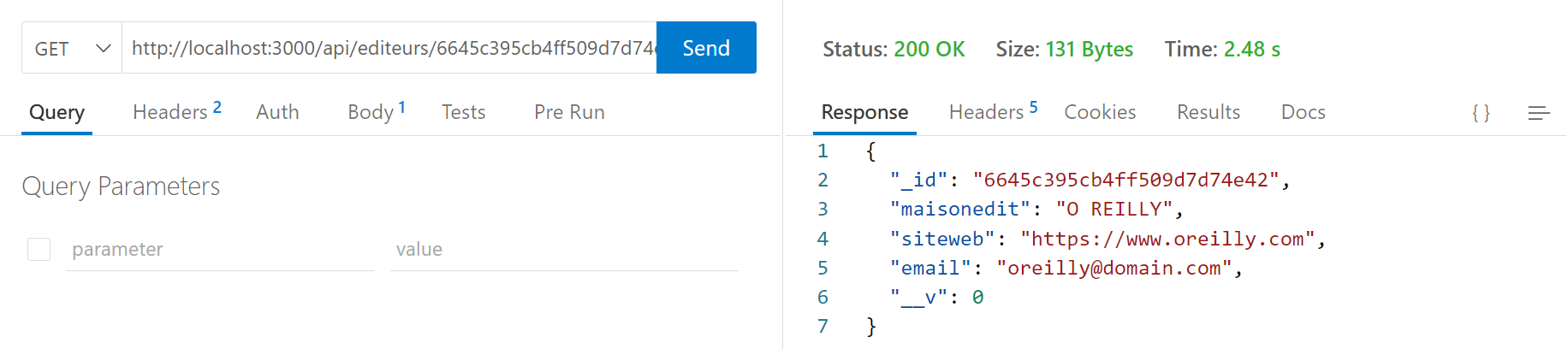
    } catch (error) {

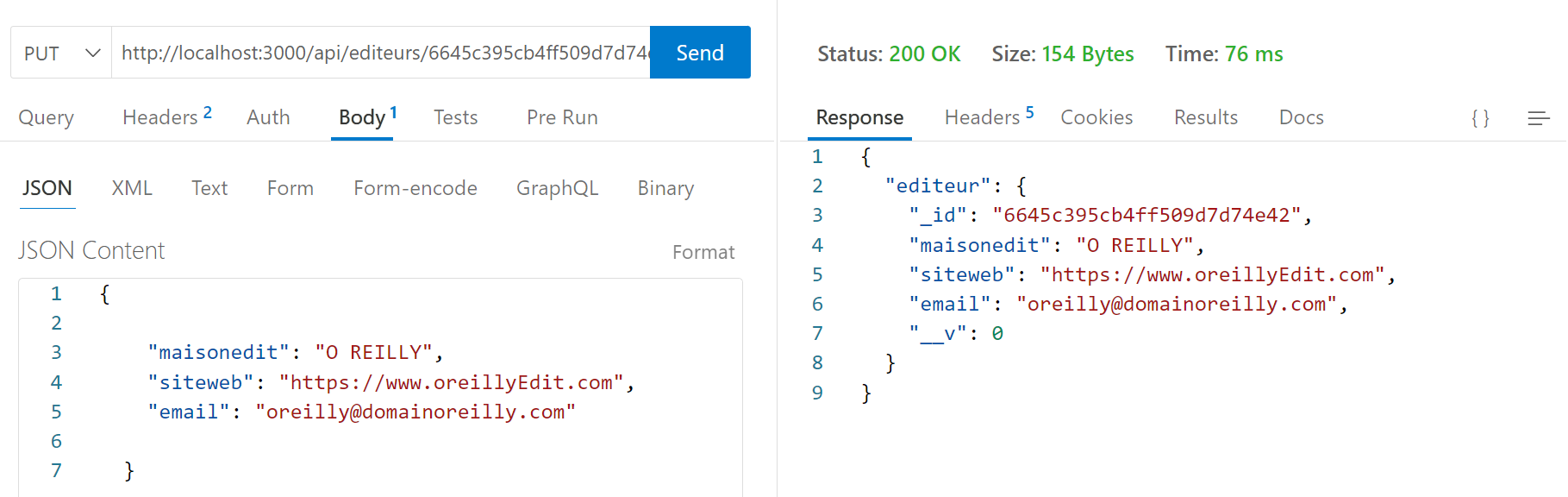
        return NextResponse.json({ message: error }, { status: HttpStatusCode.BadRequest });

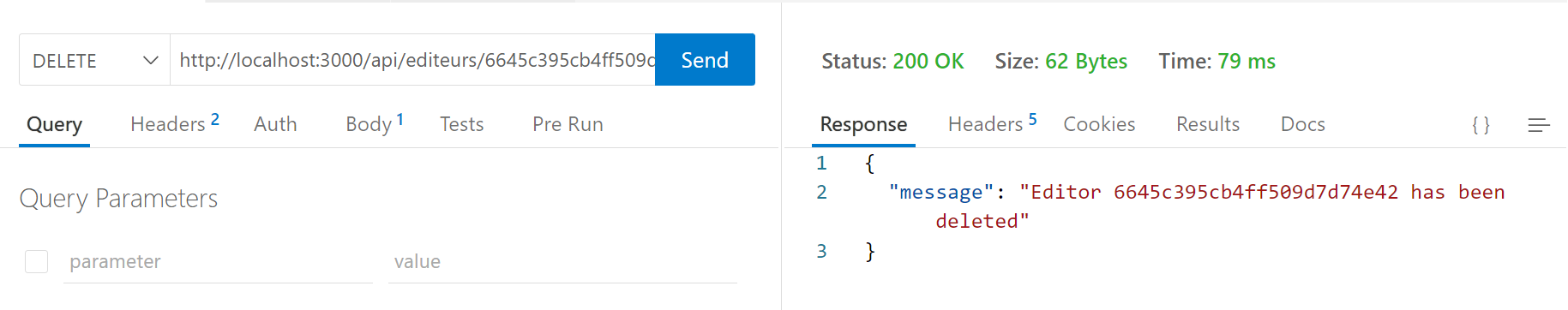
    }

}

Test Api

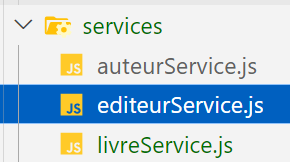






# Services

Modifier le contenu de /src/services/editeurService.js



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;

}

export const fetchEditeurById=async(editeurId)=> {

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

        method: 'GET'

    });

    const response = await res.json();

     return response;

}

export const deleteEditeur=async(editeurId) =>{

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

        method: 'DELETE'

    });

    const response = await res.json();

    return response;

}

export const addEditeur=async(editeur)=> {

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

        method: 'POST',

        headers: {

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

        },

        body: JSON.stringify(editeur),

    });

    const response = await res.json();

    return response;

}

export const editEditeur=async(editeur) =>{

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

        method: 'PUT',

        headers: {

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

        },

        body: JSON.stringify(editeur),

    });

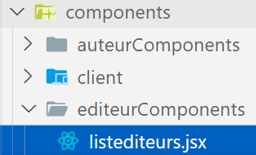
    const response = await res.json();

    return response;

}

# Affichage et suppression des éditeurs

Sous /src/components/ créer le dossier editeurComponents puis créer le fichier listediteurs.jsx



"use client"

import  { useMemo,useState } from 'react';

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

import Button from 'react-bootstrap/Button';

import {deleteEditeur} from "@/services/editeurService"

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

const Listediteurs = ({editeurs}) => {

  const [LesEditeurs,setLesEditeurs]=useState(editeurs)

    const deleteediteur=(id)=>{

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

        deleteEditeur(id)

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

         setLesEditeurs(LesEditeurs.filter((item)=>item.\_id!==id))

      })

        .catch(error=>{

            console.log(error)

           })

  }

        }

  const columns = useMemo(

    () => [

    {

    accessorKey: 'maisonedit',

    header: 'NOM',

    size: 100,

    },

    {

        accessorKey: 'siteweb',

        header: 'SITE',

        size: 100,

        },

    {

    accessorKey: 'email',

    header: 'EMAIL',

    size: 100,

    },

   {

      accessorKey: '\_id',

      header: 'actions',

      size: 100,

      Cell: ({ cell, row }) => (

      <div >

     <Button

      onClick={(e) => {

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

      }}

      variant="danger"

      size="md"

      className="text-danger btn-link delete"

      >

      <DeleteForeverIcon />

      </Button>

      </div>

      ),

      },

    ],

    [LesEditeurs],

    );

return (

<div>

<MaterialReactTable columns={columns} data={LesEditeurs} />

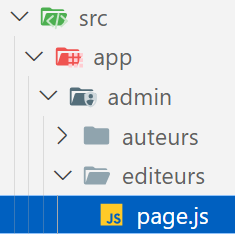
</div>

  )

}

export default Listediteurs

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



import Listediteurs from '@/components/editeurComponents/listediteurs';

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

const getEditeurs=async()=>{

const data=await fetchEditeurs()

return data;

}

const EditeurPage = async() =>{

    const editeurs=await getEditeurs()

  return (

   <div>

      <Listediteurs editeurs={editeurs}/>

    </div>

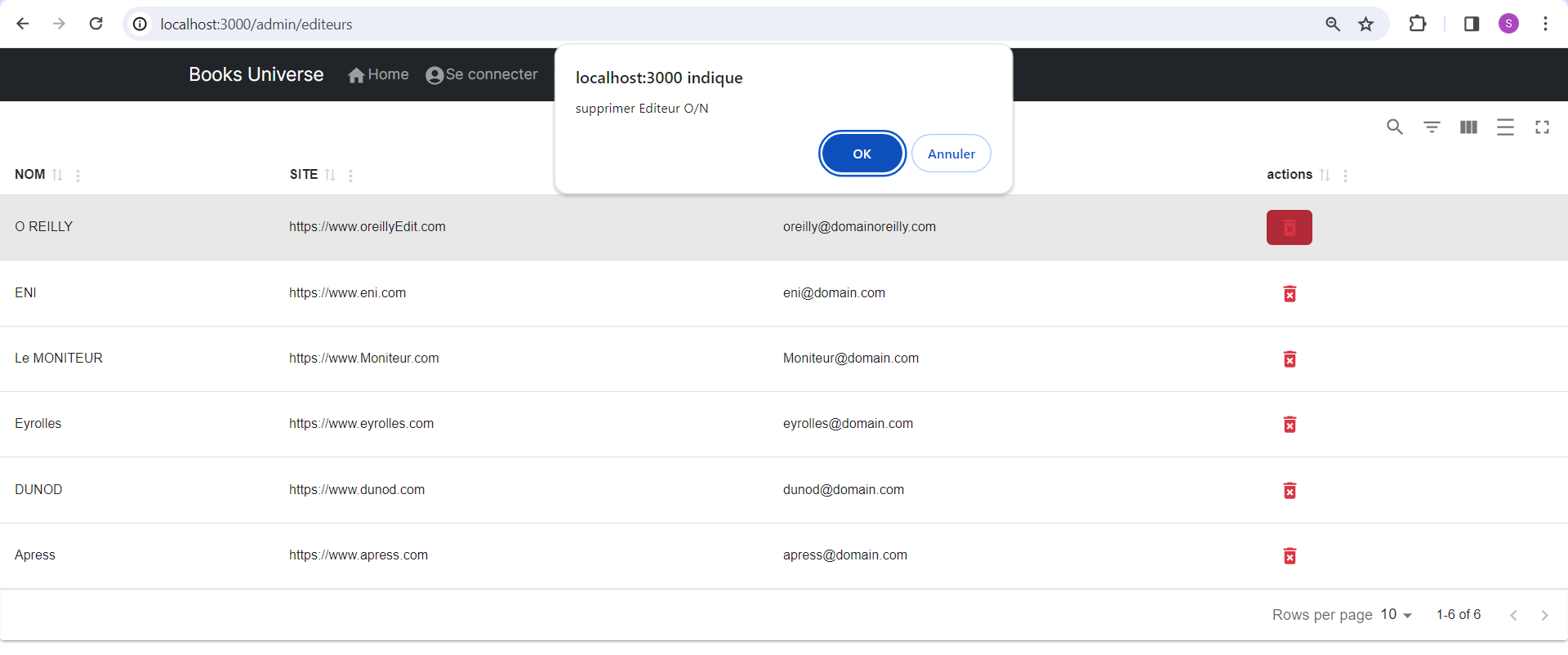
  )

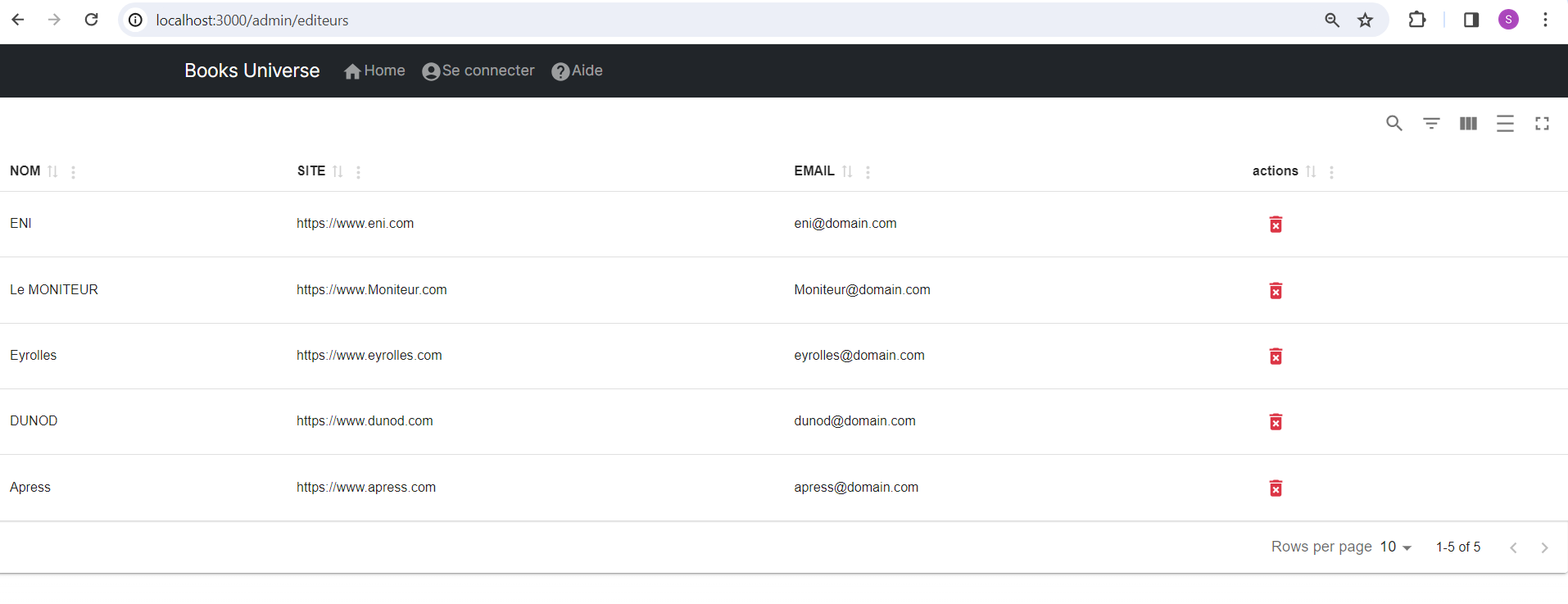
}

export default EditeurPage

Test dans le navigateur

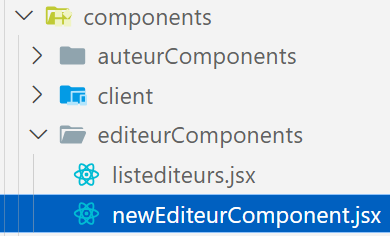
<http://localhost:3000/admin/editeurs>





# Modal ajout éditeur

Créer sous /src/components/editeurComponents le fichier newEditeurComponent.jsx



import React, { useState } from 'react';

import Button from 'react-bootstrap/Button';

import Form from 'react-bootstrap/Form';

import Col from 'react-bootstrap/Col';

import Row from 'react-bootstrap/Row';

import Modal from 'react-bootstrap/Modal';

import {addEditeur} from "@/services/editeurService"

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

const Insertediteur =({LesEditeurs,setLesEditeurs})=> {

  const [editeur,setEditeur]=useState({})

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

const [show, setShow] = useState(false);

const handleClose = () => setShow(false);

const handleShow = () => setShow(true);

const handlechange=(e)=>{

  setEditeur({...editeur,[e.target.name]:e.target.value})

  }

  const  handleSubmit = (e) => {

    e.preventDefault();

    const form = e.currentTarget;

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

                //faire le add dans la BD

               addEditeur(editeur).then((res) => {

                  console.log(res);

                  setLesEditeurs([**res.editeur**,...LesEditeurs]);

                handleReset()

                setValidated(false);

                }).catch((err) => {window.alert(err)})

            }

    setValidated(true);

    }

    const handleReset=()=>{

      setEditeur({})

        handleClose()

        }

        return (

            <div >

            <nav className="navbar navbar-expand-lg navbar-dark ">

                    <div className="container-fluid">

                    <Button

             onClick={handleShow}

            className="btn btn-outline-light"

             style={{float: 'left','margin':10,'left':10,fontFamily:'Arial'}}>

               <AddCircleOutlineIcon/> Nouveau

            </Button>

                    </div>

                  </nav>

<Modal show={show} onHide={handleClose}>

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

<Modal.Header closeButton>

 <h2>Ajouter un Editeur</h2>

</Modal.Header>

<Modal.Body>

<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="12" >

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

<Form.Control

required

type="text"

placeholder="Nom"

name="maisonedit"

value={editeur.maisonedit}

onChange={(e)=>handlechange(e)}

/>

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

Saisir Nom de l'éditeur

</Form.Control.Feedback>

</Form.Group>

</Row>

</div>

<div className='form mt-3'>

<Row className="mb-2">

<Form.Group as={Col} md="12" >

<Form.Label >Site web \*</Form.Label>

<Form.Control

required

type="url"

placeholder="Site"

name="siteweb"

value={editeur.siteweb}

onChange={(e)=>handlechange(e)}

/>

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

Saisir Site de l'éditeur

</Form.Control.Feedback>

</Form.Group>

</Row>

</div>

<div className='form mt-3'>

<Row className="mb-2">

<Form.Group as={Col} md="12" >

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

<Form.Control

required

type="email"

placeholder="Email"

name="email"

value={editeur.email}

onChange={(e)=>handlechange(e)}

/>

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

Saisir Email de l'éditeur

</Form.Control.Feedback>

</Form.Group>

</Row>

</div>

</div>

</div>

</Modal.Body>

<Modal.Footer>

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

<Button type="button" className="btn btn-warning"onClick={()=>handleReset()}>Annuler</Button>

</Modal.Footer>

</Form>

</Modal>

</div>

  )

}

export default Insertediteur

Puis modifier le contenu de listeediteurs.jsx



"use client"

import  { useMemo,useState } from 'react';

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

import Button from 'react-bootstrap/Button';

import {deleteEditeur} from "@/services/editeurService"

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

import NewEditeurComponent from './newEditeurComponent';

const Listediteurs = ({editeurs}) => {

  const [LesEditeurs,setLesEditeurs]=useState(editeurs)

    const deleteediteur=(id)=>{

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

        deleteEditeur(id)

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

         setLesEditeurs(LesEditeurs.filter((item)=>item.\_id!==id))

      })

        .catch(error=>{

            console.log(error)

           })

  }

        }

  const columns = useMemo(

    () => [

    {

    accessorKey: 'maisonedit',

    header: 'NOM',

    size: 100,

    },

    {

        accessorKey: 'siteweb',

        header: 'SITE',

        size: 100,

        },

    {

    accessorKey: 'email',

    header: 'EMAIL',

    size: 100,

    },

   {

      accessorKey: '\_id',

      header: 'actions',

      size: 100,

      Cell: ({ cell, row }) => (

      <div >

     <Button

      onClick={(e) => {

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

      }}

      variant="danger"

      size="md"

      className="text-danger btn-link delete"

      >

      <DeleteForeverIcon />

      </Button>

      </div>

      ),

      },

    ],

    [LesEditeurs],

    );

return (

<div>

    <NewEditeurComponent LesEditeurs={LesEditeurs} setLesEditeurs={setLesEditeurs}/>

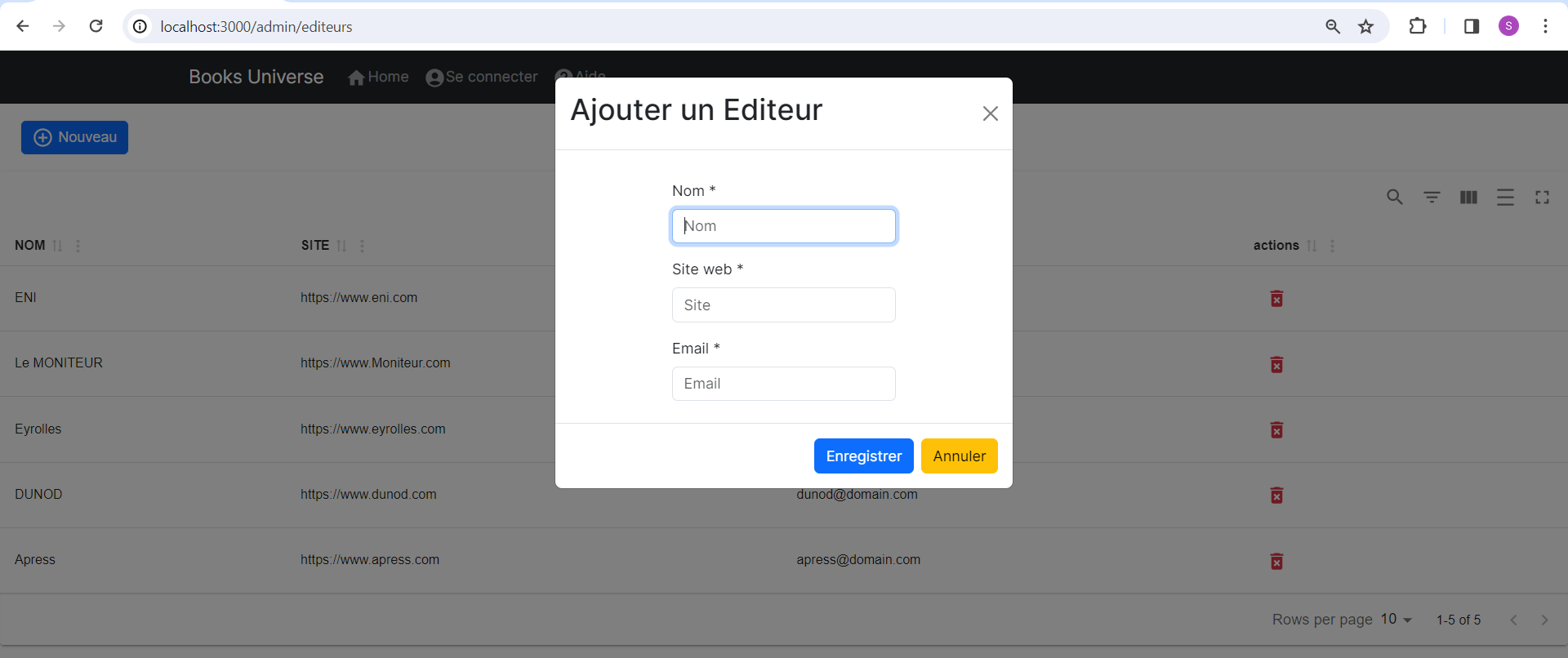
<MaterialReactTable columns={columns} data={LesEditeurs} />

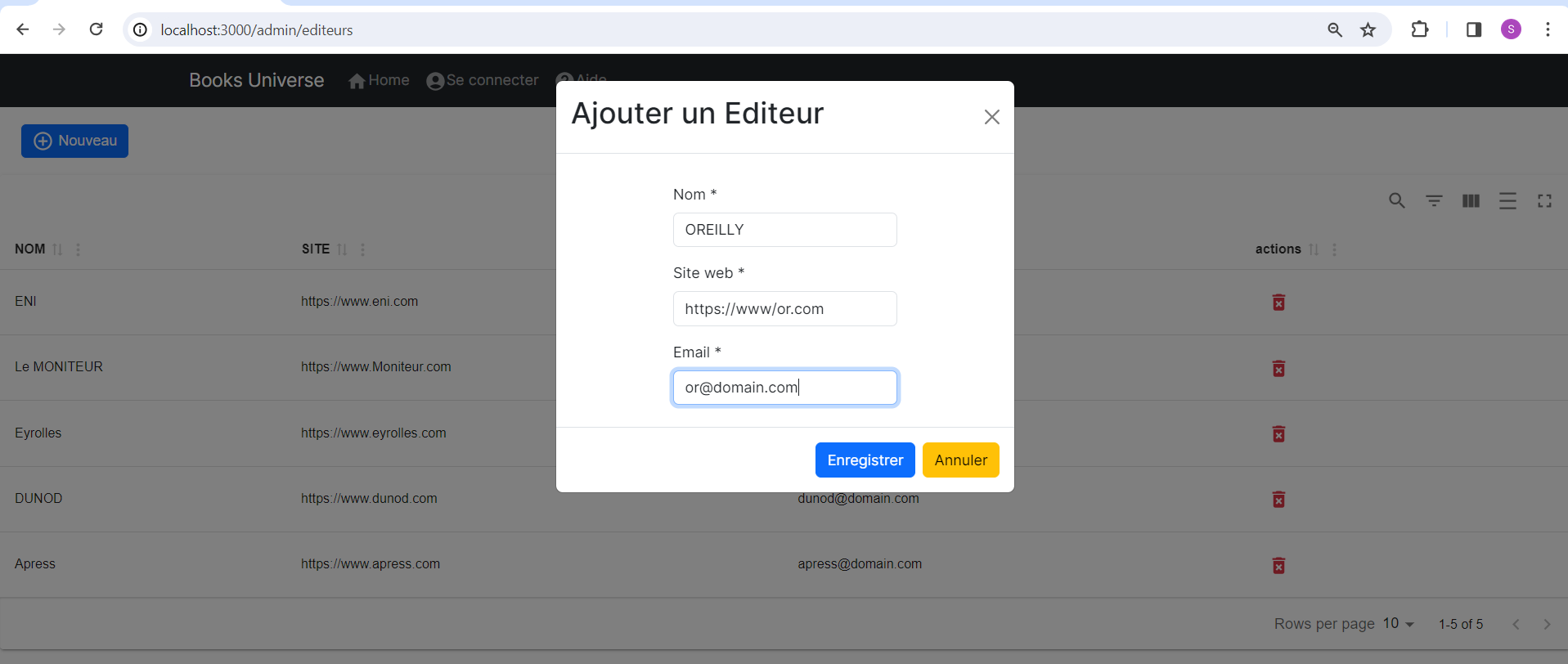
</div>

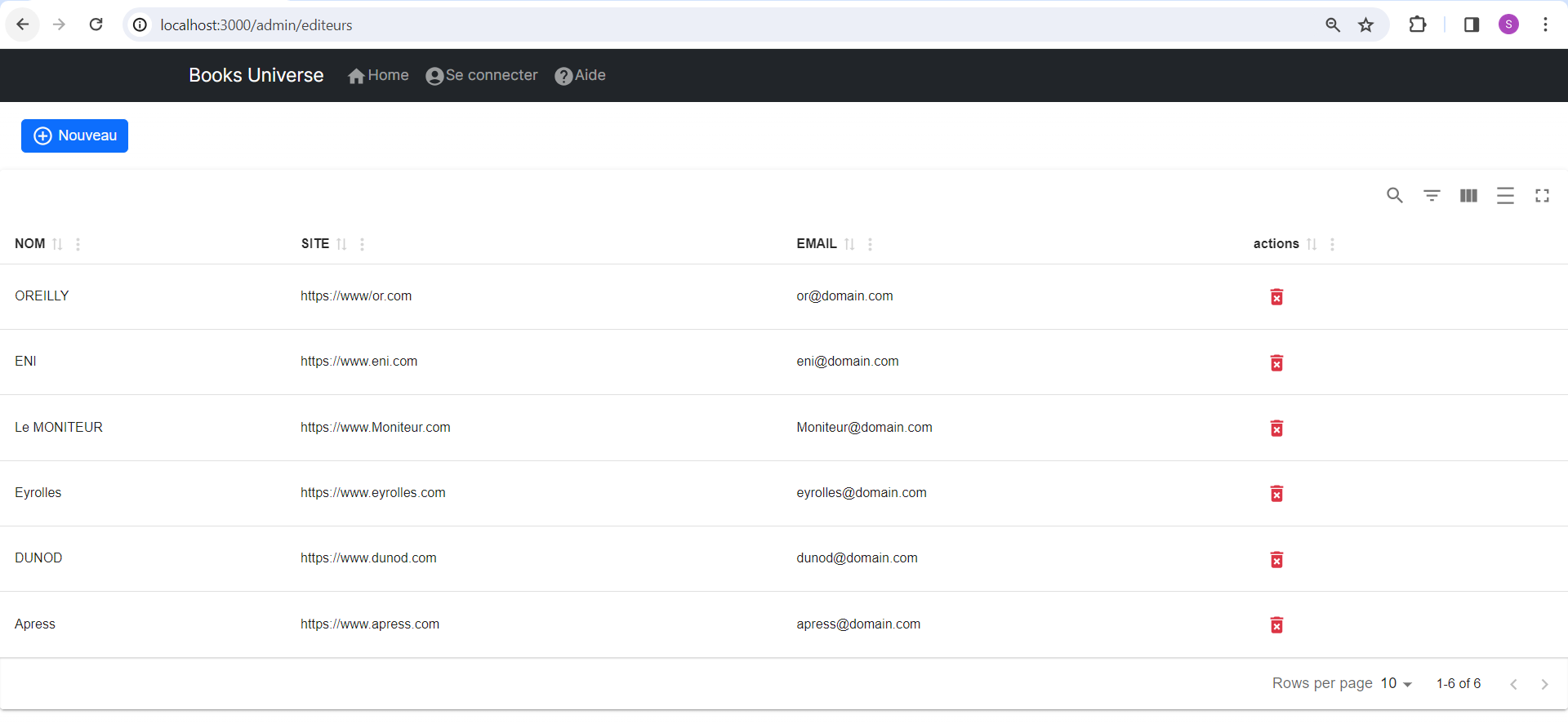
  )

}

export default Listediteurs

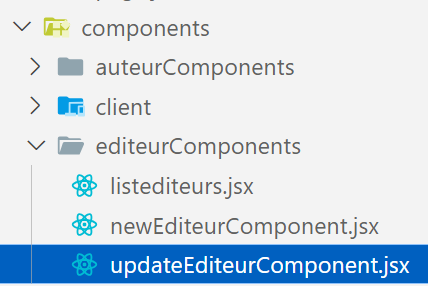






# Modal modification éditeur

Créer sous /src/components/editeurComponents le fichier updateEditeurComponent.jsx



import React, { useState } from 'react';

import Button from 'react-bootstrap/Button';

import Form from 'react-bootstrap/Form';

import Col from 'react-bootstrap/Col';

import Row from 'react-bootstrap/Row';

import Modal from 'react-bootstrap/Modal';

import {editEditeur} from "@/services/editeurService"

const Updateediteur =({ ed,LesEditeurs,setLesEditeurs,show,setShow})=> {

const [editeur,setEditeur]=useState(**ed**)

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

const handleClose = () => setShow(false);

const handlechange=(e)=>{

    setEditeur({...editeur,[e.target.name]:e.target.value})

  }

  const handleSubmit = (e) => {

    e.preventDefault();

    const form = e.currentTarget;

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

                //faire le update dans la BD

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

                   // faire le update dans le tableau affiché

                   setLesEditeurs(LesEditeurs.map((myediteur) =>myediteur.\_id === editeur.\_id ? editeur : myediteur));

                    //vider le form

                    handleClose()

                    setValidated(false);

                    })

                    .catch(error=>{

                    console.log(error)

                    alert("Erreur ! Modification non effectuée")

                    })

       }

    setValidated(true);

    }

    return (

 <div >

<Modal show={show} onHide={handleClose}>

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

<Modal.Header closeButton>

 <h2>Modifier un Editeurrrr</h2>

</Modal.Header>

<Modal.Body>

<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="12" >

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

<Form.Control

required

type="text"

placeholder="Nom"

name="maisonedit"

value={editeur.maisonedit}

onChange={(e)=>handlechange(e)}

/>

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

Saisir Nom de l'éditeur

</Form.Control.Feedback>

</Form.Group>

</Row>

</div>

<div className='form mt-3'>

<Row className="mb-2">

<Form.Group as={Col} md="12" >

<Form.Label >Site web \*</Form.Label>

<Form.Control

required

type="url"

placeholder="Site"

name="siteweb"

value={editeur.siteweb}

onChange={(e)=>handlechange(e)}

/>

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

Saisir Site de l'éditeur

</Form.Control.Feedback>

</Form.Group>

</Row>

</div>

<div className='form mt-3'>

<Row className="mb-2">

<Form.Group as={Col} md="12" >

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

<Form.Control

required

type="email"

placeholder="Email"

name="email"

value={editeur.email}

onChange={(e)=>handlechange(e)}

/>

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

Saisir Email de l'éditeur

</Form.Control.Feedback>

</Form.Group>

</Row>

</div>

</div>

</div>

</Modal.Body>

<Modal.Footer>

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

<Button type="button" className="btn btn-warning"onClick={()=>handleClose()}>Annuler</Button>

</Modal.Footer>

</Form>

</Modal>

</div>

  )

}

export default Updateediteur

Puis modifier le contenu de listeediteurs.jsx



"use client"

import  { useMemo,useState } from 'react';

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

import Button from 'react-bootstrap/Button';

import {deleteEditeur} from "@/services/editeurService"

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

import NewEditeurComponent from './newEditeurComponent';

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

import UpdateEditeurComponent from './updateEditeurComponent';

const Listediteurs = ({editeurs}) => {

const [LesEditeurs,setLesEditeurs]=useState(editeurs)

//pour edit modal

const [show, setShow] = useState(false);

const handleShow = () => setShow(true);

const [editeur,setEditeur]=useState([])

    const deleteediteur=(id)=>{

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

        deleteEditeur(id)

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

         setLesEditeurs(LesEditeurs.filter((item)=>item.\_id!==id))

      })

        .catch(error=>{

            console.log(error)

           })

  }

        }

  const columns = useMemo(

    () => [

    {

    accessorKey: 'maisonedit',

    header: 'NOM',

    size: 100,

    },

    {

        accessorKey: 'siteweb',

        header: 'SITE',

        size: 100,

        },

    {

    accessorKey: 'email',

    header: 'EMAIL',

    size: 100,

    },

   {

      accessorKey: '\_id',

      header: 'actions',

      size: 100,

      Cell: ({ cell, row }) => (

      <div >

     <Button

        onClick={()=>{setEditeur(cell.row.original);handleShow()}}

        variant="success"

        size="md"

        className="text-success btn-link edit"

        >

        <EditOutlinedIcon/>

     </Button>

     <Button

      onClick={(e) => {

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

      }}

      variant="danger"

      size="md"

      className="text-danger btn-link delete"

      >

      <DeleteForeverIcon />

      </Button>

      </div>

      ),

      },

    ],

    [LesEditeurs],

    );

return (

<div>

    <NewEditeurComponent LesEditeurs={LesEditeurs} setLesEditeurs={setLesEditeurs}/>

<MaterialReactTable columns={columns} data={LesEditeurs} />

{show ? <UpdateEditeurComponent ed={editeur} LesEditeurs={LesEditeurs} setLesEditeurs={setLesEditeurs} show={true} setShow={setShow} /> : null}

</div>

  )

}

export default Listediteurs

