Authentification avec refresh rotation et next-auth

Backend



ACCESS\_TOKEN\_SECRET=azerty

REFRESH\_TOKEN\_SECRET=qwerty



const express = require('express');

const router = express.Router();

const User = require('../models/user.js');

const  jwt  = require('jsonwebtoken');

const bcrypt = require('bcrypt');

const nodemailer=require('nodemailer');

var transporter =nodemailer.createTransport({

  service:'gmail',

  auth:{

    user:'esps421@gmail.com',

    pass:'lnrqjuzysshlrpem'

  },

  tls:{

    rejectUnauthorized:false

  }

})

require('dotenv').config()

//Register

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

    const{email,password,role,firstname,lastname,isActive,avatar}=req.body;

    const user = await User.findOne({ email })

    if (user) return res.status(404).send({ success: false, message: "User already exists" })

    const salt=await bcrypt.genSalt(10);

    const hash=await bcrypt.hash(password,salt);

    const newUser=new User({

      email:email,

      password:hash,

      role:role||"user",

      firstname:firstname ||"myfirstname",

      lastname:lastname||"mylastname",

      isActive:isActive ||true,

      avatar:avatar||"avatar.jpg"

        });

    try {

           await newUser.save();

           return res.status(201).send({ success: true, message: "Account created successfully", user: newUser })

       } catch (error) {

           res.status(409).json({ message: error.message });

       }

});

// afficher la liste des utilisateurs.

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

  try {

      const users = await User.find().select("-password");

      return res.status(200).send({ success: true, message: "Account created successfully", user: users })

  } catch (error) {

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

  }

});

// se connecter

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

    let expires = Date.now() + 1

  try {

      let { email, password } = req.body

      if (!email || !password) {

          return res.status(404).send({ success: false, message: "All fields are required" })

      }

       const user = await User.findOne({email});

         const isMatch=await bcrypt.compare(password,user.password);

           if(!isMatch) {res.status(400).json({msg:'mot de passe incorrect'});

           return} ;

      if (!user) {

          return res.status(404).send({ success: false, message: "Account doesn't exists" })

      } else {

    let isCorrectPassword = await bcrypt.compare(password, user.password)

     if (isCorrectPassword) {

              delete user.\_doc.password

              if (!user.isActive) return res.status(200).send({ success: false, message: 'Your account is inactive, Please contact your administrator' })

              const token = generateAccessToken(user);

             const refreshToken = generateRefreshToken(user);

              return res.status(200).send({ success: true, user, token,refreshToken,expiresIn: expires  })

          } else {

              return res.status(404).send({ success: false, message: "Please verify your credentials" })

          }

      }

  } catch (err) {

      return res.status(404).send({ success: false, message: err.message })

  }

 });

//Access Token

const generateAccessToken=(user) =>{

    return jwt.sign({user}, process.env.ACCESS\_TOKEN\_SECRET, { expiresIn: '60s' });

  }

  // Refresh

function generateRefreshToken(user) {

  //  return jwt.sign({user}, process.env.REFRESH\_TOKEN\_SECRET, { expiresIn: '1y' });

  return jwt.sign ({ iduser: user.\_id, role: user.role }, process.env.REFRESH\_TOKEN\_SECRET, { expiresIn: '1y'})

}

  //Refresh Route

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

    let expires = Date.now() + 3600

    const refreshtoken = req.body.refreshToken;

    console.log("RFR",refreshtoken)

      if (!refreshtoken) {

       return res.status(404).send({success: false, message: 'Token Not Found' });

          }

      else {

          jwt.verify(refreshtoken, process.env.REFRESH\_TOKEN\_SECRET, (err, user) => {

            if (err) {  console.log(err)

              return res.status(406).send({ success: false,message: 'Unauthorized' });

            }

            else {

             const token = generateAccessToken(user);

             const refreshToken = generateRefreshToken(user);

             console.log("token-------",token);

            res.status(200).send({success: true,

             token,

             refreshToken,

             expiresIn: expires

           })

              }

          });

         }

  });

  /\*\*

 \* as an admin i can disable or enable an account

 \*/

  router.put('/status/edit',  async (req, res) =>  {

    try {

        let { idUser } = req.body

        let user = await User.findById(idUser).select('+isActive')

        user.isActive = !user.isActive

        user.save()

        res.status(200).send({ success: true, user })

    } catch (err) {

        return res.status(404).send({ success: false, message: err })

    }

   })

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

    try {

    let email = req.query.email

    let user = await User.findOne({email})

    //.select('+isActive')

    user.isActive = !user.isActive

    user.save()

    //res.status(200).send({ success: true, user })

    res.redirect("https://www.google.fr/")

    } catch (err) {

    return res.status(404).send({ success: false, message: err })

    }

    })

    /\*

  Forgot password

    \*/

  router.post('/forgot-password', (req, res) => {

    const {email} = req.body;

    User.findOne({email: email})

    .then(user => {

        if(!user) {

            return res.send({Status: "User not existed"})

        }

        const token = jwt.sign({id: user.\_id}, "jwt\_secret\_key", {expiresIn: "1d"})

        var transporter = nodemailer.createTransport({

            service: 'gmail',

            auth:{

              user:'esps421@gmail.com',

              pass:'lnrqjuzysshlrpem'

            },

            tls:{

              rejectUnauthorized:false

            }

          });

          var mailOptions = {

            from: '"verify your email " <esps421@gmail.com>',

            to: email,

            subject: 'Reset Password Link',

            text: `http://localhost:3000/reset\_password/${user.\_id}/${token}`

          };

          transporter.sendMail(mailOptions, function(error, info){

            if (error) {

              console.log(error);

            } else { console.log(info)

              return res.send({Status: "Success"})

            }

          });

    })

  })

  /\*

  Reset Password

  \*/

  router.post('/reset\_password/:id/:token', async(req, res) => {

    const {id, token} = req.params

    const {password} = req.body

    jwt.verify(token, "jwt\_secret\_key", async (err, decoded) => {

        if(err) {

            return res.json({Status: "Error with token"})

        } else {

          const salt=await bcrypt.genSalt(10);

          await bcrypt.hash(password,salt)

            .then(hash => {

                User.findByIdAndUpdate({\_id: id}, {password: hash})

                .then(u => res.send({Status: "Success"}))

                .catch(err => res.send({Status: err}))

            })

            .catch(err => res.send({Status: err}))

        }

    })

  })

\*/

  module.exports = router;



const express = require('express');

const router = express.Router();

const Article=require("../models/article")

const {verifyToken} =require("../middleware/verif-token")

// afficher la liste des articles.

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

    try {

        const articles = await Article.find().populate("scategorieID").exec();

        res.status(200).json(articles);

    } catch (error) {

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

    }

});



const jwt = require('jsonwebtoken');

require('dotenv').config()

/\*\*

 \*

 \* Middleware function that check if user has token and can access or not

 \*

 \* @param {string} token - Authorization token provided by request sender

 \*

 \* @return

 \* @status 403 No token provided

 \* @Status 403 Invalid token

 \*

 \*/

const verifyToken = (req, res, next) => {

    const header = req.headers['authorization'];

    const token = header && header.split('Bearer ')[1];

    console.log("token-verif",token)

    if (!token) return res.status(405).send({ success: false, message: 'No token provided' });

    jwt.verify(token, process.env.ACCESS\_TOKEN\_SECRET, (err, decoded) => {

        if (err) return res.status(402).send({ success: false, message: 'Invalid token' });

        req.user = {}

        req.user.id = decoded.iduser

        req.user.role = decoded.role

        next()

    })

}

module.exports = { verifyToken }

Frontend



        env: {

            API\_URL: "http://localhost:3001",

            NEXTAUTH\_SECRET :"bonjour"

        }



'use client';

import NavScroll from '@/components/admin/NavScroll'

import {SessionProvider} from "next-auth/react";

export default function AdminLayout({ children,session }) {

  return (

    <>

    <SessionProvider session={session}>

       <NavScroll/>

        {children}

    </SessionProvider>

    </>

  )

}



"use client";

import React, { useState } from "react";

import {signIn} from 'next-auth/react';

import { useRouter } from 'next/navigation';

const Login = () => {

    const router = useRouter();

  const [email, setEmail] = useState("");

  const [password, setPassword] = useState("");

  const submitHandler = async (e) => {

    e.preventDefault();

    try{

      const res= await signIn('credentials',{

        redirect: false,

        email,

        password

      })

      console.log(res);

**router.push('/admin/products')**

    } catch (error) {

      console.log(error);

    }

  };

  return (

    <div className="container container-fluid">

      <div className="row mt-5 d-flex justify-content-center">

        <div className="col-10 col-lg-5 ">

          <form

            className="border border-secondary rounded p-4"

            onSubmit={submitHandler}

          >

            <h1 className="mb-4">Login</h1>

            <div className="form-outline mb-4">

              <label className="form-label" htmlFor="email\_field">

                Email address

              </label>

              <input

                type="email"

                id="email\_field"

                className="form-control"

                value={email}

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

              />

            </div>

            <div className="form-outline mb-4">

              <label className="form-label" htmlFor="password\_field">

                Password

              </label>

              <input

                type="password"

                id="password\_field"

                className="form-control"

                value={password}

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

              />

            </div>

            <button

              type="submit"

              className="btn btn-block w-100 btn-primary btn-block mb-4"

            >

              Validate

            </button>

          </form>

        </div>

      </div>

    </div>

  );

};

export default Login;



"use client"

import {useSession, signOut} from 'next-auth/react';

import Container from 'react-bootstrap/Container';

import Nav from 'react-bootstrap/Nav';

import Navbar from 'react-bootstrap/Navbar';

import NavDropdown from 'react-bootstrap/NavDropdown';

import { useRouter } from 'next/navigation';

import Link from 'next/link';

function NavScroll() {

  const router = useRouter();

  const {data} =useSession();

  return (

    <Navbar expand="lg" className="bg-body-tertiary">

      <Container>

        <Navbar.Brand href="#home">React-Bootstrap</Navbar.Brand>

        <Navbar.Toggle aria-controls="basic-navbar-nav" />

        <Navbar.Collapse id="basic-navbar-nav">

          <Nav className="me-auto">

          {data ?

<div onClick={() => {signOut()}}> Logout </div>

: <div onClick={() => router.push('/auth/login')}> LogIn </div>

}

          <Nav.Link as={Link} href="/">

            Home

            </Nav.Link>

          <Nav.Link as={Link} href="/admin/products">

            Articles

            </Nav.Link>

             <Nav.Link href="#link">Link</Nav.Link>

            <NavDropdown title="Dropdown" id="basic-nav-dropdown">

              <NavDropdown.Item href="#action/3.1">Action</NavDropdown.Item>

              <NavDropdown.Item href="#action/3.2">

                Another action

              </NavDropdown.Item>

              <NavDropdown.Item href="#action/3.3">Something</NavDropdown.Item>

              <NavDropdown.Divider />

              <NavDropdown.Item href="#action/3.4">

                Separated link

              </NavDropdown.Item>

            </NavDropdown>

          </Nav>

        </Navbar.Collapse>

      </Container>

    </Navbar>

  );

}

export default NavScroll;



import NextAuth from "next-auth";

import CredentialsProvider from "next-auth/providers/credentials";

async function refreshTokenRequest(token) {

  try {

    const response =  await fetch('http://localhost:3001/api/users/refreshToken', {

      method: 'POST',

      headers: { "Content-Type": "application/json" },

  body: JSON.stringify({refreshToken:token.refreshToken})

  });

   const data = await response.json()

    if (!response.ok) {

      throw data

    }

    token.token = data.token;

    token.refreshToken = data.refreshToken

    token.expiresIn=data.expiresIn

    return {

      ...token

         }

  } catch (error) {

    console.log(error)

    return {

      ...token,

      error: "RefreshAccessTokenError",

    }

  }

}

export const authOptions = {

secret: process.env.AUTH\_SECRET,

  providers: [

    CredentialsProvider({

       name: "Credentials",

       credentials: {

        email: { label: "email", type: "text"},

        password: { label: "Password", type: "password" },

      },

      async authorize(credentials, req) {

        const res = await fetch("http://localhost:3001/api/users/login", {

          method: "POST",

          headers: {

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

          },

          body: JSON.stringify({

            email: credentials?.email,

            password: credentials?.password,

          }),

        });

        const user = await res.json();

        if (user) {

           return user;

        } else {

           return null;

        }

      },

    }),

  ],

  callbacks: {

    async jwt({ token, user, account }) {

    if (account && user) {

      return { ...token, ...user };

    }

    if (Date.now() < token.expiresIn) {

      return { ...token,  ...user };

    }

    return refreshTokenRequest(token)

    },

    async session({ session, token, user }) {

      session.user = token ;

      session.error = token.error

      return session;

    },

  },

};

const handler = NextAuth(authOptions)

export { handler as GET, handler as POST }



import { getServerSession } from "next-auth";

import { authOptions } from "../app/api/auth/[...nextauth]/route";

const session = await getServerSession(authOptions);

import Api from "../Axios/Api";

const ARTICLE\_API="/articles"

export const fetchArticles=async()=> {

return await Api.get(ARTICLE\_API,

    {

        headers: {

          Authorization: `Bearer ${session?.user.token}`,

        }

      })

}

export const fetchArticleById=async(articleId)=> {

return await Api.get(ARTICLE\_API + '/' + articleId);

}

export const deleteArticle=async(articleId) =>{

return await Api.delete(ARTICLE\_API + '/' + articleId);

}

export const addArticle=async(article)=> {

return await Api.post(ARTICLE\_API, article);

}

export const editArticle=(article) =>{

return Api.put(ARTICLE\_API + '/' + article.\_id, article);

}



import React  from 'react';

import Listproducts from '@/components/admin/listeDesProduits';

import {fetchArticles} from "@/services/ArticlesService"

const getProducts=async()=>{

const data=await fetchArticles()

.then((results)=>{return results.data})

.catch((err)=>{console.log(err)})

return data

};

const ProductPage = async() =>{

    const produits=await getProducts()

  return (

    <div>

     {produits && <Listproducts produits={produits}/>}

    </div>

  )

}

export default ProductPage



import React from 'react'

const ListeDesProduits = ({produits}) => {

  return (

    <div>

        <table className="table table-striped">

    {

produits && produits.map(produit => (

    <tr>

    <td>{produit.designation}</td>

    <td>{produit.prix}</td>

    </tr>

    )

  )

    }

    </table>

    </div>

  )

}

export default ListeDesProduits