ATELIER 9 : AUTHENTICATION AVEC NEXT-AUTH ET PROTECTED ROUTES DANS NEXT JS

Introduction

NextAuth.js est une bibliothèque d'authentification open source complète (client/serveur) facile à implémenter, conçue à l'origine pour Next.js et Serverless.

Site officiel : https://next-auth.js.org/

NextAuth.js se targue d'être une solution open source très efficace pour l'enregistrement et l'authentification des utilisateurs dans les applications Next.js. Son approche minimaliste combinée à la prise en charge intégrée des services populaires vous permet d'ajouter une authentification à votre flux d'applications Web en quelques minutes.

Installer les dépendances

```
npm i axios mongoose bcryptjs next-auth --legacy-peer-deps
```

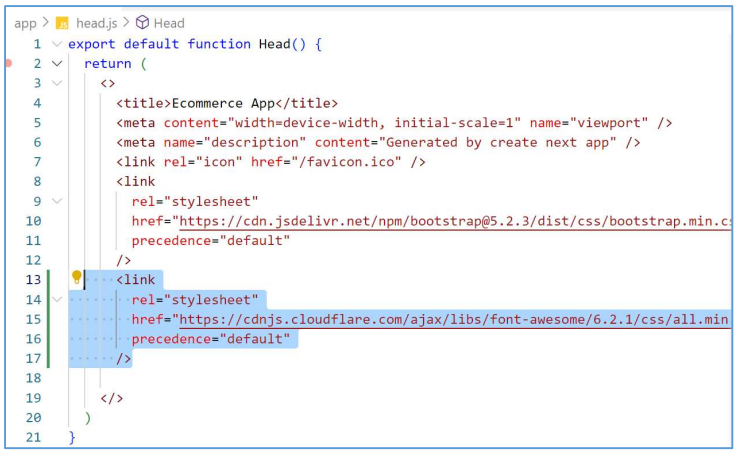
```
added 123 packages, and audited 387 packages in 37s

33 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

/app/head.js

head.js

Modifier le code en ajoutant ce link :

```
<link
        rel="stylesheet"
        href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.2.1/css/all.min.css"
        precedence="default"
      />
```



cdnjs.cloudflare.com est un réseau de diffusion de contenu ultra-rapide, fiable et disponible dans le monde entier pour les bibliothèques open source.

Cloudflare travaille avec les mainteneurs du projet cdnjs et distribue les dernières versions au fur et à mesure de leur sortie.

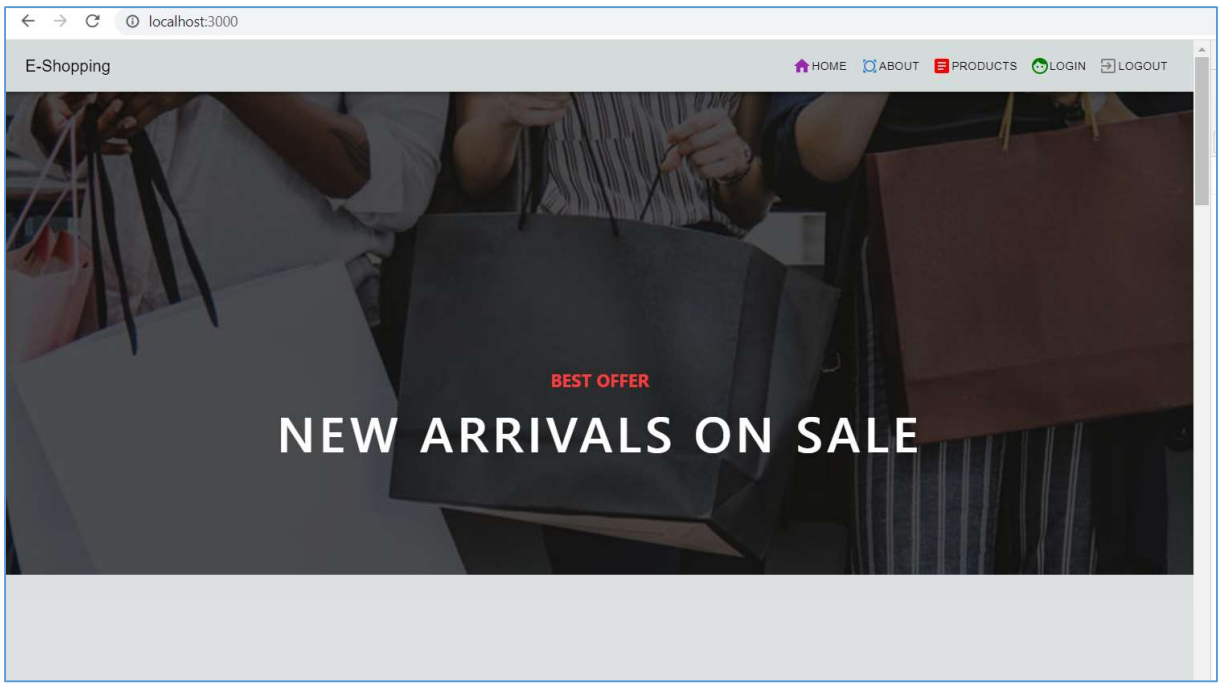
Cloudflare propose ainsi des services de CDN, sécurité et protection des données aux entreprises, développeurs, équipes et au grand public. Son réseau et ses diverses solutions permettent aux entreprises de sécuriser l'ensemble de leurs sites, applications web et outils connectés ainsi que d'améliorer leurs performances.

/components/Navbar.js


```
"use client";

import { useEffect,useState } from 'react';

import { useRouter } from 'next/navigation';

import AppBar from '@mui/material/AppBar';
import Box from '@mui/material/Box';
import Toolbar from '@mui/material/Toolbar';
import Typography from '@mui/material/Typography';
import Button from '@mui/material/Button';
import AllOutIcon from '@mui/icons-material/AllOut';
import HomeIcon from '@mui/icons-material/Home';
import ArticleIcon from '@mui/icons-material/Article';
import FaceIcon from '@mui/icons-material/Face';
import ExitToAppRoundedIcon from '@mui/icons-material/ExitToAppRounded';

function Navbar () {

    const router = useRouter();
```

```
  const [onTop, setOnTop] = useState(true);

  useEffect(() => {
    window.addEventListener('scroll', handleScroll);
  });

  const handleScroll = () => {
    if(window.pageYOffset === 0) {
      setOnTop(true);
    } else {
      setOnTop(false);
    }
  }

    return (
  <>

  <Box sx={{ flexGrow: 1 }}>
    <AppBar color={onTop ? 'transparent' : 'inherit'}>
          <Toolbar>

            <Typography variant="h6" component="div" sx={{ flexGrow: 1 }}
color="default">
              E-Shopping
            </Typography>
            <Button color="inherit" onClick={() => router.push('/')}><HomeIcon
color="secondary"/> Home </Button>

            <Button color="inherit" onClick={() =>
router.push('/about')}><AllOutIcon color="primary"/> About </Button>

            <Button color="inherit" onClick={() =>
router.push('/products')}><ArticleIcon style={{ color: 'red' }}/> Products
</Button>

            <Button color="inherit" onClick={() =>
router.push('/login')}><FaceIcon style={{ color: 'green' }}/> Login </Button>

            <Button color="inherit" onClick={() =>
router.push('/')}><ExitToAppRoundedIcon style={{ color: 'gray' }}/> Logout
</Button>

          </Toolbar>
        </AppBar>
      </Box>
     <Toolbar />
  </>

    );
```

```
}

export default Navbar ;
```



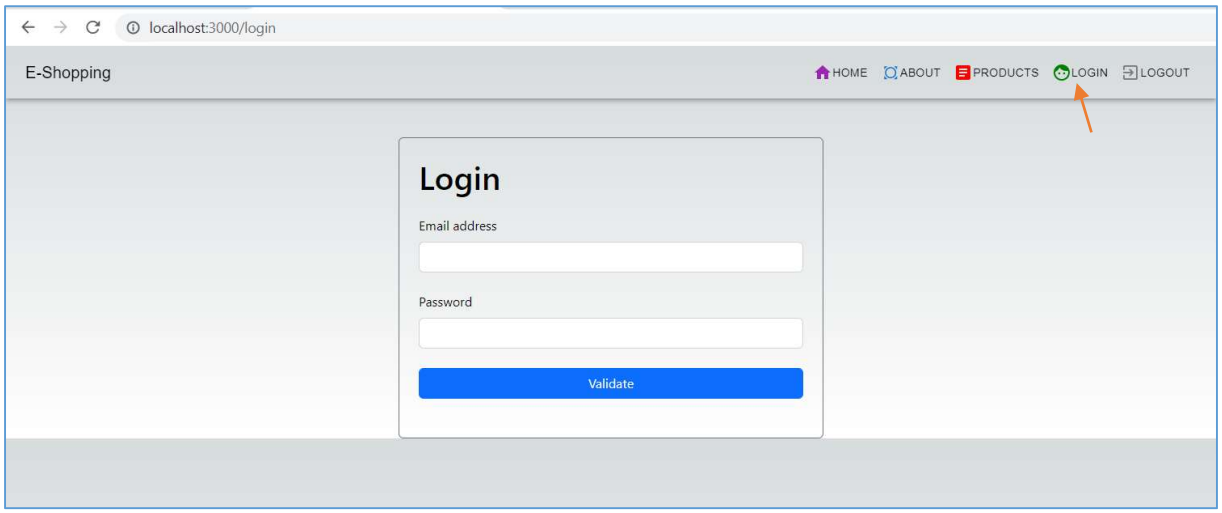
Page Login

Sous /app créer le répertoire login ayant le fichier page.js


```
"use client";

import React, { useState } from "react";

const Login = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const submitHandler = (e) => {
    e.preventDefault();
  };

  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"
            >
              Sign in
            </button>

            <div className="text-center">
              <p>
                Not a member? <Link href="/register">Register</Link>
              </p>
            </div>
          </form>
        </div>
      </div>
    </div>
  );
```

```
};

export default Login;
```



.env

Créer dans la racine du projet le fichier .env

Ajouter la ligne suivante :

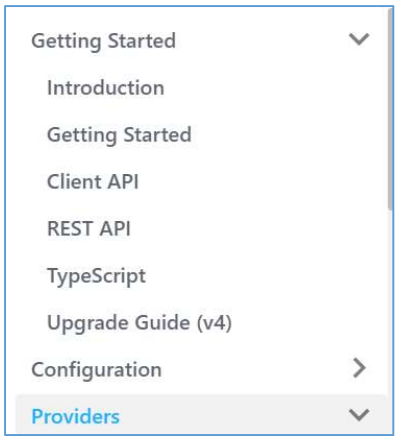
```
NEXTAUTH_SECRET = codingnexts
```

NextAuth.js : Authentication for Next.js

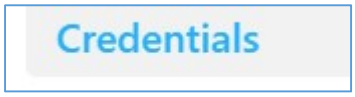
Visiter le site :

https://next-auth.js.org/getting-started/example

Choisir Providers

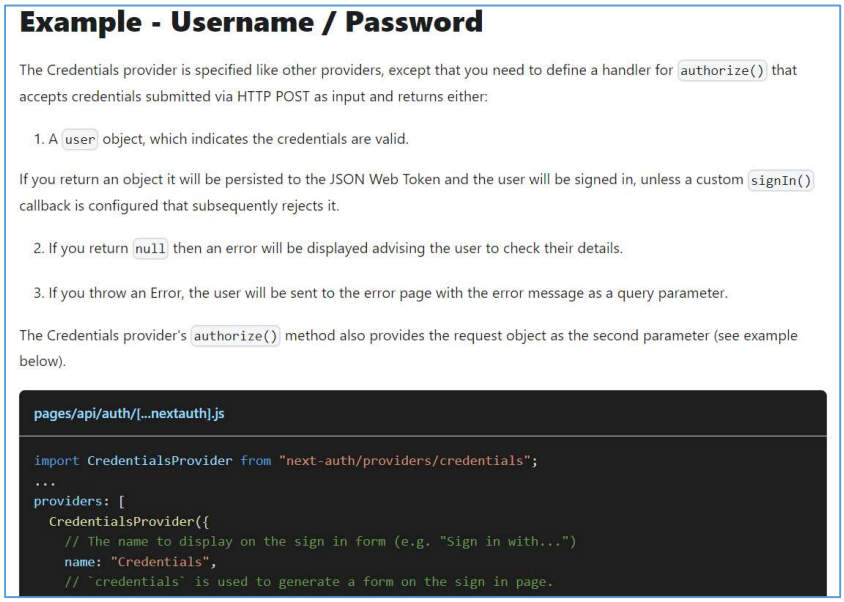


Puis cliquer sur :



On se retrouve dans la page :

https://next-auth.js.org/providers/credentials



Example - Username / Password

The Credentials provider is specified like other providers, except that you need to define a handler for `authorize()` that accepts credentials submitted via HTTP POST as input and returns either:

1. A `user` object, which indicates the credentials are valid.

If you return an object it will be persisted to the JSON Web Token and the user will be signed in, unless a custom `signIn()` callback is configured that subsequently rejects it.

2. If you return `null` then an error will be displayed advising the user to check their details.

3. If you throw an Error, the user will be sent to the error page with the error message as a query parameter.

The Credentials provider's `authorize()` method also provides the request object as the second parameter (see example below).

pages/api/auth/[...nextauth].js

```
import CredentialsProvider from "next-auth/providers/credentials";
...
providers: [
  CredentialsProvider({
    // The name to display on the sign in form (e.g. "Sign in with...")
    name: "Credentials",
    // `credentials` is used to generate a form on the sign in page.
```

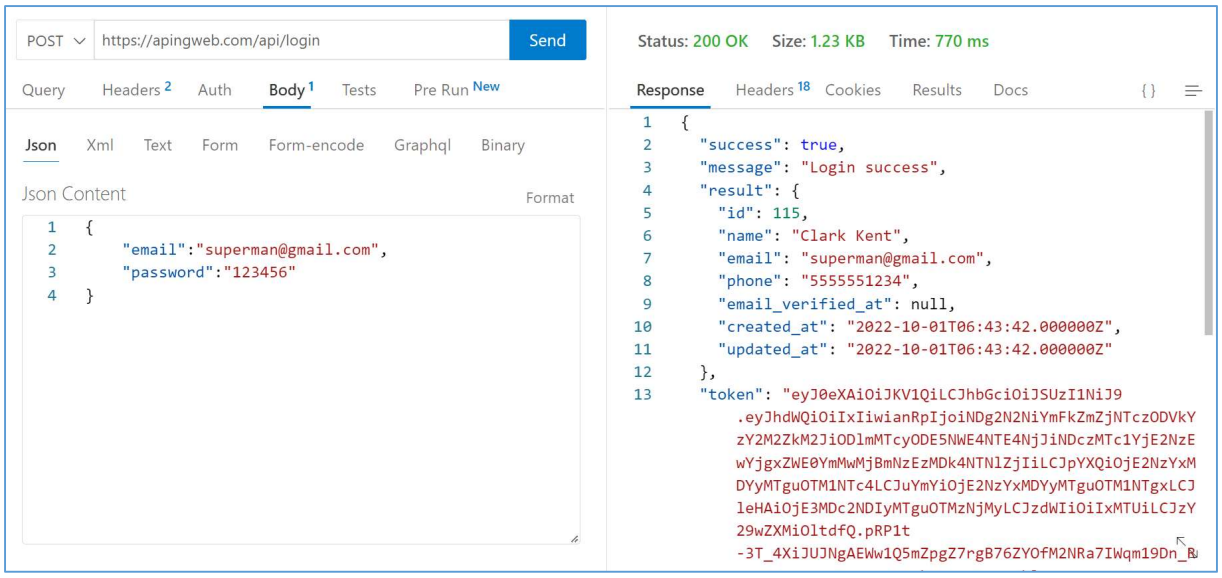
On va réaliser un code comme ce contenu

API online

On va utiliser cette API pour l'authentification.

https://apingweb.com/

Test API : https://apingweb.com/api/login



api/auth/[...nextauth].js

Créer un répertoire auth sous api

Puis créer le fichier [...nextauth].js

Y mettre ce contenu

```
import NextAuth from "next-auth";
```

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

import axios from "axios";

export default NextAuth({
    session: {
        strategy :'jwt'
    },
    providers: [
        CredentialsProvider({
            async authorize(credentials,req){

                const {email, password} = credentials;
                console.log(email, password)

                const res = await
axios.post('https://apingweb.com/api/login',{
                  email,
                  password
                    })
                  if(res){
                    const user = res.data.result;
                    const token = res.data.token;
                    console.log(user)
                    console.log(token)
                     return user;
                    }
                    else {
                        console.log("ERROR ");
                        return null;
                    }
                }
            })

    ],
    pages : {
        signIn: '/login'
    },
    secret : process.env.NEXTAUTH_SECRET
});
```

Le fournisseur Credentials est spécifié comme les autres fournisseurs, sauf que vous devez définir un gestionnaire pour authorize() qui accepte les informations d'identification soumises via HTTP POST en tant qu'entrée et renvoie soit :

1. Un objet user, qui indique que les informations d'identification sont valides.

Si vous renvoyez un objet, il sera conservé dans le jeton Web JSON et l'utilisateur sera connecté, à moins qu'un rappel personnalisé signIn() ne soit configuré pour le rejeter par la suite.

2. Si vous renvoyez null, une erreur s'affichera invitant l'utilisateur à vérifier ses coordonnées.

3. Si vous lancez une erreur, l'utilisateur sera envoyé à la page d'erreur avec le message d'erreur comme paramètre de requête.

La méthode authorize() du fournisseur Credentials fournit également l'objet de requête comme deuxième paramètre.

Login/page.js

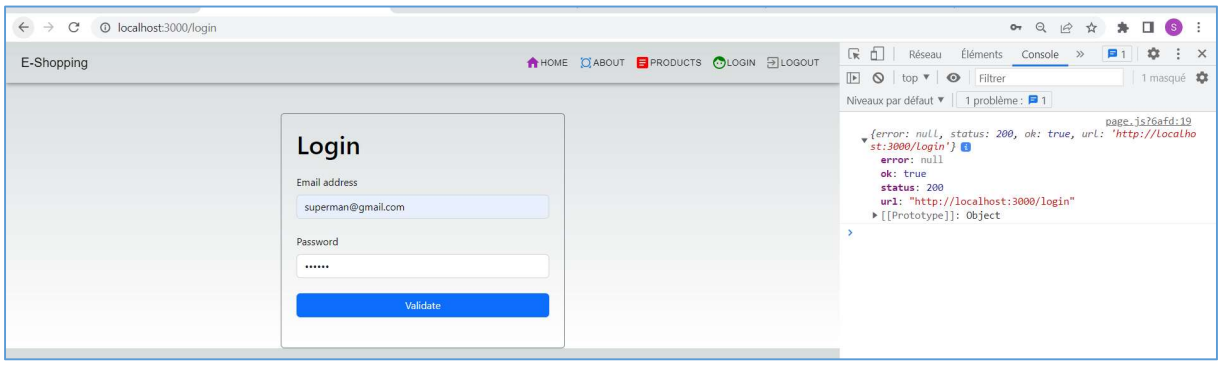
Modifier le code

```
"use client";

import React, { useState } from "react";

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

const Login = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const submitHandler = async (e) => {
    e.preventDefault();
    try{
      const data= await signIn('credentials',{
        redirect: false,
        email,
        password
      })
      console.log(data);
    } catch (error) {
      console.log(error);
    }
  };
```

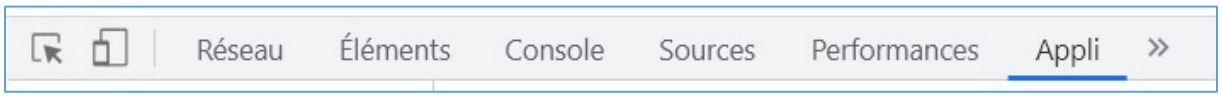
```
  return (
    <div className="container container-fluid">
      <div className="row mt-5 d-flex justify-content-center">
        <div className="col-10 col-lg-5 ">
          <form
            className="border border-secondary rounded p-4"
            onSubmit={submitHandler}
          >
            <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;
```

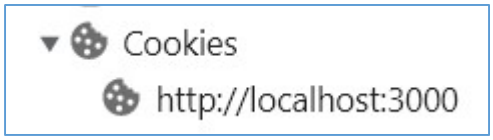
```
superman@gmail.com 123456
{
  id: 115,
  name: 'Clark Kent',
  email: 'superman@gmail.com',
  phone: '5555551234',
  email_verified_at: null,
  created_at: '2022-10-01T06:43:42.000000Z',
  updated_at: '2022-10-01T06:43:42.000000Z'
}
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhdWQiOiIxIiwianRpIjoiMGVhOWJjYjRkNzc2ZGY4MWQ1ZDdkNTM4ZTI2M2ViOWM3Y2Q1OWQwYzU5
NjM3NzFiYTJlMzhjMzY2OGIwYWI1OGZhYjZiYTkyNDYyYTZjM2UiLCJpYXQiOjE2NzYxMTA5NjcuMjM4Nzc4LCJuYmYiOjE2NzYxMTA5NjcuMjM4Nzg2L
CJleHAiOjE3MDc2NDY5NjcuMjM1MDEyLCJzdWIiOiIxMTUiLCJzY29wZXMiOltdfQ.nVucXEkUOThqQb68FqD9kln4lR1CkclmBdGsU0yjHQ310uvNeVC
Aij8-hf-DcBsS2RG0SWVPAMUGluSR5-k-N5htLdv2k5GajQ84V26-tIWXBV7V1WJ3ko7yFW_nVR23KXTYdpZA1imP9pVMwEGbT7rcGEBiDXP8fEfDfxx5
VHpWmpOGQV8IZ1KozNrwE_oRfK7uo-pK_WABk3I7jwlocjvQ3iHgDPgBRqHqgGePGfTGfs_0aOuCbMxsJdsz024Hjbj7Fmfh4P0_lMsJx0Cwb5Zt8aLFQ
nF1LjVzwV9WFfiSm7DpkLO_2WFy1MYVXI-CbSjDM7n9uz35eT9pckAzRK5V91NSN_72Ud5cnjiS8Sl4evC-BaylHee2YYYh8S5wmcpwuPzYT-SFRQ_uq_
-cUPZhf493J6W5eiyKBdyU4Z5BKsXOC3tZdu-6m0DUSKwbdwmBqXhHX5-zyAJgU6lZrfXC5R1t1iJOSvXMTJRDyv3cbMXP-sk8HFhNrd9Bz8DK19BAvpI
g7VRd_3Vuflco_znhIbzv435o4uD80QbtVP6zFdhVzNrp_RbwuiO0HJw39UKdLvuG4hHXbBFnx1USHKa7gqvIx_VaukG0ME5hAYUFPVNbaAUrLi1Tg77c
udlfAJTO98dF3VESV2s9xzgg4tD0TZ9OO1UbW1LTAh-v36s
```



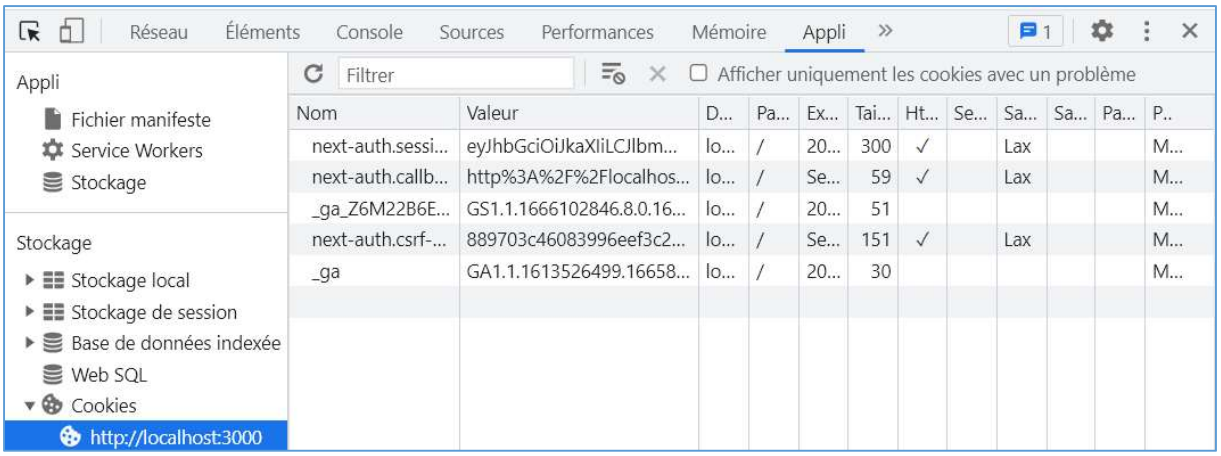
Dans l'inspecteur du navigateur choisir Application



Puis Storage/Coookies



On peut voir les cookies next dans notre navigateur



Session

Pour accéder à la session d'authentification dans notre application.

Dans app/layout.js

layout.js

Ajouter les lignes

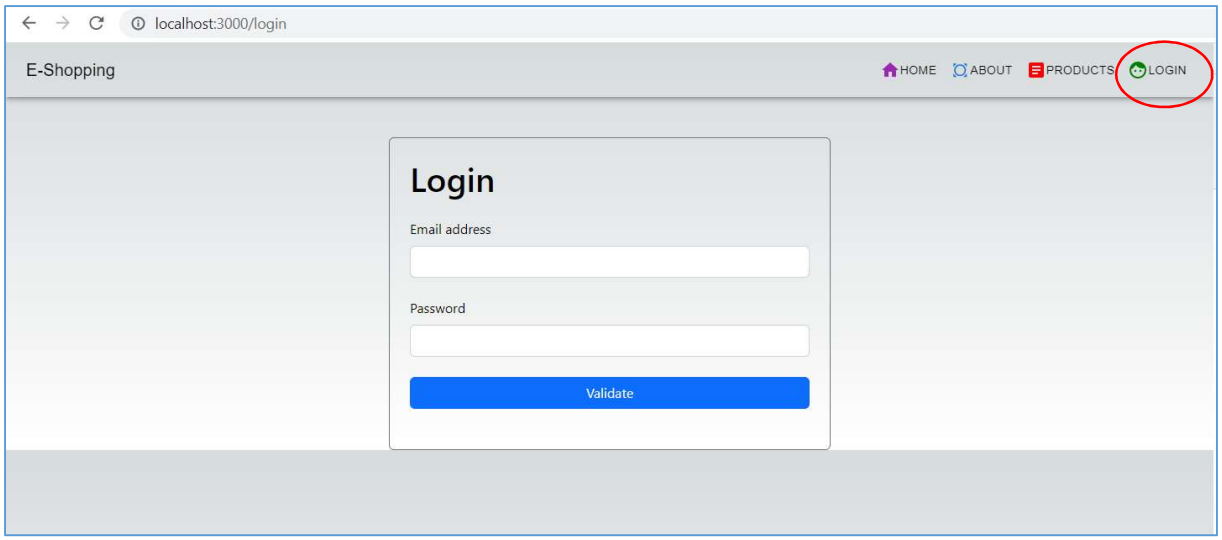
```
'use client';
import './globals.css'
import Navbar from "../components/Navbar"
import {SessionProvider} from "next-auth/react";

export default function RootLayout({ children }) {
  return (
    <html lang="en">
      {/*
        <head /> will contain the components returned by the nearest parent
        head.js. Find out more at https://beta.nextjs.org/docs/api-
reference/file-conventions/head
      */}
      <head />
      <body>
      <SessionProvider>
      <Navbar />
        {children}
      </SessionProvider>
      </body>
    </html>
  )
}
```

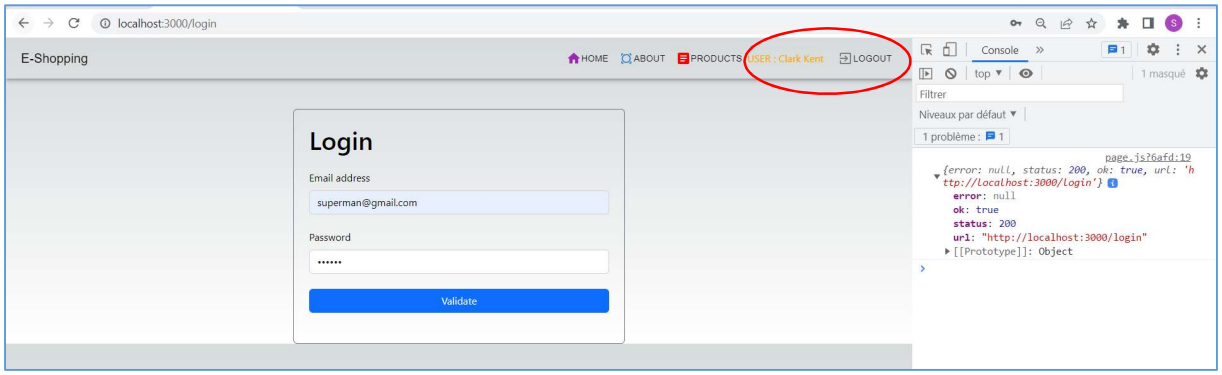
Puis aller à components/Navbar.js

Navbar.js

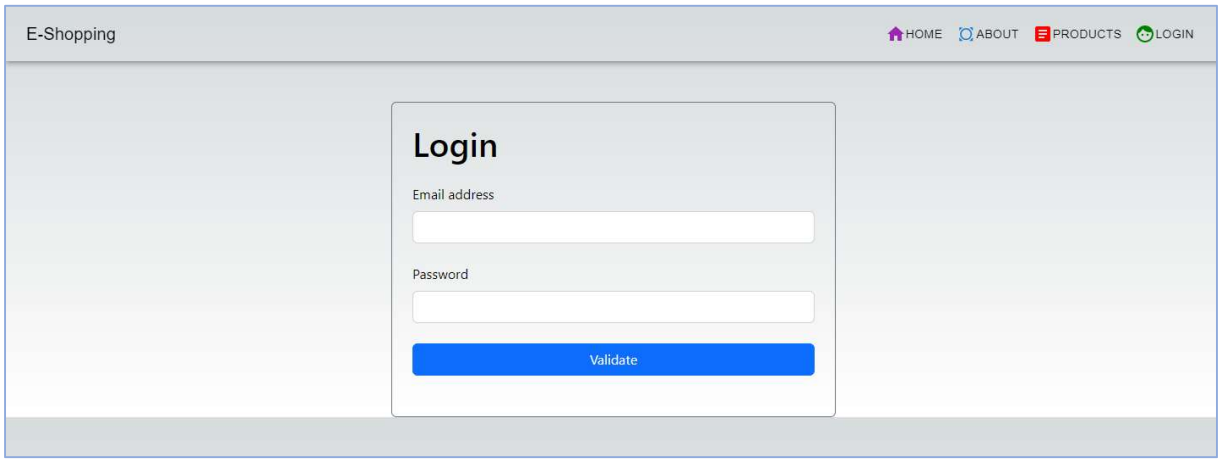
```
"use client";

import { useEffect,useState } from 'react';

import { useRouter } from 'next/navigation';

import AppBar from '@mui/material/AppBar';
import Box from '@mui/material/Box';
import Toolbar from '@mui/material/Toolbar';
import Typography from '@mui/material/Typography';
import Button from '@mui/material/Button';
import AllOutIcon from '@mui/icons-material/AllOut';
import HomeIcon from '@mui/icons-material/Home';
import ArticleIcon from '@mui/icons-material/Article';
import FaceIcon from '@mui/icons-material/Face';
import ExitToAppRoundedIcon from '@mui/icons-material/ExitToAppRounded';

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

function Navbar () {

  const {data} = useSession();


    const router = useRouter();

  const [onTop, setOnTop] = useState(true);

  useEffect(() => {
    window.addEventListener('scroll', handleScroll);
  });

  const handleScroll = () => {
    if(window.pageYOffset === 0) {
      setOnTop(true);
    } else {
      setOnTop(false);
    }
  }

    return (
```

```
  <>

  <Box sx={{ flexGrow: 1 }}>
    <AppBar color={onTop ? 'transparent' : 'inherit'}>
          <Toolbar>

            <Typography variant="h6" component="div" sx={{ flexGrow: 1 }}
color="default">
              E-Shopping
            </Typography>
            <Button color="inherit" onClick={() => router.push('/')}><HomeIcon
color="secondary"/> Home </Button>

            <Button color="inherit" onClick={() =>
router.push('/about')}><AllOutIcon color="primary"/> About </Button>

            <Button color="inherit" onClick={() =>
router.push('/products')}><ArticleIcon style={{ color: 'red' }}/> Products
</Button>
            {data?.user ? ( <>
              <span style={{ marginRight: "15px",color :"orange" }}>USER :
{data?.user?.name}</span>
              {" "}
             <Button color="inherit" onClick={() =>
signOut()}><ExitToAppRoundedIcon style={{ color: 'gray' }}/> Logout </Button>
            </>
            ): <Button color="inherit" onClick={() =>
router.push('/login')}><FaceIcon style={{ color: 'green' }}/> Login </Button>
            }
          </Toolbar>
        </AppBar>
      </Box>
     <Toolbar />
  </>

    );
}

export default Navbar ;
```





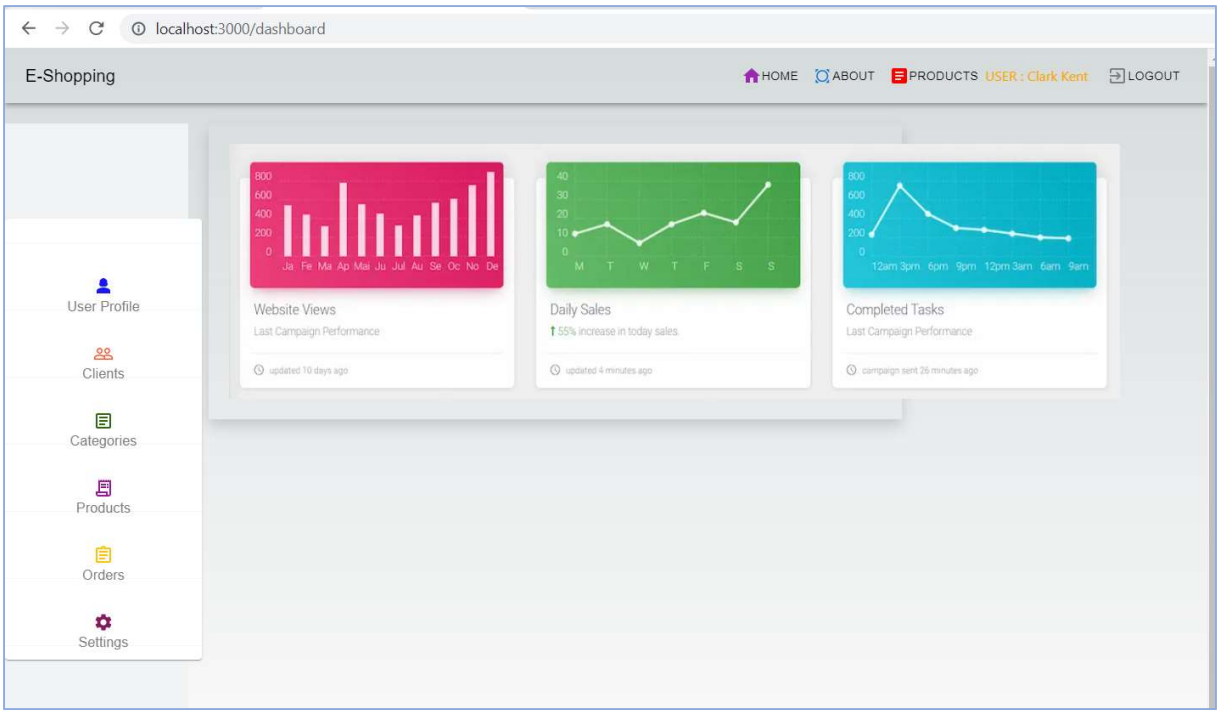
Si on clique sur Logout



Modifier le login pour se rediriger vers dashboard si l'authentification est correcte

```
"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 data= await signIn('credentials',{
        redirect: false,
        email,
        password
      })
      console.log(data);
      router.push('/dashboard')
    } 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;
```



Middleware

Dans la racine du projet créer le fichier middleware.js


```
export {default} from 'next-auth/middleware';

export const config = {matcher: ["/dashboard"]};
```

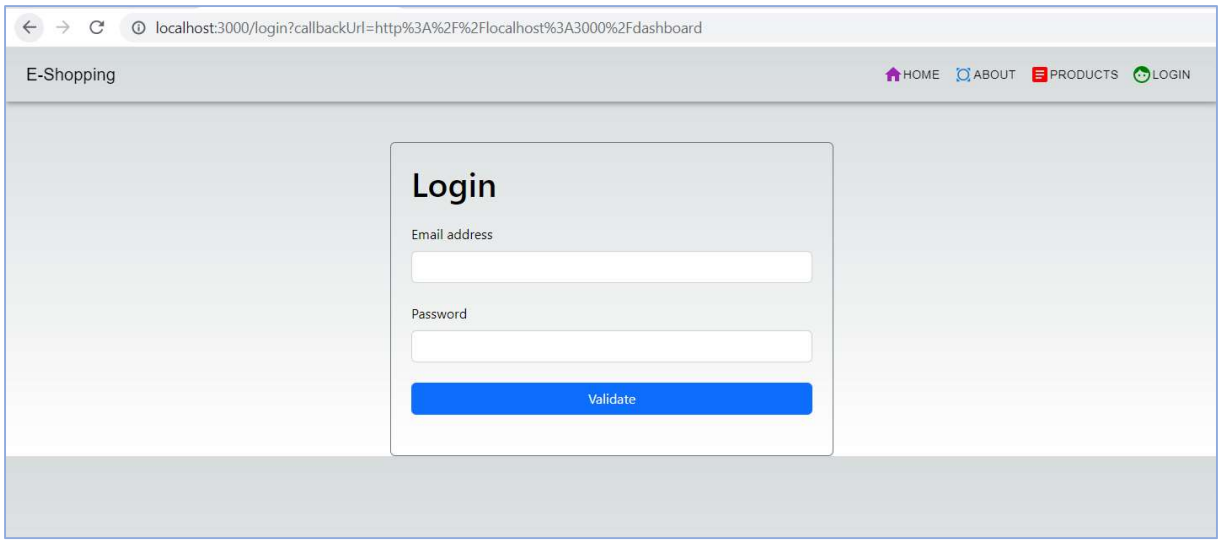
matcher vous permet de filtrer le middleware pour qu'il s'exécute sur des chemins spécifiques.

Dans le navigateur

http://localhost:3000/dashboard

devient :

http://localhost:3000/api/auth/signin?callbackUrl=%2Fdashboard



De même on va protéger toutes nos pages via le middleware :

middleware.js

```
export {default} from 'next-auth/middleware';

export const config = {matcher:
["/dashboard","/tableCategories","/tableProducts"]};
```