# AUTHENTIFICATION USER ET SHOPPING CART

Partie 1 : Authentification User

# Bootsrap

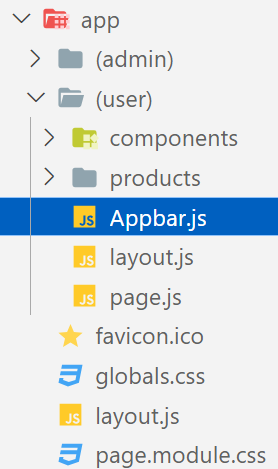
npm install bootstrap



@import url('https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css');

# Appbar

Nous disposons également de l’Appbar



import Link from "next/link";

const AppBar = () => {

return (

<nav className="navbar navbar-expand-lg bg-light">

    <ul className="navbar-nav me-auto mb-2 mb-lg-0">

    <li className="nav-item">

        <Link className="nav-link" href={"/"}>Home</Link>

    </li>

    <li className="nav-item">

        <Link className="nav-link" href={"/products"}>Proudcts</Link>

    </li>

</ul>

</nav>

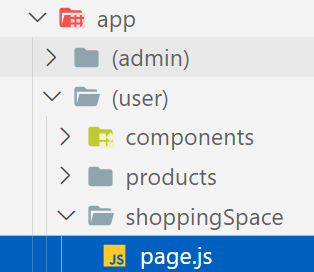
);

};

export default AppBar;

# Shopping Space

Créer :



import React from 'react'

import ProductCardShopping from '../components/user/productCardShopping'

import axios from 'axios'

const getProducts=async()=>{

const {data}=await axios.get(`${process.env.API\_URL}/api/articles`)

return data;

}

const ShoppingSpacePage = async() => {

const produits = await getProducts();

return (

<div className="row">

{produits.map((p) => (

<ProductCardShopping

key={p.\_id}

id={p.\_id}

image={p.imageart}

prix={p.prix}

des={p.designation}

/>

))}

</div>

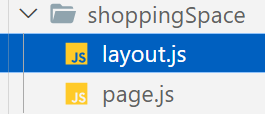
)

}

export default ShoppingSpacePage

# Layout Shopping

Créer



export default function ShoppingLayout({ children }) {

return (

<div>

<h3>Shopping Space</h3>

{children}

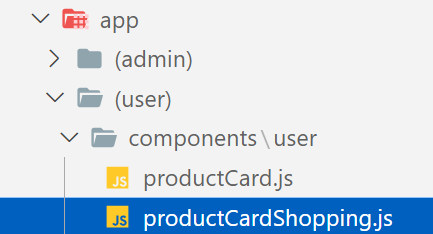
</div>

)

}

# Composant ProductCardShopping

Créer



import Image from "next/image";

import React from "react";

const ProductCardShopping = ({image,prix,des}) => {

return (

<div className="col-sm-3">

<div className="card">

<Image

src={image}

width={300}

height={200}

alt={des}

priority={true}

/>

<h6>{des}</h6>

<p>{prix}</p>

<button className="btn btn-warning">

        Add to cart

</button>

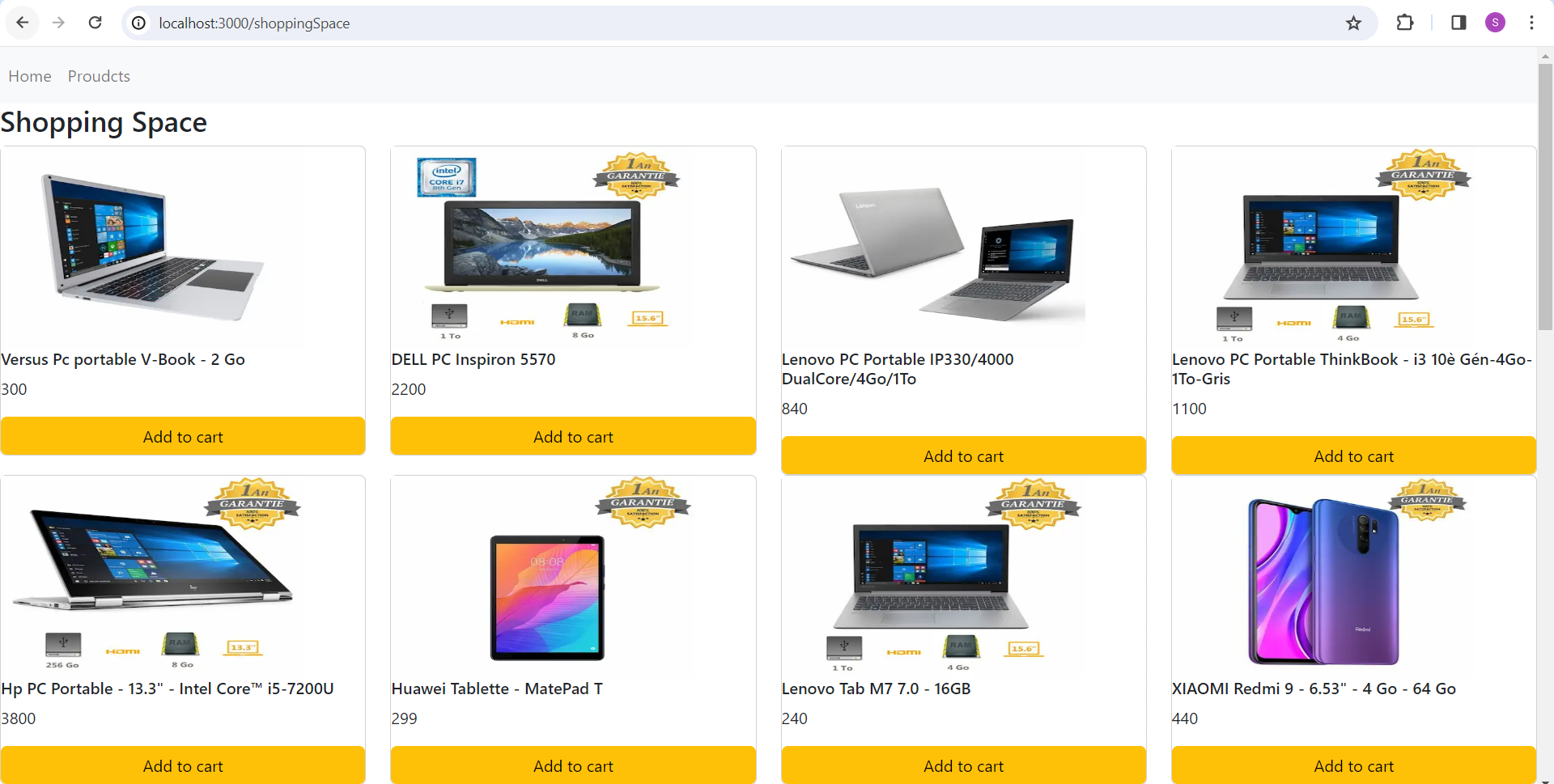
</div>

</div>

);

};

export default ProductCardShopping;

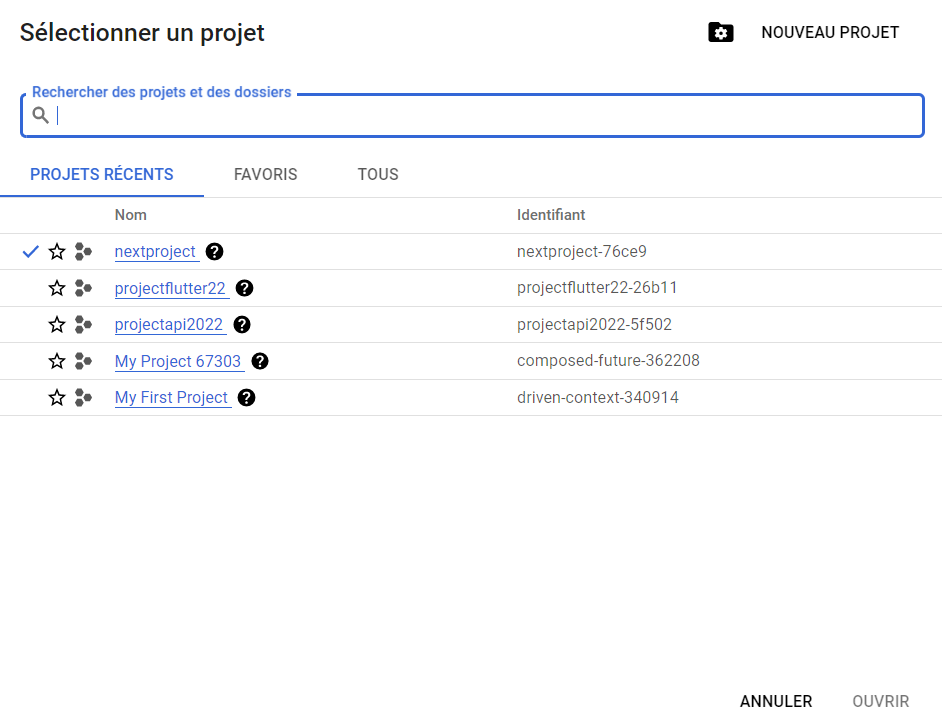


# Google Login

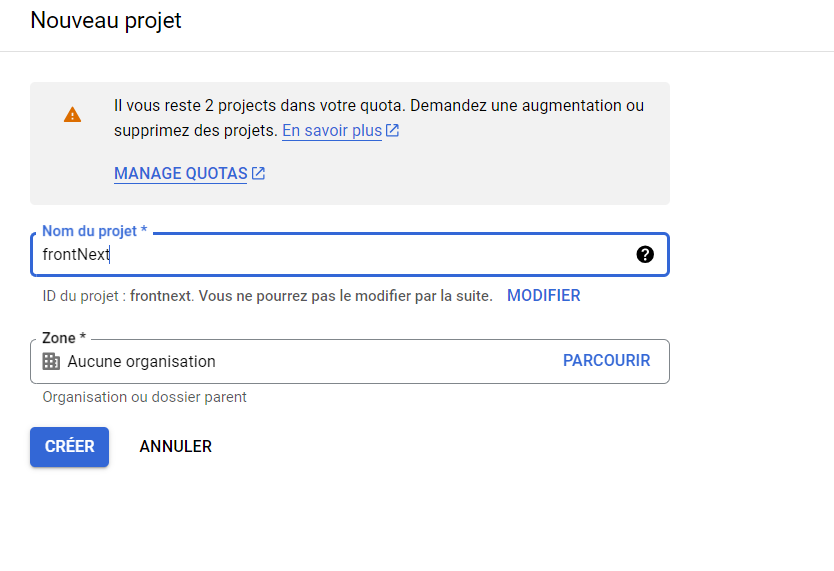
Aller au site :

<https://console.cloud.google.com/>

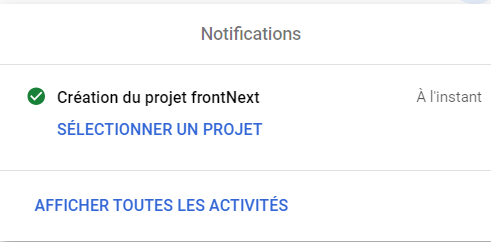




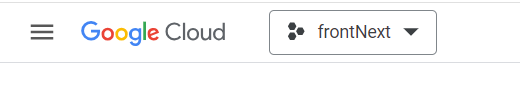




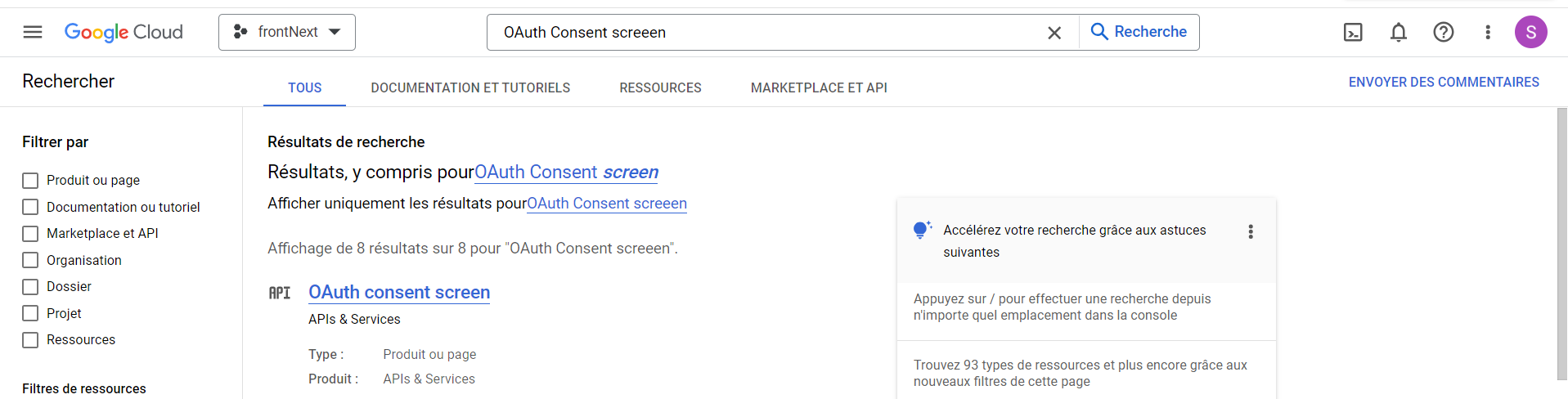


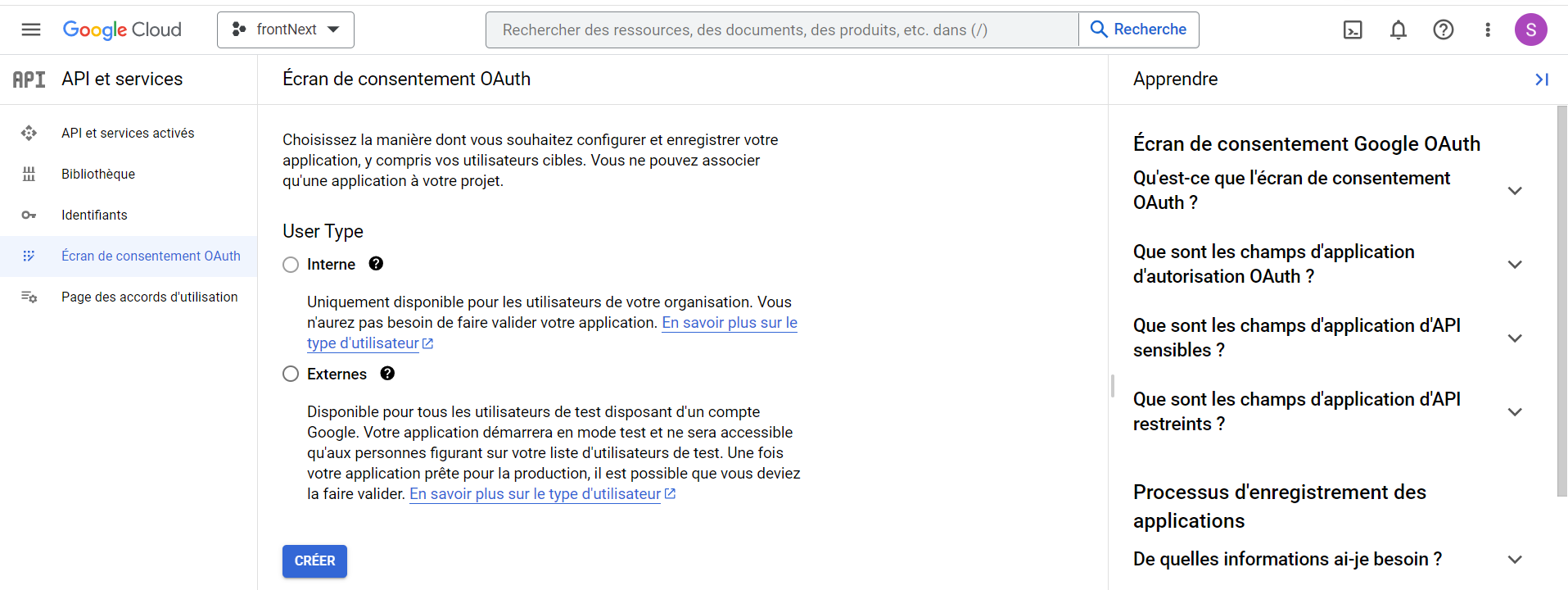


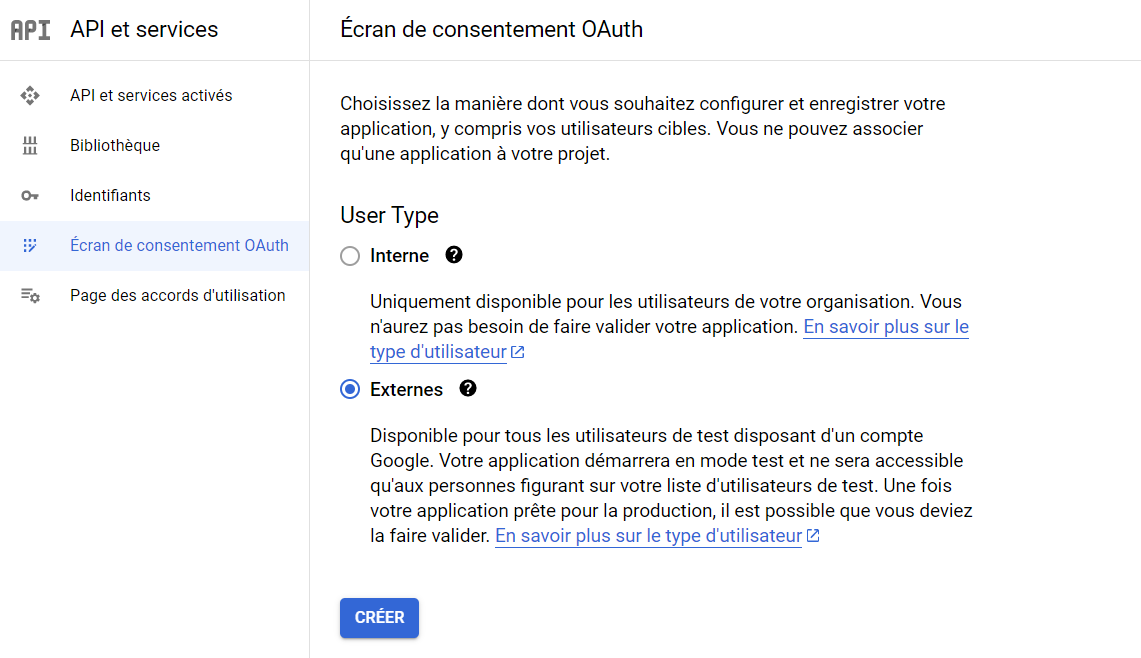


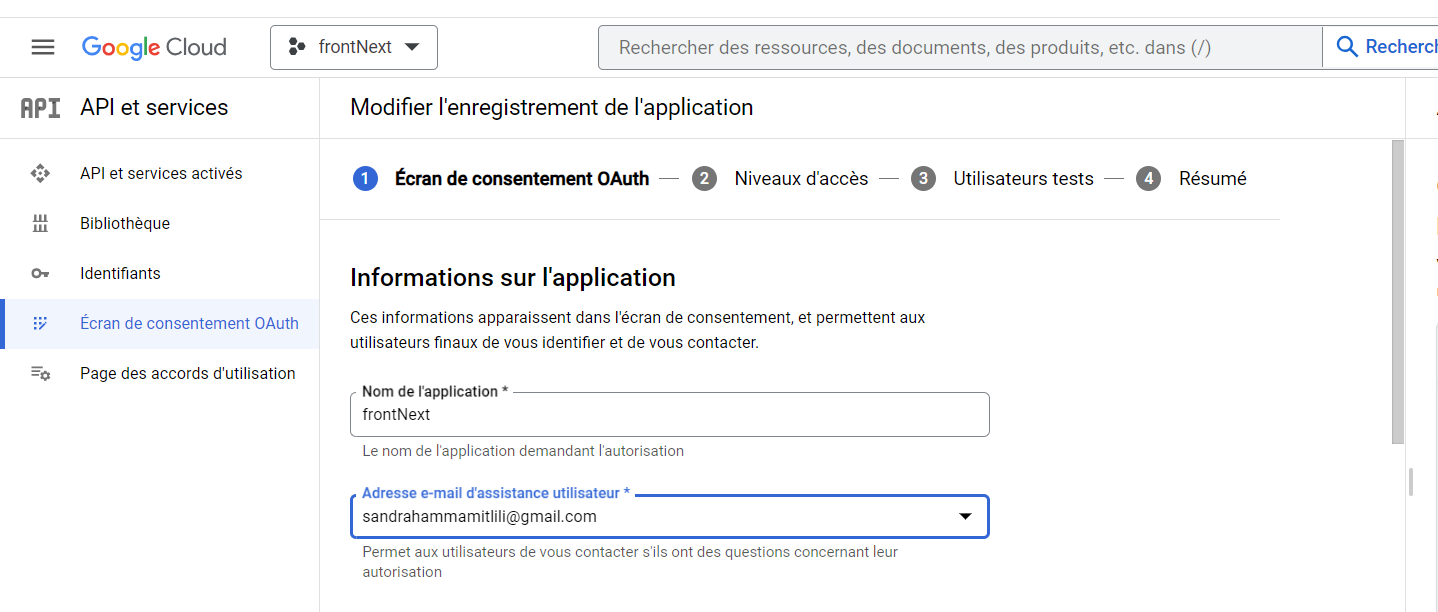






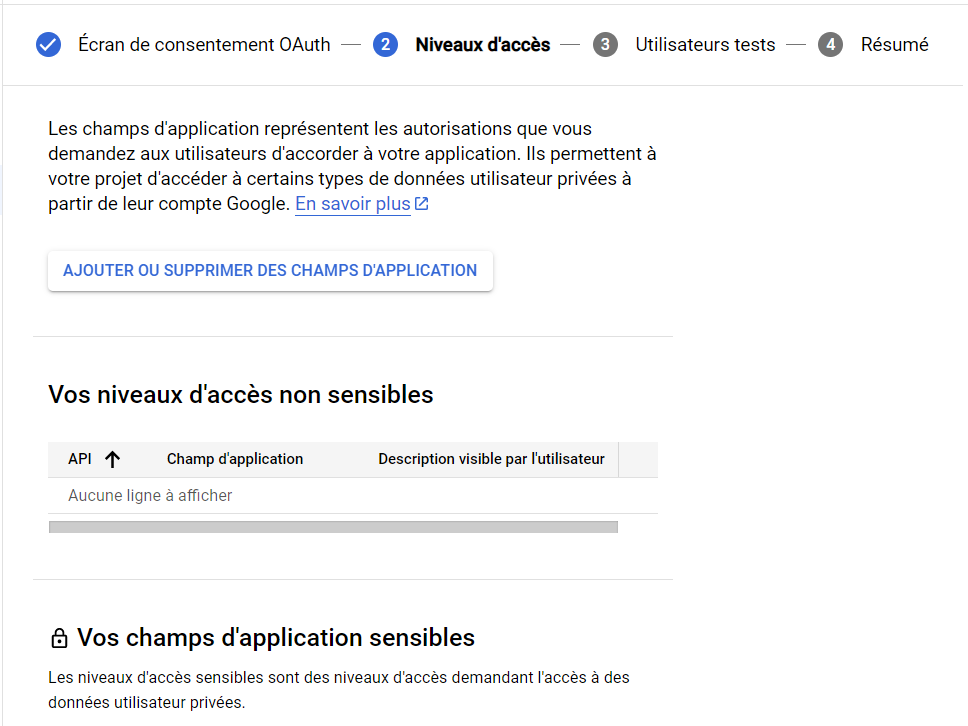


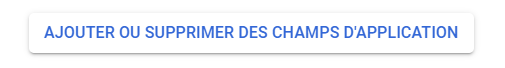


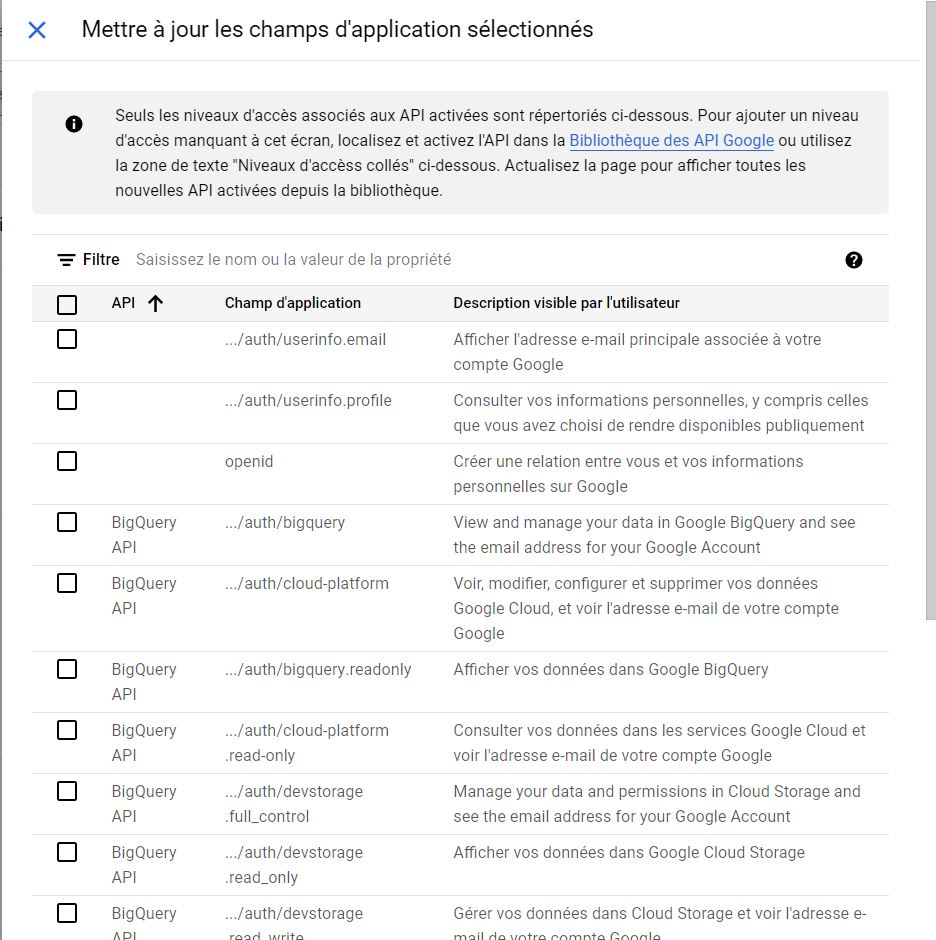


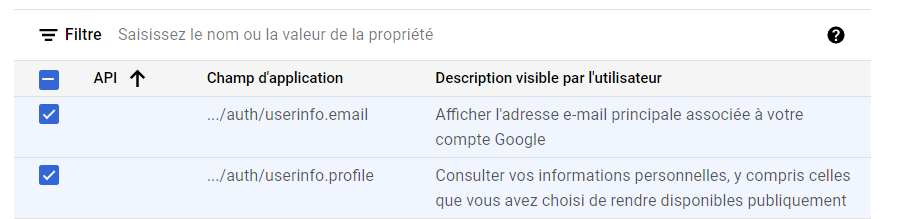


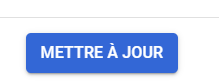


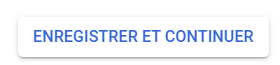


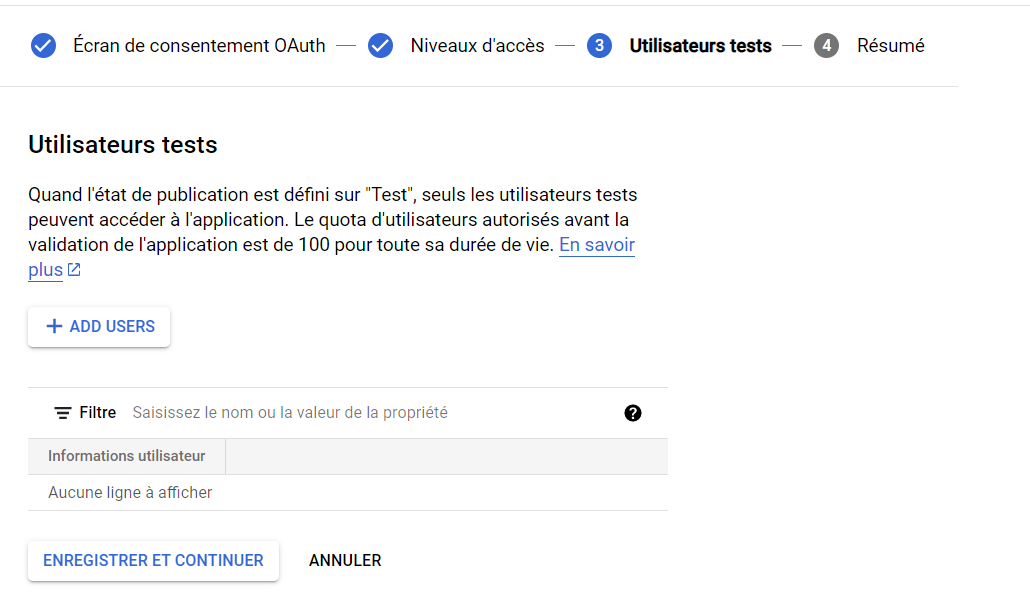


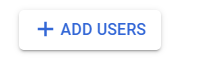


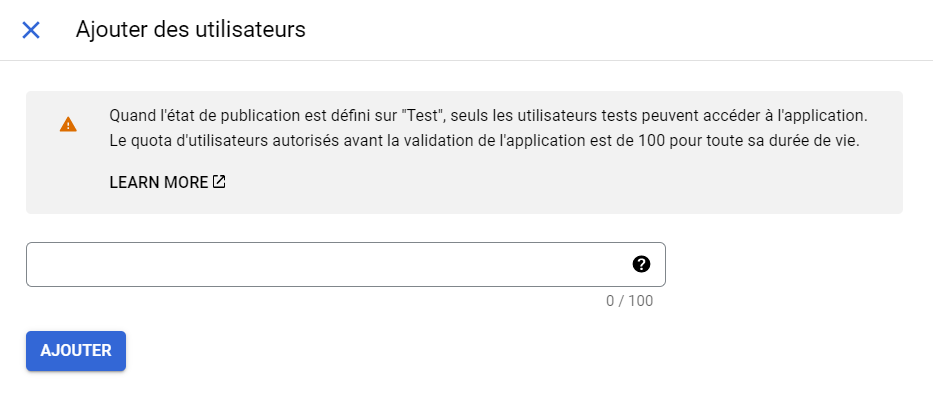








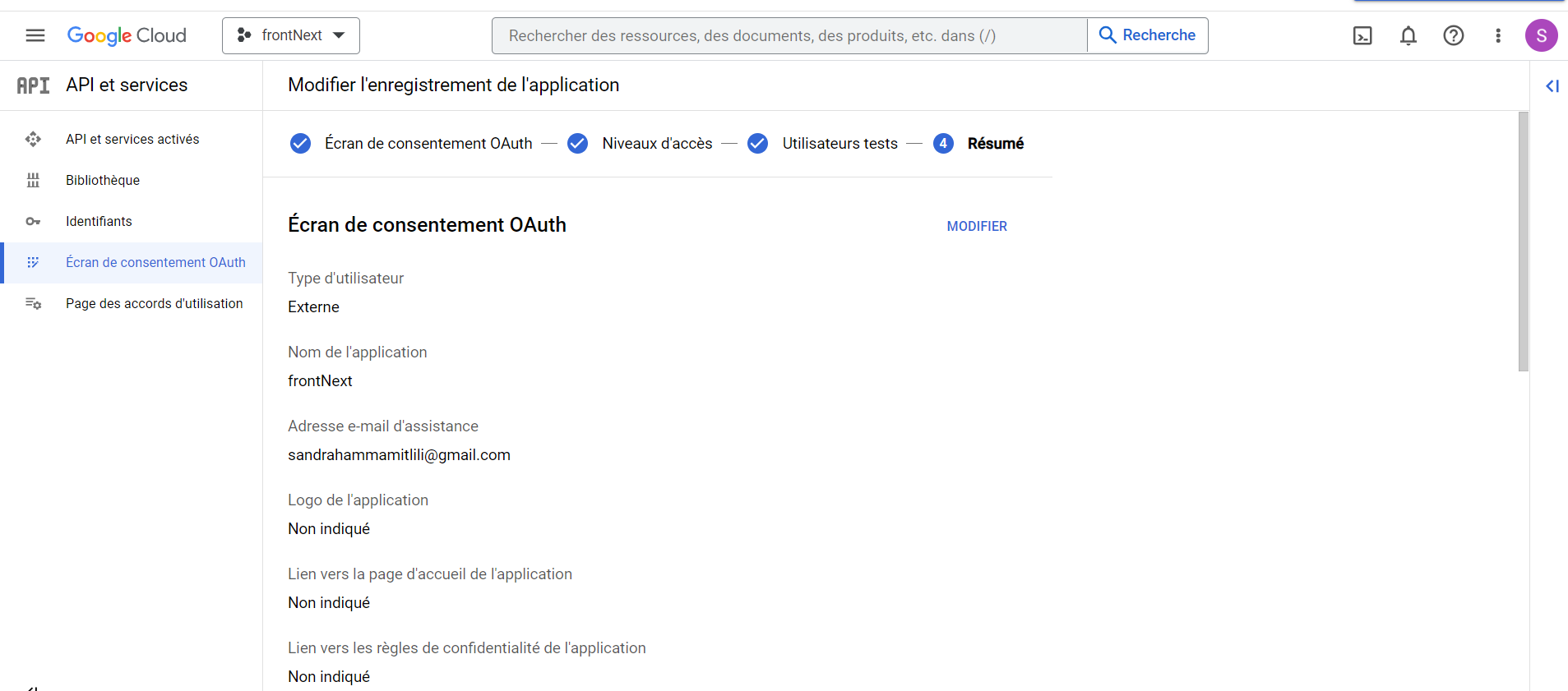


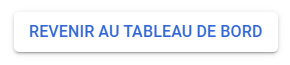


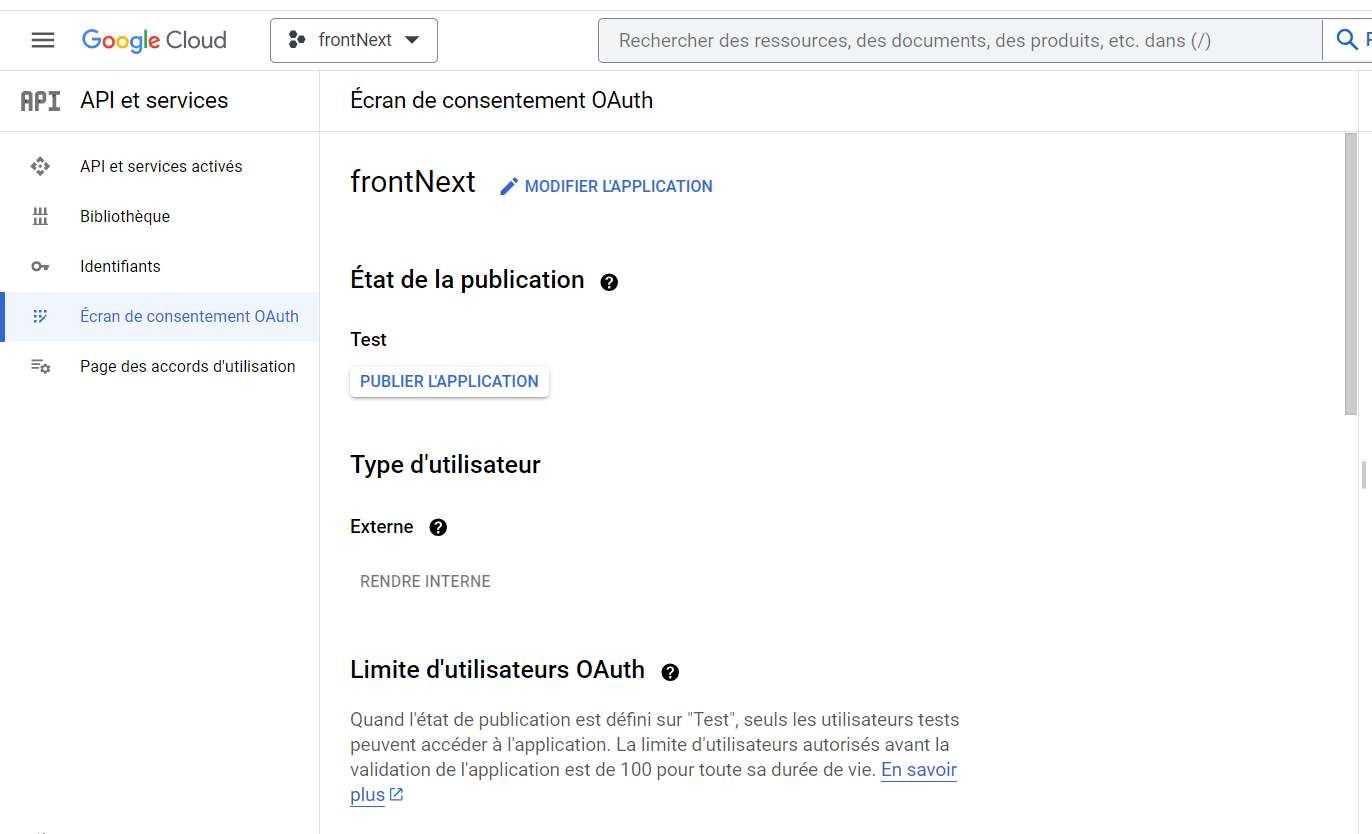




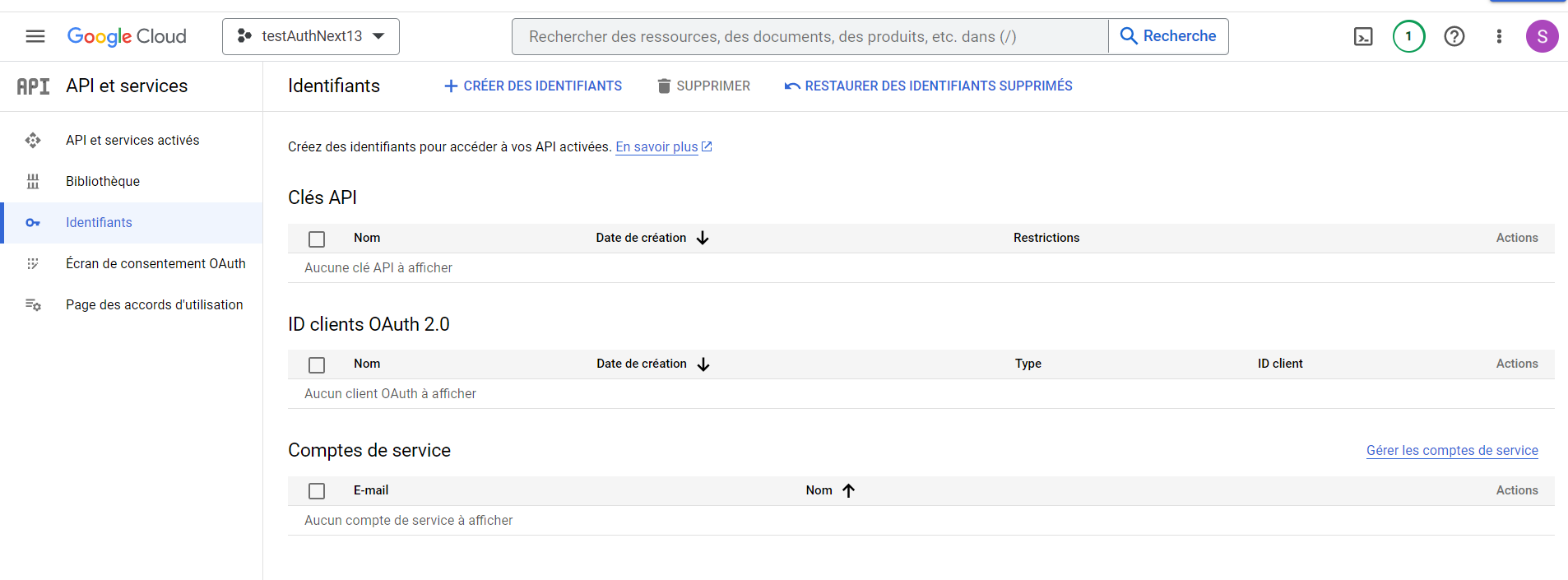




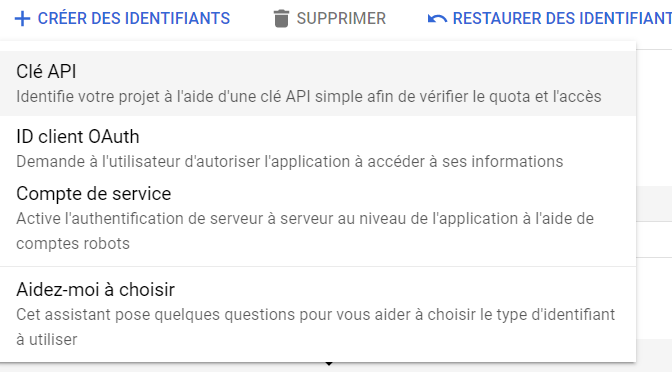




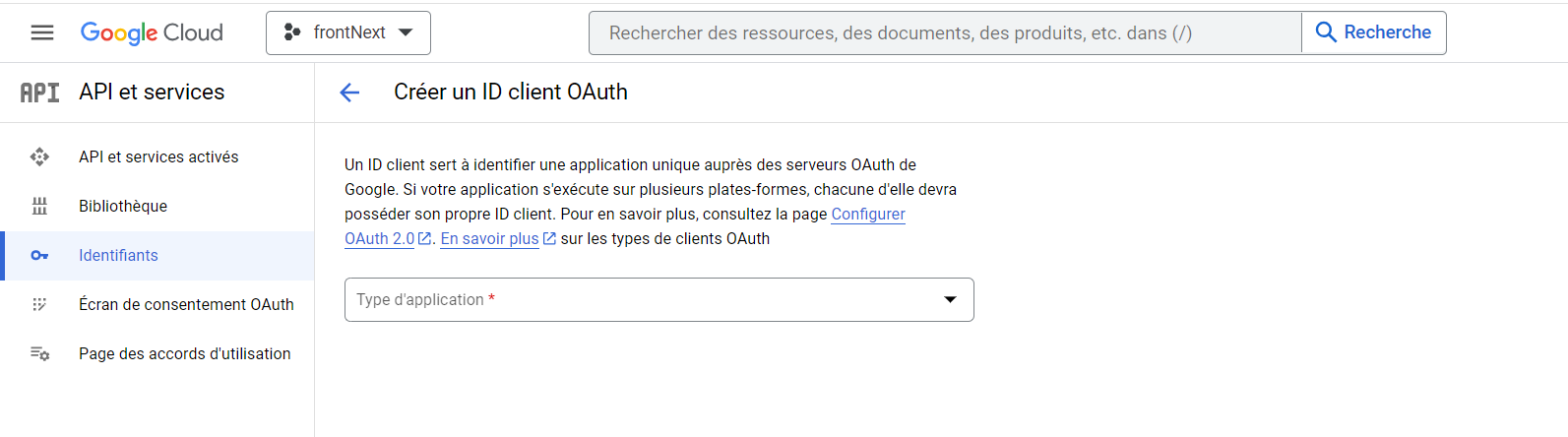




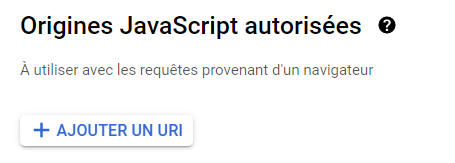


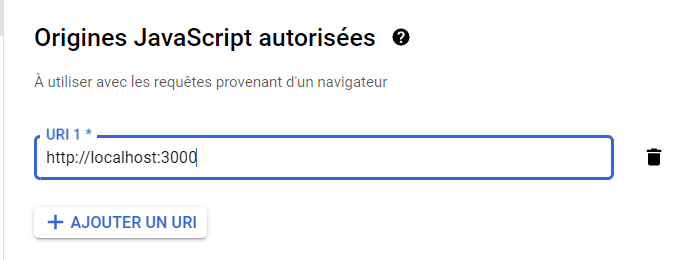






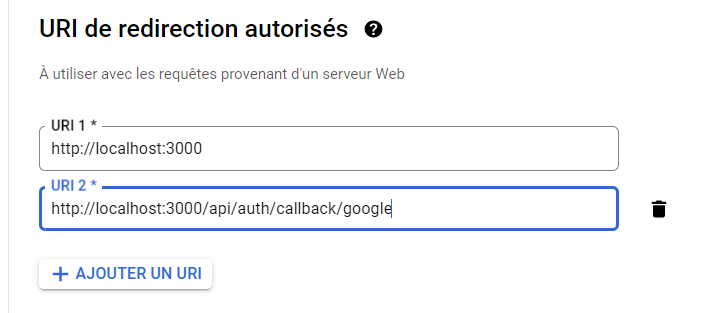




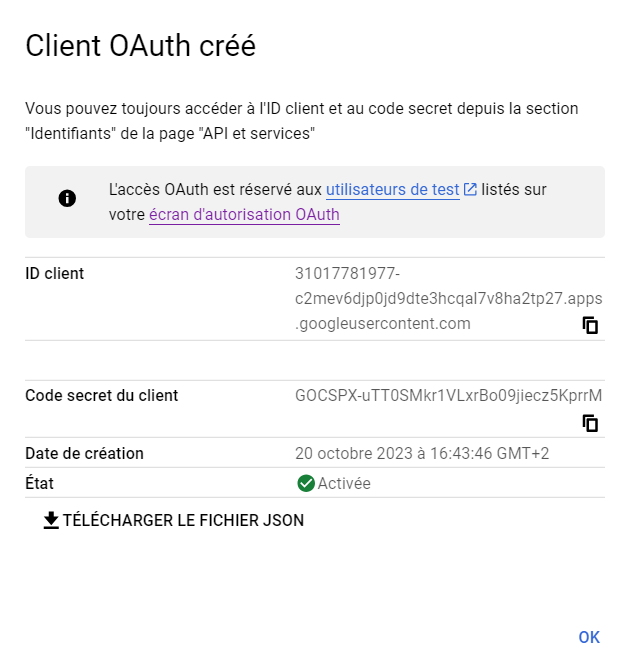












ID client : 31017781977-c2mev6djp0jd9dte3hcqal7v8ha2tp27.apps.googleusercontent.com

Code secret du client : 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

# Package next-auth

Dans le projet next installer

npm i next-auth

Créer

# signIn

Sous le répertoire /app/(user) créer le dossier signin dans le quel créer page.js



'use client';

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

import { useSearchParams } from 'next/navigation'

const SignInPage = () => {

    const searchParams = useSearchParams()

    const callbackUrl = searchParams.get('callbackUrl')

  return (

        <div className="container">

           <div className="card">

           <div className="card-head">

            <h3>Sign in to your account</h3>

           </div>

           <div className="card-body">

     <button className="btn btn-danger" onClick={() => signIn('google', { callbackUrl })}>    Continue with Google </button>

        </div>

        </div>

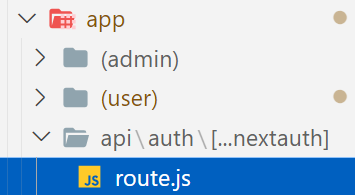
        </div>

  )

}

export default SignInPage

Puis créer /app/api/autn/[…nextauth]/route.js



import NextAuth from "next-auth";

import GoogleProvider from "next-auth/providers/google";

export const authOptions = {

    providers: [

        GoogleProvider({

            clientId: "31017781977-c2mev6djp0jd9dte3hcqal7v8ha2tp27.apps.googleusercontent.com",

            clientSecret: "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"

        })

    ],

    pages:{

        signIn: '/signin'

    }

}

const handler = NextAuth(authOptions)

export { handler as GET, handler as POST }



'use client';

import AppBar from "./Appbar"

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

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

return (

 <>

<SessionProvider session={session}>

<AppBar/>

{children}

</SessionProvider>

</>

)

}



'use client';

import Link from "next/link";

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

const AppBar = () => {

    const {data} =useSession();

return (

<nav className="navbar navbar-expand-lg bg-light">

    <ul className="navbar-nav me-auto mb-2 mb-lg-0">

    <li className="nav-item">

        <Link className="nav-link" href={"/"}>Home</Link>

    </li>

    <li className="nav-item">

        <Link className="nav-link" href={"/products"}>Proudcts</Link>

    </li>

 <li className="nav-item">

    <Link className="nav-link" href={"/shoppingSpace"} > Espace shopping  </Link>

   </li>

   <li className="nav-item">

   {data ?  ( <>

    <button className="btn btn-default"  onClick={() => {signOut();}}>Logout </button>

            </>

            ):  null

            }

    </li>

</ul>

</nav>

);

};

export default AppBar;

Modifier le code



'use client'

import Image from "next/image";

import React from "react";

import { useSession } from 'next-auth/react'

import { redirect } from 'next/navigation'

const ProductCardShopping = ({image,prix,des}) => {

const { data: session } = useSession({

        required: true,

        onUnauthenticated() {

        redirect('/signin?callbackUrl=/shoppingSpace')

        }

      })

return (

<div className="col-sm-3">

<div className="card">

<Image

src={image}

width={300}

height={200}

alt={des}

priority={true}

/>

<h6>{des}</h6>

<p>{prix}</p>

<button className="btn btn-warning">

        Add to cart

</button>

</div>

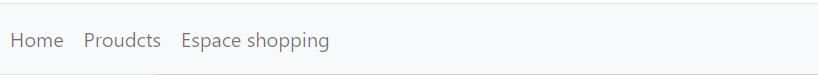
</div>

);

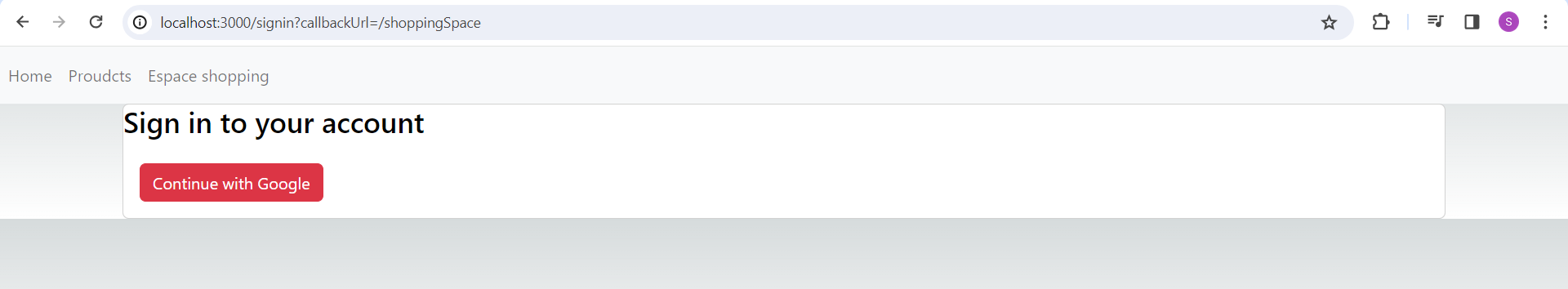
};

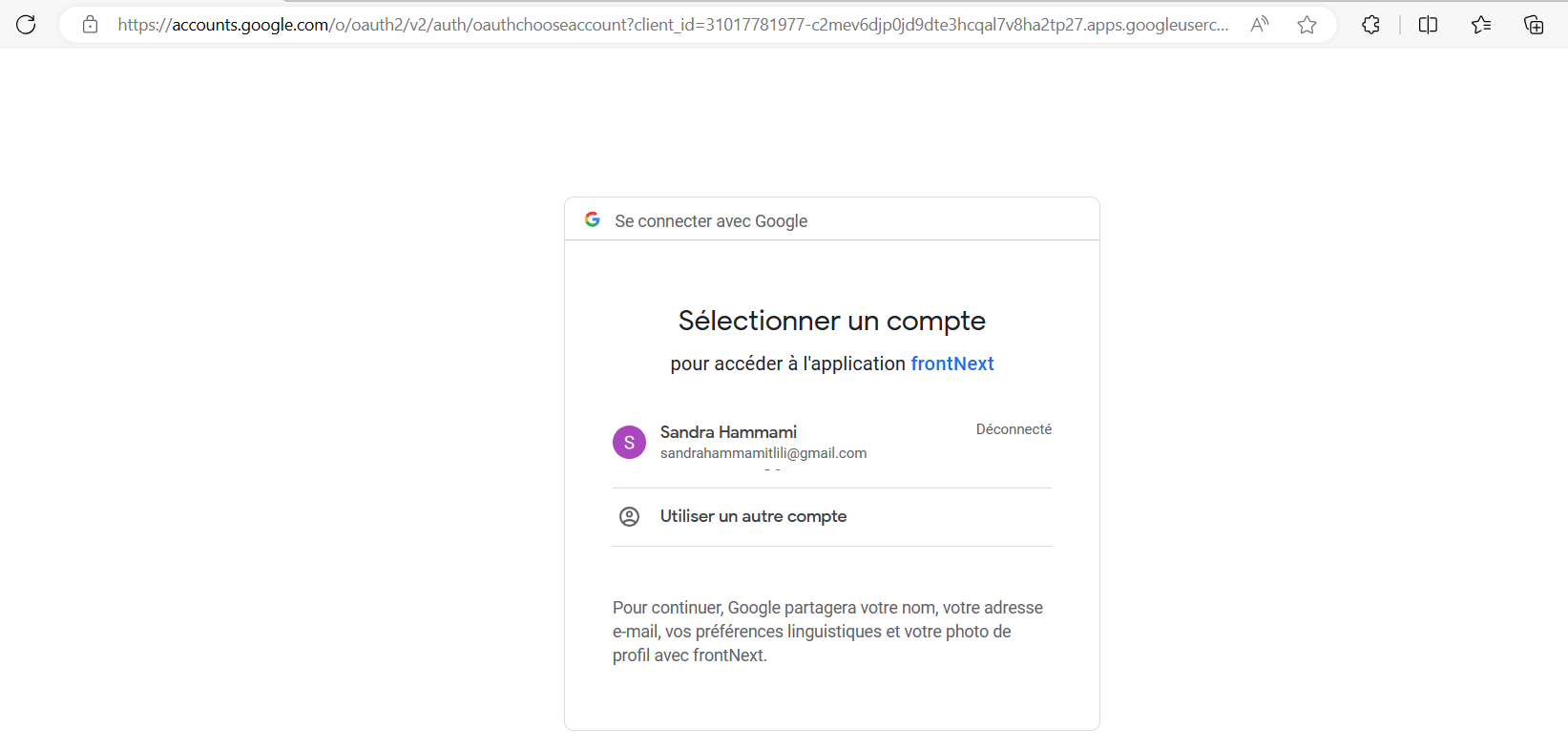
export default ProductCardShopping;

Si on choisit Espace shopping (page protégée)

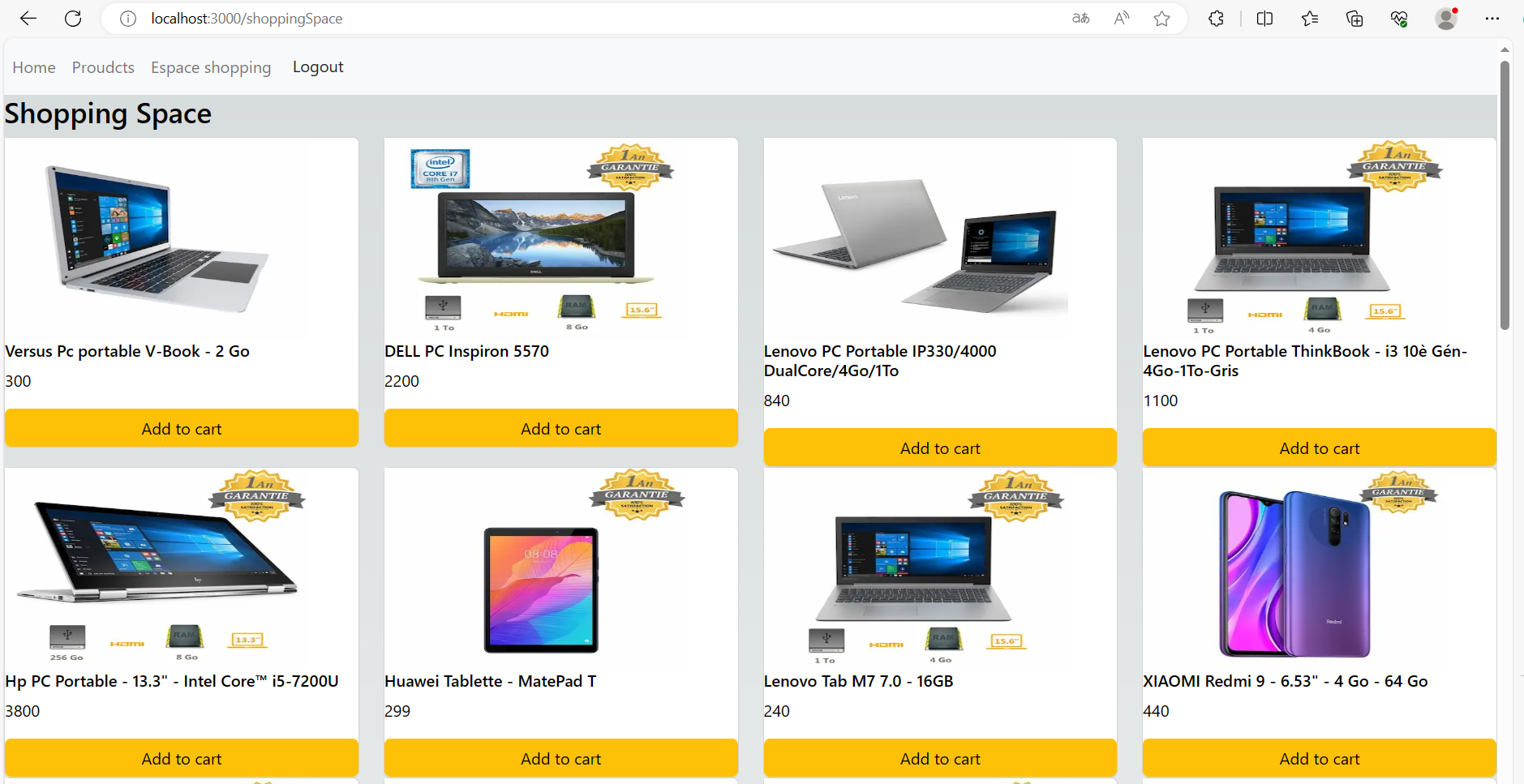


La page sigin apparaît





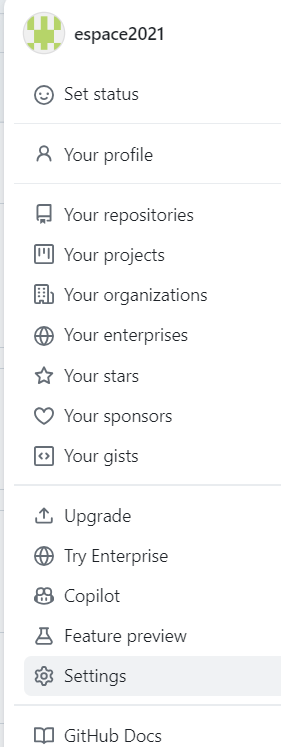
Si on clique sur logout (qui s'affiche après authentification) on a la page sigin de nouveau





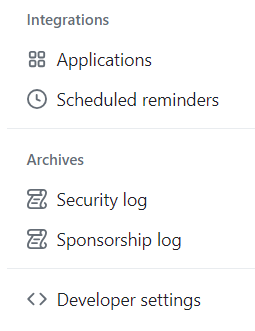
# GitHub Credentials

Connectez-vous à votre tableau de bord Github et cliquez sur l'image de profil dans le coin supérieur droit.

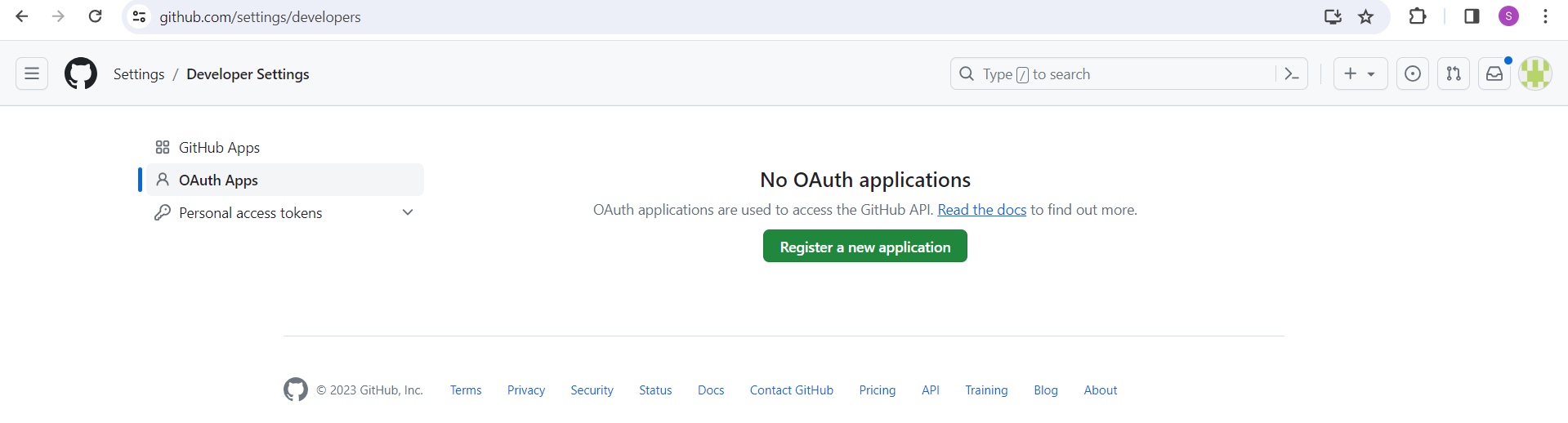


Sélectionnez Settings dans le menu déroulant :

Dans la page qui suit, faites défiler vers le bas et sélectionnez Developer settings :



Cliquez sur OAuth Apps sur la page suivante et choisissez Register a New Application :

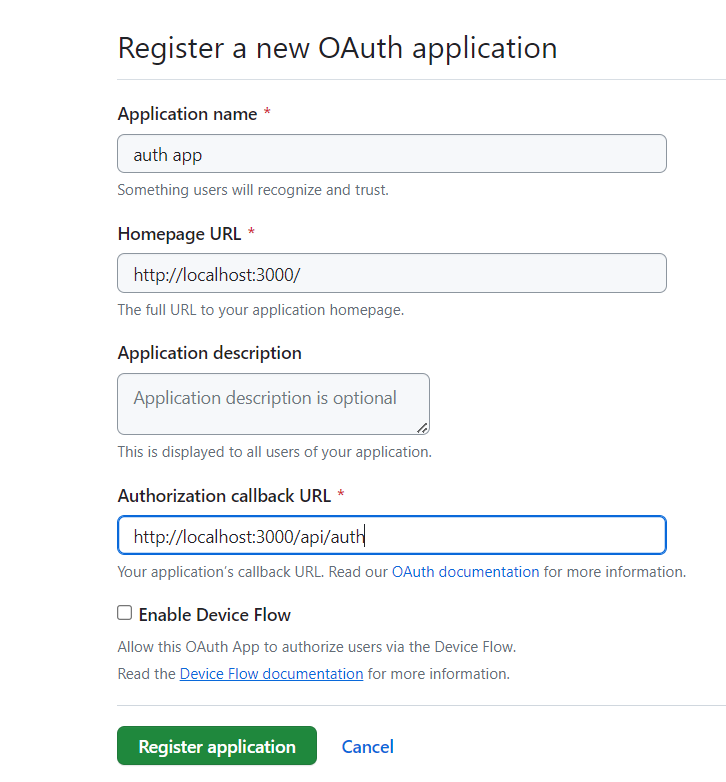


Remplissez le formulaire que vous voyez sur la page qui suit :

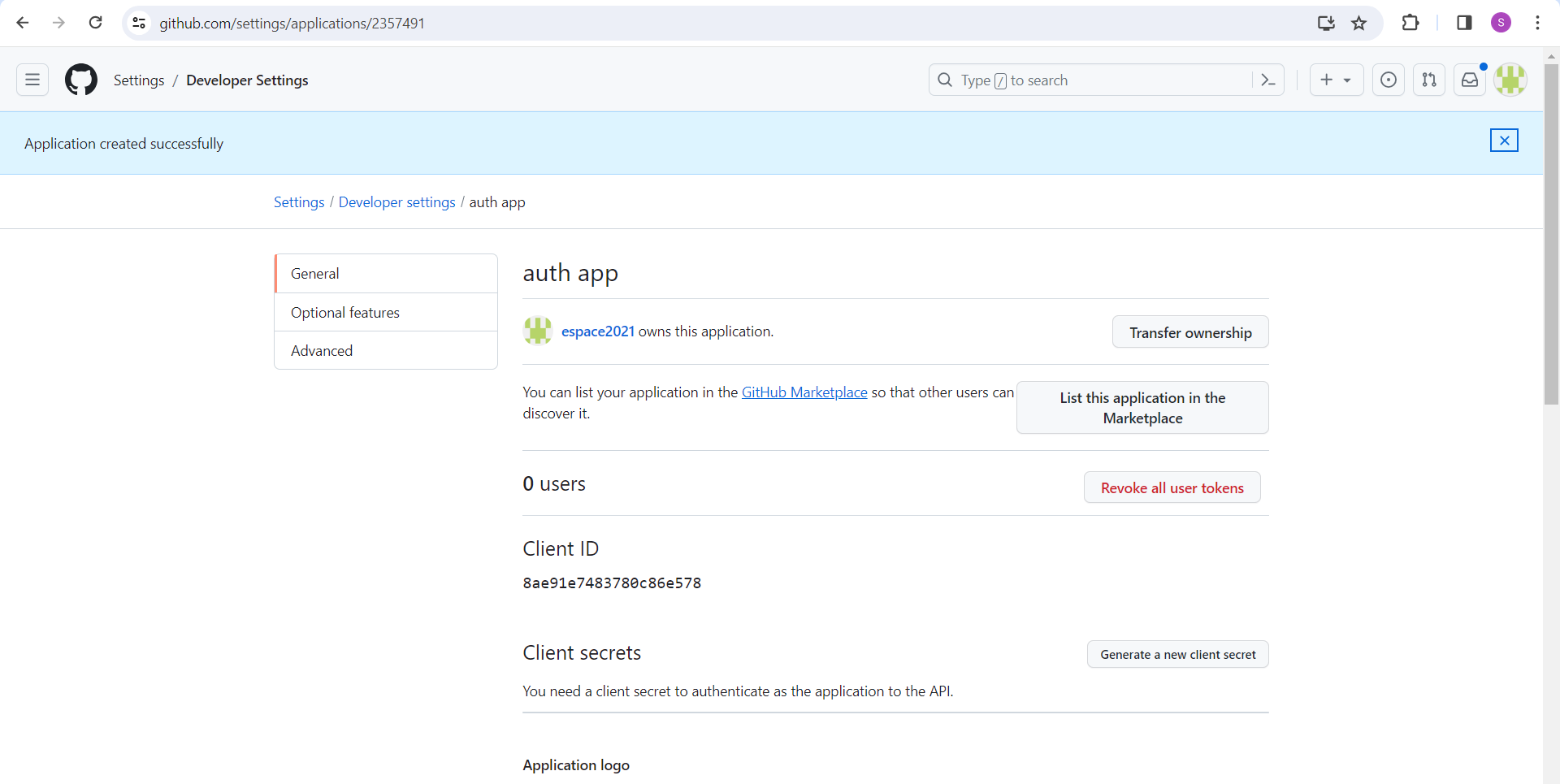
Application name - auth app

Homepage URL - http://localhost:3000/

Authorization callback URL - http://localhost:3000/api/auth



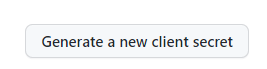
Cliquez sur le bouton Register Application. Une nouvelle page apparaît avec le GITHUB\_ID et le GITHUB\_SECRET :

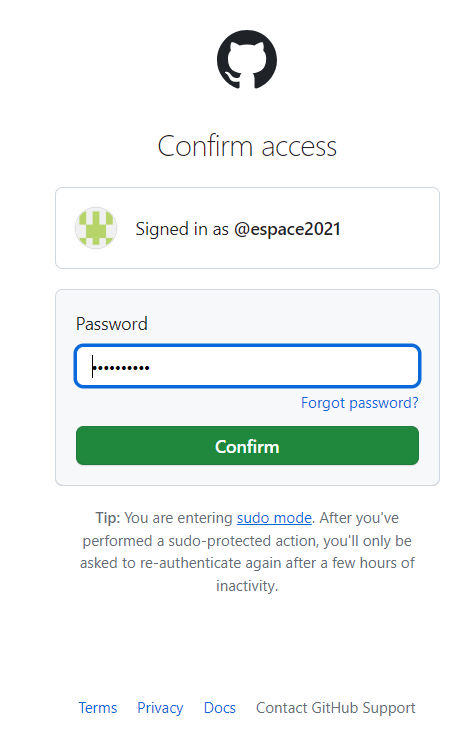


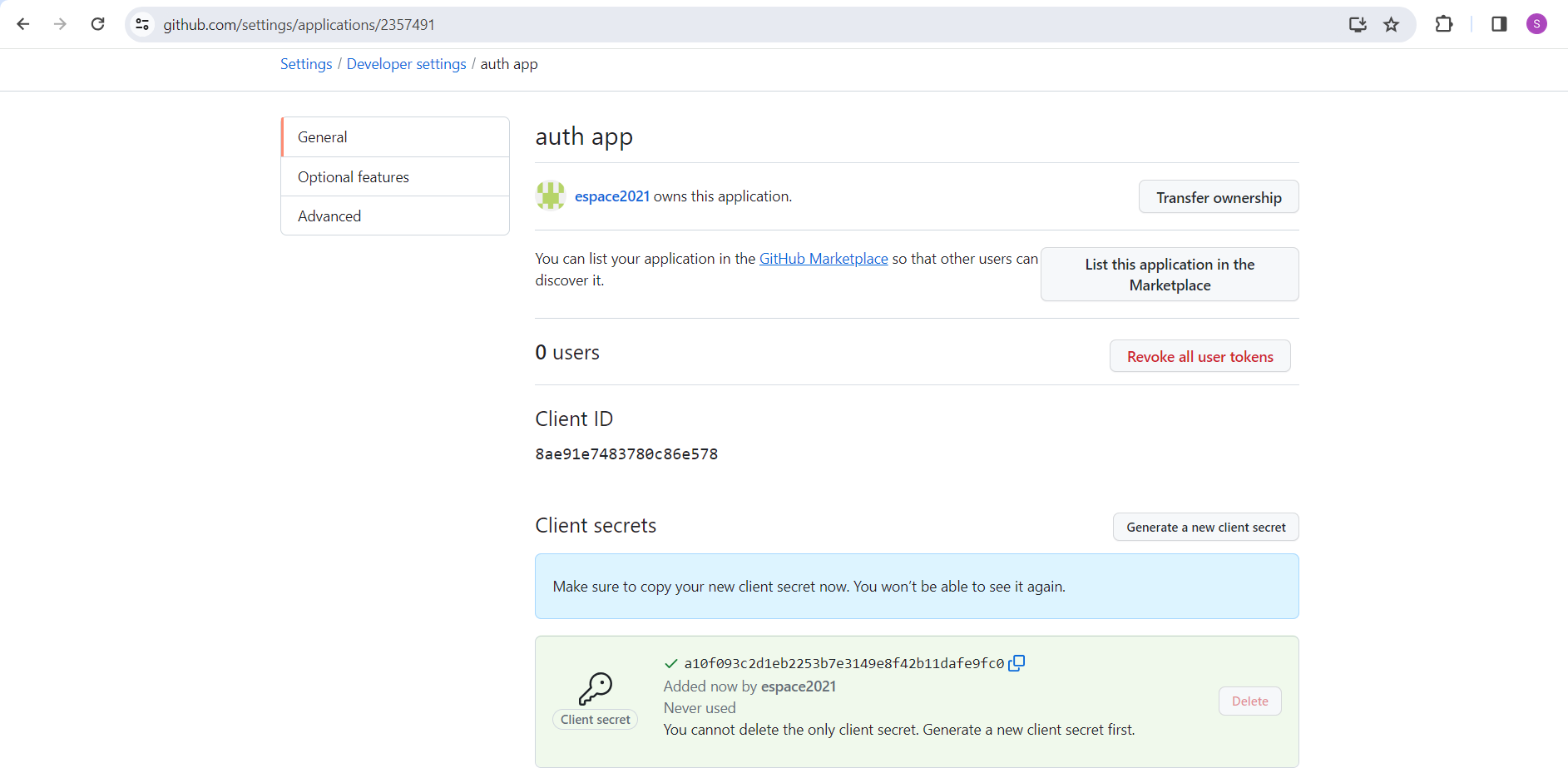
## Client ID

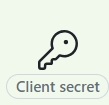
8ae91e7483780c86e578

Cliquez sur Generate a new client secret pour voir le GITHUB\_SECRET.









a10f093c2d1eb2253b7e3149e8f42b11dafe9fc0

# Modifier [..nextauth]/route.js



import NextAuth from "next-auth";

import GoogleProvider from "next-auth/providers/google";

import GitHubProvider from "next-auth/providers/github";

export const authOptions = {

    providers: [

        GitHubProvider({

            clientId: "8ae91e7483780c86e578",

            clientSecret: "a10f093c2d1eb2253b7e3149e8f42b11dafe9fc0",

          }),

        GoogleProvider({

            clientId: "31017781977-c2mev6djp0jd9dte3hcqal7v8ha2tp27.apps.googleusercontent.com",

            clientSecret: "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"

        })

    ],

    pages:{

        signIn: '/signin'

    }

}

const handler = NextAuth(authOptions)

export { handler as GET, handler as POST }

# Modifier SignIn



'use client';

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

import { useSearchParams } from 'next/navigation'

const SignInPage = () => {

    const searchParams = useSearchParams()

    const callbackUrl = searchParams.get('callbackUrl')

  return (

        <div className="container">

           <div className="card">

           <div className="card-head">

            <h3>Sign in to your account</h3>

           </div>

           <div className="card-body">

     <button className="btn btn-dark" onClick={() => signIn('github', { callbackUrl })}>    Continue with GitHub </button>

        </div>

           <div className="card-body">

     <button className="btn btn-danger" onClick={() => signIn('google', { callbackUrl })}>    Continue with Google </button>

        </div>

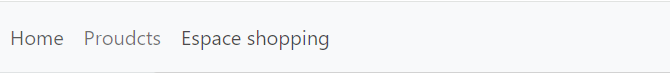
        </div>

        </div>

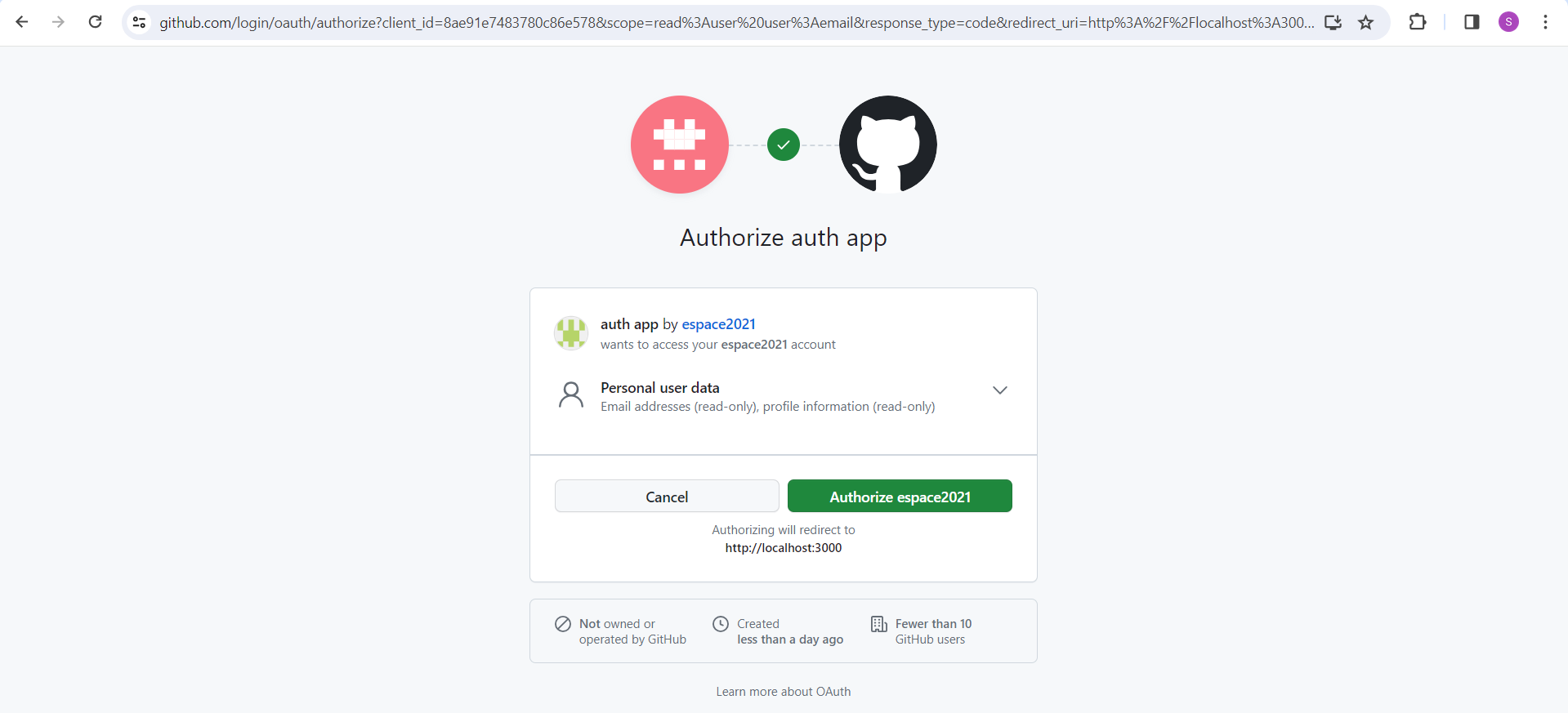
  )

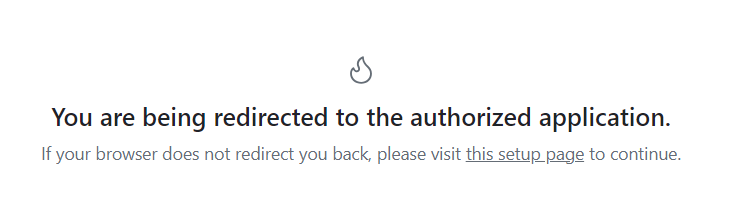
}

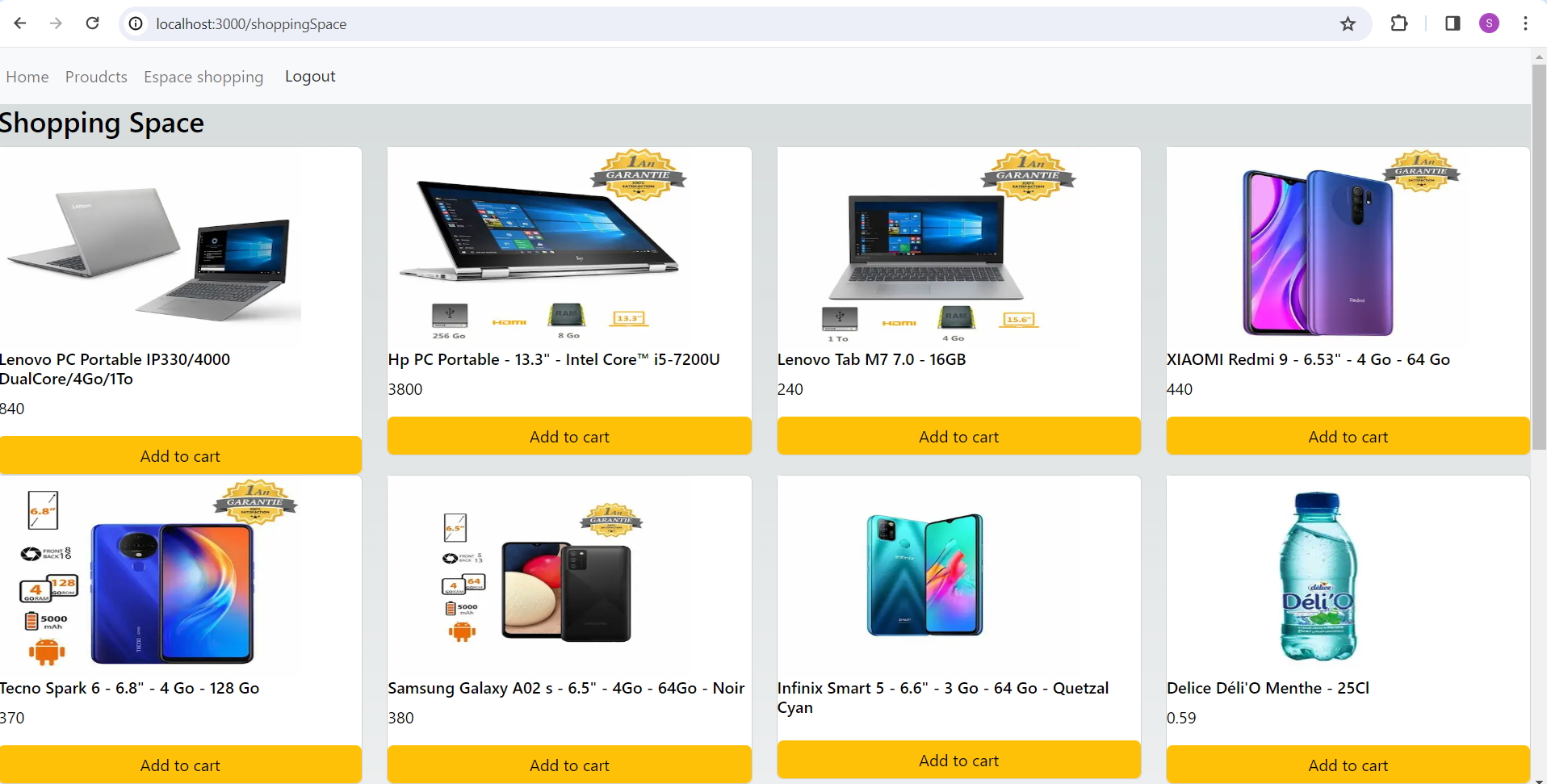
export default SignInPage











# Email Credentials

Vous pouvez utiliser n'importe quel fournisseur de messagerie proposant des services SMTP. Ces étapes utiliseront Gmail. Vous pouvez consulter Mailgun, SendGrid et autres.

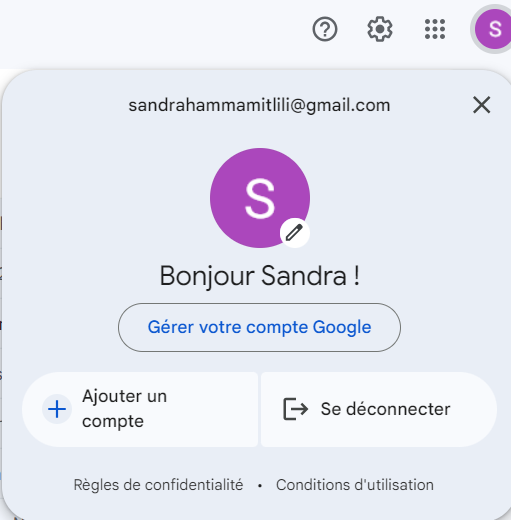
Vous aurez besoin des détails suivants :

* EMAIL\_SERVER\_HOST
* EMAIL\_SERVER\_PORT
* EMAIL\_SERVER\_USER
* EMAIL\_SERVER\_PASSWORD
* EMAIL\_FROM

Les étapes suivantes vous mèneront à votre EMAIL\_SERVER\_PASSWORD :

Connectez-vous à votre compte Gmail.

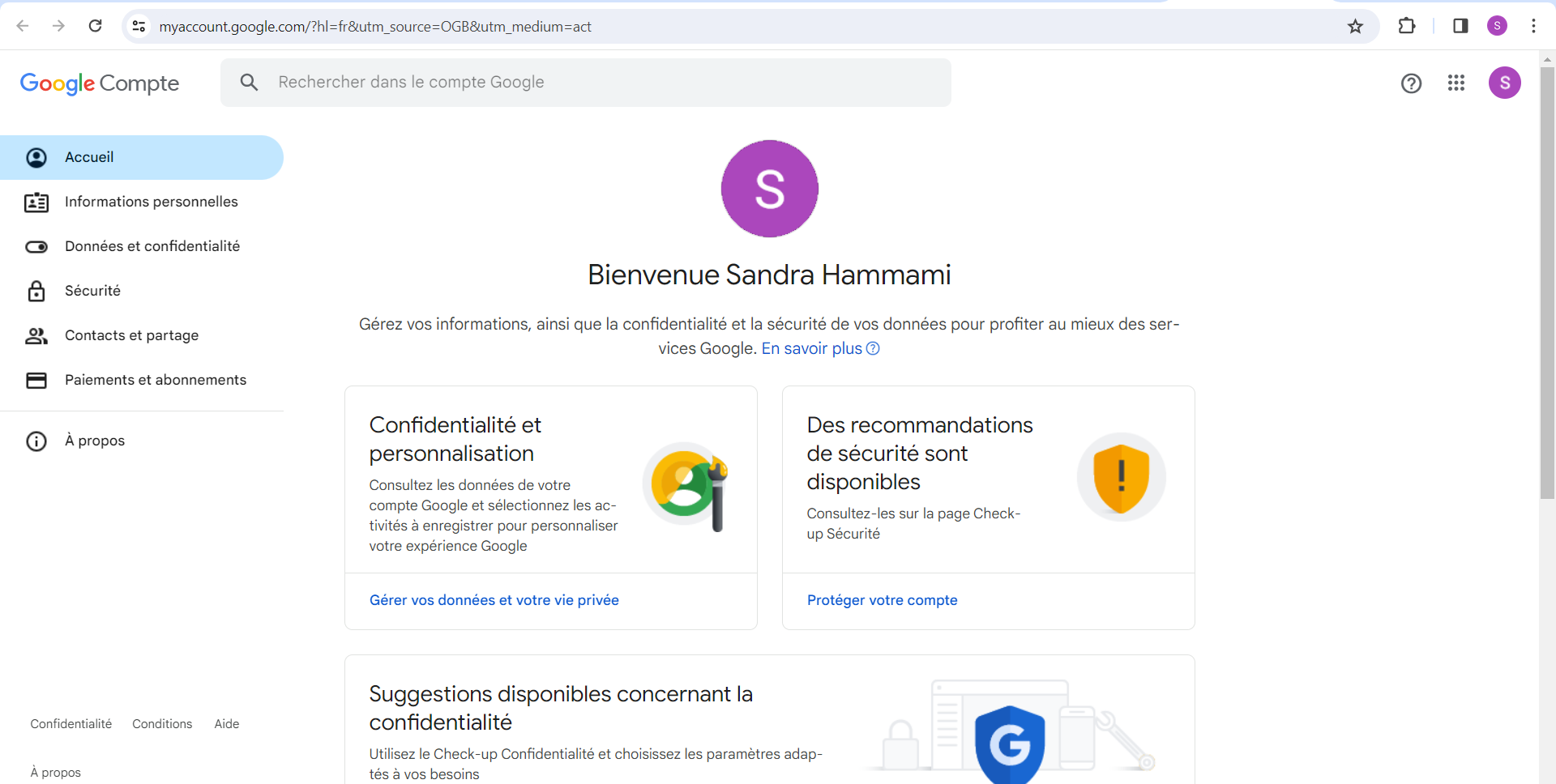
Cliquez sur votre image de profil. Il déroulera un menu.



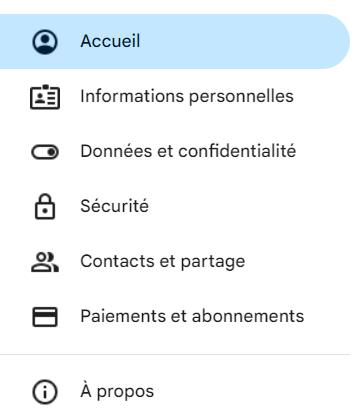
Cliquez sur le bouton Gérer votre compte Google



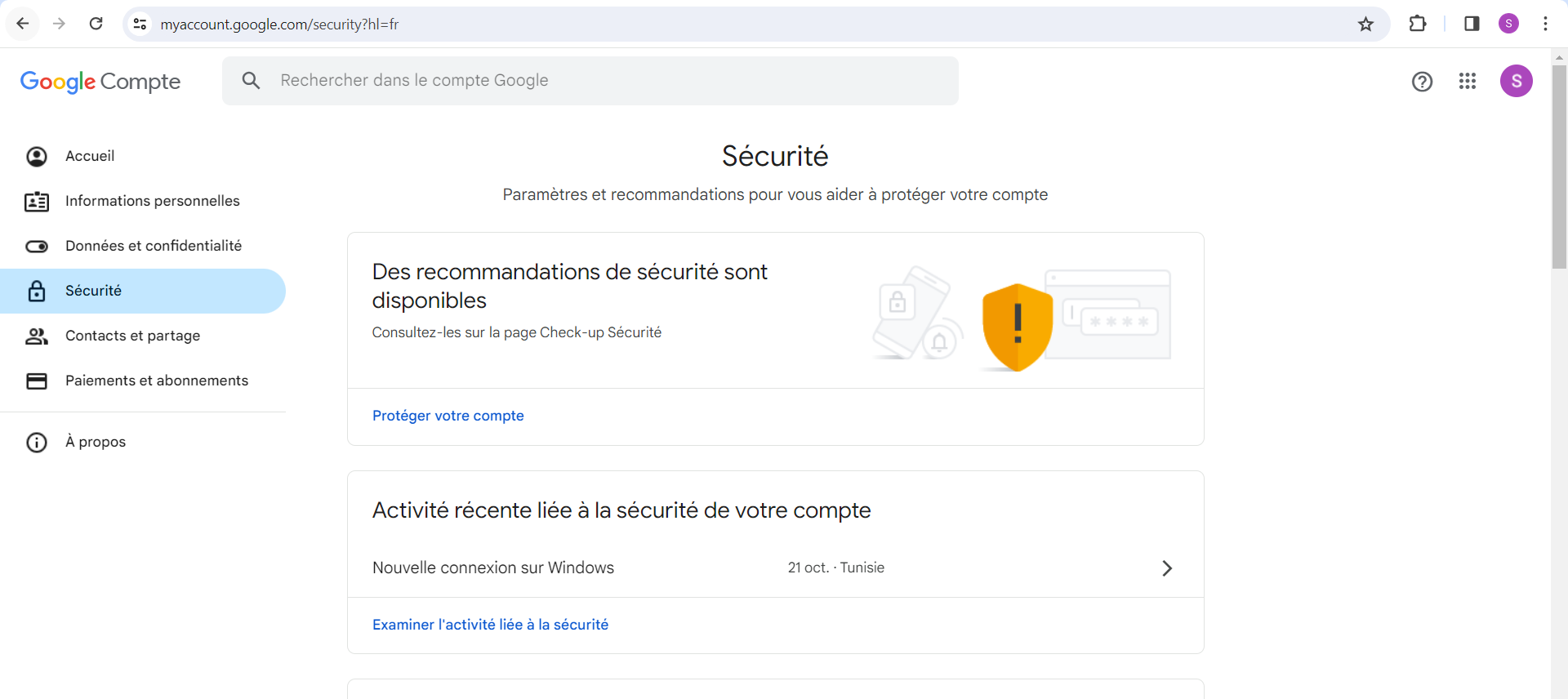
Vous serez redirigé vers la page ci-dessous :



Cliquez sur Sécurité dans le menu latéral.



La page suivante apparaîtra.

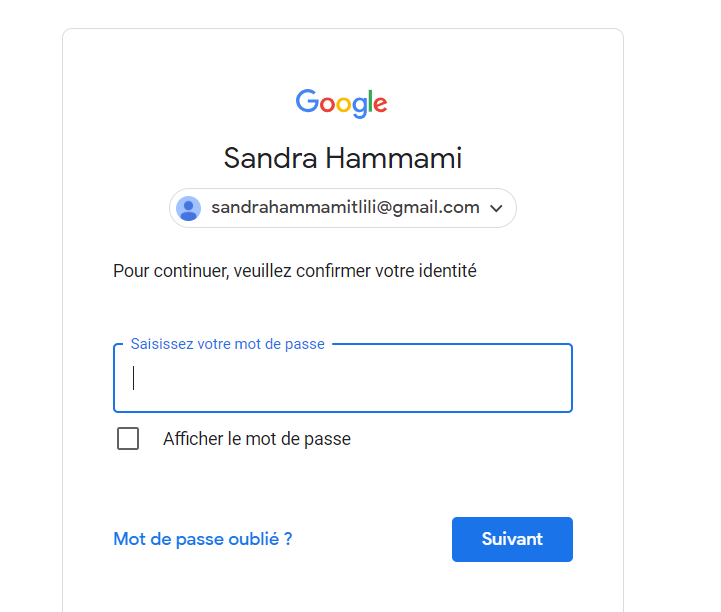


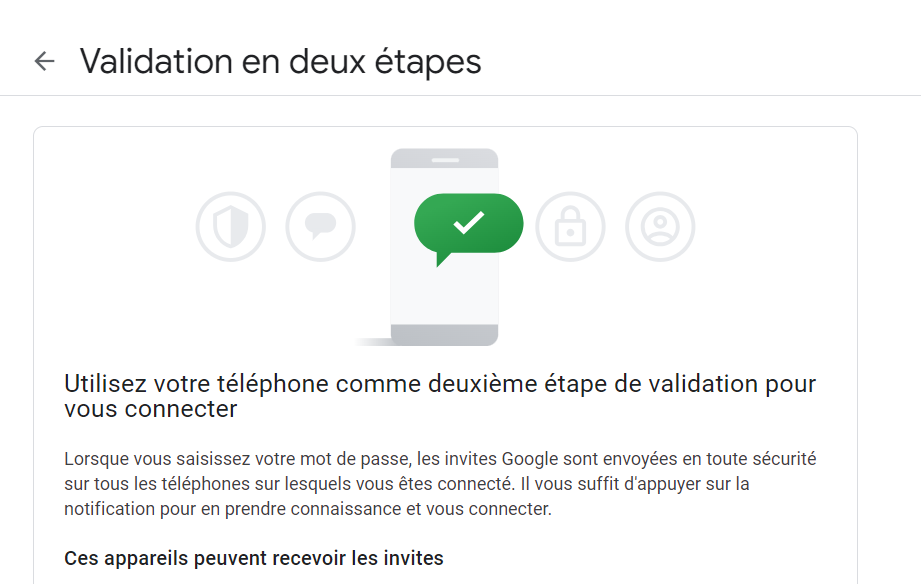
Vous devrez configurer la vérification en deux étapes pour que ce processus fonctionne.





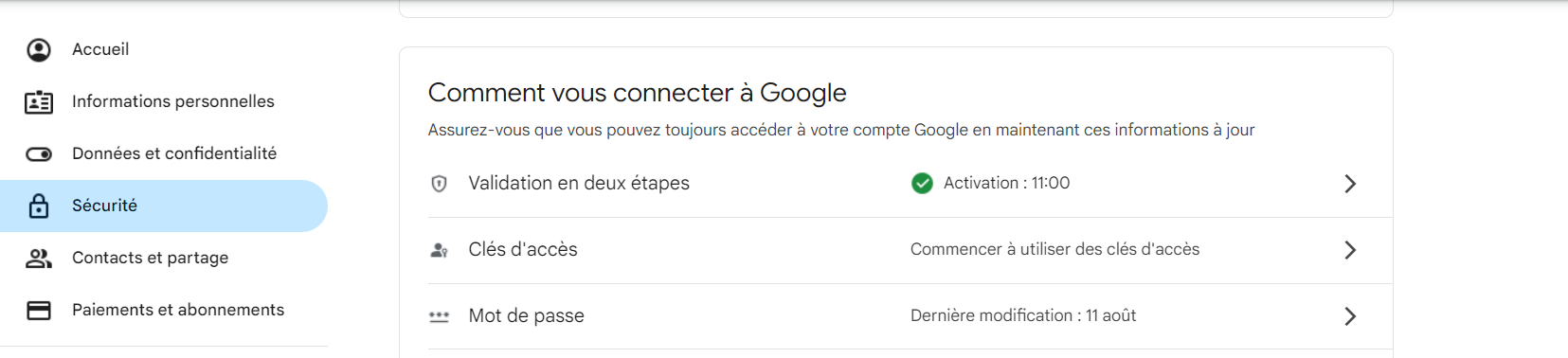
Cliquer sur commencer puis saisir le mot de passe







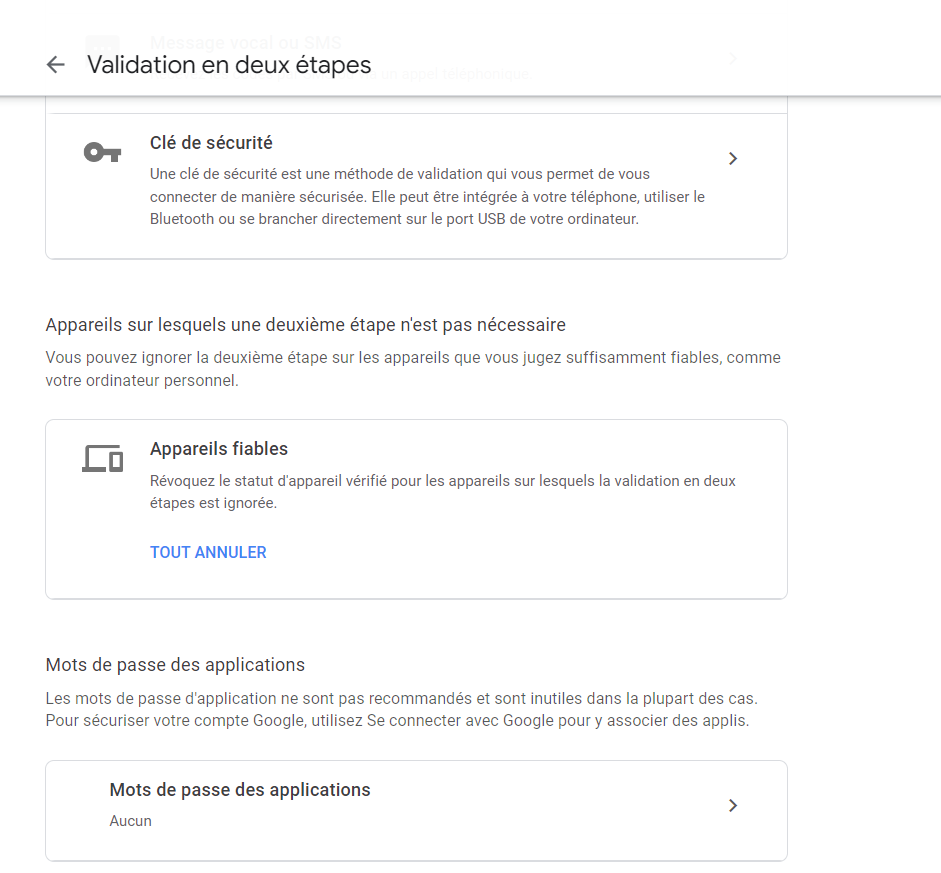
Maintenant dans la rubrique sécurité on voit que la validation en deux étapes est activée



Cliquer sur :



Après quoi vous serez redirigé vers la page suivante :

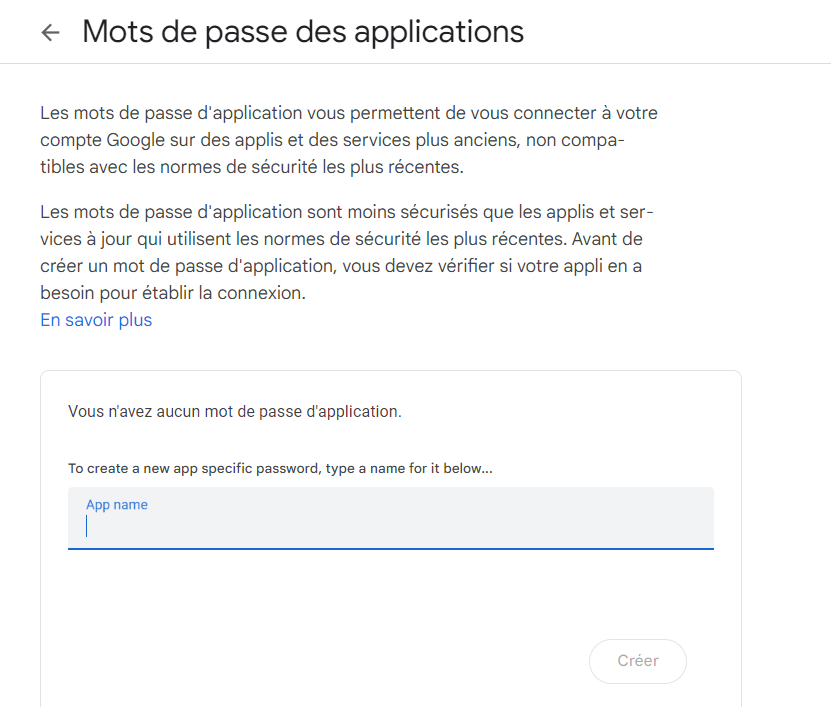


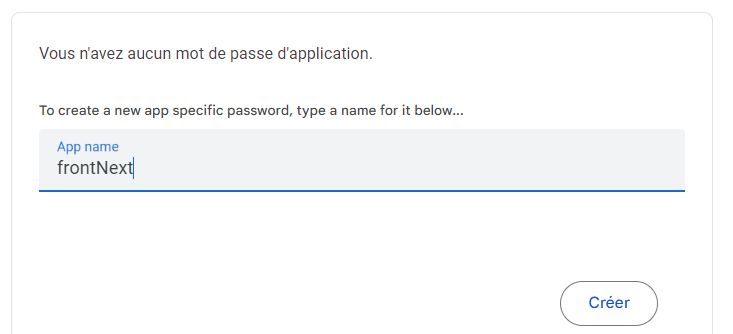
Faites défiler vers le bas et cliquez sur Mots de passe d'application, comme dans l'image ci-dessus.

Cliquer sur

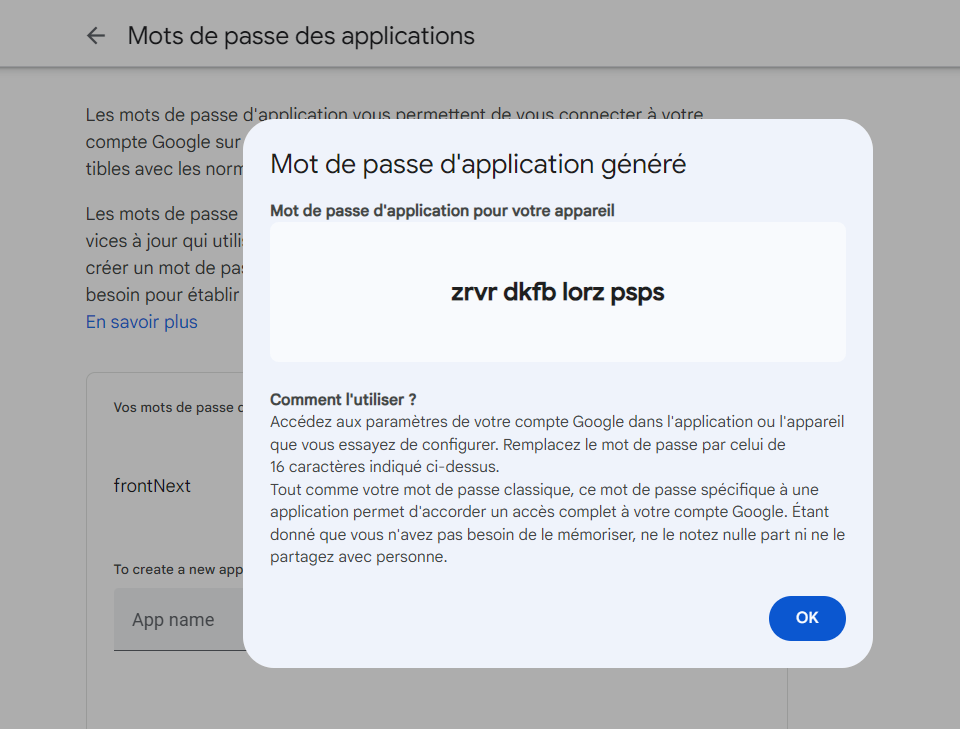


Ce qui affiche

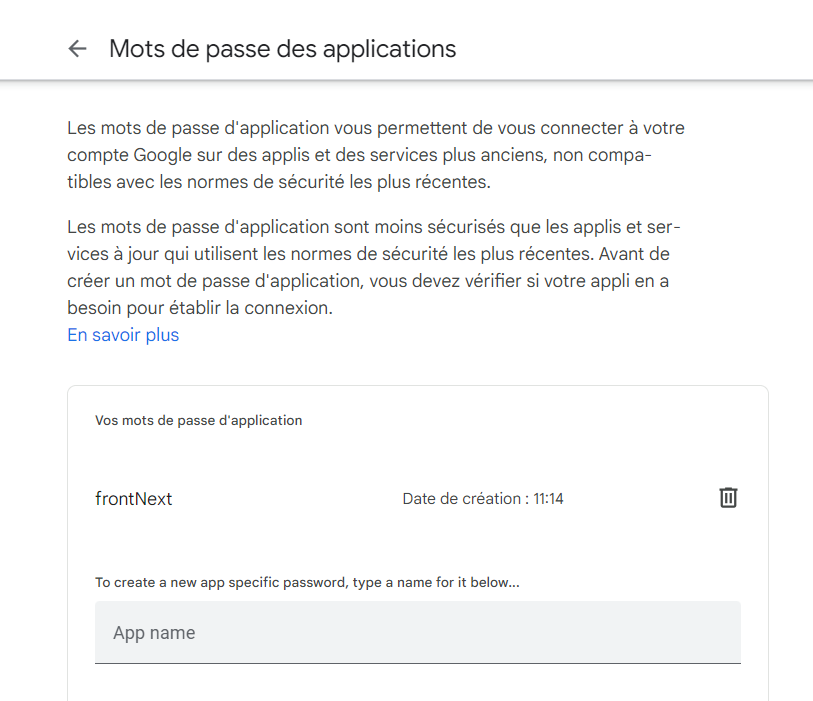




Cliquer sur créer



**zrvr dkfb lorz psps**



L’Email credentials sera comme suit :

EMAIL\_SERVER\_HOST="smtp.gmail.com"

EMAIL\_SERVER\_PORT=465

EMAIL\_SERVER\_USER="<email\_address>". Exemple “sandrahammamitlili@gmail.com”

EMAIL\_SERVER\_PASSWORD="zrvrdkfblorzpsps"

EMAIL\_FROM="<email\_address\_you\_want\_recipient\_to\_see\_as\_the\_sender>". Il n’est pas nécessaire que cet e-mail existe. Exemple “ABC@gmail.com”

# Installer package nodemailer

Pour utiliser NodeMailer avec NextAuth.js, vous devrez généralement configurer NodeMailer pour qu'il utilise un service SMTP de votre choix (par exemple, Gmail, SendGrid) et créer des fonctions pour envoyer des e-mails à partir de vos actions NextAuth.js, par exemple lors de l'inscription d'un nouvel utilisateur ou de la réinitialisation d'un mot de passe.

NodeMailer est souvent utilisé dans les applications NextAuth.js pour gérer les communications par e-mail liées à l'authentification, offrant ainsi un moyen flexible et personnalisable d'interagir avec les utilisateurs via des e-mails.

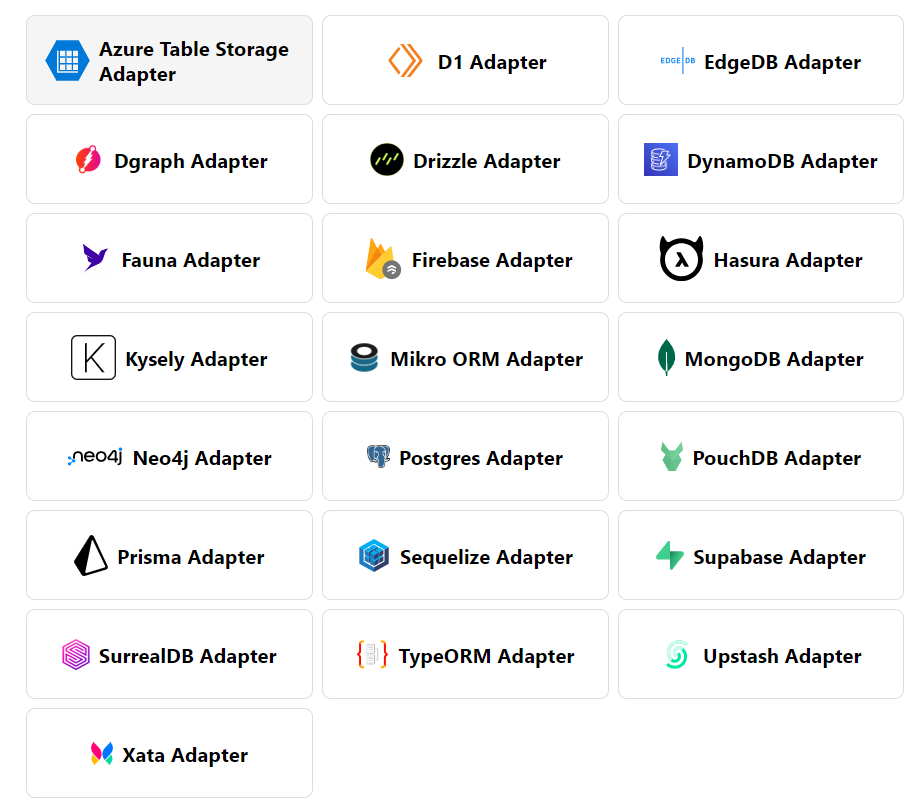
npm install nodemailer

# Adapter

Un adaptateur (ou adapter en anglais) est un composant logiciel qui sert à connecter ou à faire le lien entre deux parties d'un système informatique qui ont des interfaces incompatibles ou différentes. Les adaptateurs sont couramment utilisés pour rendre compatible l'interaction entre deux composants logiciels ou matériels qui ont été conçus de manière indépendante ou qui utilisent des protocoles différents.

Dans le contexte de l'authentification et de l'autorisation, un adaptateur peut être utilisé pour connecter un système d'authentification existant à une application.

À l'aide d'un adaptateur Auth.js / NextAuth.js, vous pouvez vous connecter à n'importe quel service de base de données ou même à plusieurs services différents en même temps. Les adaptateurs officiels répertoriés suivants sont créés et gérés par la communauté :



Nous allons choisir **MongoDB**

Installer package :

npm install -D @auth/mongodb-adapter mongodb

Créer .env :

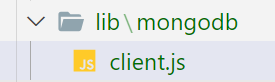


ENV=DEVELOPMENT

MONGODB\_URI\_LOCAL=mongodb://127.0.0.1:27017/DatabaseCommerce

MONGODB\_URI=mongodb+srv://hassan:messk2015@cluster0.yfhhu.mongodb.net/DatabaseCommerce?retryWrites=true&w=majority

Puis créer un simple MongoDB client. Créer dossier lib, dans lequel créer le dossier mongodb. Puis y créer le fichier client.js



import { MongoClient } from "mongodb"

const uri = process.env.MONGODB\_URI

const options = {

  useUnifiedTopology: true,

  useNewUrlParser: true,

}

let client

let clientPromise

if (!process.env.MONGODB\_URI) {

  throw new Error("Please add your Mongo URI to .env.local")

}

if (process.env.NODE\_ENV === "development") {

  // In development mode, use a global variable so that the value

  // is preserved across module reloads caused by HMR (Hot Module Replacement).

  if (!global.\_mongoClientPromise) {

    client = new MongoClient(uri, options)

    global.\_mongoClientPromise = client.connect()

  }

  clientPromise = global.\_mongoClientPromise

} else {

  // In production mode, it's best to not use a global variable.

  client = new MongoClient(uri, options)

  clientPromise = client.connect()

}

// Export a module-scoped MongoClient promise. By doing this in a

// separate module, the client can be shared across functions.

export default clientPromise

# Modifier [..nextauth]/route.js



import NextAuth from "next-auth";

import GoogleProvider from "next-auth/providers/google";

import GitHubProvider from "next-auth/providers/github";

import EmailProvider from "next-auth/providers/email";

export const authOptions = {

    providers: [

        EmailProvider({

        server: {

          host: "smtp.gmail.com",

          port: 587,

          auth: {

            user: "sandrahammamitlili@gmail.com",

            pass: "zrvrdkfblorzpsps",

          },

        },

        from: "ABC@gmail.com",

      }),

        GitHubProvider({

            clientId: "8ae91e7483780c86e578",

            clientSecret: "a10f093c2d1eb2253b7e3149e8f42b11dafe9fc0",

          }),

        GoogleProvider({

            clientId: "31017781977-c2mev6djp0jd9dte3hcqal7v8ha2tp27.apps.googleusercontent.com",

            clientSecret: "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"

        })

    ],

    pages:{

        signIn: '/signin'

    }

}

const handler = NextAuth(authOptions)

export { handler as GET, handler as POST }

# Modifier SignIn

