

# ATELIER 8 : STOCKAGE DES IMAGES AVEC FIREBASE DANS NEXT JS

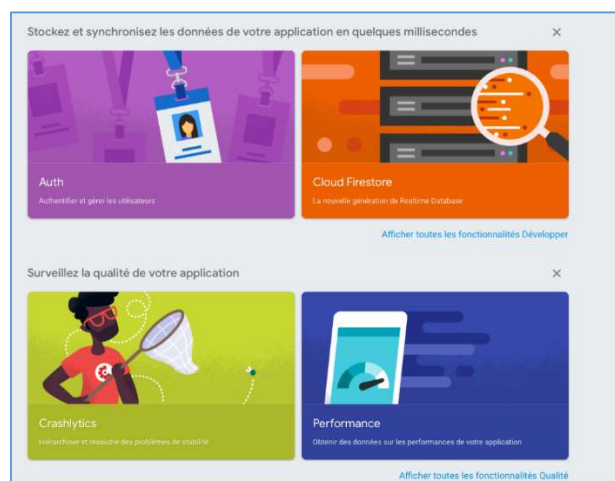
## Introduction



Firebase est le nom d'une plateforme mobile de Google qui facilite la création de back-end à la fois scalable et performant. En d'autres termes, il s'agit d'une plateforme qui permet de développer rapidement des applications pour mobile et pour le web.

Dans Firebase, vous trouverez des API intuitives regroupées dans un SDK unique. Ces API, en plus de vous faire gagner du temps, vous permettent de réduire le nombre d'intégrations que vous devez gérer par le biais de votre application.

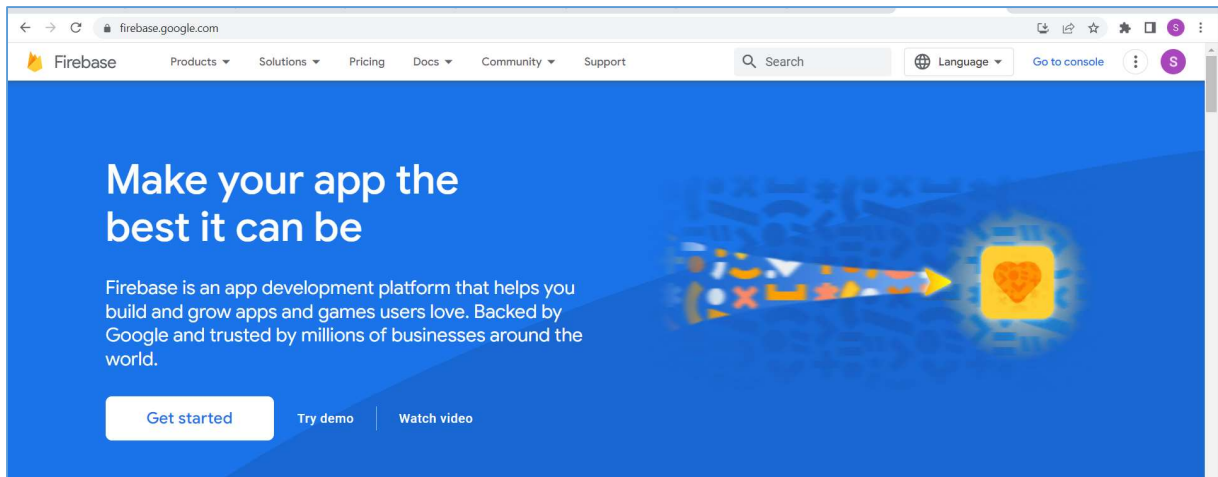
Voici quelques-uns de ces outils, des outils que les développeurs exploitent le plus dans le cadre du développement d'amplifications ou encore du test de performance des amplifications :



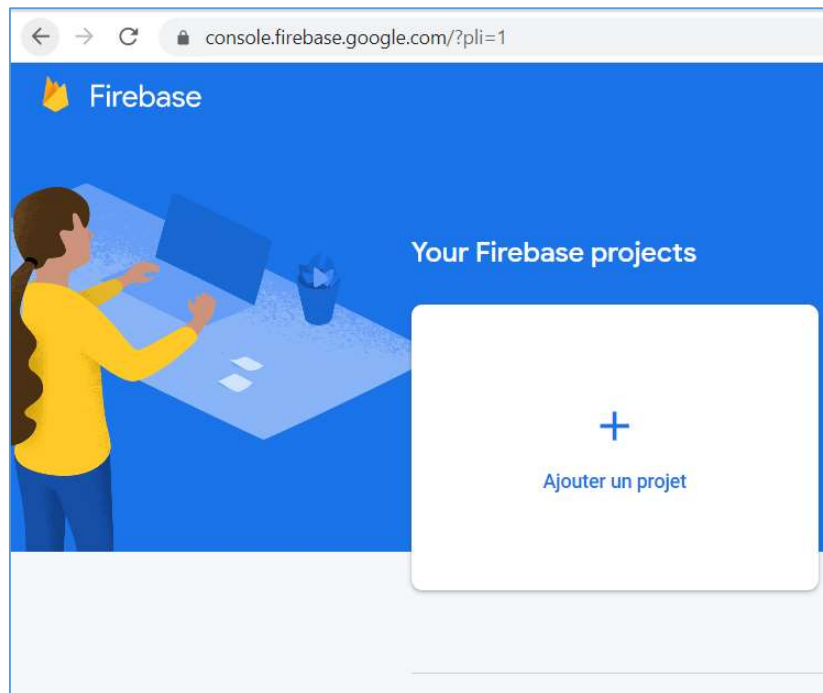
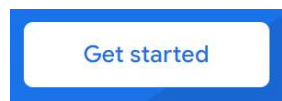
## Configurer Firebase

Accéder au site :

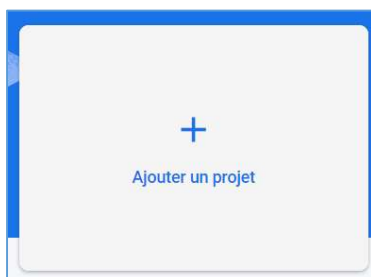
<https://console.firebase.google.com/>



Cliquer sur

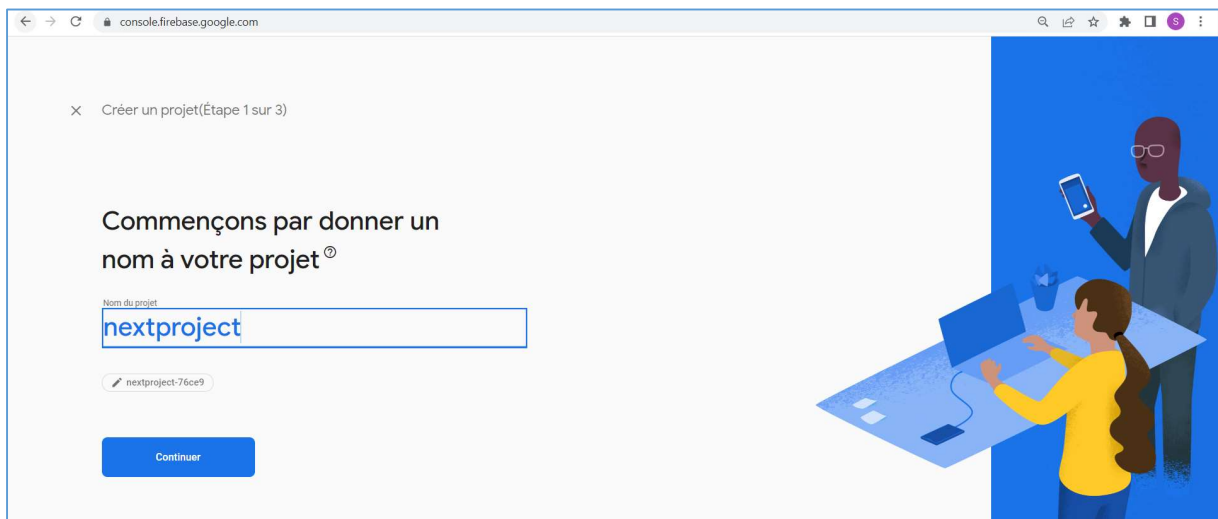


Cliquer sur

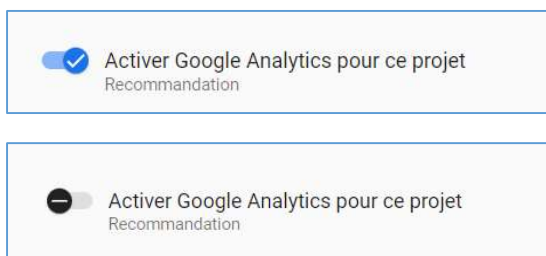


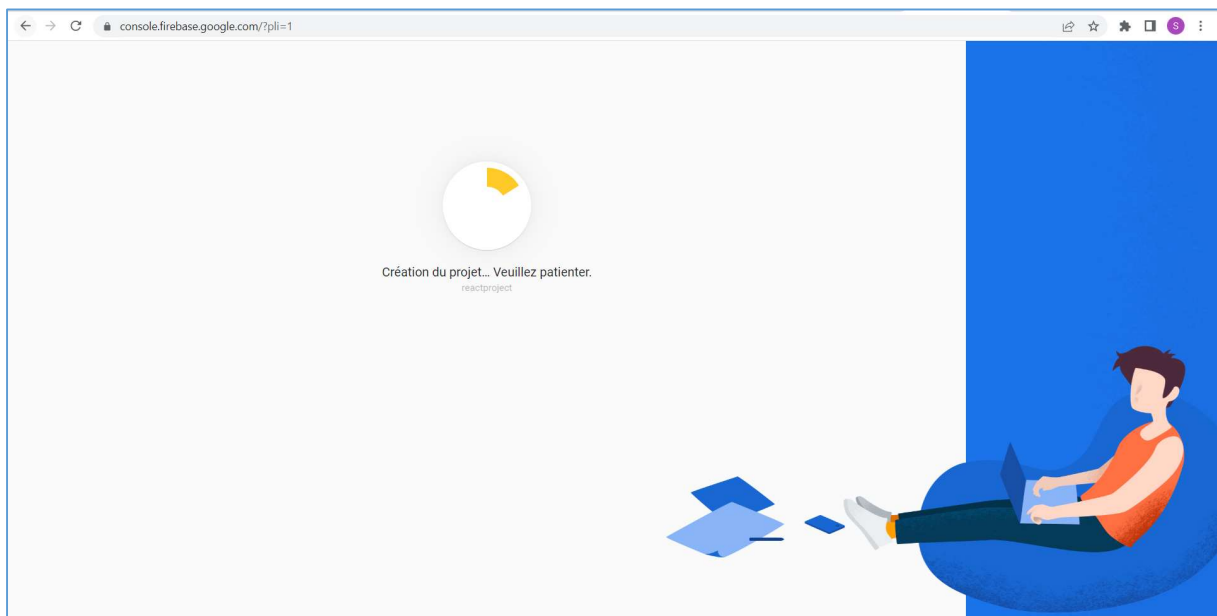
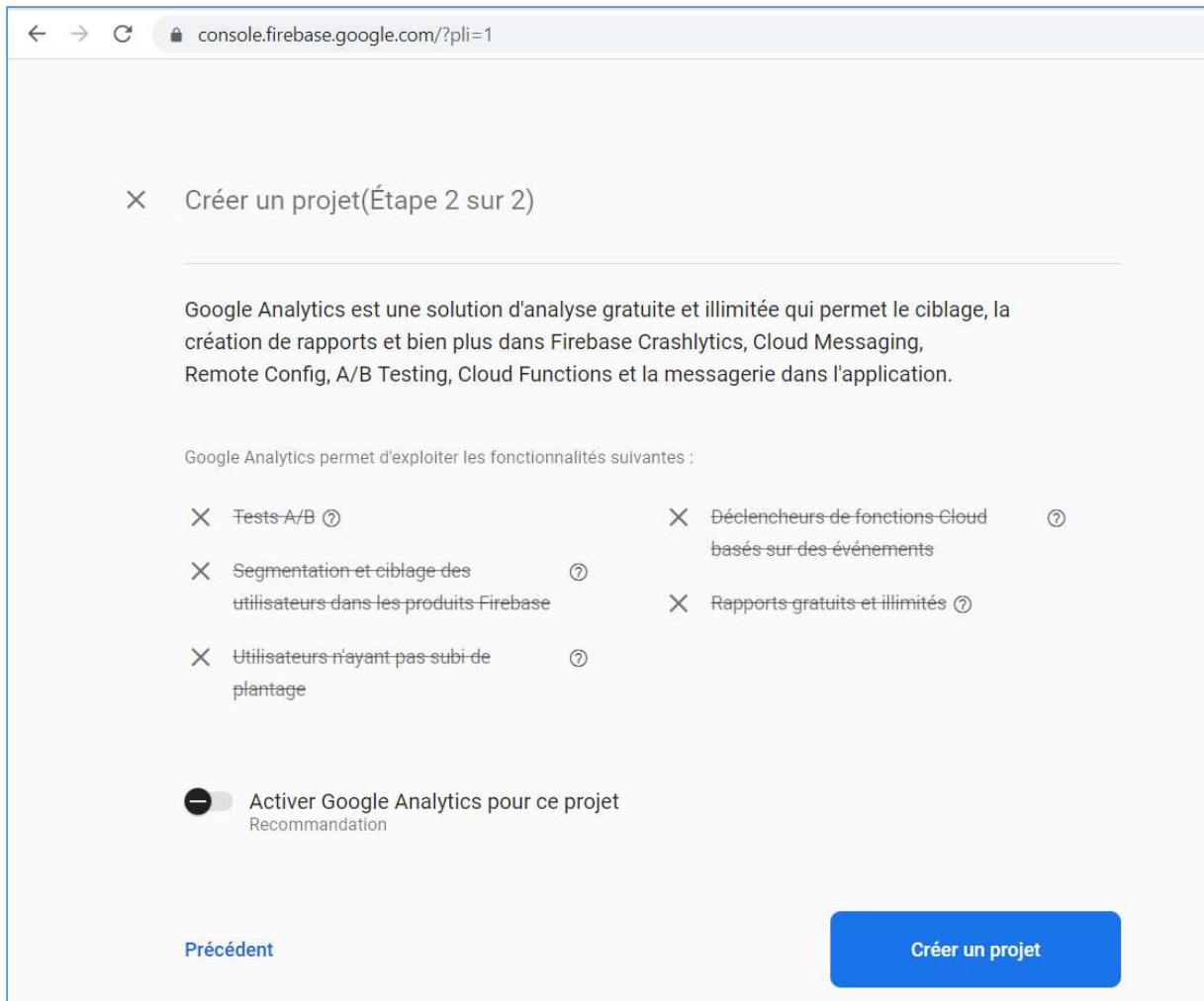


Donner un nom pour le projet puis continuer :

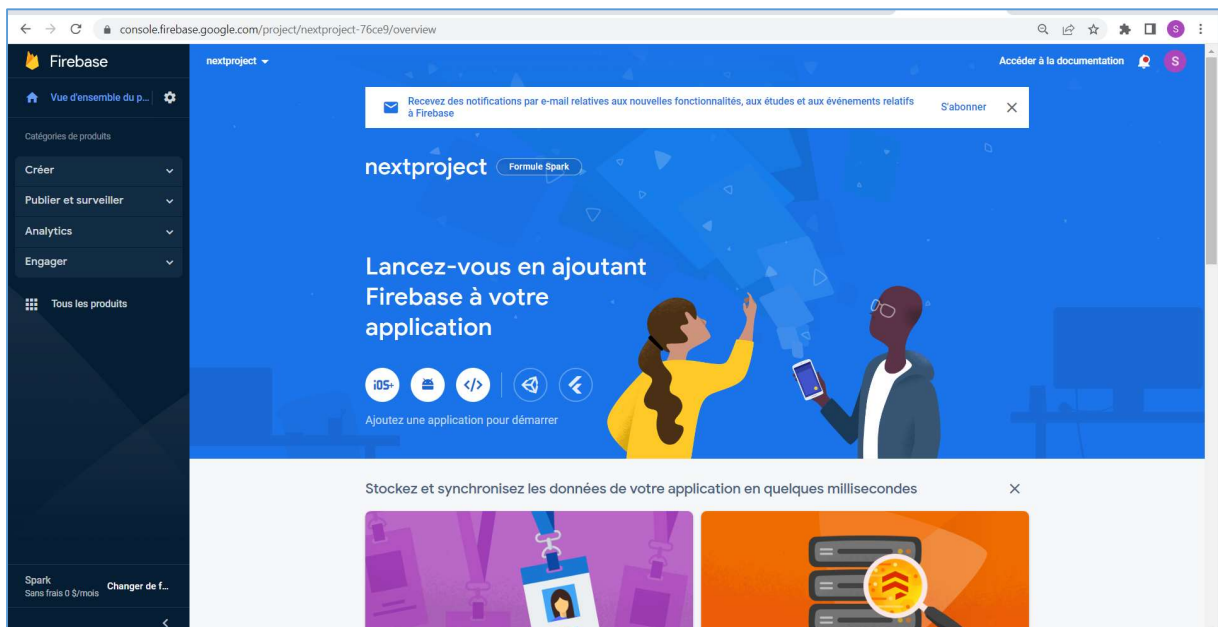
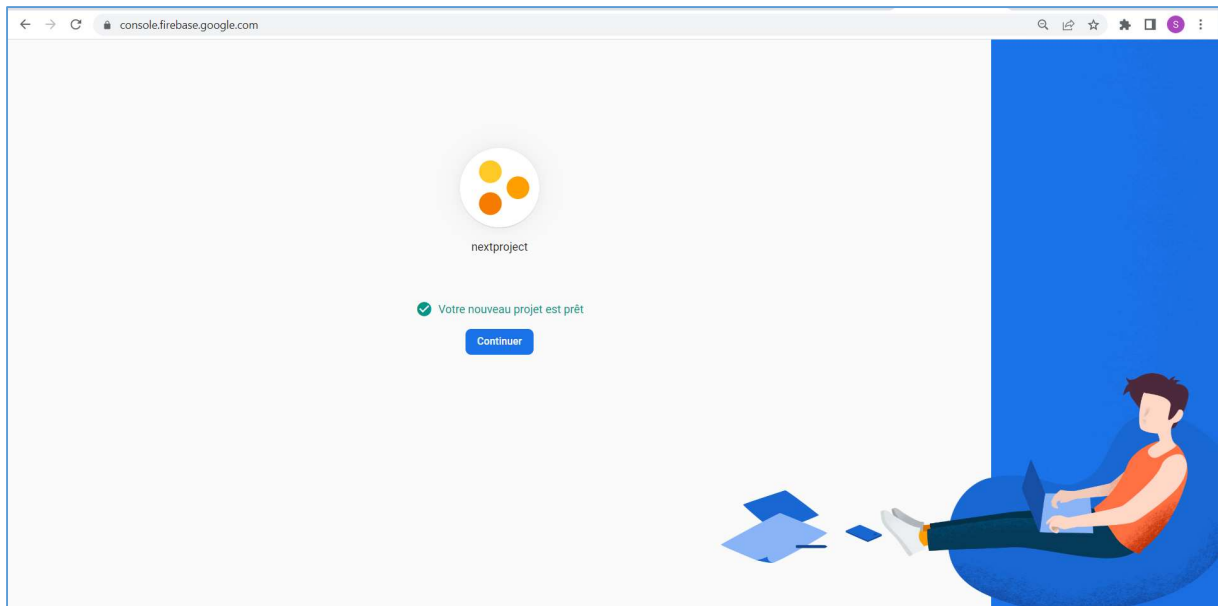


Décocher Google Analytics puis valider





Cliquer sur continuer

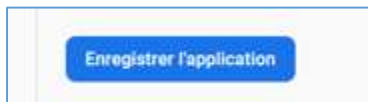
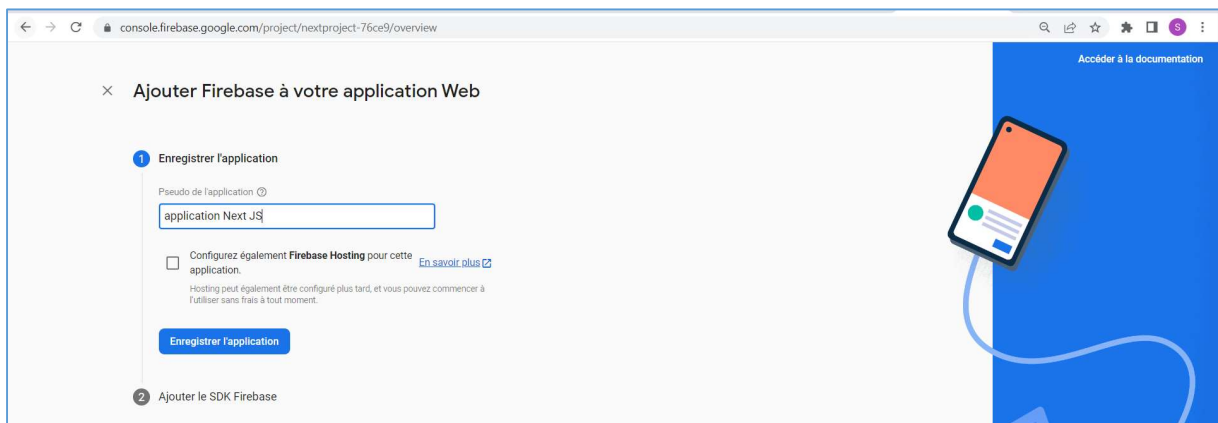
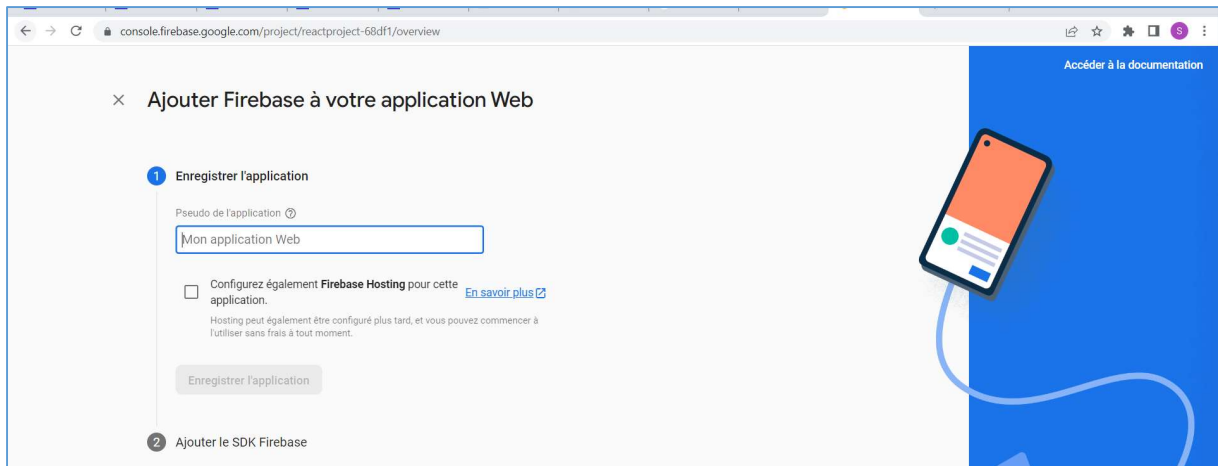


Cliquer sur l'icône Web

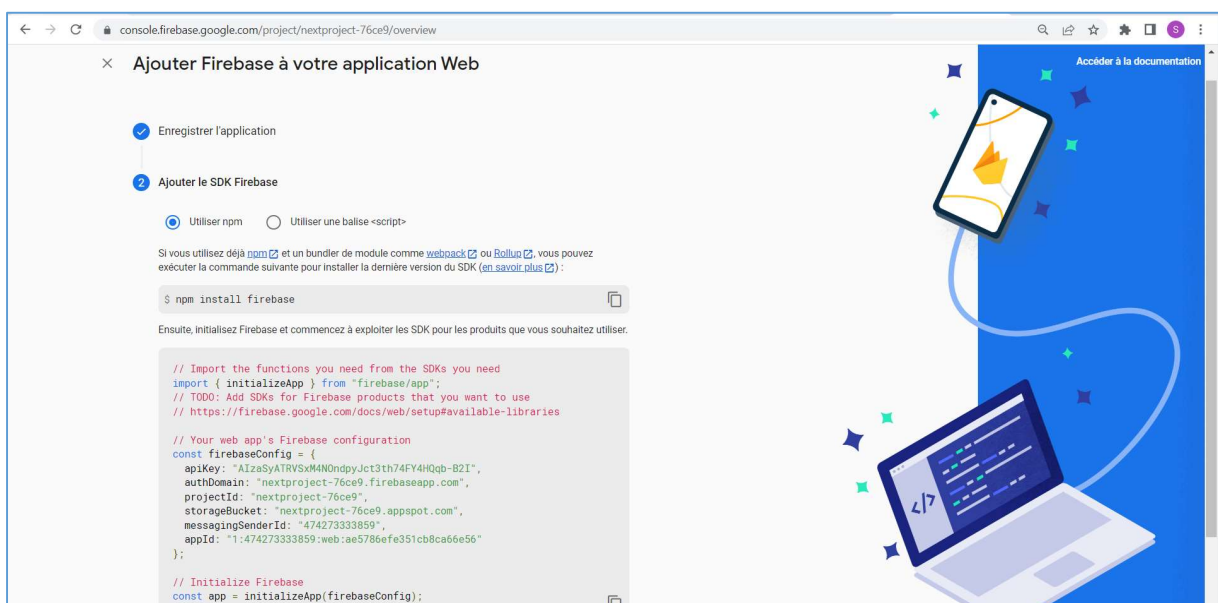


Donnez un surnom à votre application et cliquez sur S'inscrire.

Copiez l'objet de configuration fourni. Vous en aurez besoin pour connecter votre application à Firebase.



Copiez l'objet de configuration fourni. Vous en aurez besoin pour connecter votre application à Firebase.



```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";

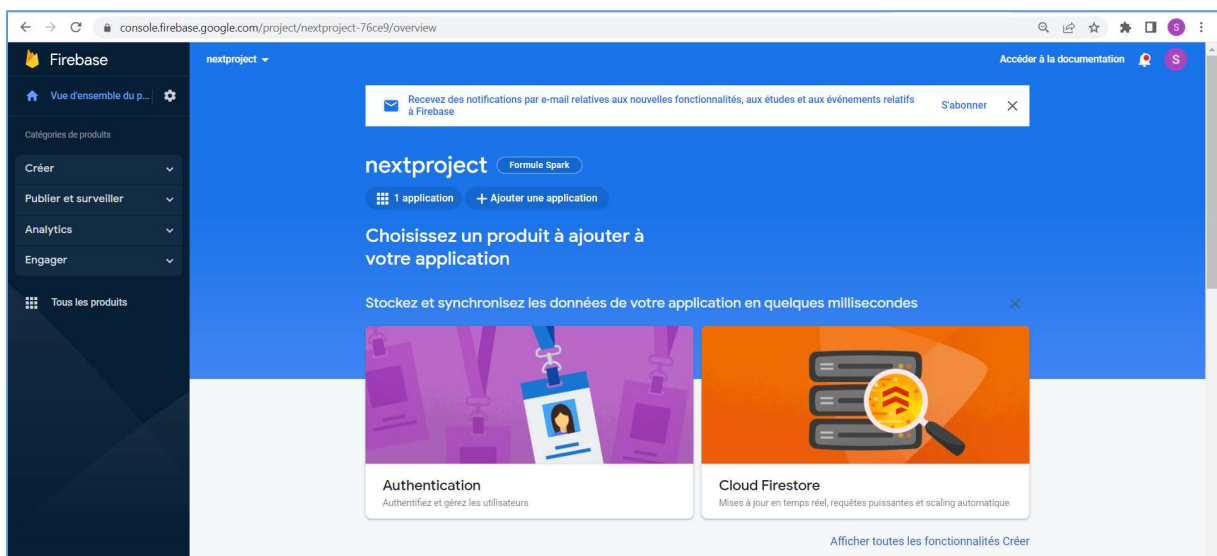
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyATRVSxM4NondpyJct3th74FY4HQqb-B2I",
  authDomain: "nextproject-76ce9.firebaseio.com",
  projectId: "nextproject-76ce9",
  storageBucket: "nextproject-76ce9.appspot.com",
  messagingSenderId: "474273333859",
  appId: "1:474273333859:web:ae5786efe351cb8ca66e56"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

Cliquer sur

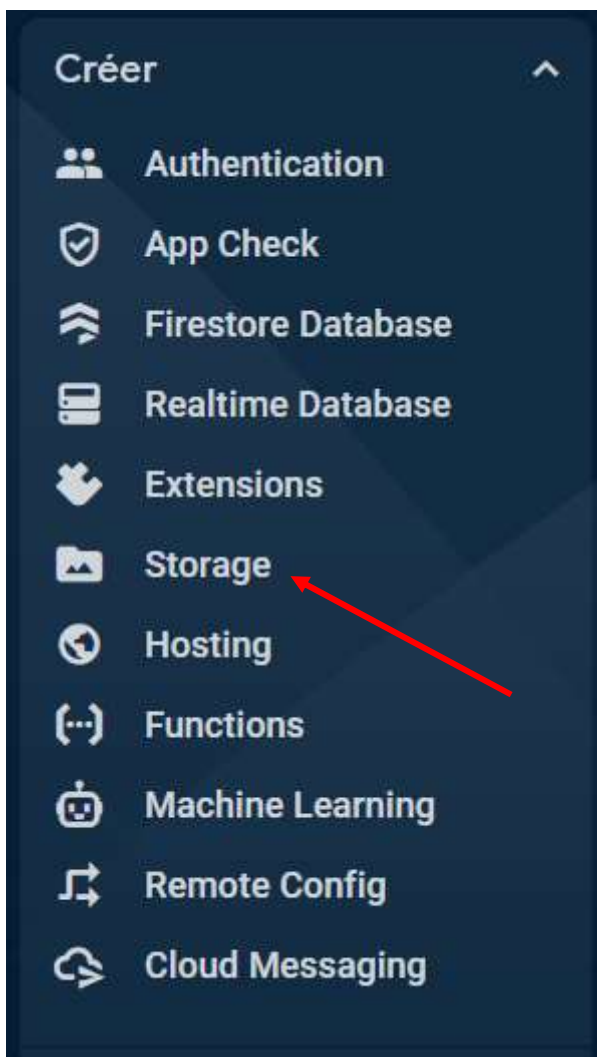
[Accéder à la console](#)



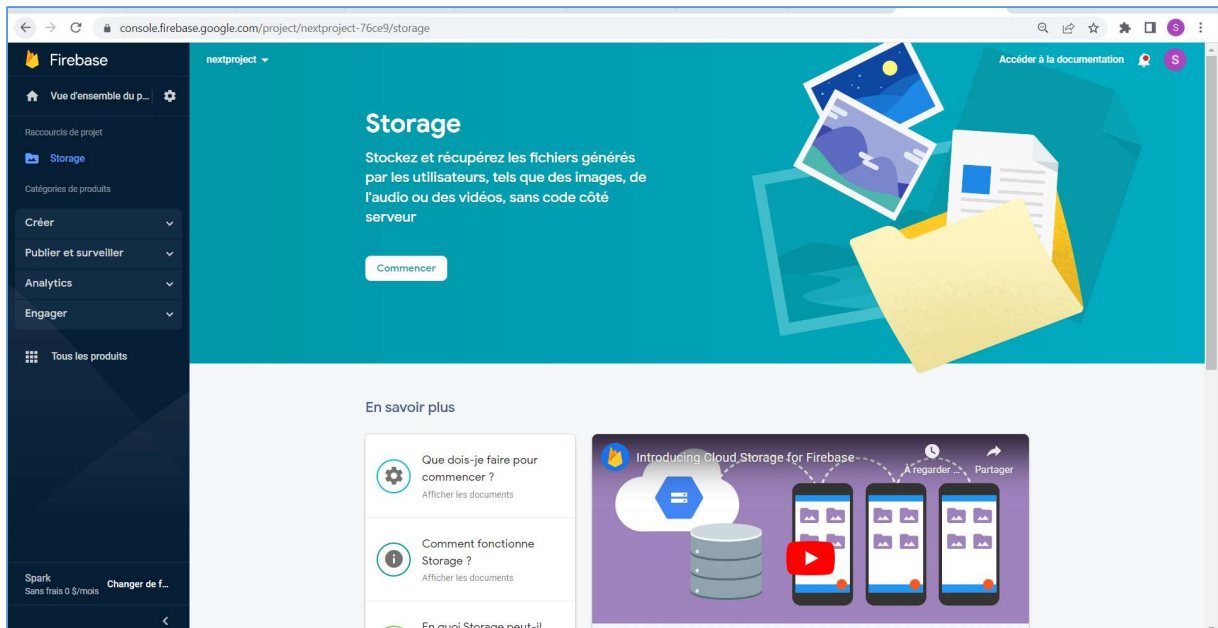
## Créer un bucket de stockage cloud

Firebase stocke les fichiers dans un bucket de stockage cloud. Suivez les étapes suivantes pour le créer :

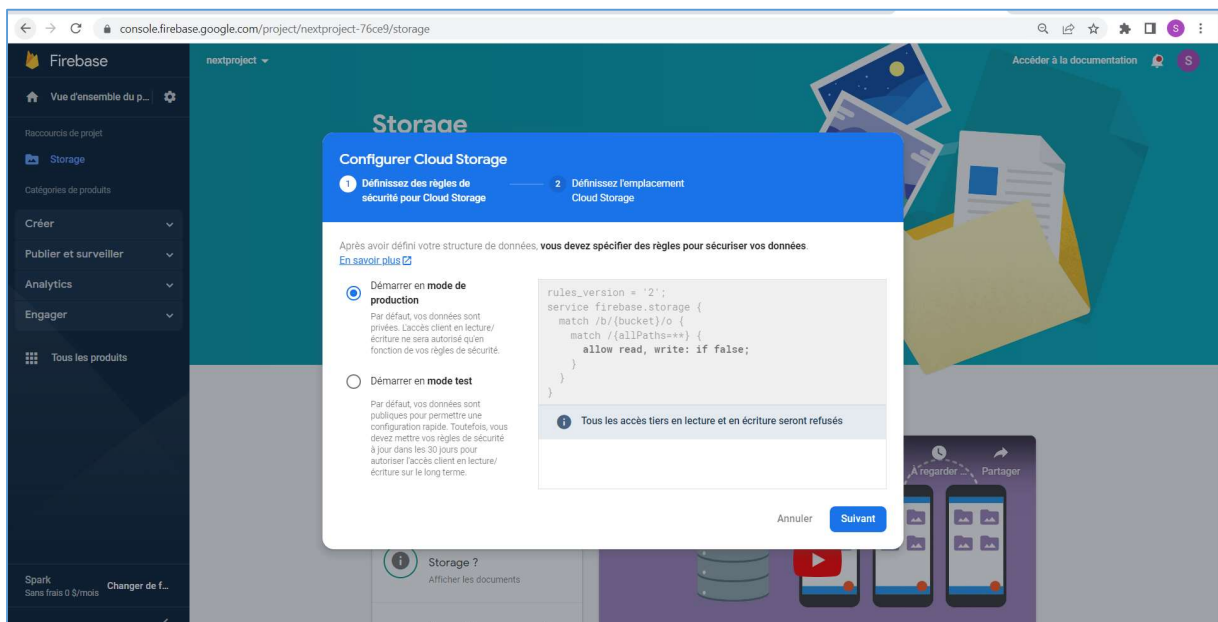
Sur la page de présentation du projet, cliquez sur l'onglet Storage dans le panneau de navigation de gauche.







Cliquer sur



Choisir démarrer en mode test puis cliquer sur suivant

## Configurer Cloud Storage

1 Définissez des règles de sécurité pour Cloud Storage

2 Définissez l'emplacement Cloud Storage

Après avoir défini votre structure de données, vous devez spécifier des règles pour sécuriser vos données.

[En savoir plus](#)

☐ Démarrer en mode de production

Par défaut, vos données sont privées. L'accès client en lecture/écriture ne sera autorisé qu'en fonction de vos règles de sécurité.

☒ Démarrer en mode test

Par défaut, vos données sont publiques pour permettre une configuration rapide. Toutefois, vous devez mettre vos règles de sécurité à jour dans les 30 jours pour autoriser l'accès client en lecture/écriture sur le long terme.

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if
        request.time < timestamp.date(2022, 10, 28);
    }
  }
}
```

⚠ Par défaut, les règles de sécurité en mode test autorisent tout utilisateur disposant de la référence de votre bucket de stockage à afficher, modifier et supprimer toutes les données qu'il contient pendant les 30 prochains jours

Annuler

Suivant

Sélectionnez l'emplacement du compartiment de stockage par défaut et cliquez sur OK

## Configurer Cloud Storage

✓ Définissez des règles de sécurité pour Cloud Storage

2 Définissez l'emplacement Cloud Storage

Le paramètre d'emplacement définit l'endroit où sont stockés votre bucket Cloud Storage par défaut et ses données.

⚠ Une fois défini, cet emplacement ne peut plus être modifié. De plus, ce paramètre d'emplacement s'appliquera par défaut à Cloud Firestore.

[En savoir plus](#)

Emplacement Cloud Storage

nam5 (us-central)

Les clients ayant souscrit une formule Blaze peuvent choisir d'autres emplacements pour des buckets supplémentaires

Annuler


OK

## Configurer Cloud Storage

1 Définissez des règles de sécurité pour Cloud Storage

2 Définissez l'emplacement Cloud Storage

Le paramètre d'emplacement définit l'endroit où sont stockés votre bucket Cloud Storage par défaut et ses données.

 Une fois défini, cet emplacement ne peut plus être modifié. De plus, ce paramètre d'emplacement s'appliquera par défaut à Cloud Firestore.

En savoir plus

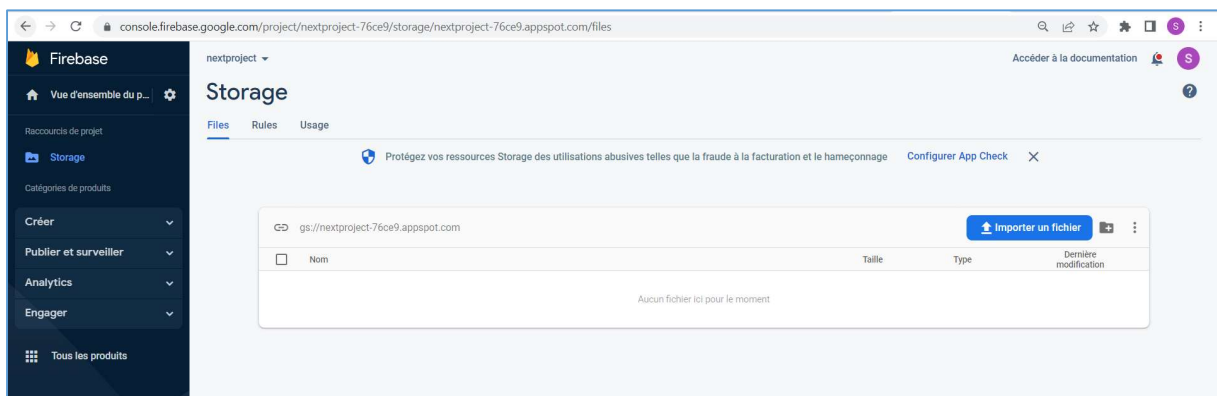
Création du bucket par défaut...

Emplacement Cloud Storage

nam5 (us-central)

Les clients ayant souscrit une formule Blaze peuvent choisir d'autres emplacements pour des buckets supplémentaires.

AnnulerOK



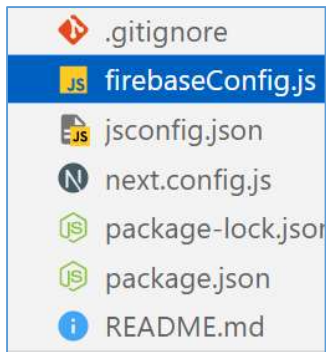
Vous êtes maintenant prêt à commencer à télécharger des fichiers sur le stockage Firebase.

## Ajouter Firebase dans Next JS

Dans votre terminal, accédez au dossier de votre projet. Exécutez la commande suivante pour installer le SDK Firebase :

```
npm install firebase --legacy-peer-deps
```

Créez un nouveau fichier appelé firebaseConfig.js dans la racine du projet.



Il faut y coller le code précédemment copié dans les étapes de création du projet dans Firebase. Puis lui ajouter l'appel de `getStorage` car on va en avoir besoin pour le stockage des images.

```
import { initializeApp } from "firebase/app";
import { getStorage } from "firebase/storage";

// Initialize Firebase
const app = initializeApp ({
  apiKey: <apiKey>,
  authDomain: <authDomain>,
  projectId: <projectId>,
  storageBucket: <storageBucket>,
  messagingSenderId: <messagingSenderId>,
  appId: <appId>,
  measurementId: <measurementId>,
});

// Firebase storage reference
const storage = getStorage(app);
export default storage;
```

Ce qui donne ce code :

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getStorage } from "firebase/storage";

// Initialize Firebase

const app = initializeApp ({
  apiKey: "AIzaSyATRVsxM4N0ndpyJct3th74FY4HQqb-B2I",
  authDomain: "nextproject-76ce9.firebaseio.com",
  projectId: "nextproject-76ce9",
  storageBucket: "nextproject-76ce9.appspot.com",
  messagingSenderId: "474273333859",
  appId: "1:474273333859:web:ae5786efe351cb8ca66e56"
});

// Firebase storage reference
const storage = getStorage(app);
export default storage;
```

La dernière ligne exporte la référence de stockage Firebase afin que vous puissiez accéder à cette instance à partir du reste de votre application.

Ensuite, sous le dossier racine créer le dossier utils. Puis il faut y créer le fichier UploadFirebase.js



La méthode Promise.resolve() « résout » une valeur donnée en une promesse. Si la valeur est une promesse, cette promesse est renvoyée.

L'opérateur await permet d'attendre la résolution d'une promesse (Promise). Il ne peut être utilisé qu'au sein d'une fonction asynchrone (définie avec l'instruction async function).

```
import storage from "../firebaseConfig";
import { ref, uploadBytesResumable, getDownloadURL } from "firebase/storage";

export const UploadFirebase=async(file)=>{
  let imageurl=""

  const storageRef = ref(storage, `/files/${file.name}`);

  const uploadTask = uploadBytesResumable(storageRef, file);

  await new Promise(resolve => {
    uploadTask.on(
      "state_changed",
      (snapshot) => {
        const percent = Math.round(
          (snapshot.bytesTransferred / snapshot.totalBytes) * 100
        );
        console.log(percent);
      },
      (err) => console.log(err),
      async () => {
        await getDownloadURL(uploadTask.snapshot.ref).then((url) => {

          imageurl=url;
          console.log(imageurl);
          resolve();
        });
      }
    );
  });

  return imageurl;
}
```

## Formulaire d'ajout

On va utiliser 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.

<https://www.npmjs.com/package/filepond>

Faire les installations suivantes :

```
npm i filepond --legacy-peer-deps
```

```
npm i react-filepond --legacy-peer-deps
```

```
npm i filepond-plugin-image-exif-orientation --legacy-peer-deps
```

```
npm i filepond-plugin-image-preview --legacy-peer-deps
```

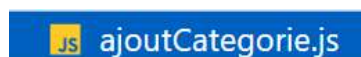
Les images seront sélectionnées puis seront uploadées dans Firebase. Ce dernier nous retournera l'url correspondantes. Pour cela il faut ajouter le site de Firebase dans le tableau domains dans le config pour que l'affichage de ces images soit autorisé.



Ajouter `firebasestorage.googleapis.com`

```
next.config.js > nextConfig > images > domains
1  /** @type {import('next').NextConfig} */
2  const nextConfig = {
3    experimental: {
4      appDir: true,
5    },
6    images: {
7      domains: ["fakestoreapi.com", "api.lorem.space", "firebasestorage.googleapis.com"],
8    },
9  }
10
11  module.exports = nextConfig
12
```

Modifier components/ajoutCategorie.js



```
"use client" ;
import React, { useState } from 'react';

import { TextField, Box, Button, Modal, Typography } from '@mui/material';

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)
```

```
import {UploadFirebase} from '../utils/UploadFirebase';
```

```
const style = {
  position: 'absolute',
  top: '50%',
  left: '50%',
  transform: 'translate(-50%, -50%)',
  width: 550,
  height: 550,
  maxHeight: 550,
  bgcolor: 'background.paper',
  border: '2px solid #000',
  boxShadow: 24,
  color: '#000',
  borderRadius: '20px',
  padding: '40px 30px 20px',
  textAlign: 'center',
};
```

```
function AjoutCat() {
```

```
  const [open, setOpen] = React.useState(false);
  const handleOpen = () => setOpen(true);
  const handleClose = () => setOpen(false);
```

```
  const [file, setFile] = useState("");
  const [name, setName] = useState("");
  const [image, setImage] = useState("");
```

```
  const handlesave=async(url)=>{
    setImage(url);
    const cat={
      name: name,
      image : url,
    };
  };
```

```
  const res = await (await
fetch('https://api.escuelajs.co/api/v1/categories', {
  method: 'POST',
  body: JSON.stringify(cat),
  headers: {
    'Content-Type': 'application/json'
```

```

    }
  })).json()
  if (res) {
    console.log('successfully inserted!')

    handleClose()
    setFile("")
    setName("");
    setImage("");
  }
  else {
    console.log(res);
  }
}

```

```

const handleUpload = (event) => {
  event.preventDefault();
  if (!file[0].file) {
    alert("Please upload an image first!");
  }
  else {
    console.log(file[0].file)
    resultHandleUpload(file[0].file,event);
  }
  if (!file[0].file) {
    alert("Please upload an image first!");
  }
};

```

```

const resultHandleUpload = async(file) => {

  try {

    const url = await UploadFirebase(file);
    console.log(url);

    handlesave(url)
  } catch (error) {
    console.log(error);
  }
}

```

```

return (
  <div>
    <Button type="button" className="btn btn-primary" onClick={handleOpen}>
      ADD
    </Button>
  </div>
)

```



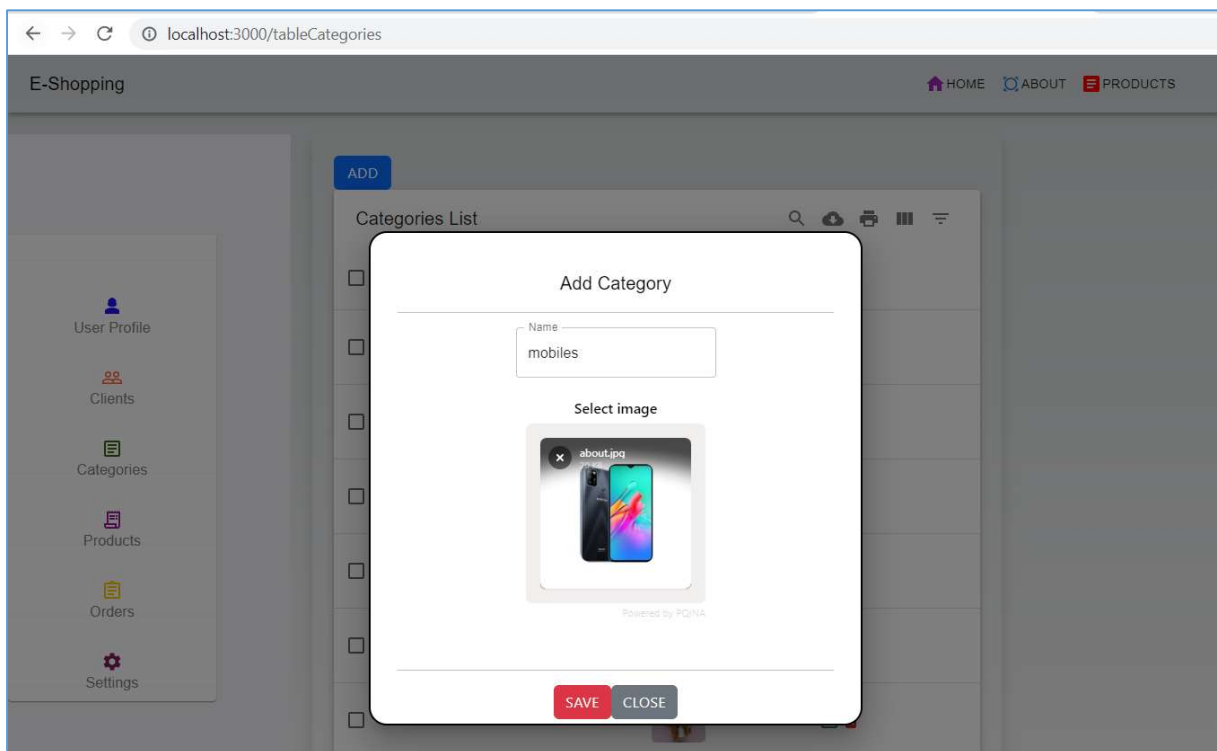
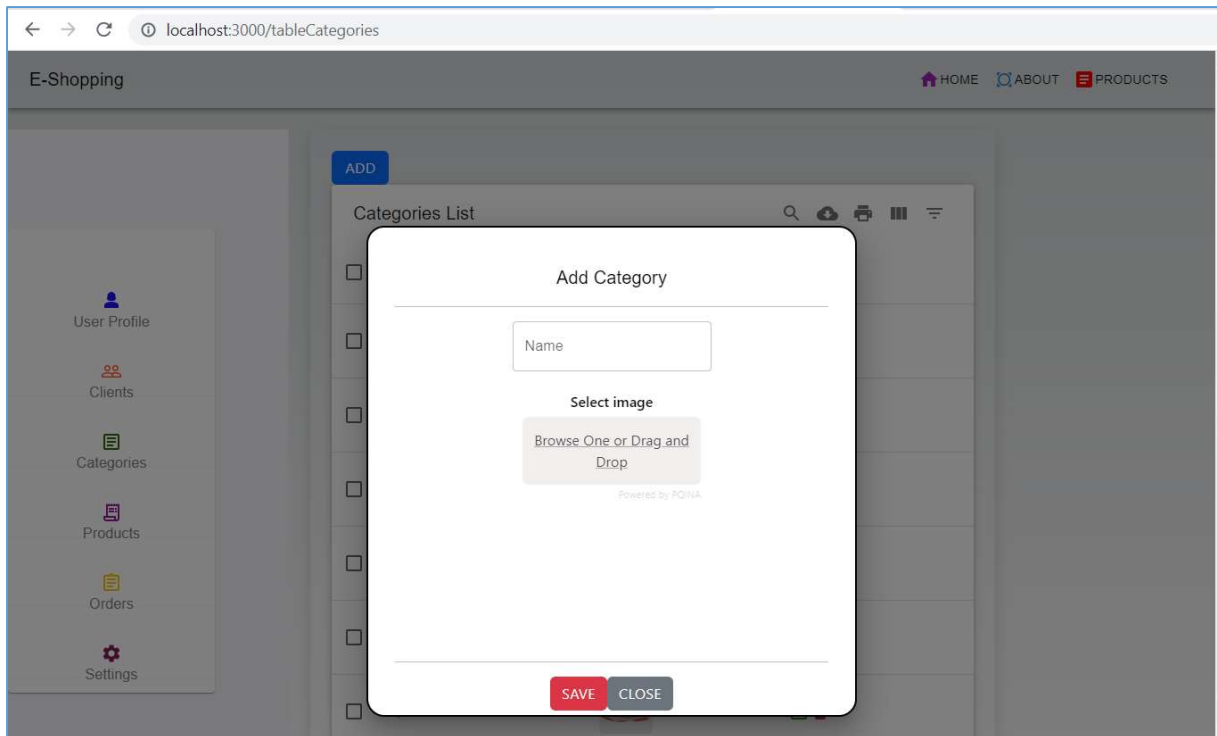
```

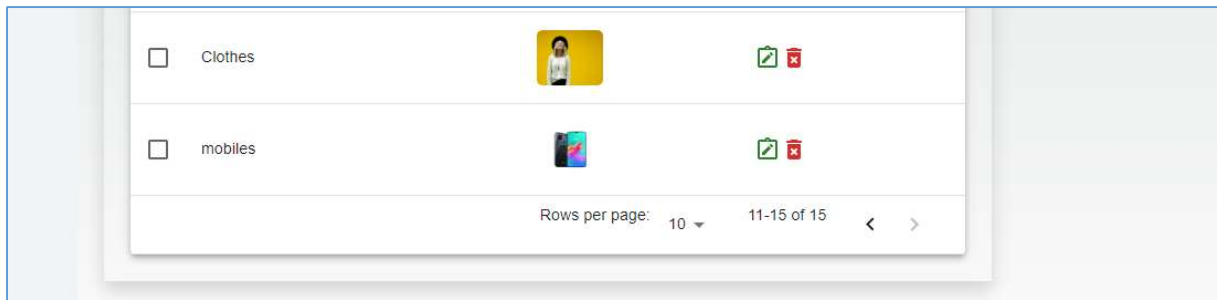
<Modal
  open={open}
  onClose={handleClose}
  aria-labelledby="modal-modal-title"
  aria-describedby="modal-modal-description"
>
  <Box sx={style}>
    <Typography id="modal-modal-title" variant="h6" component="h2">
      Add Category
    </Typography>
    <hr/>

    <div className="mb-4">
      <TextField variant="outlined"
label="Name"  onChange={e=>setName(e.target.value)} />
    </div>
    <div className="mb-4">
      <h6>Select image</h6>
      <center>
        <div style={{width:200, height:250}}>
          <FilePond
            files={file}
            allowMultiple={false}
            onupdatefiles={setFile}
            labelIdle='<span class="filepond--label-action">Browse
One</span>'
          />
        </div>
      </center>
    </div>
    <div>
      <div className="mb-3">
        <Button type="button" className="btn btn-danger"
onClick={{(event)=>handleUpload(event)}}>Save</Button>
        <Button type="button" className="btn btn-secondary"
onClick={handleClose}>Close</Button>
      </div>
    </div>
  </Box>
</Modal>
</div>
)
}

export default AjoutCat

```





## Formulaire de modification

### JS updateCategory.js

```
"use client" ;
import React,{ useState,useEffect } from 'react';

import {TextField, Box, Button, Modal, Typography } from '@mui/material';

import NoteAltOutlinedIcon from '@mui/icons-material/NoteAltOutlined';

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)

import {UploadFirebase} from '../utils/UploadFirebase';

const style = {
  position: 'absolute',
  top: '50%',
  left: '50%',
  transform: 'translate(-50%, -50%)',
  width: 550,
  height: 650,
  maxHeight: 650,
  bgcolor: 'background.paper',
  border: '2px solid #000',
  boxShadow: 24,
  color: '#000',
  borderRadius: '20px',
  padding: '40px 30px 20px',
  textAlign: 'center',
};
```

```

function updateCategory(props) {

  const [open, setOpen] = useState(false);
  const handleOpen = () => setOpen(true);
  const handleClose = () => setOpen(false);

  const [id, setId]=useState();
  const [file, setFile] = useState();
  const [name, setName] = useState();
  const [image, setImage] = useState();

  useEffect(() => {
    setId(props.categories.id);
    setName(props.categories.name);
    setImage(props.categories.image);
  }, []);

  const handlesave=async(url)=>{

    setImage(url);
    const cat={
      id: id,
      name: name,
      image : url,
    };

    const res = await (await
fetch('https://api.escuelajs.co/api/v1/categories/' + id, {
  method: 'PUT',
  body: JSON.stringify(cat),
  headers: {
    'Content-Type': 'application/json'
  }
})).json()
    if (res) {
      console.log('successfully updated!')

      handleClose()
    }
    else {
      console.log(res);
    }
  }

  const handleUpload = (event) => {
    event.preventDefault();
  }
}

```

```

    if (!file) {
      const url = image;
      handlesave(url);
    }
    else {
      console.log(file[0].file)
      resultHandleUpload(file[0].file);
    }
  };

const resultHandleUpload = async(file) => {

  try {

    const url = await UploadFirebase(file);
    console.log(url);

    handlesave(url)
  } catch (error) {
    console.log(error);
  }
}

return (
  <>
    <span onClick={handleOpen}
      style={{ cursor: 'pointer'}}>
      <NoteAltOutlinedIcon color='success' />
    </span>

    <Modal
      open={open}
      onClose={handleClose}
      aria-labelledby="modal-modal-title"
      aria-describedby="modal-modal-description"
    >
      <Box sx={style}>
        <Typography id="modal-modal-title" variant="h6" component="h2">
          Update Category
        </Typography>
        <hr/>

        <div className="mb-4">
          <TextField variant="outlined" label="Name"
value={name} onChange={e=>setName(e.target.value)} />
        </div>
        <div className="mb-4">

```

```

        {!file?<img src={image} style={{width:50, height:50}}/> :null}
        <h6>Selet new image</h6>
    <center>
        <div style={{width:200, height:250}}>
            <FilePond
                files={file}
                allowMultiple={false}
                onupdatefiles={setFile}
                labelIdle='<span class="filepond--label-action">Browse
One</span>'
            />
        </div>
    </center>
</div>
<hr/>
    <div className="mb-3">
        <Button type="button" className="btn btn-success"
onClick={(event)=>handleUpload(event)}>Update</Button>
        <Button type="button" className="btn btn-secondary"
onClick={handleClose}>Close</Button>
    </div>

</Box>
</Modal>
</>
)
}

export default updateCategory

```

