Authentification Firebase

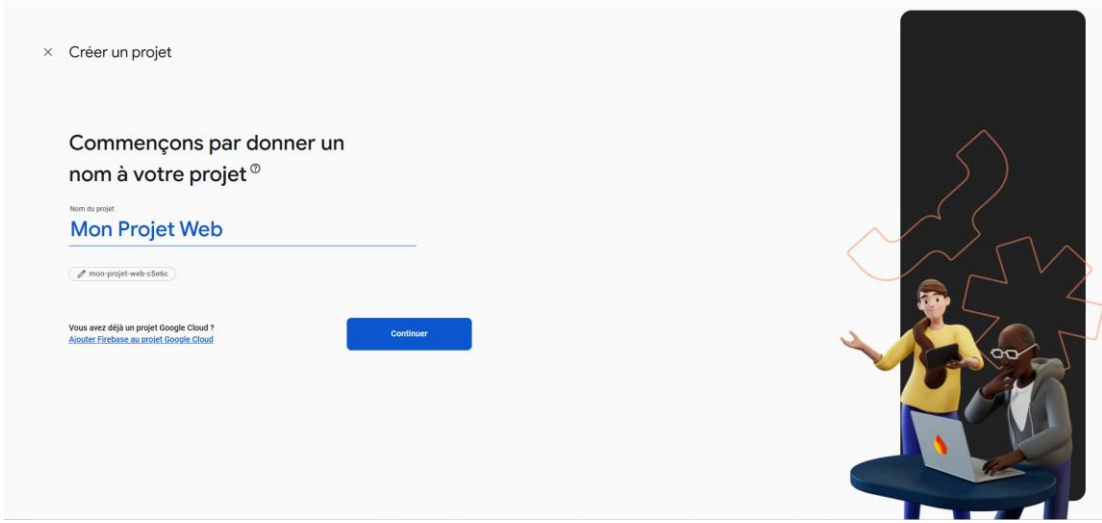
Présentation de Firebase

- Firebase est une plateforme de développement développée par Google offrant différents outils pour le développement web et d'applications mobiles
- Plusieurs outils et services sont disponibles:
  - Base de données
  - Authentification
  - Hébergement
  - Cloud
  - Notification et messagerie
  - Stockage
  - Etc.

Utilisation de Firebase

Créer un projet

- Avant toute chose, il faut avoir un compte sur Firebase
  - Rendez vous sur https://firebase.google.com/
  - On peut se connecter avec un compte Google
- Aller à la console et créer un projet
  - Le projet regroupera toutes les applications de différentes plates-formes
  - Cela peut être une application mobile, web, unity, etc.
- Suivre les étapes pour finaliser la création du projet

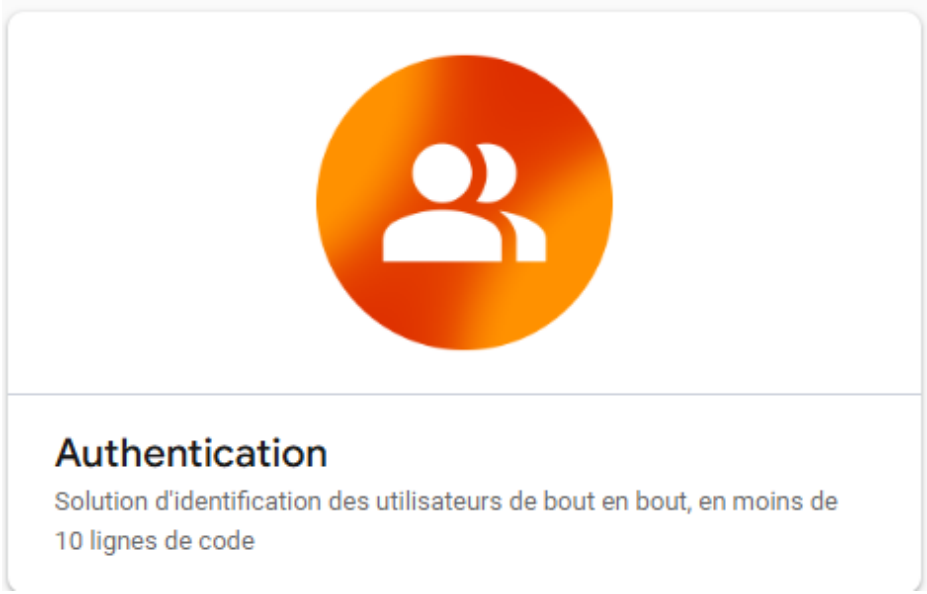


Ajouter une application au projet

- On peut maintenant enregistrer des applications à notre projet
- Ces applications sont les produits qui nous développons sur une plate-forme
- Lorsqu'on développe une application avec VueJs ou React, nous allons choisir une application web
- Une fois l'application enregistrée, vous aurez toutes les informations concernant celle-ci afin de pouvoir intégrer Firebase

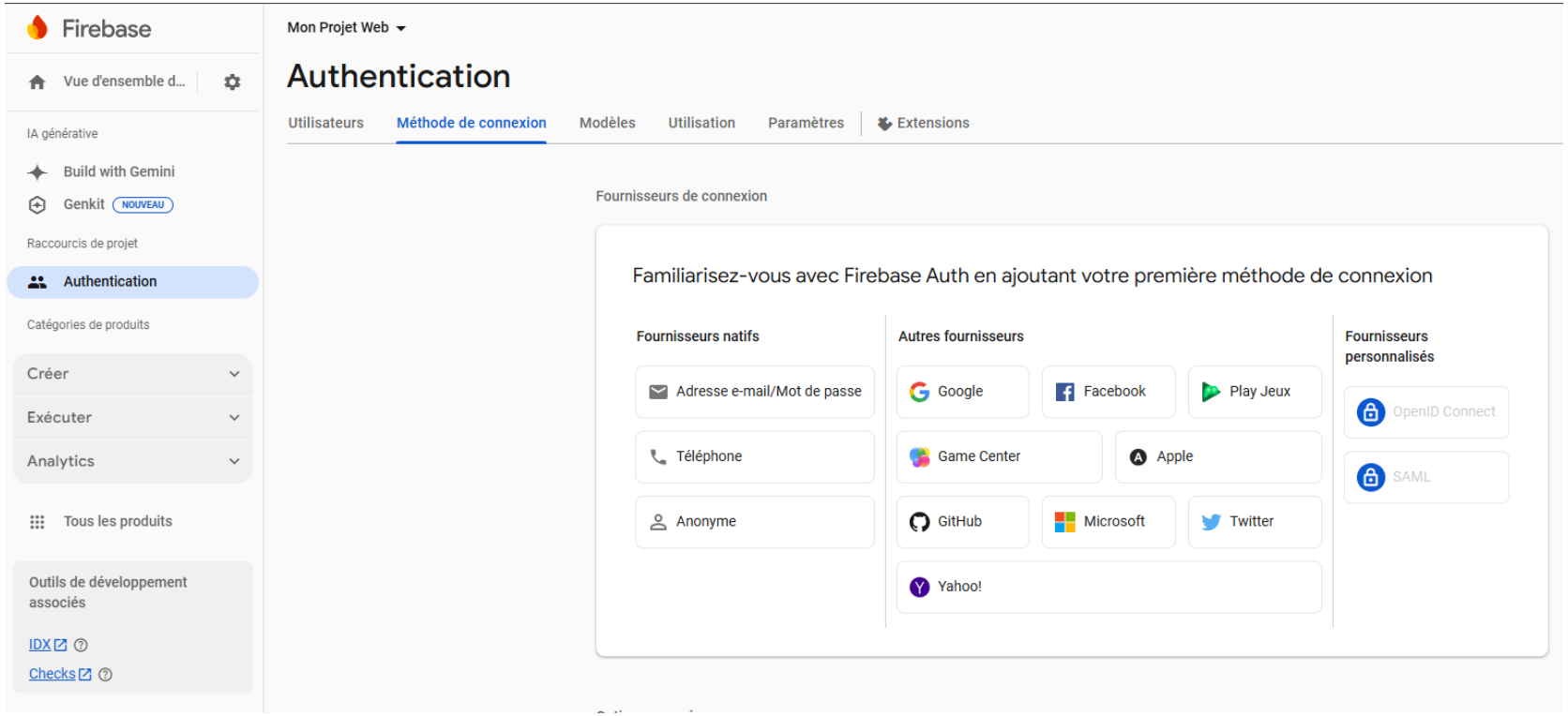
Ajout de l'authentification

- Dans la console Firebase, vous allez pouvoir ajouter le service d'authentification
- Ce produit fournit plusieurs manières aux personnes utilisant votre produit de s'identifier
- Il y a par exemple:
  - L'authentification via des fournisseurs (Google, Facebook, Github, etc.)
  - L'authentification via email et mot de passe, par téléphone ou encore de façon anonyme

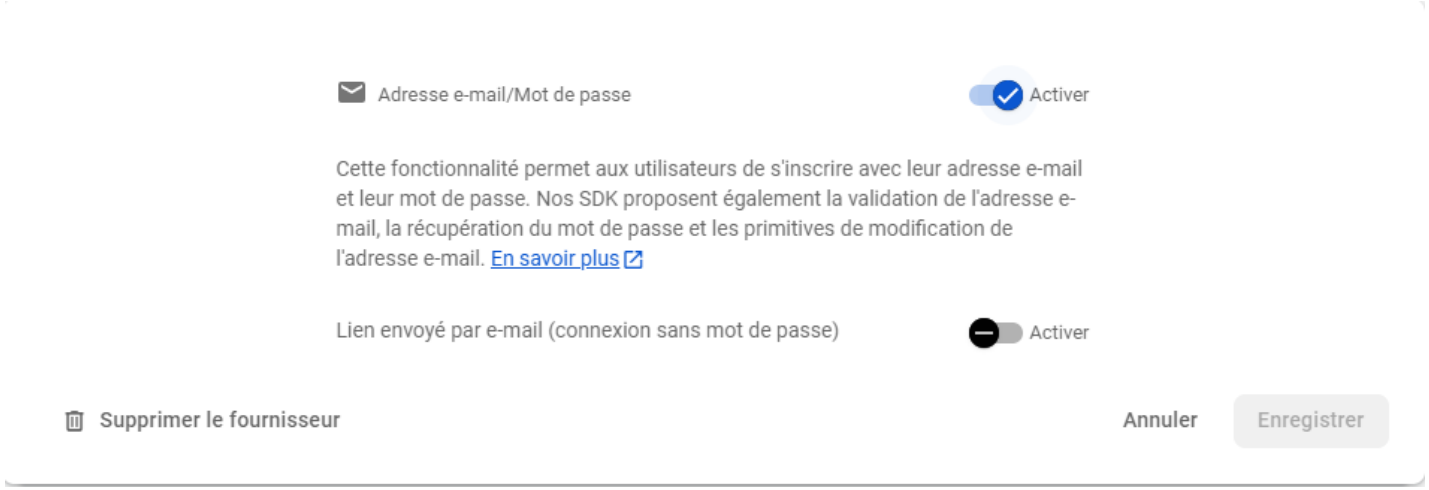


Sélectionner la méthode de connexion

- Après avoir ajouté l'authentification, il faut maintenant choisir la méthode de connexion
  - Cliquer sur l'option "Authentification" à gauche dans le menu
  - Choisir l'onglet "Méthode de connexion"



Exemples



Connexion par email et mot de passe

"Adresse e-mail/Mot de passe"

Inclure Firebase dans notre projet VueJs/React

- Il faut commencer par installer Firebase

```
npm install firebase
```

- Créez ensuite un fichier javascript dans lequel vous allez mettre toute la configuration de votre application web fournie lors de son enregistrement dans Firebase
  - Dans mon exemple, je vais nommer le fichier "firebase.js"
- Les informations de configuration se trouvent dans les paramètres généraux du projet
  - Il faudra choisir l'application désirée

Inclure Firebase dans notre projet VueJs/React

firebase.js

```
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";

const firebaseConfig = {
  apiKey: "VOTRE_API_KEY",
  authDomain: "VOTRE_AUTH_DOMAIN",
  projectId: "VOTRE_PROJECT_ID",
  storageBucket: "VOTRE_STORAGE_BUCKET",
  messagingSenderId: "VOTRE_MESSAGING_SENDER_ID",
  appId: "VOTRE_APP_ID",
  measurementId: "VOTRE_MEASUREMENT_ID",
};

//initialisation de l'application firebase
const app = initializeApp(firebaseConfig);

// ici on va chercher le service d'authentification
const auth = getAuth(app);

export { auth };
```

Créer un nouveau compte d'utilisateur

- Dans notre site web, nous allons avoir le formulaire de création de compte
- Ce formulaire devoir avoir au moins les champs d'email et de mot de passe
- Afin de pouvoir créer le compte, nous allons devoir utiliser la fonction "**createUserWithEmailAndPassword**" qu'on importera
- Cette fonction prend en paramètre:
  - L'objet auth créé dans firebase.js
  - L'email
  - Le mot de passe
- Quand la création de compte fonctionne, Firebase retourne les informations du nouveau compte, y compris l'id de l'utilisateur.
  - On peut les sauvegarder toutes ces informations
- Dans l'exemple qui suit, nous avons un composant ayant le formulaire

Créer un nouveau compte d'utilisateur

```vue
<template>
  <div>
    <form @submit="creerUtilisateur">
      <input
        type="email"
        v-model="email"
        placeholder="Entrez l'email"
      />
      <br />
      <input
        type="password"
        v-model="pwd"
        placeholder="Entrez le mot de passe"
      />
      <br />
    <button type="submit">Créer un compte</button>
    </form>
  </div>
</template>
```

```vue
<script setup>
import { ref } from "vue";
import { auth } from "../firebase";
import { createUserWithEmailAndPassword } from "firebase/auth";

const utilisateur = ref({});
const email = ref("");
const pwd = ref("");

const creerUtilisateur = async (e) => {

  e.preventDefault();
  try {
    const result = await createUserWithEmailAndPassword(auth, email.value, pwd.value);
    utilisateur.value = result.user;
    console.log(utilisateur.value);
  } catch (error) {
    console.error("Échec de la création de l'utilisateur", error);
  }
};
</script>
```

Connexion avec un compte existant

- Une fois que le compte est créé, on peut se connecter avec les informations entrées
- On devra pour la connexion utiliser la fonction "**signInWithEmailAndPassword**"
- Ici également, on fournira comme paramètres l'objet Auth, l'email et le mot de passe
- On aura en retour les informations de l'utilisateur comme lorsqu'on crée un nouveau profil

Connexion avec un compte existant

```vue
<template>
  <div>
    <form @submit="connexionEmailPassword">
      <input type="email" v-model="email"
            placeholder="Entrez l'email" />
      <br />
      <input
        type="password" v-model="pwd"
        placeholder="Entrez le mot de passe"
      />
      <br />
      <button type="submit">Se connecter</button>
    </form>

    <div v-if="connecte">
        <h3>Vous êtes connecté </h3>
    </div>

  </div>
</template>
```

```vue
<script setup>
import { ref } from "vue";
import { auth } from "../firebase";
import { createUserWithEmailAndPassword, signInWithEmailAndPassword} from
"firebase/auth";
const utilisateur = ref({});
const connecte = ref(false);
const email = ref("");
const pwd = ref("");

const connexionEmailPassword = async (e) => {
  e.preventDefault();
  console.log(email.value, pwd.value);
  try {
    const result = await signInWithEmailAndPassword(auth, email.value, pwd.value);
    utilisateur.value = result.user;
    connecte.value = true;
    console.log(utilisateur.value);
  }
  catch (error) {
    console.error("Échec de la connexion", error);
  }
};

</script>
```

Déconnexion

- La déconnexion se fera tout simplement avec la fonction "**signOut**"
- Elle prend en paramètre juste l'objet Auth
- Cette fonction ne retourne rien

Déconnexion

```vue
<template>
  <div>
      <h3>Vous êtes connecté </h3>

      <button @click="deconnexion">
         Se déconnecter
      </button>
  </div>
</template>
```

```vue
<script setup>
import { ref } from "vue";
import { auth } from "../firebase";
import { signInWithEmailAndPassword, signOut} from "firebase/auth";

const utilisateur = ref({});
const connecte = ref(false);
const email = ref("");
const pwd = ref("");

const connexionEmailPassword = async (e) => {
  e.preventDefault();
    try {
    const result = await signInWithEmailAndPassword(auth, email.value, pwd.value);
    utilisateur.value = result.user;
    connecte.value = true;
    console.log(utilisateur.value);     console.log(utilisateur.value);
  } catch (error) {
    console.error("Échec de la connexion", error);
  }
};

const deconnexion = async () => {
  try {
    const result = await signOut(auth);
    utilisateur.value = null;
    connecte.value = false;
    console.log("Déconnexion réussie", result);
  } catch (error) {
    console.error("Erreur lors de la déconnexion:", error);
  }
};

</script>
```

Mise à jour du nom affiché et de la photo de profil

- Nous avons la possibilité de mettre à jour les informations de base du profil
  - Le nom de la personne affiché
  - La photo de profil
- Cette mise à jour peut se faire dès que la personne crée son profile par exemple ou lorsque la personne se connecte
- Les informations mises à jour resteront même pour les prochaines connexions
- On utilisera la fonction "**updateProfile**" qui prend en paramètres:
  - L'utilisateur connecté
  - Un objet avec les clés "displayName" et "photoURL" et leurs valeurs
  - On peut indiquer l'une ou l'autre des clés

Mise à jour du nom affiché et de la photo de profil

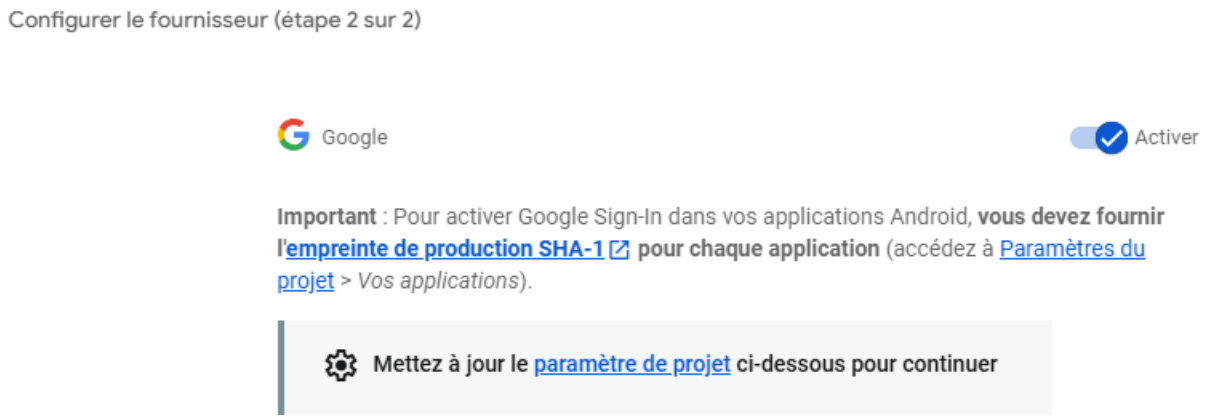
```vue
<template>
  <div>
    <form @submit="creerUtilisateur">
      <input type="text" v-model="nom" placeholder="Entrez le nom" /> <br />
      <input type="email" v-model="email" placeholder="Entrez l'email" />
      <br />
      <input
        type="password"
        v-model="pwd"
        placeholder="Entrez le mot de passe"
      />
      <br />
      <button type="submit">Créer un compte</button>
    </form>
    <div v-if="connecte">
      <h3>
        <img :src="utilisateur.photoURL" width="100" />
        Vous êtes connecté {{ utilisateur.displayName }}
      </h3>
      <button @click="deconnexion">Se déconnecter</button>
    </div>
  </div>
</template>
```

```vue
<script setup>
import { ref } from "vue";
import { auth } from "../firebase";
import {createUserWithEmailAndPassword, updateProfile,} from "firebase/auth";

const utilisateur = ref({});
const connecte = ref(false);
const nom = ref("");
const email = ref("");
const pwd = ref("");

const creerUtilisateur = async (e) => {
  e.preventDefault();
   try {
    const result = await createUserWithEmailAndPassword(
      auth,
      email.value,
      pwd.value
    );
    utilisateur.value = result.user;
    connecte.value = true;
    console.log(utilisateur.value);
    await updateProfile(utilisateur.value, {
      displayName: nom.value,
      photoURL: "https://randomuser.me/api/portraits/men/22.jpg",
    });
    console.log(utilisateur.value);
  } catch (error) {
    console.error("Échec de la connexion", error);
  }
};

</script>
```



Connexion par Google

"Google"

Mise en place

- Il faudra modifier un peu le fichier "firebase.js" afin qu'il retourne également le fournisseur en plus de l'objet Auth
- Dans le composant de la connexion, on aura un bouton qui permettra de se connecter avec un compte Google
- On aura une fenêtre pop-up qui s'ouvrira et qui permettra d'indiquer le compte Google à utiliser
- Pour faire la connexion, on utilisera la fonction "**signInWithPopup**"
- Cette fonction prend en paramètres
  - L'objet Auth
  - Le fournisseur (dans ce cas-ci ce sera Google)
- La fonction va retourner l'objet représentant l'utilisateur.
- Ici, les informations sont celles du compte utilisé pour se connecté, donc on a déjà le nom, et la photo de profil

firebase.js

```
import { initializeApp } from "firebase/app";
import { getAuth, GoogleAuthProvider } from "firebase/auth";

const firebaseConfig = {
  apiKey: "VOTRE_API_KEY",
  authDomain: "VOTRE_AUTH_DOMAIN",
  projectId: "VOTRE_PROJECT_ID",
  storageBucket: "VOTRE_STORAGE_BUCKET",
  messagingSenderId: "VOTRE_MESSAGING_SENDER_ID",
  appId: "VOTRE_APP_ID",
  measurementId: "VOTRE_MEASUREMENT_ID",
};

const app = initializeApp(firebaseConfig);

// ici on va chercher le service d'authentification
const auth = getAuth(app);

const provider = new GoogleAuthProvider();

export { auth, provider };
```

Composant de connexion

```vue
<template>
  <div>
    <button @click="creerUtilisateur">
            Se connecter avec Google
    </button>
    <div v-if="connecte">
      <h1>Bienvenue {{ utilisateur.displayName }}</h1>
    </div>
  </div>
</template>
```

```vue
<script setup>
import { ref } from "vue";
import { auth, provider } from "../firebase";
import { signInWithPopup} from "firebase/auth";

const utilisateur = ref({});
const connecte = ref(false);

const creerUtilisateur = async () => {
  try {
    const result = await signInWithPopup(auth, provider);
    utilisateur.value = result.user;
    connecte.value = true;
    console.log(utilisateur.value);
  }
  catch (error) {
    console.error("Échec de la création de compte:", error);
  }
};

</script>
```