

JS & Firebase

Nous allons utiliser Javascript avec une base de donnée automatiquement hébergée en ligne, Firebase, il s'agit d'une solution proposée par google contenu plein d'outils dont des systèmes de base de données, au même titre que AWS (Amazon Web Service) Microsoft Azure, MongoDB ou encore le concurrent direct et open-source, Supabase.

Il s'agit d'un BAAS (Backend As A Service), c'est à dire que plutôt que d'avoir à créer et gérer nous même la partie Backend de notre application nous utiliserons celui que Google (ou autre) nous met à disposition.

Dans Firebase, il y a donc plusieurs outils mais pour l'exemple nous allons utiliser la REALTIME DATABASE (une base de données en temps réel) qui est en NoSQL (Not Only SQL), en l'occurrence nous allons travailler sur une BDD organisée en JSON.

Plus d'infos ici pour comparer avec MySQL :

<https://blog.back4app.com/fr/firebase-database-contre-mysql/>

	Firebase	MySQL
Brève description	Plateforme de développement d'applications de Google.	Base de données SQL open-source
Prix	Démarrage gratuit Payez au fur et à mesure par la suite	Téléchargement gratuit
Open-Source	Non	Oui
Verrouillage du fournisseur	Oui	Non
Service géré	Oui	Hébergement géré disponible sur Oracle, AWS, Digital Ocean, etc.
Traitement des données	Firebase traite efficacement les grands ensembles de données. Il utilise des magasins à colonnes larges, à valeurs clés, à documents ou à graphes et dispose de schémas dynamiques pour faciliter les données non structurées.	MySQL est basé sur des tables et possède des schémas prédéfinis. C'est un choix privilégié pour le traitement de données complexes.
Architecture	Firebase est une base de données NoSQL qui synchronise et stocke les données en temps réel.	MySQL est un système de gestion de base de données relationnelle open source qui repose sur SQL, le langage spécifique au domaine.
Assistance linguistique	Beaucoup moins que MySQL.	Les langages de programmation pris en charge par MySQL sont bien plus nombreux que ceux pris en charge par Firebase. Ces langages comprennent Python, C++, Ada, etc.
Évolutivité	Firebase évolue horizontalement.	MySQL évolue verticalement.

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

Date révision :

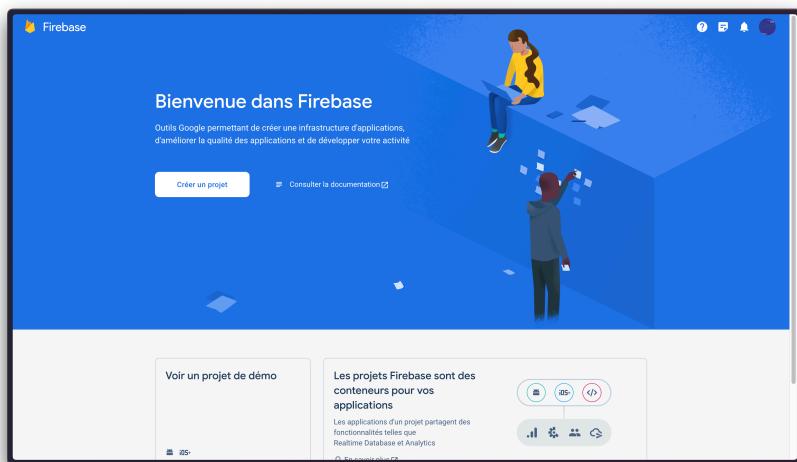
10/03/2023



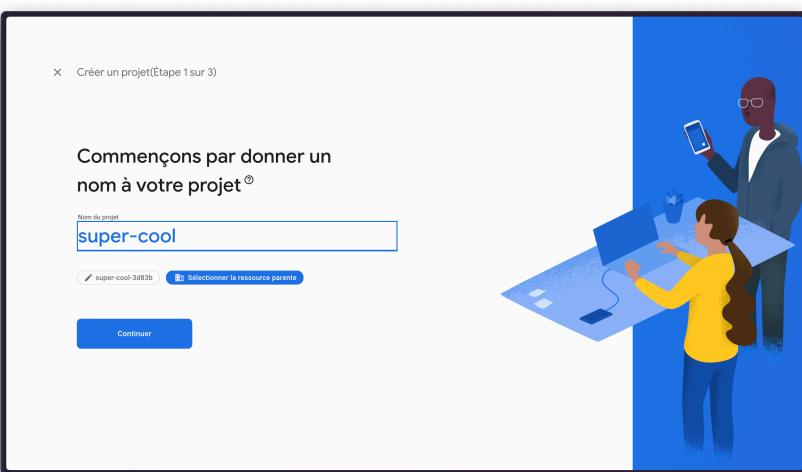
Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Set-up de firebase

Se rendre sur le site de la console Firebase
<https://console.firebaseio.google.com>



Ensuite on crée un projet dans Firebase :



Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

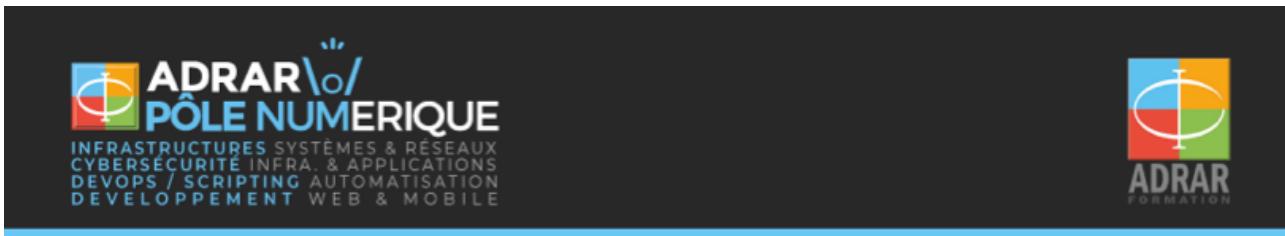
Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

Date révision :

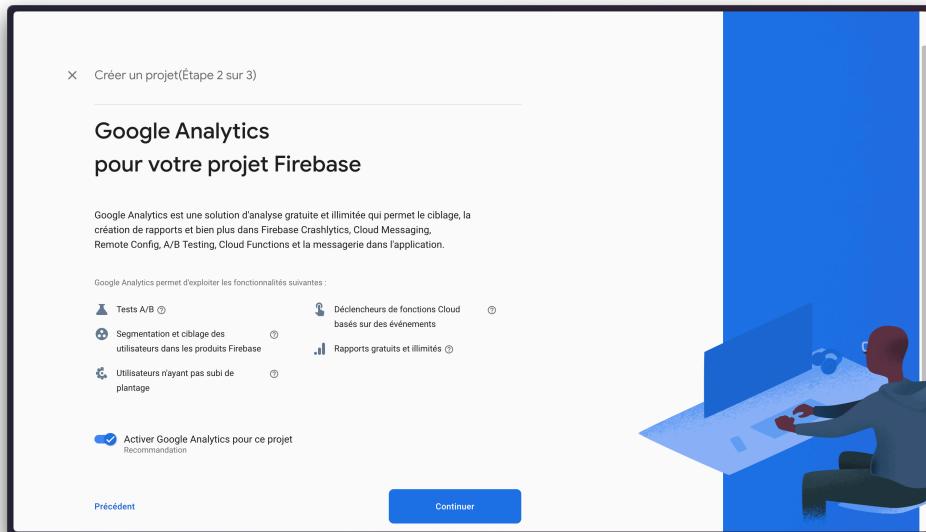
10/03/2023



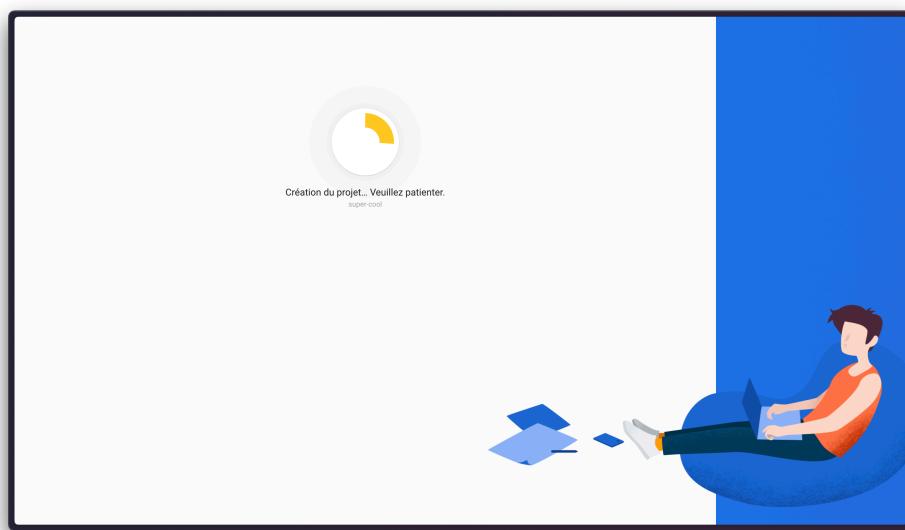
Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

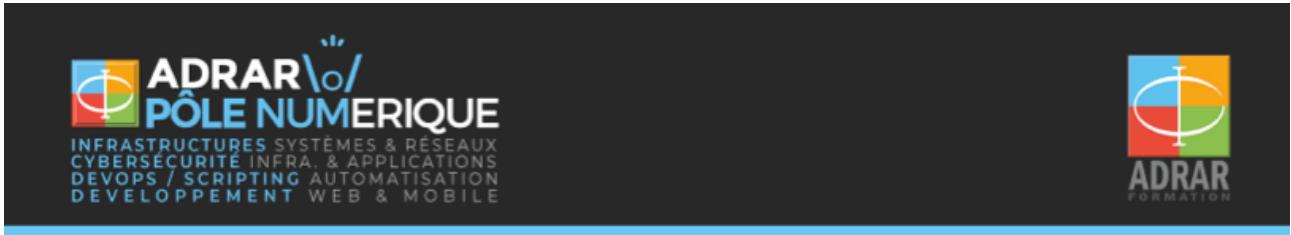


Optionnel : on peut activer Google Analytics, pour ce projet test on peut le désactiver



Ensuite Firebase crée le projet





On atterri alors sur la page d'accueil de notre projet ici « super-cool »

A screenshot of the Firebase console interface. On the left, a sidebar shows navigation options like "Créer", "Publier et surveiller", "Analytics", and "Engager". The main area displays a landing page for a project named "super-cool". The page features a large blue banner with the text "Lancez-vous en ajoutant Firebase à votre application" and icons for iOS, Android, and web development. Below the banner, there's a callout about real-time synchronization and two small images related to user authentication and database queries.

Dans l'onglet à gauche on va créer une nouvelle base de données de type **RealTime Database**

A screenshot of the Firebase console interface. The left sidebar has "Realtime Database" selected under the "Créer" category. The main area shows the same "super-cool" project landing page as the previous screenshot, with the "Realtime Database" option highlighted in the sidebar.

A footer section containing several pieces of information: "Auteur:" followed by "Jean-François Pech"; "Date création:" followed by "03/03/2023"; "Relu, validé & visé par:" followed by three names with checkboxes; "Date révision:" followed by "10/03/2023"; and a copyright notice from "ADRAR PÔLE NUMÉRIQUE TECH' CARE" stating that reproduction is prohibited without permission. The footer also includes the page number "136".



Créer une RealTime Database :

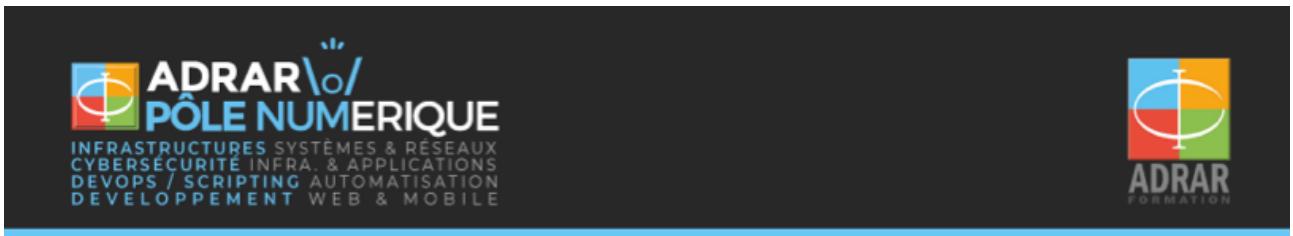
The screenshot shows the Firebase Realtime Database setup process. It starts with the main Firebase dashboard, then moves to the 'Realtime Database' configuration screen. On this screen, the location 'Etats-Unis (us-central1)' is selected. Below the configuration, there is a note about security rules in 'mode test'.

Ensuite choisir « démarrer en mode test »

The screenshot shows the 'Configure a database' step, specifically the 'Security Rules' section. The 'Mode test' option is selected. A note states: 'Par défaut, les règles de sécurité en mode test autorisent tout utilisateur disposant de la référence de votre base de données à afficher, modifier et supprimer toutes les données qu'elle contient pendant les 30 prochains jours.'

Auteur : Jean-François Pech	Date création : 03/03/2023	Relu, validé & visé par : Jérôme CHRETIENNE Sophie POULAKOS Mathieu PARIS
Date révision : 10/03/2023		 ADRAR FORMATION

Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.



Nous arrivons sur notre base de donnée
(Firebase nous a créée la base de donnée et elle est hébergée donc accessible par l'url (on pourrait aussi contacter la base via les api))

Firebase

Vue d'ensemble du p... |

Raccourcis de projet

Realtime Database

Catégories de produits

Créer

Publier et surveiller

Analytics

Engager

Tous les produits

Spark Sans frais 0 \$/mois Mettre à niveau

super-cool ▾

Realtime Database

Données Règles Sauvegardes Utilisation NOUVEAU

Protégez vos ressources Realtime Database des utilisations abusives telles que la fraude à la facturation et le hameçonnage Configurer App Check X

https://super-cool-3d83b-default.firebaseio.com/.json

Emplacement de la base de données : États-Unis (us-central1)

Ensuite on va se rendre dans les paramètres du projet afin d'enregistrer une application pour cette RealTime Database

Firebase

Vue d'ensemble du p... |

Raccourcis de projet

Realtime Database

Catégories de produits

super-cool ▾

Realtime Database

Paramètres du projet

Utilisateurs et autorisations

Utilisation et facturation

Protégez vos ressources Realtime Database

Auteur :
Jean-François Pech

Date création :
03/03/2023

Relu, validé & visé par :
Jérôme CHRETIENNE
Sophie POULAKOS
Mathieu PARIS

Date révision :
10/03/2023

Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Dans les paramètres généraux, en scrollant tout en bas :
 On va ajouter une application web donc l'icône « < / > »

Vos applications

Votre projet ne comporte aucune application

Pour démarrer, sélectionnez une plate-forme



 |
 


On renseigne un nom pour notre application web (pas besoin de firebase Hosting ça sera peut être pour plus tard si on veut déployer l'application chez Firebase):

x Ajouter Firebase à votre application Web

1 Enregistrer l'application

Pseudo de l'application [?](#)

Configurez également **Firebase Hosting** pour cette application. [En savoir plus](#)

Hosting peut également être configuré plus tard, et vous pouvez commencer à l'utiliser sans frais à tout moment.

[Enregistrer l'application](#)

2 Ajouter le SDK Firebase




Ensuite Firebase nous fournit des codes d'accès pour pouvoir contacter notre RealTime Database, on va au moins récupérer la variable firebaseConfig qu'on placera dans notre programme JS

```
// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AI[REDACTED]W",
  authDomain: "[REDACTED].com",
  databaseURL: "https://[REDACTED].firebaseapp.com",
  projectId: "s[REDACTED]",
  storageBucket: "s[REDACTED].appspot.com",
  messagingSenderId: "1[REDACTED]",
  appId: "1[REDACTED]"
};
```

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Setup-JS

```
//La variable de config pour firebase
const firebaseConfig = {
  apiKey: "-----VOS INFORMATIONS FIREBASE-----",
  authDomain: "-----VOS INFORMATIONS FIREBASE-----",
  databaseURL: "-----VOS INFORMATIONS FIREBASE-----",
  projectId: "-----VOS INFORMATIONS FIREBASE-----",
  storageBucket: "-----VOS INFORMATIONS FIREBASE-----",
  messagingSenderId: "-----VOS INFORMATIONS FIREBASE-----",
  appId: "-----VOS INFORMATIONS FIREBASE-----",
};

firebase.initializeApp(firebaseConfig);
//On va créer une référence à notre BDD
const dbRef = firebase.database().ref();
// On va également faire une ref directement dans le noeud / "table" users
const usersRef = dbRef.child("users");

const addUserBtnUI = document.getElementById("add-user-btn");
addUserBtnUI.addEventListener("click", addUserBtnClicked);

const formUserUI = document.getElementById("add-user-form");
formUserUI.addEventListener("submit", (event) => event.preventDefault());

const formUserEditUI = document.getElementById("edit-user-module");
formUserEditUI.addEventListener("submit", (event) => event.preventDefault());

const userListUI = document.getElementById("user-list");
const userDetailUI = document.getElementById("user-detail");

readUserData();

function addUserBtnClicked() {

};

function readUserData() {

};

function userClicked(event) {

};

function editButtonClicked(event) {

};

function saveUserBtnClicked() {

};

function deleteButtonClicked(event) {

}
```

Auteur :

Jean-François Pech

Relu, validé & visé par :

- Jérôme CHRETIENNE
- Sophie POULAKOS
- Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Setup-HTML

On va utiliser Firebase en mode CDN (pas besoin d'installation), ajouter les scripts de firebase à la fin du body de la page :

- firebase-app.js
- firebase-database.js

```
<!-- firebase sdk link goes here -->
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.4.2.firebaseio-app.js"></script>
<!-- <script src="https://www.gstatic.com/firebasejs/8.4.2/firebase-analytics.js"></script> -->
<script src="https://www.gstatic.com/firebasejs/8.4.2.firebaseio-database.js"></script>
<!-- JS de Bootstrap -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"
       integrity="sha384-kenU1KFdBIe4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4"
       crossorigin="anonymous"></script>
<script src="app.js"></script>
</body>
```

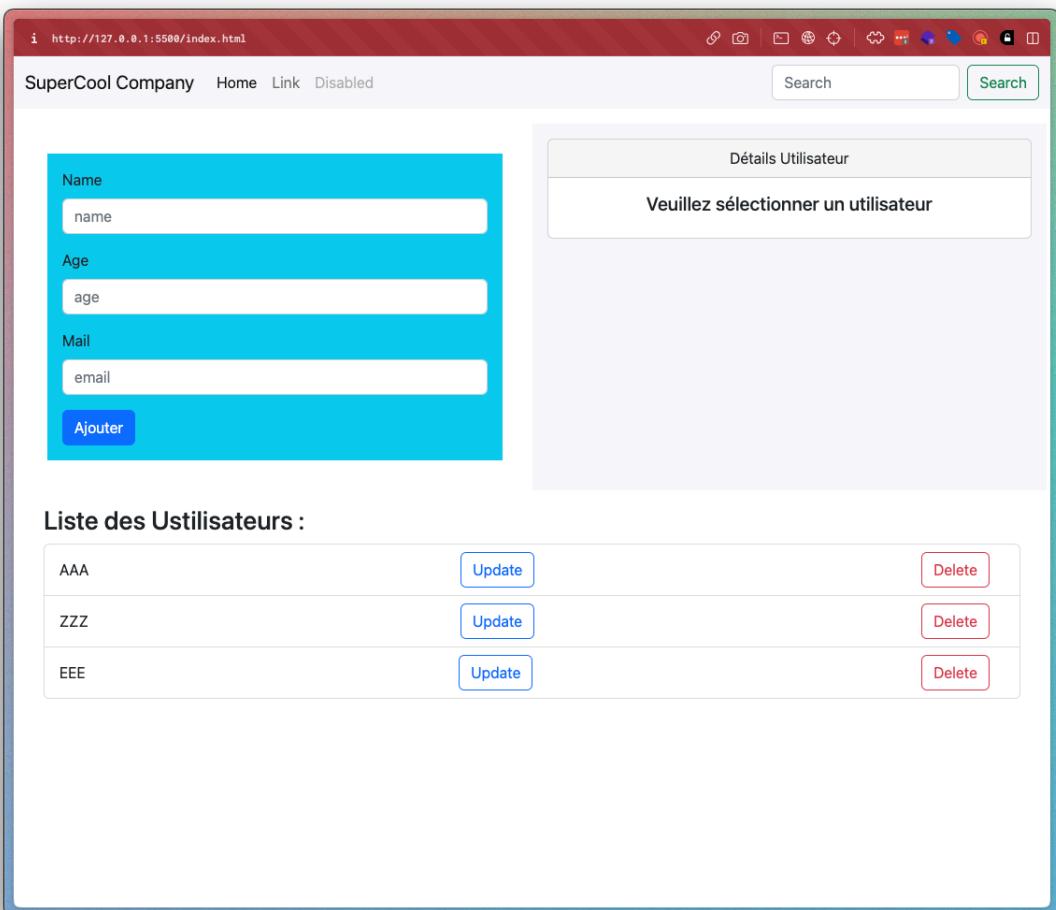
Il nous faudra :

- 1 formulaire avec id « add-user-form » (le formulaire pour ajouter un user) :
 - 3 inputs avec la classe « user-input »
 - Ces 3. Inputs ont un attribut data-key chacun avec comme valeurs respectives (name, age, mail)
 - 1 button avec id « add-user-btn »
- 1 formulaire avec id « edit-user-module » (le formulaire pour modifier un user) :
 - 3 inputs avec la classe « edit-user-input »
 - Ces 3. Inputs ont un attribut data-key chacun avec comme valeurs respectives (name, age, mail)
 - 1 button avec id « edit-user-btn »
- 1 <div> avec id « user-detail » (là où on affiche les infos détaillées d'un user) :
- 1 avec id « user-liste » (là où on affiche la liste de tous les users) :

Setup-CSS

Le formulaire pour éditer un utilisateur sera masqué de base, JS se chargera de l'afficher ou non

```
/*Edit*/
#edit-user-module {
    display: none;
}
```

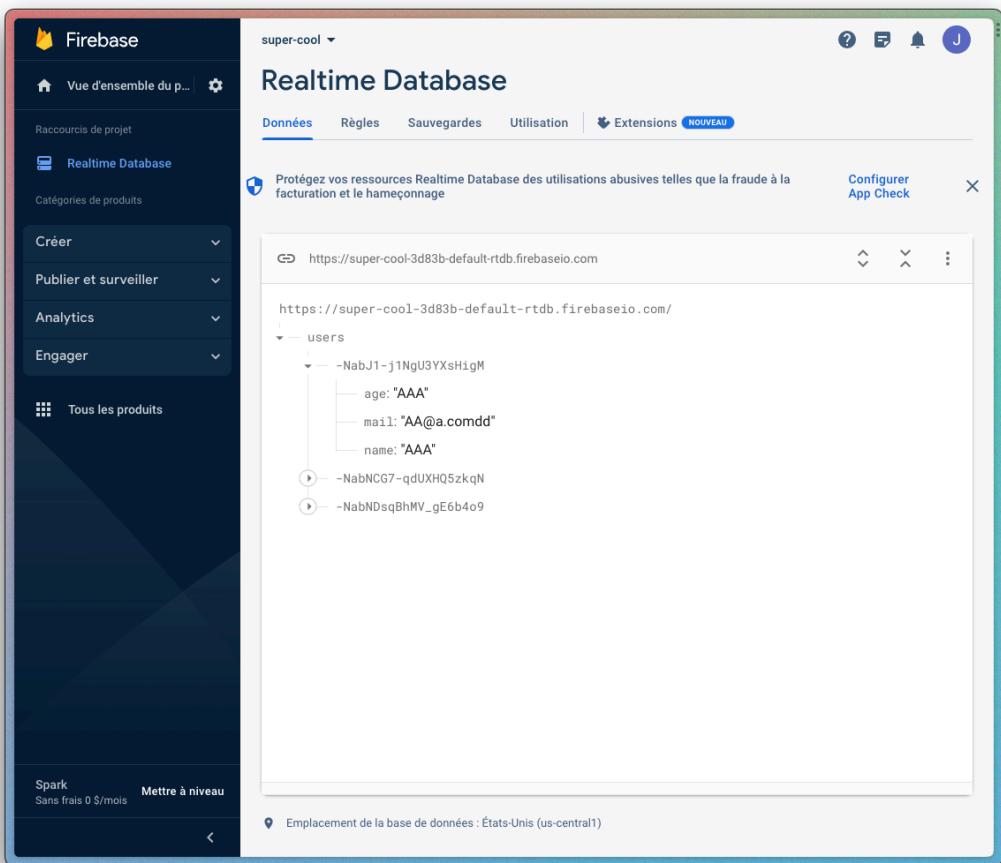


The screenshot shows a web application interface for managing users. On the left, there is a form for adding a new user with fields for Name, Age, and Mail, and a blue "Ajouter" button. On the right, there is a modal window titled "Détails Utilisateur" with the message "Veuillez sélectionner un utilisateur". Below this, there is a section titled "Liste des Utilisateurs:" displaying a table with three rows: AAA, ZZZ, and EEE. Each row has an "Update" button and a "Delete" button.

Exercice : CREATE : Ajouter un nouvel utilisateur

```
//TODO 5: Dans la f° addUserBtnClicked, Récupérer TOUS LES INPUTS avec la classe user-input 1 variable addUserInputsUI (getElementsByClassName)
//TODO 6: Dans la f° addUserBtnClicked, créer une variable newUser (qui est un objet vide)
//TODO 7: Dans la f° addUserBtnClicked, faire une boucle for pour parcourir les input dans addUserInputsUI
//TODO 8: Dans la Boucle, Pour chaque éléments parcourus on récupère Dans 1 variable key = addUserInputsUI[i].getAttribute('data-key');
//TODO 9: Dans la boucle, 1 variable value = addUserInputsUI[i].value
//TODO 10: Dans la boucle, Pour chaque clé (âge, name, email) on l'associe à notre nouvel utilisateur : newUser[key] = value
//TODO 11: après le parcours des inputs, sur usersRef on va faire un push de newUser
//TODO 12: Dans la f° addUserBtnClicked, on console log un msg type nouvel utilisateur enregistré
//TODO 13: Dans la f° addUserBtnClicked, On console log le nom et l'âge du nouvel utilisateur
//TODO 14: Dans la f° addUserBtnClicked, On ré initialise le formulaire avec l'id add-user-form
```

Après avoir testé le formulaire d'ajout peut se rendre dans la console Firebase du projet pour constater que Friable a crée un noeud « users » avec un premier objet pour notre premier utilisateur avec un id généré automatiquement par Firebase :



The screenshot shows the Firebase Realtime Database interface. On the left, there's a sidebar with project settings and a main area titled "Realtime Database". In the main area, under the "Données" tab, a single user node is displayed with the following structure:

```

https://super-cool-3d83b-default.firebaseio.com/
  users
    -NabJ1-j1NgU3YXsHigM
      age: "AAA"
      mail: "AA@a.comdd"
      name: "AAA"
    -NabNG7-qdUXHQ5zkqN
    -NabNDsqBhMV_gE6b4o9
  
```

Exercice : READ (Lecture de la BDD : Tous Les Utilisateurs)

```
/* LIRE TOUT LES USERS
//TODO : Dans la f° readUserData
//TODO : sur la variable usersRef on va utiliser une fonction .on()
//? Pour info .on() va s'utiliser comme un addEventListener
//TODO : 1er param de .on(), une string "value" (en gros dans la bdd on surveille si ya des changements de value)
//TODO : 2e param de .on(), une f° fléchée qui prend un paramètre snap
//? usersRef.on("value", (snap) => {});
//TODO : Dans la fonction fléchée : on va assigner une string vide au innerHTML de userListUI
//TODO : Sur la variable snap on va utiliser un forEach pour parcourir le tableau avec une variable temporaire childSnap
//TODO : Dans le forEach : dans une variable key on va stocker childSnap.key
//TODO : Dans le forEach : dans une variable value on va stocker childSnap.val()
//TODO : Dans le forEach : dans une variable $li on va créer un element <li></li>
//TODO : Dans le forEach : dans le innerHTML de $li on lui assigne value.name
//TODO : Dans le forEach : Sur la $li on lui rajoute un attribut 'user-key', on lui assignera la valeur stockée dans key
//TODO : Dans le forEach : dans la userListUI on va placer $li
```

Liste des Utilisateurs :

AAA

ZZZ

EEE

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

Date révision :

10/03/2023

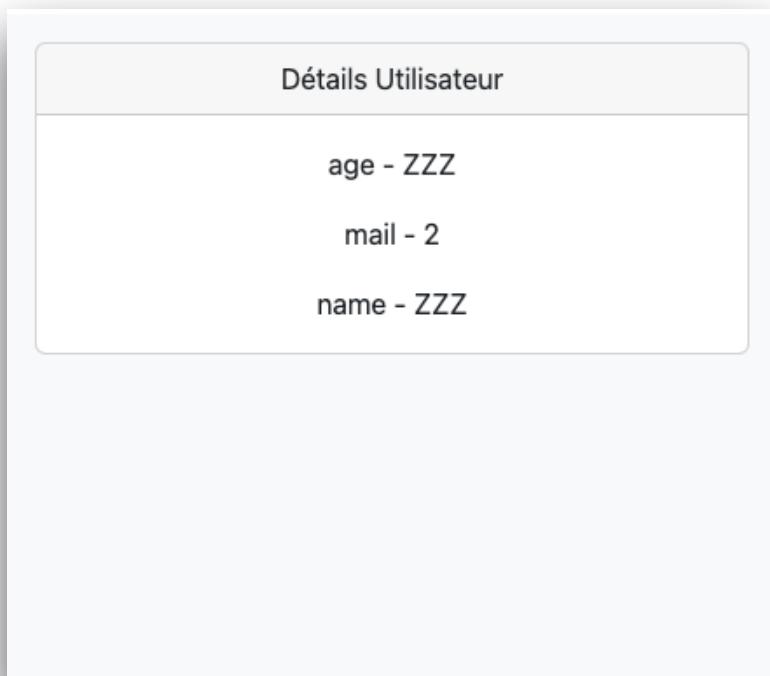


Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Exercice : READ (Lecture d'un élément de la BDD : Afficher 1 utilisateur)

```
/* LIRE 1 USER
//TODO: Dans readUserData avant le append(), on va placer un addEventListener sur $li qui écoute « click » et lance la fonction userClicked
//? Ensuite dans la fonction userClicked on capte l'évènement (on s'en sert pour savoir qui on sélectionne)
//TODO: Dans la f° userClicked, Dans 1 variable userID, on va récupérer userID via event.target.getAttribute("user-key");
//TODO: Dans la f° userClicked, 1 variable userRef va faire référence à 1 utilisateur en particulier, on lui assigne dbRef,
//? on utilise la fonction child() pour viser le noeud "users/" concaténé avec userID
//TODO: Dans la f° userClicked, 1 variable userDetailUI récupère ma div avec user-detail
//TODO: Dans la f° userClicked, Ensuite sur userRef on utilise la fonction on("value", snap =>{ })
//TODO: Dans la f° userClicked, Dans la fonction =>, on va vider l'innerHTML de userDetailUI
//TODO: Dans la f° userClicked, Ensuite sur snap on va utiliser un forEach pour parcourir le cliché (snap) de notre BDD.
//TODO: Dans la f° userClicked dans le forEach, 1 variable $p créée un élément <p>
//TODO: Dans la f° userClicked dans le forEach, On remplit le innerHTML de $p avec childSnap.key et childSnap.val()
//TODO: Dans la f° userClicked dans le forEach, On rajoute $p dans notre userDetailUI
```

Quand on click sur un utilisateur dans la liste cela va récupérer son ID, pour lire la base de donnée et afficher les informations dans la div user-detail

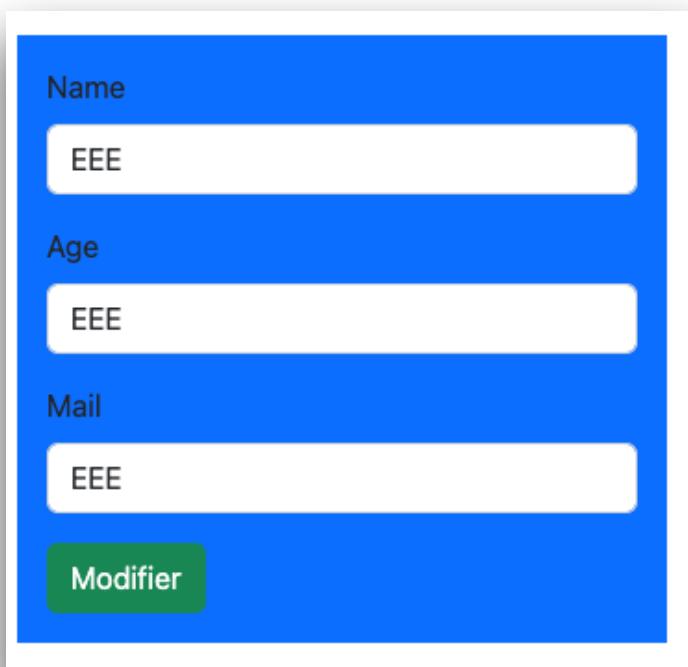


Exercice : READ (Lecture d'un élément de la BDD : Pré-remplir le formulaire d'un utilisateur)

Cette fonction est assez similaire à la précédente elle va lire les informations d'un utilisateur en base de données pour pré-remplir le formulaire de modification.

```
//*UPDATE (EDITER)
//! Dans la f° editButtonClicked
//TODO: on va modifier le display d formUserEditUI à block
//TODO: on va modifier le display du forUserUI à none
//TODO: Ensuite on va faire ceci :
//TODO: Une variable inputId qui récupère (querySelector) l'élément avec la classe .edit-userid
//TODO: A la valeur de inputId on assigne event.target.getAttribute("userid");
//TODO: Créer une variable userRef on lui assigne dbRef.child('users/' + inputId);
//TODO: Dans une variable editUserInputsUI, on récupère tous les éléments de classe edit-user-input (querySelectorAll ou autre)
//TODO: On va parcourir notre BDD avec userRef.on("value", snap => {
//TODO: dans la f° fléchée, Faire une boucle for qui parcourt les inputs editUserInputsUI,
//TODO: dans la f° fléchée dans la boucle, dans une variable key, on stock editUserInputsUI[i].getAttribute("data-key");
//TODO: dans la f° fléchée dans la boucle, Ensuite à chaque valeur de nos editUserInputsUI[i] on assigne snap.val()[key];
//TODO: En dehors de la fonction on(), Dans une variable saveBtn, on récupère notre bouton avec l id édit-user-btn
//TODO: En dehors de la fonction on(), Sur ce bouton on place un eventListener au click qui lance saveUserBtnClicked
```

Quand on clique sur le bouton update dans la liste des utilisateurs



Auteur :

Jean-François Pech

Relu, validé & visé par :

Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

Date création :

03/03/2023

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.

Exercice : UPDATE : Sauvegarder les modifications d'un utilisateur

```
/*SAVE
//TODO : Dans la f° saveUserBtnClicked :
//TODO : Dans une variable userID on récupère la VALUE de l'input avec l'id "edit-userid"
//TODO : Dans une variable userRef on fait une référence à l'utilisateur dans la BDD
//TODO : Une variable editedUserObject qui est un objet vide
//TODO : Dans une variable editUserInputsUI on récupère TOUS les éléments html qui ont la classe "edit-user-input" (querySelectorAll)
//TODO : Ensuite on va faire une boucle forEach pour parcourir les editUserInputsUI
//TODO : Dans les param de forEach(), on lui passe une fonction qui a une variable textField en paramètre
//TODO : Dans cette fonction, dans le forEach on aura une variable key qui va stocker les attributs data-key de textField (getAttribute())
//TODO : Pour chaque clé (âge, name, email) on l'associe à notre nouvel utilisateur : editedUserObject[key] = textField.value
//TODO : Ensuite en dehors de la boucle, sur notre variable userRef on utilise la f° update en lui passant editedUserObject en paramètre
//TODO : Enfin on peut remettre le display à "none" de formUserEditUI et à "block" pour formUserUI|
```

Quand on click sur le bouton « modifier » de notre formulaire de modification d'un utilisateur, cela met à jours les données d'un utilisateur en temps réel



The screenshot illustrates a user modification process. On the left, a blue-themed form titled 'Name' contains three input fields: 'ALB' for Name, '26' for Age, and 'ALB@company.com' for Mail. Below these is a green 'Modifier' button. On the right, a modal window titled 'Détails Utilisateur' displays the updated information: 'age - 26', 'mail - ALB@company.com', and 'name - ALB'. This visualizes how changes made in the UI are reflected in the database.

Exercice : DELETE (Supprimer un utilisateur)

```
/* SUPPRIMER
//TODO 1: Dans la f° readUserData, dans le forEach, juste avant $li.innerHTML = ...
//TODO 2: Dans la f° readUserData, dans le forEach, On va déclarer une variable deleteIconUI dans laquelle on va créer un élément span
//TODO 3: Dans la f° readUserData, dans le forEach, On va ensuite modifier la class de deleteIconUI en « delete-user »
//TODO 4: Dans la f° readUserData, dans le forEach, On va remplir le innerHTML de deleteIconUI avec un « X »
//TODO 5: Dans la f° readUserData, dans le forEach, deleteIconUI on lui rajoute un attribut « userid » qui prendra la valeur de key( via setAttribute)
//TODO 6: Dans la f° readUserData, dans le forEach, Enfin sur deleteIconUI on place un addEventListener qui écoute le click et lance la fonction deleteButtonClicked
//TODO 7: Créer une fonction deleteButtonClicked qui prend event en paramètre
//TODO 8: Dans la f° deleteButtonClicked, récupérer le userID via event.target.getAttribute
//TODO 9: Dans la f° deleteButtonClicked,
//TOD09-2: Faire une référence userRef à notre BDD directement sur le noeud de l'utilisateur qu'on a cliqué (référence à la table users + userID)
//TODO 10: Dans la f° deleteButtonClicked, utiliser la fonction remove sur userRef
```

On vise un utilisateur en particulier (comme dans les fonctions userClicked, editButtonClicked, et saveUserBtnClicked) pour le supprimer en temps réel de la base de données

<https://github.com/jefff404/cours-js/tree/25-firebase-cdn-js>

Conclusion



Pour aller plus loin :

<https://bradfrost.com/blog/post/front-of-the-front-end-and-back-of-the-front-end-web-development/>

Des frameworks FrontEnd :

- [Angular](#)
- [Vue](#)
- [React](#)
- [Ember](#)

Framework SSR (Server Side Rendering) :

Qu'est ce que le SSR ? Le rendu côté serveur d'une page web ou Server Side Rendering (SSR) est une technique de développement web qui consiste à créer les pages html côté serveur pour les envoyer toutes faites au navigateur.

- [Nuxt](#)
- [Next](#)
- [Svelte](#)
- [Qwik](#)

Framework Mobile - Cross Platform :

- [Ionic](#)
- [React Native](#)

De la 3D - AR-VR ? :

- [Three.js](#)

Automatisation - Testing :

- [Mocha](#)
- [Jest](#)
- [Jasmine](#)
- [Cypress](#)
- [Puppeteer](#)

Auteur :

Jean-François Pech

Date création :

03/03/2023

Relu, validé & visé par :

Jérôme CHRETIENNE
 Sophie POULAKOS
 Mathieu PARIS

Date révision :

10/03/2023



Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.