Manuel d’utilisation et de produit

Soumis par:

MyDoctor FA 1.4

Luigi Formica, 300237343

Rémi Gros-Jean, 300258238

Mary Mousa, 300110697

Chedly Redissi, 300074863

Youssef El Horri, 300136777

Sean Essimengane, 300268234

7 décembre 2022

Université d’Ottawa

Table des matières

2 Aperçu 4
2.1 Mises en garde et avertissements 5

3 Pour commencer 6
3.1. Application mobile 7
Nodejs 7
Code source 7
3.2. Firebase 7
Firebase firestore 12
Firebase auth 15
Firebase functions 16
Firebase messaging 18
La gestion de campagne de test 18
La gestion de la permission 21
3.1 Considérations pour la configuration 24
3.2 Considérations pour l'accès des utilisateurs 24
3.3 Accéder/installation du système 24
3.4 Organisation du système & navigation 24
3.5 Quitter le système 25

4 Utiliser le système 25
4.1 25

5 Dépannage & assistance 31
5.1 Messages ou comportements d'erreur 31
5.2 Considérations spéciales 31
5.3 Entretien 31
5.4 Assistance 31

6 Documentation du produit 32
6.1 Design du prototype 32
6.1.1 React Native 32
6.1.2 Base de donnée cloud Firebase 32
6.1.3 Structure de la base de code 32
6.1.4 UI et logique interne 35
6.2 Sous-système 1 du prototype 37
6.2.1 NDM (Nomenclature des Matériaux) 37
6.2.2 Liste d'équipements 37
6.2.3 Instructions 37
6.3 Essais & validation 38

7 Conclusions et recommandations pour les travaux futurs 40

9 APPENDICE I: Fichiers de conception 40

1 Introduction

Ce manuel d'utilisation et de produit (MUP) fournit les informations nécessaires à Jeff Puncher et son équipe au département de la médecine familiale de l'Université d'Ottawa pour utiliser efficacement l'application MyDoctor et pour la documentation du prototype.

L'objectif du MUP est de permettre à l'utilisateur de pouvoir utiliser MyDoctor à son plein potentiel grâce aux descriptions, images, tableaux et instructions inclus dans le MUP.

Ce MUP traite en profondeur le problème résolu, son installation, l'utilisation générale, ses fonctionnalités, les problèmes potentiels, l'entretien nécessaire, les travaux futurs possibles ainsi que la documentation de chacune des parties de MyDoctor.

2 Aperçu

Le problème auquel nous faisons face actuellement, et qui a conduit au développement de ce projet, est la pénurie de médecin. En effet, au-delà de 25 000 habitants d'Ottawa se retrouvent sans médecin de famille. Par conséquent, cela conduit à de long délai d'attente, généralement dans des salles qui se retrouvent pleines de patient qui augmente la probabilité de propager encore plus les virus. De surcroît, il n'y a aucun développement notable dans le système de santé qui puisse améliorer la situation. C'est selon cette perspective, que notre solution a pour but de faire évoluer le système et de permettre à toute famille non seulement d'avoir un médecin, mais aussi de faciliter la recherche d'un médecin de famille dans leur secteur désiré.

Notre produit a pour utilisateur principal toute personne ou famille à la recherche d'un médecin. Le besoin de ce dernier est d'avoir à disposition tous les médecins qui lui sont proches et qui sont capables de recevoir de nouveaux patients afin de prendre rendez-vous.

MyDoctor se différencie de tous les autres dans le marché vu qu'il est plus pratique que les sites web existants. Ce dernier est une application téléchargeable qui permet à l'utilisateur d'entrer son adresse et de voir tous les médecins à proximité, et ce, sans avoir à créer de compte. De plus, il affiche les médecins ayant de la place pour accueillir de nouveaux patients, et donne la possibilité à l'utilisateur de s'abonner à un médecin qui n'a pas de place afin de recevoir une alerte lorsque ce dernier est prêt à recevoir de nouveau patients.

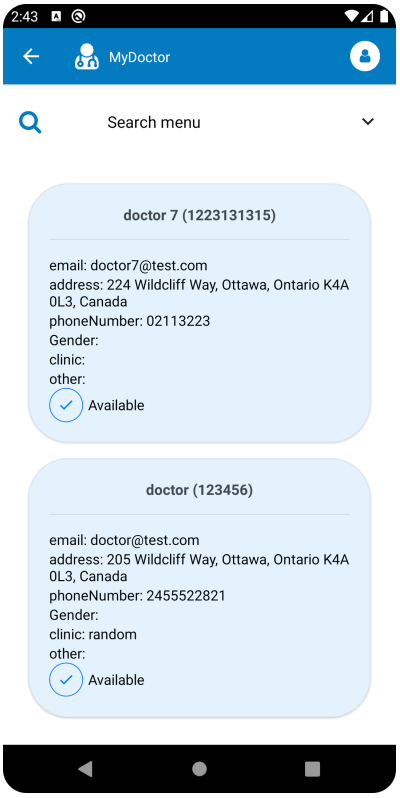


Figure 2.1

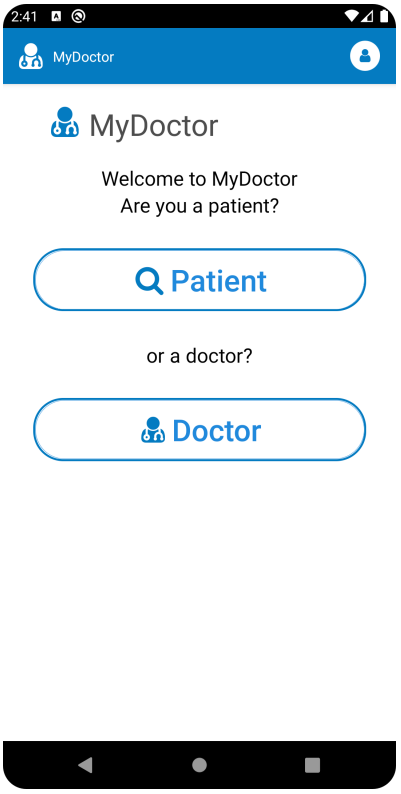


Figure 2.2

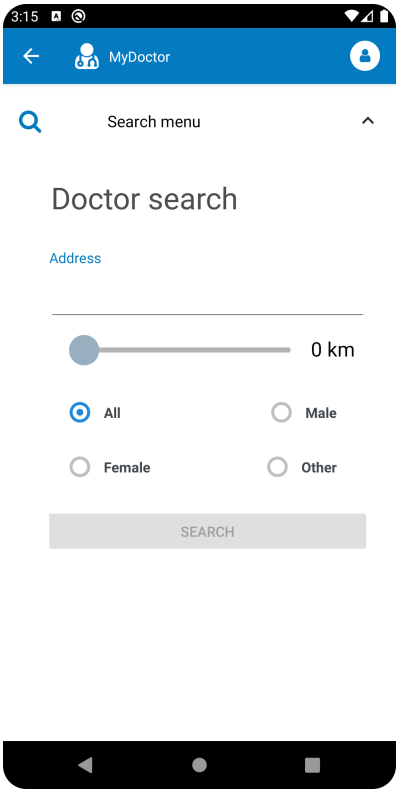


Figure 2.3

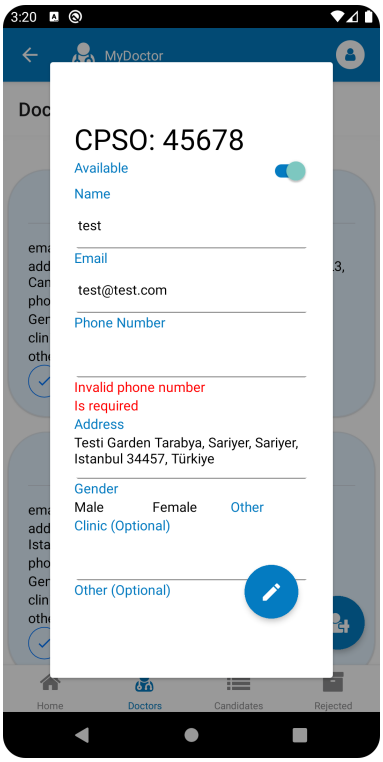


Figure 2.4

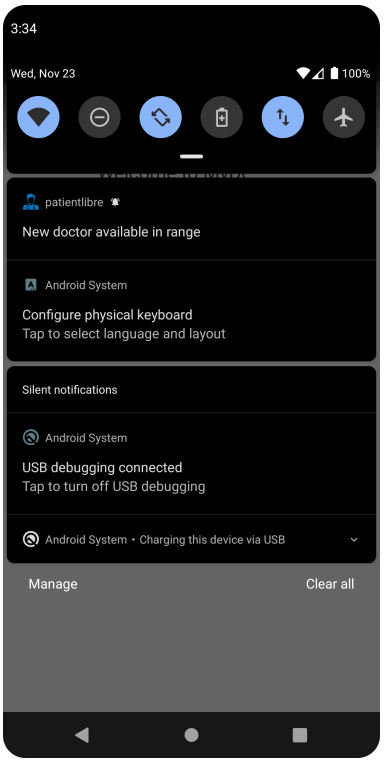
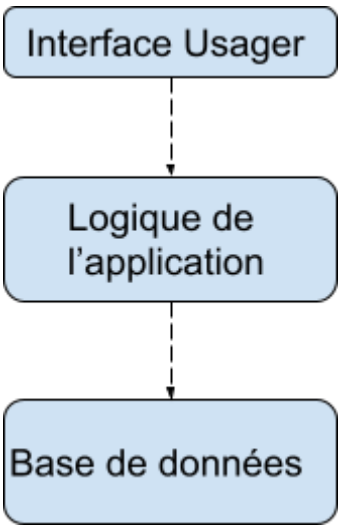


Figure 2.5

La fonction principale de notre produit est de faciliter le système de santé, le rendre plus fluide en ce qui concerne la prise de rendez-vous. Il inclut une recherche de docteur disponible dans un rayon de recherche pour une adresse entrée par l'utilisateur et donne aussi la possibilité de s'abonner à ce rayon de recherche pour de futures disponibilités.

Le système peut être séparé en trois couches principales:



Ces couches forment une hiérarchie ou les couches supérieures ont accès aux services des couches inférieures.

Chacune de ces couches peuvent être divisées en plusieurs sections qui s'occupent de certains aspects spécifiques.

Interface Usager:

- Pages admin
- Pages login, signup, et oublie de mot de passe
- Pages docteur
- Pages patient
- Style des pages
- Page d'accueil
- Entête
- Menu pour entrer l'adresse
- Texte titre

Logique de l’application:
- Logique des fonctionnalité de l’admin
- Logique d’authentification.
- Hook qui détermine l’orientation de l’appareil mobile (landscape/portrait)
- Logique des fonctionnalité des docteurs
- Logique de prédiction d'adresse et géolocalisation
- Navigation entre les pages
- Enregistrement d’address et notifications
- Contraintes utilisé par validate.js pour valider l'entrée utilisateur

Base de données:
- Validation des numéro CPSO
- Gestion des configuration Firebase
- Notifications des utilisateurs
- Activation, désactivation, et supprimer des comptes
- Création de compte

2.1 Mises en garde et avertissements

L’autorisation pour la publication de MyDoctor ainsi que l’utilisation du compte ADMIN est strictement réservée aux départements de la médecine familiale de l’Université d’Ottawa. L’utilisation des comptes docteurs requiert un numéro de CPSO valide afin d’être activé. L’accès au compte patients est permis à tous les utilisateurs et ne demande aucune autorisation spécifique pour se faire.

3 Pour commencer

Notre produit est une application mobile avec react-native et utilise firebase. D'où il y a plusieurs configurations à faire.
Pour les présentations, les configurations et les préparations ont déjà été faites. Afin d'utiliser l'application, il faut télécharger le fichier apk (pour les appareils android).
et suivre les étapes d'installations démontées par les figures 3.1 à 3.6.

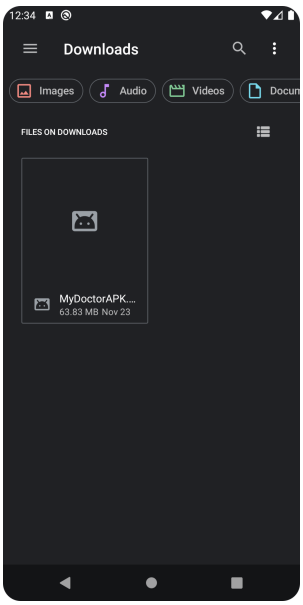


Figure 3.1: etape 1

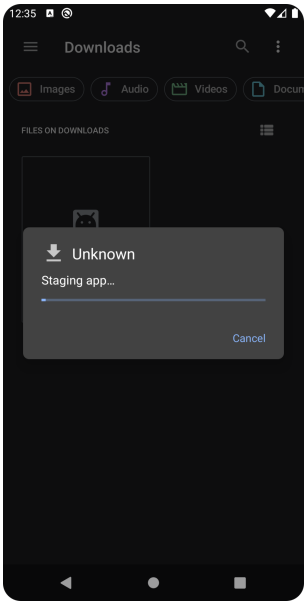


Figure 3.2: etape 2

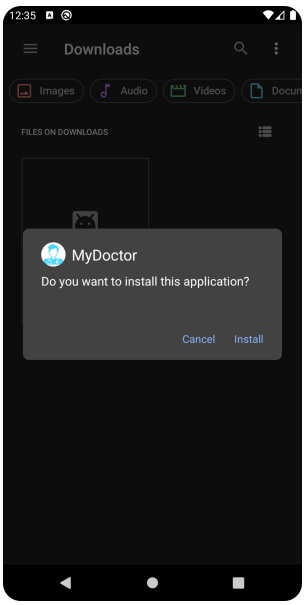


Figure 3.3: etape 3

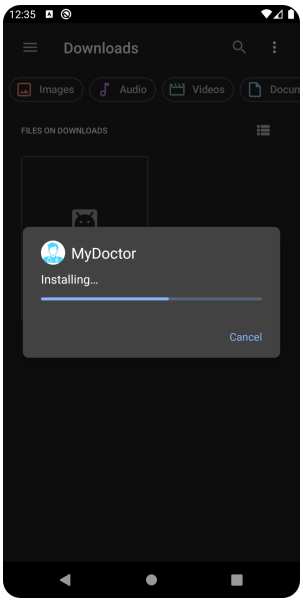


Figure 3.4: etape 4

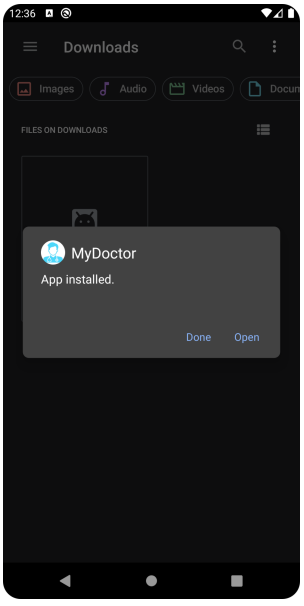


Figure 3.5: etape 5

Figure 3.6: etape 6

Ci-dessous, sont les étapes nécessaires pour la configuration et l'installation pour les développeurs.

3.1. Application mobile

Nodejs

React-native dépend des nodejs. L'application a été testé avec la version 16.13.2 qui peut être téléchargé à partir de https://nodejs.org/download/release/v16.13.2/

Code source

Pour cette étape, il suffit de procurer le code source de l'application.

Figure 3.1.2: Vue des fichiers sur github

3.2. Firebase

Notre application utilise les services firebase suivantes:

1. firebase firestore,
2. firebase auth,
3. firebase functions,
4. firebase messaging

Afin d'utiliser firebase, il faut créer un compte google puis créer un projet sur firebase. Les figures 3.2.1 a 3.2.5 montrent les étapes nécessaires pour créer un projet sur firebase.

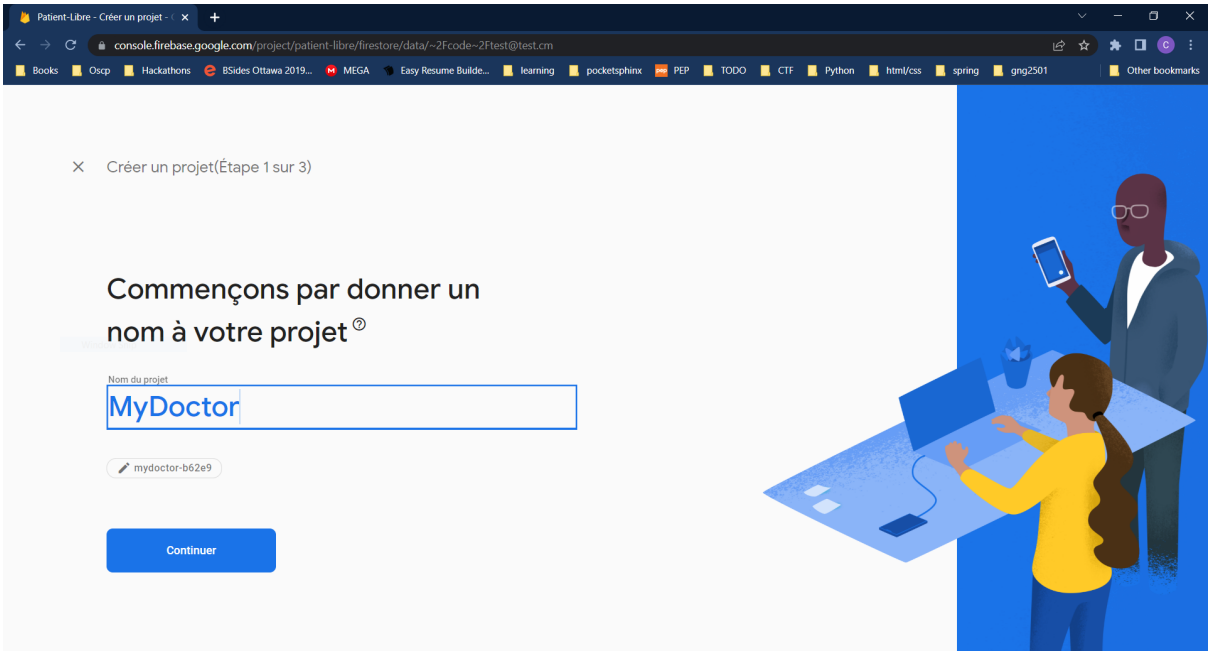


Figure 3.2.1: etape 1

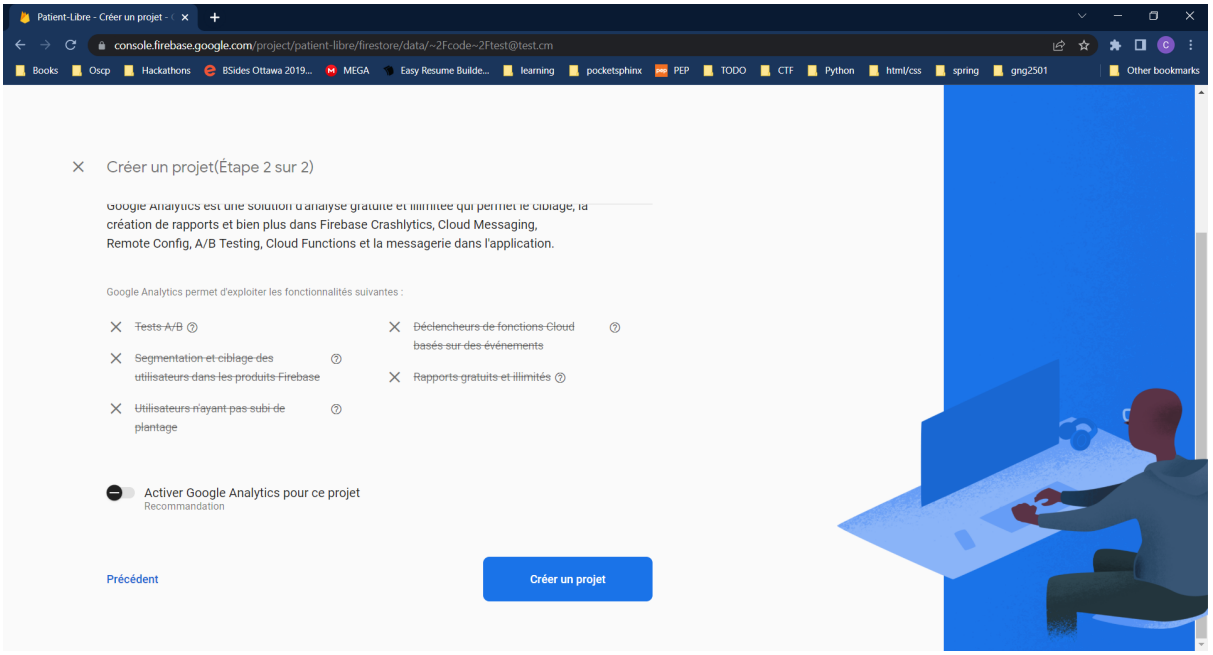


Figure 3.2.2: etape 2

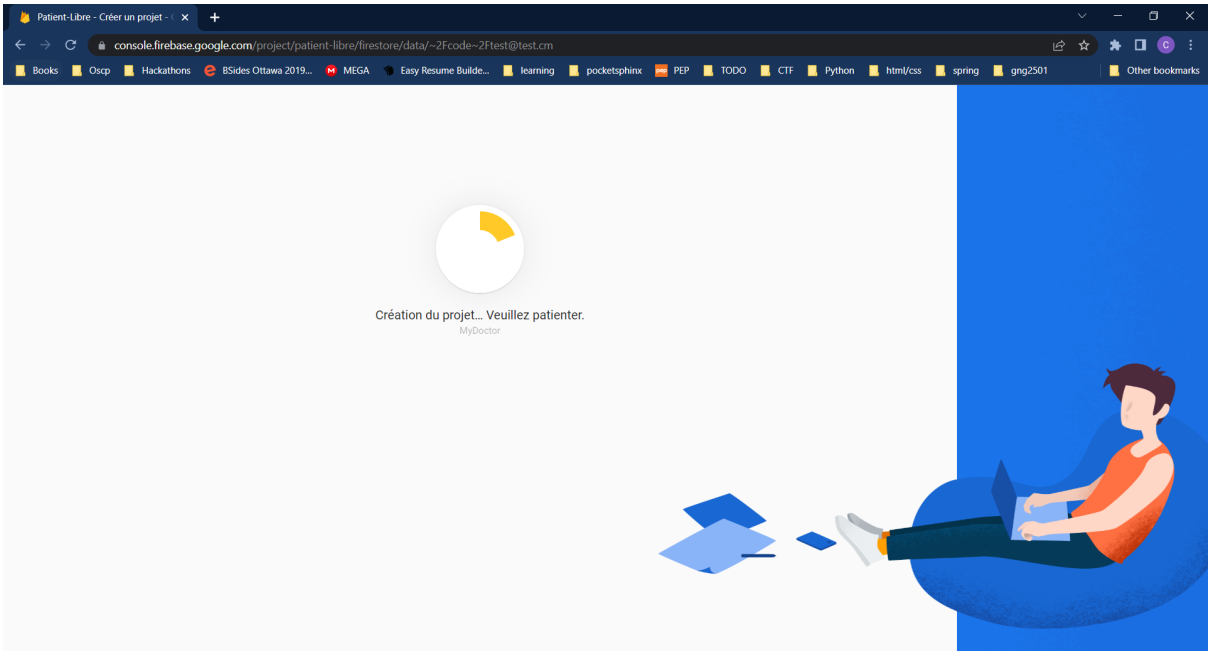


Figure 3.2.3: etape 3

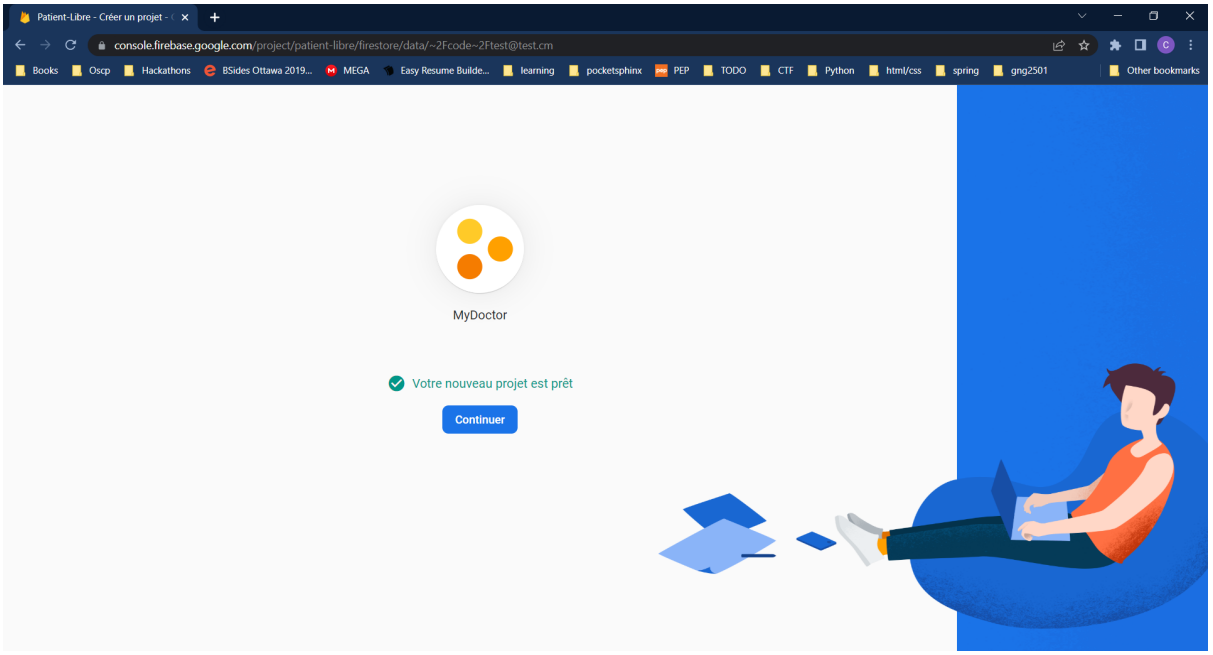


Figure 3.2.4: etape 4

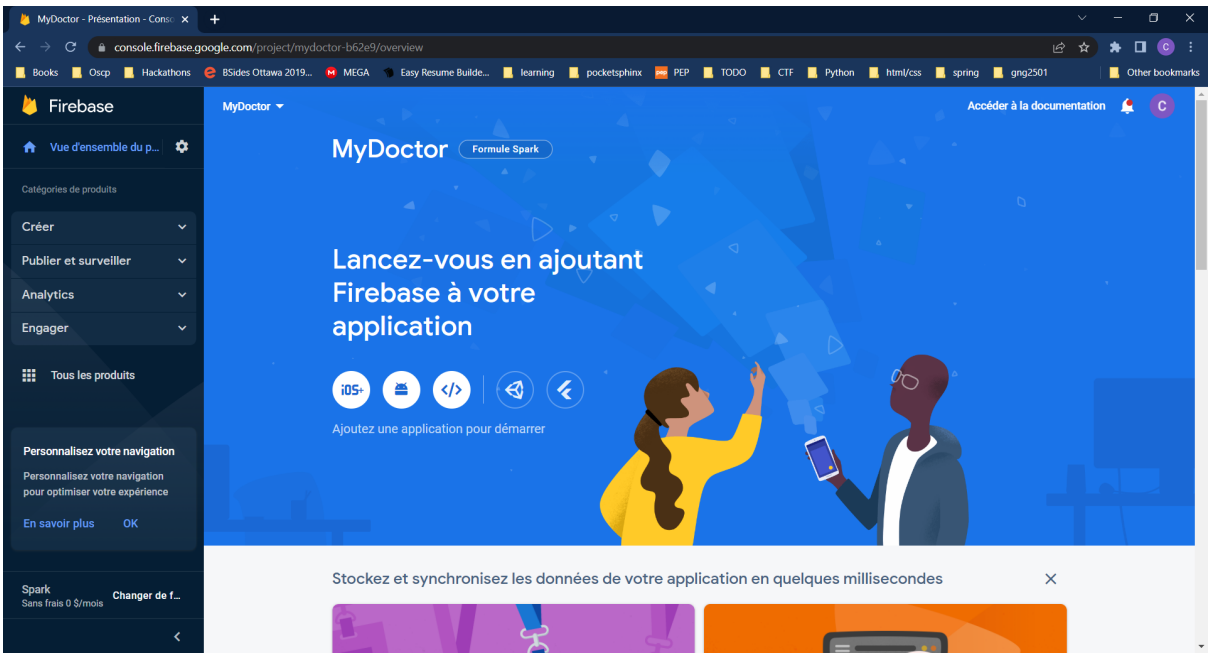


Figure 3.2.5: etape 5

Après avoir créé ce projet, il faut ajouter le fichier de configuration à l'application mobile. Firebase offre des instructions sur comment le faire, et les figures 3.2.6 à 3.2.10 montrent comment générer ces fichiers. L'exemple qu'on a pris est celui de la plateforme Android:

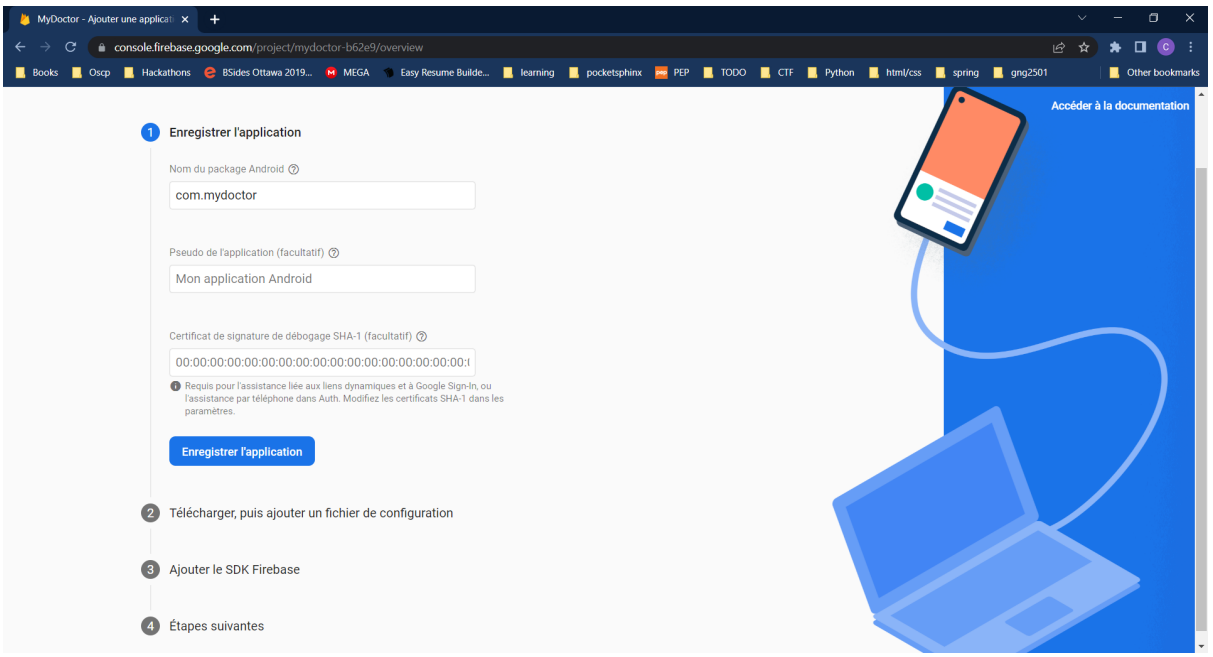


Figure 3.2.6: configuration android: etape 1

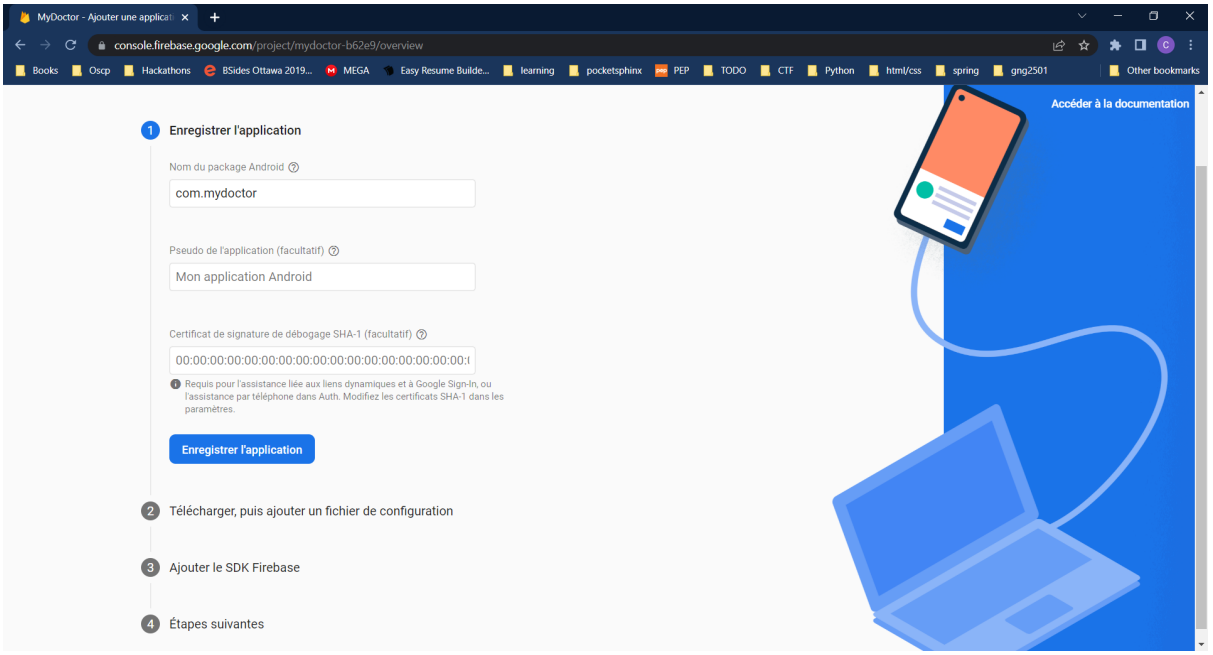


Figure 3.2.7: configuration android: etape 2

Figure 3.2.8: configuration android: etape 3

Figure 3.2.9: configuration android: etape 4

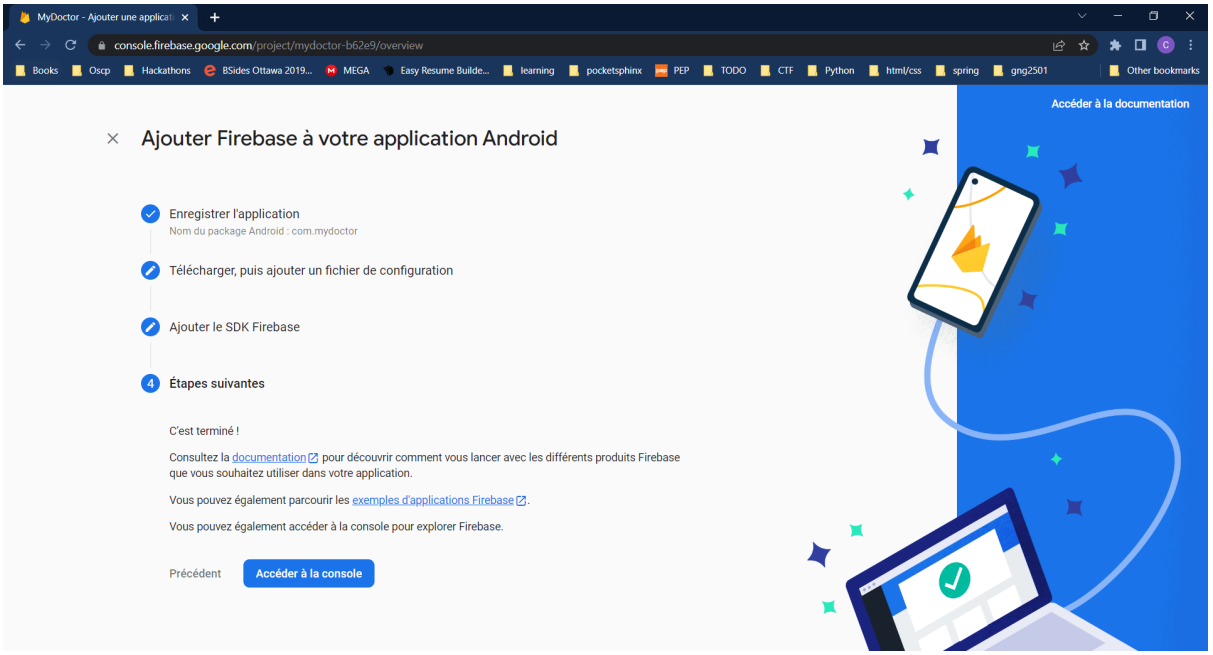


Figure 3.2.10: configuration android: etape 5

Firebase firestore

Firebase firestore est un service de base de données en temps réel. Aucune configuration compliquée n'est nécessaire pour firebase firestore, il suffit de créer une base de données en suivant les étapes 3.2.11 a 3.2.13.

La suivante est liste de role de securite de firebase, a copier dans l'entre de la figure 3.2.12:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
            // user is logged in admin
    function isAdmin() {
      return request.auth != null
          || request.auth.token.email == "admin@test.com";
    }
    function isDoctorDocument(document) {
      return request.auth != null
          || request.auth.token.uid == document;
    }

    // Match any document in the 'doctors' collection
    match /doctors/{document=**} {
      allow read: if true;
                              allow update: if isAdmin() || isDoctorDocument(document);
      allow create, delete: if isAdmin();
    }

    // Match any document in the 'codes' collection
    match /codes/{document=**} {
      allow read, write: if isAdmin();
    }

    // Match any document in the 'candidates' collection
    match /candidates/{document=**} {
      allow read, write: if isAdmin();
    }

    // Match any document in the 'rejected' collection
    match /rejected/{document=**} {
      allow read, write: if isAdmin();
    }
  }
}
```

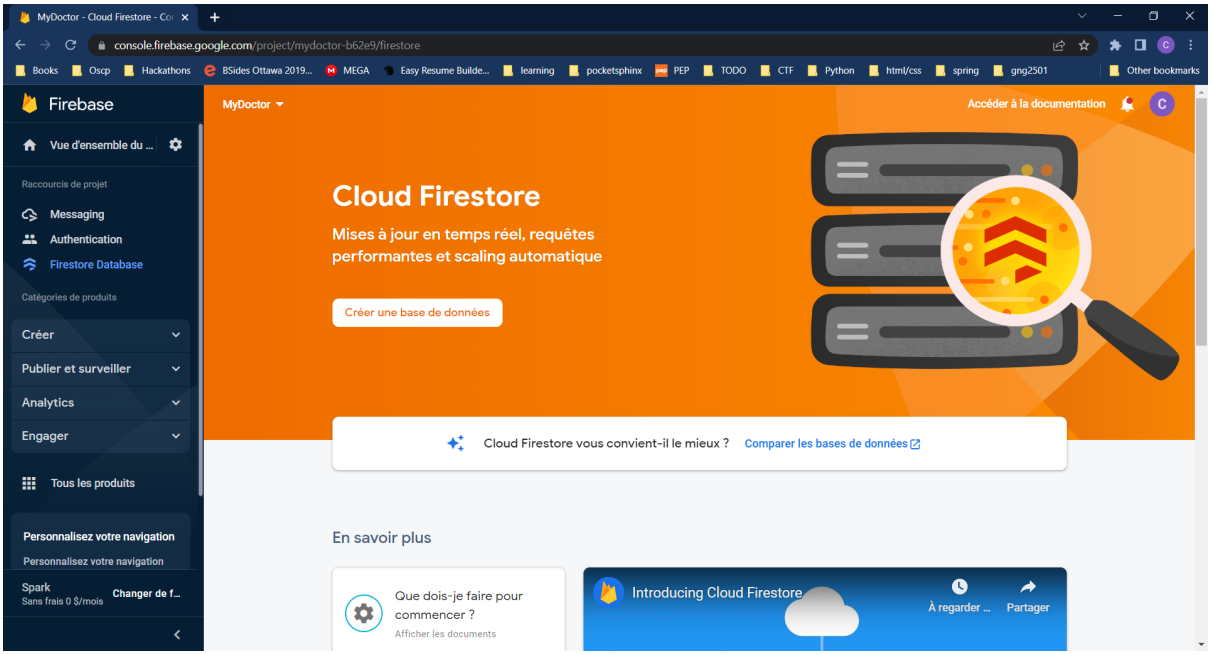


Figure 3.2.11: configuration firestore: création de base de données

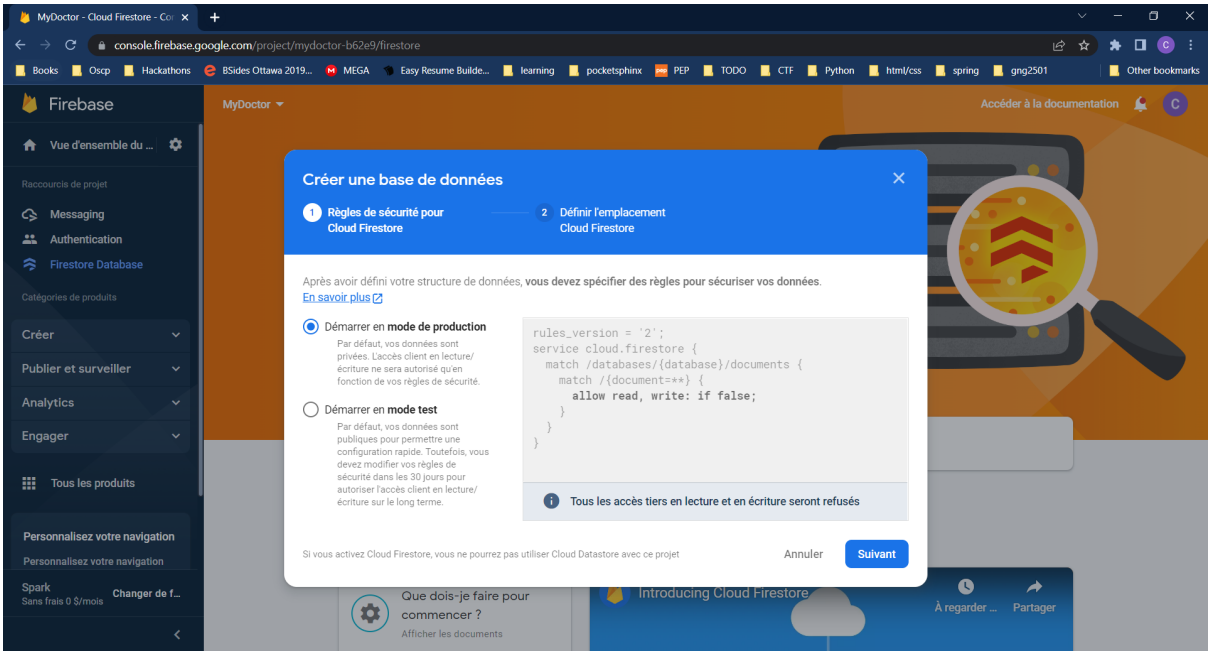


Figure 3.2.12: configuration firestore: configurations de sécurité

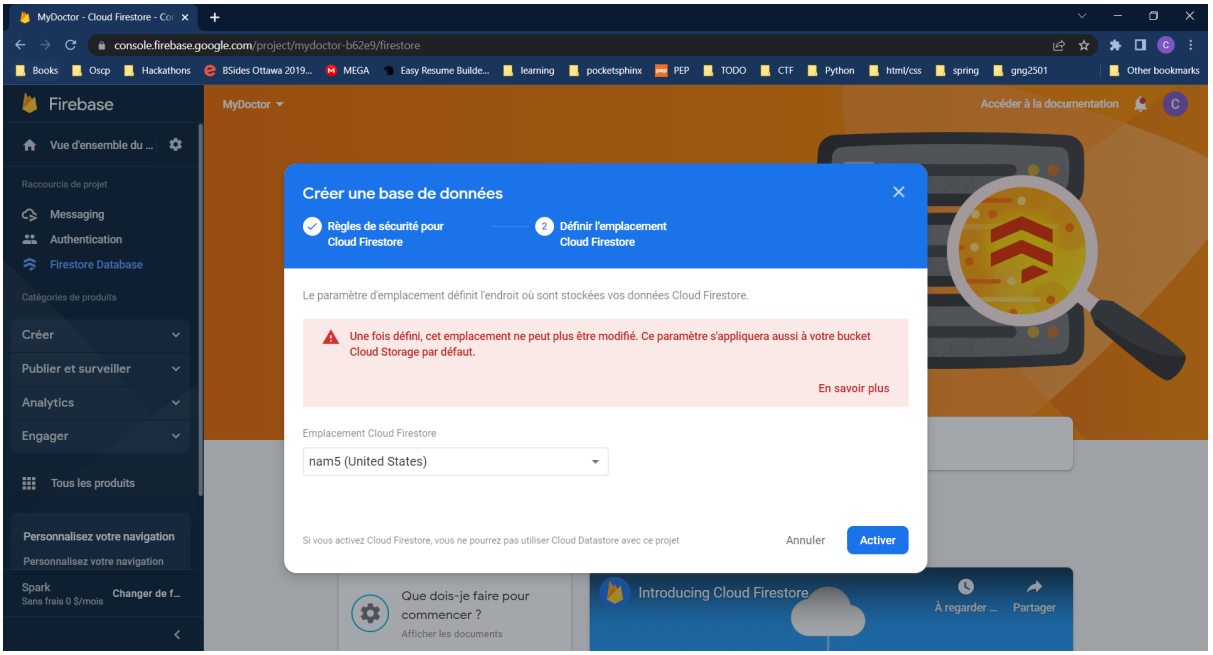


Figure 3.2.13: configuration firestore: choix de région

Firebase auth

Firebase auth est le service de gestion d'authentification de notre application. les figures 3.2.14 a 3.2.16 montre la configuration nécessaire pour firebase auth. Pour notre prototype, on utilise la méthode de connexion email et mot de passe.

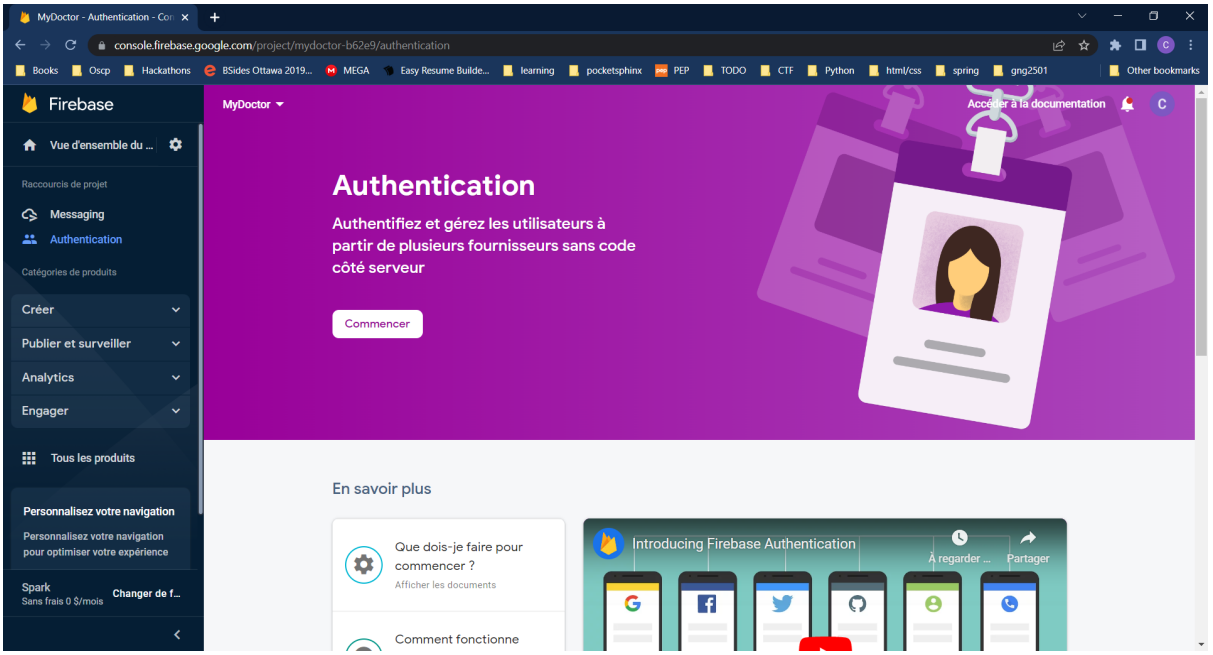


Figure 3.2.14: configuration firebase auth: créer le service

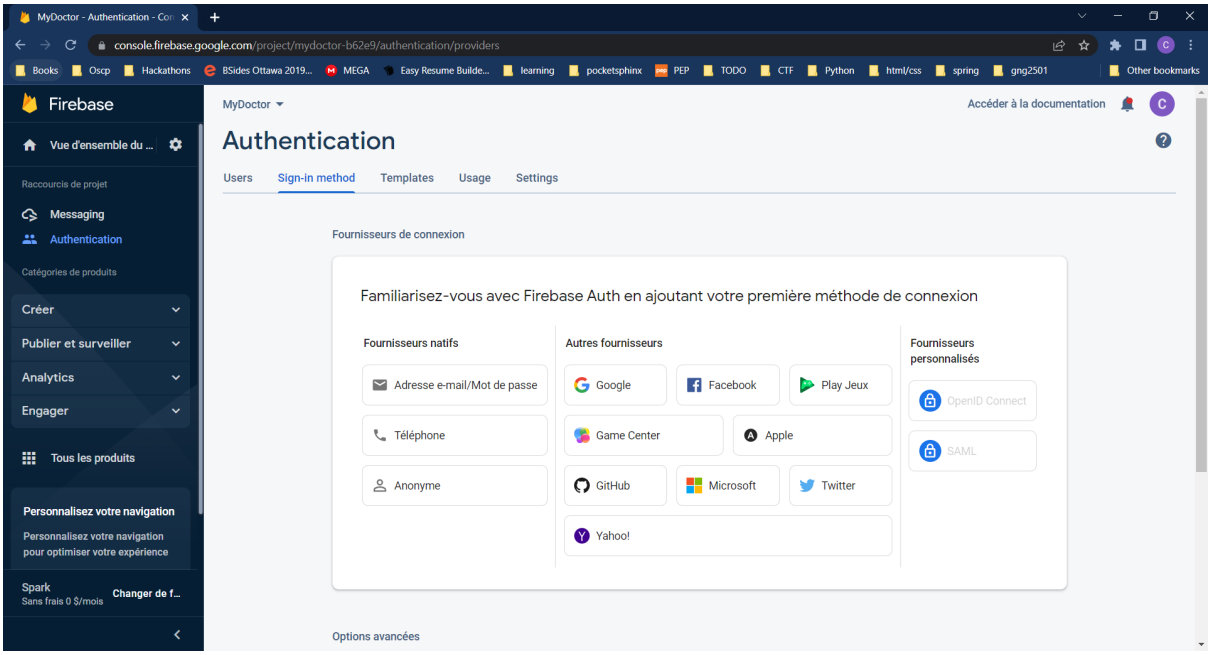


Figure 3.2.15: configuration firebase auth: choix de méthode de connexion (email et mot de passe)

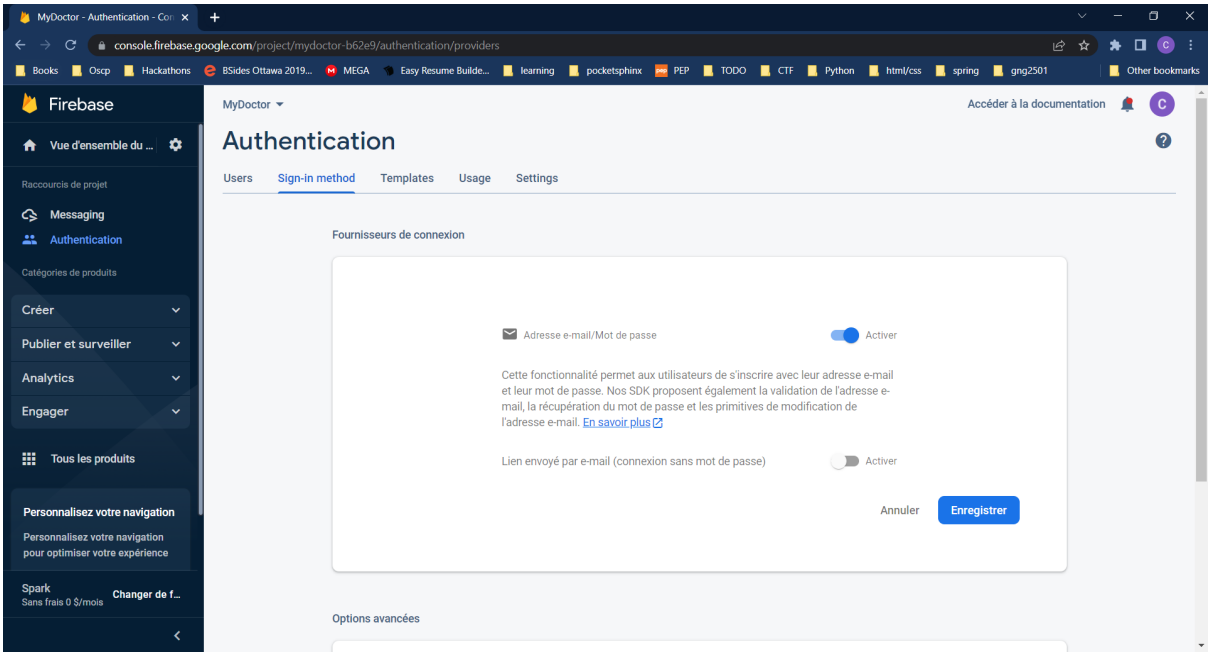


Figure 3.2.16: configuration firebase auth: fin de configuration

Et vu notre décision d'avoir un seul administrateur qui ne change pas, on utilise le courriel admin@test.com comme courriel du compte admin, et doit être créé manuellement à partir de la console.

Firebase functions

Certaines opérations plus compliquées ou délicates sont faites par des appels à firebase functions au lieu de l'application mobile, (tel que la création de compte, ou bien la suppression de compte). La figure 3.2.18 montre la page de gestion firebase functions.

Afin de configurer firebase functions il faut:

1. Procurer le code source du projet functions qui est dans notre répertoire github sous la branche functions.
2. Remplacer le fichier de configuration admin-service.json par le fichier procurer du nouveau projet. La figure 3.2.17 montre l'emplacement du fichier.

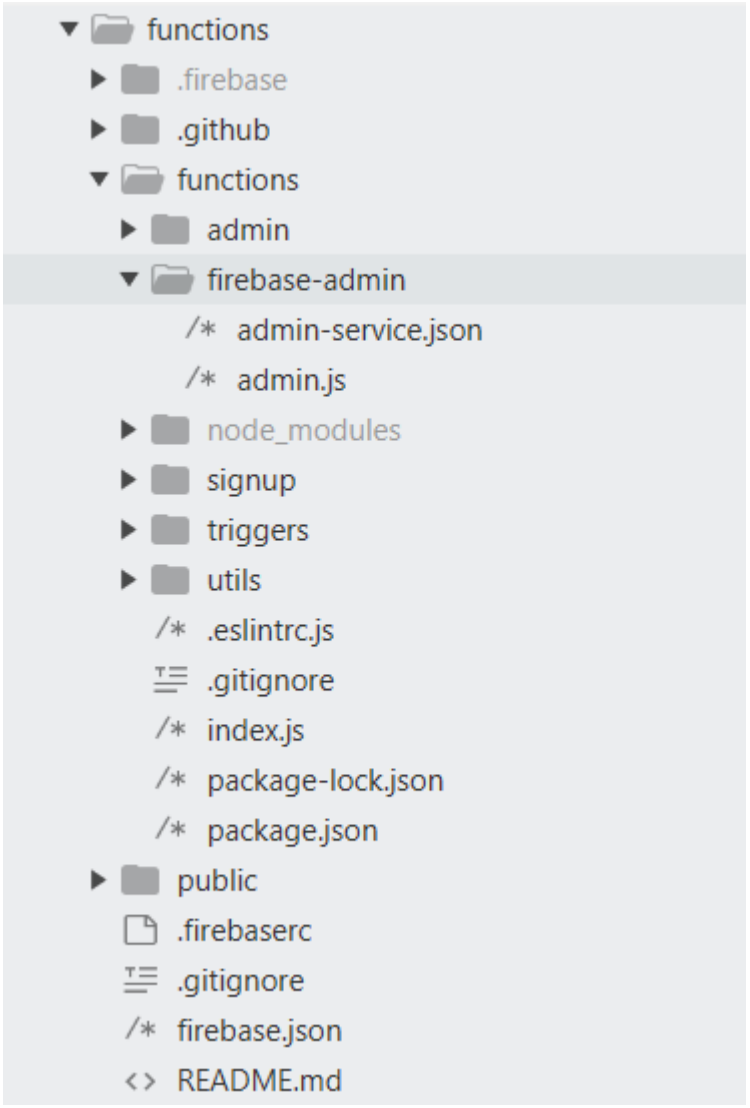


Figure 3.2.17: fichier admin-service.json

3. déployer le code en faisant un appel à `npm run deploy` dans le dossier functions.

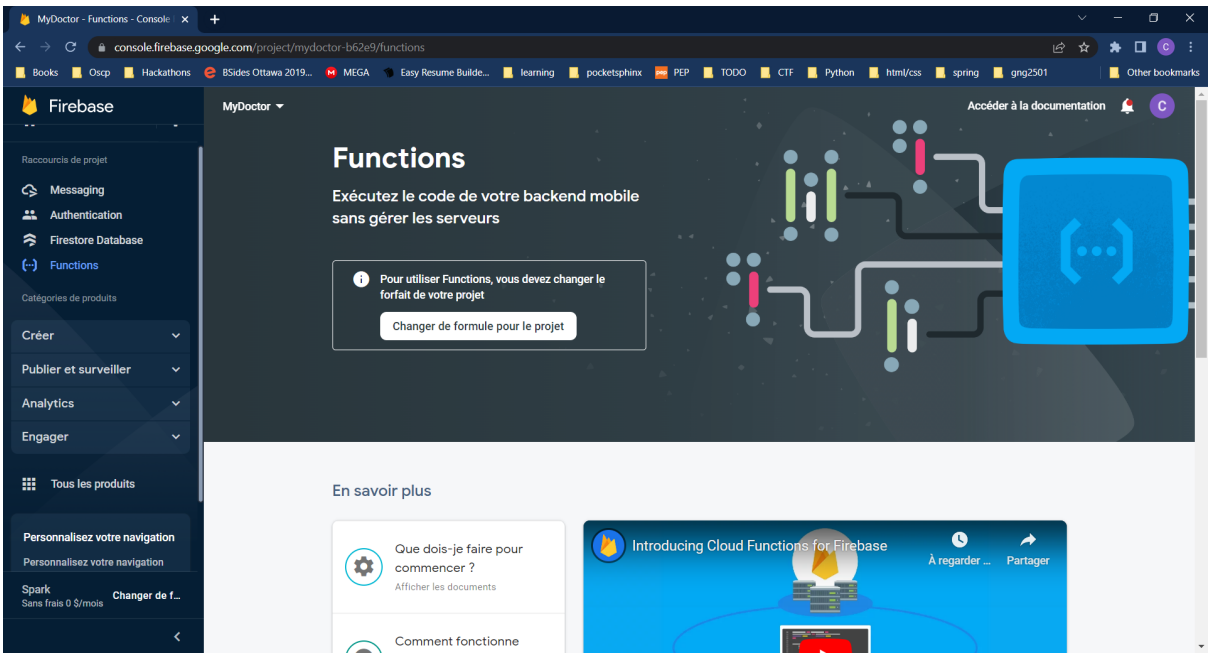


Figure 3.2.18: configuration firebase functions

Firebase messaging

Firebase messaging est le service responsable des notifications et est le plus compliqué à configurer. La configuration inclut:

La gestion de campagne de test

la gestion de campagne de test explique par les figures 3.2.19 a 3.2.24, cette section est important pour vérifier que les abonnés reçoivent les notifications

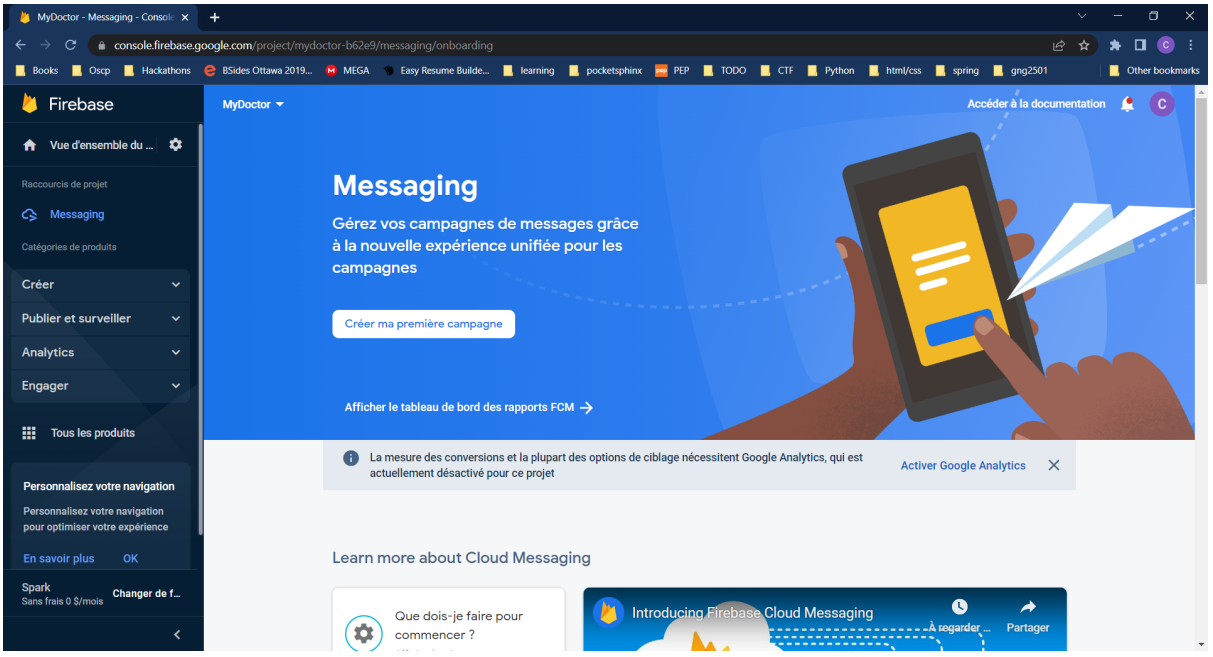


Figure 3.2.19: configuration firebase messaging: compagne de test etape 1

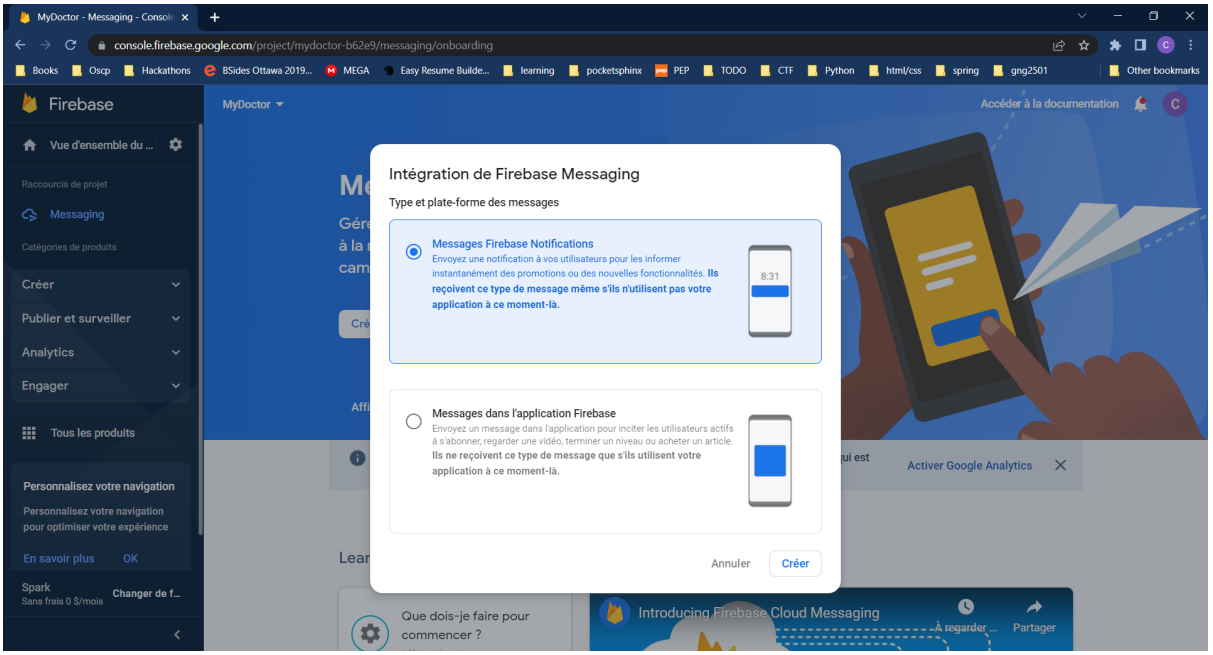


Figure 3.2.20: configuration firebase messaging: compagne de test etape 2

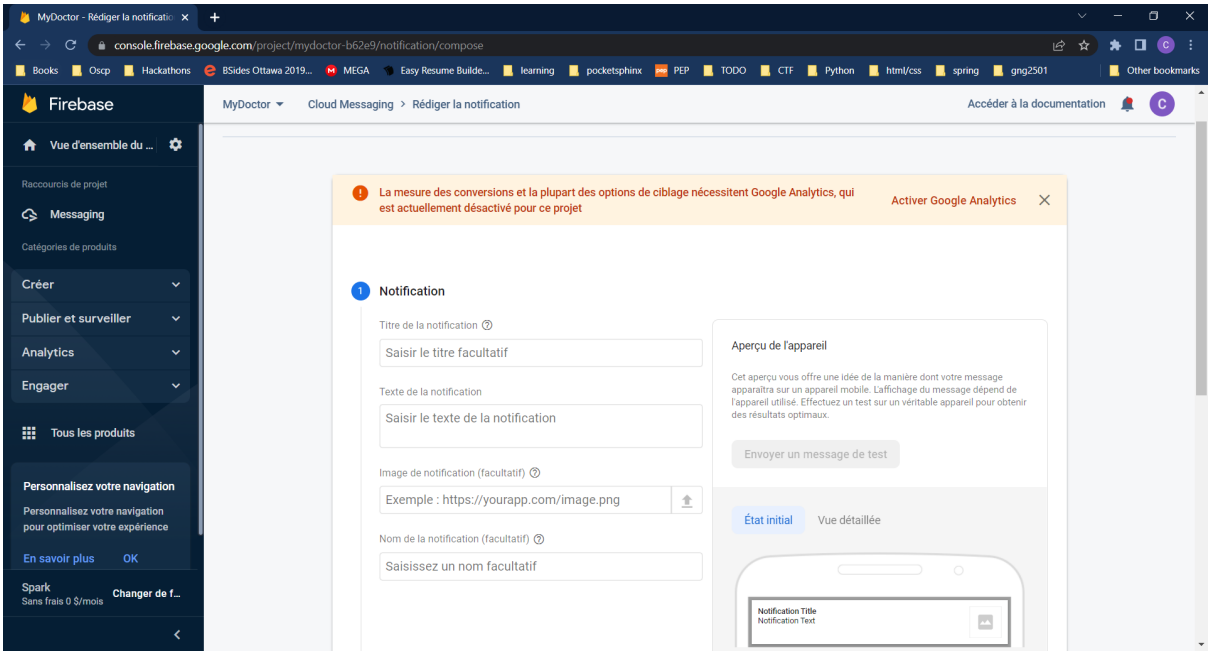


Figure 3.2.21: configuration firebase messaging: compagne de test etape 3

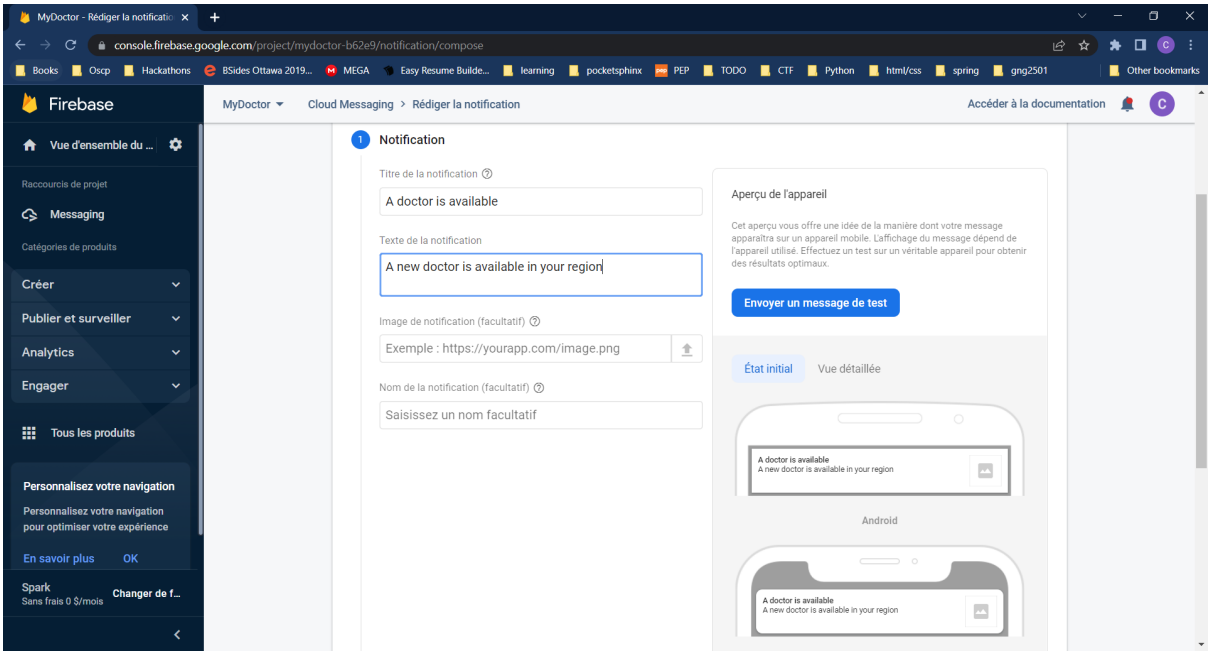


Figure 3.2.22: configuration firebase messaging: compagne de test étape 4

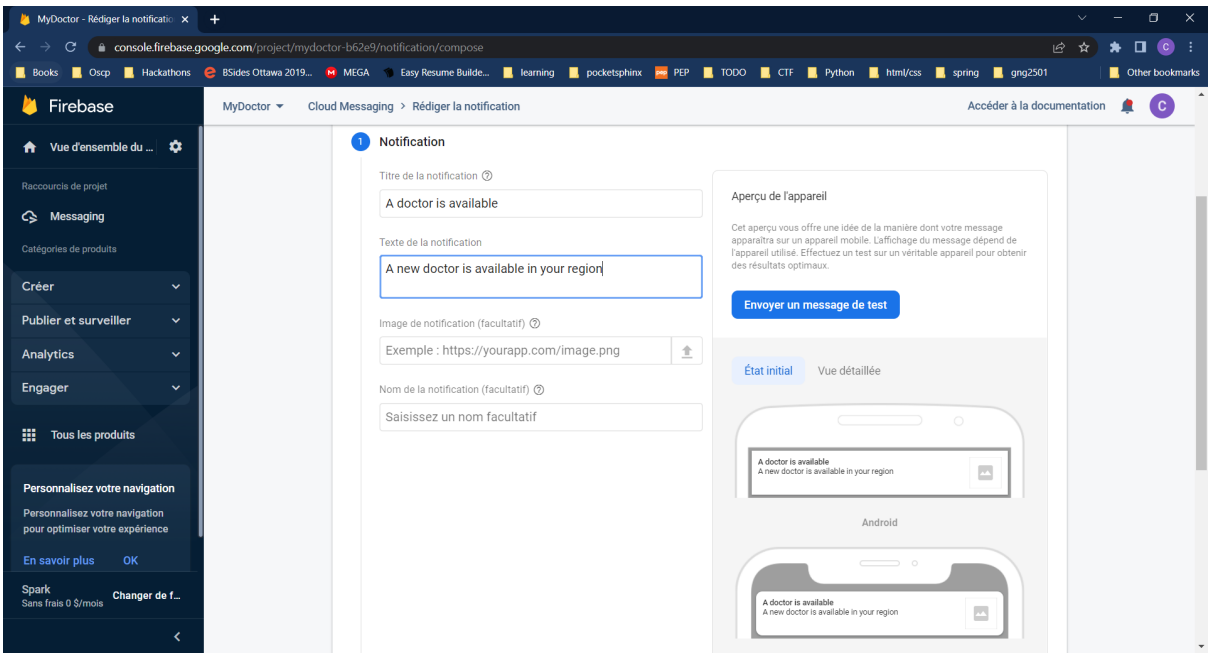


Figure 3.2.23: configuration firebase messaging: compagne de test etape 5

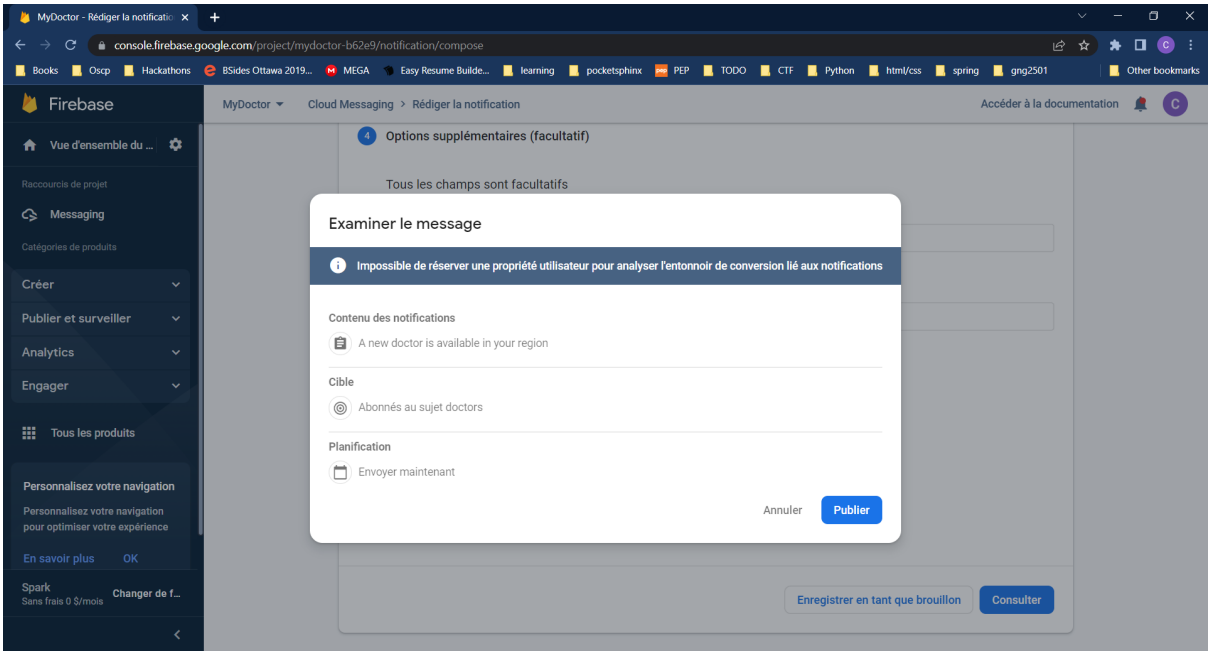


Figure 3.2.24: configuration firebase messaging:compagne de test etape 7

La gestion de la permission

la gestion de la permission explique par les figures 3.2.25 a 3.2.29, cette section permet au admin (firebase functions de créer des campagnes et notifier les patients abonnés)

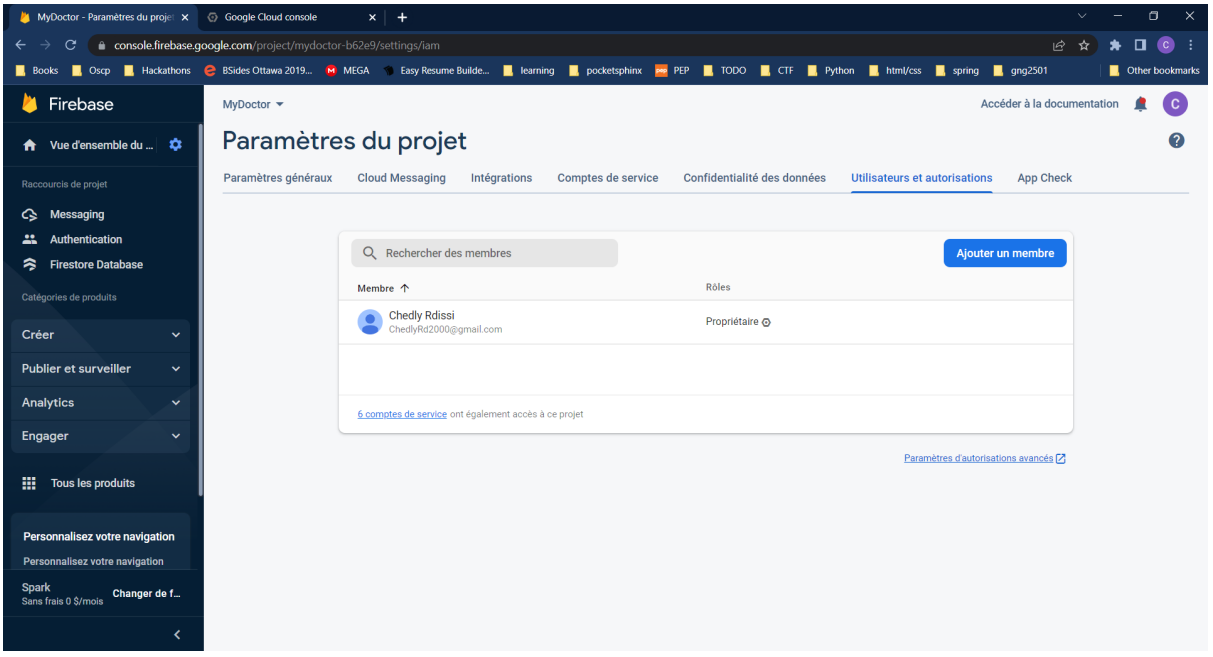


Figure 3.2.25: configuration firebase messaging: gestion de permission etape 1

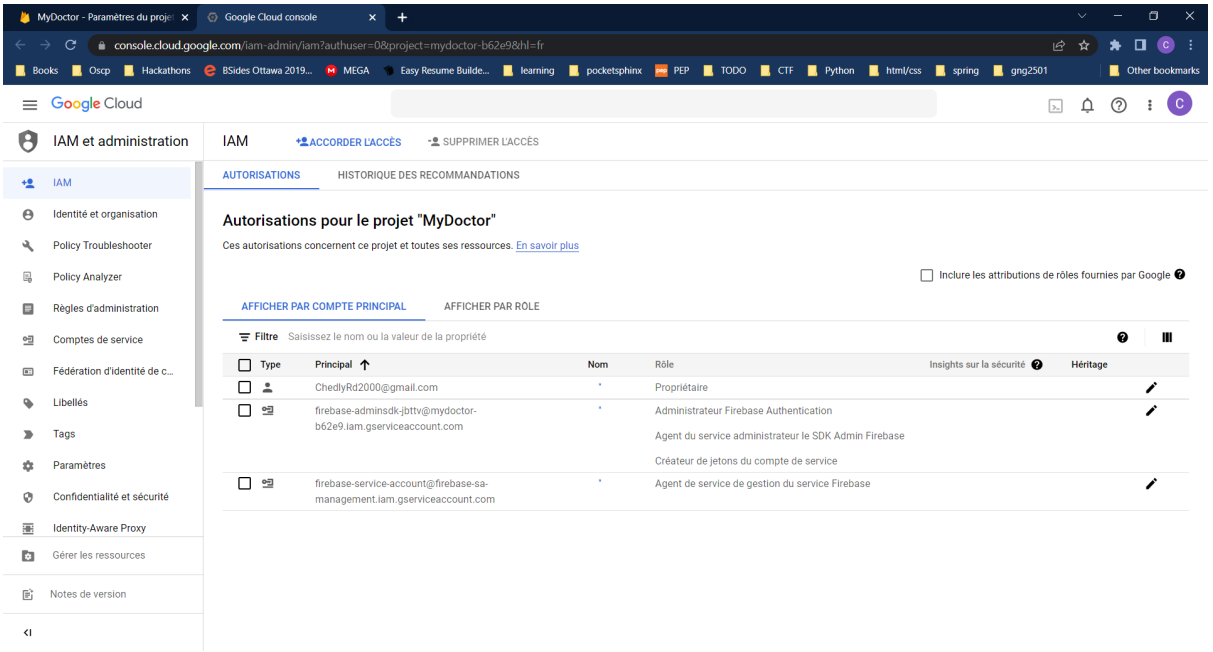


Figure 3.2.26: configuration firebase messaging: gestion de permission etape 2

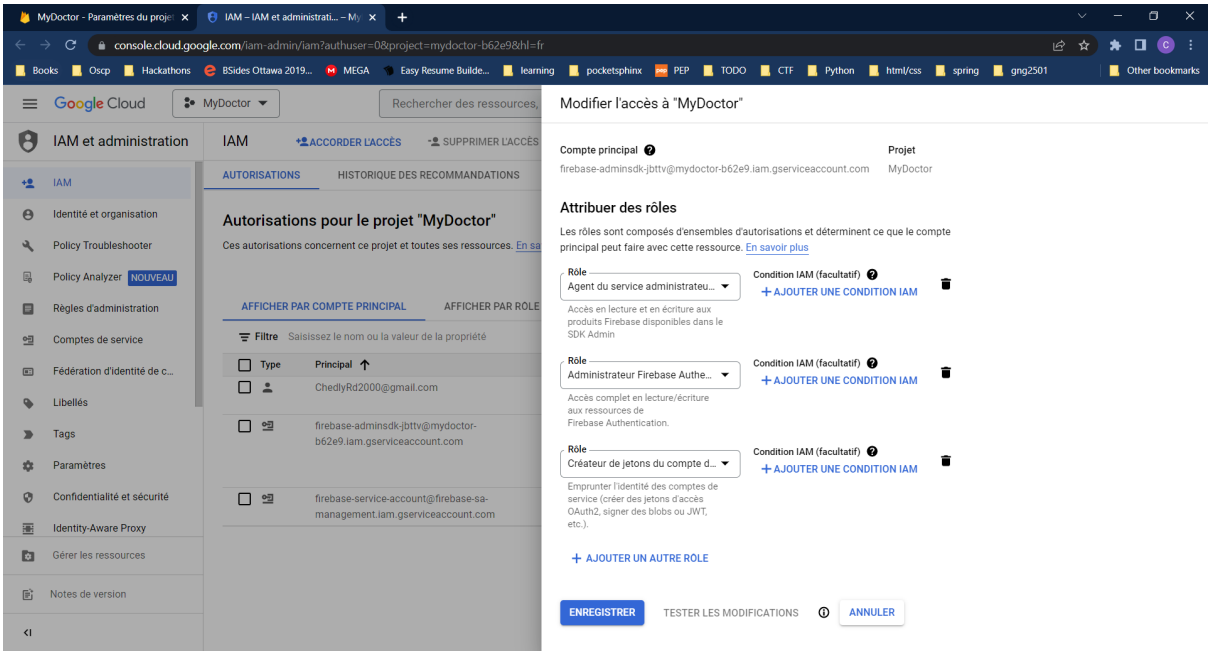


Figure 3.2.27: configuration firebase messaging: gestion de permission etape 3

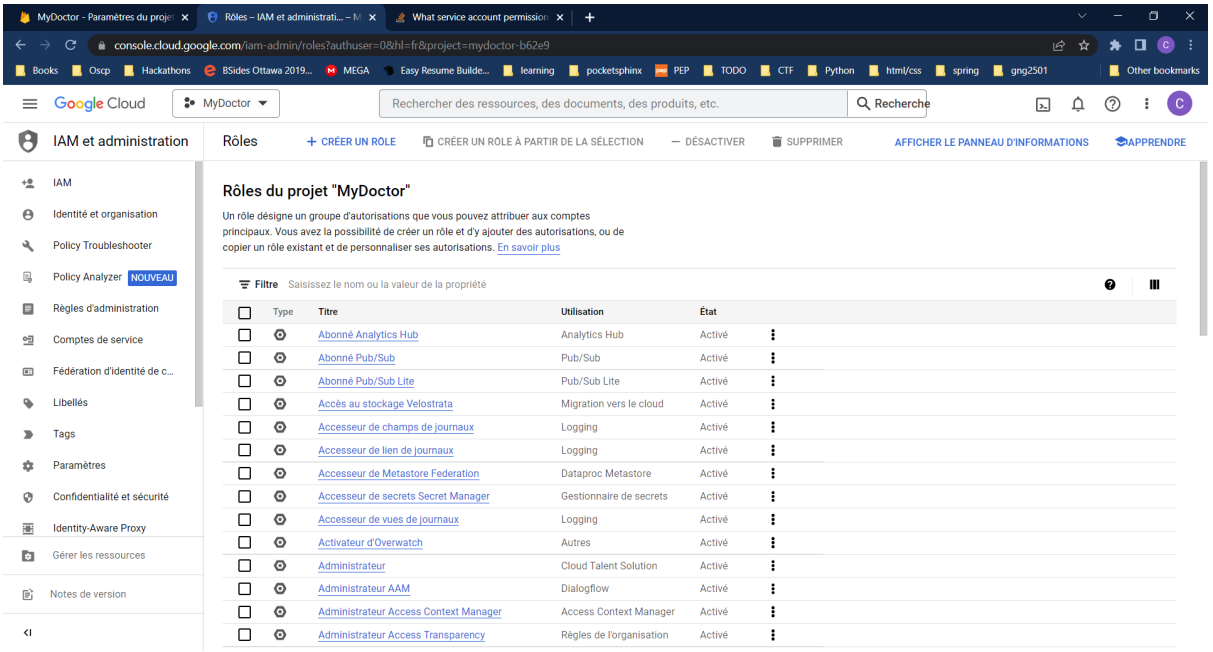


Figure 3.2.28: configuration firebase messaging: gestion de permission etape 4

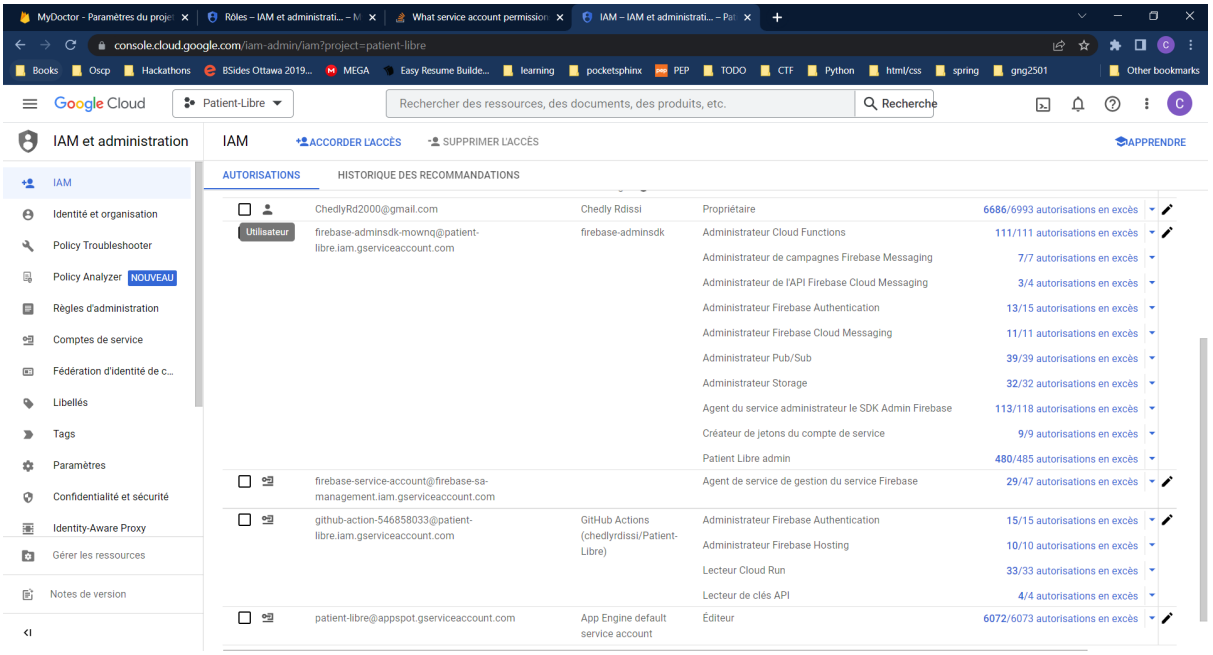


Figure 3.2.29: configuration firebase messaging: gestion de permission etape 5

3.1 Considérations pour la configuration

Afin de pouvoir installer et configurer l'application sur un téléphone intelligent, un espace libre de minimum 80MB est requis. Nous recommandons l'utilisation de téléphone intelligent plus récent afin d'avoir une expérience encore plus agréable.

3.2 Considérations pour l'accès des utilisateurs

Nous comptons avoir trois différents types d'utilisateur pour MyDoctor :

1. **ADMIN :** Utilisé par le département de la médecine familiale de l'Université d'Ottawa, a accès à chacune des fonctionnalités que l'application offre tel que la recherche de docteurs, la création de comptes DOCTEUR, la suppression de compte DOCTEUR et la possibilité de modifier les comptes DOCTEUR si nécessaire.

2. **DOCTEUR :** Utilisé par les docteurs de famille, il permet aux docteurs de créer un compte et de changer leur disponibilité lorsqu'ils sont à la recherche de nouveaux patients**.** S' ils le désirent, ils peuvent faire une recherche de docteurs tout comme l'utilisateur ADMIN et le utilisateurs PATIENT.

3. **PATIENT :** Utilisé par les membres de la société qui désirent trouver un médecin de famille, ils ne nécessitent pas la création d'un compte pour son fonctionnement. Les utilisateurs ont seulement besoin d'entrer leur adresse et un rayon de recherche afin de trouver les médecins disponibles dans leur secteur. Ils ont aussi la possibilité de s'abonner à ce secteur pour la prochaine fois qu'un médecin est disponible.

3.3 Accéder/installation du système

Afin d'accéder à MyDoctor, il suffit de télécharger l'application sur le magasin électronique respectif du téléphone; l'apple store pour IOS et le google play store pour android. Une fois l'application téléchargée, il suffit de lancer l'application en appuyant dessus. Dans le cas où un docteur nécessite une réinitialisation de leur mot de passe, il doit contacter le département de la médecine familiale de l'Université d'Ottawa afin que le compte administrateur puisse changer le mot de passe.

3.4 Organisation du système & navigation

- Gestion de place libre : Ce sous-système s'occupe des tâches de création, modification et suppression de disponibilités des docteurs

- Recherche : Ce sous-système effectue la recherche des espaces libres dans un rayon spécifié, à partir de l'adresse de l'utilisateur
- Notification : Ce sous-système s'occupe de la notification des utilisateurs
- Interface Usager : C'est l'interface graphique du système, utilisé par les docteurs et les utilisateurs
- Gestion de compte : Ce sous-système s'occupe de la création, modification et suppression des comptes et l'authentification des docteurs

p. accueil -> compte (log in/sign up- compt doc ou compt admin - patient)

compte doc -> profil

compte admin -> sign up code -> list doc. -> liste candidat / candidats rejetes

patient

3.5 Quitter le système

Afin de quitter l'application correctement, il suffit de procéder comme toute autres applications sur votre téléphone intelligent, que ce soit en appuyant sur le bouton «home» ou en glissant vers le haut.

4 Utiliser le système

Les sous-sections suivantes fournissent des instructions détaillées, étape par étape, sur la façon d'utiliser les diverses fonctions ou caractéristiques de MyDoctor.

4.1

Utilisation pour les patients :

Comme le montre les images ci-dessous, l'utilisation est assez simple. L'utilisateur n'a qu'à se rendre sur l'application et faire une recherche pour trouver des docteurs, s'il y en a (fig 4.1.3). Si les critères de recherches ne correspondent a aucun docteur, il sera plutôt rendu sur une page le lui signifiant (fig 4.1.4).

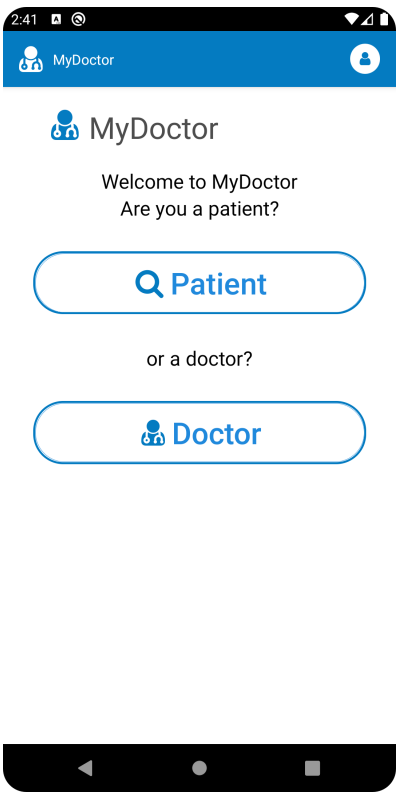


Figure 4.1.1

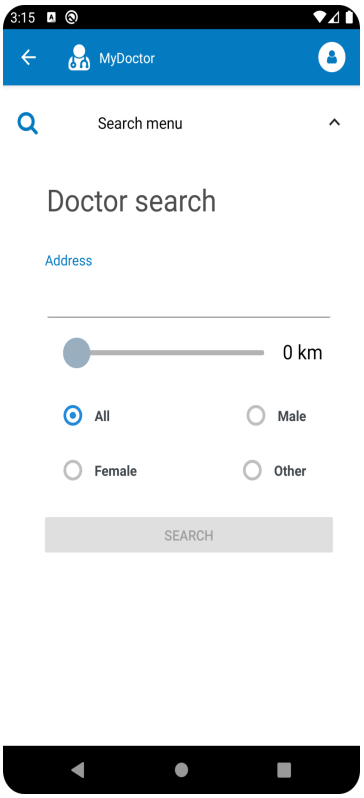


Figure 4.1.2

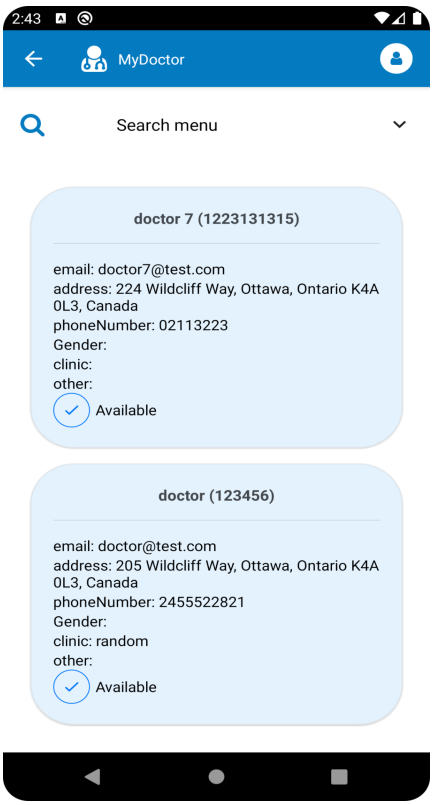


Figure 4.1.3

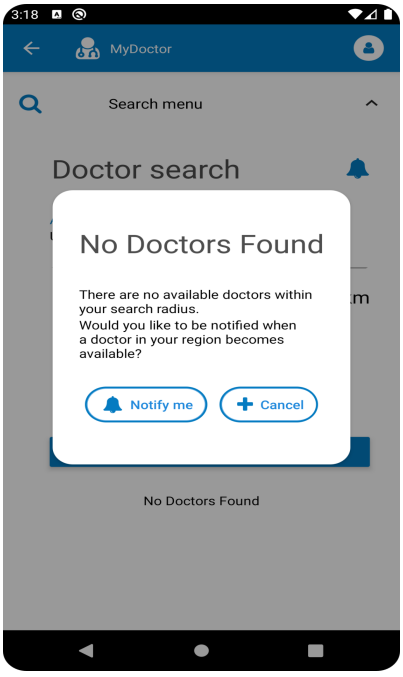


Figure 4.1.4

Utilisation pour les docteurs :

Pour un docteur, il faut sélectionner la section “doctor” et ensuite remplir ses informations(fig 4.1.6 et 4.1.7) ou créer un compte(fig 4.1.8).
Ensuite, le médecin se retrouve sur son profil ou il peut entrer ou modifier ses informations(fig4.1.10)

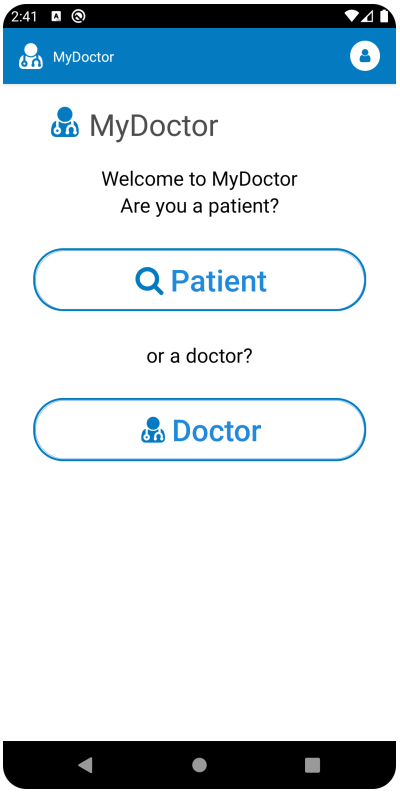


Figure 4.1.5

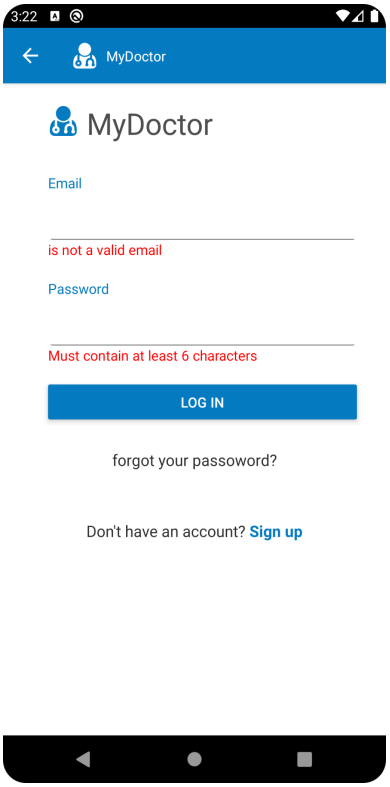


Figure 4.1.6

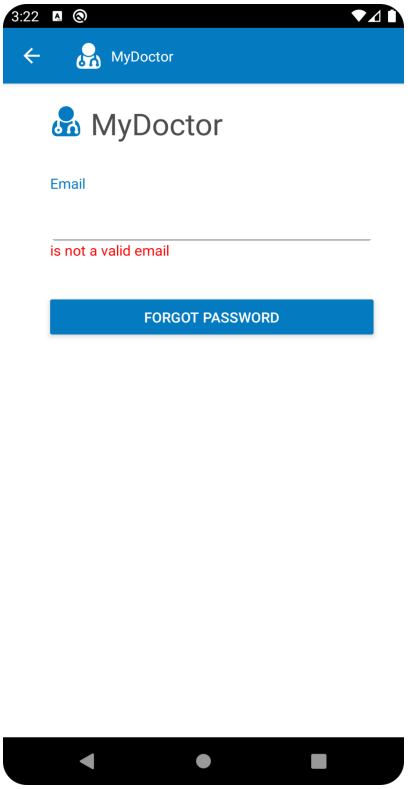


Figure 4.1.7

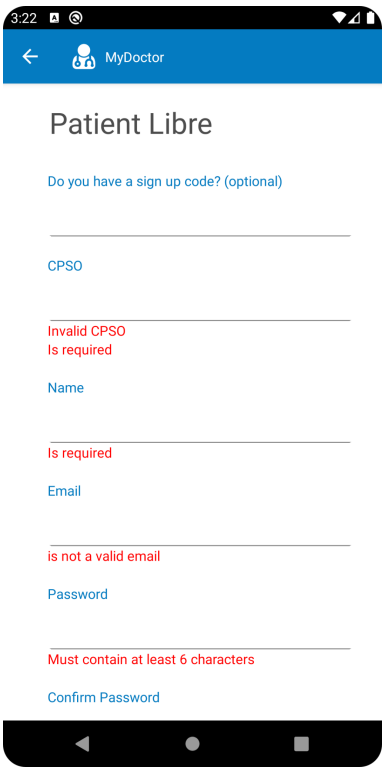


Figure 4.1.8

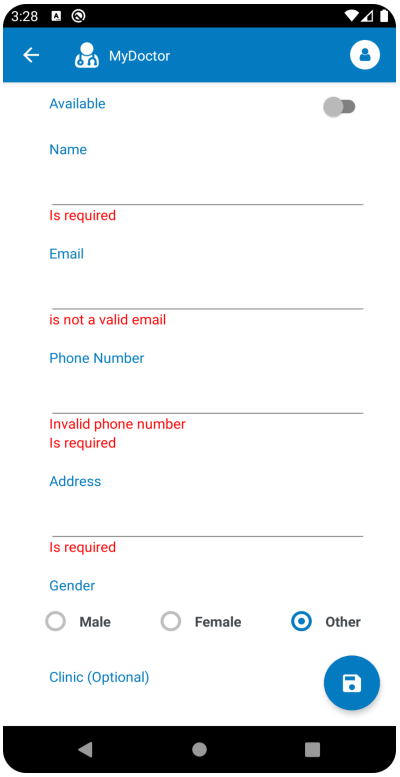


Figure 4.1.9

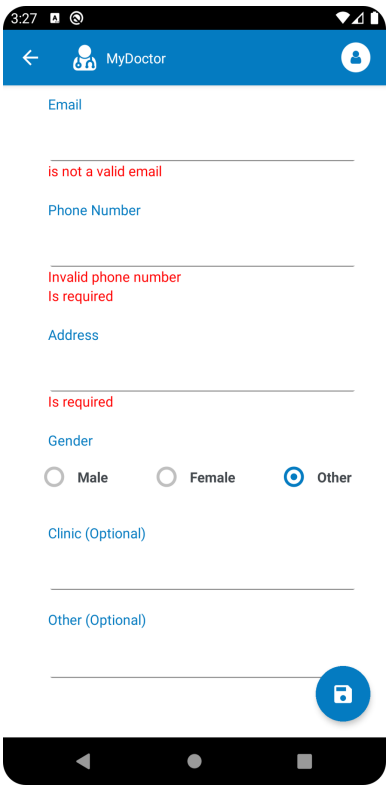


Figure 4.1.10

Utilisation pour l’administrateur :

Pour ce qui concerne l'administrateur, la connexion au profil se fait de la même façon que pour un docteur. C'est-à-dire qu’il doit entrer ses informations personnelles dans la section “docteur”. Il peut ensuite rechercher un médecin spécifiquement pour faire des modifications ou en enregistrer un nouveau(fig 4.1.14 - fig 4.1.18).

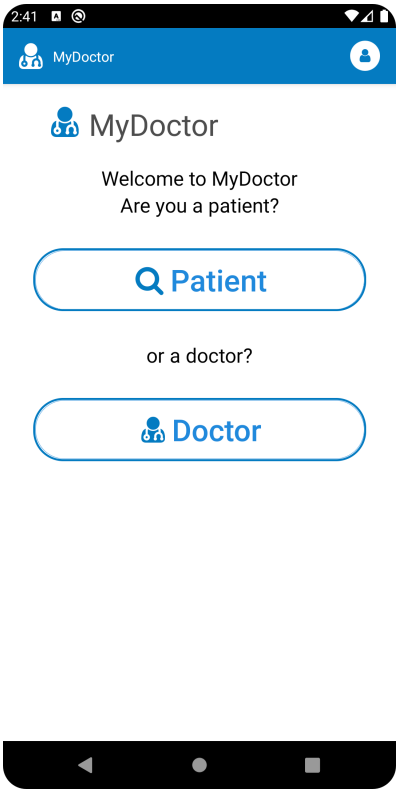


Figure 4.1.11

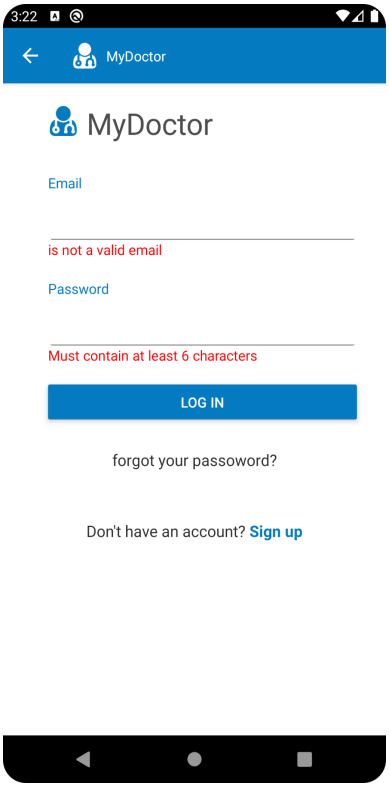


Figure 4.1.12

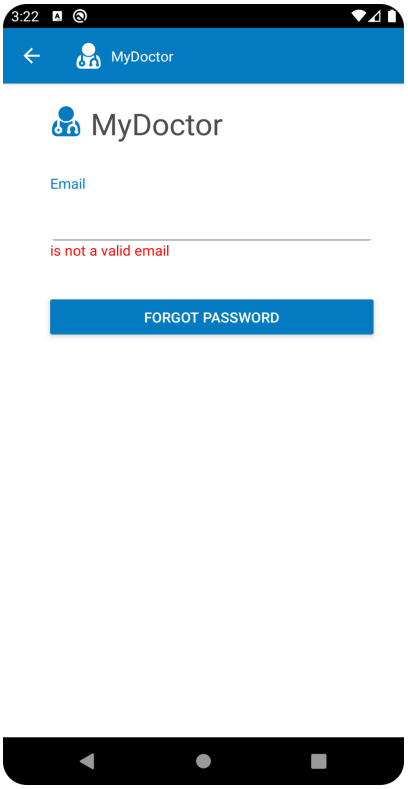


Figure 4.1.13

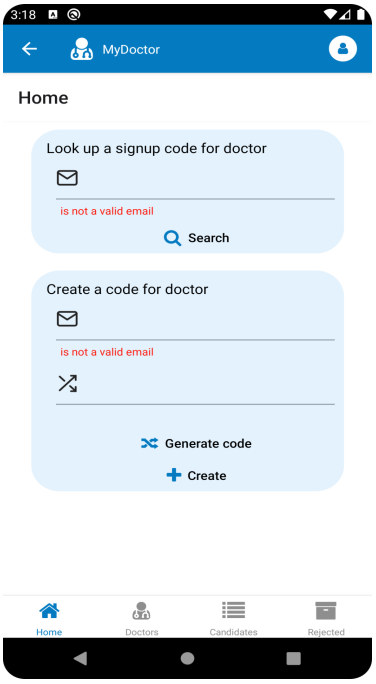


Figure 4.1.14

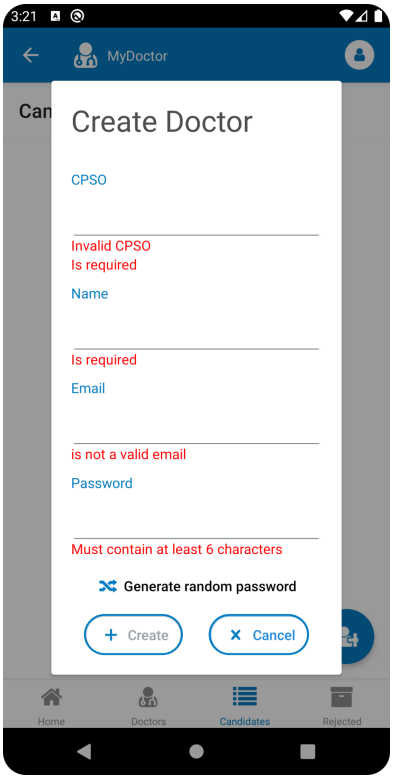


Figure 4.1.15

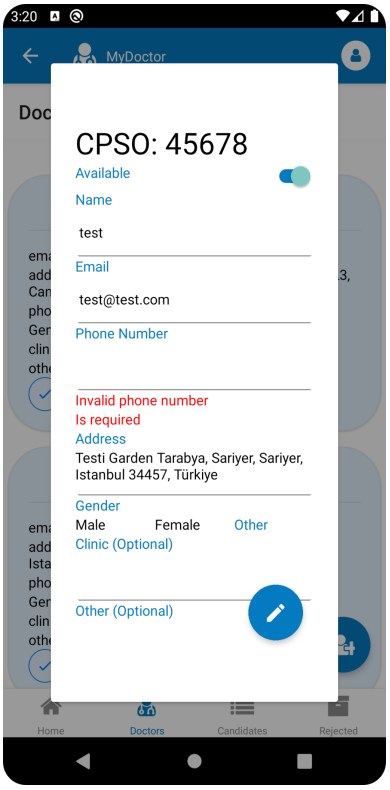


Figure 4.1.16

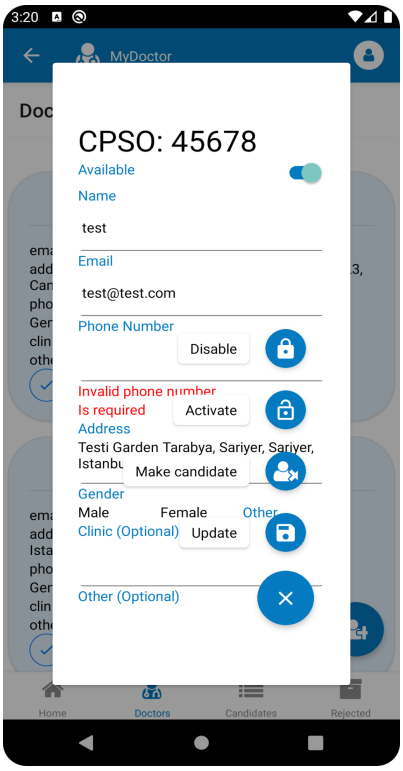


Figure 4.1.17

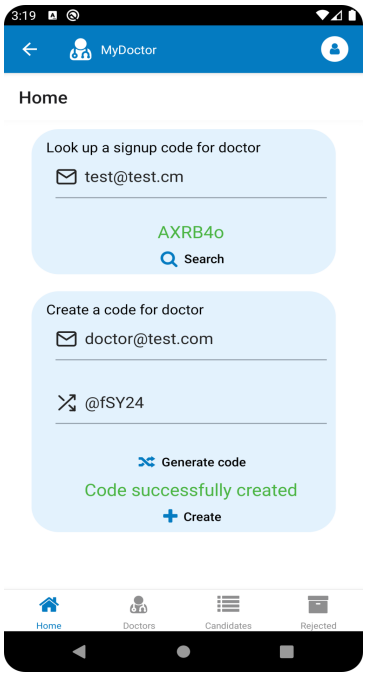


Figure 4.1.18

5 Dépannage & assistance

5.1 Messages ou comportements d’erreur

Les messages d’erreurs les plus communes seront avec les entrées de l’utilisateur, soit pour une adresse invalide, l’oublie d’une entrée nécessaire lors de la création d’un compte docteur et l’entrée d’un numéro de téléphone ou d’une adresse courriel invalide (Voir Figure 4.1.14 et 4.1.15). Ces derniers sont facilement résolus en entrant des données valides. D’autre part, nous avons des messages d’erreurs qui demandent l'assistance d’une personne spécialisée avec les logiciels afin de régler des messages d’erreurs concernant le «back-end» qui ne connect pas à Firebase et des bugs introduits par des programmeurs extérieurs.

5.2 Considérations spéciales

Lors de dépannage, l’analyse des «bugs» récoltés sera très importante. La négligence de ces «bugs» pourra mener à une expérience non agréable pour les utilisateurs qui pourrait mener à un déclin d'utilisateurs.

5.3 Entretien

L'entretien régulier qui doit être effectué sur le prototype est la récolte des «bugs» reportés par les utilisateurs ainsi que la correction de ces «bugs». B

5.4 Assistance

L’utilisateur peut obtenir une assistance d’urgence en contactant Chedly Rdissi à chedlyrd2000@gmail.com

6 Documentation du produit

6.1 Design du prototype

6.1.1 React Native

Afin de satisfaire le besoin du client de viser le plus d'utilisateurs possibles, notre application utilise React Native qui est un framework open-source offrant la possibilité de créer des applications multiplateforme, donc qui fonctionne sur les appareils Android, IOS, et même web avec le même code. Ceci réduit notre charge de travail vu qu'on n'a pas besoin de créer plusieurs bases de code pour chaque plate-forme/système d'exploitation.

6.1.2 Base de donnée cloud Firebase

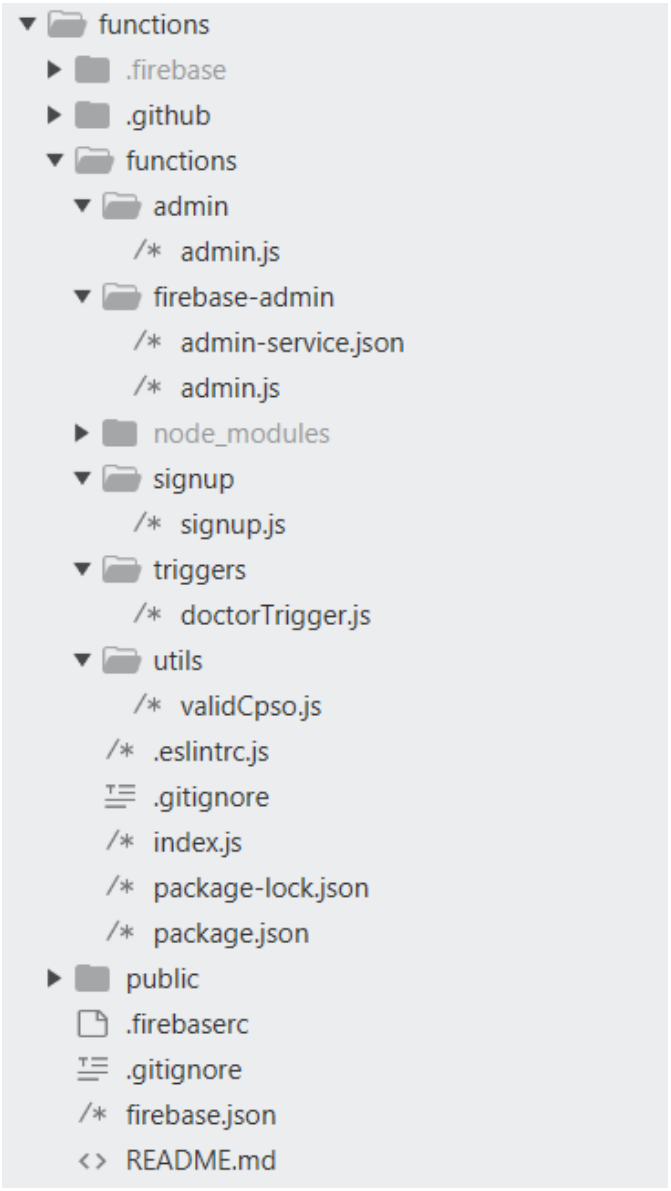
En vue de la contrainte du temps, nous avons choisi d'utiliser des services Cloud au lieu de fabriquer nos propres serveurs backend et gérer les bases de données nous même. Nous avons décidé d'intégrer Firebase dans notre application pour de nombreuses raisons. Premièrement, cela facilite beaucoup l'utilisation en raison que c'est géré par un partie tierce. Deuxièmement, dû à la petite envergure de notre projet, Firebase est complètement gratuit. Troisièmement, nos membres de l'équipe ont déjà utilisé Firebase pour d'autres projets, ce qui élimine le temps d'apprentissage. Finalement, Firebase offre plusieurs services utiles dont l'authentification, une base de données real-time, messagerie, et des fonctions Cloud.

6.1.3 Structure de la base de code

Notre base de code se sépare en deux catégories: les fonctions Firebase et l'application mobile.

Fonctions Firebase
Les fonctions firebase sont générées avec Node.js et sont utilisées pour l'envoie de notification via Firebase messaging et la gestion de comptes (i.e., création, activation, et désactivation des comptes).

Fichiers fonctions Firebase:



Cette structure de fichier inclus 5 sections:

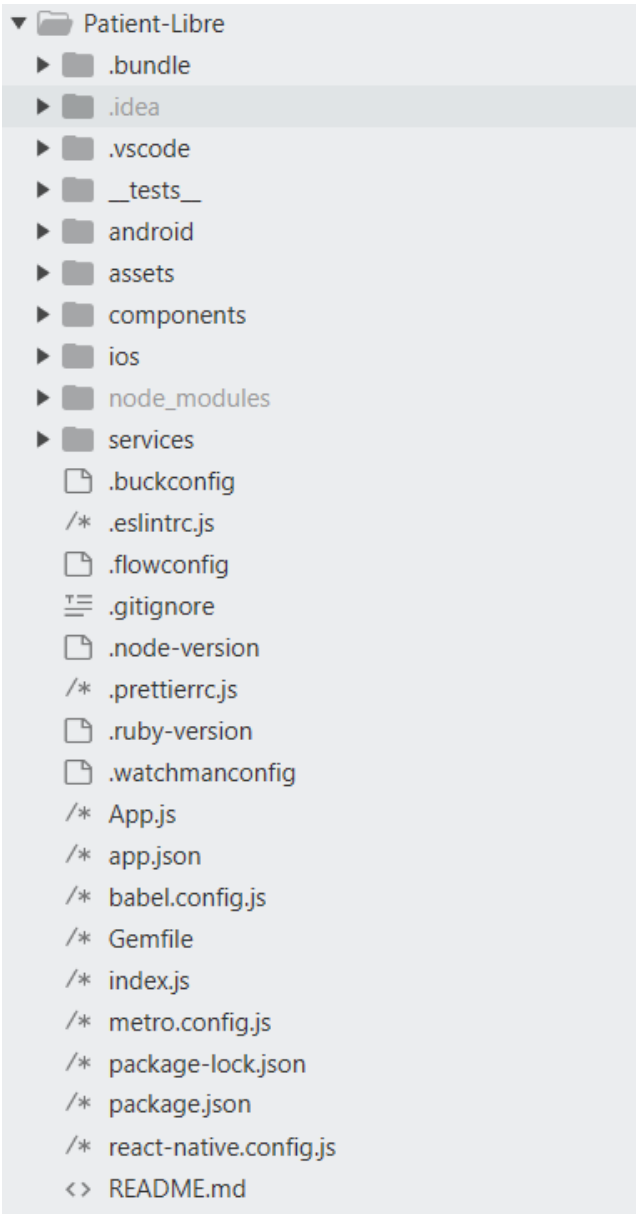
- utils: Valide les cpso des comptes docteurs.
- firebase-admin: Gère les configuration Firebase.
- triggers: Écoute aux modifications des comptes docteurs et notifie les utilisateurs selon certaines conditions.
- admin: Sert à activer, désactiver et supprimer les comptes.
- signup: Gère la création de compte soit par l'admin, ou les docteurs avec ou sans un sign up code.

La communication avec les fonctions firebase se fait avec le client firebase functions de l'application mobile.

Aussi, nous utilisons ESLint pour s'assurer qu'on suit les conventions de JavaScript et améliorer la qualité du code.

L’application mobile
Comme spécifié précédemment, on utilise React Native afin de créer une base de code qui fonctionne dans le plus de plateformes possibles.

Fichiers configurations:



La majorité des fichiers ici sont autogénérés par React Native et gère les configurations. Le fichier index.js est le point d'entrée de notre application et crée l’application ou reçoit les notifications dans l'arrière-plan.

Dans notre application, on utilise react-native-navigation pour gérer deux types de navigations:

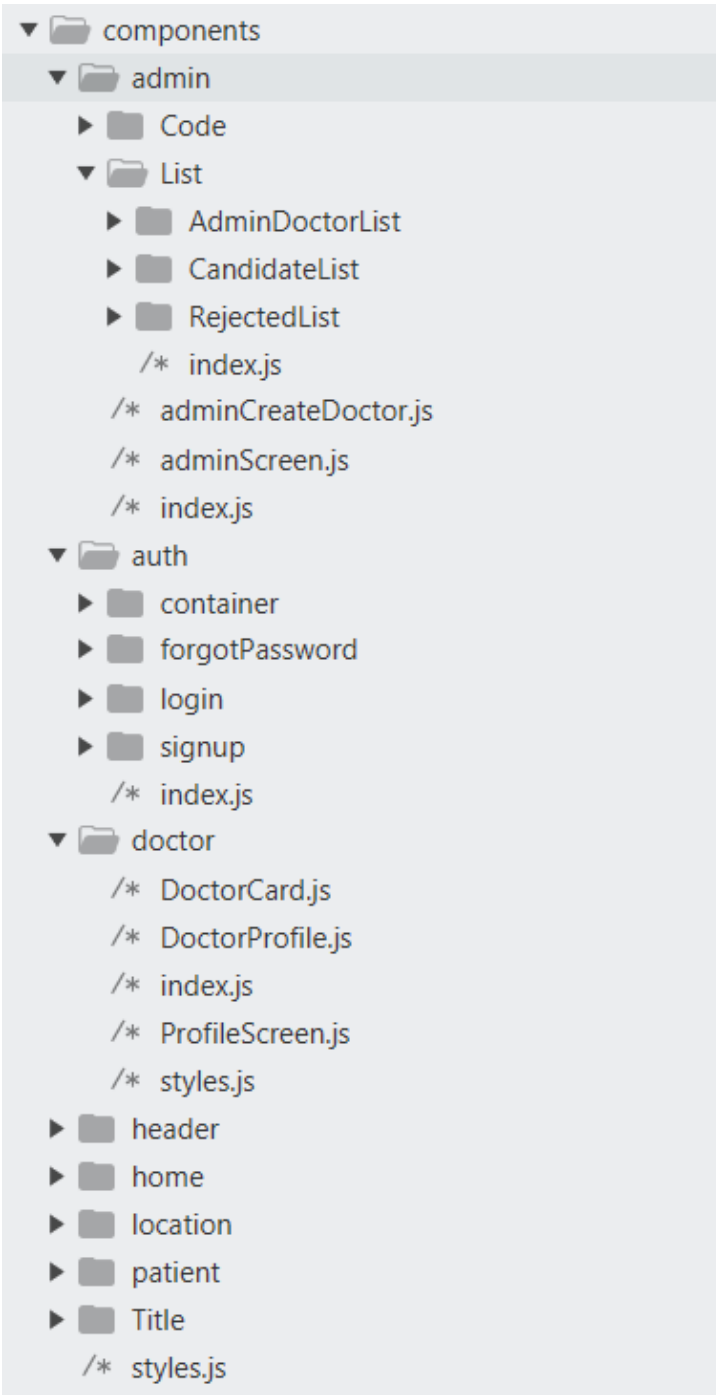
1. stack: Sert à créer des pages au dessus d’autres pages
2. tab: Sert à créer des pages dans la même stack (utilisé seulement dans la section admin)

Nous utilisons le context hook pour assurer la gestion d'authentification. Ce hook dépend de firebase auth qui implémente les fonctionnalités de login, logout, etc.

6.1.4 UI et logique interne

Nous faisons une séparation entre les composantes d'interface usager (UI) et les services qui contiennent des fonctions logiques et utiles.

Composante UI:



Les composantes UI se divisent en:

1. admin: Cette section contient toutes les pages et composantes de admin (gestion de code d'accès, liste de docteurs avec le menu de filtres, liste des candidats, et liste des rejets).
2. auth: Cette section contient les pages de login, signup, et d'oublie de mot de passe
3. doctor: Contient les composantes UI de docteurs, soit les pages ou les cartes utilisées pour afficher les docteurs au patients et l'admin.
4. patient: On a ici la page de recherche que les patients utilise.
5. styles: Contient le code css partagé par les autres composantes.
6. home: Continent la page d'accueil.
7. header: Contient l'entête de l'application.
8. location: C'est la composante pour entrer une adresse.
9. title: Contient le texte titre

Composante logique:

Les services sont les fonctions logiques et utiles de l'application et contiennent les services:

1. admin: Contient la logique de traitement des candidats, des rejets, code d'accès et gestion de comptes.
2. auth: Contient la logique pour login, logout, signup, et oublie de mot de passe.
3. orientation: Contient un hook simple pour detecter si on est en vue portrait ou landscape
4. docteurs: contient la logique de gestion, recherche, et filtres de recherche de docteurs.
5. location: contient la logique de prédiction et de géolocalisation (qui est utilisé pour prédire l'adresse en utilisant l'entrée de l'utilisateur)/
6. navigation: Ce fichier est un wrapper pour la navigation de react-native-navigation et regroupe le code de navigation.
7. subscription: qui gère le stockage d'adresse de patient lors de l'abonnement et l'affichage de notification
8. contraintes: Contient les contraintes utilisé par validate.js pour valider l'entrée utilisateur

9. outils: Contient du code d'utilité pour calculer la distance entre 2 adresses, et générer du code aléatoire.

Notre application utilise en majorité des composantes préexistantes qui ont été testées et validées.

6.2 Sous-système 1 du prototype

6.2.1 NDM (Nomenclature des Matériaux)

Ci dessous, nous avons la nomenclature des matériaux du fichier package.json qui utilise les paquets:

composante	version	Description	cout ($)	coût réel ($)
react-native	0.70.2	librairie avec des composantes native	0	0
react	18.1	dépendance de react-native	0	0
react-navigation	4.4.4	gère la navigation dans l'application	0	0
validate.js	0.13.1	utilisé pour valider les données	0	0
rneui	4.0.0-rc.6	offre des composantes native utiles	0	0
react-native-firebase	15.7.0	service pour communiquer avec firebase	0	0
notifee/react-native	7.0.4	librairie pour gérer les notifications	0	0

6.2.2 Liste d'équipements

Le seul équipement nécessaire pour la «fabrication» du logiciel est un ordinateur; portable ou non.

6.2.3 Instructions

Comme mentionné et démontré à la section 3, notre conception de notre logiciel a inclut le travail sur l'UI et les fonctionnalités à chaque étape vu qu'ils vont main dans la main. La section 3 explique en profondeur l'installation de chaque fonctionnalité pour le logicielle à l'aide d'image. Tout d'abord, nous avons fait l'initialisation de react-native ainsi que de firebase. Nous avons travailler sur un UI de base afin de pouvoir y ajouter des fonctionnalités et avoir une application qui fonctionne. Par après, nous avons implémenté la fonction de recherche ainsi que la navigation simple entre la page d'accueil, la page de création de compte docteurs, la page de recherche et la page pour se connecter à un compte docteur. Nous avons implémenté la fonction de notification qui a mené à des

changements au niveau de l'UI. Finalement, nous avons finalisé l'UI et nous avons ajouter des filtres de recherche lorsque le compte ADMIN regarde à travers les comptes DOCTEUR.

6.3 Essais & validation

Nous avons réalisé plusieurs sessions d'essais pour valider notre conception finale. Durant ces sessions d'essais, nous avons fait référence à nos spécifications cibles qui fut déterminées durant le livrable de projet B. Ces essais consistaient majoritairement de quelques membres de l'équipe qui essayaient chacune des fonctionnalités de notre prototype. De plus, nous avons aussi fait des essais des messages d'erreurs quand l'utilisateur de rentres pas une entrée valide pour le logiciel. Si-dessous, nous avons notre tableau de spécifications cibles établit lors du projet de livrable B qui inclu aussi les valeurs réelle obtenue de notre prototype final.

Critères de conception	Relation à la valeur (<, =, >)	Valeur ciblée avec unités	Méthode de vérification
Exigences fonctionnelles			
- L'envoie d'une alerte	=	OUI	Essai
- Rayon de recherche	>=	50 km	Essai
- Temps entre l'entrée et l'alerte	>	1 min	Essai
- Mise en relation médecin patient	=	OUI	Essai
- Coût	=	0 $	Essai
Contraintes			
- taille de l'application	<	7Mb	lecture de la taille du projet compile
Exigences non-fonctionnelles			
- Compatible avec les appareils mobiles	>	80%	vérification avec rapport donné par app/play store
- Délais de notification	<	1h	essai
- Temps de démarrage	<	5s	essai
- Compatibilité avec plugin de navigation	>	80%	vérification avec deux systèmes d'exploitation

Tableau 6.2.1

7 Conclusions et recommandations pour les travaux futurs

Pour conclure, ce manuel d'utilisation, comporte toutes les informations nécessaires sur la conception de notre produit, et a pour but de permettre à quiconque de pouvoir continuer sur notre piste. Ce projet, nous a été bénéfique sur plusieurs plans. Il nous a non seulement permis d'améliorer notre travail d'équipe et de gérer plus efficacement le stress généré par la charge de travail, mais aussi permis d'utiliser toutes les méthodes de conception apprise en cours et de pouvoir appliquer cela lors de nos multiples rencontres avec le client. En ce qui concerne notre prototype, il a fallu valider plusieurs étapes avant de confirmer qu'il était bien fonctionnel. C'est selon cette perspective que dans un premier temps, nous avons initialisé react native et fire base avant de travailler sur l'interface d'utilisateur (UI). Ce fut les premières étapes qui nous ont lancés dans la construction de MyDocteur. C'est sur ces faits, que les pistes les plus productives que nous suggérons, c'est de développer le UI en y ajoutant plus de fonctionnalités, et corriger les potentiels "bug" qui seront rapportés par les utilisateurs. Enfin, ce que nous ferions si nous avions quelques mois de plus pour travailler sur le projet, c'est tout d'abord de mettre en place une version en français de l'application, pour les personnes qui ne parlent pas anglais, ensuite configurer l'application afin qu'elle soit compatible avec IOS, et lui créer un site web similaire où les patients peuvent prendre rendez-vous.

8 Bibliographie

Insérez votre liste de références ici.

APPENDICES

9 APPENDICE I: Fichiers de conception

Résumez la relation de ce document avec d'autres documents pertinents. Fournir des informations d'identification pour tous les documents utilisés pour arriver à et/ou référencés dans ce document (par exemple, documents connexes et / ou d'accompagnement, documents préalables, documentation technique pertinente, etc.).

Inclure tous les fichiers de conception dans MakerRepo. Aussi fournir le lien MakerRepo pour votre projet.

Table 3. Documents référencés

Nom du document	Emplacement du document et/ou URL	Date d'émission