

TOOBIB :
UNE PLATEFORME POUR
UNE SANTE CONNECTEE,
ETHIQUE ET ACCESSIBLE
A TOUS.

STAGE DU **15/04/2024** AU **07/06/2024**

CALLET Alexis

Dossier de projet en vue de la
préparation au titre de développeur
web et web mobile.

REMERCIEMENTS

Mon stage chez InterHop a été une expérience riche en apprentissages et en défis stimulants.

J'ai pu appliquer mes connaissances techniques et acquérir de nouvelles compétences dans un environnement innovant et porteur de sens.

Ma contribution a permis de faire évoluer la plateforme Toobib et d'améliorer l'expérience utilisateur pour les patients et les praticiens de santé.

Je tiens à remercier chaleureusement les personnes qui m'ont encadré lors de ce stage, Quentin PARROT, Adrien PARROT et Corentin FORLER pour leur soutien constant et leur engagement à porter les convictions et les missions d'InterHop.

Je souhaite également remercier Alin HERCIU, stagiaire de l'ENI avec qui travailler a été un plaisir durant cette période de stage.

TABLE DES MATIERES

1. QUI SUIS-JE ?	3
1.1 MON PARCOURS PROFESSIONNEL	3
1.2 POURQUOI CETTE RECONVERSION ?	3
2. L'ENVIRONNEMENT DU STAGE	4
2.1 PRESENTATION D'INTERHOP	4
2.2 PRESENTATION DU POSTE ET DE L'ENVIRONNEMENT TECHNIQUE	5
3. PRESENTATION DES MISSIONS	8
3.1 COMPETENCES DU REFERENTIEL COUVERTES PAR LE PROJET	8
3.2 LES MISSIONS PRINCIPALES DU PROJET TOOBIB	9
4. INTEGRATION DE KEYCLOAK ET CREATION D'UN THEME PERSONNALISE	10
4.1 POURQUOI CE CHOIX ?	10
4.2 INTEGRATION DE KEYCLOAK	11
4.3 LA SECURITE APPOREE PAR KEYCLOAK POUR TOOBIB	12
4.4 CREATION DU THEME PERSONNALISE	13
5. DEVELOPPER LE CONTENU DU SITE	18
5.1 BESOINS DE TOOBIB	18
5.2 SPECIFICITES TECHNIQUES	19
5.3 CREATION DES PAGES	20
6. DEVELOPPEMENT D'API POUR TOOBIB	31
6.1 BESOINS DE TOOBIB	31
6.2 SPECIFICITES TECHNIQUES	32
6.3 CREATION DE L'API POUR LA RECHERCHE DE PRATICIENS DE SANTE	33
6.4 CREATION DE L'API POUR LA GESTION DES IMAGES	38
7. MISE EN PLACE DE TESTS	39
8. REDACTION DE LA DOCUMENTATION	40
9. CONCLUSION	41
10. ANNEXES	42

1. QUI SUIS-JE ?

Je m'appelle Alexis CALLET, j'ai décidé de mener ma reconversion professionnelle depuis maintenant plus d'un an.

La construction de mon projet professionnel m'a mené à suivre la formation de Développeur Web et Web Mobile de l'ENI.

1.1 MON PARCOURS PROFESSIONNEL

Après l'obtention de mon baccalauréat professionnel de vente, ayant apprécié mes expériences professionnelles au sein du groupe La Poste lors de multiples périodes de stage, j'ai mis mon premier pied dans le monde du travail dans l'entreprise Calligraphy Print, une imprimerie, en tant qu'agent de conditionnement.

À la suite de ça, une période sans activité professionnelle à cause de problèmes de santé m'a permis de m'épanouir de façon extra-professionnelle.

J'ai pu développer mon intérêt déjà présent et mes compétences pour le numérique et l'informatique par le biais d'initiations à l'informatique, de dépannages et plus particulièrement de gestion de communauté, de bases de données, de game design et de développement en ayant participé à un projet de modding d'un jeu vidéo en ligne avec un ami, et ce, pendant plusieurs années.

En 2019, une opportunité se présente à moi, me permettant de rejoindre l'entreprise Materiel.NET en tant qu'assembleur, où mes missions étaient d'assembler et de vérifier le bon fonctionnement d'ordinateurs sur-mesure et de séries.

1.2 POURQUOI CETTE RECONVERSION ?

Après ces quelques années chez Materiel.NET et une impossibilité à y entrevoir un futur en raison du manque d'activité provoqué par le contre-coup du COVID.

J'ai décidé de construire un projet professionnel autour de mon intérêt principal, l'informatique, mais surtout autour de mes compétences, il a donc été naturel pour moi de suivre la formation de Développeur Web et Web Mobile au sein de l'ENI.

2. L'ENVIRONNEMENT DU STAGE

2.1 PRESENTATION D'INTERHOP

InterHop est une association française à but non lucratif qui promeut, développe et met à disposition des logiciels libres et open-source pour le secteur de la santé. L'association rassemble des militants pour les logiciels libres et l'utilisation autogérée des données de santé à l'échelle locale.

Les différentes plateformes d'InterHop



Les objectifs d'InterHop

- Informer les usagers (patients et soignants) sur les enjeux du logiciel libre et des données de santé.
- Proposer un hébergement transparent, neutre et solidaire conforme aux normes de santé.
- Développer et installer des solutions alternatives et efficaces dédiées à la santé, comme "Goupile" (formulaire numérique de recueil de données) et "Toobib".
- Construire un cadre associatif de confiance, transparent et éthique autour des données.
- Protéger les données privées de santé des patients pour préserver la confiance et le secret médical.

Toobib : Une Plateforme Éthique pour la Santé Connectée

Toobib, soutenu par InterHop, est une plateforme numérique dédiée à la santé connectée, éthique et accessible. Le projet Toobib vise à améliorer l'expérience utilisateur pour les patients et les praticiens de santé.

Expérience de Stage

Lors de ma période de stage, j'ai eu la chance de bénéficier de l'expérience d'**Adrien PARROT**, ingénieur et médecin, de **Quentin PARROT**, ingénieur, et de **Corentin FORLER**, ingénieur.

Tous trois m'ont accueilli chez InterHop avec un **objectif clair** : contribuer au développement de la plateforme Web « Toobib » visant à améliorer l'accès aux soins de santé, tout en respectant les principes fondamentaux de l'éthique et de la protection des données personnelles.

Lors de ce stage j'ai été sensibilisé à la protection des données par Chantal CHARLOT, déléguée à la protection des données au sein d'InterHop, cette sensibilisation m'a permis de prendre conscience de **l'importance de la communication entre développeurs et délégués à la protection des données**.

2.2 PRESENTATION DU POSTE ET DE L'ENVIRONNEMENT TECHNIQUE

Toutes les missions qui m'ont été confiées ont été réalisées en télétravail, avec un suivi quotidien grâce aux « daily scrums » pour assurer une bonne pratique, réactivité et apprentissage.

Outils utilisés pour le travail en équipe et la communication :

Toutes ces missions ont été réalisées grâce à plusieurs outils :

- IntelliJ IDEA et Microsoft Visual Studio pour l'édition du code.
- Docker pour la mise en place d'un conteneur Keycloak en environnement local.
- Framagit pour la gestion des tickets, la gestion de dépôts Git et le travail en équipe.
- Element plateforme de communication sécurisée utilisée pour les réunions et la messagerie instantanée.

Certains éléments déjà présents au sein du projet ont nécessité l'installation d'un système Linux pour permettre leur exécution. Windows a été privilégié pour les tâches ne nécessitant pas de système Linux.

Langages utilisés :

Pour mener à bien toutes ces missions et assurer la pérennité du projet, j'ai utilisé et manipulé les langages suivants :

- HTML
- CSS
- JavaScript
- Python
- Java

Frameworks et Bibliothèques :

Dans le cadre de l'utilisation de ces langages, les frameworks et bibliothèques suivantes ont été utilisés :

- **Flask (framework Python)**
- **Tailwind CSS (framework CSS)**
- **Next.js (framework JavaScript) permettant l'utilisation de React (bibliothèque JavaScript)**
- **NPM**
 - Gestionnaire de paquets pour JavaScript qui simplifie l'installation et la gestion des bibliothèques et outils JavaScript.
- **Axios**
 - Bibliothèque qui facilite la création de requêtes HTTP dans les applications Web.
- **Formik**
 - Bibliothèque JavaScript pour la création de formulaires flexibles et efficaces avec React, offrant des fonctionnalités telles que la validation de formulaires, la gestion des erreurs et la gestion des entrées.

Outils Externes Utilisés :

Pour améliorer les fonctionnalités du projet, plusieurs API externes ont été intégrées :

- **API Annuaire de santé :**
 - Cette API permet d'accéder à une base de données de professionnels de santé, fournissant des informations détaillées et actualisées sur les praticiens. Elle est essentielle pour le système de recherche de praticiens de santé.

- **API FHIR (Fast Healthcare Interoperability Resources) :**
 - FHIR est une norme pour l'échange électronique d'informations de santé. Elle permet d'assurer l'interopérabilité entre différents systèmes de santé, facilitant ainsi l'échange et l'intégration des données de santé.
- **API BAN du gouvernement français (Base Adresse Nationale) :**
 - La BAN est une base de données gouvernementale qui fournit des adresses géographiques précises et normalisées.
- **Keycloak :**
 - Outil open-source de gestion des identités et des accès, offrant de nombreuses fonctionnalités comme :
 - **Authentification unique (SSO) :** Permet aux utilisateurs de se connecter à plusieurs applications avec une seule identité.
 - **Gestion des accès basée sur les rôles :** Contrôle qui peut accéder à quelles ressources dans vos applications.
 - **Authentification multifacteur :** Ajoute une couche de sécurité supplémentaire aux applications en exigeant plusieurs facteurs d'authentification, tels qu'un mot de passe et un code PIN envoyé par SMS.
- **VMware Workstation :**
 - Logiciel de virtualisation puissant qui permet aux utilisateurs de créer et d'exécuter plusieurs machines virtuelles sur un seul ordinateur physique.

Gestion des données :

Pour la gestion des données, divers outils et technologies ont été utilisés :

- **Langage de base de données : SQL**
- **Système de gestion de base de données : PostgreSQL**
 - PostgreSQL est un système de gestion de base de données relationnelle open-source.
 - PostgreSQL permet également de réaliser, par le biais de fonctions présentes en son sein, de réaliser des calculs permettant la géolocalisation.
- **Outil de gestion : DBeaver**
 - DBeaver est un outil de gestion de base de données universel et open-source qui supporte plusieurs types de bases de données, permettant une administration facile et intuitive de PostgreSQL.

3. PRESENTATION DES MISSIONS

3.1 COMPETENCES DU REFERENTIEL COUVERTES PAR LE PROJET

Nom et prénom du candidat :

CALLET Alexis

Document complété d'un commun accord entre le stagiaire et le responsable du stage en entreprise à joindre au rapport d'activité.

Compétences Voir le détail dans le référentiel d'emploi, d'activités et de compétences	Cocher les compétences mises en œuvre lors du projet en entreprise (en totalité ou partiellement)
C1 - Installer et configurer son environnement de travail en fonction du projet web ou web mobile	<input type="checkbox"/>
C2 - Maquetter des interfaces utilisateur web ou web mobile	<input type="checkbox"/>
C3 - Réaliser des interfaces utilisateur statiques web ou web mobile	<input type="checkbox"/>
C4 - Développer la partie dynamique des interfaces utilisateur web ou web mobile	<input type="checkbox"/>
C5 - Mettre en place une base de données relationnelle	<input type="checkbox"/>
C6 - Développer des composants d'accès aux données SQL et NoSQL	<input type="checkbox"/>
C7 - Développer des composants métier coté serveur	<input type="checkbox"/>
C8 - Documenter le déploiement d'une application dynamique web ou web mobile	<input type="checkbox"/>

Observations éventuelles :

Du stagiaire : Très bon retour de cette expérience enrichissante. L'équipe a été superbe, accompagnement complet et régulier.

De l'entreprise :

avis d'avant travail
excellent
Parfait

Nom et Prénom du stagiaire :

Callet Alexis

Signature :



Entreprise :

Intercept

Nom du responsable de stage :



Signature :



3.2 LES MISSIONS PRINCIPALES DU PROJET TOOBIB

Mon stage s'est déroulé autour de missions définies en amont et complétées par des tâches spécifiques pendant la période de stage, issues du système de tickets de Framagit.

Intégration de Keycloak et création d'un thème personnalisé :

- Intégration complète de Keycloak pour connecter les utilisateurs et permettre de réaliser des actions de changement de mot de passe, de déconnexion de Keycloak, depuis Toobib.org.
- Création d'un thème personnalisé pour Toobib.org, adapté aux besoins spécifiques de la plateforme.
- **Outils utilisés :** Keycloak, HTML, TailwindCSS, CSS, React, Next.js, Postman, JavaScript, Java, Docker.

Enrichissement du contenu :

- Mise en place de nouvelles pages, telles que "Outils", "Cotisations", "Profil", "Modifier mon profil" et "Trouver son Toobib", avec une gestion sécurisée des routes.
- **Outils utilisés :** React, React Router, Next.js., JavaScript, Postman, TailwindCSS, CSS, HTML.

Connectivité API :

- Développement d'interfaces pour interagir avec différentes API REST, notamment ProSanté Connect (authentification), l'annuaire des professionnels de santé et la base d'adresses nationale.
- **Outils utilisés :** Postman, Axios, Next.js.

Création d'API :

- Conception et réalisation de deux API :
 - Une API permettant aux utilisateurs d'ajouter et de modifier leur image de profil.
 - Une API pour la recherche de professionnels de santé, avec dans un second temps la possibilité de filtrage par localisation.
- **Outils utilisés :** VMWare, Linux, PostgreSQL, DBeaver, Python, Flask, SQLAlchemy.

Tests unitaires et d'intégration :

- Mise en place de tests unitaires et de tests d'intégration pour garantir la qualité et la fiabilité des développements réalisés.
- **Outils utilisés :** Jest.

Mon stage chez InterHop a été une expérience riche en apprentissages et en défis stimulants notamment concernant la profondeur des outils à disposition pour la manipulation des données de santé. J'ai pu mettre en pratique mes connaissances techniques et développer de nouvelles compétences dans un environnement innovant et porteur de sens.

4. INTEGRATION DE KEYCLOAK ET CREATION D'UN THEME PERSONNALISE



4.1 POURQUOI CE CHOIX ?

Au-delà du côté pratique de Keycloak, aussi bien côté utilisateur que du côté de la gestion, permettant une **authentification rapide** à plusieurs applications depuis un seul et même compte, tout en proposant une **gestion des utilisateurs centralisée**, rendant l'administration facile, Keycloak répond à un besoin fondamental, celui d'être un **pilier de la sécurité**.

En plus d'être sécurisé, Keycloak est outil **open-source**, apportant une extensibilité et une personnalisation pour que les diverses interfaces sur lesquelles l'utilisateur est redirigé, puissent être ressenties comme une simple extension de Toobib, sa communauté de développeurs étant nombreuses, les mises à jours sont nombreuses et correspondent aux besoins exprimés par ses utilisateurs.

En permettant la création de thèmes personnalisés, répond donc aux besoins de Toobib de créer un cadre de confiance et sécurisé, tout en s'assurant que les données nécessaires à l'inscription d'un utilisateur correspondent aux besoins de Toobib de pouvoir identifier un professionnel de santé grâce à son numéro RPPS ou AMELI (numéros d'identification des professionnels de santé en France).

Annexe 1 : Diagramme de séquence Keycloak.

Toobib a fait le choix d'une solution d'authentification performante, fiable et répondant aux enjeux actuels de la sécurité numérique. Cette décision permettra à la plateforme de continuer à offrir à ses utilisateurs une expérience sécurisée et de qualité, tout en favorisant son développement et son innovation.

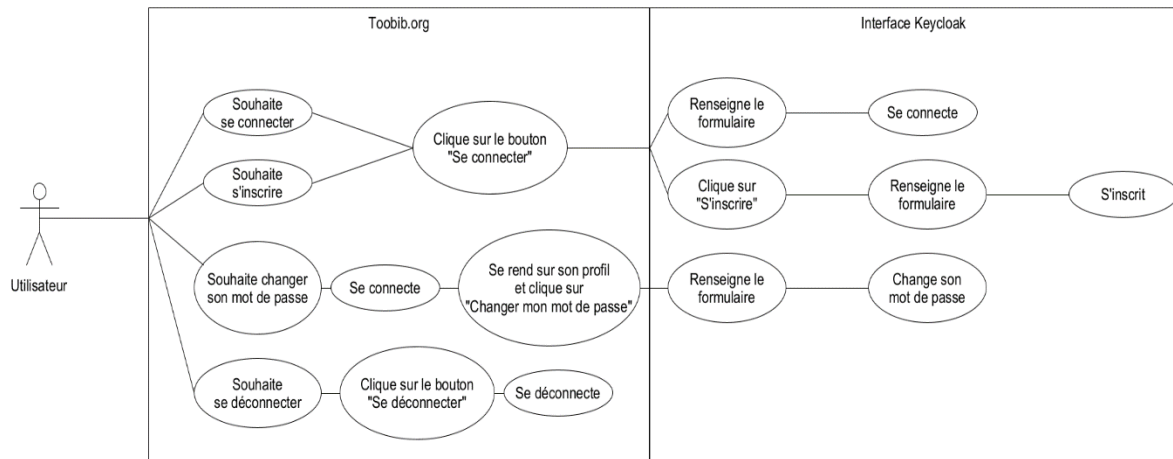
4.2 INTEGRATION DE KEYCLOAK

L'intégration de Keycloak au sein de l'application passe avant tout par sa mise en route avec **Docker**, permettant la création d'un conteneur Keycloak grâce à la configuration du fichier « docker-compose.yml » que l'on peut modifier de façon à exécuter Keycloak dans un environnement local, permettant ainsi de tester son bon fonctionnement avec **Postman**, en envoyant des requêtes sur les terminaux (endpoints) de Keycloak.

Annexe 2 : Extrait du fichier « docker-compose.yml ».

Pour permettre la bonne intégration de Keycloak sur Toobib, il a d'abord fallu déterminer de quelles fonctionnalités proposées par Keycloak nous allions avoir besoin, il a donc été choisi de rendre possible les actions suivantes sur Keycloak pour un utilisateur depuis Toobib :

Diagramme de cas d'utilisations



Chacune de ces actions correspond à un terminal (endpoint) de Keycloak, nous envoyons une requête sur ces différents endpoints au sein du code.

```
Exemple : `${keycloakServer}/protocol/openid-connect/logout?post_logout_redirect_uri=${redirectURI}&client_id=${clientID}`
```

Ici la variable `keycloakServer` correspond au nom de domaine utilisé pour Keycloak, il dépend de son utilisation locale ou non, « `logout` » est mon endpoint, me permettant de lui faire comprendre que la requête envoyée concerne une déconnexion et `redirectURI` lui, indique l'URL de redirection utilisé, il doit être au préalable autorisé dans la configuration de Keycloak afin d'assurer la sécurité de l'utilisateur.

Annexe 3 : Extrait de code du bouton de changement de mot de passe.

4.3 LA SECURITE APPOREE PAR KEYCLOAK POUR TOOBIB

La sécurité mise en œuvre par Keycloak assure la pérennité de l'application et la protection des données des utilisateurs, ses principales fonctionnalités incluent :

- **Authentification sécurisée**, prise en charge des protocoles standard et prévention des attaques par force brute.
- **Gestion des autorisations**, attributs et balises d'autorisation, et décisions d'autorisation basées sur des règles.
- **Protection des données** avec chiffrement, sécurité des sessions et conformité aux normes.
- **Audit et traçabilité** avec journalisation détaillée, détection des intrusions et rapports et analyses.
- **La gestion des URL de redirection :**

Valid redirect URIs ⓘ	http://localhost:3000/api/auth/callback/keycloak	⊖
	http://https://dev.front.toobib.org/mdp-modifie	⊖
	https://dev.front.toobib.org/*	⊖
	⊕ Add valid redirect URIs	
Valid post logout redirect URIs ⓘ	http://localhost:3000	⊖
	https://dev.front.toobib.org/*	⊖
	http://https://dev.front.toobib.org/mdp-modifie	⊖
	⊕ Add valid post logout redirect URIs	

- **Un panneau d'administration puissant**, il permet par exemple de gérer les durées des sessions, ainsi que la durée d'inactivité maximale de l'utilisateur avant de le déconnecter, d'activer ou non des méthodes d'authentification à double facteurs, l'authentification par biométrie ou encore l'ajout de moyens de connexion tierces, comme ProSanté Connect dans le cas de Toobib.

Ma veille technologique concernant Keycloak s'est déroulée sur de multiples plateformes, grâce à son côté open-source, le site Internet de Keycloak dans un premier temps, mais surtout Github, avec la résolution de tickets et l'explications de centaines de développeurs qui collaborent sur le dépôt Keycloak.

Lien du dépôt Git de Keycloak : <https://github.com/keycloak/keycloak>

4.4 CREATION DU THEME PERSONNALISE

La création du thème personnalisé de Keycloak pour Toobib a nécessité la manipulation du conteneur Keycloak utilisé localement. Dans son architecture, le conteneur Keycloak comprend un dossier appelé « **themes** » dans lequel il est possible d'ajouter un thème personnalisé.

L'architecture d'un thème est plutôt simple. Elle est composée d'un dossier « **admin** », qui comprend les propriétés du thème, permettant par exemple de gérer les langues disponibles, ainsi que d'un dossier « **login** » qui contient les éléments du thème.

Le dossier « login » inclut :

- Un dossier « **messages** » contenant les éléments de texte apportés par le thème personnalisé.
- Un dossier « **resources** » contenant les fichiers CSS, les images nécessaires à la documentation du thème et sources des logos par exemple, ainsi que les scripts, codés en JavaScript, pour les fonctionnalités.
- Les fichiers FTL sont des modèles FreeMarker (FreeMarker Template Language) qui définissent la structure des vues interprétées par le navigateur. Ils contiennent du HTML avec des instructions dynamiques en FreeMarker et peuvent inclure des fragments de code en Java pour générer du contenu dynamique.

Annexe 4 : Capture d'écran de l'architecture d'un thème personnalisé

Afin de **répondre au besoin** de Toobib d'intégrer son identité visuelle sur ses interfaces Keycloak, la manipulation des fichiers FTL ainsi que des différents fichiers CSS a été nécessaire.

J'ai donc eu carte blanche, je n'avais pas de maquette à suivre, seule la satisfaction de l'équipe lors des réunions journalières guidaient mon travail, j'ai alors commencé par manipuler le fichier CSS ainsi que le fichier « **template.ftl** » qui représente la vue de base répétée sur tous les autres fichiers .ftl afin d'y intégrer le logo de Toobib.

```
<body class="{properties.kcBodyClass!}">
<div class="{properties.kcLoginClass!}">
  <div id="kc-header" class="{properties.kcHeaderClass!}">
    <div id="kc-header-wrapper" class="{properties.kcHeaderWrapperClass!}">
      
    </div>
  </div>
</div>
```

Mais également le fichier CSS du thème, dans lequel **aucun tri n'est effectué de base**, j'ai donc décidé de séparer le CSS de base du CSS que j'ai ajouté, afin d'assurer que les développeurs amenés à travailler sur le thème à l'avenir puissent le faire dans les meilleures conditions possibles.

Annexe 5 : Capture d'écran du thème de base de Keycloak

Capture d'écran du thème personnalisé pour Toobib (version Ordinateur)



Connectez-vous à Toobib

Nom d'utilisateur ou courriel


Mot de passe

☐ Se souvenir de moi

[Mot de passe oublié ?](#)

Connexion

Ou se connecter avec

 **PRO SANTE CONNECT**
Un service du ministère chargé de la Santé

Nouveau sur Toobib ?

[Créer un compte](#)


Français ▼

Capture d'écran du thème personnalisé pour Toobib (version Mobile)



Connectez-vous à Toobib

Nom d'utilisateur ou courriel

Mot de passe 

☐ Se souvenir de moi

[Mot de passe oublié ?](#)

Connexion

Ou se connecter avec



Nouveau sur Toobib ? [Créer un compte](#)

Français ▼

L'autre besoin de Toobib lui, est fonctionnel, de base, l'inscription d'un utilisateur ne comporte pas de champ lui permettant d'indiquer son numéro d'identification RPPS (numéro d'identification national des professionnels de santé) il a donc fallu répondre à ce besoin en modifiant le fichier « register.ftl » qui lui, contient les éléments générés par la page d'inscription de Keycloak.

Ajout du champ « RPPS » au sein du code du fichier « register.ftl »

```
<div class="${properties.kcFormGroupClass!} ${messagesPerField.printIfExists('rpps',properties.kcFormGroupErrorClass!)}">
  <div class="${properties.kcLabelWrapperClass!}">
    <label for="rppsId" class="${properties.kcLabelClass!}">${msg("rppsId")}</label>
  </div>
  <div class="${properties.kcInputWrapperClass!}">
    <input
      type="text"
      id="rppsId"
      class="${properties.kcInputClass!}"
      name="rppsId"
      value="${(register.formData['user.attributes.rpps']! '')}"
      minlength="11"
      maxlength="11"

      aria-invalid="<#if messagesPerField.existsError('rppsId')>true</#if>"

    />
    <span hidden id="input-error-rppsId" class="${properties.kcInputErrorMessageClass!}" aria-live="polite">
      </span>
    </div>
  </div>
</div>
```

Ce champ a également nécessité l'ajout d'éléments textuels dans les fichiers messages, dans les langues souhaitées par Toobib (Français et Anglais), ce champ est primordial à l'identification d'un professionnel de santé, l'utilisateur est donc obligé de le renseigner lors de son inscription, or, Keycloak ne me permet de directement modifier le fonctionnement du formulaire, j'ai donc dû ajouter un script me permettant de valider la saisie au sein du champ « rppsId ».

Extrait de code du script « RppsValidation.js »

```
document.addEventListener('DOMContentLoaded', function() {
  var form = document.getElementById('kc-register-form');
  var inputRppsId = document.getElementById('rppsId');
  var selectedLanguage = document.getElementById('kc-current-locale-link');

  // Je récupère la valeur de mon message d'erreur dans mon formulaire (register.ftl)
  var errorMessage = document.getElementById('input-error-rppsId');

  form.addEventListener('submit', function(event) {
    var inputRppsIdValue = inputRppsId.value.trim();
    var regex = /^[0-9]{11}$/;

    if (!inputRppsIdValue || inputRppsIdValue.length !== 11 || !regex.test(inputRppsIdValue)) {

      // Vérification sur le langage utilisé par l'utilisateur (valeur de l'attribut outerText de la variable prenant pour valeur l'id "kc-current-locale-link")
      // afin d'adapter le message d'erreur utilisé
      if(selectedLanguage.outerText === "Français"){
        errorMessage.textContent = 'Veuillez renseigner un identifiant RPPS valide de 11 chiffres.';
      }else if(selectedLanguage.outerText === "English"){
        errorMessage.textContent = 'Please specify a valid RPPS Identifier of 11 digits.';
      }else{
        errorMessage.textContent = 'Please specify a valid RPPS Identifier of 11 digits.';
      }

      // Ici j'utilise le comportement d'affichage des messages d'erreur de Keycloak par défaut en mettant la valeur de l'attribut "aria-invalid" à true
      inputRppsId.setAttribute('aria-invalid', 'true');

      // Je mets l'attribut "hidden" de mon message d'erreur à faux,
      // car je ne peux pas utiliser les conditions par défaut de Keycloak,
      // je dois donc faire une logique ici, qui fait simplement en sorte qu'un élément HTML apparaisse si l'on parvient à cet endroit du code
      errorMessage.hidden = false;

      // J'empêche la soumission du formulaire en cas d'erreur
      event.preventDefault();
    }else{
      errorMessage.hidden = true;
      inputRppsId.setAttribute('aria-invalid', 'false');
    }
  });
});
```

Grâce à ce script, que je prends soin d'importer dans mon fichier « **register.ftl** », j'assure qu'à la soumission du formulaire, l'utilisateur se verra indiqué que le fait de renseigner ce champ est obligatoire s'il ne l'a pas renseigné ou n'a pas respecté la structure d'un numéro d'identification RPPS, qui est de 11 caractères.

Message d'erreur affiché à l'utilisateur en cas de soumission du formulaire non conforme

Identifiant RPPS

!

Veuillez renseigner un identifiant RPPS valide de 11 chiffres.

En cas de soumission valide, je laisse Keycloak garder son comportement par défaut et venir, grâce à l'ajout de ce champ, me créer un attribut « **rppslid** » pour chacun de mes utilisateurs, attribut que je peux retrouver sur mon panneau d'administration et qui permet aux administrateurs de la plateforme, de valider l'inscription ainsi que l'adhésion à l'association de l'utilisateur mais également pour de futures tâches de développement nécessitant la récupération de cet identifiant.

Key	Value	
PractitionerRole	Epithésiste	⊖
Cabinet	{ "id": { "Lieu": "CABINET ORTHOPHONISTE MME PROS...	⊖
rppslid	00039100730	⊖

Pour conclure concernant Keycloak, j'ai choisi de présenter l'intégration de cet outil ainsi que la création du thème personnalisé car c'était un objectif important de mon stage, la compréhension de l'outil et la possibilité pour Toobib de faire perdurer son cadre de confiance étaient primordiaux pour la plateforme.

5. DEVELOPPER LE CONTENU DU SITE

[Outils](#)[Cotisations](#)[Blog](#)[Mon profil](#)[Trouver son Toobib](#)[Se déconnecter](#)

La plateforme web pour des services numériques éthiques en santé

Toobib met à disposition une boîte à outils libre, open-source et éthique pour les patients et les professionnels de santé. L'association sensibilise et forme les professionnels de santé aux enjeux du numérique.

[Nous contacter](#)

5.1 BESOINS DE TOOBIB

Concernant le contenu proposé sur Toobib.org, **les besoins de la plateforme ont été exprimés lors de nos réunions quotidiennes** avec soumission d'une maquette afin d'identifier au mieux, les éléments qui doivent être rendus sur les pages du site.

Les points suivants ont été soulignés :

1. Présentation des Services :

- Une page « Nos outils » doit être créée, présentant les outils utilisés par l'association Toobib.
- Cette page doit être accessible depuis la page d'accueil du site.
- Le contenu proposé sur cette page doit pouvoir être géré par la simple manipulation d'un fichier .md(markdown).

2. Système de Recherche de Praticiens :

- L'intégration d'une page pour le système de recherche de praticiens de santé est primordiale pour atteindre l'objectif final de Toobib.
- Afin de correspondre à un souhait fort concernant la performance du système de recherche, il a été décidé de modifier les formulaires de recherche et de s'éloigner par conséquent de la maquette.

3. Cotisations et Adhésions :

- Il est essentiel de mettre en avant les cotisations et adhésions à l'association.
- Cette page doit être accessible depuis la page d'accueil du site.

4. Espace Personnel :

- L'utilisateur doit pouvoir profiter d'un espace personnel dynamique et réactif.
- La modification de son profil doit être fluide, tout en assurant la justesse des données avec une fonctionnalité lui permettant de remplir le formulaire de modification de son profil avec des informations disponibles via l'API FHIR.

5.2 SPECIFICITES TECHNIQUES

L'application Toobib est développée avec le framework **Next.js**, ce qui en fait une application principalement composée du langage **JavaScript**.

Elle utilise également la bibliothèque **React**, ce qui permet le développement d'interfaces utilisateur réactives et dynamiques. L'utilisation de React garantit une expérience utilisateur fluide et interactive grâce à la manipulation efficace du DOM virtuel.

Le contenu des pages statiques telles que « Blog » et « Nos outils » est stocké dans des fichiers **Markdown (.md)**. Cette approche permet un ajout et une suppression de contenu faciles, tout en maintenant un format lisible et modifiable par les non-développeurs.

Les outils externes utilisés par l'application afin de répondre aux besoins de Toobib sont les suivants :

- **Keycloak :**
 - Système d'authentification des utilisateurs et mesures de sécurité pour protéger les utilisateurs que nous avons vu précédemment.
- **Les API externes :**
 - **FHIR** : FHIR est une norme pour l'échange électronique d'informations de santé. Elle permet d'assurer l'interopérabilité entre différents systèmes de santé, facilitant ainsi l'échange et l'intégration des données de santé.
 - **Annuaire de Santé** : Cette API permet d'accéder à une base de données de professionnels de santé, fournissant des informations détaillées et actualisées sur les praticiens. Elle est essentielle pour le système de recherche de praticiens de santé.
 - **BAN** : La BAN (base adresse nationale) est une base de données gouvernementale qui fournit des adresses géographiques précises et normalisées.
- **Les API internes :**
 - **Toobib - API pour la gestion des images** : API développée pendant la période de stage.
 - **Toobib - API pour la recherche de praticiens de santé** : API développée pendant la période de stage.
- **Bibliothèque**
 - **Formik** : Bibliothèque JavaScript open-source pour la gestion des formulaires dans les applications React. Elle vise à améliorer la gestion des formulaires en fournissant des outils et des abstractions pour gérer les états des formulaires, la validation, et la soumission.
 - **Yup** : Bibliothèque JavaScript open-source utilisée pour la validation des schémas d'objets JavaScript.

5.3 CREATION DES PAGES

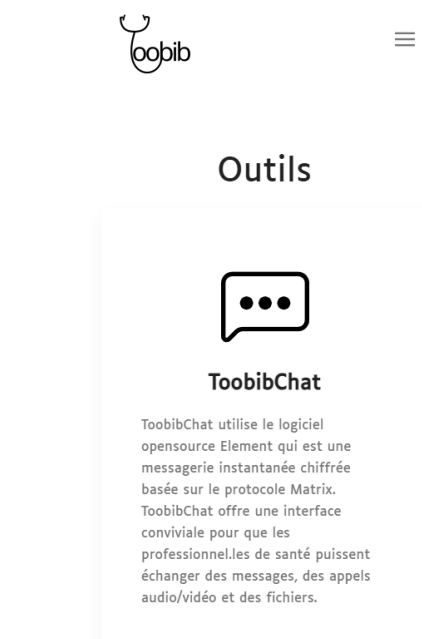
Création de la page statique « Nos outils »

La première page créée durant la période de stage a été la page « Nos outils », page statique permettant la présentation des outils utilisés par l'association Toobib et qui doit permettre l'ajout et la lecture de contenu, par modification de fichiers Markdown.

Capture d'écran de la page « Nos outils » version ordinateur



Capture d'écran de la page « Nos outils » version mobile



Extrait de code de la page « Nos outils »

```
return (
  <>
    <section className="section">
      <div className="container">
        {markdownify(title, {tag: "h1", className: "text-center font-normal"})}
        <div className="grid grid-cols-2 lg:grid-cols-3">
          {articles.map((article, index : number) => (
            <div key={index} className="col-span-3 mt-6 md:col-span-1">
              <div className="p-12 shadow flex flex-col justify-between">
                <div>
                  style={{
                    display: "flex",
                    justifyContent: "center",
                    alignItems: "center",
                    height: "100%",
                  }}
                >
                  <Image
                    className="my-2"
                    height={128}
                    width={128}
                    src={article.logo}
                    alt={article.title}
                  />
                </div>
                <div className="text-center">
                  {markdownify(article.title, {tag: "h3", className: "text-center"})}
                </div>
                <div className="my-6 md:min-h-60 lg:min-h-72 xl:min-h-64">
                  <div>
                    {markdownify(article.description)}
                  </div>
                </div>
              </div>
            </div>
          ))}
        </div>
      </div>
    </section>
  </>
);
```

Ici, la bonne gestion des colonnes a été primordiale afin d'assurer un affichage correct dans toutes les résolutions, que ce soit sur un ordinateur de bureau, une tablette ou un smartphone.

Création de la page statique « Cotisations » avec intégration de l'iframe HelloAsso

La page « **Cotisations** » intègre la campagne d'adhésion de Toobib créée sur HelloAsso, il est donc nécessaire d'intégrer l'iframe proposé par ce service afin de rendre possible l'adhésion à l'association depuis la plateforme Toobib.

Annexe 6 : Capture d'écran de la page « Cotisations » version ordinateur

Concernant la page « **Cotisations** », l'élément « **iframe** » est un élément qui peut être très rigide et ne pas permettre une intégration de qualité sur toutes les résolutions.

Pour cette raison, j'ai décidé d'utiliser le hook « **useEffect** » de **React**, afin, qu'au chargement de ma page, je puisse détecter la résolution de mon utilisateur.

Extrait de code du hook personnalisé « useWindowSize »

```
1+ usages  KKMYA *
const useWindowSize = () : {height: undefined, width: undefined} => {

  // Déclare la variable d'état windowSize avec ses propriétés width et height initialisées à undefined.
  // setWindowSize est le setter permettant de mettre à jour windowSize.
  const [windowSize : {height: undefined, width: undefined} , setWindowSize] = useState( initialState: {
    width: undefined,
    height: undefined,
  });

  // Utilise useEffect pour exécuter le code au chargement de la page.
  useEffect( effect: () => {
    // Déclare une fonction pour récupérer les dimensions actuelles de la fenêtre.
    1+ usages  KKMYA
    const handleResize = () :void => {
      setWindowSize( value: {
        width: window.innerWidth,
        height: window.innerHeight,
      });
    };

    // Ajoute un écouteur d'événements à la fenêtre pour mettre à jour les dimensions lorsqu'elles changent.
    window.addEventListener( type: 'resize', handleResize);

    // Exécute handleResize pour définir les dimensions initiales.
    handleResize();

    // Nettoie l'écouteur d'événements lors du démontage du composant pour éviter les fuites de mémoire.
    return () :void => {
      window.removeEventListener( type: 'resize', handleResize);
    };
  }, deps: []); // Le tableau de dépendances vide signifie que cet effet s'exécute une seule fois après le premier rendu

  // Retourne l'objet windowSize contenant les dimensions actuelles de la fenêtre.
  return windowSize;
};

1+ usages  KKMYA
export default useWindowSize;
```

La création de ce hook personnalisé me permet ensuite, de pouvoir effectuer un rendu différent en fonction de la résolution de mon utilisateur, où je décide de rendre possible le scroll dans l'iframe lors de l'utilisation de petites résolutions, la fonction `removeEventListener` me permet d'éviter une fuite de mémoire, problème rencontré lors de la première création du hook qui a nécessité cet ajout.

Annexe 7 : Extrait de code de la page « Cotisations »

Création des espaces dynamiques « Mon profil » et « Modifier mon profil »

Dans l'espace personnel de l'utilisateur, les données transitent entre Keycloak et les différentes API utilisées sur ces pages, **Keycloak stocke et est capable de retourner**, toutes les informations que l'utilisateur a renseigné lors de son inscription ou qu'il a mit à jour en modifiant son profil, l'API de la gestion des images permet la mise à jour et l'affichage de l'image de profil de l'utilisateur, l'API FHIR quand à elle, joue un rôle important dans une fonctionnalité importante pour l'expérience de l'utilisateur en lui permettant, en un clic, de mettre à jour son profil avec les données récupérée grâce à son identifiant RPPS.

Annexe 8 : Maquette de la page « Mon profil »

Sur la base de la maquette fournie, il a donc été décidé la réalisation de l'espace personnel « Mon profil », celui-ci devait permettre l'affichage d'un maximum d'informations contenues dans les attributs Keycloak, ou générées par la mise à jour du profil par l'utilisateur.

Capture d'écran de la page « Mon profil » version ordinateur



Annexe 9 : Capture d'écran de la page « Mon profil » version mobile

Extraits de code de la page « Mon profil »

```
useEffect( effect: () :void => {
  1+ usages 1 Alin1233
  const fetchData = async () : Promise<void> => {
    if (session) {
      const response : any = await getAccount(session.accessToken);

      setAccount(response);
      if (response && response.attributes) {
        setPhotoUrl(
          {value: process.env.NEXT_PUBLIC_PHOTO_DOMAIN +
            '/photo?id=${response.attributes.rppsId}&t=${Date.now()}',
          });
      }
    }
  };
  fetchData();
}, [session, photoUpdated]);
```

Ici, au chargement de ma page, **je m'assure qu'une session est en cours** pour récupérer la photo correspondante à mon utilisateur.

```
if (!account) {
  return (
    <>
    <section className="section">
      <div className="container utilities-body py-44 pb-96">
        <h1 className="text-center font-normal">Veuillez vous connecter</h1>
        <div className="flex flex-col justify-center items-center space-y-4 section p-12 mt-6">
          <button
            className="btn btn-primary flex items-center justify-center w-72"
            onClick={() : Promise<SignInResponse> => signIn( provider: "keycloak")}
          >
            Se connecter
          </button>
        </div>
      </div>
    </section>
    </>
  );
}
```

Si aucun compte n'est connecté à la session en cours, bien que la page mon profil ne soit pas accessible directement depuis la page d'accueil, **j'empêche l'accès à la page « Mon profil » en redirigeant mon utilisateur vers une page de connexion** si ce dernier se rend sur l'URL de cette page.

```
if (session) {
  if (account.attributes === undefined) {
    router.push( url: "/modifier-profil");
  }
}
```

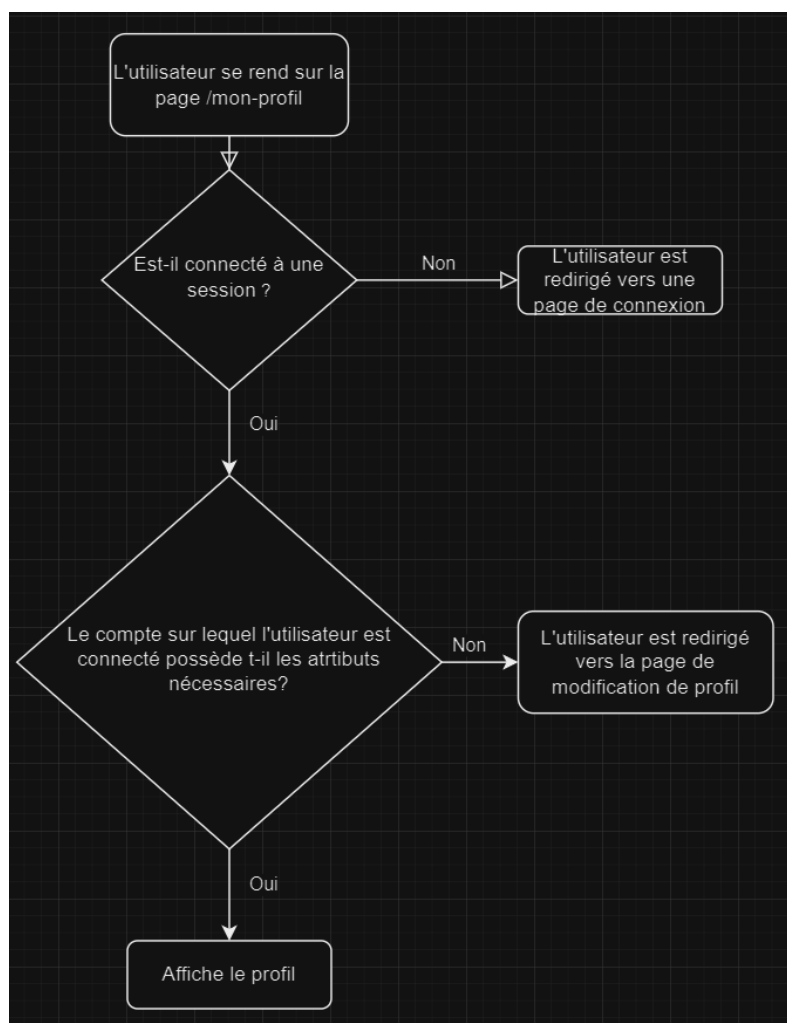
Si mon utilisateur est bien lié à une session, mais que les attributs que j'essaie de lire dans l'affichage de son profil sont « undefined » je le dirige vers la modification de son profil.

```
{Object.entries(  
  JSON.parse(  
    text: account.attributes.Cabinet === undefined  
      ? "{}"  
      : account.attributes.Cabinet,  
  ),  
)}.map(([key : string , value]) => (  
  <React.Fragment key={key}>  
    <CabinetDisplay id={key} data={value} />  
    <br />  
  </React.Fragment>  
)})
```

Mapping sur l'objet JSON retourné par Keycloak, qui contient l'attribut « Cabinet ».

Grâce à toutes ces conditions et en m'assurant que l'utilisateur soit bel et bien passé par l'étape de la modification de son profil, **je réponds au besoin de Toobib d'afficher un maximum d'informations pertinentes et à jour** à propos de mon utilisateur, qui lui, est informé ou redirigé sur les éléments adéquats de l'application, **en fonction de ce que l'on attend de lui.**

Diagramme de flux pour l'affichage et les redirections de la page « Mon profil »



Annexe 10 : Maquette de la page « Modifier mon profil »

Lors des réunions journalières avec l'équipe, il a été décidé d'intégrer le changement de la photo directement depuis la page « Mon profil », nous avons alors commencer le développement en nous basant sur la maquette fournie, sans y intégrer le changement de photo, jusqu'à discuter d'une idée de fonctionnalité avec l'équipe, qui serait la suivante :

Permettre à l'utilisateur de mettre à jour son profil en utilisant des données fiables qui sont disponibles à son sujet publiquement, via l'API FHIR.

Capture d'écran de la page « Modifier mon profil » version ordinateur

Modifier mon profil

Remplir le formulaire grâce à l'annuaire des professionnels de santé

Informations personnelles

Prénom

Nom de famille

Profession

Numéro RPPS

Numéro de téléphone

Email

Conventionnement

Adresse(s)

Ajouter un lieu de consultation



Sauvegarder

Annuler

Annexe 11 : Capture d'écran de la page « Modifier mon profil » version mobile

Afin de réaliser la fonctionnalité attendue, il est nécessaire de faire en sorte que le bouton placé au-dessus du formulaire interroge l'API FHIR avec un numéro RPPS, c'est pourquoi l'attribut Keycloak est nécessaire lors de l'inscription, on s'assure qu'il est présent par défaut, si jamais il ne l'est pas, le bouton n'est pas fonctionnel.

Extrait de code de la méthode « handleFhirClick »

```
const handleFhirClick = async () : Promise<void> => {
  const practitioner : any = await getPractitionerAndPractitionerRole(rppsId);

  if (practitioner.error) {
    setErrorNotif({value: true});
    return;
  }
  setErrorNotif({value: false});
  let professionCode;
  practitioner.data?.entry[1]?.resource?.code[0]?.coding.forEach((code) : void => {
    if (parseInt(code.code)) {
      professionCode = code.code;
    }
  });
  const profession : string = await getProfessionString(professionCode);
  const orgId : string = extractOrgIdFromFhirPractitionerBundle(practitioner.data);
  let fhirData : object = {};

  if (orgId) {
    orgId.forEach(async (id, index : number) : Promise<void> => {
      const org : object = await getFormattedOrganizationData(id);
      formik.setFieldValue( field: `Cabinet.${index + 1}.Lieu`, org.Lieu);
      formik.setFieldValue( field: `Cabinet.${index + 1}.CodePostal`, org.CodePostal);
      formik.setFieldValue( field: `Cabinet.${index + 1}.Adresse`, org.Adresse);
      formik.setFieldValue(
        field: `Cabinet.${index + 1}.PhoneLieuConsultation`,
        org.PhoneLieuConsultation
      );
      formik.setFieldValue( field: `Cabinet.${index + 1}.Ville`, org.Ville);
    });
    fhirData = { ...fhirData, profession: profession };
    formik.setFieldValue( field: "PractitionerRole", fhirData.profession);
  }
};
```

Cette méthode consulte l'API FHIR à l'url suivant :

[https://gateway.api.esante.gouv.fr/fhir/Practitioner? id=\\${rppsId}& revinclude=PractitionerRole%3Apractitioner](https://gateway.api.esante.gouv.fr/fhir/Practitioner? id=${rppsId}& revinclude=PractitionerRole%3Apractitioner)

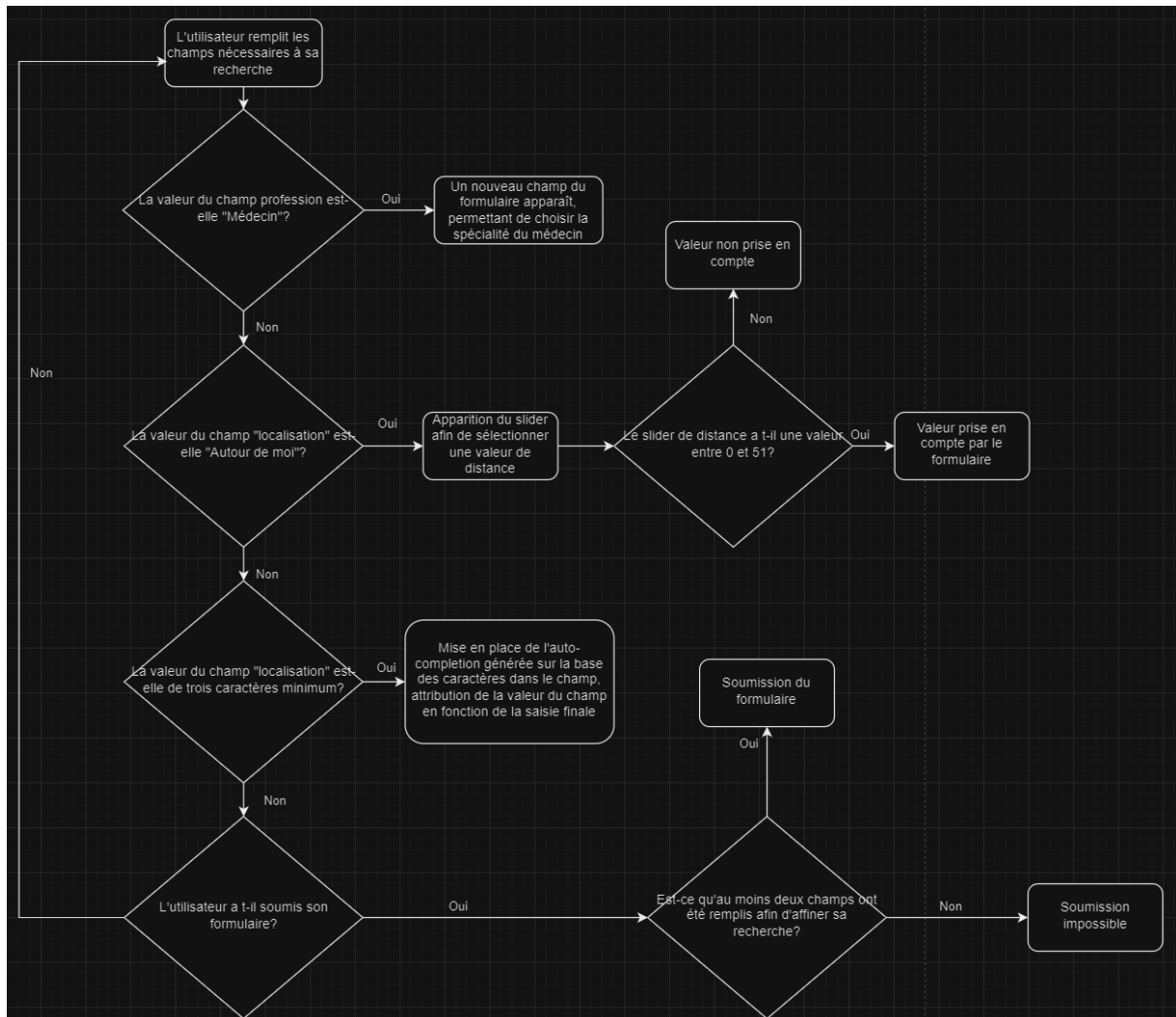
L'objet JSON qui nous est retourné me sert à retrouver toutes les informations me permettant de remonter les informations supplémentaires nécessaires, comme la profession, avec le code de profession, ainsi que l'organisation, avec le code de l'organisation, s'il y'en a une.

Avec toutes ces informations, **je peux maintenant attribuer les valeurs aux champs de mon formulaire** et permettre à l'utilisateur de mettre à jour ses données rapidement.

L'ajout d'une telle fonctionnalité a soulevé des questions lors de notre présentation devant l'agence du numérique en santé, une éventuelle fonctionnalité supplémentaire à ce formulaire serait de pouvoir également choisir de demander à mettre à jour l'annuaire de santé, avec les données renseignées dans ce formulaire.

Afin de mieux comprendre le comportement du formulaire voici un diagramme de flux du formulaire de recherche :

Diagramme de flux du comportement du formulaire de la recherche de praticiens de santé



En plus de ce comportement se rajoute des vérifications réalisées par Formik et Yup, afin de valider le schéma des données soumises dans le formulaire, je prend également soin de bien faire en sorte qu'une ville ne puisse pas être saisie en plus de laisser une valeur au champ « rangeAround » lié au slider en réinitialisant sa valeur à null, dès que l'utilisateur n'indique plus la valeur « Autour de moi » dans le champ.

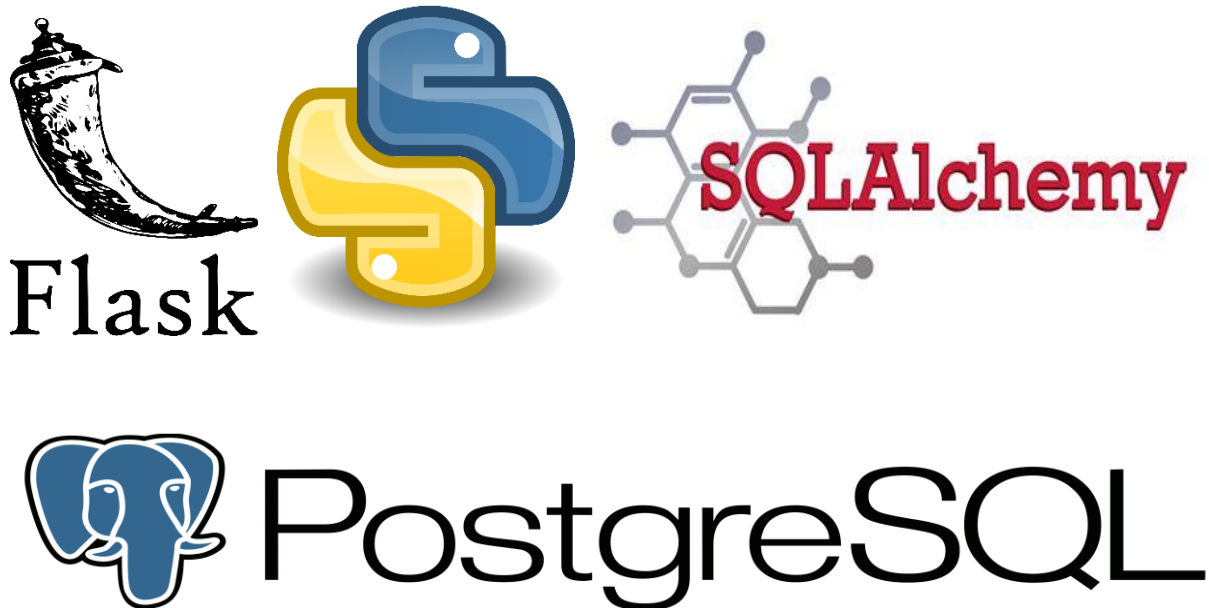
Pour ne pas trop réaliser de requêtes sur l'API qui génère l'auto-complétion des villes, je m'assure également de réaliser ma méthode avec la fonction `debounce` de JavaScript, qui me permet d'affecter un délai à cette dernière afin qu'elle ne soit pas effectuée en boucle.

Extrait de code de la fonction me permettant de générer les villes affichées dans l'auto-complétion du champ de mon formulaire.

```
const fetchCities = useCallback(debounce( func: (userInput) :void => {
  fetch( {url: 'https://geo.api.gouv.fr/communes?nom=${userInput}&boost=population&limit=4&fields=code_nom,code_postaux,centre'} ) Promise<Response>
    .then(response : Response => response.json() Promise<any>)
    .then(data => {
      const citiesArray = data.map(city => ({
        ville: city.nom,
        longitude: city.centre.coordinates[0],
        latitude: city.centre.coordinates[1]
      }));
      setCities(citiesArray);
    }) Promise<void>
    .catch(error => {
      console.error('Error fetching data:', error);
    });
}, wait: 200), deps: []);
```

Créer les pages statiques et dynamiques a nécessité de gérer efficacement la mise en page et d'assurer une compatibilité optimale sur diverses résolutions d'écran. Intégrer la campagne d'adhésion HelloAsso a renforcé mes compétences en utilisation dynamique de React. Le développement des espaces « Mon profil » et « Modifier mon profil » m'ont permis de maîtriser les intégrations avec Keycloak et l'API FHIR, assurant une gestion sécurisée des informations utilisateur. Enfin, concevoir la page de recherche « Trouver son Toobib » m'a appris à optimiser les systèmes de recherche et à valider les données de manière performante.

6. DEVELOPPEMENT D'API POUR TOOBIB



6.1 BESOINS DE TOOBIB

Les besoins de la plateforme Toobib nécessitant le développement d'API ont été exprimés lors de réunions journalières, pendant lesquelles nous avons discuté des enjeux auxquels la plateforme doit absolument répondre afin d'assurer qualité des données, performance et une bonne expérience utilisateur, ces besoins sont les suivants :

- **Un système de recherche performant :**
 - L'équipe ayant déjà tenté d'effectuer un système de recherche uniquement en utilisant l'API FHIR, s'est vue être confrontée à un temps de réponse trop long, de parfois d'une trentaine de secondes pour des requêtes assez basiques, le développement de nos propres outils est donc nécessaire.
 - Le respect d'une maquette concernant l'affichage des résultats.
 - Des données de qualité, avec la contrainte de devoir mettre à jour les résultats trouvés par l'API FHIR.
- **Une personnalisation du profil sécurisée :**
 - Permettre la personnalisation des profils des utilisateurs par la modification de leurs images de profil afin d'améliorer l'expérience utilisateur et ce, de façon sécurisée.

6.2 SPECIFICITES TECHNIQUES

Les API sont développées avec le framework Flask, ce qui en fait une application principalement composée de Python.

L'API pour la recherche de praticiens de santé utilise PostgreSQL comme système de gestion de base de données relationnelles. Les fonctionnalités de PostgreSQL nous seront grandement utiles, pour par exemple réaliser la géolocalisation des praticiens de santé.

Pour l'interaction avec la base de données, j'utilise DBeaver, un outil universel de gestion de bases de données, offrant une interface graphique conviviale pour la manipulation des données et la gestion des schémas.

L'environnement de développement est basé sur Linux, installé sur une machine virtuelle avec VMWare Workstation.

Les outils externes utilisés par les API afin de répondre aux besoins spécifiques sont les suivants :

- **Les API externes :**
 - **FHIR** : FHIR est une norme pour l'échange électronique d'informations de santé. Elle permet d'assurer l'interopérabilité entre différents systèmes de santé, facilitant ainsi l'échange et l'intégration des données de santé.
 - **Annuaire de Santé** : Cette API permet d'accéder à une base de données de professionnels de santé, fournissant des informations détaillées et actualisées sur les praticiens. Elle est essentielle pour le système de recherche de praticiens de santé.
 - **BAN** : La BAN (base adresse nationale) est une base de données gouvernementale qui fournit des adresses géographiques précises et normalisées.
- **ORM :**
 - **SQLAlchemy** : Bibliothèque SQL toolkit et ORM (Object Relational Mapper) pour Python. Elle fournit une interface flexible et puissante pour interagir avec les bases de données relationnelles en utilisant des objets Python.

6.3 CREATION DE L'API POUR LA RECHERCHE DE PRATICIENS DE SANTE

La création de l'API a d'abord nécessité l'installation de l'environnement de travail. J'ai commencé par installer VMware Workstation, ce qui m'a permis de configurer une machine virtuelle avec Linux Mint. Ensuite, j'ai installé Microsoft Visual Studio, PostgreSQL et DBeaver. Après cela, les dépendances de Python et le framework Flask ont été installés, me permettant d'exécuter un script de création de base de données.

Création et peuplement de la base de données

Annexe 14 : Extrait du script « Makefile »

A l'exécution de ce script, on vient configurer la connexion à la base de données, puis les fichiers CSV sont téléchargés à partir de l'annuaire de santé et on vient exécuter nos scripts SQL de création de table.

La base de données créée est assez simple, elle ne possède aucune contrainte de clés primaires et est peuplée de près d'1 millions de praticiens de santé.

Ce qui va complexifier cette base de données, c'est la nécessité d'offrir à l'utilisateur des filtres poussés, qui amélioreront son expérience, c'est pourquoi, avec la création du filtre par localisation, des changements ont dû être effectués sur cette base de données, pour nous permettre l'intégration de ce filtre sur la plateforme Toobib.

Annexe 15 : Schéma de base de données

Pour permettre cette intégration, la première modification nécessaire a été d'ajouter une table « **coordonnees** », avec sa clé primaire, ses différentes colonnes et aussi la colonne « **geom** », qui sera peuplée par la suite.

Extrait du script de création de la table « coordonnees »

```
DROP TABLE IF EXISTS coordonnees;

CREATE TABLE coordonnees
(
    coordonnees_id SERIAL PRIMARY KEY,
    latitude DOUBLE PRECISION,
    longitude DOUBLE PRECISION,
    code_postal INTEGER,
    libelle_commune character varying,
    label character varying,
    code_insee character varying,
    rue character varying,
    house_nr character varying,
    quality boolean,
    geom GEOGRAPHY(Point, 4326)
);
```

Extrait du script de la modification de la table « ps_libreaccs_personne_activite » ajoutant la clé étrangère « fk_coordonnees »

```
ALTER TABLE IF EXISTS ps_libreaccs_personne_activite
|   ADD COLUMN IF NOT EXISTS coordonnees_id INTEGER;

ALTER TABLE IF EXISTS ps_libreaccs_personne_activite
|   ADD CONSTRAINT fk_coordonnees
|   FOREIGN KEY (coordonnees_id)
|   REFERENCES coordonnees(coordonnees_id);
```

La création de cette clé primaire me permet de créer une clé étrangère dans la table « **ps_libreaccs_personne_activite** », table qui contient toutes les informations nécessaires au système de recherche, je veille à ce qu'elle fasse référence à la colonne « **coordonnees_id** » de la table « **coordonnees** ».

Avec la création de la table « **coordonnees** », nous avons désormais une table prête à accueillir des informations concernant les organisations des professionnels de santé, il est donc maintenant nécessaire de la peupler.

Pour se faire, **nous avons réalisé un script**, ce script, initialement, demandait un temps total de **18 heures** afin d'être exécuté entièrement, nous avons donc retravaillé la requête initiale afin que cette exécution soit plus rapide, **d'environ 3 heures** et ce, sans la mise en place de multithreading, qui aurait pu permettre l'exécution de plusieurs requêtes simultanées, **nous avons décidé de notifier cet axe d'amélioration à l'équipe**, sans avoir le temps de pouvoir l'intégrer.

Extrait du script qui peuple la table « **coordonnees** », montrant la requête initiale permettant la sélection distincte de chaque objet de ma base de données composés d'un numéro de finess qui n'est pas « null » pour ensuite les trier de façon à garantir que chaque numéro de finess n'apparaît qu'une seule fois.

```
WITH ranked_data AS (
SELECT finess,
       numero_voie,
       indice_repetition_voie,
       libelle_type_de_voie,
       libelle_voie,
       code_postal,
       libelle_commune,
       coordonnees_id,
       ROW_NUMBER() OVER (PARTITION BY finess ORDER BY finess) AS row_num
FROM external.ps_libreaccs_personne_activite
WHERE finess IS NOT NULL
)
SELECT finess,
       numero_voie,
       indice_repetition_voie,
       libelle_type_de_voie,
       libelle_voie,
       code_postal,
       libelle_commune,
       coordonnees_id
FROM ranked_data
WHERE row_num = 1;

""")
```

Le script construit ensuite l'adresse complète d'une organisation, basée sur son numéro de FINESS avec les colonnes « **numero_voie**, **indice_repetition_voie**, **libelle_type_de_voie**, **libelle_voie**, **code_postal**, **libelle_commun** » afin d'interroger l'API BAN avec l'adresse construite.

Extrait du script qui peuple la table « coordonnees », montrant la méthode appelée pour interroger l'API BAN, en passant en paramètre l'adresse construite à l'aide des informations contenues dans la base de données.

```
def get_from_ban_api(param):  
    url = f"https://api-adresse.data.gouv.fr/search/?q={param}"  
    try:  
        response = requests.get(url)  
        response.raise_for_status()  
        return response.text  
    except requests.RequestException as e:  
        print(f"Error making API request: {e}")  
        return None
```

Cette méthode, une fois terminée, nous renvoi un objet JSON, qui, une fois interprété, nous permet d'ajouter des valeurs dans les colonnes « **quality** » à vrai ou faux, **VRAI** si l'objet JSON qui nous a été retourné par l'API était composé **d'un seul élément**, garantissant ainsi une donnée de qualité, car le résultat trouvé avec les informations passées en paramètre de la requête ont permis d'affiner au maximum les possibilités de résultats, ou à **FAUX**, si ce dernier était composé de **plus d'un élément**.

Cet objet JSON nous permet donc de peupler les colonnes latitude et longitude de notre table « **coordonnees** », en veillant à récupérer la clé générée par l'**auto-incrémentation** de la colonne « **coordonnes_id** », afin de l'attribuer aux praticiens de santé concernés par le numéro de finess qui est utilisé par le script.

Les valeurs contenues dans les colonnes « **latitude** » et « **longitude** », nous permettent désormais de peupler la colonne « **geom** » de la table « **coordonnes_id** » grâce à une fonctionnalité de PostgreSQL.

Extrait du script qui peuple la table « geom », montrant l'utilisation de la fonctionnalité ST_MakePoint de PostgreSQL pour créer des objets géométriques.

```
try:  
    with engine.connect() as connection:  
        print("Connected to DB")  
        print("-" * 40)  
  
        populate_geom_column_query = text("""  
        UPDATE external.coordonnees SET geom = ST_SetSRID(ST_MakePoint(longitude, latitude), 4326);  
        """)  
        connection.execute(populate_geom_column_query)  
        connection.commit()  
        print("Geom column populated")  
  
except OperationalError as e:  
    print(f"Not connected: {e}")
```

Cette méthode assure que la colonne « **geom** » est peuplée dans la table « **coordonnees** » et mentionne l'utilisation du SRID 4326 pour le système de référence spatiale, en utilisant des coordonnées en degrés de latitude et de longitude **pour représenter des emplacements sur la surface de la Terre.**

Une fois ces étapes terminées, nous disposons de toutes les données nécessaires à la création de notre vue matérialisée, qui servira de table pour le système de recherche.

Extrait du script de création de la vue matérialisée

```
CREATE MATERIALIZED VIEW recherche AS
SELECT p.identification_nationale_pp AS inpp
, p.nom AS nom
, coalesce(multi_replace(lower(public.unaccent(p.nom)), '{" " : " ", "-":""}'::jsonb), '') AS nom_normalise
, p.prenom AS prenom
, coalesce(multi_replace(lower(public.unaccent(p.prenom)), '{" " : " ", "-":""}'::jsonb), '') AS prenom_normalise
, p.libelle_profession AS profession
, coalesce(multi_replace(lower(public.unaccent(p.libelle_profession)), '{" " : " ", "-":""}'::jsonb), '') AS profession_normalise
, p.code_savoirfaire AS savoir_faire
, coalesce(multi_replace(lower(public.unaccent(p.libelle_savoirfaire)), '{" " : " ", "-":""}'::jsonb), '') AS savoir_faire_normalise
, p.libelle_commune AS commune
, coalesce(multi_replace(lower(public.unaccent(p.libelle_commune)), '{" " : " ", "-":""}'::jsonb), '') AS commune_normalise
, c.latitude
, c.longitude
FROM ps_libreaccs_personne_activite p
LEFT JOIN
coordonnees c ON p.coordonnees_id = c.coordonnees_id;
```

Ce script SQL crée une vue matérialisée nommée « **recherche** » qui normalise les données de la table « **ps_libreaccs_personne_activite** » en les enrichissant avec des informations de la table « **coordonnees** » grâce à une jointure basée sur les valeurs des colonnes « **coordonnees_id** » présentes dans les deux tables.

Cette vue est conçue pour optimiser les recherches en fournissant des données normalisées et enrichies, prêtes à être interrogées.

Schéma de la vue matérialisée

recherche	
ABC	inpp
ABC	nom
ABC	nom_normalise
ABC	prenom
ABC	prenom_normalise
ABC	profession
ABC	profession_normalise
ABC	savoir_faire
ABC	savoir_faire_normalise
ABC	commune
ABC	commune_normalise
123	latitude
123	longitude

Avec cette vue matérialisée, il est désormais possible de **créer notre méthode de recherche**, celle-ci va récupérer les données qui lui sont envoyées par la requête effectuée par le système de recherche.

```
def get_praticioner_by_data(cls, form_data):
    if(form_data.get('rangeAround')):
        distance = form_data.get('rangeAround').strip()
    if (form_data.get('position[latitude]') and form_data.get('position[longitude]')):
        lat = form_data.get('position[latitude]')
        long = form_data.get('position[longitude]')
    if(form_data.get('ville')):
        ville = form_data.get('ville').strip()
        ville = ville.lower()

    if(form_data.get('profession')):
        profession = form_data.get('profession').strip()
        profession = profession.lower()
```

Ici nous récupérons les paramètres de la recherche uniquement s'ils sont présents, en veillant à assainir les valeurs qui y sont présentes.

Annexe 16 : Extrait de la suite du code de la méthode recherche

Grâce à SQLAlchemy je réalise mes requêtes en utilisant les variables que j'ai défini grâce aux paramètres de la recherche de mon utilisateur.

Afin de répondre au besoin exprimé par la plateforme Toobib d'assurer une qualité de données optimale, nous mettons à jour les données des résultats qui nous ont été retournés avec l'API FHIR, il a été décidé de l'effectuer dans le front-end afin de rendre plus simple l'affichage des données du back-end ou de l'API FHIR si cette dernière venait à être non disponible.

Pour assurer un temps de réponse optimal, et veiller à répondre au deuxième besoin de la plateforme, celui d'assurer une recherche performante, je réalise les requêtes uniquement sur les résultats de ma page en cours, grâce à la pagination, je porte à un maximum de 12, le nombre d'éléments qui nécessiteront des requêtes par page.

Annexe 17 et 18 : Extraits de code de la pagination et de la mise à jour des données des éléments avec l'API FHIR.

Grâce à cette gestion, je peux afficher les résultats de la recherche en fonction de si l'API FHIR est disponible, je veille donc à ce que mon service reste le plus souvent possible opérationnel et je notifie mon utilisateur si les données présentées proviennent si jamais cette dernière n'est pas disponible et que les résultats présentés sont ceux de l'annuaire de santé et non de l'API FHIR qui elle, contient des données bien plus à jour.

Annexe 19 et 20 : Maquette et interface actuelle des résultats de recherche

6.4 CREATION DE L'API POUR LA GESTION DES IMAGES

Pour le développement de l'API de gestion des images de profil, nous avons choisi de stocker les images sur le serveur et d'intégrer l'identifiant unique généré par Keycloak avant d'envoyer notre requête sur le serveur back-end. Cela nous permet de nous assurer que la requête provient bien d'un utilisateur du site. Nous veillons ensuite à envoyer les informations nécessaires et à construire le nom du fichier.

Extrait de code du comportement du formulaire lors de sa soumission

```
1+ usages  Aline1233
const ChangePhotoForm = ({
  rppsId,
  onPhotoChange,
  token,
  onPhotoUploadError,
}) => {
  return (
    <Formik
      initialValues={{ photo: null }}
      validationSchema={validationSchema}
      onSubmit={async (values : FormikValues, { setSubmitting }) : Promise<void> => {
        const originalFile = values.photo;
        const newFileName : string = `${rppsId}${originalFile.name.substring(originalFile.name.lastIndexOf( searchElement: "."))}`;
        const newFile = new File([originalFile], newFileName, {
          type: originalFile.type,
        });

        const formData : FormData = new FormData();
        formData.append( name: "photo", newFile);
        formData.append( name: "token", token);
        formData.append( name: "id", rppsId);
        const response : ... = await uploadPhoto(formData);
        if (response.error) {
          onPhotoUploadError();
        }
        onPhotoChange();
        setSubmitting( isSubmitting: false);
      }}
    >
```

Annexe 21 : Méthode d'upload de la photo de l'utilisateur sur le serveur

La classe Photos est composée de deux méthodes, la première récupère les paramètres de la requête soumise par hook « useEffect » de la page du profil, le RPPS ID est utilisé pour localiser la photo, le chemin du dossier est construit dynamiquement en fonction de l'environnement dans le quel le script est lancé et la photo est retournée si sa récupération s'est bien déroulée.

Annexe 22 : Méthode de validation du token Keycloak

La deuxième elle, récupère les paramètres de la requête soumise lors de la soumission du formulaire, puis permet le téléchargement d'une image après validation du jeton Keycloak. Si le fichier est valide et présent, il est sauvegardé dans le dossier des photos avec un nom basé sur l'ID RPPS.

7. MISE EN PLACE DE TESTS

Divers tests ont été réalisés avec le framework Jest, framework développé par Facebook, ces tests sont là pour vérifier que le comportement des composants testés est le comportement attendu.

- **Tests unitaires réalisés :**

- **Test unitaire sur la page 404 :** Le test vérifie que le chargement du composant « NotFound » fait bien apparaître et convertit le contenu du fichier Markdown qui lui est associé.
- **Test unitaire sur la page de Cotisations :** Le test vérifie que la page laisse bien apparaître les différents articles présents dans le fichier Markdown et que l'iframe HelloAsso pour la campagne d'adhésion à Toobib est bel et bien généré.
- **Tests unitaires sur le Header et le Footer :** Les tests vérifient que les éléments générés dans les composants Header et Footer sont bels et bien ceux présents dans le fichier « menu.json » et que le fait de cliquer sur les éléments qui y sont présents dirigent bien l'utilisateur sur les bonnes pages.
- **Test unitaire sur la page du profil :** Les tests assurent l'affichage correct des informations disponibles sur la page du profil, ainsi que le fait que plusieurs lieux de consultations soient bel et bien indiqués, quand l'attribut Keycloak de l'utilisateur contient plusieurs lieux de consultations.

Annexe 23 : Extrait de code d'un test avec le framework Jest

Grâce à la réalisation de ces tests et à la création de pipelines CI/CD (Continuous Integration/Continuous Deployment) sur Framagit, ces tests sont effectués à chaque commit sur une branche, permettant de vérifier l'intégrité des composants testés au sein de l'application.

8. REDACTION DE LA DOCUMENTATION

Pour assurer la maintenabilité du projet, plusieurs actions de documentation ont été réalisées durant ma période de stage, notamment la création et la modification de fichiers « README » afin d'offrir une vue d'ensemble du projet ou des modifications effectuées, ce qui est très important lors d'un premier contact avec un projet.

Annexe 24 : Création d'un fichier « README » pour l'API pour la gestion d'images

Annexe 25 : Création d'un fichier « README » pour le thème personnalisé Keycloak

Annexe 26 : Modification du fichier « README » de l'application Toobib

Afin d'améliorer la maintenance, le code a été commenté lors de la mise en place de composants ou de fonctionnalités complexes, l'historique des changements est quant à lui, géré par l'outil Framagit.

9. CONCLUSION

Maintenant arrivé au terme de ma formation de Développeur Web et Web Mobile au sein de l'ENI, il est bon de réfléchir à l'impact positif de ces expériences sur mon parcours. L'objectif initial de la formation était de maîtriser les compétences présentes dans le référentiel pour finalement les appliquer dans un environnement professionnel réel et j'espère avoir pu répondre à ces attentes.

Mon stage chez InterHop a été particulièrement enrichissant. J'ai eu l'opportunité de contribuer à l'évolution de la plateforme Toobib, un projet porteur de sens. Ma principale réalisation a été le développement d'un système de recherche de professionnels de santé, ce qui m'a permis de mettre en pratique mes compétences et d'apporter une valeur ajoutée significative au projet.

Ces expériences ont eu un impact profond sur mon développement professionnel et personnel. Elles m'ont aidé à organiser mon travail de manière plus efficace et à affiner ma méthode de réflexion. Le domaine de la santé a également suscité mon intérêt.

À court terme, je souhaite poursuivre ma formation au sein de l'ENI en alternance avec le cursus Concepteur Développeur d'Applications. Bien que la poursuite de cette alternance ne soit malheureusement pas possible avec InterHop, je reste déterminé à trouver une entreprise où je pourrai continuer à apprendre.

À long terme, mon objectif est de travailler dans un environnement qui a du sens. Le domaine de la santé m'a particulièrement plu pour cette raison, et je souhaite continuer à évoluer dans des projets qui apportent une valeur ajoutée significative à la société.

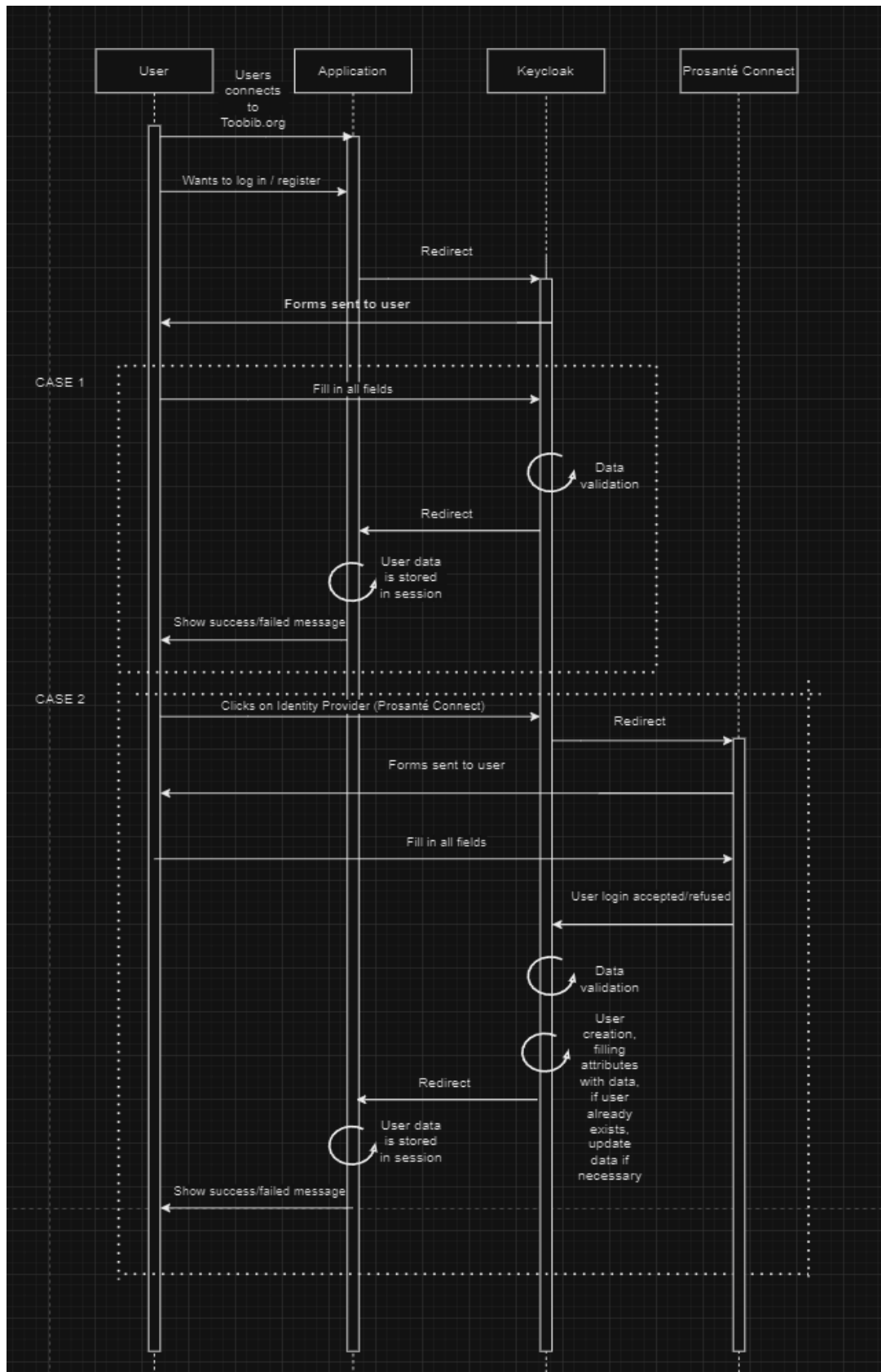
Je profite de cette conclusion pour encore une fois remercier les personnes qui m'ont encadrées durant cette période de stage, l'accompagnement était très complet et les trois personnes qui nous ont accompagnés Alin et moi étaients très complémentaires.

Je remercie également Alin HERCIU, avec qui j'ai maintenant travaillé sur deux projets différents et avec qui il a toujours été agréable de travailler ainsi que Romain BOUGOUIN, Eli N'DIATH, Quélène DELILLE WONG et Hugo L'HOPITAL tous trois élèves de l'ENI, avec qui travailler tout au long de ces huit mois de formation nous a tous chacun poussés un peu plus vers le haut.

[Retrouvez l'article rédigé par Alin et moi pour la plateforme Toobib ici.](#)

10. ANNEXES

ANNEXE 1



ANNEXE 2

```
# https://medium.com/@gantong.eu/docker-compose-file-for-keycloak-with-without-postgresql-deployment-dcf5028829c8
# reverse proxy
# - https://www.keycloak.org/server/reverseproxy
# - https://www.olvid.io/keycloak/installation/

version: '3'

volumes:
  keycloak-db-data:
    driver: local

networks:
  my-network:
    driver: bridge

services:
  # https://www.keycloak.org/server/db
  # https://hub.docker.com/\_/postgres
  postgres:
    image: postgres:15.6
    container_name: postgres-keycloak
    restart: always
    volumes:
      - keycloak-db-data:/var/lib/postgresql/data
    environment:
      POSTGRES_DB: keycloak
      POSTGRES_USER: my_keycloak
      POSTGRES_PASSWORD: ${POSTGRES_PWD}
    networks:
      - my-network

  adminer:
    image: adminer
    container_name: adminer-keycloak
    restart: always
    ports:
      - "${ADMINER_PORT}:8080"
    depends_on:
      - postgres
    networks:
      - my-network
```

ANNEXE 3

```
const ChangePasswordButton = () => {
  const router = useRouter();
  const { data: session } = useSession();

  // Fonction pour rediriger vers la page de changement de mot de passe
  1+ usages  ▲ KIMMYA
  const handleChangePassword = () : void => {
    if (session) {
      const clientID = process.env.NEXT_PUBLIC_CLIENT_ID;
      const redirectURI : string = encodeURIComponent(process.env.NEXT_PUBLIC_KEYCLOAK_REDIRECT_URL_PWD);
      const keycloakServer = process.env.NEXT_PUBLIC_ISSUER;
      const keycloakURL : string = `${keycloakServer}/protocol/openid-connect/auth?client_id=${clientID}&redirect_uri=${redirectURI}&response_type=code&scope=openid&kc_action=UPDATE_PASSWORD`;
      // const keycloakLocalURL = 'http://localhost:9892/realms/Toobii/protocol/openid-connect/auth?client_id=${clientID}&redirect_uri=${redirectURI}&response_type=code&scope=openid&kc_action=UPDATE_PASSWORD';
      router.push(keycloakURL);
    } else {
      router.push(🔗 "/connexion");
    }
  };

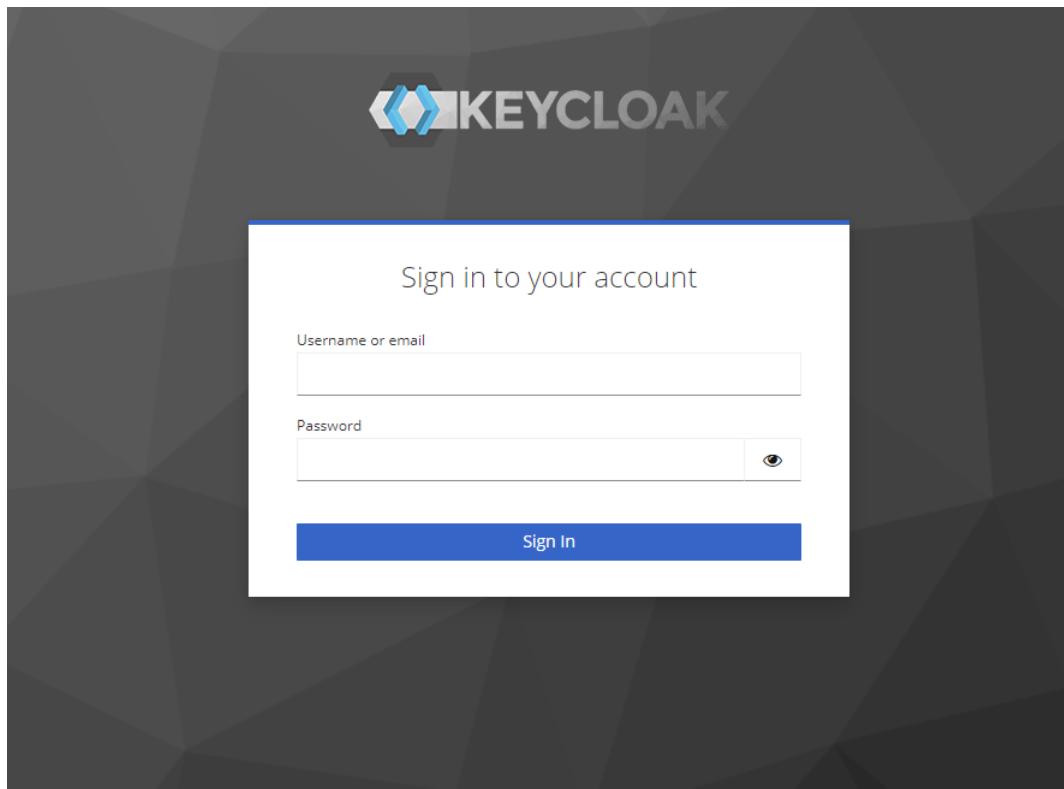
  return (
    <button className="btn btn-primary z-0 py-[14px] md:w-full w-full flex items-center text-center justify-center" onClick={handleChangePassword}>Changer le mot de passe</button>
  );
};

1+ usages  ▲ KIMMYA
export default ChangePasswordButton;
```

ANNEXE 4

- ▼ login
 - ▼ messages
 - 📁 Resource Bundle 'messages'
 - ⚙ messages_en.properties
 - ⚙ messages_fr.properties
 - ▼ resources
 - > css
 - > img
 - > scripts
 - 📄 error.ftl
 - 📄 login.ftl
 - 📄 login-otp.ftl
 - 📄 login-reset-password.ftl
 - 📄 login-update-password.ftl
 - 📄 register.ftl
 - 📄 template.ftl
 - ⚙ theme.properties
- 📄 README.md

ANNEXE 5



ANNEXE 6

Cotisations

Pourquoi cotiser ?

Toobib oeuvre pour des communs numériques en santé et promeut une utilisation des données de santé éthique et transparente, respectueuse du droit à la protection des données personnelles dont les données couvertes par le secret médical sont une composante particulièrement sensible.

Toobib fait de la sensibilisation et de la formation auprès des professionnels de santé sur les enjeux du traitement des données de santé.

Toobib.org promeut, développe et met à disposition des outils, documents, guides, logiciels libres et open-sources par et pour les professionnel.le.s de santé.

Par virement bancaire

Des virements de banque à banque sont possibles sans frais : renseigne-toi auprès de ta banque !

Domiciliation : CRÉDIT COOPÉRATIF

Identification du compte pour une utilisation nationale

Banque : 42559
Guichet : 10000
Compte : 08026979059
Clé RIB : 32
BIC : CCOPFRPPXXX

Identification du compte pour une utilisation internationale (IBAN)

FR76 4255 9100 0008 0269 7905 932

Via Hello Asso

0 adhérent

Choix de l'adhésion

Adhérents

Coordonnées

Récapitulatif

Association

120€

-

0

+

Souhaitez-vous faire un don à TOOBIB en plus de votre adhésion ?

☒ Pas de don

☐ 5 €


☐ 10 €

☐ 20 €

☐ Montant de votre choix

Montant à payer pour la durée de l'adhésion :


0 €



HelloAsso est une entreprise sociale et solidaire, qui fournit gratuitement ses technologies de paiement à l'organisme [TOOBIB](#). Une contribution au fonctionnement de HelloAsso, modifiable et facultative, vous sera proposée avant la validation de votre paiement.

Étape suivante >

CRÉÉ ET DIFFUSÉ AVEC

 helloasso

ANNEXE 7

```
const { width, height } = useWindowSize();

if (width < 425) {
  return (
    <>
    <section className="section">
      <div className="container h-full shadow">
        <h1 className="text-center font-normal">{title}</h1>
        <div className="section p-12">
          {articles.map((article, index) => (
            <div className="py-6" key={index}>
              {markdownify(article.title, {tag: "h3", className: "mb-5"})}
              {article.description && markdownify(article.description, {tag: "p"})}
              {article.descriptions[0].text !== "helloasso" &&
                article.descriptions &&
                markdownify(article.descriptions[0].text, {tag: "p"})}
              {article.descriptions[0].text === "helloasso"}
            </div>
          ))}
        </div>
        <iframe
          id="haWidget"
          className="w-full h-full fadeIn scrollingNo"
          data-testid="helloasso-iframe"
          src="https://www.helloasso.com/associations/toobib/adhesions/cotisation-toobib/widget"
          allowFullScreen
        ></iframe>
      </div>
    </section>
    </>
  );
};
```


ANNEXE 8

Docteur GUY Martin

INFIRMIER

+33 22 11 55 66 44
Fanti.pierre@truc.com

Conventionnement :



Adresse

Cabinet médical George Sand
12 rue Camille Claudel
75002 Paris

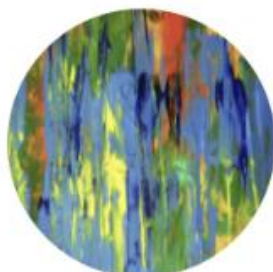
Informations supplémentaires

4e étage avec ascenseur
Accès PMR



ANNEXE 9

Mon profil



Modifier mon profil

Changer le mot de passe

Modifier la photo de profil

Choisir un fichier

Aucun fichier choisi

Téléchargement

Adrien PARROT

Orthophoniste

Numéro RPPS :

Numéro de téléphone :

Adresse e-mail : adrien.parrot@caramail.fr

Conventionnement :

ANNEXE 10

Rectifier mes données

Nom

Prénom

Adresse Électronique

Téléphone

Photo

Conventionnement

Adresse

Lieu de consultation

Adresse

Code Postal

Ville

Informations complémentaires

Téléphone du lieu de consultation

Étage

Ascenseur

Accès handicapés

Autre

Veux-tu envoyer ces informations à l'organisme tuteur ?

Notre Délégué à la Protection des Données s'en charge pour toi :)

Envoyer

ANNEXE 11

Modifier mon profil

Remplir le formulaire grâce à
l'annuaire des professionnel.les de
santé

Informations personnelles

Prénom

Nom de famille

Profession

Numéro RPPS

Numéro de téléphone

ANNEXE 12

Trouve ton Toobib | Ville  

ANNEXE 13

Rechercher un professionnel de santé

Nom du Toobib
Médecin
Veuillez sélectionner la spécialité du médecin
Spécialité
Autour de moi



Dans un rayon de : 27 km

ANNEXE 14

```
download:
echo "download csv ..."
echo "from service.annuaire.sante.fr ..."
rm -f download/*
wget --no-check-certificate https://service.annuaire.sante.fr/annuaire-sante-webservices/V300/services/extraction/PS_LibreAcces -P download/
wget --no-check-certificate https://service.annuaire.sante.fr/annuaire-sante-webservices/V300/services/extraction/Porteurs_CPS_CPF -P download/
wget --no-check-certificate https://service.annuaire.sante.fr/annuaire-sante-webservices/V300/services/extraction/Extraction_Correspondance_MSSante -P download/
mv download/PS_LibreAcces download/PS_LibreAcces.zip
mv download/Porteurs_CPS_CPF download/Porteurs_CPS_CPF.zip
mv download/Extraction_Correspondance_MSSante download/Extraction_Correspondance_MSSante.zip
unzip download/PS_LibreAcces.zip -d download/
unzip download/Porteurs_CPS_CPF.zip -d download/
unzip download/Extraction_Correspondance_MSSante.zip -d download/

correction-csv:
sh correction.sh

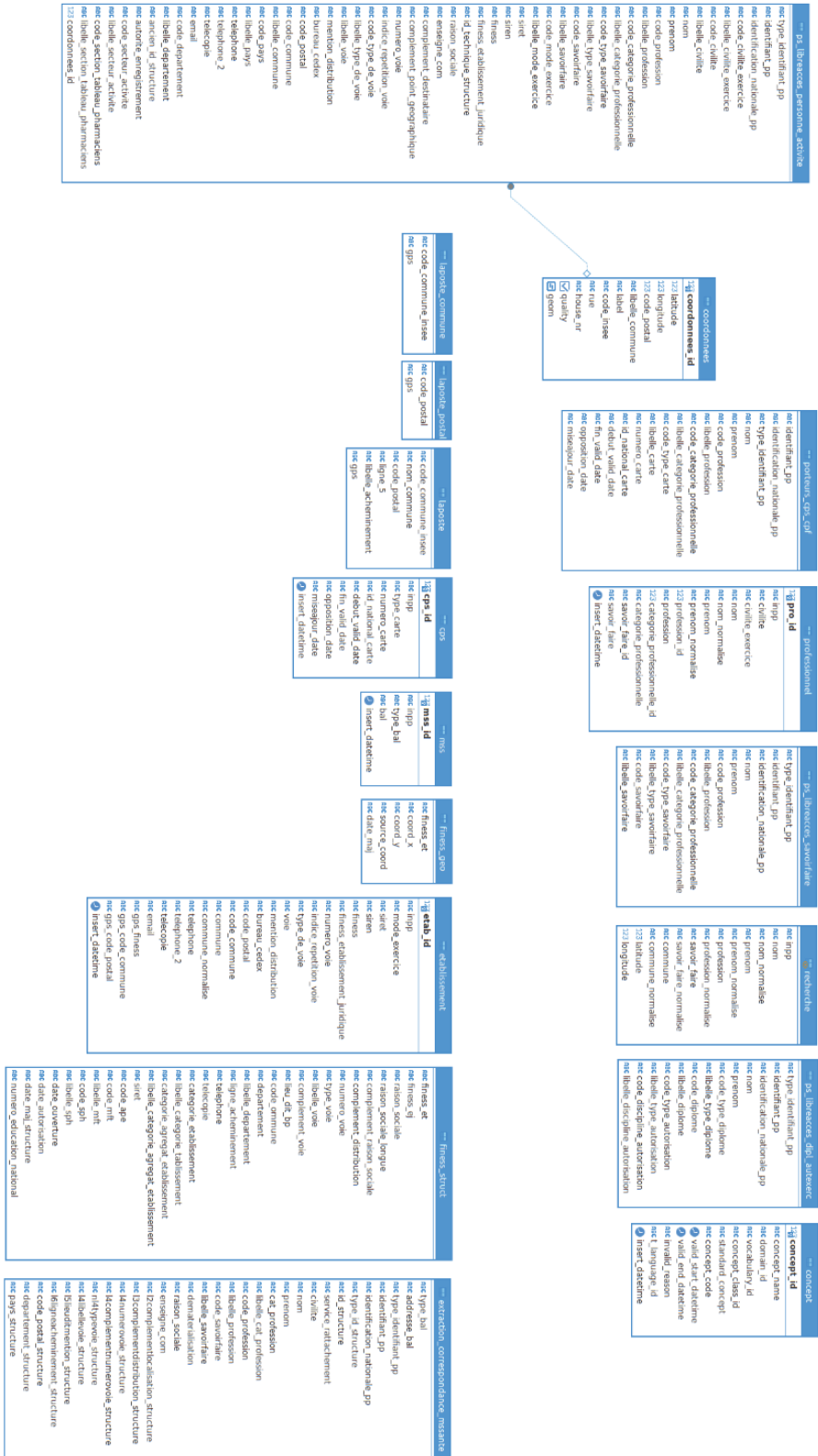
external-create-schema:
psql "${PG_CONF}" -c 'DROP SCHEMA IF EXISTS "${EXTERNAL_SCHEMA}" CASCADE;'
psql "${PG_CONF}" -c 'CREATE SCHEMA "${EXTERNAL_SCHEMA}";'
echo "${EXTERNAL_DB_ARGUMENTS}"
psql "${EXTERNAL_DB_ARGUMENTS}" -f "external-create-table.sql"
psql "${EXTERNAL_DB_ARGUMENTS}" -f "external-constraints.sql"
psql "${EXTERNAL_DB_ARGUMENTS}" -f "functions.sql"

external-load-row:
echo "Load csv in db..."
psql "${EXTERNAL_DB_ARGUMENTS}" -f "external-load-row-data.sql"

external-load-api:
echo "Build api table..."
psql "${EXTERNAL_DB_ARGUMENTS}" -f "external-load-api.sql"

external-load-coordonnees:
echo "Build coordonnees table..."
psql "${EXTERNAL_DB_ARGUMENTS}" -f "coordonnees_table.sql"

create-materialized-view:
psql "${MATERIALIZED_VIEW_ARGUMENTS}" -f "load-materialized-view.sql"
```



ANNEXE 16

```
if form_data.get('toobib'):
    name = form_data.get('toobib').strip()
    name = name.split(' ')
    if len(name) == 2:
        nom, prenom = name
        query = query.filter(and_(
            cls.nom.ilike(nom),
            cls.prenom.ilike(prenom),
        ))
    elif len(name) == 1:
        nom = name[0]
        query = query.filter(or_(
            cls.nom.ilike(nom),
            cls.prenom.ilike(nom),
        ))

if form_data.get('ville'):
    query = query.filter(or_(
        cls.commune.ilike(ville),
    ))

if form_data.get('profession'):
    query = query.filter(or_(
        cls.profession.ilike(profession),
    ))

if(form_data.get('specialty')):
    specialty = form_data.get('specialty').strip()
    print(specialty)

    csv_file_path = 'app_flask/data/specialties.csv'
    code_savoirfaire_list = fetch_specialties_by_code_ref(specialty, csv_file_path)

    query = query.filter(cls.code_savoirfaire.in_(code_savoirfaire_list))

if (form_data.get('rangeAround') and form_data.get('position[latitude]') and form_data.get('position[longitude]')):
    search_point = geofunc.ST_SetSRID(
        geofunc.ST_MakePoint(long, lat),
        4326
    ).cast(Geography)
    distance_column = geofunc.ST_Distance(cls.geom, search_point).label('distance')
    query = query.add_columns(distance_column).filter(
        geofunc.ST_DWithin(
            cls.geom,
            search_point,
            int(distance) * 1000
        )
    )
```


ANNEXE 17

```
useEffect( effect: () : void => {  
  if (results) {  
    1+ usages 1 KKMYA +1  
    const fetchPageData = async () : Promise<void> => {  
      const indexOfLastResult : number = currentPage * resultsPerPage;  
      const indexOfFirstResult : number = indexOfLastResult - resultsPerPage;  
  
      const inppArray = results  
        .slice(indexOfFirstResult, indexOfLastResult)  
        .map((result) => result.inpp);  
  
      const fhirResults : (Awaited<...>){} = await Promise.all(  
        inppArray.map(async (inpp) : Promise<any> => {  
          const fhirResult : any = await getPractitionerAndPractitionerRole(inpp);  
          return fhirResult;  
        }  
      ),  
    );  
  
    for (let i : number = 0; i < fhirResults.length; i++) {  
      if (fhirResults[i] && results[i]) {  
        fhirResults[i].latitude = results[i].latitude;  
        fhirResults[i].longitude = results[i].longitude;  
      }  
    }  
  
    const practitioners : any[] = await CreatePractitioners(fhirResults);  
  
    setCurrentResults(practitioners);  
  };  
  
  fetchPageData();  
}, deps: [currentPage, results]);
```

ANNEXE 18

```
for (let i : number = 0; i < practitionersBundle.length; i++) {

  firstName =
    practitionersBundle[
      i
    ]?.data?.entry?.[1]?.resource?.extension?.[0]?.valueHumanName?.given?.[0].toUpperCase();
  familyName =
    practitionersBundle[
      i
    ]?.data?.entry?.[1]?.resource?.extension?.[0]?.valueHumanName?.family.toUpperCase();

  if (!familyName) {
    familyName = "non renseigné";
  }

  if (!firstName) {
    firstName = "non renseigné";
  }

  practitionersBundle[i]?.data?.entry?.[1].resource.code[0].coding.forEach(
    (code) : void => {
      if (parseInt(code.code)) {
        jobId = code.code;
      }
    },
  );

  const profession = await getProfessionString(jobId);

  if (
    practitionersBundle[i].data?.entry?.[1].resource.specialty?.[0] !==
    undefined
  ) {
    const jobSpecialty =
      practitionersBundle[i].data?.entry?.[1].resource.specialty[0]?.coding[0]
        .code;
    specialtyString = await getSpecialtyString(jobSpecialty);
  } else {
    specialtyString = null;
  }

  if (practitionersBundle[i].inpp !== undefined) {
    inpp = practitionersBundle[i].inpp;
  }
}
```

ANNEXE 19

Résultats de votre recherche :

TUTTI Gaël
AUDIOPROTHÉSISTE
Paris
Consulter

JOSS Loïc
PSYCHOLOGUE
Paris
Consulter

TIO Cindy
SAGE-FEMME
Paris
Consulter

MARTIN Denis
OSTÉOPATHE
Paris
Consulter

LOCA Max
OPHTALMOLOGUE
Paris
Consulter

JACK Tom
ORTHOPHONISTE
Paris
Consulter

TUTTI Gaël
AUDIOPROTHÉSISTE
Paris
Consulter

GUY Martin
INFIRMIER
Paris
Consulter

ANNEXE 20

Rechercher un professionnel de santé

Nom du Toobib

Médecin

Veuillez sélectionner la spécialité du médecin

Spécialité

Autour de moi

Dans un rayon de : 20 km

Rechercher

8 résultats pour votre recherche

**HADJOUEL
AMBRE**
Médecin à OMBREE D
ANJOU
Qualifié en Médecine
générale (SM)
Consulter

**KURTA MAUPAS
DOMINIQUE**
Médecin à PLECHATEL
Qualifié en Médecine
générale (SM)
Consulter

**GENDRY
LYDIE**
Médecin à CRAON
Spécialiste en Médecine
générale (SM)
Consulter

**LE PONNER
CHRISTIAN**
Médecin à MARTIGNE
FERCHAUD
Spécialiste en Médecine
générale (SM)
Consulter

ANNEXE 21

```
from flask_restful import Resource, reqparse
from flask import send_from_directory, request
import os
from werkzeug.utils import secure_filename
from ..helpers.keycloak import validate_keycloak_token

class Photos(Resource):
    def get(cls):
        request_args = request.args
        if(request_args.get('id')):
            id = request_args.get('id')
            # Get the directory of the current script
            script_dir = os.path.dirname(os.path.realpath(__file__))
            # Get the parent directory
            app_dir = os.path.dirname(script_dir)
            # Construct the path to the photos directory
            folder_path = os.path.join(app_dir, 'photos')

            for filename in os.listdir(folder_path):
                if os.path.splitext(filename)[0] == id:
                    return send_from_directory(folder_path, filename)

            return 'Photo not found', 404
        else:
            return 'No id', 404
    def post(cls):
        #validate token
        token_value = request.form.get('token')
        if validate_keycloak_token(token_value) == False:
            return 'Error', 400

        if 'photo' not in request.files:
            return 'No photo part', 400
        file = request.files['photo']

        if file.filename == '':
            return 'No selected file', 400

        rpps_id = request.form.get("id")

        script_dir = os.path.dirname(os.path.realpath(__file__))
        app_dir = os.path.dirname(script_dir)
        folder_path = os.path.join(app_dir, 'photos')
        if not os.path.exists(folder_path):
            os.makedirs(folder_path)

        filename = secure_filename(file.filename)
        name, ext = os.path.splitext(filename)
        filename = f"{rpps_id}{ext}"
        file_path = os.path.join(folder_path, filename)
        file.save(file_path)

        return {'message': 'Photo uploaded successfully', 'filename': filename}, 201
```

ANNEXE 22

```
import requests
import os
def validate_keycloak_token(token):

    keycloak_host = os.getenv("KEYCLOAK_HOST")
    realm_name = os.getenv("REALM_NAME")
    client_id = os.getenv("CLIENT_ID")
    client_secret = os.getenv("CLIENT_SECRET")

    url = f"https://{keycloak_host}/realms/{realm_name}/protocol/openid-connect/token/introspect"

    payload={
        "token": token,
        "client_id": client_id,
        "client_secret": client_secret
    }
    try:
        response = requests.post(url, data=payload)
        result = response.json()
        return result.get("active", False)
    except Exception as e:
        print(f"Error validating token: {e}")
    return False
```

ANNEXE 23

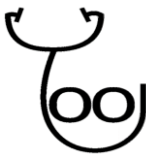
```
import { render, screen } from "@testing-library/react";
import "@testing-library/jest-dom";
import NotFound from "@layouts/404";

describe("NotFound", () => {
    const fakeData = {
        frontmatter: { title: "Test Title" },
        content: "Test Content",
    };

    it("renders without crashing", () => {
        render(<NotFound data={fakeData} />);
        expect(
            screen.getByRole("heading", { name: "Test Title" }),
        ).toBeInTheDocument();
    });

    it("renders content correctly", () => {
        render(<NotFound data={fakeData} />);
        expect(screen.getByText("Test Content")).toBeInTheDocument();
    });

    it("converts markdown content correctly", () => {
        const fakeDataMd = {
            ...fakeData,
            content: "**Test** Content",
        };
        render(<NotFound data={fakeDataMd} />);
        expect(screen.getByText("Test").closest("strong")).toBeInTheDocument();
        expect(screen.getByText("Content")).toBeInTheDocument();
    });
});
```



ANNEXE 24

Photo API

This is a Flask API designed for managing user profile pictures on Toobib.org.

Table of Contents

- 1. Getting Started
- 2. Installation
- 3. Configuration
- 4. Running the Project

Getting Started

This section will guide you through the setup process to get the project up and running on your local machine.

Installation

Before you start, make sure you have Python installed on your machine. Then, install the required libraries by running the following commands:

```
pip install flask
pip install flask_restful
pip install flask_cors
pip install requests
pip install python-dotenv
```

Configuration

Before running the repository, create a .env file and add the following configurations. These are required for the API to communicate with Keycloak:

```
KEYCLOAK_HOST=" "
REALM_NAME=" "
CLIENT_ID=" "
CLIENT_SECRET=" "
```

Running the Project

```
flask --app __init__ run
```



ANNEXE 25

TOOBIB THEME

Getting started

Any file duplicated in the custom theme from the base theme will overwrite the base file located in the default theme.

The file `template.ftl` represents the model replicated on each page; it is imported by all `.ftl` files (except the template one) with:

```
1 #import "template.ftl" as layout
```

The files in the `messages` folder contain all the resources called by the "nested" function in the `.ftl` files.

Resource to learn more about the structure of themes: [Customizing Keycloak Themes](#)

Various JS scripts (such as the `passwordVisibility` script, which allows displaying or hiding the password in the input field) are present in Keycloak's base folders but can also be included in the customized theme, customized scripts are available at `pscToobib2/login/resources/scripts/rppsValidation.js`.

Most of the CSS used for this theme has been moved to the bottom of the CSS file found at `import/theme/pscToobib2/login/resources/css/psc.css`

Example of code

Button with `passwordVisibility` function:

```
<button class="${properties.kcFormPasswordVisibilityButtonClass}" kc-form-password-visibility-button" type="button" aria-label="${msg('showPassword')}"
  aria-controls="password" data-password-toggle tabindex="4"
  data-icon-show="${properties.kcFormPasswordVisibilityIconShow}" data-icon-hide="${properties.kcFormPasswordVisibilityIconHide}"
  data-label-show="${msg('showPassword')}" data-label-hide="${msg('hidePassword')}">
  <i class="${properties.kcFormPasswordVisibilityIconShow}" aria-hidden="true"></i>
</button>
```

JS file to import to access `passwordVisibility` function: `<script type="module" src="${url.resourcesPath}/js/passwordVisibility.js"></script>`

Other resources

- [Spring Security with Keycloak and Custom Themes](#)
- [Custom Themes in Keycloak](#)

Additionally, several options can be activated in the Keycloak administration panel.

For example, you can enable the dropdown menu to switch between languages by going to Realm Settings > Localization, activating Internationalization, and then selecting the languages that will be available on your Keycloak pages.

ANNEXE 26

- Configurer le fichier `.env.local`

Afin de s'assurer du bon fonctionnement de toutes les fonctionnalités de Toobib, on s'assure de la bonne configuration du fichier `.env.local` contenant les variables d'environnement.

```
#LOCALHOST KEYCLOAK (Utilisé en environnement local)

KEYCLOAK_CLIENT_ID="CLIENT_NAME" (à créer dans le panel d'administration Keycloak (Client > Create Client))
KEYCLOAK_CLIENT_SECRET="CLIENT_KEY" (à récupérer dans le panel d'administration Keycloak (Client > CLIENT_NAME > Credentials > Client Secret))
KEYCLOAK_ISSUER="https://<KEYCLOAK_SERVER_URL>/realms/<REALM_NAME>" (URL utilisée par exemple dans la redirection vers la fonction Keycloak)
NEXT_PUBLIC_CLIENT_ID="CLIENT_NAME" (Variable d'environnement publique utilisée par exemple dans la configuration de la redirection du changement de mot de passe)
NEXT_PUBLIC_ISSUER="https://<KEYCLOAK_SERVER_URL>/realms/<REALM_NAME>" (Variable d'environnement publique de l'URL utilisée par exemple dans la redirection vers la fonction Keycloak)

#ONLINE KEYCLOAK (Utilisé en environnement de test/de production)

KEYCLOAK_CLIENT_ID="CLIENT_NAME" (à créer dans le panel d'administration Keycloak (Client > Create Client))
KEYCLOAK_CLIENT_SECRET="CLIENT_KEY" (à récupérer dans le panel d'administration Keycloak (Client > CLIENT_NAME > Credentials > Client Secret))
KEYCLOAK_ISSUER="https://<KEYCLOAK_SERVER_URL>/realms/<REALM_NAME>" (URL utilisée par exemple dans la redirection vers la fonction Keycloak)
NEXT_PUBLIC_CLIENT_ID="CLIENT_NAME" (Variable d'environnement publique utilisée dans la configuration de la redirection du changement de mot de passe)
NEXT_PUBLIC_ISSUER="https://<KEYCLOAK_SERVER_URL>/realms/<REALM_NAME>" (Variable d'environnement publique de l'URL utilisée par exemple dans la redirection vers la fonction Keycloak)

#FHIR KEY
NEXT_PUBLIC_FHIR_KEY="FHIR_KEY" (Variable d'environnement utilisée pour la récupération des informations FHIR)

#REDIRECT URL's (à valider pour chaque client Keycloak sur le panel d'administration Keycloak (Clients > CLIENT_NAME > Settings > Valid redirect URL))
NEXT_PUBLIC_KEYCLOAK_REDIRECT_URL_PWD="http://<APP_SERVER_URL>/mdp-modifie" (Variable d'environnement publique de l'URL de redirection vers la fonction Keycloak)
```