

Rapport de Stage

**Concepteur Développeur d’Applications**



**AFPA**

407 Avenue de la Gironde

59640 DUNKERQUE

[**https://www.afpa.fr**](https://www.afpa.fr)

*Période en centre de formation :*

**18/10/2022** au **03/03/2023**

**IEMN**

Avenue Henri Poincaré

59650 VILLENEUVE D’ASCQ

**https://www.iemn.fr**

*Période en stage :*

**06/03/2023** au **02/06/2023**

Table des matières

[Remerciements - 3 -](#_Toc136611541)

[IEMN - 4 -](#_Toc136611542)

[Présentation de l’entreprise - 4 -](#_Toc136611543)

[Présentation de l’équipe - 5 -](#_Toc136611544)

[Environnement de travail - 6 -](#_Toc136611545)

[Outils Utilisés - 6 -](#_Toc136611546)

[Technologies Utilisées - 7 -](#_Toc136611547)

[Présentation du projet - 9 -](#_Toc136611548)

[Présentation du projet en français - 9 -](#_Toc136611549)

[Présentation du projet en anglaise / Project presentation in english - 10 -](#_Toc136611550)

[Tâches effectuées - 11 -](#_Toc136611551)

[Duplication de projets et changement de termes pour certains boutons - 11 -](#_Toc136611552)

[Ajout de tests unitaires - 14 -](#_Toc136611553)

[Système de logs - 16 -](#_Toc136611554)

[Relance de projet par mail - 17 -](#_Toc136611555)

[Traduction de l’application - 21 -](#_Toc136611556)

[Responsive de certaines parties de l’application - 22 -](#_Toc136611557)

[Conclusion - 25 -](#_Toc136611558)

# Remerciements

Je tiens tout d’abord à remercier ma formatrice Mme Martine POIX pour sa pédagogie, son soutient et sa disponibilité lors de la formation. Ainsi que mes collègues de formation pour leurs aides et les bons moments de camaraderies apportés lors de la période de formation.

Je remercie également Mr Billel GUERBOUKHA qui fut mon tuteur lors de mon stage ainsi que Mr Benjamin LECHA pour leurs aides et conseils lors de mon apprentissage des technologies utilisées dans l’entreprise ainsi que pour leurs disponibilités tout au long du stage.

Et je remercie pour finir Mr Dino EL HANI, Mr Nicolas GOGUILLON, Mlle Lucie CAUDRON, Mr Antoine LEVA et Mr Julien PRONIER pour leurs aides lors du stage mais aussi pour les bons moments apportés lors de celui-ci.

# IEMN

## Présentation de l’entreprise

L’IEMN (Institut d’Electronique, de microélectronique et de nanotechnologie) fondé en 1992 est une UMR (Unité Mixte de Recherche) du CNRS (Centre National de la Recherche Scientifique). L’IEMN est un établissement partenaire de plusieurs écoles et universités notamment l’université de Lille et l’université de Valenciennes avec plusieurs programmes de recherches et d’enseignements communs. Depuis 2003 l’IEMN fait également parti du réseau RENATECH qui est le réseau académique Français des équipements de pointe dans le domaine de la micro et nanotechnologie piloté par le CNRS. Le réseau RENATECH est composée de 5 grandes centrales de micro et nanotechnologies : le C2N (Paris-Saclay), l’IEMN (Lille), le LAAS (Toulouse), le LTM (Grenoble), FEMTO-ST (Besançon).

L’IEMN fait également partie du réseau RENATECH+ qui est constituée du réseau RENATECH (les 5 grandes centrales citées précédemment) ainsi que de 27 centres dits « régionaux » de plus petite taille. Chaque centrale possède sa spécialité et ses propres ressources.

## Présentation de l’équipe

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquementUne image contenant texte, clipart

Description générée automatiquementUne image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquementUne image contenant texte, clipart

Description générée automatiquementUne image contenant texte, clipart

Description générée automatiquementUne image contenant texte, clipart

Description générée automatiquement

**Julien PRONIER**

**Antoine LEVA**

**Nicolas GOGUILLON**

*Développeur*

**Billel GUERBOUKHA**

*Développeur*

**Lucie CAUDRON**

**Yoan DECONINCK**

**Dino El HANI**

*Développeur*

**Benjamin LECHA**

*Chef de projet*

# Environnement de travail

## Outils Utilisés



Gitlab nous sert pour le versioning du projet. Gitlab possède des outils tels qu’une liste et un tableau pour pouvoir créer des tâches comme des fix, features, etc… par les différents membres du projet. Gitlab possède également différents outils collaboratifs pour les tâches comme une liste ou un tableau pour affecter les tâches du projet aux différents développeurs.

Zulip nous a servi pour la communication, c’est un logiciel qui peut être téléchargé sur un poste de travail mais qui est également accessible sur un navigateur internet ce qui fait que l’on peut l’utiliser sur n’importe quel système d’exploitation et sur une grande variété d’appareils (mobile, tablette, ordinateur). Sur Zulip on dispose de plusieurs canaux pour pouvoir communiquer entre nous sur différents sujets, d’envoyer des messages privés aux autres utilisateurs et de pouvoir s’envoyer des captures d’écrans.

Pour pouvoir coder nos différentes fonctionnalités, j’avais le choix entre VS Code et PyCharm. J’ai eu l’occasion d’utiliser les 2 lors de mon stage. Chaque IDE possède ses propres avantages : PyCharm est meilleur pour les débutants sur Python car c’est un IDE spécialisé dans ce langage et VS Code est plus versatile grâce aux nombreuses extensions de ses créateurs et de la communauté.

## Technologies Utilisées





Python est un langage de programmation open-source dit interprété. Python possède une syntaxe « simple » par rapport à d’autres langages, c’est l’indentation qui limite les différents blocs de code. Il favorise la programmation structurée, fonctionnelle et orienté objet. C’est également une technologie multiplateforme qui possède une large communauté et qui est utilisée dans plusieurs domaines tels que le développement web, la data science, l’automatisation de tâches, la création de jeux et bien d’autres.

Flask est un framework Python qui est classé comme microframework car il est très léger et a pour objectif de garder un noyau simple mais extensible. En effet il ne possède pas de système d’authentification, de validation de formulaire ou autre. Mais de nombreux paquets nous permettent facilement d’ajouter ces fonctionnalités. Ces paquets sont installables en ligne de commande. On peut installer tous les paquets d’un projet en une seule ligne de commande s’ils sont listés dans un fichier de « requirements ».

Docker est une plateforme de conteneurisation qui permet de créer des conteneurs pour nos applications. C’est une alternative aux machines virtuelles. Par rapport aux machines virtuelles, un conteneur est plus léger, plus simple et partage son kernel et son système d’exploitation avec d’autres conteneurs, ce qui est un avantage non négligeable. Docker permet grâce aux conteneurs d’avoir le même environnement sur toutes les machines où l’application est déployée et développée. Docker est aussi une très bonne solution pour créer des applications micro service grâce à l’indépendance des différents conteneurs.



AJAX de son nom complet **Asynchronous Javascript And XML** permet d’envoyer des requêtes JavaScript asynchrones ce qui permet une mise à jour de contenu sans rechargement de page.

JavaScript permet d’implémenter des mécaniques complexes à une page web.

Pour faire le responsive de l’application, j’ai utilisé Bootstrap 4.

HTML5 (**HyperText Markup Language**) est utilisé pour créer la structure et le contenu des pages web et CSS3 (**Cascading Style Sheets**) est utilisé pour soigner l’apparence de nos différentes pages.

# Présentation du projet

## Présentation du projet en français

J’ai été positionné sur le projet Repotech, c’est une application de gestion de projet. Cette application sert à déposer des projets de recherche dans différentes centrales du réseau RENATECH. L’application est composée de 2 interfaces différentes : l’interface administrateur et l’interface utilisateur. Pour accéder à l’application il faut obligatoirement avoir un compte. Il y a 3 types d’utilisateurs pour permettre de séparer l’accès à ces 2 interfaces : l’administrateur national, les administrateurs locaux et les utilisateurs.

L’administrateur national et les administrateurs locaux ont accès aux 2 interfaces tandis que les utilisateurs ont uniquement accès à l’interface utilisateur.

Dans l’interface utilisateur on peut effectuer une demande de faisabilité de projet si on ne connait pas la centrale qui effectuera le projet. Cette demande pourra être confirmée ou infirmée par un administrateur. On peut également consulter nos demandes de faisabilités avec un historique si elles sont refusées, acceptées ou en attente de confirmation. Si à contrario on connait la centrale qui effectuera le projet, on peut déposer un projet. Dans ce projet on pourra affecter des utilisateurs, partenaires et créer des étapes.

Dans l’interface administrateur on a 2 types d’administrateurs : l’administrateur local qui peut uniquement interagir avec les projets de sa centrale et l’administrateur national qui lui peut interagir avec les projets de toutes les centrales du réseau RENATECH. Les administrateurs peuvent valider au non les faisabilités, modifier les projets, les dupliquer, les supprimer et effectuer des relances par mail aux demandeurs et project leaders directement depuis l’application pour les différents projets. Ils peuvent également effectuer des migrations d’établissements et d’utilisateurs en cas de doublon. L’application est développée par l’IEMN un institut de recherche du CNRS pour RENATECH et elle est développée en Python-Flask et Jinja pour faire des templates dans le front. Mon but a été d’ajouter des fonctionnalités dans l’application, d’effectuer des requêtes vers l’API REST de l’application et d’améliorer le design et l’ergonomie de l’application pour les utilisateurs finaux.

## Présentation du projet en anglaise / Project presentation in english

I was positionned on the Repotech Project, this is a project management application. This application is used to deposit research projects in differents RENATECH network plants. The application is composed of 2 differents user interfaces: the admin interface and the user interface. To access the application, you need to have an account. There are 3 types of users to separate the access to those two interfaces: the national admin, the local admins and the users.

The national and local admins have access to both interfaces while the users only have access to the user interface.

In the user interface you can ask the network for a feasibility request if you do not know the plant that will take care of the project. This demand can be confirmed or refused by an admin. You can also consult your feasibilities demands with an history to see if they are refused, accepted or pending confirmation. But if you already know with plant will do your project you can directly do a project deposit. In this project you will be able to affect users, partners and create steps.

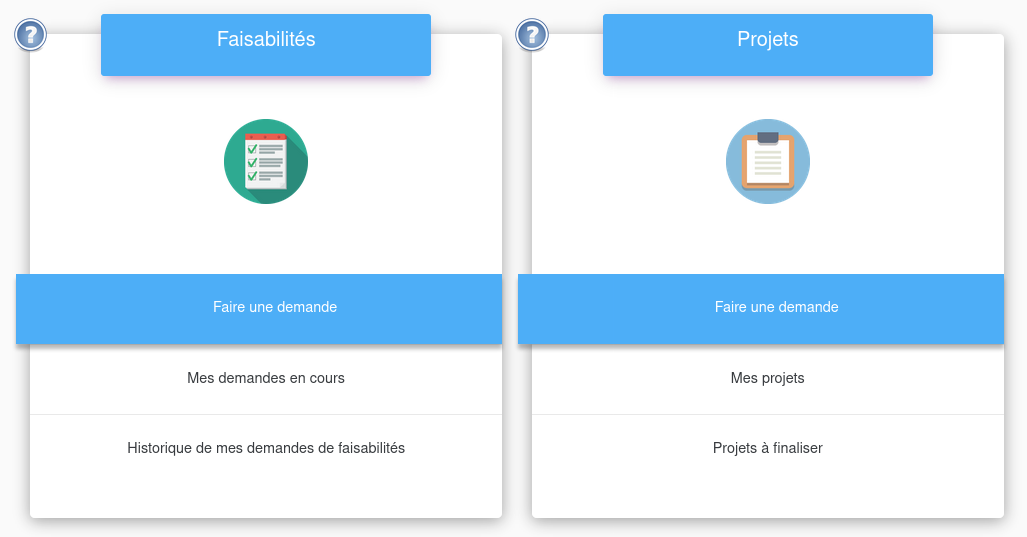
In the administrator interface, we have two types of admins: the local admin who can only interact with projects from his plant and the national admin who can interact with every project existing from every plant of the RENATECH network. The administrators can validate or not the feasibilities requests, modify the projects, delete them, send a reminder email about the projects to the applicant and the project leaders directly from the application. They can also migrate establishments and users in case of duplicates. The application is developped by the IEMN a research institute of the CNRS for RENATECH and the app is developped in Python-Flask and jinja to make templates in the front-end. My role was to add multiple functionalities in the application, to make requests to the API REST of the app and to upgrade the design and ergonomics of the application for the end user.

# Tâches effectuées

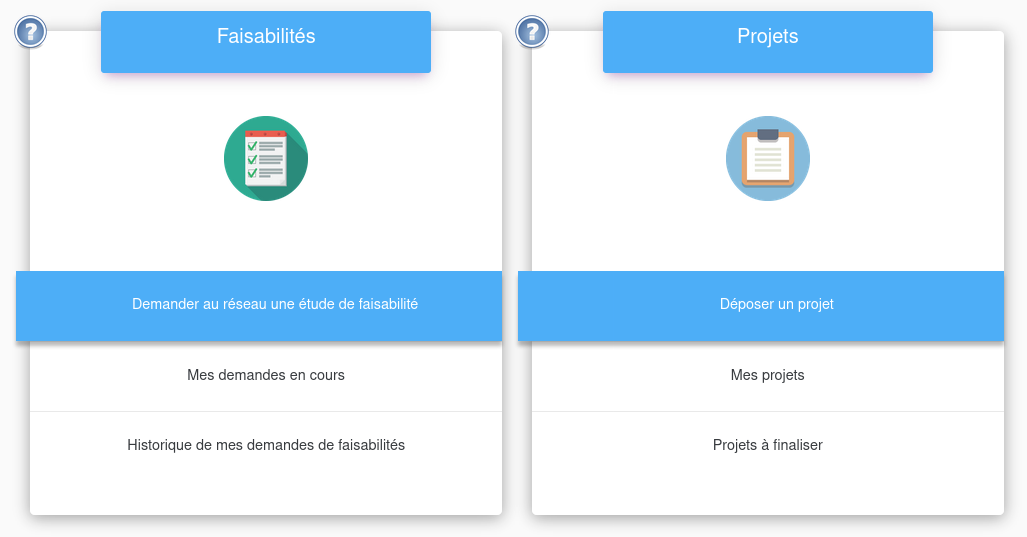
## Duplication de projets et changement de termes pour certains boutons

Après avoir passé environ une semaine pour m’habituer à l’environnement de travail (Python, Docker) de l’entreprise mais aussi pour apprendre Python-Flask grâce à la documentation de celui-ci mais aussi à un tutoriel conseillé par l’entreprise. Ma première tâche fut de changer les termes pour deux boutons dans la page d’accueil de la partie utilisateur de l’interface dans la page d’accueil. En effet 2 boutons avaient un terme très similaire alors qu’ils faisaient des actions très différentes, ce qui rendait les utilisateurs confus. J’ai donc modifié ce terme pour 2 autres termes bien explicite de ce que ces 2 boutons faisaient ces 2 termes sont : « Demander au réseau une étude de faisabilité » et « Déposer un projet ». J’ai également changé les termes d’autres boutons pour correspondre avec ceux changés précédemment.

**Changement termes boutons faisabilité et demande de projet (Avant) :**



**Changement termes boutons faisabilité et demande de projet (Après) :**



J’ai ensuite dû rajouter une nouvelle fonctionnalité qui est la possibilité de dupliquer un projet existant. En effet l’application étant une application de gestion de projet de nombreux projets sont créés sur cette application. Un projet est composé d’un nombre conséquent d’informations comme un numéro de projet, un titre, un acronyme, un contexte, une description, un type, une thématique, une ou plusieurs ressource(s), un ou plusieurs financement(s), la centrale qui effectuera le projet, des partenaires, une équipe, des étapes, des faits marquants. Lors de la création d’un projet ou de la demande de faisabilité d’un projet les éléments obligatoires sont indiqués avec une astérix, on peut également sauvegarder la demande de projet en tant que brouillon si pour une raison ou une autre la demande de projet ne peut pas être finalisée sur le moment.

Tous ces éléments impliquent que parfois des projets se ressemblent en tous points (mis à part le numéro du projet, les membres et partenaires affectés au projet, les dates de début et de fin du projet ainsi que les étapes et les faits marquants). Donc pour simplifier le travail des utilisateurs de l’application pour qu’ils n’aient pas à saisir de nouveau toutes les informations dans le formulaire de création de projet, ce qui peut être long et fastidieux. J’ai dû créer une fonction qui duplique toutes les informations d’un projet mis à part son numéro de projet (qui est un numéro unique), les dates de début et de fin, le statut du projet ainsi que les étapes et les faits marquants qui elles seront toujours différentes. La duplication de projet peut être effectuée à la fois dans l’interface utilisateur mais aussi dans l’interface administrateur.

L’utilisateur a le choix de repositionner les mêmes membres et partenaires du projet d’origine sur le projet dupliqué et il a également le choix de saisir une date de fin pour le projet. Cependant s’il choisit de ne pas saisir de date de fin pour le projet lors de la duplication, le projet se termine automatiquement au bout d’un an car un projet doit obligatoirement avoir une date de fin. Une des plus grandes difficultés de cette tâche fût de récupérer toutes les informations du projet que l’on veut dupliquer car certaines informations ne sont pas dans la même table que celle du projet en lui-même notamment les informations sur les partenaires, les financements, etc... Une autre difficulté était le numéro de projet. Le numéro de projet doit être unique et affecté à un seul et unique projet, il a donc fallu que je récupère en base de donnée le dernier numéro de projet enregistré et que je l’instancie de 1 tout en vérifiant que le numéro du projet dupliqué n’existe pas déjà en base de donnée. Si c’est le cas j’empêche l’enregistrement du projet dupliqué en base de donnée et je retourne une erreur. Lors du développement cette tâche j’ai eu différentes erreurs et pour y remédier j’ai utilisé la documentation officielle de Python-Flask ainsi que des forums anglophones de personnes ayant eu des problèmes similaires notamment sur le forum StackOverflow.

## Ajout de tests unitaires

J’ai également développé les tests unitaires de l’application afin de valider l’intégrité des fonctionnalités du projet à la demande de l’équipe de développement. Afin de simplifier le travail des développeurs et pouvoir faire l’intégration continue de l’application. Pour effectuer les tests unitaires j’ai utilisé Pytest, un framework Python qui permet d’écrire et d’exécuter des tests unitaires dans une application Python.

Des tests unitaires pour tester l’authentification à l’application étaient déjà présents dans l’application, ils ont été créés par mon tuteur de stage. Avant d’effectuer ma tâche j’ai donc passé environ un jour et demi à étudier les tests déjà faits pour les comprendre et j’ai également étudié la documentation officielle de Pytest. Après cela j’ai donc créé des fonctions de tests qui appellent les différentes fonctions du projet à tester et j’ai dû saisir le résultat attendu pour qu’à la fin de la batterie de tests on compare le résultat attendu avec celui obtenu.

**Exemple d’une fonction de test unitaire :**

def \_access\_project\_page(client):

\_connect(client)

project\_request\_page = client.get("/project\_request/new\_project", follow\_redirects=True)

check\_project\_D = client.get("/check\_project/D", follow\_redirects=True)

check\_project\_B = client.get("/check\_project/B", follow\_redirects=True)

assert b'Demande de Projet' in project\_request\_page.data

assert b'Projets auxquels je suis affili' in check\_project\_D.data

assert b'finaliser' in check\_project\_B.data

Dans ce cas précis on teste l’accès à certaines pages de l’application, on appelle tout d’abord une fonction « \_connect » qui possède les informations de connexion d’un utilisateur. Puis on appelle plusieurs routes d’accès de différentes pages ici la route pour la création d’un projet ainsi que les pages qui comportent la liste des projets soit des projets déjà créés ("/check\_project/D") soit des brouillons ("/check\_project/B").

Ensuite le **assert** sert à indiquer le résultat attendu. Ici on indique qu’on attend le titre des différentes pages.

Pour les tests unitaires on peut soit tester les fonctions d’un seul fichier de test soit tous les fichiers de tests qui sont situés dans un dossier. S’il y a une erreur lors des tests on a un message qui nous indique le nombre de tests qui sont passés et le nombre de tests qui ont échoués avec les messages d’erreurs de ceux-ci pour que l’on sache où le problème se situe.

Pour exécuter les tests on utilise la ligne de commande avec la commande « pytest ***nom\_de\_fichier\_test*** ou ***nom\_du\_dossier de test*** », on peut également utiliser des « flags » ce qui peut aider. Par exemple il y a le flag ***-s*** qui permet d’afficher les prints du test dans le terminal, ***-v*** qui fournit les détails à chaque fichier de test ou encore ***--disable-warnings*** qui n’affiche pas les alertes des tests.



J’ai écrit les tests unitaires pour qu’ils couvrent la majeure partie des fonctions de l’application. Mon tuteur de stage a ensuite pu utiliser mon travail afin de créer des pipelines sur Gitlab afin d’automatiser ces tests unitaires pour qu’à chaque mise en développement et chaque mise en production les tests s’exécutent automatiquement. Ce qui empêche les régressions qui pourraient survenir en cas d’oubli d’exécution des tests par les développeurs.

## Système de logs

Je me suis aussi attelé à la création d’un système de logs sur différentes actions effectuées par un utilisateur dans l’application. Jusqu’à maintenant seuls deux personnes de l’équipe de développement avaient accès aux logs système de l’application ce qui posait problème quand ces deux personnes étaient absentes. Après avoir consulté mon tuteur de stage nous avons convenus que la meilleure solution était de faire un fichier de logs dans le projet pour avoir un historique local mais aussi d’avoir une table dans la base de données pour un historique « long » mais aussi pour que l’équipe entière puisse avoir accès à ces logs. Après quelques recherches pour faire les logs de l´application j´ai choisi d’utiliser le paquet Flask-Logging qui nous permet de faire des logs personnalisés pour l´application. J´ai également choisi Flask-logging car en étudiant la documentation officielle j’ai trouvé qu’il peut aussi nous permettre de pouvoir automatiquement envoyer un mail à une adresse mail de notre choix quand un log d´erreur est renvoyé par l´application après quelques configurations, ce qui permet une plus grande réactivité en cas d’erreur dans celle-ci car l´utilisateur n´a pas besoin de contacter l´équipe de développement en cas de problème bloquant, l´application le fait elle-même. Grâce au travail que j’ai effectué sur les logs mon tuteur de stage pourra les implémenter dans Grafana-Loki qui est une interface graphique de monitoring qui permet de consulter les logs de l’application, les gérer et des effectuer statistiques de celles-ci. C’est un complément de Grafana-Prometheus qui permet de faire le monitoring de l’utilisation des ressources de l’application, ce qui permet d’avoir un vue d’ensemble de l’application et de pouvoir prévoir et localiser les différents problèmes pouvant survenir dans l’application.

## Relance de projet par mail

J’ai aussi dû m’occuper de la relance de projet par mail. C’est une fonctionnalité demandée par les administrateurs qui leur permet d’envoyer des mails à propos d’un projet bien particulier car ils ont parfois besoin de demander des choses au demandeur et aux project leaders par rapport à certains projets effectués ou en cours. Comme c’est une fonctionnalité qui ne sert qu’aux administrateurs, le bouton de « relance de projet » n’est présent que dans l’interface administrateur. Pour l’envoi de mail j’ai utilisé Flask-Mail qui est un paquet de Flask installable sur le projet qui été déjà présent avant mon arrivé qui nous permet de facilement envoyer des mails, j’ai donc étudié la documentation officielle de Flask-Mail afin de pouvoir par la suite faire la fonction d’envoi de mail.



Quand on clique sur le bouton de relance une modal apparaît et permet à l’administrateur de saisir un message qui sera dans le corps du mail ainsi que le sujet du message qui sera envoyé au demandeur ainsi qu’aux project leaders affectés au projet.

**Modal pour la saisie du sujet et du message du mail :**

**Modal pour la saisie du sujet et du message du mail (remplie) :**

Après avoir validé l’envoi du mail, la fonction recherche les e-mails des project leaders et du demandeur pour les mettre en destinataires et pour faciliter la discussion par mail on récupère l’email de l’expéditeur et on l’ajoute dans le champ « reply-to » de la fonction car le mail est envoyé depuis une adresse mail « no-reply ». De cette manière quand un mail de relance aura été envoyé, les destinataires pourront répondre à l’expéditeur même s’ils ne connaissaient pas l’e-mail de celui-ci.

Quand le mail a bien été envoyé on a un message qui apparaît dans la page du projet côté administrateur qui nous confirme le bon envoi de celui-ci.

**Confirmation de l’envoi du mail pour la relance de projet :**



Et voici le mail que les project leaders et le demandeur recevront :

**Mail reçu par le demandeur et les project leaders du projet :**



Ici on voit bien le nom et prénom de la personne qui reçoit le mail mais aussi le numéro du projet concerné comme un utilisateur peut être positionné sur plusieurs projets, il faut pouvoir savoir de quel projet on parle. On a également le message précédemment saisi dans la modal et un bouton qui permet de directement aller sur la page côté utilisateur du projet concerné. Pour faire ressortir ce message envoyé par l’expéditeur du reste du mail j’ai ajouté un fond à celui-ci et j’ai également mis le message en italique. La difficulté de cette tâche était de réussir à ajouter les adresses mail des membres du projet en tant que destinataire. Pour cela je parcours dans une boucle la réponse JSON des utilisateurs du projet en question et j’ajoute les emails dans un tableau. Puis j’envoie ce tableau à la fonction qui envoi le message et j’indique que ce tableau est le « recipients », les personnes à qui le serveur doit envoyer les mails.

## Traduction de l’application

Je me suis également attelé à la traduction de l’application, l’application étant utilisé à l’international par différents collaborateurs du réseau RENATECH et de l’IEMN, il fallait que l’application soit traduite au moins en anglais. Une traduction existait déjà auparavant, cependant la quasi-totalité des textes de l’application ont changés. Ce qui a engrangé la disparition des traductions auparavant présentes dans l’application.

Pour faire la traduction après avoir fait quelques recherches j’ai décidé d’utiliser le paquet Flask-Babel car la traduction se fait dans un fichier binaire ce qui est beaucoup plus rapide pour accéder aux traductions par rapport à une traduction en base de données, une traduction en base de données prends également plus de place qu’un fichier binaire. J’ai également eu l’occasion d’utiliser Flask-Babel lors du tutoriel pour m’habituer à Python-Flask. Grâce à Flask-Babel on peut « tagguer » les termes que l’on veut traduire et si on retrouve les mêmes termes à différents endroits de l’application (que ce soit dans le même fichier ou non) ces termes sont regroupés dans le fichier binaire et au-dessus du terme à traduire on a en commentaire le nom du fichier et le numéro de la ligne où le terme à traduire se situe ce qui simplifie grandement la traduction.

## Responsive de certaines parties de l’application

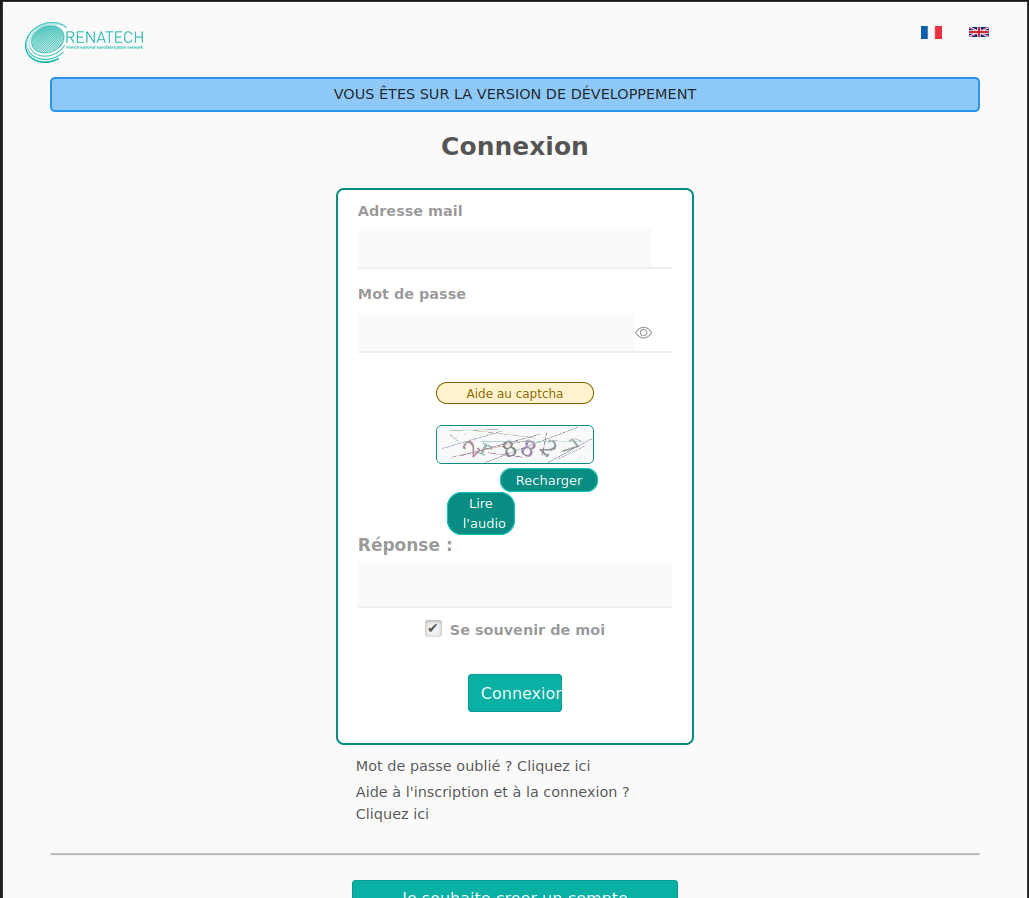
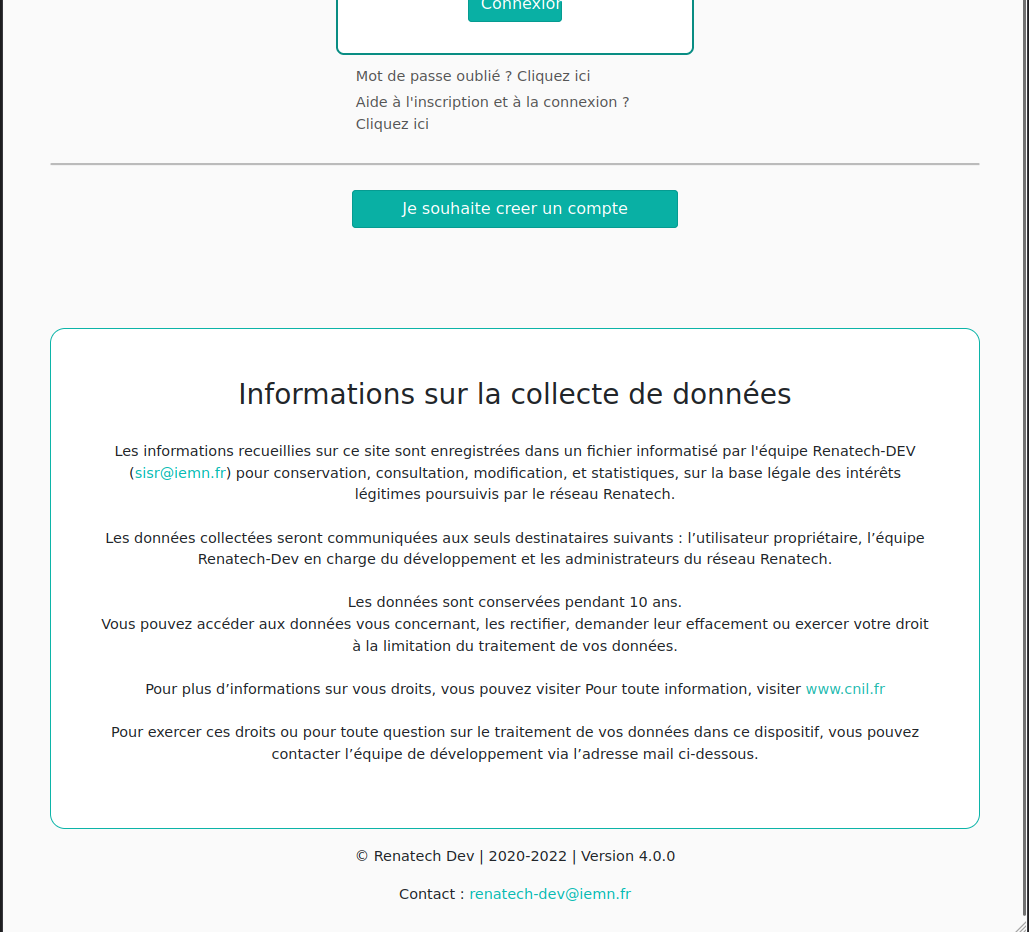
J’ai également dû m’occuper du responsive de l’application dans l’interface utilisateur car un retour des différents utilisateurs était que l’application était difficilement utilisable sur mobile, tablette mais également sur certaines résolutions d’écrans d’ordinateurs. J’ai donc utilisé Bootstrap 4 et retravaillé le CSS afin de faire en sorte que l’application supporte une plus grande variété de résolutions d’écrans allant du mobile au desktop en passant par les tablettes et les laptops. Mon travail s’est porté sur les 4 pages les moins responsives et donc les plus problématiques :

* La page de Connexion
* La page d’Accueil (index)
* La page de Dépôt de projet
* La page de Détails d’un projet

Pour l’exemple voici l’évolution de la page de connexion ainsi que la page d’accueil avant et après le passage en responsive :

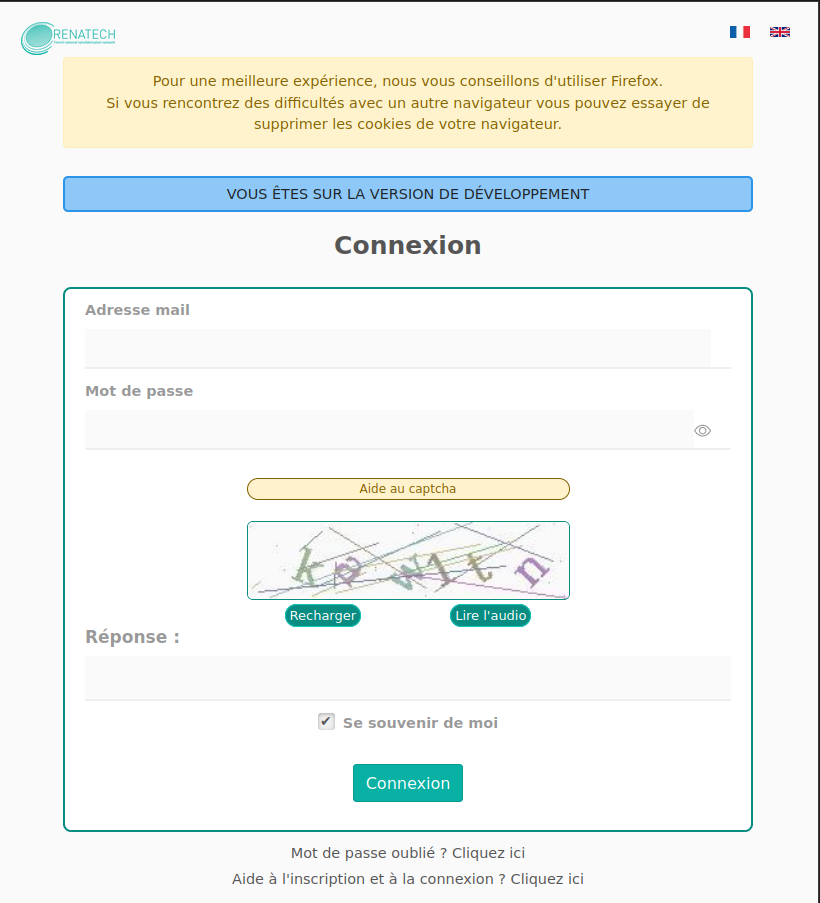
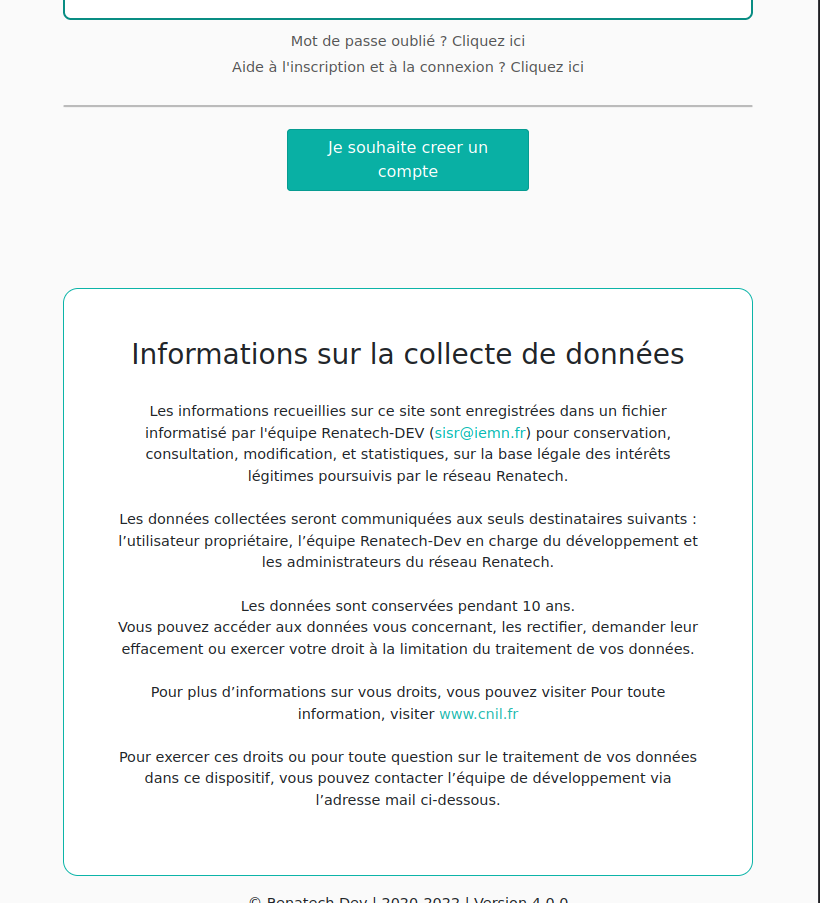
Pour la page de connexion la version ordinateur ne nécessitait pas de modification par rapport aux versions tablette et mobile qui posaient vraiment problème.

**Page de connexion version tablette (avant responsive) :**



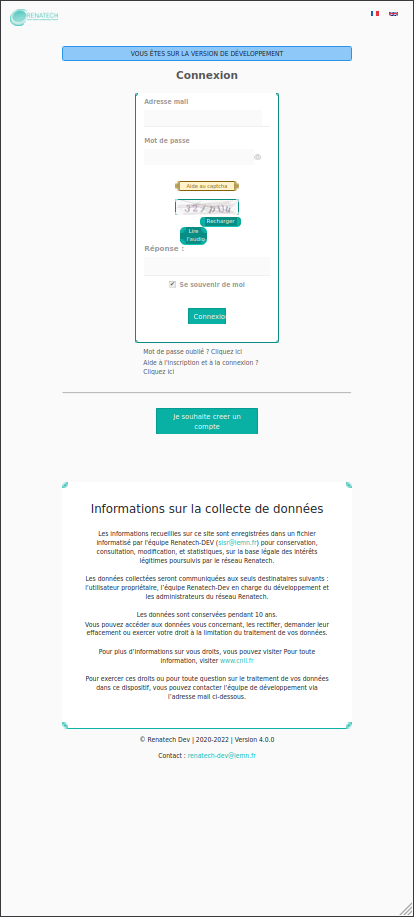
Ici on voit que les boutons sont mal placés et que le texte dépasse de certains d’entre eux. Et que la partie connexion en elle-même est petite par rapport à l’espace qu’elle pourrait occuper. Ce qui peut engranger des difficultés lors de la saisie des identifiants pour se connecter sur ce type de résolution.

**Page de connexion version tablette (après responsive) :**



Quand je me suis occupé du responsive pour la version tablette j’ai fait en sorte que les encadrés fassent la même taille, ce qui rend l’expérience plus plaisante visuellement. Grâce à l’aggrandissment de l’encadré de la partie de la connexion, les boutons pour le captcha ne se chevauchent plus et sont bien espacés l’un de l’autre. Les inputs des identifiants de connexion et de la réponse du captcha sont aussi plus grand ce qui apporte un meilleur confort pour les écrans de tablettes.

**Page de connexion version mobile (avant responsive) :**

Comme on peut le voir ici la page de connexion en version mobile posait vraiment problème on ne pouvait pas faire grand-chose comme les éléments étaient vraiment petits, en plus d’avoir les mêmes problèmes que la version tablette de la page.

# Conclusion

Ce stage m’a beaucoup apporté j’ai pu apprendre une nouvelle technologie que j’apprécie beaucoup et je suis également très heureux d’avoir pu effectuer mon stage dans un lieu comme l’IEMN où j’ai pu rencontrer des personnes très talentueuses et très attentionnées.

Ils m’ont chacun apportés quelque chose et j’espère avoir pu faire de même. J’ai beaucoup appris à leurs côtés.

J’ai aussi eu la chance d’avoir été positionné sur un projet aussi important et qui est utilisé quotidiennement par nombre de personnes pour un réseau aussi important que le réseau RENATECH.