

Rapport de PAE

**Concepteur Développeur d’Applications**

**IEMN**

Avenue Henri Poincaré

59650 VILLENEUVE D’ASCQ

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

*Période en entreprise :*

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

**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**

Table des matières

[1 Listing des compétences abordées lors du projet - 3 -](#_Toc137146101)

[2 Remerciements - 4 -](#_Toc137146102)

[3 IEMN - 5 -](#_Toc137146103)

[3.1 Présentation de l’entreprise - 5 -](#_Toc137146104)

[3.2 Présentation de l’équipe - 6 -](#_Toc137146105)

[4 Environnement de travail - 7 -](#_Toc137146106)

[4.1 Outils Utilisés - 7 -](#_Toc137146107)

[4.2 Technologies Utilisées - 9 -](#_Toc137146108)

[5 Présentation du projet - 12 -](#_Toc137146109)

[5.1 Présentation du projet en français - 12 -](#_Toc137146110)

[5.2 Présentation du projet en anglaise / Project presentation in English - 13 -](#_Toc137146111)

[5.3 Plus d’informations sur le projet Repotech et mon travail sur celui-ci - 14 -](#_Toc137146112)

[5.3.1 Quelques chiffres sur le projet Repotech - 14 -](#_Toc137146113)

[5.3.2 Les enjeux de mon travail - 14 -](#_Toc137146114)

[5.3.3 Cheminement de l’application du développement vers la mise en production - 14 -](#_Toc137146115)

[5.3.4 Schéma du fonctionnement de l’application - 15 -](#_Toc137146116)

[6 Tâches effectuées - 16 -](#_Toc137146117)

[6.1 Duplication de projets et changement de termes pour certains boutons - 16 -](#_Toc137146118)

[6.2 Ajout de tests unitaires - 22 -](#_Toc137146119)

[6.3 Système de logs - 25 -](#_Toc137146120)

[6.4 Relance de projet par mail - 28 -](#_Toc137146121)

[6.5 Traduction de l’application - 32 -](#_Toc137146122)

[6.6 Responsive de certaines parties de l’application - 34 -](#_Toc137146123)

[7 Conclusion - 41 -](#_Toc137146124)

[8 Sources - 42 -](#_Toc137146125)

# Listing des compétences abordées lors du projet

|  |  |  |
| --- | --- | --- |
| Concevoir et développer des composants d’interface utilisateur en intégrant les recommandations de sécurité | Maquetter une application | 🗶 |
| Développer une interface utilisateur de type desktop | ✓ |
| Développer des composants d’accès aux données | ✓ |
| Développer la partie front-end d’une interface utilisateur web | ✓ |
| Développer la partie back-end d’une interface utilisateur web | ✓ |
| Concevoir et développer la persistance des données en intégrant les recommandations de sécurité | Concevoir une base de données | 🗶 |
| Mettre en place une base de données | 🗶 |
| Développer des composants dans le langage d’une base de données | ✓ |
| Concevoir et développer une application multicouche répartie en intégrant les recommandations de sécurité | Collaborer à la gestion d’un projet informatique et à l’organisation de l’environnement de développement | ✓ |
| Concevoir une application | ✓ |
| Développer des composants métier | ✓ |
| Construire une application organisée en couches | ✓ |
| Préparer et exécuter les plans de tests d’une application | ✓ |
| Préparer et exécuter le déploiement d’une application | 🗶 |

# 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 notre chef de projet pour leurs aides et conseils lors de mon apprentissage des technologies utilisées dans l’entreprise ainsi que pour leurs disponibilités tout au long de la période d’application en entreprise.

Et je remercie pour finir Mr David LEGUICHER, Mr Sébastien JASNIAK, Mr Samuel TENON, Mr Abdoul BACHIR SEYDOU, 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) et 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 d’une trentaine de centres régionaux de plus petite taille. Chaque centrale possède sa spécialité et ses propres ressources.

Une image contenant texte, diagramme, carte

Description générée automatiquement

## Présentation de l’équipe

# Environnement de travail

## Outils Utilisés



Linux est un noyau de système d’exploitation qui permet de gérer l’exécution d’application. Linux est disponible en open-source. Linux a permis de donner naissance à des centaines de systèmes d’exploitation et de distributions Linux. Ces distributions peuvent être utilisées sur un ordinateur, un objet connecté ou encore un serveur. Lors de ma période d’application en entreprise j’ai utilisé la distribution Ubuntu. Linux est très largement utilisé dans le cloud et de grandes infrastructures du cloud reposent sur lui notamment Google et Amazon. Linux est également un socle d’une des dernières innovations du cloud : Docker qui repose sur le Linux Container du système open-source.

Gitlab nous sert pour le versioning du projet. Gitlab possède des outils tels qu’une liste et un tableau pour pouvoir créer et suivre l’avancement 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.

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.



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 textuels 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.

## Technologies Utilisées



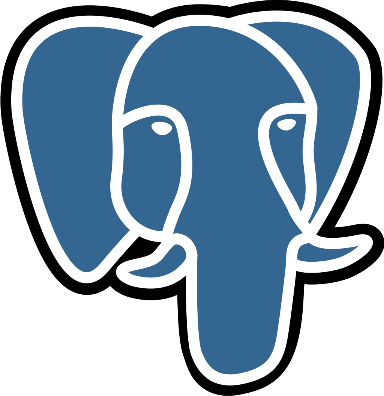


Docker est une plateforme de conteneurisation qui permet de créer des conteneurs pour nos applications. C’est une alternative aux machines virtuelles. Par rapport aux machines virtuelles, un conteneur est plus léger, plus simple et partage son kernel et son système d’exploitation avec d’autres conteneurs, 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.

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. Cependant on peut également installer les paquets de Flask avec une interface graphique en utilisant l’IDE PyCharm. On peut installer tous les paquets d’un projet existant en une seule ligne de commande s’ils sont listés dans un fichier de « requirements ».

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.





Adminer est une application web qui offre une interface graphique pour différents systèmes de gestion de base de données (par exemple : MySQL, SQLite, PostgreSQL, etc…). Cette application est réalisée sous PHP et a pour particularité d’être entièrement contenu dans un seul fichier et se présente comme une alternative légère par rapport à PHPMyAdmin. Mais si on le souhaite, on peut ajouter un fichier CSS pour en modifier la présentation. Adminer était autrefois connu sous le nom de PHPMinAdmin.

Pour la base de données, l’entreprise a décidé d’utiliser PostgreSQL qui est un SGBDRO (Système de gestion de base de données relationnelle et objet) open-source développé en 1986. L’un des avantages de PostgreSQL est qu’il y a une meilleure prise en charge des données volumineuses. Mais aussi car l’entreprise utilise PostgREST qui permet de transformer la BDD en API REST pour l’application.

Traefik est un reverse-proxy français développé en langage Go, un concurrent de C et Pascal. Un reverse-proxy permet de faire l’intermédiaire entre un réseau publique et un réseau privé. Par exemple sur le réseau privé on trouvera les applications qui ne sont pas accessibles depuis l’extérieur pour des raisons de sécurité. Le reverse-proxy doit donc connaitre les différentes routes de l’application pour pouvoir transférer les requêtes effectuées vers le bon service.



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

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

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.

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 directement déposer un projet. Dans ce projet on pourra y affecter des utilisateurs et des partenaires, créer des étapes et créer des faits marquants. Lors de la saisie d’une faisabilité ou d’un projet, les utilisateurs peuvent également l’enregistrer en tant que brouillon pour le continuer de le remplir et valider la saisie plus tard pour la création de la faisabilité ou du projet.

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 et les supprimer. Ils peuvent aussi créer des comptes utilisateurs, renvoyer des mails d’activation de compte au besoin, mais aussi consulter les informations d’un utilisateur, modifier ses informations ou encore supprimer un compte utilisateur. 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.

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

I was positioned on the Repotech project. This is a project management application. This application is used to deposit research projects in different RENATECH network plants. The application is composed of 2 different user interfaces: the administration 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 what plant will do your project you can directly do a project deposit. In this project you will be able to affect users and partners to it, create steps and create highlights. While filling a feasibility request or creating a project the users can save it as a draft to continue its creation and validate the entry later.

In the administration 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 and delete them. They can also create user accounts, resend activation e-mails if needed, but also consult users informations, modify them or delete a user account. They can also migrate establishments and users in case of duplicates. The application is developed by the IEMN a research institute of the CNRS for RENATECH and the app is developed in Python-Flask and Jinja to make templates in the front-end.

## Plus d’informations sur le projet Repotech et mon travail sur celui-ci

### Quelques chiffres sur le projet Repotech :

En quelques chiffres le projet Repotech c’est :

Une image contenant texte, capture d’écran, Police, logo

Description générée automatiquement

### Les enjeux de mon travail :

Mon but a été d’ajouter des fonctionnalités demandées par les utilisateurs ou les administrateurs 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 mais aussi de pouvoir aider à la finalisation de la mise en place de l’intégration et du déploiement continu grâce à des tests unitaires dans l’application et de permettre une surveillance accrue des services mis en production grâce à un système de logs afin d’augmenter le taux de productivité et de réactivité de l’équipe en cas de soucis dans l’application.

### Cheminement de l’application du développement vers la mise en production :

* Chaque développeur dispose de sa propre branche de développement Gitlab où il développe les tâches qui lui sont attribuées.
* Ensuite quand il finit sa tâche, il la publie sur une branche commune à tous les développeurs.
* Quand un certain nombre de tâches sont effectuées et des tests en local réalisés on passe sur une branche « dev » et sur la mise en ligne sur un site similaire en tous points au site qui est en production qui fait office de bac à sable afin de laisser les administrateurs de tester les nouvelles fonctionnalités et d’avoir un retour de leur part en cas de bugs ou d’incompréhensions sur les fonctionnalités rajoutées.
* Si le retour des administrateurs est positif alors on effectue une mise en production sur la branche principale.

### Schéma du fonctionnement de l’application :

Une image contenant capture d’écran, texte, diagramme, cercle

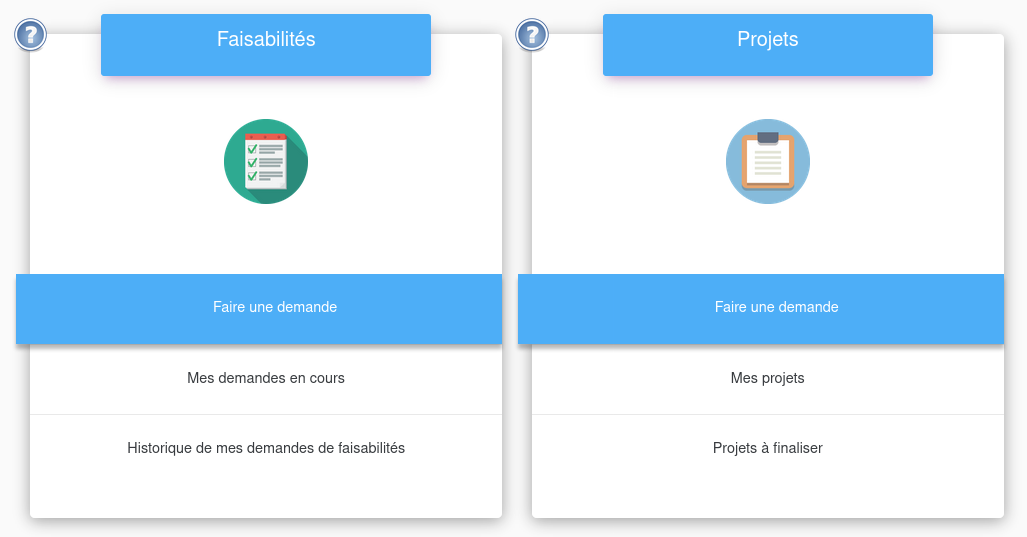
Description générée automatiquement

# Tâches effectuées

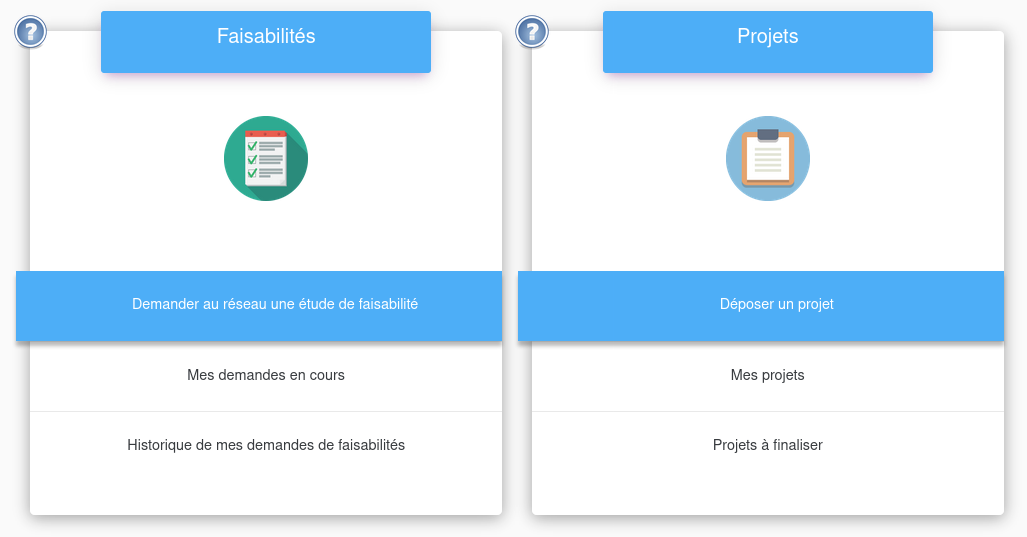
## Duplication de projets et changement de termes pour certains boutons

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 officielle de celui-ci mais aussi grâce 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, afin de commencer doucement et de m’imprégner du code de l’application. 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) :**

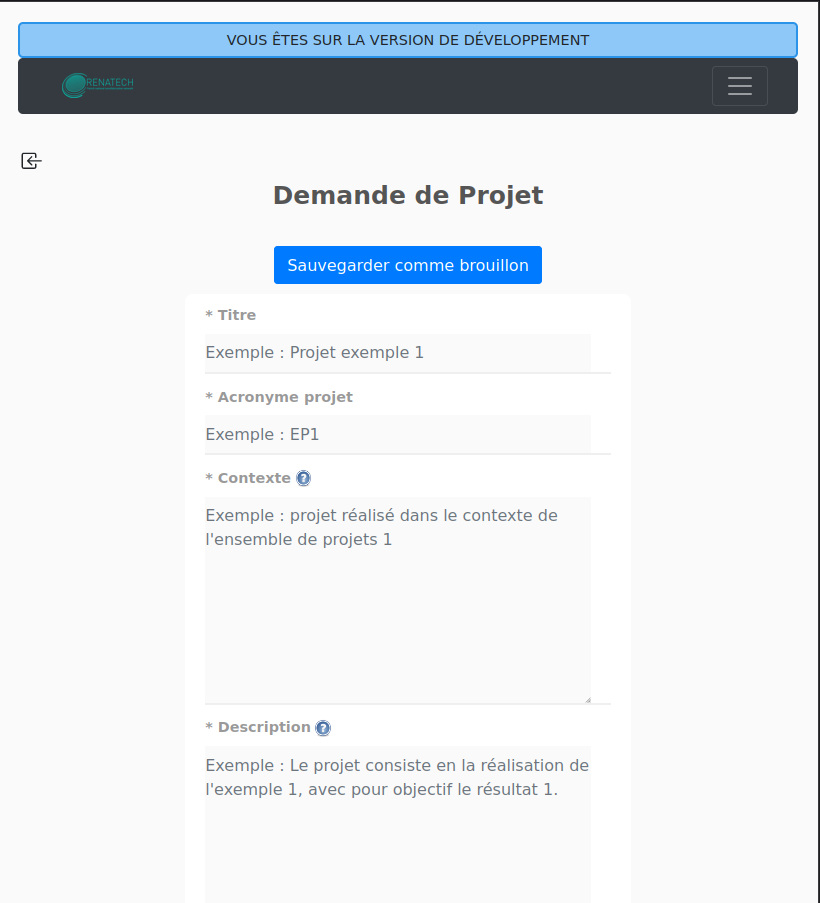


**Changement termes boutons faisabilité et demande de projet (Après) :**



Par la suite, j’ai 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 un astérisque, 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.

**Formulaire de demande de projet :**



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.

**Extrait de code de la fonction de la duplication :**

projet = Projets(

*numero\_projet*=new\_project\_number,

*titre\_projet*=data\_projet[0]["titre\_projet"],

*UUID*=uuid.uuid4(),

*contexte*=data\_projet[0]["contexte"],

*description\_projet*=data\_projet[0]["description\_projet"],

*acronyme\_projet*=data\_projet[0]["acronyme\_projet"],

*reference\_interne*="Non communiqué",

*confidentiel*=data\_projet[0]["confidentiel"],

*developpement\_technologique*=data\_projet[0]["developpement\_technologique"],

*id\_type\_projet*=data\_projet[0]["id\_type\_projet"],

*id\_thematique\_projet*=data\_projet[0]["id\_thematique\_projet"],

*id\_statut\_projet*=1,

*id\_etablissement*=data\_projet[0]["id\_etablissement"],

*date\_demande*=datetime.utcnow(),

*date\_debut\_projet*=None,

*date\_fin\_projet*=date\_fin,

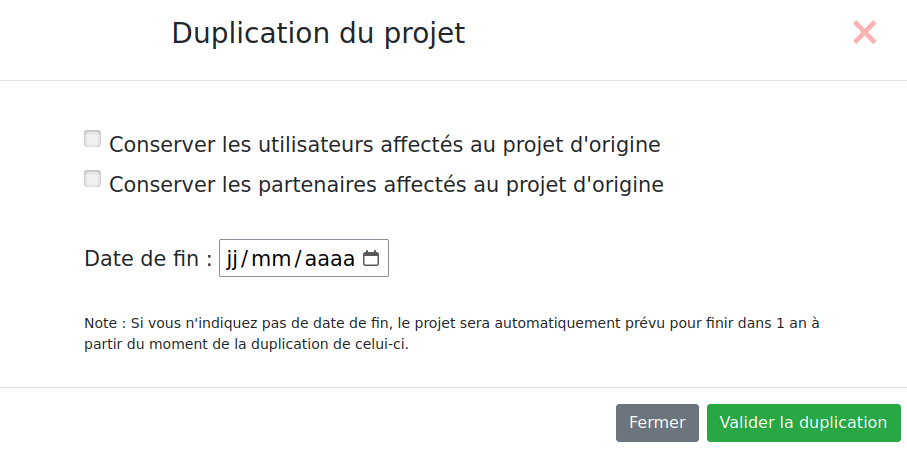
*coordinateur\_projet*=data\_projet[0]["coordinateur\_projet"],

*multiplateforme* = data\_projet[0]["multiplateforme"]

)

L’utilisateur possède la liberté de repositionner les mêmes membres et/ou partenaires du projet d’origine sur le projet dupliqué et il a également le choix de saisir une date de fin pour le projet grâce à une modale qui apparait quand l’utilisateur clique sur le bouton de « duplication de projet » où l’on retrouve 2 cases à cocher et un sélecteur de date. L’envoi de la décision de l’utilisateur pour les membres, partenaires et date de fin est envoyée à la fonction python par un appel AJAX.

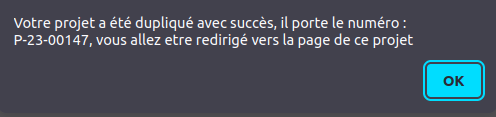
**Modale de duplication de projet :**



On doit également prévenir l’utilisateur que 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. On fait cela car un projet doit obligatoirement avoir une date de fin.

Si la duplication est réussie alors une alerte navigateur s’affiche avec une confirmation de la duplication et le numéro du projet dupliquer. Juste après l’utilisateur est redirigé vers le projet dupliqué.

**Duplication de projet réussie :**



A l’opposé en cas d’échec de la duplication un message d’erreur apparaît et rien d’autre ne se passe, l’utilisateur reste sur la page du projet d’origine et aucune donnée n’est envoyée en base de données.

Pour la duplication de projet du côté de l’interface administrateur je vérifie dès le début le rôle de l’utilisateur pour vérifier s’il possède soit le rôle d’administrateur local soit celui d’administrateur national à des fins de sécurité dans l’hypothèse où d’une manière ou une autre quelqu’un aurait réussi à accéder à l’interface administrateur alors qu’il n’en aurait pas les droits. Si l’utilisateur ne possède aucun rôle d’admin alors il est déconnecté et renvoyé sur la page de connexion.

**Extrait de code de la vérification du rôle admin :**

if current\_user.id\_statut\_utilisateur != 2 and current\_user.id\_statut\_utilisateur != 3:

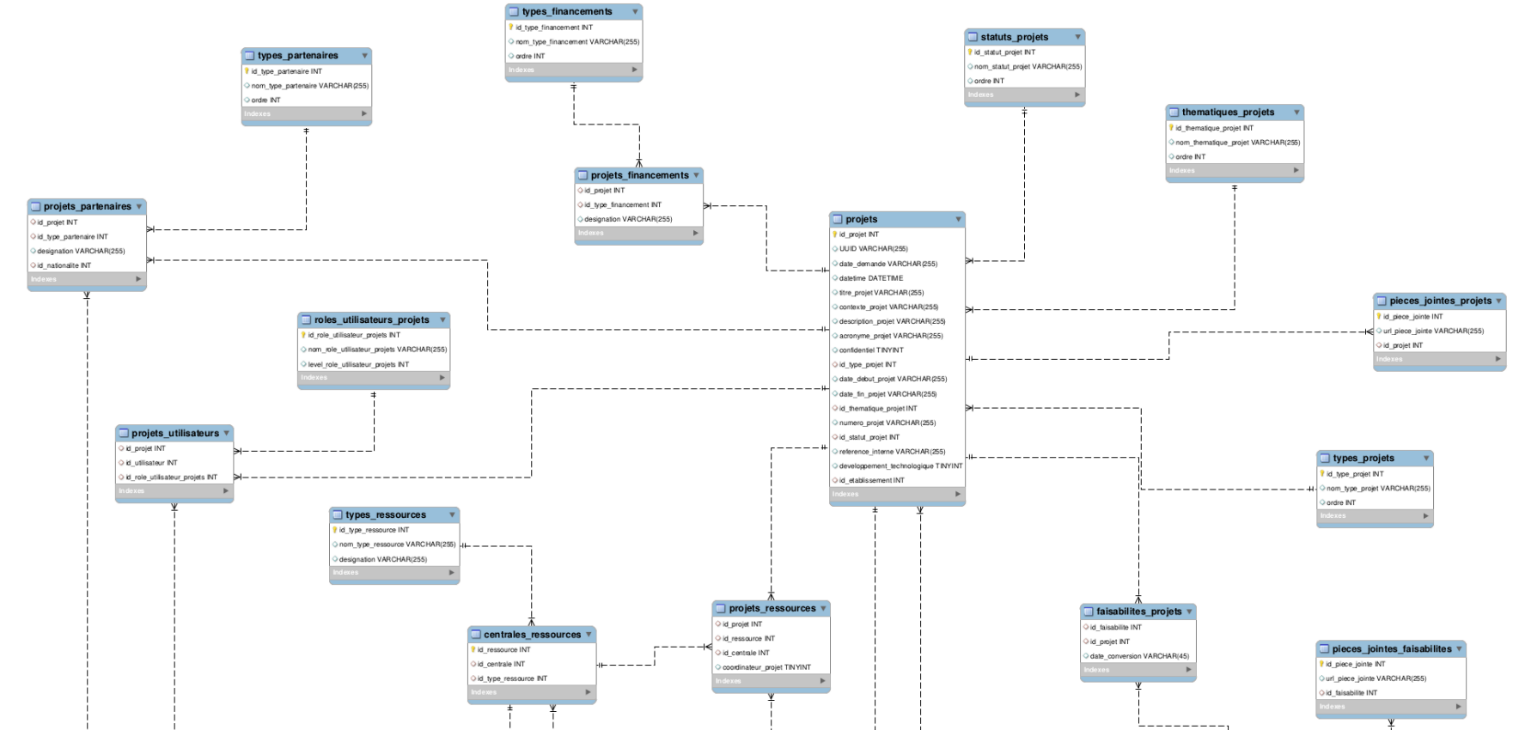
        current\_app.logger.error('Tentative de duplication de projet par un utilisateur qui n\'est pas un administrateur')

        functionApp.save\_log('Tentative de duplication de projet par un utilisateur qui n\'est pas un administrateur', 1, 9, *id\_current\_utilisateur*=current\_user.id\_utilisateur)

        return {'dupliquer': False, 'userRole': True, 'localAdmin': False}

Une des plus grandes difficultés de la duplication de projet 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... Afin de pouvoir garantir l’intégrité des données du projet dupliqué.

**Partie du schéma de la base de données :**



Comme on peut le voir la table projets est reliée à un nombre conséquent de tables dans la base de données. Ce qui rend la tâche ardue car il faut récupérer les données du projet d’origine de toutes ces tables et les réintroduire en les liants avec le projet dupliqué.

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 données 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 données. Si c’est le cas j’empêche l’enregistrement du projet dupliqué en base de données 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 dans une application Python.

Des tests pour 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 réalisés 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 comme on ne peut pas accéder aux différentes pages de l’application sans être connecté à celle-ci. 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 un terminal avec la commande « pytest ***nom\_de\_fichier\_de\_test*** ou ***nom\_du\_dossier de test*** », on peut également utiliser des « flags » ce qui peut aider pour le debugage. Par exemple il y a le flag ***-s*** qui permet d’afficher les print du test dans le terminal, ***-v*** qui fournit les détails à chaque fichier de test ou encore ***--disable-warnings*** qui permet de ne pas afficher les alertes des tests dans la console ou le terminal.

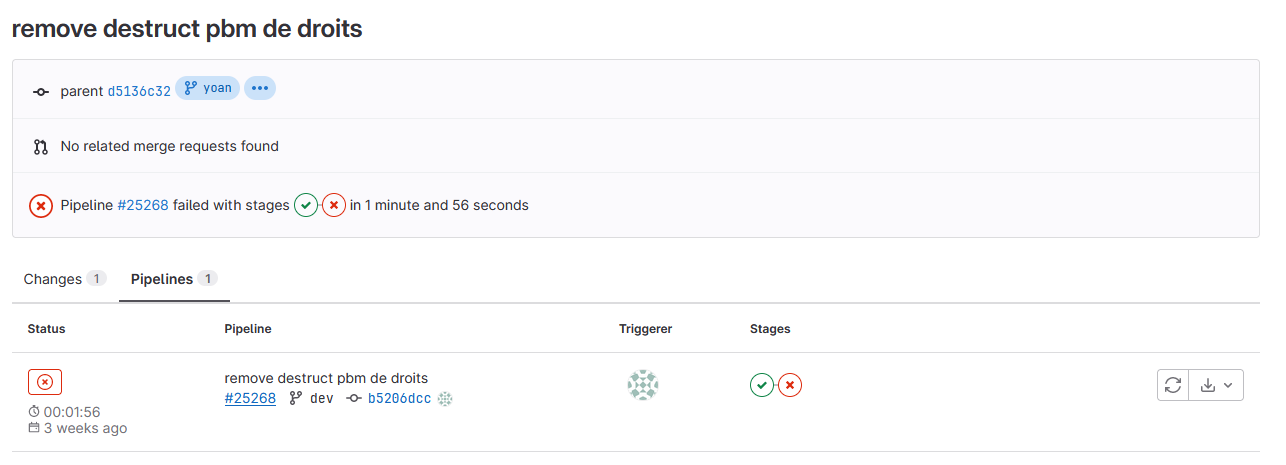
**Capture d’écran du terminal après exécution des tests unitaires en local :**



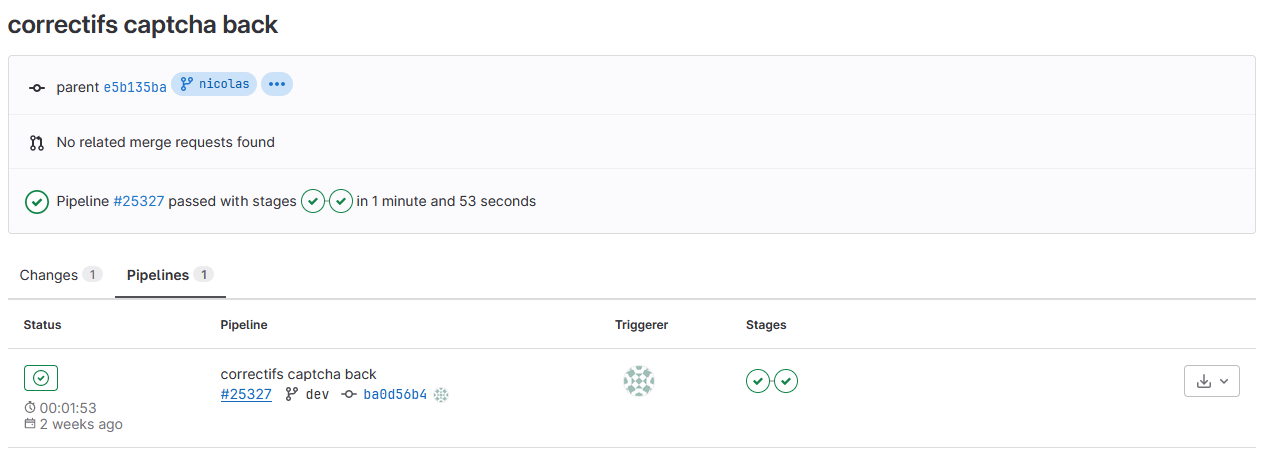
J’ai écrit les tests unitaires pour qu’ils couvrent la majeure partie des fonctions de l’application. Mon tuteur de stage a ensuite pu utiliser mon travail afin de créer des pipelines via Gitlab-CI/CD afin d’automatiser ces tests unitaires pour qu’à chaque mise en développement et chaque mise en production les tests s’exécutent automatiquement. Ce qui empêche les régressions qui pourraient survenir en cas d’oubli d’exécution des tests en local par les développeurs et permet de s’inscrire dans un contexte d’intégration et de déploiement continue de l’application.

La difficulté de cette tâche était de comprendre les différentes fonctions de l’application afin de saisir le résultat attendu correcte pour le bon fonctionnement des tests.

**Echec d’un pipeline sur Gitlab :**



**Réussite d’un pipeline sur Gitlab :**



On peut voir sur Gitlab les différents pipelines et si le pipeline échoue, on peut voir à quel moment l’échec à eu lieu afin de pouvoir effectuer les vérifications et modifications nécessaires.

## 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 ou indisponibles pour diverses raisons. 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´aura pas besoin de contacter l´équipe de développement en cas de problème bloquant, l´application le fera d’elle-même. Ce qui sera utile dans le futur. 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 des statistiques de celles-ci. Grafana-loki est similaire à Grafana Prometheus qui lui permet de faire le monitoring de l’utilisation des ressources de l’application (en matière de stockage, d’utilisation de processeur ou encore de mémoire), ce qui permet d’avoir rapidement une vue d’ensemble de l’application et de pouvoir prévoir et localiser les différents problèmes pouvant survenir dans l’application dans l’immédiat ou dans le futur. Ce qui sera très utile pour l’équipe de développement.

Voici un extrait des logs côté front :

[2023-05-24 11:38:20,487] - [INFO] front : Accès à la page de finalisation de la transformation de la faisabilité F-23-00004 en projet par ADMIN Admin

[2023-05-24 11:40:03,323] - [INFO] front : Transformation de la faisabilité F-23-00004 en projet par ADMIN Admin

[2023-05-24 11:57:49,554] - [INFO] front : Accès à la page de dépôt de projet par ADMIN Admin

[2023-05-24 11:58:52,949] - **[ERROR]** front : Une erreur est survenue lors de la soumission du formulaire du projet P-23-00067

[2023-05-24 13:06:57,273] - [INFO] front : Accès à la page de dépôt de projet par ADMIN Admin

[2023-05-24 13:12:26,275] - [INFO] front : Accès à la page de dépôt de projet par ADMIN Admin

[2023-05-24 13:13:23,822] - [INFO] front : Enregistrement d'un projet crée par ADMIN Admin

[2023-05-24 13:13:36,966] - [INFO] front : Accès à la page de dépôt de projet par ADMIN Admin

[2023-05-24 13:14:34,666] - [INFO] front : Une demande de projet vient d'être enregistrée comme brouillon par ADMIN Admin

[2023-05-24 14:41:28,527] - **[ERROR]** front : Une erreur est survenue lors de la duplication du projet numéro P-23-00068 effectuée par ADMIN Admin car le numéro de projet existe déjà

[2023-05-24 15:40:08,253] - [INFO] front : Accès à la page de finalisation du projet ayant l'id 72 par ADMIN Admin

[2023-05-24 15:40:25,822] - [INFO] front : Modification du brouillon du projet ayant l'id numéro 72 par ADMIN Admin

Et voici un extrait des logs côté back :

[2023-05-26 16:05:16,241] - [INFO] back : Suppression du partenaire Toto à la suite d'une migration par ADMIN Admin

[2023-05-26 16:08:04,857] - [INFO] back : Ajout d'une étape sur le projet P-23-00076 par ADMIN Admin

[2023-05-26 16:08:09,663] - [INFO] back : Modification d'une étape sur le projet P-23-00076 par ADMIN Admin

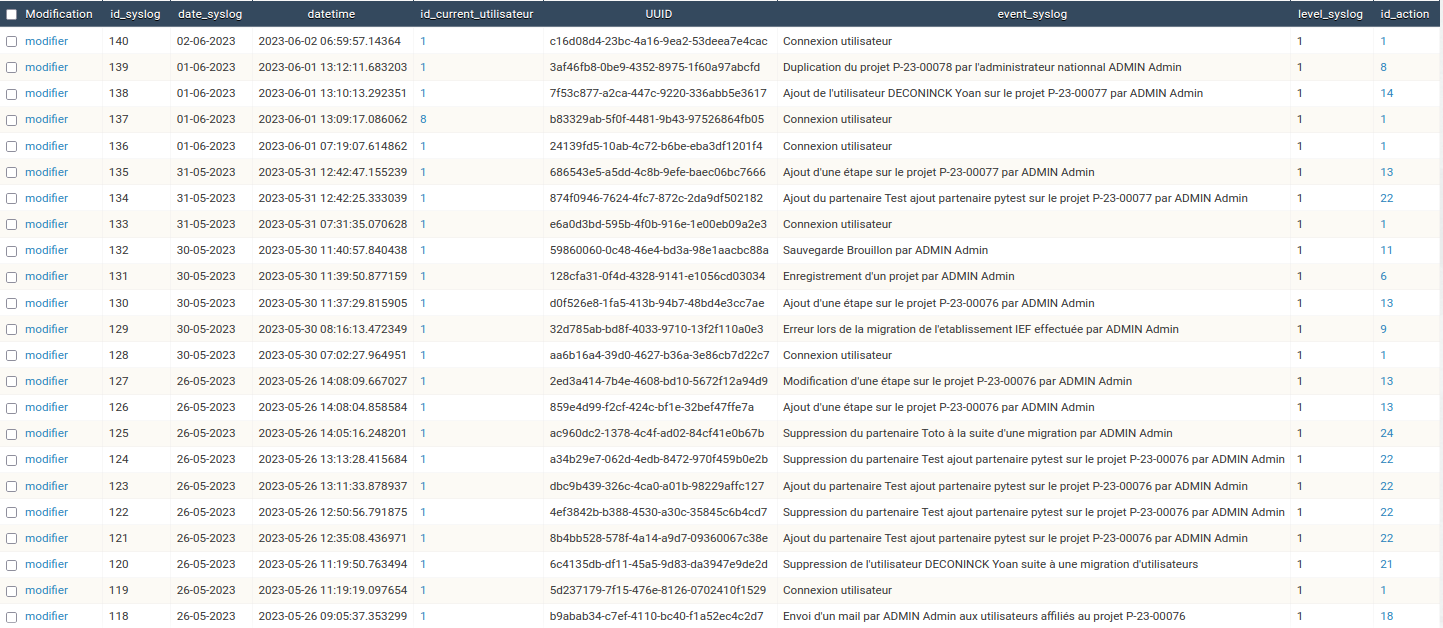
[2023-05-30 10:16:13,470] - **[ERROR]** back : Erreur lors de la migration de l'etablissement IEF effectuée par ADMIN Admin

[2023-05-30 13:37:29,814] - [INFO] back : Ajout d'une étape sur le projet P-23-00076 par ADMIN Admin

[2023-06-01 15:10:13,285] - [INFO] back : Ajout de l'utilisateur DECONINCK Yoan sur le projet P-23-00077 par ADMIN Admin

[2023-06-01 15:12:11,681] - [INFO] back : Duplication du projet : P-23-00078 effectuée par l'administrateur nationnal ADMIN Admin

Pour finir voici une capture d’écran des logs enregistrés en base de données :



J’ai également modifié la table actions de la base de données afin de pouvoir catégoriser les différentes actions effectuées par l’utilisateur dans l’application. On retrouve l’id de l’action dans la table des logs de la BDD.

**Extrait du script SQL de la table actions de la base de données :**

*-- Adminer 4.8.1 PostgreSQL 11.8 (Debian 11.8-1.pgdg90+1) dump*

DROP TABLE IF EXISTS "actions";

DROP SEQUENCE IF EXISTS actions\_id\_action\_seq;

CREATE SEQUENCE actions\_id\_action\_seq INCREMENT 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1;

CREATE TABLE "public"."actions" (

    "id\_action" integer DEFAULT nextval('actions\_id\_action\_seq') NOT NULL,

    "nom\_action" character varying(255),

    CONSTRAINT "actions\_pkey" PRIMARY KEY ("id\_action")

) WITH (oids = false);

CREATE INDEX "ix\_actions\_nom\_action" ON "public"."actions" USING btree ("nom\_action");

INSERT INTO "actions" ("id\_action", "nom\_action") VALUES

(1, 'Connexion'),

(2, 'Création de compte'),

(3, 'Création d''établissement'),

(4, 'Changement de mot de passe'),

(5, 'Message via le formulaire de contact'),

(8, 'Duplication de projet'),

(9, 'Erreur'),

(6, 'Dépôt de projet'),

(7, 'Modification de projet'),

(10, 'Suppression brouillon'),

(11, 'Dépôt brouillon'),

(12, 'Dépôt faisabilité'),

(13, 'Dépôt étape'),

(14, 'Ajout utilisateur projet'),

(15, 'Ajout fait marquant'),

(16, 'Suppression utilisateur projet'),

(17, 'Activation compte'),

(18, 'Mail relance projet'),

(19, 'Suppression utilisateur equipe'),

(20, 'Suppression utilisateur etablissement'),

(21, 'Suppression utilisateur'),

(22, 'Ajout partenaire projet'),

(23, 'Suppression partenaire projet'),

(24, 'Suppression partenaire'),

(25, 'Modification informations utilisateur');

## 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 les admins 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 était déjà présent avant mon arrivé et 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 modale 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.

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

**Modale 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 au lieu d’essayer de répondre à l’adresse no-reply.

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 demandeur ou un project leader peut être positionné sur plusieurs projets en même temps, il faut pouvoir savoir de quel projet on parle. On a également le message précédemment saisi dans la modale et un bouton qui permet de directement aller sur la page du projet concerné dans l’interface utilisateur. Pour faire ressortir le message envoyé par l’administrateur 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 » autrement dit, les personnes à qui le serveur doit envoyer les mails.

**Extrait du code pour envoyer les destinataires dans la fonction d’envoi de mail :**

tab\_users = []

    for user\_data in json\_utilisateur:

        utilisateurs = Utilisateurs()

        utilisateurs.nom = user\_data["nom"]

        utilisateurs.prenom = user\_data["prenom"]

        utilisateurs.email = user\_data["email"]

        utilisateurs.id\_role\_utilisateur\_projets = user\_data["id\_role\_utilisateur\_projets"]

        tab\_users.append(utilisateurs.email)

    try:

        send\_mail.send\_mail\_project\_remainder(projet, utilisateurs, expediteur, sujet, message, tab\_users)

        data = json.dumps(1)

        flash("Votre mail a bien été envoyé.")

        return data

    except OSError as err:

        flash(

            "Une erreur est survenue lors de l'envoi de votre mail.")

        data = json.dumps(2)

        return data

## 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 avaient 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, son emplacement et le numéro de la ligne où le terme à traduire se situe ce qui simplifie grandement la traduction. Il faut également prendre en compte qu’il faut un fichier de traduction par langue.

**Extrait de code des tags pour la traduction :**

<div *class*="bloc\_menu\_\_elem bloc\_menu\_\_elem-1">

{{ \_('Demander au réseau une étude de faisabilité') }}

</div>

<div *class*="bloc\_menu\_\_elem bloc\_menu\_\_elem-2">

{{ \_('Mes demandes en cours') }}

</div>

<div *class*="bloc\_menu\_\_elem bloc\_menu\_\_elem-3">

{{ \_('Historique de mes demandes de faisabilités') }}

</div>

Pour extraire les textes taggués dans un fichier *.pot* on utilise la commande « pybabel extract –F babel.cfg –k \_l –o messages.pot . ». Avec cette commande pybabel va regarder dans le fichier babel.cfg pour savoir à quels endroits il doit extraire le texte.

Ensuite on utilise la commande « pybabel init –i messages.pot –d app/translations –l en » qui va créer un fichier messages.po où l’on pourra traduire les termes de l’application qui ont été taggués.

**Extrait de code du fichier de traduction :**

#: templates/check\_single\_project.html:801

#: templates/check\_single\_project.html:807

msgid "Nom du partenaire"

msgstr "Name of the partner"

#: templates/create\_academic\_establishment.html:28

#: templates/create\_establishment.html:28

msgid "Création d'un établissement"

msgstr "Creation of an establishment"

Ici on a en commentaire l’emplacement du texte d’origine qui va être traduit, msgid (pour message id) correspond au texte d’origine de l’application et msgstr (pour message string) qui correspond au texte qui sera affiché quand l’utilisateur aura choisi une langue autre que celle d’origine de l’application. Cela signifie que pour chaque terme de l’application il faut traduire manuellement dans le fichier de traduction. On peut également utiliser des services tierces tels que le Google Cloud Translation API ou le Microsoft Translator Text API, cependant ces deux services sont payants.

Enfin on utilise la commande « pybabel compile –d app/translations » qui va compiler le fichier messages.po en messages.mo qui sera prêt à l’emploi par l’application pour la traduction de celle-ci.

## Responsive de certaines parties de l’application

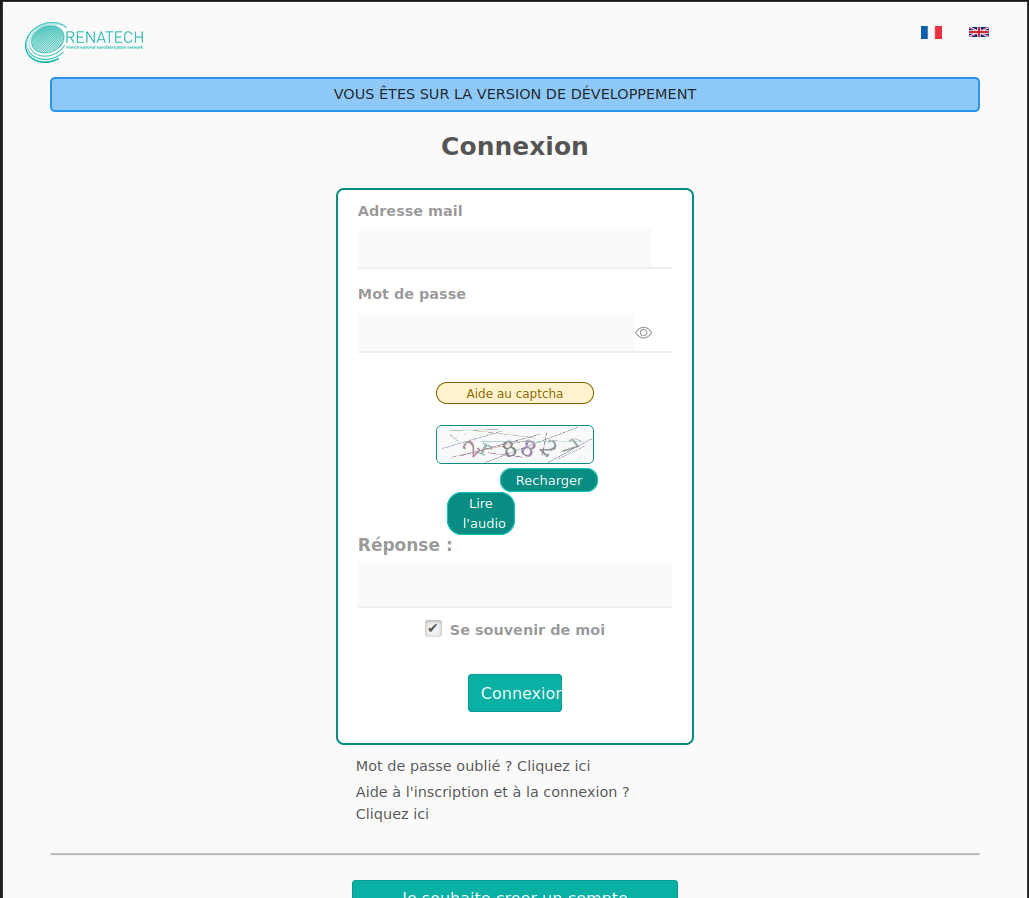
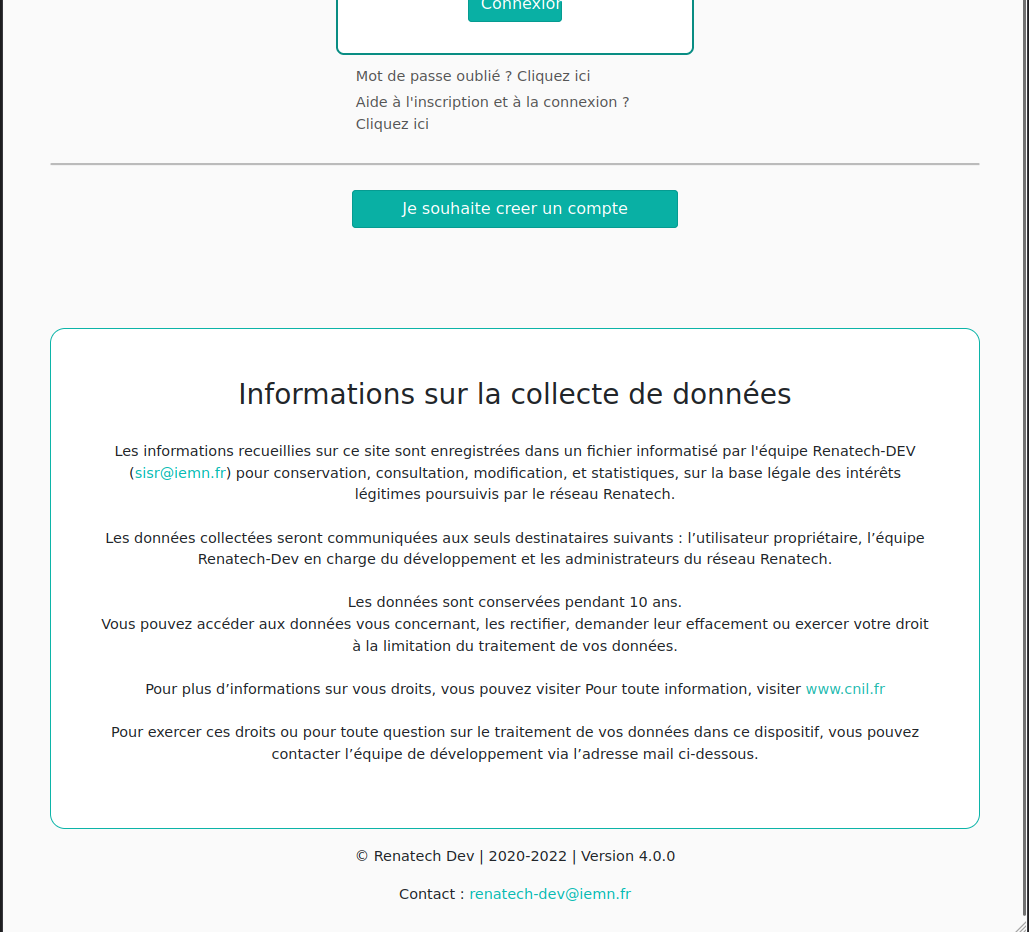
J’ai également dû m’occuper du responsive de l’application dans l’interface utilisateur car un retour des différents utilisateurs était que l’application était difficilement utilisable sur mobile, tablette mais également sur certaines résolutions d’écrans d’ordinateurs. J’ai donc utilisé Bootstrap 4 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 :

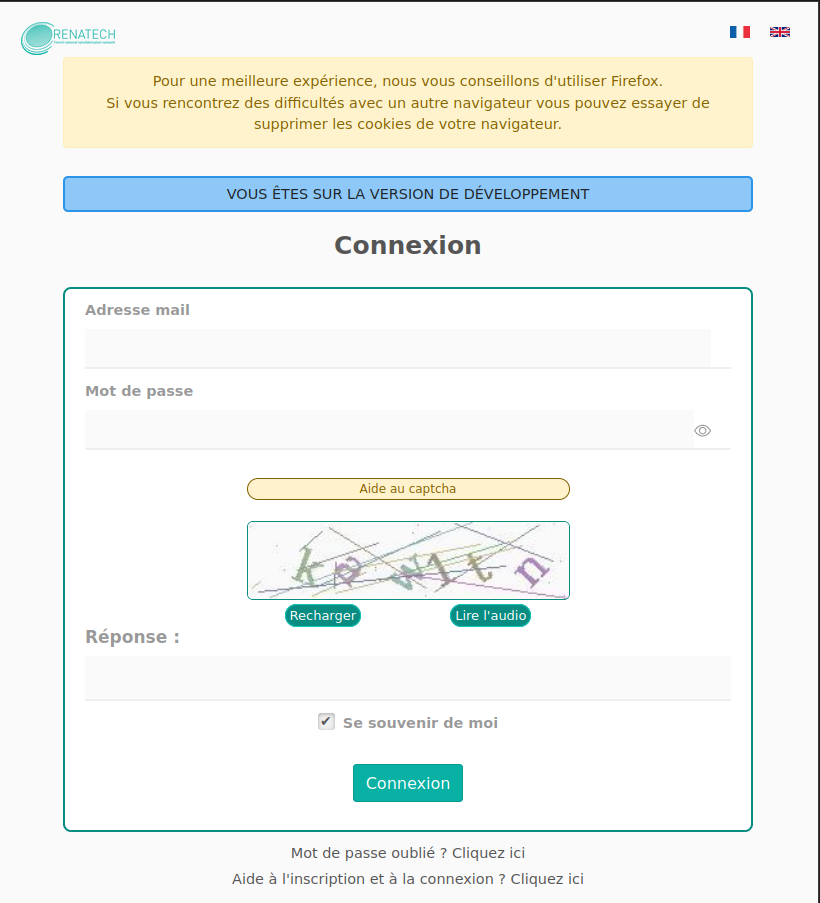
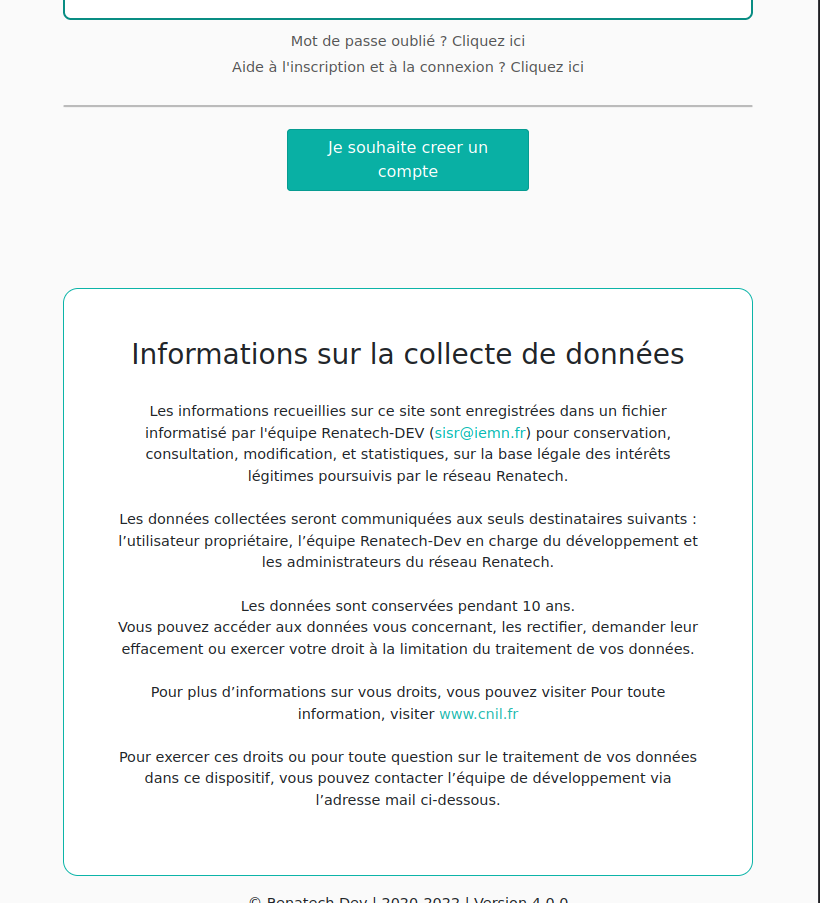
Pour la page de connexion la version pour ordinateur ne nécessitait pas de modification par rapport aux versions tablette et mobile qui posaient vraiment problème.

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



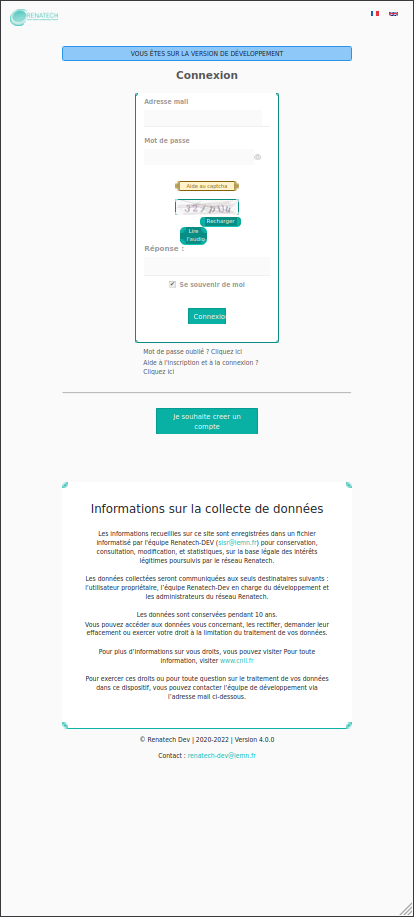
Ici on voit que les boutons sont mal placés et que le texte dépasse de certains d’entre eux. Et que la partie connexion en elle-même est petite par rapport à l’espace qu’elle pourrait occuper. Ce qui peut engranger des difficultés lors de la saisie des identifiants pour se connecter sur ce type de résolution.

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



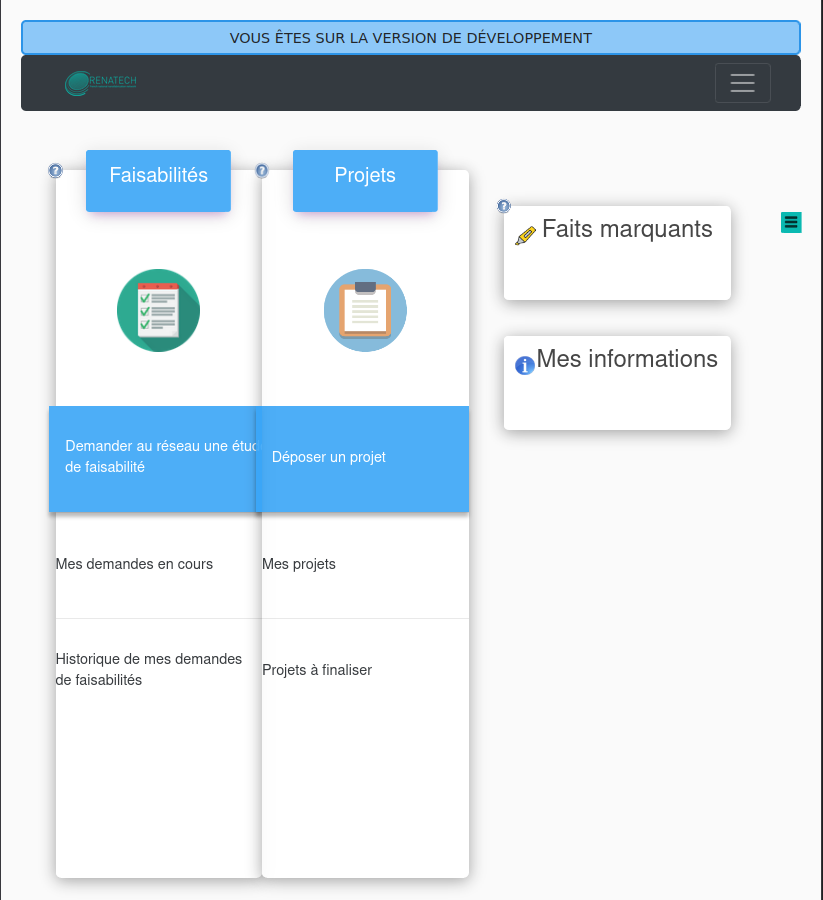
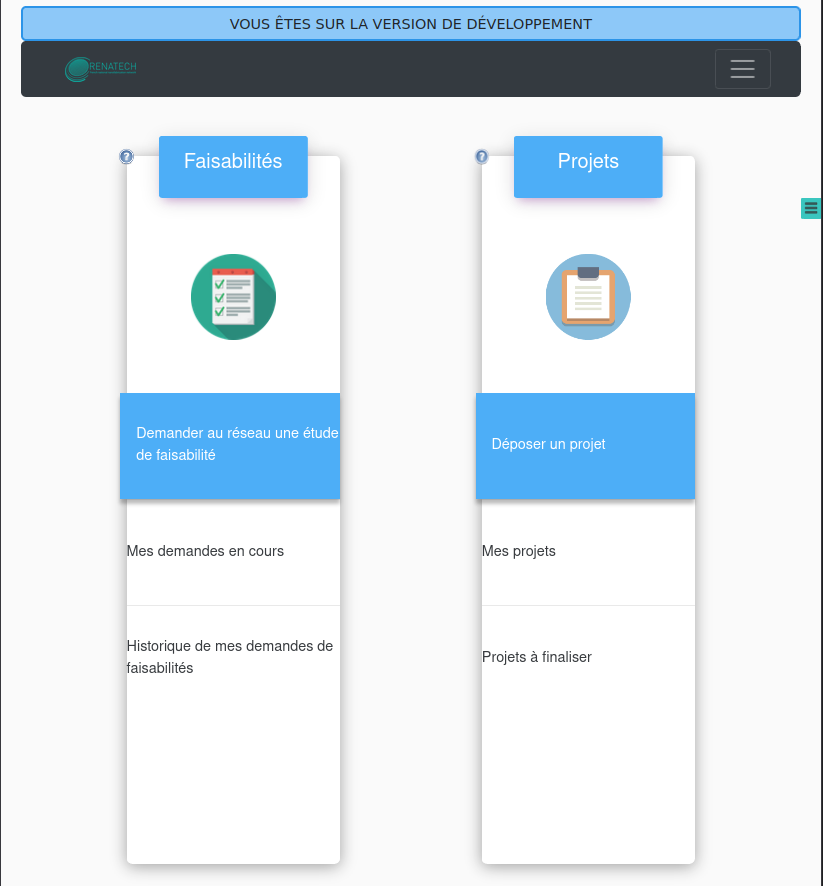
Quand je me suis occupé du responsive pour la version tablette j’ai fait en sorte que les encadrés fassent la même taille, ce qui rend l’expérience plus plaisante et cohérente visuellement. Grâce à l’aggrandissment de l’encadré de la partie de la connexion, les boutons pour le captcha ne se chevauchent 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.

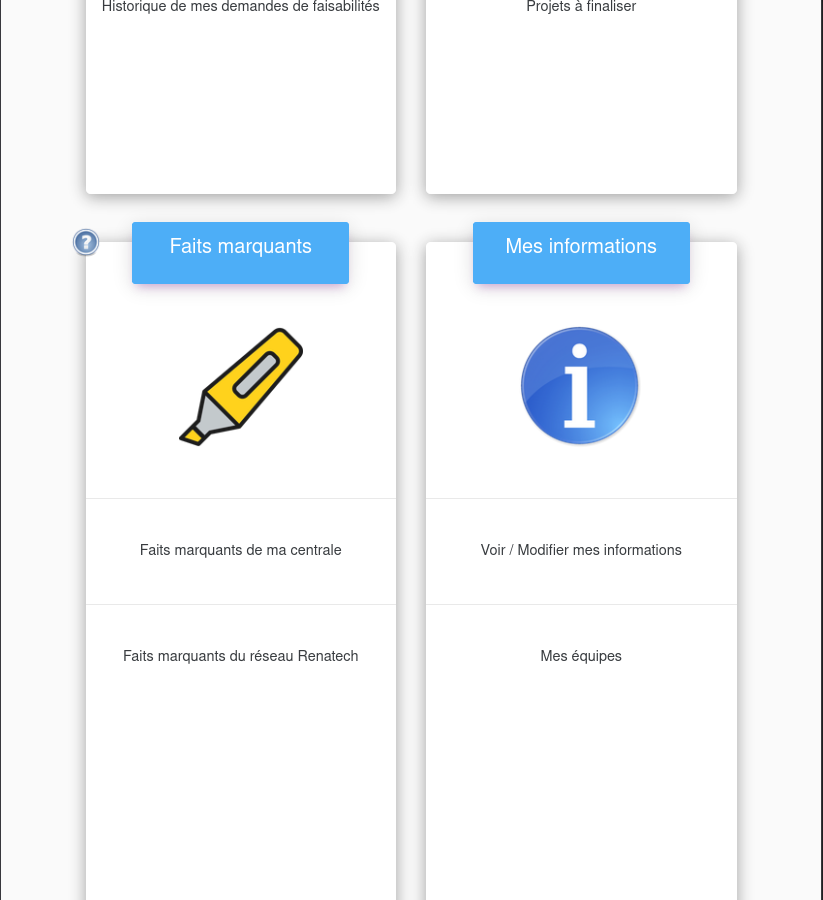
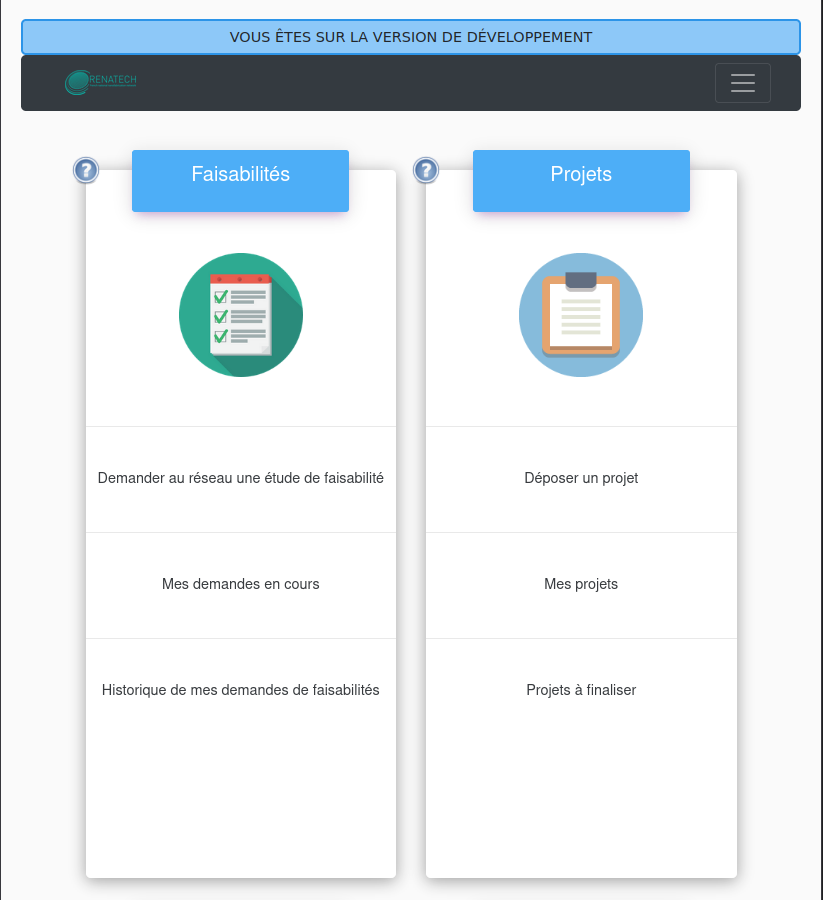
**Page de connexion version mobile (après responsive) :**

  
Comme on peut le voir ci-dessus la page de connexion version mobile après responsive est probablement la page où le changement est le plus flagrand j’ai fait de mon mieux pour que l’expérience sur mobile soit agréable pour les utilisateurs. Je ne me suis pas beaucoup attardé sur le captcha car il a été retiré peu après la réalisation de ma tâche car il posait des problèmes aux utilisateurs pour pouvoir se connecter à l’application.

**Page d’accueil version tablette (avant responsive) :**

Comme on peut le voir ici la page d’accueil sur un écran de type tablette est difficilement utilisable les boites d’aides dans le coin supérieur gauche des colonnes est inutilisable et quand on appuie sur le bouton menu qui se situe à droite les colonnes se chevauchent.

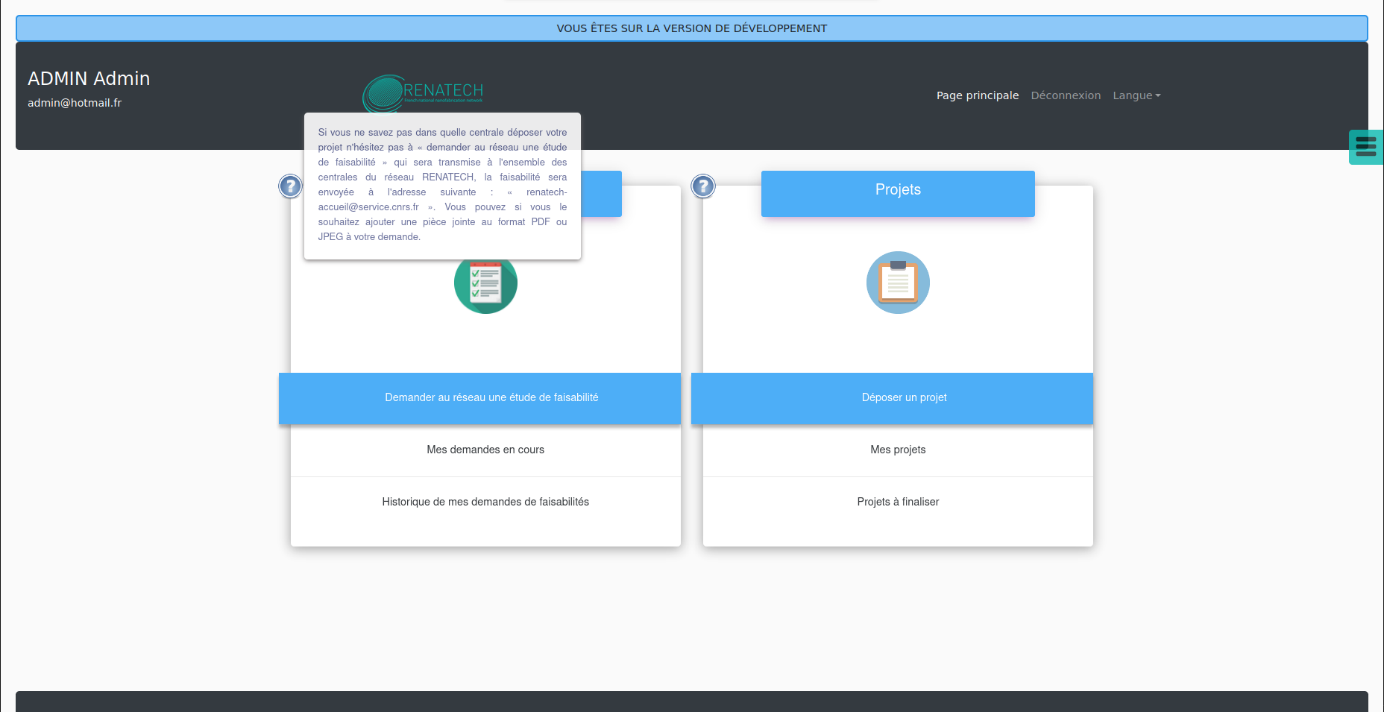
**Page d’accueil version tablette (après responsive) :**



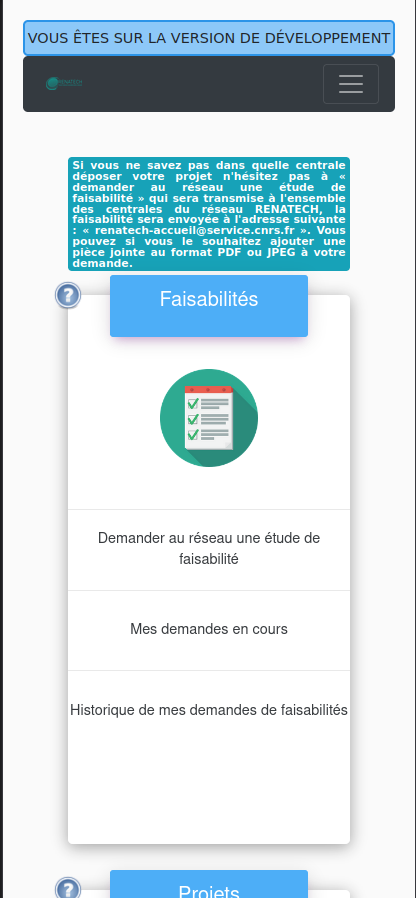
Pour remédier à ce problème j’ai fait à ce que quand la taille d’écran est égale ou inférieure à la taille d’un écran de tablette 2 nouvelles colonnes sont créés pour les faits marquants et les informations de l’utilisateur. Mais si on atteint une taille de type desktop l’affichage revient à l’affichage précédent qui ne pose aucun problème sur une taille d’écran de type desktop.

Pour les bulles infos pour l’aide sur desktop quand on passe notre curseur sur celles-ci une box apparait avec un texte d’aide, cela posait problème sur tablette et mobile comme nous n’avons pas de curseur. Donc quand j’ai retravaillé le responsive, j’ai pris la décision de laisser l’icône d’aide mais quand un utilisateur sur mobile ou tablette touchera cette icône un badge apparait avec le texte d’aide.

**Infobulle version desktop :**



**Infobulle version mobile (et tablette) :**



J’ai utilisé un badge car le popup de la version desktop aurait posé problème. En effet sur la version desktop il faut que le curseur reste sur l’icône pour pouvoir lire le texte d’aide or cela est impossible sur mobile comme il n’y a pas de curseur.

# Conclusion

Ce stage m’a beaucoup apporté et m’a été très instructif, 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 ont su me donner confiance en moi-même et 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.

La formation m’a également beaucoup apporté, j’y ai rencontré des personnes très aidantes et attentionnées qui ont su pousser le groupe vers le haut. Cela m’a beaucoup apporté sur le plan personnel.

# Sources

IEMN <https://www.iemn.fr>

CNRS <https://www.cnrs.fr/fr/le-cnrs>

RENATECH <https://www.renatech.org>

<https://cat.opidor.fr/index.php/RENATECH%2B>

<https://www.iemn.fr/cellule-technologique-renatech>

Python-Flask <https://flask.palletsprojects.com/en/2.3.x/>

Flask Mega Tutorial <https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>

Stack Overflow <https://stackoverflow.com>

Pytest <https://docs.pytest.org/en/7.3.x/>

Flask-Logging <https://flask.palletsprojects.com/en/2.3.x/logging/>

Flask-Mail <https://pythonhosted.org/Flask-Mail/>

Flask-Babel <https://python-babel.github.io/flask-babel/>