Projet de Groupe 2:
Mission Logement

Rapport Final

21/03/2025

Réalisé en Mars 2025 par Samer AYYOUB, Lorenzo BARBO, Quentin BUSSARD, Arman MOLA, Corentin POULET

Encadrant : Jean-Yves MARTIN
Client : Stéphanie BREUIL

I] Contexte

Le projet Mission Logement a été initié pour répondre aux besoins spécifiques du Service Vie Étudiante (VE) de l'École Centrale de Nantes en matière de gestion de l'attribution des logements étudiants. Traditionnellement, cette mission repose sur une permanence téléphonique assurée par des étudiants en contrats temporaires, qui collectent les demandes de logement des futurs étudiants de première année admis avec une réponse "Oui définitif" sur la plateforme SCEI. Les informations recueillies permettent d'établir un classement basé sur des critères de priorité, tel que le statut de boursier et la distance domicile-école, avant d'être transmises à la résidence pour une première répartition.

Cependant, cette procédure traditionnelle présente plusieurs limites, notamment des difficultés de recrutement des étudiants pour la permanence téléphonique, une saturation du standard téléphonique, et un manque de disponibilité pour des conseils personnalisés. Pour pallier ces difficultés, une transition vers une solution numérique a été proposée.

Le logiciel développé dans le cadre du projet Mission Logement repose sur un formulaire en ligne accessible via une plateforme web sécurisée. Ce formulaire simplifie la collecte des informations et réduit la charge sur le standard téléphonique. Le système inclut une authentification basée sur la liste des étudiants "Oui définitif", garantissant que seules les personnes éligibles peuvent accéder à la plateforme. Les informations collectées sont ensuite accessibles aux administrateurs et aux assistants, leur permettant d'examiner les dossiers et de traiter les demandes spécifiques. Une fois cette phase de traitement terminée, les données enrichies et validées sont exportées sous un format structuré pour la répartition finale des logements.

Le développement du logiciel répond à plusieurs priorités, notamment la sécurité des données, l'automatisation de la collecte des données, le traitement manuel des dossiers, et la facilitation de l'exportation des données. L'application a été développée avec le framework Java Spring, une base de données PostgreSQL, et une modélisation UML, assurant une architecture robuste et évolutive.

II] Rappels du cahier des charges et de la conception :

1. Cahier des charges

Le cahier des charges est segmenté en 3 grandes parties.

- Connexion et vérification (priorité)

Fonction 1 : Récupérer la liste et les informations des OUI DEF sur SCEI	
Objectif	Recenser l'ensemble des élèves qui pourront potentiellement demander une place à la rez. Ainsi que les informations (identité, ville, adresse mail, date de naissance, numéro de candidat) sur lesquelles il y aura un traitement.
Description	L'utilisation du logiciel par les étudiants ne pourra avoir lieu uniquement sous condition d'être en état "OUI DEF" sur SCEI. L'acquisition se fera grâce à un document CSV que l'admin téléversera sur le logiciel.
Contraintes / règles de gestion	Seul l'admin peut téléverser la liste des étudiants "OUI DEF" et en extraire le contenu sur le logiciel
Niveau de priorité	Haute

Fonction 2 : Première connexion sécurisée pour les étudiants avant la date limite	
Objectif	Permettre aux étudiants de s'inscrire sur la plateforme de la mission de manière sécurisée et avant la date limite décidée par l'admin.
Description	Les étudiants ayant répondu "OUI DEF" avant la date limite recevront un mail sur leur adresse, trouvée grâce à la fonction 1, dans lequel se trouvera un lien qui les enverra vers une plateforme de connexion où ils créeront leur identifiant et leur mot de passe.

Contraintes / règles de gestion	L'admin pourra décider de la date limite sur une plateforme dédiée. L'adresse mail sur laquelle sera envoyée le lien correspondra à celle liée au compte SCEI du candidat. Le lien devra ne plus être valide à la suite de la création des logins. Ainsi, l'étudiant veillera à bien les noter. Si l'étudiant cherche à créer ses logins après la date limite, le lien ne sera pas fonctionnel.
Niveau de priorité	Haute

Fonction 3 : Donner les informations concernant la mission logement	
Objectif	Afficher les différentes informations durant les différentes phases
Description	Le logiciel devra afficher à l'écran des candidats les informations suivantes selon la phase dans laquelle la mission logement se situe : ● la date de fin d'inscription ainsi que le numéro de téléphone du standard de la mission logement durant la phase d'inscription ● la date à partir de laquelle sera disponible le verdict (avoir une chambre ou non, en colocation ou non) après la fin de la mission logement ● Si un élève a créé ses identifiants mais n'a pas répondu à la mission logement seule la page d'information apparaît (et non son formulaire ou la réponse aux voeux)
Contraintes / règles de gestion	L'admin devra décider des dates de la mission logement sur le logiciel à l'avance, sinon la date du 1er Janvier 1970 sera affichée. L'affichage sera mis à jour à chaque début de nouvelle phase et les candidats déconnectés.
Niveau de priorité	Moyenne

Fonction 4 : Empêcher plusieurs connexions simultanée sur un même compte	
Objectif	Empêcher les candidats, les assistants ou l'admin de se connecter depuis différents appareils sur un même compte.
Description	Les candidats, assistants et l'admin ne pourront pas se connecter simultanément sur plusieurs appareils à leur compte pour empêcher de surcharger le serveur et éviter tout conflit d'information ou fraude
Contraintes / règles de gestion	

Niveau de priorité	Basse

Fonction 5 : Permettre aux utilisateurs de se connecter	
Objectif	Permet aux candidats, à l'admin et aux assistants de se connecter sur le logiciel de la mission logement.
Description	Les étudiants pourront se connecter avec les logins qu'ils auront créés avant le début de la mission logement. L'admin et ses assistants le feront avec des identifiants et mots de passe qui leur seront fournis.
Contraintes / règles de gestion	Les candidats devront nécessairement avoir créé leurs logins avant le début de la mission logement, sans quoi ils n'y auront pas accès. L'admin et ses assistants se connectent sur une plateforme que les candidats pour éviter qu'un candidat puisse avoir accès à l'ensemble du système. Au bout d'un certain temps d'inactivité, l'utilisateur sera déconnecté de sa session Ce sera l'admin (VE) qui créera les comptes des assistants
Niveau de priorité	Haute

Fonction 6 : Paramétrer la mission logement	
Objectif	Permet à l'admin de gérer les différents paramètres de la mission logement
Description	L'admin doit pouvoir paramétrer la date de la mission logement, le contenu des mails ainsi que les textes "statiques" modifiables sur le site.
Contraintes / règles de gestion	

Niveau de priorité	Haute

- Complétion du document gestion de la collecte des réponses

Fonction 1 : Donner accès à un formulaire prérempli	
Objectif	Les candidats, après s'être connectés à la mission logement, pourront remplir un formulaire pour la rez.
Description	Le formulaire demandera les informations suivantes : ● nom (*) ● prénom (*) ● genre (*)(!) ● numéro de téléphone ● adresse postale (*) ● adresse mail (*) ● bourse(*) (!) ● voeu du type de logement ● besoin logement adapté (PMR,...)(*) (!) ● commentaires Les informations préremplies grâce aux informations disponibles sur SCEI sont marquées d'un symbole '*' Les informations marquées d'un '!' lèvent une alerte lorsqu'elles sont modifiées par rapport au prérempli et notifient l'admin qui doit vérifier le statut de la demande La modification de l'adresse postale devra se faire uniquement en appelant le standard
Contraintes / règles de gestion	Dans le cas d'une demande de bourse ou d'un logement adapté, le téléversement de pièces justificatives sera demandé avant d'enregistrer le formulaire. Après avoir rempli le formulaire, le candidat devra enregistrer son formulaire.
Niveau de priorité	Haute

Fonction 2 : Modification et envoi des données	
Objectif	Les candidats pourront modifier leur formulaire jusqu'à validation total du formulaire
Description	Le formulaire chargera la dernière version enregistrée pour chaque candidat. Le

	candidat pourra alors la modifier s'il le faut. Pour enregistrer les modifications, il devra appuyer sur le bouton de validation qui rappelera que l'action est définitive, la date de la validation sera de plus récupérée.
Contraintes / règles de gestion	Le candidat pourra modifier autant de fois qu'il le souhaite son formulaire. Cependant, seule la version validée sera prise en compte dans la base de données à la fin de la phase de demande de logement.
Niveau de priorité	Haute

Fonction 3 : Aspect légal	
Objectif	Le candidat doit être notifié que ses informations seront gardées le long de la procédure de la mission logement.
Description	Le candidat pourra cocher pour accepter que ses informations soient conservées par le logiciel durant toute la procédure de la mission logement.
Contraintes / règles de gestion	Le candidat ne pourra pas enregistrer ses informations sans avoir accepter que ses informations soient stockées dans la base de données pendant toute la durée de la mission logement. Toutes les informations seront détruites automatiquement à la fin de la mission logement.
Niveau de priorité	Moyenne

Fonction 4 : Dépôt de documents	
Objectif	Lorsqu'il est nécessaire, le candidat doit pouvoir déposer les documents qui lui sont demandés.
Description	Sous chaque rubrique, une zone, initialement verrouillée, sera débloquée, lorsqu'il souhaite faire une demande de bourse ou de logement adapté, dans laquelle le candidat pourra téléverser les documents nécessaires.
Contraintes / règles de gestion	En cas d'erreur, le candidat doit pouvoir supprimer et ajouter le nouveau document.
Niveau de priorité	Moyenne

Fonction 5 : Modification par l'admin ou ses assistants	
Objectif	Les admins et ses assistants peuvent consulter et modifier toutes les informations de chaque candidat ainsi que valider le statut d'un dossier en cas d'alerte.
Description	Lorsque le dossier d'un candidat lève une alerte, l'admin ou un assistant doit l'analyser et le valider ou non. De plus, sur demande (notamment grâce au standard), les assistants et l'admin pourront modifier toute information. Ils pourront également laisser des commentaires.
Contraintes / règles de gestion	Si le dossier du candidat n'est pas validé, notamment lors d'une demande de bourse ou d'un logement adapté, le candidat devra en être notifié. Lors d'une demande de modification, le candidat, afin de justifier de son identité, devra fournir au standard les informations suivantes : son nom, prénom, adresse mail, numéro de candidat et son numéro de téléphone.
Niveau de priorité	Moyenne

- Résultats et système d'alerte

Fonction 1 : Notifier l'admin d'une alerte	
Objectif	Lorsqu'un candidat souhaite alerter qu'il est boursier ou qu'il a besoin d'un logement adapté, il téléverse ses documents. Ce qui doit envoyer une notification d'alerte à l'admin et les assistants.
Description	L'admin et ses assistants sont notifiés qu'un dossier nécessite une vérification. Ils ont accès à l'ensemble des dossiers à vérifier sur une interface.
Contraintes / règles de gestion	
Niveau de priorité	Moyenne

Fonction 2 : Export des formulaires	
Objectif	Permet à l'admin d'exporter les informations des formulaires afin d'effectuer le traitement
Description	L'admin pourra exporter les données sous le format csv ou excel afin de traiter et classer les dossiers
Contraintes / règles de	

gestion	
Niveau de priorité	Haute

Fonction 3 : Transmettre les résultats aux candidats	
Objectif	Une fois la période d'affectation terminée, les candidats doivent avoir accès à leur dossier.
Description	En se reconnectant à leur compte après la période d'affectation, les candidats pourront savoir s' ils ont obtenu une place à la rez ou non.
Contraintes / règles de gestion	
Niveau de priorité	Basse

Note : ce cahier des charges a évolué en cours de projet (voir plus loin)

De plus, nous avons convenu 3 livrables : l'application, un mode d'emploi utilisateur et un mode d'emploi technique développeur.

2. Diagramme des cas d'utilisation

On définit par ailleurs les acteurs du projet : la VE (administrateur de de la mission logement), les élèves qui participent à la mission logement, et les assistants (les élèves en contrat étudiant) qui traiteront les dossiers avec la VE.

Avec le cahier des charges précédents et ces acteurs on obtient le diagramme suivant :

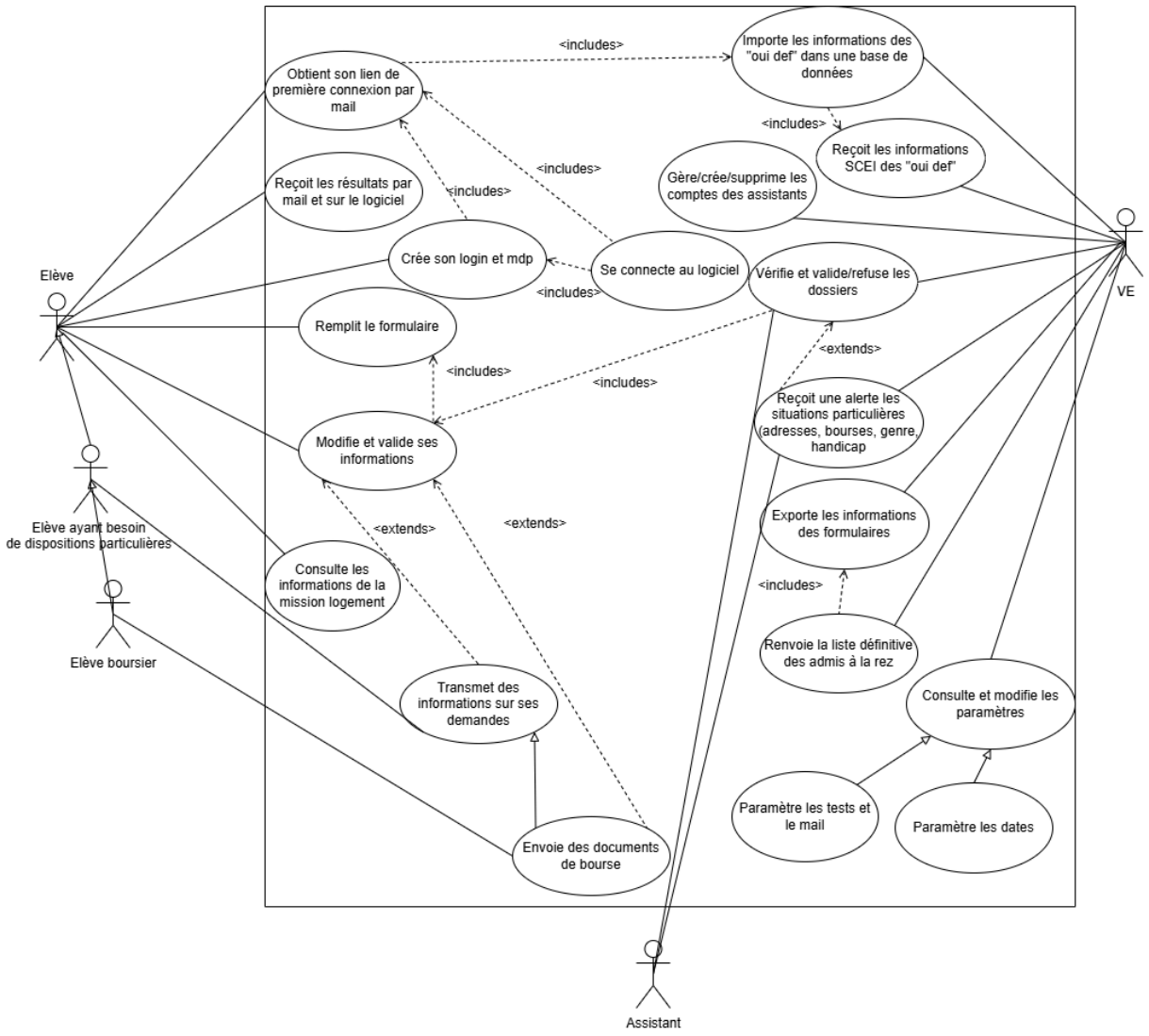

Figure 1 : Diagramme des cas d'utilisation

3. Base de données

Pour la structure de la base de données relationnelle, la base de données du dernier projet d'application a été modifié pour être cohérente avec les besoins mis à jour du projet de groupe :

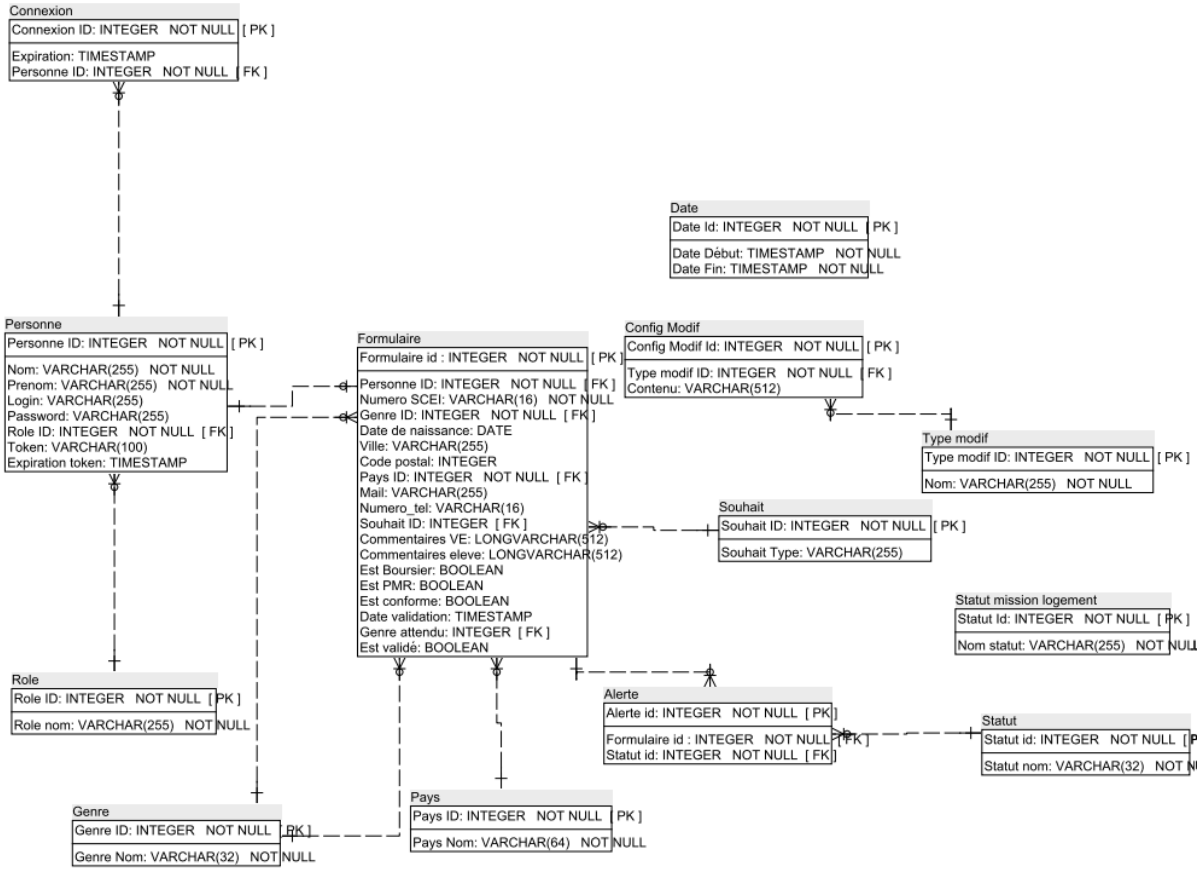

Figure 2 : Modèle physique de la base de données

Nous détaillons maintenant les choix de ce modèle :

- La table connexion permet de gérer le temps de connexion
- Un formulaire contient les informations du formulaire remplis mais aussi s'il est validé par l'élève, s'il est conforme pour être traité ainsi que sa date de validation. Il contient aussi s'il a obtenu une chambre à la fin du traitement. Elle contient aussi le token de première connexion (utilisé pour les élèves seulement afin de générer un lien unique pour se connecter, une fois les identifiants créés, on supprime le token et le lien devient inaccessible), il contient aussi un genre attendu qui est le même que le genre obtenu sur SCEI (il sert à comparer avec le genre rentré par l'élève sur l'application)
- La table date contient les dates de la mission logement
- La table de souhait contient les types d'appartement que l'on peut souhaiter (coloc, seul, peu importe)
- Le statut d'une alerte peut être traitée ou en cours
- La table statut mission logement représente l'état ouvert ou fermé de la mission logement. Elle contient une seule ligne.
- configModif et typeModif représentent les textes statiques de l'application et où les placer dans l'interface

III] Déroulement du projet

Le projet Mission Logement a suivi une méthodologie rigoureuse et progressive, en s'appuyant sur plusieurs étapes clés pour assurer un développement structuré et efficace.

1. Prise de connaissance de PAPPL et cadrage du projet

Au début du projet, nous avons pris connaissance du travail réalisé lors du précédent Projet d'Application (PAPPL). Cette première phase nous a permis d'analyser l'état de l'application, d'identifier les éléments à améliorer et d'aligner nos objectifs avec les besoins du Service Vie Étudiante (VE).
Nous avons ensuite cadré le projet en définissant précisément les fonctionnalités attendues, les contraintes techniques et les priorités de développement.

2. Organisation du travail en équipes

Afin d'optimiser l'efficacité du développement, nous avons divisé le groupe plusieurs équipes en fonction des groupements de tâches à effectuer (voir annexes et planning).

Le travail de développement a été aidé par l'utilisation d'un Github et l'utilisation de branches pour chaque grand compartiment de l'application. Ces branches étaient les suivantes :

- Main
- Import
- Export
- Initialisation
- Paramètres
- Première connexion
- ImportExport
- Mail
- Token
- Suppression données
- Traitement des dossiers

3. Communication et collaboration

Tout au long du projet, nous avons mis en place une communication fluide et efficace entre les membres de l'équipe. Des réunions régulières ont été organisées pour faire le point sur l'avancement des tâches, résoudre les problèmes rencontrés et valider les choix techniques.
En parallèle, des échanges fréquents avec le Service Vie Étudiante ont permis d'obtenir des validations intermédiaires et d'adapter le projet aux besoins réels des utilisateurs.

4. Semaine de projet et développement intensif

Une semaine complète a été dédiée à un développement intensif. Durant cette période, nous avons avancé sur plusieurs aspects critiques du projet, notamment :

- L'intégration de la gestion des comptes et de l'authentification.
- L'amélioration de l'interface utilisateur pour une meilleure expérience.
- L'implémentation des fonctionnalités de collecte et de traitement des dossiers.
- La mise en place des mécanismes de validation et d'export des données.

Cette semaine de travail intensif a permis d'effectuer des tests approfondis et d'améliorer la cohérence globale du système.

5. Validation et ajustements

À la suite du développement, nous avons réalisé une série de tests pour identifier d'éventuels dysfonctionnements et apporter des corrections. Des ajustements ont également été effectués en fonction des retours des utilisateurs et de l'encadrant du projet.

Toutefois, les tests unitaires sont absents du projet, car la quasi-totalité des fonctions ne se vérifient pas automatiquement (des tests d'intégration auraient été plus judicieux).

6. Bilan de la collaboration

La bonne communication au sein de l'équipe a été un facteur clé de succès du projet. La répartition claire des tâches et l'implication de chacun ont permis d'atteindre les objectifs fixés dans les délais impartis.

IV] Réalisations

1. Rappel de l’état de l’application en fin de PAPPL

Dans cette partie, on rappelle l’état de l’application à la fin du premier projet d’application, terminé en janvier.

A ce moment là :

- La page de connexion était fonctionnelle, mais il n’y avait pas de hachage des mots de passe.
- La navigation entre les pages n’étaient pas sécurisées
- Sur la vue admin :
    - L'accueil était fonctionnelle
    - On ne pouvait pas gérer les assistants, seule la page de la liste des assistants existait
    - L’admin ne pouvait pas paramétrer l’application
    - Il pouvait accéder à la liste des dossiers transmis, mais il ne pouvait pas consulter les détails d’un dossier
- Sur la vue assistant :
    - Tout comme la vue admin, les assistant ont accès à la liste des dossiers transmis mais n’ont pas accès aux détails

- Sur la vue étudiant :
    - L’accueil des étudiants était fonctionnel
    - Ils avaient accès au formulaire à compléter, mais ne pouvait ni l’enregistrer ni le soumettre

2. Avancement à la fin du PGROU

L'ensemble des fonctionnalités est basé sur le modèle MVC (Model View Controller) du framework Java Spring :

- Les views sont des fichiers .jsp qui contiennent le code HTML, CSS et JavaScript classique dans le développement web, c'est ce que le client voit
- Les controller récupèrent grâce aux Servlet (objets qui assurent la communication des requêtes http des vues au serveur), récupèrent dans le Model les données nécessaires à chaque routes et renvoient vers une autre vue (voir exemple en annexe).
- Le Model est constitué de deux éléments :
    - Les Entités qui sont les représentations des tables de la base de données, cette représentation est automatiquement adaptée à la base de données par l'ORM (Object-Relational-Mapping) de Spring : JPA (Java Persistence API).
    - De plus, nous utilisons des repositories pour communiquer avec la base de données. Grâce à JPA, on assure la persistance des données, c'est-à-dire la correspondance entre l'objet de java et l'élément de la base de données lors d'une requête (voir exemple en annexe).

Nous détaillons maintenant les réalisations de l'application.

a. Côté Admin :

Paramétrage

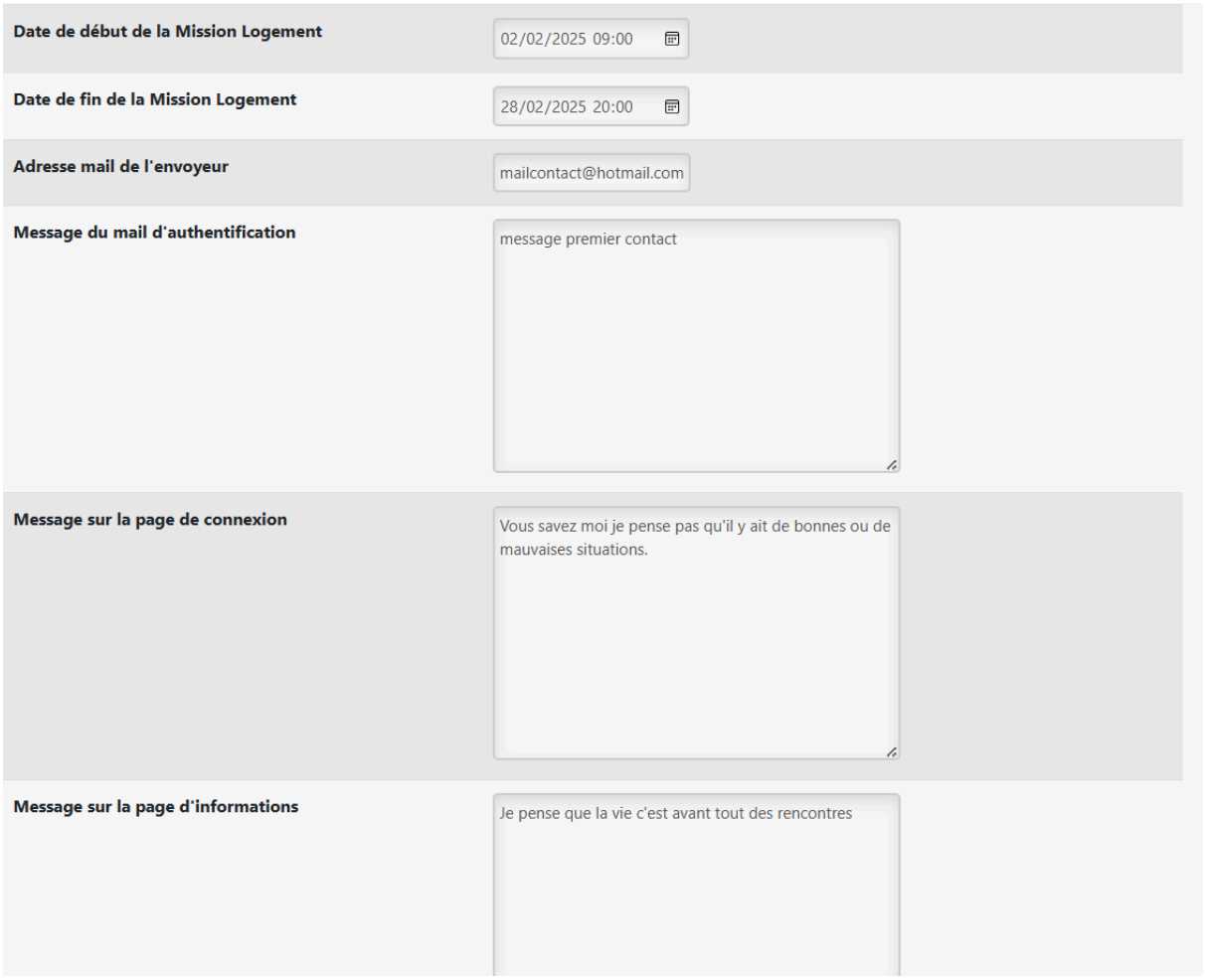

Figure 3 : Haut de la page de paramétrage

Une fonctionnalité primordiale du logiciel est la configuration de la mission logement. Par configuration, il entend non seulement le choix du début et de la fin de l'ouverture de la mission logement mais aussi l'adresse mail utilisée ou le choix de la plupart des textes d'informations (le contenu du mail envoyé à chaque étudiant, par exemple).

Pour cela, nous avons donc créé la page "Configuration", uniquement accessible à l'admin. Sur cette page, l'admin pourra choisir les dates de début et de fin de

l'ouverture de la mission logements grâce aux deux premiers champs.Ensuite, il peut également choisir de changer l'adresse mail utilisée par la mission logement grâce au troisième champ. Finalement, il peut, s'il le souhaite, modifier certaines zones de textes de l'application grâce aux derniers champs.

Lorsque l'admin a fini la configuration de la mission logement, il lui suffit de cliquer sur le bouton "Enregistrer" en bas de page pour enregistrer et appliquer toutes les modifications qu'il a faites. De plus, lorsque l'admin se retrouve sur la page de configuration, nous avons fait en sorte que les versions actuelles des différents champs lui sont montrées.

Pour l'utilisation du contenu des différents champs :

- Si certains ont été modifiés, on enregistre leur nouvelle valeur dans la table config_modif en précisant quelle type de modif a été faite (grâce à une table de nomenclature type_modif)
- Pour récupérer la valeur actuelle de chaque champs, on récupère dans la table config_modif, pour chaque type de modification, la ligne ayant l'identifiant le plus grand
- Pour l'affichage des valeurs, on utilise des controllers pour chacune des utilisation (un controller pour l'affichage du texte d'information sur la page de connexion par exemple)

Figure 4 : Bas de la page de paramétrage

Une étape essentielle de la sécurité de l’application est la sécurisation de la création des comptes. Nous avons conçu un système de gestion de la première connexion où, après l’import de la liste des élèves ayant répondu "OUI DEF" après les concours, nous générons un token unique par élève. Ce token est stocké dans la base de données avec une date d’expiration légèrement supérieure à la durée de la mission.

Cette étape de génération des tokens est déclenchée lorsque l’administrateur clique sur le bouton "Envoi de mail", ce qui envoie à chaque étudiant un e-mail contenant un lien sécurisé intégrant son token unique.

Ce lien redirige l’étudiant vers la page de première connexion, où il doit saisir son numéro SCEI pour renforcer la sécurité. Il choisit ensuite son identifiant et son mot de passe. À cette étape, nous vérifions :

1. Si le numéro SCEI saisi correspond au numéro associé au token,
2. Si l’élève ne possède pas déjà un compte existant. (la column login est de valeur null)

Si toutes les vérifications sont validées, nous enregistrons l’identifiant et le mot de passe haché dans la base de données. Une fois le compte de l’étudiant créé, le token est supprimé de la base de données, rendant ainsi le lien d’activation invalide.

Malheureusement, en raison de problèmes techniques liés à l’envoi des emails et des autorisations nécessaires auprès de la DSI, ce système, bien que implémenté, n'est pas encore utilisé.

Afin de garantir une application fonctionnelle pour juillet 2025, nous avons mis en place une alternative simplifiée, tout en maintenant un bon niveau de sécurité.

Nouvelle Procédure de Première Connexion

1. Envoi d’un email contenant un lien commun à tous les étudiants.
2. Saisie de l’email sur la page de connexion.
    - Si l’email correspond à un étudiant présent dans la base (après l’import des "OUI DEF"), l’accès à l’étape suivante est validé.

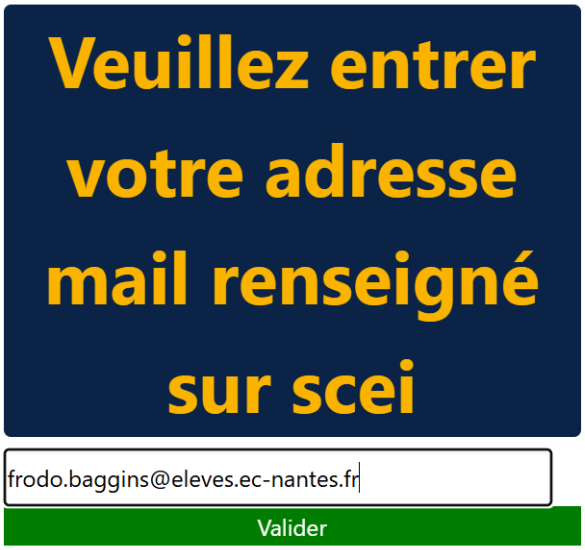

Figure 5 : Page de vérification du mail lors de la première identification

3. Vérification d'identité :
    - L'étudiant doit saisir son numéro SCEI.
    - Nous vérifions si l'email et le numéro SCEI correspondent à la même personne.

Figure 6 : Page de vérification du mail lors de la première identification

4. Création du compte :
    - Si les informations sont correctes et que l'étudiant n'a pas encore de compte, il peut définir son identifiant et son mot de passe.
    - Comme dans la version initiale, le mot de passe est haché avant d'être stocké.

Ce nouveau système assure un équilibre entre simplicité et sécurité, tout en nous permettant de respecter les délais pour la mise en production de l'application.

Import des données

Une des fonctionnalités indispensables développée durant le PGROU est l'import des données.
Cette fonction permet à l'administrateur de choisir un fichier csv contenant une liste d'élèves qui seront alors automatiquement ajoutés à la base de données. Pour cela on utilise un *BufferedReader* qui lit la première ligne du fichier, détectant le séparateur utilisé et attribuant chaque colonne à un champ différent. Le programme lit ensuite chaque ligne du fichier en créant les entitées correspondantes de la base de données au fur et à mesure de l'avancée.
Il peut arriver que le document csv soit mal configuré : nom de colonne mal orthographié, information manquante,... Si jamais cela arrive le programme saute la ligne illisible (ou le fichier complet si besoin) et écrit dans un fichier texte l'heure de l'erreur, le numéro de la ligne concerné, l'erreur et la conséquence de celle-ci.

Pays	Nom	Prénom	Date de naissance	Mail	Numero SCEI	Ville	Genre	Code Postal
France	Baggins	Bilbo	15/01/2002	bilbo.baggins@eleves.ec-nantes.fr	27589	Nantes	Homme	44000
France	Baggins	Frodo	17/05/2003	frodo.baggins@eleves.ec-nantes.fr	10004	Paris	Homme	75009

```
2025-03-18 09:55:28 -
temp14404228964640136122FichierImportParfait.csv : la colonne Nom
n'a pas ete trouvee, annulation de l'import
2025-03-18 09:59:07 - temp690248102872720397 1rapportErreur.txt :
la colonne Genre n'a pas ete trouvee, annulation de l'import
2025-03-18 10:08:41 - Import terminé !
2025-03-18 17:51:36 - Import terminé !
2025-03-18 20:00:30 - Import terminé !
2025-03-18 20:05:43 - Import terminé !
2025-03-18 20:07:14 - Import terminé !
2025-03-18 20:10:15 - Import terminé !
2025-03-18 20:16:06 - Import terminé !
2025-03-20 15:20:22 - Import terminé !
```

Figure 7 : Fichier d'import et rapport d'erreur

Export des données

Une autre fonctionnalité implémentée pendant la semaine de projet était l'export des données des formulaires au format CSV.
Pour cela, on récupère les formulaires puis on écrit les données dans un fichier grâce à un BufferedWriter. Une fois ce fichier écrit, il est créé sur le serveur.
Ensuite il faut récupérer le fichier sur le serveur et l'envoyer sur le client :
On crée alors un ResponseEntity qui renvoie le fichier au client. Pour cela, on récupère le fichier en créant un objet File à partir du chemin sur le serveur. On utilise ensuite une méthode sendFile fournie par notre encadrant qui va obtenir le fichier et l'envoyer au client (voir annexe).
Le fichier CSV obtenu reprend les colonnes de la table formulaire et personne de la base de données et est nommé avec l'instant de l'import pour assurer l'unicité du document.

Figure 8 : Liste des dossiers transmis et fichier d'export

Sécurisation des connexions

L'un des aspects clés de notre application est la gestion des connexions et la sécurisation des pages et fonctionnalités. Pour cela, nous avons mis en place un système de connexion centralisé qui assure :

- L'authentification des utilisateurs,
- La gestion des droits d'accès selon les rôles,
- Le contrôle du statut de la mission pour restreindre certaines actions,
- L'actualisation et l'expiration des connexions pour renforcer la sécurité.

Accès sécurisé aux fonctionnalités

À chaque tentative d'accès à une page ou une fonctionnalité sécurisée, le système vérifie si l'utilisateur est bien authentifié et possède le rôle requis. Dans le cas contraire, l'accès est refusé. De plus, toutes les connexions expirées sont automatiquement supprimées, et si une connexion est valide, sa durée est prolongée pour éviter les déconnexions inopinées.

Authentification et durée de connexion

Lorsqu'un utilisateur s'authentifie, une connexion temporaire est créée avec une durée de validité de 30 minutes. Toute activité de l'utilisateur prolonge cette durée, garantissant ainsi une session active tant qu'il utilise l'application.

Gestion du statut de la mission

Certaines fonctionnalités sont accessibles uniquement à des moments précis de la mission. Par exemple :

- Avant la mission : certaines pages sont restreintes aux administrateurs,
- Pendant la mission : les élèves ont accès à des outils spécifiques,
- Après la mission : certaines actions ne sont plus disponibles.

Le système ajuste donc automatiquement les pages accessibles et les redirections en fonction du statut actuel de la mission. Le statut de la mission est modifiable uniquement par l'admin dans la page de configuration.

Actuellement, les seules fonctionnalités restreintes par ce système sont celles du formulaire côté élève. Le formulaire est accessible uniquement lorsque la mission est en cours.

Affichage et gestion des vues

Pour garantir une navigation fluide, toutes les pages de l'application prennent en compte les informations de connexion. Cela permet d'éviter les erreurs d'accès et d'assurer que l'utilisateur est toujours identifié lorsqu'il navigue.

Envoi de mails

Lors du lancement de la mission logement, il faut envoyer un lien personnalisé aux étudiants contenant un token, pour s'assurer que ce lien ne soit utilisé uniquement par

l'étudiant concerné. Il faut donc l'envoyer de manière personnalisée, c'est-à-dire qu'il faut créer un mail unique pour chaque étudiant. Ainsi, l'objectif de cette fonctionnalité est de créer une boîte d'envoi qui permet d'envoyer un mail unique à l'adresse mail de chaque étudiant.

Comment ça marche :

- On récupère le mail des étudiants.
- On génère un token.
- On envoie un mail à chaque adresse mail avec JavaMail, qui est configurée avec les informations de l'école et son serveur SMTP (Simple Mail Transfer Protocol) qui est le protocole classique d'envoi de mail.

Malheureusement, cette fonctionnalité n'est pas disponible cette année (voir difficultés)

Suppression des données

Pour respecter la RGPD et ne pas avoir de problème lors de l'année suivante, il est nécessaire de posséder un moyen de supprimer l'ensemble des données élèves de la base de données.
Ainsi l'administrateur à accès à un bouton *supprimer les données* qui déclenche des commandes SQL TRUNCATE et DELETE sur les tables contenant les données élèves. Les tables de nomenclatures et les entités administrateurs ne sont pas modifiées par cette action ce qui permet d'être immédiatement prêt à déclencher une nouvelle session au besoin.

Traitement d'un dossier

Lors du traitement d'un dossier par un assistant ou un administrateur, après relecture du dossier, celui-ci doit pouvoir valider, refuser ou enregistrer un dossier. Lorsque les mails étaient prévus, un mail devait être envoyé en cas de refus ou de validation.
Un assistant ou l'administrateur peut écrire un commentaire sur le traitement du dossier, lorsqu'il y a refus de ce dossier, celui-ci est affiché au début du formulaire sur la vue de l'étudiant.
Pour faire cela, il y a trois routes (valider, enregistrer, refuser) qui enregistrent les nouvelles informations du formulaire puis met à jour le statut est_conforme de la base de données. De plus, si une alerte est résolue, elle est considérée comme traitée. Enfin, le pictogramme de la liste des dossiers est mis à jour en fonction du statut.

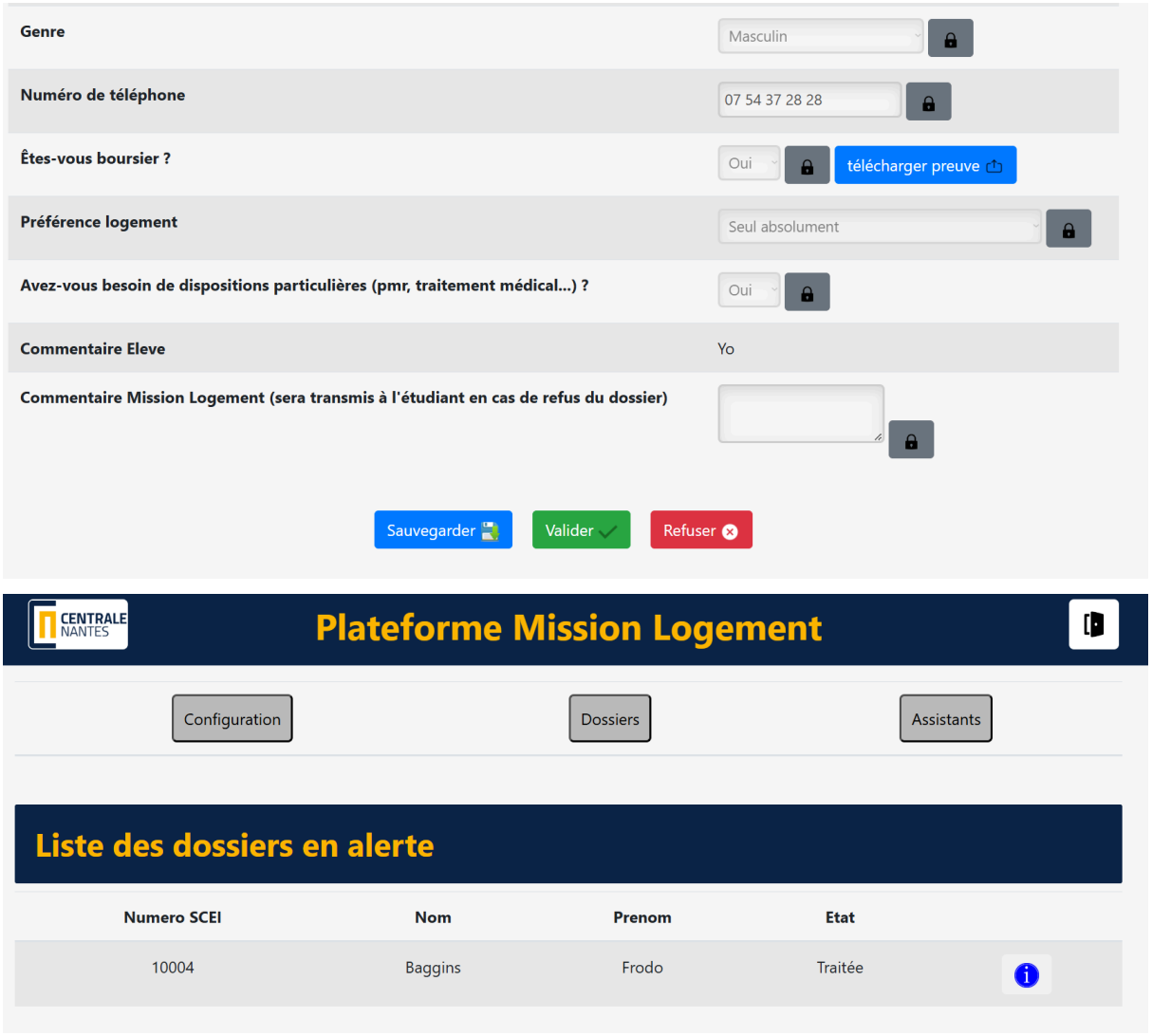

Figure 9 : Bas de la page de traitement d'un dossier et accueil de l'administrateur

Liste de tous les dossiers et traitement d'urgence

A l'origine il avait été imaginé que les administrateurs et assistants n'avaient accès qu'aux formulaires validés par les étudiants. Néanmoins il a été soulevé qu'il pouvait arriver qu'un étudiant n'ayant pas accès à internet veuille remplir son dossier en appelant le standard. Pour répondre à ce besoin il à été nécessaire d'imaginer une seconde page qui affichera l'ensemble des dossiers et permettra à l'assistant de modifier les informations non modifiables par l'étudiant et de valider le dossier à la place de l'étudiant.
Néanmoins pour respecter la RGPD nous ne pouvons pas avoir accès aux champs rentrés par l'étudiant tant que celui-ci n'a pas validé lui-même. Ainsi si un administrateur valide un formulaire, cela le vide entièrement (exception faites des

champs sur lequel l'élève n'a pas la main) avant de la valider. L'administrateur pourra ensuite modifier librement l'ensemble du dossier lorsque celui-ci aura été transmis.

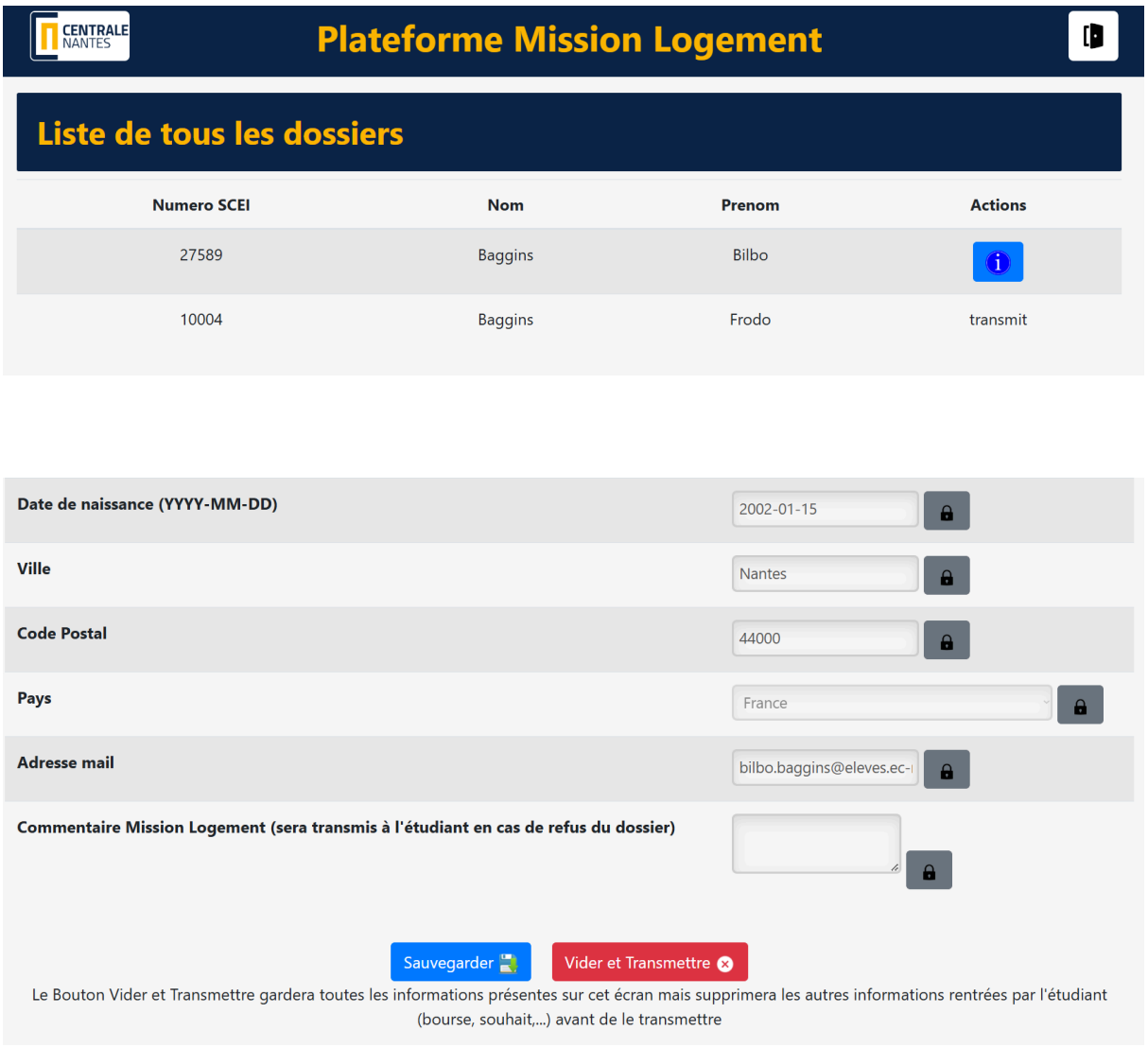

Figure 10 : Liste de tous les dossiers et vue d'un dossier non transmis

Gestion du compte admin

En cas de fausse manipulation menant à la suppression de l'administrateur ou à la perte de son mot de passe, il est nécessaire de posséder un moyen permettant de créer un nouvel admin ou de changer son mot de passe.

Pour répondre à ce problème nous avons imaginé un .jar qui serait présent sur la machine de la mission logement, qui détecte si un admin est présent dans la bdd et en fonction permet d'en créer un nouveau ou de modifier celui présent.

Cette application fonctionne en ligne de commande, et utilise les fonctions de hachage de Spring.

b. Côté Assistant :

Création, suppression et édition des assistants :

Lors de la mission logement, la vie étudiante (admin) est aidée par des étudiants (assistants). Il faut donc une fonctionnalité permettant de créer, modifier ou supprimer des assistants.

Pour cela, lorsque l'admin se connecte, il peut accéder à la page de gestion des assistants. Il est la seule personne à pouvoir accéder à cette fonctionnalité. Sur cette page, il peut effectuer trois actions :

- Ajouter un assistant :
  il faut qu'il ajoute ces informations :
  - Prénom
  - Nom
  - Login
  - Mot de passe
- Supprimer un assistant (à la fin de la mission)

  Cette action supprime tous les droits, informations de l'assistant en question.

- Modifier un assistant :

  - Tous les paramètres peuvent être modifiés

Bonnes pratiques :

- À la fin de la mission, supprimer tous les assistants pour éviter de laisser traîner des mots de passe qui peuvent être compromis.
- Si un assistant oublie son mot de passe, il est très simple de le réinitialiser.

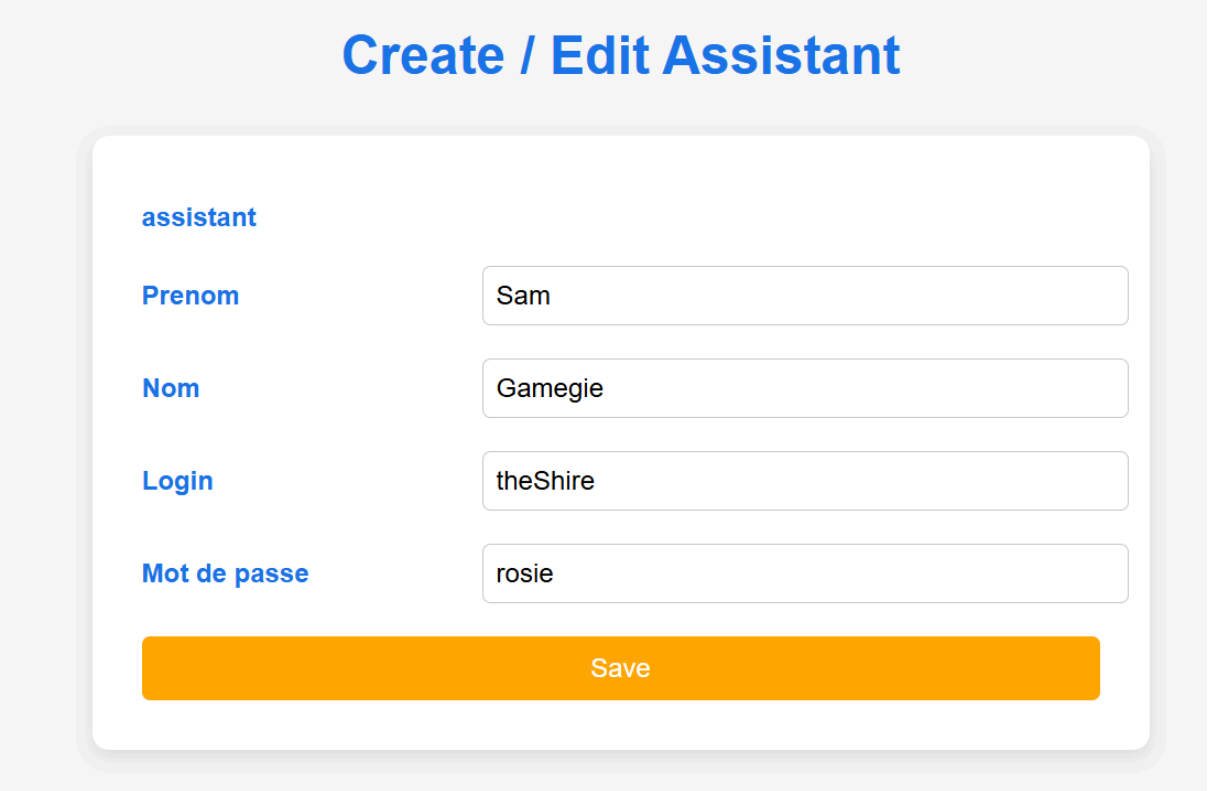

Figure 11 : Création d'un assistant et liste des assistants

Concernant le traitement des dossiers, le fonctionnement est le même que pour l'administrateur, sauf que l'export des données n'est pas possible.

c. Côté Étudiant :

Figure 12 : Page de connexion de l'application

Lorsque les mails étaient encore prévus, le fonctionnement était le suivant : L'étudiant recevait le mail avec son token dans l'URL menant à la page de première connexion, il pouvait ensuite entrer son numéro SCEI, qui était ensuite comparé à celui associé au token, il y avait donc contrôle de l'identité fiable (token+SCEI), il pouvait ensuite rentrer ses identifiants voulus pour ensuite se connecter avec, on détruisait ensuite le token. Tout cela se réglait avec des appels à la base de données pour récupérer le numéro SCEI attendu et vérifier que le login n'était pas déjà utilisé. A présent, comme les mails ne sont pas envoyés automatiquement, il n'y a plus de token. La personne passe donc par un lien commun à tous rentre son mail qui est vérifié dans la base de données qui l'envoie ensuite, si le mail est correct sur la page de première connexion habituelle, cependant …

Pour gérer la réinitialisation des mots de passe en cas d'oubli, nous avons implémenté un système similaire à celui de la première connexion.

Procédure de réinitialisation :

1. Demande de réinitialisation :
    - Lorsqu'un étudiant clique sur "Mot de passe oublié", il est redirigé vers une page où il doit saisir son email.
    - Si cet email correspond à un étudiant enregistré dans la base de données, un token unique est généré et envoyé à l'étudiant par email.
2. Réinitialisation :

    - Le lien envoyé par email contient ce token unique et redirige l’étudiant vers une page de réinitialisation du mot de passe.
    - L’étudiant saisit alors son nouveau mot de passe.
3. Vérification et sauvegarde :
    - Le système vérifie que le token correspond bien à un étudiant existant dans la base de données et que l’étudiant dispose d’un compte valide.
    - Si tout est valide, le mot de passe est haché et enregistré dans la base de données et le token supprimé.

Malheureusement, en raison des problèmes techniques liés à l'envoi des emails automatiques personnalisés, nous n'avons pas pu mettre ce système en place dans l’application actuelle.

Solution alternative pour l’application actuelle :

- Lorsqu’un étudiant clique sur "Mot de passe oublié", il est redirigé vers une page qui informe l’étudiant que le service de réinitialisation est actuellement indisponible.
- Il est précisé que l’étudiant doit contacter la mission logement par téléphone.
- L’administrateur ou l’assistant pourra alors saisir les informations de l’élève pour faire la demande grâce à la liste de tous les dossiers.

Remplissage formulaire

Quand les étudiants se connectent, ils doivent remplir un formulaire avec différentes informations nécessaires pour faire une demande de logement.

Techniquement, il y a eu plusieurs contraintes dans ce formulaire :

- Pour valider un bouton, il faut que tous les champs obligatoires soient remplis ; sinon, on ne peut pas effectuer d’actions sur celui-ci.
- Une fois le formulaire soumis, on ne peut plus le modifier.

Ainsi, nous avons dû paramétrer le formulaire avec des conditions pour vérifier si les champs étaient validés. Ensuite, nous avons dû ajouter certaines fonctionnalités comme "required" pour obliger l’utilisateur à remplir les champs.

Finalement, quand on sauvegarde le formulaire, il faut que les informations apparaissent à la prochaine ouverture de celui-ci. Au début, une valeur par défaut est présente dans la base de données, que l’on change lorsqu'une personne sauvegarde le formulaire pour l’afficher la fois suivante.

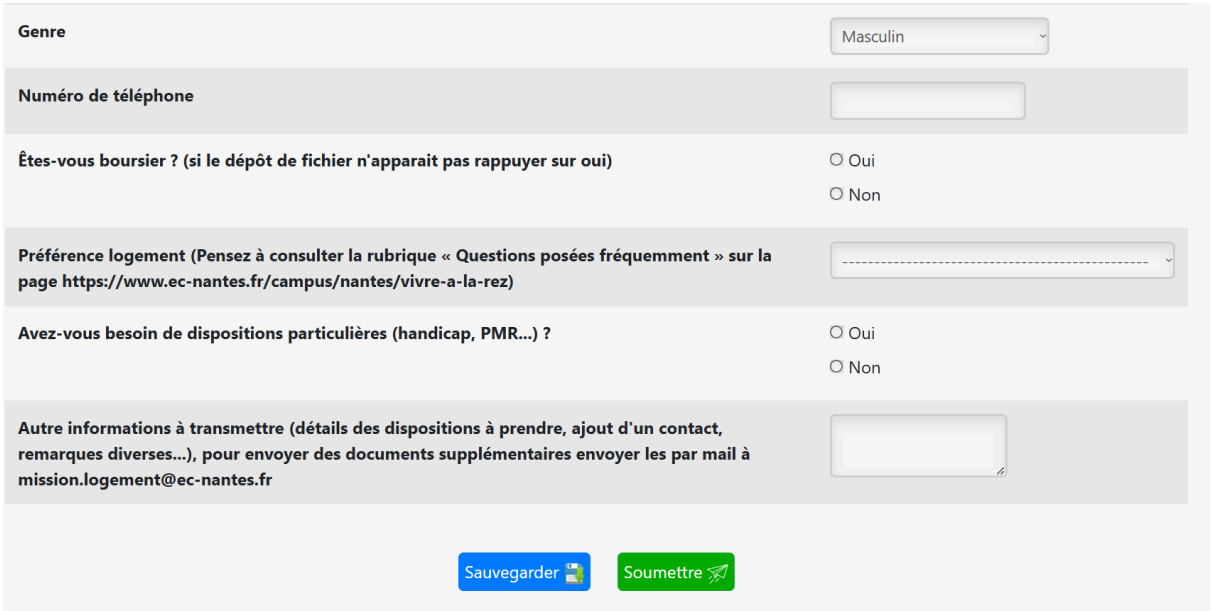

Figure 13 : Bas de la page de remplissage d'un formulaire

Note : lorsque leur dossier est refusé ou validé, il devait y avoir un mail automatiquement envoyé. Avec les difficultés des mails, les mails devront être envoyés manuellement. En cas de refus le commentaire laissé par l'assistant ou l'admin est placé en rouge au dessus du formulaire de l'élève

Informations pour un étudiant

Un étudiant, lors du remplissage de son formulaire, peut nécessiter de l'aide de la part de la mission logement. Il peut alors aller consulter la page d'informations où se trouvent, par exemple l'adresse mail ou le numéro de téléphone de la permanence (à la place de "Je pense que la vie c'est avant tout des rencontres" ci-dessus).

Pour afficher ce texte, on utilise un controller qui fait une requête à la base de donnée pour récupérer le texte à afficher. Plus précisément, dans la table config_modif, on demande à récupérer le contenu (le texte) de la ligne ayant l'identifiant le plus grand et ayant comme type de modification message_page_informations.

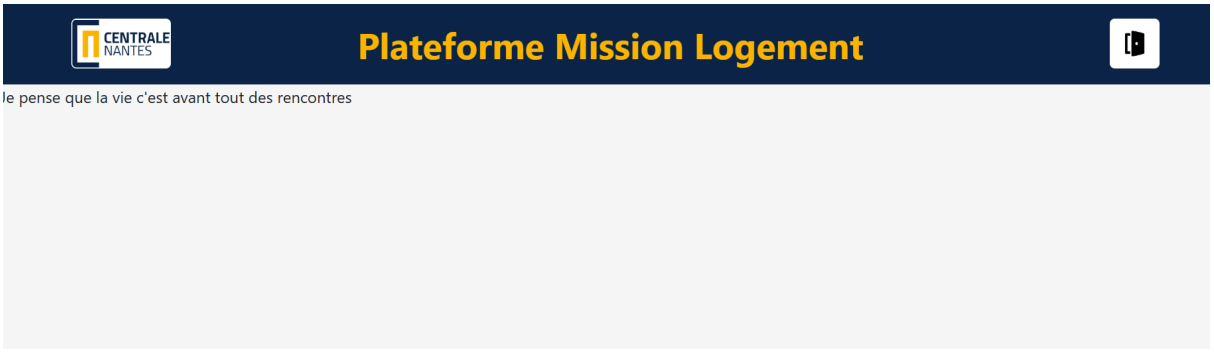

Figure 14 : Page d'informations accessible aux élèves

Import de documents

Concernant l'import de document, cette problématique n'a été réglée qu'en toute fin de projet. En effet, le moyen trouvé ne fonctionne pas pour une raison inconnue. On a donc dû changer de méthode en partenariat avec notre encadrant et cela a permis d'implémenter l'import d'un document de bourse sur le formulaire (voir annexe). Il peut ensuite être récupéré sur la page de dossier de la vue admin/assistant.

V] Difficultés rencontrées

A la fin du projet, un sujet a émergé, comment mettre en production l'application ? Cependant, les dialogues avec la DSI de l'école Centrale ont mis en évidence que l'application ne pourrait pas être déployée sur un serveur de l'école dès cette année. Notre encadrant utilisera ses ressources pour résoudre ce problème cette année.

Pour les mails, pour les mêmes raisons comme évoqué précédemment, le non-accès à un serveur SMTP ne nous permet pas d'envoyer des mails aux étudiants de manière généralisée, ce qui a amené au retour aux mails manuels.

De plus, nous devions permettre une meilleure gestion des listes de dossiers grâce à la bibliothèque Data Table de CSS. Toutefois des erreurs ont gêné son implémentation et la fonctionnalité n'est donc pas disponible.

VI] Bilan et futur du projet

Ce qu'il reste à faire :

Pour le futur projet, plusieurs ajouts sont à faire sur le projet (voir guide du développeur pour les détails) :

- Mettre en pratique le code fait pour envoyer des mails et étendre les mails aux validation, refus de dossier ainsi qu'aux réinitialisation de mots de passe
- Le code bien que fonctionnel, souffre de plusieurs défauts :
    - Le trop grand appel aux ModelAndView au lieu de faire des redirections aux routes, ce qui augmente le risque d'erreur
    - Le hachage des mots de passe est possiblement trop lent (en tout cas la connexion à l'application était très lente lors du développement à cause de ce hachage), à voir s'il est possible d'optimiser les requêtes SQL de l'application ou le hachage)
    - Le code souffre de plusieurs longueurs et a besoin d'être simplifié (répétition de code, fonctions mal séparées).

Note : nous n'avons pas pu utiliser Sonar sur le projet car le GitHub était un repository privé et était peu compatible avec le Framework Spring.

- Un autre problème du code est le suivant : il fonctionne beaucoup avec les identifiants de la base de données (genreId, paysId, souhaitId…) et surtout de manière explicite. Donc, si la base de données est déplacée (détruite et reconstruite) et que les identifiants sont mélangés, il en résultera un dysfonctionnement de l'application.
- Concernant l'évolution des fonctionnalités de l'application nous pouvons citer :
    - Le calcul automatique des tiers de distance pour pouvoir classer les dossiers
    - L'envoi des résultats sur la plateforme (c'est le sens de la table logement et type_appart de la base de données).
- Nous avons tenté d'implémenter DataTable sans succès, nous invitons à réessayer de l'implémenter à l'avenir
- Enfin des mesures de sécurité plus élaborées sont à rajouter (limite de taille pour les fichiers entrant par exemple, protocole https).
- Concernant l'import de bourse, un problème est découverte en fin de développement, si l'on coche oui que l'on importe sa notification de bourse, qu'on enregistre puis qu'on revient sur le form plus tard, la bourse n'est pas présente dans l'input (mais est bien présente sur le serveur), que l'on revient puis soumettons sans remettre la bourse, le fichier de bourse précédent est

écrasé. On a donc écrit un message disant de bien mettre sa bourse lors de la soumission.

VII] Conclusion

En conclusion, le projet fournit une application fonctionnelle et prête à être utilisée cette année pour la mission logement de juillet 2025.
Toutefois, des améliorations significatives sont à ajouter à l'application et nous recommandons donc de renvoyer l'application en projet pour l'année prochaine afin d'améliorer l'application.

VIII] Annexes

1. Planning final à jour

Mission Logement

Tâche	Priorité	Propriétaire	État	Date de début	Date de fin	Nombre d'heures
Programmer réunion de lancement	P3	Corentin P	Terminé	06/01/2025	06/01/2025	0,5
Réunion du groupe/Intégratiion des nouveaux membres	P3	Tous	Terminé	06/01/2025	06/01/2025	1
Réunion de lancement du PGROU	P3	Tous	Terminé	10/01/2025	10/01/2025	0,5
Rapport de suivi 1	P2	Samer A	Terminé	10/01/2025	10/01/2025	0,5
Réunion de rédaction du cahier des charges et de répartition des tâches	P3	Tous	Terminé	16/01/2025	16/01/2025	0,5
Rapport de suivi 2	P2	Quentin B	Terminé	16/01/2025	17/01/2025	0,5
Diagramme des cas d'utilisation	P3	Corentin P	Terminé	16/01/2025	19/01/2025	0,5
Rédaction du cahier des charges	P3	Corentin P	Terminé	16/01/2025	19/01/2025	2
Réunion d'avancement	P2	Tous	Terminé	24/01/2025	24/01/2025	1
Rapport de suivi 3	P2	Quentin B	Terminé	24/01/2025	24/01/2025	0,5
Réunion de lancement semaine de projet	P3	Tous	Terminé	27/01/2025	27/01/2025	0,5
Réunion d'avancement encadrant 1 et 2	P3	Tous	Terminé	27/01/2025	27/01/2025	0,5
Gestion token de première connexion	P2	Samer A	Terminé	27/01/2025	27/01/2025	7
Création des premiers identifiants	P2	Corentin P	Terminé	27/01/2025	27/01/2025	7
Développement de la partie paramétrage des assistants	P2	Lorenzo B	Terminé	27/01/2025	29/01/2025	3
Import des informations de SCEI	P2	Quentin B	Terminé	27/01/2025	29/01/2025	9
Export des formulaires	P2	Corentin P	Terminé	29/01/2025	29/01/2025	4
Réunion client bilan semaine de projet	P2	Tous	Terminé	31/01/2025	31/01/2025	0,5
Réunion encadrant bilan semaine de projet	P2	Tous	Terminé	31/01/2025	31/01/2025	0,5
Rapport de suivi 4	P2	Corentin P	Terminé	30/01/2025	31/01/2025	1,5
Développement de la sécurité de l'application	P3	Samer A	Terminé	27/01/2025	20/02/2025	12
Développement de la partie paramètres de l'admin	P3	Arman M	Terminé	27/01/2025	20/02/2025	8

Paramétrages textes et dates	P2	Arman M Quentin B	Terminé	27/01/2025	20/02/2025	6
Développement de la partie envoi du formulaire	P3	Lorenzo B	Terminé	29/01/2025	20/02/2025	8
Rapport de suivi 5	P2	Arman M	Terminé	07/02/2025	07/02/2025	0,5
Création de la page de consultation des dossiers et d'un dossier	P3	Quentin B Arman M	Terminé	17/02/2025	21/02/2025	3
Réinitialisation Mots de passes	P2	Samer A	Terminé	17/02/2025	21/02/2025	6
Rapport de suivi 6	P2	Corentin P	Terminé	14/02/2025	14/02/2025	0,5
Rapport de suivi 7	P2	Quentin B	Terminé	27/02/2025	28/02/2025	0,5
Gestion des mots de passe (users et admin)	P2	Quentin B et Samer	Terminé	17/02/2025	04/03/2025	6
Gestion des alertes	P3	Corentin P	Terminé	03/03/2025	04/03/2025	4
Développement de la partie d'attente de l'application	P2	Samer A	Terminé	03/03/2025	07/03/2025	4
Rapport de suivi 8	P2	Corentin P	Terminé	07/03/2025	07/03/2025	0,5
Gestion envoi du mail de première connexion	P2	Lorenzo B	Bloqué	27/01/2025	14/03/2025	15
Validation, refus, sauvegarde d'un dossier traité	P3	Corentin P	Terminé	17/02/2025	10/03/2025	6
Implémentation de Data Table	P1	Arman M	Bloqué	24/02/2025	14/03/2025	9,5
Supression des données	P2	Quentin B	Terminé	03/03/2025	10/03/2025	4
Création de la page des dossiers non transmis et de leurs valida	P2	Quentin B	Terminé	11/03/2025	13/03/2025	5
Réunion de test	P3	Tous	Terminé	10/03/2025	10/03/2025	1
Rapport de suivi 9	P2	Corentin P	Terminé	14/03/2025	14/03/2025	1
Mise en place de la premier connexion provisoire	P1	Lorenzo B	Terminé	11/03/2025	14/03/2025	2
Import d'une bourse	P2	Corentin P	Terminé	10/03/2025	19/03/2025	4
Export de document	P2	Corentin P	Terminé	10/03/2025	19/03/2025	4
Finission des fonctionnalités	P2	Tous	Terminé	10/03/2025	19/03/2025	3
Tests de validation de l'application	P3	Tous	Terminé	10/03/2025	19/03/2025	3
Rapport final	P3	Tous	Terminé	17/03/2025	21/03/2025	4
Livrables à rendre	P3	Tous	Terminé	E37 19/03/2025	20/03/2025	4
Rapport de suivi 10	P2	Quentin B	Terminé	24/02/2025	21/03/2025	0,5
Positionnement individuel compétences	P3	Tous	Terminé	21/03/2025	21/03/2025	1

2. Comptage des heures et des ressources

a. Comptage des heures

A l'issue du projet voici le décompte des heures (voir planning) :

Corentin Poulet : 55,5h
Samer Ayyoub : 56,5h
Quentin Bussard : 56h
Lorenzo Barbo : 49h
Arman Mola : 48h

b. Ressources et coûts du projet

Dans cette partie nous évaluons les coûts de ce projet :

Coût humain : Sur les deux projets 390,5h ont été consacré à l'application (à 3 puis 5) : ¼ homme.an
Soit 20000 € de coût humain

Coût matériel : Un serveur pour héberger l'application, un serveur Web type Tomcat, un hébergement de base de données :
1000€
L'application n'étant pas critique et un mode dégradé (le mode actuel) étant facilement réutilisable, aucun élément d'architecture supplémentaire n'est prévu.

Cette application serait toutefois adéquate à être déployée sur une architecture de Cloud public, avec un abonnement.

Ces coûts ne prennent en compte que les coûts de production et non les coûts de maintenance.

3. Bouts de codes

```
/**
 * Controller permettant d'afficher la page d'identification
 * @return Le ModelAndView lié à la page de connexion
 */
@RequestMapping(value="index.do")
public ModelAndView handleIndexGet(){
    Optional<ConfigModif> configInformationPopUpOpt = configModifRepository.findTopByTypeNomOrderByModifI
    ConfigModif configInformationPopUp = configInformationPopUpOpt.get();
    String texteInformationPopUp = configInformationPopUp.getContenu();

    ModelAndView returned = new ModelAndView("index");
    returned.addObject("textePopUp",texteInformationPopUp);

    return returned;
}
```

Figure 15 : Un méthode de controller type

```java
public static File getFileFromRequest(HttpServletRequest request, String value) {
    boolean isMultipart = ServletFileUpload.isMultipartContent(request);
    if (isMultipart) {
        MultipartHttpServletRequest request1 = (MultipartHttpServletRequest) request;
        MultipartFile temp1 = request1.getFile(value);
        System.out.println(temp1);
        File file = new File(pathwayFichier+"logement.tmp");
        if (temp1 != null) {
            try (
                OutputStream os = new FileOutputStream(file)) {
                os.write(temp1.getBytes());
            } catch (IOException ex) {
            }
        }
        if (request.getAttribute("MULTIPART") == null) {
            request.setAttribute(MULTIPART, new ArrayList<File>());
        }
        ArrayList<File> tempFiles = (ArrayList<File>) (request.getAttribute("MULTIPART"));
        tempFiles.add(file);
        return file;
    }
    return null;
}
```

Figure 16 : La méthode d'import de fichier bourse sur le serveur

```java
private static ResponseEntity<InputStreamResource> sendFile(String fileName, File theFile, MediaType media
    if ((fileName != null) && (!fileName.isEmpty()) && (theFile != null) && (mediaType != null)) {
        try {
            return ResponseEntity.ok()
                    .header(HttpHeaders.CONTENT_DISPOSITION, "attachment;filename=\"" + fileName + "\"")
                    .contentType(mediaType)
                    .body(new InputStreamResource(new FileInputStream(theFile)));
        } catch (FileNotFoundException ex) {
            Logger.getLogger(AdminController.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    return ResponseEntity.unprocessableEntity().body((InputStreamResource) null);
}
```

Figure 17 : La méthode de téléchargement de fichier sur client

```java
/**
* Mise à jour d'une personne
* @param id L'identifiant de la personne
* @param firstName Le prénom de la personne
* @param lastName Le nom de la personne
* @param Login Le login de la personne
* @param Password Le mot de passe de la personne
* @return La personne mise à jour si elle a réussie
*/
@Override
public Personne update(int id, String firstName, String lastName, String Login, String Password) {
    Personne item = null;
    if (id > 0) {
        item = personneRepository.getReferenceById(id);
    }
    if ((item != null) && (firstName != null) && (lastName != null)) {
        item.setPrenom(firstName);
        item.setNom(lastName);
        item.setLogin(Login);
        personneRepository.saveAndFlush(item);
    }
    return item;
}
```

Figure 18 : Une méthode de repository type