Rendu final

18 / 03 / 2024

Client
Mme Stéphanie BREUIL
Vie Étudiante de Centrale Nantes

Équipe Projet
Vianney de PONTHAUD
Elsa ARRIVE
Céline OURAOU
Zoé EYMARD

Encadrant Ecole
M. Jean-Yves MARTIN

CENTRALE NANTES

Sommaire

Présentation du projet

Le projet a été proposé par le service de Vie Étudiante de l'école Centrale Nantes. Chaque année, l'école reçoit de nouveaux élèves de première année issus des concours aux grandes écoles. Pour faciliter l'arrivée de ces élèves à Nantes, une partie d'entre eux peut profiter de la résidence étudiante présente sur le campus.

Jusqu'à aujourd'hui, un standard téléphonique était mis en place juste après la sortie des résultats de concours et les élèves devaient le contacter rapidement pour avoir la possibilité d'habiter dans la résidence. Cependant, les périodes d'appel sont courtes et si un élève n'est pas disponible, il ne pourra pas profiter de cet avantage.

La solution développée est donc avant tout à destination du service de Vie Étudiante dans le cadre de la Mission Logement, mais devra aussi être accessible via une interface aux élèves inscrits sur le SCEI (Service de Concours Écoles d'Ingénieurs) ayant un "oui définitif" validé la veille de la Mission Logement

Déroulement du projet

Comparaison organisation initiale et exécution

Lors de la phase de délimitation des bornes du projet, nous avions mis au point des dates clé et un découpage des tâches détaillé dans un GANTT (Annexe 5). Les dates clé sont les suivantes:

Janvier

- 26 janvier : rendu du cahier des charges. ***En pratique***, respecté

Février

- 19 février : objectif partie étudiants fonctionnelle. ***En pratique***, le 23/02/2023

Mars

- 4 mars : date limite de début des tests fonctionnels. ***En pratique***, les fonctionnalités côté administrateurs n'étaient pas toutes finies, nous n'avons pas pu lancer les tests généraux d'intégration et unitaires.
- 18 mars 2024 : rendu du rapport final. ***En pratique***, respecté
- 20 mars 2024 : transmission des livrables. ***En pratique***, respecté
- 22 mars 2024 : soutenance.

Communication au cours du projet

En ce qui concerne la communication dans notre groupe, nous avons su communiquer et nous entraider. Nous avons décidé de passer la grande majorité des heures de projet à travailler en présentiel à Centrale et le reste du temps nous avons communiqué sur une conversation messenger dédiée. Cependant, nous avions assez vite dévié du GANTT initial, avec des tâches qui demandaient plus de travail que prévu et des tâches qui impliquent de nombreuses modifications de structure, très

chronophages. Ainsi, mieux tenu à jour, nous aurions pu mieux utiliser le GANTT pour se tenir au courant des tâches en cours et faciliter la répartition des tâches.

Nous avons assuré un suivi régulier avec notre client, la Vie Étudiante. Nous avons bien tenu des réunions d'avancement de projet régulièrement (2 à 3 semaines) pour avoir des retours sur nos progrès et que Madame Breuil sache où nous en étions.

Bien que nous ayons développé une bonne partie des fonctionnalités requises, en l'absence de tests unitaires et de charge, nous ne pourrons pas déployer le logiciel pour la mission logement 2024 comme espéré en début du projet.

Réflexion sur des axes d'amélioration

Nous sommes assez fiers du travail que nous avons fourni, cependant, avec du recul, nous aurions pu faire une organisation légèrement différente. En effet, nous avons essayé de développer pour l'application toutes les fonctionnalités en parallèle. Or, si nous avions isolé les fonctionnalités prioritaires dès le début, et fait un meilleur découpage du projet en phases précises, nous aurions peut-être pu finir une version minimaliste du logiciel (contenant le recueil des souhaits des élèves et la gestion administrative minimale) déployable à la fin du projet.

Travail effectué

Ici sont présentées les différentes fonctionnalités du logiciel à ce jour. Plus de détails sur l'utilisation de chacune sont données dans les livrets à l'intention des différents intervenants.

1- Fonctionnalités administrateur et ses assistants

Fonctionnalités uniquement pour les administrateurs

Ces fonctionnalités sont disponibles dans l'onglet "Gestion admin" côté administrateur de l'application. Cet onglet n'est pas accessible par les personnes ayant un rôle d'assistant.

Modification des dates

L'ajout des dates nous a permis de gérer l'affichage des pages côté étudiants. Ainsi, lorsque la mission est ouverte, l'étudiant à accès au questionnaire. Si la mission est clôturée mais les résultats pas encore disponibles, il aura une page d'attente et si les résultats sont disponibles il est renvoyé vers la page où il peut consulter son affectation (colocation, studio ou sans appartement).

Côté front :

Comme détaillé dans le mode d'emploi administrateur, l'administrateur a la main sur les dates liées à la mission logement. Cela lui permet de gérer les dates d'ouverture de la mission logement, fermeture du dépôt des demandes et de dévoilement des résultats.

Côté back :

Pour la gestion des dates, nous avons créé une table date dans la base de données, l'année de chaque session faisant office d'identifiant. À chaque année (chaque session de la mission logement) correspondent les trois dates citées ci-dessus. Lorsqu'elles sont modifiées dans l'application, la base de données est mise à jour. La concordance de la chronologie est bien vérifiée lors de la sauvegarde (la clôture ne peut pas avoir lieu avant l'ouverture par exemple).

Gestion des assistants

Afin de répondre à l'exigence de restriction de l'accès à certaines fonctionnalités administrateurs aux assistants, nous avons créé une fonctionnalité gestion des assistants. En effet, une personne connectée à un compte assistant n'aura pas accès à l'onglet gestion admin où sont accessibles les fonctions réservées aux administrateurs.

Côté front:

Comme détaillé dans le mode d'emploi administrateur, il est possible de créer des comptes assistants pour les étudiants qui vont participer à la mission logement. Cette page se présente sous la forme d'une liste des assistants, l'administrateur a la main pour :

- Ajouter un assistant
- Modifier un assistant (changer un mot de passe si nécessaire)
- Supprimer un assistant

Pour la liste, il faut faire le lien avec la base de données, nous nous sommes basés sur la page d'affichage de la liste des étudiants que nous avions déjà créée.

Pour assurer que seuls les utilisateurs ayant le rôle d'administrateur aient accès à l'onglet "Gestion admin" sur la page d'accueil admin, nous avons mis en place une vérification du rôle directement dans le code HTML de la page. On vérifie le rôle de l'utilisateur connecté. Si cet utilisateur a le rôle "Admin", alors le bouton pour accéder à la gestion admin est affiché dans le bandeau supérieur de l'application. Sinon, le bouton n'est pas affiché, limitant ainsi l'accès uniquement aux utilisateurs ayant les droits appropriés.

Côté back:

Gérer les assistant a demandé l'ajout d'un nouveau rôle pour les personnes donc des modifications notamment dans les classes Role.java (item), et ApplicationInitializer.java pour créer le rôle dans la base de données dès l'initialisation de la base de données.

Nous avons fait face à un problème d'affichage, lorsqu'on affichait les assistants, leur rôle affiché était élève, pourtant, ils avaient bien les droits d'assistant et étaient enregistrés comme tel dans la base de données. L'erreur provenait d'une inversion de l'ordre des rôles dans la base de données, lors de l'initialisation de l'application.

Suppression de la base de données

À la fin d'une session, il faut supprimer les données (c'est important légalement parlant mais aussi pour le fonctionnement de l'application pour les futures sessions). Les données supprimées sont les connexions, étudiants et administrateurs (et les personnes associées), et les logements

.

Côté front:

Pour la gestion de la suppression de la base de données, nous avons mis en place beaucoup de sécurité pour que cela ne soit pas effectué par erreur par un administrateur. Toutes les étapes sont détaillées dans le mode d'emploi administrateur.

Côté back:

Pour supprimer les données proprement, il a fallu faire les suppressions des données dans les tables dans un ordre correct (un étudiant étant une personne on ne peut pas supprimer la personne avant l'étudiant par exemple). De plus, nous avons fait attention à conserver la personne ayant le rôle

administrateur dans la suppression des données, sinon elle n'aurait plus accès à l'application ce qui serait problématique.

Initialisation de la base des données gérant les communes

Pour le bon fonctionnement de l'application, il est nécessaire de pouvoir calculer la distance d'une ville à Nantes, cela permet de classer les élèves par distance pour l'affectation des logements. Nous avons donc besoin de la longitude et latitude des villes, ce que nous avons trouvé sur un site officiel. Nous avons donc mis en place l'import de ces informations dans le logiciel. Le fichier se trouve dans le GIT et sera parmi les documents.

Dans un premier temps, nous souhaitions le faire à chaque initialisation de l'application, puis nous nous sommes vite rendu compte que cela prenait trop de temps à chaque démarrage. Ainsi, nous avons décidé de faire un unique import. Il suffira à l'administrateur d'appuyer sur un bouton (comme expliqué dans la notice administrateur) pour tout importer une unique fois.

Import du fichier des logements

Afin de répartir correctement les étudiants dans les logements, il était nécessaire d'importer les logements, c'est-à-dire, leur nom, leur type (colocation, studio ou PMR), le genre et le nombre de places disponibles si c'est une colocation avec déjà un ou une étudiante affectée. On utilise un fichier type que le gestionnaire de la résidence fournit à la vie étudiante pour l'import.

Côté front

On permet à l'administrateur de téléverser le fichier dans une page dédiée aux imports. Dans la notice il est précisé le format d'encodage nécessaire à la réussite de l'import. En effet, il faut absolument que le fichier soit encodé avec le format UTF-8, sinon cela cause des problèmes avec les accents et caractères spéciaux.

Le fichier est ensuite sauvegardé dans l'application (dans le dossier target, un dossier FichierRez est créé pour le contenir) et les données dans la base de données. L'utilisateur peut vérifier la réussite en regardant la liste des logements.

Côté back:

Afin de pouvoir associer un type de logement aux logements, il a fallu les initialiser. Nous nous sommes basés sur la structure utilisée pour rôle (en rapport avec les personnes) pour initialiser les types d'appartements. On ajoute les types dans l'item et on les crée dans l'initialiseur de l'application.

Ensuite, lorsqu'on reçoit le fichier csv, on le sauvegarde et on lit les lignes pour stocker les logements dans la base de données. Pour avoir un calcul correct du nombre de places restantes, il faut faire attention à stocker le nombre initial de place dans un appartement, et de mettre à jour le nombre de place lorsqu'il y a des étudiants (internationaux, bachelors) qui sont déjà affectés à cet appartement. Ces derniers sont eux aussi enregistrés dans la base de données mais avec un numéro SCEI égal à -1 pour qu'ils ne fassent pas partie du tri lors de l'attribution des logements. N'ayant pas leurs dates de naissance pour les identifier s'il y a des homonymes, on regarde le triplet nom, prénom, mail. Cela sera utile pour l'export final pour la résidence, il y aura un fichier avec l'intégralité des membres de la résidence, pas uniquement ceux qui ont été acceptés par SCEI.

Pour affecter à l'appartement le bon type, on se base sur le nombre de places dans l'appartement.

- Égal à 1 c'est un studio

- Sinon c’est une colocation

L'exception concerne les logements type PMR. La mention PMR , dans les fichiers reçus de la résidence, se situe dans la colonne prénom. Donc on regarde aussi la colonne prénom pour affecter le bon type.

Pour éviter des problèmes de nombre de place dans les appartements, il faut importer le fichier de la résidence une unique fois. Ainsi, si l’administrateur importe un nouveau fichier, les données liées à l’import précédent (élèves pré-inscrits et logements) sont automatiquement supprimées. L’administrateur en est prévenu dans le mode d’emploi et par une fenêtre de confirmation qui s’affiche.

Import des fichiers SCEI

Pour le logiciel, il a été choisi de créer des accès au formulaire uniquement pour les étudiants qui auraient validé le choix Centrale Nantes sur SCEI. Ainsi, il est nécessaire que l’administrateur puisse importer la liste des étudiants admis qui lui est transmis par l’administration.

Côté front:

Tout comme le fichier des logements, il est possible de téléverser le fichier dans la partie import de l’administration. Cela l’enregistre dans l’application, dans le dossier *target/ECNLogement-1.0/FichierSCEI,* puis le traite pour enregistrer les données dans la base. Pour que les caractères spéciaux soient correctement traités, il est nécessaire de téléverser un fichier csv encodé en UTF-8, comme spécifié dans la notice administrateur. Une fois le fichier importé, l’utilisateur peut vérifier l’import car il est redirigé vers la liste des inscrits.

Côté back:

On lit des fichiers type avec des en-têtes prédéfinis (fichier type fourni dans la doc). Puis ligne par ligne, on importe les étudiants, en les mettant en non validé par défaut. On va chercher dans la base de données des communes pour récupérer le code commune associé.

La vie étudiante voulait être capable de faire plusieurs imports pour ouvrir la mission le plus tôt possible. Ainsi, il est possible d’importer plusieurs fichiers sans écraser les données, on enregistrera uniquement les étudiants non présents dans la base de données.

Gestion de l’affichage

Pour permettre la pérennité de l'application web, il a fallu réfléchir aux textes à afficher du côté élève. Après discussion avec le client, nous avons convenu qu’il fallait pouvoir permettre à la Vie Étudiante de modifier les textes principaux du formulaire. Par conséquent, nous avons implémenté cette fonctionnalité dans l’application.

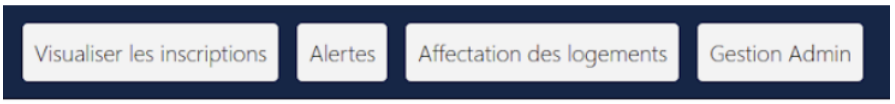
Côté front :

Un bouton dans l’onglet “Gestion Admin” permet d’accéder à la page de modification des textes. L’administrateur doit seulement choisir quel texte il souhaite modifier et remplacer le texte par le nouveau. Il n’est cependant pas possible de créer un nouveau texte !

Côté back :

Nous avons simplement ajouté la table *Texte* à la base de données. Cette table est très simple. L’identifiant est le nom du texte en question et son seul attribut le contenu du texte.

Fonctionnalités pour les assistants et administrateurs



Gestion des inscriptions

Le bouton “Gestion des inscriptions” permet de visualiser la liste des élèves importés (c’est-à-dire les élèves ayant répondu “Oui définitif” sur SCEI. Les élèves importés sont par défaut en “non confirmé” et n’ont pas d’identifiant (seules les informations fournies par SCEI sont remplies). Lorsqu’un élève crée son compte sur notre site, son identifiant est rempli. Lorsqu’il valide son formulaire de demande de pré-réservation de logement, ses informations remplies sont mises à jour et il est mis automatiquement en “confirmé”, sauf s’il est dans un cas ayant besoin d’une confirmation admin.

Côté Front

Depuis la page de visualisation de la liste des élèves, on peut :

- Afficher les informations complètes de l’élève via le bouton “I” bleu
- Modifier ou compléter les informations de l’élève via le bouton crayon
- Supprimer l’élève via le bouton poubelle
- Ajouter un élève via le bouton “+”

On peut trier la liste selon chaque critère, ainsi que faire des recherches via la barre de recherche.

Côté Back

La modification de l’élève passe par la méthode EleveSaveAdmin.do du contrôleur EleveController. Elle permet notamment, en plus de sauvegarder les mêmes éléments que la méthode EleveSave.do, de sauvegarder la confirmation de l’élève et les informations supplémentaires ajoutées par les admins.

Gestion des alertes

Nous n’avons pas eu le temps de développer cette fonctionnalité. Elle sera donc mise en place par ceux qui reprendront ce projet.

2- Fonctionnalités étudiant

Création d'un compte

La première fois que l'élève accède au site, il doit pouvoir se créer un compte puis s'y reconnecter ensuite. Pour cela, on lui crée un identifiant, et il se choisit un mot de passe.

Côté Front
Lors de sa première connexion, on fait rentrer à l'élève son nom, prénom et numéro SCEI. On le renvoie ensuite vers une page pour se créer un compte : on lui attribue un identifiant (l'élève doit bien noter cet identifiant car il ne peut pas le changer ensuite), et on lui demande de choisir un mot de passe et de le confirmer. Pour faire cela, nous avons mis deux champs d'entrée de texte de type "password", ce qui permet de mettre le mot de passe tapé en "points", avec une icône "œil" pour voir les lettres tapées. Pour éviter les erreurs, nous avons rajouté une contrainte d'égalité entre les deux mots de passe tapés pour que l'élève puisse valider son mot de passe choisi seulement s'il a écrit deux fois la même chose.

Côté Back
Après la page de première connexion, on vérifie si l'élève est bien dans notre base de données établie à partir des fichiers d'imports SCEI (grâce à son nom, prénom et numéro SCEI).
S'il ne l'est pas, on le renvoie sur une page lui expliquant qu'il n'est pas encore marqué en "Oui définitif", ce qui est nécessaire pour se créer un compte.
S'il l'est, on lui crée un identifiant unique qui correspond à son numéro SCEI concaténé à son personne_id dans notre base de données.
Pour conserver son mot de passe dans notre base de données, on a utilisé une fonction de hachage encryptPassword et lorsque l'élève se reconnecte, on vérifie que le hachage du mot de passe qu'il rentre correspond au hachage dans notre base de données avec la fonction checkPassword.

Formulaire de demande de pré-réservation

Une fois connecté, l'élève doit remplir le formulaire de demande de pré-réservation d'un logement, et on doit enregistrer ces informations dans notre base de données.

Côté Front
Tous les élèves doivent au moins remplir les informations suivantes : date de naissance, pays, mail, téléphone, ville, boursier ou non, genre, souhait de logement.
Selon les cas, nous avons affiché les informations de manière un peu différente :

- Si l'élève est français, un champ supplémentaire pour le code postal apparaît. De plus, le champ pour la ville est un menu déroulant pour qu'il choisisse sa ville. Cela permet d'être sûr de reconnaître le nom de la ville de l'élève qui est nécessaire pour les français pour faire le classement de distance.
- Si l'élève est international, le champ pour sa ville est un champ de texte libre, car seules les communes françaises sont déjà initialisées dans notre base de données.
- Si l'élève est boursier, un bouton s'affiche pour qu'il puisse téléverser un pdf depuis ses fichiers (étape obligatoire s'il a coché oui pour la bourse).

De plus, une case "informations supplémentaires" est disponible à la fin si l'élève souhaite apporter des précisions (notamment les élèves PMR sont incités à le préciser à cet endroit)

Pour éviter les erreurs, nous avons précisé les types de champs dans le code de la page html, par exemple :

- Le champ pour le mail accepte une entrée du type …@...

- Le champ pour le numéro de téléphone accepte une entrée composée d'entre 10 et 13 caractères pouvant être des chiffres ou des + (on accepte donc à la fois les numéros français et les internationaux)
- Le champ pour le pays est un menu déroulant
- Pour la bourse, le genre et le souhait de logement, nous avons mis des boutons radios (réponses prédéfinies dont seulement une est sélectionnable)

Côté Back

L'enregistrement des données remplies par l'élève se fait via la méthode EleveSave.do du contrôleur EleveController.
On commence par exporter la notification de bourse potentiellement téléversée par l'élève. On vérifie d'abord que le répertoire où l'on va la placer est créé, sinon on le crée.
On récupère le fichier à partir de la requête et on lui donne un nom unique (pour ne pas qu'un fichier soit écrasé par un autre dans le répertoire) en le nommant Nom_Prénom_bourse_x, x étant un nombre aléatoire. On vérifie que l'élève est boursier (pour ne pas créer de fichier vide dans le répertoire d'arrivée), s'il est bien boursier, on copie le fichier dans le répertoire.
Pour mettre les informations entrées par l'élève dans la base de données, on crée une instance provisoire d'Eleve, on utilise les fonctions set pour lui donner en attribut toutes les informations entrées par l'élève, puis on "update" l'élève de la base de données par l'élève provisoire. Lorsque l'on récupère des requêtes des chaînes de caractères qui doivent être reconnues, comme le pays, on utilise une fonction qui remplace tous les caractères spéciaux par des caractères normaux et qui met le tout en majuscules.

Page de confirmation et modification d'information

Une fois le formulaire rempli, l'élève est redirigé vers une page de confirmation.
Si un élève ayant rempli le formulaire se reconnecte, il sera dirigé vers une autre version du questionnaire, où toutes les informations le concernant sont affichées mais seules certaines informations peuvent être modifiées : son caractère boursier (et le téléversement de sa notification de bourse le cas échéant), son souhait de logement, et ses informations supplémentaires facultatives.
Lorsqu'il valide, il est à nouveau dirigé vers la page de confirmation.

Temps de travail et estimation des coûts

Estimation du volume horaire de chacun :

	Céline	Elsa	Vianney	Zoé	Total
Nombre d'heure	72	75	90	35	272

En prenant comme valeur, 450€/jour (ETP : 7h/jr), on peut estimer grossièrement le coût du projet à : 17 500€

Difficultés rencontrées

Estimation du temps de travail

Tout au long du projet nous avons rencontré diverses difficultés, pas toutes liées directement au code. Tout d'abord, la gestion de projet a été délicate. En effet, ayant peu d'expérience en développement, il nous a été difficile d'estimer correctement le temps pour développer chacune des fonctionnalités, et c'est ce qui nous a empêché d'avoir un logiciel opérationnel aujourd'hui. Il aurait peut-être fallu définir plusieurs versions, en testant à chaque version obtenue. Nous aurions ainsi pu développer uniquement la partie formulaire pour la pré-réservation et le classement des étudiants en laissant les affectations à la Vie Étudiante (en manuel).

Multiplication des bugs

Au début, nous pensions que tout ce que voulait le client était réalisable dans le temps imparti, mais plus nous avancions dans le projet, plus nous avons eu des difficultés et plus les bugs engendrés étaient difficiles à résoudre. Ainsi, un bug minime pouvait être très chronophage puisqu'il faut prendre en compte tous les fichiers concernés. De plus, lors de la résolution d'un problème, on en découvre un autre et ainsi de suite. Tous les petits défauts cachés font leur apparition et demandent une correction.

Utilisation du Git

Un autre point de difficultés rencontrées a été la gestion du git. Pour pouvoir utiliser Sonar, il a fallu modifier la structure des fichiers dans le git, ce qui a entraîné une suppression du travail des autres lorsqu'il a fallu *pull* la version partagée..

Pour résumer, certaines difficultés sont d'ordre informatique (gestion du git et bugs) mais la plupart ne le sont pas. Elles sont plutôt dûes à des manques dans la gestion de projet et la communication (avec le client et entre les membres du projet).

Suite du projet, perspectives futures

Si l'application n'est pas achevée à la fin de ce projet, la poursuite du projet est plutôt claire, et la suite du développement aussi. Dans la *Notice développeur*, nous avons explicité ce qui a été fait, ce qui doit être vérifié, et ce qui devra être fait.

Une *Notice projet étudiant* a été constituée pour aider le prochain groupe de projet à reprendre notre travail. Elle est à utiliser conjointement à la *Notice développeur*.

Conclusion

Pour conclure, ce projet a été pour nous l'occasion d'apprendre la programmation Web en profondeur et de vivre le développement d'un projet en contact avec le client. Nous avons eu une bonne communication avec ce dernier. La leçon principale que nous retenons de ce projet de groupe est le fait qu'il faut être extrêmement prudent sur l'application que l'on vend. En effet, même une fonctionnalité basique peut-être longue à mettre en place, et mieux vaut une application restreinte mais fonctionnelle qu'une application complexe mais bancale.

Livrables

Livrable 1 : Cahier des charges

Dans le dossier

Livrable 2 : Mode d'emploi Administrateur

Dans le dossier *Livrables*

Il s'agit du mode d'emploi pour guider l'administrateur dans l'utilisation de l'application. Toutes les consignes de mise en place de l'application et ses fonctionnalités sont explicitées dans ce document.

Livrable 3 : Mode d'emploi étudiant

Dans le dossier *Livrables*

Le mode d'emploi étudiant est le guide d'utilisation de l'application pour l'étudiant cherchant à réserver un logement. Ce guide est destiné au client, pour qu'il possède le processus d'inscription d'un élève et ses possibilités.

Livrable 4 : Notice développeur

Dans le dossier *Livrables*

Cette notice contient les informations complémentaires à la Javadoc de l'application. Elle contient notamment la structure de la base de données et des commentaires sur les différentes tables et attributs.

Livrable 5 : Notice projet étudiant

Dans le dossier *Livrables*

Nous avons écrit cette notice à destination des étudiants qui reprendront ce projet. Elle contient la liste des outils nécessaires au développement, des explications sur les méthodes de développement Web. Il vient compléter la Notice développeur pour les étudiants.