RAPPORT D'ALTERNANCE 2023-2024

Développeur Full stack

Licence Informatique Générale

Et

Bachelor informatique - Parcours Ingénierie logicielle et DevOps

LABARCHE Nathan

Tuteurs de la formation : Desraux Ludovic & De Pellgrin Alexandre
Coordinatrice Pédagogique ESIEE-IT : Mme Aude FAUGUET
Coordinatrice Formation CNAM : Mme Sandrine BERTOUX

Alexandre De Pellgrin – Tuteur (DSI)	**Ludovic Desraux - Tuteur (Chef de projet)**

Table des matières

Remerciements

Tout d’abord, je tiens à exprimer ma profonde gratitude envers mon employeur, l’ENSEA, ainsi que mon établissement ESIEE-IT, pour m'avoir offert cette opportunité d'alternance enrichissante. Je souhaite adresser mes remerciements les plus sincères à l'équipe du SRI de l’ENSEA, avec laquelle j'ai eu le privilège de travailler tout au long de cette année. Je tiens particulièrement à exprimer ma reconnaissance envers Monsieur Desraux et Monsieur De Pellgrin pour l’opportunité de travaillé sur projet aussi complet. Enfin, je souhaite remercier chaleureusement toutes les personnes qui prendront le temps de lire ce rapport. Je vous souhaite une agréable lecture.

Introduction

Ce document d'apprentissage retrace mon parcours professionnel durant cette année passée, une période caractérisée par une évolution constante. Au début, n'ayant pas trouvé de contrat d'alternance avant le début de ma formation, j'ai été soutenu activement par l'ESIEE-IT à travers plusieurs ateliers consacrés à la recherche d'emploi en alternance.

Cette démarche s'est révélée fructueuse, ce qui m'a permis de commencer mon apprentissage à l'ENSEA le 16 octobre 2023. Cette transition a été unique car, en tant qu'étudiant en école d'ingénieur, j'ai été confronté à une situation différente en intégrant l'aspect professionnel d'une école supérieur.

De cette manière, j'ai pu explorer l'organisation interne d'une prestigieuse école, qui présente des similitudes étonnantes avec une entreprise privée. Effectivement, l'ENSEA possède tous les services indispensables pour assurer le bon déroulement d'une entreprise, tels qu'un Service des Ressources Informatiques qui comprend une équipe spécialisée dans le support informatique, des chefs de projet, des administrateurs réseaux et bien sûr, un responsable du service informatique supervisant l'ensemble des activités au sein du service.

C'est dans ce département que j'ai acquis une expérience en tant que développeur, avec pour principale responsabilité la création d'une application web interne. L'objectif de cette application est de rendre la création de comptes utilisateurs plus facile et automatisée, tout en gérant les accès associés en fonction des rôles définis. Plus précisément, lorsqu'on crée un compte sur cette application, une série requête api d'API est lancé afin d'accéder aux divers outils et services requis, comme 365 ou Galactus.

Dans ce rapport, nous commencerons par une analyse du fonctionnement et de l'organisation de l'ENSEA, afin de mieux appréhender le contexte dans lequel s'inscrit mon expérience. Nous explorerons ensuite le processus complet de création de l'application Infocentre, depuis sa conception jusqu'à son déploiement. Nous examinerons les objectifs initiaux, les changements qui ont eu lieu au cours du développement, ainsi que l'organisation mise en œuvre pour réaliser ce projet. Nous nous pencherons aussi sur les problèmes rencontrés cette année, ainsi que la relation entre ma formation et les tâches accomplies au sein de l'entreprise.

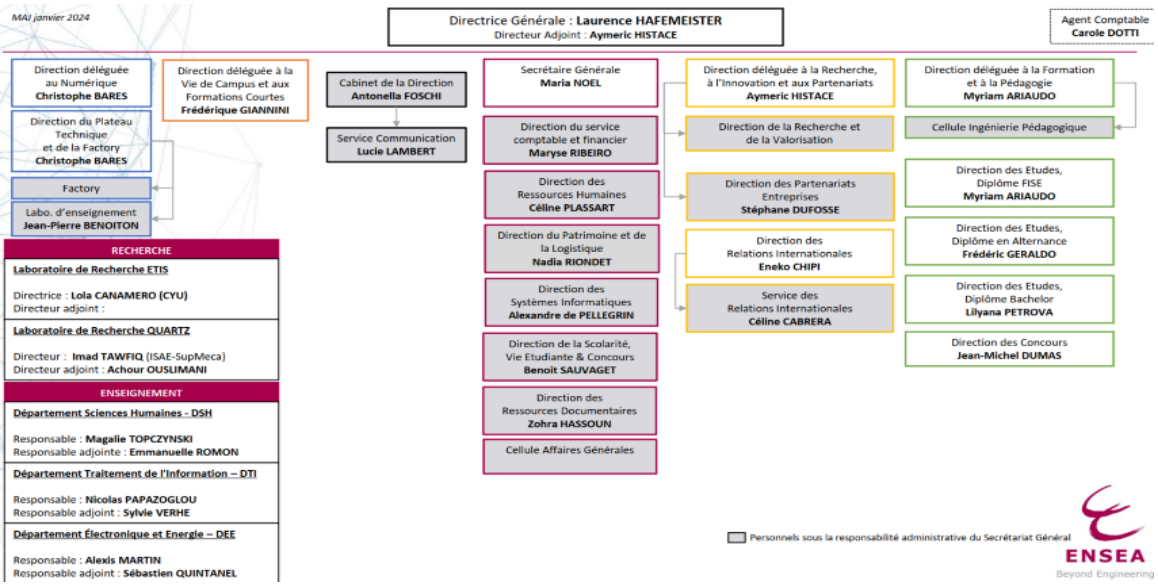
Enfin, une conclusion permettra de dresser un bilan de cette année d'alternance, en mettant en lumière les aspects positifs et négatifs de cette expérience, tout en identifiant les axes d'amélioration pour l'avenir.

Présentation de l’organisation

L'ENSEA, École nationale supérieure de l'électronique et de ses applications, est un établissement de premier plan de l'enseignement supérieur français dans les domaines de l'électronique, de l'informatique et des télécommunications. L'ENSEA a été fondée en 1952 à Paris dans le cadre de l'essor de l'électronique et des télécommunications en France et à l'échelle mondiale. L'objectif initial de l'école était de former des ingénieurs de premier ordre afin de répondre aux besoins grandissants de l'industrie électronique en plein essor.

L'ENSEA s'est imposée au fil des décennies comme un établissement de premier plan, attirant des étudiants et des chercheurs venant du monde entier. Grâce à son dévouement envers l'innovation et la recherche de pointe, l'école a réussi à se classer en tête des écoles d'ingénieurs. De nos jours, l'ENSEA poursuit sa mission de formation des futurs ingénieurs et chercheurs, tout en jouant un rôle actif dans l'évolution des connaissances et des technologies dans ses domaines d'expertise. Elle incarne une vision audacieuse et un engagement sans faille envers l'excellence académique, l'innovation technologique et la responsabilité sociale et environnementale. Fondée sur des valeurs fondamentales telles que l'excellence, l'ouverture, l'engagement, la cohésion et l'audace, l’ENSEA s'efforce de former une nouvelle génération d'ingénieurs et de chercheurs capables d'élaborer des solutions technologiques responsables et de haute qualité, tout en répondant aux défis socio-économiques et environnementaux actuels.

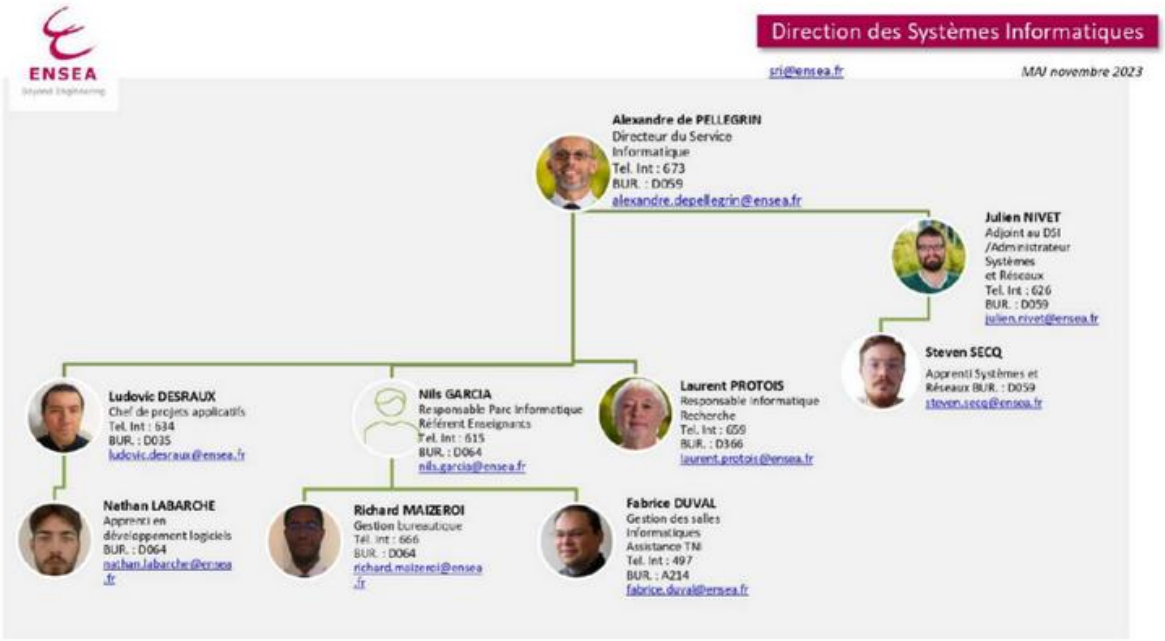
L'objectif principal de la direction de l'ENSEA est de maintenir ces valeurs et objectifs qui sont à la base de l'excellence de l'établissement. En particulier, cette clé essentielle des grandes entreprises, l'intégration. Effectivement, l'établissement est dirigé par Madame Laurence HAFEMEISTER depuis 2014. C'est la première femme à diriger l'école et elle sera représentante de l'école pendant deux mandats successifs. La priorité de la direction a toujours été d'accepter tout le monde sans distinction. Voici comment s’organise la direction de l’établissement :



J'ai eu la chance d'intégrer le Service des Ressources Informatiques (SRI) de l'ENSEA, un cadre prestigieux. Le SRI joue un rôle clé dans l'infrastructure de l'établissement scolaire, avec pour mission de fournir un soutien technique et une expertise dans le domaine des systèmes d'information et de communication. Grâce à son savoir-faire et à son expérience, le SRI occupe une position stratégique dans la gestion et l'amélioration des ressources informatiques de l'établissement, tout en garantissant le bon déroulement de ses activités académiques et administratives.

Le SRI est constitué d'une équipe pluridisciplinaire composée de personnes compétentes dans différents domaines de l'informatique, tels que le support technique, la gestion de projets, la sécurité informatique et la gestion des réseaux. Encadré par le Directeur des Systèmes d'Information (DSI), le SRI s'engage à proposer des solutions novatrices et fiables afin de satisfaire les exigences particulières de l'ENSEA, tout en assurant la sécurité et l'intégrité des données institutionnelles.

En tant que développeur au sein du SRI, j'ai été plongée dans un cadre dynamique et stimulant, où l'innovation et la collaboration sont au centre de chaque projet. Grâce à ma collaboration étroite avec mes collaborateurs et les divers acteurs de l'école, j'ai pu pleinement comprendre les défis et les possibilités associés à la gestion des systèmes d'information dans un environnement académique exigeant. Voici l'organigramme du SRI :



Cette section introductive se conclut sur une transition fluide vers la description détaillée des missions que j'ai accomplies au sein du SRI, mettant en lumière leur importance stratégique dans le cadre des activités de l'école et leur contribution à l'atteinte des objectifs organisationnels.

Contexte

L'ENSEA a lancé le projet « Infocentre » afin de satisfaire un besoin grandissant de modernisation de son système de gestion des comptes utilisateurs. La précédente application « Gescomptes », qui était devenue obsolète et remplie de fonctionnalités obsolètes, ne satisfaisait plus aux exigences actuelles en matière de technologie et de sécurité.

Ce système devait être remplacé par une solution plus solide, performante et sécurisée, capable de gérer les comptes de différents utilisateurs : étudiants, enseignants, vacataires et autres intervenants. L'infocentre présente un avantage par rapport à son prédécesseur, à savoir l'automatisation et la création autonomes de chaque accès aux outils externes.

Lors de nos discussions pour la création de l'application, j'ai aussi voulu rendre l'application plus agréable et plus ergonomique pour les utilisateurs, sans pour autant surcharger les feuilles de styles. Nous avons essayé de trouver un bon équilibre afin que le prochain développeur souhaitant faire des modifications puisse en faire sans être contraint de passer des heures sur l'apparence, tout en souhaitant rendre l'expérience utilisateur plus agréable que la solution existante.

Objectifs du Projet

Développement d'une Application Web Moderne :

1. **Élaboration et Réalisation** : Créer "Infocentre", une application web contemporaine utilisant le Framework Django, afin d'optimiser la gestion des comptes utilisateurs. Il est essentiel que l'application soit intuitive, sécurisée et facile à utiliser.
2. **Caractéristiques :** Création, modification et suppression de compte effectuées de manière simplifiée. Appel API pour la mise en place d'accès externes à l'application intégrée.
3. **Automatisation des Processus** : Réduire les délais et les erreurs liés aux processus manuels en automatisant la création des comptes et l'accès aux applications telles que 365 ou Galactus. Une API spécialement développée pour l'application collecte les informations d'un compte choisi pour les transmettre aux API des outils externes. De cette manière, il est possible que la création des accès soit autonome par rapport à son application. Par exemple, en cas de difficultés lors de la création du compte Galactus, le compte 365 pourra néanmoins se créer.
4. **Introduction à Django et Python** : Utiliser ce projet comme une opportunité pour introduire et familiariser l'équipe du Service des Ressources Informatiques (SRI) avec le Framework Django et le langage de programmation Python, alignant ainsi les compétences techniques de l'équipe sur les technologies modernes et efficaces.
5. **Amélioration de l'Infrastructure Technologique** : Fournir une infrastructure technologique mise à jour qui soutient l'efficacité opérationnelle et renforce la sécurité des données utilisateurs.

   Le projet visait non seulement à améliorer les services informatiques offerts par l'ENSEA mais également à enrichir l'expertise technique interne en adoptant des outils de développement modernes et des pratiques de programmation avancées.

La conception

La conception d'Infocentre est passé par plusieurs étapes en commençant par le fait de comprendre qui seront les utilisateurs. Cette partie a été faites lorsque je n'étais pas encore membre de l'équipe. Les technologies ont aussi été déterminées avant mon arrivée. Le SRI avait décidé d'utiliser je Framework Django du langage python pour la création de cette application ainsi que d'utiliser l'annuaire LDAP déjà existant dans l'écosystème de l'école pour la récupération des données.

Par conséquent, j'ai décidé après mon auto-formation de proposé des diagrammes de conception afin de valider auprès de mes tuteurs l'objectif et la logique de l'application. Ensuite, j'ai proposé quelques idées concernant le style graphique ainsi que l'affichage des données. Nous verrons plus tard que nous sommes loin de l'application finale à la suite de plusieurs changement opérés lors de son développement. J'ai utilisé plusieurs applications pour la partie conceptuelle de l'application comme Looping ou encore Figma. Les notions que j'ai vu en Ux Design ou encore dans nos cours de base de données conceptuels m'ont permis de mieux comprendre quels sont les objectifs de la conception avant même le développement de la solution.

Enfin le déploiement de l'application est fait en collaboration Alexandre De Pellgrin (DSI) afin de conteneuriser l'application via l'outils Docker permettant ainsi à l'application d'être déployable facilement.

Organisation

Au cours de la réalisation de mon projet, j'ai eu l'opportunité de travailler de manière autonome, ce qui m'a permis de bénéficier d'une grande flexibilité dans l'organisation et la gestion de mon temps.

En tant que seul membre de l'équipe, j'ai pu directement contrôler toutes les étapes du projet, depuis l'auto-formation sur Django et LDAP jusqu'à la mise en œuvre de fonctionnalités complexes comme la modification des comptes utilisateurs dans la base de données et l'intégration avec le système CAS SSO.

Cette autonomie m'a permis de développer une compréhension approfondie et multifacette du projet, en prenant des décisions rapides et en ajustant les objectifs et les délais en fonction des besoins réels du projet, comme illustré dans le planning fourni. Ce mode de travail indépendant a non seulement accéléré le processus de développement, mais a également renforcé mes compétences en gestion de projet et en résolution de problèmes.

Bien évidemment, cette organisation n'est pas forcément la plus optimale car en étant seul sur un projet, on peut manquer à certains moment un manque de recul sur notre projet. C'est pour cela que chaque semaine, nous faisions un point avec mes tuteurs quant à l'état du projet et de la mission en cours. Cela nous a aussi permis de comprendre les obstacles futurs et de réadapter le projet pour qu'il soit le plus clair possible.

Vous trouverez ci-dessous mon planning de travail tout au long de ce projet. Certaines taches ont été ajouté au fur et à mesure lorsqu'une correction était nécessaire.

La réalisation

Le premier objectif quant à la réalisation du projet était d'être à l'aise sur la technologie. Mon expérience en python m'as permis de ne pas perde de temps sur le langage en lui-même. Cependant je me devais de me former sur le Framework Django afin de comprendre l'étendu des possibilités qui m'était proposé. J'ai donc commencé mon apprentissage en développant mes compétences grâce à plusieurs à plusieurs documentations comme OpenClassroom qui propose plusieurs exercice Django pour les initiés à python. J'ai aussi appris des aspects plus techniques via la documentation Django officiel. Enfin mon auto-formation Django pour la réalisation du projet a surtout été accompagné par des tutos YouTube tel que celui de Traversy Media, tuto complet montrant plusieurs outils mis à disposition par Django. Ce tutoriel m'as permis de comprendre aussi les bonnes manières du développement comme la compréhension du modèle MVT : models, views, templates.
Ce modèle de développement consiste à séparer la gestion des données, la logique de traitement et la partie visible de l'application.

Le modèle est la partie de gestion des données. Plus simplement, c'est dans le modèle que l'on va interagir avec notre base de données. Chaque model déclaré dans ce fichier représente une table avec ces champs. Nous pouvons faire le parallèle avec le modèle MVC, souvent utilisé en JAVA lors de programmation orienté objet que nous avons explorer lors de nos de programmation objet avancé ou encore lors de paradigme de programmation.
La vue est responsable de la partie logique de l'application. Elle sera chargée du traitement des requêtes et des réponses. La vue interagit directement avec le modèle pour récupérer les données nécessaires et les traiter.
Enfin le Template est la partie visible de l'application. Globalement on parle ici des fichiers html de l'application qui afficherons notamment les données gérées par les vues. Django offre la possibilité d'utilisé des outils backend directement dans les fichiers html comme des « if / else » ou encore d'intégrer directement l'appel d'une vue depuis ces fichiers.

Pour donner un exemple simple et concret, lors de la création d'une page de login, le modèle s'occupera de récupérer les données des comptes existants pour les transmettre à la vue. La vue elle va pouvoir faire les vérifications de l'existence du compte ou non et agir en conséquence : valider et rediriger vers une page en cas de succès ou gérer les erreurs potentielles. Le Template affichera la page de login qu'appellera la vue pour vérifier les informations entrées dans le formulaire.

Après avoir compris les principes fondamentaux de Django, je me suis penché sur l'utilisation des annuaires LDAP. Monsieur Desraux m'as mis à disposition un annuaire de tests que je pouvais exploiter comme bon me semble.

Une fois tous ces concepts compris, j'ai pu commencer le développement de l'application.

La première étape était de réussir à récupérer des données de l'annuaire LDAP pour les afficher sur l'application web. Pour réussir, il me fallait me connecter à cet annuaire. Grace à la bibliothèque ldap3 de Django, la configuration et l'exécution de requête a été grandement

facilité. Pour une question de sécurité, il est primordial de fermer la connexion à l'annuaire pour éviter toute intrusion lorsque les requêtes ne sont pas envoyées.

Ensuite, j'ai enregistré les données LDAP dans un modèle LdapData afin de pouvoir les enregistrer et les afficher dans un tableau. Les données enregistrer dans le modèle sont ensuit transformer en objet python pouvoir être manipuler. Pour une meilleure compréhension, j'ai aussi ajouté un système de pagination afin d'afficher les données dans un tableau affichable sur une page web. J'ai aussi ajouté un système de tri par ordre croissant et décroissant avec une barre de recherche pour faciliter l'utilisation du tableau. Après cela, plusieurs vues ont été créer pour la création, la modification ou encore la suppression des comptes LDAP.

Après cela, je me suis tourné vers la partie Login/logout/registration. Tout simplement, l'idée était de créer des formulaires de login et d'inscription simple. Concernant le formulaire d'inscription, je voulais qu'il ne soit accessible que par les administrateurs. Par conséquent, j'ai créé une vue admin_required qui me permet d'être appelé sur les vues souhaité afin mettre une dépendance sur l'accès de la vue en question.

```python
def admin_required(login_url=None):
    return user_passes_test(lambda u: u.is_active and u.role == 'admin', login_url=login_url)

@admin_required(login_url='/login/')
def registerPage(request):
    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')  # Assurez-vous d'avoir une vue de connexion
    else:
        form = CustomUserCreationForm()
    return render(request, 'home/login_register.html', {'form': form})
```

C'est à partir de maintenant que nous avons eu des changements de scopes avec mes tuteurs sur plusieurs aspects d'Infocentre. La première était concernant ce fonctionnement de login/registration. Lors de l'une de nos réunions mensuelles, j'ai évoqué le système d'enregistrement propre à l'ENSEA actuellement mis en place sur plusieurs outils en interne. Ce système repose un ce qu'on appelle le CAS SSO. Le CAS ou Central Authentic Service est un système SSO : Single Sign-On. Cela permet aux utilisateurs d'accéder à plusieurs applications en ne s'authentifiant qu'une seule fois. Pour exemple, nous avons un intranet et un une application RH pour avoir un récapitulatif de notre nombre d'heure travaillé, notre planning, nos CP … Le CAS SSO permet lors de la première connexion à l'intranet par exemple, de m'authentifier auprès de nos servers ce qui me permet d'accéder à l'outils RH sans y mettre mes identifiants. Lors de l'authentification, le CAS génère un ticket d'authentification qui permet de valider notre identité auprès des applications.

En plus de son côté pratique, les identifiants des utilisateurs sont centralisés et non stockés dans l'application réduisant le risque de compromission. Cette centralisation facilite aussi les administrateurs qui gèrerons l'identités et les permissions aux applications à un seul et même endroit.

Mon but était de faire une application le plus homogène possible par rapport aux applications déjà existante. Par conséquent le système de login/registration a été mis de côté pour appeler le CAS SSO lors de l'authentification.

Le prochain changement de Scopes concerne l'utilisation de l'annuaire LDAP. Mes tuteurs ont souhaité que l'on fasse une mise au point quant à l'utilisation de l'annuaire LDAP. Leur souhait était que chaque nouveau compte soit créé dans une base de données tiers. Nous nous sommes donc questionnés sur l'utilité de l'annuaire LDAP dans ce contexte. Nous avons convenu que l'annuaire LDAP existant nous permettra de connaitre les comptes déjà existants mais n'allait pas être responsable de la création des nouveaux. J'ai donc réadapté mon code afin de faire en sorte de pouvoir accéder aux donner LDAP et ceux de la BDD.

```python
def display_ldap_and_db_data(request):
    query = request.GET.get('q', '')
    order = request.GET.get('order', 'asc')

    # Récupération et fusion des données depuis les modèles en fonction de la requête de recherche
    if query:
        ldap_data = LdapData.objects.filter(name__icontains=query)
        comptes_bd = Compte.objects.filter(name__icontains=query)
    else:
        ldap_data = LdapData.objects.all()
        comptes_bd = Compte.objects.all()
    all_accounts = list(ldap_data) + list(comptes_bd)

    # Appliquer le tri sur les données LDAP uniquement, ajuster si nécessaire pour les comptes_bd
    if order == 'desc':
        all_accounts.sort(key=lambda x: x.name, reverse=True)
    else:
        all_accounts.sort(key=lambda x: x.name)

    # Suppression et rechargement des données LDAP (vérifiez la nécessité de cette étape)
    LdapData.objects.all().delete()
    save_ldap_data_to_model()

    paginator = Paginator(all_accounts, 20)
    page = request.GET.get('page', 1)

    try:
        ldap_and_db_data = paginator.page(page)
    except PageNotAnInteger:
        ldap_and_db_data = paginator.page(1)
    except EmptyPage:
        ldap_and_db_data = paginator.page(paginator.num_pages)

    form = UpdateEmployeeForm()

    # Inclusion des paramètres de recherche et de tri dans le contexte
    context = {
        'ldap_and_db_data': ldap_and_db_data,
        'form': form,
        'query': query,
        'order': order
    }

    return render(request, 'home/ldap_template.html', context)
```

Ce code permet donc de transformer en objet les données LDAP et les données provenant de la BDD afin de les fusionner pour ne faire qu'un. All_accounts représente cette fusion des données.

Actuellement nous pouvons donc consulter l'ensemble des comptes existant et modifier/ajouter/supprimer les comptes présent sur la base de données.

La dernière étape du développement de ce projet fût de mettre en place une API permettant de récupérer les données d'un compte afin de les transmettre à une API externes.

Pour cette partie-là, nous avons fait un bond en arrière afin de revenir à l'étape de conception. En effet il nous fallait définir exactement comment nous voulions que les données soient transmises. Dans un premier temps, nous nous sommes rendu compte que suivant le compte créé, les accès ne seront pas les mêmes. Les comptes sont créés pour plusieurs personnes que ce soient des élèves, des vacataires, des professeurs ou encore des associations. En fonction du rôle attribué au compte, les accès diffèrent. Ainsi il nous fallait avoir un moyen de différencier les comptes avec l'opportunité d'ajouter des rôles selon les besoins. J'ai donc séparé la table User de la table Rôle afin de différencier ces 2 caractéristiques.

Enfin j'ai pu me mettre sur le développement de l'API directement. Pour cela je me suis aidé de la documentation Django officielle ainsi que du même tutoriel YouTube que lors de mon auto-formation. La création d'une API en Django passe par plusieurs étapes que je vais vous expliquer en résumé afin de comprendre comment cela fonctionne.
Tout d'abord, qu'est qu'une API : « ***Une API est une interface qui permet à deux applications logicielles de s'interfacer et d'interagir. Elle comprend des instructions et des standards qui permettent aux développeurs d'accéder aux données et aux fonctionnalités d'une application sans avoir besoin de connaître les détails internes de son fonctionnement*** » **(**Michael Goodwin, 09 April 2024 **|** **https://www.ibm.com/topics/api)**

Pour revenir à la configuration de l'api, j'ai utilisé la bibliothèque djangorestframework. Outre la configuration au sein des paramètre de l'application, l'API utilise un fichier serializeurs.py qui définit de quel modèle va dépendre l'API. En l'occurrence le modèle des Comptes. Ensuite, j'ai créé une vue dans laquelle j'ai déclaré le comportement de l'API en fonction des requêtes que l'on utilise. Dans notre cas nous avons besoins des requête POST, PUT et DELETE. Dans ce même fichier j'ai défini l'url de l'API externe que je souhaite interroger. Ensuite, j'ai créé une class par action. Nous avons une class Create, Update, Delete. Chacun transmet des informations et donnent des ordres différents à l'API. La différence avec le reste des vues qui communiquent directement avec les templates, j'ai utilisé du javascript pour la transmission des données du template à la vue. L'utilisation d javascript dans ce cas précis permet dans un premier temps un meilleur dynamisme dans l'utilisation de l'API. Cela permet à ce que la page ne se rafraichisse pas à chaque appel API que nous faisons. Ensuite, le javascript va permettre l'envoie asynchrone des requêtes permettant la récupération et l'envoi des données en arrière-plan. Pour résumé, cela rend l'application plus dynamique et fluide dans son utilisation.

Ma mission principale est donc terminée à ce moment. Cependant, il nous faut désormais valider les données nécessaires aux API externe afin de n'omettre aucun champ. Ensuite, je procèderai avec l'aide de Monsieur DE Pellgrin à la mise en production de l'application Pour ce faire, nous utiliserons Docker pour la conteneurisation de l'application. Cela nous permettra de déployer l'application avec toutes ces dépendances de façon plus simple et automatique.

Missions annexes

Mon rôle au sein de l'ENSEA étant de concevoir une application de A à Z implique que je n'avais pas vraiment de mission quotidienne si ce n'est de travailler sur ce projet. Cependant, tout au long de mon année d'alternance certaines habitudes faisait aussi partie de mes tâches à accomplir. Comme évoqué précédemment, le SRI voulait profiter d'Infocentre pour introduire la technologie Django dans l'établissement. Par conséquent, j'avais pour mission d'explorer les bases de Django afin de les transmettre à mes collaborateurs. J'ai donc décidé lors de mon auto-formation de prendre des notes sur la conception d'un projet Django et ses avantages. J'ai aussi pris plusieurs notes sur la technique pure. Comprendre le fonctionnement de certains fichiers et bien évidemment des outils que nous propose Django comme le Dashboard administrateur ou encore l'ORM. Ainsi de manière hebdomadaire, j'ai pu présenter différents aspects avec les avantages et les inconvénients de l'utilisation de Django.

Ma curiosité personnelle m'a aussi amené à m'intéresser aux missions de mes collaborateurs Nils et Richard avec qui nous partageons le bureau. Ces derniers assignés au parc informatique et à la gestion bureautique avaient quotidiennement des missions de support interne auprès de différents intervenants professeurs comme élèves. J'ai pu les assister dans certaines tâches de manière exceptionnelle comme le changement de matériel d'un ordinateur ou même essayer de les épauler sur la réflexion d'un problème informatique rencontré par les utilisateurs.

Difficultés rencontrées

L'objectif étant de concevoir seul un projet dans son ensemble, certaines difficultés était à prévoir et j'ai essayé de m'y préparer le plus possible. Une des premières difficultés que j'ai rencontrées concerne l'organisation. Mes cours de management de projet m'auront été bien utile pour combler à certains aspects que je n'avais pas pris en compte en premier lieu. Dès le début, je me posais plusieurs questions sur le temps qu'allait me prendre l'auto-formation. Je devais trouver le bon équilibre entre se former assez pour être à l'aise sur le sujet mais ne pas prendre trop de temps pour avoir le temps de finir le projet à temps. Je me devais ensuite d'anticiper le temps de développement par services. Par exemple, je ne pouvais pas me permettre de passer beaucoup de temps sur la partie login au détriment de l'API.
Par conséquent je devais prioriser certaines tâches ayant une criticité plus importante que d'autres.

Une autre difficulté rencontrée a concerné la conception de l'application. Nous avions avec mes tuteurs une approche plus ou moins précise du rendue finale d'Infocentre. Cependant, le temps nous a amenés à réflexion sur l'objectif et le rendu de l'application. Nous avons donc dû revoir le scope du projet et j'ai dû adapter mon développement à ces changements.

Enfin, une difficulté que nous avions essayé d'anticiper mais qui nous a tout de même freiner dans l'élaboration d'Infocentre a été le fait d'être seul sur ce projet. Bien évidemment que nous étions tous au courant de la difficulté de la tâche et ce dès lors de notre premier entretien. Mes tuteurs avaient insisté, à juste titre, sur ma capacité à être autonome. Cependant nous sommes tous les trois rendu compte des difficultés rencontré et ceux malgré nos efforts. Nous avons essayé de communiquer le plus possible que l'avancement du projet mais force est de constater que cela n'a pas empêché certains quiproquos à certains moments.

Compétences acquises

Avant de procéder au bilan de cette année enrichissante, voici l'ensemble des compétences que j'ai pu acquérir au cours de l'année. Chaque tableau correspond à un bloc de compétence à valider selon France Compétences. (*https://www.francecompetences.fr/recherche/rncp/35717/#anchor1*)

Compétences Acquises en Entreprise	**UE Correspondantes à Valider**
Développement d'une application web avec Django	**RNCP35717BC02 - Piloter un projet informatique** - Concevoir l'architecture du SI - Mettre en place les procédures d'exploitation, d'administration et de maintenance du système - Intégrer l'expérience utilisateur dans la conception de la solution - Coder en différents langages et sur différentes plateformes
Gestion de projet de A à Z (Planification, développement, déploiement)	**RNCP35717BC02 - Piloter un projet informatique** - Manager les équipes affectées au projet - Assurer la mise en œuvre, le suivi, le contrôle et le reporting du projet
Automatisation de processus avec des API	**RNCP35717BC07 - Ingénierie des logiciels et systèmes** - Proposer les technologies et outils nécessaires à la mise en œuvre - Piloter un projet de développement logiciel et système
Mise en œuvre et intégration de systèmes de sécurité (CAS SSO)	**RNCP35717BC04 - Assurer la sécurisation des données et des traitements d'un projet** - Déterminer les mesures de sécurité pour réduire ou éviter les risques - Assurer la mise en œuvre d'un système d'authentification des utilisateurs - Assurer la sauvegarde des données
Conception et utilisation de bases de données relationnelles	**RNCP35717BC02 - Piloter un projet informatique** - Concevoir l'architecture du SI - Mettre en place les procédures d'exploitation, d'administration et de maintenance du système
Activité de veille technologique et mise à jour des compétences	**RNCP35717BC03 - Créer et développer des projets numériques innovants et créateurs de valeur** - Maîtriser l'intégration des nouvelles technologies dans l'organisation
Formation et partage de connaissances en Django avec les collaborateurs	**RNCP35717BC02 - Piloter un projet informatique** - Accompagner le changement

Compétences Acquises en Entreprise	UE Correspondantes à Valider
Création d'une interface utilisateur ergonomique et intuitive	**RNCP35717BC02 - Piloter un projet informatique** - Intégrer l'expérience utilisateur dans la conception de la solution
Analyse et recommandation de solutions pour les projets en cours	**RNCP35717BC01 - Analyser le SI et conseiller sur son évolution** -Effectuer des recommandations sur les projets informatiques en cours - Déterminer les solutions d'évolution adaptées
Modification du scope du projet pour l'adapter à l'objectif final	**RNCP35717BC01 - Analyser le SI et conseiller sur son évolution** - Déterminer les solutions d'évolution adaptées - Évaluer chaque solution en termes de coûts, de délai et de valeur apportée

Conclusion et bilan

Conclusion et Bilan Professionnel

Durant cette année d'alternance au sein de l'ENSEA, j'ai eu l'opportunité de travailler sur un projet significatif et de développer une application web. Mon rôle en tant que développeur dans le Service des Ressources Informatiques (SRI) m'a permis de participer activement à la création et à la mise en œuvre de l'application "Infocentre", qui vise à moderniser et automatiser la gestion des comptes utilisateurs.

Réalisation des Objectifs

Mon travail a commencé par une phase d'auto-formation sur le Framework Django et les annuaires LDAP. Cette formation m'a permis de développer une application robuste et fonctionnelle, conforme aux exigences initiales définies par mes tuteurs. "Infocentre" permet désormais de créer, modifier et supprimer des comptes utilisateurs de manière automatisée, en intégrant des appels API pour la gestion des accès à divers outils et services, comme 365 et Galactus.

Difficultés et Solutions

L'un des défis majeurs a été de travailler de manière autonome tout en maintenant une communication efficace avec mes tuteurs. Les réunions hebdomadaires avec mes tuteurs ont été cruciales pour ajuster les objectifs et les priorités du projet en fonction des retours et des obstacles rencontrés. Cela a permis de surmonter les difficultés et d'ajuster le scope du projet en fonction des besoins évolutifs.

Contribution à l'Organisation

L'application "Infocentre" a pour but d'améliorer l'efficacité du SRI en automatisant des processus auparavant manuels et en fournissant une interface utilisateur plus intuitive et ergonomique. L'intégration du CAS SSO (Central Authentication Service Single Sign-On) a renforcé la sécurité des identifiants utilisateurs, tout en simplifiant l'expérience utilisateur à travers un système de connexion unique.

Résultats et Déploiement

Le déploiement de l'application "Infocentre" est en cours lorsque je rendrais ce rapport. Le déploiement devrait s'effectuer via la conteneurisation via docker de l'application. Cette étape sera faite par Monsieur De Pellgrin que je pourrais assister afin de pouvoir participer à l'entièreté du projet.

Cette année d'alternance m'a permis de développer des compétences techniques solides et de gagner en autonomie et en rigueur professionnelle. La satisfaction de voir une application complète et fonctionnelle, créée de bout en bout, est une source de fierté. Les défis rencontrés et surmontés m'ont permis d'acquérir une maturité professionnelle et une confiance en mes compétences, me préparant ainsi à affronter de futurs défis professionnels avec assurance.

Bilan de formation

Cette année de formation m'as permis de développer de nouvelle compétence notamment de gestion de projet, ainsi que de consolider des compétences acquises de mes formations précédentes.

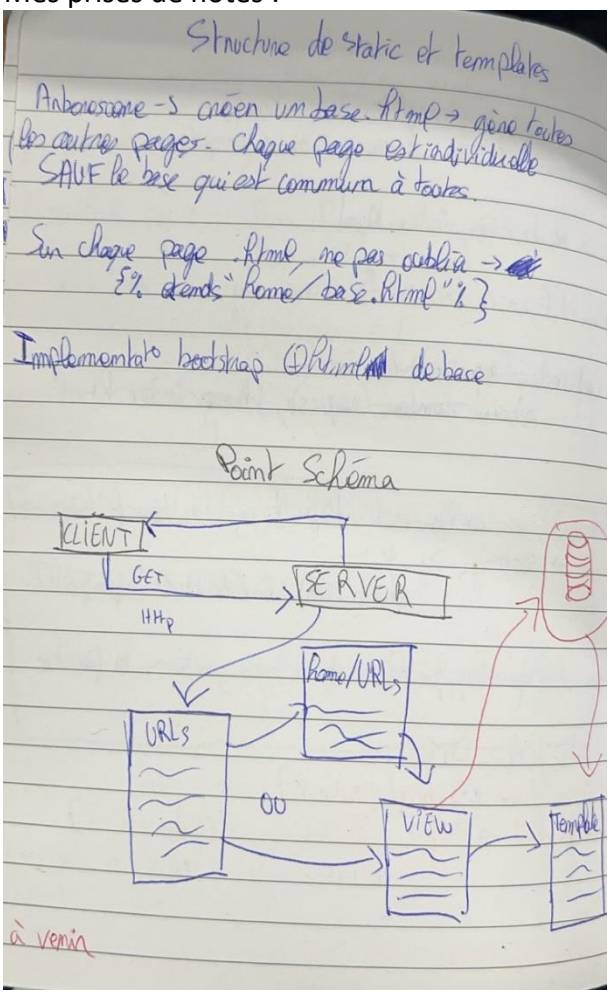
J'ai pu mieux comprendre certains aspects de la gestion de projet qui m'étais inconnu et qui m'ont énormément apporté lors de la réalisation de mon projet professionnel. Les différents projets tel que la piscine Python m'ont permis d'expérimenter de nouvelle façon de travailler en équipe. D'autres projet tels que MiniTeams durant le cours de Linux et Principe de programmation m'auront permis de me surpasser sur des sujets où je me sentais moins à l'aise.

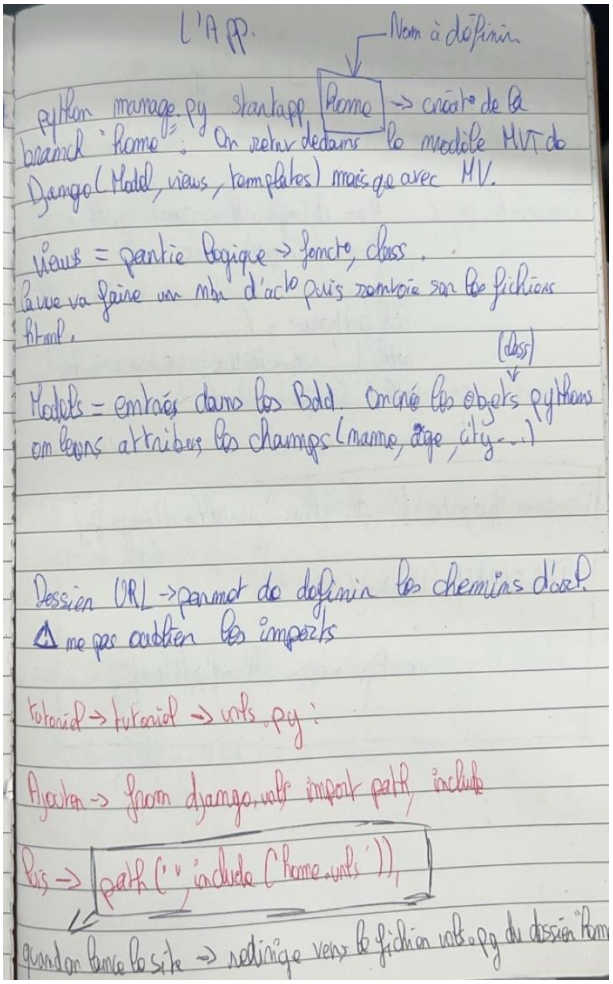
Cette année a aussi été riche humainement. J'ai observé une vraie solidarité au sein de notre classe ou tout le monde essayait de se tirer vers le haut. J'ai pu aussi faire des rencontres intéressantes avec nos professeurs qui étaient tous à l'écoute. Cela m'a aussi aidé concernant mon futur afin de comprendre les enjeux d'un formateur, métier auquel j'aspire.

Ces pour ces raisons que je vais continuer au sein de l'ESIEE-IT pour mon master afin d'entrer officiellement sur le marché du travail en étant le plus compétent possible. J'espère pouvoir continuer de développer de nouvelles compétences au sein du diplôme Bac+5 IL&MSI.
J'ai la motivation de sortir de l'ESIEE-IT le plus compétent possible avec d'exercer ce métier de formateur de la meilleure des façons.

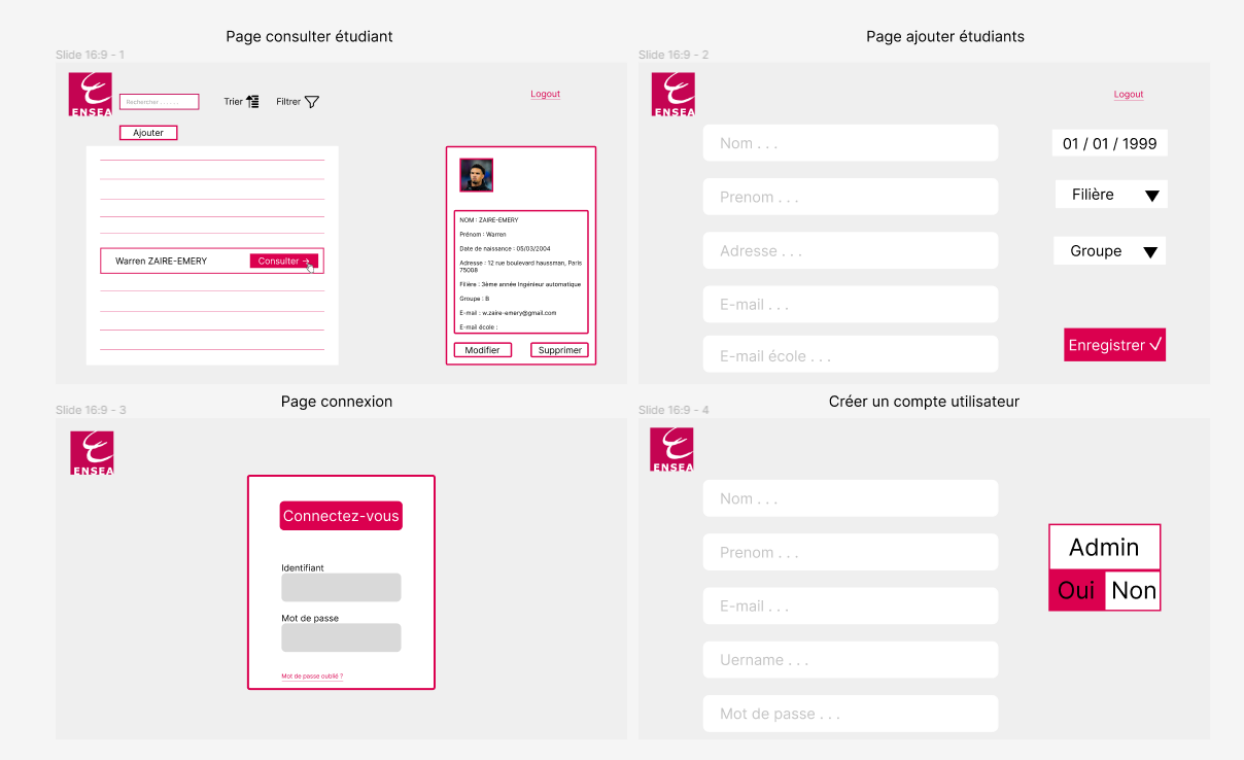
Annexes

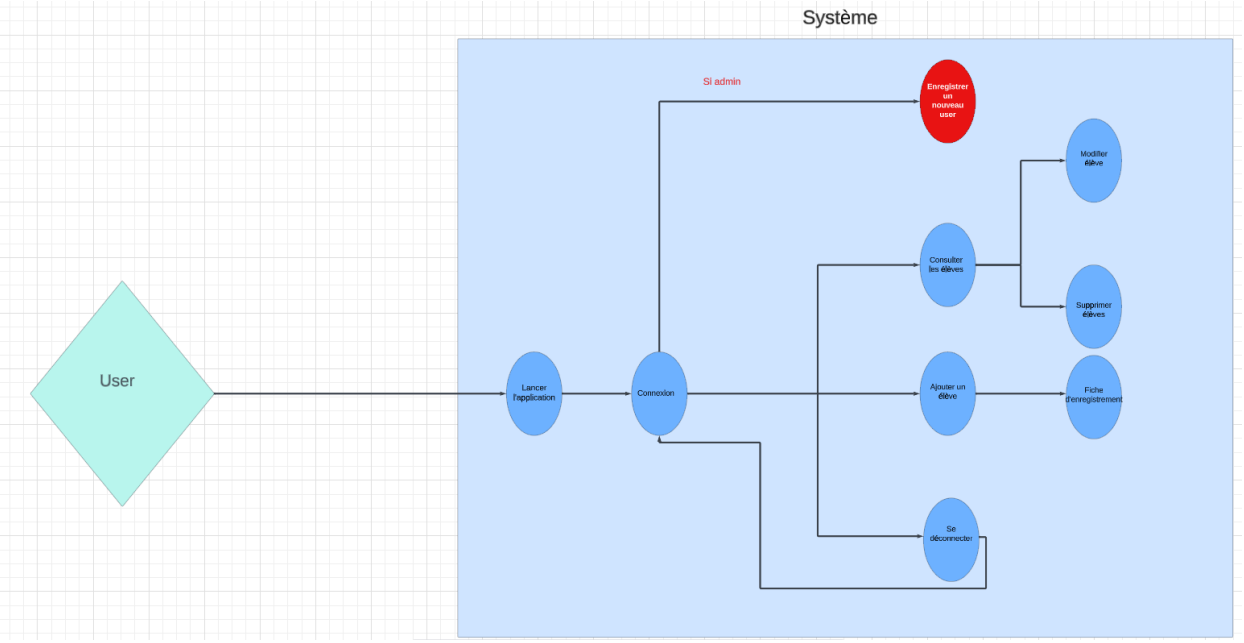
Mes prises de notes :



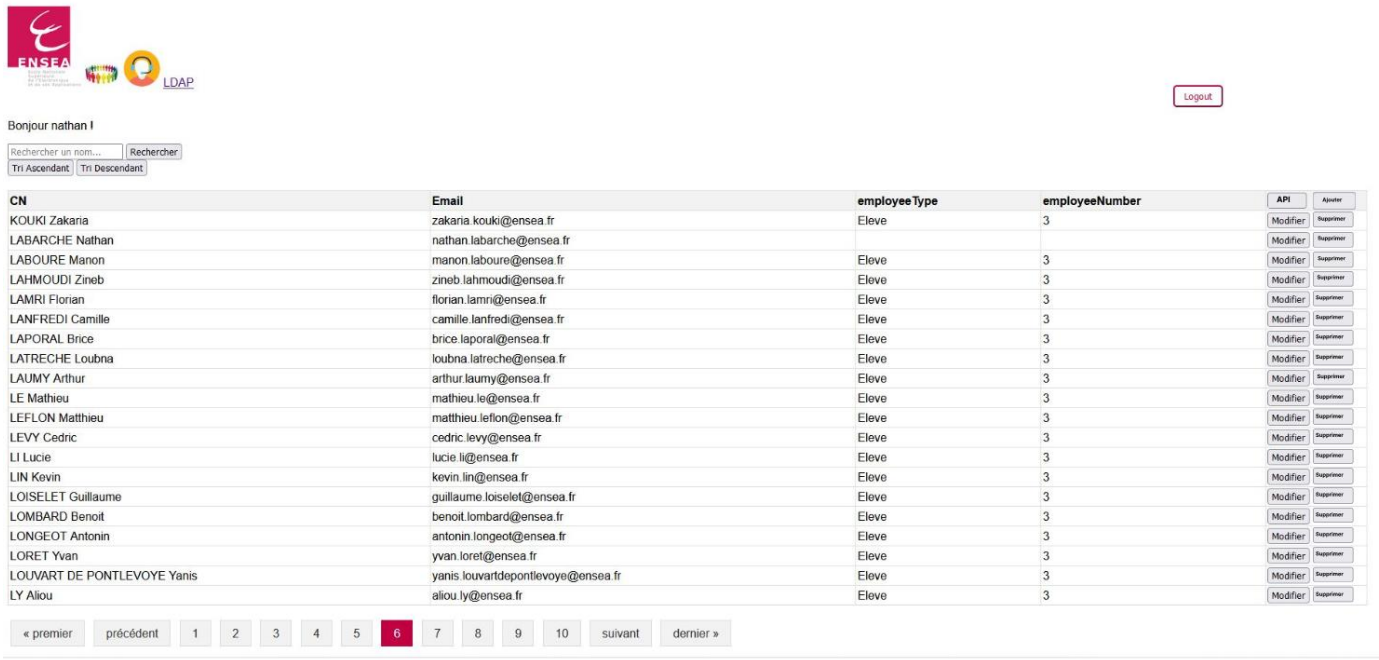


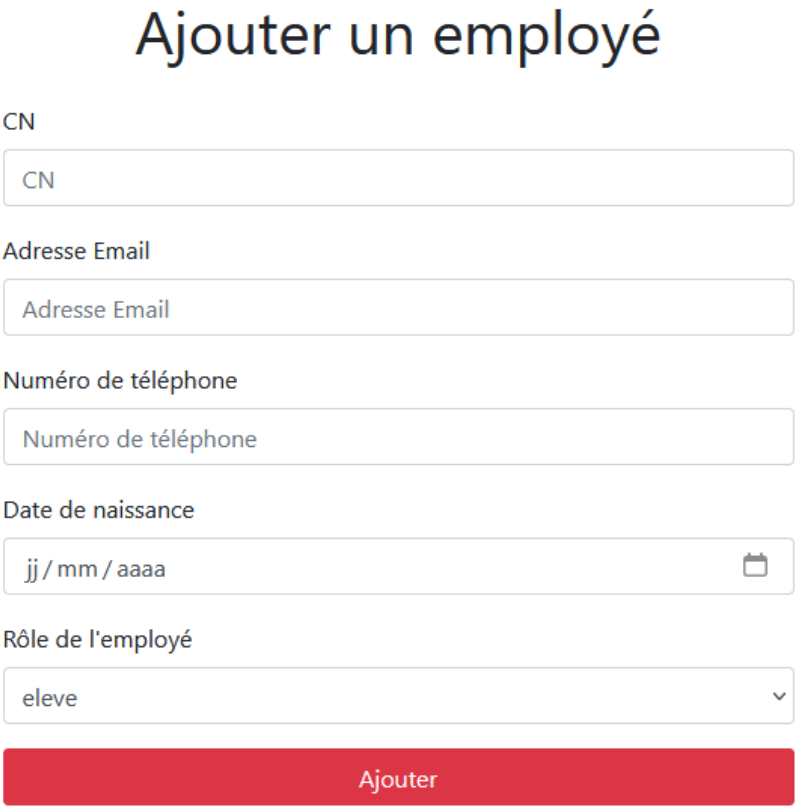
Les premières phases de conception :



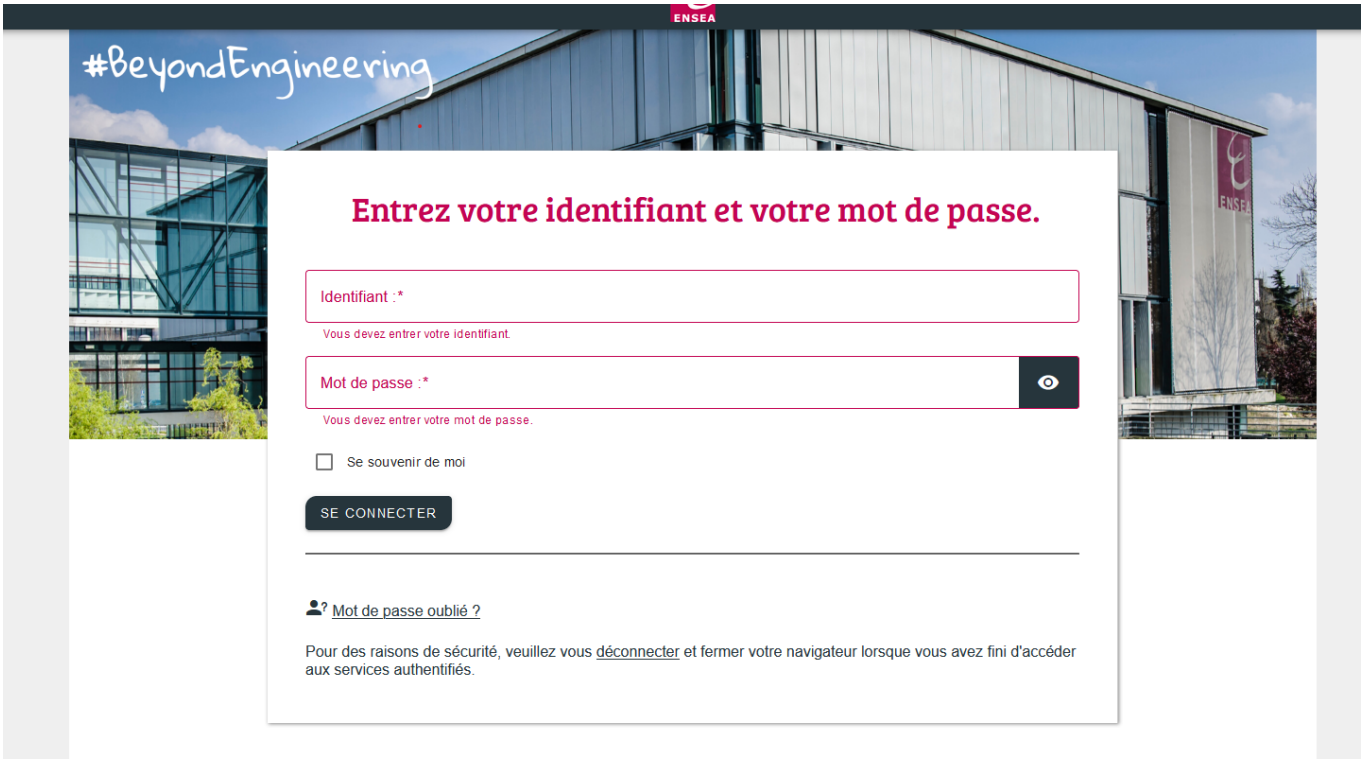


L’application :





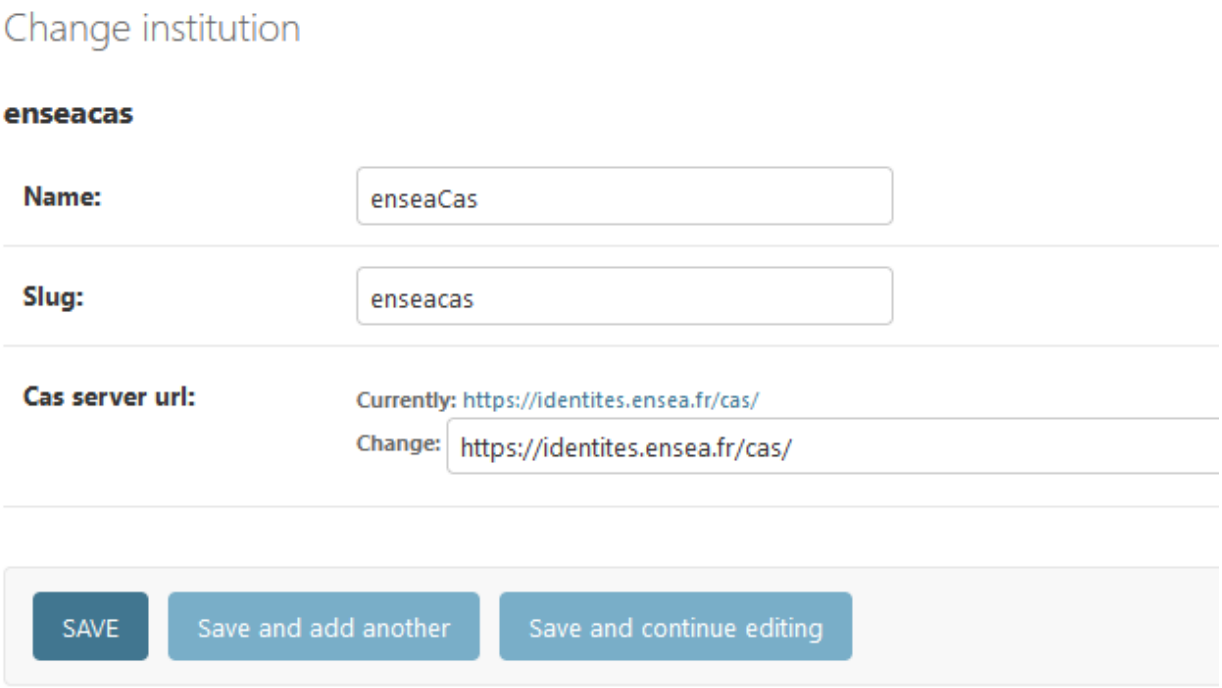
Login via CAS SSO :



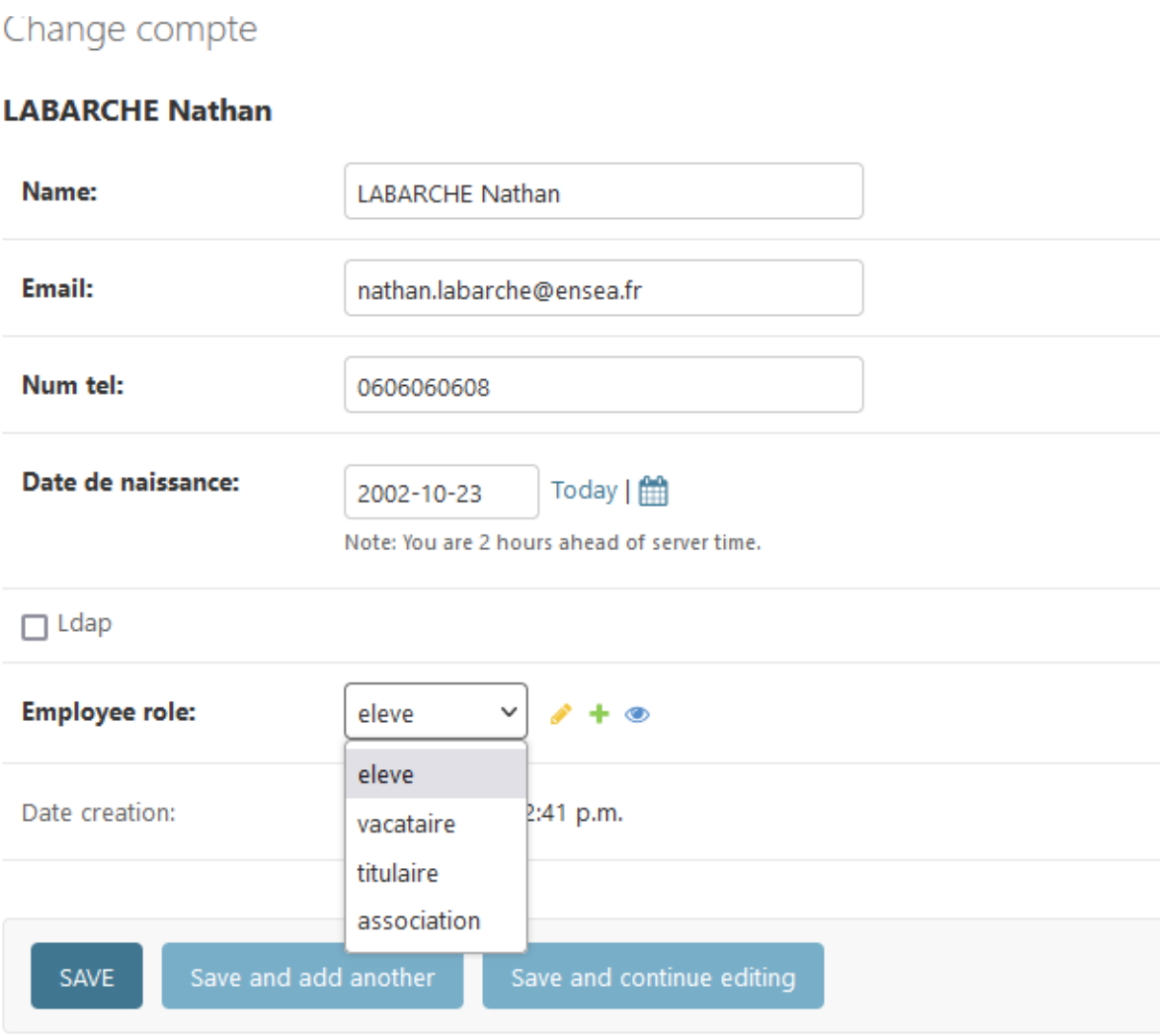
Jeux de données utilisé pour les tests :

ABBAD Samira	samira.abbad@ensea.fr	3	Eleve
ABESSIRA Ethan	ethan.abessira@ensea.fr	3	Eleve
AHMED Ines	ines.ahmed@ensea.fr	3	Eleve
AIT DJOUDI OUFELLA Idir	idir.aitdjoudioufella@ensea.fr	3	Eleve
AMARIR Youssef	youssef.amarir@ensea.fr	3	Eleve
ANEMICHE Nassim	nassim.anemiche@ensea.fr	3	Eleve
ANOT-DELCOURT Claire	claire.anot-delcourt@ensea.fr	3	Eleve
APPOURCHAUX Leo	leo.appourchaux@ensea.fr	3	Eleve
ARIOUICH Hamza	hamza.ariouich@ensea.fr	3	Eleve
AURIAC Robin	robin.auriac@ensea.fr	3	Eleve
BABA-AISSA Remi	remi.baba-aissa@ensea.fr	3	Eleve
BARRIE Mathys	mathys.barrie@ensea.fr	3	Eleve
BAYI GWET Bryan Royce	bryanroyce.bayigwet@ensea.fr	3	Eleve
BAZAIA Tancrede	tancrede.bazaia@ensea.fr	3	Eleve
BEAUDET Theo	theo.beaudet@ensea.fr	3	Eleve
BEN AMAR Rayen	rayen.benamar@ensea.fr	3	Eleve
BENOIT Dimitrii	dimitrii.benoit@ensea.fr	3	Eleve
BERDIJ Souhaila	souhaila.berdij@ensea.fr	3	Eleve
BERLIOZ Clement	clement.berlioz.22@ensea.fr	3	Eleve
BERNEDE Guillaume	guillaume.bernede@ensea.fr	3	Eleve
BERREE Teva	teva.berree@ensea.fr	3	Eleve
BICHEMIN Fannie	fannie.bichemin@ensea.fr	3	Eleve
BIEHLER Gauthier	gauthier.biehler@ensea.fr	3	Eleve
BITAUD--CANOEN Lael	lael.bitaud--canoen@ensea.fr	3	Eleve
BOUAZZAOUI Leila	leila.bouazzaoui@ensea.fr	3	Eleve
BOUDISSA Adem	adem.boudissa@ensea.fr	3	Eleve
BOUHRAZNINE Hicham	hicham.bouhraznine@ensea.fr	3	Eleve

Django admin – Configuration du CAS :



Django admin création de compte :



Annexes techniques :

Exemple pour le fichier models.py de l'application :

```python
class Compte(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField(unique=True)
    num_tel = models.CharField(max_length=20)
    date_de_naissance = models.DateField()
    date_creation = models.DateTimeField(auto_now_add=True)
    ldap = models.BooleanField(default=False)
    employee_role = models.ForeignKey(EmployeRole, on_delete=models.CASCADE, default=1)
    def __str__(self):
        return f"{self.name} "
```

Voici un exemple de views.py :

- Création de comptes :

```python
def creer_compte(request):
    if request.method == 'POST':
        form = CompteForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('ldap_data')  # Rediriger vers la page d'accueil ou une autre page après la création du compte
    else:
        form = CompteForm()
    return render(request, 'home/ajouter_employee.html', {'form': form})

def is_ldap_user(user_id):
    try:
        user = Compte.objects.get(pk=user_id)
        return user.ldap
    except Compte.DoesNotExist:
        return False
```

Configuration de l'API.

Tout d'abord nous avons le serializers qui va récupérer les données présentes dans le model Compte :

```python
from rest_framework import serializers
from .models import Compte

class CompteSerializer(serializers.ModelSerializer):
    class Meta:
        model = Compte
        fields = '__all__'
```

Maintenant voici la configuration pour envoyer à l'API externe :

```python
def send_to_external_api(data, method='post', endpoint=''):
    external_api_url = f'https://api.externe.com/{endpoint}'
    if method == 'post':
        response = requests.post(external_api_url, json=data)
    elif method == 'put':
        response = requests.put(external_api_url, json=data)
    elif method == 'delete':
        response = requests.delete(external_api_url, json=data)
    else:
        response = requests.get(external_api_url, params=data)
    return response
```

Voici un exemple de configuration de l'api. Ici nous avons pour la modification :

```python
class CompteRetrieveUpdateAPIView(generics.RetrieveUpdateAPIView):
    queryset = Compte.objects.all()
    serializer_class = CompteSerializer

    def perform_update(self, serializer):
        response = send_to_external_api(serializer.validated_data, method='put', endpoint=str(serializer.instance.id))
        if response.status_code == 200:
            serializer.save()
        else:
            raise Exception("Failed to send data to external API")
```

Enfin les urls qui redirigeront vers les vues associées :

```
path('api/comptes/', CompteListCreateAPIView.as_view(), name='compte-list-create'),
path('api/comptes/<int:pk>/', CompteRetrieveUpdateAPIView.as_view(), name='compte-retrieve-update'),
path('api/comptes/<int:pk>/delete/', CompteDestroyAPIView.as_view(), name='compte-destroy'),
```

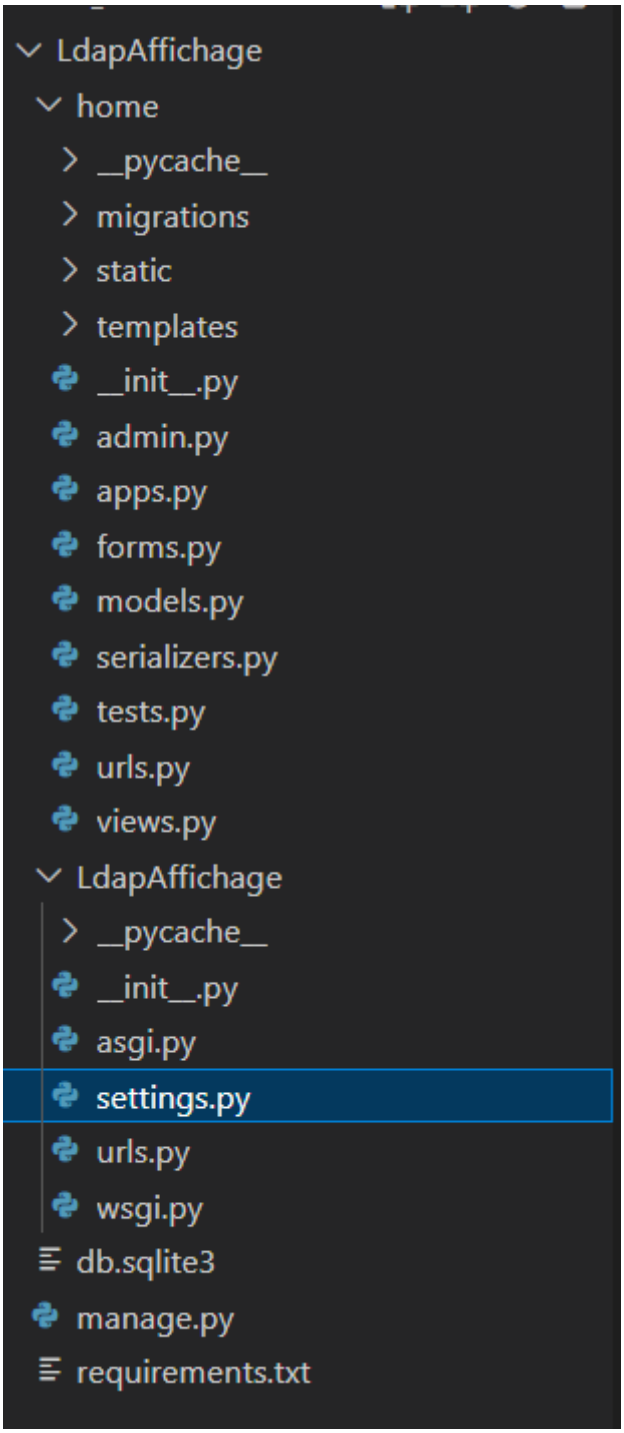
Pour terminer la couche visible, les templates html :

Ici on voit un tableau qui affiche le nom, le mail, le type d'employé, le numéro de l'employé.

Ensuite on a les différents appels de l'API avec un bouton pour chaque appel.
La boucle for permet de mettre toutes ces données pour chaque ligne du tableau.

```
<tbody>
  {% for entry in ldap_and_db_data %}
  <tr class="employee-row" data-email="{{ entry.email }}" data-employee-type="{{ entry.employeeType }}"
    data-cn="{{ entry.cn }}" data-employee-number="{{ entry.employeeNumber }}">
    <td>{{ entry.name }}</td>
    <td>{{ entry.email }}</td>
    <td>{{ entry.employeeType }}</td>
    <td>{{ entry.employeeNumber }}</td>
    <td>
      <button type="button" id="modify" class="btn btn-modify" data-toggle="modal"
        data-target="#modifyEmployeeModal{{ entry.id }}">Modifier</button>
    </td>
    <td>
      <button type="button" onclick="callApi('{{ entry.id }}')">Appel API</button>
      <button type="button" onclick="updateCompte('{{ entry.id }}')">Modifier</button>
      <button type="button" onclick="deleteCompte('{{ entry.id }}')">Supprimer</button>
    </td>
  </tr>
  {% endfor %}
</tbody>
```

Arborescence du projet :



Exemple du fichier settings.py pour la configuration du CAS SSO (pour une question de sécurité, j'ai décidé de cacher l'url de connexion au CAS de l'entreprise) :

```
HOME_URL = "/home/"
LOGIN_URL = "/accounts/login/"
LOGIN_REDIRECT_URL="/home/"
UNIAUTH_LOGIN_DISPLAY_STANDARD = False
UNIAUTH_LOGOUT_CAS_COMPLETELY = True

CAS_SERVER_URL = '
```

Fichier requirements.txt avec toutes les dépendances pour le bon fonctionnement du projet :

```
asgiref==3.8.1
certifi==2024.2.2
cffi==1.16.0
charset-normalizer==3.3.2
cryptography==42.0.7
Django==5.0.4
idna==3.7
paypal-checkout-serversdk==1.0.3
paypalhttp==1.0.1
paypalrestsdk==1.13.3
pillow==10.3.0
pycparser==2.22
pyOpenSSL==24.1.0
python-decouple==3.8
requests==2.31.0
six==1.16.0
sqlparse==0.5.0
tzdata==2024.1
urllib3==2.2.1
```

Enfin voici tous les liens vers les documentations qui m'auront été utiles durant cette année :

https://www.youtube.com/watch?v=PtQiiknWUcI&t=24141s&ab_channel=TraversyMedia

https://docs.djangoproject.com/fr/5.0/

https://medium.com/@ldgoodridge95/adding-cas-authentication-to-your-django-app-with-django-uniauth-13ff4e1e7bfa

https://openclassrooms.com/fr/courses/7192416-mettez-en-place-une-api-avec-django-rest-framework

https://django-auth-ldap.readthedocs.io/en/latest/

https://stackoverflow.com/questions/54549903/django-authentication-ldap-full-example