POLYTECH MARSEILLE
Aix*Marseille Université

INFO 5A

Rapport de Sécurité de systèmes d'informations

Système d'annuaire sécurisé

SMIRANI Ibtihell, **ES-SMAHI** Hussam, **ELOMARI** Lokmane, **BAIDO** Florian, **EL ALLAM** Ayman

Professeur : LAURENT-BURLE Guillaume

Année universitaire : 2023-2024

Table des matières

Table des figures

Introduction

Ce rapport présente le projet réalisé dans le cadre du module sécurité de systèmes d'informations.

Le projet vise à développer un site web exigeant l'authentification des utilisateurs. Les visiteurs ont la possibilité de se connecter s'ils possèdent un compte ou de créer un compte s'ils n'en ont pas. Cependant, lors de la première connexion, l'accès sera restreint. Le système comprend plusieurs rôles, notamment celui de simple utilisateur, qui permet d'ajouter des personnes à son annuaire et de le consulter. L'administrateur, quant à lui, détient tous les droits, lui permettant de révoquer ou d'attribuer des droits aux utilisateurs. De plus, chaque utilisateur a la faculté de fermer son compte à tout moment.

Nous introduirons dans un premier temps la conception du diagramme de cas d'utilisation. Ensuite, nous entamerons avec la présentation de l’architecture logicielle et du systéme de l’application. Dans le cadre de l'amélioration de l'architecture, Nous enchainerons par la suite par l’explication de la migration vers l’architecture microservice. Et nous finirons par aborder la conteneurisation de l'application en utilisant Docker.

I. Conception de l'application

Pour mieux comprendre les rôles gérés dans notre application ainsi que les différentes fonctionnalités on a élaboré le diagramme de cas d'utilisation.

1. Diagramme de cas d'utilisation

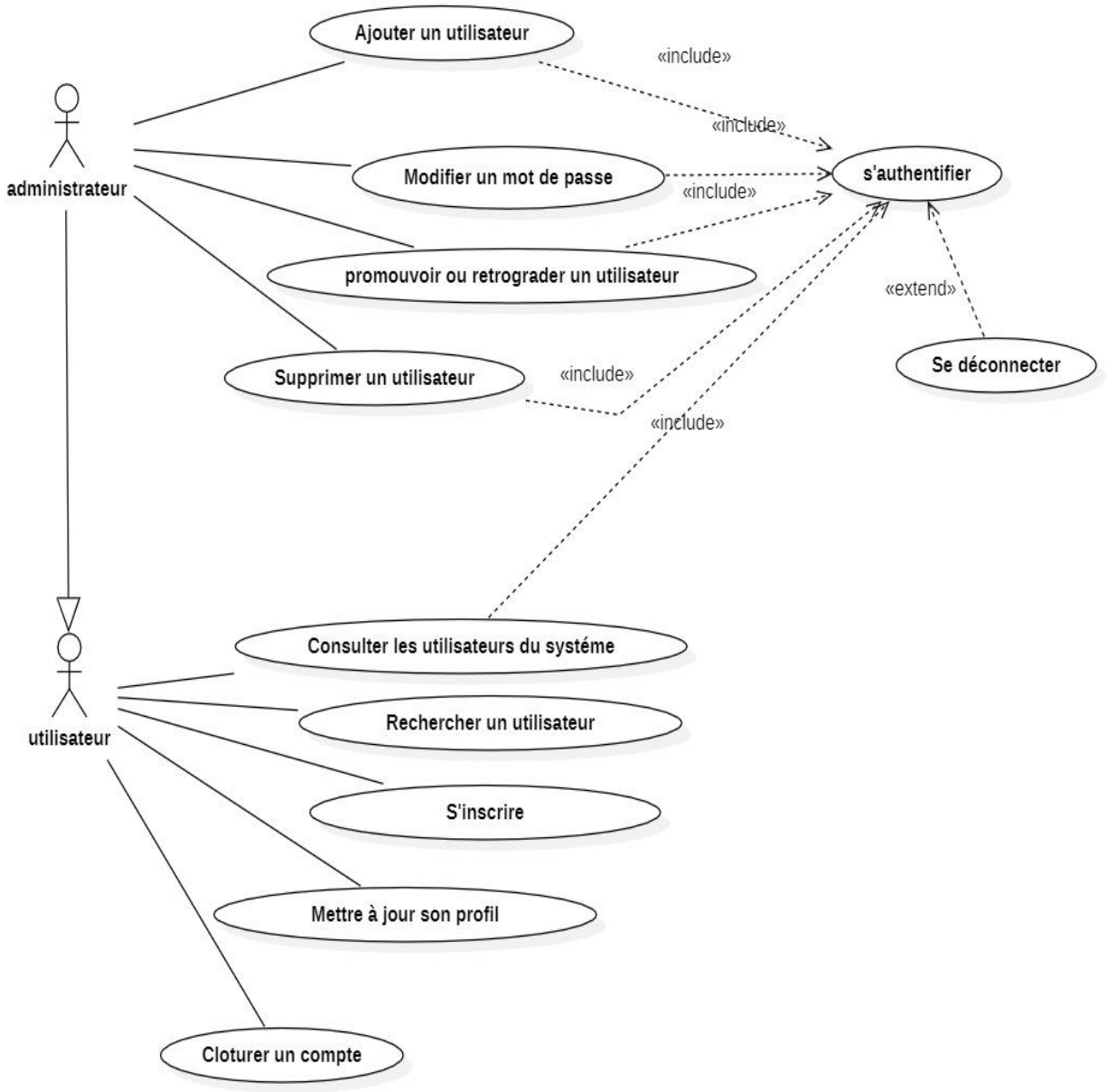

Figure 1: Diagramme de cas d'utilisation

2. Diagramme de classe

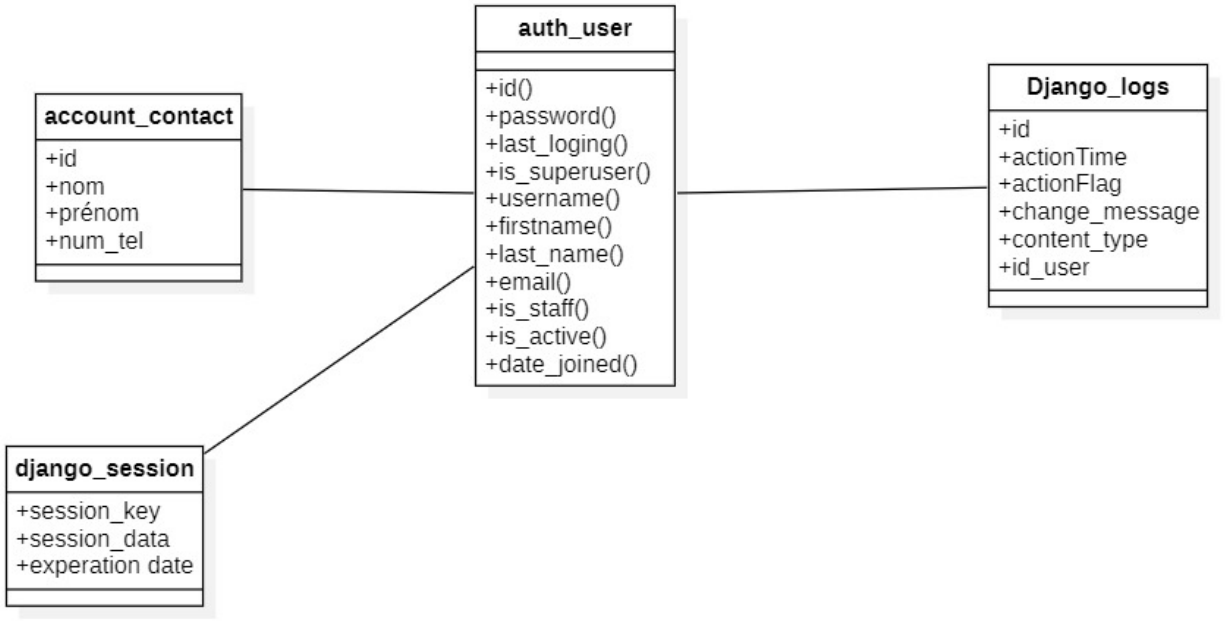

Figure 2 : Diagramme de classe

Un utilisateur regroupe l'ensemble de ses données de compte, incluant le nom d'utilisateur, l'adresse e-mail, et le mot de passe, ainsi que des autorisations telles que "is_staff", qui détermine si l'utilisateur est autorisé à modifier, ajouter, ou supprimer des contacts. De plus, le rôle de "is_superuser" confère à l'utilisateur des privilèges administratifs pour la modification, l'ajout ou la suppression d'autres comptes d'utilisateurs. La classe de session Django est utilisée pour gérer la session de l'utilisateur, tandis que Django Logs enregistre les audits des actions effectuées par l'utilisateur. Enfin, la table "contact" stocke des informations spécifiques telles que le nom, le prénom et le numéro de téléphone associés à chaque contact.

II. Architecture du projet

1. Outils utilisés

Coté Client (Django)

Django est un Framework web open source en Python. Il a pour but de rendre le développement d'applications web simple et basé sur la réutilisation de code.

Coté serveur (HTML)

HTML est un langage de balisage utilisé pour structurer le contenu d'une page web. Il s'agit d'un langage fondamental dans le développement web, utilisé pour décrire la structure et l'organisation des éléments sur une page web.

2. Architecture Microservice

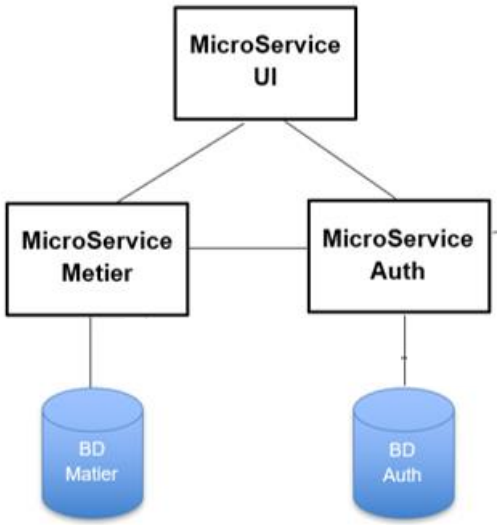

Figure 3 : Architecture logicielle

Dans notre architecture basée sur des microservices, nous avons mis en place 2 MS distincts comme le montre la figure ci-dessus pour répondre aux besoins spécifiques de notre application.

Le deuxième service est le MS-Authentification. Ce microservice est chargé de gérer les mécanismes d'authentification en utilisant des Tokens JWT et d'autorisation au sein de notre application.

Le troisième service est le MS-métier. Ce service représente le cœur fonctionnel de notre application. Il implémente la logique métier spécifique à notre domaine d'activité qui peut s'agir de fonctionnalités telles que la gestion des utilisateurs, des etc.

Après avoir décomposé notre projet en microservices, nous avons décidé de le containeriser à l'aide de Docker. Cette approche offre plusieurs avantages tels que la portabilité des microservices, facilitant leur déploiement sur différentes plateformes ainsi qu'une gestion des dépendances et des configurations devient plus simple grâce à l'encapsulation des microservices dans des conteneurs.

3. Architecture logicielle

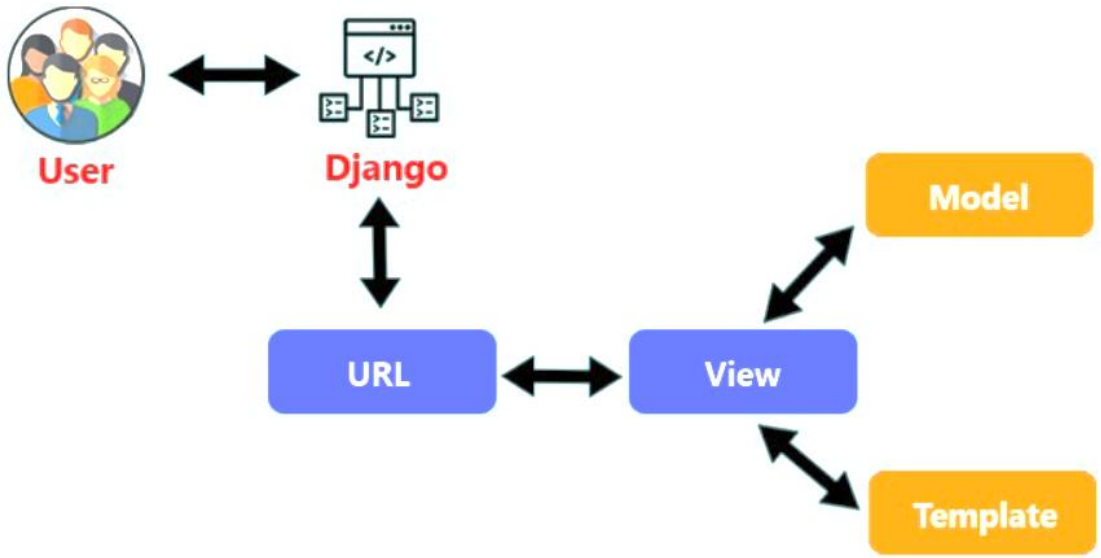

Figure 4 : Architecture logicielle

L'une des particularités de Django réside dans son utilisation de l'architecture MVT (Modèle-Vue-Template). Ce qui rend Django unique, c'est sa capacité à gérer lui-même la partie contrôleur, prenant en charge les requêtes du client et les droits associés aux actions.

L'architecture MVT se compose de trois éléments principaux : le Modèle, la Vue et le Template.

- Les modèles : sont représentés par des classes dans le fichier models.py. Par exemple, la classe Contact définit la structure des données associées aux contacts, avec chaque attribut correspondant à un champ dans la base de données.

- Les vues : sont implémentées dans le fichier views.py. Chaque fonction de vue gère une partie spécifique de l'application, telles que la modification de mots de passe, la gestion des profils, l'affichage de listes d'utilisateurs ou de contacts, etc. Ces vues récupèrent les données du modèle associé et utilisent des informations sur la requête et les paramètres de l'URL pour générer un rendu HTML à partir des données du modèle.
- Les templates : généralement des fichiers HTML, sont utilisés pour générer le rendu visuel des pages et sont associés aux fonctions de vue. Ces fichiers définissent la structure et la présentation des pages web, intégrant les données récupérées par les vues.

Lorsqu'un utilisateur accède à une page du site Django, le framework, en suivant les règles de routage URL définies, exécute la vue correspondante. Cette vue récupère les données des modèles, génère un rendu HTML à partir du template et renvoie le résultat au navigateur de l'utilisateur.

4. Architecture du système

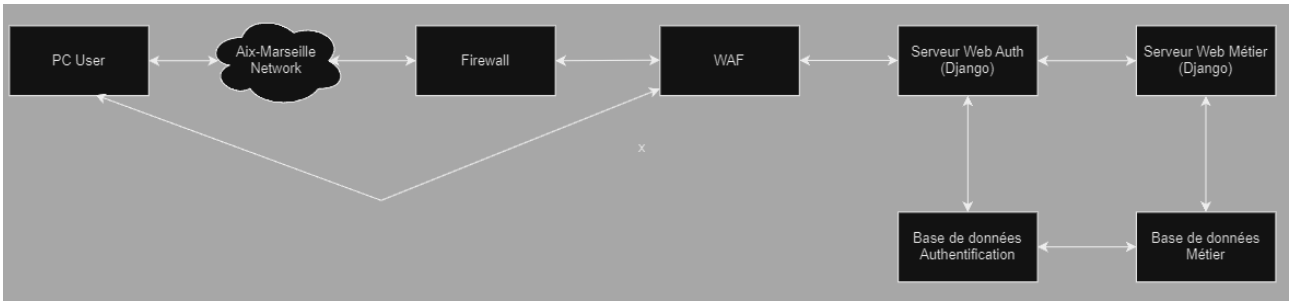

Figure 5 : Architecture du système

Lorsqu'un utilisateur souhaite accéder au site, il traverse le Web Application Firewall (WAF) et est ensuite redirigé vers le serveur web d'authentification pour procéder à l'authentification. Une fois authentifié avec succès, l'utilisateur est redirigé vers le serveur web Métier. Ce serveur web interroge la base de données pour vérifier les informations fournies par l'utilisateur telles que le nom d'utilisateur et le mot de passe. Une fois connecté, l'utilisateur obtient un jeton (JWT) pour garantir son statut d'authentification et empêcher un accès non autorisé au site.
Sur le serveur web, l'utilisateur peut effectuer diverses actions en fonction de ses droits d'accès. Lorsqu'une action est initiée, une requête est alors émise à la base de données.

III. Les tests

Les tests générés pour APIs dans Django appartiennent principalement à la catégorie des tests fonctionnels. Ces tests évaluent le comportement attendu de l'application du point de vue de l'utilisateur final en simulant les interactions réelles avec l'interface utilisateur et en vérifiant les réponses générées. Dans le cadre de ces tests, l'authentification de l'utilisateur est simulée à l'aide de données de test, et les résultats sont analysés pour s'assurer que les fonctionnalités de connexion et d'inscription fonctionnent correctement. Les assertions sont utilisées pour valider des éléments tels que les redirections après une inscription réussie, la gestion des erreurs en cas de données incorrectes, et d'autres comportements spécifiques de l'application.

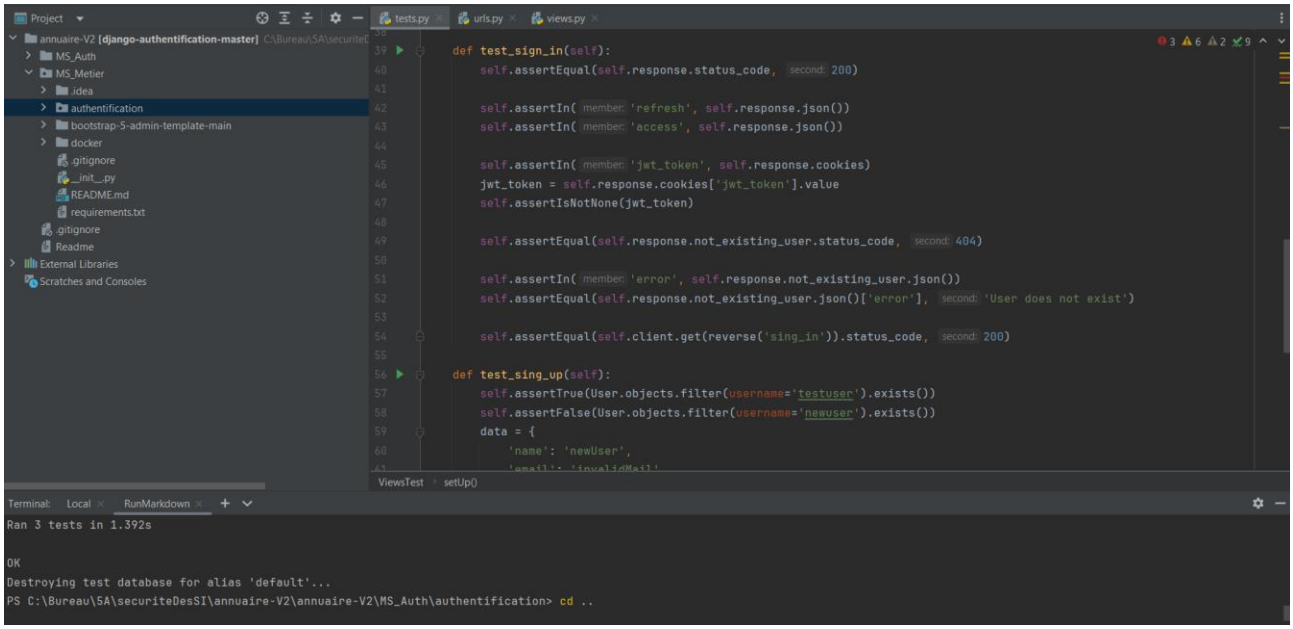

Figure 6 : Tests du microservice 'Authentification'

Conclusion

En conclusion, ce projet de sécurité des systèmes d'informations a été mené avec succès en mettant en place un site web sécurisé nécessitant

l'authentification des utilisateurs. La conception a été minutieusement élaborée, débutant par la modélisation des cas d'utilisation et des classes, puis évoluant vers une architecture logicielle robuste, basée sur des microservices et containerisée avec Docker.

L'introduction de l'architecture MVT de Django a été soulignée, mettant en avant sa capacité unique à gérer la partie contrôleur et à prendre en charge les requêtes utilisateur. Les composants clés, Modèle, Vue et Template, ont été clairement définis dans le contexte du développement de l'application.

L'architecture du système, avec son Web Application Firewall (WAF) et l'utilisation de microservices pour l'authentification et les fonctionnalités métier, a été présentée de manière détaillée. Ce rapport met également en lumière les tests fonctionnels effectués, notamment le test du microservice 'Authentification', assurant la qualité et le bon fonctionnement des fonctionnalités d'authentification et d'inscription.

En somme, ce projet démontre une approche complète et réfléchie en matière de sécurité des systèmes d'informations, intégrant des technologies modernes et des pratiques avancées de développement web pour aboutir à une application robuste et sécurisée.