Sécurité

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.

Architecture logicie

Après l'authentification du client dans la partie MS_Auth, cette dernière crypte les informations du client à l'aide de JWT et les transmet à la deuxième partie via des requêtes HTTP. La partie MS_Metier décode le JWT pour vérifier l'existence de l'utilisateur. Si l'utilisateur existe, il se connecte avec sa session et ses privilèges. En revanche, si l'utilisateur n'est pas présent dans la base de données de la partie métier, cela signifie qu'il s'agit d'un nouveau compte. Dans ce cas, l'utilisateur est recréé dans la base de données et une nouvelle session est initiée. À l'origine, l'utilisateur dispose d'un rôle basique de consultation des contacts, jusqu'à ce que l'administrateur lui attribue les privilèges d'éditeur.

▼ Projet énoncé :

Projet: Système d’annuaire sécurisé

Vous êtes responsable de la conception et de la mise en œuvre d’un système d’annuaire sécurisé. Votre équipe de projet aura la charge de:

- Définir l'architecture logicielle
- Définir l’architecture du système
- Procéder au développement de la solution logicielle
- Procéder à la mise en place de l'infrastructure
- Automatiser le déploiement et les test via CI/CD

Le projet sera réalisé en équipes de 5 personnes maximum

Les livrables

Chaque équipe projet devra rendre:

- Spécification logicielle
    - Use cases
    - Architecture
    - Structure des données
- Spécification de l’architecture
    - Configuration des serveurs
    - Politique de sécurité
    - Détails des flux entre les systèmes
    - Gestion du code source et contrôle de version
        - Un dépôt git par système/sous système
            - branches master/develop
        - Pipeline CI/CD
            - Build
            - Test
            - Deploy
    - Compte rendu de réalisation
        - Document académique relatant vos choix et décrivant le déroulement de votre projet

        - Inclure les livrables décrits en annexe

Cahier des charges 1/5

L’annuaire sécurisé proposera ces fonctionnalités:

- Outils d'administration
    - Ajouter manuellement un utilisateur
    - Changer un mot de passe
    - Promouvoir/Rétrograder un utilisateur
    - Supprimer un utilisateur
- Service d’annuaire
    - Lister les utilisateurs du système
    - Seul un utilisateur connecté peut lister les autres utilisateurs
    - Recherche basique

    L’annuaire sécurisé proposera ces fonctionnalités:

    - Compte d’utilisateur
        - Inscription
        - Mise à jour du profil
        - Connexion
        - Déconnexion
        - Clôture du compte

    Contraintes de sécurité:

    - La base de données doit être sécurisée
    - Les mots de passe doivent être protégés
    - Le design doit respecter
        - Les principes du moindre privilège
        - Les principes de défense en profondeur
        - Permettre la traçabilité des actions
    - Respecter les normes d’implémentation industrielles

- Le projet contient peu de fonctionnalités mais elles doivent être rigoureusement développés et explicités

▼ Cahier de charge :

Pour créer un système d'annuaire sécurisé en utilisant Django avec une base de données MySQL, suivez ces étapes de développement:

1. Spécification logicielle

Use Cases

1. Administrateur:
    - Ajouter un utilisateur
    - Changer le mot de passe d'un utilisateur
    - Promouvoir/Rétrograder un utilisateur
    - Supprimer un utilisateur
2. Service d’annuaire:
    - Liste des utilisateurs (accessible uniquement aux utilisateurs connectés)
    - Recherche basique
3. Compte d’utilisateur:
    - Inscription
    - Mise à jour du profil
    - Connexion
    - Déconnexion
    - Clôture du compte

Architecture

- Modèles Django pour les utilisateurs, les rôles, etc.
- Vues pour les différentes fonctionnalités
- URLConf pour gérer les routes

- Templates HTML pour l'interface utilisateur

Structure des données

1. Utilisateur (nom, prénom, email, mot de passe, rôle)
2. Rôle (administrateur, utilisateur)

2. Spécification de l’architecture

Configuration des serveurs

- Serveur Django avec une base de données MySQL
- Utilisation de l'ORM Django pour interagir avec la base de données

Politique de sécurité

- Utilisation de Django's built-in User pour la gestion des utilisateurs
- Utilisation d'authentification et d'autorisation pour restreindre l'accès aux fonctionnalités
- Cryptage des mots de passe avec des fonctions de hachage sécurisées
- Utilisation de HTTPS pour sécuriser les communications

Détails des flux entre les systèmes

- Flux entre le frontend (HTML/CSS/JavaScript) et le backend (Django)
- Flux entre le backend et la base de données MySQL

3. Gestion du code source et contrôle de version

Dépôt Git

- Création d'un dépôt Git pour le projet
- Utilisation de branches master/develop pour le développement

CI/CD Pipeline

- Mise en place d'un pipeline CI/CD avec les étapes Build, Test, et Deploy
- Utilisation d'outils comme Jenkins, GitLab CI, ou GitHub Actions

4. Automatisation du déploiement

- Utilisation d'outils comme Docker pour containeriser l'application
- Utilisation d'outils d'orchestration comme Docker Compose
- Déploiement automatisé sur un serveur

5. Compte rendu de réalisation

- Document académique décrivant les choix architecturaux, les étapes de développement, et les défis rencontrés
- Inclusion des livrables spécifiés dans le cahier des charges

Contraintes de sécurité

- Utilisation de middleware Django pour la sécurité
- Configuration appropriée des paramètres de sécurité Django
- Sécurisation de la base de données (accès limité, configurations sécurisées)
- Utilisation de JWT (JSON Web Tokens) pour gérer les sessions utilisateur de manière sécurisée