

**Dédicaces**

A tous ceux qui nous ont éclairé le chemin de la connaissance.

A nos chers parents qui nous ont déployé tous les moyens moraux et matériels pour faire de nous ce que nous sommes devenu.

A nos chers enseignants pour leurs remarques qui nous ont permis de mener cette mémoire à bon terme.

A vous cher lecteur.

**Remerciements**

Avant d’entamer ce rapport, nous profitons de l’occasion pour exprimer nos profonds remerciements à toutes les personnes qui ont contribués de près ou de loin à la réalisation et l’élaboration de ce modeste rapport.

Nous tenons à remercier infiniment l’ensemble du personnel de l’Ecole Marocaine des Sciences de l’Ingénieure et spécialement Mr ABOUABID Hamza notre enseignant de Python et Frameworks, Mr BOURAHI Tarik, Mr ERRAJI Naoufal Responsable de la filière et Mme BADRITIJANE Fatimazohra chef de département pour le soutien qu’ils nous ont apporté. Nos surveillants Mr ATTAR Jalal, Mr BENHMAMA Hicham, Mr BOUTRABA Ayoub et Mr JOBAIR Morad.

Jamais nous saurons présenter ce travail sans adresser nos profonds et sincères remerciements à tous nos chers enseignants.

**Table des matières**

**Introduction générale1**

**Chapitre 1 : Analyse des Besoins et Spécifications2**

**1. Introduction3**

**2.Analyse Détaillée du Cahier des Charges3**

**3. Fonctionnalités Attendues 3**

**3. 1. Gestion des utilisateurs3**

**3. 2. Gestion des missions 3**

**3. 3. Recherche et inscription3**

**3. 4. Suivi et communication 4**

**3. 5. Administration 4**

**3. 6. Spécification des besoins non fonctionnels4**

**5. Présentation des cas d’utilisation5**

**5. 1. Présentation des acteurs5**

**5. 2. Description des cas d’utilisation5**

**5. 3. Diagramme des cas d’utilisation global7**

**Chapitre 2 : Conception de l'Application 9**

**1. Introduction10**

**2. Modélisation dynamique10**

**2. 1. Diagramme de séquence10**

**2. 2. Diagramme d’état-transition11**

**2. 3. Diagrammes d’activité12**

**3. Modélisation statique14**

**3. 1. Diagramme de classes14**

**3. 2. Modèle relationnel16**

**3. 3. Dictionnaire de données 16**

**3. 4. Description des Principaux Modules et Composants 18**

**4. Architecture logiciel18**

**5. Conclusion19**

**Chapitre 3 : Réalisation et Développement20**

**1. Introduction21**

**2. Environnement Logiciel21**

**3. Présentation du Code Source22**

**3. 1. Structure des Dossiers22**

**3. 2. Principaux Fichiers23**

**3. Conclusion24**

**Conclusion générale25**

**Annexe A : Extrait des codes26**

**Annexe B : Extrait des fonctionnalités de l’application29**

**Liste des figures**

**Figure 1 : Diagramme de cas d’utilisation 8**

**Figure 2 : Diagramme de séquence système10**

**Figure 3 : Diagramme d’état-transition11**

**Figure 4 : Diagramme d’activité13**

**Figure 5 : Diagramme de classes15**

**Figure 6 : Structure des dossiers22**

**Figure 7 : Extrait forms.py26**

**Figure 8 : Extrait views.py26**

**Figure 9 : Extrait models.py27**

**Figure 10 : Extrait urls.py28**

**Figure 11 : Acceuil29**

**Figure 12 : Missions29**

**Figure 13 : Modérations Missions30**

**Figure 14 : Utilisateurs30**

**Figure 15 : API31**

**Liste des tableaux**

**Tableau 1 : Dictionnaire de données17**

**Introduction générale**

Dans le contexte actuel, les associations rencontrent des difficultés croissantes pour mobiliser des bénévoles, tandis que de nombreux citoyens souhaitent s'engager dans des actions solidaires mais ne disposent pas d'une plateforme centralisée et efficace pour trouver des missions adaptées. Selon les données récentes, le nombre d'associations actives est en constante augmentation, mais la participation bénévole stagne, voire diminue dans certaines régions. Face à ce constat, le projet vise à développer une plateforme web innovante, baptisée "Missions Bénévolat," permettant de faciliter la mise en relation entre les associations et les bénévoles, en optimisant la recherche et la gestion des missions.

La problématique centrale à laquelle ce projet tente de répondre est la suivante : Comment concevoir et implémenter une plateforme web qui optimise la mise en relation entre les associations et les bénévoles, en tenant compte de leurs besoins spécifiques et de leurs contraintes, tout en assurant une expérience utilisateur intuitive et engageante ?

Ce rapport de projet détaillera les différentes étapes de la conception et du développement de la plateforme "Missions Bénévolat." Dans un premier temps, nous présenterons une analyse approfondie des besoins et des spécifications fonctionnelles, en nous basant sur le cahier des charges. Ensuite, nous décrirons l'architecture de l'application, en mettant en évidence les choix technologiques et les principaux modules implémentés. La troisième partie sera consacrée à la présentation des fonctionnalités de la plateforme, en illustrant leur fonctionnement à travers des exemples concrets. Nous aborderons également les aspects liés aux tests et à la validation, en détaillant la stratégie de tests mise en œuvre et les résultats obtenus. Enfin, nous conclurons en dressant un bilan du projet, en identifiant les points forts et les points faibles, et en proposant des perspectives d'amélioration future.



1. **Introduction**

L'analyse des besoins et des spécifications est une étape cruciale dans le cycle de développement d'un projet informatique. Elle permet de comprendre en profondeur les exigences des utilisateurs et de définir les fonctionnalités et les caractéristiques techniques de la solution à mettre en œuvre. Cette phase garantit que le produit final répondra aux attentes des parties prenantes et atteindra les objectifs fixés.

1. **Analyse Détaillée du Cahier des Charges**

Le cahier des charges précise les fonctionnalités clés de la plateforme, qui vise à faciliter la mise en relation entre associations et bénévoles. Il inclut une gestion complète des utilisateurs avec des profils différenciés et un système d’inscription sécurisé, une gestion dynamique des missions permettant leur création, modification et catégorisation, ainsi qu’un moteur de recherche avancé avec filtres adaptés aux compétences et disponibilités. La plateforme intègre également des outils de suivi et de communication, tels que la messagerie interne, les notifications et l’évaluation post-mission, tout en offrant un volet administratif robuste pour la modération, la gestion des signalements et l’analyse statistique. Ces fonctionnalités garantissent une expérience utilisateur fluide et une administration efficace, conformément aux objectifs pédagogiques et techniques définis.

1. **Fonctionnalités Attendues**
   1. **Gestion des utilisateurs**

* Inscription/connexion des utilisateurs
* Profils différenciés : bénévoles et associations
* Tableau de bord personnalisé
* Vérification des organisations (modération)
  1. **Gestion des missions**
* Création, modification, suppression de missions
* Informations : titre, description, lieu, date, compétences requises
* Catégorisation par type d’activité et cause
* Upload de photos et documents d’information
  1. **Recherche et inscription**
* Recherche par critères (lieu, date, type d’activité)
* Filtrage avancé par compétences et disponibilités
* Système d’inscription aux missions
* Liste d’attente pour missions complètes
  1. **Suivi et communication**
* Historique des participations
* Messagerie entre associations et bénévoles
* Notifications et rappels
* Évaluation post-mission
  1. **Administration**
* Modération des associations et missions
* Statistiques de participation
* Gestion des signalements
* Mise en avant de missions urgentes

1. **Spécification des besoins non fonctionnels**

La spécification des besoins non fonctionnels pour ce projet de plateforme de mise en relation entre associations et bénévoles porte sur plusieurs aspects essentiels garantissant la qualité, la robustesse et la pérennité du système. Elle inclut notamment des exigences de performance afin d’assurer une navigation fluide et un temps de réponse rapide, même en cas de forte affluence d’utilisateurs simultanés, ce qui est crucial pour maintenir une expérience utilisateur optimale. La sécurité est également un point central, avec des mécanismes stricts d’authentification, de protection des données personnelles des bénévoles et des associations, ainsi qu’une gestion rigoureuse des accès pour prévenir toute intrusion ou manipulation non autorisée.

L’ergonomie et l’accessibilité sont aussi prises en compte, avec une interface intuitive et responsive, compatible avec différents appareils (ordinateurs, tablettes, smartphones) et navigateurs, afin de toucher un large public et faciliter l’usage par des profils variés. La maintenabilité du code est prévue pour permettre des évolutions futures aisées, en s’appuyant sur une architecture modulaire conforme au modèle MVT de Django, facilitant la correction rapide des bugs et l’ajout de nouvelles fonctionnalités. Enfin, la disponibilité et la fiabilité du service sont garanties par une infrastructure stable, minimisant les interruptions et assurant la continuité des opérations, notamment pour la gestion des missions urgentes et les notifications en temps réel. Ces spécifications non fonctionnelles sont indispensables pour répondre aux objectifs pédagogiques et techniques du projet tout en assurant une plateforme performante, sécurisée et évolutive.

1. **Présentation des cas d’utilisation**
   1. **Présentation des acteurs**

Le système de bénévolat implique trois acteurs principaux, chacun avec des rôles et des responsabilités distinctes :

* **Utilisateur (Bénévole) :** Les bénévoles s'inscrivent et se connectent à la plateforme, créent et gèrent leur profil, recherchent des missions qui correspondent à leurs compétences et intérêts, s'inscrivent à ces missions, consultent leur historique de participation, communiquent avec les associations via la messagerie, évaluent les missions auxquelles ils ont participé, signalent les problèmes ou abus, et consultent des statistiques sur leurs activités de bénévolat.
* **Association :** Les associations s'inscrivent et se connectent, gèrent leur profil d'organisation, créent, modifient et suppriment des missions, téléchargent des documents pertinents, utilisent la messagerie pour communiquer avec les bénévoles, consultent leur tableau de bord avec des informations sur leurs missions et leurs bénévoles, consultent les évaluations des missions qu'elles ont proposées, gèrent les signalements et accèdent aux statistiques relatives à leurs activités sur la plateforme.
* **Administrateur :** L'administrateur modère les associations et les missions pour assurer la qualité et la conformité, gère les signalements, met en avant les missions urgentes, vérifie les organisations, évalue la plateforme et peut consulter l'API.
  1. **Description des cas d’utilisation**

Le diagramme de cas d'utilisation représente les interactions entre les acteurs et le système, décrivant les actions que chaque acteur peut effectuer :

* **Inscription/Connexion :** Permet aux utilisateurs (bénévoles et associations) de créer un compte et de se connecter à la plateforme.
* **Créer Profil :** Permet aux utilisateurs de créer et de gérer leur profil avec des informations pertinentes.
* **Rechercher Missions :** Permet aux bénévoles de rechercher des missions en fonction de divers critères tels que le lieu, la date, le type d'activité et les compétences requises.
* **S'inscrire à une Mission :** Permet aux bénévoles de s'inscrire à une mission qui les intéresse.
* **Gérer Historique :** Permet aux bénévoles de consulter leur historique de participation aux missions.
* **Envoyer/Répondre à un Message :** Facilite la communication entre les bénévoles et les associations.
* **Tableau de Bord Association :** Fournit aux associations une vue d'ensemble de leurs missions, de leurs bénévoles et d'autres informations pertinentes.
* **Evaluation Post-Mission :** Permet aux bénévoles d'évaluer les missions auxquelles ils ont participé, et aux administrateurs d'évaluer la plateforme.
* **Signaler un Problème/Abus :** Permet aux utilisateurs de signaler tout problème ou abus sur la plateforme.
* **Consulter Statistiques :** Permet aux utilisateurs et aux administrateurs de consulter des statistiques sur l'utilisation de la plateforme.
* **Créer/Modifier/Supprimer une Mission :** Permet aux associations de gérer leurs missions.
* **Uploader des Documents :** Permet aux associations de télécharger des documents pertinents pour leurs missions.
* **Vérifier Organisation :** Permet à l'administrateur de vérifier les informations des associations.
* **Modérer Associations et Missions :** Permet à l'administrateur de modérer le contenu de la plateforme.
* **Gérer Signalements :** Permet à l'administrateur de gérer les signalements d'abus.
* **Mettre en Avant Missions Urgentes :** Permet à l'administrateur de mettre en avant les missions urgentes.
* **Consulter Evaluations :** Permet aux Associations et Administrateurs de consulter les évaluations.
* **Consulter API :** Permet aux administrateurs de consulter l'API.

Ces cas d'utilisation assurent que la plateforme répond aux besoins des bénévoles, des associations et de l'administrateur, en facilitant la mise en relation, la gestion des missions et le suivi des activités de bénévolat.

* 1. **Diagramme des cas d’utilisation global**

 Il offre une vue d'ensemble des interactions entre les acteurs (Utilisateur, Association, Administrateur) et le système de bénévolat. Il illustre les principales fonctionnalités offertes par la plateforme, telles que l'inscription, la recherche de missions, la gestion des profils, etc., et montre comment chaque acteur interagit avec ces fonctionnalités.

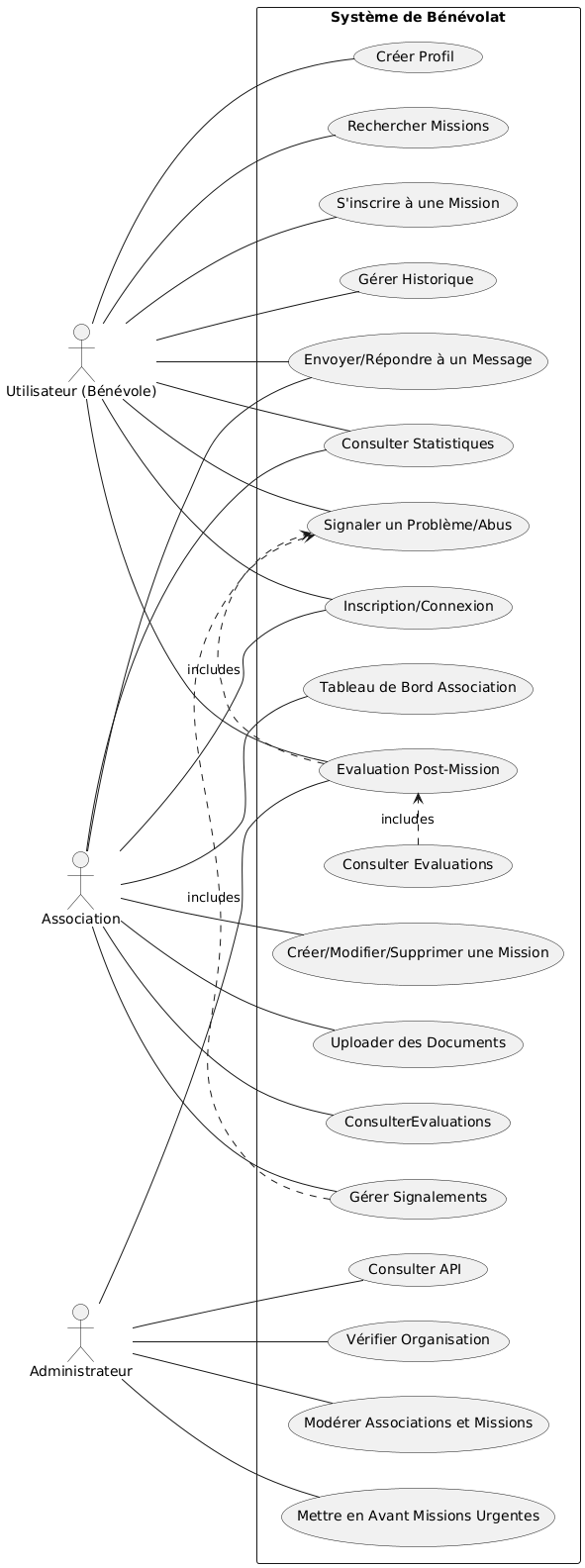


Figure 1 : Diagramme de cas d'utilisation



1. **Introduction**

La conception de l'application est une étape cruciale qui transforme les besoins et les spécifications définis précédemment en un plan détaillé de la solution logicielle. Cette phase consiste à définir l'architecture de l'application, à concevoir le modèle de données, à élaborer l'interface utilisateur et à choisir les technologies appropriées pour atteindre les objectifs fixés. Une conception bien pensée est essentielle pour garantir la robustesse, la maintenabilité, l'évolutivité et la convivialité de l'application.

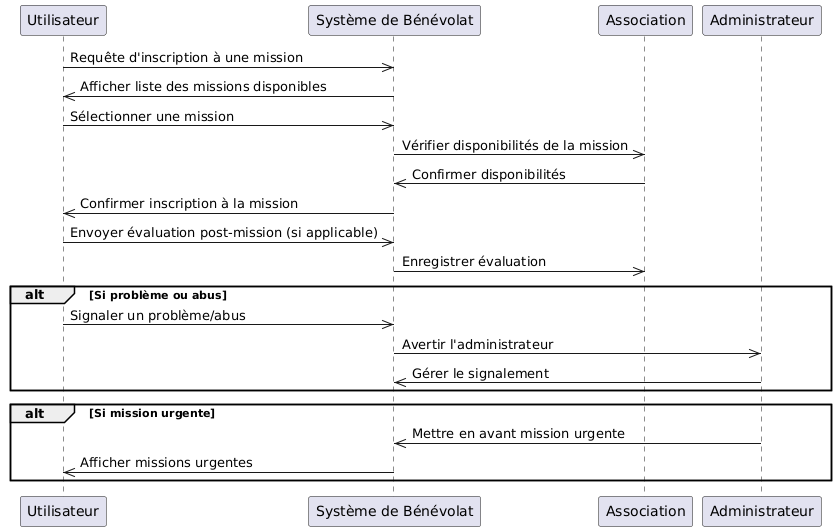
1. **Modélisation dynamique** 
   1. **Diagramme de séquence**

Figure 2 : Diagramme de séquences système

Il met en évidence la séquence des interactions entre l'Utilisateur et le Système pour un cas d'utilisation spécifique, comme l'inscription à une mission. Il illustre les messages échangés entre l'Utilisateur et le Système, ainsi que les opérations réalisées par le Système pour répondre à la requête de l'Utilisateur.

* 1. **Diagramme d’état-transition**

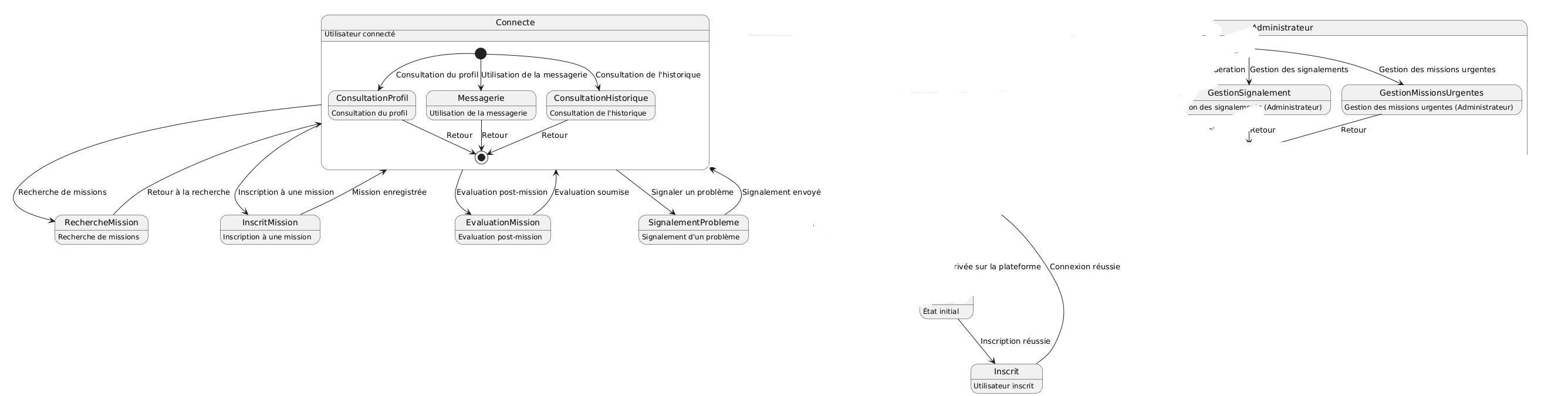
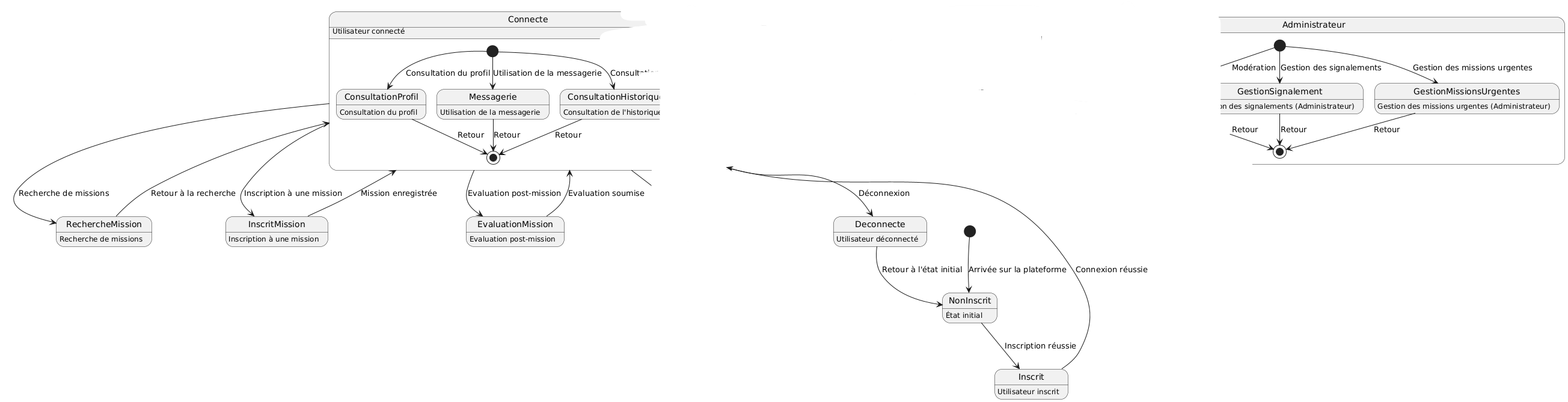
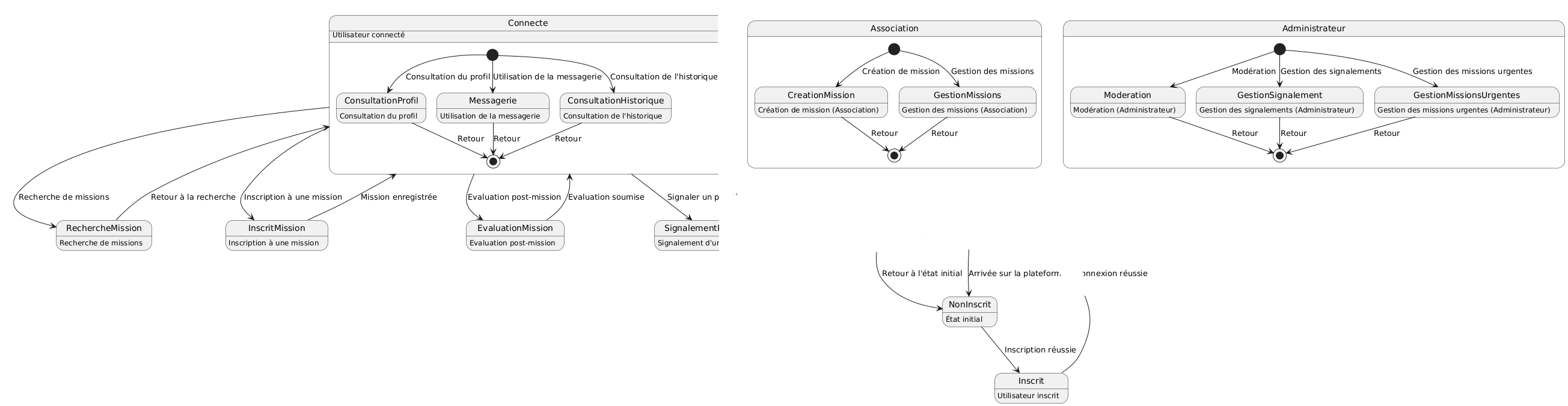
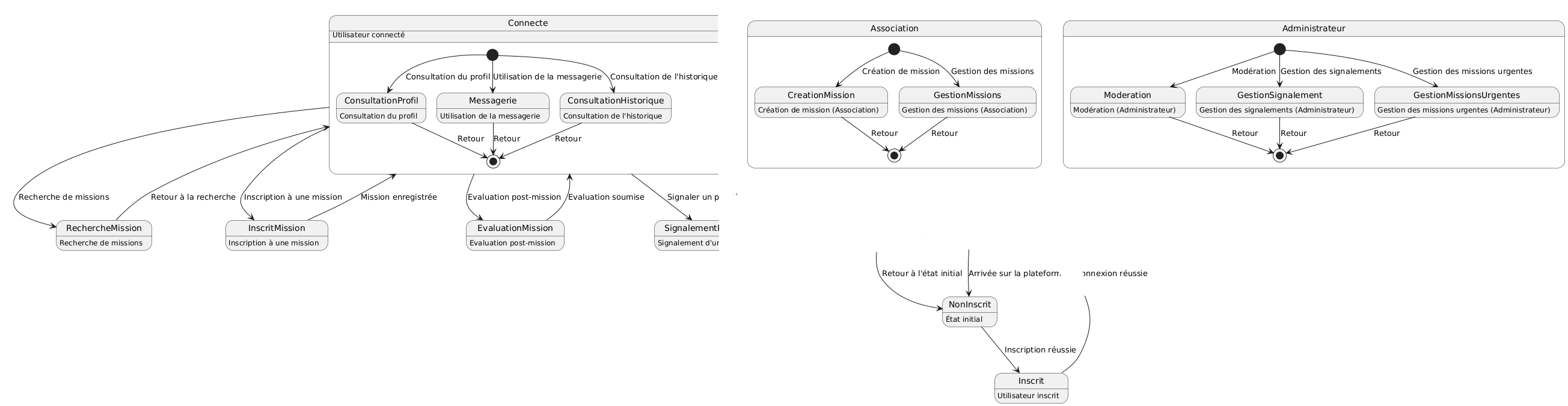
Il modélise le comportement d'un objet (par exemple, un utilisateur) en décrivant les différents états possibles de cet objet et les transitions entre ces états. Il permet de visualiser les différentes étapes par lesquelles passe un objet au cours de son cycle de vie et les événements qui déclenchent ces transitions.

Figure 3 : Diagramme d'état-transition

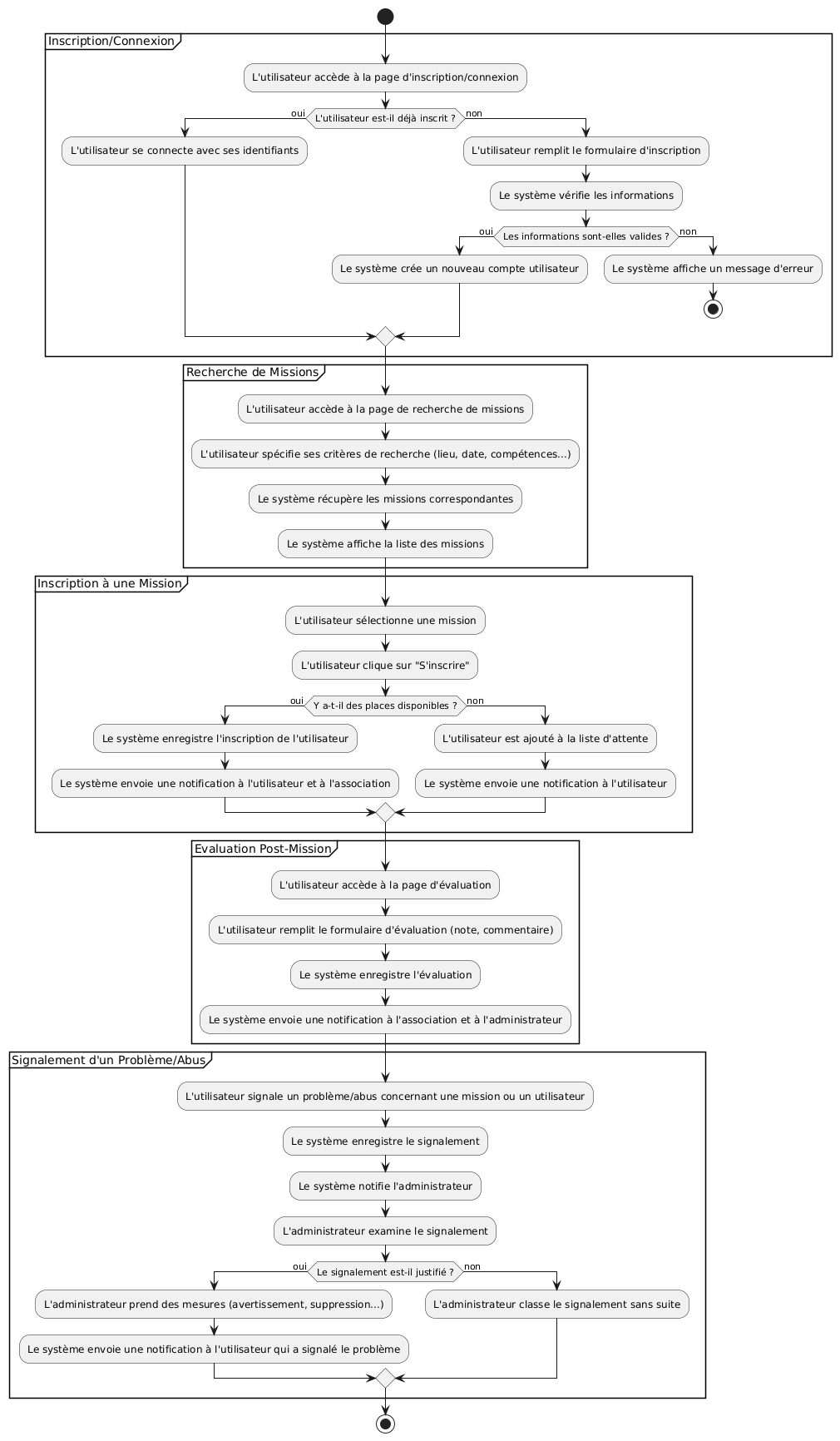




* 1. **Diagrammes d’activité**

Il décrit le flux d'activité pour un processus spécifique, comme l'inscription d'un bénévole à une mission. Il illustre les différentes étapes du processus, les décisions à prendre et les actions à réaliser, en mettant en évidence le déroulement des opérations et les alternatives possibles.

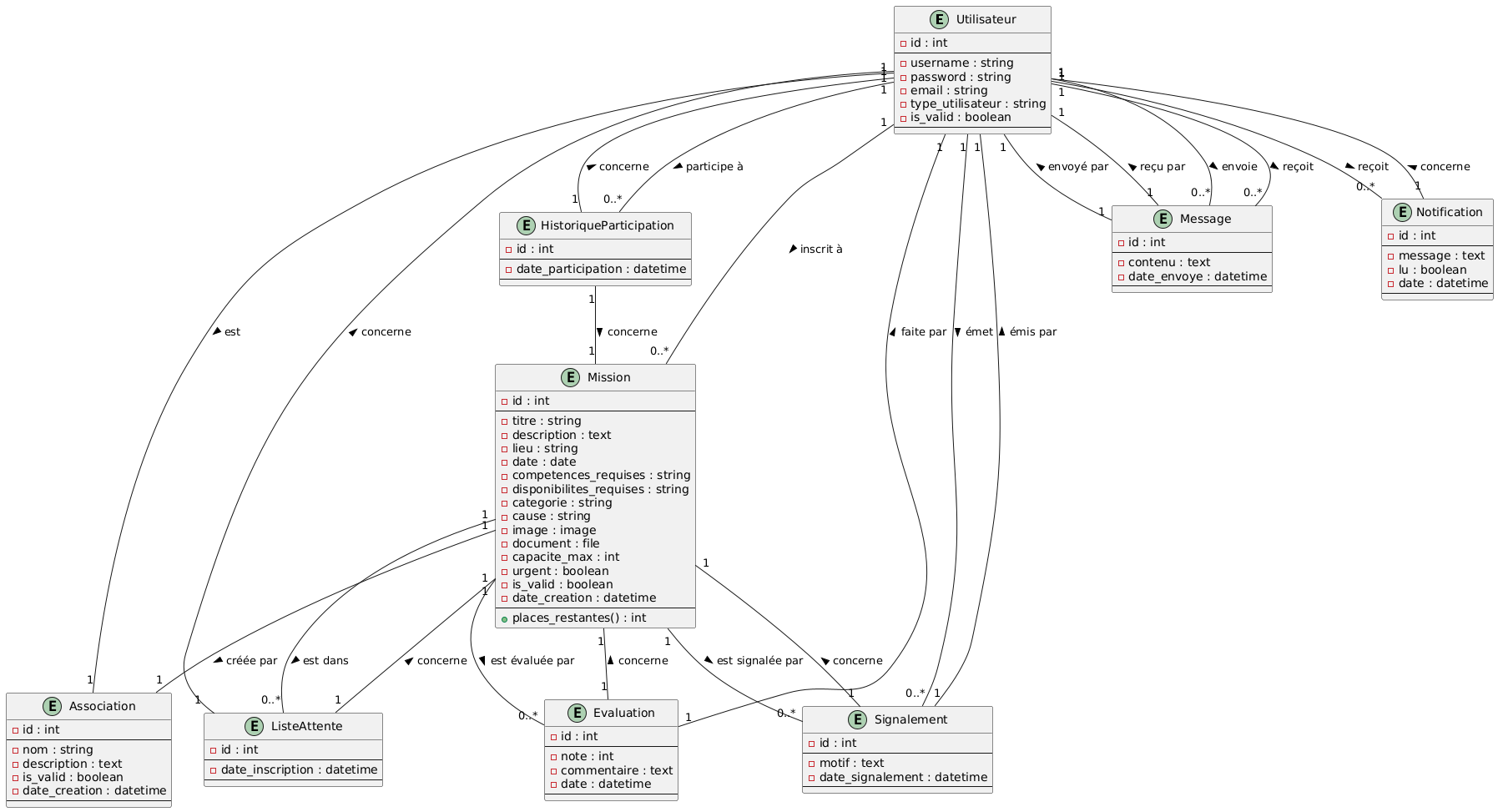
Figure 4 : Diagramme d'activité



1. **Modélisation statique**
   1. **Diagramme de classes**

Il représente la structure statique du système en décrivant les classes, leurs attributs et leurs relations. Il permet de visualiser les entités importantes du système, telles que Utilisateur, Mission, Association, et de comprendre comment elles sont liées entre elles.

Figure 5 : Diagramme de classes



* 1. **Modèle relationnel**

Le modèle relationnel de votre application repose principalement sur la gestion des utilisateurs, des missions proposées par des associations, ainsi que sur la participation des bénévoles à ces missions. La table Utilisateurs stocke les informations des différents types d’utilisateurs (bénévoles ou associations). Chaque mission est créée par une association et contient des informations détaillées telles que le titre, la description, la date, la capacité maximale, etc. Les bénévoles peuvent s’inscrire aux missions via une relation ManyToMany, et s’ils sont en liste d’attente, cela est géré par une table spécifique. Des évaluations, signalements et historiques de participation permettent de suivre la qualité et l’activité des missions. Enfin, un système de messagerie et de notifications assure la communication entre utilisateurs.

* Utilisateurs (id\_utilisateur, type\_utilisateur, username, password, email, is\_valid, first\_name, last\_name, téléphone, adresse)
* Missions (id\_mission, id\_association, titre, description, lieu, date, competences\_requises, disponibilites\_requises, categorie, cause, image, document, capacite\_max, urgent, is\_valid, date\_creation)
* Missions\_Inscrits (id\_mission, id\_utilisateur)
* ListeAttente (id\_liste\_attente, id\_mission, id\_benevole, date\_inscription)
* Evaluation (id\_evaluation, id\_mission, id\_benevole, note, commentaire, date)
* Signalement (id\_signalement, id\_mission, id\_utilisateur, motif, date\_signalement)
* HistoriqueParticipation (id\_historique, id\_utilisateur, id\_mission, date\_participation)
* Message (id\_message, id\_expediteur, id\_destinataire, contenu, date\_envoye)
* Notification (id\_notification, id\_utilisateur, message, lu, date)
  1. **Dictionnaire de données**

Tableau 1 : Dictionnaire de données

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Attribut | Type | Contraintes / Description |
| Utilisateurs | id\_utilisateur | INT (PK, auto-incr) | Identifiant unique utilisateur |
|  | type\_utilisateur | VARCHAR(20) | 'benevole' ou 'association' |
|  | username | VARCHAR | Nom d'utilisateur (unique) |
|  | password | VARCHAR | Mot de passe hashé |
|  | email | VARCHAR | Email utilisateur (unique) |
|  | is\_valid | BOOLEAN | Validation du compte |
|  | first\_name | VARCHAR | Prénom utilisateur |
|  | last\_name | VARCHAR | Nom utilisateur |
|  | téléphone | VARCHAR (optionnel) | Numéro de téléphone (non défini dans le modèle original) |
|  | adresse | VARCHAR (optionnel) | Adresse (non défini dans le modèle original) |
| Missions | id\_mission | INT (PK, auto-incr) | Identifiant unique mission |
|  | titre | VARCHAR(255) | Titre de la mission |
|  | description | TEXT | Description détaillée |
|  | lieu | VARCHAR(255) | Lieu de la mission |
|  | date | DATE | Date de la mission |
|  | competences\_requises | VARCHAR(255) | Compétences demandées (optionnel) |
|  | disponibilites\_requises | VARCHAR(255) | Disponibilités demandées (optionnel) |
|  | categorie | VARCHAR(100) | Catégorie de la mission |
|  | cause | VARCHAR(100) | Cause soutenue |
|  | image | VARCHAR (chemin) | Chemin vers l’image (optionnel) |
|  | document | VARCHAR (chemin) | Chemin vers le document (optionnel) |
|  | id\_association | INT (FK) | Référence à l’association (Utilisateur de type 'association') |
|  | capacite\_max | INT | Capacité maximale de bénévoles |
|  | urgent | BOOLEAN | Indique si la mission est urgente |
|  | is\_valid | BOOLEAN | Validation de la mission |
|  | date\_creation | DATETIME | Date de création automatique |
| Missions\_Inscrits | id\_mission | INT (FK) | Référence à une mission |
|  | id\_utilisateur | INT (FK) | Référence à un utilisateur inscrit |
|  |  |  | PK composée (id\_mission, id\_utilisateur) |
| ListeAttente | id\_liste\_attente | INT (PK, auto-incr) | Identifiant unique |
|  | id\_mission | INT (FK) | Référence à la mission |
|  | id\_benevole | INT (FK) | Référence au bénévole |
|  | date\_inscription | DATETIME | Date d’inscription en liste d’attente |
|  |  |  | UNIQUE(id\_mission, id\_benevole) |
| Evaluation | id\_evaluation | INT (PK, auto-incr) | Identifiant unique |
|  | id\_mission | INT (FK) | Référence à la mission |
|  | id\_benevole | INT (FK) | Référence au bénévole |
|  | note | INT | Note attribuée (ex: 1 à 5) |
|  | commentaire | TEXT | Commentaire de l’évaluation |
|  | date | DATETIME | Date de l’évaluation |
| Signalement | id\_signalement | INT (PK, auto-incr) | Identifiant unique |
|  | id\_mission | INT (FK) | Mission concernée |
|  | id\_utilisateur | INT (FK) | Utilisateur ayant signalé |
|  | motif | TEXT | Motif du signalement |
|  | date\_signalement | DATETIME | Date du signalement |
| HistoriqueParticipation | id\_historique | INT (PK, auto-incr) | Identifiant unique |
|  | id\_utilisateur | INT (FK) | Utilisateur participant |
|  | id\_mission | INT (FK) | Mission concernée |
|  | date\_participation | DATETIME | Date de participation |
| Message | id\_message | INT (PK, auto-incr) | Identifiant unique |
|  | id\_expediteur | INT (FK) | Utilisateur expéditeur |
|  | id\_destinataire | INT (FK) | Utilisateur destinataire |
|  | contenu | TEXT | Contenu du message |
|  | date\_envoye | DATETIME | Date d’envoi |
| Notification | id\_notification | INT (PK, auto-incr) | Identifiant unique |
|  | id\_utilisateur | INT (FK) | Utilisateur destinataire |
|  | message | TEXT | Message de notification |
|  | lu | BOOLEAN | Indique si la notification a été lue |
|  | date | DATETIME | Date de création de la notification |

* 1. **Description des Principaux Modules et Composants**
* Module Utilisateurs : Gestion des inscriptions, des profils (bénévoles et associations), de l’authentification et de la modération.
* Module Missions : Création, modification, suppression, catégorisation des missions, gestion des compétences requises et des documents associés.
* Module Recherche : Moteur de recherche avancé avec filtrage par critères (lieu, date, type d’activité, compétences, disponibilités).
* Module Communication : Messagerie interne, système de notifications et de rappels, gestion des évaluations post-mission.
* Module Administration : Modération des contenus, gestion des signalements, statistiques de participation, mise en avant des missions urgentes.

1. **Architecture logicielle**

L'architecture de la plateforme "Missions Bénévolat" est basée sur le modèle Modèle-Vue-Template (MVT) de Django, qui permet de séparer les préoccupations et de structurer l'application en trois couches distinctes :

* Modèles (Models) : Les modèles définissent la structure des données et les règles métier. Ils représentent les entités de l'application, telles que les utilisateurs (bénévoles et associations), les missions, les compétences, les inscriptions, etc. Les modèles sont implémentés en utilisant l'ORM (Object-Relational Mapping) de Django, qui permet d'interagir avec la base de données MySQL de manière abstraite.
* Vues (Views) : Les vues gèrent la logique de l'application et interagissent avec les modèles pour récupérer et manipuler les données. Elles reçoivent les requêtes HTTP, effectuent les traitements nécessaires et renvoient une réponse HTTP, qui peut être un template HTML ou un code d'état (par exemple, pour une API).
* Templates : Les templates définissent l'interface utilisateur de l'application. Ils sont écrits en HTML et peuvent contenir des balises Django pour afficher les données dynamiques et exécuter des instructions de contrôle (par exemple, des boucles et des conditions).

En plus du modèle MVT, l'architecture de l'application comprend les composants suivants :

* Contrôleurs d'URL (URL Dispatcher) : Les contrôleurs d'URL mappent les URL aux vues appropriées. Ils permettent de définir des URL conviviales et de structurer l'application en modules logiques.
* Formulaires (Forms) : Les formulaires facilitent la création et la validation des données saisies par les utilisateurs. Ils sont implémentés en utilisant Django Forms et permettent de gérer les erreurs de saisie et de garantir la cohérence des données.
* Authentification (Authentication) : Le système d'authentification de Django est utilisé pour gérer les utilisateurs, les mots de passe et les permissions. Il permet de sécuriser l'application et de contrôler l'accès aux différentes fonctionnalités.
* Middleware : Les middleware sont des composants qui interceptent les requêtes et les réponses HTTP. Ils permettent de réaliser des traitements transversaux, tels que la gestion des sessions, la sécurité et la journalisation.
* Fichiers Statiques : Les fichiers statiques (CSS, JavaScript, images) sont servis par un serveur web dédié (par exemple, Nginx) pour améliorer les performances de l'application.

1. **Conclusion**

Ce chapitre a présenté les choix de conception clés pour la plateforme "Missions Bénévolat." Nous avons détaillé l'architecture de l'application basée sur le modèle MVT de Django, qui permet une séparation claire des responsabilités entre les modèles de données, les vues et les templates. Nous avons également décrit le modèle de données, en mettant en évidence les relations entre les différents entités (utilisateurs, missions, compétences, etc.). Enfin, nous avons présenté l'interface utilisateur, en expliquant les principes d'ergonomie et de convivialité qui ont guidé nos choix de conception. La phase de conception est essentielle pour assurer le succès du projet. Elle permet de traduire les besoins et les spécifications en un plan concret de la solution logicielle. Les choix de conception que nous avons effectués dans ce chapitre serviront de base pour la phase de développement, qui sera décrite dans le chapitre suivant.



1. **Introduction**

Ce chapitre décrit le processus de mise en œuvre de la plateforme "Missions Bénévolat", en partant des spécifications définies dans les chapitres précédents. Il détaille les étapes de développement, les outils utilisés, les choix techniques effectués et les difficultés rencontrées. L'objectif est de fournir une vue d'ensemble de la réalisation du système, en mettant en évidence les aspects clés de l'implémentation.

1. **Environnement Logiciel**

L'environnement de développement et de déploiement de la plateforme "Missions Bénévolat" est composé des éléments suivants :

* Système d'exploitation : Linux Mint 22.1.
* Langage de programmation : Python 3.12.3.
* Framework web : Django 5.1.7.
* Base de données : MariaDB.
* Serveur web : Le serveur de développement est intégré à Django (utilisé pour le développement local), mais pour le déploiement en production, un serveur web dédié comme Gunicorn ou Apache2 serait utilisé en conjonction avec Django.

Outils de développement :

* IDE : Visual Studio Code (VS Code).
* Gestionnaire de paquets : pip.
* Système de contrôle de version : Git (avec GitHub).
* Interface d'administration de la base de données : phpMyAdmin.

Bibliothèques et dépendances :

* Django Forms.
* Django Api Restful.
* Django Authentication.
* Bootstrap.
* JavaScript.

1. **Présentation du Code Source**
   1. **Structure des dossiers**

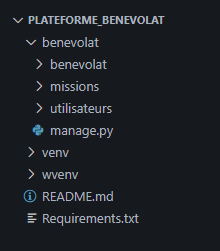
Notre projet Django est organisé selon une structure modulaire qui facilite la maintenance, la scalabilité et la collaboration. À la racine du projet, on trouve le fichier manage.py, qui sert d’interface en ligne de commande pour gérer le projet (lancement du serveur, migrations, etc.). Ce fichier est accompagné d’un dossier principal portant le nom du projet, qui contient les fichiers de configuration essentiels tels que settings.py (paramètres globaux), urls.py (routes principales), wsgi.py et asgi.py (interfaces serveur).

Les applications Django (apps) sont placées dans des dossiers distincts directement à la racine. Chaque app est une unité fonctionnelle autonome comprenant ses propres fichiers models.py (définition des données), views.py (logique métier), urls.py (routes spécifiques), forms.py (gestion des formulaires), ainsi que des dossiers templates et static pour les fichiers HTML, CSS, JavaScript et images.

Les dossiers templates peuvent être organisés par app pour éviter les conflits de noms et faciliter la réutilisation, en suivant la convention templates/app\_name/. De même, les fichiers statiques sont souvent regroupés dans un dossier static global ou répartis dans chaque app.

Cette organisation modulaire permet de séparer clairement les responsabilités, de faciliter les tests, et de rendre le projet plus lisible et maintenable, notamment dans des projets collaboratifs ou de grande taille.

Figure 6 : Structure des dossiers



* 1. **Principaux fichiers**
* views.py : Ce fichier contient la logique métier de votre application. Les fonctions ou classes définies dans views.py reçoivent les requêtes HTTP, traitent les données (par exemple, en interrogeant la base de données via les modèles), puis retournent une réponse, souvent sous forme de page HTML ou de données JSON. Les vues font le lien entre les modèles, les formulaires et les templates.
* urls.py : Dans urls.py, vous définissez les routes (URL patterns) de votre application. Ce fichier fait correspondre une URL à une vue spécifique, permettant ainsi de diriger chaque requête utilisateur vers la logique appropriée. Il peut exister un fichier urls.py au niveau du projet et dans chaque application Django, facilitant la modularité.
* forms.py : Ce fichier sert à définir des formulaires Django, utilisés pour la saisie et la validation de données côté utilisateur. Les classes de forms.py permettent de générer des formulaires HTML, de valider les entrées et de les transformer en objets Python utilisables dans vos vues ou modèles.
* settings.py : settings.py regroupe toutes les configurations globales du projet : base de données, chemins des fichiers statiques et médias, applications installées, paramètres de sécurité, internationalisation, etc. Modifier ce fichier permet d’adapter le comportement général de votre projet à différents environnements (développement, production, etc.).
* manage.py : C’est le script principal pour l’administration du projet Django. Il permet de lancer le serveur de développement, d’effectuer des migrations, de créer des utilisateurs administrateurs, et d’exécuter d’autres commandes utiles à la gestion du projet. Il sert d’interface en ligne de commande pour interagir avec l’application.
* serializer.py : Ce fichier est utilisé principalement dans les applications Django REST Framework. Il définit la façon dont les objets Python (modèles) sont convertis en formats de données (JSON, XML) pour les API, et inversement. Les serializers gèrent la validation et la transformation des données échangées via l’API.
* models.py : Dans models.py, vous définissez la structure des données de votre application sous forme de classes Python. Chaque classe correspond à une table de la base de données, et chaque attribut à une colonne. Les modèles permettent d’effectuer des opérations CRUD (création, lecture, mise à jour, suppression) sur les données de façon simple et cohérente.
* custom\_tags.py : Ce fichier contient des balises personnalisées pour le moteur de templates Django. Les “custom template tags” et “filters” permettent d’ajouter des fonctionnalités spécifiques dans les templates HTML, comme le formatage de données ou l’inclusion de logique conditionnelle, améliorant ainsi la réutilisabilité et la clarté des vues côté interface.

1. **Conclusion**

Ce chapitre a présenté les différentes étapes de la réalisation de la plateforme "Missions Bénévolat." Nous avons décrit l'environnement de développement et de déploiement, les fonctionnalités implémentées et le processus de déploiement. Les choix techniques que nous avons effectués, tels que l'utilisation de Django, de MariaDB et de Bootstrap, ont permis de développer une application robuste, performante et facile à maintenir. La phase de réalisation est essentielle pour transformer les spécifications en une application fonctionnelle. Les efforts que nous avons déployés dans ce chapitre ont permis de concrétiser la vision de la plateforme "Missions Bénévolat" et de la rendre accessible aux associations et aux bénévoles.

**Conclusion générale**

Ce projet a permis de développer une plateforme web complète, "Missions Bénévolat", visant à faciliter la mise en relation entre les associations et les bénévoles. En se basant sur le modèle MVT de Django, nous avons créé une application structurée, robuste et facile à maintenir.

Les objectifs pédagogiques initiaux ont été atteints, notamment la maîtrise du modèle MVT, l'implémentation d'un système avec différents types d'utilisateurs, la gestion de relations complexes entre modèles, la mise en place d'un système de recherche avancée et l'implémentation d'un système de notification. Les fonctionnalités essentielles définies dans le cahier des charges ont été implémentées avec succès, notamment la gestion des utilisateurs, la gestion des missions, la recherche et l'inscription, le suivi et la communication, ainsi que l'administration.

Les choix technologiques effectués, tels que l'utilisation de Django, de MariaDB, de Bootstrap et de JavaScript, ont permis de créer une application performante, sécurisée et conviviale. Malgré les difficultés rencontrées lors du développement, notamment la mise en place du système de recherche avancée et la gestion des erreurs de formulaire, nous avons su les surmonter grâce à une approche méthodique et à l'utilisation des ressources disponibles. Ce projet a permis de renforcer nos compétences en développement web avec Django, en gestion de base de données et en gestion de projet agile.

En conclusion, la plateforme "Missions Bénévolat" est une solution viable et efficace pour faciliter l'engagement citoyen et soutenir les associations à but non lucratif. Nous espérons que cette plateforme contribuera à renforcer le tissu social et à promouvoir les valeurs de solidarité et de bénévolat.

**Annexe A : Extrait de code**

Figure 7 : Extrait de forms.py

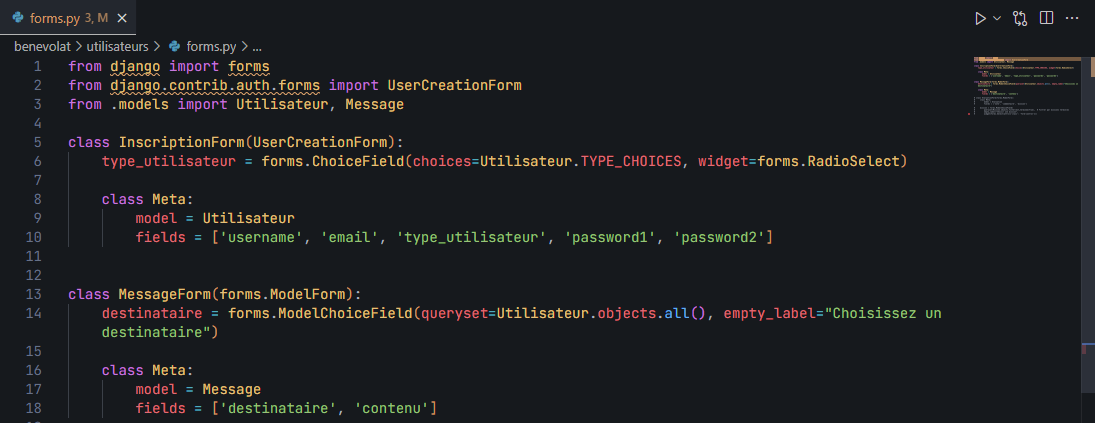
****

Figure 8 : Extrait de views.py

****

Figure 9 : Extrait de models.py

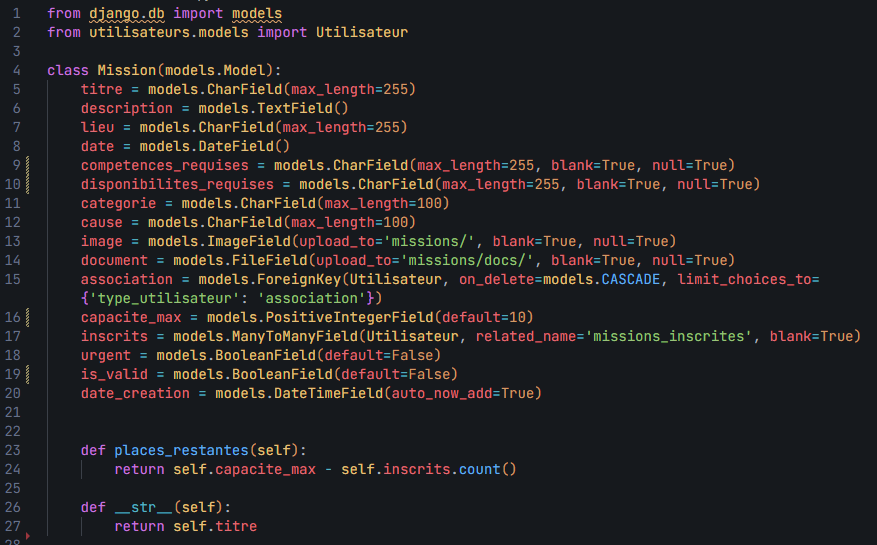
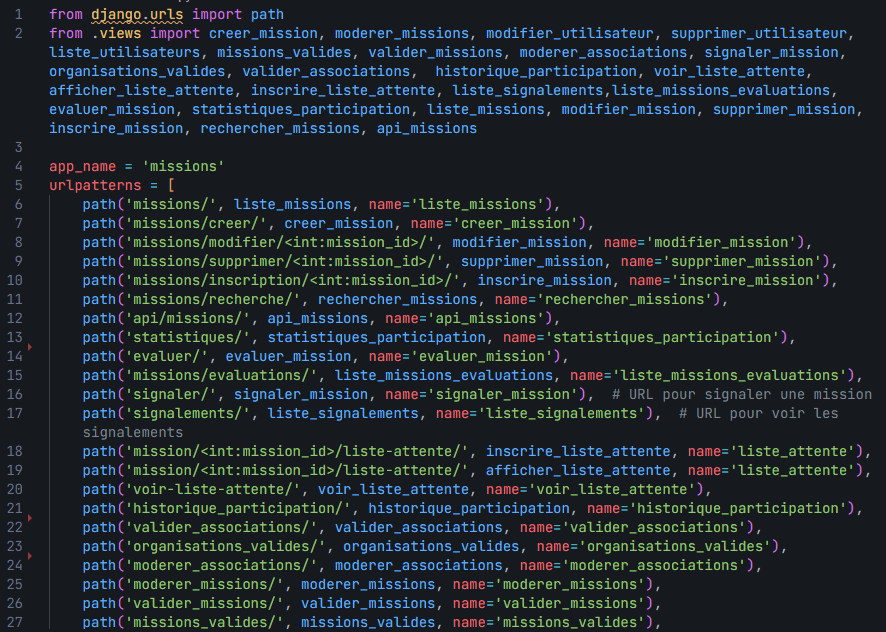
****

Figure 10 : Extrait de urls.py

****

**Annexe B : Extrait des Fonctionnalités de l’application**

Figure 11 : Accueil

****

Figure 12 : Missions

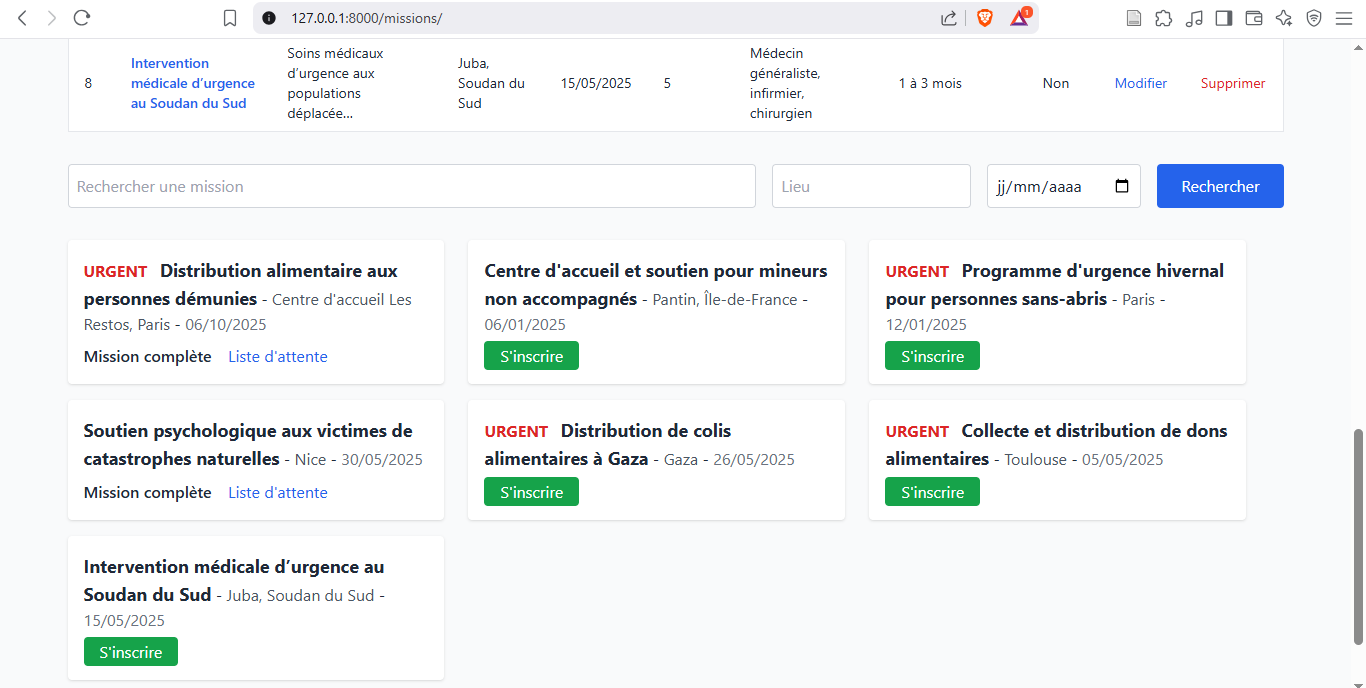
****

Figure 13 : Modération missions

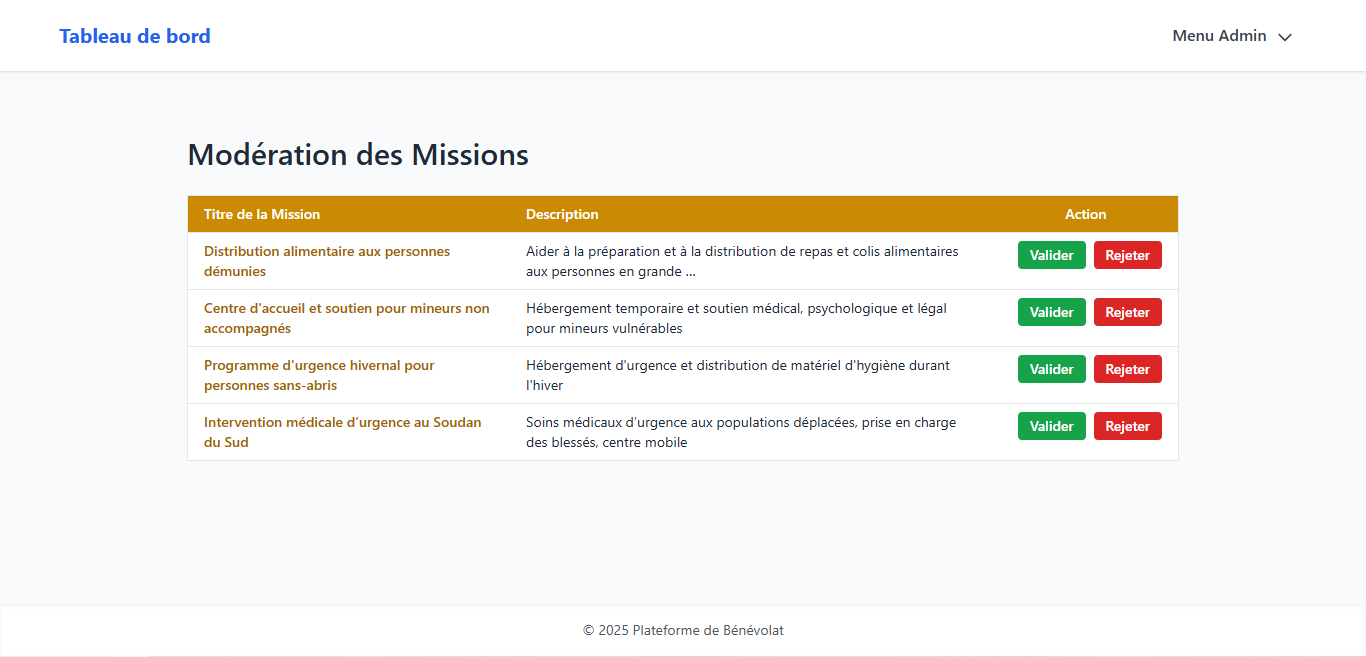
****

Figure 14 : Utilisateurs

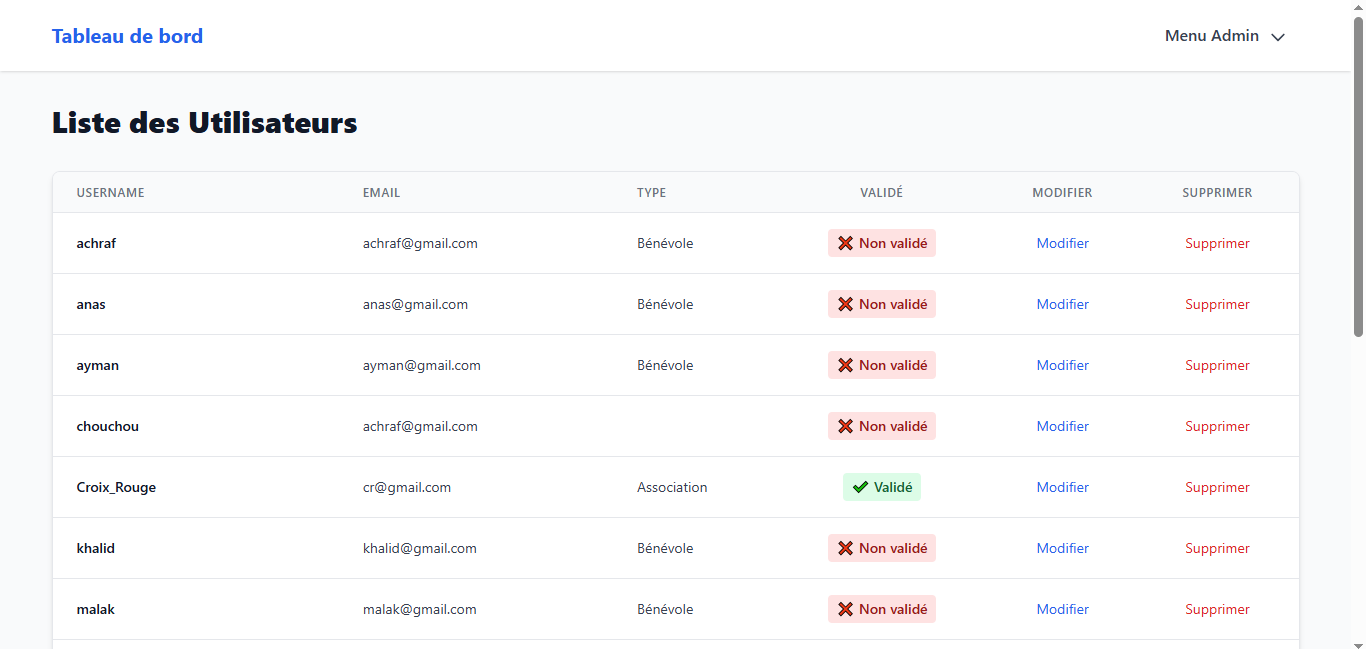
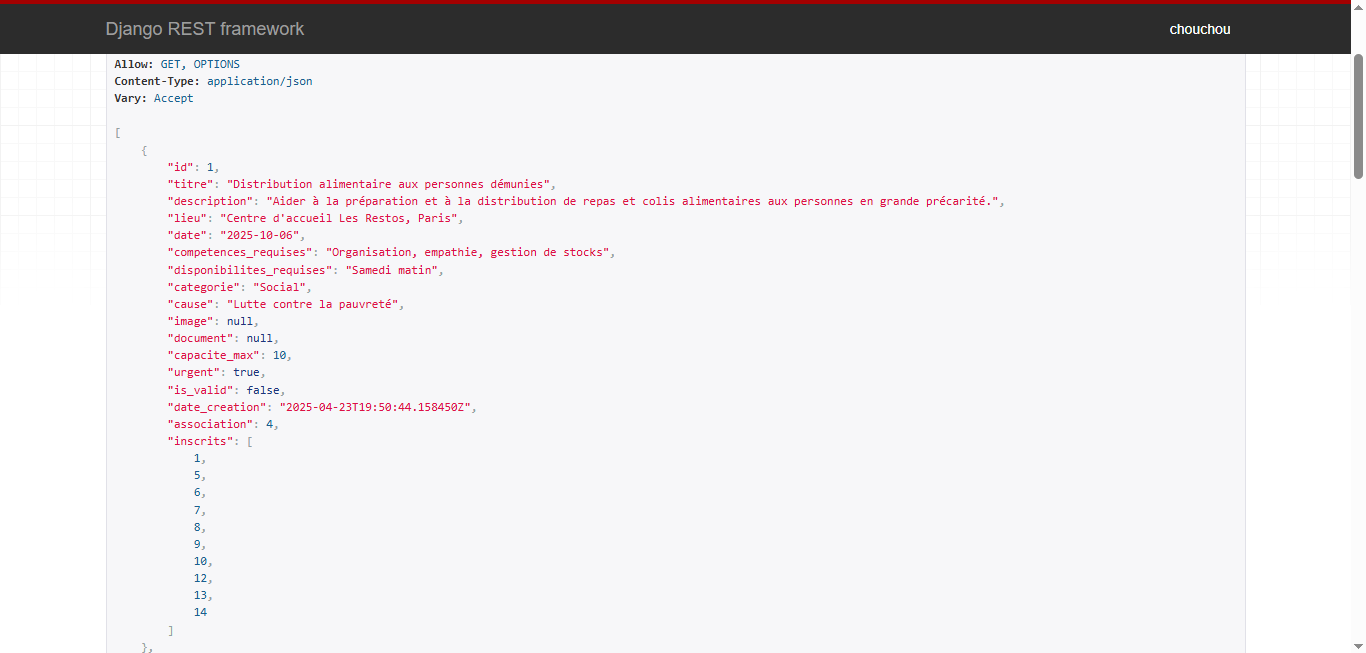
****

Figure 15 : API

****