Documentation interne

ASADI : ASsistant ADministratif Intelligent

Groupe:
L3W1

Encadrant:
David Janisek
Auteurs:
Djaffer Abdel Malik, Gourmelen Thomas, Ponnoussamy Valentin, Traore Ali

Version du document:
1.0

Résumé
Dans le cadre de l'UE Projet Informatique à l'université Paris Cité, nous devons par équipes de quatre mener un projet tout le long du semestre 6 sous la tutelle d'un encadrant. Ce projet vise à développer un assistant numérique capable de répondre automatiquement aux questions administratives, d'expliquer des procédures et d'offrir un apprentissage interactif via des scénarios immersifs, en s'appuyant sur une base documentaire centralisée.

Sommaire

1 Introduction

ASADI est une application web intelligente développée dans le cadre du projet de fin d'année de Licence 3 Informatique à l'Université Paris Cité. Imaginée et réalisée par le groupe L3W1, cette solution a pour ambition de moderniser et simplifier l'accès aux informations administratives souvent jugées complexes, obscures ou difficiles à naviguer.

ASADI s'appuie sur les avancées récentes de l'intelligence artificielle, notamment via la technologie RAG (Retrieval-Augmented Generation), qui combine des capacités de recherche documentaire avec la génération de texte. Cela permet à l'assistant d'apporter des réponses précises, contextualisées et sourcées, en s'appuyant sur une base de connaissances centralisée et maintenue à jour par les administrateurs.

L'utilisateur peut ainsi interagir de manière naturelle avec l'assistant pour poser ses questions, explorer des procédures administratives détaillées, ou encore approfondir ses connaissances à travers des scénarios immersifs et des quiz pédagogiques.

1.1 Objectifs fonctionnels et techniques

1.1.a Objectifs fonctionnels

L'application ASADI (ASsistant ADministratif Intelligent) vise à améliorer la compréhension et la gestion des démarches administratives à l'aide de l'intelligence artificielle. Les objectifs fonctionnels décrivent ce que le système doit faire d'un point de vue utilisateur :

1. Assistance aux démarches administratives
   - Répondre à des questions d'ordre administratif en langage naturel.
   - Fournir des réponses précises, contextualisées et accompagnées de sources vérifiables.
   - Permettre à l'utilisateur de spécifier un espace de travail pour affiner les réponses (ex : RH, Comptabilité).

2. Explication des procédures
   - Présenter de manière claire et interactive les étapes d'une procédure administrative.
   - Adapter les explications selon le profil ou la demande de l'utilisateur.

3. Apprentissage interactif
   - Proposer des quiz pour tester les connaissances de l'utilisateur.
   - Simuler des scénarios administratifs réalistes pour favoriser l'apprentissage par la pratique.
   - Générer un feedback personnalisé basé sur les performances.

4. Gestion documentaire
   - Permettre aux administrateurs d'ajouter, supprimer ou modifier les documents de la base de connaissances.
   - Organiser les documents par espaces de travail thématiques.

5. Gestion des comptes
   - Offrir un système d'inscription, de connexion, de réinitialisation de mot de passe.
   - Distinguer les droits entre utilisateurs simples et administrateurs.

1.1.b Objectifs techniques

Les objectifs techniques décrivent les contraintes et choix d'implémentation pour garantir performance, sécurité et maintenabilité :

1. Architecture modulaire Django
   - Utilisation du framework Django pour une structure MVC claire.
   - Séparation des modules en applications : documents, quiz, scenario, utilisateurs, workspace, etc.

2. Utilisation d'un moteur RAG (Retrieval-Augmented Generation)
   - Indexation des documents dans une base vectorielle.
   - Recherche sémantique via embeddings pour une réponse IA plus pertinente.
   - Génération de texte à l'aide d'un LLM (modèle de langage pré-entraîné).

3. Base de données vectorielle auto-hébergée
   - Utilisation de ChromaDB pour stocker et interroger les vecteurs.
   - Structuration de la base pour permettre des filtres par espace de travail.

4. Sécurité et droits d'accès
   - Authentification sécurisée avec mots de passe hachés.
   - Sessions utilisateur protégées.
   - Gestion des rôles pour limiter les actions aux utilisateurs autorisés.

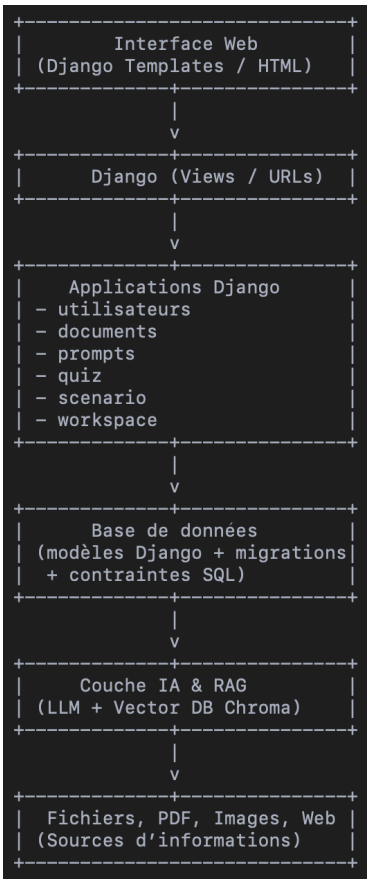
5. Compatibilité multiplateforme
   - Application web responsive.
   - Support des formats .pdf, .docx, .txt, HTML, et images OCR.

6. Documentation et tests
   - Génération automatique de documentation via Sphinx.
   - Mise en place de tests unitaires et fonctionnels (Django TestCase, pytest).

1.2 Architecture du projet

Le projet ASADI repose sur une architecture modulaire typique des projets Django, enrichie par une couche d'IA. Chaque application (ou "app") du projet est responsable d'un domaine fonctionnel précis (ex : documents, quiz, utilisateurs, etc.). L'architecture favorise la séparation des responsabilités, la réutilisabilité des composants, et la maintenabilité du code.

1.2.a Schéma général



1.2.b Composants principaux

- **Frontend** (interface utilisateur) : basé sur les templates Django (templates/). Le style CSS est personnalisé selon les besoins du projet, avec une ergonomie claire.
- **Backend Django** : contient la logique métier. Chaque app gère ses propres modèles, vues, routes, formulaires et tests.
- **Base de données** : stocke les utilisateurs, documents, quiz, scénarios, résultats, espaces de travail, etc. Le moteur est SQLite en développement, avec une possibilité d'extension vers PostgreSQL en production.
- **Couche IA (RAG + LLM) :**
  - **RAG** : le système de génération augmentée par récupération permet d'extraire du contenu de documents indexés pour générer des réponses contextuelles.
  - **LLM** : modèle de langage utilisé pour formuler les réponses à partir des informations extraites.
  - **ChromaDB** : base de données vectorielle utilisée pour stocker les représentations vectorielles des documents.
- **Fichiers** : les documents fournis (.pdf, .docx, etc.) sont analysés puis indexés pour alimenter la base de connaissances du RAG.

2 Description des applications

2.1 Utilisateurs

2.1.a Rôle général

L'application utilisateurs gère l'ensemble des opérations liées à l'inscription et l'authentification, la gestion de mot de passe, la gestion des rôles et aux droits d'accès. Elle constitue la base du système de sécurité et de personnalisation de l'application ASADI.

2.1.b Modèles

Modèle personnalisé basé sur AbstractBaseUser et PermissionsMixin avec pour champs :

- email : adresse mail unique (identifiant principal),
- username, nom, prenom : informations d'identification.
- role : définit si l'utilisateur est "administrateur" ou "utilisateur".
- niveau : niveau de l'utilisateur (utile dans le mode apprentissage).
- is_active / is_staff pour contrôler l'accès administrateur et l'activation du compte.
- date_inscription pour suivre le nombre d'utilisateurs et établir des statistiques.

2.1.c Vues principales

Authentification personnalisée

- LoginView : affichage du formulaire de connexion.
- LogoutView: permet de se déconnecter de manière sécurisée.

Gestion des utilisateurs

- ListeUtilisateursView: affichage de la liste des utilisateurs non administrateurs.
- DetailUtilisateurView: consultation des détails d'un utilisateur donné.
- SupprimerUtilisateurView: suppression possible uniquement par les administrateurs.

Mot de passe

- ResetPasswordView: vue personnalisée permettant de réinitialiser son mot de passe (envoi d'un e-mail de confirmation).

2.1.d Formulaires

CreationUtilisateurForm : formulaire de création de compte avec contrôles de validation (mot de passe, email, etc.).

ConnexionForm : formulaire de connexion standard.

ResetPasswordForm : formulaire de réinitialisation du mot de passe.

2.1.e Administration Django

Le modèle Utilisateur est enregistré dans l'interface d'administration via une classe UtilisateurAdmin personnalisée, permettant :

- L'affichage de champs personnalisés,
- Des filtres sur les rôles et l'activité,
- Une interface d'ajout/modification adaptée au modèle custom.

2.1.f URLs

Les routes définies incluent :

`/login` : connexion

`/logout` : déconnexion

`/utilisateurs/` : liste des utilisateurs

`/utilisateurs/id` : détail d'un utilisateur

`/utilisateurs//supprimer` : suppression d'un utilisateur

`/mot-de-passe-oublie`: début de la procédure de réinitialisation

2.1.g Objectifs

- Permettre l'authentification sécurisée de tous les utilisateurs.
- Gérer les comptes et leurs rôles (utilisateur simple, admin).
- Offrir des opérations de gestion (modification, suppression, réinitialisation) conformes aux droits.

3 Système RAG

Le système RAG (Retrieval-Augmented Generation) est au cœur de l'application ASADI. Il permet de générer des réponses enrichies, contextualisées et fiables en combinant deux approches :

- Récupération d'informations (Retrieval) : recherche de contenu pertinent dans une base de documents vectorialisés.
- Génération de texte (Generation) : production de la réponse à l'aide d'un modèle de langage (LLM) sur la base des informations récupérées.

Cette approche améliore la pertinence des réponses et permet de s'appuyer sur des documents vérifiables tout en réduisant les hallucinations du modèle IA afin de répondre de manière factuelle et vérifiable, garantir la pertinence et la traçabilité des informations, fournir une expérience utilisateur fluide et intelligente et enfin, s'adapter à différents cas d'usage (réponse libre, quiz, scénario).

3.1 Fonctionnement du système

Le pipeline RAG d'ASADI suit les étapes suivantes :

1. Indexation des documents :
   - Extraction du texte brut depuis divers formats (.pdf, .docx, .txt, .html, images via OCR).
   - Segmentation en chunks.
   - Encodage vectoriel via un modèle d'embedding.
   - Stockage dans ChromaDB, la base de données vectorielle.
2. Interrogation utilisateur :
   - L'utilisateur pose une question via l'interface ou le chatbot.
   - Le système détecte éventuellement un espace de travail (thématique).
3. Recherche contextuelle :
   - Transformation de la question en vecteur.
   - Recherche des documents les plus similaires dans ChromaDB.
   - Sélection des meilleurs passages à inclure dans le prompt.
4. Génération de réponse :
   - Création d'un prompt structuré incluant la question et les extraits trouvés.
   - Envoi au modèle LLM.
   - Génération d'une réponse structurée, avec citation des sources.

3.2 Technologies utilisées

Composant	Outils utilisées
Embedding	SentenceTransformers / OpenAI
Base vectorielle	ChromaDB
Moteur de génération	Modèle LLM (Pléiade)
Format de document	PDF, DOCX, TXT, HTML, PNG, etc.

3.3 Espaces de travail

Le RAG est sensibilisé à l'espace de travail utilisé par l'utilisateur. Cela signifie que :

- Si l'utilisateur choisit "RH", les documents issus de l'espace RH sont privilégiés.
- Cela permet de filtrer les résultats et de contextualiser les réponses.

4 Intégration du LLM

L'intégration du LLM (Large Language Model) constitue le cœur du fonctionnement intelligent de l'application ASADI. Elle permet de transformer une requête utilisateur en une réponse précise, contextualisée et argumentée, en s'appuyant sur une combinaison entre un moteur de recherche documentaire (RAG) et un modèle de génération de texte. Ce module repose principalement sur l'appel à une API distante hébergeant le modèle de langage.

4.1 Méthode d'intégration

L'intégration est réalisée par une classe personnalisée nommée PleiadeLLM, qui hérite de CustomLLM (issue de LlamaIndex). Elle encapsule toute la logique d'interaction avec l'API distante. L'API est accessible à l'URL : https://pleiade.mi.parisdescartes.fr/api/chat/completions et sécurisée par une clé stockée dans un fichier .env.

4.2 Construction du prompt

Le prompt envoyé au LLM suit une structure rigoureuse :

- Prompt système : définit le rôle de l'assistant (ex. : assistant administratif rigoureux et factuel).
- Contexte : insertion des extraits de documents les plus pertinents, obtenus via la base vectorielle (ChromaDB) et rerankés par un modèle CrossEncoder.
- Question utilisateur : ajoutée à la fin.
- Historique de conversation : chaque prompt est enrichi avec les échanges précédents pour maintenir la cohérence du dialogue.

Cette structure assure des réponses logiques, contextualisées, et surtout sourcées.

4.3 Utilisation dans l'application

Le LLM est utilisé dans différentes fonctionnalités de l'application :

- Chatbot principal : pour répondre aux questions administratives.
- Génération de titres : pour nommer automatiquement les prompts et les quiz.
- Amélioration de la formulation : nettoyage ou reformulation des requêtes utilisateur.
- Résumé de documents : synthèse automatique de contenus ajoutés à la base.

4.4 Optimisations mises en œuvre

Plusieurs optimisations ont été intégrées pour améliorer la pertinence et les performances :

- Recherche hybride : combinaison de recherche vectorielle (ChromaDB) et BM25.
- Filtrage par espace de travail : permet de restreindre les résultats aux documents d'un domaine spécifique.
- Reranking intelligent : amélioration de l'ordre des résultats via un modèle CrossEncoder.
- Température faible (0.3) : pour limiter la créativité du modèle et favoriser la cohérence.
- Timeout de 60 secondes : pour éviter les blocages prolongés.

4.5 Gestion des erreurs

La robustesse du système est assurée par :

- Une gestion explicite des erreurs réseau ou de réponse vide.
- Un message utilisateur : « Je n'ai pas pu obtenir de réponse du LLM. Veuillez réessayer plus tard. »
- Un contrôle préalable de la présence de documents dans le contexte avant l'appel du LLM.

4.6 Contraintes techniques

Bien que des options locales aient été envisagées (ex. : Ollama), le modèle est exécuté à distance pour des raisons de performance et de cohérence. L'utilisation de LlamaIndex comme couche

d'orchestration entre le LLM, la base vectorielle et Django permet une intégration fluide dans l'architecture du projet.

5 Traitement des documents

Le module de traitement des documents constitue la première étape du pipeline RAG de l'application ASADI. Il est chargé d'extraire, normaliser, segmenter et indexer les contenus textuels des fichiers importés dans la base de connaissances. Ce processus garantit que les réponses générées par le LLM sont basées sur des sources fiables, bien structurées et représentatives du contenu original.

5.1 Formats pris en charge

Le système ASADI supporte un large éventail de formats couramment utilisés dans les démarches administratives :

- Textuels : .pdf, .docx, .doc (via conversion), .txt, .html, .md, .rtf
- Bureautiques : .xlsx, .csv, .pptx
- Visuels : PDF scannés et images (via OCR)

Cette compatibilité étendue permet d'indexer la majorité des documents rencontrés en contexte professionnel ou administratif.

5.2 Extraction de texte

La phase d'extraction repose sur un ensemble de bibliothèques spécialisées selon le type de fichier :

Format	Bibliothèque utilisée
PDF non scanné	pdfplumber
PDF scanné	pdf2image + pytesseract (OCR)
HTML	BeautifulSoup
XLSX/CSV	pandas
PPTX	python-pptx
RTF	striprtf

Pour les fichiers complexes (PDF avec tableaux, PowerPoint), des traitements spécifiques sont appliqués afin de restituer le contenu de manière structurée et exploitable.

5.3 Nettoyage et normalisation

Le texte extrait est nettoyé de manière ciblée :

- Suppression des espaces superflus
- Ajout de retours à la ligne pour préserver la hiérarchie du document
- Formatage des tableaux avec délimiteurs pour faciliter leur interprétation
- Organisation des feuilles Excel en blocs de texte par onglet

Une fonction clean_text() existe mais est volontairement désactivée afin de ne pas perturber la qualité de récupération du contexte par le RAG.

5.4 Segmentation en chunks

Le contenu est ensuite découpé en segments appelés chunks, selon deux méthodes principales :

- Segmentation intelligente : basée sur la structure logique (titres, sections)
- Segmentation récursive : via RecursiveCharacterTextSplitter (taille : 500 caractères, chevauchement : 150)

Des adaptations spécifiques sont prévues selon le format :

- Excel ⟶ un chunk par feuille
- PowerPoint ⟶ un chunk par diapositive

Cette stratégie permet de conserver la cohérence sémantique et d'optimiser la pertinence des résultats lors de la recherche vectorielle.

5.5 Génération des embeddings

Chaque chunk est vectorisé à l'aide du modèle all-mpnet-base-v2 de SentenceTransformers, reconnu pour ses performances dans les tâches de similarité textuelle. Les vecteurs générés servent à alimenter la base vectorielle utilisée dans le système RAG.

5.6 Indexation dans la base vectorielle

Les vecteurs sont stockés dans ChromaDB, au sein d'une collection unique asadi_collection. Le système utilise :

- PersistentClient pour garantir la persistance des données,
- un découpage par workspace pour filtrer les documents selon leur domaine d'application (ex : RH, Comptabilité).

L'indexation est déclenchée automatiquement à l'ajout d'un fichier via l'interface administrateur. Un script d'ingestion ingest_all_documents() est également disponible pour réindexer massivement les documents d'un répertoire.

5.7 Métadonnées associées

Chaque chunk stocke des métadonnées essentielles pour le RAG :

- source : chemin du fichier d'origine
- chunk_index : position du chunk dans le document
- start_index : index de départ du texte original
- workspace : domaine de rattachement
- is_summary : si le chunk correspond à un résumé
- feuille_courante, nomsFeuilles : spécifiques aux fichiers Excel

Ces informations enrichissent le contexte des réponses et permettent des filtrages précis.

5.8 Suppression de documents

Lorsqu'un administrateur supprime un document :

- Le fichier est retiré du système de fichiers.
- Les vecteurs correspondants sont supprimés de ChromaDB via une requête delete(where={« source »: ...}).
- L'opération est propagée à tous les workspaces associés, assurant la cohérence de la base de connaissances.

6 Gestions des workspaces

Le système de **workspaces** dans ASADI constitue un mécanisme central pour l'organisation des documents et la contextualisation des réponses du LLM. Il permet à l'utilisateur de spécifier un périmètre de recherche thématique, garantissant ainsi des réponses plus pertinentes et ciblées.

6.1 Rôle des workspaces

Les workspaces ont trois fonctions principales :

- **Organisation des documents** : ils regroupent les ressources documentaires par thème (ex. : RH, Juridique, Comptabilité).
- **Filtrage sémantique** : ils agissent comme un filtre dans le pipeline RAG, limitant la recherche vectorielle aux documents d'un domaine spécifique.
- **Interface utilisateur** : ils sont visibles et sélectionnables via une interface modale accessible à tout moment, offrant à l'utilisateur un contrôle précis sur le contexte de sa requête.

Le workspace devient ainsi un **contexte implicite** transmis à chaque requête du LLM.

6.2 Droits d'accès et gestion

La gestion des workspaces est **réservée aux administrateurs** :

- Création et suppression via l'interface d'administration.
- Association dynamique lors de l'import de nouveaux documents.
- Relation ManyToMany entre documents et workspaces, permettant à un document d'appartenir à plusieurs espaces.

Les utilisateurs simples peuvent uniquement les consulter et les sélectionner comme filtres.

6.3 Modélisation

Le modèle `Workspace` dans Django est défini comme suit :

```
class Workspace(models.Model):
    name = models.CharField(max_length=100, unique=True)
    contenue = models.CharField(max_length=120)
    document = models.ManyToManyField('documents.Document', blank=True,
related_name='workspaces')
    date_creation = models.DateTimeField(auto_now_add=True)
```

Il surcharge la méthode `save()` pour synchroniser la création ou le renommage de dossiers physiques dans le système de fichiers.

6.4 Liaison avec les documents

- **Relation ManyToMany** définie dans le modèle `Workspace`
- Lors de l'ajout d'un document, celui-ci peut être lié à plusieurs workspaces
- Les fichiers sont également **déplacés physiquement** dans des sous-dossiers correspondant aux workspaces pour une organisation plus lisible

6.5 Intégration avec ChromaDB

Lors de l'indexation :

- Le champ `workspace` est injecté comme **métadonnée** dans chaque chunk vectorisé
- Il est utilisé comme **filtre dans la clause `where`** des requêtes ChromaDB :

```
where={"workspace": workspace} if workspace else None
```

Lorsqu'un document change de workspace :

- Ses chunks sont supprimés de ChromaDB
- Ils sont **réindexés avec la nouvelle valeur de workspace**

6.6 Contrôles et restrictions

- **Nom unique** imposé par `unique=True`
- **Vérification lors de la suppression** : les documents liés ne sont pas supprimés mais déplacés vers le dossier racine
- **Longueurs limitées** : 100 caractères pour le nom, 120 pour la description
- **Caractères spéciaux** : aucune validation explicite, mais déconseillés car les noms sont utilisés dans le système de fichiers

7 Fonctionnalités additionnelles

L'application ASADI intègre un ensemble de fonctionnalités complémentaires visant à enrichir l'expérience utilisateur, à personnaliser les interactions et à faciliter la maintenance pour les administrateurs. Bien que certaines soient secondaires, elles renforcent l'ergonomie, la sécurité et la souplesse du système.

7.1 Historique des conversations

ASADI conserve l'historique complet des échanges entre l'utilisateur et l'assistant :

- **Affichage dynamique** : l'historique est accessible dans le menu latéral gauche.
- **Suppression manuelle** : chaque session peut être supprimée individuellement.
- **Génération automatique de titres** : les conversations sont nommées automatiquement via la fonction `generate_prompt_title()`.
- **Conservation du contexte** : chaque entrée conserve les prompts, les réponses, la date et le workspace actif (via l'URL).

L'utilisateur peut reprendre à tout moment une conversation précédente, avec rechargement du contexte conversationnel et du workspace associé. Cela permet de maintenir une continuité logique dans les échanges avec l'IA.

7.2 Accessibilité et ergonomie

ASADI intègre plusieurs mécanismes facilitant la navigation et l'utilisation :

- **Modales d'information** : affichage des détails de quiz, scénarios ou profil utilisateur.
- **Tooltips** : infobulles descriptives sur les boutons et actions clés.
- **Overlay de chargement** : animation visuelle lors des requêtes vers le LLM.
- **Interface responsive** : prise en charge des résolutions variées.
- **Confirmations** : messages de validation avant suppression (quiz, scénarios…).

Bien que basique, cette couche ergonomique participe à l'amélioration de l'expérience utilisateur.

7.3 Sécurité applicative

La sécurité est assurée par plusieurs moyens standard :

- **Authentification via email et mot de passe**.
- **Contrôle d'accès** :
  - `@login_required` pour les routes privées,
  - `@user_passes_test(est_admin)` pour l'accès administrateur.
- **Protection CSRF** sur tous les formulaires.
- **Validation stricte** des formats de fichiers à l'upload.

Toutefois, certains points restent à améliorer :

- Pas de journalisation avancée des actions.
- Absence de protection explicite contre les abus ou les attaques de type prompt injection.

7.4 Scripts et outils techniques

Plusieurs scripts en ligne de commande sont disponibles pour la maintenance :

- `list_chroma_contents.py` : exploration des contenus indexés dans ChromaDB.
- `reset_chroma.py` : réinitialisation complète de la base vectorielle.
- `sync_svn_git.sh` : synchronisation entre SVN et Git.
- `ingesterDonnee.py` : ingestion en masse de documents.
- `extractions.py` : extraction spécialisée de contenu (OCR, Excel, etc.).

Ces outils facilitent la gestion hors interface web et permettent une automatisation des tâches administratives.

8 Sécurité de l'application

L'application ASADI repose sur les mécanismes de sécurité standard fournis par le framework Django. Elle met en place plusieurs protections pour l'authentification, le contrôle d'accès et le traitement des fichiers. Toutefois, certains aspects restent à renforcer, notamment concernant l'interaction avec le LLM, la journalisation et la conformité réglementaire.

8.1 Authentification et gestion des utilisateurs

- **Hashage des mots de passe** : Django utilise PBKDF2 avec SHA256. La méthode `set_password()` est utilisée et aucun mot de passe n'est stocké en clair.
- **Réinitialisation** : La classe personnalisée `ResetPasswordView` envoie un lien de réinitialisation sécurisé par email.
- **Sécurité des sessions** : Sessions gérées par `SessionMiddleware`, protection CSRF via `CsrfViewMiddleware`, mais sans timeout explicite.
- **Tentatives de connexion** : Aucune limite sur les tentatives ni captcha, mais les messages d'erreur restent génériques.

8.2 Contrôle d'accès

- **Vues Django sécurisées** : `@login_required` pour les pages privées, `@user_passes_test(est_admin)` pour les administrateurs.
- **Séparation des rôles** : Basée sur les groupes Django, avec des vérifications dans le backend et les templates.
- **Affichage conditionnel** : Les actions admin sont masquées côté template selon le rôle utilisateur.
- **Protection CSRF** systématique sur les formulaires.

8.3 Sécurité des fichiers

- **Vérification des types** : Seules certaines extensions sont acceptées (.pdf, .docx, .xlsx, etc.).
- **Limites de taille** : 50 Mo maximum, défini dans `settings.py`.
- **Stockage** : Les fichiers sont enregistrés localement dans `media/documents/`, organisés par workspace.
- **Risque d'exécution** : Aucun traitement actif des fichiers ; ils sont uniquement lus pour en extraire du texte. Pas de scan antivirus.

8.4 Sécurité des interactions avec le LLM

- **Nettoyage des prompts** : Aucun filtrage ou modification des prompts avant envoi.
- **Protection contre les prompt injections** : Aucune mesure spécifique.
- **Affichage des réponses** : Les réponses générées sont affichées sans contrôle ni vérification.

8.5 Journalisation et suivi

- **Suivi des connexions** : Seule la dernière connexion est enregistrée (`last_login`). Pas de suivi d'action (ajout, suppression…).
- **Absence d'audit trail** : Aucun système d'audit ou de log complet.
- **Alerte** : Aucune alerte automatique ni détection d'abus.

8.6 Conformité RGPD

- **Pages légales** : Politique de confidentialité et CGU présentes.
- **Consentement** : Aucun mécanisme explicite de consentement à la collecte des données.
- **Suppression des données** :
  - Seuls les administrateurs peuvent supprimer un compte.
  - Pas de système d'anonymisation ou d'effacement automatique.

8.7 Tests de sécurité

- **Tests automatisés** : Fichiers `tests.py` présents mais non implémentés.
- **Tests de vulnérabilité** : Aucun test de pénétration ni outil d'analyse de sécurité automatique configuré.

8.8 Bilan

ASADI bénéficie des protections par défaut de Django (authentification, CSRF, gestion des sessions), mais des efforts sont nécessaires pour :

- Mieux sécuriser les interactions avec le LLM (prompt injection, filtrage des réponses).
- Améliorer la traçabilité via une journalisation approfondie.
- Se conformer davantage au RGPD (effacement, export, consentement).
- Mettre en place des tests de sécurité réguliers.

9 Conclusion

L'architecture modulaire du projet, basée sur Django, permet une grande évolutivité, tout en assurant la séparation des responsabilités et la maintenabilité du code. Les fonctionnalités avancées telles que les workspaces thématiques, l'historique conversationnel enrichi, les scénarios interactifs et les outils d'administration renforcent l'expérience utilisateur autant que les capacités fonctionnelles du système.

Si la sécurité de base est bien assurée via les outils fournis par Django (authentification, CSRF, validation des fichiers), des pistes d'amélioration demeurent en matière de protection avancée contre les attaques, de journalisation, de respect du RGPD, ainsi que dans l'intégration de tests automatisés.