**UNIVERSITE DE KINSHASA**



**FACULTE DE SCIENCES ET TECHNOLOGIES**

**MENTION MATHEMATIQUES, STATISTIQUE ET INFORMATIQUE**

**B.P 190 KINSHASA XI**

**Rapport du travail pratique dans le cadre du séminaire de Python avancé :**

**Développement web avancé : Création d'une application web en utilisant Flask ou Django avec des fonctionnalités avancées comme l'authentification utilisateur, l'API REST, etc**

**Mise en place d’une application web Django de réservation de vols**

**KIPULU NGIMBI Serge Mario**

**Prof MASAKUNA Jordan**

2023-2024

Contenu

[I. Problématique des Applications Web et de Django 2](#_Toc174823131)

[I.1. Introduction 2](#_Toc174823132)

[I.2. Sécurité des Applications Web 2](#_Toc174823133)

[I.3. Scalabilité et Performance 2](#_Toc174823134)

[I.4. Expérience Utilisateur (UX) 2](#_Toc174823135)

[I.5. Gestion des Données et Compliance 2](#_Toc174823136)

[II. Cadre théoriques 4](#_Toc174823137)

[II.1. Historique et Evolution de Django 4](#_Toc174823138)

[II.2. Comparaison avec d'Autres Frameworks 4](#_Toc174823139)

[II.3. Avantages de Django pour le Développement d'Applications Web 4](#_Toc174823140)

[II.4. Limites et Inconvénients de Django 4](#_Toc174823141)

[III. Questions de Recherche sur les Applications Web en Django 5](#_Toc174823142)

[IV. Fonctionnement Detaillé de Notre Application 6](#_Toc174823143)

[IV.1. Architecture Générale de l'Application 6](#_Toc174823144)

[IV.2. Fonctionnalités Clés de l'Application 6](#_Toc174823145)

[IV.3. Interface Utilisateur 6](#_Toc174823146)

[IV.4. Sécurité et Conformité 7](#_Toc174823147)

[V. Discussion et Commentaire 8](#_Toc174823148)

[V.1. Réflexion sur le Développement du Projet 8](#_Toc174823149)

[V.2. Défis et Obstacles Rencontrés 8](#_Toc174823150)

[V.3. Perspectives d'Amélioration 8](#_Toc174823151)

# I. Problématique des Applications Web et de Django

## I.1. Introduction

Les applications web sont au cœur de l'écosystème numérique moderne, permettant une interaction en temps réel entre les utilisateurs et les services en ligne. Cependant, le développement d'applications web efficaces et fiables présente une série de défis. Ces défis sont particulièrement accentués dans des domaines tels que la réservation de vols, où les utilisateurs attendent non seulement une expérience fluide, mais aussi une sécurité de leurs données personnelles et financières.

## I.2. Sécurité des Applications Web

La sécurité est l'un des principaux enjeux du développement d'applications web. Les cyberattaques sont de plus en plus sophistiquées, et les applications web doivent être conçues pour résister à une variété de menaces, notamment les injections SQL, le cross-site scripting (XSS), et les attaques par force brute. Django adresse cette problématique en intégrant par défaut des protections contre ces menaces, telles que le hachage des mots de passe avec un sel, la gestion des sessions sécurisées, et la protection CSRF (Cross-Site Request Forgery).

## I.3. Scalabilité et Performance

Avec l'augmentation du nombre d'utilisateurs d'internet, les applications web doivent être capables de gérer des charges de plus en plus importantes. La scalabilité d'une application est cruciale pour maintenir sa performance à mesure que le nombre de requêtes simultanées augmente. Django facilite cette tâche en permettant une gestion efficace des bases de données via son ORM (Object-Relational Mapping) et en offrant des outils pour la mise en cache et la gestion de sessions distribuées.

## I.4. Expérience Utilisateur (UX)

L'expérience utilisateur est un autre facteur clé dans le succès d'une application web. Une interface utilisateur bien conçue, intuitive et réactive peut significativement améliorer la satisfaction de l'utilisateur et augmenter les taux de conversion. Django offre des outils pour créer des interfaces utilisateur dynamiques et réactives en combinaison avec des frameworks front-end comme Bootstrap. De plus, la gestion des formulaires de Django simplifie le processus de création d'interfaces utilisateur qui interagissent directement avec les données back-end.

## I.5. Gestion des Données et Compliance

Enfin, la gestion des données est un aspect central de toute application web. Les réglementations comme le RGPD (Règlement Général sur la Protection des Données) imposent des normes strictes sur la manière dont les données des utilisateurs sont collectées, stockées et traitées. Django aide les développeurs à respecter ces normes grâce à son modèle de données structuré et à ses outils de gestion des migrations de base de données.

# II. Cadre théoriques

## II.1. Historique et Evolution de Django

Django a été initialement développé en 2003 pour répondre aux besoins de sites de presse qui devaient publier des articles rapidement et avec un contenu constamment mis à jour. Ce framework a depuis évolué pour devenir l'un des outils les plus populaires pour le développement d'applications web. Sa popularité repose sur son approche "batteries-included", où une multitude de fonctionnalités essentielles sont intégrées par défaut, facilitant ainsi le développement rapide de projets robustes.

## II.2. Comparaison avec d'Autres Frameworks

Django est souvent comparé à d'autres frameworks web, comme Flask, Ruby on Rails, et Laravel. Flask, par exemple, est un micro-framework en Python connu pour sa flexibilité et sa légèreté. Contrairement à Django, Flask ne fournit pas de structure imposée, ce qui donne aux développeurs une liberté totale mais nécessite également plus de configuration manuelle. Ruby on Rails, quant à lui, partage avec Django une philosophie similaire d'inclusivité et de rapidité de développement, mais repose sur le langage Ruby. Laravel, basé sur PHP, est également reconnu pour son élégance et ses fonctionnalités étendues, bien que Django soit souvent préféré pour sa communauté active et sa compatibilité avec l'écosystème Python.

## II.3. Avantages de Django pour le Développement d'Applications Web

Django est apprécié pour ses nombreux avantages dans le développement d'applications web. L'un de ses principaux atouts est son ORM, qui permet de manipuler des bases de données relationnelles en utilisant des objets Python plutôt que des requêtes SQL directes, ce qui simplifie grandement le développement et la maintenance. De plus, Django inclut un système d'administration automatisé, généré à partir des modèles de données, qui permet aux développeurs de gérer les données de l'application de manière efficace et sans effort supplémentaire de codage.

## II.4. Limites et Inconvénients de Django

Cependant, Django n'est pas sans inconvénients. Sa courbe d'apprentissage peut être abrupte pour les débutants, notamment en raison de sa nature monolithique. De plus, bien que Django soit très flexible, il peut parfois être moins adapté à des projets qui nécessitent une personnalisation fine et où un micro-framework comme Flask pourrait être plus approprié. Enfin, le déploiement de projets Django peut être plus complexe que celui de projets développés avec des frameworks plus légers.

# III. Questions de Recherche sur les Applications Web en Django

Dans le cadre du développement de notre application de réservation de vols, plusieurs questions de recherche ont émergé :

1. Comment Django facilite-t-il l'intégration des fonctionnalités avancées comme l'authentification utilisateur et les API REST ?
2. Quels sont les défis communs rencontrés lors du développement d'applications web avec Django ?
3. Comment Django se compare-t-il à d'autres frameworks web en termes de performance et de scalabilité ?
4. Quelles sont les meilleures pratiques pour optimiser une application Django pour une utilisation à grande échelle ?

# IV. Fonctionnement Detaillé de Notre Application

## IV.1. Architecture Générale de l'Application

L'application de réservation de vols que nous avons développée suit une architecture MVC (Model-View-Controller), typique des projets Django. Cette architecture sépare clairement les différentes responsabilités de l'application, facilitant ainsi le développement, la maintenance, et l'évolution du projet. Les modèles définissent la structure des données et les interactions avec la base de données, les vues gèrent la logique de l'application et traitent les requêtes des utilisateurs, tandis que les templates génèrent le HTML affiché dans le navigateur.

## IV.2. Fonctionnalités Clés de l'Application

1. **Authentification Utilisateur** :
   * L'application utilise le système d'authentification de Django pour gérer l'inscription, la connexion, et la réinitialisation de mot de passe des utilisateurs. Nous avons personnalisé le processus d'authentification en ajoutant des vérifications de sécurité supplémentaires, comme la validation des adresses e-mail et la protection contre les attaques par force brute.
   * Django propose également un middleware pour gérer les sessions des utilisateurs, garantissant que les données sensibles sont protégées et que les utilisateurs restent connectés de manière sécurisée.
2. **Recherche et Réservation de Vols** :
   * Les utilisateurs peuvent rechercher des vols en fonction de divers critères, tels que la date, la destination, et le nombre de passagers. L'application interagit avec une API externe pour récupérer des informations en temps réel sur les vols disponibles.
   * Le processus de réservation est conçu pour être aussi simple et intuitif que possible. Après avoir sélectionné un vol, l'utilisateur peut entrer ses informations personnelles et procéder à la réservation. Une fois la réservation confirmée, un e-mail de confirmation est envoyé à l'utilisateur.
3. **Gestion des Données** :
   * Les données sur les vols, les utilisateurs, et les réservations sont stockées dans une base de données relationnelle, gérée via l'ORM de Django. Cette approche permet une gestion simplifiée des données et réduit le risque d'erreurs liées à la manipulation manuelle de SQL.
   * Nous avons également mis en place des outils de migration pour gérer l'évolution du schéma de la base de données au fur et à mesure que l'application se développe. Les migrations sont une fonctionnalité clé de Django, permettant de maintenir la cohérence des données à travers les différentes versions de l'application.

## IV.3. Interface Utilisateur

L'interface utilisateur de l'application a été développée en utilisant des templates HTML de Django, stylisés avec Bootstrap pour assurer une réactivité et une compatibilité multi-plateformes. L'objectif principal était de créer une interface intuitive, permettant aux utilisateurs de naviguer facilement et de compléter leurs réservations sans friction.

Voici les interfaces principales de notre application

**Page de connexion :**

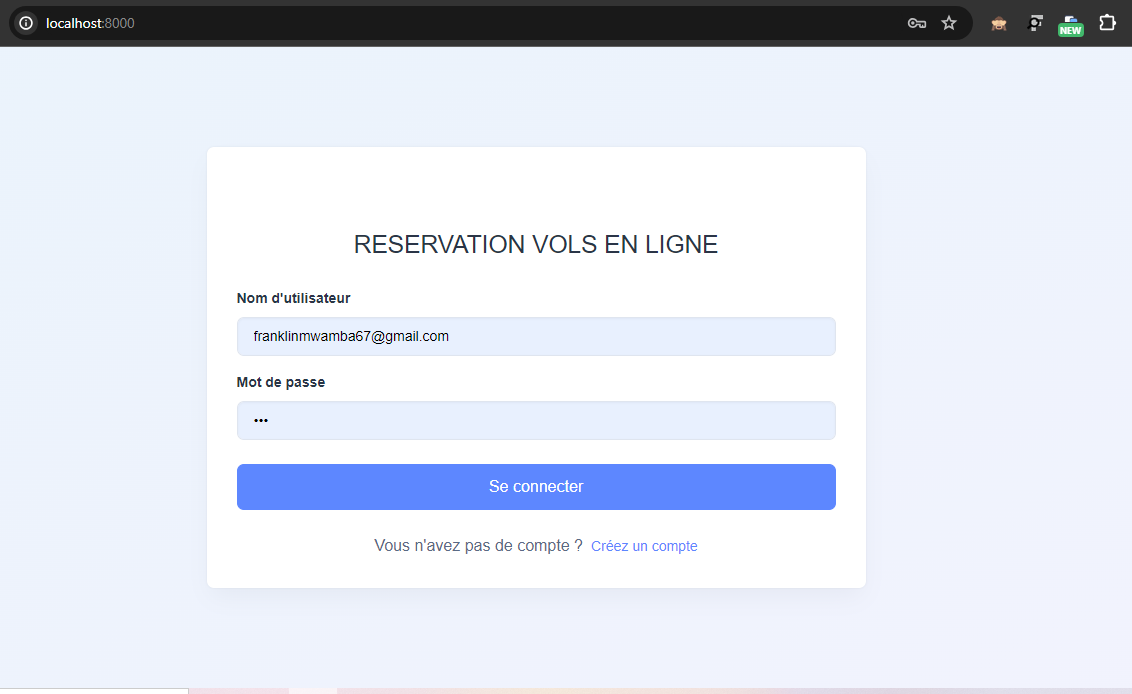


Figure : Page de connexion

**Page de création de compte :**

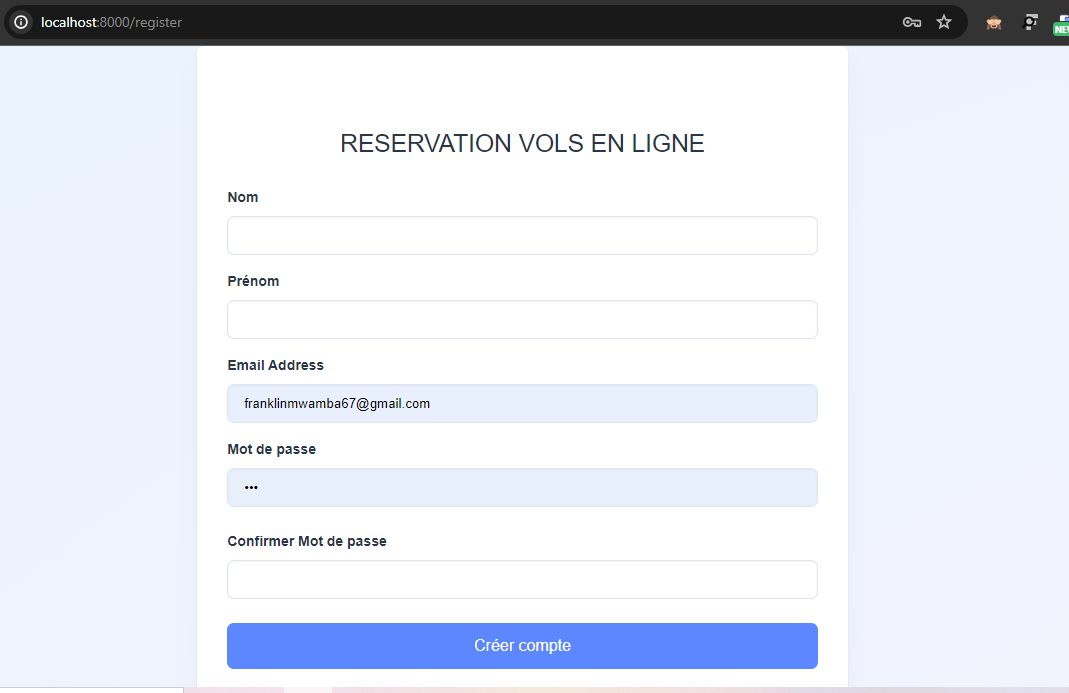


Figure : page de création de compte

**Page de recherche de vol et de réservation :**

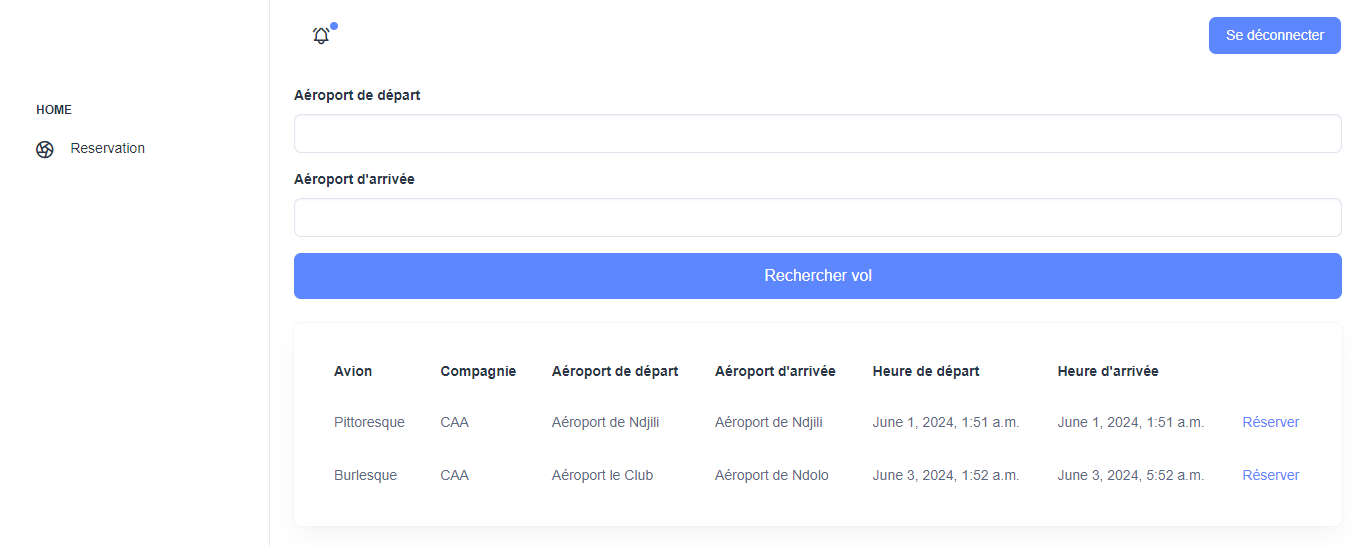


Figure : Liste vols

Django admin nous permet de créer de vols.

## IV.4. Sécurité et Conformité

En matière de sécurité, l'application implémente plusieurs des bonnes pratiques recommandées par Django, y compris l'utilisation du système d'authentification intégré, la gestion sécurisée des sessions, et l'application de politiques strictes pour les formulaires et les entrées utilisateur. De plus, nous avons pris des mesures pour nous assurer que l'application est conforme aux régulations telles que le RGPD, en implémentant des fonctionnalités permettant aux utilisateurs de contrôler leurs données personnelles.

# V. Discussion et Commentaire

## V.1. Réflexion sur le Développement du Projet

Le développement de cette application de réservation de vols a permis d'explorer en profondeur les fonctionnalités avancées de Django, ainsi que d'apprendre à surmonter divers défis techniques. Django s'est révélé être un outil puissant, offrant une grande flexibilité et de nombreuses fonctionnalités intégrées qui ont facilité le développement rapide et sécurisé de l'application. Le choix de Django a été particulièrement pertinent dans le contexte d'une application nécessitant une gestion complexe des utilisateurs et des transactions sécurisées.

## V.2. Défis et Obstacles Rencontrés

Plusieurs défis ont été rencontrés au cours du développement. L'un des principaux obstacles a été la gestion des dépendances et la configuration des environnements de développement. Django, bien que complet, nécessite une bonne connaissance de l'écosystème Python et des outils de gestion d'environnements virtuels pour éviter les conflits entre les différentes bibliothèques utilisées.

Un autre défi majeur a été l'optimisation des performances, en particulier dans le contexte de la gestion des sessions utilisateur et de la mise en cache des résultats des recherches de vols. Bien que Django offre des solutions pour ces problématiques, leur mise en œuvre nécessite une compréhension approfondie des mécanismes sous-jacents.

## V.3. Perspectives d'Amélioration

Pour l'avenir, plusieurs pistes d'amélioration peuvent être envisagées pour l'application. Tout d'abord, la mise en œuvre de mécanismes de scalabilité plus avancés pourrait être explorée, notamment en utilisant des techniques de load balancing et de mise en cache distribuée pour améliorer les performances sous forte charge.

En termes de sécurité, bien que l'application implémente déjà les bonnes pratiques de Django, des améliorations pourraient être apportées en intégrant des solutions de détection et de prévention des intrusions (IDS/IPS) pour renforcer encore plus la protection des données utilisateurs.

Enfin, l'interface utilisateur pourrait être affinée pour améliorer l'accessibilité et l'expérience utilisateur, en particulier pour les utilisateurs mobiles. Une attention particulière pourrait être portée à la rapidité de chargement des pages et à la réduction du nombre de clics nécessaires pour compléter une réservation.