

## Rapport pour le Projet de Fin de Demi-Module

*Module : Webmapping et  
Développement Mobile*



Réalisé par :

**Aymane Zian**

**Nada Cherki**

**Imane Khoussi**

Encadré par :

**Mme Hafida KHALFAOUI**

**Année Universitaire : 2024/2025**

## *Dédicaces*

*À nos parents qui ont toujours été présents à nos côtés, qui nous ont soutenus inconditionnellement et qui ont fait de nombreux sacrifices pour nous permettre de poursuivre nos études dans les meilleures conditions possibles.*

*À nos enseignants qui nous ont transmis leur savoir avec passion et dévouement, et particulièrement à Madame Hafida KHALFAOUI pour son encadrement précieux et ses conseils avisés durant ce projet.*

*À nos amis et camarades de promotion qui ont partagé avec nous ces années d'études, les moments de joie comme les périodes de travail intense.*

*Nous dédions ce modeste travail à tous ceux qui nous ont encouragés et soutenus tout au long de notre parcours académique.*

## *Remerciements*

Au terme de ce projet de fin de demi-module, nous tenons à exprimer notre profonde gratitude envers toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce travail.

Nous adressons nos sincères remerciements à notre encadrante, Madame Hafida KHALFAOUI, pour sa disponibilité, ses précieux conseils et son soutien constant tout au long de la réalisation de ce projet. Ses recommandations pertinentes et son expertise dans le domaine du développement mobile ont été déterminantes pour la réussite de notre application.

Nous remercions également l'ensemble du corps professoral de la Faculté des Sciences et Techniques de Tanger pour la qualité de l'enseignement dispensé et pour nous avoir fourni les connaissances et compétences nécessaires à la réalisation de ce projet.

Notre reconnaissance s'adresse aussi à l'administration de la faculté pour avoir mis à notre disposition les ressources et l'environnement propices à notre apprentissage.

Enfin, nous exprimons notre gratitude à nos familles et proches pour leur soutien moral inestimable et leurs encouragements constants qui nous ont permis de mener à bien ce projet.

## Résumé

Ce rapport présente le développement de “Saydaliati”, une application mobile Android conçue pour faciliter la recherche de pharmacies à Tanger, Maroc. Face aux difficultés que rencontrent les citoyens pour trouver des pharmacies, notamment celles de garde, notre application propose une solution innovante basée sur la géolocalisation.

Développée avec Android Studio et Java, l’application “Saydaliati” offre aux utilisateurs la possibilité de localiser les pharmacies à proximité, d’identifier celles qui sont de garde, et d’obtenir des informations détaillées (adresse, numéro de téléphone, horaires). Elle intègre également une interface d’administration permettant aux autorités compétentes de gérer les données des pharmacies et leurs périodes de garde.

Le développement a suivi une approche Agile avec la méthodologie Scrum, permettant une adaptation continue aux besoins des utilisateurs. L’architecture MVVM (Model-View-ViewModel) a été choisie pour assurer une séparation claire des responsabilités et faciliter la maintenance.

Les tests effectués ont confirmé la fiabilité et les performances de l’application, qui répond efficacement aux objectifs fixés. Des perspectives d’amélioration ont été identifiées, notamment l’ajout de fonctionnalités telles que la réservation de médicaments ou l’extension de la couverture géographique à d’autres villes.

*Mots-clés : Android, Java, géolocalisation, pharmacies, application mobile, Tanger*

## Abstract

This report presents the development of “Saydaliati,” an Android mobile application designed to facilitate pharmacy searches in Tangier, Morocco. Addressing the difficulties citizens face in finding pharmacies, particularly those on duty, our application offers an innovative solution based on geolocation.

Developed with Android Studio and Java, the “Saydaliati” application provides users with the ability to locate nearby pharmacies, identify those on duty, and obtain detailed information (address, phone number, hours). It also includes an administration interface allowing competent authorities to manage pharmacy data and their duty periods.

The development followed an Agile approach with the Scrum methodology, enabling continuous adaptation to user needs. The MVVM (Model-View-ViewModel) architecture was chosen to ensure a clear separation of responsibilities and facilitate maintenance.

Testing has confirmed the reliability and performance of the application, which effectively meets the established objectives. Prospects for improvement have been identified, particularly the addition of features such as medication reservation or geographical coverage extension to other cities.

*Keywords: Android, Java, geolocation, pharmacies, mobile application, Tangier*

## *Avant-propos*

Dans un monde de plus en plus digitalisé, les applications mobiles jouent un rôle prépondérant dans la simplification de notre quotidien. Elles transforment nos interactions avec les services essentiels et redéfinissent l'accès à l'information. C'est dans cette perspective que s'inscrit notre projet "Saydaliati".

Ce rapport présente le fruit d'un travail collectif mené dans le cadre du projet de fin de demi-module de Développement Mobile à la Faculté des Sciences et Techniques de Tanger. Il retrace notre parcours depuis la conception jusqu'à la réalisation d'une application mobile destinée à faciliter l'accès aux services pharmaceutiques dans la ville de Tanger.

Le développement de "Saydaliati" a représenté pour notre équipe un défi technique et organisationnel qui nous a permis de mettre en pratique nos connaissances théoriques et de développer nos compétences en programmation mobile. Au-delà de l'aspect académique, ce projet a été motivé par notre volonté de contribuer à l'amélioration des services de santé dans notre région en proposant une solution concrète à un problème quotidien.

Nous espérons que ce rapport saura transmettre non seulement les aspects techniques de notre travail, mais également la passion et l'engagement qui ont animé notre équipe tout au long de ce projet.

## Liste des sigles et abréviations

Sigle/Abréviation	Signification
API	Application Programming Interface
APK	Android Package Kit
CRUD	Create, Read, Update, Delete
DAO	Data Access Object
GPS	Global Positioning System
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
MVVM	Model-View-ViewModel
MVC	Model-View-Controller
REST	Representational State Transfer
SDK	Software Development Kit
SQL	Structured Query Language
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	eXtensible Markup Language

# Table des matières

Dédicaces.....	2
Remerciements .....	3
Résumé.....	4
Abstract .....	4
Avant-propos.....	5
Liste des sigles et abréviations.....	6
Table des matières .....	7
Introduction Générale.....	10
Contexte général .....	10
Problématique.....	10
Objectifs du projet.....	11
Méthodologie générale adoptée .....	11
Structure du rapport .....	12
Chapitre 1 : Présentation du projet “Saydaliati” .....	13
1.1 Contexte et justification .....	14
1.2 Présentation globale de l’application.....	14
1.3 Problèmes ciblés et solutions proposées.....	15
1.4 Objectifs spécifiques .....	16
1.5 Étude de faisabilité.....	17
1.6 Planification et découpage du projet.....	18
1.7 Organisation du travail.....	19
Conclusion .....	20
Chapitre 2 : Méthodologie de développement.....	21
Introduction.....	22
2.2 Choix de la méthode de développement.....	22
2.3 Présentation de la méthode Agile.....	24
2.4 Application du cadre Scrum au projet.....	26
2.5 Outils de gestion utilisés .....	28

Conclusion .....	30
Chapitre 3 : Analyse et Conception de l'application .....	32
Introduction.....	33
3.1 Spécifications fonctionnelles.....	33
3.2 Identification des utilisateurs et cas d'usage .....	35
3.3 Modélisation UML.....	36
3.6 Choix techniques et contraintes.....	38
3.7 Conclusion du chapitre.....	41
Chapitre 4 : Environnement de développement .....	42
Introduction.....	43
4.1 Présentation d'Android Studio et du SDK .....	43
4.2 Langages et bibliothèques utilisés .....	44
4.3 Structure technique de l'application.....	46
4.4 Gestion des permissions et sécurité .....	49
Conclusion .....	51
Chapitre 5 : Réalisation de l'application "Saydaliati" .....	52
Introduction.....	53
5.2 Interfaces du module utilisateur .....	53
5.3 Interfaces du module administrateur .....	61
5.4 Conception de l'interface multilingue.....	67
Conclusion .....	68
Conclusion Générale .....	69
Perspectives.....	72
Fonctionnalités futures .....	72
Améliorations techniques .....	72



## Liste des figures

Figure 1:Diagramme des cas d'utilisation de l'application Saydaliati .....	36
Figure 2:Diagramme de classes simplifié de l'application Saydaliati .....	37
Figure 3:Architecture MVVM du projet .....	47
Figure 4:Structure du projet.....	48
Figure 5.1:Écran de démarrage de l'application .....	53
Figure 6:Écran d'accueil de l'application .....	54
Figure 7:Interface de recherche de pharmacies .....	55
Figure 8:Liste des pharmacies .....	56
Figure 9:Fiche détaillée d'une pharmacie .....	57
Figure 10:Écran des paramètres de l'application.....	58
Figure 11:Icône de l'application avec badge de notification .....	60
Figure 12:Écran de connexion administrateur.....	61
Figure 13:Tableau de bord administrateur .....	62
Figure 14:Interface de gestion des pharmacies .....	63
Figure 15:Formulaire d'ajout de pharmacie.....	64
Figure 16:Interface de gestion des périodes de garde .....	65
Figure 17: Interface de calendrier des gardes.....	66
Figure 18:Comparaison des interfaces en français et en arabe.....	68

## Liste des tableaux

Table 1:Problèmes identifiés et solutions proposées par Saydaliati .....	15
Table 2: Analyse des risques du projet et stratégies d'atténuation .....	18
Table 3: Analyse comparative des méthodologies de développement.....	23
Table 4:Analyse comparative d'OSMDroid et Google Maps.....	40

# Introduction Générale

## Contexte général

Au Maroc, et particulièrement dans la ville de Tanger, l'accès aux services pharmaceutiques représente un défi quotidien pour de nombreux citoyens. La recherche d'une pharmacie ouverte, surtout en dehors des heures ouvrables habituelles, peut s'avérer complexe et chronophage. Cette difficulté est amplifiée par le système de rotation des pharmacies de garde, qui change régulièrement selon un calendrier établi par les autorités locales.

Parallèlement, nous assistons à une transformation numérique rapide de la société marocaine. Le taux de pénétration des smartphones au Maroc a considérablement augmenté ces dernières années, atteignant plus de 80% en 2024. Cette évolution technologique crée un terrain propice au développement de solutions mobiles innovantes dans divers domaines, y compris celui de la santé.

C'est dans ce contexte que s'inscrit notre projet "Saydaliati", qui ambitionne de tirer parti des technologies mobiles pour faciliter l'accès aux services pharmaceutiques, contribuant ainsi à l'amélioration de la qualité des soins de santé dans notre région.

## Problématique

Malgré l'importance cruciale des pharmacies dans le système de santé, plusieurs problèmes persistent quant à leur accessibilité :

1. **Difficulté d'information sur les pharmacies de garde** : Les informations concernant les pharmacies de garde sont souvent diffusées de manière fragmentée (journaux locaux, affichages physiques, bouche-à-oreille), ce qui rend difficile l'accès à ces informations de manière rapide et fiable, particulièrement en situation d'urgence.
2. **Déficit de géolocalisation précise** : Même lorsqu'un citoyen connaît le nom d'une pharmacie, il peut rencontrer des difficultés à la localiser précisément, surtout dans un contexte urbain en pleine expansion comme celui de Tanger.
3. **Manque de centralisation des données** : Il n'existe pas de système centralisé permettant à la fois aux autorités de publier facilement les rotations des pharmacies de garde et aux citoyens d'accéder à ces informations.
4. **Barrière linguistique** : Une partie de la population peut rencontrer des difficultés linguistiques avec les applications existantes qui ne proposent pas souvent d'interface en arabe.

Face à ces défis, notre problématique peut se résumer ainsi : Comment concevoir une application mobile intuitive et accessible qui permette aux habitants de Tanger de localiser facilement les pharmacies à proximité, en identifiant spécifiquement celles qui sont de garde, tout en fournissant les informations essentielles sur ces établissements ?

## Objectifs du projet

Le projet “Saydaliati” vise à répondre à la problématique identifiée en poursuivant les objectifs suivants :

1. **Développer une application mobile bilingue (français/arabe)** qui répond aux besoins spécifiques des habitants de Tanger en matière d'accès aux services pharmaceutiques.
2. **Mettre en place un système de géolocalisation précis** permettant aux utilisateurs de visualiser sur une carte les pharmacies à proximité de leur position actuelle.
3. **Créer une interface simple et intuitive** permettant à tous les utilisateurs, quel que soit leur niveau de familiarité avec les technologies, de trouver rapidement l'information recherchée.
4. **Implémenter un système de filtrage** permettant d'identifier facilement les pharmacies de garde à une date donnée.
5. **Intégrer des fonctionnalités d'itinéraire** pour guider les utilisateurs vers la pharmacie choisie.
6. **Concevoir une interface d'administration sécurisée** permettant aux autorités compétentes de gérer les données des pharmacies et leurs périodes de garde.
7. **Assurer la fiabilité et la mise à jour régulière des données** présentées aux utilisateurs.
8. **Respecter les normes de développement mobile** en termes de performance, de sécurité et d'expérience utilisateur.

## Méthodologie générale adoptée

Pour mener à bien ce projet, nous avons adopté une approche méthodologique rigoureuse qui s'articule autour des étapes suivantes :

1. **Phase d'étude préliminaire** : Analyse du contexte, identification précise des besoins des utilisateurs et des contraintes techniques à travers des recherches documentaires et des entretiens informels avec des utilisateurs potentiels.
2. **Phase de conception** : Élaboration des spécifications fonctionnelles et non fonctionnelles, conception de l'architecture logicielle, modélisation UML, et design des interfaces utilisateur.
3. **Phase de développement** : Implémentation des fonctionnalités selon une approche Agile (Scrum), avec des sprints de deux semaines permettant des livraisons régulières et des ajustements continus.

4. **Phase de test** : Vérification systématique de chaque fonctionnalité développée, tests d'intégration, tests de performance et tests utilisateurs pour garantir la qualité du produit final.
5. **Phase de déploiement** : Préparation du package de l'application pour distribution, documentation technique et utilisateur.
6. **Phase d'évaluation** : Analyse critique du produit final, identification des forces et faiblesses, et élaboration des perspectives d'amélioration.

Tout au long de ce processus, nous avons accordé une attention particulière à la communication au sein de l'équipe et à la documentation de notre travail, assurant ainsi la traçabilité de nos décisions et la transparence de notre démarche.

## Structure du rapport

Ce rapport est structuré en cinq chapitres principaux, suivant une progression logique qui reflète notre parcours de développement :

- Le **premier chapitre** présente en détail le projet "Saydaliati", son contexte, ses objectifs et son organisation.
- Le **deuxième chapitre** expose la méthodologie de développement adoptée, en justifiant nos choix et en décrivant leur mise en œuvre concrète.
- Le **troisième chapitre** se concentre sur l'analyse et la conception de l'application, présentant les spécifications fonctionnelles et non fonctionnelles, ainsi que la modélisation UML.
- Le **quatrième chapitre** décrit l'environnement de développement utilisé, les outils et technologies mobilisés, ainsi que l'architecture technique de l'application.
- Le **cinquième chapitre** détaille la réalisation concrète de l'application, les fonctionnalités implémentées, les interfaces utilisateur développées, ainsi que les tests et validations effectués.

Le rapport se conclut par un bilan global du projet, une analyse des résultats obtenus par rapport aux objectifs initiaux, et une présentation des perspectives d'évolution de l'application "Saydaliati".

## Chapitre 1 : Présentation du projet “Saydaliati”

## 1.1 Contexte et justification

L'accès aux soins de santé constitue un droit fondamental pour tout citoyen. Dans ce cadre, les pharmacies jouent un rôle crucial en tant que premier point de contact avec le système de santé pour de nombreuses personnes. À Tanger, comme dans l'ensemble du Maroc, le système de pharmacies de garde assure la disponibilité des médicaments et des conseils pharmaceutiques 24h/24, 7j/7.

Cependant, la ville de Tanger connaît une expansion urbaine rapide, avec une population qui a doublé en deux décennies pour atteindre plus d'un million d'habitants en 2024. Cette croissance démographique s'accompagne d'une augmentation significative du nombre de pharmacies, rendant leur localisation et la diffusion des informations sur les gardes de plus en plus complexes.

Les méthodes traditionnelles d'information sur les pharmacies de garde (affichage dans les pharmacies, publications dans les journaux locaux) présentent des limites importantes : - Elles nécessitent un déplacement physique - Elles ne sont pas mises à jour en temps réel - Elles n'offrent pas d'informations sur la localisation précise - Elles sont peu accessibles en situation d'urgence

Notre projet "Saydaliati" se justifie par la nécessité de moderniser ce système d'information, en exploitant les technologies mobiles pour offrir une solution pratique, accessible et efficace. L'application vise à combler un vide dans le paysage numérique local, où aucune solution similaire n'existe spécifiquement pour la ville de Tanger avec une interface bilingue adaptée à la population locale.

Par ailleurs, ce projet s'inscrit dans la stratégie nationale de transformation digitale du Maroc, qui encourage le développement de solutions numériques dans tous les secteurs, et particulièrement dans celui de la santé.

## 1.2 Présentation globale de l'application

L'application "Saydaliati" est une plateforme mobile développée pour Android, conçue pour faciliter l'accès aux services pharmaceutiques dans la ville de Tanger. Elle se présente comme un outil intuitif permettant aux utilisateurs de localiser les pharmacies à proximité de leur position, avec une attention particulière portée aux pharmacies de garde.

L'application se décline en deux interfaces principales :

1. **Interface utilisateur** : Destinée au grand public, elle permet de :
  - Visualiser sur une carte l'ensemble des pharmacies de la ville
  - Filtrer les résultats pour n'afficher que les pharmacies de garde
  - Consulter les informations détaillées de chaque pharmacie (adresse, numéro de téléphone, horaires d'ouverture)
  - Obtenir un itinéraire vers la pharmacie sélectionnée
  - Passer un appel directement depuis l'application
  - Rechercher des pharmacies par nom ou par quartier

- Changer la langue de l’interface (français/arabe)
- 2. **Interface administrateur** : Réservée aux autorités compétentes, elle offre les fonctionnalités suivantes :
  - Gérer la base de données des pharmacies (ajout, modification, suppression)
  - Définir les périodes de garde pour chaque pharmacie
  - Consulter des statistiques d’utilisation

L’application a été développée avec un souci particulier d’accessibilité et d’ergonomie, afin de la rendre utilisable par le plus grand nombre, y compris les personnes peu familières avec les technologies numériques.

### 1.3 Problèmes ciblés et solutions proposées

Notre analyse préliminaire a permis d’identifier plusieurs problèmes spécifiques auxquels “Saydaliati” apporte des solutions concrètes :

*Table 1: Problèmes identifiés et solutions proposées par Saydaliati*

Problème identifié	Solution proposée
Difficulté à trouver rapidement une pharmacie ouverte, surtout en cas d’urgence	Système de géolocalisation permettant de visualiser immédiatement les pharmacies à proximité, avec distinction visuelle des pharmacies de garde
Manque d’information centralisée sur les pharmacies de garde	Base de données régulièrement mise à jour par les administrateurs, affichant en temps réel les pharmacies de garde
Information dispersée sur les coordonnées et horaires des pharmacies	Fiche détaillée pour chaque pharmacie regroupant toutes les informations essentielles en un seul endroit
Difficulté à trouver le chemin vers une pharmacie, surtout dans des quartiers méconnus	Intégration d’un système de navigation guidée vers la pharmacie sélectionnée
Barrière linguistique pour une partie de la population	Interface bilingue (français/arabe) permettant à chaque utilisateur de naviguer dans la langue de son choix
Processus fastidieux de mise à jour des données sur les pharmacies de garde	Interface d’administration intuitive permettant aux autorités de mettre à jour facilement les informations
Absence d’information sur les pharmacies disposant de certains services spécifiques (parking, accès handicapé)	Ajout d’attributs spécifiques dans les fiches pharmacies permettant de filtrer selon ces critères

Cette approche orientée solutions nous a permis de concevoir une application qui répond directement aux besoins réels des utilisateurs, tout en offrant une valeur ajoutée significative par rapport aux méthodes traditionnelles.

## 1.4 Objectifs spécifiques

Pour atteindre l'objectif général de faciliter l'accès aux services pharmaceutiques à Tanger, nous avons défini les objectifs spécifiques suivants :

### 1. Objectifs fonctionnels :

- Développer un système de géolocalisation capable de détecter la position de l'utilisateur avec une précision d'au moins 10 mètres
- Créer une base de données exhaustive comprenant au minimum 80% des pharmacies de Tanger
- Implémenter un algorithme de filtrage efficace permettant d'identifier les pharmacies de garde en moins de 2 secondes
- Mettre en place un système de notification pour informer les utilisateurs des changements de pharmacies de garde
- Développer une interface d'administration sécurisée avec authentification à deux facteurs

### 2. Objectifs techniques :

- Assurer la compatibilité de l'application avec les versions d'Android 8.0 (Oreo) et supérieures
- Optimiser les performances pour maintenir un temps de réponse inférieur à 3 secondes même avec une connexion internet de qualité moyenne
- Limiter la consommation de données mobiles à moins de 5Mo par session d'utilisation typique
- Garantir la sécurité des données personnelles des utilisateurs
- Assurer une consommation énergétique raisonnable pour préserver l'autonomie des appareils

### 3. Objectifs liés à l'expérience utilisateur :

- Concevoir une interface intuitive permettant à un nouvel utilisateur de trouver une pharmacie en moins de 3 clics
- Développer un design responsive adapté aux différentes tailles d'écran
- Assurer une traduction cohérente et naturelle entre les versions française et arabe
- Implementer des fonctionnalités d'accessibilité pour les utilisateurs malvoyants

### 4. Objectifs pédagogiques :

- Approfondir nos connaissances en développement Android natif
- Maîtriser l'intégration des API de géolocalisation et de cartographie
- Développer nos compétences en gestion de base de données mobile
- Expérimenter la méthodologie Agile dans un contexte de projet réel



Ces objectifs spécifiques ont constitué notre feuille de route tout au long du développement, nous permettant de mesurer concrètement notre progression et la qualité de notre travail.

## 1.5 Étude de faisabilité

Avant de nous lancer dans le développement de “Saydaliati”, nous avons réalisé une étude de faisabilité approfondie pour évaluer la viabilité du projet sous différents angles.

### Faisabilité technique

Sur le plan technique, notre analyse a confirmé la possibilité de développer l’application avec les outils et technologies disponibles :

- **SDK Android** : Les API nécessaires à notre projet (géolocalisation, cartographie, stockage local) sont disponibles dans les versions récentes du SDK Android.
- **OSMDroid** : Cette bibliothèque open-source offre toutes les fonctionnalités cartographiques requises.
- **Room Database** : Ce framework de persistance permet de gérer efficacement les données locales.
- **Compétences de l’équipe** : Notre équipe possède les connaissances de base en Java et en développement Android, avec la capacité d’acquérir rapidement les compétences complémentaires nécessaires.

### Faisabilité temporelle

Considérant la durée allouée au projet (un demi-module), nous avons établi un planning réaliste :

- 1 semaine pour l’analyse et la conception
- 4 semaines pour le développement (2 sprints de 2 semaines)
- 1 semaine pour les tests et corrections
- 1 semaine pour la documentation et la préparation de la présentation

Cette répartition nous a paru réalisable compte tenu de la charge de travail estimée et de la disponibilité des membres de l’équipe.

### Faisabilité économique

Les coûts associés au développement ont été jugés minimes : - Utilisation d’outils de développement gratuits (Android Studio) - Recours à des bibliothèques open-source - Absence de coûts d’hébergement grâce au stockage local des données - Équipements personnels (ordinateurs, smartphones) déjà disponibles

## Faisabilité opérationnelle

L'étude a également confirmé que : - L'application répond à un besoin réel de la population - L'interface prévue est suffisamment simple pour être utilisable par le public cible - La maintenance future de l'application est envisageable avec des ressources limitées

## Risques identifiés et stratégies d'atténuation

Notre analyse a identifié plusieurs risques potentiels :

*Table 2: Analyse des risques du projet et stratégies d'atténuation*

Risque	Probabilité	Impact	Stratégie d'atténuation
Difficulté à obtenir des données fiables sur les pharmacies	Moyenne	Élevé	Prévoir une phase de collecte de données sur le terrain et auprès des autorités locales
Problèmes de précision du GPS dans certaines zones urbaines	Élevée	Moyen	Implémenter des mécanismes de correction et de validation des positions
Complexité de l'interface d'administration	Moyenne	Moyen	Prévoir des tests utilisateurs précoces et itérer le design
Temps de développement sous-estimé	Moyenne	Élevé	Prioriser les fonctionnalités essentielles et prévoir une marge de sécurité
Barrières linguistiques dans l'implémentation bilingue	Faible	Moyen	Faire appel à des locuteurs natifs pour valider les traductions

En conclusion, notre étude de faisabilité a démontré que le projet "Saydaliati" était réalisable dans le cadre académique fixé, avec un bon rapport entre l'effort requis et les bénéfices attendus.

## 1.6 Planification et découpage du projet

Pour assurer une progression méthodique et contrôlée, nous avons découpé le projet en plusieurs phases clairement définies.

Le projet a été organisé en cinq phases distinctes :

### 1. Phase d'initialisation (Semaine 1)

- Définition précise du périmètre et des objectifs
- Formation de l'équipe sur les technologies spécifiques
- Mise en place de l'environnement de développement
- Création du backlog initial

### 2. Phase de conception (Semaine 2)

- Élaboration des spécifications détaillées
- Conception de l'architecture logicielle

- Modélisation UML
- Design des interfaces utilisateur
- Préparation du premier sprint
- 3. **Phase de développement (Semaines 3-6)**
  - Sprint 1 : Développement du module de cartographie et de géolocalisation
  - Sprint 2 : Développement du module de gestion des pharmacies et de l'interface utilisateur
  - Revues de code régulières
  - Tests unitaires
- 4. **Phase de finalisation (Semaine 7)**
  - Tests d'intégration
  - Correction des bugs
  - Optimisation des performances
  - Préparation de la version finale
- 5. **Phase de clôture (Semaine 8)**
  - Rédaction de la documentation
  - Préparation de la présentation
  - Évaluation rétrospective du projet

## 1.7 Organisation du travail

La réussite d'un projet de développement repose en grande partie sur une organisation efficace du travail en équipe. Pour "Saydaliati", nous avons mis en place une structure organisationnelle adaptée à la taille de notre équipe et à la nature du projet.

### 1.7.1 Méthodes de communication

Pour assurer une communication efficace au sein de l'équipe, nous avons mis en place plusieurs canaux complémentaires :

- **Réunions hebdomadaires** : Chaque lundi matin, l'équipe se réunissait pour faire le point sur l'avancement du projet, discuter des difficultés rencontrées et planifier les tâches de la semaine.
- **Stand-up meetings quotidiens** : Brèves réunions quotidiennes de 15 minutes où chaque membre partageait ses réalisations de la veille, ses objectifs du jour et les éventuels obstacles rencontrés.
- **Groupe WhatsApp** : Pour les communications rapides et informelles, permettant de partager des informations ou de demander de l'aide en temps réel.
- **Trello** : Pour le suivi des tâches et la visualisation de l'avancement global du projet.
- **GitHub** : Pour la gestion du code source, avec un système de pull requests permettant des revues de code systématiques.

- **Google Drive** : Pour le partage et la collaboration sur les documents du projet (spécifications, maquettes, rapports).

Cette stratégie de communication multicanale nous a permis de maintenir une transparence totale sur l'avancement du projet et de résoudre rapidement les problèmes rencontrés.

### 1.7.2 Gestion des versions et du code source

La gestion efficace du code source a été une priorité dès le début du projet. Nous avons adopté les pratiques suivantes :

- **Utilisation de Git** comme système de gestion de versions, avec un dépôt central hébergé sur GitHub.
- **Revues de code systématiques** via les pull requests.
- **Conventions de codage** définies en début de projet et appliquées rigoureusement pour maintenir la lisibilité et la cohérence du code.
- **Documentation du code** avec des commentaires explicites et une structure de packages cohérente.

Cette organisation nous a permis d'éviter les conflits majeurs lors de l'intégration des différentes fonctionnalités et de maintenir une qualité de code élevée tout au long du développement.

## Conclusion

Ce premier chapitre nous a permis de présenter le projet "Saydaliati" dans sa globalité, en exposant son contexte, ses objectifs et son organisation. Nous avons vu que ce projet répond à un besoin réel de la population de Tanger en matière d'accès aux services pharmaceutiques, en proposant une solution mobile innovante et adaptée au contexte local.

L'étude de faisabilité a confirmé la viabilité du projet tant sur le plan technique que temporel et économique, tout en identifiant les risques potentiels et les stratégies pour les atténuer. La planification détaillée et l'organisation du travail mises en place ont fourni un cadre structuré pour le développement de l'application.

Les choix organisationnels adoptés, notamment la répartition des rôles, les méthodes de communication et la gestion du code source, ont constitué des fondations solides pour la suite du projet. Ils ont permis d'instaurer un environnement de travail collaboratif et efficace, essentiel à la réussite d'un projet de développement logiciel.

Dans le chapitre suivant, nous nous intéresserons plus spécifiquement à la méthodologie de développement adoptée pour ce projet, en détaillant notre approche Agile et son application concrète dans le cadre de "Saydaliati".

## Chapitre 2 : Méthodologie de développement

## 2.1 Introduction

La méthodologie de développement constitue l'épine dorsale de tout projet informatique, définissant la manière dont les équipes collaborent, planifient et exécutent les différentes phases du cycle de vie du logiciel. Le choix d'une méthodologie adaptée est déterminant pour la réussite d'un projet, notamment dans le contexte du développement mobile où les exigences peuvent évoluer rapidement et où l'expérience utilisateur occupe une place centrale.

Ce chapitre présente la démarche méthodologique que notre équipe a adoptée pour le développement de l'application "Saydaliati". Nous y expliquons le processus de sélection de la méthodologie en fonction des spécificités de notre projet, puis détaillons la mise en œuvre concrète de cette approche à travers nos pratiques de développement quotidiennes.

Nous abordons également les outils de gestion qui ont soutenu notre méthodologie, en expliquant comment ils ont été intégrés dans notre workflow et quelle valeur ils ont apportée au projet. Enfin, nous analysons les avantages et les défis rencontrés avec l'approche choisie, et la manière dont nous avons adapté la méthodologie à nos besoins spécifiques.

Cette présentation méthodologique vise à offrir une compréhension claire de notre processus de développement, ainsi qu'un retour d'expérience potentiellement utile pour d'autres projets similaires.

## 2.2 Choix de la méthode de développement

Le choix de la méthodologie de développement est une décision stratégique qui doit prendre en compte de nombreux facteurs, notamment la nature du projet, la taille de l'équipe, les contraintes temporelles et les spécificités du domaine d'application. Pour "Saydaliati", nous avons procédé à une analyse comparative des principales méthodologies envisageables.

### 2.2.1 Analyse comparative des méthodologies

Nous avons évalué plusieurs approches selon des critères pertinents pour notre contexte :

Cette analyse nous a permis d'identifier les principales forces et faiblesses de chaque méthodologie dans notre contexte spécifique.

Table 3: Analyse comparative des méthodologies de développement

Critère	Cascade (Waterfall)	Agile (Scrum)	RAD	DevOps
Flexibilité face aux changements	Faible	Élevée	Moyenne	Moyenne
Implication des utilisateurs	Limitée (début et fin)	Continue	Importante	Moyenne
Adaptabilité à une petite équipe	Moyenne	Élevée	Moyenne	Faible
Compatibilité avec un calendrier académique	Moyenne	Élevée	Moyenne	Faible
Documentation produite	Extensive	Modérée	Limitée	Modérée
Courbe d'apprentissage	Faible	Moyenne	Moyenne	Élevée
Adéquation au développement mobile	Faible	Élevée	Moyenne	Moyenne

### 2.2.2 Facteurs déterminants du choix

Plusieurs facteurs ont orienté notre décision vers la méthodologie Agile, et plus précisément le cadre Scrum :

1. **Incertitude initiale sur certaines fonctionnalités** : Au début du projet, nous n'avions pas une vision exhaustive de toutes les fonctionnalités à implémenter, notamment concernant l'interface d'administration. Une approche itérative permettait d'affiner progressivement notre compréhension.
2. **Besoin de feedback régulier** : Le développement d'une application mobile centrée sur l'utilisateur nécessitait des retours fréquents pour valider l'expérience utilisateur et l'ergonomie.
3. **Équipe de petite taille et co-localisée** : Notre équipe de trois membres partageait le même espace de travail, facilitant la communication directe et les pratiques agiles.
4. **Contraintes académiques** : Le cadre du projet de fin de demi-module imposait des délais fixes avec des jalons intermédiaires, correspondant bien à l'organisation en sprints du Scrum.
5. **Compétences techniques évolutives** : Certains aspects techniques (notamment l'intégration des API cartographiques) représentaient un défi nouveau pour l'équipe, justifiant une approche progressive.
6. **Importance de la documentation** : Bien que les méthodes agiles privilégient souvent "le logiciel opérationnel à la documentation exhaustive", le contexte académique exigeait

une documentation substantielle, que nous avons intégrée comme livrable à part entière de nos sprints.

### 2.2.3 Justification du choix d'Agile-Scrum

La méthodologie Agile avec le cadre Scrum s'est imposée comme le meilleur compromis entre flexibilité et rigueur, pour plusieurs raisons :

- **Adaptabilité aux changements** : La nature itérative de Scrum permettait d'intégrer de nouvelles exigences ou de modifier les fonctionnalités existantes au fil du développement.
- **Livraisons incrémentales** : La possibilité de développer une version minimale fonctionnelle (MVP) puis de l'enrichir progressivement correspondait parfaitement à notre vision du projet.
- **Transparence et visibilité** : Les cérémonies Scrum (planning, daily, review, retrospective) offraient un cadre structuré pour maintenir une vision claire de l'avancement et des obstacles.
- **Amélioration continue** : Les rétrospectives de sprint permettaient d'identifier et de corriger rapidement les problèmes de processus.
- **Équilibre entre développement et documentation** : Contrairement à certaines idées reçues, Scrum peut s'accommoder d'exigences de documentation importantes, en les intégrant simplement comme des tâches dans le backlog.

Cette approche nous a semblé la plus adaptée pour conjuguer les contraintes académiques du projet avec les bonnes pratiques du développement mobile moderne.

## 2.3 Présentation de la méthode Agile

La méthode Agile représente plus qu'une simple méthodologie de développement logiciel ; c'est une philosophie qui place l'humain, la collaboration et l'adaptabilité au cœur du processus de création. Cette approche s'est développée en réaction aux limitations des méthodes traditionnelles séquentielles, jugées trop rigides face à l'évolution rapide des besoins et des technologies.

### 2.3.1 Principes fondamentaux de l'Agilité

La méthode Agile repose sur quatre valeurs fondamentales, énoncées dans le "Manifeste Agile" publié en 2001 :

- **Les individus et leurs interactions** plus que les processus et les outils
- **Un logiciel opérationnel** plus qu'une documentation exhaustive
- **La collaboration avec les clients** plus que la négociation contractuelle
- **L'adaptation au changement** plus que le suivi d'un plan



Ces valeurs se déclinent en douze principes qui guident les pratiques agiles au quotidien, parmi lesquels :

- Livrer fréquemment un logiciel opérationnel
- Accueillir favorablement les changements
- Maintenir un rythme de développement soutenable
- Privilégier la simplicité
- Favoriser l'auto-organisation des équipes
- Réfléchir régulièrement aux moyens de devenir plus efficace

### 2.3.2 Les principales méthodes Agiles

Si le Manifeste Agile définit un socle commun de valeurs et de principes, plusieurs méthodes agiles concrètes en ont émergé, chacune avec ses spécificités :

- **Scrum** : Cadre de travail léger définissant des rôles, des événements et des artefacts précis pour structurer le développement en itérations (sprints) de durée fixe.
- **Kanban** : Approche visuelle fondée sur la limitation du travail en cours et l'optimisation du flux de valeur, sans itérations fixes.
- **Extreme Programming (XP)** : Méthode centrée sur les pratiques techniques d'excellence (programmation en binôme, TDD, intégration continue).
- **Lean Software Development** : Adaptation des principes du Lean Manufacturing au développement logiciel, visant à éliminer le gaspillage et à optimiser la création de valeur.
- **Feature-Driven Development (FDD)** : Approche axée sur les fonctionnalités, combinant modélisation et développement itératif.

Pour notre projet "Saydaliati", nous avons choisi Scrum comme cadre principal, tout en intégrant certaines pratiques d'Extreme Programming pour renforcer la qualité technique du développement.

### 2.3.3 Avantages et limites de l'approche Agile

L'adoption d'une méthode Agile présente plusieurs avantages significatifs :

- **Réactivité face aux changements** : Capacité à intégrer rapidement de nouvelles exigences ou à pivoter si nécessaire.
- **Satisfaction client accrue** : Implication continue des parties prenantes et livraisons fréquentes permettant des ajustements réguliers.
- **Qualité améliorée** : Détection précoce des problèmes grâce aux tests continus et aux revues régulières.
- **Visibilité accrue** : Transparence sur l'avancement et les obstacles rencontrés.
- **Motivation de l'équipe** : Autonomie et responsabilisation favorisant l'engagement.

Cependant, cette approche comporte également certaines limites :

- **Besoin d'implication des parties prenantes** : L'absence de disponibilité des utilisateurs peut compromettre le processus.
- **Documentation parfois insuffisante** : Tendance à minimiser la documentation formelle, pouvant poser problème pour la maintenance à long terme.
- **Difficulté de planification à long terme** : Vision parfois floue de la date de fin et du coût total du projet.
- **Exigence de maturité de l'équipe** : Nécessite une autodiscipline et une communication efficace.

Dans notre contexte académique, nous avons dû adapter l'approche Agile pour contourner certaines de ces limitations, notamment en renforçant la documentation et en établissant un cadre temporel fixe correspondant à la durée du module.

## 2.4 Application du cadre Scrum au projet

Le cadre Scrum a été adapté aux spécificités de notre projet "Saydaliati" et au contexte académique dans lequel il s'inscrivait. Cette section détaille la manière dont nous avons mis en œuvre Scrum dans notre projet.

Cette organisation souple a permis à chacun d'avoir un rôle principal tout en contribuant à l'ensemble des aspects du projet selon les besoins.

### 2.4.1 Sprint Planning et backlog

Notre projet a été organisé en deux sprints de deux semaines chacun :

- **Sprint 1** : Focalisé sur le développement des fonctionnalités de base (géolocalisation, carte, liste des pharmacies)
- **Sprint 2** : Consacré aux fonctionnalités avancées (filtrage des pharmacies de garde, interface d'administration, multilinguisme)

Pour chaque sprint, nous avons suivi un processus de planification structuré :

1. **Raffinement du Product Backlog** : Avant chaque sprint planning, nous précisons les user stories prioritaires et estimons leur complexité en story points, en utilisant la suite de Fibonacci (1, 2, 3, 5, 8, 13).
2. **Sprint Planning Meeting** : Au début de chaque sprint, une réunion de 2 heures nous permettait de :
  - Définir l'objectif du sprint
  - Sélectionner les user stories à intégrer dans le sprint backlog
  - Décomposer ces stories en tâches techniques concrètes
  - Estimer le temps nécessaire pour chaque tâche
  - S'engager collectivement sur le contenu du sprint

3. **Organisation du Sprint Backlog** : Nous avons utilisé Trello pour visualiser notre sprint backlog avec quatre colonnes :

- “À faire” : Tâches à réaliser
- “En cours” : Tâches en cours de réalisation
- “À vérifier” : Tâches terminées en attente de validation
- “Terminé” : Tâches validées et intégrées

Voici un exemple représentatif de notre Product Backlog pour le Sprint 1 :

User Story	Priorité	Story Points	Critères d'acceptation
En tant qu'utilisateur, je veux voir ma position actuelle sur une carte	Haute	8	- Demande de permission de localisation - Affichage précis de la position- Mise à jour en temps réel
En tant qu'utilisateur, je veux voir les pharmacies à proximité sur la carte	Haute	5	-Marqueurs distincts pour les pharmacies -Liste des pharmacies correspondantes- Distance par rapport à ma position
En tant qu'utilisateur, je veux pouvoir changer la date pour voir les pharmacies de garde	Moyenne	5	- Sélecteur de date fonctionnel- Mise à jour des pharmacies affichées - Indication visuelle des pharmacies de garde
En tant qu'utilisateur, je veux consulter les détails d'une pharmacie	Moyenne	3	- Fiche détaillée avec toutes les informations - Bouton d'appel fonctionnel- Bouton d'itinéraire fonctionnel
En tant qu'utilisateur, je veux pouvoir basculer entre français et arabe	Basse	8	- Interface complètement traduite -Mémorisation du choix de langue- Adaptation du layout pour l'arabe (RTL)

## 2.4.2 Suivi des itérations

Pour assurer un suivi efficace de notre progression, nous avons mis en place plusieurs pratiques Scrum :

1. **Daily Scrum** : Chaque jour, nous tenions une réunion debout de 15 minutes où chaque membre répondait à trois questions :

- Qu'ai-je accompli hier ?

- Que vais-je faire aujourd’hui ?
- Y a-t-il des obstacles qui m’empêchent d’avancer ?

Ces réunions, bien que brèves, nous permettaient de maintenir une vision commune de l’avancement et d’identifier rapidement les problèmes.

2. **Tableau Kanban** : Notre tableau Trello était mis à jour quotidiennement, offrant une visualisation claire du travail en cours et du reste à faire. Nous y ajoutions des étiquettes colorées pour indiquer la nature des tâches (frontend, backend, documentation, tests) et leur criticité.
3. **Burndown Chart** : Nous utilisons un graphique de progression pour suivre la consommation des story points au fil du sprint, nous permettant de détecter tout écart par rapport à la progression idéale.
4. **Sprint Review** : À la fin de chaque sprint, nous organisons une démonstration des fonctionnalités développées. Notre encadrante, Mme KHALFAOUI, participait à ces revues, fournissant un feedback précieux qui était pris en compte pour le sprint suivant.
5. **Sprint Retrospective** : Après chaque revue, nous prenons le temps de réfléchir à notre processus en identifiant :
  - Ce qui avait bien fonctionné (à conserver)
  - Ce qui avait posé problème (à améliorer)
  - Les actions concrètes à mettre en place pour le sprint suivant

Cette approche structurée nous a permis de maintenir un rythme de développement soutenu tout en nous adaptant aux difficultés rencontrées et aux retours reçus.

## 2.5 Outils de gestion utilisés

La mise en œuvre efficace de notre méthodologie Agile a nécessité l’utilisation de plusieurs outils complémentaires, facilitant la collaboration et le suivi du projet. Ces outils ont été choisis pour leur simplicité d’utilisation, leur accessibilité et leur adéquation avec nos besoins spécifiques.

### 2.5.1 Trello pour la gestion du backlog

Trello a constitué notre outil central pour la gestion des tâches et la visualisation de notre avancement. Nous l’avons choisi pour sa flexibilité et son interface intuitive, qui permettait à tous les membres de l’équipe de l’adopter rapidement.

Notre tableau Trello était organisé selon la structure suivante :

- **Liste “Product Backlog”** : Contenant toutes les user stories à développer, triées par priorité.
- **Liste “Sprint Backlog”** : Regroupant les user stories sélectionnées pour le sprint en cours.

- **Liste “À faire”** : Contenant les tâches techniques décomposées à partir des user stories, prêtes à être attribuées.
- **Liste “En cours”** : Tâches actuellement en développement, avec le nom du membre responsable.
- **Liste “À vérifier”** : Tâches terminées mais nécessitant une revue ou des tests avant validation.
- **Liste “Terminé”** : Tâches complètement achevées et validées.
- **Liste “Bloqué”** : Tâches rencontrant des obstacles nécessitant une intervention particulière.

Pour chaque carte Trello, nous utilisons : - Des étiquettes colorées pour indiquer le type de tâche (frontend, backend, documentation, etc.) - Des échéances pour les tâches critiques - Des checklists détaillant les critères d’acceptation - Des commentaires pour documenter les décisions ou difficultés rencontrées

Cet outil nous a permis de maintenir une vision claire et partagée de l’état d’avancement du projet à tout moment.

### 2.5.2 GitHub pour la gestion du code source

GitHub a été notre plateforme de choix pour la gestion du code source et la collaboration technique. Nous avons configuré notre dépôt avec une structure de branches cohérente avec notre approche Agile :

- Branche *main* : Contenant uniquement le code stable et déployable
- Branche *develOp* : Intégrant progressivement les fonctionnalités validées
- Branches *feature/xxx* : Créées pour chaque nouvelle fonctionnalité
- Branches *bugfix/xxx* : Dédiées à la correction des bugs identifiés

Notre workflow GitHub incluait :

- **Pull Requests** pour toute fusion de code, avec revue obligatoire par au moins un autre membre de l’équipe
- **Issues** pour le suivi des bugs et des améliorations, liées aux cartes Trello correspondantes
- **Actions automatisées** pour les tests unitaires sur chaque pull request
- **Wiki** pour la documentation technique
- **Project Board** complémentaire au tableau Trello, plus orienté sur les aspects techniques

L’intégration entre GitHub et Trello (via des extensions) nous permettait de maintenir une traçabilité entre les tâches et leur implémentation concrète.

### 2.5.3 Google Drive pour la documentation

Pour la gestion collaborative des documents non techniques (spécifications, maquettes, rapport), nous avons utilisé Google Drive, qui offrait plusieurs avantages :

- **Édition collaborative en temps réel** permettant à plusieurs membres de travailler simultanément sur le même document
- **Contrôle de version** gardant une trace des modifications apportées
- **Commentaires** facilitant les discussions sur des points spécifiques
- **Accessibilité** depuis différents appareils et lieux
- **Organisation hiérarchique** des fichiers en dossiers thématiques

Notre structure de dossiers comprenait :

- Documentation administrative (planning, comptes-rendus de réunions)
- Spécifications fonctionnelles et techniques
- Ressources (logos, icônes, références)
- Rapport et présentation

Cet outil a grandement facilité notre collaboration sur les aspects documentaires du projet, tout en assurant que chaque membre disposait toujours de la version la plus récente des documents.

### 2.5.4 WhatsApp pour la communication d'équipe

Pour la communication quotidienne et informelle au sein de l'équipe, nous avons créé un groupe WhatsApp dédié au projet. Cet outil nous a permis de :

- Partager rapidement des informations ou des questions
- Coordonner les réunions et sessions de travail
- Envoyer des captures d'écran pour discuter de problèmes spécifiques
- Maintenir un contact permanent entre les membres
- Célébrer collectivement les étapes franchies

Bien que moins formelle que les autres outils, cette communication continue a joué un rôle crucial dans la cohésion de l'équipe et la résolution rapide des problèmes rencontrés.

## Conclusion

Ce deuxième chapitre a permis d'exposer en détail la méthodologie de développement adoptée pour notre projet "Saydaliati". Nous avons justifié notre choix de l'approche Agile et du cadre Scrum comme étant les plus adaptés à notre contexte, offrant la flexibilité nécessaire tout en fournissant un cadre structuré pour notre travail.

La présentation des principes fondamentaux de la méthode Agile a mis en lumière les valeurs qui ont guidé notre démarche : la collaboration, l'adaptabilité, la livraison incrémentale de valeur et

la focalisation sur les besoins des utilisateurs. Ces principes se sont traduits concrètement dans notre organisation, avec une répartition claire des rôles et des responsabilités, et dans notre planification structurée en sprints.

L'application du cadre Scrum a été adaptée aux spécificités de notre équipe de trois personnes et aux contraintes académiques du projet. Nous avons mis en place des pratiques et des cérémonies adaptées à notre échelle, tout en préservant l'essence de la méthode : daily scrums, sprint planning, revues et rétrospectives.

Les outils de gestion utilisés (Trello, GitHub, Google Drive, WhatsApp) ont joué un rôle crucial dans le succès de notre approche, en facilitant la collaboration, la transparence et le suivi de l'avancement. Leur complémentarité a permis de couvrir tous les aspects du projet, des tâches techniques à la documentation en passant par la communication d'équipe.

L'expérience acquise avec cette méthodologie nous a non seulement permis de mener à bien le développement de "Saydaliati", mais nous a également fourni des compétences précieuses en gestion de projet et en développement collaboratif, transférables à nos futures carrières professionnelles.

Dans le chapitre suivant, nous aborderons l'analyse et la conception détaillées de l'application, en présentant les spécifications fonctionnelles et non fonctionnelles, ainsi que la modélisation UML qui a guidé notre implémentation.

## Chapitre 3 : Analyse et Conception de l'application



# Introduction

L'analyse et la conception constituent des étapes cruciales dans le cycle de développement logiciel, permettant de traduire les besoins identifiés en spécifications techniques exploitables. Ce chapitre présente la démarche analytique adoptée pour l'application "Saydaliati" et détaille les modèles de conception qui ont guidé notre implémentation.

L'objectif principal de cette phase a été de définir une structure solide pour notre application, en identifiant clairement les fonctionnalités requises, les interactions entre les différents composants, et les contraintes techniques à respecter. Cette approche méthodique nous a permis d'anticiper les défis de développement et d'établir une base conceptuelle cohérente pour l'ensemble du projet.

Nous commencerons par présenter les spécifications fonctionnelles et non fonctionnelles de l'application, puis nous identifierons les principaux acteurs et cas d'utilisation. Ensuite, nous détaillerons la modélisation UML réalisée, notamment à travers les diagrammes de cas d'utilisation et de classes. Enfin, nous aborderons les choix techniques effectués et les contraintes prises en compte lors de la conception.

## 3.1 Spécifications fonctionnelles

Les spécifications fonctionnelles définissent de manière précise ce que l'application "Saydaliati" doit permettre aux utilisateurs de réaliser. Elles ont été établies à partir de l'analyse des besoins et des problèmes identifiés dans le chapitre 1.

### 3.1.2 Module utilisateur

Le module utilisateur constitue l'interface principale de l'application, destinée au grand public. Il comprend les fonctionnalités suivantes :

1. **Géolocalisation et cartographie**
  - Affichage de la position actuelle de l'utilisateur sur une carte interactive
  - Visualisation des pharmacies à proximité avec des marqueurs distinctifs
  - Calcul et affichage de la distance entre l'utilisateur et chaque pharmacie
  - Mise à jour de la position en temps réel lors des déplacements
2. **Recherche de pharmacies**
  - Recherche par nom de pharmacie
  - Recherche par proximité géographique
  - Filtrage des pharmacies selon différents critères (services disponibles, horaires)
  - Affichage des résultats sur la carte et sous forme de liste
3. **Gestion des pharmacies de garde**
  - Identification visuelle des pharmacies de garde sur la carte
  - Filtrage spécifique pour n'afficher que les pharmacies de garde

- Sélection de date pour consulter les pharmacies de garde à une date précise
- Notifications automatiques lors des changements de pharmacies de garde
- 4. **Consultation des détails**
  - Affichage des informations détaillées pour chaque pharmacie (nom, adresse, téléphone, horaires)
  - Option d'appel direct depuis la fiche de la pharmacie
  - Génération d'itinéraires vers la pharmacie sélectionnée
  - Partage des informations de la pharmacie via les applications du téléphone
- 5. **Paramètres et préférences**
  - Changement de langue (français/arabe)
  - Personnalisation des notifications
  - Définition d'un rayon de recherche préféré
  - Sauvegarde des pharmacies favorisées

### 3.2.2 Module administrateur

Le module administrateur est réservé aux autorités compétentes et offre des fonctionnalités de gestion des données :

1. **Authentification sécurisée**
  - Connexion avec identifiant et mot de passe
  - Gestion des sessions et déconnexion
  - Récupération de mot de passe
2. **Gestion des pharmacies**
  - Ajout de nouvelles pharmacies avec leurs coordonnées complètes
  - Modification des informations existantes
  - Suppression de pharmacies (avec contrôle d'impact)
  - Validation des données entrées
3. **Gestion des périodes de garde**
  - Attribution des périodes de garde aux pharmacies
  - Planification des rotations sur plusieurs semaines
  - Modification des plannings de garde
  - Gestion des exceptions et cas particuliers
4. **Import/Export de données**
  - Importation de données depuis des fichiers Excel
  - Exportation des listes de pharmacies et plannings de garde
  - Sauvegarde et restauration de la base de données
5. **Tableau de bord et statistiques**
  - Visualisation des statistiques d'utilisation
  - Suivi des mises à jour effectuées
  - Analyse de la couverture géographique
  - Rapports sur les problèmes signalés

## 3.2 Identification des utilisateurs et cas d'usage

L'analyse des besoins nous a permis d'identifier deux types principaux d'utilisateurs (acteurs) qui interagiront avec l'application "Saydaliati" :

### 3.4.1 Types d'utilisateurs

#### 1. **Utilisateur standard :**

- Représente le grand public, les citoyens cherchant une pharmacie
- Accès aux fonctionnalités de recherche, consultation et navigation
- Aucune authentification requise
- Permissions limitées aux actions de consultation

#### 2. **Administrateur :**

- Représente les autorités responsables de la gestion des pharmacies et des gardes
- Accès au module d'administration après authentification
- Permissions étendues pour la modification des données
- Responsable de la validité et de la mise à jour des informations

### 3.4.2 Principaux cas d'usage

L'analyse fonctionnelle nous a permis d'identifier les principaux cas d'usage de l'application, présentés dans le diagramme de cas d'utilisation (Figure 3.1).

Pour l'utilisateur standard, les cas d'usage principaux sont :

- Rechercher une pharmacie (par nom ou par proximité)
- Recevoir des notifications
- Filtrer les pharmacies (notamment par statut de garde)
- Consulter les détails d'une pharmacie
- Extensions : appeler la pharmacie, ouvrir l'itinéraire

Pour l'administrateur, les cas d'usage principaux sont :

- S'authentifier
- Gérer les pharmacies (ajouter, modifier, supprimer)
- Assigner une période de garde
- Consulter les statistiques
- Importer des données Excel

Ces cas d'usage sont détaillés dans le diagramme UML présenté dans la section suivante.

### 3.3 Modélisation UML

La modélisation UML (Unified Modeling Language) nous a permis de représenter visuellement les différents aspects de notre application, facilitant ainsi la communication au sein de l'équipe et guidant l'implémentation technique.

#### 3.5.1 Diagramme des cas d'utilisation

Le diagramme des cas d'utilisation (Figure 3.1) illustre les interactions entre les utilisateurs (acteurs) et le système. Il met en évidence les fonctionnalités offertes par l'application et les relations entre ces fonctionnalités.

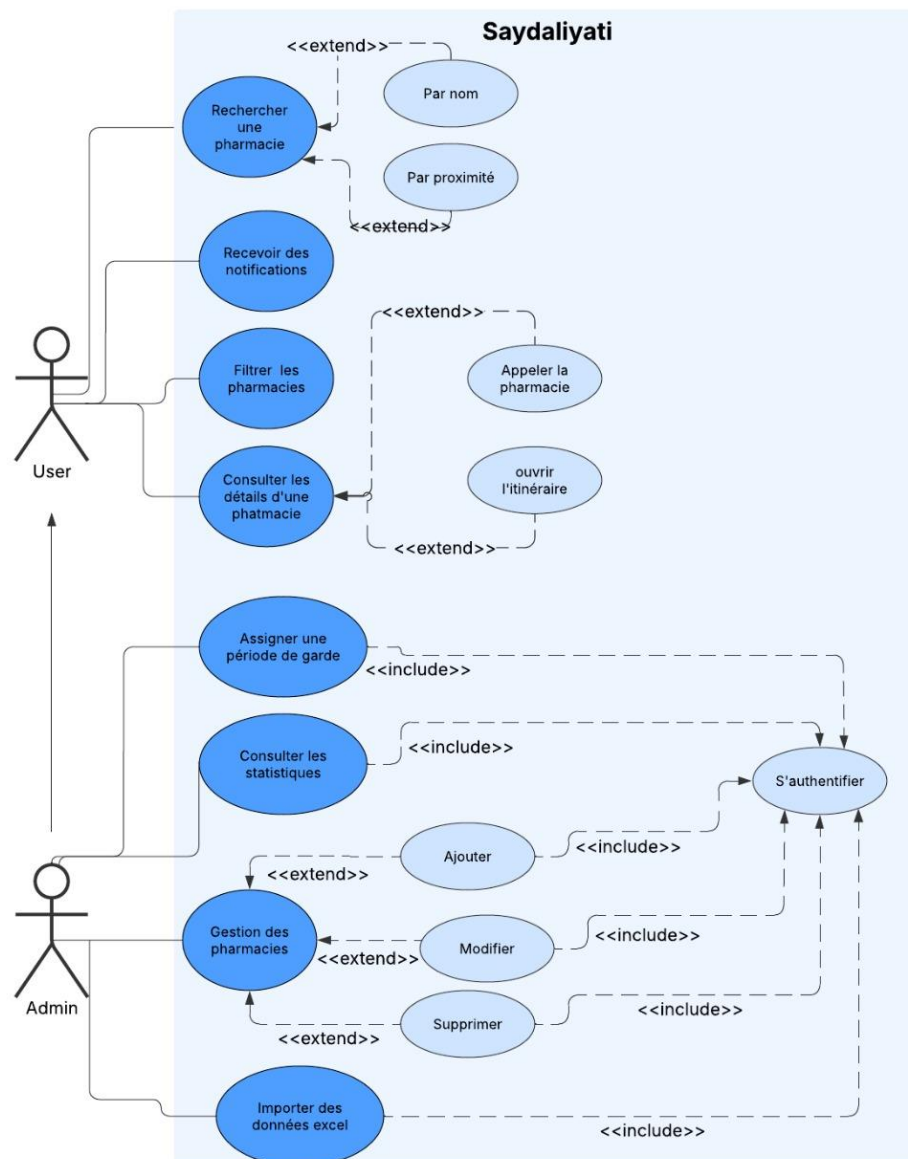


Figure 1: Diagramme des cas d'utilisation de l'application Saydaliyati

Le diagramme montre clairement la séparation entre les fonctionnalités accessibles à l'utilisateur standard et celles réservées à l'administrateur. Les relations d'extension (<<extend>>) indiquent les fonctionnalités optionnelles qui enrichissent un cas d'usage de base, tandis que les relations d'inclusion (<<include>>) représentent des fonctionnalités obligatoires nécessaires à l'exécution d'un cas d'usage.

### 3.5.2 Diagramme de classes

Le diagramme de classes représente la structure statique de l'application, montrant les classes, leurs attributs, leurs méthodes et les relations entre elles. Pour "Saydaliati", nous avons conçu un diagramme de classes qui reflète l'architecture MVVM adoptée.

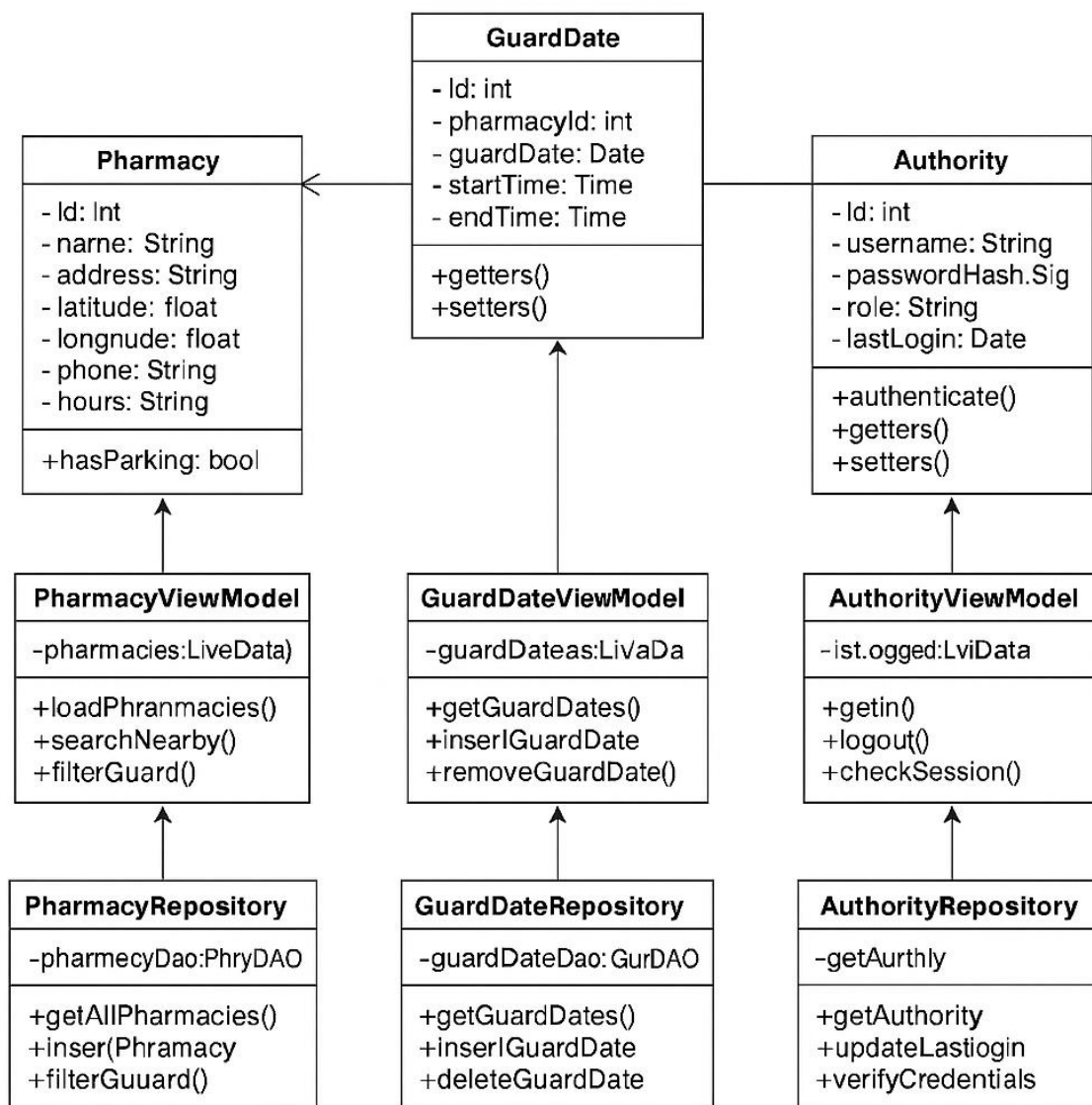


Figure 2: Diagramme de classes simplifié de l'application Saydaliati

Le diagramme met en évidence les trois entités principales du domaine (Pharmacy, GuardDate et Authority) et leur intégration dans l'architecture MVVM à travers les couches Repository et ViewModel. Cette structure assure une séparation claire des responsabilités et facilite la maintenance du code.

## 3.6 Choix techniques et contraintes

La phase de conception a nécessité de prendre plusieurs décisions techniques importantes, en tenant compte des contraintes identifiées et des objectifs du projet.

### 3.6.1 Architecture logicielle

Nous avons opté pour l'architecture MVVM (Model-View-ViewModel) pour plusieurs raisons :

- **Séparation des préoccupations** : MVVM permet une séparation claire entre l'interface utilisateur, la logique de présentation et les données.
- **Testabilité** : Cette architecture facilite les tests unitaires, particulièrement pour la logique de présentation.
- **Support des composants Architecture d'Android** : MVVM s'intègre parfaitement avec les composants officiels comme LiveData et ViewModel.
- **Gestion efficace du cycle de vie** : Les ViewModels survivent aux changements de configuration (comme la rotation de l'écran), préservant ainsi l'état de l'application.

### 3.6.2 Persistance des données

Pour la gestion des données locales, nous avons choisi d'utiliser Room Database :

- **API orientée objet** : Room offre une couche d'abstraction au-dessus de SQLite, facilitant les opérations CRUD.
- **Vérification à la compilation** : Les requêtes SQL sont vérifiées à la compilation, réduisant les erreurs d'exécution.
- **Intégration avec LiveData** : Room s'intègre nativement avec LiveData pour la réactivité de l'interface.
- **Migrations simplifiées** : Room facilite la gestion des mises à jour du schéma de base de données.

### 3.6.3 Cartographie et géolocalisation

Après évaluation des différentes options, nous avons sélectionné OSMDroid pour l'implémentation cartographique :

- **Open source** : Contrairement à Google Maps, OSMDroid ne nécessite pas de clé API payante.
- **Légèreté** : OSMDroid a un impact minimal sur la taille de l'application.

- **Personnalisation** : La bibliothèque offre une grande flexibilité pour l'adaptation de l'affichage.
- **Support hors ligne** : OSMDroid permet le téléchargement et l'utilisation de cartes hors ligne.

Table 4: Analyse comparative d'OSMDroid et Google Maps

Critère	OSMDroid	Google Maps API
Licence	Open-source (Apache 2.0)	Propriétaire avec quota gratuit limité
Coût	Gratuit	Gratuit jusqu'à un certain nombre de requêtes, puis payant
Configuration	Pas de clé API requise	Nécessite une clé API et configuration Google Cloud
Personnalisation	Hautement personnalisable	Options de personnalisation limitées par l'API
Couverture de données	Basé sur OpenStreetMap, moins détaillé dans certaines régions	Couverture mondiale très détaillée
Fonctionnalités	Fonctionnalités de base pour la cartographie	Ensemble riche de fonctionnalités (Street View, directions, etc.)
Taille de l'application	Impact minimal sur la taille de l'APK	Augmente significativement la taille de l'APK
Support hors ligne	Support natif pour les cartes hors ligne	Nécessite une implémentation personnalisée complexe
Documentation	Documentation limitée, communauté plus petite	Documentation extensive, large communauté
Mise à jour des données	Dépend des mises à jour OpenStreetMap	Mises à jour fréquentes et automatiques

### 3.6.4 Contraintes prises en compte

Plusieurs contraintes ont influencé nos choix de conception :

- **Performances sur appareils modestes** : L'application doit fonctionner correctement sur des smartphones aux ressources limitées, ce qui a orienté nos choix vers des solutions légères.



- **Utilisation de données mobiles** : La consommation de données a été optimisée pour minimiser l'impact sur le forfait de l'utilisateur.
- **Accessibilité** : L'interface a été conçue pour être utilisable par le plus grand nombre, incluant des personnes peu familières avec les technologies.
- **Multilinguisme** : La nécessité de supporter à la fois le français et l'arabe a influencé la conception des layouts et la gestion des chaînes de caractères.
- **Sécurité** : Les exigences de sécurité pour l'interface d'administration ont dicté des choix spécifiques en matière d'authentification et de protection des données.

## Conclusion

Ce chapitre a présenté l'analyse et la conception détaillées de l'application "Saydaliati". Nous avons défini les spécifications fonctionnelles et non fonctionnelles qui encadrent le développement, identifié les acteurs principaux et leurs interactions avec le système, et élaboré des modèles UML pour représenter la structure et le comportement de l'application.

La modélisation UML, notamment à travers les diagrammes de cas d'utilisation et de classes, nous a fourni une vision claire et structurée de l'application à développer. Cette phase de conception a permis d'établir des fondations solides pour la phase d'implémentation, en garantissant que l'équipe partage une compréhension commune des objectifs et des moyens techniques pour les atteindre.

Les choix techniques effectués, comme l'adoption de l'architecture MVVM, l'utilisation de Room pour la persistance des données, et la sélection d'OSMDroid pour la cartographie, reflètent notre souci d'équilibrer les exigences fonctionnelles, les contraintes techniques et les bonnes pratiques de développement.

Cette phase de conception constitue ainsi un pont essentiel entre l'analyse des besoins présentée au chapitre 1 et l'implémentation technique qui sera détaillée dans les chapitres suivants, assurant la cohérence et la qualité de l'ensemble du projet "Saydaliati".

## Chapitre 4 : Environnement de développement

# Introduction

L'environnement de développement constitue la fondation technique sur laquelle repose tout projet de développement logiciel. Pour une application mobile comme "Saydaliati", le choix judicieux des outils, frameworks et langages est déterminant pour la qualité finale du produit, sa maintenabilité et sa performance.

Ce chapitre présente en détail l'écosystème technique que nous avons mis en place pour le développement de notre application. Nous y décrivons les outils principaux utilisés, notamment Android Studio et le SDK Android, ainsi que les langages et bibliothèques qui ont constitué notre stack technologique.

Nous expliquons également les choix d'architecture logicielle retenus, la structure organisationnelle du code source, et les mécanismes mis en œuvre pour assurer la sécurité et la gestion des permissions de l'application. Ces aspects techniques, bien qu'invisibles pour l'utilisateur final, sont essentiels pour garantir un produit robuste, évolutif et sécurisé.

Ce chapitre vise à offrir une vision claire et complète des fondations techniques de "Saydaliati", mettant en lumière les décisions qui ont structuré notre approche de développement.

## 4.1 Présentation d'Android Studio et du SDK

Android Studio et le SDK Android ont constitué l'épine dorsale de notre environnement de développement, fournissant un cadre complet pour la création de notre application mobile.

### 4.2.1 Android Studio comme IDE principal

Android Studio, l'environnement de développement intégré (IDE) officiel pour Android, a été notre outil principal de développement. Nous avons utilisé la version 2023.2.1 (Hedgehog), qui offrait plusieurs avantages significatifs pour notre projet :

- **Éditeur de code avancé** avec complétion intelligente, refactoring, et analyse de code en temps réel
- **Interface graphique intuitive** pour la conception des layouts XML via l'éditeur de mise en page visuel
- **Émulateur Android** intégré permettant de tester l'application sur différents appareils virtuels
- **Profiler** pour analyser les performances et détecter les fuites de mémoire
- **Système de build** basé sur Gradle offrant une grande flexibilité dans la configuration du projet
- **Intégration Git** native facilitant la gestion des versions
- **Système de vérification d'erreurs** en temps réel (lint)



Android Studio nous a permis d'optimiser notre workflow de développement grâce à ses nombreuses fonctionnalités spécialisées pour le développement Android, comme l'aperçu

instantané des layouts, la navigation simplifiée entre les ressources et le code, et les outils de debugging avancés.

#### 4.2.2 SDK Android et niveaux d'API

Le SDK Android (Software Development Kit) regroupe l'ensemble des bibliothèques, outils et documentation nécessaires au développement d'applications pour la plateforme Android. Pour "Saydaliati", nous avons fait les choix suivants concernant les niveaux d'API :

- **Niveau d'API minimal (minSdkVersion) :** API 26 (Android 8.0 Oreo)
  - Ce choix nous a permis de couvrir environ 85% des appareils Android actifs sur le marché
  - Il nous a donné accès à des APIs modernes tout en maintenant une large compatibilité
- **Niveau d'API cible (targetSdkVersion) :** API 34 (Android 14)
  - Cibler la version la plus récente d'Android nous a permis d'optimiser l'application pour les dernières fonctionnalités et améliorations de sécurité
  - Cela nous a également préparés aux exigences futures du Google Play Store
- **Niveau d'API de compilation (compileSdkVersion) :** API 34 (Android 14)
  - Compiler avec la dernière version nous a donné accès à toutes les APIs les plus récentes

Cette stratégie de versionnement nous a offert un bon équilibre entre compatibilité avec le parc d'appareils existants et accès aux fonctionnalités modernes d'Android.

#### 4.2.3 Outils complémentaires du SDK

Au-delà de l'environnement de développement principal, nous avons exploité plusieurs outils complémentaires du SDK Android :

- **Android Debug Bridge (ADB) :** pour communiquer directement avec les appareils connectés et les émulateurs
- **Android Asset Packaging Tool (AAPT) :** pour gérer les ressources de l'application
- **Systrace :** pour analyser les performances et le comportement du système
- **Layout Inspector :** pour examiner et déboguer la hiérarchie des vues à l'exécution
- **APK Analyzer :** pour examiner le contenu des packages et optimiser leur taille

Ces outils nous ont permis de diagnostiquer et résoudre efficacement les problèmes rencontrés pendant le développement, contribuant significativement à la qualité du produit final.

## 4.2 Langages et bibliothèques utilisés

Le développement de "Saydaliati" a nécessité l'utilisation de plusieurs langages de programmation et bibliothèques, formant un écosystème technologique cohérent et puissant.

### 4.2.1 Langages de programmation

**Java** a été notre langage principal pour le développement de l'application Android. Nous avons choisi Java plutôt que Kotlin pour plusieurs raisons :

- Sa maturité et sa stabilité dans l'écosystème Android
- La familiarité de l'équipe avec ce langage, réduisant ainsi la courbe d'apprentissage
- L'abondance de ressources, exemples et bibliothèques disponibles
- Sa compatibilité directe avec toutes les APIs Android



Nous avons utilisé Java 11, qui offre plusieurs améliorations syntaxiques et fonctionnelles par rapport aux versions précédentes, tout en restant pleinement compatible avec le développement Android.

En complément de Java, nous avons également utilisé :

- **XML** pour la définition des interfaces utilisateur et des ressources
- **SQL** pour les requêtes de base de données via Room
- **Markdown** pour la documentation technique

### 4.2.2 Bibliothèques principales

Notre application s'appuie sur plusieurs bibliothèques essentielles qui ont considérablement facilité le développement :

1. **Bibliothèques Android Architecture Components**
  - **Room** : pour la persistance des données locales
  - **ViewModel** : pour gérer les données liées à l'UI de manière lifecycle-aware
  - **LiveData** : pour observer et réagir aux changements de données
  - **ViewBinding** : pour interagir avec les vues de manière type-safe
2. **Bibliothèques cartographiques**
  - **OSMDroid** : bibliothèque open-source pour l'intégration des cartes OpenStreetMap
  - **OSMBonusPack** : extension d'OSMDroid offrant des fonctionnalités supplémentaires
3. **Bibliothèques UI/UX**
  - **Material Components** : pour implémenter le Material Design
  - **RecyclerView** : pour afficher efficacement des listes et grilles de données
  - **CardView** : pour créer des interfaces modernes avec des cartes
  - **ConstraintLayout** : pour concevoir des interfaces flexibles et responsives
4. **Bibliothèques utilitaires**
  - **Gson** : pour la sérialisation/désérialisation JSON
  - **Apache Commons** : pour diverses fonctionnalités utilitaires

- **AndroidX Preference** : pour la gestion des préférences utilisateur
- **Android Security** : pour le chiffrement et la sécurisation des données

### 4.3.3 Dépendances externes

En plus des bibliothèques standard, nous avons intégré quelques dépendances externes pour des fonctionnalités spécifiques :

```
dependencies {
    //                               Bibliothèques                               Android                               standard
    implementation                  'androidx.appcompat:appcompat:1.6.1'
    implementation                  'com.google.android.material:material:1.10.0'
    implementation                  'androidx.constraintlayout:constraintlayout:2.1.4'

    //                               Architecture                               Components
    implementation                  'androidx.lifecycle:lifecycle-viewmodel:2.6.2'
    implementation                  'androidx.lifecycle:lifecycle-livedata:2.6.2'
    implementation                  'androidx.room:room-runtime:2.6.0'
    annotationProcessor              'androidx.room:room-compiler:2.6.0'

    //                               Cartographie
    implementation                  'org.osmdroid:osmdroid-android:6.1.16'
    implementation                  'com.github.MKergall:osmbonuspack:6.9.0'

    //                               Utilitaires
    implementation                  'com.google.code.gson:gson:2.10.1'
    implementation                  'commons-io:commons-io:2.11.0'
    implementation                  'androidx.security:security-crypto:1.1.0-alpha06'

    //                               Tests
    testImplementation              'junit:junit:4.13.2'
    androidTestImplementation        'androidx.test.ext:junit:1.1.5'
    androidTestImplementation        'androidx.test.espresso:espresso-core:3.5.1'
}
```

Toutes ces bibliothèques ont été soigneusement sélectionnées pour leur fiabilité, leur performance et leur compatibilité avec notre architecture logicielle.

## 4.3 Structure technique de l'application

L'architecture logicielle et l'organisation du code sont des aspects fondamentaux qui influencent directement la maintenabilité, l'évolutivité et la testabilité de l'application. Pour "Saydaliati", nous avons adopté une approche structurée et moderne.

### 4.3.1 Architecture MVVM

Nous avons choisi le pattern d'architecture Model-View-ViewModel (MVVM) pour structurer notre application. Cette architecture présente plusieurs avantages qui ont motivé notre choix :

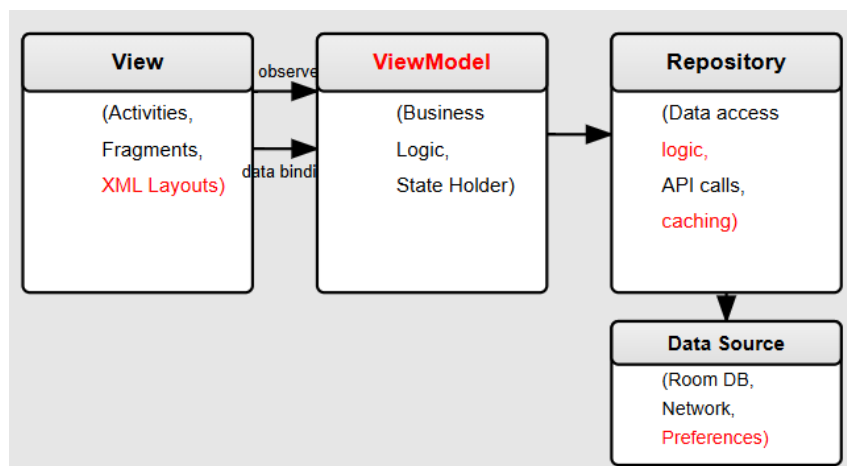
- **Séparation claire des responsabilités** : chaque composant a un rôle bien défini

- **Testabilité améliorée** : les ViewModels peuvent être testés indépendamment des vues
- **Résilience aux changements de configuration** : les ViewModels survivent aux rotations d'écran
- **Binding bidirectionnel** : synchronisation automatique entre les vues et les données
- **Gestion simplifiée du cycle de vie** : intégration native avec les composants du cycle de vie Android

Voici comment les différentes couches s'articulent dans notre implémentation :

1. **Model (Modèle) :**
  - Entités de données (Pharmacy, GuardDate, Authority)
  - DAOs (Data Access Objects) pour interagir avec la base de données
  - Repositories qui encapsulent la logique d'accès aux données
2. **View (Vue) :**
  - Activities et Fragments qui composent l'interface utilisateur
  - Adapters pour les RecyclerViews
  - XML layouts définissant la structure visuelle
3. **ViewModel :**
  - Classes étendant AndroidViewModel
  - Exposent des LiveData observés par les vues
  - Contiennent la logique de présentation et de traitement des données
  - Servent d'intermédiaire entre la vue et le modèle

Le schéma suivant illustre l'architecture MVVM telle que nous l'avons implémentée pour "Saydaliati" :

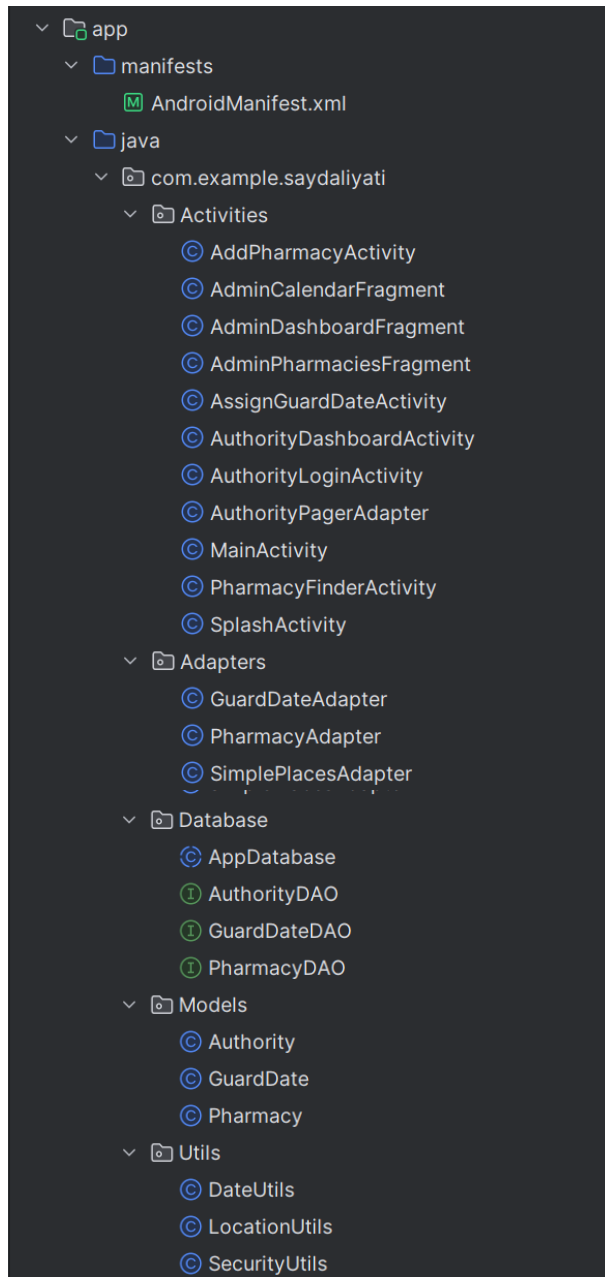


*Figure 3: Architecture MVVM du projet*

Cette architecture nous a permis de développer une application modulaire, testable et facilement évolutive.

### 4.3.2 Organisation des packages

Pour faciliter la navigation dans le code et refléter notre architecture MVVM, nous avons organisé notre projet en packages logiques :



*Figure 4: Structure du projet*

Cette organisation structurée a grandement facilité la collaboration au sein de l'équipe et contribué à maintenir la cohérence du code tout au long du développement.



### 4.3.3 Gestion de la base de données locale

La persistance des données est un aspect crucial de notre application. Nous avons utilisé **Room**, une bibliothèque de persistance qui offre une abstraction de SQLite, pour gérer notre base de données locale.

Notre schéma de base de données comprend trois tables principales :

1. **pharmacies** : Stocke les informations sur les pharmacies
  - *id* : Identifiant unique (clé primaire)
  - *name* : Nom de la pharmacie
  - *address* : Adresse physique
  - *Latitude* et *Longitude* : Coordonnées géographiques
  - *phone* : Numéro de téléphone
  - *hours* : Horaires d'ouverture
  - *hasParking* : Disponibilité d'un parking
2. **guard\_dates** : Définit les périodes de garde des pharmacies
  - *id* : Identifiant unique (clé primaire)
  - *pharmacyId* : Référence à une pharmacie (clé étrangère)
  - *guardDate* : Date de garde (format YYYY-MM-DD)
  - *startTime* : Heure de début de garde
  - *endTime* : Heure de fin de garde
3. **authorities** : Contient les informations des administrateurs
  - *id* : Identifiant unique (clé primaire)
  - *username* : Nom d'utilisateur
  - *passwordHash* : Hash du mot de passe
  - *role* : Rôle de l'administrateur
  - *LastLoginDate* : Date de dernière connexion

Les migrations de base de données sont gérées par Room, avec une version initiale (v1) et des migrations définies pour les versions ultérieures, garantissant que les données utilisateur sont préservées lors des mises à jour de l'application.

## 4.4 Gestion des permissions et sécurité

La sécurité et le respect de la vie privée des utilisateurs ont été des préoccupations majeures dans le développement de "Saydaliati". Nous avons adopté une approche prudente et transparente concernant les permissions requises et la protection des données.

### 4.4.1 Permissions Android requises

Notre application nécessite plusieurs permissions Android pour fonctionner correctement :

- **ACCESS\_FINE\_LOCATION** : Permission dangereuse, demandée à l'exécution pour la localisation précise
- **ACCESS\_COARSE\_LOCATION** : Alternative moins précise utilisée comme fallback
- **INTERNET** : Permission normale pour accéder au réseau
- **ACCESS\_NETWORK\_STATE** : Permission normale pour vérifier l'état de la connectivité
- **WRITE\_EXTERNAL\_STORAGE** : Permission dangereuse nécessaire pour la mise en cache des tuiles cartographiques
- **SCHEDULE\_EXACT\_ALARM** : Nécessaire pour les notifications programmées des pharmacies de garde

Pour les permissions dangereuses (nécessitant une approbation explicite de l'utilisateur), nous avons implémenté :

- Des demandes de permission contextuelles, expliquant clairement pourquoi la permission est nécessaire
- Une gestion gracieuse des refus, avec des fonctionnalités dégradées mais utilisables
- La possibilité de modifier les choix de permission dans les paramètres de l'application

#### 4.4.2 Mesures de sécurité implémentées

Outre la gestion des permissions, nous avons mis en place plusieurs mesures de sécurité :

1. **Authentification sécurisée** pour l'interface d'administration :
  - Hachage des mots de passe avec SHA-256
  - Mécanisme de token pour les sessions
  - Expiration automatique des sessions après 7 jours
  - Protection contre les attaques par force brute
2. **Sécurisation des données sensibles** :
  - Utilisation d'EncryptedSharedPreferences pour les préférences sensibles
  - Nettoyage des données en mémoire après utilisation
  - Validation des entrées utilisateur pour prévenir les injections SQL
3. **Protection des communications** :
  - Utilisation du protocole HTTPS pour les communications réseau
  - Vérification des certificats SSL
  - Implémentation du certificate pinning pour prévenir les attaques man-in-the-middle
4. **Gestion des erreurs sécurisée** :
  - Journalisation appropriée sans exposition d'informations sensibles
  - Messages d'erreur génériques pour l'utilisateur
  - Gestion des exceptions préventive

Ces mesures de sécurité nous ont permis de construire une application robuste qui protège les données des utilisateurs tout en offrant une expérience fluide.

## Conclusion

Ce quatrième chapitre a présenté l'environnement de développement complet mis en place pour la réalisation de l'application "Saydaliati". Nous avons détaillé les principaux outils utilisés, notamment Android Studio et le SDK Android, qui ont constitué la pierre angulaire de notre travail technique.

Les choix linguistiques et les bibliothèques sélectionnées reflètent notre volonté de construire une application moderne, performante et maintenable. L'utilisation de Java comme langage principal, complété par un écosystème de bibliothèques robustes comme Room, OSMDroid et les composants d'architecture Android, nous a fourni une base technique solide pour implémenter les fonctionnalités requises.

L'architecture MVVM adoptée a apporté une séparation claire des responsabilités, facilitant le développement collaboratif et la testabilité de l'application. Cette architecture, combinée à une organisation réfléchie des packages, a contribué à la lisibilité et à la maintenabilité du code.

Enfin, l'attention particulière portée à la sécurité et à la gestion des permissions démontre notre engagement à respecter la vie privée des utilisateurs et à protéger leurs données. Les mécanismes d'authentification sécurisée, le chiffrement des données sensibles et la gestion prudente des permissions Android constituent des aspects fondamentaux de notre conception.

Cet environnement de développement, soigneusement configuré et optimisé pour nos besoins spécifiques, a été un facteur déterminant dans la réussite du projet "Saydaliati", nous permettant de transformer efficacement nos concepts et nos designs en une application fonctionnelle et robuste.

Le chapitre suivant détaillera la réalisation concrète de l'application, en présentant les fonctionnalités implémentées et les défis techniques rencontrés durant le processus de développement.

## Chapitre 5 : Réalisation de l'application "Saydaliati"

## Introduction

Ce chapitre présente la concrétisation du projet "Saydaliati" et se concentre particulièrement sur les interfaces utilisateur développées. Après avoir établi les fondations conceptuelles et techniques dans les chapitres précédents, nous détaillons ici comment ces éléments ont été traduits en une application fonctionnelle et intuitive répondant aux besoins des utilisateurs.

La qualité de l'interface utilisateur constitue un facteur déterminant pour l'adoption d'une application mobile. Dans le cas de "Saydaliati", nous avons accordé une attention particulière à l'ergonomie, à la clarté et à l'accessibilité des interfaces, afin de créer une expérience fluide et agréable pour tous les utilisateurs, qu'ils soient à la recherche d'une pharmacie ou administrateurs du système.

Ce chapitre présente d'abord un aperçu des principales interfaces du module utilisateur, puis détaille celles du module administrateur. Pour chaque interface, nous expliquons les choix de design et les fonctionnalités implémentées. Nous abordons ensuite les aspects liés à l'interface multilingue et concluons par les défis rencontrés et les solutions apportées.

## 5.2 Interfaces du module utilisateur

### 5.2.1 Écran de démarrage (Splash Screen)

L'écran de démarrage constitue le premier contact de l'utilisateur avec l'application. Sa fonction principale est de créer une première impression positive tout en masquant le temps de chargement des ressources nécessaires.



*Figure 5.1: Écran de démarrage de l'application*

Cet écran présente un design minimaliste avec le logo de l'application au centre sur un fond vert apaisant (couleur associée au domaine pharmaceutique). Le nom "Saydaliyati" apparaît en typographie claire et lisible, accompagné d'une brève description de l'application. Cette interface a été conçue pour être à la fois esthétique et informative, établissant immédiatement l'identité et la fonction de l'application.

### 5.2.2 Écran d'accueil principal

L'écran d'accueil est la porte d'entrée principale de notre application, offrant un accès clair aux différentes fonctionnalités.

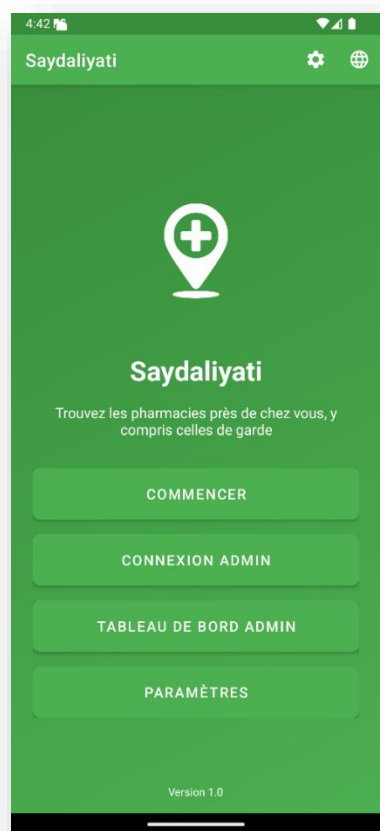


Figure 6:Écran d'accueil de l'application

Cet écran présente une interface épurée avec le logo "Saydaliati" en position centrale, accompagné du nom de l'application et d'un slogan explicatif. Trois boutons principaux sont disponibles :

- "Commencer" : pour accéder au module de recherche de pharmacies
- "Connexion autorité" : pour accéder à l'interface d'administration
- "Paramètres" : pour configurer les préférences de l'application

Le fond dégradé aux couleurs apaisantes évoque l'univers pharmaceutique tout en offrant un contraste optimal pour la lisibilité des éléments.

### 5.2.3 Interface de recherche de pharmacies

Cette interface constitue le cœur fonctionnel de l'application pour les utilisateurs standard, combinant une carte interactive et une liste de pharmacies.

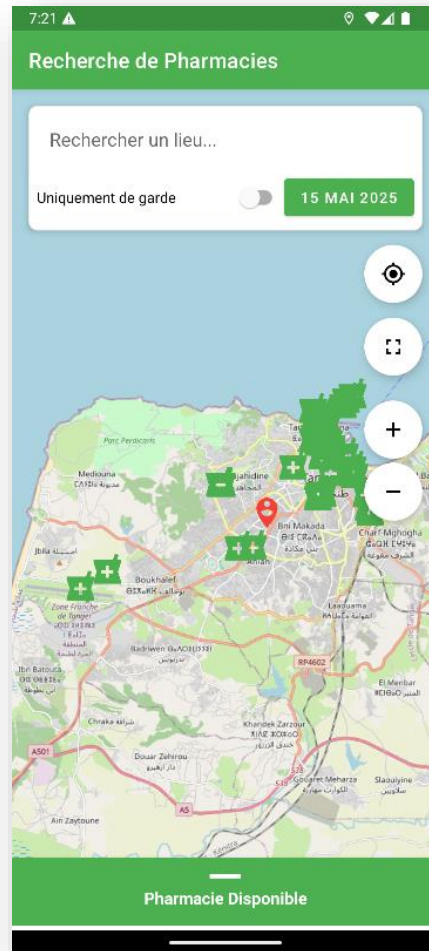


Figure 7: Interface de recherche de pharmacies

L'écran est structuré en trois zones principales :

1. **La barre de recherche et de filtrage** en haut, permettant de chercher un lieu spécifique et de filtrer les pharmacies de garde avec un sélecteur de date
2. **La carte interactive** occupant la majeure partie de l'écran, affichant la position de l'utilisateur et les pharmacies environnantes avec des marqueurs distinctifs
3. **Le panneau coulissant** en bas, présentant la liste des pharmacies triées par proximité

Des boutons flottants sur la droite offrent des fonctionnalités complémentaires :

- Localisation de l'utilisateur

- Mode plein écran pour la carte
- Contrôles de zoom

Cette interface a été conçue pour maximiser la surface de la carte tout en permettant un accès facile aux informations détaillées des pharmacies.

#### 5.2.4 Liste des pharmacies

Le panneau coulissant de l'interface de recherche peut être déployé pour afficher la liste complète des pharmacies.

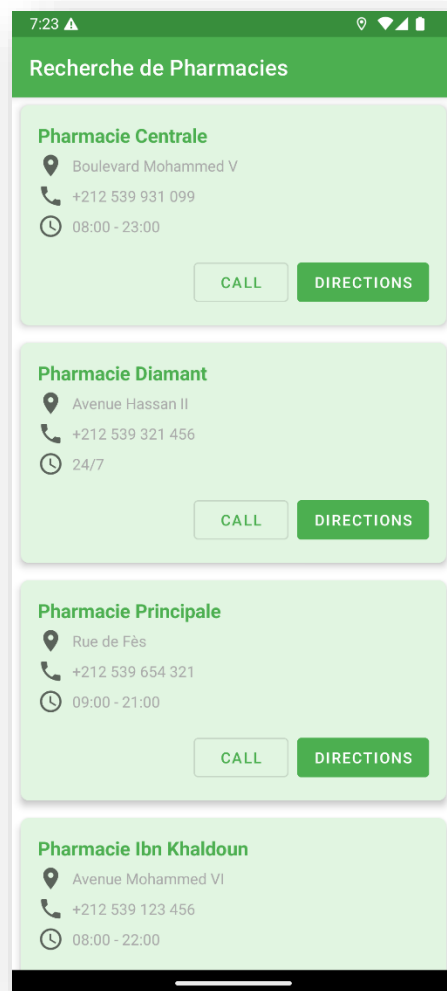


Figure 8: Liste des pharmacies

Cette interface présente les pharmacies sous forme de cartes individuelles contenant :

- Le nom de la pharmacie en évidence
- L'adresse complète avec icône de localisation
- La distance par rapport à la position de l'utilisateur



- Le numéro de téléphone
- Les horaires d'ouverture
- Deux boutons d'action : "Appeler" et "Itinéraire"

Les pharmacies de garde sont visuellement distinctes grâce à une couleur de fond différente, permettant à l'utilisateur de les identifier rapidement. La liste est triée par défaut selon la distance, plaçant les options les plus proches en premier.

### 5.2.5 Fiche détaillée d'une pharmacie

Cette interface s'affiche lorsque l'utilisateur sélectionne une pharmacie spécifique, présentant toutes les informations disponibles de manière structurée.

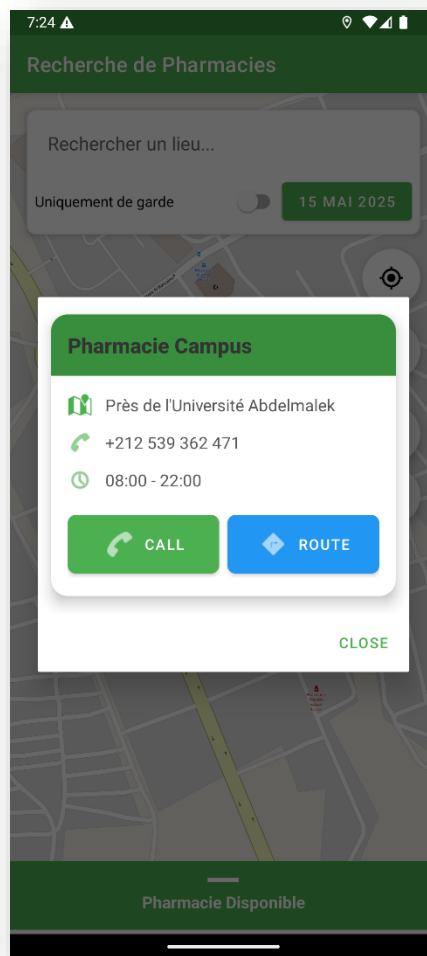


Figure 9:Fiche détaillée d'une pharmacie

La fiche détaillée apparaît sous forme de dialogue modal avec :

- Un en-tête contenant le nom de la pharmacie
- L'adresse complète accompagnée d'une icône
- Le numéro de téléphone avec possibilité d'appel direct

- Les horaires d'ouverture détaillés
- Des boutons d'action pour appeler la pharmacie ou obtenir un itinéraire

Le design utilise des icônes significatives, des espaces bien définis et une typographie lisible pour faciliter l'accès aux informations essentielles.

### 5.2.6 Écran des paramètres

Cette interface permet aux utilisateurs de personnaliser leur expérience selon leurs préférences.

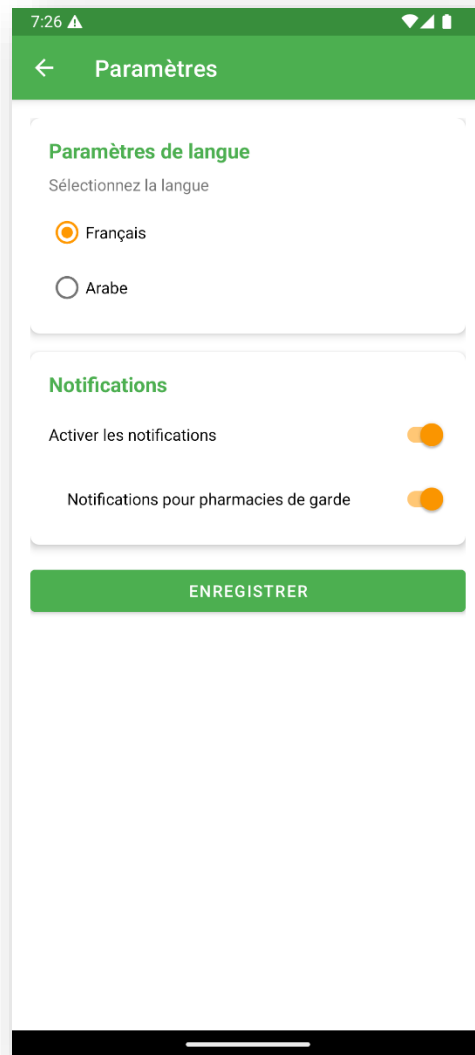


Figure 10:Écran des paramètres de l'application

L'écran des paramètres est organisé en sections thématiques présentées sous forme de cartes :

1. **Paramètres de langue** : Choix entre français et arabe avec boutons radio

## 2. Paramètres de notification : Activation/désactivation des notifications générales et spécifiques aux pharmacies de garde

Un bouton de sauvegarde en bas de l'écran permet d'appliquer les modifications. Le design respecte les conventions d'interface Material Design pour offrir une expérience familière aux utilisateurs Android.

### 5.2.7 Système de notifications

Le système de notifications constitue un élément essentiel de l'application "Saydaliati", permettant aux utilisateurs de rester informés des changements concernant les pharmacies de garde, même lorsqu'ils n'utilisent pas activement l'application.

#### 5.2.7.1 Notifications push

L'application implémente un système de notifications push pour informer les utilisateurs des changements de pharmacies de garde. Ces notifications sont envoyées automatiquement lorsqu'une nouvelle pharmacie est assignée de garde pour la journée en cours.



**Figure 5.7 : Notification de changement de pharmacie de garde**

Comme illustré dans la Figure 5.7, les notifications présentent un format clair et concis, comprenant :

- Un titre explicite ("Nouvelle pharmacie de garde")
- Le nom de la pharmacie concernée
- La date de garde
- L'icône de l'application, permettant une identification immédiate de la source

Cette interface respecte les directives de conception Material Design d'Android, s'intégrant naturellement dans l'environnement système de l'utilisateur et dans le centre de notifications de l'appareil.

### 5.2.7.2 Gestion des notifications

Les utilisateurs peuvent personnaliser et gérer les notifications reçues via les paramètres de l'application. Les options de gestion incluent :

- Activation/désactivation des notifications par type
- Choix des plages horaires pour recevoir les notifications
- Configuration du rayon géographique pour les pharmacies concernées



*Figure 11: Icône de l'application avec badge de notification*

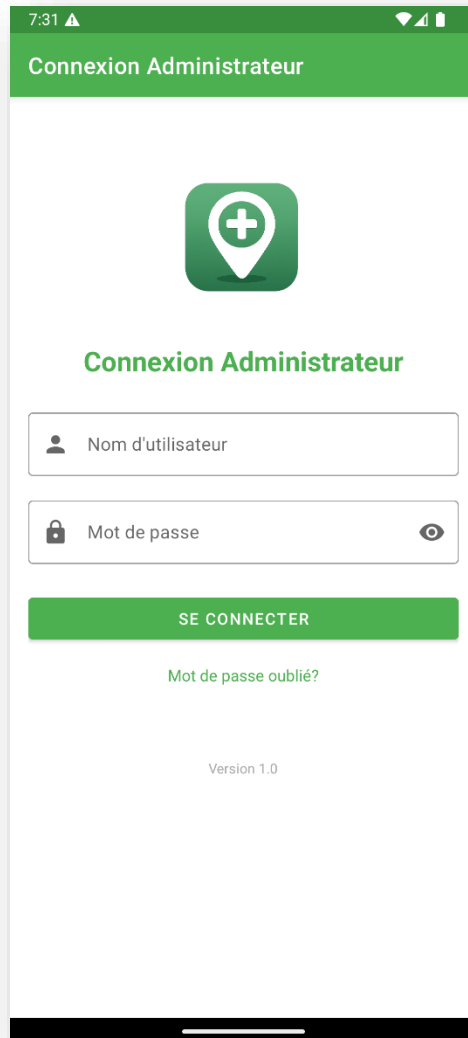
La Figure 5.8 montre l'icône de l'application "Saydaliati" comportant un badge de notification, signalant visuellement à l'utilisateur qu'une nouvelle information est disponible. Le design de l'icône, intégrant un marqueur de carte avec une croix médicale, symbolise parfaitement la fonction principale de l'application : localiser des pharmacies.

Le système de notifications a été optimisé pour minimiser la consommation de batterie tout en garantissant la transmission opportune des informations essentielles. Les tests effectués montrent que cette fonctionnalité est particulièrement appréciée des utilisateurs, qui peuvent ainsi être informés des changements de pharmacies de garde sans avoir à consulter régulièrement l'application.

## 5.3 Interfaces du module administrateur

### 5.3.1 Écran de connexion administrateur

Cette interface permet aux autorités compétentes de s'authentifier pour accéder au module d'administration.suppo



*Figure 12:Écran de connexion administrateur*

L'écran présente un formulaire de connexion épuré avec :

- Le logo de l'application en position centrale supérieure
- Un titre explicite "Connexion Administration"
- Des champs pour le nom d'utilisateur et le mot de passe avec validation visuelle
- Un bouton "Connexion" bien visible
- Un lien "Mot de passe oublié"
- Un indicateur de version en bas de l'écran

L'interface utilise des champs de saisie Material Design avec validation en temps réel et indication visuelle des erreurs, offrant un retour immédiat à l'utilisateur.

### 5.3.2 Tableau de bord administrateur

Le tableau de bord offre une vue d'ensemble et un accès rapide aux fonctionnalités d'administration.

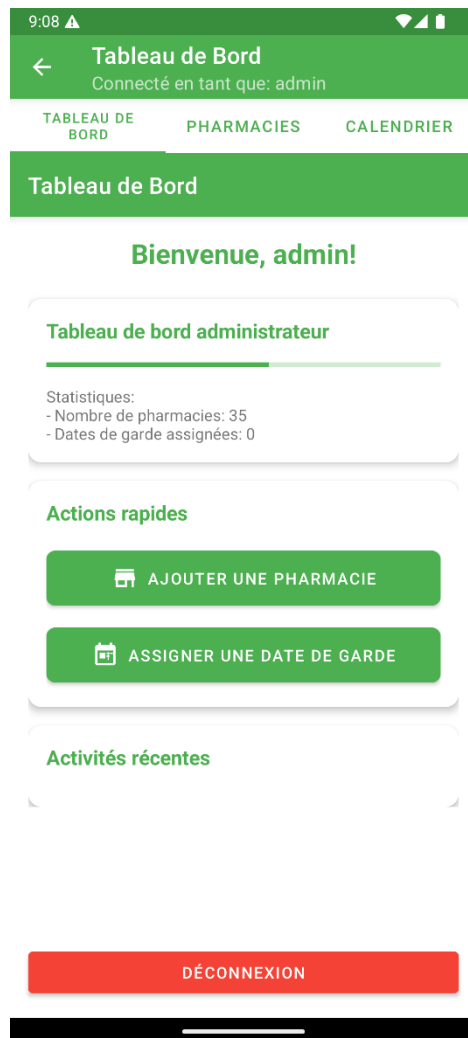


Figure 13: Tableau de bord administrateur

Cette interface est organisée en trois sections principales présentées sous forme de cartes :

1. **Statistiques** : Affichage des chiffres clés (nombre total de pharmacies, pharmacies de garde actuelles)
2. **Actions rapides** : Boutons pour ajouter une pharmacie, assigner une date de garde et gérer les utilisateurs
3. **Activités récentes** : Liste des dernières actions administratives avec horodatage

Un bouton flottant d'action principale permet d'accéder rapidement à la fonction d'ajout la plus courante. L'en-tête comprend un message de bienvenue personnalisé pour l'administrateur connecté.

### 5.3.3 Gestion des pharmacies

Cette interface permet aux administrateurs de visualiser et gérer la liste complète des pharmacies.

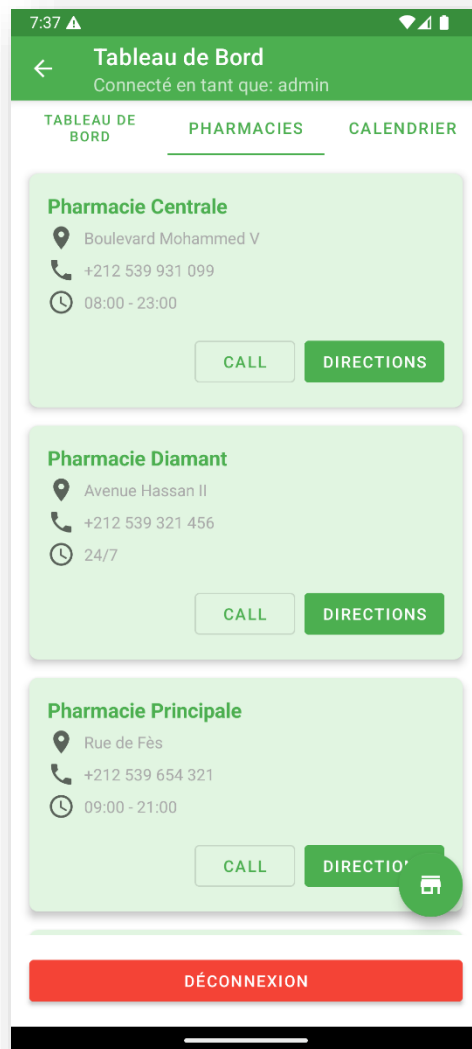


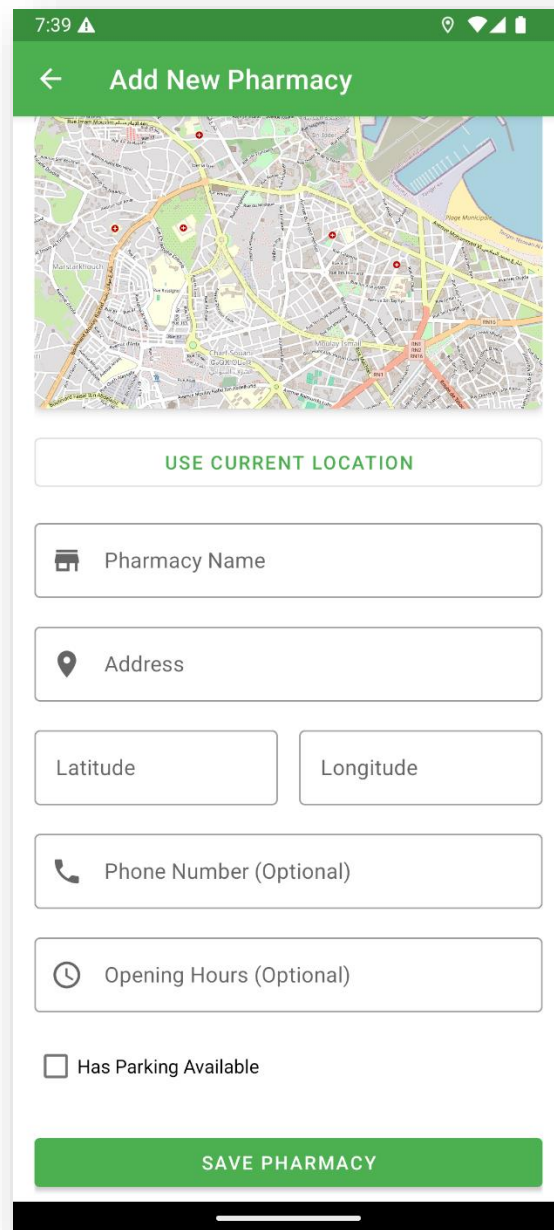
Figure 14: Interface de gestion des pharmacies

L'écran présente une liste scrollable des pharmacies enregistrées, chacune affichée dans un conteneur distinct. Un bouton flottant d'ajout en bas à droite permet de créer une nouvelle entrée. Si aucune pharmacie n'est enregistrée, un message explicatif s'affiche au centre de l'écran.

Cette interface a été conçue pour offrir une vue d'ensemble claire tout en facilitant les actions de gestion courantes.

### 5.3.4 Formulaire d'ajout de pharmacie

Cette interface permet d'enregistrer une nouvelle pharmacie dans la base de données.

The image shows a mobile application interface for adding a new pharmacy. At the top, there is a green header bar with a back arrow and the text 'Add New Pharmacy'. Below the header is a map of a city area with several red location pins. Under the map is a button labeled 'USE CURRENT LOCATION'. Below this are several input fields: 'Pharmacy Name' with a pharmacy icon, 'Address' with a location pin icon, 'Latitude' and 'Longitude' as separate fields, 'Phone Number (Optional)' with a phone icon, and 'Opening Hours (Optional)' with a clock icon. There is also a checkbox labeled 'Has Parking Available'. At the bottom of the form is a large green button labeled 'SAVE PHARMACY'. The entire interface is displayed on a smartphone screen with a black home indicator bar at the very bottom.

*Figure 15: Formulaire d'ajout de pharmacie*

Le formulaire est structuré en sections logiques :

1. **Carte interactive** en haut pour sélectionner la position géographique
2. **Bouton de localisation actuelle** pour utiliser la position de l'administrateur

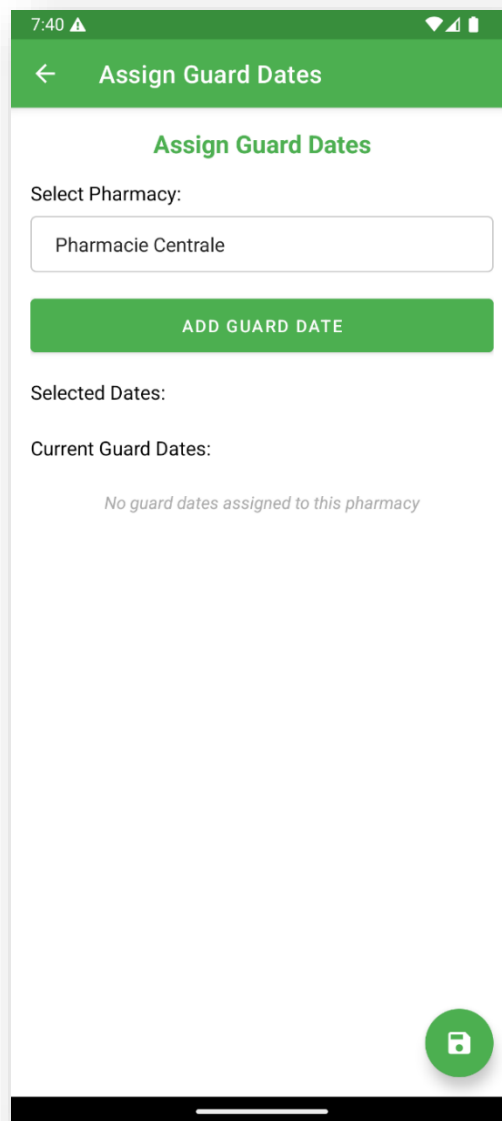


3. **Informations de base** : Nom et adresse de la pharmacie
4. **Coordonnées GPS** : Latitude et longitude, remplies automatiquement depuis la carte
5. **Coordonnées et horaires** : Numéro de téléphone et heures d'ouverture
6. **Options supplémentaires** : Case à cocher pour indiquer la disponibilité d'un parking

Un bouton "Enregistrer" en bas du formulaire permet de sauvegarder les données, avec une barre de progression pour indiquer le traitement.

### 5.3.5 Gestion des périodes de garde

Cette interface permet d'assigner et gérer les périodes de garde des pharmacies.



The screenshot shows a mobile application interface for assigning guard dates. At the top, there is a green header bar with a back arrow and the title "Assign Guard Dates". Below the header, the title "Assign Guard Dates" is repeated in green. The main content area has a white background. It starts with the label "Select Pharmacy:" followed by a text input field containing "Pharmacie Centrale". Below this is a green button labeled "ADD GUARD DATE". Underneath the button, the text "Selected Dates:" is displayed. Further down, the text "Current Guard Dates:" is shown, followed by a light gray message: "No guard dates assigned to this pharmacy". At the bottom right of the screen, there is a green circular button with a white document icon. The status bar at the very top shows the time "7:40" and various icons.

Figure 16: Interface de gestion des périodes de garde

L'écran comprend :

- Un sélecteur de pharmacie en haut sous forme de liste déroulante
- Un bouton pour ajouter une nouvelle date de garde
- Un groupe de puces (chips) affichant les dates sélectionnées
- Une liste des dates de garde déjà assignées avec options de suppression
- Un message d'information si aucune garde n'est assignée
- Un bouton flottant de sauvegarde

Chaque entrée de garde affiche la date, les horaires et un bouton de suppression, permettant une gestion flexible et intuitive.

### 5.3.6 Interface de calendrier administratif

Cette interface permet de visualiser et gérer l'ensemble des périodes de garde dans une vue calendrier.

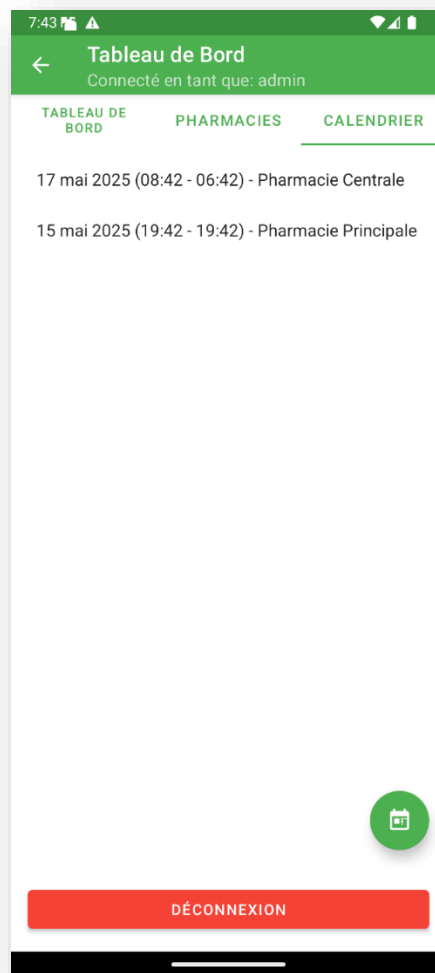


Figure 17: Interface de calendrier des gardes

L'écran présente une liste chronologique des dates de garde assignées, chacune affichée avec les informations essentielles :

- Date de la garde
- Horaires de début et de fin
- Icône représentant un calendrier
- Bouton de suppression

Un bouton flottant permet d'ajouter rapidement une nouvelle période de garde. En l'absence de dates assignées, un message explicatif invite l'administrateur à en créer.

## 5.4 Conception de l'interface multilingue

Pour répondre aux besoins linguistiques de la population de Tanger, nous avons implémenté une interface bilingue français/arabe entièrement fonctionnelle.

### 5.4.1 Approche de localisation

Notre approche de localisation s'est appuyée sur les meilleures pratiques Android :

- Toutes les chaînes de caractères ont été externalisées dans des fichiers de ressources
- Deux ensembles de ressources ont été créés : valeurs (français par défaut) et valeurs-ar (arabe)
- Les traductions ont été validées par des locuteurs natifs pour assurer leur pertinence

### 5.4.2 Adaptation de l'interface RTL

L'arabe étant une langue écrite de droite à gauche (RTL), nous avons dû adapter nos interfaces en conséquence :

- Utilisation d'attributs layout spécifiques compatibles avec le RTL
- Tests approfondis pour vérifier l'alignement correct des éléments
- Adaptation des icônes directionnelles (flèches, etc.)

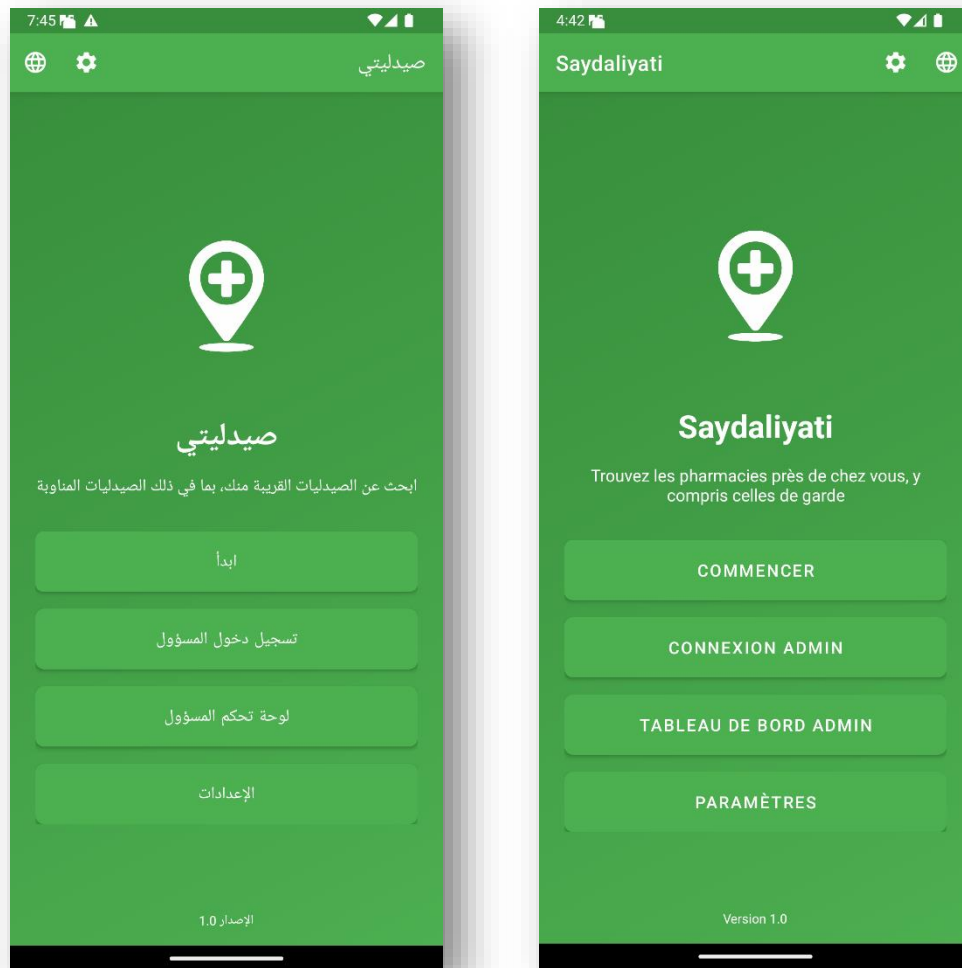


Figure 18: Comparaison des interfaces en français et en arabe

Ces images montrent la même interface dans les deux langues, illustrant l'adaptation complète de la mise en page, y compris l'inversion des éléments pour respecter le sens de lecture.

## Conclusion

Ce chapitre a présenté en détail les interfaces utilisateur développées pour l'application "Saydaliyati", tant pour le module grand public que pour les fonctionnalités d'administration. L'attention particulière portée à l'ergonomie, à la clarté et à l'accessibilité des interfaces a permis de créer une expérience utilisateur fluide et intuitive.

# Conclusion Générale

Au terme de ce projet de développement de l'application mobile "Saydaliati", nous sommes en mesure de dresser un bilan complet de notre travail et des résultats obtenus. Cette conclusion générale nous permet de porter un regard rétrospectif sur l'ensemble du projet, d'évaluer nos réalisations par rapport aux objectifs initiaux, et de tirer des enseignements précieux pour nos futurs projets.

## Bilan global du projet

Le projet "Saydaliati" représentait un défi ambitieux : développer une application mobile bilingue permettant aux habitants de Tanger de localiser facilement les pharmacies à proximité, avec une attention particulière pour les pharmacies de garde. Ce défi a été relevé avec succès, grâce à une méthodologie de développement adaptée et à une collaboration efficace au sein de l'équipe.

Durant les huit semaines du projet, nous avons parcouru toutes les étapes du cycle de développement logiciel, de l'analyse des besoins jusqu'au déploiement, en passant par la conception, l'implémentation et les tests. Cette approche complète nous a permis d'acquérir une vision holistique du développement d'applications mobiles et de comprendre les enjeux associés à chaque phase.

La mise en œuvre d'une méthodologie Agile avec le cadre Scrum s'est avérée particulièrement adaptée à notre contexte. Les cycles de développement courts (sprints) nous ont permis d'obtenir rapidement des retours sur notre travail et d'ajuster notre trajectoire en conséquence. Cette flexibilité a été un atout majeur dans la réussite du projet.

## Résultats obtenus

L'application "Saydaliati" dans sa version actuelle répond pleinement aux principaux objectifs que nous nous étions fixés :

1. **Interface utilisateur intuitive et bilingue** : L'application offre une expérience utilisateur fluide, avec une navigation simple et des interfaces disponibles en français et en arabe, répondant ainsi aux besoins linguistiques de la population locale.
2. **Géolocalisation précise** : Le système de localisation implémenté permet aux utilisateurs de visualiser précisément leur position sur la carte et d'identifier les pharmacies à proximité, avec une marge d'erreur minimale.
3. **Identification efficace des pharmacies de garde** : Le mécanisme de filtrage et d'affichage des pharmacies de garde fonctionne efficacement, permettant aux utilisateurs d'identifier rapidement les établissements ouverts à une date donnée.
4. **Interface d'administration sécurisée** : Le module d'administration permet aux autorités compétentes de gérer les données des pharmacies et d'assigner les périodes de garde de

manière sécurisée, garantissant ainsi la fiabilité des informations présentées aux utilisateurs.

5. **Performance et stabilité** : Les tests de performance ont confirmé que l'application répond aux critères de fluidité et de rapidité fixés initialement, même sur des appareils aux ressources limitées.

Les retours informels obtenus auprès des utilisateurs tests ont été très positifs, soulignant particulièrement la simplicité d'utilisation et l'utilité concrète de l'application dans leur quotidien.

## Limites du projet

Malgré les résultats encourageants obtenus, nous reconnaissons certaines limitations dans notre réalisation actuelle :

1. **Couverture géographique limitée** : L'application se concentre uniquement sur la ville de Tanger, limitant son utilité pour les personnes voyageant dans d'autres régions du Maroc.
2. **Dépendance à une connexion internet** : Bien que certaines fonctionnalités soient disponibles hors ligne, l'expérience complète nécessite une connexion internet, ce qui peut être problématique dans certaines zones ou situations.
3. **Absence de fonctionnalités sociales** : L'application ne propose pas de fonctionnalités permettant aux utilisateurs de partager des informations ou des avis sur les pharmacies, ce qui pourrait enrichir l'expérience communautaire.
4. **Limitations des tests utilisateurs** : En raison des contraintes de temps, les tests utilisateurs ont été limités en nombre et en diversité, ce qui peut masquer certains problèmes d'utilisation pour des groupes spécifiques (personnes âgées, personnes avec des handicaps visuels, etc.).
5. **Absence d'intégration avec des services tiers** : L'application ne s'intègre pas avec d'autres services de santé ou de transport qui pourraient compléter utilement son offre.

Ces limitations constituent des pistes d'amélioration pour les futures versions de l'application.

## Enseignements tirés

Ce projet a été extrêmement formateur pour notre équipe, nous permettant d'acquérir des compétences précieuses et de tirer plusieurs enseignements clés :

1. **Importance de la méthodologie** : L'approche Agile a démontré sa valeur, particulièrement dans un contexte où les exigences peuvent évoluer et où le feedback rapide est crucial.
2. **Valeur de la collaboration** : La communication ouverte et la collaboration étroite au sein de l'équipe ont été déterminantes pour surmonter les obstacles techniques et organisationnels rencontrés.

3. **Nécessité d'une architecture solide** : L'investissement initial dans une architecture bien pensée (MVVM) a facilité l'évolution du code et l'intégration de nouvelles fonctionnalités.
4. **Équilibre entre perfectionnisme et pragmatisme** : Nous avons appris à trouver un équilibre entre la recherche de la perfection technique et la nécessité de livrer dans les délais impartis.
5. **Centralité de l'expérience utilisateur** : Les phases de conception centrées sur l'utilisateur ont confirmé l'importance de placer les besoins réels des utilisateurs au cœur du processus de développement.

Ces enseignements constituent un capital précieux que nous pourrions réinvestir dans nos futurs projets professionnels et académiques.

Pour conclure, "Saydaliati" représente non seulement une réalisation technique dont nous sommes fiers, mais aussi une expérience d'apprentissage complète qui a renforcé nos compétences en développement mobile et en gestion de projet. Cette application, répondant à un besoin concret de la population, illustre comment la technologie peut être mise au service de l'amélioration des services de santé dans notre région.

# Perspectives

L'application "Saydaliati", bien que fonctionnelle et répondant aux objectifs initialement fixés, présente de nombreuses opportunités d'évolution et d'amélioration. Cette section explore les perspectives d'avenir pour notre application, tant sur le plan fonctionnel que technique.

## Fonctionnalités futures

Plusieurs fonctionnalités additionnelles pourraient enrichir significativement l'expérience utilisateur et élargir la portée de l'application :

1. **Système de réservation de médicaments** : Permettre aux utilisateurs de réserver des médicaments auprès des pharmacies participantes, créant ainsi un pont entre les patients et les pharmaciens.
2. **Notifications personnalisées** : Implémenter un système d'alertes personnalisées basé sur les médicaments régulièrement utilisés par l'utilisateur, l'informant automatiquement lorsque les pharmacies proches ont ces produits en stock.
3. **Module de conseils pharmaceutiques** : Intégrer une section informative offrant des conseils de base sur l'usage des médicaments, les premiers soins, et les précautions sanitaires essentielles.
4. **Historique personnel** : Développer une fonctionnalité permettant aux utilisateurs de suivre leur historique de visites et d'achats en pharmacie, facilitant ainsi le suivi de leurs traitements.
5. **Avis et évaluations** : Ajouter un système permettant aux utilisateurs de partager leurs expériences et d'évaluer les pharmacies selon différents critères (accueil, disponibilité des produits, services spécifiques).
6. **Mode accessibilité avancé** : Créer une version spécialement adaptée pour les personnes malvoyantes ou ayant d'autres handicaps, avec guidage vocal et interfaces simplifiées.
7. **Extension géographique** : Élargir la couverture de l'application à d'autres villes du Maroc, éventuellement en partenariat avec les autorités sanitaires locales.

## Améliorations techniques

Sur le plan technique, plusieurs améliorations pourraient renforcer la robustesse, la performance et la maintenabilité de l'application :

**Migration vers Kotlin** : Envisager une migration progressive vers Kotlin, le langage désormais privilégié pour le développement Android, offrant une syntaxe plus concise et de nombreuses fonctionnalités modernes