République Tunisienne
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Monastir
Institut Supérieur d'Informatique de Mahdia

PROJET FEDERE – METHODE AGILE

Présenté à
Institut Supérieur d'Informatique de Mahdia

Par
Zeineb Ben Jeddou

DÉVELOPPEMENT D'UNE APPLICATION D'ASSISTANCE AUX PERSONNES ÂGÉES

Encadré par :
Mme OUNI

Année universitaire 2024/2025

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Dans un contexte où le vieillissement de la population est de plus en plus présent, il devient essentiel de développer des solutions technologiques qui répondent aux besoins des personnes âgées, en particulier en matière de gestion de leur santé et de leur quotidien.

L'objectif principal de ce projet est de concevoir une application mobile destinée à assister les personnes âgées dans la gestion de leur suivi médical, de leurs rappels de médicaments, et de leurs rendez-vous médicaux, tout en leur permettant d'interagir facilement avec leurs proches et soignants.

L'application mobile que nous proposons s'articule autour de plusieurs fonctionnalités clés : la gestion des rappels pour médicaments et rendez-vous, l'interaction avec les proches et soignants via un partage de calendrier et des notifications, ainsi qu'un accès facilité à un chatbot intelligent conçu pour aider les utilisateurs à programmer leurs rappels et répondre à leurs questions fréquentes. De plus, une fonctionnalité de bouton SOS est intégrée pour garantir une réponse rapide en cas d'urgence.

La conception de l'interface se fera avec une attention particulière à l'accessibilité, en simplifiant la navigation, en optimisant les éléments visuels (comme les polices et les couleurs), et en proposant une assistance vocale pour une meilleure interaction. La base de données sera gérée à l'aide de MySQL et l'application sera développée en Flutter, garantissant ainsi une expérience fluide et réactive sur différentes plateformes. Le backend sera assuré par Spring Boot, permettant une gestion efficace et sécurisée des données.

Ce projet vise à fournir aux utilisateurs âgés un outil simple et intuitif qui améliore leur qualité de vie, leur permet de rester autonomes et de mieux gérer leur santé, tout en assurant la tranquillité d'esprit de leurs proches et soignants.

Nous commençons ce rapport par une introduction au contexte du projet, suivie d'une étude de l'existant afin d'identifier les besoins auxquels notre solution doit répondre. Ensuite, nous proposons une méthodologie adaptée pour la réalisation du projet, en nous appuyant sur une approche structurée. Le rapport est organisé en plusieurs chapitres : Un chapitre sera consacré à la spécification des besoins, à la définition du pattern et du style architectural, ainsi qu'au pilotage du projet selon la méthodologie Agile Scrum. Nous y présenterons également le backlog du produit et le découpage des fonctionnalités en trois sprints.

Les chapitres suivants traiteront du développement de l'application. Le premier portera sur la gestion des rappels et des interactions entre utilisateurs. Le second se focalisera sur l'implémentation du chatbot et du journal de suivi, tandis que le troisième concernera l'intégration des fonctionnalités d'urgence, notamment le bouton SOS et les notifications.

Un dernier chapitre sera dédié à la description de l'environnement de travail, à l'analyse architecturale et aux choix technologiques adoptés pour garantir la performance et la

fiabilité de notre application.

Chapitre 1

Étude préalable

1.1 Introduction

Dans ce chapitre, nous analysons le système de gestion de projet et de réclamation existant afin d'identifier la problématique et de proposer une solution. Ensuite, nous présenterons la méthodologie adoptée pour la réalisation de ce projet.

1.2 Étude et critique de l'existant

Ne disposant pas d'un système informatique dédié à l'assistance des personnes âgées, la gestion des rappels de médicaments, des rendez-vous médicaux et de la communication avec les proches repose principalement sur des méthodes traditionnelles, telles que les notes manuscrites ou les appels téléphoniques.

Cet état des lieux présente plusieurs inconvénients majeurs :

- Une difficulté à organiser et à suivre les rappels de médicaments et de rendez-vous.
- Une absence de communication efficace entre les personnes âgées, leurs proches et les soignants.
- Un manque de réactivité en cas d'urgence, en raison de l'absence d'un système d'alerte structuré.
- L'absence d'un historique centralisé permettant un suivi médical et organisationnel fiable.

L'analyse des solutions existantes est une étape essentielle pour mieux comprendre la problématique et définir les objectifs à atteindre. Elle permet d'identifier les forces et les faiblesses des solutions actuelles afin de proposer une approche optimisée.

Actuellement, plusieurs applications facilitent la gestion des rappels et le suivi médical, comme **Medisafe** et **MyTherapy**. Par ailleurs, des plateformes comme **Google Calendar** ou **Pill Reminder** offrent des fonctionnalités de rappel, mais elles ne sont pas spécifiquement adaptées aux besoins des personnes âgées.

1.2.1 Critique de l'existant

Dans le cadre de notre étude, nous avons analysé plusieurs solutions existantes d'assistance aux personnes âgées afin d'identifier leurs avantages et leurs limites. Cette analyse nous permet de mieux comprendre les besoins non couverts et d'adapter notre projet en conséquence.

Medisafe : Medisafe est une application mobile qui aide les utilisateurs à gérer leur prise de médicaments grâce à des rappels personnalisés et des notifications.

Avantages :

— Interface conviviale et facile à utiliser.
— Envoi de notifications pour rappeler la prise de médicaments.
— Possibilité de partager les informations avec un proche ou un soignant.

Inconvénients :

— Fonctionnalités limitées pour les personnes âgées ayant des difficultés avec les smartphones.
— Ne permet pas une interaction avancée avec les soignants.
— Absence d'un système de gestion des urgences.

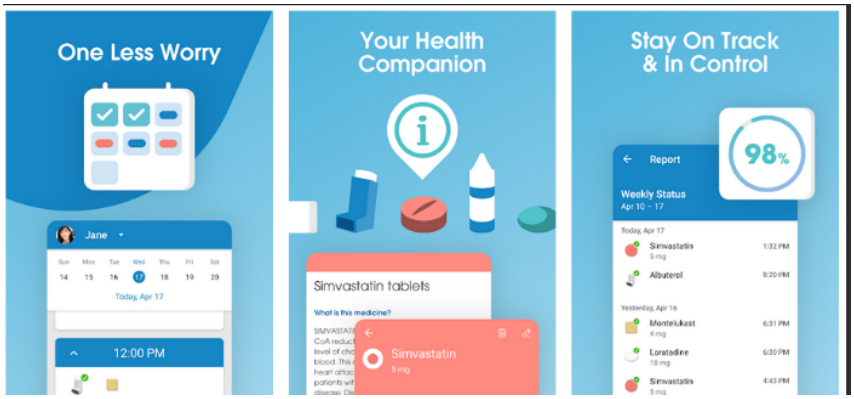

FIGURE 1.1 – Application Medisafe

Pill Reminder – Meds Alarm : Avantages :

— Facilité d'utilisation avec une interface intuitive.
— Possibilité d'ajouter plusieurs médicaments et d'organiser les prises en fonction des heures de la journée.

Inconvénients :

— Pas d'interaction avec des proches ou des soignants.
— Aucune fonctionnalité de gestion des urgences ni de journal de suivi.
— Pas d'intégration avec des assistants vocaux ou des options d'accessibilité avancée.
— Interface peu attractive.

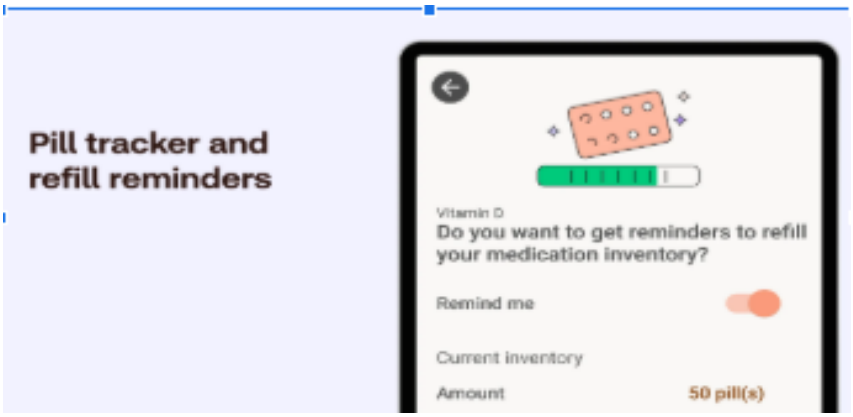

FIGURE 1.2 – Application Pill Reminder

Google Calendar : Avantages :

— Gratuit et disponible sur tous les appareils.
— Permet de planifier des rappels avec répétition automatique.
— Partage de calendrier avec les proches.

Inconvénients :

— Non spécifiquement conçu pour les personnes âgées.
— Pas de fonctionnalités de gestion des médicaments ou d'urgence.
— Dépendance à un compte Google.

FIGURE 1.3 – Application Google Calender

1.3 Solution proposée

L'analyse des solutions existantes a mis en évidence plusieurs limites qui empêchent une gestion efficace des rappels médicaux et des interactions entre les personnes âgées, leurs proches et les soignants.

Pour pallier ces insuffisances, nous proposons de développer une application mobile intelligente d'assistance aux personnes âgées, intégrant des fonctionnalités adaptées à leurs besoins spécifiques.

Fonctionnalités proposées :

— **Gestion des rappels** : prise de médicaments et rendez-vous médicaux.
— **Interaction** : partage de calendrier, notifications, et messagerie.
— **Accessibilité** : interface intuitive avec boutons larges, polices lisibles, et assistance vocale.
— **Chatbot intelligent** : pour aider à programmer les rappels et répondre aux questions.
— **Journal de suivi** : historique des rappels et interactions.
— **Bouton SOS** : envoi d'une alerte géolocalisée aux proches en cas d'urgence.

Toutes ces fonctionnalités sont intégrées dans un système sécurisé, performant et évolutif.

1.4 Choix méthodologiques

1.4.1 Formalisme de modélisation

Pour la spécification des besoins, nous avons opté pour le langage de modélisation UML, qui permet la description et la visualisation des exigences ainsi que la définition de l'architecture logicielle.

Nous avons utilisé :

- **Diagramme de cas d'utilisation** : pour visualiser le comportement global de l'application.
- **Diagramme de classes** : pour définir la structure de l'application.
- **Diagramme de séquences** : pour décrire les interactions entre les éléments du système et les acteurs.

1.4.2 Méthodologie Scrum

Depuis plusieurs années, les méthodes agiles, et en particulier Scrum, ont été adoptées pour assurer une gestion réactive et itérative des projets.

Présentation de la méthode Scrum : Scrum est une méthode agile créée en 2002. Le nom vient du rugby, où la "mêlée" représente l'organisation collective autour d'un objectif. Les projets Scrum sont découpés en itérations courtes appelées **Sprints** (1 à 3 semaines), permettant une évaluation et planification régulières.

Principaux éléments :

- **Product Backlog** : Liste des besoins fonctionnels (sous forme de User Stories).
- **User Story** : Description littérale et non technique d'une fonctionnalité.
- **Sprint Backlog** : Liste des User Stories à développer pendant un Sprint.
- **Sprint** : Itération courte ayant un objectif précis.
- **Task** : Une User Story est décomposée en tâches élémentaires.

Répartition des rôles :

- **Product Owner** : Représente le client final, priorise les besoins.
- **Scrum Master** : Facilite le processus Scrum, lève les obstacles.
- **Équipe de développement** : Auto-organisée, multidisciplinaire, réalise les tâches.

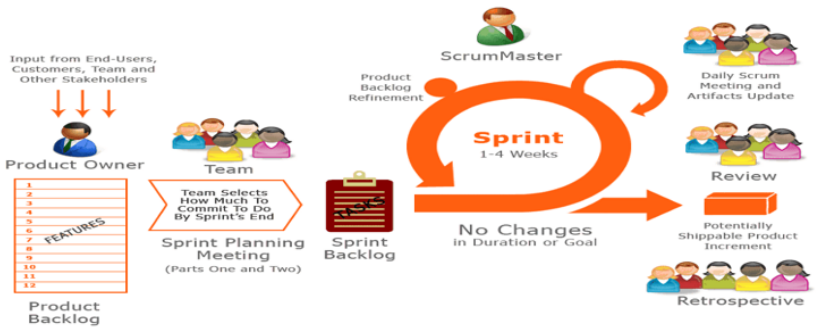

FIGURE 1.4 – Processus Scrum

Processus Scrum : Chaque Sprint commence par une réunion de planification pour fixer les objectifs, puis s'exécute avec une revue de Sprint et une rétrospective. Un diagramme illustre généralement ce cycle complet.

1.5 Conclusion

Dans ce chapitre, nous avons effectué une étude critique de l'existant en présentant notre solution qui va remédier aux insuffisances de l'application actuelle. Nous avons terminé le chapitre en introduisant la méthodologie ainsi que le formalisme de modélisation que nous avons adopté pour la conception du projet que nous allons spécifier dans le chapitre suivant.

Chapitre 2

Sprint Zéro

2.1 Introduction

Dans ce chapitre intitulé **Sprint Zéro**, nous présentons les premières étapes préparatoires du projet. Le Sprint Zéro (ou Sprint 0) ne se termine pas forcément par une livraison, mais permet de définir les bases du projet : exigences, participants, backlog initial, etc.

2.2 Spécification des besoins

2.2.1 Identification des acteurs

Un acteur est une entité externe interagissant avec le système. Dans notre application, les principaux acteurs sont :

Acteur	Description	Actions principales
Utilisateur (personne âgée / proche)	Utilisateur principal de l'application. Bénéficie de l'assistance pour les rappels, la communication et la sécurité.	— Recevoir des rappels. — Gérer les tâches quotidiennes. — Contacter les proches ou soignants. — Activer le bouton SOS.
Soignant	Professionnel de santé assurant le suivi.	— Accéder aux rappels. — Consulter le journal de suivi. — Être alerté en cas d'urgence.

TABLE 2.1: Tableau d'acteurs

2.2.2 Spécifications des besoins fonctionnels

Pour la personne âgée :

— Être assistée via rappels et notifications.
— Pouvoir communiquer avec proches et soignants.
— Utiliser un chatbot pour poser des questions.
— Suivre ses traitements et rendez-vous.
— Améliorer son autonomie.

Pour l'aidant :

— Recevoir des alertes de suivi.
— Accéder aux données de santé.
— Aider la personne âgée sans intrusion.

2.2.3 Spécifications des besoins non fonctionnels

— **Extensibilité** : ajout futur de nouvelles fonctionnalités.
— **Sécurité** : confidentialité et authentification sécurisée.
— **Interface graphique** : lisibilité et ergonomie pour utilisateurs seniors.
— **Maintenance** : code source clair et documenté.
— **Performance** : réactivité et temps de réponse rapides.

2.3 Pattern et style architectural

2.3.1 Style architectural – Architecture trois-tiers

L'architecture trois-tiers (*three-tier architecture*) permet une séparation logique des rôles dans l'application :

— **Client** : Interface utilisateur (mobile Flutter) ;
— **Serveur d'application (middleware)** : Spring Boot, qui traite les requêtes et applique la logique métier ;
— **Serveur de base de données** : MySQL, pour stocker les données des utilisateurs, rappels, profils, etc.
Cette séparation permet une meilleure performance, sécurité et évolutivité.

2.3.2 Pattern architectural – MVC

Le modèle MVC (Modèle-Vue-Contrôleur) structure l'application en trois composants :

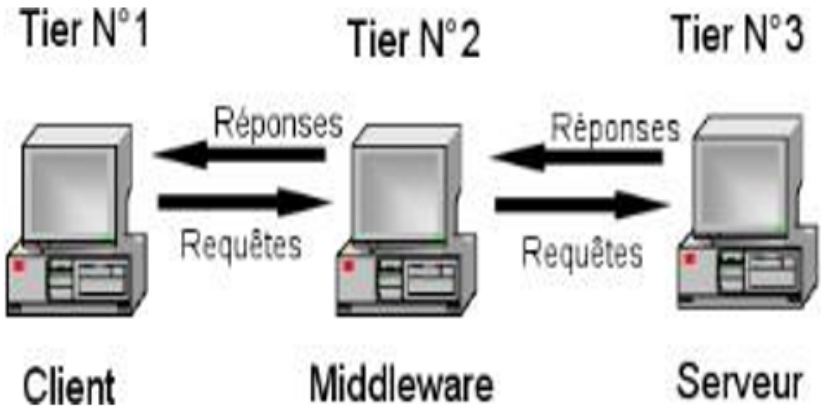

FIGURE 2.1 – modèle MVC

— **Modèle** : Contient la logique métier et gère les données (ex. : objets Utilisateur, Médicament).
— **Vue** : Interface utilisateur, affichant les informations (UI Flutter).
— **Contrôleur** : Traite les entrées utilisateurs, appelle les modèles et met à jour la vue.
Ce découpage rend le développement modulaire et facilite la maintenance.

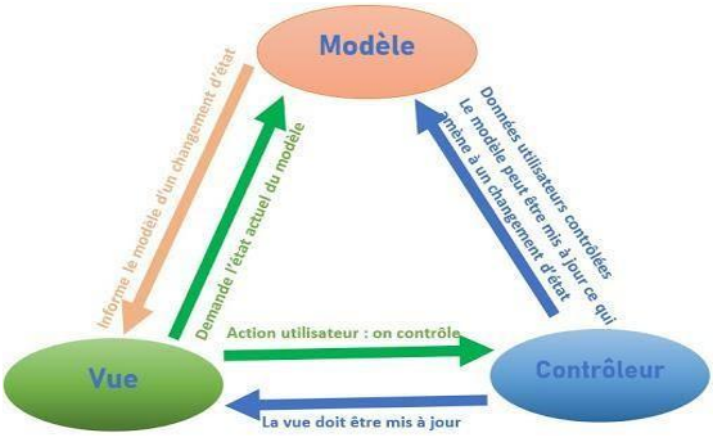

FIGURE 2.2 – architecture MVC

2.3.3 Pattern architectural – MVVM

MVVM (Model-View-ViewModel) est une variante moderne du MVC adaptée aux frameworks comme Flutter :
— **Model** : Gestion des données comme dans MVC.
— **View** : Affichage, sans logique métier.
— **ViewModel** : Lien entre vue et modèle, gère les données observables et la logique d'interface.
Ce modèle est adapté aux architectures réactives et à l'usage des états dans Flutter.

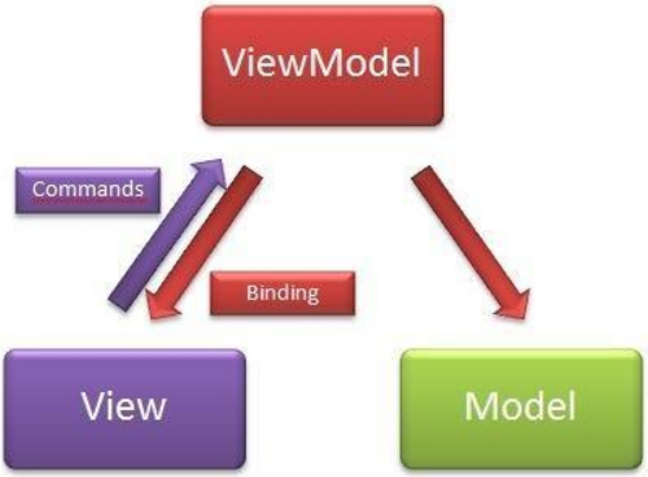

FIGURE 2.3 – architecture MVVM

2.4 Planning du traitement des cas d'utilisation

2.4.1 Importance et exigences

Certaines fonctionnalités sont plus prioritaires que d'autres. Pour cela, une planification par niveau d'importance est utilisée : élevé, moyen, faible.
Cette hiérarchisation permet de développer d'abord les fonctionnalités critiques et dépendantes des autres.

2.4.2 Backlog produit

ID : Identifiant unique de la fonctionnalité.

Fonctionnalité : Désignation de la fonctionnalité à implémenter.

User Story : Description du besoin utilisateur formulé

Priorité : Niveau d'importance de la user story

ID	**Fonctionnalité**	**ID Story**	**User Story**	**Priorité**
1	Authentification	1.1	- En tant qu'utilisateur, je peux m'authentifier pour accéder à l'application.	Élevée
2	Gestion utilisateurs	2.1	- En tant que personne âgée, je peux mettre à jour mes informations personnelles.	Élevée
		2.2	- En tant qu'aidant, je peux consulter la liste des personnes âgées sous ma responsabilité.	Moyenne
		2.3	- En tant qu'aidant, je peux afficher des infos par utilisateur	Moyenne

3	Aide et assistance	3.1	- En tant que personne âgée, je peux envoyer une demande d'aide à un aidant.	Élevée
		3.2	- En tant qu'aidant, je peux voir et répondre aux demandes d'aide	Élevée
		3.3	- En tant qu'aidant, je peux suivre l'état des demandes d'assistance.	Moyenne
4	Notifications	4.1	- En tant que personne âgée, je peux recevoir des rappels pour mes médicaments.	Élevée
		4.2	- En tant qu'aidant, je peux être notifié lorsqu'une personne âgée demande de l'aide.	Élevée
5	Messagerie	5.1	- En tant que personne âgée, je peux envoyer des messages à mon aidant.	Moyenne
		5.2	- En tant qu'aidant, je peux envoyer des messages aux personnes âgées dont je m'occupe.	Moyenne
6	Suivi de santé	6.1	- En tant que personne âgée, je peux enregistrer mes données de santé (tension, glycémie...).	Faible
		6.2	- En tant qu'aidant, je peux consulter l'historique des données de santé d'une personne âgée.	Moyenne

TABLE 2.2: Backlog global

2.4.3 Structure et découpage du projet – Sprints

Sprint : Itération de développement Agile.

Description du contenu : Fonctionnalités à réaliser durant le sprint.

Durée : Temps alloué en jours pour le sprint.

Sprint	**Description du contenu**	**Durée**
Sprint 1	Authentification + Mise à jour des infos + Consultation des profils	10 jours
Sprint 2	Envoi/réception des demandes d'aide + Notifications médicales + Alertes d'urgence	15 jours
Sprint 3	Fonctionnalité de messagerie	4 jours
Sprint 4	Enregistrement et historique des données santé	7 jours

TABLE 2.3: Décomposition des sprints

2.5 Technologies et outils de développement

2.5.1 Développement back-end

Spring Boot : Spring Boot est un framework Java qui facilite la création d'applications autonomes et prêtes à l'emploi. Il permet une configuration simplifiée et une gestion efficace des services backend.

FIGURE 2.4 – Spring Boot

MySQL : MySQL est un système de gestion de base de données relationnelle (SGBDR) open-source. Il est performant, évolutif et compatible avec SQL. Il permet de stocker les données liées aux utilisateurs, rappels, messages, etc.

FIGURE 2.5 – MySQL

Chatbot Gemini : Gemini est un modèle avancé d'intelligence artificielle développé par Google. Il permet d'intégrer un chatbot intelligent pour assister l'utilisateur : rappels, navigation, questions fréquentes, etc.

FIGURE 2.6 – Gemini

2.5.2 Développement front-end

Flutter : Flutter est un framework UI open-source développé par Google. Il permet de créer des applications mobiles multiplateformes (Android et iOS) avec un seul code source en Dart.

FIGURE 2.7 – Flutter

Ses avantages incluent :

— Développement rapide avec hot reload.
— Interface fluide et moderne.
— Intégration facile avec des APIs (Firebase, REST, etc.)

2.5.3 Environnement de développement

Matériel utilisé

FIGURE 2.8 – Matériel utilisé

— **Modèle** : ASUS TUF Gaming F15 (FX507VV-LP287W)
— **Processeur** : Intel Core i7-13620H (10 cœurs, jusqu'à 4.9 GHz)
— **Mémoire vive** : 16 Go DDR5 4800 MHz
— **Stockage** : SSD NVMe PCIe 4.0 de 512 Go
— **Carte graphique** : NVIDIA GeForce RTX 4060 8 Go GDDR6 (TGP jusqu'à 140W)
— **Écran** : 15,6" FHD (1920 x 1080), IPS, 144 Hz, 250 nits
— **Système d'exploitation** : Windows 11

Outils et environnements logiciels

— **IDE Back-end** : IntelliJ IDEA

FIGURE 2.9 – IntelliJ IDEA

— **IDE Front-end** : Visual Studio Code

FIGURE 2.10 – Visual Studio Code

— **SDK Flutter** : Flutter SDK 3.x

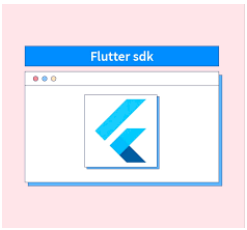

FIGURE 2.11 – Flutter SDK 3.x

— **JDK** : OpenJDK 17

FIGURE 2.12 – OpenJDK 17

— **Base de données** : MySQL Server 8.x

FIGURE 2.13 – MySQL Server 8.x

— **Outils de test** : Postman (pour les API REST)

FIGURE 2.14 – Postman

Android Studio (pour l'émulation Android)

FIGURE 2.15 – Android Studio

2.6 Modélisation UML

La modélisation UML permet de visualiser le comportement et la structure du système à travers des diagrammes standardisés.

2.6.1 Diagramme de cas d'utilisation

Ce diagramme montre les interactions entre les différents acteurs (personne âgée, aidant) et les fonctionnalités du système.

- Authentification
- Modification des informations
- Consultation des informations
- Envoi de demandes d'aide
- Réponse aux alertes
- Consultation de l'historique

FIGURE 2.16 – Diagramme des cas d'utilisation

2.6.2 Diagramme de classes

Ce diagramme représente les entités principales de l'application et leurs relations.

- Utilisateur (Personne âgée, Aidant)
- DemandeAide
- Notification

— Rappel
— Message
— SuiviSanté

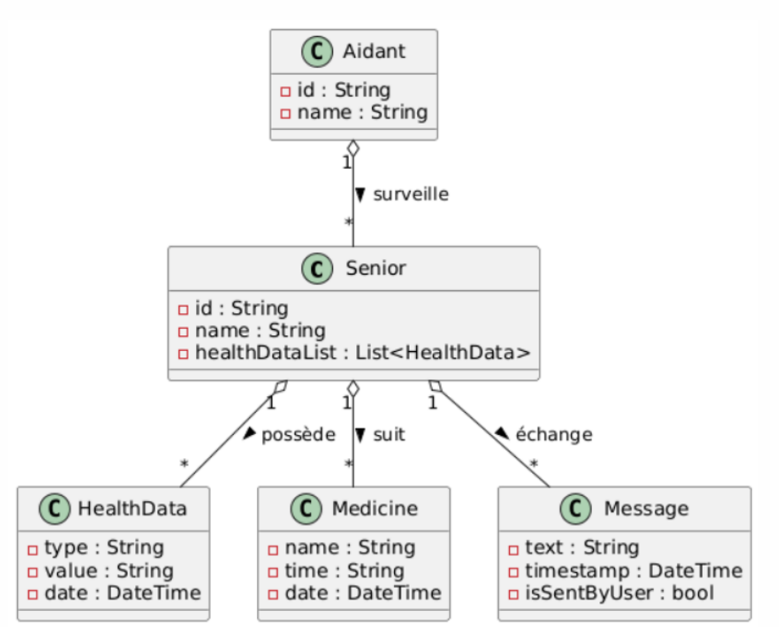

FIGURE 2.17 – Diagramme de classes

2.6.3 Diagramme de séquences

Ce diagramme illustre les échanges entre les composants lors d'un scénario précis (ex : authentification ou alerte SOS).

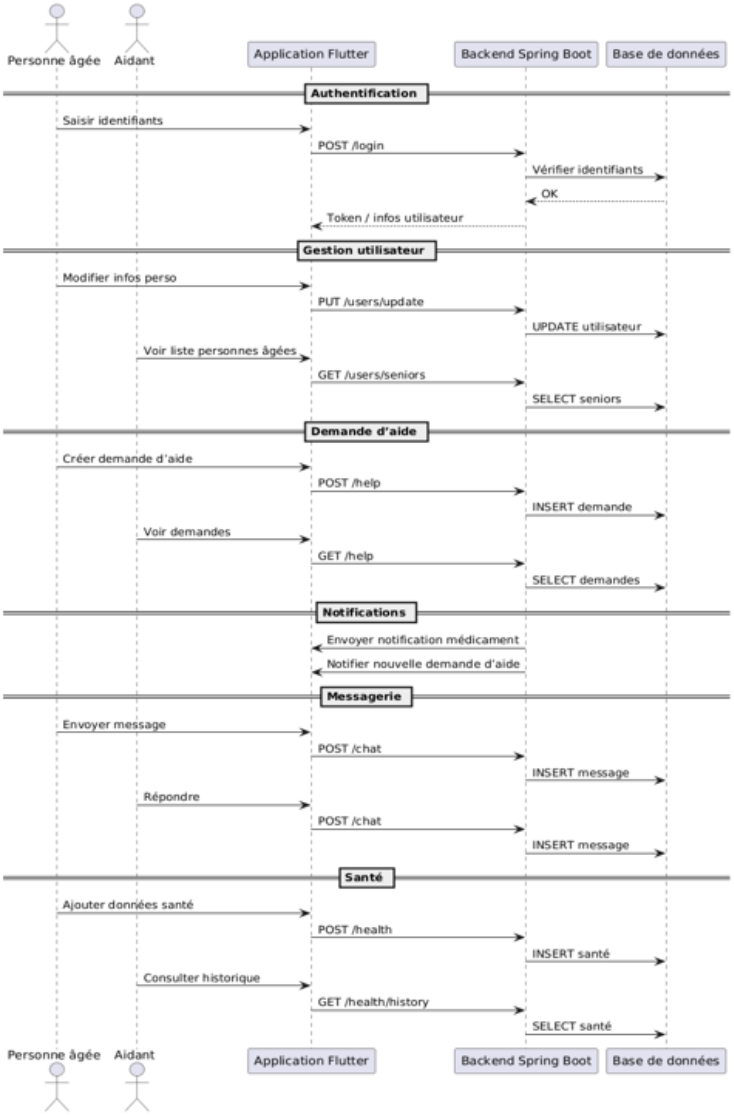

FIGURE 2.18 – Diagramme de séquences

2.7 Conclusion

Dans ce chapitre, nous avons planifié notre travail en suivant les instructions de l'approche de conception Scrum. Ainsi, nous avons défini les spécifications et les exigences de l'application à développer. Après, nous avons publié le backlog du produit contenant une liste des fonctionnalités attendues de l'application et nous avons cité quelques prototypes d'interfaces. De plus, nous avons préparé la répartition du sprint . Dans le chapitre suivant, nous allons commencer à modéliser le projet en commençant par le premier sprint.

Chapitre 3

Sprint 1

3.1 Introduction

Dans le chapitre précédent, nous avons défini la spécification des exigences. Ensuite, nous avons divisé le projet en sprints. Chaque sprint spécifie une période durant laquelle le développement doit être fait et revu. Dans ce sens, nous détaillons dans ce chapitre le premier sprint qui comprend l'authentification de l'utilisateur, la mise à jour des informations personnelles, ainsi que la consultation de la liste des informations personnelles par l'aidant.

3.2 Backlog du sprint

ID : Identifiant de la User Story

User Story : Besoins exprimés par l'utilisateur.

Tâche : Actions nécessaires pour réaliser la User Story

ID	**User Story**	**Tache**
1.1	En tant qu'utilisateur, je peux m'authentifier.	-Réaliser les diagrammes des cas d'utilisation, de séquence et de classes : « S'authentifier » -Développer le cas « S'authentifier » (Frontend + Backend) -Tester le cas « S'authentifier » (tests fonctionnels + unitaires)
2.1	En tant que personne âgée, je peux mettre à jour mes informations personnelles.	-Réaliser les diagrammes des cas d'utilisation, de séquence et de classes : « Modifier profil » -Développer la fonctionnalité « Modifier profil » (modification nom, âge, santé...) -Tester la mise à jour des informations personnelles

2.2	En tant qu'aidant, je peux consulter la liste des personnes âgées sous ma responsabilité.	- Réaliser les diagrammes des cas d'utilisation, de séquence et de classes : « Consultation utilisateurs » -Développer la consultation des profils des personnes âgées par l'aidant -Tester la consultation des informations côté aidant

TABLE 3.1: Backlog du sprint 1

3.3 Spécifications fonctionnelles

3.3.1 Classification des cas d'utilisation par acteur

Personne agée : S'authentifier , Modifier ses informations personnelles , Consulter ses informations personnelles

Aidant : S'authentifier , Modifier ses informations personnelles , Consulter ses informations personnelles

3.3.2 Diagramme de cas d'utilisation de sprint 1

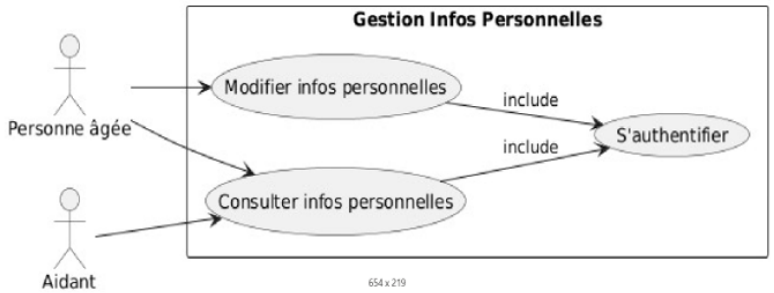

FIGURE 3.1 – Diagramme de cas d'utilisation de sprint 1

- La personne âgée peut réaliser l'ensemble des cas d'utilisation relatifs à la gestion de son compte. Elle peut ainsi s'authentifier, modifier ses informations personnelles, et consulter ses informations personnelles.
- L'aidant dispose des mêmes fonctionnalités. Il peut également s'authentifier, modifier ses informations personnelles et consulter les informations personnelles de la personne aidée.
- Chaque action inclut obligatoirement une phase d'authentification préalable, représentée par une relation «include» dans le diagramme.

3.3.3 Cas d'utilisation : S'authentifier

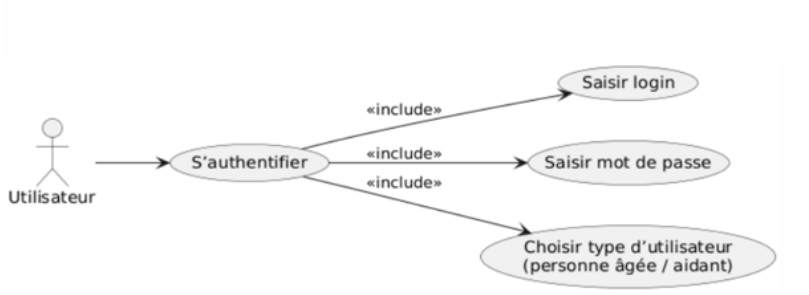

FIGURE 3.2 – cas d'utilisation : s'authentifier

Acteurs : Personne âgée, Aidant
Préconditions : L'utilisateur est enregistré dans le système.
Postcondition : Accès à l'espace personnel (aidant ou senior).

Scénario principal :

1. L'utilisateur lance l'application.
2. Il saisit son adresse e-mail et mot de passe.
3. Il choisit son rôle (senior ou aidant).
4. Il clique sur "Se connecter".
5. Le système valide et redirige vers la page d'accueil.

Scénario alternatif :

— Email ou mot de passe incorrect → message d'erreur.
— Champ vide → affichage d'un message de validation.

3.3.4 Cas d'utilisation : Modifier les informations personnelles

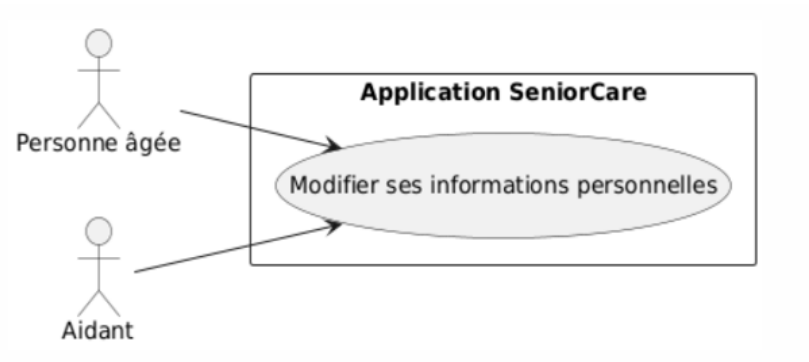

FIGURE 3.3 – Cas d'utiliation : Modifier les informations personnelles

Acteurs : Utilisateur authentifié
Préconditions : Accès à la section "Mon Profil".
Postcondition : Les nouvelles informations sont sauvegardées.

Scénario principal :

1. Accès à “Mon Profil”.
2. Clic sur “Modifier”.
3. Saisie des nouvelles informations.
4. Validation.
5. Mise à jour dans la base + message de confirmation.

Scénario alternatif :

— Champ vide ou invalide → message d’erreur, annulation de la mise à jour.

3.3.5 Cas d’utilisation : Consulter les informations personnelles

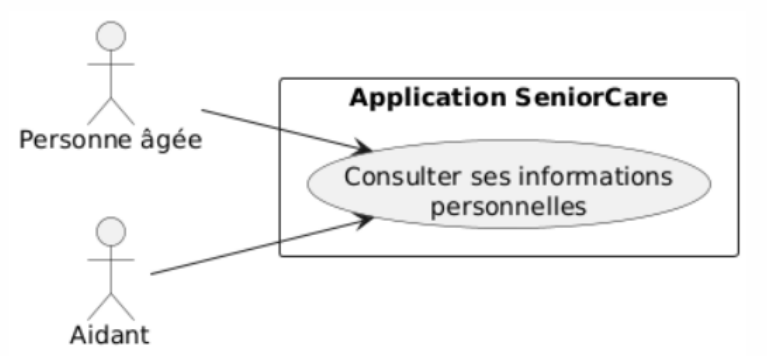

FIGURE 3.4 – Cas d’utilisation : Consulter les informations personnelles

Acteurs : Utilisateur connecté
Préconditions : Authentification réussie
Postcondition : Affichage des données

Scénario principal :

1. Clic sur l’icône “Mon Profil”
2. Chargement des données
3. Affichage sur écran

Scénario alternatif :

— Erreur de chargement → affichage d’un message d’erreur

3.4 Interfaces utilisateur

3.4.1 Écran d’authentification

Cet écran permet à l’utilisateur (personne âgée ou aidant) de saisir ses identifiants pour accéder à son espace personnel.

— Champs : nom d’utilisateur, Mot de passe
— Bouton : Se connecter

— Options : Mot de passe oublié, Créer un compte

FIGURE 3.5 – Ecran d'authentification

3.4.2 Écran de création de compte

— Saisie : Nom, Mot de passe
— Sélection du rôle (senior ou aidant)
— Bouton : Créer un compte

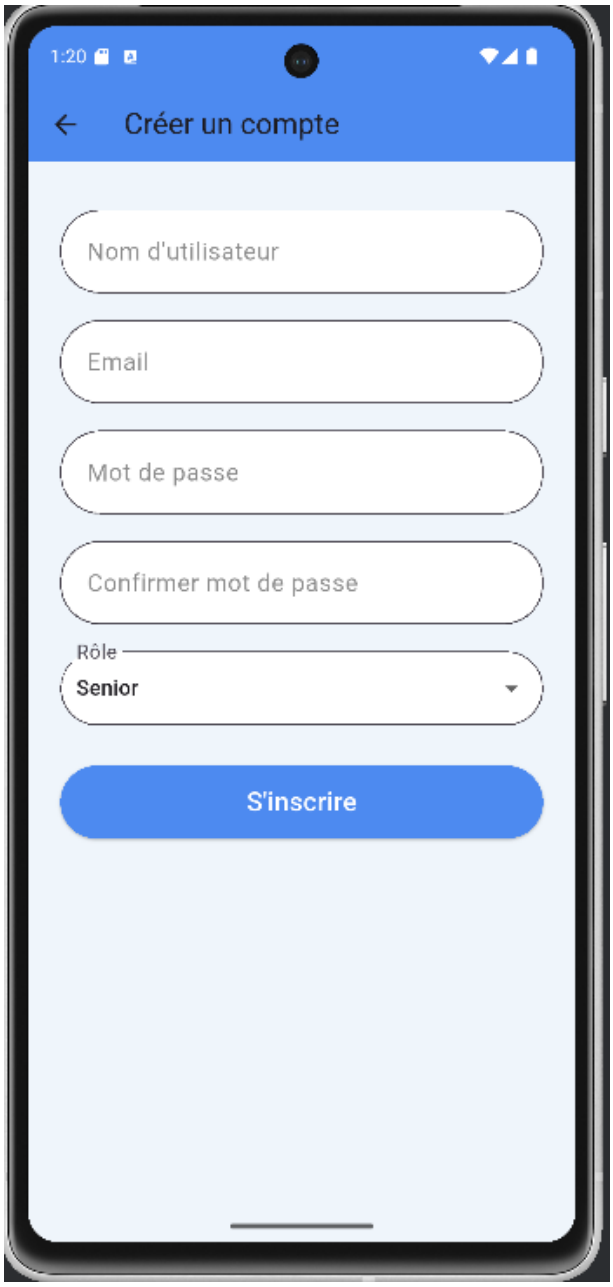

FIGURE 3.6 – Ecran de création de compte

3.4.3 Écran du mot de passe oublié

— Saisie : Email
— Bouton : Envoyer

FIGURE 3.7 – Ecran du mot de passe oublié

3.4.4 Tableau de bord Aidant

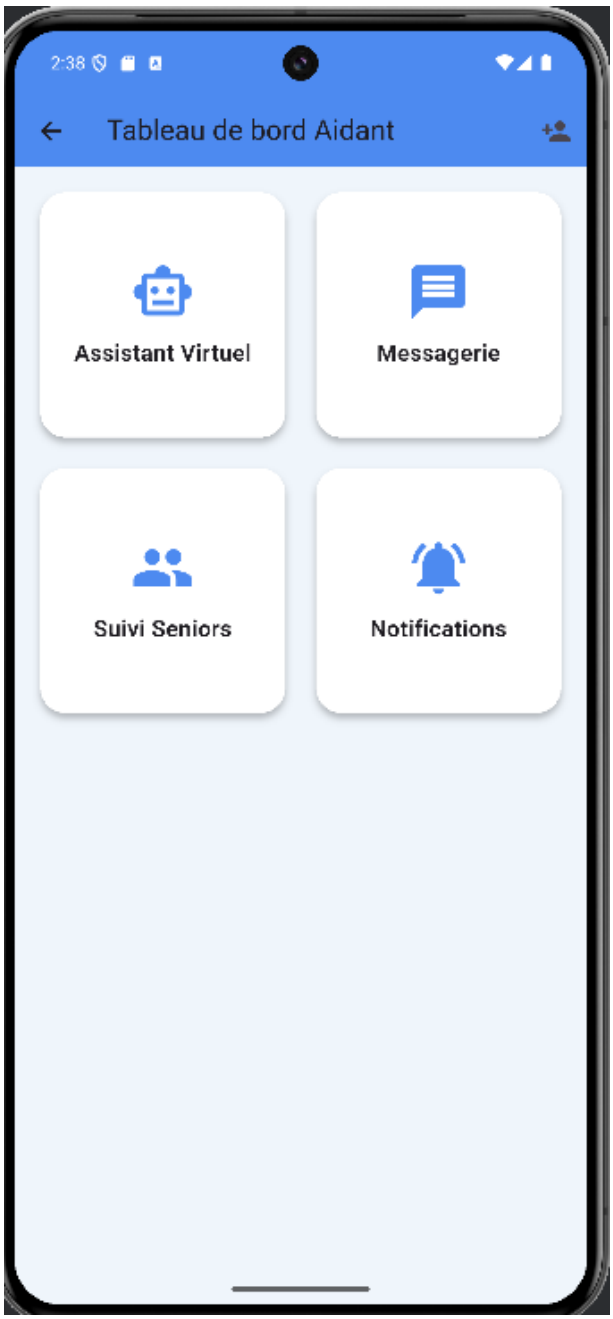

FIGURE 3.8 – Tableau de bord Aidant

3.4.5 Tableau de bord Senior”

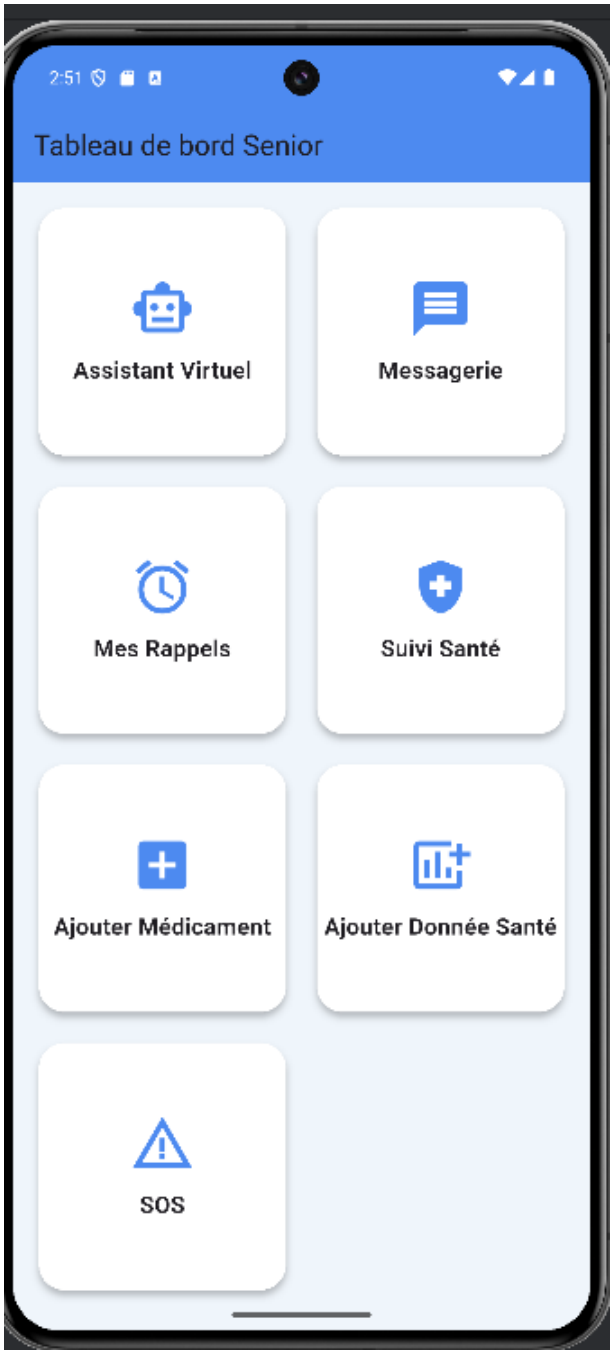

FIGURE 3.9 – Tableau de bord Senior

3.5 Conclusion

Dans la première phase du projet, nous avons réussi à vérifier le premier sprint. Maintenant, nous avons le premier incrément de l’application. Dans le prochain chapitre, nous allons concevoir et développer un deuxième Sprint