



Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes

FILIERE : GENIE LOGICIEL

application Web/Mobile de suivi de Budget

Realise par :
BAKHOUCH Nisrine
EL BOUZIYANI Anas

Sous la direction de :
Pr EL HAMLAOUI Mahmoud
Membre de Jury :
Pr NASSAR Mahmoud

Département Informatique
ENSIAS - UM5
Année Académique 2023/2024

Remerciement

Au moment où nous pensons avoir fait uvre utile, qu'il nous soit permis d'exprimer nos vifs remerciements et notre profonde gratitude à Monsieur le Professeur HAMLAOUI Mahmoud, qui a bien voulu et en dépit de ses multiples engagements tant professionnels que personnels, accepter de diriger et de parrainer ce travail. Qu'il veuille bien nous permettre de rendre hommage à ses qualités exceptionnelles, à son savoir-faire, et à sa contribution bénéfique à la réalisation de ce travail.

Nos remerciements vont aussi et de manière respectueuse et combien reconnaissante au grand Monsieur le Professeur NASSAR Mahmoud membre éminent de notre jury, qui nous a consacré son précieux temps afin d'assister et soutenir notre projet.

Par la même occasion, nous tenons à remercier très chaleureusement les Forces Armées Royales, la Direction de l'Enseignement Supérieur, et la totalité du personnel de l'ENSIAS pour tous les efforts louables qu'ils ont consentis pour nous permettre de réussir ce modeste travail, digne de lauréats de cette prestigieuse école.

Résumé

Notre projet académique de fin d'année se concentre sur le développement d'une application mobile dédiée à la gestion de syndic de copropriété afin de digitaliser les tâches manuelles et de renforcer la transparence des opérations. Pour atteindre ces objectifs, une approche Agile, en particulier la méthodologie Scrum, a été adoptée pour gérer le projet de manière itérative.

En termes de réalisation nous avons utilisé «UML» pour la conception, et nous avons adopté les meilleures pratiques de développement logiciel ainsi que les technologies modernes telles que «Android Jetpack Compose» pour le développement de l'interface utilisateur, «Gradle» pour la gestion de build et «Firebase» pour les services backend.

Le résultat final est une application mobile fonctionnelle offrant une interface utilisateur intuitive et conviviale.

Mots clés : Agile, Application, Android, Dagger-Hilt, Firebase, Gradle, JetPack-Compose, Mobile, Scrum, UML.

Abstract

The main focus of our academic project is the development of a mobile application for managing a joint property in order to optimize the existing processes of budget managing and enforce the transparency of operations. In order to achieve our goals, we opted for a Scrum/Agile approach, Meanwhile we used the latest technologies (Android JetPack Compose for ui, MVVM as a model, Firebase for backend...) the end result is a fully functional application

Keywords : Agile, Application, Android, Dagger-Hilt, Firebase, Gradle, JetPack-Compose, Mobile, Scrum, UML.

Table des matières

Remerciements	3
Résumé	4
Abstract	5
Table des figures	10
Liste des tableaux	11
1 Introduction générale	12
2 Contexte général de projet	13
2.1 Problématique	13
2.2 Objectif	13
2.3 L'analyse des besoins	13
2.3.1 Besoins fonctionnels	14
2.3.2 Besoins non fonctionnels	14
3 Spécifications techniques	15
3.1 Méthodologie	15
3.1.1 Agile manifesto	15
3.1.2 Scrum/Agile	15
3.1.2.1 Les intervenants	15
3.1.2.2 La conduite du Scrum Agile	16
3.2 Technologies	17
3.2.1 Android	17
3.2.2 Kotlin	17
3.2.3 Les bonnes pratiques	17
3.2.3.1 Conposition basique d'un logiciel android	17
3.2.4 Règles à suivre :	18
3.2.4.1 Modèle MVVM	18
3.2.5 GitHub	19
3.2.5.1 GitHub Issues	19
3.2.5.2 GitHub projects	19
3.2.5.3 GitHub Actions	19
3.2.6 Android JetPack Compose	20
3.2.7 Gradle	20
3.2.8 Firebase	21

3.2.9	Firebase FirebaseAuth	21
3.2.10	Firebase Firestore	21
3.2.11	DaggerHilt (injecteur de dependances)	21
4	Déroulement des Sprints	22
4.1	Sprint 0 : Mise en Oeuvre	22
4.1.1	Identification des acteurs	22
4.1.2	UserStories (Backlog du produit)	22
4.1.3	Prototypage des interfaces	23
4.1.3.1	L'authentification	23
4.1.3.2	Ajout de cotisation et de dépense	24
4.1.3.3	L'affichage des situations	25
4.1.4	Déroulement	26
4.2	Sprint 1	27
4.2.1	Spécifications fonctionnelles	27
4.2.1.1	Inscription	27
4.2.1.2	Connexion	27
4.2.1.3	Réinitialisation du mot de passe	27
4.2.2	Sprint Backlog	27
4.2.3	Conception	28
4.2.3.1	Diagramme de cas d'utilisation	28
4.2.3.2	Diagramme de classe	29
4.2.4	Réalisation	31
4.2.4.1	Interface de connexion	31
4.2.4.2	Interface d'inscription et de réinitialisation de mot de passe	31
4.3	Sprint 2	36
4.3.1	Spécifications fonctionnelles	36
4.3.1.1	Consulter la situation des mois	36
4.3.1.2	Consulter les opérations de chaque mois	36
4.3.2	Sprint Backlog	36
4.3.3	Conception	37
4.3.3.1	Diagramme de cas d'utilisation	37
4.3.3.2	Diagramme de classe	38
4.3.4	Réalisation	39
4.3.4.1	Les interfaces	39
4.4	Sprint 3	41
4.4.1	Spécifications fonctionnelles	41
4.4.1.1	Gérer les cotisations	41
4.4.1.2	Gérer les dépenses	41
4.4.2	Sprint Backlog	41
4.4.3	Conception	42
4.4.3.1	Diagramme de cas d'utilisation	42
4.4.3.2	Diagramme de classe	43
4.4.4	Réalisation	45
4.4.4.1	L'accès au menu latéral	45
4.4.4.2	Ajout de dépense	45
4.4.4.3	Ajout et modification de type de dépense	48
4.4.4.4	Ajout de cotisation	49

4.4.4.5	Suppression des opérations	51
5	Conclusion	52
	Bibliographie	52
A	Annexes	54

Table des figures

3.1	Schéma d'une architecture d'application android	18
3.2	Architecture MVVM dans android avec FireBase	18
4.1	Prototype de l'écran d'inscription	23
4.2	Prototype de l'écran de connexion	23
4.3	Prototype de l'écran de réinitialisation du mot de passe	23
4.4	Prototype du menu latérale	24
4.5	Prototype de l'écran d'ajout des contributions	24
4.6	Prototype de l'écran d'ajout des dépenses	24
4.7	Prototype d'affichage pour l'administrateur	25
4.8	Prototype d'affichage pour l'utilisateur	25
4.9	Prototype d'affichage des opérations	25
4.10	Le diagramme de GANTT	26
4.11	BurnDownChart	26
4.12	Le diagramme de cas d'utilisation pour Sprint 1	29
4.13	Diagramme de classe métier pour Sprint 1	30
4.14	Le diagramme de Class pour Sprint 1	31
4.15	Écran de connexion (light mode)	32
4.16	Écran de connexion (dark mode)	32
4.17	Écran d'inscription	33
4.18	Écran de réinitialisation de mot de passe	33
4.19	Animation lors de connexion	34
4.20	Exemple d'erreur 1	34
4.21	Exemple d'erreur 2	35
4.22	Exemple d'erreur 3	35
4.23	Le diagramme de cas d'utilisation pour Sprint 2	37
4.24	Diagramme de classe métier pour Sprint 2	38
4.25	Le diagramme de Class pour Sprint 2	39
4.26	Écran de la liste des mois	40
4.27	Écran de la liste des opérations de janvier 2024	40
4.28	Le diagramme de Class pour Sprint 3	42
4.29	Diagramme de classe métier pour Sprint 3	43
4.30	Le diagramme de Class pour Sprint 3	44
4.31	Menu latéral accessible par l'administrateur	45
4.32	Message d'erreur si l'administrateur essaye d'ajouter ou supprimer une opération avec une date ancienne	45
4.33	Exemple d'interface pour ajouter des dépenses	46
4.34	Exemple d'opération en cours (ajout de dépense)	46

4.35 Calendrier pour sélectionner la date de dépense	47
4.36 Exemple de message de réussit d'ajout de dépenses	47
4.37 Interface d'ajout d'un nouveau type de dépense a la liste des types des dépenses	48
4.38 Interface de modification d'un type de dépense	48
4.39 Exemple d'interface pour ajouter des cotisation	49
4.40 Exemple d'opération en cours (ajout de cotisation)	49
4.41 Exemple de message de réussit d'ajout de cotisation	50
4.42 Exemple de suppression d'une opération par l'administrateur	51
4.43 Message de réussit	51
A.1 Diagramme de classe métier	54
A.2 Product Backlog de sprint 1	55
A.3 Product Backlog de sprint 2	55
A.4 Product Backlog de sprint 3	56

Liste des tableaux

4.1	Backlog du 1er sprint	28
4.2	Backlog du 2eme sprint	36
4.3	Backlog du 3eme sprint	41

Introduction générale

La gestion efficace des syndics représente un enjeu majeur dans le secteur immobilier. Néanmoins, les approches traditionnelles reposant sur des processus manuels se révèlent souvent inefficaces, entraînant des erreurs et des retards. Notre objectif principal était de créer une solution innovante afin de transformer les processus de gestion conventionnels en une approche numérique pour optimiser les processus de gestion, renforcer la transparence des opérations et permettre une accessibilité pour toutes les parties prenantes. Dans ce rapport, nous détaillons notre démarche méthodologique, les outils et technologies utilisées, ainsi que les résultats obtenus tout au long du processus de développement de l'application, nous analyserons comment notre application peut rendre les processus de gestion des copropriétés plus efficaces, transparents et conviviaux, ce qui se traduira par une meilleure expérience pour toutes les parties impliquées.

Contexte général de projet

2.1 Problématique

La gestion des copropriétés constitue un défi de taille pour de nombreux syndics, requieront une coordination minutieuse d'une pluralité de tâches telles que la communication avec les résidents, la collecte des cotisations, la gestion des dépenses. Ces processus sont fréquemment confrontés à des inefficacités et des erreurs humaines lorsqu'elles sont conduites manuellement ou selon des méthodes conventionnelles.

Dans le cadre de notre cursus académique de fin d'année, nous avons choisi d'investiguer le développement d'une solution innovante pour répondre à ces défis. Ainsi, l'élaboration d'une application mobile spécifiquement dédiée à la gestion de syndic de copropriété offrant une opportunité significative d'optimiser les processus existants, d'améliorer la communication entre les différentes parties prenantes et de renforcer la transparence des opérations.

2.2 Objectif

Dans ce contexte, notre projet de fin d'année se fixe pour but la conception et le développement d'une application mobile de gestion de syndic de copropriété. Notre objectif est de répondre à l'interrogation suivante : de quelle manière une application mobile dédiée peut-elle concourir à l'optimisation des procédures de gestion, à l'amélioration de la communication entre les résidents et le syndic, ainsi qu'au renforcement de la transparence des opérations, afin de satisfaire de manière efficiente les besoins des propriétaires ?

2.3 L'analyse des besoins

Dans cette section, nous procédons à une analyse approfondie des exigences inhérentes à notre application mobile consacrée à la gestion du syndic de copropriété. Cette démarche nous permettra de formuler des recommandations spécifiques pour la conception et le développement de notre solution de gestion de syndic de copropriété, assurant ainsi une réponse adéquate aux défis et aux besoins des utilisateurs. De plus, nous identifierons les fonctionnalités essentielles ainsi que les exigences techniques et de performance.

2.3.1 Besoins fonctionnels

Les besoins fonctionnels définissent les fonctionnalités spécifiques que l'application doit offrir pour répondre aux attentes des utilisateurs et aux exigences opérationnelles. Voici une élargissement des exigences fonctionnelles de l'application :

- Surveillance de l'état budgétaire : L'application doit permettre aux utilisateurs, en particulier aux administrateurs, de suivre de manière précise et en temps réel l'état financier global de la copropriété. Cela implique de fournir des informations détaillées sur les revenus et les dépenses, ainsi que sur le solde actuel du budget.
- Résumé mensuel du budget : Les utilisateurs devraient pouvoir visualiser de manière claire et concise un résumé mensuel du budget, mettant en évidence les entrées et les sorties d'argent, les cotisations perçues, les dépenses engagées et le solde restant.
- Historique des transactions mensuelles : Pour une gestion transparente et une traçabilité des finances, il est nécessaire de fournir un historique détaillé des transactions effectuées pour un mois donné. Cela permettra aux utilisateurs de comprendre en détail les mouvements financiers et de vérifier la validité des opérations.
- Différenciation des droits d'accès : une distinction claire est établie entre les utilisateurs réguliers et les administrateurs. Les administrateurs devraient avoir des priviléges étendus pour effectuer des opérations de gestion, tandis que les utilisateurs réguliers devraient avoir un accès restreint aux fonctionnalités essentielles
- Fonctionnalités administratives :
 - + **Ajouter des cotisations** : Permettre aux administrateurs d'ajouter de nouvelles cotisations et de définir leurs montants respectifs.
 - + **Modifier des cotisations** : Autoriser les administrateurs à apporter des modifications aux cotisations existantes, que ce soit pour ajuster les montants ou les périodicités.
 - + **Ajouter des dépenses** : Offrir aux administrateurs la possibilité d'enregistrer de nouvelles dépenses engagées par la copropriété.
 - + **Modifier des dépenses** : Permettre aux administrateurs de corriger toute erreur ou inexactitude dans les dépenses enregistrées.
 - + **Ajouter un nouvel utilisateur** : Autoriser les nouveaux utilisateurs à créer de nouveaux comptes pour avoir l'accès aux données.
 - + **Réinitialiser le mot de passe d'un utilisateur** : Offrir la possibilité aux utilisateurs de réinitialiser les mots de passes en cas de besoin, garantissant ainsi la sécurité des comptes.

2.3.2 Besoins non fonctionnels

concernent les aspects techniques et de performance de l'application notamment :

- **Sécurité** : L'application doit garantir la confidentialité et l'intégrité des données.
- **Performance** : L'application doit être rapide et offrir une expérience utilisateur optimale.
- **Extensibilité** : Il est important de concevoir l'application de manière à ce qu'elle puisse être étendue avec de nouvelles fonctionnalités à l'avenir.
- **Maintenance** : Il est essentiel de mettre en œuvre une architecture qui facilite la maintenance de l'application.

Spécifications techniques

3.1 Méthodologie

La méthodologie est un ensemble de principes utilisés pour guider un processus spécifique. Dans notre projet, nous optons pour la méthodologie Agile.

3.1.1 Agile manifesto

La conduite d'un projet logiciel selon la méthodologie Agile[2] suppose une approche distincte qui privilégie la communication transparente, valorise l'individu et soutient les changements radicaux, tout en favorisant l'auto-organisation des équipes, les incitant à prendre des décisions autonomes et à s'adapter rapidement aux changements.

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

3.1.2 Scrum/Agile

Selon «The Scrum Guide™», Scrum est « un framework léger qui aide les personnes, les équipes, et les organisations à générer de la valeur grâce à des solutions adaptatives à des problèmes complexes » [1]. Scrum est le framework agile le plus largement utilisé et le plus populaire. Le terme agile décrit un ensemble spécifique de principes et de valeurs fondamentaux pour l'organisation et la gestion d'un travail complexe. Bien qu'il ait ses racines dans le développement de logiciels, Scrum fait aujourd'hui référence à un cadre léger utilisé dans tous les secteurs pour fournir des produits et services complexes et innovants qui ravissent réellement les clients. C'est simple à comprendre, mais difficile à maîtriser.

3.1.2.1 Les intervenants

Scrum compte trois responsabilités (précédemment appelées «rôles») garantissant que chaque aspect du travail partagé est géré de manière efficace.

1. **Propriétaire du produit** : Le propriétaire du produit communique l'objectif du produit, possède les user stories du produit et veille à ce que l'équipe s'attaque toujours au travail de la plus haute valeur. Il connaît et comprend le domaine ainsi que le marché de son produit.

2. **Maître Scrum** : Le maître Scrum guide et dirige l'organisation dans son adoption et sa pratique du Scrum. Le maître Scrum aide l'équipe à construire le produit et à devenir la meilleure équipe possible en les guidant dans l'utilisation du Scrum et l'adaptation des principes agiles. Il coach l'équipe vers une utilisation efficace des événements et des artefacts. Sa journée peut inclure l'aide à la gestion des obstacles rencontrés par l'équipe, et il est souvent essentiel à la croissance de l'équipe dans son ensemble.
3. **Développeurs** : Professionnels de l'équipe Scrum qui travaillent ensemble pour créer n'importe quel aspect du produit. Ils créent les incrémentums de produit pendant les sprints.

3.1.2.2 La conduite du Scrum Agile

Il y a cinq événements dans le cadre Scrum. Ces événements sont des occasions précieuses pour inspecter et adapter le produit ou la manière dont l'équipe travaille ensemble.

1. **Le sprint** : Au cœur de Scrum, une période limitée dans le temps (moins d'un mois et fréquemment de 1 à 2 semaines) pendant laquelle un ou plusieurs incrémentums sont créés. Le sprint contient tous les autres événements.
2. **Planification de sprint** : L'ensemble de l'équipe Scrum établit l'objectif du sprint. Les développeurs prévoient le travail qu'ils estiment pouvoir accomplir pendant le sprint pour soutenir l'objectif, et comment le travail choisi sera effectué. Sur la base de l'objectif du sprint et des prévisions, un plan initial est également créé.
3. **Scrum quotidien** : Pendant le scrum quotidien, les développeurs inspectent la progression vers l'objectif du sprint et adaptent les plans si nécessaire. C'est un événement quotidien bref dirigé par les développeurs pour inspecter et adapter. Il est limité dans le temps (15 minutes à 20 minutes). Le scrum quotidien n'est pas la seule opportunité pour l'équipe d'adapter ses plans, elle communique souvent sur les points nécessaires en dehors de cet événement. Lors du scrum quotidien, l'équipe peut synchroniser son travail, identifier les blocages et discuter de la collaboration qui doit avoir lieu. Le scrum quotidien aide l'équipe à comprendre si ses derniers plans les rapprocheront de l'objectif du sprint.
4. **Revue de sprint** : L'ensemble de l'équipe Scrum inspecte le résultat du sprint et détermine les adaptations futures. Un livrable est communiqué au propriétaire du produit pour avoir ces remarques et savoir si ce livrable répond partiellement à ces besoins. Le backlog du produit est adapté en fonction des remarques du propriétaire du produit.
5. **Rétrospective de sprint** : La conclusion du sprint, la rétrospective est l'occasion pour l'équipe d'inspecter ses propres interactions, collaborations, processus, outils et tout autre facteur qu'elle juge pertinent pour sa capacité à s'améliorer continuellement.

3.2 Technologies

3.2.1 Android

Android est un système d'exploitation mobile open source fondé sur le noyau Linux, il est développé par un consortium d'entreprises, « Open Handset Alliance», [11] sponsorisé par Google. Android est défini comme étant une pile de logiciels, c'est-à-dire un ensemble de logiciels destinés à fournir une solution intégrale pour les appareils mobiles, smartphones et tablettes tactiles. Cette pile comporte un système d'exploitation (comprenant un noyau Linux), les applications essentielles telles que le navigateur web, le téléphone et le carnet d'adresses ainsi que des logiciels intermédiaires entre le système d'exploitation et les applications.

Android est distribué en open source sous licence Apache. La licence autorise les constructeurs qui intègrent Android dans leurs appareils à y apporter des modifications leur permettant de se distinguer de leurs concurrents, ce qui a été adopté par la quasi-totalité des fabricants de produits concurrents de l'iPhone.

3.2.2 Kotlin

Kotlin est un langage de programmation orienté objet et fonctionnel, avec un typage statique, permet de compiler pour la machine virtuelle Java, JavaScript, et vers plusieurs plateformes en natif (grâce à LLVM).[12] Son développement provient principalement d'une équipe de programmeurs chez JetBrains basée à Saint-Pétersbourg en Russie (son nom vient de l'île de Kotline, près de St. Pétersbourg).

Google annonce pendant la conférence Google I/O 2017 que Kotlin devient le second langage de programmation officiellement pris en charge par Android après Java. Le 8 mai 2019, toujours lors de la conférence Google I/O, Kotlin devient officiellement le langage de programmation voulu et recommandé par le géant américain Google pour le développement des applications Android. Pivotal Software annonce le 4 janvier 2017 le support officiel de Kotlin sur la cinquième version du Framework Spring.

3.2.3 Les bonnes pratiques

Dans la documentation Android, on trouve les recommandations ou les "bonnes pratiques" à suivre dans la conception d'une application Android.

3.2.3.1 Conposition basique d'un logiciel android

Généralement, un logiciel Android se compose de trois couches fondamentales.

- **Couche UI** : consiste à afficher les données de l'application à l'écran. Chaque fois que les données changent, soit à la suite d'une interaction de l'utilisateur (par exemple, si celui-ci appuie sur un bouton) ou d'une entrée externe (telle qu'une réponse du réseau), l'UI doit être mise à jour pour refléter les modifications.
- **Couche de domaine** : (facultative) chargée d'encapsuler une logique métier complexe, ou une logique métier simple qui est réutilisée par plusieurs ViewModels.
- **Couche de données** : contient la logique métier. La logique métier est ce qui donne de la valeur à votre application. Elle repose sur des règles qui déterminent la manière dont votre application crée, stocke et modifie les données.

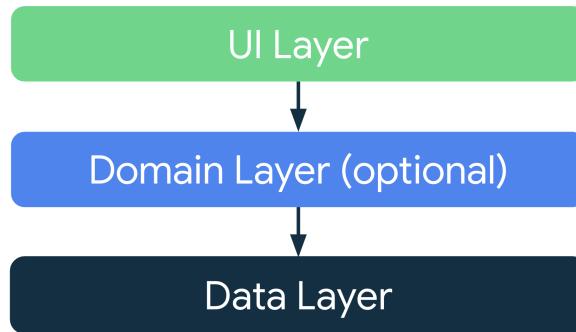


FIGURE 3.1 – Schéma d'une architecture d'application android

source : <https://www.developer.android.com>

3.2.4 Règles à suivre :

Selon[7], il est fortement recommandé de suivre les règles suivantes :

1. Ne stockez pas de données dans les composants UI d'une application.
2. Réduisez les dépendances aux classes Android.
3. Créez des limites de responsabilité bien définies entre les différents modules de votre application
4. Exposez le moins d'éléments possible dans chaque module.

3.2.4.1 Modèle MVVM

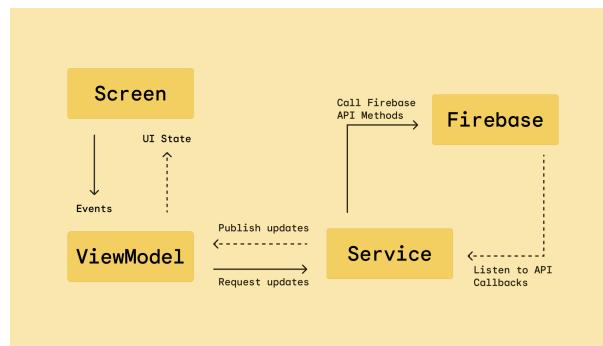


FIGURE 3.2 – Architecture MVVM dans android avec FireBase

source : firebase.blog

Le modèle MVVM (Modele-View-ViewModel) est un Design pattern (Modèle de conception) visant à séparer la logique de la présentation d'une application, il est basé sur 3 couches[10] :

1. **Model** : Le modèle contient les données liés à la logique métier. Il peut s'agir par exemple d'entités issues de bases de données ou encore d'API externes.

2. **View** : La vue est la description de l'interface graphique, elle fait le lien entre les actions de l'utilisateur et le modèle de vue.
Elle définit où et comment sont placés les composants graphiques sur l'interface et décrit les liaisons de données (Data Bindings) entre les valeurs affichées et le modèle de vue.
3. **ViewModel** : Le modèle de vue est chargé de transformer et organiser les modèles métiers afin d'exposer les données à afficher par la vue.

3.2.5 GitHub

GitHub est une plateforme qui offre la possibilité de stocker, partager et collaborer avec d'autres le code source. En stockant ce dernier dans un "référentiel" sur GitHub, il est possible :

1. Suivre et gérer les modifications apportées à votre code au fil du temps.
2. Permettre à d'autres personnes de réviser votre code et de faire des suggestions pour l'améliorer.
3. Collaborer sur un projet partagé sans craindre que les modifications aient un impact sur le travail des collaborateurs avant l'intégration.

3.2.5.1 GitHub Issues

Le GitHub Issues constituent des entités créées dans un dépôt pour planifier, discuter et suivre le travail, en assignant des responsabilités, en fixant des priorités et en ajoutant des étiquettes pour classer les problèmes[5]. En intégrant GitHub Issues dans notre projet qui adopte la méthodologie agile Scrum a offert une organisation transparente du travail, une collaboration efficace et une surveillance stratégique de la progression tout au long du cycle de développement logiciel. Ces Issues servent à représenter les user stories, à assigner des tâches aux membres de l'équipe, suivre l'avancement du travail et discuter des obstacles lors des réunions quotidiennes et des revues de Sprint.

3.2.5.2 GitHub projects

GitHub projects propose des fonctionnalités complémentaires à GitHub Issues permettant ainsi aux équipes de mieux planifier. Ils offrent une vue d'ensemble organisé des backlogs de chaque sprint. Grâce à GitHub Projects, l'avancement de notre travail est surveillé en temps réel, facilitant ainsi la coordination et la communication au sein de notre équipe. À la fin de chaque Sprint, les données visuelles fournies par GitHub Projects permettent d'examiner le travail accompli et de discuter des améliorations à apporter lors des revues de Sprint et des rétrospectives.

3.2.5.3 GitHub Actions

GitHub Actions est une plateforme d'intégration et livraison continue (CI/CD) qui permet d'automatiser le pipeline de génération, de test et de déploiement[4]. Il permet de créer des workflows qui compilent et testent chaque demande d'intégration de code à la branche principale (pull request) adressée à votre dépôt, ou déployer en production après validations des tests.

3.2.6 Android JetPack Compose

Jetpack Compose est un kit d'outils moderne conçu pour simplifier le développement des interfaces utilisateur[9]. Il suit un modèle de programmation réactif à la concision et à la facilité d'utilisation du langage de programmation Kotlin. Il adopte une approche déclaratif.

3.2.7 Gradle

Gradle est un moteur de production[6] fonctionnant sur la plateforme Java. Il permet de construire des projets en Java, Kotlin, Scala, Groovy voire C++. L'outil a été développé pour la compilation d'exécutables multi-projets, qui tendent à être gourmands en espace. Son fonctionnement est basé sur une série de tâches de compilation qui sont exécutées de manière séquentielle ou parallèle.

3.2.8 Firebase

Firebase[3] est une plateforme qui regroupe un ensemble d'outils pour l'hébergement des bases de données, offre les services d'authentification, gestionnaire de notifications et de publicités, surveillance de l'état des applications

3.2.9 Firebase FirebaseAuth

Firebase Authentication fournit des services backend, des SDK faciles à utiliser et des bibliothèques prêtes à l'emploi pour authentifier les utilisateurs auprès d'une application. Il prend en charge l'authentification à l'aide de mots de passe, de numéros de téléphone, de fournisseurs d'identité fédérés populaires tels que Google, Facebook et Twitter, etc.

3.2.10 Firebase FireStore

Cloud Firestore est une base de données flexible et évolutive pour le développement mobile, Web et serveur à partir de Firebase et Google Cloud. Il maintient les données synchronisées entre les applications clientes via realtime listeners et offre une prise en charge hors ligne pour mobile et Web afin de permettre la création des applications réactives qui fonctionnent quelle que soit la latence du réseau ou de la connectivité internet.

3.2.11 DaggerHilt (injecteur de dépendances)

Hilt est une bibliothèque d'injection de dépendances pour Android[8] qui réduit le code récurrent nécessaire pour injecter manuellement les dépendances dans un projet. Hilt offre une méthode standard pour utiliser l'injection de dépendances dans une application en fournissant des containers pour chaque classe Android du projet et en gérant automatiquement leur cycle de vie. Hilt repose sur la bibliothèque d'injection de dépendances Dagger et permet d'intégrer Dagger à une application Android de manière standard.

Déroulement des Sprints

4.1 Sprint 0 : Mise en Oeuvre

Avant de débuter la conception avec Scrum/agile, il est essentiel d'identifier d'abord les acteurs, de comprendre leurs besoins et attentes, ainsi que de mettre en place la réalisation du backlog.

4.1.1 Identification des acteurs

les principaux acteurs sont :

- **Administrateur (gérant du syndique) :**
 1. gérer la recette ;
 2. marquer les dépenses ;
 3. marquer les cotisations des membres.
- **Utilisateur (membre du syndique) :**
 1. consulter la situation.

4.1.2 UserStories (Backlog du produit)

basant sur les besoins des acteurs on a définis les UserStories suivantes :

- En tant qu'administrateur, je veux avoir la possibilité de m'authentifier en utilisant mon login et mon mot de passe de manière sécurisée afin d'exploiter l'application.
- En tant qu'utilisateur de l'application, je veux avoir la possibilité de m'authentifier en utilisant mon login et mon mot de passe de manière sécurisée afin d'exploiter l'application.
- En tant qu'utilisateur, je veux avoir la possibilité de créer un nouveau compte utilisateur afin de l'utiliser.
- En tant qu'utilisateur, je veux avoir la possibilité de réinitialiser mon mot de passe afin de retrouver l'accès à mon compte.
- En tant qu'administrateur, je veux avoir la possibilité de désactiver le compte d'un utilisateur de l'application afin d'annuler les droits d'accès a un utilisateur.
- En tant qu'administrateur, je veux avoir la possibilité d'ajouter un type de dépenses à la liste des dépenses prédefinies dans l'application afin de mieux catégoriser les dépenses.
- En tant qu'administrateur, je veux avoir la possibilité de modifier un type de dépense à la liste des dépenses prédefinies dans l'application afin de mieux catégoriser les dépenses.

- En tant qu'administrateur, je veux avoir la possibilité d'ajouter une contribution à la liste des contributions en spécifiant l'utilisateur, le montant, et la date (qui doit être le jour même par défaut) afin d'enregistrer les contributions.
- En tant qu'administrateur, je veux avoir la possibilité de supprimer une contribution de la liste des contributions afin de réctifier les enregistrements incorrects.
- En tant qu'administrateur, je veux avoir la possibilité d'ajouter une dépense à la liste des dépenses en spécifiant le type, le montant, et la date (qui doit être le jour même par défaut) afin d'enregistrer les dépenses.
- En tant qu'administrateur, je veux avoir la possibilité de supprimer une dépense de la liste des dépenses afin de rectifier les enregistrements incorrects.
- En tant qu'utilisateur, je veux voir les revenus, les dépenses, et le solde de chaque mois afin de mieux saisir la situation.
- En tant qu'utilisateur, je veux voir en détail les dépenses et les revenus par mois afin de mieux saisir la situation de chaque mois.

4.1.3 Prototypage des interfaces

4.1.3.1 L'authentification

Les interfaces dédiées à l'authentification des utilisateurs



FIGURE 4.1 – Prototype de l'écran d'inscription



FIGURE 4.2 – Prototype de l'écran de connexion



FIGURE 4.3 – Prototype de l'écran de réinitialisation du mot de passe

4.1.3.2 Ajout de cotisation et de dépense

Les interfaces dédiées à l'ajout des cotisation et des dépenses



FIGURE 4.4 – Prototype du menu latérale

Figure 4.5 : Prototype de l'écran d'ajout des contributions. L'écran est intitulé "Ajout de cotisations". Il contient les champs suivants : "utilisateur" (avec un bouton de déroulement), "Date dd/mm/aaaa" (avec un calendrier) et "Montant 0 DH". En bas de l'écran se trouve un bouton "Ajouter".

FIGURE 4.5 – Prototype de l'écran d'ajout des contributions

Figure 4.6 : Prototype de l'écran d'ajout des dépenses. L'écran est intitulé "Ajout de dépenses". Il contient les champs suivants : "type de dépense" (avec un bouton de déroulement et un symbole de plus +), "Date dd/mm/aaaa" (avec un calendrier) et "Montant 0 DH". En bas de l'écran se trouve un bouton "Ajouter".

FIGURE 4.6 – Prototype de l'écran d'ajout des dépenses

4.1.3.3 L'affichage des situations

Les interfaces dédiées à l'affichage des situations

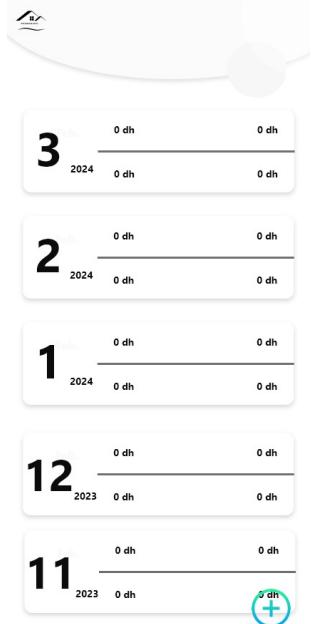


FIGURE 4.7 – Prototype d'affichage pour l'administrateur

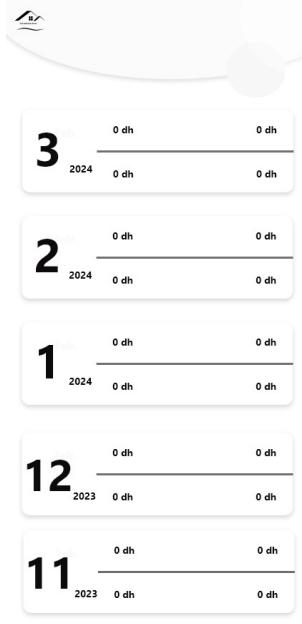


FIGURE 4.8 – Prototype d'affichage pour l'utilisateur

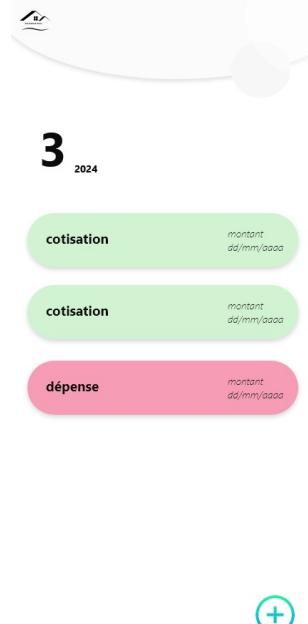


FIGURE 4.9 – Prototype d'affichage des opérations

4.1.4 Déroulement

Le projet s'est déroulé sur 3 sprints, chaque sprint a duré 2 semaines donc le projet a pris 6 semaines pour le réaliser.

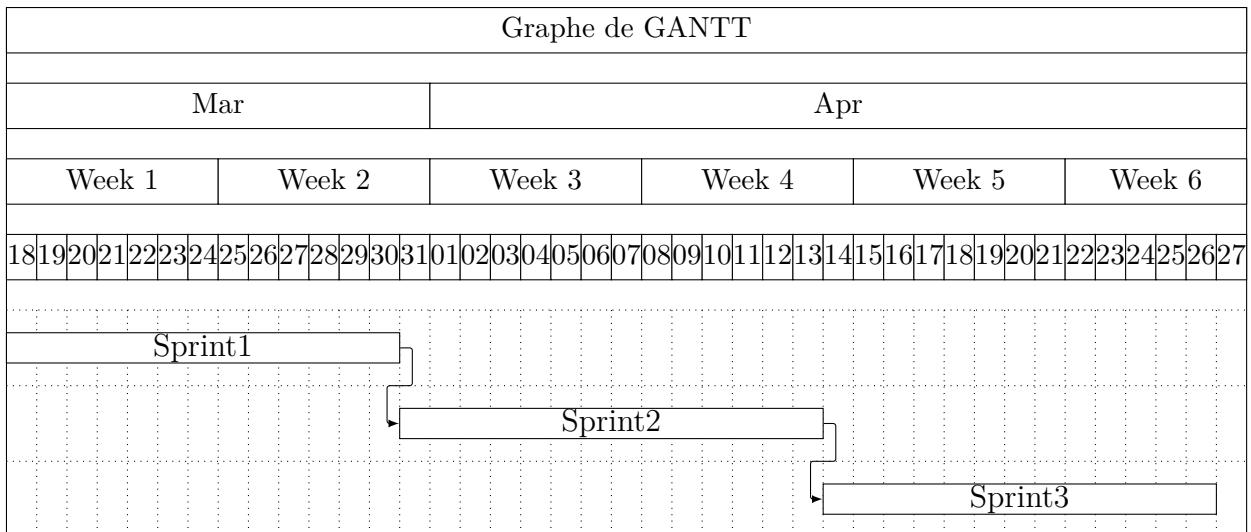


FIGURE 4.10 – Le diagramme de GANTT

Le nombre totale des tâches est 62, les détails de ces tâches sont dans l'annexe figure A.2,figure A.3 et figure A.4,la vélocité de l'équipe était suffisante pour finir le projet avant la fin du 3ème sprint, le diagramme Burndown illustre ça.

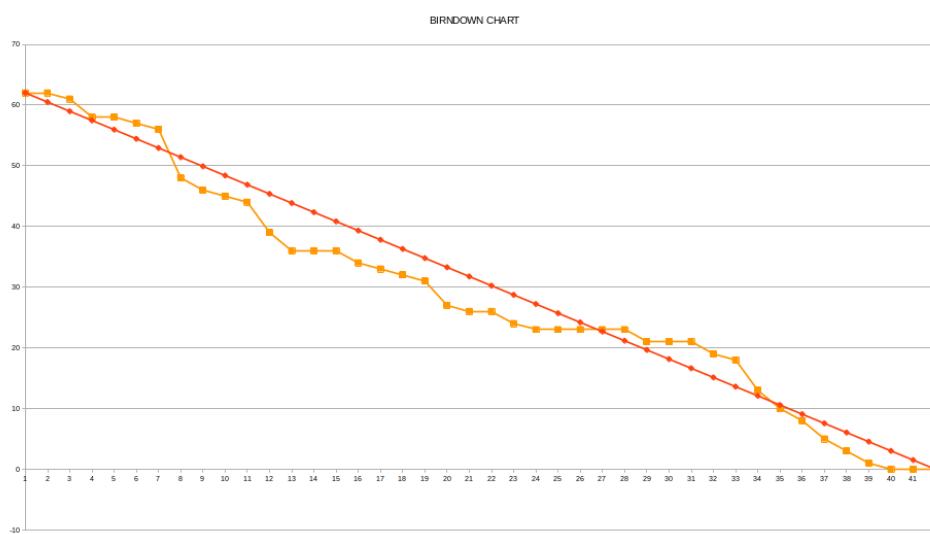


FIGURE 4.11 – BurnDownChart

4.2 Sprint 1

4.2.1 Spécifications fonctionnelles

4.2.1.1 Inscription

Cette opération permet à un utilisateur de créer un nouveau compte dans l’application. L’utilisateur fournit ses informations personnelles telles que son nom, son prénom, son adresse e-mail et un mot de passe pour créer son compte. Après une vérification des informations fournies le nouvel utilisateur sera enregistrer dans la base de données.

4.2.1.2 Connexion

L’opération de connexion permet à un utilisateur enregistré d’accéder à son compte. L’utilisateur saisit son adresse e-mail et son mot de passe dans les champs prévus à cet effet. L’application vérifie les informations saisies et authentifie l’utilisateur si les informations sont correctes.

4.2.1.3 Réinitialisation du mot de passe

Cette opération permet à un utilisateur de réinitialiser son mot de passe en cas d’oubli. L’utilisateur fournit son adresse e-mail associée à son compte. L’application envoie un lien de réinitialisation par mail à l’utilisateur. Ce dernier peut ensuite cliquer sur le lien pour choisir un nouveau mot de passe et le mettre à jour dans la base de données.

4.2.2 Sprint Backlog

Le sprint Backlog contient les UserStories relatant la connexion, l’authentification, renouvellement du mot de passe, et l’inscription.

GitHub ID	Sprint Backlog	Acteur	Priorité
#2	en tant qu'administrateur, je veux avoir la possibilité de m'authentifier en utilisant mon login et mon mot de passe de manière sécurisée afin d'exploiter l'application	Administrateur	ELVEE
#3	En tant qu'utilisateur de l'application, je veux avoir la possibilité de m'authentifier en utilisant mon login et mon mot de passe de manière sécurisée afin d'exploiter l'application	Utilisateur	ELEVEE
#4	En tant qu'utilisateur, je veux avoir la possibilité de créer un nouveau compte utilisateur afin de l'utiliser	Utilisateur	MOYEN
#5	En tant qu'utilisateur, je veux avoir la possibilité de réinitialiser mon mot de passe afin de retrouver l'accès à mon compte	Utilisateur	MOYEN

TABLE 4.1 – Backlog du 1er sprint

4.2.3 Conception

Au cours de la phase initiale de conception du premier Sprint, nous avons choisi de focaliser notre attention sur les «user stories» liées à l'authentification. Les exigences se concentrent sur la possibilité de se connecter à l'application en utilisant un identifiant et un mot de passe, ainsi que sur la capacité de créer un compte et de réinitialiser le mot de passe en cas de besoin.

4.2.3.1 Diagramme de cas d'utilisation

D'après le Sprint Backlog, nous observons la présence de deux acteurs principaux : l'utilisateur et l'administrateur. Ces deux acteurs ont besoin de se connecter via un identifiant et un mot de passe. De plus, ils doivent tous deux avoir la capacité de réinitialiser le mot de passe. Seul l'utilisateur aura le droit de s'inscrire.

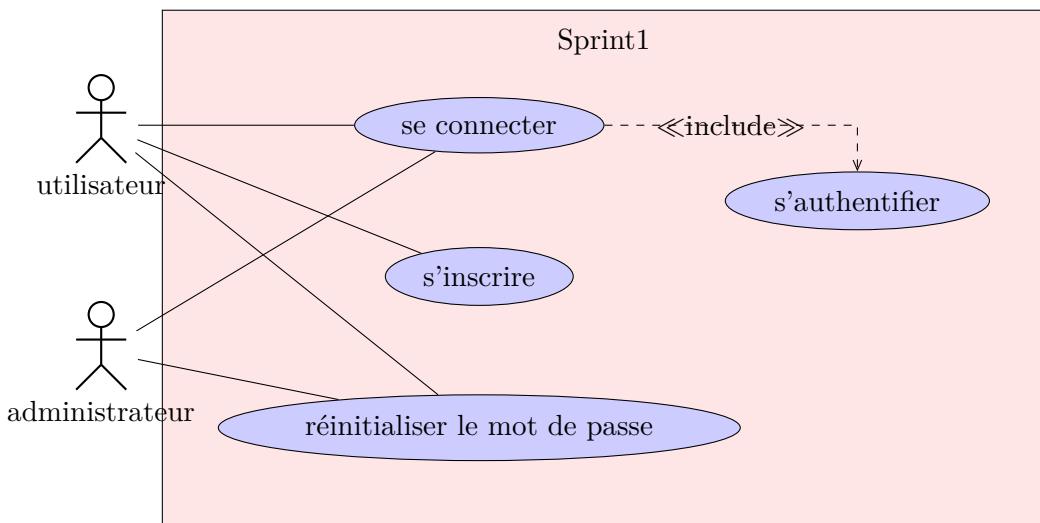


FIGURE 4.12 – Le diagramme de cas d'utilisation pour Sprint 1

4.2.3.2 Diagramme de classe

basé sur les meilleures pratiques citées auparavant, nous avons opté pour le modèle MVVM. Nous avons dû créer 3 vues pour :

- la connexion ;
- l'inscription ;
- la réinitialisation du mot de passe.

Ces vues ont été créées en utilisant Jetpack Compose, un framework d'interface utilisateur déclaratif basé sur des fonctions plutôt que sur des classes «Activity» ou les anciens fichiers XML, ce qui explique pourquoi les vues ne sont pas mentionnées dans le diagramme de classes.

Pour les modèles de données, nous avons dû créer un modèle pour l'utilisateur ; une classe contenant une variable booléenne décrivant s'il s'agit d'un administrateur ou non, et plusieurs variables de type chaîne de caractère contenant le nom, le nom de famille, l'identifiant et l'e-mail.

Pour le viewModel : nous avons opté pour un ViewModel unifié car cela n'est pas au contraire des meilleures pratiques et pour simplifier la base du code.

La classe authViewModel est une classe qui hérite de la classe ViewModel et contient 3 valeurs mutableState utilisées pour mettre à jour les interfaces graphiques de chaque vue : LoginUiState, registerUiState, resetUiState pour les vues de connexion, d'inscription et de réinitialisation respectivement.

Le viewModel contient également une valeur accountService injectée lors de la création du viewModel (en utilisant le constructeur) grâce à Dagger-Hilt Dependency Injector.

Le accountService est une interface où nous avons déclaré toutes les fonctions nécessaires. Nous avons créé une autre classe qui étend cette interface appelée FirebaseAccountService afin de faciliter le support d'autres backends.

Dans le AuthViewModel, nous avons implémenté toutes les fonctions nécessaires appelées par les vues. Par exemple : login est une fonction appelé lorsque l'utilisateur clique sur le bouton «connexion». Il récupère l'e-mail et le mot de passe à partir de LoginUiState, puis il passe ces paramètres et appelle la méthode authenticate de l'instance FirebaseAccountService.

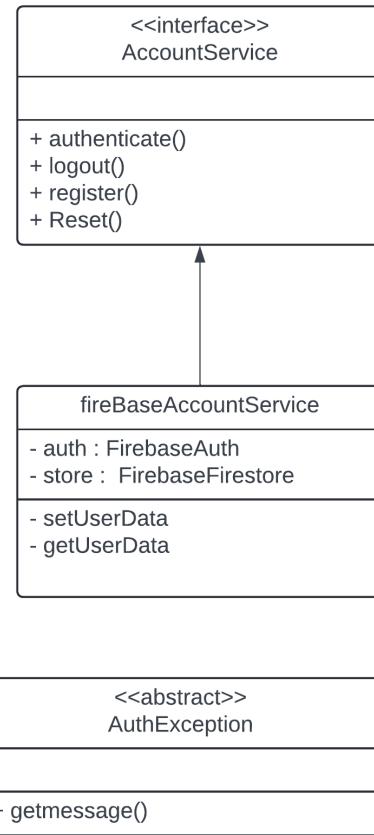


FIGURE 4.13 – Diagramme de classe métier pour Sprint 1

Pour personnaliser les messages d’erreurs de Snackbar, nous avons dû créer nos propres exceptions, la classe **AuthException**. Toutes sortes d’exceptions survenues lors de l’authentification sont des exceptions spécifiques à Firebase qui sont converties en exceptions héritées de **AuthException**. Ensuite, ces exceptions sont gérées par la fonction **loginExceptionHandler** du **AuthViewModel**, qui récupère simplement le message de l’exception et l’affiche à l’aide d’un Snackbar.

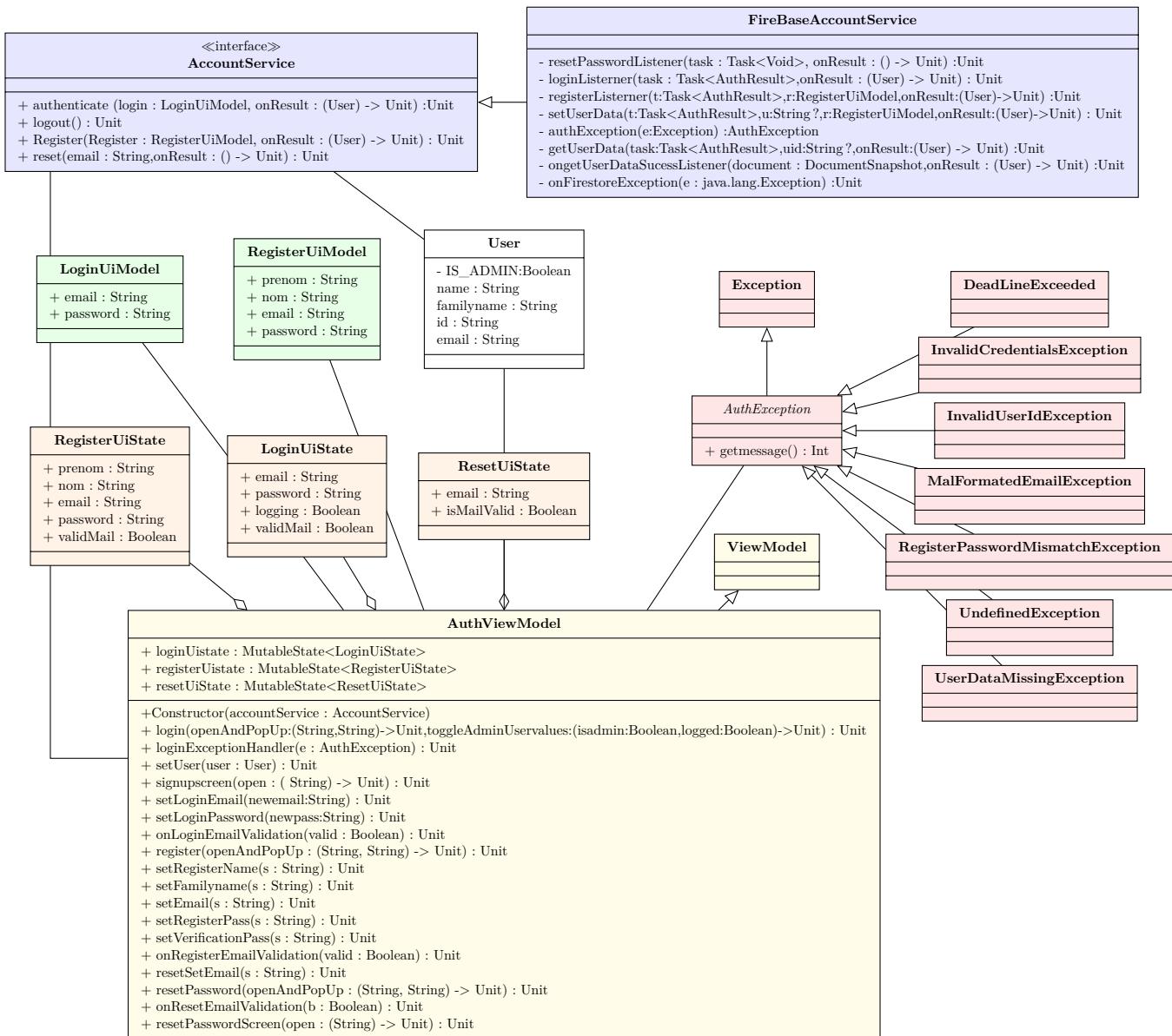


FIGURE 4.14 – Le diagramme de Class pour Sprint 1

4.2.4 Réalisation

4.2.4.1 Interface de connexion

L’interface de connexion en mode Clair et Sombre.

4.2.4.2 Interface d’inscription et de réinitialisation de mot de passe

L’interface d’inscription à gauche avec les différents champs nécessaires. À droit, l’interface de réinitialisation de mot de passe.



FIGURE 4.15 – Écran de connexion (light mode)



FIGURE 4.16 – Écran de connexion (dark mode)



FIGURE 4.17 – Écran d’inscription



FIGURE 4.18 – Écran de réinitialisation de mot de passe

L'interface de connexion à gauche avec animation (attent de réponse du serveur firebase). À droit, la même interface reportant que l'e-mail ou le mot de passe est incorrect.

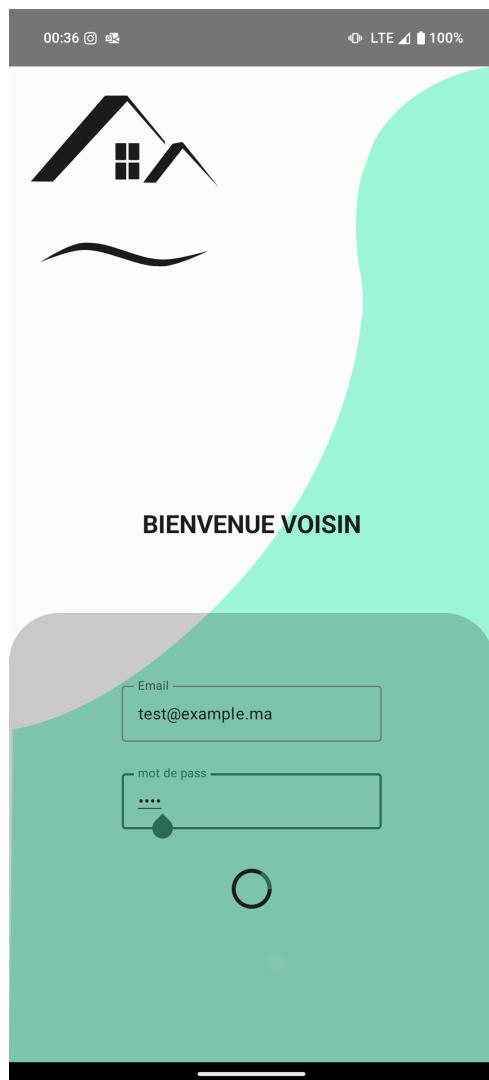


FIGURE 4.19 – Animation lors de connexion



FIGURE 4.20 – Exemple d'erreur 1

Exemple d'erreur de l'interface de connexion à gauche (mail malformé). À droit, la même interface reportant une erreur inconnue.



FIGURE 4.21 – Exemple d'erreur 2



FIGURE 4.22 – Exemple d'erreur 3

4.3 Sprint 2

4.3.1 Spécifications fonctionnelles

4.3.1.1 Consulter la situation des mois

Cette fonctionnalité permet à l'utilisateur de visualiser la liste des mois disponibles, avec les données associées à chaque mois telles que les revenus, les dépenses et le solde. Ces informations peuvent aider l'utilisateur à suivre les finances mois par mois et avoir une vue d'ensemble des finances sur une période donnée.

4.3.1.2 Consulter les opérations de chaque mois

Cette fonctionnalité offre à l'utilisateur la possibilité de consulter en détail les revenus et les dépenses par mois. Une liste des opérations serait affiché en précisant la date de l'opération et le montant de l'opération, le cotiseur s'il s'agit d'une cotisation ou le type de dépense s'il s'agit d'une dépense.

4.3.2 Sprint Backlog

GitHub ID	Sprint Backlog	Acteur	Priorité
#23	En tant qu'utilisateur, je veux voir les revenus, les dépenses, et le solde de chaque mois afin de mieux saisir la situation	Utilisateur	ELEVÉE
#24	En tant qu'utilisateur je veux voir en détail les dépenses et les revenus par mois afin de mieux saisir la situation de chaque mois,	Utilisateur	ELEVÉE

TABLE 4.2 – Backlog du 2eme sprint

4.3.3 Conception

Dans le deuxième Sprint, nous avons décidé de nous attaquer aux backlogs liés à la visualisation des données. Les besoins sont les suivants : montrer la situation globale et présenter une situation spécifique pour chaque mois.

4.3.3.1 Diagramme de cas d'utilisation

Nous allons conserver nos principaux acteurs comme lors du dernier sprint, mais cette fois, les deux acteurs auront la possibilité de :

1. Voir la situation globale ;
2. Voir les opérations pour un mois spécifique.

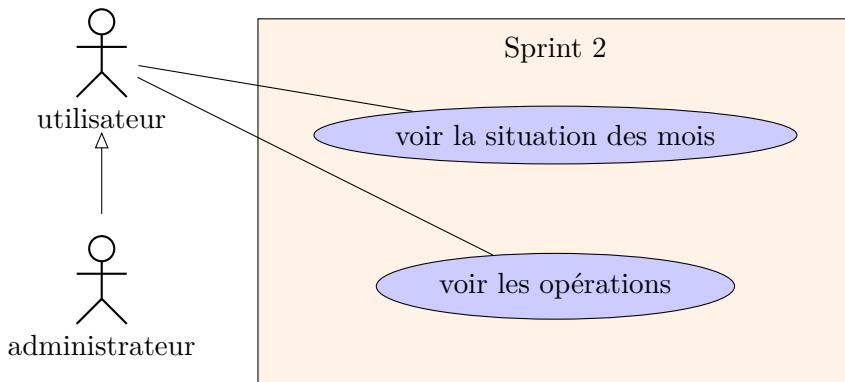


FIGURE 4.23 – Le diagramme de cas d'utilisation pour Sprint 2

4.3.3.2 Diagramme de classe

Suivant le même principe du modèle MVVM, nous avons créé deux vues : une pour afficher la liste des mois à l'aide d'un LazyColumn et une autre vue pour l'affichage des opérations d'un mois sélectionné, qui utilisent également un LazyColumn.

En ce qui concerne les modèles, nous avons créé 3 classes :

1. Month : représente les données d'un mois ;
2. SpendType : représente un type de dépense ;
3. Opération : qui représente une opération (soit une contribution soit une dépense).

Normalement, nous devons avoir deux classes qui héritent de l'opération et chacune représente un type spécifique d'opération, mais étant donné que Firestore utilise une base de données NoSQL et afin d'optimiser les requêtes de la base de données, nous avons décidé d'éviter l'héritage (nous avons trouvé dans nos expériences que Firestore ne gère pas bien l'héritage).

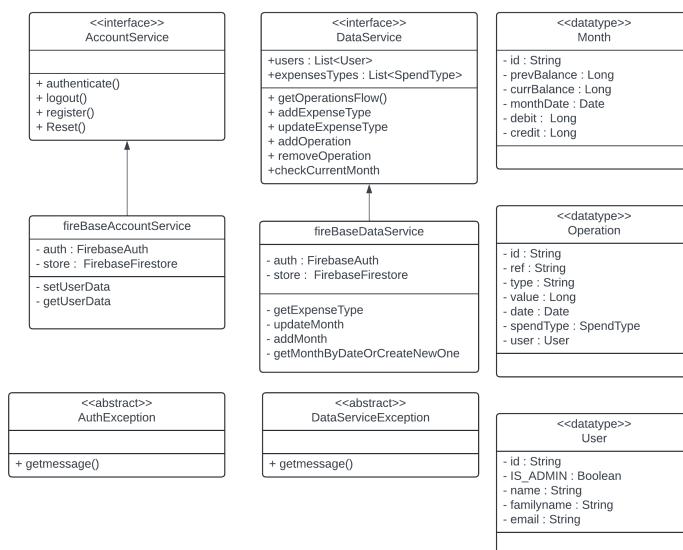


FIGURE 4.24 – Diagramme de classe métier pour Sprint 2

L'interface DataService est destinée à être implémenté par tous les types de fournisseurs de services, dans notre cas la classe FireBaseDataService l'implémente.

Pour le ViewModel, nous avons créé une classe appelée MonthViewModel. Au début, la vue demande à MonthViewModel un Flow<List<Month>>, que le ViewModel récupère à partir du FireBaseDataService. Ensuite, ce Flow est utilisé pour afficher les données de manière réactive (les changements dans la base de données Firestore sont reflétés directement et instantanément dans l'interface utilisateur).

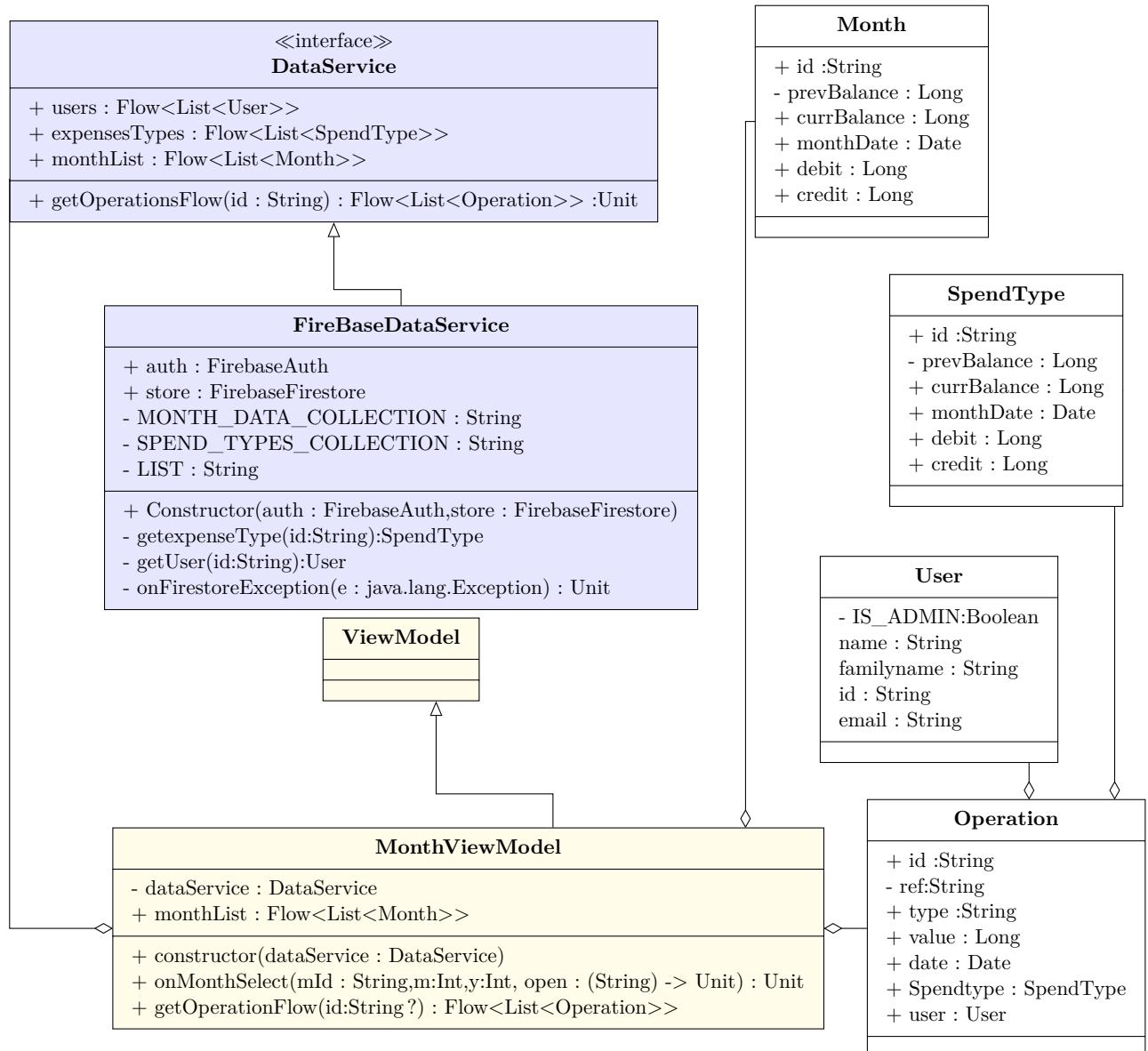


FIGURE 4.25 – Le diagramme de Class pour Sprint 2

4.3.4 Réalisation

4.3.4.1 Les interfaces

L'interface démontrant la liste des mois à gauche, la liste des opérations effectuées en mois de janvier à droite.

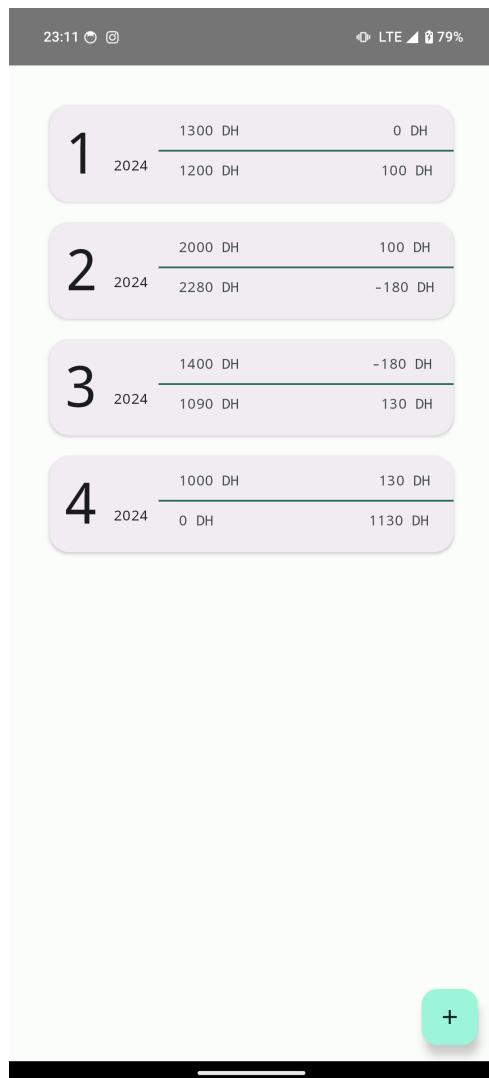


FIGURE 4.26 – Écran de la liste des mois



FIGURE 4.27 – Écran de la liste des opérations de janvier 2024

4.4 Sprint 3

4.4.1 Spécifications fonctionnelles

4.4.1.1 Gérer les cotisations

Dans cette fonctionnalité, nous devons fournir à l'administrateur du syndicat les outils nécessaires pour ajouter ou supprimer des contributions. Chaque contribution crée doit inclure le nom de contributeur et la date à laquelle elle a eu lieu.

4.4.1.2 Gérer les dépenses

Pour les dépenses, la même histoire s'applique avec une seule exception : la dépense doit inclure un type de dépense au lieu d'un nom d'utilisateur.

De plus, l'application doit permettre à l'administrateur d'ajouter de nouveaux types de dépenses ou de changer le nom des autres.

4.4.2 Sprint Backlog

GitHub ID	Sprint Backlog	Acteur	Priorité
#18	En tant qu'Administrateur, je veux avoir la possibilité d'ajouter un type de dépenses à la liste des dépenses prédefinies dans l'application afin de mieux catégoriser les dépenses	Administrateur	ELEVÉE
#20	En tant qu'utilisateur je veux avoir la possibilité d'ajouter une dépense à la liste des dépenses en spécifiant le type, le montant, et la date (qui doit être le jour même par défaut) afin d'enregistrer les dépenses mois.	Administrateur	ELEVÉE
#19	En tant qu'utilisateur je veux avoir la possibilité de modifier un type de dépenses dans la liste des dépenses prédefinies dans l'application afin de mieux catégoriser les dépenses.	Administrateur	ELEVÉE
#21	En tant qu'utilisateur je veux avoir la possibilité d'ajouter une contribution parmi celles déjà enregistrées dans l'application afin de gérer la situation.	Administrateur	ELEVÉE
#20	En tant qu'utilisateur je veux avoir la possibilité de supprimer une dépense de la liste des dépenses afin d'éliminer les enregistrements incorrects.	Administrateur	MOYEN

TABLE 4.3 – Backlog du 3eme sprint

4.4.3 Conception

Dans le troisième sprint, nous avons abordé le reste des backlogs produits parce qu'ils se concentrent tous sur la gestion des opérations.

Nous avons dû donner à l'administrateur la possibilité de créer ou de supprimer de nouvelles contributions, d'ajouter et de modifier des types de dépenses, ainsi que de créer ou de supprimer des dépenses.

4.4.3.1 Diagramme de cas d'utilisation

En analysant les backlogs de sprint, nous constatons qu'un seul acteur est impliqué, à savoir l'administrateur. Ce dernier est responsable des actions suivantes : l'ajout ou la suppression d'une dépense, l'ajout ou la modification d'un type de dépense, ainsi que l'ajout ou la suppression d'une cotisation.

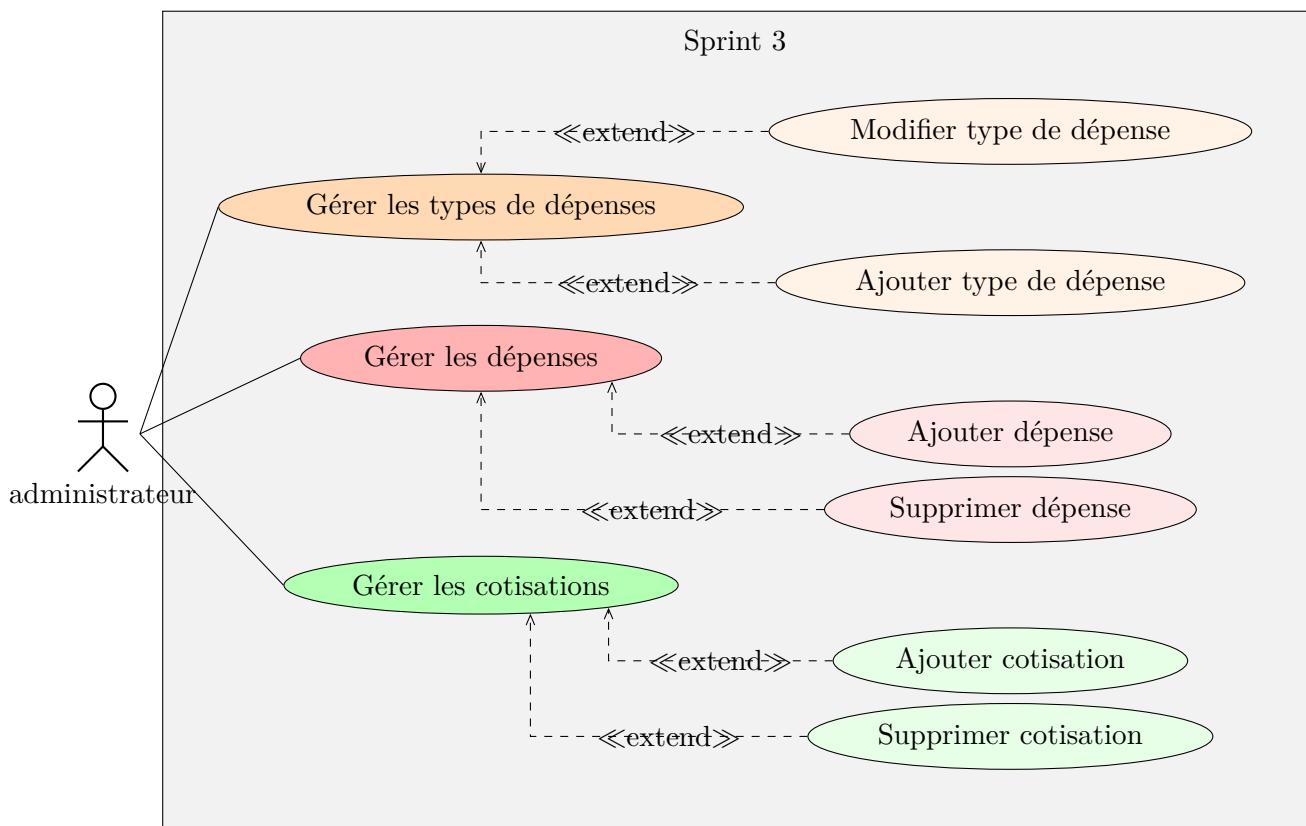


FIGURE 4.28 – Le diagramme de Class pour Sprint 3

4.4.3.2 Diagramme de classe

En suivant le même principe, nous avons créé deux vues où l'administrateur peut ajouter des contributions et des dépenses. Ces deux vues utilisent des UiStates (ContributionUiState et ExpenseUiState) pour obtenir les données à afficher sur l'interface utilisateur.

Pour les modèles, nous avons utilisé les mêmes modèles créés lors du deuxième sprint.

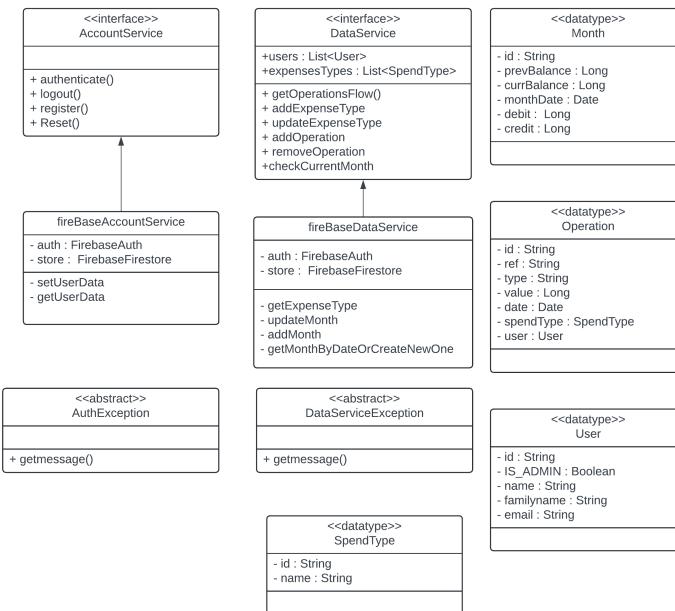


FIGURE 4.29 – Diagramme de classe métier pour Sprint 3

Nous avons également chargé l'interface DataService et la classe FireBaseDataService avec les fonctions nécessaires pour atteindre les objectifs de ce sprint. Nous avons ajouté :

1. addExpenseType : nous permet d'ajouter un nouveau type de dépense
2. updateExpenseType : met à jour le nom d'un type de dépense existant
3. addOperation : pour les contributions ou les dépenses, cette fonction nous permet d'ajouter une nouvelle opération à la base de données et de mettre à jour le mois concerné en conséquence (mise à jour du solde créditeur ou débiteur)
4. removeOperation : également pour les contributions et les dépenses, cette fonction supprime l'enregistrement de l'opération concernée et met à jour le mois concerné en conséquence.

Pour ces deux vues, nous avons utilisé OperationViewModel pour gérer la logique derrière les vues implémentées. Il contient toutes les fonctions différentes appelées par l'interface utilisateur et dispose des fonctions nécessaires pour mettre à jour les vues en fonction des réponses du service ou des interactions de l'utilisateur. Il convient de noter que les opérations liées à la suppression des opérations (qu'il s'agisse de cotisations ou de dépenses) sont gérées via le MonthViewModel, car nous avons opté pour la gestuelle de glissement pour supprimer les opérations.

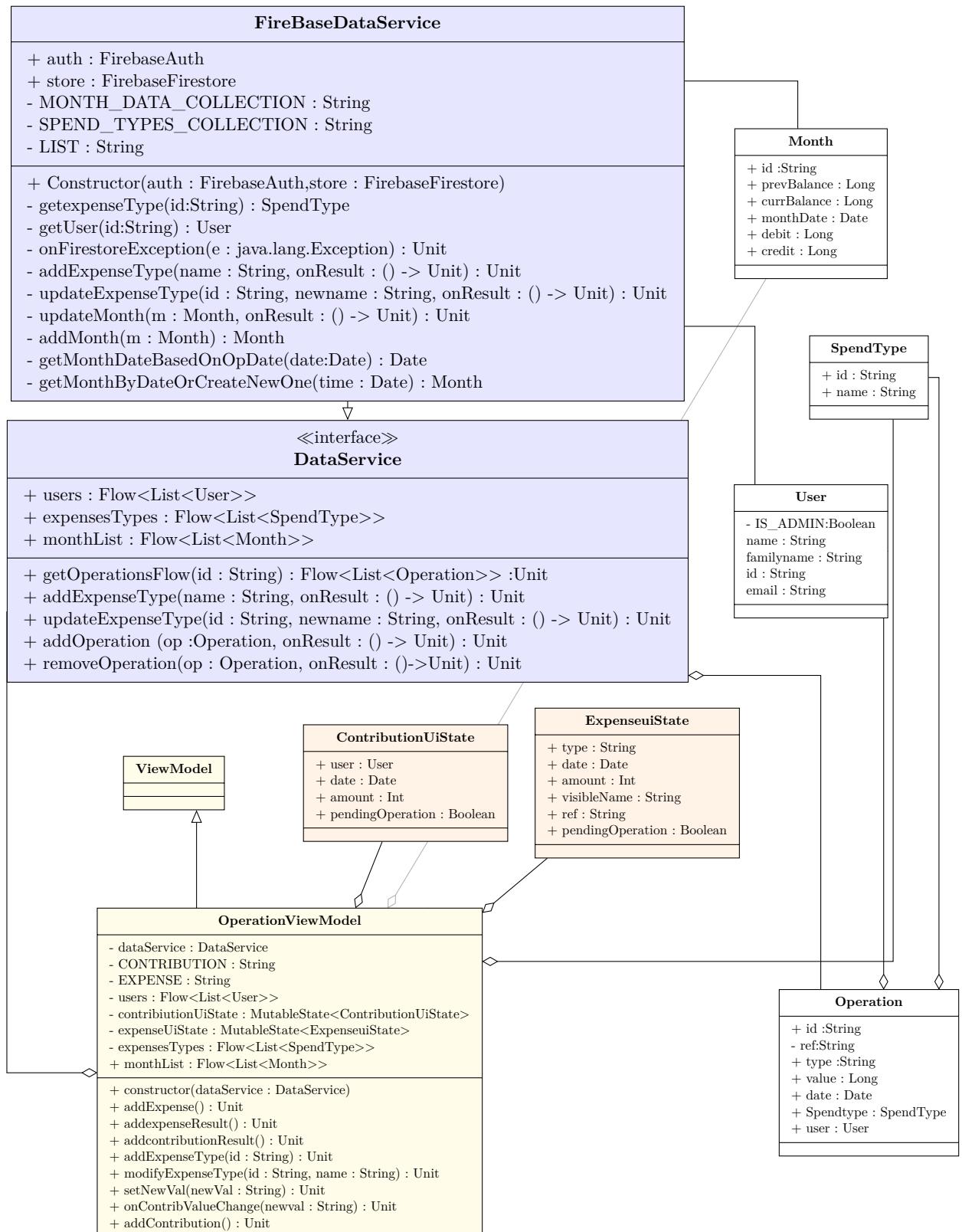


FIGURE 4.30 – Le diagramme de Class pour Sprint 3

4.4.4 Réalisation

4.4.4.1 L'accès au menu latéral

Le menu latéral (pour l'administrateur) à gauche, a droit un message d'erreur en cas de suppression d'une opération non effectué au mois courant.

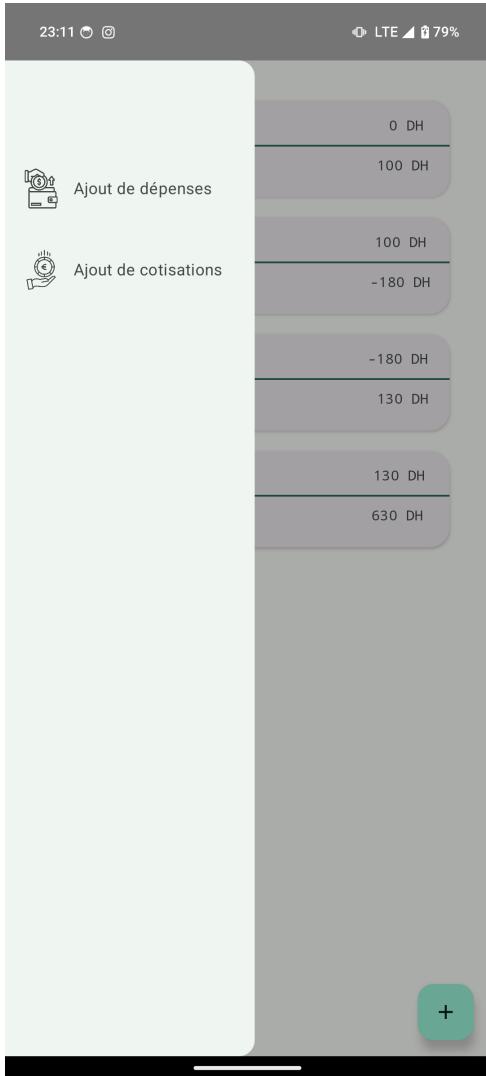


FIGURE 4.31 – Menu latéral accessible par l'administrateur



FIGURE 4.32 – Message d'erreur si l'administrateur essaye d'ajouter ou supprimer une opération avec une date ancienne

4.4.4.2 Ajout de dépense

L'interface d'ajout des dépenses et modification des types des dépenses.



FIGURE 4.33 – Exemple d’interface pour ajouter des dépenses



FIGURE 4.34 – Exemple d’opération en cours (ajout de dépense)

Interface pour sélectionner la date (calendrier) à gauche, message de réussir d'ajout d'opération à droit.

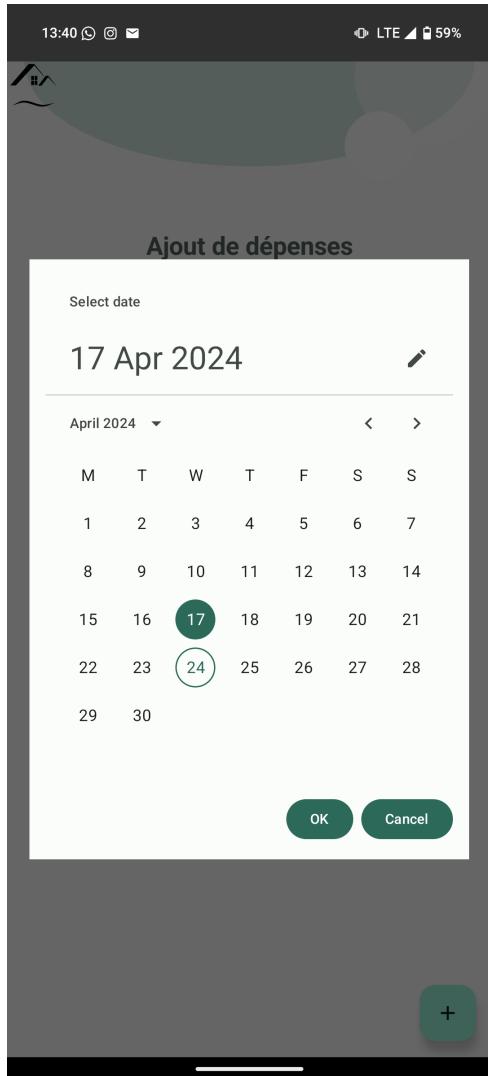


FIGURE 4.35 – Calendrier pour sélectionner la date de dépense

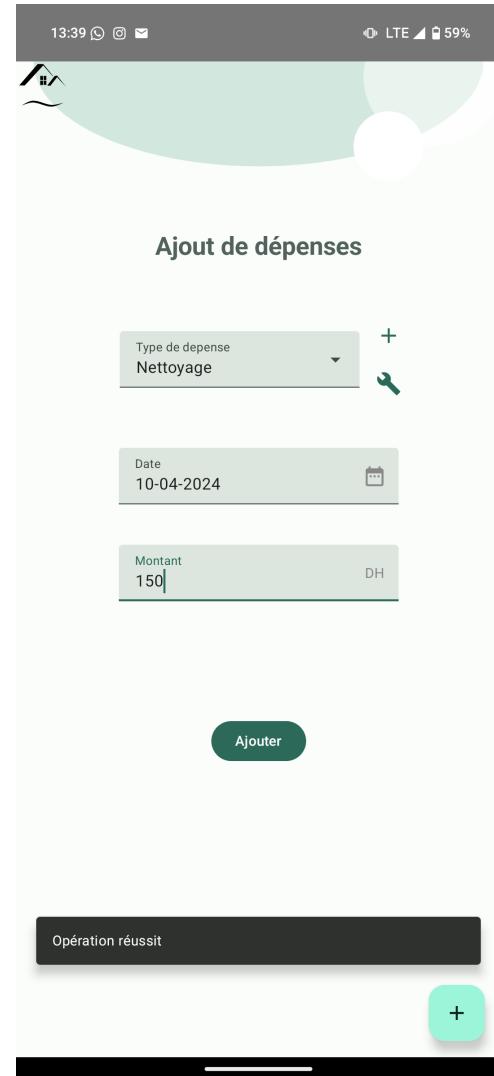


FIGURE 4.36 – Exemple de message de réussit d'ajout de dépenses

4.4.4.3 Ajout et modification de type de dépense

Interface d'ajout d'un nouveau type de dépense et de modification d'un type de dépense

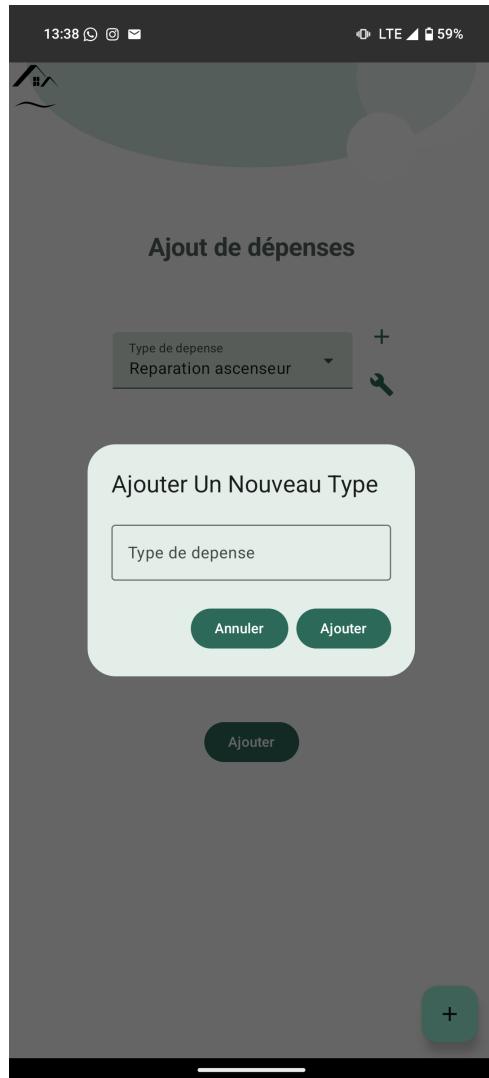


FIGURE 4.37 – Interface d'ajout d'un nouveau type de dépense à la liste des types des dépenses



FIGURE 4.38 – Interface de modification d'un type de dépense

4.4.4.4 Ajout de cotisation

Interface d'ajout de cotisation.



FIGURE 4.39 – Exemple d'interface pour ajouter des cotisations

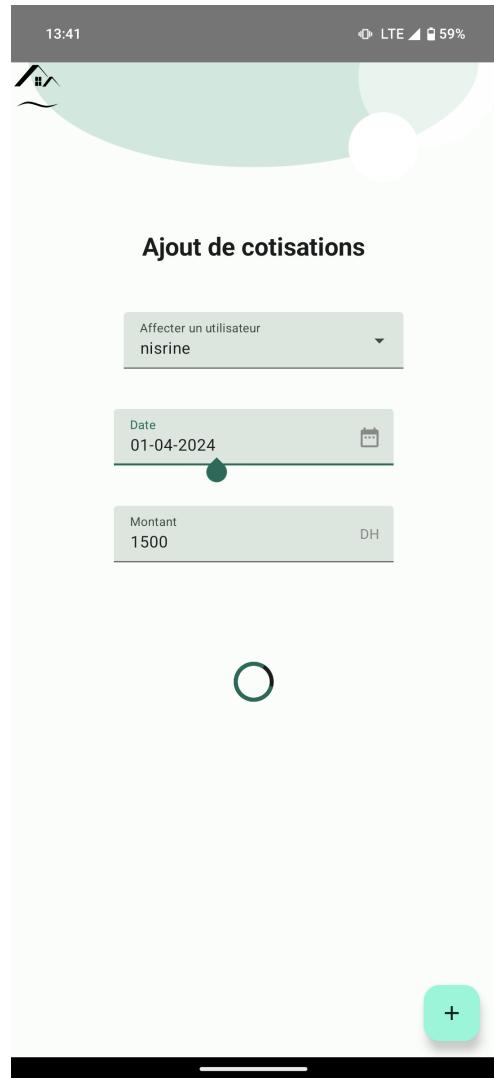


FIGURE 4.40 – Exemple d'opération en cours (ajout de cotisation)

Interface montrant le message de réussir d'ajout de cotisation.

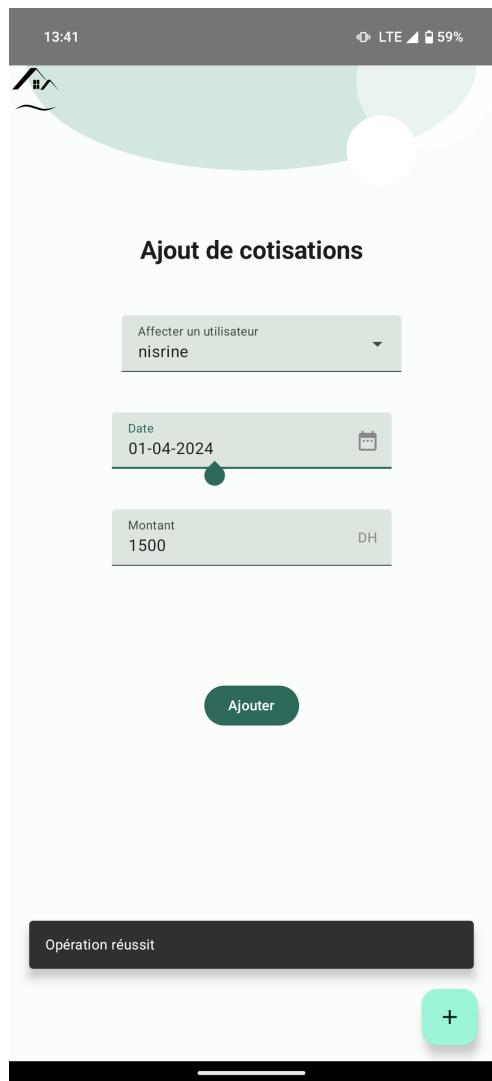


FIGURE 4.41 – Exemple de message de réussit d'ajout de cotisation

4.4.4.5 Suppression des opérations

Interface d'opération de mois 2 montrant la possibilité de glissement pour suppression.



FIGURE 4.42 – Exemple de suppression d'une opération par l'administrateur



FIGURE 4.43 – Message de réussite

Conclusion

En guise de conclusion, nous avons développé une application mobile dédiée à la gestion des syndics de copropriété, visant à améliorer les processus existants et à accroître la transparence des opérations. Pour ce faire, nous avons adopté une approche Agile, en particulier la méthodologie Scrum, pour une gestion itérative du projet. En matière de développement, nous avons suivi les normes les plus rigoureuses du génie logiciel et tiré parti de technologies de pointe telles qu'Android Jetpack Compose pour l'interface utilisateur, Gradle pour la construction, Firebase pour les services backend. Ce faisant, nous avons abouti à une application mobile opérationnelle, dotée d'une interface utilisateur fluide et ergonomique. Parallèlement, ce projet nous a permis de raffiner notre méthodologie de travail et de développer notre esprit d'équipe.

Bibliographie

- [1] Scrum ALLIANCE. *about scrum*. URL : <https://www.scrumalliance.org/about-scrum>.
- [2] Kent BECK et al. *Manifesto for Agile Software Development*. 2001. URL : <http://www.agilemanifesto.org/>.
- [3] *Build Documentation / Firebase Documentation*. <https://firebase.google.com/docs/build>. [Accessed 21-04-2024].
- [4] *Comprendre GitHub Actions - GitHub Enterprise Cloud Docs*. <https://docs.github.com/fr/enterprise-cloud@latest/actions/learn-github-actions/understanding-github-actions>. [Accessed 21-04-2024].
- [5] GITHUB. *À propos de GitHub et Git - Documentation GitHub*. <https://docs.github.com/fr/get-started/start-your-journey/about-github-and-git>. [Accessed 21-04-2024].
- [6] *Gradle Wikipédia — fr.wikipedia.org*. <https://fr.wikipedia.org/wiki/Gradle>. [Accessed 24-04-2024].
- [7] *Guide de l'architecture des applications / Android Developers — developer.android.com*. <https://developer.android.com/topic/architecture?hl=fr>. [Accessed 21-04-2024].
- [8] *Injection de dépendances avec Hilt / Android Developers*. <https://developer.android.com/training/dependency-injection/hilt-android?hl=fr>. [Accessed 21-04-2024].
- [9] *Principes de base de Jetpack Compose à/à Android Developers — developer.android.com*. <https://developer.android.com/codelabs/jetpack-compose-basics?hl=fr>. [Accessed 21-04-2024].
- [10] Arkance SYSTEMS. *Qu'est ce que le pattern MVVM ? — arkance-systems.fr*. <https://www.arkance-systems.fr/blog/pattern-mvvm/>. [Accessed 21-04-2024].
- [11] WIKIPEDIA. *Android — Wikipedia, The Free Encyclopedia*. <http://fr.wikipedia.org/w/index.php?title=Android&oldid=214052893>. [Online ; accessed 11-April-2024]. 2024.
- [12] WIKIPEDIA. *Kotlin (langage) — Wikipedia, The Free Encyclopedia*. [http://fr.wikipedia.org/w/index.php?title=Kotlin%20\(langage\)&oldid=212445709](http://fr.wikipedia.org/w/index.php?title=Kotlin%20(langage)&oldid=212445709). [Online ; accessed 11-April-2024]. 2024.

Annexes

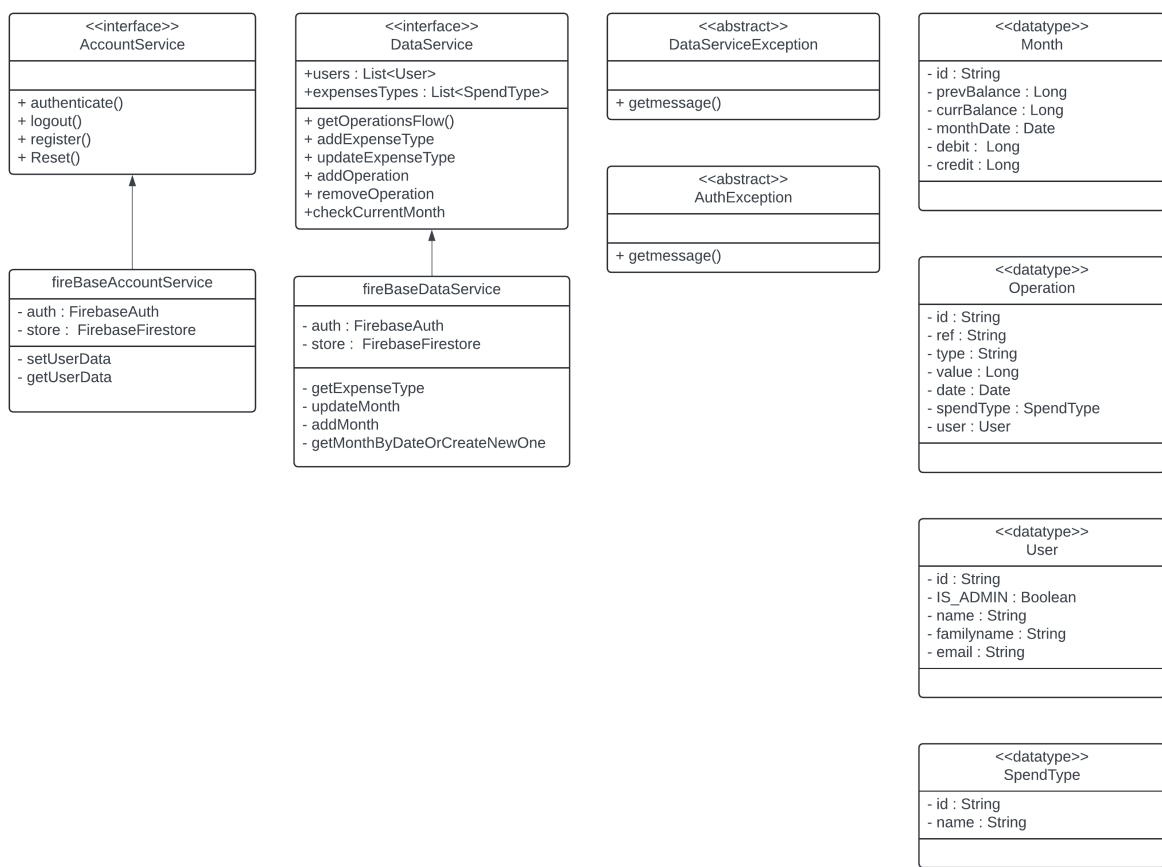


FIGURE A.1 – Diagramme de classe métier

list des products backlog

ANNEXES

GITHUB ISSUE ID	BACKLOG TASK & ID	PRIORITY	ASSIGNED TO	STATUS	SPRINT	SPRINT 1																
						17/03/24	18/03/24	19/03/24	20/03/24	21/03/24	22/03/24	23/03/24	24/03/24	25/03/24	26/03/24	27/03/24	28/03/24	29/03/24	30/03/24			
#2	As an administrator, I want to be able to authenticate to the app using my login and password in order to use the app	HIGH		DONE	1	17%	20%	33%	56%	61%	78%	89%	100%	100%	100%	100%	100%	100%	100%	100%		
	create LOGIN UI		BAKHOUCH NISRINE	DONE		50%	80%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	create model		EL BOUZYANI ANAS	DONE		0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	create the viewModel		EL BOUZYANI ANAS	DONE		0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	create service		EL BOUZYANI ANAS	DONE		0%	0%	0%	50%	50%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	link firebase project with the application		EL BOUZYANI ANAS	DONE		100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	implement firebase auth logic		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	implement firebase firestore logic		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	test		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	create a theme / color template for the app		BAKHOUCH NISRINE	DONE		0%	0%	0%	50%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
#3	As a user, I want to be able to authenticate to the app using my login and password in order to use the app	HIGH		DONE	1	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	adapt viewModel		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	adapt Service		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	test		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
#4	As a user, I want to be able to register my new account on the app in order to have access	HIGH		DONE	1	0%	0%	0%	0%	0%	0%	0%	19%	39%	66%	86%	100%	100%	100%	100%	100%	
	create REGISTER UI		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	0%	50%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	adapt viewModel for register		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	80%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	adapt Service for register		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	0%	50%	100%	100%	100%	100%	100%	100%	100%	100%
	add register UState		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%
	implement navigation logic		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	20%	60%	100%	100%	100%	100%	100%	100%
	adapt firebase firestore logic		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%
	test		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%
#5	As a user, I want to be able to reset my password in order to recover my account to regain access to my lost account	HIGH		DONE	1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	50%	100%	100%	100%	
	create RESET UI		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%
	adapt viewModel for reset		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%
	adapt Service for reset		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%
	add register UState		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%
	add reset to navgraph		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%
	test		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%
#6	As an administrator, I want to be able to deactivate a specific user's account in order to deny access to the user	LOW		CANCELED	1	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	

FIGURE A.2 – Product Backlog de sprint 1

GITHUB ISSUE ID	BACKLOG TASK & ID	PRIORITY	ASSIGNED TO	STATUS	SPRINT	SPRINT 2																
						0%	17%	25%	50%	67%	83%	88%	92%	100%	100%	100%	100%	100%	100%	100%	100%	
#23	As a user, I want to see each month's income, expenses, and balance to better understand the situation	LOW		DONE	2	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	set month UI		BAKHOUCH NISRINE	DONE		0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	implement model		EL BOUZYANI ANAS	DONE		0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	Understand Kotlin Flows and implement them		EL BOUZYANI ANAS	DONE		0%	0%	50%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	set parameters on NavGraph and onClickCallBack on viewmodel		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	implement needed functions on services		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	debug and test		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	25%	50%	100%	100%	100%	100%	100%	100%	100%	100%	100%
#24	As a user, I want to see monthly expenses and income in detail to better understand each month's situation	LOW		DONE	2	0%	0%	0%	0%	0%	40%	40%	40%	80%	90%	100%	100%	100%	100%	100%	100%	
	set interface for both admin and user		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	add suitable listState		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	update the viewModel to handle ui requests		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	
	add needed function on the service		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	
	test and code review		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	50%	100%	100%	100%	100%	100%	

FIGURE A.3 – Product Backlog de sprint 2

GITHUB ISSUE ID	BACKLOG TASK & ID	PRIORITY	ASSIGNED TO	STATUS	SPRINT	SPRINT 3															
						14/04/24	15/04/24	16/04/24	17/04/24	18/04/24	19/04/24	20/04/24	21/04/24	22/04/24	23/04/24	24/04/24	25/04/24	26/04/24	27/04/24		
#18	As an administrator, I want to have the ability to add an expense type to the list of predefined expenses in the app to better categorize expenses	LOW	BAKHOUCH NISRINE	DONE	3	13%	25%	25%	75%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%		
	adapt firebase firestore logic		BAKHOUCH NISRINE	DONE		50%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	implement logic to add an expense type		EL BOUZYANI ANAS	DONE		0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	make suitable UI		BAKHOUCH NISRINE	DONE		0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	test		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
#19	As an administrator, I want to have the ability to edit an expense type from the list of predefined expenses in the app to better categorize expenses	LOW	BAKHOUCH NISRINE	DONE	3	0%	0%	0%	0%	0%	67%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
	adapt service for renaming expense type		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	implement UI		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	test		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
#21	As an administrator, I want to be able to add user contribution of the month in order to better report the situation	LOW	BAKHOUCH NISRINE	DONE	3	0%	0%	0%	0%	0%	10%	60%	100%	100%	100%	100%	100%	100%	100%	100%	
	implement side navigation panel		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	50%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	adapt navigation graph		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	add function to service		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	add suitable UI		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
	test		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
#22	As an administrator, I want the ability to remove an expense from the expense list to remove incorrect records.	LOW	BAKHOUCH NISRINE	DONE	3	0%	0%	0%	0%	0%	0%	0%	0%	0%	50%	100%	100%	100%	100%	100%	
	implement swipe to delete logic		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	
	adapt service		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%	
	add logic to recalculate month details		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%
	test		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%
	As an administrator, I want to be able to remove a user contribution of the month in order to fix errors	LOW	BAKHOUCH NISRINE	DONE	3	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	50%	100%	100%	100%	
	adapt service		EL BOUZYANI ANAS	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%
	test		BAKHOUCH NISRINE	DONE		0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	100%	100%	100%	100%	100%

FIGURE A.4 – Product Backlog de sprint 3