



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

FILIERE : GENIE LOGICIEL

application Web/Mobile de suivi de Budget

Realise par :

BAKHOUCHE Nisrine
EL BOUZIYANI Anas

Sous la direction de :

Pr EL HAMLAOUI Mahmoud

Department Informatique
ENSIAS - UM5
Année Académique 2023/2024

Remerciments

Au terme de ce projet nous exprimons notre profonde gratitude envers notre encadreur Pr.EL HAMLAOUI Mahmoud pour sa vigilance constante et ses remarques constructives qui ont enrichi notre travail. Nous le remercions chaleureusement pour sa disponibilité et son précieux accompagnement, et nous avons été honorés de bénéficier de ses directives et ses conseils éclairés tout au long du projet.

Resume

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 d'optimiser les processus existants 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 développement nous avons utilise les meilleures pratiques de developpement logiciel et les technologies modernes tells que Android Jetpack Compose pour le développement de l'interface utilisateur, Gradle pour la gestion de build, Firebase pour les services backend et UML pour la conception. Le resultat final est une application mobile fonctionnelle offrant une interface utilisateur intuitive et conviviale.

Abstract

TO BE IMPLEMENTED

Table des matières

Remerciments	3
Resume	4
Abstract	5
Table des figures	7
Liste des tableaux	8
1 Introduction générale	9
2 Contexte générale de projet	10
2.1 Problematique	10
2.2 objectif	10
2.3 l'analyse des besoins	10
2.3.1 Besions fonctionnels	11
2.3.2 Besoins non fonctionnels	11
3 Spécification techniques	12
3.1 méthodologie	12
3.1.1 Agile manifesto	12
3.1.2 Scrum/Agile	12
3.1.2.1 les intervennats	12
3.1.2.2 la conduite du Scrum Agile	13
3.2 technologies	14
3.2.1 Android	14
3.2.2 Kotlin	14
3.2.3 les bonnes pratiques	14
3.2.3.1 conposition basique d'un logiciel android	15
3.2.4 Regles a suivre :	15
3.2.4.1 modèle MVVM	15
3.2.5 GitHub	16
3.2.5.1 GitHub Issues	16
3.2.5.2 GitHub projects	17
3.2.5.3 GitHub Actions	17
3.2.6 Android JetPack Compose	17
3.2.7 Gradle	17
3.2.8 Firebase	17

3.2.9	Firebase FirebaseAuth	17
3.2.10	Firebase Firestore	17
3.2.11	daggerHilt (injecteur de dependances)	18
4	Deroulement des Sprints	19
4.1	Sprint 0 : Mise en Oeuvre	19
4.1.1	identification des acteurs	19
4.1.2	userStories (Backlog du produit)	19
4.1.3	prototypage des interfaces	20
4.1.3.1	l'authentification	20
4.1.3.2	ajout de cotisation et de depenses	21
4.1.3.3	l'affichage des situations	22
4.2	Sprint 1	22
4.2.1	specification fonctionnel	22
4.2.1.1	S'inscrire	22
4.2.1.2	Se connecter	22
4.2.1.3	Réinitialisation le mot de passe	22
4.2.2	Sprint Backlog	23
4.2.3	conception	23
4.2.3.1	diagramme de cas d'utilisation	23
4.2.3.2	diagramme de classe	24
4.2.4	Realisation	25
4.2.4.1	interface de connexion	25
4.2.4.2	interface d'inscription et de renitialisation de mot de passe	27
4.3	Sprint 2	30
4.3.1	specification fonctionnel	30
4.3.1.1	consulter la situation des mois	30
4.3.1.2	consulter les operations de chaque mois	30
4.3.2	Sprint Backlog	30
4.3.3	conception	30
4.3.3.1	diagramme de cas d'utilisation	30
4.3.3.2	diagramme de classe	31
4.3.4	Realisation	33
4.3.4.1	les interfaces	33
4.4	Sprint 3	34
4.4.1	specification fonctionnel	34
4.4.2	Sprint Backlog	34
4.4.3	conception	34
4.4.3.1	diagramme de cas d'utilisation	34
4.4.3.2	diagramme de classe	34
4.4.4	Realisation	34
4.4.4.1	interface pour gérer les opération	34
4.4.4.1.1	ajouter type de dépense	34
4.4.4.1.2	modifier type de dépense	34
4.4.4.1.3	ajouter de dépense	34
4.4.4.1.4	ajouter cotisation	34
4.4.4.1.5	supprimer operation	34

A annexes

37

Table des figures

3.1	logo de scrum Agile	12
3.2	logo d'android	14
3.3	logo de Kotlin	14
3.4	Schéma d'une architecture d'application android	15
3.5	architecture MVVM dans android avec FireBase	16
3.6	logo de GitHub	16
4.1	prototype de l'ecran d'inscription	20
4.2	prototype de l'ecran de connexion	20
4.3	prototype de l'ecran de réinitialisation de mot de passe	20
4.4	prototype de menu latérale	21
4.5	prototype de l'ecran d'ajout des contributions	21
4.6	prototype de l'ecran d'ajout des dépenses	21
4.7	prototype d'affichage pour l'administrateur	22
4.8	prototype d'affichage pour l'utilisateur	22
4.9	prototype d'affichage des opérations	22
4.10	le diagramme de cas d'utilisation pour Sprint 1	24
4.11	le diagramme de Class pour Sprint 1	25
4.12	ecran de connexion (light mode)	26
4.13	ecran de connexion (dark mode)	26
4.14	ecran d'inscription	27
4.15	ecran de reinitialisation de mot de passe	27
4.16	animation lors de connexion	28
4.17	exemple d'erreur 1	28
4.18	exemple d'erreur 2	29
4.19	exemple d'erreur 3	29
4.20	le diagramme de cas d'utilisation pour Sprint 2	31
4.21	le diagramme de Class pour Sprint 2	32
4.22	ecran de la liste des mois	33
4.23	ecran de la liste des operations de janvier 2024	33
4.24	le diagramme de Class pour Sprint 2	36

Liste des tableaux

Introduction générale

La gestion efficace des syndics représente en enjeu majeur dans le secteur immobilier, Néanmoins, les approches traditionnelles reposant sur des processus manuels et papier 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 détaillerons notre démarche méthodologique, les outils et technologies utilisés, 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 générale de projet

2.1 Problematique

La gestion des copropriétés constitue un défi de taille pour de nombreux syndicats, requérant 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 offre 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 aux 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. l'analyse des besoins

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 un 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 privilèges 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 oeuvre une architecture qui facilite la maintenance de l'application.

Spécification 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 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

" [beck2001agile]

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 » 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.



FIGURE 3.1 – logo de scrum Agile

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. **Développeurs** : Professionnels de l'équipe Scrum qui travaillent ensemble pour créer n'importe quel aspect du produit. Ils créent l'incrément(s) de produit pendant le sprint. Les personnes possédant les compétences nécessaires à la construction du produit assument la responsabilité de développeur. Selon la nature du produit, les compétences seront différentes.
2. **Propriétaire du produit** : Le propriétaire du produit développe et communique l'objectif du produit, possède le backlog du produit et veille à ce que l'équipe s'attaque toujours au travail de la plus haute valeur. Il équilibre également les besoins des parties prenantes, des clients et de l'équipe. Il connaît et comprend le domaine ainsi que le marché de ces produits, et il est passionné par la fourniture de résultats que les clients et les utilisateurs veulent et dont ils ont besoin.
3. **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'incarnation 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 ainsi que des individus qui la composent.

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 (et parfois les deux).

1. **Le sprint** : Au cur de Scrum, une période limitée dans le temps (moins d'un mois et fréquemment de 1 à 2 semaines) pendant laquelle un ou plusieurs incréments 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. La planification doit être limitée dans le temps à un maximum de 8 heures pour un sprint d'un mois, avec une durée plus courte pour les sprints plus courts. Sur la base de l'objectif du sprint et des prévisions, un plan initial est également créé. L'équipe Scrum peut inviter d'autres personnes à la planification du sprint pour obtenir des conseils ou des contributions sur le travail pertinent.
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 à 15 minutes. Le scrum quotidien n'est pas la seule opportunité pour l'équipe d'adapter ses plans ; elle communique souvent sur les pivots nécessaires en dehors de cet événement. Lors du scrum quotidien, l'équipe peut synchroniser son travail quotidien, 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 et à pivoter si nécessaire.
4. **Revue de sprint** : L'ensemble de l'équipe Scrum inspecte le résultat du sprint avec les parties prenantes et détermine les adaptations futures. Les parties prenantes sont invitées à donner leur avis sur ce que l'équipe Scrum a accompli jusqu'à présent

et sur la direction future du développement du produit. Le backlog du produit est adapté en fonction de ces conversations.

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.

[scrumalliance]

3.2 technologies

3.2.1 Android

Android est un système d'exploitation mobile open source fondé sur le noyau Linux et développé par un consortium d'entreprises, le Open Handset Alliance, 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 clé en main pour les appareils mobiles, smartphones et tablettes tactiles. Cette pile comporte un système d'exploitation (comprenant un noyau Linux), les applications clés 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 [wiki:Android]



FIGURE 3.2 – logo d'android

3.2.2 Kotlin

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



FIGURE 3.3 – logo de Kotlin

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 composition 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.

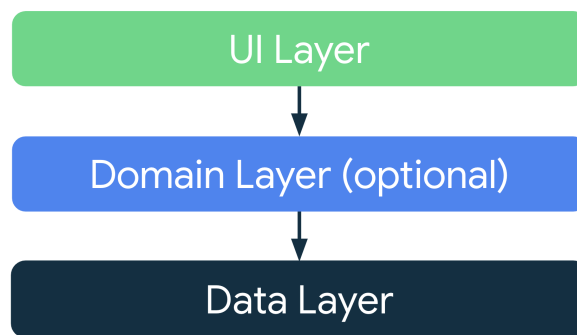


FIGURE 3.4 – Schéma d'une architecture d'application android

source : developer.android.com

3.2.4 Regles a suivre :

selon le site developer.android.com il est fortement recommandé de suivre les regles suivantes :

1. **Ne stockez pas de données dans les composants 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.**

[[androidGuideLarchitecture](#)]

3.2.4.1 modèle MVVM

le modele MVVM (Modele-View-ViewModele) est un Design pattern (Modèle de conception) visant à séparer la logique de présentation d'une application en 3 couches

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.

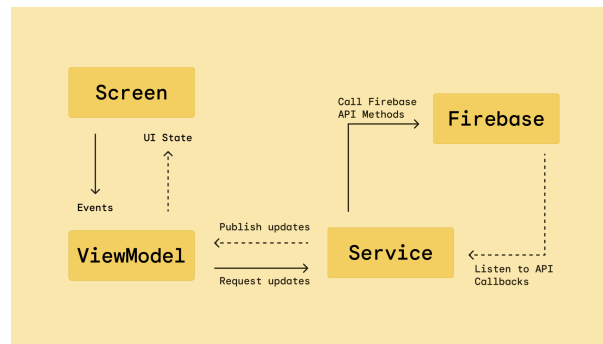


FIGURE 3.5 – architecture MVVM dans android avec FireBase

source : firebase.blog

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.

[arkancesystemsQuestPattern]

3.2.5 GitHub

GitHub est une plateforme basée sur le cloud qui offre la possibilité de stocker, partager et collaborer avec d'autres sur l'écriture de code. En stockant le code dans un "référentiel" sur GitHub, vous pouvez : Suivre et gérer les modifications apportées à votre code au fil du temps. Permettre à d'autres personnes de réviser votre code et de faire des suggestions pour l'améliorer. Collaborer sur un projet partagé sans craindre que vos modifications aient un impact sur le travail de vos collaborateurs avant que vous ne soyez prêt à les intégrer



FIGURE 3.6 – logo de GitHub

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. Intégrer les GitHub Issues dans notre projet qui adopte la méthodologie agile Scrum offre 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. [githubProposGitHub]

3.2.5.2 GitHub projects

GitHub projects propose des fonctionnalités de gestion de projet intégrées qui permettent aux équipes de planifier, organiser et suivre leur travail. Ils offrent une vue visuelle des tâches, des étapes de workflow et des deadlines. Ils peuvent inclure des Issues, des Pull Requests et d'autres éléments. 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 continue et livraison continue (CI/CD) qui vous permet d'automatiser votre pipeline de génération, de test et de déploiement. Vous pouvez créer des workflows qui créent et testent chaque demande de tirage (pull request) adressée à votre dépôt, ou déployer des demandes de tirage fusionnées en production. [[Comprendre GitHub](#)]

3.2.6 Android JetPack Compose

Jetpack Compose est un kit d'outils moderne conçu pour simplifier le développement des interfaces utilisateur. Il allie un modèle de programmation réactif à la concision et à la facilité d'utilisation du langage de programmation Kotlin. Il est entièrement déclaratif.

3.2.7 Gradle

3.2.8 Firebase

Firebase est une plate-forme de développement d'applications et un ensemble d'outils pour l'hébergement, qui permet l'envoi de notifications et de publicités, la remontée des erreurs et des clics effectués dans l'application

3.2.9 Firebase FirebaseAuth

Firebase Authentication fournit des services backend, des SDK faciles à utiliser et des bibliothèques d'interface utilisateur 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.

TO BE IMPLEMENTED

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. Comme Firebase Realtime Database, 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 que vous puissiez créer des applications réactives qui fonctionnent quelle que soit la latence du réseau ou la connectivité Internet.

3.2.11 daggerHilt (injecteur de dépendances)

Hilt est une bibliothèque d'injection de dépendances pour Android qui réduit le code récurrent nécessaire pour injecter manuellement des 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.

Deroulement des Sprints

4.1 Sprint 0 : Mise en Oeuvre

Avant de débiter 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

le sprincipaux acteurs sont :

— **Administrateur (gérant du syndique) :**

1. gérer la recette ;
2. marquer les dépenses ;
3. marque 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 definis 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 a 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 prédéfinies dans l'application afin de mieux catégoriser les dépenses.
- En tant qu'administrateur, je veux avoir la possibilité de modifier un type de dépenses à la liste des dépenses prédéfinies 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 rectifier 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

The registration screen features the 'RESIDENCE RIAD' logo at the top. Below the logo is the text 'WELCOME NEIGHBOR'. The form contains four input fields labeled 'nom', 'prenom', 'email', and 'pass word', followed by a 's'inscrire' button.

FIGURE 4.1 – prototype de l’écran d’inscription

The login screen features the 'RESIDENCE RIAD' logo at the top. Below the logo is the text 'WELCOME NEIGHBOR'. The form contains two input fields labeled 'email' and 'mot de pass', followed by a 'se connecter' button.

FIGURE 4.2 – prototype de l’écran de connexion

The password reset screen features the 'RESIDENCE RIAD' logo at the top. Below the logo is the text 'WELCOME NEIGHBOR'. The form contains one input field labeled 'email', followed by a 'reinitialiser le mot de pass' button.

FIGURE 4.3 – prototype de l’écran de réinitialisation de mot de passe

4.1.3.2 ajout de cotisation et de depenses

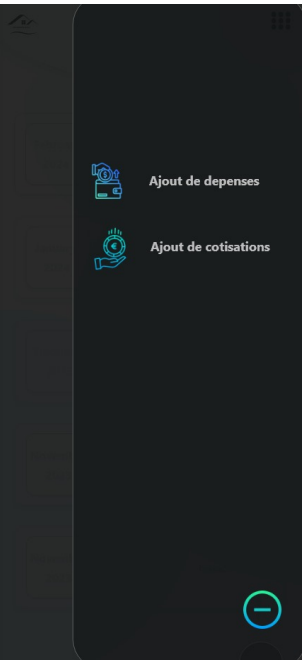


FIGURE 4.4 – proto-
type de menu latérale



FIGURE 4.5 – proto-
type de l'écran d'ajout
des contributions



FIGURE 4.6 – proto-
type de l'écran d'ajout
des dépenses

4.1.3.3 l'affichage des situations

Prototype d'affichage pour l'administrateur. L'interface présente une liste de cinq situations numérotées 1 à 5. Chaque situation est affichée dans une carte blanche avec un en-tête gris. Les cartes 1 à 4 ont un en-tête gris clair, tandis que la carte 5 a un en-tête gris foncé. Chaque carte contient un grand chiffre (1 à 5), une date (2024 pour 1 à 4, 2023 pour 5), et deux champs de texte '0 dh'.

FIGURE 4.7 – proto-
type d'affichage pour
l'administrateur

Prototype d'affichage pour l'utilisateur. L'interface présente une liste de cinq situations numérotées 1 à 5. Chaque situation est affichée dans une carte blanche avec un en-tête gris. Les cartes 1 à 4 ont un en-tête gris clair, tandis que la carte 5 a un en-tête gris foncé. Chaque carte contient un grand chiffre (1 à 5), une date (2024 pour 1 à 4, 2023 pour 5), et deux champs de texte '0 dh'.

FIGURE 4.8 – proto-
type d'affichage pour
l'utilisateur

Prototype d'affichage des opérations. L'interface présente une liste de trois opérations numérotées 1 à 3. Chaque opération est affichée dans une carte blanche avec un en-tête gris. Les cartes 1 et 2 ont un en-tête gris clair, tandis que la carte 3 a un en-tête gris foncé. Chaque carte contient un grand chiffre (1 à 3), une date (2024 pour 1 à 2, 2023 pour 3), et deux champs de texte '0 dh'.

FIGURE 4.9 – proto-
type d'affichage des
opérations

4.2 Sprint 1

4.2.1 specification fonctionnel

4.2.1.1 S'inscrire

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 enregistré dans la base de données.

4.2.1.2 Se connecter

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 le 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 e-mail à l'utilisateur. L'utilisateur 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

Gi- tHub 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	Adminis- trateur	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 a mon compte	Utilisateur	MOYEN

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 les Sprint Backlogs sélectionnés, 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.

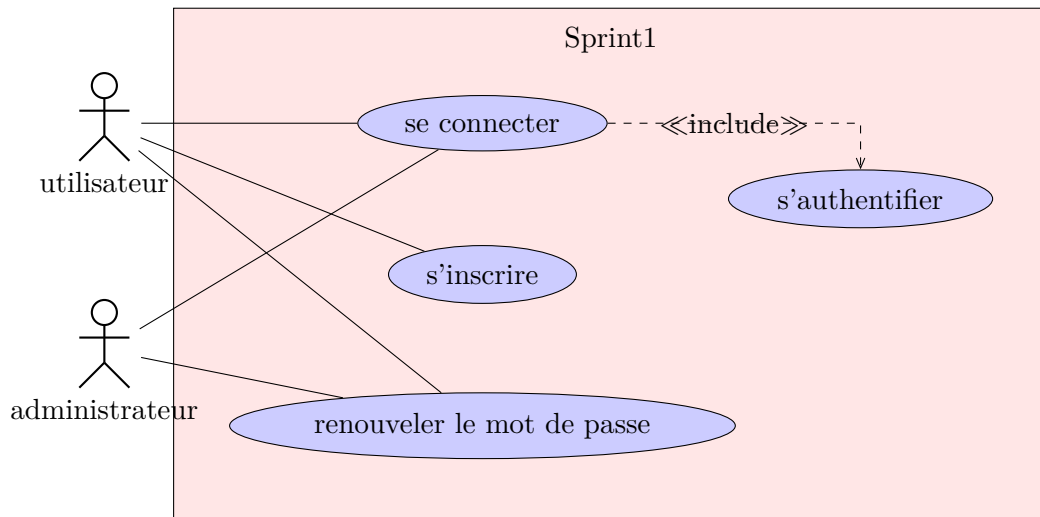


FIGURE 4.10 – le diagramme de cas d'utilisation pour Sprint 1

4.2.3.2 diagramme de classe

basé sur les meilleures pratiques citées ci-dessous, 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 le modèle, 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 de 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 les passe ces parametres et appelle la méthode authenticate de l'instance FireBaseAccountService.

Pour personnaliser les messages d'erreur 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.

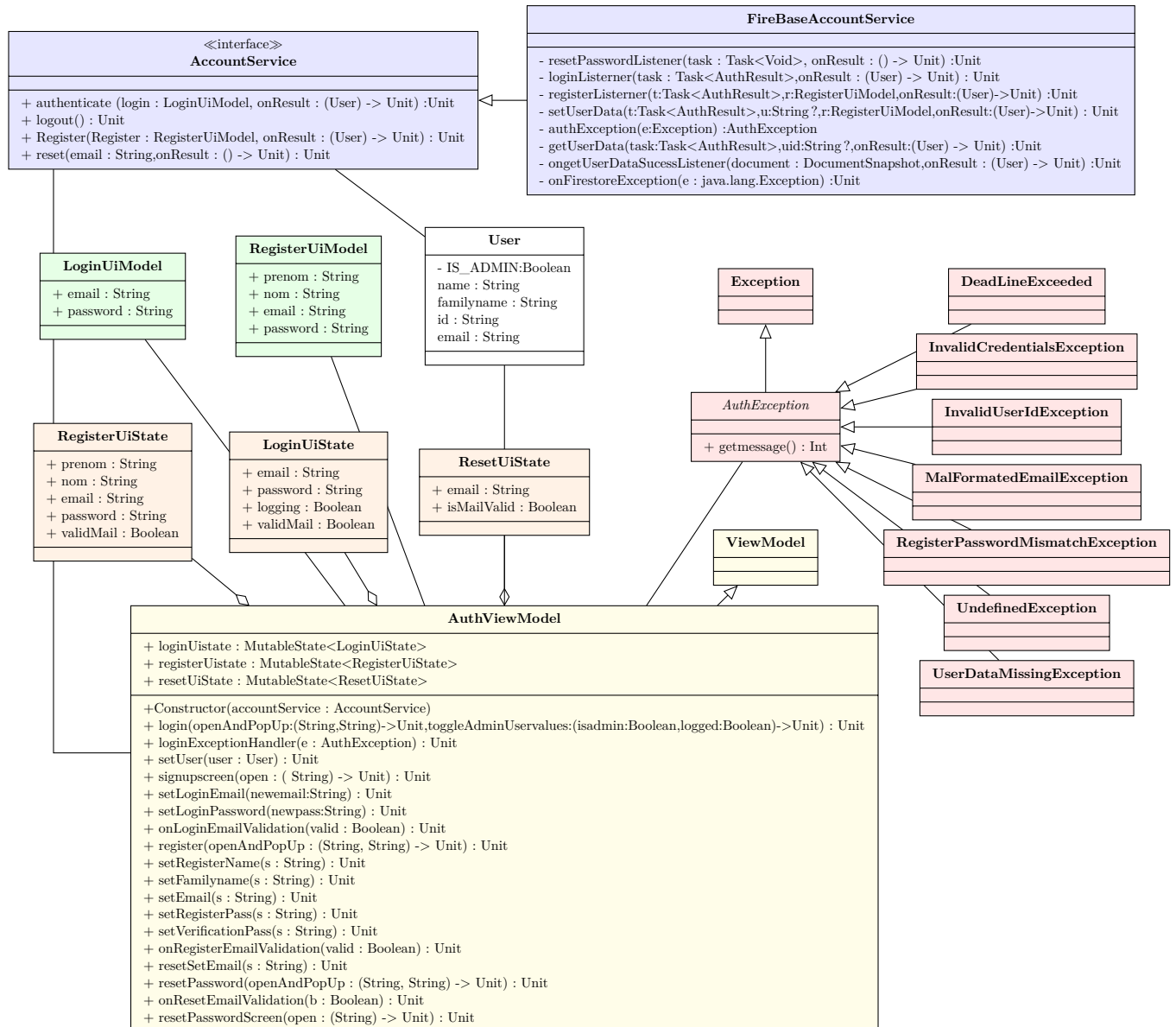


FIGURE 4.11 – le diagramme de Class pour Sprint 1

4.2.4 Realisation

4.2.4.1 interface de connexion



FIGURE 4.12 – écran de connexion (light mode)



FIGURE 4.13 – écran de connexion (dark mode)

4.2.4.2 interface d'inscription et de renitialisation de mot de passe

23:09 78%

BIENVENUE VOISIN

nom

prenom

Email

mot de passe

retapez le mot de passe

s'inscrire

FIGURE 4.14 – ecran d'inscription

23:09 78%

BIENVENUE VOISIN

Email

réinitialiser le mot de passe

FIGURE 4.15 – ecran de reinitialisation de mot de passe



FIGURE 4.16 – animation lors de connexion



FIGURE 4.17 – exemple d'erreur 1



FIGURE 4.18 – exemple d’erreur 2



FIGURE 4.19 – exemple d’erreur 3

4.3 Sprint 2

4.3.1 specification fonctionnel

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 à avoir une vue d'ensemble des finances sur une période donnée.

4.3.1.2 consulter les operations 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 operations serait affiché en précisant la date de l'operation et montant d'operation, le cotisateur s'il s'agit d'une cotisation ou le type de depense s'il s'agit d'une depense.

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	ELVEE
#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	ELEVEE

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.

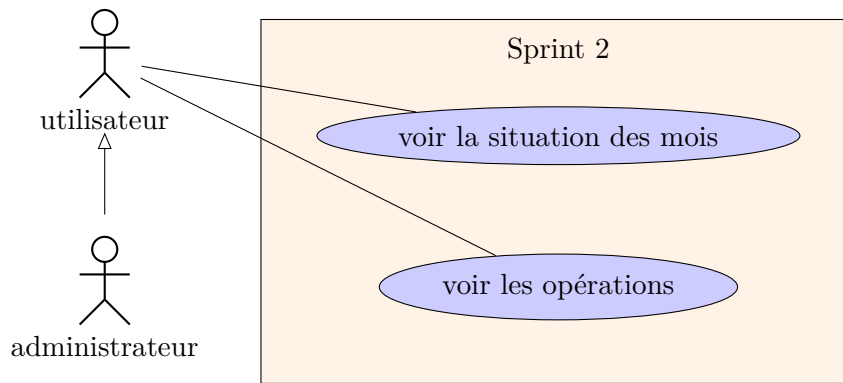


FIGURE 4.20 – le diagramme de cas d'utilisation pour Sprint 2

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. Operation : qui représente une opération (soit une contribution soit une dépense).

Normalement, nous devrions avoir deux classes qui héritent de l'operation 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).

L'interface DataService est destinée à être étendue par tous les types de fournisseurs de services, dans notre cas la classe FirebaseDataService l'étend.

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).

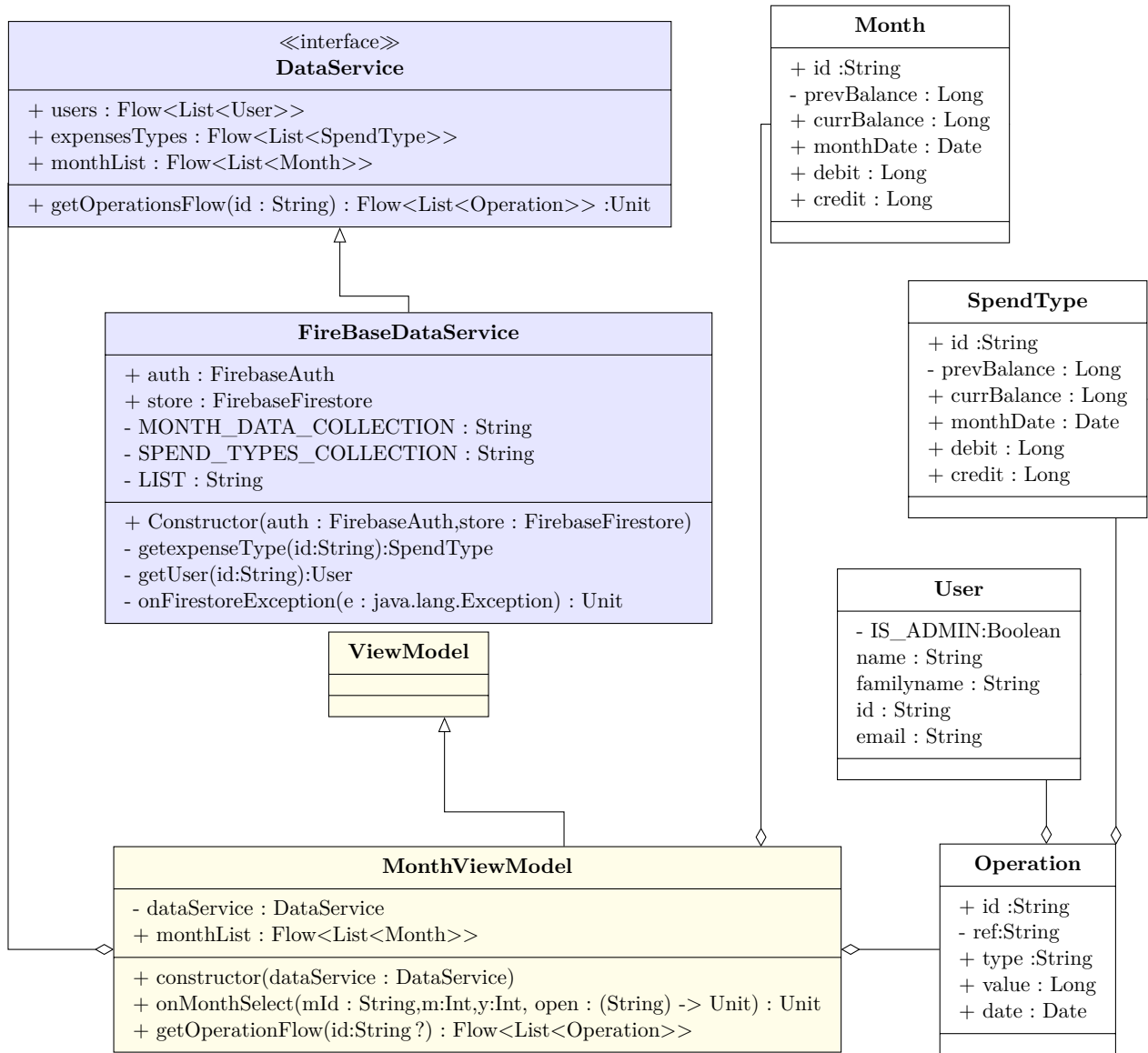


FIGURE 4.21 – le diagramme de Class pour Sprint 2

4.3.4 Realisation

4.3.4.1 les interfaces

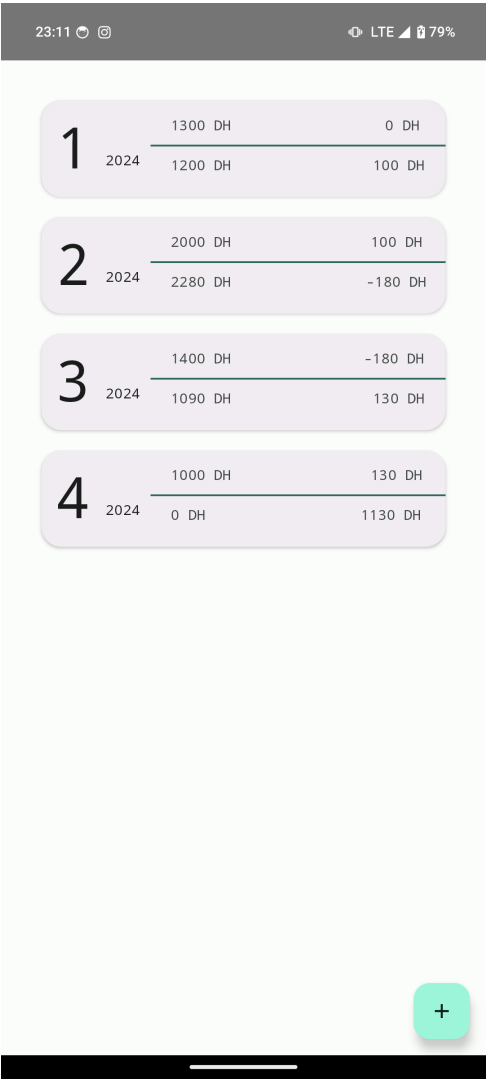


FIGURE 4.22 – ecran de la liste des mois



FIGURE 4.23 – ecran de la liste des operations de janvier 2024

4.4 Sprint 3

4.4.1 specification fonctionnel

4.4.2 Sprint Backlog

Gi- tHub 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 prédéfinies dans l'application afin de mieux catégoriser les dépenses	Adminis- trateur	ELVEE
#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.	Adminis- trateur	ELEVEE
#19	En tant qu'utilisateur je veux avoir la possibilité de modifier un type de dépenses dans la liste des dépenses prédéfinies dans l'application afin de mieux catégoriser les dépenses.	Adminis- trateur	ELEVEE
#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.	Adminis- trateur	ELEVEE
#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.	Adminis- trateur	MOYEN

4.4.3 conception

4.4.3.1 diagramme de cas d'utilisation

4.4.3.2 diagramme de classe

4.4.4 Realisation

4.4.4.1 interface pour gérer les opération

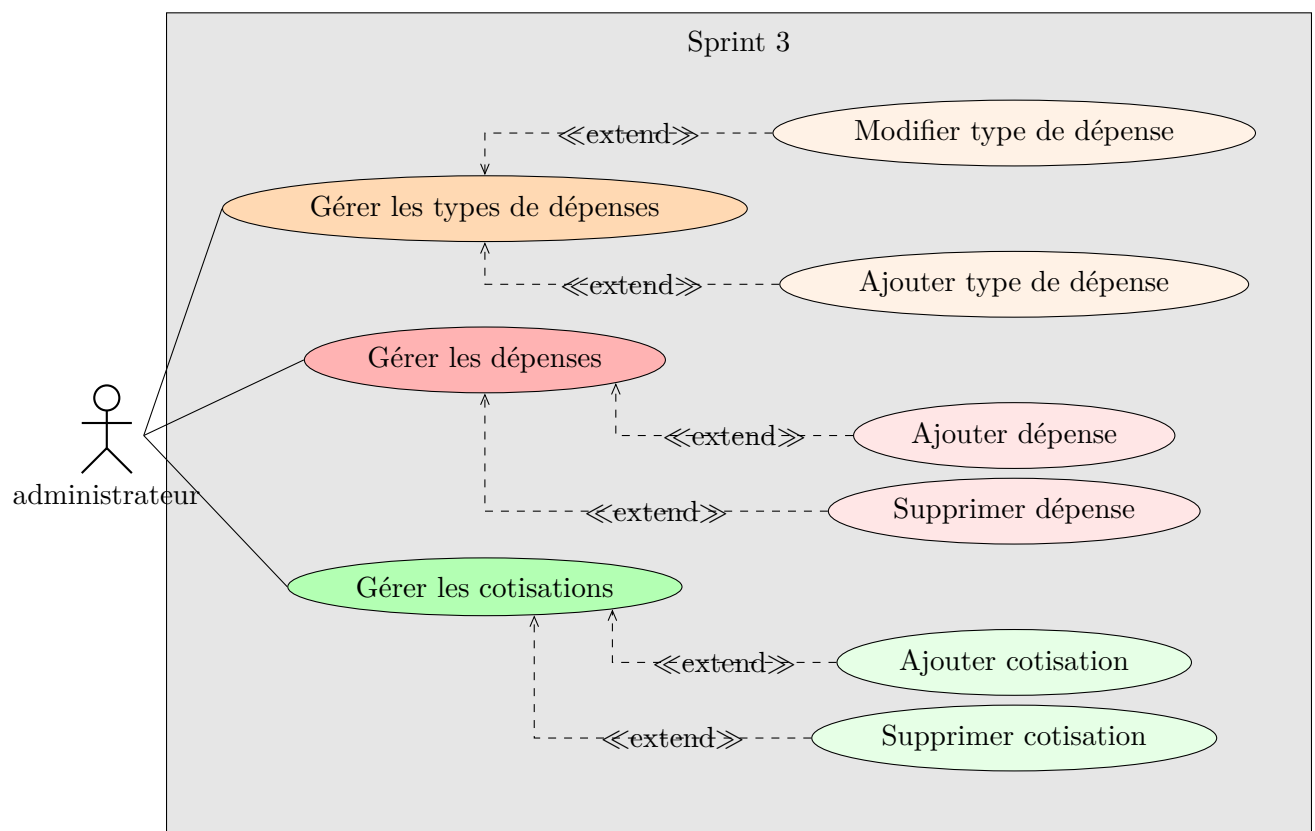
4.4.4.1.1 ajouter type de dépense

4.4.4.1.2 modifier type de dépense

4.4.4.1.3 ajouter de dépense

4.4.4.1.4 ajouter cotisation

4.4.4.1.5 supprimer operation



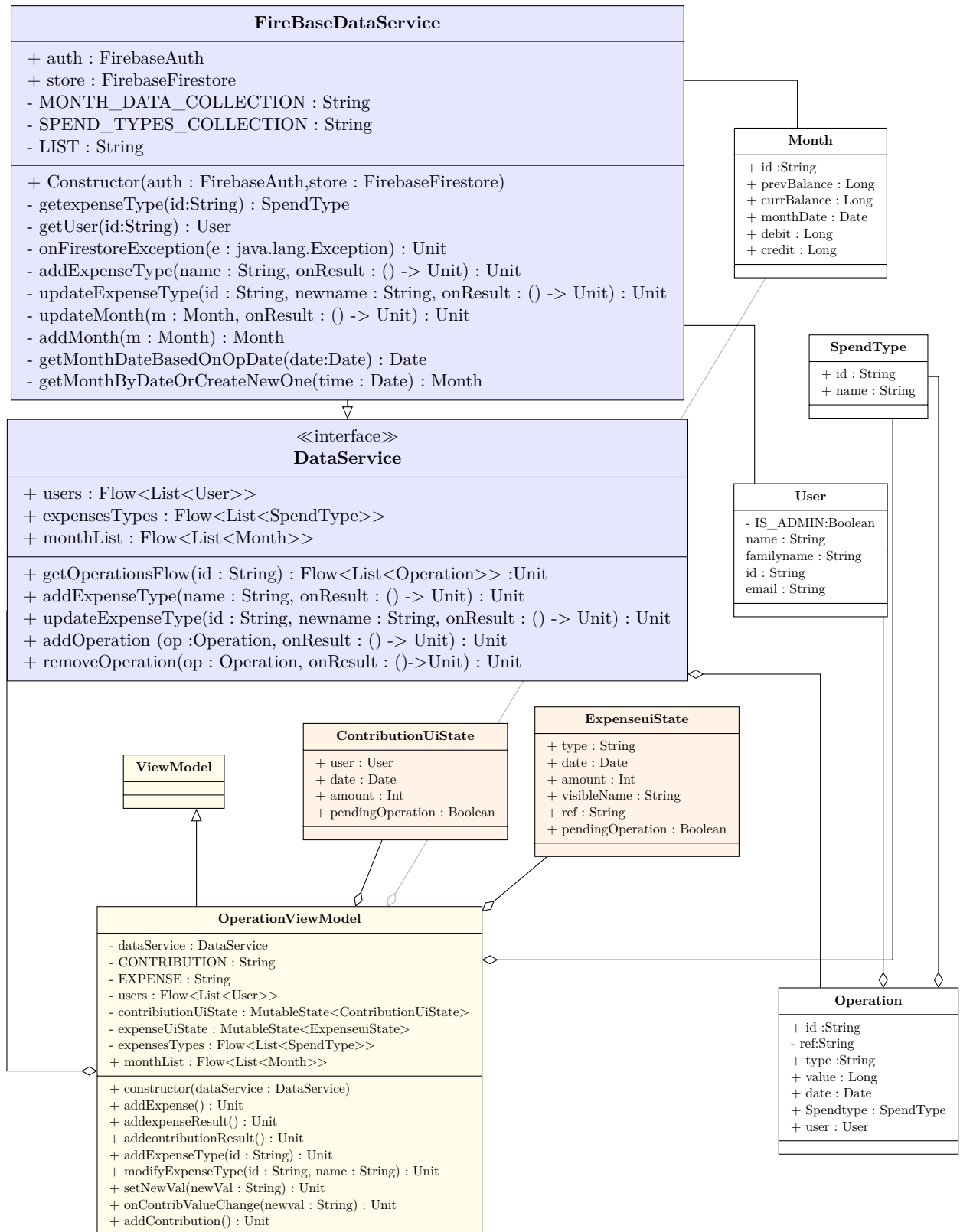


FIGURE 4.24 – le diagramme de Class pour Sprint 2

annexes

TO BE IMPLEMENTED