

Université de Mons
Faculté Des Sciences
Département d’Informatique

**Projet de modélisation logicielle
Rapport de modélisation**

Professeur :

Tom MENS

Assistants :

Jeremy DUBRULLE

Gauvin DEVILLEZ

Sébastien BONTE

Auteurs :

Pignozi AGBENDA

Thomas BERNARD

Théo GODIN

Ugo PROIETTI



Année académique 2021-2022

Projet de Modélisation Logicielle

AGBENDA Christian, BERNARD Thomas, GODIN Theo et PROIETTI Ugo

Table des matières

1	Introduction	1
2	Applications de bases	1
2.1	Application1 : Gestion de portefeuilles financiers	1
2.1.1	Vue d'ensemble	1
2.1.2	Diagramme de cas d'utilisation	1
2.1.3	Interaction Overview Diagram	2
2.1.4	Diagramme de classes	3
2.1.5	Diagrammes de séquences	3
2.2	Application 2 : Application de gestion pour institutions financières	3
2.2.1	Vue d'ensemble	3
2.2.2	Diagramme de cas d'utilisation	3
2.2.3	Interaction Overview Diagram	4
2.2.4	Diagramme de classes	4
2.2.5	Diagramme de séquences	4
2.3	Serveur	4
2.3.1	Vue d'ensemble	4
2.3.2	Diagramme d'entité-relation	4
2.3.3	API	5
2.4	Interface Graphique	8
2.4.1	Application clients	8
3	Extensions	14
3.1	Extension 1, Gestion des cartes - Godin Théo :	14
3.2	Extension 2, Gestion des devises et virements internationaux - Proietti Ugo :	14
3.3	Extension 5, Gestion des contrats d'assurance - Bernard Thomas :	14
3.3.1	Vue d'ensemble	14
3.3.2	Application 1	15
3.3.3	Diagramme des cas d'utilisation	15
3.3.4	Interaction Overview Diagram	16
3.3.5	Diagrammes de classe	17
3.3.6	Diagrammes de séquence	18
3.3.7	Application 2	20
3.3.8	Diagramme des cas d'utilisitation	20

3.3.9	Interaction Overview Diagram	20
3.3.10	Diagramme de classes	20
3.3.11	Diagrammes de séquences	23
3.3.12	Serveur	23
3.3.13	Diagramme d'entité relation	23
3.3.14	Interface graphique de l'extension	24
3.4	Extension 6, Paiement et gestion des fraudes - Agbenda Pignozi :	24

1 Introduction

2 Applications de bases

2.1 Application1 : Gestion de portefeuilles financiers

2.1.1 Vue d'ensemble

L'application 1 est l'application qui est destinée aux clients. Il s'agit de l'application sur laquelle nous avons commencé à travailler et donc celle qui a subi le plus de modifications au total. Il est possible que certains concepts soient modifiés lors de l'implémentation notamment ceux qui n'ont pas été approfondis au maximum. Cette application sera étendue par l'entièreté du groupe.

2.1.2 Diagramme de cas d'utilisation

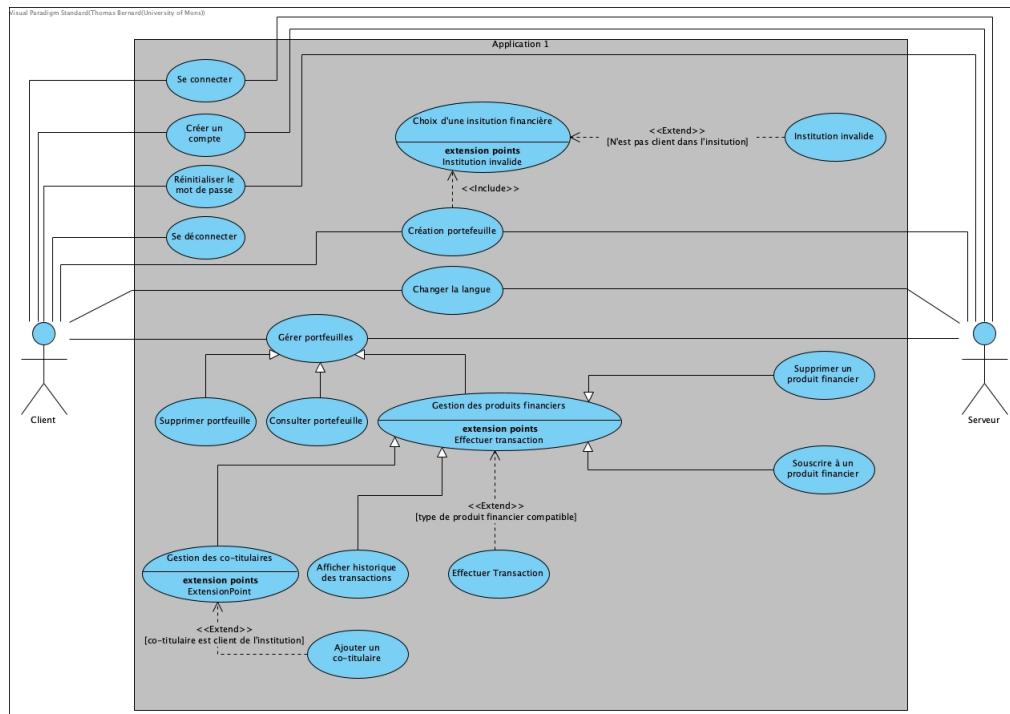


FIGURE 1 – Diagramme des cas d'utilisation de l'Application 1.

2.1.3 Interaction Overview Diagram

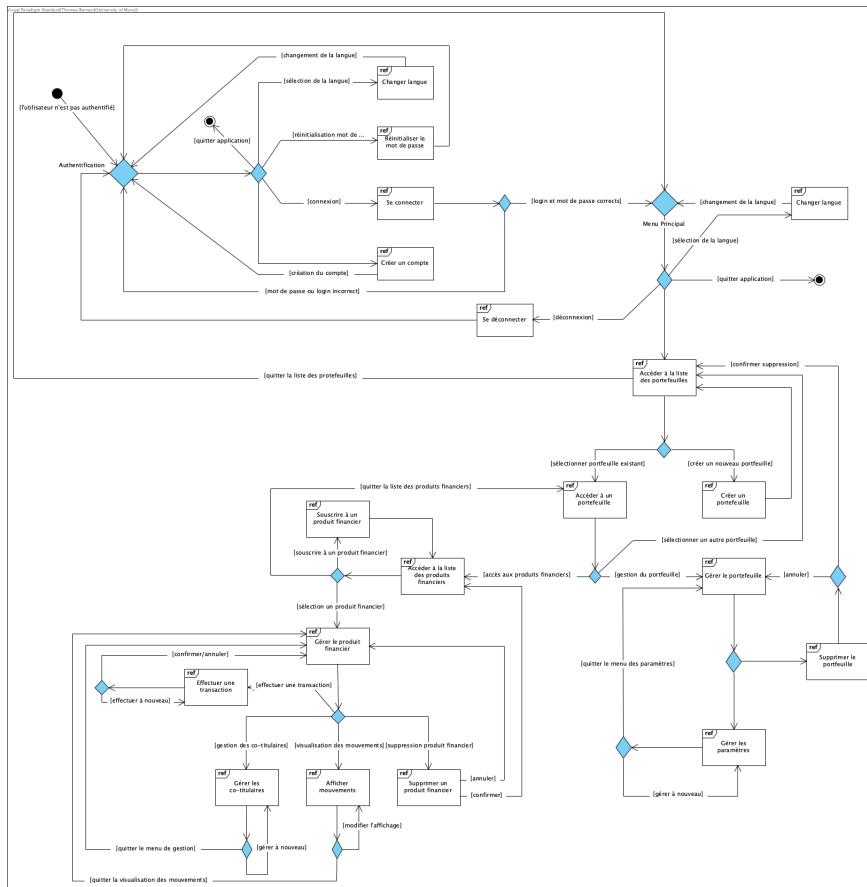


FIGURE 2 – Interaction Overview Diagram Application 1.

Le schéma des interactions de l'application utilisateurs est composée de 3 parties principales. Une partie dédiée à l'accès à l'application (authentification), une partie pour la l'accès et la gestion des portefeuilles financiers la dernière pour l'accès et la gestion des produits financiers.

2.1.4 Diagramme de classes

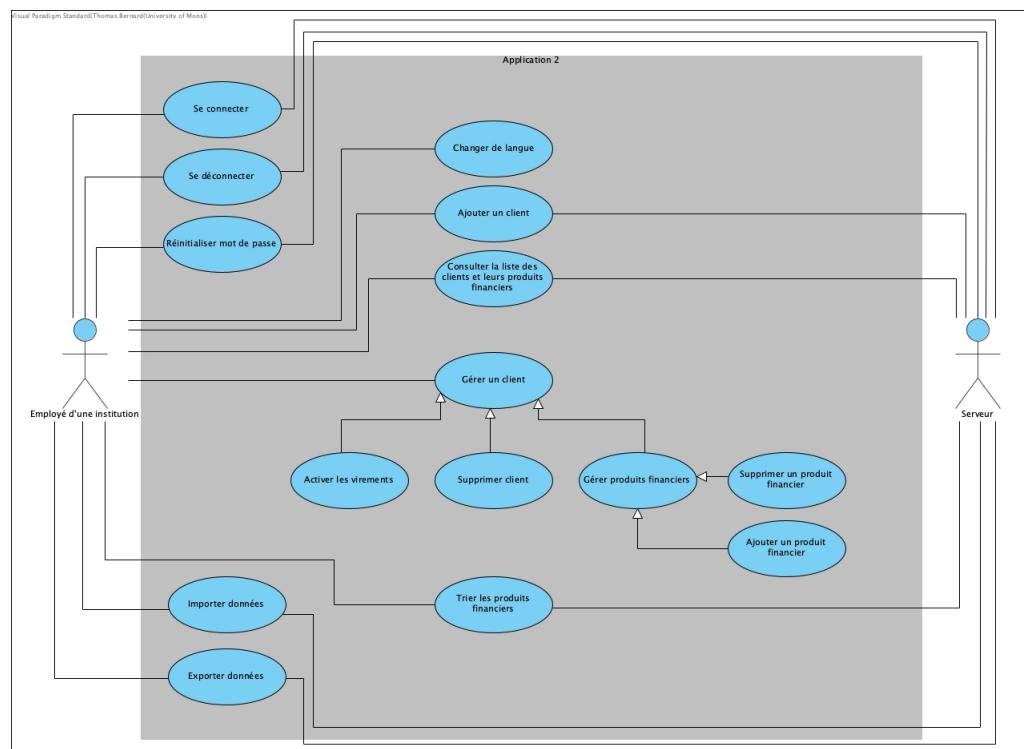
2.1.5 Diagrammes de séquences

2.2 Application 2 : Application de gestion pour institutions financières

2.2.1 Vue d'ensemble

L'application 2 est l'application destinée aux institutions. Cette application leur permet de gérer leurs clients ainsi que les produits de ceux-ci. La conception de cette application est basée sur celle de l'application 1. En effet l'application 2 reprend certains des aspects de l'application 1. Tout en étant plus restrictive. En effet, l'application 2 n'a pas d'accès à la notion de portefeuille. Toutefois, elle possède des outils de gestion qui ne font pas partie de l'application 1.

2.2.2 Diagramme de cas d'utilisation



2.2.3 Interaction Overview Diagram

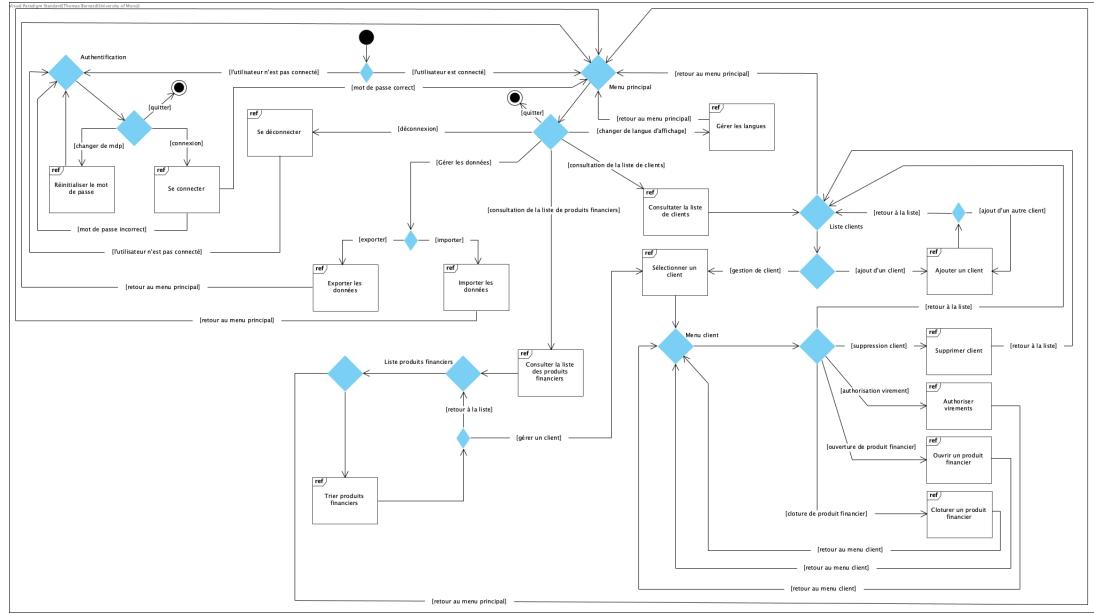


FIGURE 3 – Interaction Overview Diagram Application 2.

L'utilisateur d'un application d'institution financière devra d'abord s'authentifié via le menu d'authentification schématisé dans le coin supérieur gauche du diagramme.

Il devra utiliser le numéro swift et mot de passe de l'institution pour laquelle il travaille. Il pourra ensuite choisir de rester connecté à l'application.

Le menu principal de l'application permet d'accéder à la liste des clients et ainsi depouvoir effectuer des opérations sur chaque client de l'institution ou d'en ajouter de nouveaux.

La partie gestion de clients est schématisée dans la partie droite du diagramme.

2.2.4 Diagramme de classes

2.2.5 Diagramme de séquences

2.3 Serveur

2.3.1 Vue d'ensemble

2.3.2 Diagramme d'entité-relation

Tables utilisateurs : Pour notre base de données nous avons essayé de séparer au mieux les tables afin que celles-ci restent simples de compréhension et faciles d'accès. Nous avons une table **USER_PASSCLI** qui sert de table

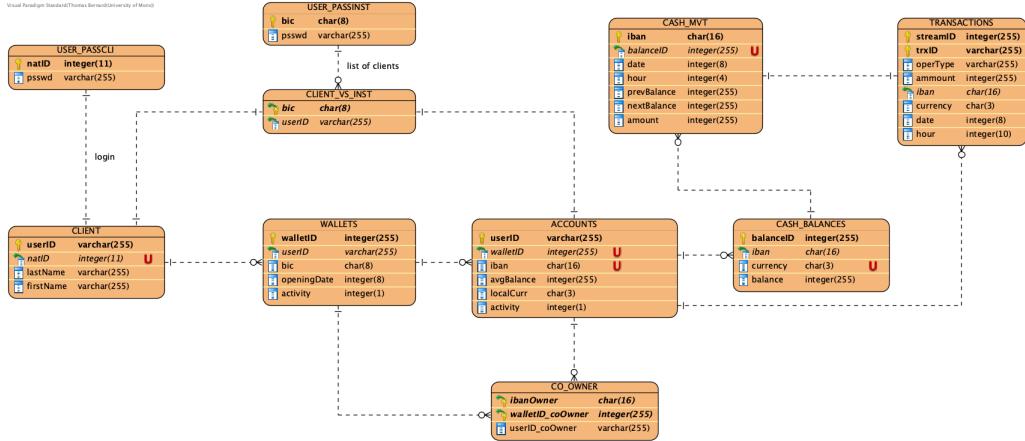


FIGURE 4 – Diagramme d'entité-relation de l'application.

d'authentification. Elle contient le mot de passe **psswd** ainsi que le numéro de registre national de l'utilisateur **natID** si les deux correspondent alors on peut accéder à la table **CLIENT** qui contient quant à les informations personnelles du client ainsi que son nom d'utilisateur **userID**.

Tables des institutions : Afin que les institutions n'aient pas accès à la notion de portefeuilles nous avons introduit 2 tables qui leurs sont propres et qui permettent de faire le lien entre les clients de la table **CLIENT** et leurs produits financiers de la table **ACCOUNTS**.

La table **USER_PASSINST** est à l'image de la table **USER_PASSCLI** une table d'authentification pour les institutions à l'aide de leur identifiant **bic**.

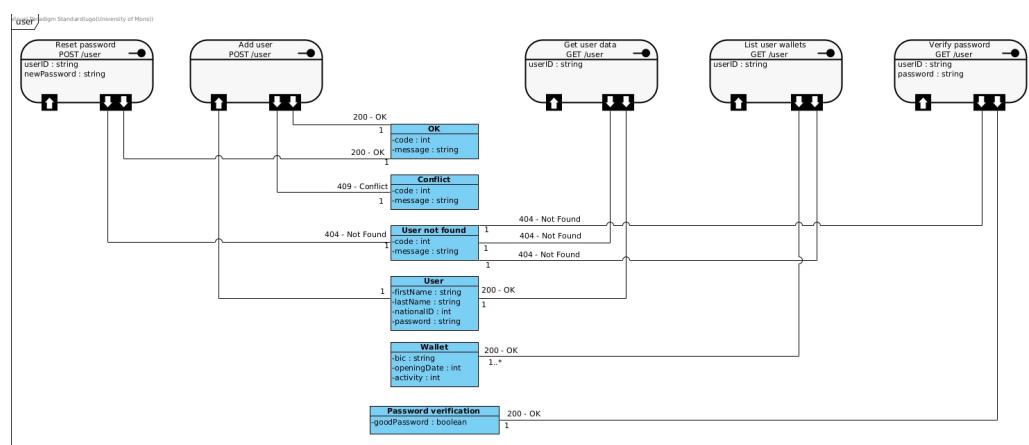
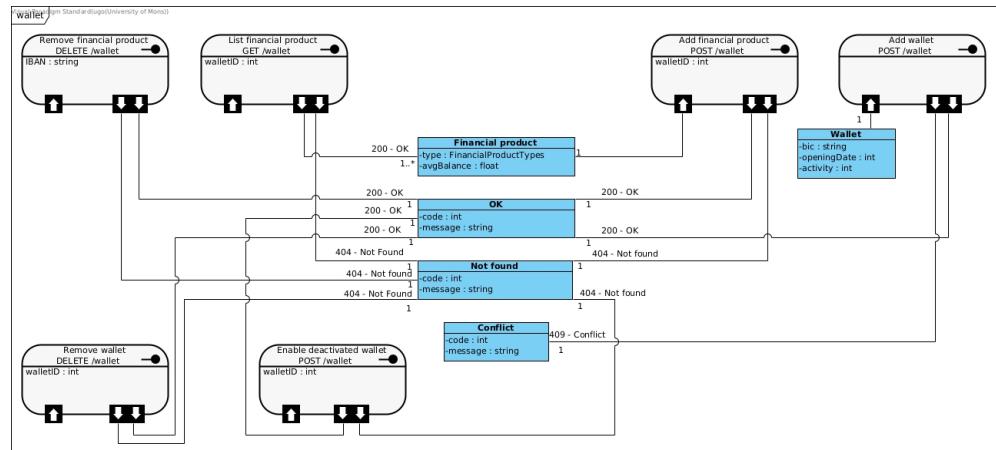
La table **CLIENT_VS_INST** associe chaque client et institution si pour un **userID** il possède au moins un produit financier dans l'institution. Lorsqu'un **wallet** est créé le **userID** du client ainsi que le **bic** de l'institution dans lequel il est créé sont ajoutés à la table ce qui permet ainsi à l'institution de récupérer la liste de ses clients assez aisément à l'aide de cette table. De plus elle peut ensuite se servir du **userID** afin de récupérer la liste des produits du client (table **ACCOUNTS** ainsi que ses informations personnelles (table **CLIENT**).

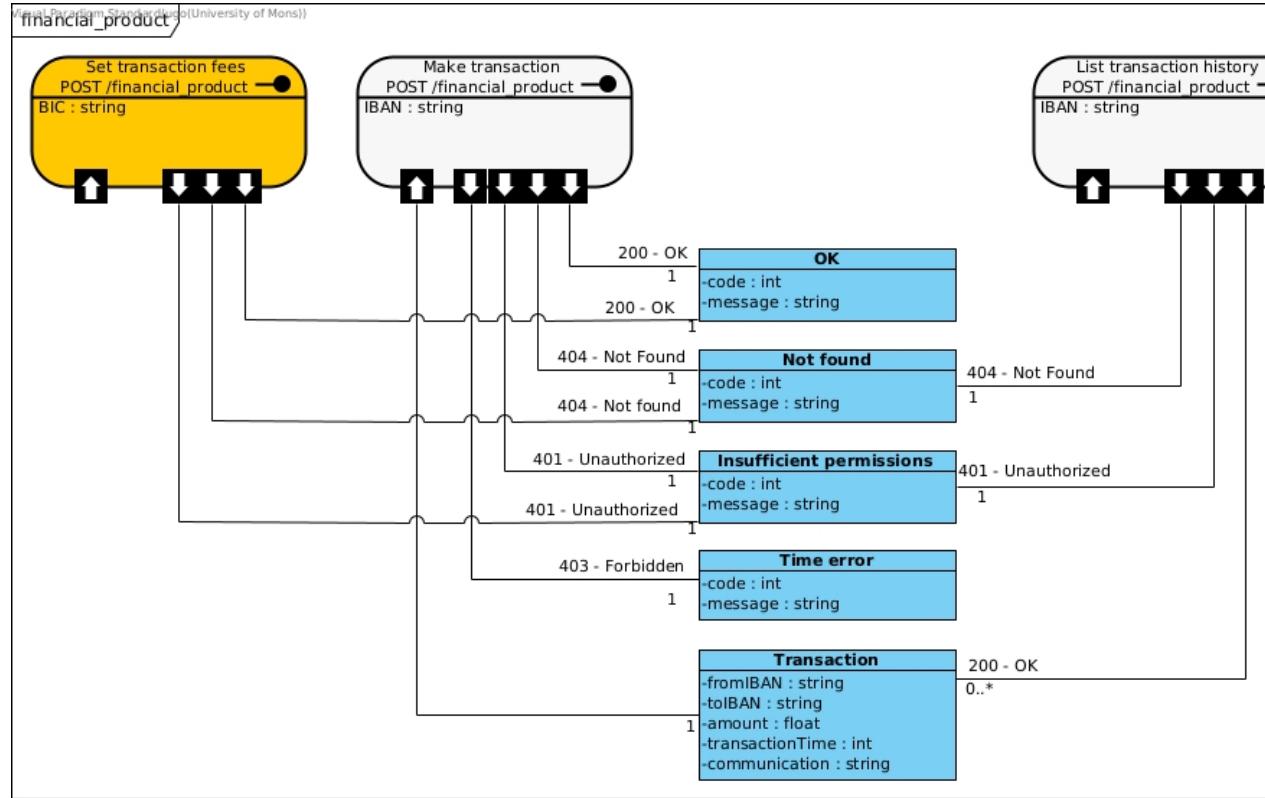
2.3.3 API

L'API est de type REST. Elle a été divisée en 3 endpoints (`/user` ; `/wallet` ; `/financial_product`) correspondant chacun à un type de donnée pouvant être édité.

Du à un manque de compréhension, nous n'avons pas utilisé les erreurs de la série 500 (problème serveur) dans le diagramme mais nous avons bien utilisé les erreurs de la série 400 correspondant à un problème dans la demande à l'API.

Nous utilisons les erreurs 401 - Unauthorized , 403 - Forbidden , 404 - Not found , 409 - Conflict.





2.4 Interface Graphique

2.4.1 Application clients

Cette section décrit les différents interfaces auxquels les utilisateurs de l'application auront accès pour gérer leurs comptes, portefeuilles et produits financiers.

1. LoginScreen

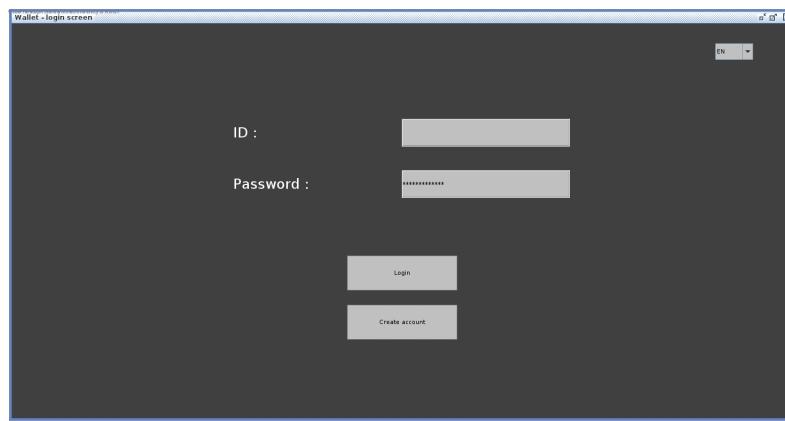


FIGURE 5 – Login Scene

L'écran de connexion est la première interface affichée lors du démarrage de l'application.

Cette interface permet à l'utilisateur de se connecter via ses identifiants ou d'accéder à l'écran de création de compte.

La langue d'affichage peut également être changée directement depuis le coin supérieur droit.

2. RegisterScreen

Un utilisateur peut s'enregister via cet écran en entrant les données demandées.

Le numéro de registre national étant unique, le client ne sera pas en mesure de se créer plusieurs compte sur l'application.

Une fois le compte créé, l'utilisateur sera amené à un écran de connexion afin de procéder à l'authentification.

3. MainMenuScreen

Une fois qu'un utilisateur s'est authentifié via le LoginScreen, le menu principal lui est affiché. Il permet à l'utilisateur d'accéder à toutes ses

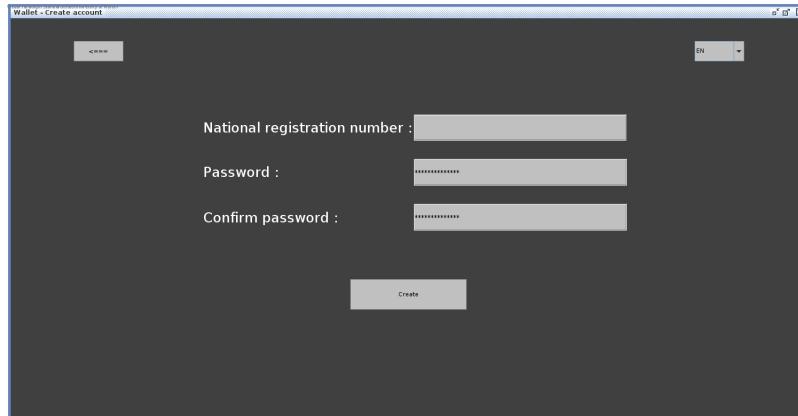


FIGURE 6 – Login Scene

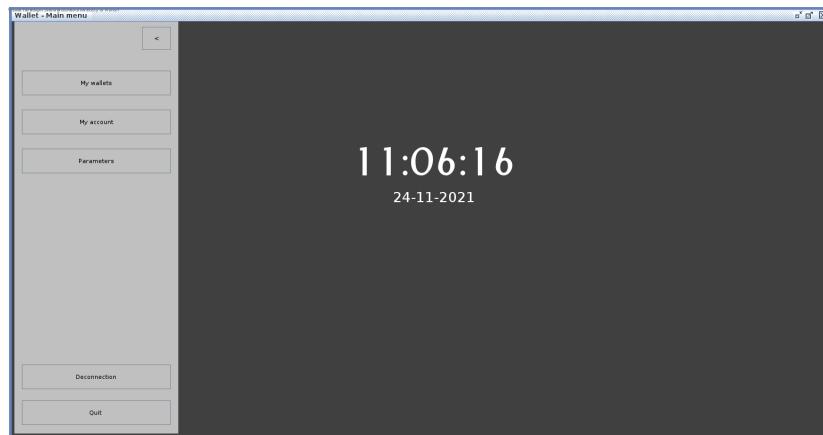


FIGURE 7 – Login Scene

données via le menu à gauche de l'écran. Il contient un bouton “My wallets” qui mène l'utilisateur vers l'écran de gestion de ses portefeuilles financiers.

Un bouton “My account” permet à l'utilisateur d'avoir un écran d'aperçu de ses données. Il pourra aussi changer de mot de passe via cet écran.

Le bouton “Parameters” mène à l'écran des paramètres de l'app.

Le bouton “Deconnection” permet de se déconnecter afin de changer d'utilisateur (il mène à l'écran de connexion).

Et le bouton “Quit” déconnecte l'utilisateur et ferme l'application.

4. WalletsListScreen

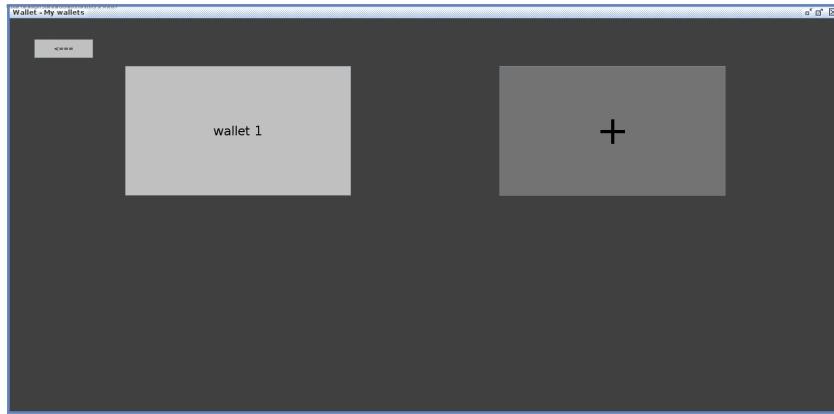


FIGURE 8 – Wallet List Scene

Cet écran affiche tous les portefeuilles de l'utilisateur ainsi qu'un bouton permettant d'en ajouter de nouveaux.

Les boutons de chaque portefeuille mènent l'utilisateur aux écrans affichant les détails de ceux-ci tandis que le bouton d'ajout de portefeuille envoie sur l'écran de création de portefeuille.

L'utilisateur ne pourra créer qu'un portefeuille par institution dont il est client.

Un bouton de retour en arrière est aussi présent sur l'écran des portefeuilles afin de permettre à l'utilisateur de retourner au menu principal.

5. AddWalletScreen

AddWalletScreen propose à l'utilisateur de choisir une institution financière pour laquelle il veut avoir un portefeuille financier visible dans le menu des portefeuilles (WalletsListScreen). Si il est client de cette institution, la demande sera validée et le portefeuille sera accessible.

Un bouton retour en arrière se trouve dans le coin supérieur gauche de l'écran. Ce bouton renvoie vers la liste des portefeuilles de l'utilisateur.

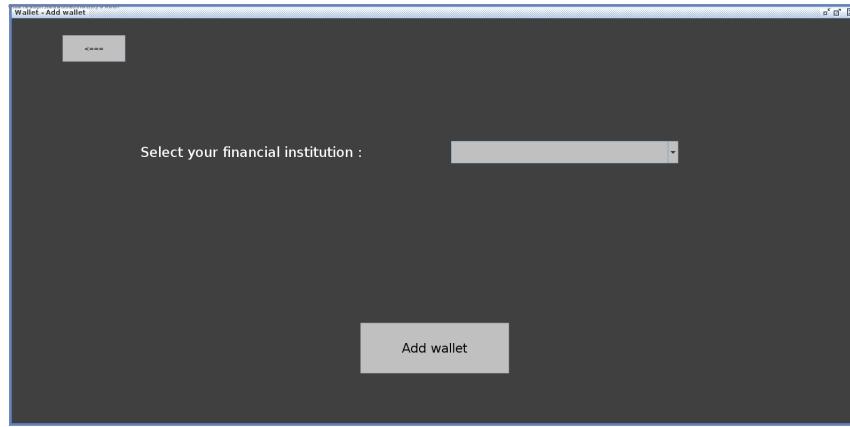


FIGURE 9 – Scene to add a wallet

6. ProductsListScreeen



FIGURE 10 – List of the products

Chaque bouton permet à l'utilisateur de consulter les détails d'un de ses produits financiers.

Il peut faire une demande d'ajout de nouveau produit financier. Une fois que celle-ci sera validée par l'institution, le produit créé apparaîtra dans la liste du menu précédent.

Un bouton de retour en arrière permet de revenir au menu précédent manuellement et donc d'annuler la création du produit financier.

7. ProductMenuScreen

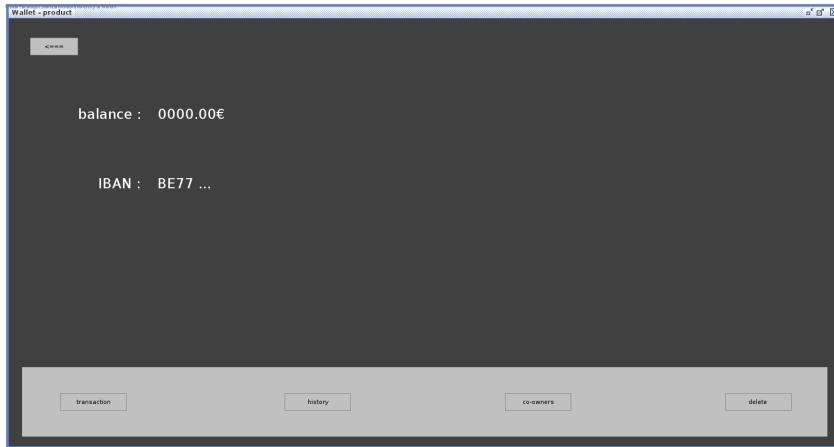


FIGURE 11 – Product menu

Cet écran est affiché à l'utilisateur lorsqu'il clique sur un de ses produits financiers.

Les informations générales du produit financier (telles que le solde du compte et le numéro de compte) y sont disposées et les boutons du bas de l'écran permettent à l'utilisateur de procéder à différentes opérations sur ce produit financier.

Effectuer une transaction se fait via le bouton "transaction", consulter l'historique des transactions se fait via le bouton "history", la gestion des co-titulaires se fait via le bouton "co-owners" et enfin la suppression du produit se fait via le bouton "delete".

Le bouton "delete" demandera à l'utilisateur de confirmer son action avant de procéder à la désactivation du produit.

Un bouton de retour en arrière est également disponible afin de revenir à l'écran du portefeuille dans lequel se trouve le produit financier courant.

8. TransactionScreen

L'écran de transaction demande plusieurs informations à l'utilisateur afin d'effectuer une transaction d'un compte A à un compte B.

Si ce menu a été accédé via le menu d'un produit financier, le compte envoyeur ne peut pas être modifié par l'utilisateur. Sinon, il doit sélectionner un compte parmi tous ceux qu'il possède.

Le nom et l'IBAN du receveur doit ensuite être entré dans leurs champs respectifs. Finalement, le montant de la transaction doit être précisé et optionnellement une communication peut être jointe à la transaction.

Une fois toutes les données entrées, le bouton "confirm" va afficher un résumé de la transaction, l'utilisateur pourra alors décider de confirmer

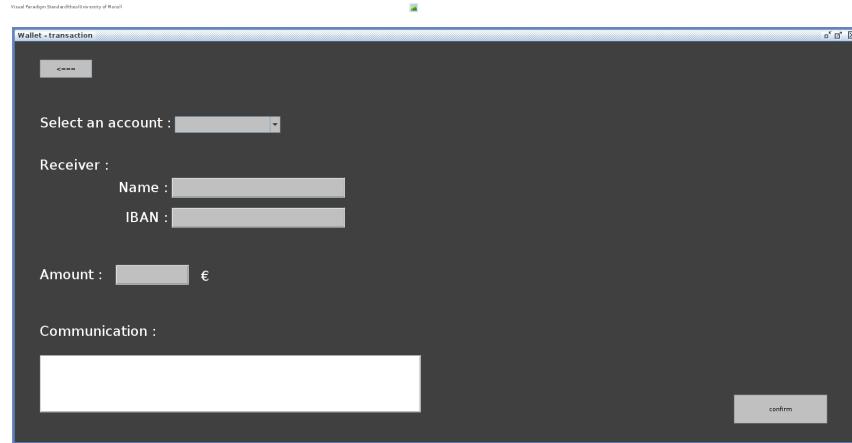


FIGURE 12 – Transaction scene

ou annuler la transaction. Le bouton retour arrière renvoie vers le menu du produit financier et annule la transaction courante.

9. ProductHistoryScreen

lastName firstName IBAN	
Actor date	amount
Cofeo Services SA 24 november	€-0,70
Di Vino & Pasta 24 november	€-4,00
Delhaize 22 november	€-2,20
Di Vino & Pasta 21 november	€-4,00

FIGURE 13 – History of the product

Cet écran affiche toutes les transactions effectuées depuis et vers le produit courrant.

Pour chaque transaction, l'envoyeur ou destinataire est affiché ainsi que la date de l'opération et le montant.

Le bouton retour arrière mènera l'utilisateur vers le menu du produit.

10. OwnersScreen

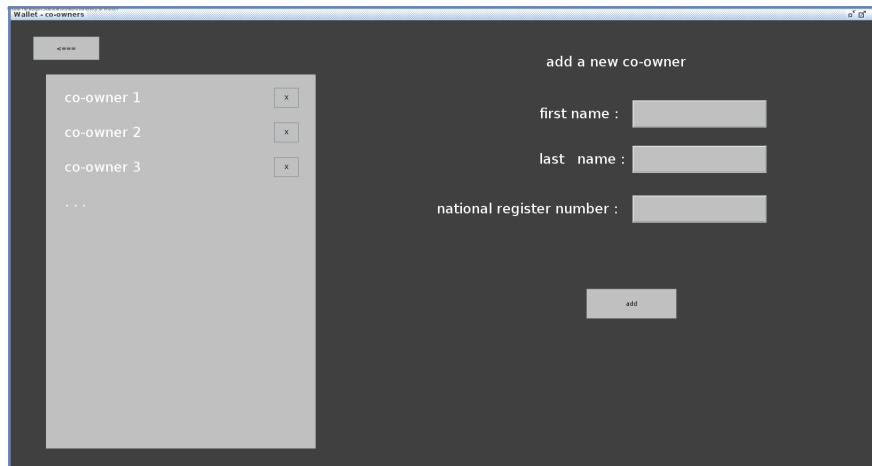


FIGURE 14 – Owners scene

L'écran de gestion des co-titulaires affiche une liste de toutes les personnes partageant le compte.

L'utilisateur peut supprimer ou ajouter des co-titulaire via ce menu.

Le nom, prénom et numéro de registre national sont nécessaires afin d'ajouter un nouveau co-titulaire.

Le bouton de retour arrière renvoie l'utilisateur vers le menu du produit.

3 Extensions

3.1 Extension 1, Gestion des cartes - Godin Théo :

3.2 Extension 2, Gestion des devises et virements internationaux - Proietti Ugo :

3.3 Extension 5, Gestion des contrats d'assurance - Bernard Thomas :

3.3.1 Vue d'ensemble

Le but de cette extension est de rajouter la gestion de contrats d'assurances divers à la fois pour les client smais également pour les institutions. L'extension se base tout de même sur la structure de l'application 1 car c'est dans celle-ci qu'elle la plus utilisée. En effet, dans l'appliction 2 elle est gérée comme les autres produits financiers. Il n'y a réellement que la réponse à une demande de devis qui diffère. Afin de rendre les diagrammes plus visibles j'ai changé les couleurs

des éléments rajoutés. Rose pour le diagramme d'entité relation et jaune pour les autres diagrammes.

3.3.2 Application 1

3.3.3 Diagramme des cas d'utilisation

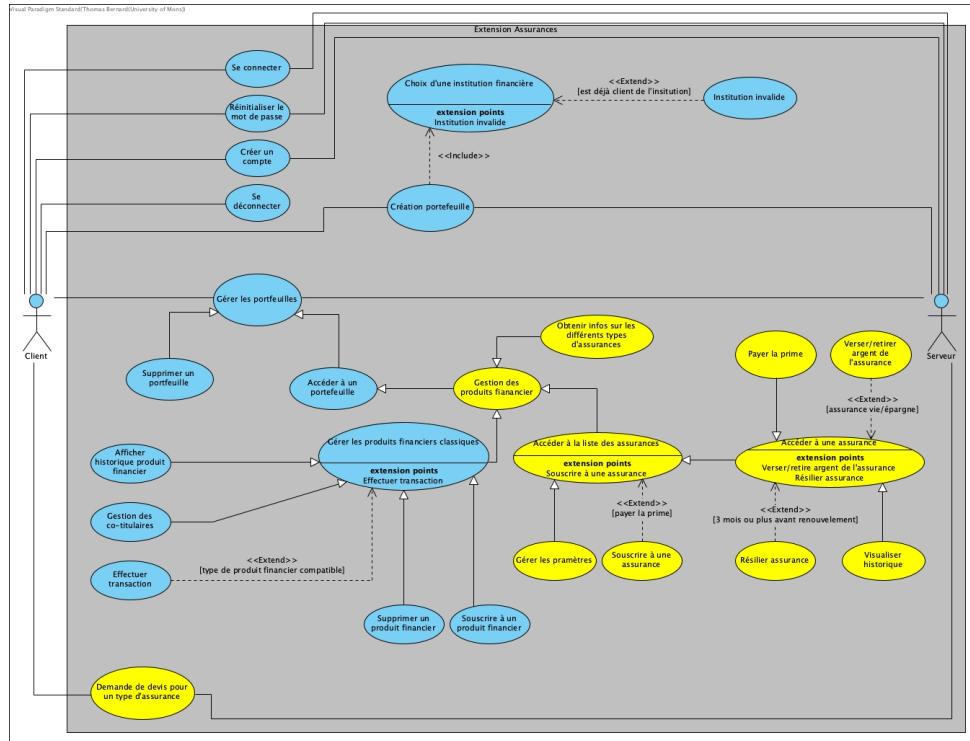


FIGURE 15 – Diagramme des cas d'utilisation de l'app 1 avec extension

J'ai rajouté un ensemble de divers cas d'utilisations qui sont propres aux assurances mais toutefois semblables aux cas d'utilisation relatifs aux produits financiers classiques. Il n'y a aucune remarque particulière à faire concernant les cas d'utilisation car le modèle est calqué sur le diagramme de base.

3.3.4 Interaction Overview Diagram

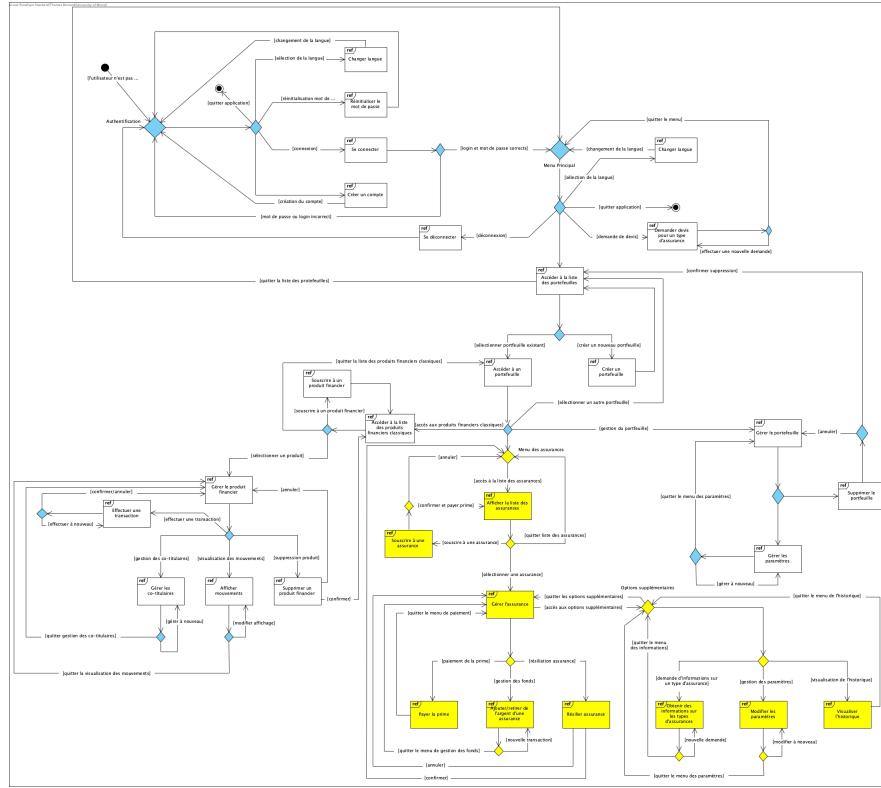


FIGURE 16 – Interaction Overview Diagram de l'app 1 avec extension

Encore une fois ce diagramme se base sur celui de l'application 1. Notons qu'il est maintenant possible de demander un devis depuis le menu principal. Il y a également une différenciation entre les produits financiers classiques et les assurances afin de les séparer en deux scènes différentes plus tard dans la GUI.

3.3.5 Diagrammes de classe

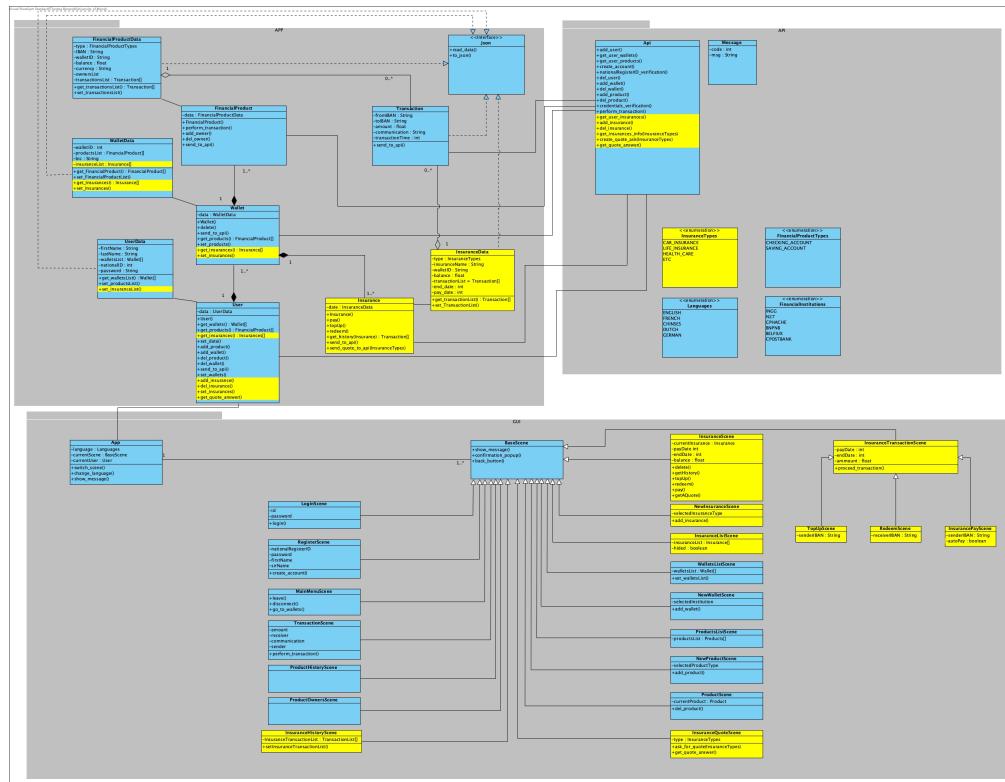


FIGURE 17 – Diagramme de classes de l'app 1 avec extension

Dans la partie logique de l'application (package APP) deux classes ont été rajoutées : **Insurance** et **InsuranceData**. La première contient le constructeur et un attribut *data* qui est une objet de type *InsuranceData*. La classe contient également les méthodes liées aux assurances. La classe **Insurance** est liée à la classe *wallet* de même manière que la classe **FinancialProduct**. La classe **Insurance** possède une méthode *send_to_api()* et est connectée au package API. La classe **InsuranceData** implémente l'interface **Json** qui permet d'écrire et le dire des fichiers :json de données et d'ainsi mettre ses données à jour.

Les classes de bases dans lesquelles se trouvaient des listes de produits, des setter et des getter se sont vues ajouter des setter, des getter et des listes mais cette fois-ci pour les assurances.

Dans la partie serveur du diagramme de classe (package API) j'ai rajouté une enumération **InsurancesTypes** qui contient les différents types d'assurances

afin de pouvoir interpréter correctement les données envoyées à la classe **Api**. Dans la classe **Api** des méthodes ont été rajoutées afin de pouvoir ajouter, supprimer et obtenir les assurances d'un client. Il y a également une méthode permettant d'obtenir des informations sur un ou plusieurs types d'assurances dans le cadre de devis.

Dans la partie interface graphique (package GUI) il s'agit principalement de nouvelles scènes rajoutées à la manière des scènes de l'application de base. Il n'y a pas de point particulier à expliquer ici.

3.3.6 Diagrammes de séquence

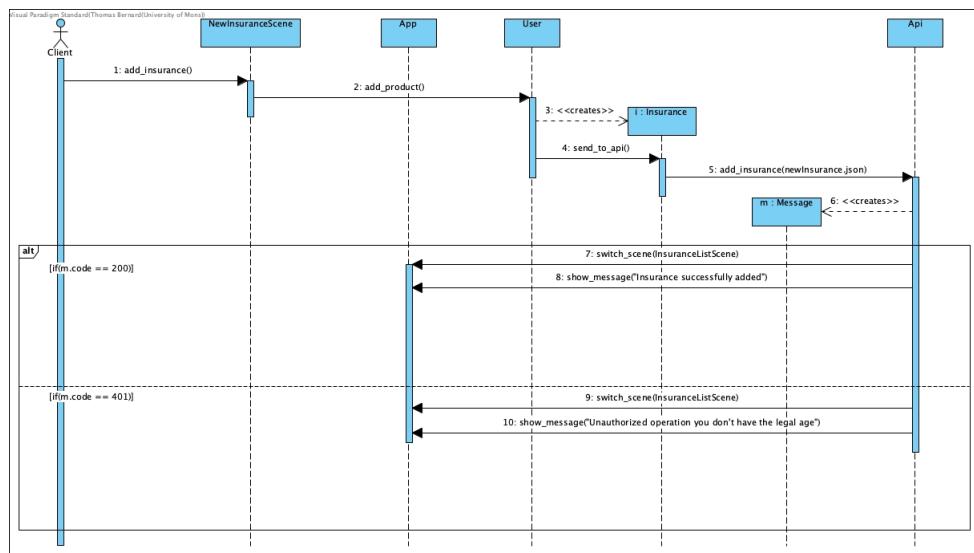


FIGURE 18 – Souscrire à une assurance

Souscrire à une assurance : Lorsque le client souhaite souscrire à une assurance l'application envoie à l'API une requête contenant les informations du demandeur ainsi que les informations de l'assurance demandée. L'API ajoute l'assurance à la base de données. Si l'ajout a bien eu lieu alors la scène change sur celle de la liste des assurances qui lors de sa création récupère les assurances du client. L'application affiche le message qui dit que l'assurance a bien été ajoutée. Si l'ajout n'a pas lieu le client est redirigé sur la scène de la liste des assurances et un message l'informe de l'erreur.

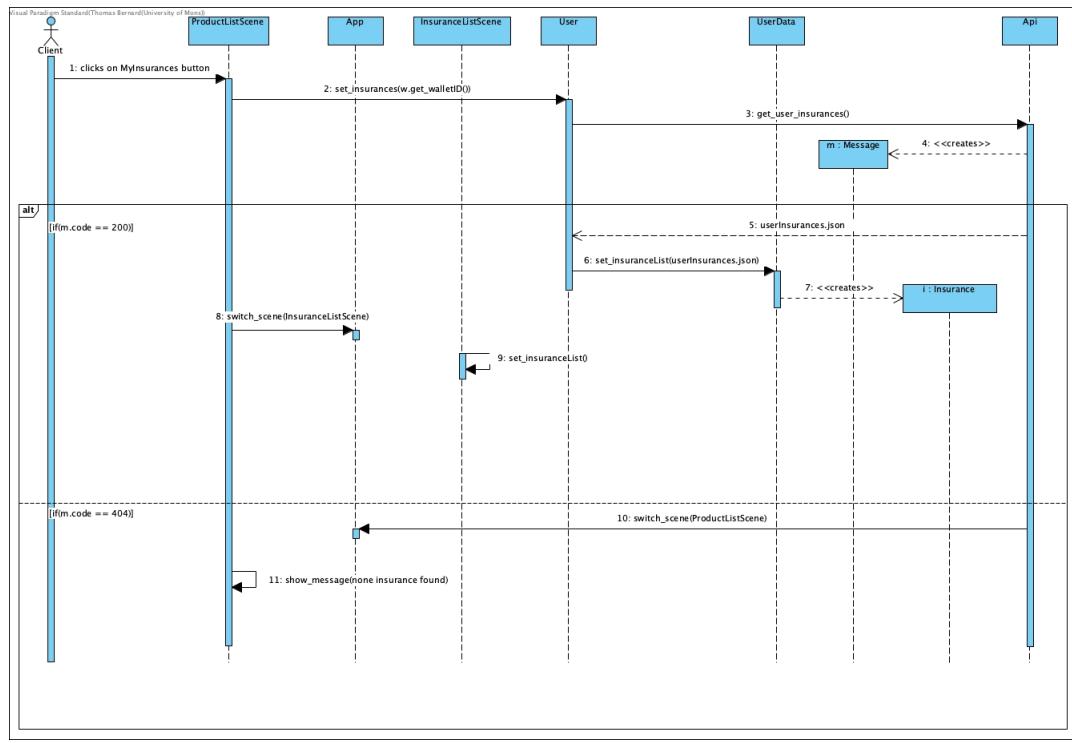


FIGURE 19 – Afficher la liste des assurances

Afficher la liste des assurances :

3.3.7 Application 2

3.3.8 Diagramme des cas d'utilisation

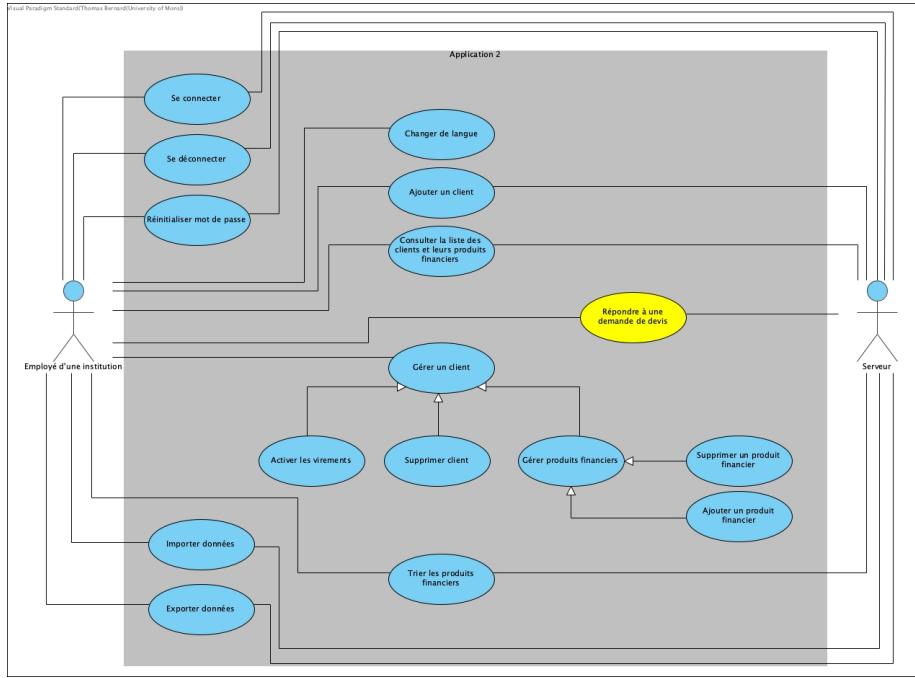


FIGURE 20 – Diagramme des cas d'utilisation de l'app 2 avec extension

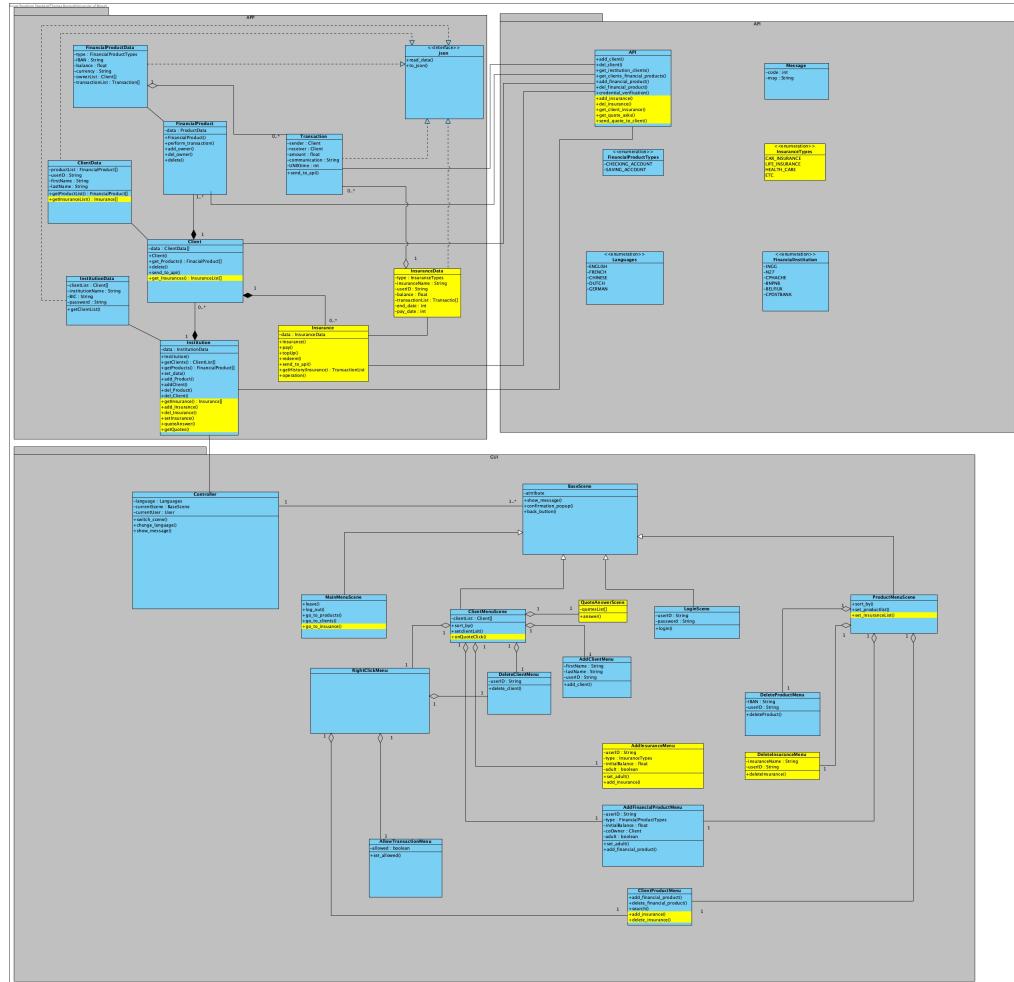
Le diagramme de cas d'utilisation de l'app 2 ne contient qu'une seule modification. Il s'agit du use case permettant de répondre à un devis. En effet lorsqu'une institution liste les produits d'un client elle liste également ses assurances. Il n'est donc pas nécessaire d'en rajouter plus.

3.3.9 Interaction Overview Diagram

Il n'y a également qu'un seul cas d'utilisation rajouté qui est celui de réponse à un devis. Par souci de taille du diagramme il n'a pas été ajouté au rapport. L'interaction a été rajoutée au niveau du menu de gestion des clients. Une fois l'utilisation terminée, le user de l'institution est ramené vers le menu des produits.

3.3.10 Diagramme de classes

Au niveau de la logique de l'application (package APP) mise à part le fait que la classe **Insurance** est maintenant reliée à la classe **Client** du fait que les institutions n'ont pas la notion de portefeuilles et qu'une méthode *quoteAnswer()*



ait été rajoutée à la classe **Institution** permettant d'envoyer le devis au client, il n'y a pas de changements importants par rapport à celui de l'application 1.

Au niveau de la partie serveur de l'application (package API) le seul changement à noter est l'ajout d'une méthode *answerQuote()* dans la classe **Api** qui permet à l'API de générer des réponses à une demande de devis.

Au niveau de la partie interface graphique de l'application (package GUI). Une méthode permettant d'accéder à la scène liée aux assurances a été ajoutée à la classe **MainMenuScene** et une méthode permettant d'accéder à la scène de demande des devis a été ajoutée dans la classe **ClientMenuScene**. Enfin une méthode permettant d'afficher les assurances des clients a été ajoutée à la classe **ProductMenuScene**.

Pour ce qui est des classes ajoutées, elles sont au nombre de 3.
Il s'agit des classes suivantes :

1. **QuoteAnswerScene** : qui est la scène où toutes les demandes de devis sont listées et qui contient un bouton permettant d'y répondre.
2. **AddInsuranceMenu** cette scène permet d'ajouter une assurance pour un client en particulier

3.3.11 Diagrammes de séquences

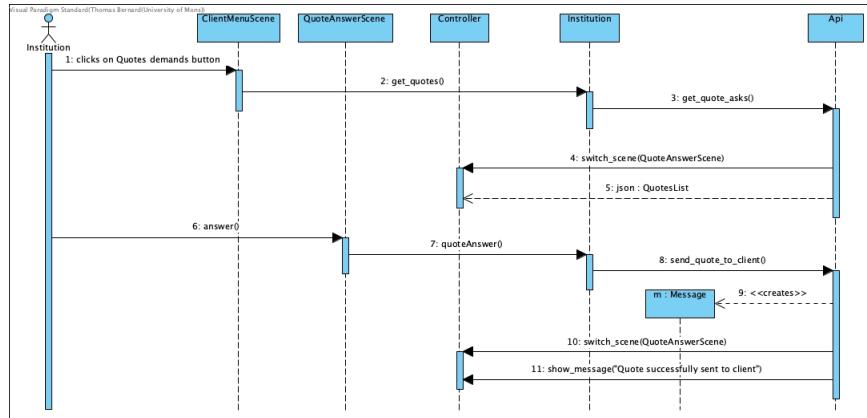


FIGURE 21 – Réponse à une demande de devis

L'insititution lorsque décide d'accéder à la scène des demande devis envoie une requête à l'API qui va rechercher la liste des demandes de devis qui sont propres à l'institution dans la base de données. Ainsi quand la scène est affichée la liste l'est également. Lorsque que l'insititution appuie sur answer une requête est envoyée à l'API qui va chercher les informations liées au type d'assurance demandé par le client et le stocke dans la base de données.

3.3.12 Serveur

3.3.13 Diagramme d'entité relation

Le diagramme d'entité relation a été étendu à l'aide de 3 tables : *INSURANCES*, *ISNURANCE_TYPE* et *INSURANCE_PRICE*.

La table *INSURANCES* contient toutes les assurances des clients. Elle peut être accédée par 2 tables : par la table *WALLETS* grâce au walletID qui lui est lié dans le cadre de l'application client et par la table *CLIENT_VS_INST* grâce au userID qui est lié à chaque assurances dans le cadre de l'application insititution.

La table *INSURANCE_TYPE* est une table intermédiaire contenant toutes les assurances proposées dans l'application. On peut obtenir le type d'une assurance depuis la table *INSURANCES* grâce au insuranceName.

La table *INSURANCE_PRICE* contient les informations pratiques liées à chaque type d'assurance. Car un type d'assurance peut avoir plusieurs nom (insName) notamment les assurances vies. Cette table est accessible depuis la table *INSURANCE_TYPE* grâce au insName.

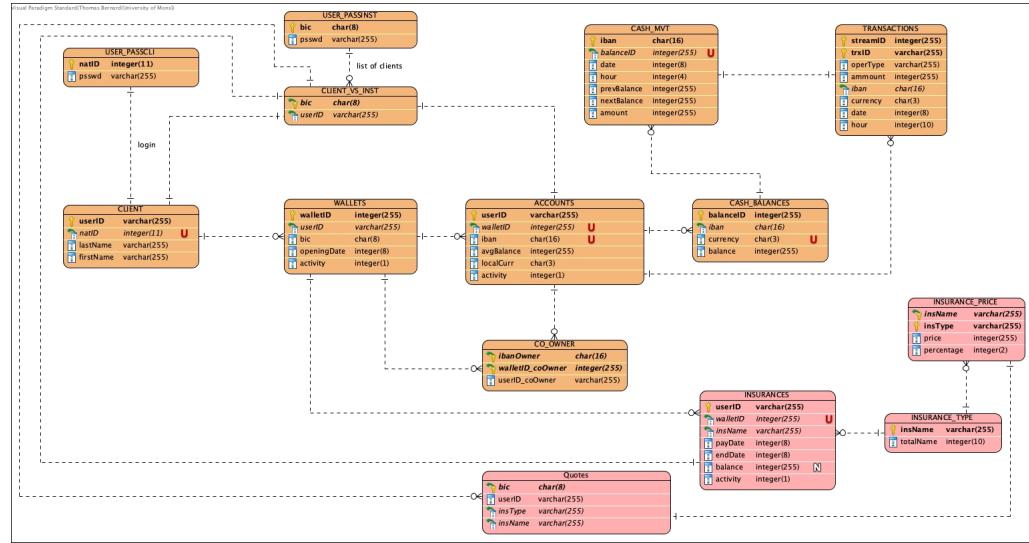


FIGURE 22 – Diagramme d'entité relation avec extension

3.3.14 Interface graphique de l'extension

Plusieurs scènes ont été rajoutées contenant les fonctionnalités de l'extension et des scènes de l'application de base ont été modifiées afin de donner accès à ces nouvelles scènes.

3.4 Extension 6, Paiement et gestion des fraudes - Ag-benda Pignozi :