Université de Mons
Faculté Des Sciences

Rapport de modélisation du projet de Génie Logiciel

S-INFO-015

Professeur :
Tom MENS
Assistants :
Jeremy DUBRULLE
Gauvain DEVILLEZ
Sébastien BONTE

Auteurs :
Augustin HOUBA
Arnaud MOREAU
Cyril MOREAU
François VION

Année académique 2021-2022

Table des matières

Chapitre 1

Partie commune

1.1 Introduction et choix de conception des diagrammes de conception UML

1.1.1 Use case diagrams

1.1.2 Interaction overview diagrams

En se basant sur le *use case diagram*, il est possible d'établir un *interaction overview diagram*, décrivant plutôt la structure et le *flow* général d'exécution du programme. Ce diagramme ne va pas dans les détails de chaque cas d'utilisation (cette tâche est réservées à chacun des *sequence diagrams* associés à chaque *use case*) mais décrit plutôt le chemin que l'utilisateur peut suivre durant l'exécution de l'application, en faisant usage des fonctionnalités mises à disposition. Par exemple, au démarrage de l'application, l'utilisateur peut se connecter, afficher la liste de ses portefeuilles, afficher l'historique d'un compte et l'exporter au format JSON, puis sélectionner un nouveau portefeuille et le désactiver, et enfin se déconnecter de l'application avant de fermer cette-dernière. Il est possible de suivre ce chemin d'exécution dans l'*interaction overview diagram*, et si l'on souhaite obtenir plus de détails sur le fonctionnement de l'exportation de l'historique, on peut suivre la référence au *sequence diagram* associé.

Deux *interaction overview diagrams* ont été réalisés. Le premier décrit l'application client, tandis que le second décrit l'application institution financière. Les deux applications sont chacunes divisées en deux parties, à savoir :

1. Application client
   (a) L'authentification ;
   (b) L'accès à l'application en tant que client.
2. Application institution
   (a) L'authentification ;
   (b) L'accès à l'application en tant qu'institution.

Comme on peut le voir, chacune des application utilise le système d'authentification avec les fonctionnalités de connexion, création de compte, etc. Ensuite, chaque application aura des fonctionnalités bien distinctes.

Évidemment, si l'on crée un compte client, il ne faudra pas entrer les mêmes informations que pour un compte institution, mais le flux d'exécution reste le même.

Authentification Cette première partie est commune aux deux applications. Elle est accessible à tous, même aux personnes qui ne sont pas enregistrées dans la base de données ou pour qui l'application n'est pas destinée. Cette partie ne communique avec la base de données que lors de l'inscription, afin de vérifier si, par exemple, l'adresse email n'est pas déjà enregistrée dans cette-dernière, ou afin d'enregistrer un nouvel utilisateur. La seule table à laquelle l'application accèdera durant l'exécution de cette partie est la table *Users*, ce qui permet d'empêcher d'accéder aux tables contenant des informations sensibles sans être connecté à son compte.
Lors de l'ouverture de l'application, si aucune connexion n'a été enregistrée (c'est-à-dire si on accède à l'application sans s'être précédemment connecté), l'utilisateur arrive sur la fenêtre d'authentification. Depuis celle-ci, il peut effectuer plusieurs actions :

1. Changer la langue ;
2. Se connecter ;
3. Créer un compte.

Quitter l'application *est considéré comme un comportement trivial.*

Choix de conception Les deux diagrammes se basent sur deux points centraux, représentés par deux grands diamants : *Authentification* et *Accès à l'application en tant que client/institution*. L'*accès à l'application en tant que client/institution* est ce qui différentie le diagramme de l'application client de celui de l'application institution. Chaque application se base sur le point central qui lui correspond, à partir duquel l'utilisateur peut prendre toutes les déicisions qu'il désire afin d'utiliser à sa guise les différentes fonctionnalités mises à sa disposition. Cette structure a été choisie puiqsu'elle concorde avec la structure qui sera implémentée graphiquement : un menu principal à partir duquel l'utilisateur peut accéder à différentes catégories, fonctionnalités... Typiquement, une flèche de *Control flow* quittant le diamant *Accès à l'application en tant que client* correspondra au clic d'un des boutons du menu principal (voir *User Interface diagrams*), mis à part la fermeture de l'application. Par exemple, pour l'application client, *Afficher les notifications* sera atteint si l'utilisateur décide d'afficher ses notifications via le bouton disponible sur l'écran principal de l'application. Un autre exemple pour l'application institution serait *Accéder à la liste des demandes*. Ce point du flot d'exécution serait atteint si l'utilisateur cliquait sur le bouton lui permettant d'accéder aux demandes faites au près de l'institution.

1.1.3 Class diagrams

Les diagrammes de classes sont divisés en trois parties : la partie logique, la partie API et la partie GUI (Interface graphique). L'ensemble des diagrammes de classes ont été pensé afin de pouvoir utiliser des classes ou des concepts identique pour les deux applications.

1.1.4 Sequence diagrams

L'ensemble des diagrammes de séquence du projet se veulent simplifiées et plus abstraits que le programme réel afin de fournir une meilleur lisibilité et compréhension.

1.2 Diagrammes de conception UML : application client

1.2.1 Use case diagram

L'énoncé étant globalement assez clair sur les fonctionnalités demandées, très peu de décisions importantes ont dû être prises pour les diagrammes de cas d'utilisation. Généralement tout faisait partie des concepts basique d'une application (Ex : Réïnitialisation du mot de passe) ou alors était clairement spécifié dans l'énoncé. Le use case *Introduire une demande* est une généralisation de toutes les demandes qu'un utilisateur peur envoyer (Demande d'accès à ses comptes, demande de virements,...). La fonctionnalité n'a pas été plus détaillée dans ce diagramme pour plus de clarté. Les descriptions semi-structurées de chaque cas d'utilisations de ce diagramme sont dans un document PDF séparé afin de garantir une certaine clarté.

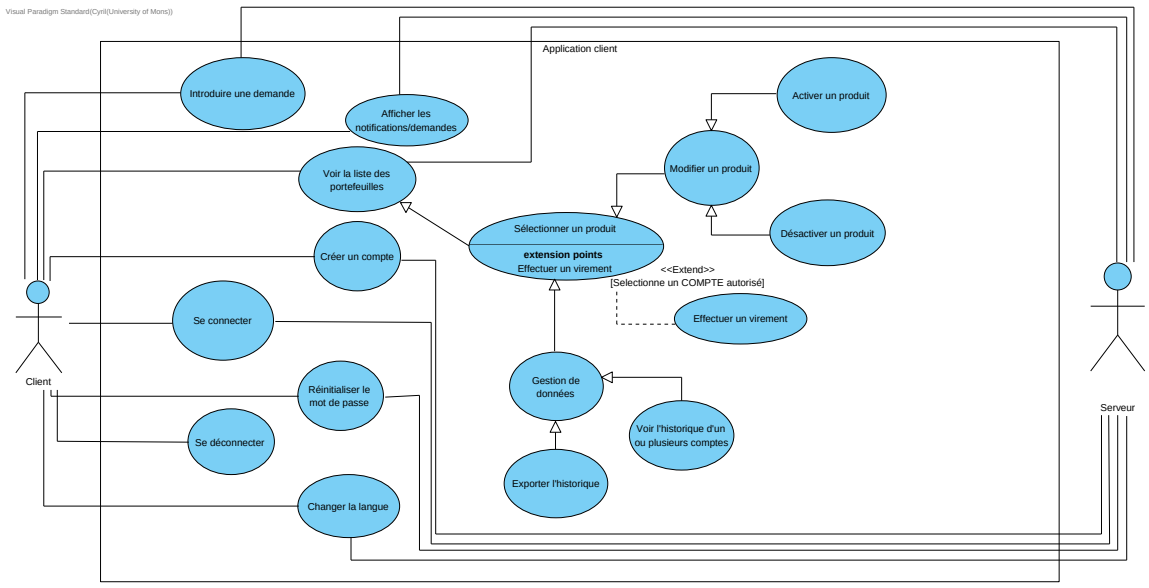


FIGURE 1.1 – Diagramme de case d'utilisation de l'application client.

1.2.2 Interaction overview diagram

Changer la langue L'utilisateur peut accéder à une interface lui permettant de changer sa langue préférée. Cela mettra à jour sa langue préférée dans la base de données et adaptera la langue de l'interface. Un menu déroulant affichera chaque fichier situé dans le répertoire dédié dont le nom respecte la forme *LL.json*, à savoir deux lettres déterminant la langue (par exemple *fr*, *en*, *de*, *nl*, *es*...) et au format JSON.

Se connecter L'utilisateur qui désire se connecter doit posséder un compte (pré-condition triviale). En entrant son nom d'utilisateur ou adresse email ainsi que son mot de passe, il peut s'identifier et accéder à son compte et ses portefeuilles.

Créer un compte L'utilisateur qui ne possède pas de compte et souhaite en créer un doit passer par l'interface de création de compte. Plusieurs informations sont requises : un nom, un prénom, un numéro national au format correct, une adresse email non utilisée et au format correct, un nom d'utilisateur libre, un mot de passe et sa confirmation, ainsi qu'une langue favorite. En confirmant qu'il a enregistré son mot de passe dans une base de donnée locale et sécurisée (comme un gestionnaire de mot de passe), nous encourageons l'utilisateur à protéger son mot de passe. Si toutes les informations sont remplies correctement, il pourra créer un compte et l'utliser pour se connecter à l'application.

Accès à application client, en tant que client La seconde partie n'est accessible qu'en se connectant avec les identifiants d'un compte enregistré dans la base de données. Lors de la connexion, l'utilisateur arrive sur l'écran principal, et peut, depuis celui-ci, accéder à toutes les fonctionnalités. Après s'être connecté, le client peut effectuer plusieurs actions :

1. Accéder à la liste de ses produits financiers
2. Introduire une demande

Quitter l'application, Changer la langue *et* Se déconnecter *sont considéré comme des comportements triviaux.*

Accéder à la liste de ses produits financiers La liste des produits financiers de l'utilisateur s'affiche à l'écran, et les fonctionnalités sont accessibles si l'utilisateur sélectionne un produit.

Sélectionner un produit Lorsque l'utilisateur a sélectionné un produit, de nouvelles fonctionnalités deviennent utilisables :

1. Modifier un portefeuille ;
2. Effectuer un virement ;
3. Voir l'historique d'un compte.

Modifier un portefeuille La modification du portefeuille permet à l'utilisateur de basculer (*toggle*) l'état du portefeuille sélectionner. Si le portefeuille est activé, l'utilisateur pourra le désactiver, et si le portefeuille est désactivé, l'utilisateur pourra le ré-activer. Après avoir basculé l'état du portefeuille, l'utilisateur est envoyé sur l'écran précédent, à savoir la sélection d'un portefeuille.

Effectuer un virement Si l'utilisateur souhaite effectuer un virement, il en a la possibilité en accédant à cette étape du flot d'exécution. Il faut entrer un montant et l'IBAN du destinataire ; le destinataire, le message et la date sont optionnels.

Voir l'historique d'un/plusieurs compte(s) Après avoir sélectionné un portefeuille, l'utlisateur peut demander à visualiser l'historique d'un des comptes associés à ce portefeuille de manière plus détaillée. Pour cela, deux techniques sont mises à sa disposition :

1. La visualisation en liste, qui affiche une liste détaillée des transactions d'un ou plusieurs comptes à intervalles de temps différentes ;
2. La visualisation graphique qui permet d'afficher des graphiques représentant ce même historique, mais de manière plus condensée et *user-friendly*. L'utilisateur peut aussi visualiser un ou plusieurs comptes, à intervalles de temps différentes.

Lorsque l'utilisateur sélectionne un portefeuille, il peut afficher les comptes associés à ce dernier. Il peut ensuite sélectionner un compte et visualiser l'historique des transactions de ce compte sous forme d'une liste ou de graphiques. Il peut aussi ajouter des comptes à visualiser avec le premier compte sélectionné.

Introduire une demande En retournant à l'accès à l'application en tant qu'utilisateur, le client peut aussi introduire une demande. Deux types de demandes sont supportés :

1. Demande de permission de virement : l'utilisateur peut demander à une institution la permission d'effectuer des virements depuis l'application ;
2. Demande d'ajout de produit financier : l'utilisateur peut demander à une institution la permission d'afficher des produits financiers dans l'application.

L'utilisateur peut aussi consulter l'état des demandes effectuées, à savoir : validée, refusée ou en attente.

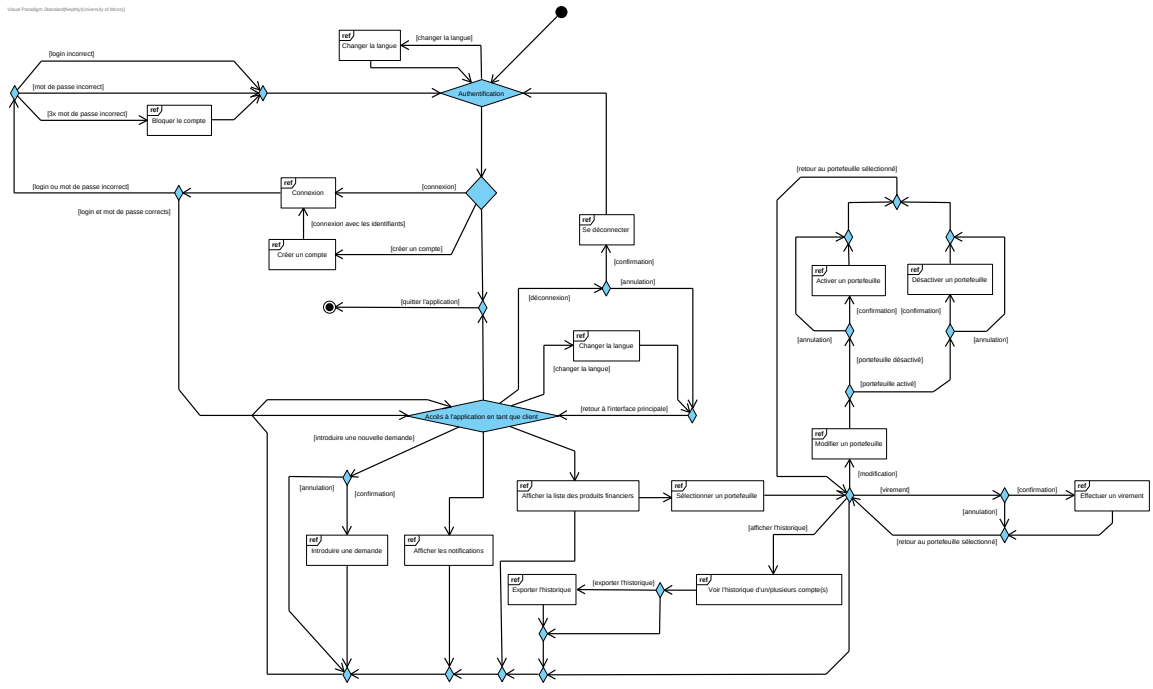


FIGURE 1.2 – Interaction overview diagram : application client

1.2.3 Class diagram

Partie logique La partie logique possède une partie commune aux 2 applications ainsi qu'une partie individuelle pour l'application banque.

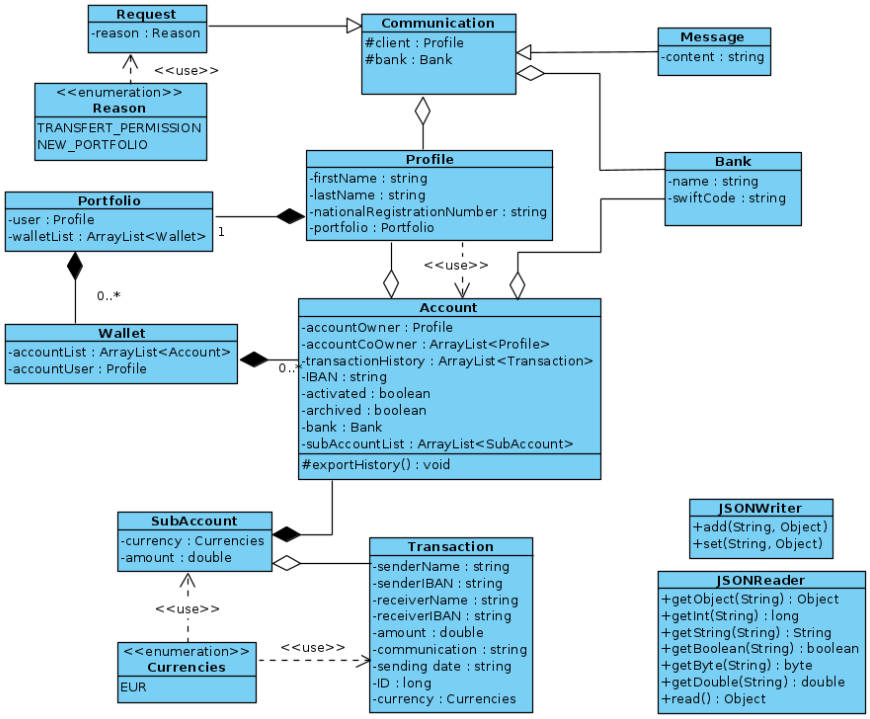


FIGURE 1.3 – Diagramme de classe de la partie logique commune

Partie commune La partie commune est composée des différents objets qui lient un client à une banque. La classe Profile est la plus utilisée, elle représente un client avec comme attributs le prénom, le nom, le numéro de registre national et son portfolio . Les banques possèdent aussi une classe pour les représenter. Il s'agit de la classe Bank qui a comme attribut un nom et un code Swift. L'utilisation de numéro de registre national et de code swift garantit que le client ou la banque est bien unique. Ensuite, il y a une classe Account qui représente un compte en banque, il possède comme attributs un titulaire, un co-titulaire, un historique de transaction, un IBAN (qui garantit l'unicité du compte), une banque chez laquelle il se trouve, une monnaie, deux booléens qui déterminent si le compte est activé et/ou archivé ainsi qu'une liste de sous compte. Ceux ci font partie de l'extension B mais ont été ajouté dans la partie commue afin de permettre à l'application de base d'intéragir avec la base de donnée qui elle, est commune. Les classes SubAccount et Currencies seront donc expliquées dans l'extension B. La classe Account possède aussi une méthode qui permet d'exporter l'historique de transactions. La classe Transaction possède tous les éléments d'une transaction bancaire, à savoir le nom et l'IBAN de l'envoyeur et du récepteur, le montant de la transaction, une communication, une date d'envoi, un ID et une monnaie. La classe Wallet représente l'ensemble des comptes d'un client chez une même banque. Elle est caractérisée par une liste de compte (objet Account) et

d'un utilisateur (objet Profile). La classe Portfolio représente l'ensemble des wallets d'une personne. L'une des dernières classe est la classe Communication. Elle sert à établir des communications entre le client et la banque. Une communication est caractérisée par un client (objet Profile) et une banque (objet Banque). Il s'agit d'une classe mère de 2 classes filles à savoir Request et Message. Request permet au client de créer une requête et à la banque d'y répondre, elle est caractérisée par une énumération de raisons et Message permet à la banque de transmettre un message au client. Enfin, les classes JSONWriter et JSONReader permettent d'écrire et de lire dans des fichiers JSON.

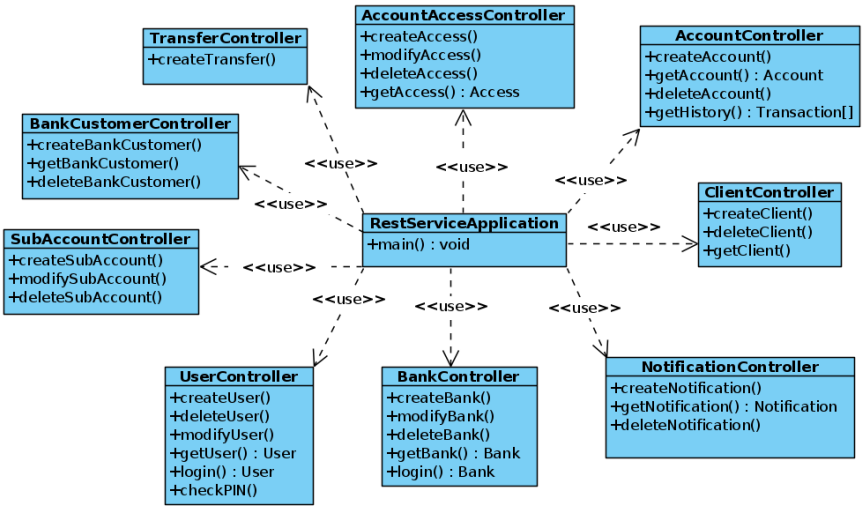


FIGURE 1.4 – Diagramme de classe de la API commune

Partie API La partie API est commune aux 2 applications et permet globalement d'ajouter des données dans la base de données, de les gérer, les supprimer ou les envoyer au client. Tout cela se fait via les classes associées au type de donnée (User, Bank, Account, AccountAccess,Transfer, Client,BankCustomer, SubAccount et Notification) et leur méthode create() qui ajoute, delete() qui supprime, modify() qui modifie et get() qui envoie les données au client.

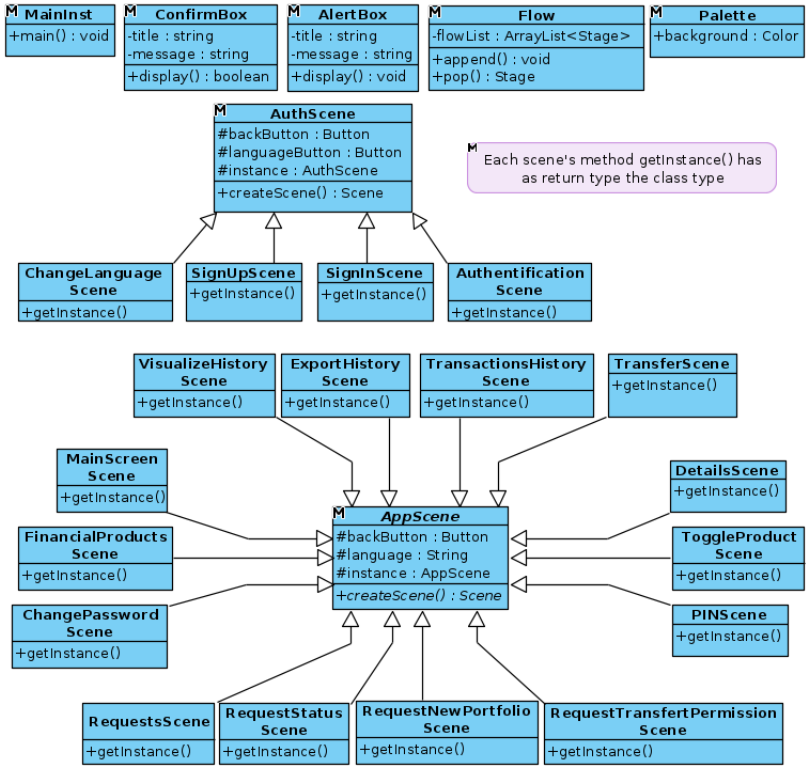


FIGURE 1.5 – Diagramme de classe de la partie GUI commune

Partie GUI La partie GUI est composée de toutes les classes de scenes ainsi que des classes “outil”. Les classes scenes ont toutes la même structure mais il en existe deux sortes. La première sorte est composée des scenes d’authentification (AuthScene) et la deuxième est composée des scenes de l’application une fois connecté (AppScene). La seule différence entre elles est la présence d’un bouton pour changer la langue dans les scenes d’authentification. Les classes abstraites AppScene et AuthScene ont en commun comme caractéristique un bouton de retour ainsi que la langue utilisé dans les fenêtres et comme méthode la méthode createScene qui crée le contenu de la page en fonction de la page sur laquelle on se trouve. Le design patern singleton a été utilisé pour les scenes. En effet, chaque classe Scene possède une méthode statique getInstance() qui permet de garantir qu’il n’existe qu’une instance de l’objet en cours d’utilisation. Pour les classes “outils”, on a ConfirmBox, AlertBox, Flow et Palette. AlertBox et ConfirmBox représentent des fenêtres d’alerte et de confirmation. Elles ont comme attribut un titre et un message et ont comme méthode la méthode display() qui affiche cette fenêtre et renvoie un booléen dans le cas de confirm box. La classe Flow représente une liste de toutes les pages parcourues et sert pour l’utilisation du bouton back, elle a comme attribut une liste d’élément Stage et comme méthode les méthodes append() et pop() qui servent à ajouter des éléments en fin de liste et les retirer en les lisants. La méthode Pallette sert pour avoir un fond d’écran de couleur uni défini par son attribut background. Enfin la classe Main sert à lancer l’application.

1.2.4 Sequence diagram

Les fonctionnements des actions possibles au sein des applications étant sensiblement les mêmes, ceux ci seront décrits en utilisant deux diagrammes de séquence “type” afin d’éviter de créer une multitude de diagrammes redondants. Cependant, certains fonctionnements sont plus particuliers et seront donc expliqués via des diagrammes de séquence uniques.

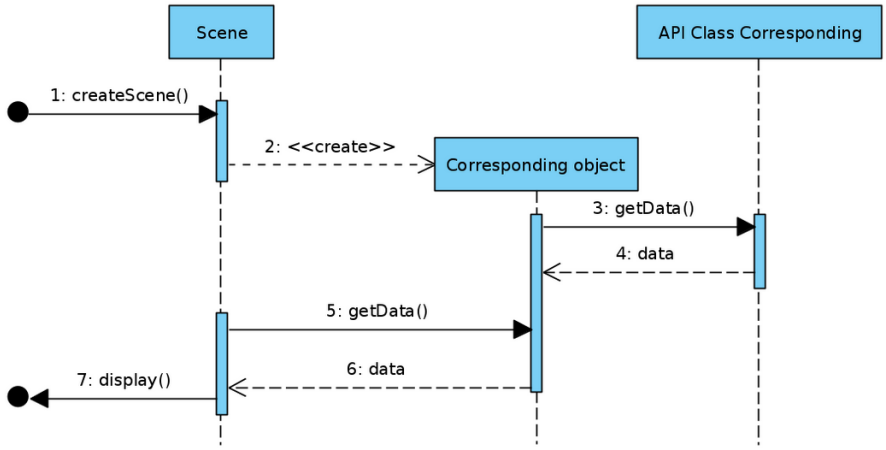


FIGURE 1.6 – Diagramme de séquence du premier fonctionnement type

Le premier comportement type commence par la création d’une scene particulière(1). Cette scene va instancier un objet correspondant au contexte de la scene (2) (Si on est sur une fenêtre de compte en banque cela créera un objet Account). Cet objet va obtenir les données qui le caractérise via la base de donnée. Cela se fait via une méthode get() de la classe API correspondant à l’objet (3 et 4) (Par exemple, la classe Account utilisera la classe AccountController). Enfin l’objet peut être utilisé afin d’afficher des données sur une scene(5,6 et 7).

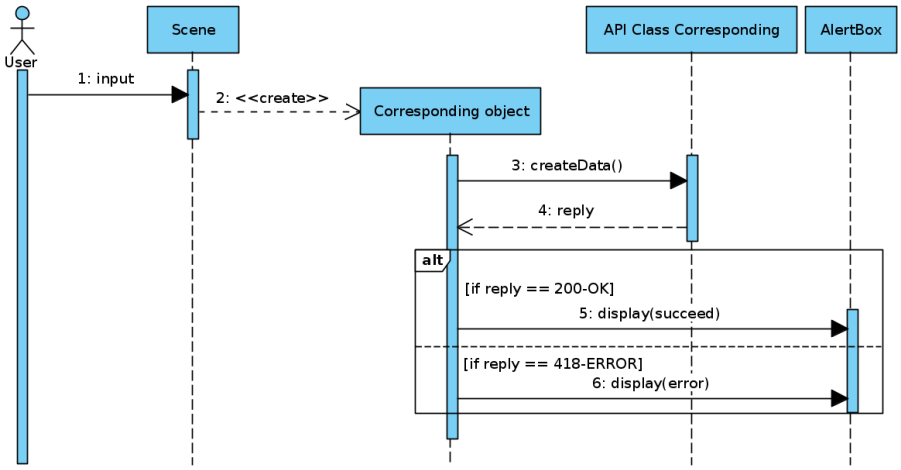


FIGURE 1.7 – Diagramme de séquence du deuxième fonctionnement type

Le deuxième comportement type possible commence par une entrée de l'utilisateur via l'interface graphique via une scene particulière (1). Ces entrées vont permettre de créer un objet correspondant au contexte de la scene (2). Celui ci sera utilisé afin d'ajouter des données à la base de données via un appel create() sur la classe de l'API correspondant (3). Cet appel renverra une réponse (4), soit 200-OK si l'opération s'est bien déroulée, soit 418-ERROR si il y a eu une erreur ou que les données entrées sont déjà existantes. Selon la réponse, une petite fenêtre viendra signaler à l'utilisateur si l'opération s'est bien déroulée ou non (5 et 6).

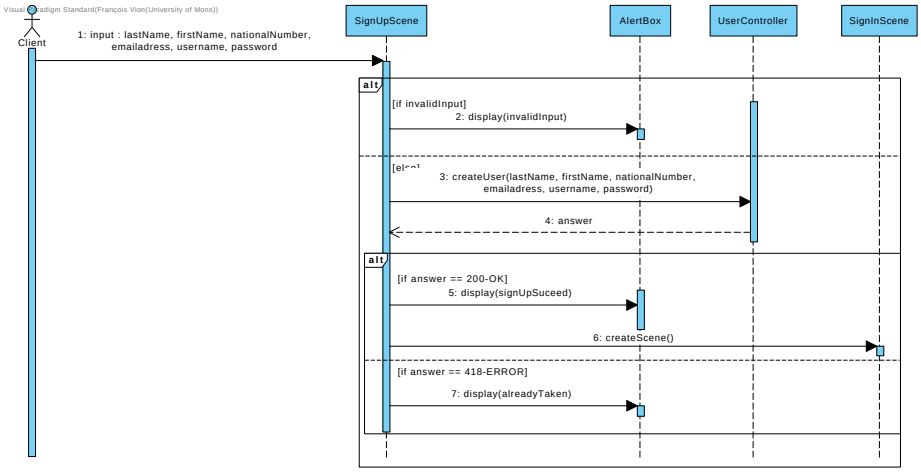


FIGURE 1.8 – Diagramme de séquence de la création d'un compte

La création d'un compte se fait grâce à la scene SignUpScene qui prend comme entrée le nom, le prénom, le numéro de registre national, l'adresse email, un pseudonyme et un mot de passe (1). S'ils sont invalides, une fenêtre AlertBox est créée afin de le signaler à l'utilisateur (2). S'ils sont valides, la scene fait un appel createUser() à la classe d'API UserController (3), prenant en paramètre les inputs rentrés à l'étape (1). UserController renvoie une réponse à cet appel (4), à savoir 200-OK ou 418-ERROR, comme dans le premier fonctionnement. Si la réponse est 200-OK, une fenêtre affiche que l'opération est réussie (5) puis la scene suivante est chargée (6). Si la réponse est 418-ERROR, une fenêtre informe l'utilisateur que l'une de ses données est déjà utilisée (7).

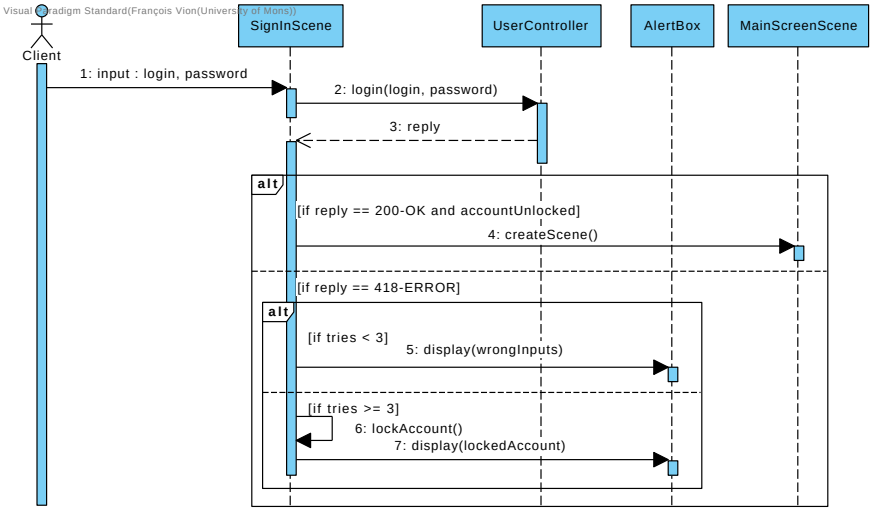


FIGURE 1.9 – Diagramme de séquence de la création d'un compte

La connexion se fait grâce à la fenêtre SignInScene qui reçoit les entrées login et password de l'utilisateur (1). La scene va faire un appel login() à la classe d'API UserController avec comme paramètre les entrées de l'utilisateur (2). UserController va envoyer une réponse à la scene (3) avec 200-OK si l'identifiant et le mot de passe sont correct et 418-ERROR si ils sont incorrects. Si la réponse est 200-OK et que le compte n'est pas bloqué, la connexion sera réussie et la prochaine scene sera chargée (4). Si la réponse est 418-ERROR et que l'utilisateur a fait moins de 3 essais incorects, une fenêtre affichera que les entrées sont incorrectes grâce à la classe AlertBox (5). Si c'est le 3eme essai ou plus, le compte sera bloqué (6) et une fenêtre affichera que le compte est bloqué grâce à la classe AlertBox (7).

1.3 Diagrammes de conception UML : application institution

1.3.1 Use case diagram

Contraiement à l'application présentée précédemment, cette application est utilisée par une institution financière et, comme pour l'application 1, très peu de décision ont été prise car tout était assez clair dans l'énoncé. Notons cependant que le cas d'utilisation *Gérer les demandes* permet à l'utilisateur d'approuver ou de refuser une demande mais il devra effectuer les demandes manuellement (par exemple si la demande est d'ajouter un produit financier, il devra l'ajouter manuellement et ensuite accepter la demande.)
La description semi-détaillée de chaque cas d'utilisation de ce diagramme est également dans un PDF attaché à ce rapport.

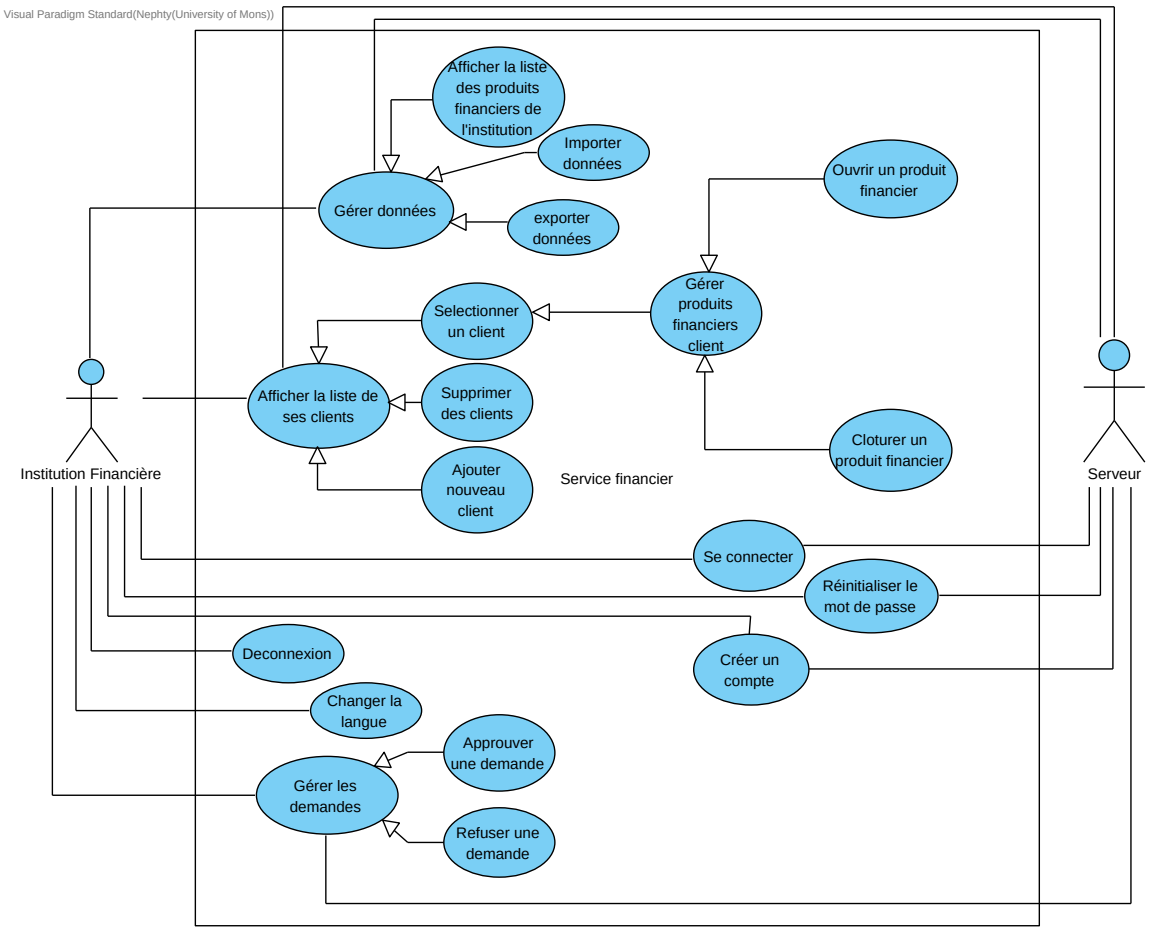


FIGURE 1.10 – Diagramme de case d'utilisation de l'application pour un service financier.

1.3.2 Interaction overview diagram

Accès à application institution financière, en tant qu'institution financière La seconde partie n'est accessible qu'en se connectant avec les identifiants d'un compte enregistré dans la base de données. Lors de la connexion, l'utilisateur (le responsable de l'institution, l'économe, le gestionnaire...) arrive sur l'écran principal, et peut, depuis celui-ci, accéder à toutes les fonctionnalités. Après s'être connecté, l'utilisateur peut effectuer plusieurs actions :

1. Afficher la liste des clients de l'institution ;
2. Gérer les données de l'institution ;
3. Accéder à la liste des demandes faites auprès de l'institution.

Quitter l'application, Changer la langue *et* Se déconnecter *sont considéré comme des comportements triviaux.*

Afficher la liste des clients de l'institution L'utilisateur peut accéder à la liste des clients et peut ajouter un client, exporter les données de ses clients ou en sélectionner un afin d'accéder aux détails.

Sélectionner un client Après avoir sélectionné un client, l'utilisateur peut supprimer le client de la base de données, clôturer un compte ou en ouvrir un nouveau (de n'importe quel type). *Ces trois comportements sont considérés comme triviaux.*

Ajouter un client L'utilisateur peut ajouter un nouveau client en spécifiant les informations nécéssaires.

Exporter les données de ses clients (implémente le tri et la recherche) Lorsque l'utilisateur effectue une recherche ou un tri de ses clients, une liste différente de clients sera affichée à l'écran. Il pourra alors exporter cette liste "personnalisée" au format JSON ou CSV, qui contiendra les informations à disposition de la banque à propos de ses clients.

Gérer les données de l'institution Grâce à cette fonctionnalité, l'utilisateur connecté peut importer des données ou exporter toutes les données de tous les clients.

Importer des données L'import de données peut servir, par exemple, à restaurer une liste de client dans la base de données pour la mettre à disposition de la banque via l'interface graphique. L'utilisateur peut choisir un fichier au format JSON qui sera lisible par l'application.

Exporter toutes les données de tous les clients L'export de toutes les données des clients permet d'exporter toutes les données en un clic ; c'est un moyen alternatif, plus rapide, mais qui ne permet pas le tri ou la recherche de mots clés.

Accéder à la liste des demandes L'utilisateur peut, finalement, accéder aux demandes, qui sont de deux types : demandes de permission de transferts et demande d'ouverture de compte. En sélectionnant l'une ou l'autre catégorie, il pourra approuver ou refuser une demande.

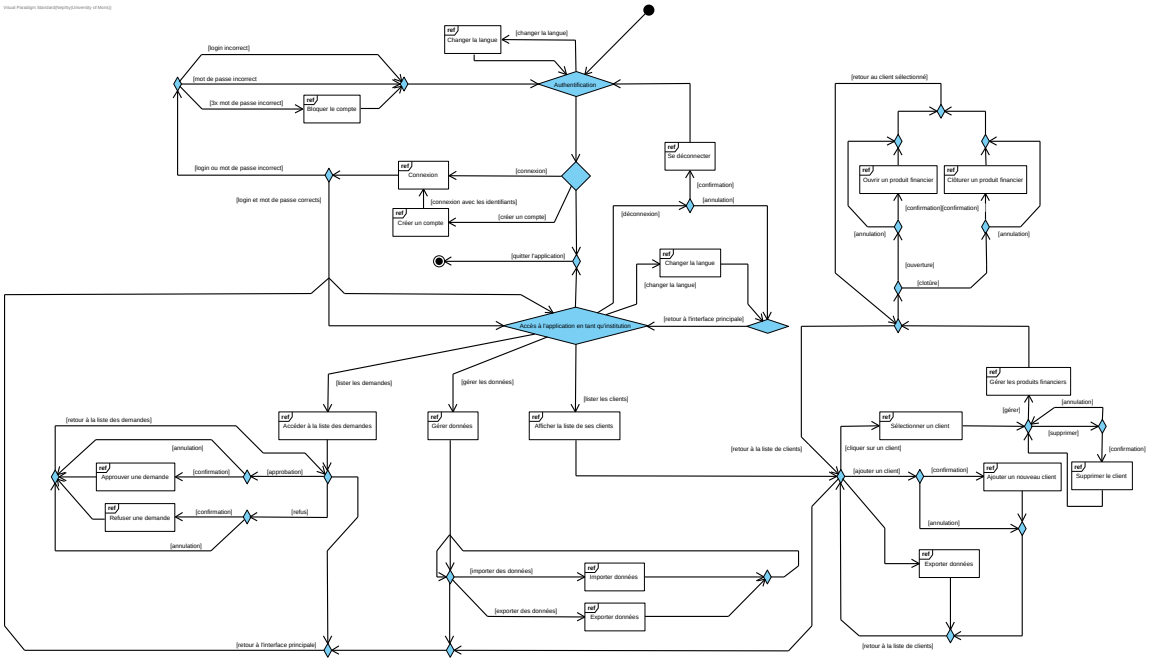


FIGURE 1.11 – Interaction overview diagram : application institution

1.3.3 Class diagram

Comme dit dans la partie diagramme de classe de l'application client, celui de l'application pour institution financières possède beaucoup de classes ou de concepts en commun avec celle ci.

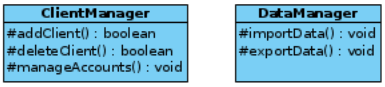


FIGURE 1.12 – Classes ajoutées dans la partie logique

Partie logique La partie banque ajoute 2 classes qui permettent de gérer les clients et les données à savoir ClientManager et DataManager. ClientManager permet grâce à ces méthodes d'ajouter des clients, d'en supprimer ou de gérer leur compte et DataManager permet d'importer des données et d'en exporter.

Partie API La partie API est exactement la même que l'application client

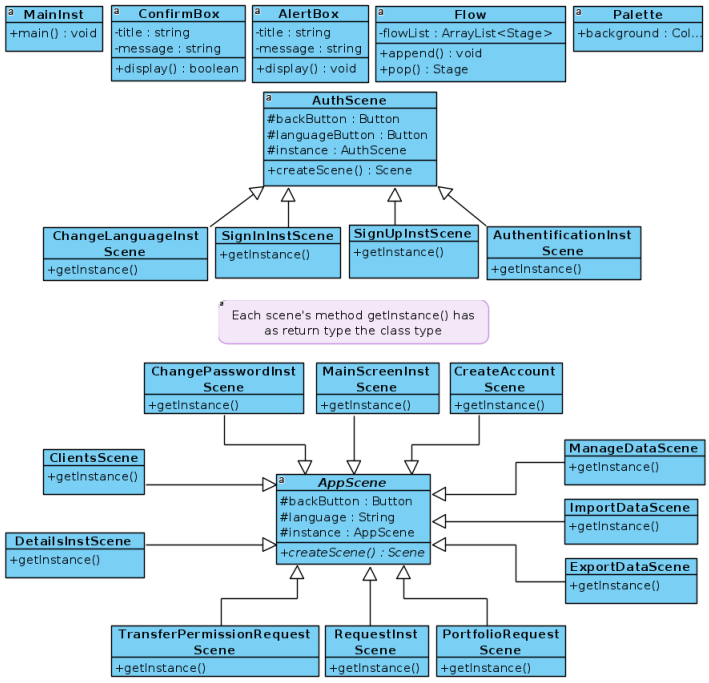


FIGURE 1.13 – Diagramme de classe de la partie GUI

Partie GUI La partie GUI s'articule de la même manière que celle pour l'application client. On a les classes AuthScene et AppScenes qui sont des classes mères de toutes les autres scenes ainsi que les classes ConfirmBox, AlertBox, Flow et Palette qui sont en commun avec le GUI de la partie client. Le mot clé "Inst" a parfois été ajouté au nom des classes qui ont le même nom que celles de l'application client mais qui ne sont pas les mêmes.

1.3.4 Sequence diagram

Introduction En ce qui concerne les diagrammes de séquence de l'application pour les institutions. Nous avons fait le choix pour un bon nombre d'entre eux de ne pas les réaliser pour l'une de deux raisons

1. Le diagramme est considéré comme trivial comme expliqué dans la partie client
2. Le diagramme a un équivalent dans la partie client et serait donc redondant

Nous avons donc réalisé les diagrammes de séquence des interactions les plus complexes.

Accéder à la liste des demandes Cette interaction permet à la banque d'accéder à la liste des demandes que ses utilisateurs ont postés. Elle démarre à la création de la scène RequestScene (1) qui va réaliser un appel à l'API avec getNotification en précisant le numéro SWIFT de la banque dont l'on veut récupérer les notifications(2). Le serveur répondra avec une liste de notifications(3). Une fois que l'application a reçu les données, elle affichera la liste avec la méthode display(4).

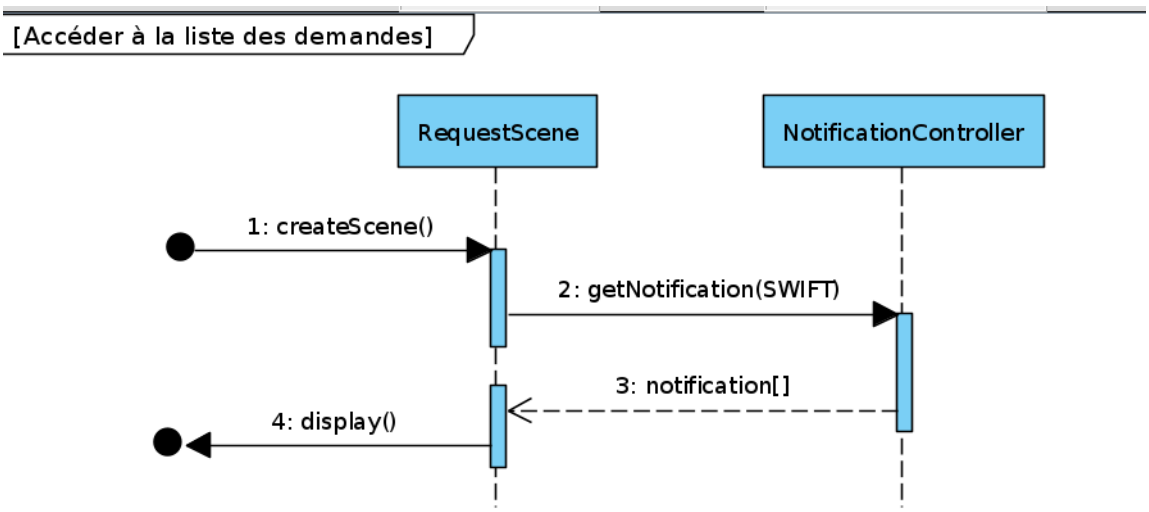


FIGURE 1.14 – Accéder à la liste des demandes

Approuver une demande Lorsqu'un objet Request recoit une demande d'approbation(1), il enverra à l'API une demande pour créer une notification afin d'informer l'utilisateur qui a introduit la requête que sa demande a été acceptée(2). Ensuite elle enverra une 2e demande à l'API pour cette fois si delete cette request du serveur(3).

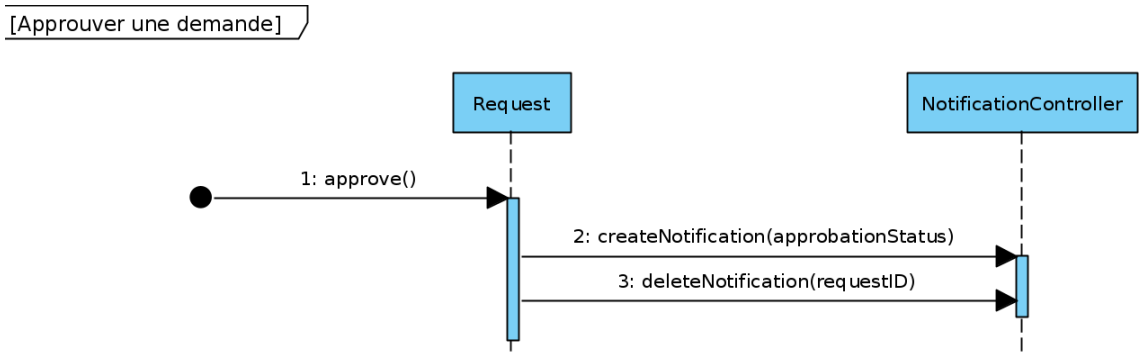


FIGURE 1.15 – Approuver une demande

Ouvrir un produit financier L'interaction commence quand la banque entre les donnnés du produit financier qui sont : clientName, clientID, IBAN, accountType (1). Lorsqu'il actionnera le bouton pour créer le produit, la scène fera un appel à l'API pour créer le produit(2). Si les données reçues par le WalletController sont correctes, Une notification sera créer pour informer le client que son produit a été ouvert(4). Ensuite la scène informera la banque que la requête a été effectuée avec succès(5). Si les données reçues par le WalletController sont incorrectes, l'application affichera un message d'erreur(6).

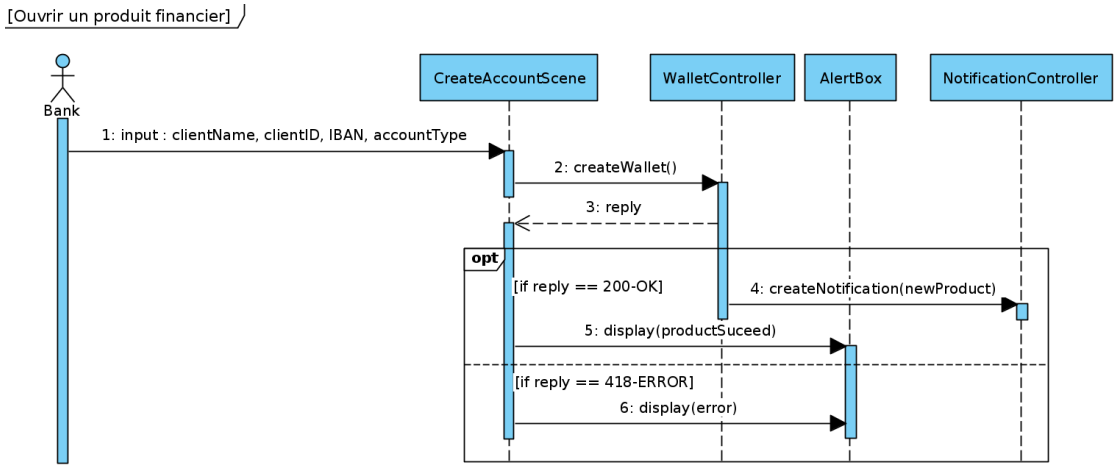


FIGURE 1.16 – Ouvrir un produit financier

1.4 Modèle de données

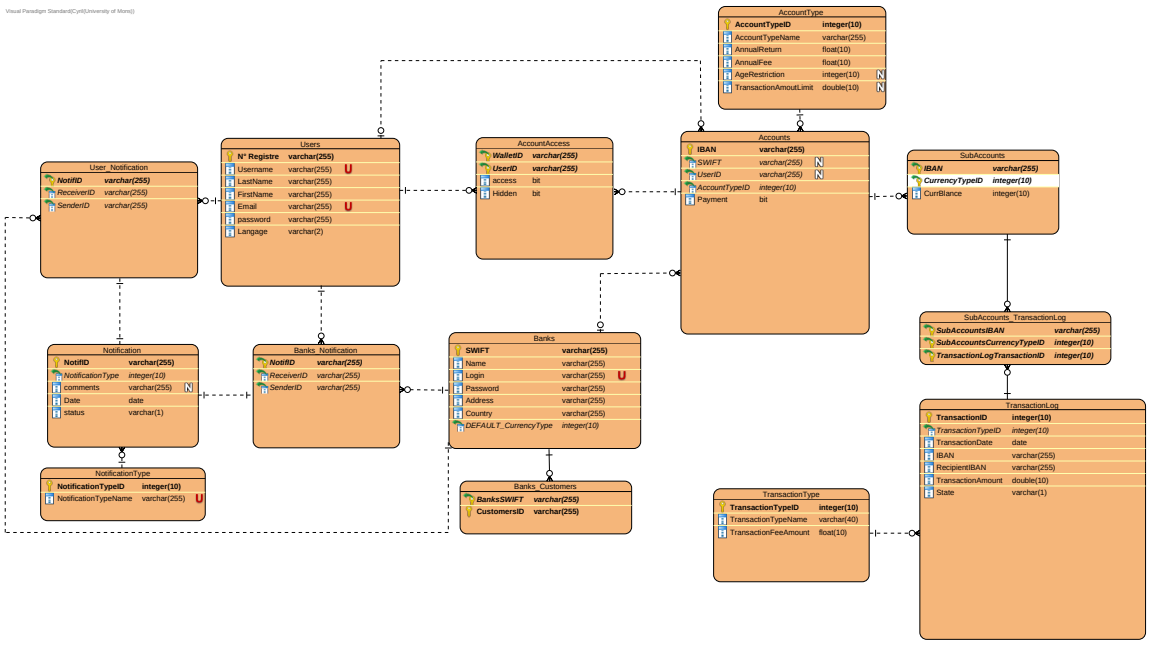


FIGURE 1.17 – Diagramme d'entite relation

Tout d'abord, certaines parties de ce modèle de données sont présentes car toutes les application (commune + extension) se basent sur le même modèle de données. L'application de base à donc été adaptée afin de respecter chaque extension. Vous trouvez la base de donnée entière dans un pdf attaché à ce rapport.

Avec ce modèle de donnée, nous avons essayé de couvrir au maximum les besoins de l'application tout en gardant un schéma assez compréhensible. Les deux entités *USER* et *BANKS* sont les entités centrale du schéma, elle permettent de gérer les données des utilisateurs ainsi que des banques. Chaque user est identifié grace à son numéro de registre nationnal et chaque banque est identifiée grâce à son numéro SWIFT.
L"entité *Accounts* permet de stocker tous les comptes de l'application. Il existe plusieurs type de compte où certains ont un rendement annuel, l'entité *AccountType* permet de les différenciers. Veuillez noter que les spécificités de chaque compte sera commun à toutes les banque par soucis de facilité. Un compte est divisé en plusieurs sous-comptes permettant de faciliter l'implémentations de certains extension mais dans cette vesrion, un compte ne possèdera qu'un seul Subaccount. L'entité *AccountAccess* permet, quant à elle, de donner les accès à un utilisateur à un compte. Elle permet donc à un utilisateur d'être co-titulaire c'est à dire d'avoir accès à un compte sans être le titulaire de celui-ci.
Plusieurs entités sont utilisées pour les notification. Chaque notification est identifiée par un *NotificationID* qui permet de connaitre tous les informations concernant cette notification (par exemple le type de notification ou l'état de celle-ci). Deux entités sont créée, permettant de différencier les notifications des clients et les notifications des banques.
L'entité *TransactionLog* permet de stocker l'historique des transaction effectuées ainsi que celles à effectuer. L'etats des transaction est connu grâce à l'attribut *status*. Remarquez également que nous ne connaissons pas

le nouveau montant du compte après la transaction. Nous avons décidé de calculer les montants en ajoutant ou diminuant le montant des transactions au montant actuel du compte. Cela prendra effectivement plus de temps à calculer cependant cela permet de rendre la table TransactionLog plus concise et permet aussi d'éviter certaines erreurs.

1.5 Design du REST API

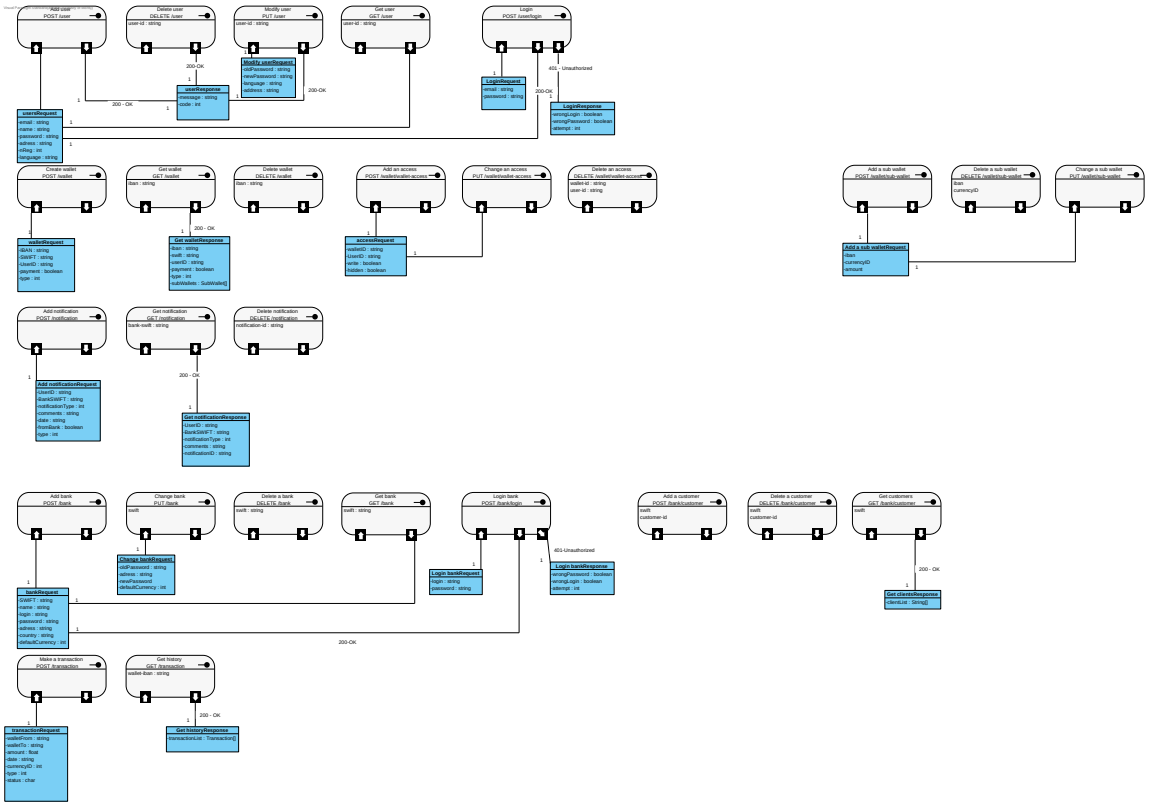


FIGURE 1.18 – Diagramme de l'API

Introduction Ce diagramme décrit le fonctionnement de l'api. Nous l'avons réalisé en se basant sur le use case diagramme ainsi que celui de relation-overview. On s'est permis quelques simplifications comme lorsqu'une réponse n'est constitué que d'un code elle est considérée comme triviale et n'est donc pas représenté. En revanche si le corps de la réponse est plus complet (notamment pour les méthodes de login) il sera précisé.

4 méthodes HTTP sont utilisées dans ce schéma :

1. **GET** : Méthode utilisée lorsqu'il est nécessaire d'accéder à des données de la BDD. Elle prendra en général comme paramètre(s) le(s) clé(s) primaire de la table ou la recherche est effectuée.
2. **POST** : Méthode utilisée lorsqu'il est nécessaire de rajouter des données dans la BDD. Elle requiert en général les attributs de la table ou la création se fera.
3. **PUT** : Méthode utilisée lorsqu'il est nécessaire de modifier des données dans la BDD. Elle prend en paramètre le(s) clé(s) primaire de la table ou la modification est effectuée ainsi que le(s) paramètre(s) à modifier. Si ils sont plusieurs, ils seront généralement optionnel et seulement ceux précisé seront modifiés.

4. **DELETE** : Méthode utilisée lorsqu'il est nécessaire de supprimer des données dans la BDD. Elle prendra en général comme paramètre(s) le(s) clé(s) primaire de la table ou la suppression est effectuée.

\user

1. **GET** : Elle retournera une instance de User. Le paramètre user-id est requis.
2. **POST** : Cette méthode est employée lors de la création d'un nouveau compte.
3. **PUT** : Elle permet de modifier la langue de préférence de l'utilisateur ou le mot de passe La méthode s'adaptera en fonction des données fournies. Soit le paramètre language aura été spécifié soit les paramètres oldPassword et newPassword.
4. **DELETE** : Le paramètre user-id est requis cette méthode supprime simplement un user.

\user\login

1. **GET** : Lorsqu'un user se connecte si les crédentiels entrés sont correct, elle renvoie le profil sinon elle renvoie des informations sur ce qui s'est mal déroulé.

\account

1. **GET** : Renvoie une instance d'un account ainsi qu'une liste de tous ses sub-accounts
2. **POST** : Utile pour la création d'un nouveau compte. Elle créera notamment un sub-account avec la currency par défaut de la banque associée.
3. **DELETE** : Suppression d'un compte ainsi que les sub-account et account-access associés

\account\account-access

1. **POST** : Création d'un nouvel accès.
2. **PUT** : Modifier le statut d'un accès le paramètre write ou hidden sera spécifié. accountID et userID correspondent aux clés primaires de la table.
3. **DELETE** : account-id et user-id correspondent aux clés primaires de la table.

\account\sub-account

1. **POST** : Crée un nouveau sub-account
2. **PUT** : Modifie la balance dans un account. Le paramètre amount reprrésente ce qu'il faut rajouter, il peut être négatif.
3. **DELETE** : Suppression d'un sub-account. Les paramètres sont les clés primaires de la table.

\notification

1. **GET** : Renvoie la notification correspondant à l'id
2. **POST** : Crée une notification. Le paramètre fromBank indique si le destinateur et la banque ou l'utilisateur ce qui sera utile pour savoir dans quelle table rajouter la notification.
3. **DELETE** : Suppression de la notification correspondant à l'id

\bank

1. **GET** : Retourne une instance de banque.
2. **POST** : Création d'une banque.
3. **DELETE** : Suppression d'une banque, swift est la clé primaire de la table.
4. **PUT** : Les paramètres modifiables sont password, adress, country et defaultCurrency

\bank\login

1. **GET** : Lorsqu'une banque se connecte si les crédentiels entrés sont correct, elle renvoie le profil sinon elle renvoie des informations sur ce qui s'est mal déroulé.

\bank\customer

1. **GET** : Renvoie la liste de tous les clients appartenant à la banque dont le swift eest passé en paramètre.
2. **POST** : Ajoute un nouveau client.
3. **DELETE** : Supprime le client (customer-id) de la bank (swift). Les paramètres sont les clés primaires.

\transaction

1. **GET** : Renvoie une liste des transactions d'un portefeuille qu'il en soit l'émetteur ou le receveur
2. **POST** : Crée une nouvelle transaction.

1.6 Maquette de l'interface utilisateur : application client

Fenêtres disponibles

- Auth ;
- Sign in ;
- Sign up ;
- Change password ;
- Change language ;
- Main screen ;
- Financial products ;
- Toggle product ;
- Enter PIN to confirm ;
- Details ;
- Transaction history ;
- Visualize history ;
- Export history ;
- Transfer ;
- Requests ;
- Request transfer permission ;
- Request new portfolio ;
- Requests status ;
- Notifications.

Fenêtre *Auth*

Accès : en ouvrant l'application sans être connecté, en cliquant sur les boutons *Back* des fenêtres *Sign in* et *Sign up* ou en cliquant sur le bouton *Sign out* de la fenêtre *Main screen*.
Contenu : les boutons *Sign in*, *Sign up* et *Language*.

- Le bouton *Sign in* : envoie l'utilisateur sur la fenêtre *Sign in* ;
- Le bouton *Sign up* : envoie l'utilisateur sur la fenêtre *Sign up* ;
- Le bouton *Language* : envoie l'utilisateur sur la fenêtre *Change language*.

Fenêtre *Sign in*

Accès : en cliquant sur le bouton *Sign in* de la fenêtre *Auth* ou en créant un compte sur la fenêtre *Sign up*.
Contenu : les boutons *Back*, *Sign in* et *Language*, le champ de texte *Username or email...* et le champ de mot de passe *Password...* .

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Auth* ;
- Bouton *Sign in* : si les identifiants sont corrects, connecte l'utilisateur et l'envoie sur la fenêtre *Main screen*, sinon, affiche une erreur ;
- Le bouton *Language* : envoie l'utilisateur sur la fenêtre *Change language* ;
- Le champ de texte *Username or email...* : permet à l'utilisateur d'entrer son nom d'utilisateur ou son email ;
- Le champ de mot de passe *Password...* : permet à l'utilisateur d'entrer son mot de passe de manière discrète.

Fenêtre *Sign up*

Accès : en cliquant sur le bouton *Sign up* de la fenêtre *Auth*.

Contenu : les boutons *Back*, *Sign up* et *Language*, les champs de texte *Last name...*, *First name...*, *NRN...*, *Email address...*, *Username...*, les champs de mot de passe *Password...* et *Confirm password...*, le label *Favorite language*, le menu déroulant *Language* et la case à cocher *Confirmation* .

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Auth* ;
- Bouton *Sign up* : si l'adresse email est au format correct, l'adresse email et le nom d'utilisateur ne sont pas déjà pris par un autre utilisateur, le numéro national est correctement formatté, le mot de passe est confirmé et la case à cocher est cochée, enregistre un nouveau profil dans la base de données et envoie l'utilisateur sur la fenêtre *Sign in* avec le champ de texte *Username or email...* pré-complété ;
- Le bouton *Language* : envoie l'utilisateur sur la fenêtre *Change language* ;
- Le champ de texte *Last name...* : permet à l'utilisateur d'entrer son nom ;
- Le champ de texte *First name...* : permet à l'utilisateur d'entrer son prénom ;
- Le champ de texte *NRN...* : permet à l'utilisateur d'entrer son numéro national ;
- Le champ de texte *Email address...* : permet à l'utilisateur d'entrer son adresse email ;
- Le champ de texte *Username...* : permet à l'utilisateur d'entrer un nom d'utilisateur ;
- Le champ de mot de passe *Password...* : permet à l'utilisateur d'entrer un mot de passe de manière discrète ;
- Le champ de mot de passe *Confirm password...* : permet à l'utilisateur d'entrer la confirmation du mot de passe de manière discrète ;
- Le menu déroulant *Language* : permet à l'utilisateur de sélectionner sa langue préférée ;
- La case à cocher *Confirmation* : demande à l'utilisateur de confirmer qu'il a sauvegardé son mot de passe de manière locale, sécurisée et permanente.

Fenêtre *Change password*

Accès : en cliquant sur le bouton *Change password* de la fenêtre *Main screen*.
Contenu : les boutons *Back*, *Change password* et *Language* et les champ de mot de passe *Current password*, *New password* et *Confirm new password* et la case à cocher *Confirmation*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;
- Le bouton *Change password* : change le mot de passe lié à l'utilisateur si la case à cocher est cochée, que l'ancien mot de passe est correct et que la confirmation du nouveau mot de passe est correcte ;
- Le bouton *Language* : envoie l'utilisateur sur la fenêtre *Change language* ;
- Le champ de mot de passe *Current password* : permet à l'utilisateur d'entrer son mot de passe actuel de manière discrète ;
- Le champ de mot de passe *New password* : permet à l'utilisateur d'entrer le nouveau mot de passe désiré de manière discrète ;
- Le champ de mot de passe *Confirm new passsword* : permet à l'utilisateur d'entrer la confirmation du nouveau mot de passe de manière discrète ;
- La case à cocher *Confirmation* : demande à l'utilisateur de confirmer qu'il a mis à jour son mot de passe précédemment enregistré de manière locale, sécurisée et permanente.

Fenêtre *Change language*

Accès : en cliquant sur le bouton *Language* de la fenêtre *Auth*, *Sign in*, *Sign up*, *Reset password* ou *Main screen*.
Contenu : les boutons *Back*, *Add...* et *Confirm*, le label *Choose a language* et la liste *Available languages*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *précédente* ;

— Le bouton *Add...* : permet de sélectionner un fichier JSON correspondant à une langue afin de l'ajouter à la liste de langues disponibles ;
— Le bouton *Confirm* : envoie l'utilisateur sur la fenêtre *précente* et modifie la langue de l'application par la nouvelle langue sélectionnée ;
— La liste *Available languages* : contient une liste de toutes les langues disponibles et permet à l'utilisateur d'en sélectionner une.

Fenêtre *Main screen*

Accès : en se connectant à son compte de la fenêtre *Sign in*, en cliquant sur le bouton *Back* de la fenêtre *Financial products*, en cliquant sur les boutons *Change password* ou *Back* de la fenêtre *Change password*, en ayant entré trois fois un code PIN incorrect via la fenêtre *Enter PIN to confirm*, en cliquant sur le bouton *Back* de la fenêtre *Financial products* ou en cliquant sur le bouton *Back* de la fenêtre *Requests*.
Contenu : les boutons *Sign out*, *Language*, *Financial products*, *Requests* et *Change password*.
— Le bouton *Sign out* : envoie l'utilisateur sur la fenêtre *Auth* et déconnecte l'utilisateur ;
— Le bouton *Language* : envoie l'utilisateur sur la fenêtre *Change language* ;
— Le bouton *Financial products* : envoie l'utilisateur sur la fenêtre *Financial products* ;
— Le bouton *Requests* : envoie l'utilisateur sur la fenêtre *Requests* ;
— Le bouton *Change password* : envoie l'utilisateur sur la fenêtre *Change password*.

Fenêtre *Financial products*

Accès : en cliquant sur le bouton *Financial products* de la fenêtre *Main screen* ou en cliquant sur le bouton *Back* des fenêtres *Toggle product* et *Details*.
Contenu : les boutons *Back*, *Edit* et *Details*, le label *Choose a product* et la liste *Products*.
— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;
— La liste *Products* : affiche la liste des produits financiers associés au compte (c'est-à-dire ceux dont l'utilisateur est titulaire ou co-titulaire) ;
— Le bouton *Edit* : envoie l'utilisateur sur la fenêtre *Toggle product* ;
— Le bouton *Details* : envoie l'utilisateur sur la fenêtre *Details*.

Fenêtre *Toggle product*

Accès : en cliquant sur le bouton *Edit* de la fenêtre *Financial products*.
Contenu : les boutons *Back* et *Toggle* et le label *Toggle selected product*.
— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Financial product* ;
— Le bouton *Toggle* : envoie l'utilisateur sur la fenêtre *Enter PIN to confirm* et active/désactive le produit financier sélectionné selon son état actuel si la réponse de la fenêtre *Enter PIN to confirm* est positive (c'est-à-dire si le code PIN est correct).

Fenêtre *Enter PIN to confirm*

Accès : en cliquant sur le bouton *Toggle* de la fenêtre *Toggle product* ou sur le bouton *Confirm* de la fenêtre *Transfer* si le virement est effectué.
Contenu : les boutons *Back* et *Confirm*, un bouton par chiffre allant de 0 à 9, un bouton *Delete* et le label *Entered code*.
— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *précédente* ;
— Le bouton *Confirm* : renvoie à la fenêtre précédente avec une réponse positive si le code entré est bon, affiche une erreur si le code entré est mauvais et que l'utilisateur a fait moins de trois tentatives

erronées, ou renvoie à la fenêtre *Main screen* et bloque le compte si l'utilisateur a fait trois tentatives erronées ;
- Les boutons allant de 0 à 9 : entre le chiffre correspondant et modifie le label *PIN* ;
- Le bouton *Delete* (<) : supprime la dernière entrée du code PIN.

Fenêtre *Details*

Accès : en cliquant sur le boutons *Details* de la fenêtre *Financial products* ou en cliquant sur le bouton *Back* de la fenêtre *Transaction history*.
Contenu : Les boutons *Back*, *Transfer* et *History*, les labels *Choose an account* et *Product* et la liste *Accounts*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Financial products* ;
- Le bouton *Transfer* : envoie l'utilisateur sur la fenêtre *Transfer* ;
- Le bouton *History* : envoie l'utilisateur sur la fenêtre *Transaction history* ;
- La liste *Accounts* : affiche les comptes associés au produit financier précédemment sélectionné.

Fenêtre *Transaction history*

Accès : en cliquant sur le bouton *History* de la fenêtre *Details* ou en cliquant sur le bouton *Back* de la fenêtre *Export...*, en cliquant sur le bouton *Back* de la fenêtre *Visualize*, en cliquant sur les boutons *Export to JSON format* ou *Export to CSV format* de la fenêtre *Export history*.
Contenu : les boutons *Back*, *Visualize* et *Export history*, les labels *Account name* et *Account ID* et la liste *Transactions*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Details* ;
- Le bouton *Visualize* : envoie l'utilisateur sur la fenêtre *Visualize history*
- Le bouton *Export history* : envoie l'utilisateur sur la fenêtre *Export history* ;
- La liste *Transactions* : contient les transactions passées.

Fenêtre *Visualize history*

Accès : en cliquant sur le bouton *Visualize* de la fenêtre *Transaction history*.
Contenu : les boutons *Back*, *Export...*, *Add* et *Remove*, les labels *Product name*, *Product ID*, *Time scale* et *Add/remove acccount*, les menus déroulants *Time scale* et *Accounts* et la zone d'affiche des graphiques.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Transaction history* ;
- Le bouton *Export* : envoie l'utilisateur sur la fenêtre *Export history* et lui permet d'exporter l'historique avec une échelle de temps différente ;
- Le bouton *Add* : ajoute le produit sélectionné à visualiser ;
- Le bouton *Remove* : retire le produit sélectionné visualisé ;
- Le menu déroulant *Time scale* : permet à l'utilisateur de choisir une échelle de temps à utiliser pour les graphiques ;
- le menu déroulant *Accounts* : permet à l'utilisateur de sélectionner un produit à ajouter/enlever de la visualisation.

Fenêtre *Export history*

Accès : en cliquant sur le bouton *Export...* de la fenêtre *Transaction history* ou en cliquant sur le bouton *Export...* de la fenêtre *Visualize history*.
Contenu : les boutons *Back*, *Choose path...*, *Export to JSON format* et *Export to CSV format* et les labels *Export location* et *Selected path*

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Transaction history* ;
— Le boutton *Choose path...* : ouvre l'explorateur de fichier afin de déterminer le chemin de destination ;
— Le bouton *Export to JSON format* : exporte l'historique au format JSON ;
— Le bouton *Export to CSV format* : exporte l'historique au format CSV.

Fenêtre *Transfer*

Accès : en cliquant sur le bouton *Transfer* de la fenêtre *Financial products*.
Contenu : les boutons *Back* et *Confirm*, les labels *Amount*, *Recipient*, *IBAN*, *Message* et *Date* et les champs de texte *Amount*, *Recipient*, *IBAN*, *Message* et *Date*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Financial products* ;
— Le bouton *Confirm* : envoie l'utilisateur sur la fenêtre *Enter PIN to confirm* si le montant, l'IBAN et la date sont correctement formatés, et effectue le virement si la réponse de la fenêtre *Enter PIN to confirm* est positive ;
— Le champ de texte *Amount* : permet à l'utilisateur d'entrer la somme à envoyer ;
— Le champ de texte *Recipient* : permet à l'utilisateur d'entrer le destinataire ;
— Le champ de texte *IBAN* : permet à l'utilisateur d'entrer l'IBAN du destinataire ;
— Le champ de texte *Message* : permet à l'utilisateur d'entrer la communication liée au virement ;
— Le champ de texte *Date* : permet à l'utilisateur d'entrer la date de planification du virement.

Fenêtre *Requests*

Accès : en cliquant sur le bouton *Request* de la fenêtre *Main screen*, en cliquant sur le bouton *Back* de la fenêtre *Request transfer permission* ou en cliquant sur le bouton *Back* de la fenêtre *Request new portfolio*.
Contenu : les boutons *Back*, *Transfer permission*, *New portfolio* et *Requests status*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;
— Le bouton *Transfer permission* : envoie l'utilisateur sur la fenêtre *Request transfer permission* ;
— Le bouton *New portfolio* : envoie l'utilisateur sur la fenêtre *Request new portfolio* ;
— Le bouton *Requests status* : envoie l'utilisateur sur la fenêtre *Requests status*.

Fenêtre *Request transfer permission*

Accès : en cliquant sur le bouton *Transfer permission* de la fenêtre *Requests*.
Contenu : les boutons *Back* et *Send request*, le label *Select portfolio* et le menu déroulant *Portfolio*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Requests* ;
— Le bouton *Send request* : affiche une confirmation que la demande a bien été envoyée si un portfeuille a été sélectionné ;
— Le menu déroulant *Portfolio* : contient les portefeuilles qui ne possèdent pas déjà la permission d'effectuer des virements.

Fenêtre *Request new portfolio*

Accès : en cliquant sur le bouton *New portfolio* de la fenêtre *Requests*.
Contenu : les boutons *Back* et *Send request*, le label *Select SWIFT* et le menu déroulant *SWIFT*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Requests* ;
— Le bouton *Send request* : affiche une confirmation que la demande a bien été envoyée si un code SIWFT a été sélectionné ;
— Le menu déroulant *SWIFT* : contient les codes SWIFT existants dans la base de donnée.

Fenêtre *Requests status*

Accès : en cliquant sur le bouton *Requests status* de la fenêtre *Requests*.

Contenu : Le bouton *back* et la liste *Requests*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Requests* ;
- La liste *Requests* : affiche toutes les requêtes effectuées ainsi que leur status (validée, refusée, en attente).

Fenêtre *Notifications*

Accès : en cliquant sur le bouton *Notifications* de la fenêtre *Main screen*.

Contenu : le bouton *Back* et la liste *Notifications*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;
- La liste *Notifications* : contient les notifications de l'utilisateur.

Illustration du diagramme Par souci de clareté, ce diagramme est disponible en annexe du rapport.

1.7 Maquette de l'interface utilisateur : application institution

Fenêtres disponibles

- Auth ;
- Sign in ;
- Sign up ;
- Change password ;
- Change language ;
- Main screen ;
- Client ;
- Details ;
- Export data ;
- Create client account ;
- Requests ;
- Transfer permission requests ;
- Portfolio requests ;
- Manage data ;
- Import data.

Fenêtre *Auth*

Accès : en ouvrant l'application sans être connecté, en cliquant sur les boutons *Back* des fenêtres *Sign in* et *Sign up* ou en cliquant sur le bouton *Sign out* de la fenêtre *Main screen*.
Contenu : les boutons *Sign in*, *Sign up* et *Language*.

- Le bouton *Sign in* : envoie l'utilisateur sur la fenêtre *Sign in* ;
- Le bouton *Sign up* : envoie l'utilisateur sur la fenêtre *Sign up* ;
- Le bouton *Language* : envoie l'utilisateur sur la fenêtre *Change language*.

Fenêtre *Sign in*

Accès : en cliquant sur le bouton *Sign in* de la fenêtre *Auth* ou en créant un compte sur la fenêtre *Sign up*.
Contenu : les boutons *Back*, *Sign in* et *Language*, le champ de texte *SWIFT...* et le champ de mot de passe *Password...* .

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Auth* ;
- Bouton *Sign in* : si les identifiants sont corrects, connecte l'utilisateur et l'envoie sur la fenêtre *Main screen*, sinon, affiche une erreur ;
- Le bouton *Language* : envoie l'utilisateur sur la fenêtre *Change language* ;
- Le champ de texte *SWIFT...* : permet à l'utilisateur d'entrer le code SWIFT de la banque ;
- Le champ de mot de passe *Password...* : permet à l'utilisateur d'entrer le mot de passe du compte de la banque de manière discrète.

Fenêtre *Sign up*

Accès : en cliquant sur le bouton *Sign up* de la fenêtre *Auth*, les champs de texte *SWIFT...*, *City...*, *Country...*, *Name...* et *Password...*, les champs de mot de passe *Password...* et *Confirm password...*, le label *Favorite language*, le menu déroulant *Language* et la case à cocher *Confirmation*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Auth* ;

— Bouton *Sign up* : si le code SWIFT est au format correct et disponible, les champs de texte *City...*, *Country...* et *Name...* sont remplis, le mot de passe est confirmé, la langue favorite est sélectionnée et la case est cochée, enregistre un nouveau profil dans la base de données et envoie l'utilisateur sur la fenêtre *Sign in* avec le champ de texte *SWIFT...* pré-complété ;
— Le bouton *Language* : envoie l'utilisateur sur la fenêtre *Change language* ;
— Le champ de texte *SWIFT...* : permet à l'utilisateur d'entrer le code SWIFT de la banque ;
— Le champ de texte *City...* : permet à l'utilisateur d'entrer la ville dans laquelle la banque est basée ;
— Le champ de texte *Country...* : permet à l'utilisateur d'entrer le pays dans lequel la banque est basée ;
— Le champ de texte *Name...* : permet à l'utilisateur d'entrer le nom de la banque ;
— Le champ de mot de passe *Password...* : permet à l'utilisateur d'entrer un mot de passe de manière discrète ;
— Le champ de mot de passe *Confirm password...* : permet à l'utilisateur d'entrer la confirmation du mot de passe de manière discrète ;
— Le menu déroulant *Language* : permet à l'utilisateur de sélectionner sa langue préférée ;
— La case à cocher *Confirmation* : demande à l'utilisateur de confirmer qu'il a sauvegardé son mot de passe de manière locale, sécurisée et permanente.

Fenêtre *Change password*

Accès : en cliquant sur le bouton *Change password* de la fenêtre *Main screen*.
Contenu : les boutons *Back*, *Change password* et *Language* et les champ de mot de passe *Current password*, *New password* et *Confirm new password* et la case à cocher *Confirmation*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;
— Le bouton *Change password* : change le mot de passe lié à l'utilisateur si la case à cocher est cochée, que l'ancien mot de passe est correct et que la confirmation du nouveau mot de passe est correcte ;
— Le bouton *Language* : envoie l'utilisateur sur la fenêtre *Change language* ;
— Le champ de mot de passe *Current password* : permet à l'utilisateur d'entrer son mot de passe actuel de manière discrète ;
— Le champ de mot de passe *New password* : permet à l'utilisateur d'entrer le nouveau mot de passe désiré de manière discrète ;
— Le champ de mot de passe *Confirm new passsword* : permet à l'utilisateur d'entrer la confirmation du nouveau mot de passe de manière discrète ;
— La case à cocher *Confirmation* : demande à l'utilisateur de confirmer qu'il a mis à jour son mot de passe précédemment enregistré de manière locale, sécurisée et permanente.

Fenêtre *Change language*

Accès : en cliquant sur le bouton *Language* de la fenêtre *Auth*, *Sign in*, *Sign up*, *Reset password* ou *Main screen*.
Contenu : les boutons *Back*, *Add...* et *Confirm*, le label *Choose a language* et la liste *Available languages*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *précédente* ;
— Le bouton *Add...* : permet de sélectionner un fichier JSON correspondant à une langue afin de l'ajouter à la liste de langues disponibles ;
— Le bouton *Confirm* : envoie l'utilisateur sur la fenêtre *précente* et modifie la langue de l'application par la nouvelle langue sélectionnée ;
— La liste *Available languages* : contient une liste de toutes les langues disponibles et permet à l'utilisateur d'en sélectionner une.

Fenêtre *Main screen*

Accès : en se connectant à son compte de la fenêtre *Sign in*, en cliquant sur le bouton *Back* de la fenêtre *Clients*, en cliquant sur les boutons *Change password* ou *Back* de la fenêtre *Change password*, en cliquant sur le bouton *Back* de la fenêtre *Financial products* ou en cliquant sur le bouton *Back* de la fenêtre *Manage data*.

Contenu : les boutons *Sign out*, *Language*, *Clients*, *Manage requests*, *Manage data* et *Change password*.

- Le bouton *Sign out* : envoie l'utilisateur sur la fenêtre *Auth* et déconnecte l'utilisateur ;
- Le bouton *Language* : envoie l'utilisateur sur la fenêtre *Change language* ;
- Le bouton *Clients* : envoie l'utilisateur sur la fenêtre *Clients* ;
- Le bouton *Manage requests* : envoie l'utilisateur sur la fenêtre *Requests* ;
- Le bouton *Manage data* : envoie l'utilisateur sur la fenêtre *Manage data* ;
- Le bouton *Change password* : envoie l'utilisateur sur la fenêtre *Change password*.

Fenêtre *Clients*

Accès : en cliquant sur le bouton *Clients* de la fenêtre *Main screen*, en cliquant sur le bouton *Back* de la fenêtre *Details*, en cliquant sur le bouton *Back*, *Export to JSON format* ou *Export to CSV format* de la fenêtre *Export data* ou en cliquant sur le bouton *Back* de la fenêtre *Add client*.

Contenu : les boutons *Back*, *Search*, *Export data...*, *Add client* et *Details*, les labels *Choose a client*, *Search name or ID*, *Sort by*, le champ de texte *Enter search...*, le menu déroulant *Sort* et la liste *Clients*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;
- La liste *Clients* : affiche la liste des clients associés à l'institution (c'est-à-dire ceux étant titulaires ou co-titulaires d'un compte dans cette institution) ;
- Le bouton *Export data...* : envoie l'utilisateur sur la fenêtre *Export data* ;
- Le bouton *Add client* : envoie l'utilisateur sur la fenêtre *Add client* ;
- Le bouton *Details* : envoie l'utilisateur sur la fenêtre *Details* ;
- Le bouton *Search* : effectue une recherche en fonction des critères entrés par l'utilisateur ;
- Le champ de texte *Enter search...* : permet à l'utilisateur d'entrer sa recherche ;
- Le menu déroulant *Sort* : permet à l'utilisateur de choisir un critère de tri des résultats.

Fenêtre *Add client*

Accès : en cliquant sur le bouton *Add client* de la fenetre *Clients*.

Contenu : les boutons *Back* et *Add client*, le label *Client NRN* et le champ de texte *NRN...*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Clients* ;
- Le bouton *Add client* : ajoute un client dans la table des clients de la banque dans la base de données ;
- Le champ de texte *NRN...* : permet à l'utilisateur d'entrer le numéro de registre national du client.

Fenêtre *Details*

Accès : en cliquant sur le bouton *Details* de la fenêtre *Clients*, en cliquant sur les boutons *Back*, *Export to JSON format* ou *Export to CSV formate* de la fenêtre *Export data* ou en cliquant sur le bouton *Back* de la fenêtre *Create account*.

Contenu : les boutons *Back*, *Remove client*, *Search*, *Export data...*, *Create account* et *Close account*, les labels *Choose an account*, *Client*, *Search IBAN* et *Sort by*, le champ de texte *Enter search...*, le menu déroulant *Sort* et la liste *Accounts*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Clients* ;
- Le bouton *Remove client* : enlève le client de la banque ;
- Le bouton *Search* : effectue une recherche en fonctino des critères entrés par l'utilisateur ;

— Le bouton *Export data...* : envoie l'utilisateur sur la fenêtre *Export data* ;
— Le bouton *Create account* : envoie l'utilisateur sur la fenêtre *Create account* ;
— Le bouton *Close account* : supprime le compte de ce client ;
— Le champ de texte *Enter search...* : permet à l'utilisateur d'entrer sa recherche ;
— Le menu déroulant *Sort* : permet à l'utilisateur de choisir un critère de tri des résultats.

Fenêtre *Export data*

Accès : en cliquant sur le bouton *Export data...* de la fenêtre *Clients* ou *Details* ou en cliquant sur le bouton *Export all client data* de la fenêtre *Manage data*.
Contenu : les boutons *Back*, *Choose path...*, *Export to JSON format* et *Export to CSV format* et les labels *Export location* et *Selected path*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *précédente* ;
— Le bouton *Choose path...* : ouvre l'explorateur de fichier afin de déterminer le chemin de destination ;
— Le bouton *Export to JSON format* : exporte l'historique au format JSON ;
— Le bouton *Export to CSV format* : exporte l'historique au format CSV.

Fenêtre *Create client account*

Accès : en cliquant sur le bouton *Create account* de la fenêtre *Details*.
Contenu : les boutons *Back* et *Create account*, les labels *Account type*, *Client name*, *IBAN* et *Client ID* et le menu déroulant *Type*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Details* ;
— Le bouton *Create account* : envoie l'utilisateur sur la fenêtre *Details* et crée le compte ;
— Le menu déroulant *Type* : permet à l'utilisateur d'entrer le type de compte à créer.

Fenêtre *Requests*

Accès : en cliquant sur le bouton *Manage requests* de la fenêtre *Main screen*, en cliquant sur le bouton *Back* de la fenêtreTransfer permission requests ou en cliquant sur le bouton *Back* de la fenêtre *Portfolio requests*.
Contenu : les boutons *Back*, *Transfer permission requests* et *Portfolio requests*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;
— Le bouton *Transfer permission requests* : envoie l'utilisateur sur la fenêtre *Transfer permission requests* ;
— Le bouton *Portfolio requests* : envoie l'utilisateur sur la fenêtre *Portfolio requests*.

Fenêtre *Transfer permission requests*

Accès : en cliquant sur le bouton *Transfer permission requests* de la fenêtre *Requests*.
Contenu : les bouton *Back*, *Deny* et *Approve*, le label *Choose a request* et la liste *Requests*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Requests* ;
— Le bouton *Deny* : refuse la requête sélectionnée ;
— Le bouton *Approve* : accepté la requête sélectionnée ;
— La liste *Requests* : liste les requêtes des utilisateurs.

Fenêtre *Portfolio requests*

Accès : en cliquant sur le bouton *Portfolio requests* de la fenêtre *Requests*.
Contenu : les bouton *Back*, *Deny* et *Approve*, le label *Choose a request* et la liste *Requests*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Requests* ;

— Le bouton *Deny* : refuse la requête sélectionnée ;
— Le bouton *Approve* : accepté la requête sélectionnée ;
— La liste *Requests* : liste les requêtes des utilisateurs.

Fenêtre *Manage data*

Accès : en cliquant sur le bouton *Manage data* de la fenêtre *Main screen* ou en cliquant sur le bouton *Back* de la fenêtre *Import data*.

Contenu : les boutons *Back*, *Import data* et *Export all client data*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;
— Le bouton *Import data* : envoie l'utilisateur sur la fenêtre *Import data* ;
— Le bouton *Export all client data* : envoie l'utilisateur sur la fenêtre *Export data*.

Fenêtre *Import data*

Accès : en cliquant sur le bouton *Import data* de la fenêtre *Manage data*.

Contenu : les boutons *Back*, *Choose file...* et *Import file* et les labels *Choose file* et *Selected file*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Manage data* ;
— le bouton *Choose file...* : ouvre l'explorateur de fichier afin de déterminer le fichier à importer ;
— Le bouton *Import file* : importe les données des fichiers.

Illustration du diagramme Par souci de clareté, ce diagramme est disponible en annexe du rapport.

Chapitre 2

Extension A : Gestion de cartes - Augustin HOUBA

2.1 Diagrammes de conception UML : application client

2.1.1 Use case diagram

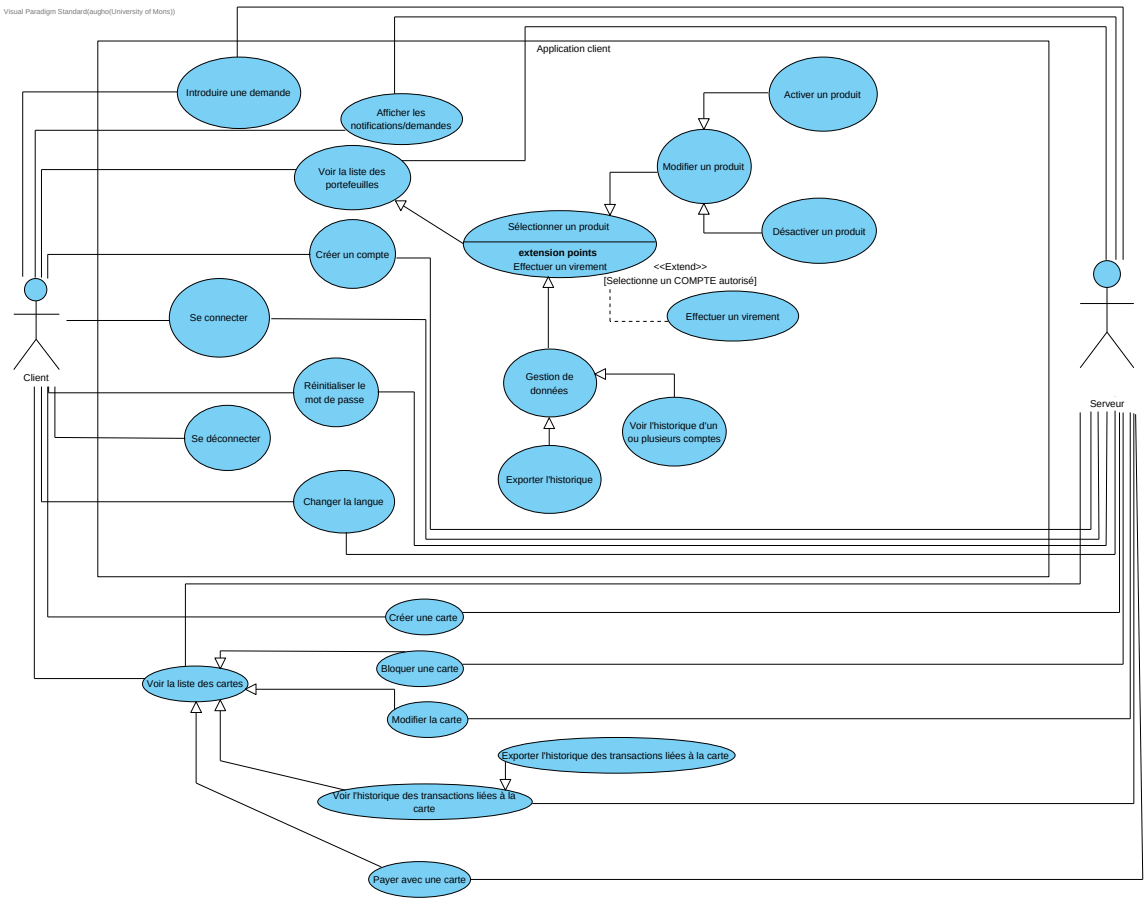


FIGURE 2.1 – Use case diagram - Extension 1

Voir la liste des cartes

Acteur principal	Utilisateur et Serveur
Description	Permet à un utilisateur d'afficher la liste des cartes
Préconditions	L'utilisateur doit être connecté
PostConditions	/
Scénario principal	1. L'utilisateur entre les données de la carte 2. L'application envoie une demande de création de carte à l'API 3. L'API répond que la demande de création de carte a été ajoutée ou qu'une erreur s'est produite
Scénario alternatif	1. L'utilisateur entre les données de la carte 2. L'application envoie une demande de création de carte à l'API 3. L'API répond qu'une erreur s'est produite 4. L'application affiche un message d'erreur
Trigger	Lorsque l'utilisateur appuie sur le bouton pour ajouter une carte
Fréquence d'utilisation	Rarement utilisée

Créer une carte

Acteur principal	Utilisateur et Serveur
Description	Permet à un utilisateur de créer une nouvelle carte
Préconditions	L'utilisateur doit être connecté et avoir introduit les données nécessaires à la création de la carte
PostConditions	Une nouvelle carte a été crée
Scénario principal	1. L'utilisateur entre les données de la carte 2. L'application envoie une demande de création de carte à l'API 3. L'API répond que la demande de création de carte a été ajoutée ou qu'une erreur s'est produite
Scénario alternatif	1. L'utilisateur entre les données de la carte 2. L'application envoie une demande de création de carte à l'API 3. L'API répond qu'une erreur s'est produite 4. L'application affiche un message d'erreur
Trigger	Lorsque l'utilisateur appuie sur le bouton pour ajouter une carte
Fréquence d'utilisation	Rarement utilisée

Bloquer une carte

Acteur principal	Utilisateur et Serveur
Description	Permet à un utilisateur de bloquer une carte
Préconditions	L'utilisateur doit avoir sélectionner une carte
PostConditions	L'API répond que la demande de blocage de la carte a été ajoutée ou qu'une erreur s'est produite
Scénario principal	1. Il actionne le bouton pour envoyer la demande à l'API 2. L'API répond que la demande de blocage de la carte a été ajoutée
Scénario alternatif	1. Il actionne le bouton pour envoyer la demande à l'API 2. L'API répond qu'une erreur s'est produite 3. Un message d'erreur est affiché
Trigger	Lorsque l'utilisateur actionne le bouton Stop Card
Fréquence d'utilisation	Très rare

Modifier une carte

Acteur principal	Utilisateur et Serveur
Description	Permet à un utilisateur de modifier une carte
Préconditions	L'utilisateur doit avoir sélectionner une carte
PostConditions	L'API répond que la demande de modification de la carte a été ajoutée ou qu'une erreur s'est produite
Scénario principal	1. L'utilisateur modifie les données qu'il veut changer 2. Il actionne le bouton pour envoyer la demande à l'API 3. L'API répond que la demande de modification de la carte a été ajoutée
Scénario alternatif	1. L'utilisateur modifie les données qu'il veut changer 2. Il actionne le bouton pour envoyer la demande à l'API 3. L'API répond qu'une erreur s'est produite 4. Un message d'erreur est affiché
Trigger	Lorsque l'utilisateur actionne le bouton Submit après avoir modifier des données
Fréquence d'utilisation	Peu fréquent

Voir l'historique des transactions liées à la carte

Acteur principal	Utilisateur et Serveur
Description	Permet à un utilisateur d'exporter les transactions au format json
Préconditions	L'utilisateur doit être connecté et avoir sélectionné une carte
PostConditions	La liste des transactions est affichée
Scénario principal	1. L'utilisateur appuie sur le bouton history 2. Une demande pour recevoir l'historique des transactions est effectuée 3. L'API renvoie les données 4. Les données sont affichées
Scénario alternatif	1. L'utilisateur appuie sur le bouton history 2. Une demande pour recevoir l'historique des transactions est effectuée 3. L'API répond qu'une erreur s'est produite 4. L'application display une erreur et propose de refresh
Trigger	Lorsque l'utilisateur appuie sur le bouton pour voir l'historique
Fréquence d'utilisation	Moyennement Fréquent

Acteur principal	Utilisateur et Serveur
Description	Permet à un utilisateur d'exporter les transactions au format json
Préconditions	L'utilisateur doit être connecté et avoir la liste des transactions en local
PostConditions	Un fichier json contenant les transactions de la carte est crée
Scénario principal	1. L'utilisateur appuie sur le bouton pour exporter 2. Un fichier json contenant les transactions de la carte est crée
Scénario alternatif	/
Trigger	Lorsque l'utilisateur appuie sur le bouton pour exporter
Fréquence d'utilisation	Peu fréquent

Exporter l'historique des transactions liées à la carte

Acteur principal	Utilisateur et Serveur
Description	Permet à un utilisateur de payer avec une carte
Préconditions	L'utilisateur doit avoir sélectionner une carte et avoir entré le compte destinataire ainsi que le montant(les autres infos non-nécessaires au paiement par carte seront auto-générées).
PostConditions	L'API répond que la demande de transaction a été effectuée ou qu'une erreur s'est produite
Scénario principal	1. L'utilisateur sélectionne la carte avec laquelle il veut payer 2. Il entre le destinataire et le montant 3. Il appuie sur le bouton pour payer et envoit la demande à l'API 4. L'API répond que le demande a été faite avec succès 5. L'application affiche que la transaction a été effectuée
Scénario alternatif	1. L'utilisateur sélectionne la carte avec laquelle il veut payer 2. Il entre le destinataire et le montant 3. Il appuie sur le bouton pour payer et envoit la demande à l'API 4. L'API répond qu'une erreur s'est produite 5. Un message d'erreur est affiché
Trigger	Lorsque l'utilisateur actionne le bouton Pay dans la scène Card Management
Fréquence d'utilisation	Fréquent

2.1.2 Interaction overview diagram

Introduction Pour mon extension qui est la gestion de cartes je n'ai pas jugé nécessaires de modifier l'interaction overview diagram de l'application pour l'institution car il n'y avait rien à changé Je vais donc uniquement parler des ajouts que j'ai réalisé à celui de l'application client.

Gestions de cartes Cette partie sert de point d'entrée pour les fonctionnalités de cette extension. L'utilisateur a plusieurs options disponibles :

1. Payer avec une carte
2. Voir la liste des cartes

Payer avec une carte Ici l'utilisateur pour faire un paiement en utilisant une de ses cartes. Il pourra :

1. Annuler le paiement
2. Confirmer le paiement

Dans les deux cas il sera renvoyer vers la gestion de cartes.

Voir la liste des cartes Cette partie couvre les flows de la gestion de liste des cartes de crédit et de débit car ils sont similaires. Lorsque l'utilisateur à la liste devant lui, il peut :

1. Créer une carte
2. Sélectionner une carte

Créer une carte Pour cette interaction l'utilisateur aura la possibilité de valider sa demande de création ou d'annuler. Les deux cas aboutiront à un retour à la liste des cartes.

Sélectionner une carte Lorsque l'utilisateur sélectionne une carte il a la possibilité de :

1. Modifier une carte -> Retourne directement à la sélection de carte
2. Bloquer une carte -> Retourne directement à la sélection de carte
3. Voir l'historique des transactions de la carte

Voir l'historique des transactions de la carte Ici l'utilisateur pourra visualiser l'historique de la carte qu'il avait sélectionné. Il pourra ensuite

1. Retourner à la liste des cartes
2. Exporter son historique -> Retourner directement à la liste des cartes

2.1.3 Class diagram

Introduction Ce document parle des modifications apportées au diagramme de classe de la partie commune pour l'extensions de gestions de cartes.

Partie Logique La classe abstraite *AbstractCard* décrit le comportement commun pour entre les cartes de crédit et débit. Comme la gestion de l'historique et le blocage d'une carte. Les classes *CreditCard* et *DebitCard* spécifie les différences entre ces 2 types de cartes notamment avec les transaction et la récupération des données de la cartes. De plus la carte de Crédit a 2 attributs supplémentaires *maxAmount* et *availableAmount* qui indique respectivement le plafond de la carte ainsi que le montant restant pour le mois. L'énumération CardBrand contient la liste des différentes marques de cartes utilisée **Classes rajoutées** :

- AbstractCard
- CreditCard
- DebitCard
- CardBrand

Patie GUI Pour chaque nouvelle scène, une classe a été rajoutée. Ces classes respectent toujours le design pattern Singleton
Classes rajoutées :

- CardPayScene
- CardManagementScene
- AddCreditCardScene
- AddDebitCardScene
- CreditCardScene
- DebitCardScene

Partie API Pour chaque nouvel endpoint, une classe a été rajoutée les méthodes correspondent aux méthodes HTTP disponibles à cet endpoint.
Classes rajoutées :

- CardController
- DebitCardController
- CreditCardController

2.1.4 Sequence diagram

Introduction Ce document parle des modifications apportées au diagramme de classe de la partie commune pour l'extensions de gestions de cartes.

Partie Logique La classe abstraite *AbstractCard* décrit le comportement commun pour entre les cartes de crédit et débit. Comme la gestion de l'historique et le blocage d'une carte. Les classes *CreditCard* et *DebitCard* spécifie les différences entre ces 2 types de cartes notamment avec les transaction et la récupération des données de la cartes. De plus la carte de Crédit a 2 attributs supplémentaires *maxAmount* et *availableAmount* qui indique respectivement le plafond de la carte ainsi que le montant restant pour le mois. L'énumération CardBrand contient la liste des différentes marques de cartes utilisée **Classes rajoutées** :

- AbstractCard
- CreditCard
- DebitCard
- CardBrand

Patie GUI Pour chaque nouvelle scène, une classe a été rajoutée. Ces classes respectent toujours le design pattern Singleton
Classes rajoutées :

- CardPayScene
- CardManagementScene
- AddCreditCardScene
- AddDebitCardScene
- CreditCardScene
- DebitCardScene

Partie API Pour chaque nouvel endpoint, une classe a été rajoutée les méthodes correspondent aux méthodes HTTP disponibles à cet endpoint.
Classes rajoutées :

- CardController
- DebitCardController
- CreditCardController

2.2 Diagrammes de conception UML : application institution

2.2.1 Use case diagram

Il n'y a pas de changement par rapport à l'application de base

2.2.2 Interaction overview diagram

Il n'y a pas de changement par rapport à l'application de base

2.2.3 Class diagram

Il n'y a pas de changement par rapport à l'application de base

2.2.4 Sequence diagram

Il n'y a pas de changement par rapport à l'application de base

2.2.5 Maquette de l'interface utilisateur : application client

Introduction Ce document décrit les modifications effectuées à l'UI pour l'extension des cartes. Les scènes modifiées sont en couleur orange.

Scènes modifiées

- Sign in ;
- Main screen ;
- Transaction history ;
- Cards Management ;
- Credit Card ;
- Debit Card ;
- Card Pay
- Add a Credit Card ;
- Add a debit Card ;

Scène *Sign In* Une méthode de connexion alternative est ajoutée afin de permettre de se connecter via sa carte de banque. Les textfields CardNumber et pin permettront d'entrer les données de la carte avec laquelle on veut se connecter.

Scène *Main screen* **Contenu modifié/ajouté** : le bouton*Cards*

- Le bouton *Cards* : envoie l'utilisateur sur la fenêtre *Cards Management* ;

Scène *Transaction History* Cette scène se comporte de la meme manière que si l'on regardait les transactions d'un certain compte. Je me suis permis de la recycler pour les cartes car que ce soit pour un compte ou une carte les données utilisées pour représenter l'historique sont des transactions sous forme de tableau. Les transactions d'une carte étant un sous-ensemble des transactions d'un compte. La manière pour visualiser ou exporter sera aussi identique.

Scène *Cards Management* **Contenu** : les boutons*Manage debit card*, *Manage credit card* et *Pay*.

- Le bouton *Manage credit card* : envoie l'utilisateur sur la fenêtre *Credit Card* ;
- Le bouton *Manage debit card* : envoie l'utilisateur sur la fenêtre *Debit Card* ;
- Le bouton *Pay* : envoie l'utilisateur sur la fenêtre *CardPay* ;

Scène *Credit Card*

- Le bouton *add* : envoie l'utilisateur sur la fenêtre *Add Credit Card* ;
- La liste *Cards list* : Affiche la liste des cartes de crédit ;
- La combo box *Change linked account* : permet de changer le compte lié à la carte. Elle contient la liste des comptes liés au profil ;
- La check box *Foreign Transaction* : permet d'autoriser ou pas les transactions étrangères ;
- La check box *Negative transaction* : permet d'autoriser ou pas les virements mettant le compte en négatif ;
- Le bouton *Submit* : permet d'envoyer la requete de modification au serveur ;
- Le bouton *Stop Card* : Envoie une requête pour bloquer la carte ;

— Le bouton *History* : envoie l'utilisateur sur la fenêtre *Transaction History*
— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Cards Management*

Scène *Debit Card*
— Le bouton *add* : envoie l'utilisateur sur la fenêtre *Add Debit Card* ;
— La liste *Cards list* : Affiche la liste des cartes de débit ;
— La combo box *Change linked account* : permet de changer le compte lié à la carte. Elle contient la liste des comptes liés au profil ;
— La check box *Foreign Transaction* : permet d'autoriser ou pas les transactions étrangères ;
— La check box *Negative transaction* : permet d'autoriser ou pas les virements mettant le compte en négatif ;
— Le bouton *Submit* : permet d'envoyer la requete de modification au serveur ;
— Le bouton *Stop Card* : Envoie une requête pour bloquer la carte ;
— Le bouton *History* : envoie l'utilisateur sur la fenêtre *Transaction History*
— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Cards Management*

Scène *Card Pay*
— La liste *Cards list* : Affiche la liste des cartes ;
— Le text field *Iban* : permet de spécifier le compte destinataire ;
— Le text field *Amount* : permet de spécifier le montant de la transaction ;
— Le bouton *Pay* : permet d'envoyer la requete de transaction au serveur ;
— Le bouton *History* : envoie l'utilisateur sur la fenêtre *Transaction History*
— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Cards Management*

Scène *Add a Debit Card*
— La combo box *Change linked account* : permet de changer le compte lié à la carte. Elle contient la liste des comptes liés au profil ;
— La combo box *Brand* : permet de changer la marque de la carte ;
— La check box *Foreign Transaction* : permet d'autoriser ou pas les transactions étrangères ;
— La check box *Negative transaction* : permet d'autoriser ou pas les virements mettant le compte en négatif ;
— Le textfield *Pin* : détermine le pin de la carte
— Le bouton *Submit* : permet d'envoyer la requete de modification au serveur et retourne vers Debit Card ;
— Le bouton *Cancel* : envoie l'utilisateur sur la fenêtre *Debit Card*

Scène *Add a Credit Card*
— La combo box *Change linked account* : permet de changer le compte lié à la carte. Elle contient la liste des comptes liés au profil ;
— La combo box *Brand* : permet de changer la marque de la carte ;
— La check box *Foreign Transaction* : permet d'autoriser ou pas les transactions étrangères ;
— La check box *Negative transaction* : permet d'autoriser ou pas les virements mettant le compte en négatif ;
— Le textfield *Pin* : détermine le pin de la carte
— Le textfield *MaxAmount* : détermine le montant qui sera mis à disposition chaque mois sur la carte ;

— Le bouton *Submit* : permet d'envoyer la requete de modification au serveur et retourne vers Debit Card ;
— Le bouton *Cancel* : envoie l'utilisateur sur la fenêtre *Debit Card*

2.2.6 Maquette de l'interface utilisateur : application institution

Introduction Ce document décrit l'utilisation de l'interface graphique de l'application institution en se basant sur la maquette de l'interface.

Scènes modifiées
— Requests ;

Scène *Requests* Elle assume le rôle d'approuver ou refuser une demande. Peut importe son type. Cela comprends notamment les demandes pour bloquer une carte ou en créer une nouvelle.
Contenu :
— Le bouton *Deny* : Va communiquer à l'api que la requête a été refusée ;
— Le bouton *Approve* : Va communiquer à l'api que la requête a été acceptée ;
— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main Screen*
— La list *RequestList* : Contient la liste des requêtes de la banque ;

2.2.7 Design du REST API

Introduction Ce diagramme décrit les ajouts et modifications apportés à l'api pour l'extension de gestion de cartes.

\transaction

1. **GET** : Un paramètre de recherche *card-id* a été rajouté. Si il est spécifié, l'api saura que la transaction est effectuée par une carte. Elle ignorera alors le paramètre *wallet-iban* et retournera la liste des transactions liées à la carte.
2. **POST** : Un paramètre de recherche *card-id* a été rajouté. Si il est spécifié, l'api saura que la transaction est effectuée par une carte

\card

1. **DELETE** : Le paramètre card-id est requis cette méthode supprime simplement une carte.

\card\debit-card

1. **POST** : Création d'une nouvelle carte de débit .
2. **PUT** : Modifier les paramètres de la carte de débit. *card-id* est la clé primaire. Les autres paramètres sont ceux qui seront modifiés.
3. **GET** : Retourne une liste d'instances de DebitCard liées à un user *user-id*

\card\credit-card

1. **POST** : Création d'une nouvelle carte de crédit.
2. **PUT** : Modifier les paramètres de la carte de crédit. *card-id* est la clé primaire. Les autres paramètres sont ceux qui seront modifiés.
3. **GET** : Retourne une liste d'instances de CreditCard liées à un user *user-id*

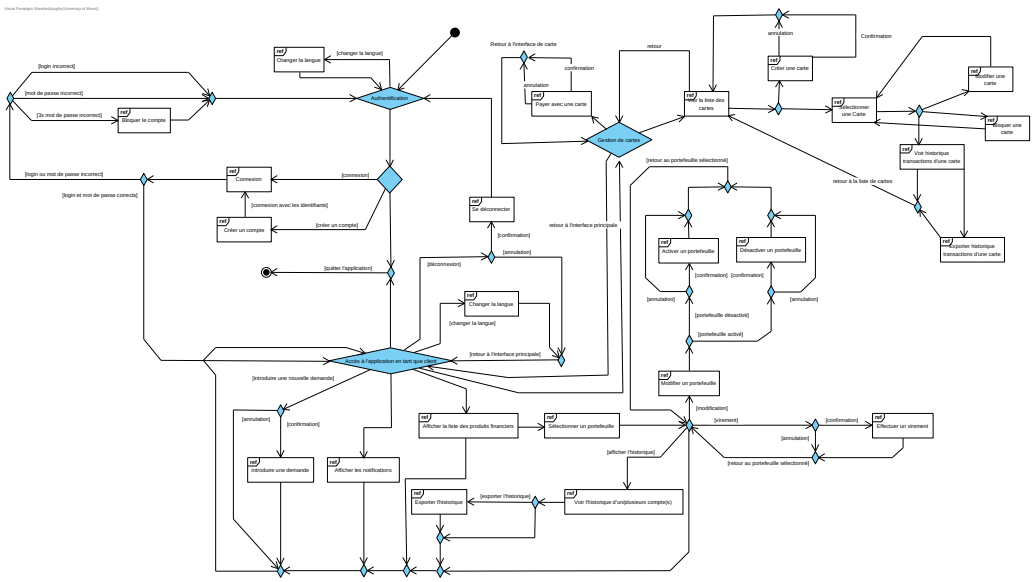


FIGURE 2.2 – Interaction overview diagram - Extension 1

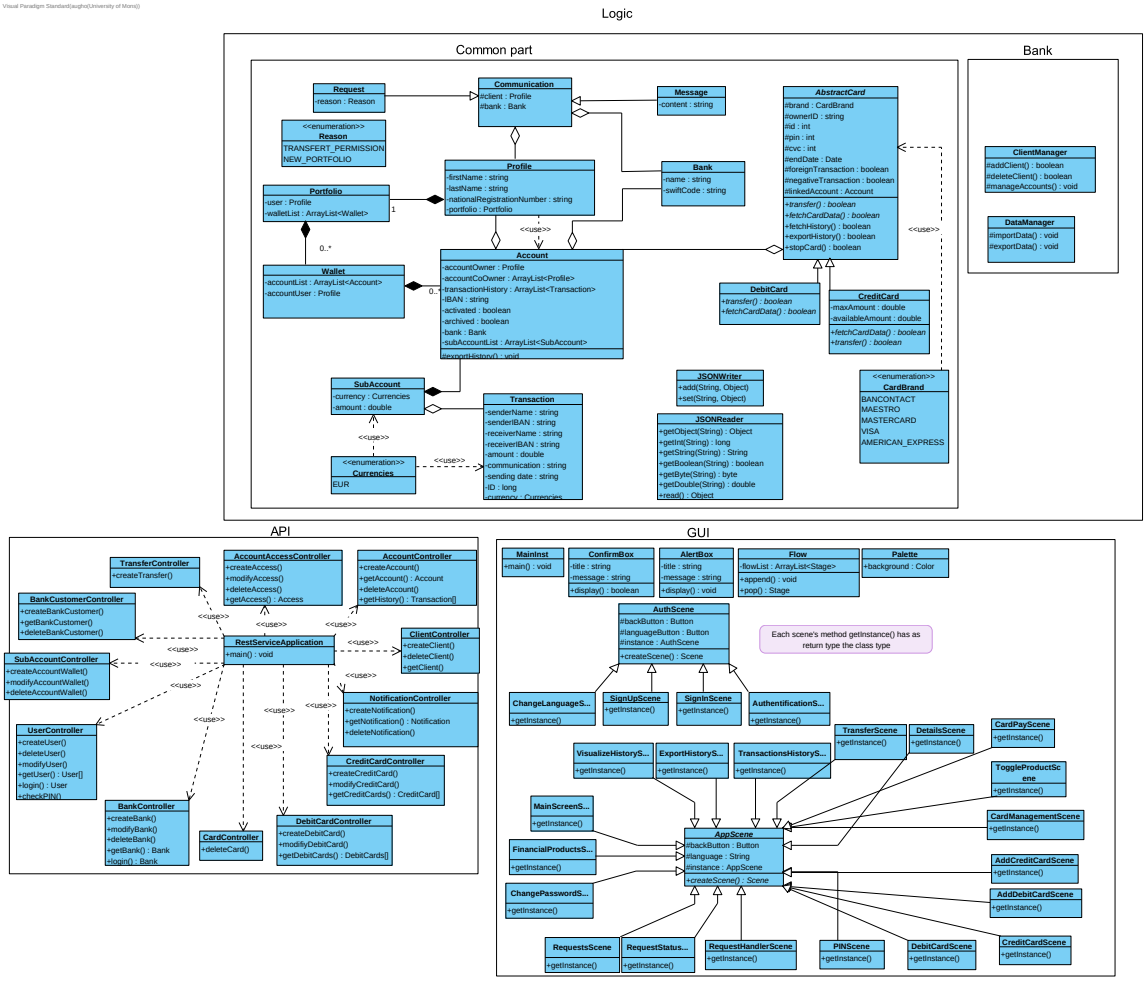


FIGURE 2.3 – Diagramme de classes - Extension 1

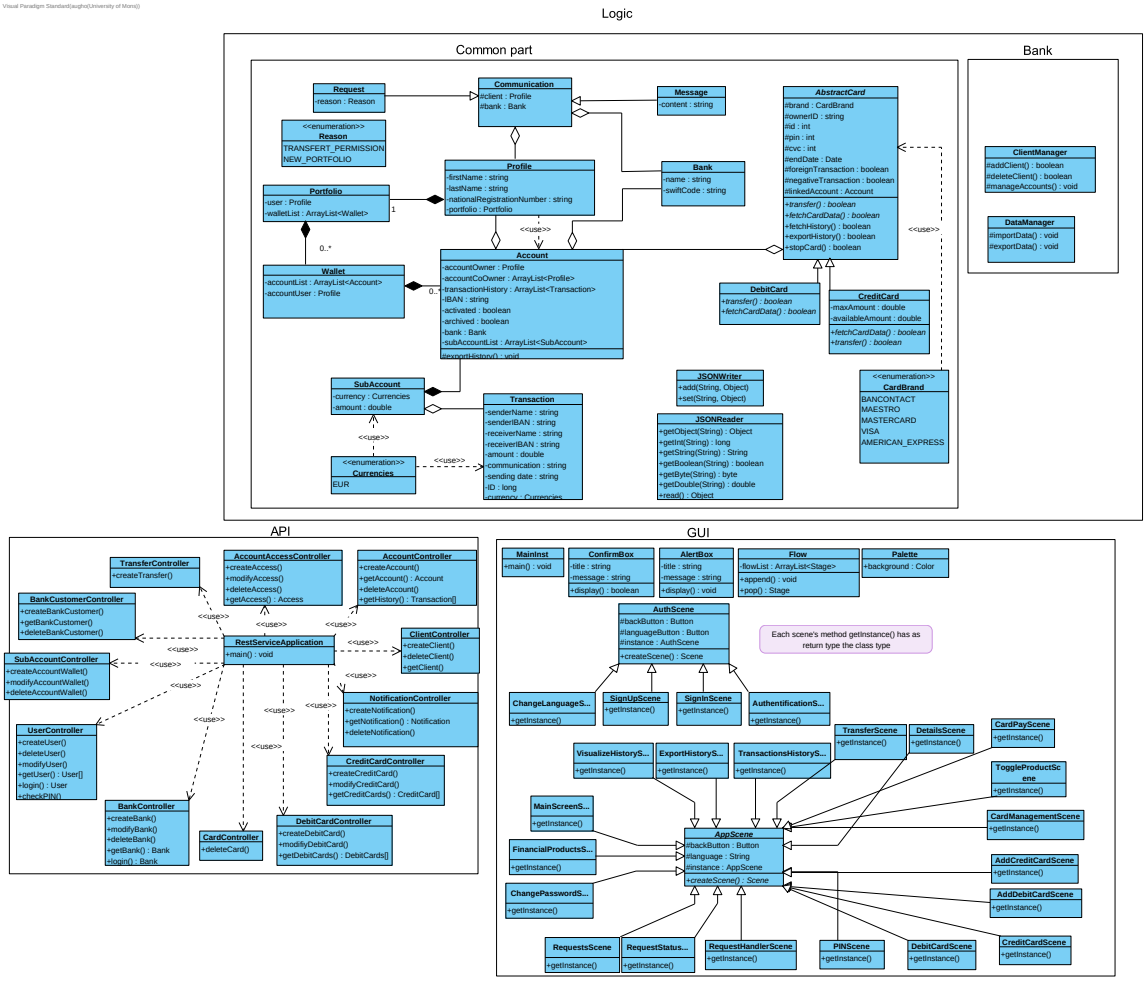


FIGURE 2.4 – Diagramme de classes - Extension 1

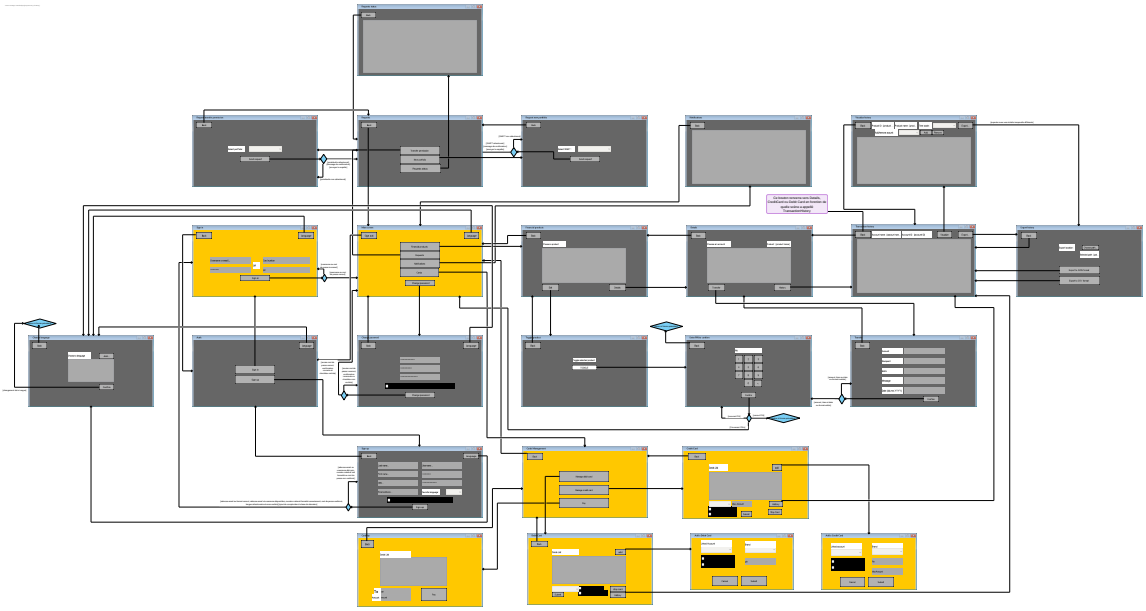


FIGURE 2.5 – Maquette de l'interface utilisateur : application client - Extension 1

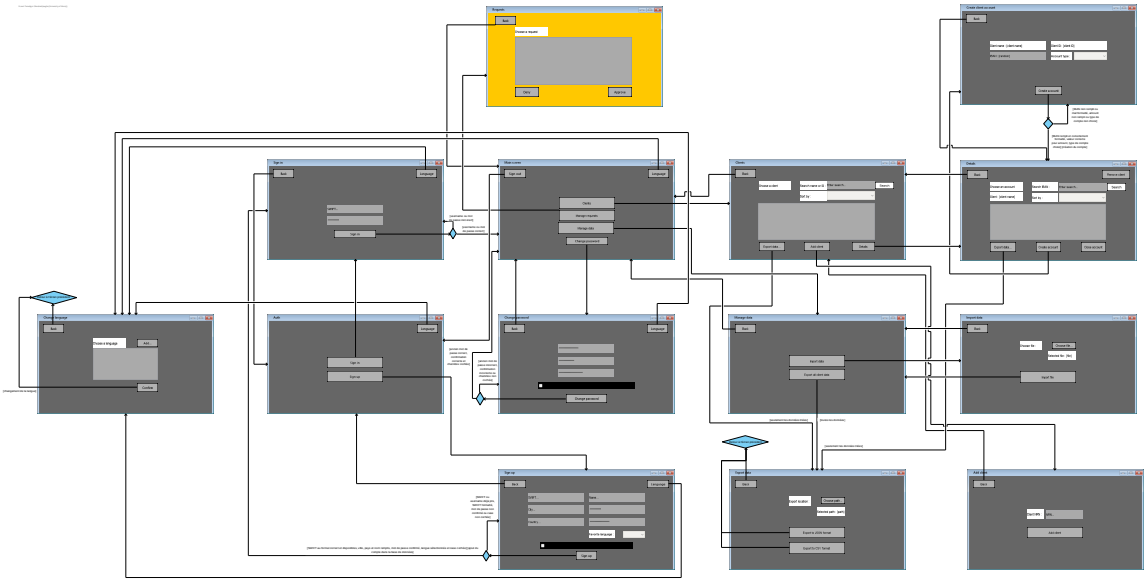


FIGURE 2.6 – Maquette de l'interface utilisateur : application institution - Extension 1

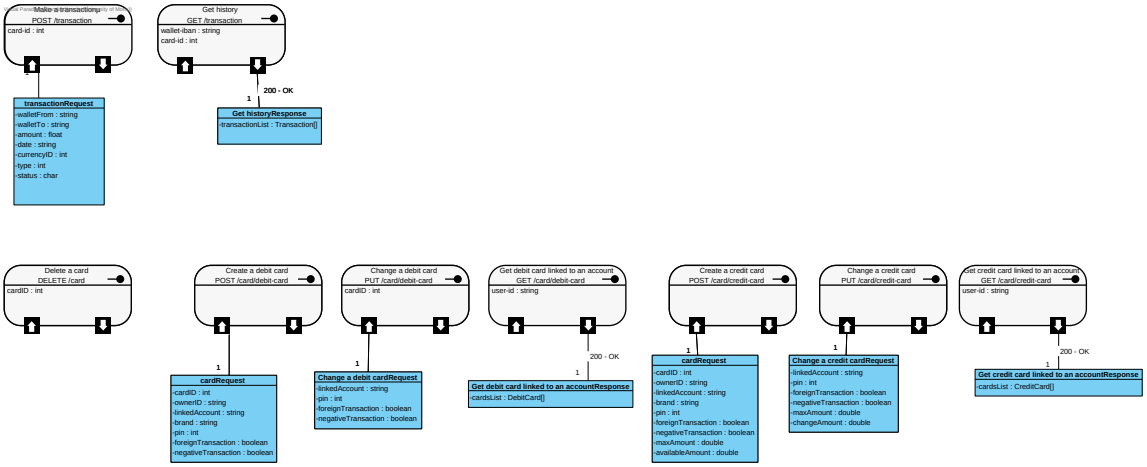


FIGURE 2.7 – Maquette de l'interface utilisateur : application client - Extension 1

Chapitre 3

Extension B : Gestion de devises et virements internationaux - Cyril MOREAU

3.1 Diagrammes de conception UML : application client

3.1.1 Use case diagram

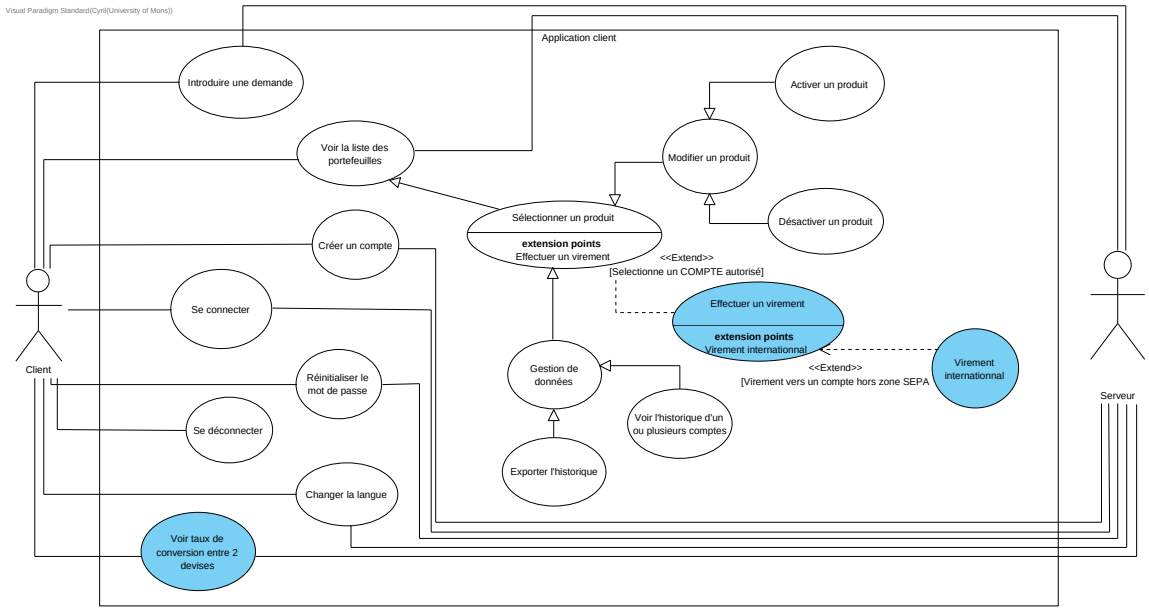


FIGURE 3.1 – Diagramme de case d'utilisation de l'application client de l'extension 2.

Pour cette extension, un nouveau type de virement a été ajouté : les virements internationnaux qui sont une extension du virement. Un virement internationnal ne se produit que lorsqu'un virement est effectué vers un compte dans une institution se trouvant hors zone SEPA.
Un autre use case a été ajouté permettant au client de voir le taux de conversion des devises disponibles dans l'application. Un graphique permettant de visualiser le taux de conversion de ces devises au cours du temps sera également affiché. Ce graphique sera calculé grâce aux fichiers JSON présent sur le serveur. En effet, chaque jour, le serveur enverra une requête à l'api *ExchangeRate* permettant d'avoir le taux de conversion de chaque devise. Ces données seront stockées chaque jour dans un fichier JSON présent sur le serveur.

3.1.2 Interaction overview diagram

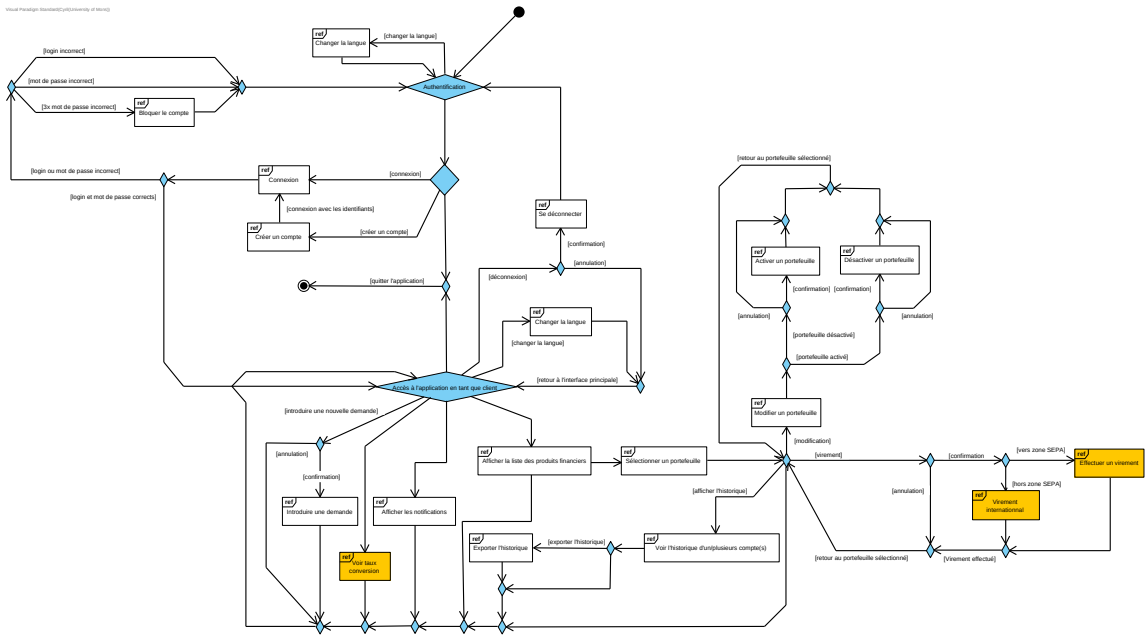


FIGURE 3.2 – Interaction overview diagram de l'application client de l'extension 2.

Les changements effectué pour cette extensions sont mis en évidence en orange.
Comme très peu de cas d'utilisation ont été ajoutés car les changements majeurs résident dans une gestion interne, très peu de changement sont également effectué dans l'interaction overview diagram.
Tout d'abord, les virements ont été modifié. L'utilisateur à maintenant la possibiblité de choisir entre effectuer un virement internationnal(hors zone SEPA) ou un virement normal(En zone SEPA).
Il a également la possibilité de consulter le taux de conversion d'une devise vers une autre après s'est authentifié.

3.1.3 Class diagram

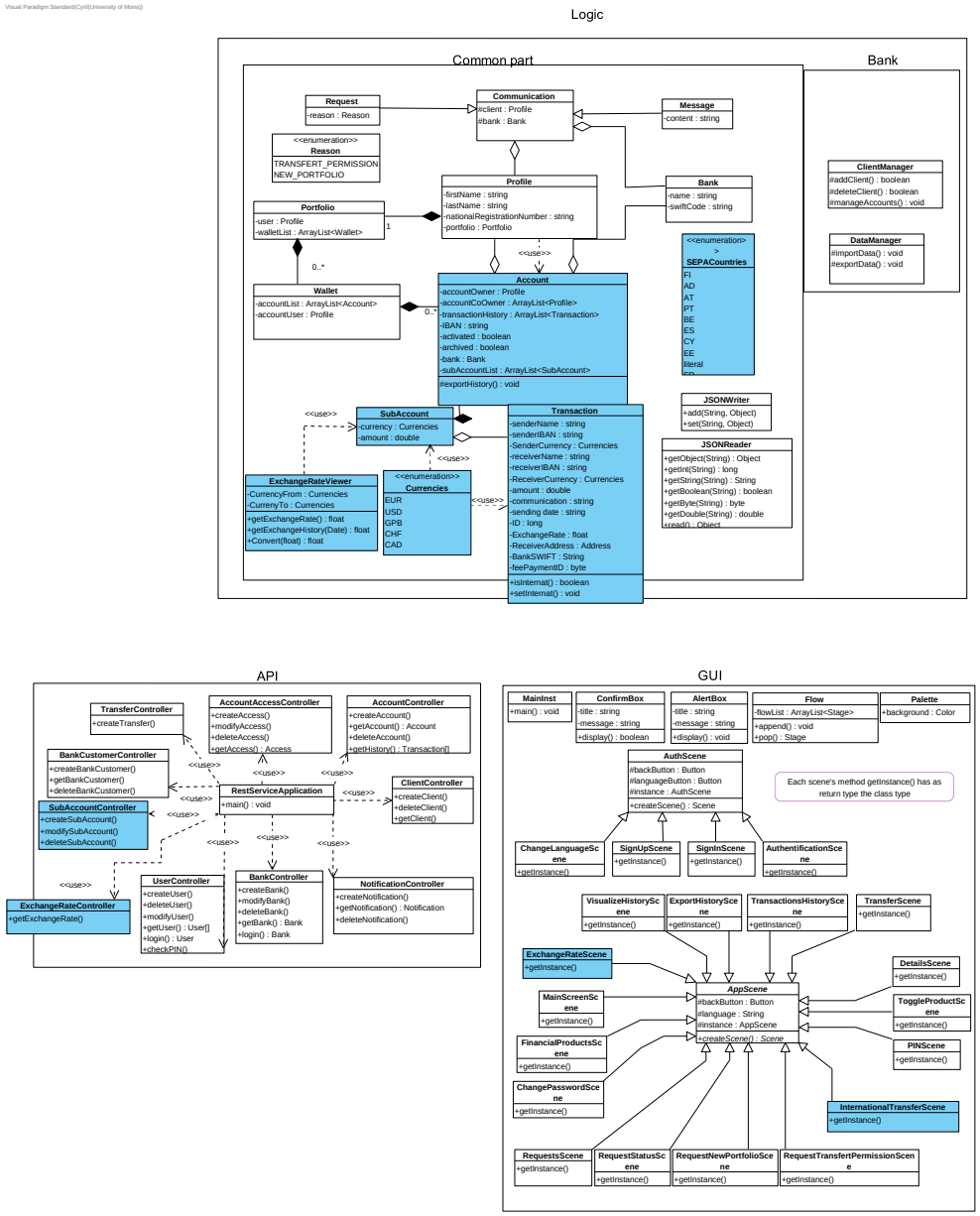


FIGURE 3.3 – Diagramme de classe de l'application client de l'extension 2

Introduction Contrairment à la partie commune, les diagrammes de classes des deux applications sont légèrement différent. Certaines classes ne se trouve uniquement dans l'une ou l'autre application.

Partie logique Tout d'abord, un changement majeur a été effectué dans le gestion des comptes de la partie commune à cause de cette extension. En effect, dans la base de donnée, un compte est divisé en sous-compte en fonction de leur devise. Dans la partie commune, uniquement l'EUR est implémenté donc il n'ont qu'un seul sous compte.
Cela implique donc quelques changement dans les classes correspondant aux comptes :

- Une classe *SubAccount* a été ajoutée à cause de ce changement dans la base de donnée (Dans le cas de cette extension, cela permet de gérer les comptes multidevise).
- La classe *Account* contient maintenant une liste de *SubAccount*.
- Une enumeration *SEPACountries* a été ajoutée afin de garder la liste des pays de la zone SEPA. Cette liste n'est pas modifiée très souvent, une enumeration est donc suffisante.
- Une enumeration *Currencies* a également été ajoutée afin de garder la liste des devises supportées par l'application. Lors de l'implémentation, cette technique pourrait être modifiée afin de permettre aux institution d'ajouter de nouvelles devises.
- Une nouvelle classe *ExchangeRateViewer* a été ajoutée afin de permettre à l'application de connaitre les taux de change entre 2 devises et de convertir un montant d'une devise à une autre.
- La classe *Transaction* a également été adaptée afin de permettre d'effectuer des virements internationnaux et des virements entre 2 comptes ayant des devises différentes.
- La classe *Address* permet de regrouper les données de l'address du bénéficaire d'une transaction (city, street,...).

Partie API Ici, très peu de changement ont été effectué par rapport à la partie commune. Uniquement 2 controller ont été ajouté afin d'accéder/modifier des information sur le serveur.

Partie GUI Finalementn, uniquement 2 nouvelles classes correspondant aux 2 nouvelles fenêtres ont été ajoutée. Toujours en respectant le design pattern singleton.

3.1.4 Sequence diagram

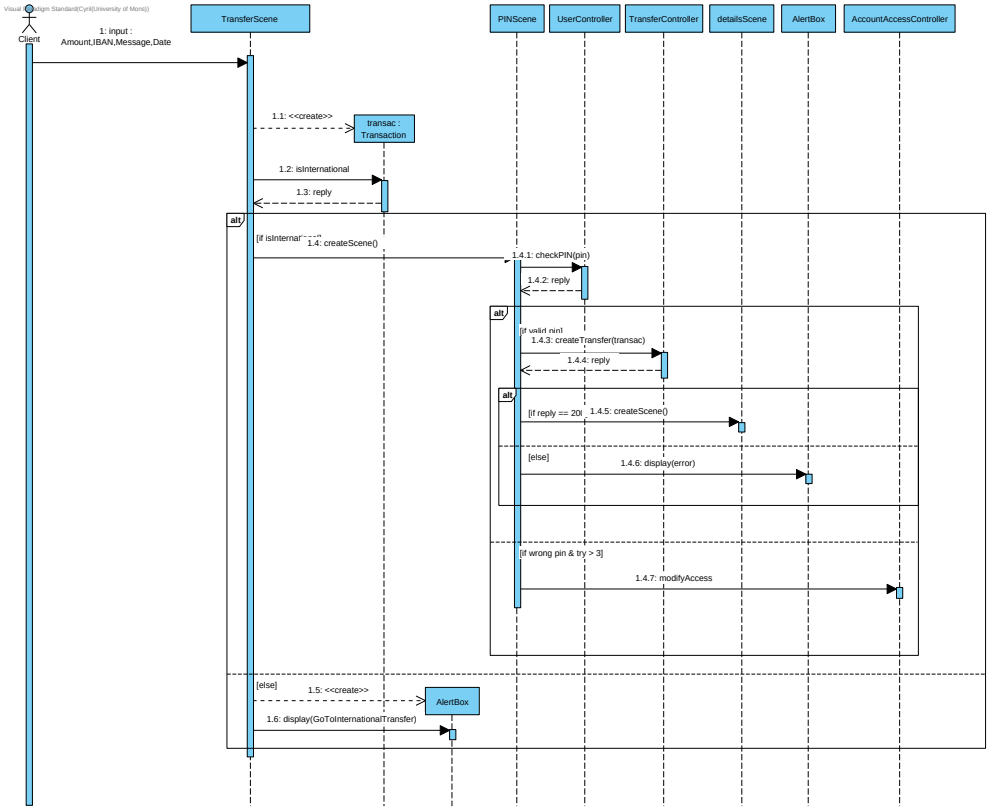


FIGURE 3.4 – Effectuer un virement

La fonction permettant d'effectuer un virement a légèrement été modifiée car l'application va vérifier si la banque du compte vers lequel le client souhaite effectuer un virement est bien située en zone SEPA (1.2). Si ce n'est pas le cas, le client sera invité à effectuer un virement international(1.6).

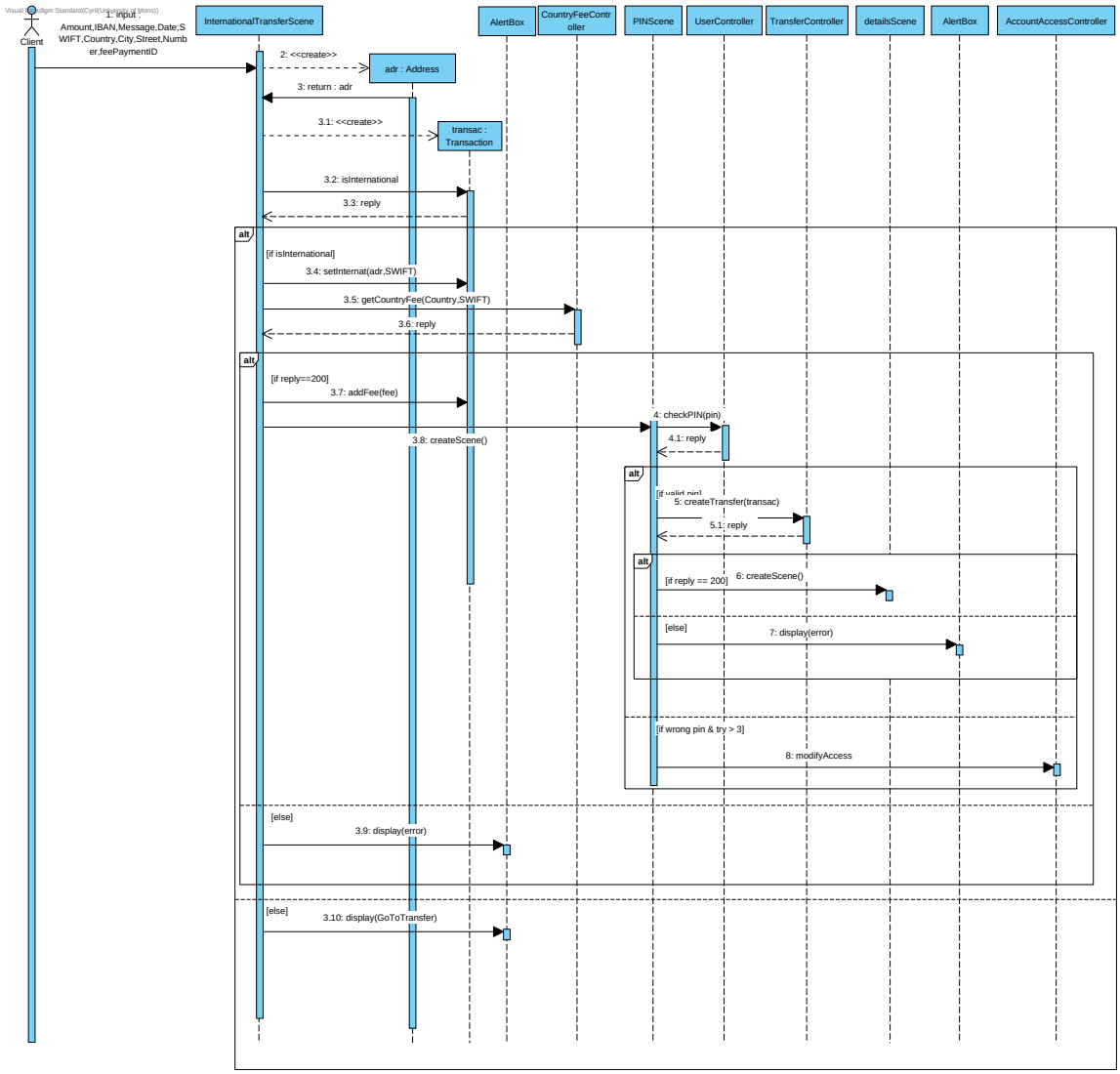


FIGURE 3.5 – Virement international

Effectuer un virement international est assez similaire à la fonction effectuer un virement. Les seules différences résident dans le fait que le client doit rentrer l'adresse exacte du bénéficiaire et le code SWIFT de la banque (1).
De plus, des frais s'ajoute en fonction du pays vers lequel il souhaite effectuer le virement. (3.7)
Pour le reste, cela est similaire au virement normal.

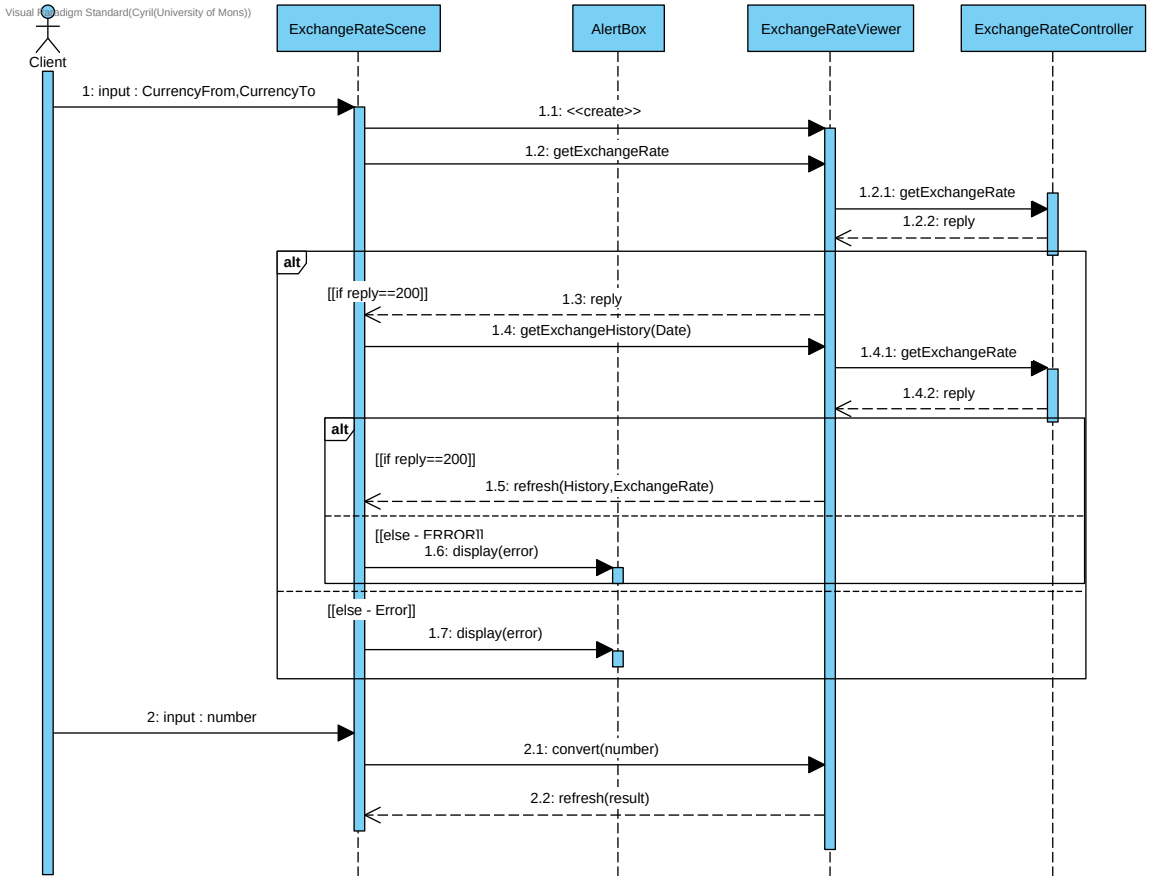


FIGURE 3.6 – Voir taux conversion

Calculer le taux de conversion est dans le fond assez simple, l'application va récupérer le taux de change du jour et va ensuite récupérer l'historique afin de pouvoir afficher le graphique.
Le client peut ensuite entrer une valeur à convertir qui sera calculée grace à la classe *ExchangeRateViewer*.
Le reste n'est globalement que de la gestion d'erreur renvoyée par l'API.

3.2 Diagrammes de conception UML : application institution

3.2.1 Use case diagram

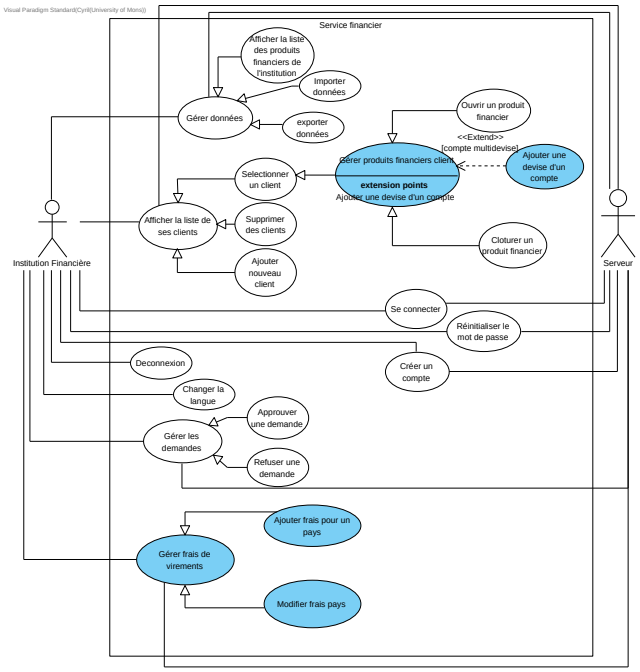


FIGURE 3.7 – Diagramme de case d'utilisation de l'application service financier de l'extension 2.

Ici, peu de use case ont dû être ajouté. Seul une extension du use case *Gérer produits financiers client* qui n'est disponible uniquement pour les comptes multidevises.
Des cas d'utilisation ont également été ajouté afin de permettre aux institution d'ajouter des frais aux pays ne se trouvent pas en zone SEPA, si les pays ne sont pas dans cette liste, le virement ne peux pas être effectué.
Elle permet d'ajouter une devise au compte. Ici également, la description semi-structurée des nouveaux use case se trouve dans un PDF attaché à ce rapport.

3.2.2 Interaction overview diagram

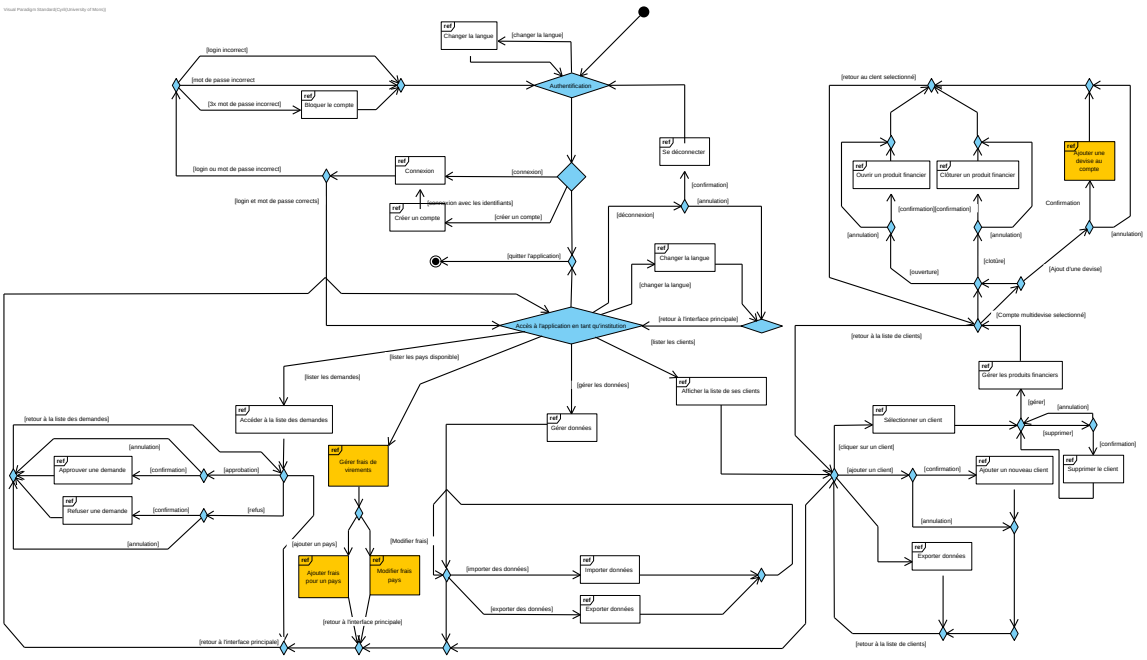


FIGURE 3.8 – Interaction overview diagram de l'application service financier de l'extension 2.

Dans ce diagramme, plusieurs choses ont été ajoutées :
Tout d'abords, lorsque l'institution souhaite gérer un produit financier, si celui-ci est un compte multi-devise, elle a la possibilité d'ajouter une devise. Ensuite, l'institution à maintenant la possibilité, après s'être authentifié, de gérer les pays vers lesquels elle autorise d'effectuer des virements (pays hors zone SEPA). Elle peut ajouter un pays dans la liste et modifier les frais lorsqu'un client effectue un virement vers un compte d'une banque située dans un de ces pays.

3.2.3 Class diagram

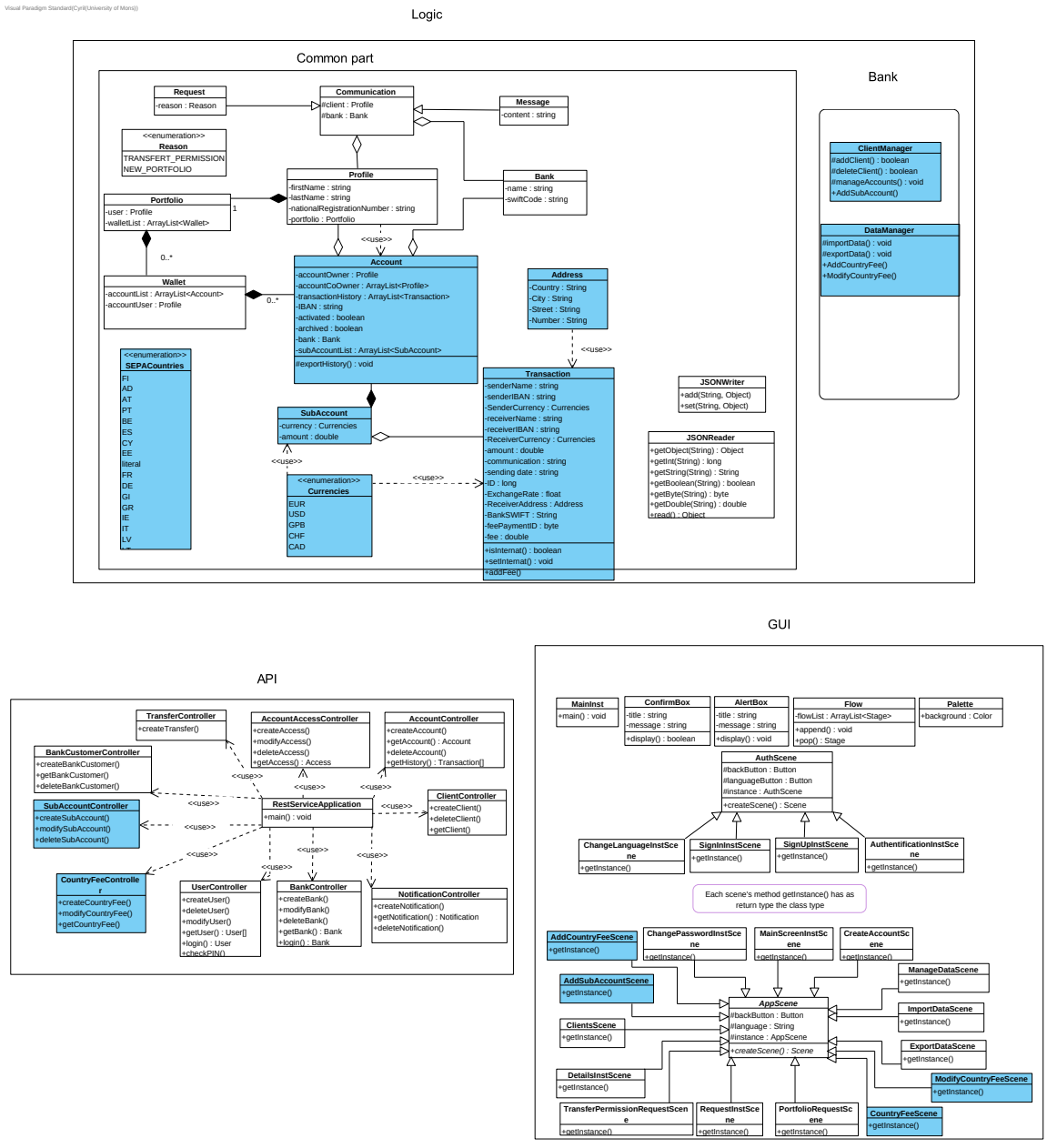


FIGURE 3.9 – Diagramme de classe de l'application service financier de l'extension 2

Introduction : Comme dit dans la partie client, des changements ont été effectué entre les deux applications. Ce diagramme est donc légèrement différent du diagramme de l'application client. Cependant, certains changement dans ce diagramme sont également dans la partie client et y sont déjà expliqué.

Partie logique

- La classe *ClientManager* a été modifié afin de permettre aux institutions d'ajouter un sous-compte grace à la méthode *addSubAccount()*.
- La classe *DataManager* a également été modifiée afin de permettre aux institutions d'ajouter ou de modifier les frais de virement vers un pays hors zone SEPA.

Partie API Comme dans la partie client, peu de changement ont été effectué dans cette partie.
Uniquement 2 classes ont été ajoutée :

- Une classe *SubAccountController* a été ajoutée afin de permettre aux institution d'ajouter et de modifier un sous-compte.
- Une classe *CountryFeeController* a également été ajoutée afin de pouvoir ajouter/modifier des frais sur un pays hors zone SEPA.

Les méthodes de ces classes seront appelée par les classe *ClientManager* et *DataManager* afin qu'uniquement la partie logique n'appelle ces classes et de garder le schéma assez clair.

Partie GUI Encore une fois, une classe par fenêtre a été ajoutée tout en respectant le design pattern singleton.

3.2.4 Sequence diagram

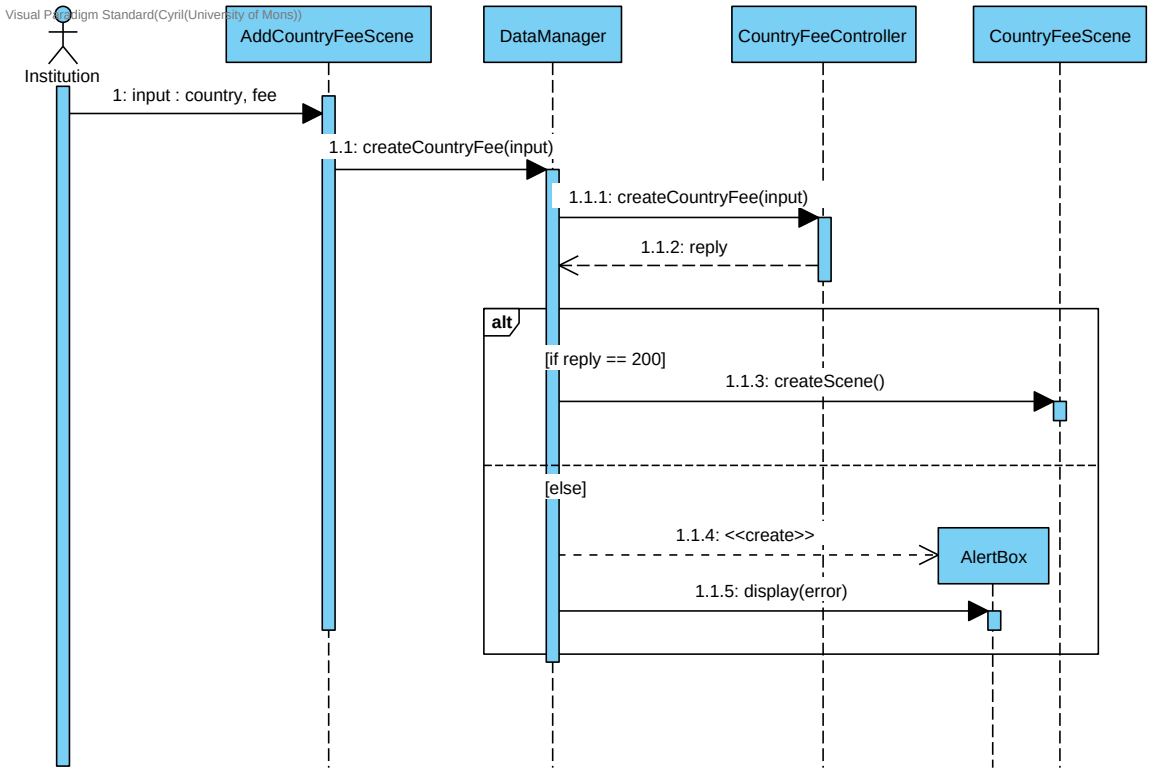


FIGURE 3.10 – Ajouter frais pour un pays

Ajouter les frais pour un pays est globalement assez simple. On appelle le *DataManager* qui appelle le countryFeeController afin d'ajouter les frais dans la table correspondante. Le reste est juste les gestions d'erreurs grace à la classe *AlertBox*.

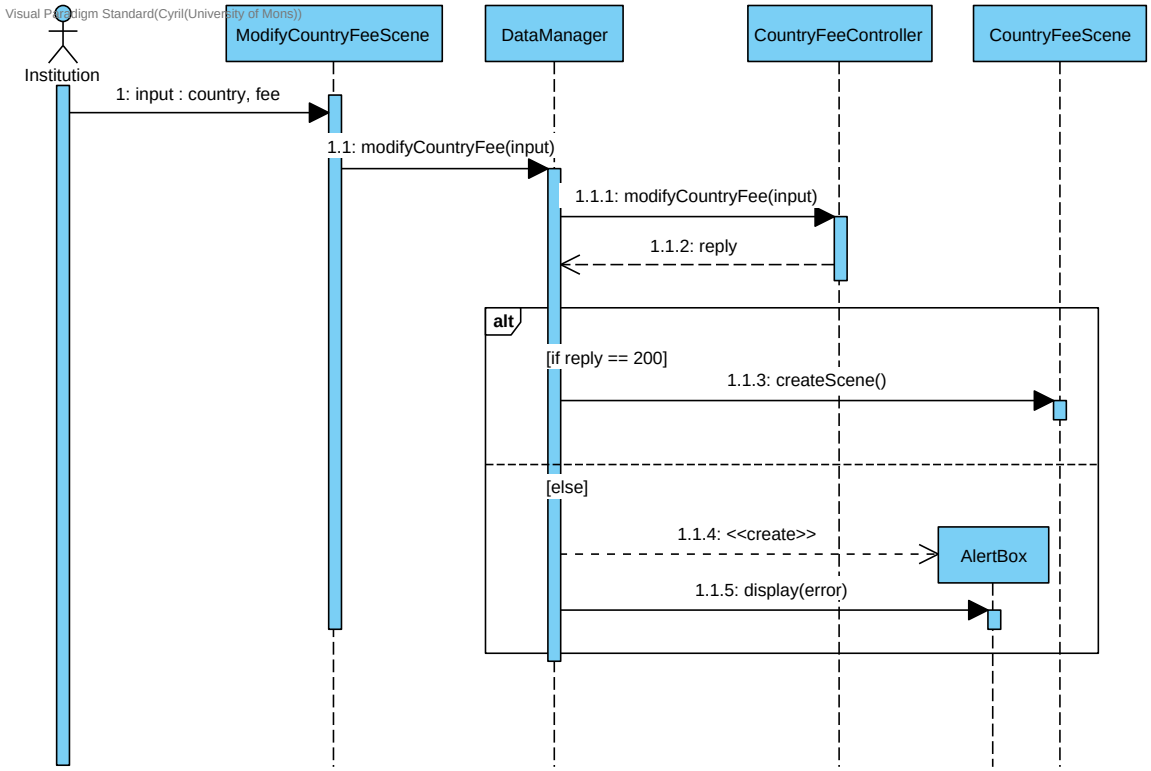


FIGURE 3.11 – Modifier frais pour un pays

Modifier des frais pour un pays est pareil que le sequence diagram pour ajouter des frais. La seule différence est qu'on appelle la méthode *ModifyCountryFee* au lieu de *AddCountryFee*.

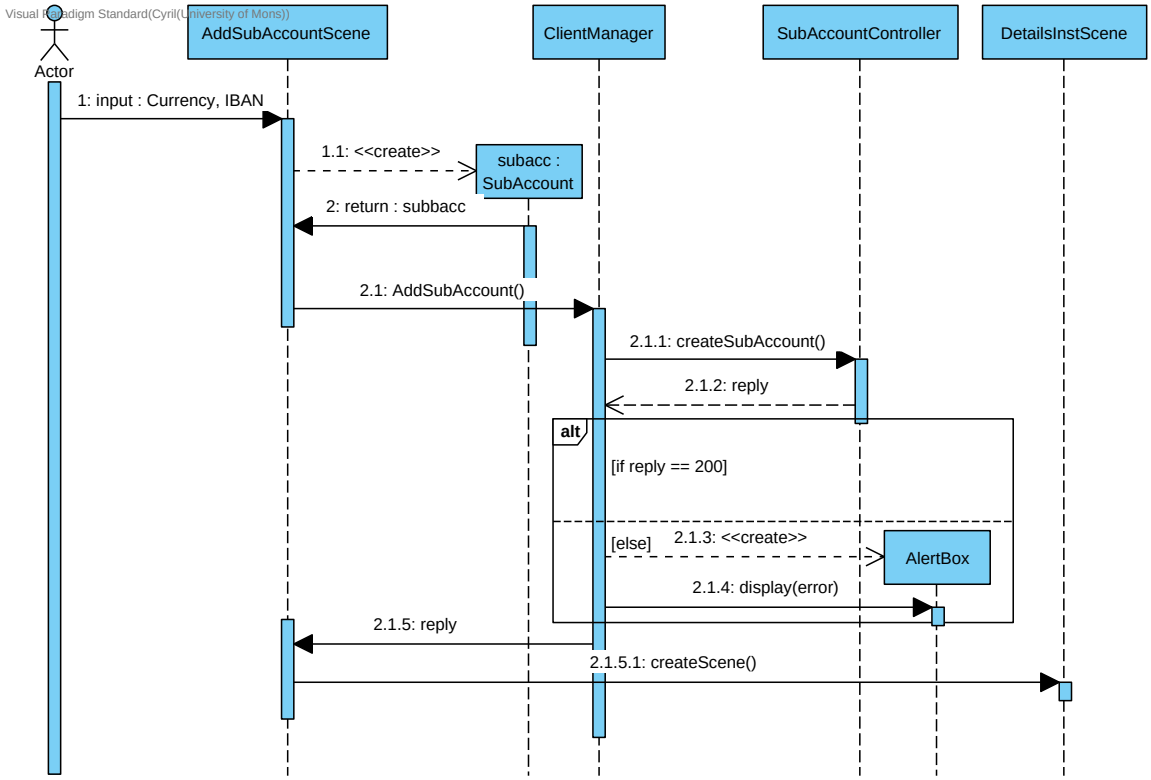


FIGURE 3.12 – Ajouter une devise pour un compte

Cette sequence crée tout d'abord le *subAccount* demandé, elle va ensuite faire appel au *ClientManager* qui va appeller le *SubAccountController* ce qui va ajouter un subAccount dans la base de donnée.
L'erreur potentielle est gérée en ensuite l'application nous renvoie dans tous les cas vers la fenêtre Details.

3.3 Modèle de données

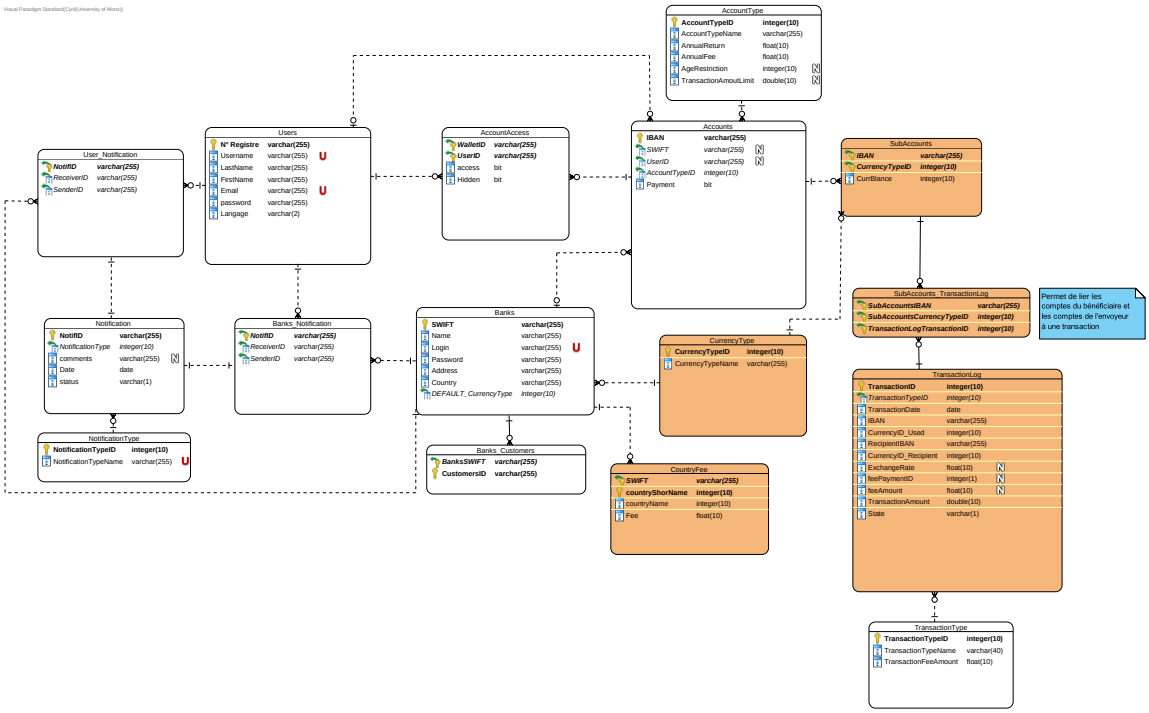


FIGURE 3.13 – Diagramme d'entité relation de l'extension 2.

Pour cette extension, de nombreux changements ont été effectués au niveau du diagramme d'entité relation.
Tout d'abord, chaque compte contient plusieurs subaccounts qui sont différencier par le type de devise. Ce qui permet de créer des comptes multi-devises. Si un compte n'est pas multi-devise, il ne possèdera qu'un seul subaccount.
L'ajout de ces SubAccount implique quelques changement de l'entité *TransactionLog* car il faut effectuer le virement depuis un subaccount vers un autre subaccount. C'est pour cela que les deux attributs *CurrencyI_Used* et *CurrencyID_Recipient* ont été ajouté car ils permettent de décrire de quel devise a été envoyé l'argent et vers quelle devise. Un attribut est également ajouté afin de connaitre le taux de conversion entre les deux devise au moment de la transaction.
Une entité *CountryFee* a également été ajoutée afin de connaitre les frais de virement internationaux pour chaque banque. Si un utilisateur souhaite effectuer un virement vers un pays qui n'est pas dans la liste de sa banque, le virement ne peut pas être effectué.

3.4 Maquette de l'interface de l'application client

Par soucis de clarté, le diagramme correspondant est attaché en annexe à ce rapport.

Nouvelles fenêtres disponibles
- Exchange rate
- International transfer

Fenêtres modifiées
- Details
- Main screen

Fenêtre Exchange rate
Accès : En cliquant sur le bouton *ExchangeRate* de la fenêtre *Main screen*.
Contenu : Les boutons *Back* et *Convert*, les labels *From*, *To* et *converted number*, les deux comboBox permettant de choisir les currencies et un panel *Graph*
- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen*
- Le bouton *Convert* : permet de valider le choix de devise et convertir le nombre sélectionné. Affiche Egalement l'historique de taux de change entre ces devises

Fenêtre International transfer
Accès : En cliquand sur le bouton *International Transfer* de la fenêtre *Details*.
Contenu : les boutons *Back* et *Confirm*, les labels *Amount*, *Recipient*, *IBAN*, *Message*, *Date*, *SWIFT*, *country*, *City*, *Street*, *Number* et les champs de texte *Amount*, *Recipient*, *IBAN*, *Message*, *Date*,*SWIFT*, *country*, *City*, *Street*, *Number*.
- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen*
- Le bouton *Confirm* : envoie l'utilisateur sur la fenêtre *Enter PIN to confirm* si les informations sont correctes, on effectue le virement si la réponse de la fenêtre *Enter PIN to confirm* est positive
- Le champ de texte *Recipient* : permet à l'utilisateur d'entrer le destinataire
- Le champ de texte *IBAN* : permet à l'utilisateur d'entrer l'IBAN du destinataire
- Le champ de texte *Message* : permet à l'utilisateur d'entrer la communication liée au virement
- Le champ de texte *Date* : permet à l'utilisateur d'entrer la date de planification du virement
- Le champ de texte *SWIFT* : permet à l'utilisateur d'entrer Code swift de la banque du bénéficiaire
- Le champ de texte *Country* : permet à l'utilisateur d'entrer Pays du bénéficiaire
- Le champ de texte *City* : permet à l'utilisateur d'entrer Ville du bénéficiaire
- Le champ de texte *Street* : permet à l'utilisateur d'entrer Rue du bénéficiaire
- Le champ de texte *Number* : permet à l'utilisateur d'entrer Numéro de maison du bénéficiaire

Fenêtre *Details* **Nouvel accès :** en cliquant sur le bouton *Back* de la fenêtre *International transfer*. **Nouveau contenu :** le bouton *International transfer*.
- Le bouton *International transfer* : envoie l'utilisateur sur la fenêtre *International transfer*.

Fenêtre *Main screen* **Nouvel accès :** en cliquant sur le bouton *Back* de la fenêtre *ExchangeRate* **Nouveau contenu :** le bouton *Exchange rate*
- Le bouton *ExchangeRate* : envoie l'utilisateur sur la fenêtre *ExchangeRate*.

3.5 Maquette de l'interface de l'application pour une institution

Nouvelles fenêtres disponibles

- AddCountryFee
- CountryFee
- MldifyCountryFee
- AddSubAccount

Fenêtre *AddCountryFee*

Accès : En cliquant sur le bouton *Add* de la fenêtre *CountryFee*

Contenu : Les boutons *Back* et Confirm ainsi que les champs de texte *CountryName* et Fee

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *CountryFee*
- Le bouton *Confirm* ajoute les frais sur ce pays dans la base de donnée et ramène à la fenêtre *CountryFee*
- Le champ de texte *CountryName* : permet à l'utilisateur d'entrer Le nom du pays
- Le champ de texte *Fee* : permet à l'utilisateur d'entrer Les frais ajouté à ce pays.

Fenêtre *ModifyCountryFee* Cette fenêtre est globalement similaire à la fenêtre *AddCountryFee*.

Fenêtre *CountryFee*

Accès : En cliquand sur le bouton *Manage Country Fee* de la fenêtre *Main screen*.

Contenu : Les boutons *Back*,*Add* et *Modify* ainsi que la liste *Countries*

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen*
- Le bouton *Add* : envoie l'utilisateur sur la fenêtre *AddCountryFee*
- Le bouton *Modify* : envoie l'utilisateur sur la fenêtre *ModifyCountryFee*
- La liste *Countries* affiche la liste des pays qui ont déjà des frais dans cette banque.

AddSubAccount

Accès : En cliquand sur le bouton *AddSubAccount* de la fenêtre *Details*.

Contenu : Les boutons *Back* et *Confirm*, les labels *Client IBAN*, *Client Name*, *Select new Currency* et la comboBox *Currencies*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Details*
- Le bouton *Confirm* permet d'ajouter le sous compte et de revenir à la fenêtre *Details*.
- La comboBox *Currencies* permet de sélectionner la devise du nouveau subAccount.

Chapitre 4

Extension C : Gestion des contrats d'assurance - François VION

4.1 Diagrammes de conception UML : application client

4.1.1 Use case diagram

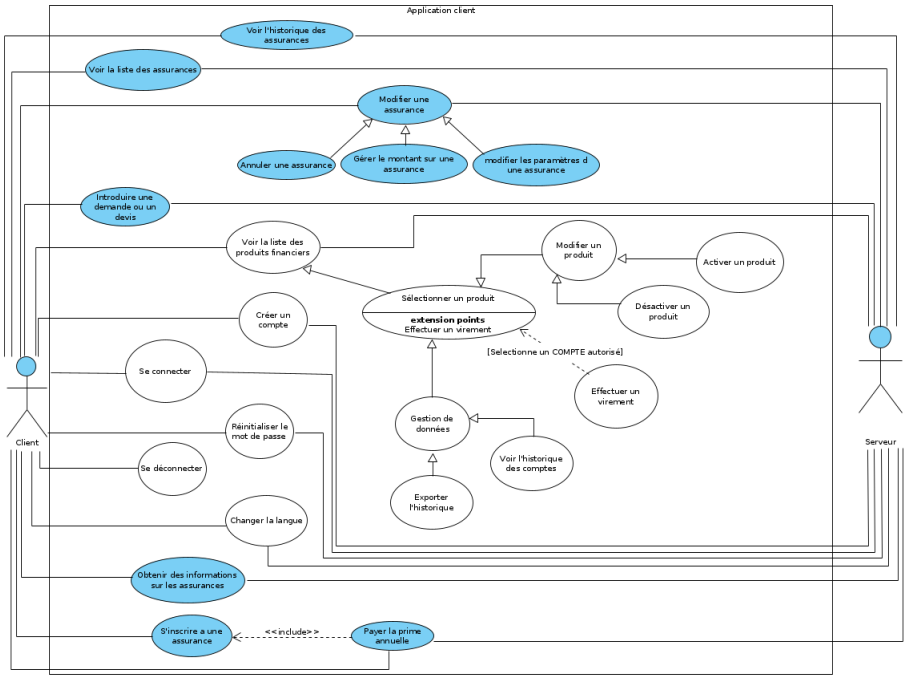


FIGURE 4.1 – Use case de l'application client

Le use case de l'extension assurance rajoute dix cas. Ceux si seront détaillés à part du rapport dans un fichier pdf qui reprend les descriptions semi-structurées. Il y a cependant quelques particularités comme le use case "Introduire une demande" qui a été modifié en "Introduire une demande ou un devis", le use case "Modifier une assurance" qui comprend plusieurs spécifications et le use case "S'inscrire à une assurance" qui inclus "Payer la prime anuelle" car l'inscription se valide via le premier payement.

4.1.2 Interaction overview diagram

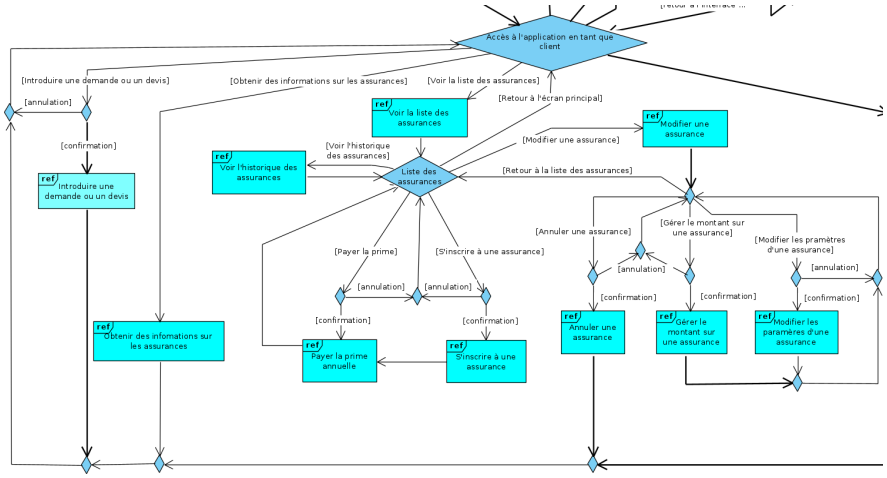


FIGURE 4.2 – Ajout fait à l'Interaction Overview Diagram de l'application de base pour la partie client

L'interaction overview diagram de l'extension assurance ajoute ses uses cases à celui de base. La grande partie de ceux ci sont disponibles après avoir listé les assurances. Après cela, on peut voir l'historique des assurances, s'inscrire à une assurance en payant la prime annuelle, payer la prime annuelle d'une assurance et modifier une assurance. Si l'on souhaite modifier une assurance, trois choix s'offrent à nous. Soit on peut annuler une assurance, soit gérer le montant sur une assurance, soit modifier les paramètres d'une assurance. A côté de cela on peut obtenir des informations sur les assurances et introduire un devis. Pour tout les cas importants, une confirmation est demandée avant d'effectuer l'action en question.

4.1.3 Class diagram

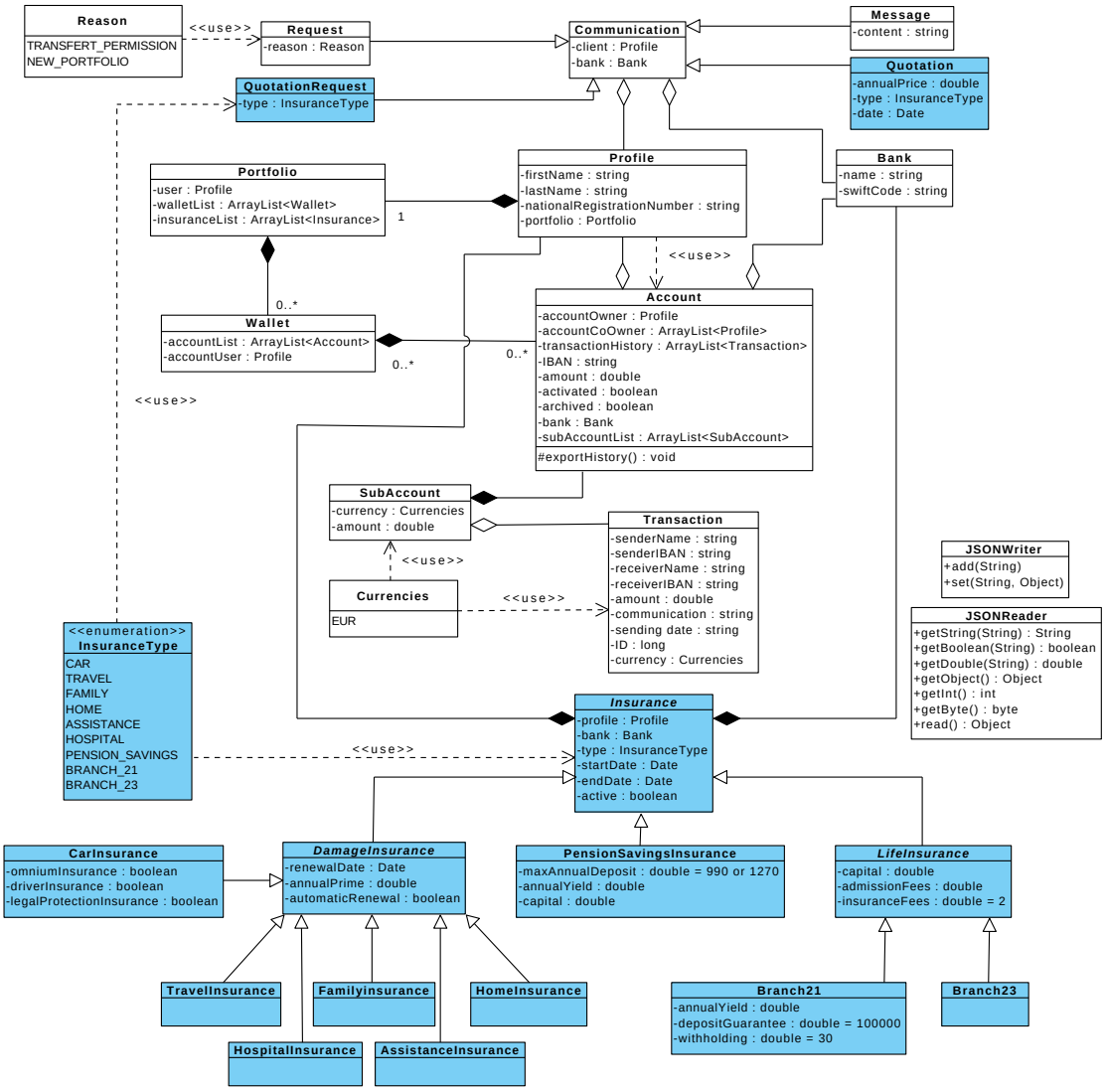


FIGURE 4.3 – Diagramme de classe de la partie logique de l'extension assurance

Partie logique La partie logique ne modifie les classes déjà existantes mais en ajoute. Toutes celles-ci sont en commun avec l'application banque. Tout d'abord, on a la classe mère Insurance qui représente une assurance grâce à un assuré et un assureur (Profile et Bank), un type d'assurance, une date de début, une date de fin (Pas forcément définie) et son état d'activité pour savoir si elle est cloturée ou en cours. Chaque

type d'assurance possède sa classe fille. Les assurances de dégats sont caractérisées par une date de renouvellement, une prime annuelle et une option pour activer le renouvellement automatique. Seule l'assurance voiture ajoute quelques options supplémentaires à savoir l'assurance omnium, l'assurance conducteur et l'assurance protection juridique. Les assurances épagne pension sont caractérisées par un maximum annuel de dépôt, un rendement annuel et un capital. Les assurances vies sont caractérisées par un capital, des frais d'entrée et des taxes d'assurances. Ces assurances sont sous 2 formes. La première est la branche 21 qui ajoute un rendement annuel, une guarantie de dépôt et un précompte mobilier. La deuxième est la branche 23 et elle n'ajoute rien de plus. Afin de pouvoir gérer les devis entre le client et la banque, deux classes filles de la classe Communication ont été ajoutées. Il s'agit de la classe QuotationRequest qui représente une demande pour un certain type d'assurance et de la classe Quotation qui possède les informations a envoyer au client à savoir la prime anuelle pour un type d'assurance ainsi que la date du devis.

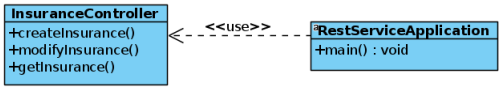


FIGURE 4.4 – Ajout au diagramme de classe de la partie API de l'extension assurance

Partie API La partie API n'ajoute qu'une classe InsuranceController qui permettra de créer, modifier et recevoir une assurance de la base de données. Elle est en commun avec la partie banque.

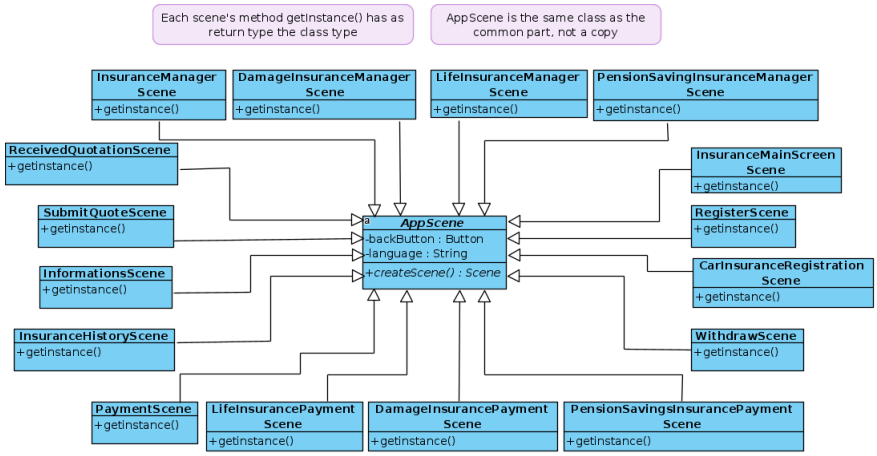


FIGURE 4.5 – Ajout au diagramme de classe de la partie GUI de l'extension assurance

Partie GUI La partie GUI ajoute des scenes à la partie de base et ne modifie que le mainScreenScene. Celles-ci héritent tous de AppScene qui est la même classe que la partie commune. Elle a juste isolée du reste de l'application sur le diagramme pour une meilleure visibilité.

4.1.4 Sequence diagram

Pour suivre le même principe que les diagrammes de séquence de la partie commune, seuls les diagrammes de séquences des cas d'utilisation ne se limitant pas à la création et la manipulation d'un objet instancié via l'API seront détaillés.

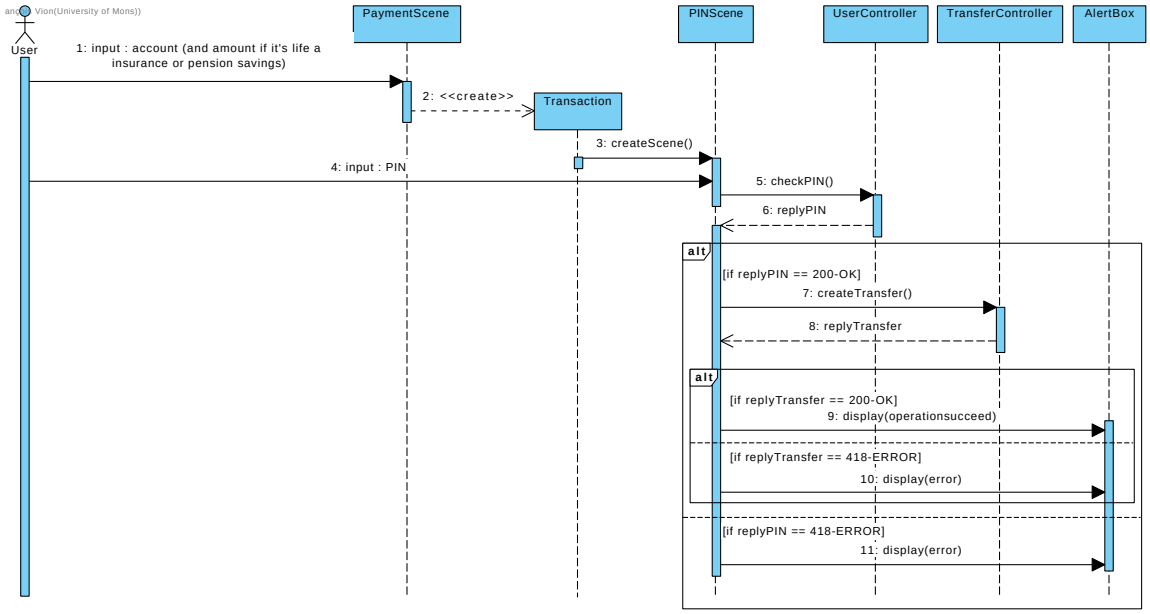


FIGURE 4.6 – Diagramme de séquence du use case "Payer la prime annuelle"

Pour payer la prime annuelle, le client rentre sur la scene le compte avec lequel il compte payer ainsi que le montant à payer s'il s'agit d'une assurance vie ou épargne pension (1). Un objet Transaction est créé (2). Celui-ci va créer la scene de confirmation avec le code PIN (3). Ensuite, l'utilisateur va rentrer son code PIN (4). Celui-ci sera vérifié via l'API (5) et elle renverra une réponse (6). Si cette réponse est 200-OK, alors on crée le transfert via l'API (7) qui renvoie une réponse (8). Si la réponse au transfert est bonne (200-OK), alors une fenêtre est affichée pour signaler que l'opération a bien été effectuée (9). Si l'une des 2 réponses est mauvaise (418-ERROR), on affiche une fenêtre qui signale qu'il y a eu une erreur (10 et 11).

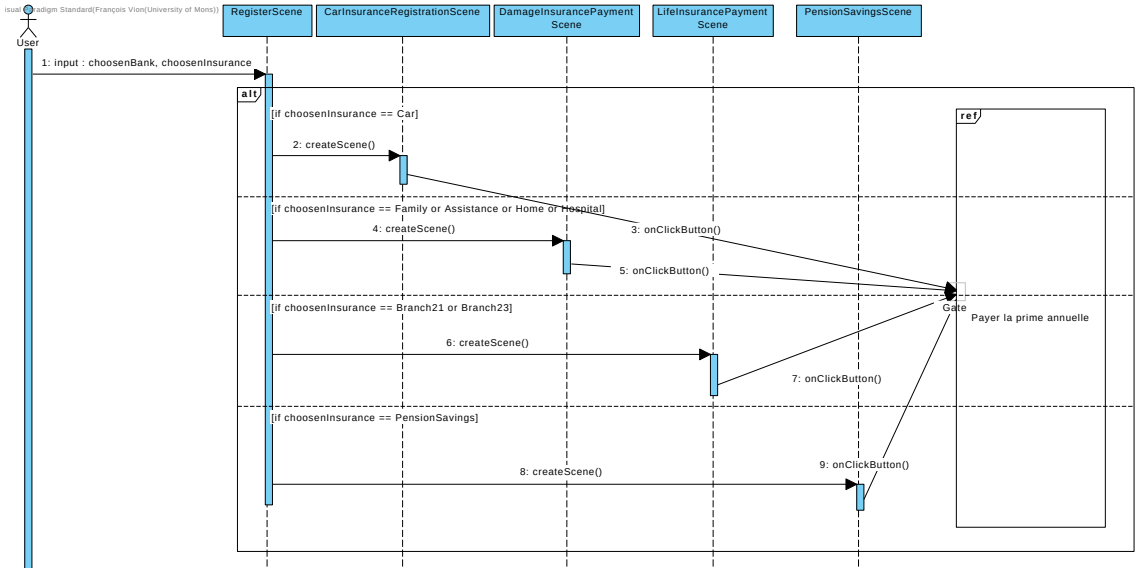


FIGURE 4.7 – Diagramme de séquence du use case "S'inscrire à une assurance"

S'inscrire à une assurance commence par le choix de banque et d'assurance par l'utilisateur (1). Ensuite, plusieurs cas se distinguent même s' ils fonctionnent de la même manière. Une scene correspondante au type d'assurance est créée (2, 4,6 et 8) et reprend les informations sur l'assurance choisie ainsi que le choix des options possibles dans le cas de l'assurance voiture. Enfin, en cliquant sur le bouton de la fenêtre, on accède à la fenêtre de payement afin de confirmer l'inscription (3,5,7 et 9). Le payement ne sera pas décrit car il s'agit exactement du cas cité précédemment dans le rapport.

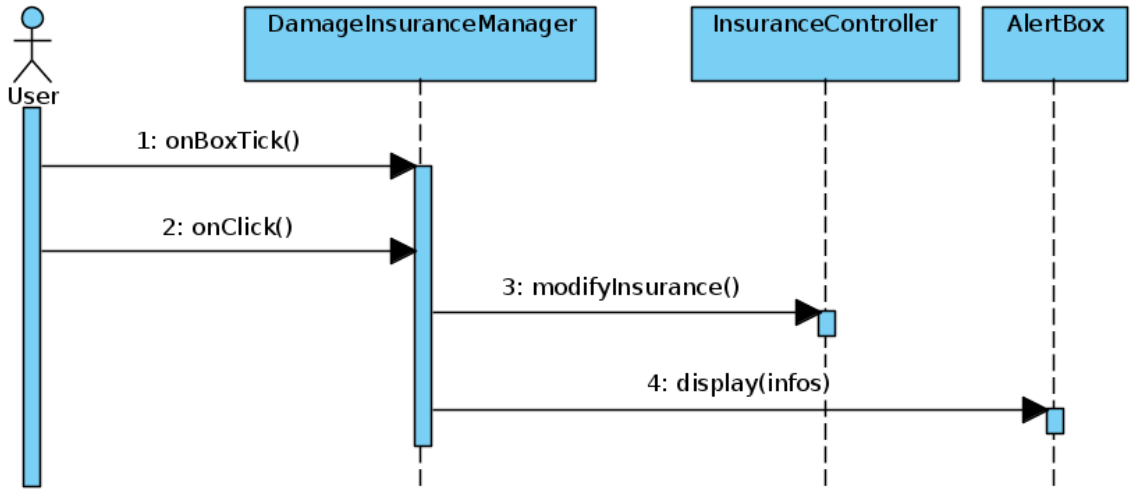


FIGURE 4.8 – Diagramme de séquence du use case "Modifier les paramètres d'une assurance"

Modifier les paramètres d'une assurance se fait via une scene avec des boutons "checkBox". L'utilisateur commence par les cocher ou décocher (1) puis clique sur le bouton "save changes" (2). Une fois cela fait, la scene fait un appel à l'API pour modifier l'assurance dans la base de données (3). Enfin, une fenêtre reprenant les informations suite à la modification est affichée.

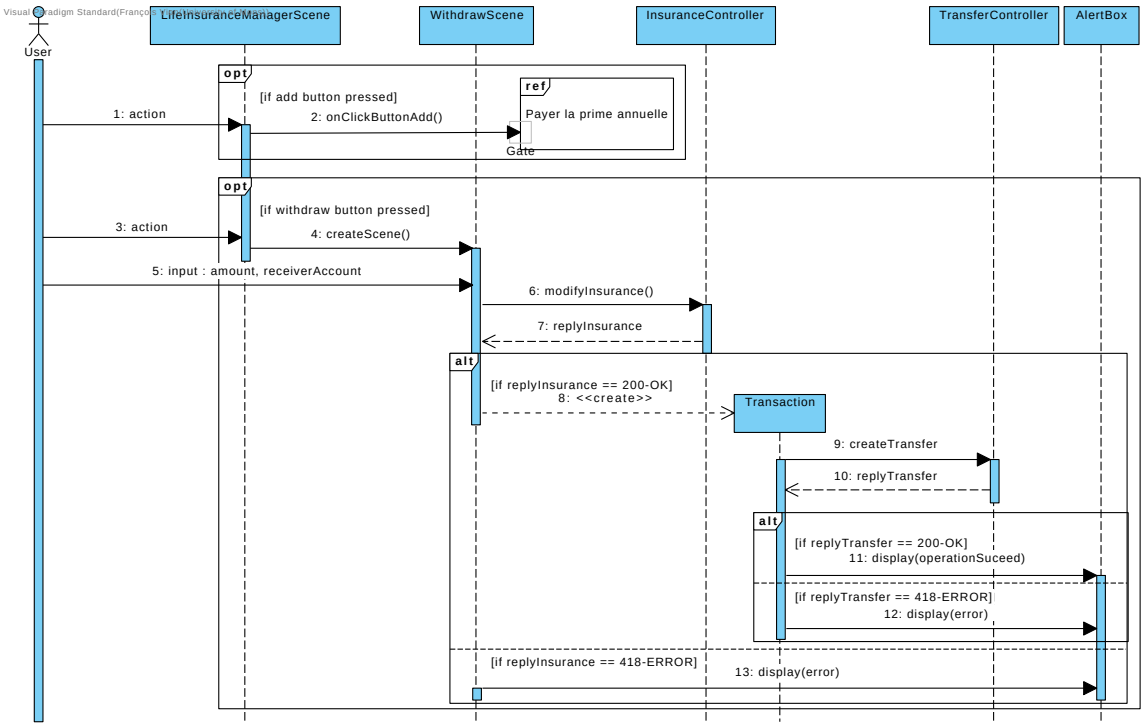


FIGURE 4.9 – Diagramme de séquence du use case "Gérer le montant sur une assurance"

On peut gérer le montant sur une assurance seulement pour les assurances vies. Cela se fait via un ajout ou un retrait d'argent. Si l'utilisateur clique sur le bouton d'ajout (1), il est redirigé vers le payement d'une prime (2) déjà décrite dans ce rapport. Si il clique sur le bouton de retrait (3), une scene est créée (4) dans laquelle l'utilisateur rentrera le montant qu'il souhaite retirer ainsi que le compte sur lequel il compte le recevoir (5). La scene fait ensuite un appel à l'API pour modifier l'assurance (6), celle-ci renvoie une réponse (7). Si la réponse est bonne, alors un objet Transaction est créé (8). Celui-ci va faire un appel à l'API pour créer un transfert (9) et recevoir une réponse de celle-ci (10). Si cette réponse est bonne, alors une fenêtre affichera que l'opération s'est effectuée avec succes. Si l'une des deux réponse est mauvase, une fenêtre affichera qu'il y a eu une erreur.

4.2 Diagrammes de conception UML : application institution

4.2.1 Use case diagram

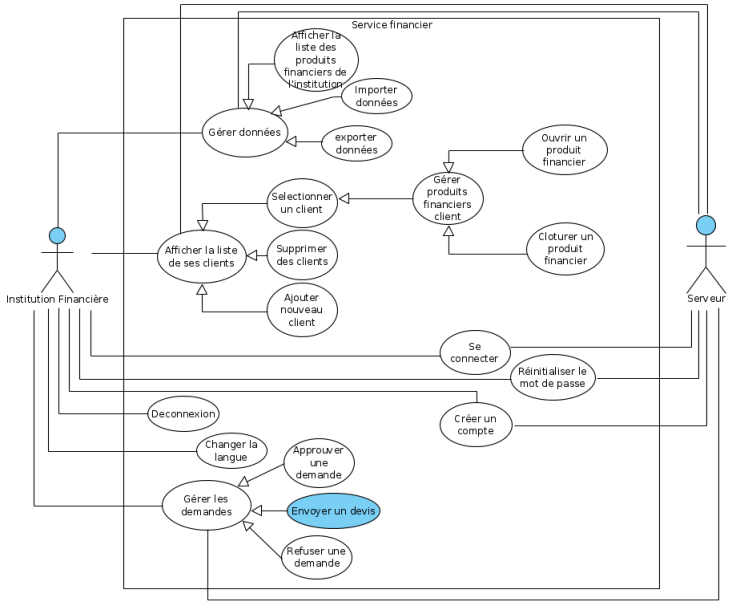


FIGURE 4.10 – Use case de l'application banque pour l'extension assurance

Le use case de la partie banque n'ajoute que le cas "Envoyer un devis" qui est une spécification de gérer les demandes. Il sera néanmoins détaillé comme les autres à part.

4.2.2 Interaction overview diagram

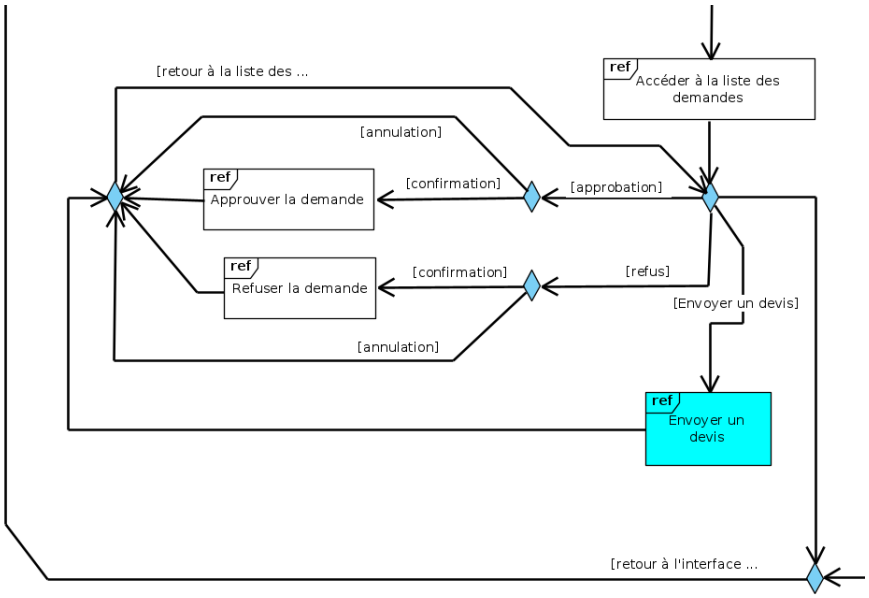


FIGURE 4.11 – Interaction overview diagram de l'application banque pour l'extension assurance

L'interaction overview diagram de l'application banque de l'extension assurance n'ajoute que l'option d'envoyer un devis lorsque l'on accède à la liste des demandes.

4.2.3 Class diagram

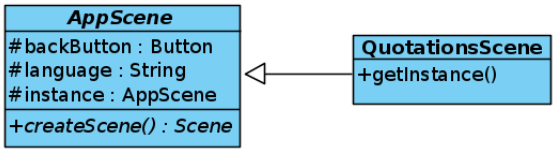


FIGURE 4.12 – Ajout au diagramme de classe de la partie GUI de l'extension assurance

Comme dit dans la partie client de l'extension assurance, les classes de logique et d'API ajoutées sont en commun avec la partie banque. Cependant la partie GUI ajoute une scene à l'application à savoir QuotationScene qui permet de gérer les demandes de devis. La classe AppScene est la même que celle des autres diagrammes, elle a juste été isolée ici pour une meilleur lisibilité.

4.2.4 Sequence diagram

Le seul diagramme de séquence de cette partie (Envoyer un devis) pouvant être réalisé sous la forme d'un fonctionnement type tel que décrit dans les diagrammes de séquence de la partie commune, celui-ci ne sera pas décrit. Il s'agit en effet de la création d'un objet via les entrées d'un utilisateur afin d'être ajoutées par après dans la base de données via l'API (deuxième diagramme type).

4.3 Modèle de données

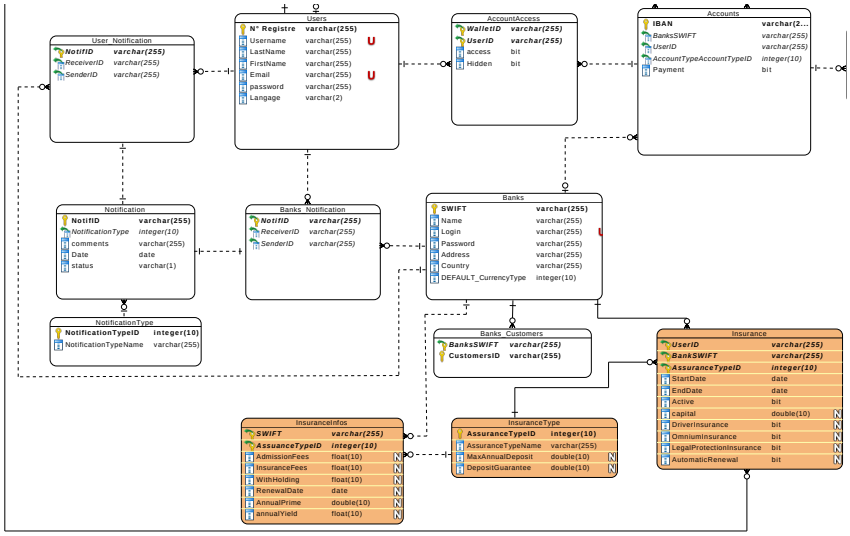


FIGURE 4.13 – Diagramme d'entité-relation comprenant la partie assurance

Pour la base de données, ses entités n'ont pas été modifiées mais cependant trois nouvelles ont été ajoutées. Globalement, il s'agit de données qui ne vont changer que rarement. Tout d'abord, l'entité Insurance contient les données qui dépendent du client comme le capital sur une assurance ou encore les option d'une assurance. Ensuite, l'entité InsuranceInfos reprend toutes les données d'une assurance qui dépendent de la banque comme le prix de l'assurance, le rendement pour les assurances vie, les différents frais ainsi que la date de renouvellement. Enfin, l'entité InsuranceType contient les données qui ne dépendent ni de la banque et ni du client comme le nom de l'assurance, le dépot maximum ou la guarantie de dépôt.

4.4 Maquette de l'interface utilisateur et institution

Introduction Cete section décrit l'utilisation de l'interface graphique de l'extension assurance en se basant sur la maquette de l'interface. Seule les deux dernières fenêtres sont utilisées dans l'application pour banque, toutes les autres font partie de l'application client. Par soucis de clarte, ce diagramme est disponible en annexe du rapport.

Fenêtres disponibles

- Main screen
- Insurance main screen
- Informations on the insurances
- Submit a quote
- Received quotation
- Insurance manager
- Insurance history
- Register for insurances
- Car insurance registration
- Life insurance payment
- Damage insurance payment
- Pension savings insurance payment
- Payment
- Enter PIN to confirm
- Life insurance manager
- Withdraw
- Damage insurance manager
- Pension savings insurance manager
- Requests
- Quotations

Fenêtre *Main screen*
Accès : en se connectant à l'application via la fenêtre Sign in, en ayant terminé un payement, en appuyant sur le bouton back de la fenêtre Insurance main screen ou via les fenêtre Change language ou Change password.
Contenu : les boutons Sign out, Language, Insurance, Financial product, Requests et Change password.

- Le bouton *Sign out* : envoie l'utilisateur sur la fenêtre *Sign out* ;
- Le bouton *Language* : envoie l'utilisateur sur la fenêtre *Change language* ;
- Le bouton *Insurance* : envoie l'utilisateur sur la fenêtre *Insurance main screen*.
- Le bouton *Financial product* : envoie l'utilisateur sur la fenêtre *Financial product*
- Le bouton *Requests* : envoie l'utilisateur sur la fenêtre *Requests*
- Le bouton *Change password* : envoie l'utilisateur sur la fenêtre *Change password*

Fenêtre *Insurance main screen*
Accès : en cliquant sur Insurance dans main screen, en ayant annulée une assurance, ou en cliquant sur le bouton des back des fenêtres Sumit a quote, Informations on the insurances et Insurance manager.
Contenu : Les boutons Back, Insurance manager, Submit a quote, Informations on the insurances

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;

— Le bouton *Insurance manager* : envoie l'utilisateur sur la fenêtre *Insurance manager*
— Le bouton *Submit a quote* : envoie l'utilisateur sur la fenêtre *Submit a quote* ;
— Le bouton *Informations on the insurances* : envoie l'utilisateur sur la fenêtre *Informations on the insurances*

Fenêtre *Informations on the insurances*
Accès : en cliquant sur le bouton Informations on the insurances de la fenêtre Insurance main screen
Contenu : Le bouton back et des zones de texte
— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Insurance main screen* ;
— Les zones de textes contiennent des informations sur les assurances

Fenêtre *Submit a quote*
Accès : en cliquant sur le bouton Submit a quote de la fenêtre Insurance main screenou via le bouton Back de la fenêtre Received quotation
Contenu : Les boutons Back, Received quotation et Submit a quote ainsi que les listes de sélection Choose a bank et Choose an insurance.
— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Insurance main screen* ;
— Le bouton *Reveived quotation* : envoie l'utilisateur sur la fenêtre *Received quotation*
— Le bouton Submit a quote qui effectue la demande de devis.
— La liste sélectionnable Choose a bank.
— La liste sélectionnable Choose an insurance.

Fenêtre *Received quotation*
Accès : en cliquant sur le bouton Received quotation de la fenêtre Submit a quote
Contenu : Les boutons Back et Register for insurance et la liste des devis reçus.
— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Submit a quote* ;
— Le bouton Register for insurance qui envoie l'utilisateur vers la fenêtre Life insurance payment s'il a choisi une assurance vie, vers la fenêtre Damage insurance payment s'il a choisi une assurance famille, assistance, maison ou hopital, vers la fenêtre Car insurance registration s'il a choisi une assurance voiture et vers la fenêtre Pension savings insurance payment s'il a choisi une assurance épargne pension.
— La liste sélectionnable des devis reçus

Fenêtre *Insurance manager*
Accès : en cliquant sur le bouton Insurance manager de la fenêtre Insurance main screen ou en cliquand sur les boutons Back des fenêtres Insurance history, Life insurance manager, Pension savings insurance manager et Damage insurance manager.
Contenu : Les boutons Back, Register for insurance, Pay the annual premium, See insurance's history, Manage insurance et Cancel insurance ainsi que la liste sélectionnable Choose an insurance.
— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Insurance main screen* ;
— Le bouton *Register for insurance* : envoie l'utilisateur sur la fenêtre *Register for insurance*
— Le bouton Pay the annual premium qui envoie l'utilisateur vers la fenêtre Life insurance payment s'il a choisi une assurance vie, vers la fenêtre Damage insurance payment s'il a choisi une assurance famille, assistance, maison ou hopital, vers la fenêtre Car insurance registration s'il a choisi une

assurance voiture et vers la fenêtre Pension savings insurance payment s'il a choisi une assurance épargne pension.
- Le bouton *See insurance's history* : envoie l'utilisateur sur la fenêtre *Insurance history*
- Le bouton Manage insurace qui envoie l'utilisateur vers la fenêtre Life insurance manager s'il a choisi une assurance vie, vers la fenêtre Damage insurance manager s'il s'agit d'une assurance de dégâts ou vers la fenêtre Pension savings insurance manager s'il s'agit d'une assurance épargne pension.
- Le bouton Cancel insurance qui annule l'assurance selectionnée.
- La liste sélectionnable des assurances de l'utilisateur.

Fenêtre *Insurance history*
Accès : en cliquant sur le bouton See insurance's history de la fenêtre Insurance manager.
Contenu : Un bouton Back et la liste de l'historique des assurances.
- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Insurance manager* ;
- La liste de l'historique des assurances avec les assurances terminée surlignées en gris.

Fenêtre *Register for insurances*
Accès : en cliquant sur le bouton Register for insurance de la fenêtre Insurance manager ou Received quotation ou en cliquand sur le bouton Back des fenêtres Car insurance registration, Life insurance payment, Damage insurance payment ou Pension savings payment
Contenu : Les boutons Back et Registration ainsi que les listes sélectionnables Choose a bank et Choose an insurance.
- Le bouton Back qui renvoie l'utilisateur à la fenêtre précédente.
- La liste sélectionnable Choose a bank.
- La liste sélectionnable Choose an insurance.
- Le bouton Registration qui envoie l'utilisateur vers la fenêtre Life insurance payment s'il a choisi une assurance vie, vers la fenêtre Damage insurance payment s'il a choisi une assurance famille, assistance, maison ou hopital, vers la fenêtre Car insurance registration s'il a choisi une assurance voiture et vers la fenêtre Pension savings insurance payment s'il a choisi une assurance épargne pension.

Fenêtre *Car insurance registration*
Accès : en cliquant sur le bouton Registration de la fenêtre Register for insurance ou le bouton Register for insurance de la fenêtre Received quotation après voir selectionné une assurance voiture.
Contenu : Les boutons Back et Pay the annual prime, les informations sur l'assurance choisie ainsi que des checkBox pour les options.
- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Register for insurance* ;
- Les informations sur le type d'assurance
- Les checkBox Omnium insurance, Driver insurance et Legal protection Insurance
- Le bouton *Pay the annual prime* : envoie l'utilisateur sur la fenêtre *Payment*

Fenêtre *Life insurance payment*
Accès : en cliquant sur le bouton Registration de la fenêtre Register for insurance ou le bouton Register for insurance de la fenêtre Received quotation après voir selectionné une assurance vie.
Contenu : Les boutons Back et Add funds ainsi que les informations sur l'assurance.
- Le bouton renvoie à la fenêtre précedent, soit Register for insurance, soit Received quotation. ;

— Les informations sur l'assurance
— Le bouton *Add funds* : envoie l'utilisateur sur la fenêtre *Payment*

Fenêtre *Damage insurance payment*
Accès : en cliquant sur le bouton Registration de la fenêtre Register for insurance ou le bouton Register for insurance de la fenêtre Received quotation après voir selectionné une assurance famille, assistance, habitation ou hopital.
Contenu : Ls boutons Back et Pay the annual prime ainsi que les informations sur l'assurance.
— Le bouton renvoie à la fenêtre précedent, soit Register for insurance, soit Received quotation. ;
— Les informations sur l'assurance
— Le bouton *Pay the annual prime* : envoie l'utilisateur sur la fenêtre *Payment*

Fenêtre *Pension savings insurance payment*
Accès : en cliquant sur le bouton Registration de la fenêtre Register for insurance ou le bouton Register for insurance de la fenêtre Received quotation après voir selectionné une assurance épargne pension.
Contenu : Ls boutons Back et Pay the annual prime ainsi que les informations sur l'assurance.
— Le bouton renvoie à la fenêtre précedent, soit Register for insurance, soit Received quotation. ;
— Les informations sur l'assurance
— Le bouton *Pay the annual prime* : envoie l'utilisateur sur la fenêtre *Payment*

Fenêtre *Payment*
Accès : en cliquant sur le bouton Pay the annual prim ou add funds des fenêtres Life insurance payment, Damage insurance payment, Pension savings insurance payment, Pension savings insurance manager ou Life insurance manager.
Contenu : Les boutons Back et Confirm, les textFields Amount, IBAN, Message et Date qui sont tous vérouillées sauf les cas où il s'agit d'un montant libre,alors celui-ci est dévérouiller. Une liste déroulante Account.
— Le bouton Back renvoie à la fenêtre précédente.
— Le champ de texte *Amount* : permet à l'utilisateur d'entrer le montant libre
— La liste déroulante Account qui permet de choisir le compte débité.
— Le champ de texte *IBAN* : permet à l'utilisateur d'entrer le compte IBAN du receveur
— Le champ de texte *Message* : permet à l'utilisateur d'entrer une communication pour la transaction
— Le champ de texte *Date* : permet à l'utilisateur d'entrer la date souhaitée pour la transaction

Fenêtre *Enter PIN to confirm*
Accès : en cliquant sur le bouton Confirm de la fenêtre Payment ou Withdraw de la fenêtre Withdraw.
Contenu : Les boutons Back et Confirm. Un textField pour rentrer son PIN ainsi que 10 boutons numérotés pour le taper à la souris.
— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Payment* ;
— Le champ de texte *PIN* : permet à l'utilisateur d'entrer son code PIN
— Les 10 boutons numérotés qui entrent les numéros dans le textField PIN
— Le bouton Confirm qui effectue la transaction si le code PIN est correct, si c'est le cas, l'utilisateur est redirigé vers la fenêtre Main screen.

Fenêtre *Life insurance manager*
Accès : en cliquant sur le bouton Manage insurance de la fenêtre Insurance manager en ayant selectionné une assurance vie.
Contenu : Les boutons Back, Add funds et Withdraw funds ainsi que des informations sur l'assurance.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Insurance manager* ;
- Le bouton *Add funds* : envoie l'utilisateur sur la fenêtre *Payment*
- Le bouton *Withdraw funds* : envoie l'utilisateur sur la fenêtre *Withdraw*
- Les informations de l'assurance.

Fenêtre *Withdraw*
Accès : en cliquant sur le bouton Withdraw funds de la fenêtre Life insurance manager.
Contenu : Les boutons Back et Withdraw, le textField amount et la liste déroulante Receiver account.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Life insurance manager* ;
- Le bouton *Withdraw* : envoie l'utilisateur sur la fenêtre *Enter PIN to confirm*
- Le champ de texte *Amount* : permet à l'utilisateur d'entrer le montant a retirer
- La liste déroulante pour choisir le compte receveur.

Fenêtre *Damage insurance manager*
Accès : en cliquant sur le bouton Manage insurance de la fenêtre Insurance manager en ayant selectionné une assurance de dégâts.
Contenu : Les boutons Back et Make changes, les informations sur l'assurance choisie et les checkBox Automatic renewal et Omnium insurance, Driver insurance et Legal protection insurance dans le cas d'une assurance voiture.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Insurance manager* ;
- Le bouton Make change qui effectue les changements
- La checkBox Automatic renewal a cocher/décocher
- Les checkBox Omnium insurance, Driver insurance et Legal protection insurance a cocher/décocher dans le cas d'une assurance voiture.

Fenêtre *Pension savings insurance manager*
Accès : en cliquant sur le bouton Manage insurance de la fenêtre Insurance manager en ayant selectionné une assurance épargne pension.
Contenu : Les boutons Back, Add funds ainsi que des informations sur l'assurance.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Insurance manager* ;
- Le bouton *Add funds* : envoie l'utilisateur sur la fenêtre *Payment*
- Les informations de l'assurance.

Fenêtre *Requests*
Accès : en cliquant sur le bouton Requests de la fenêtre main screen
Contenu : Les boutons Back, Quotations, Transfer permission requests et Portfolio requests

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;
- Le bouton *Quotations* : envoie l'utilisateur sur la fenêtre *Quotations*
- Le bouton *Transfer permission requests* : envoie l'utilisateur sur la fenêtre *Transfer permission requests*
- Le bouton *Portfolio requests* : envoie l'utilisateur sur la fenêtre *Portfolio requests*

Fenêtre *Quotations*
Accès : en cliquant sur le bouton Quotations de la fenêtre Requests
Contenu : Les boutons Back et Send quotation ainsi que la liste sélectionnable des demandes de devis

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Requests* ;
- Le bouton Send quotation qui envoie le devis au client.
- La liste sélectionnable des demande de devis reçues.

Chapitre 5

Extension D : Paiements et gestion de fraudes - Arnaud MOREAU

5.1 Diagrammes de conception UML : application client

5.1.1 Use case diagram

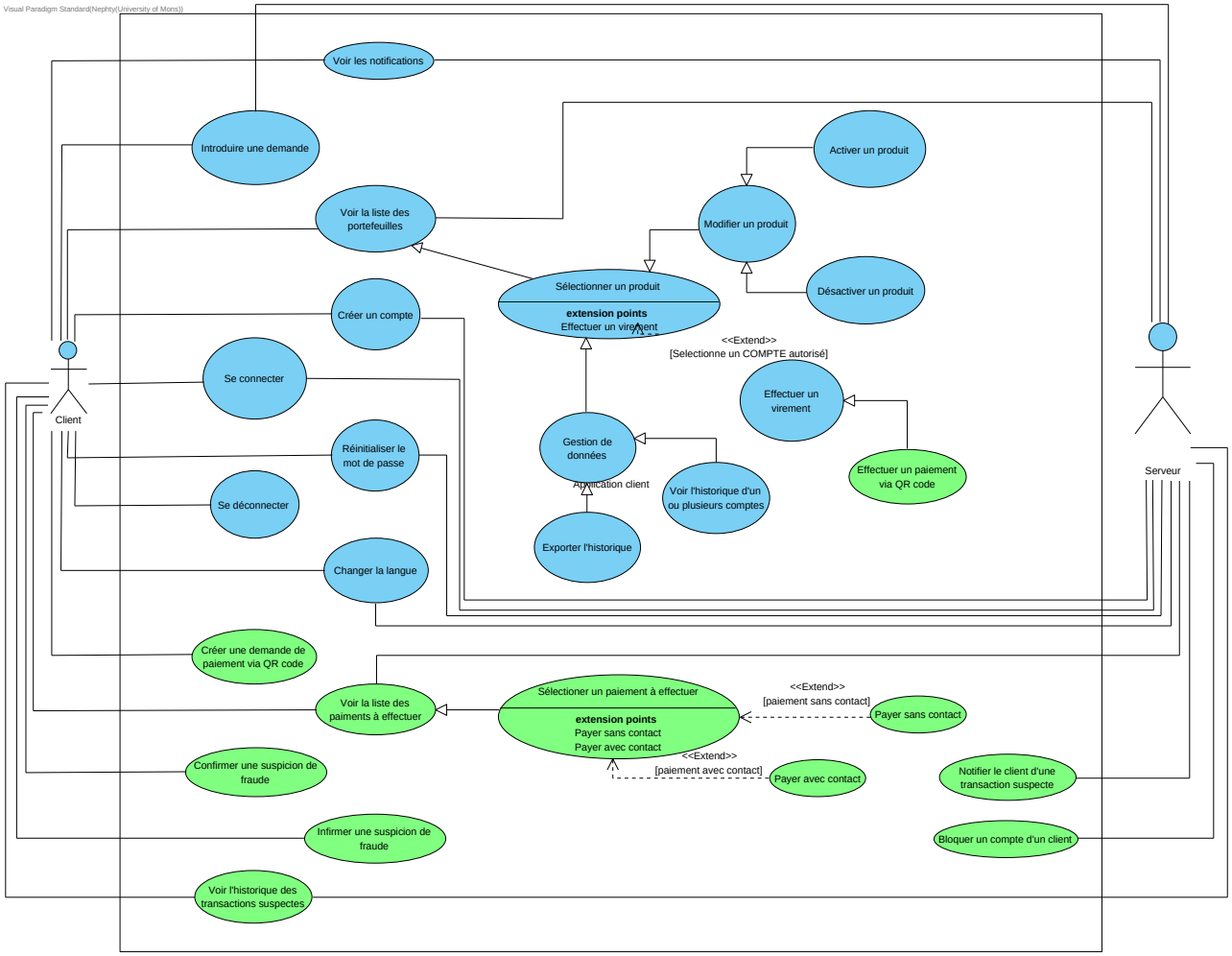


FIGURE 5.1 – Nouveau diagramme de cas d'utilisation de l'application client

Structure Globalement, les nouveaux cas d'utilisation sont séparés en trois catégories, comme le spécifie l'énoncé : les paiements par QR code, la simulation de paiement avec et sans contact et la gestion de fraude. Les cas d'utilisation qui ont été ajoutés correspondent aux fonctionnalités principales de l'application. Une description semi-structurée de chacun des cas d'utilisation peut être trouvée en annexe.

5.1.2 Interaction overview diagram

Nouvelles références :

- Créer une demande de paiement via QR code ;
- Effectuer un paiement via QR code ;
- Voir l'historique des transactions suspectes ;
- Infirmer une suspicion de fraude ;
- Confirmer une suspicion de fraude ;
- Voir les paiements à effectuer ;
- Sélectionner un paiement à effectuer ;
- Payer avec contact ;
- Payer sans contact.

Créer une demande de paiement via QR code L'utilisateur a la possibilité de créer une image contenant un QR code, qui pourra transmettre des données (montant, destinataire et message) et qui pourra plus tard être utilisé par un autre utilisateur afin d'effectuer une transaction correspondant aux données.

Effectuer un paiement via QR code L'utilisateur peut aussi sélectionner une image dans ses fichiers locaux contenant un QR code créé par un autre utilisateur auparavant. Ce QR code pourra alors être lu et une transaction pourra avoir lieu.

Voir l'historique des transactions suspectes Le client peut visualiser l'ensemble des transactions qui auront été marquées comme suspectes dans la base de données. Par la suite, il pourra confirmer ou infirmer cette suspicion.

Infirmer une suspicion de fraude Le client ayant accédé aux transactions suspectes, il pourra infirmer cette suspicion et baisser le niveau de sécurité de son compte. Typiquement, plus ce niveau est élevé, plus l'utilisateur est restreint, afin d'empêcher toute utilisation frauduleuse de son compte.

Confirmer une suspicion de fraude Le client ayant accédé aux transactions suspectes, il pourra confirmer cette suspicion et augmenter le niveau de sécurité de son compte.

Voir les paiements à effectuer L'utilisateur peut accéder à une liste de paiements prévus, situés dans un fichier local au format JSON. Cette liste de paiements peut être réglée, moyennant un ou plusieurs paiements avec ou sans contact.

Sélectionner un paiement à effectuer L'utilisateur peut sélectionner un paiement parmi ceux proposés dans la liste, afin d'effectuer la transaction.

Payer avec contact Après avoir sélectionner le paiement qu'il souhaite effectuer, l'utilisateur a la possibilité de payer avec contact, ce qui signifie qu'il devra entrer son code PIN afin de confirmer le paiement.

Payer sans contact Après avoir sélectionner le paiement qu'il souhaite effectuer, l'utilisateur a la possibilité de payer sans contact, ce qui signifie que le paiement sera effectuer sans avoir besoin d'entrer le code PIN. Ce moyen de paiement sera soumis à plus de restrictions (notamment une limite, comme explicité pour l'*Interaction Overview Diagram* de l'application institution) afin d'éviter les utilisations frauduleuses.

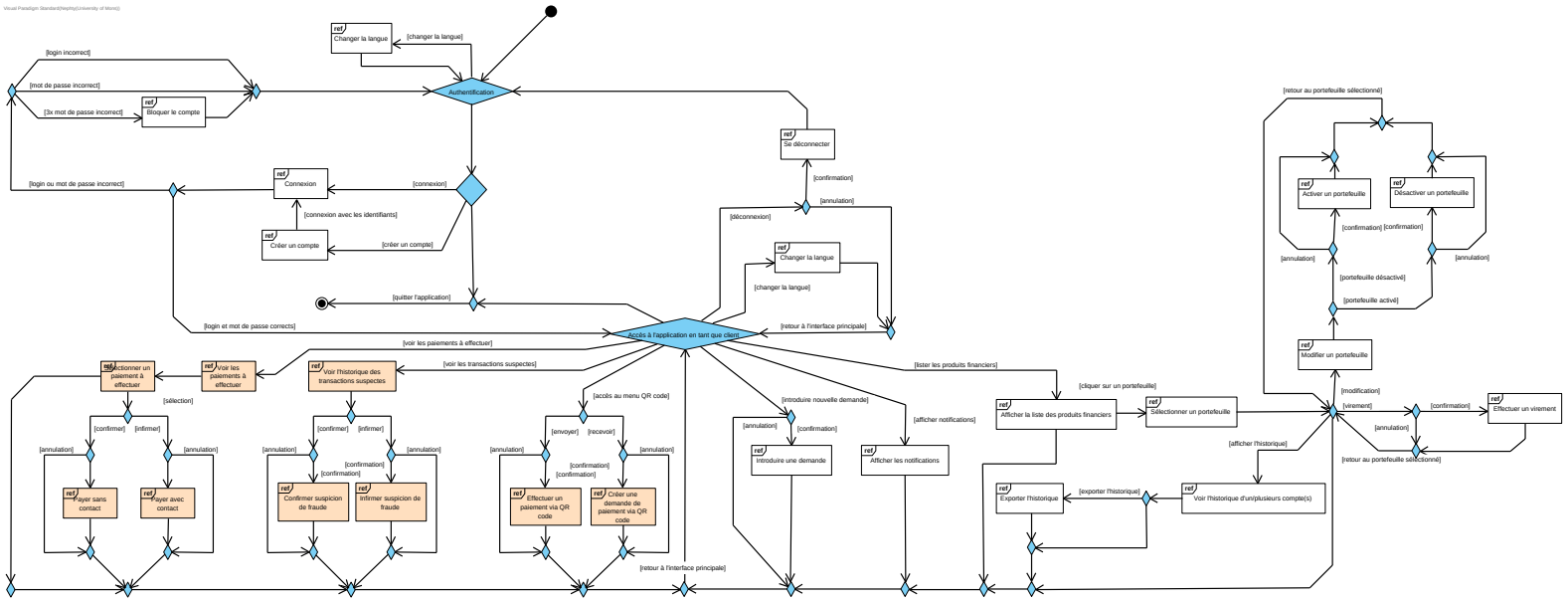


FIGURE 5.2 – Nouveau diagramme d'interaction de l'application client

5.1.3 Class diagram

Structure Le diagramme de classe de l'application client suit globalement la même structure que celui de la partie commune. Des ajouts ont été faits au GUI : de nouvelles fenêtres requises par les nouvelles fonctionnalités sont disponibles et respectent toujours le design pattern Singleton. Un utilitaire (classe *QRCode*) a été ajouté à la partie logique, ainsi qu'un ensemble d'attributs qui serviront notamment à la détection de fraude et à l'imposition de limites par l'institution financière. Trois énumérations ont aussi été ajoutées : *SEPACountries*, *Country* et *State*, servant respectivement à identifier les pays faisant partie de la zone SEPA, les pays à partir desquels on peut effectuer une transaction et l'état d'une transaction. Quelques méthodes ont été ajoutés à l'API afin d'obtenir certaines données plus facilement, sans devoir implémenter de nouvelles méthodes dans le code de l'application.

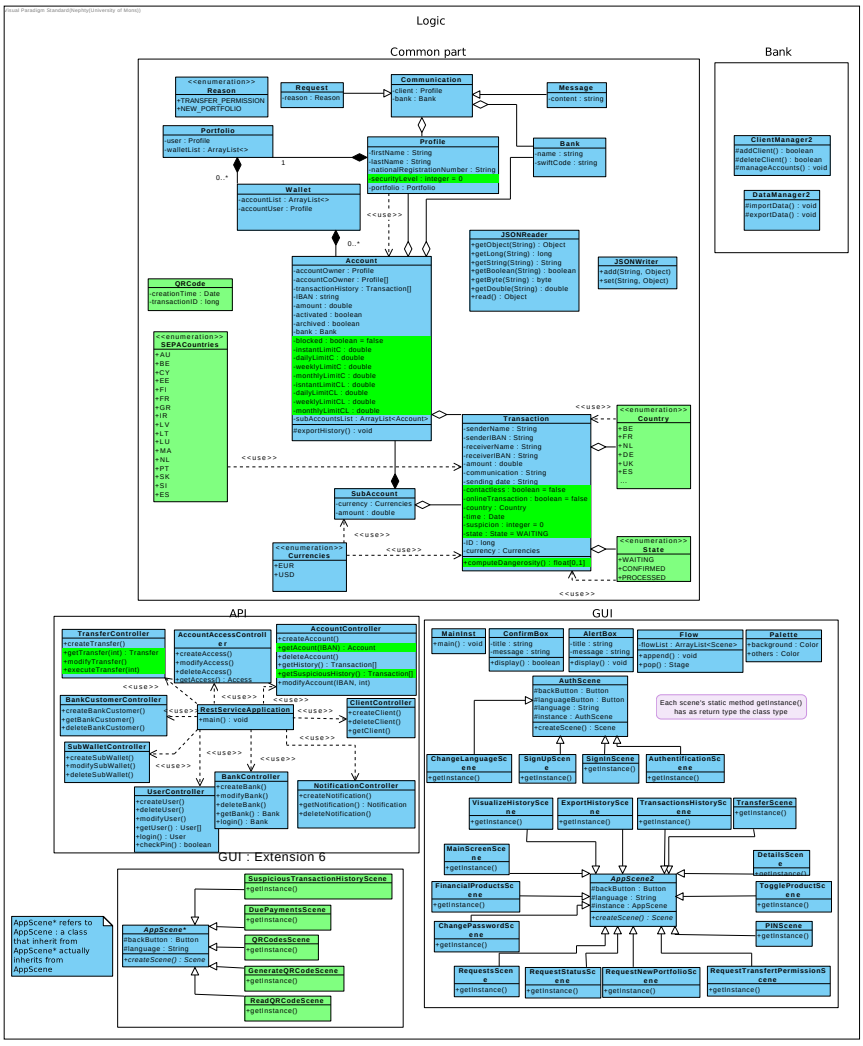


FIGURE 5.3 – Nouveau diagramme de classes de l'application client

5.1.4 Sequence diagram

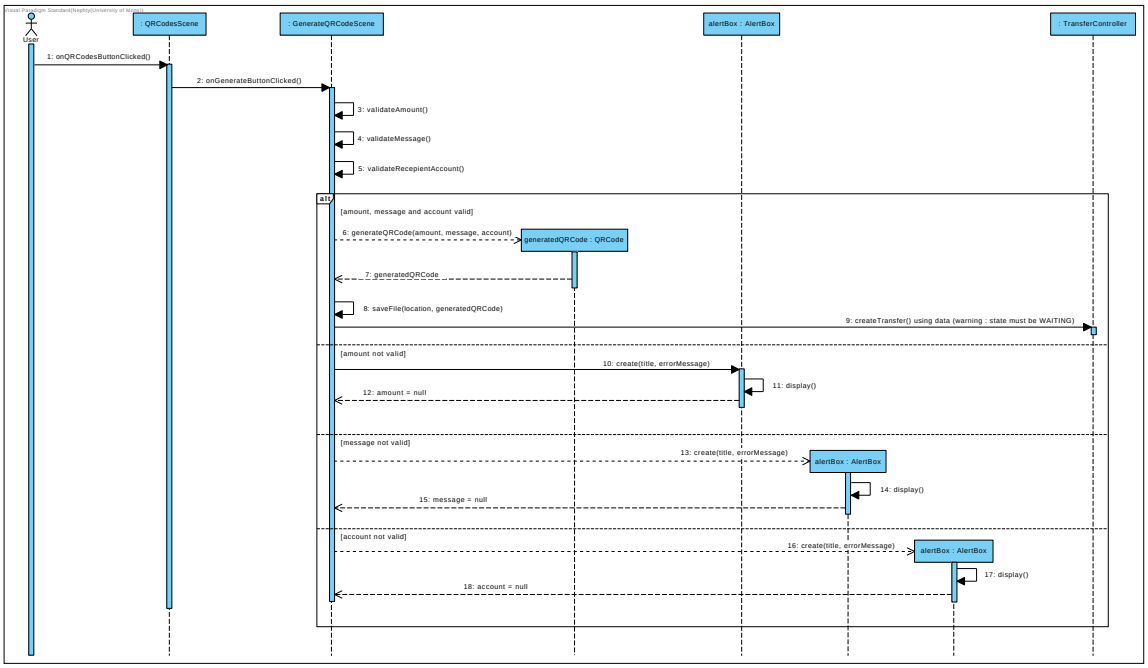


FIGURE 5.4 – Créer une demande de paiement via QR code

Description La création de demande de paiement via QR code se déroule en plusieurs étapes, en suivant le scénario principal décrit dans la description semi-structurée du cas d'utilisation correspondant. Afin d'optimiser les communications avec le serveur, tout se déroule localement, jusqu'à la fin du processus, où la transaction en attente est ajouté à la base de données (9). Toutes les vérifications (3) (4) (5) locales sont effectuées afin de distinguer chacun des scénarios alternatifs et donner un retour adapté à l'utilisateur via différentes *AlertBox*.

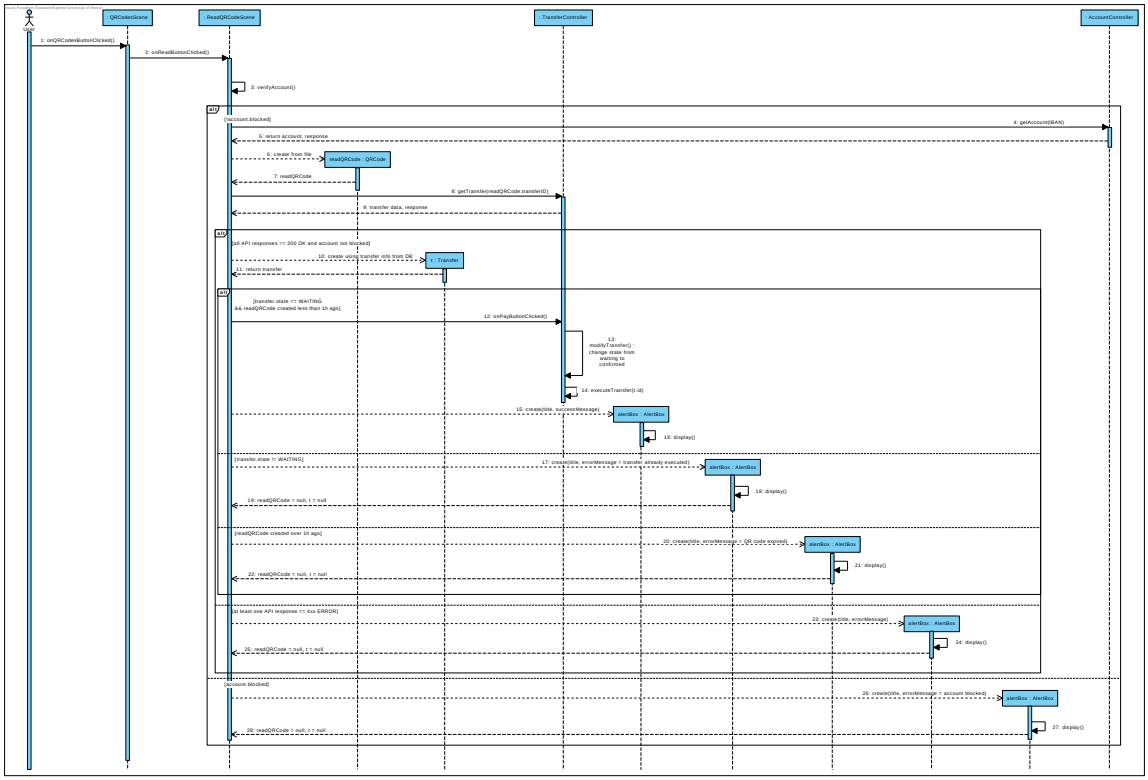


FIGURE 5.5 – Effectuer un paiement via QR code

Description Le paiement via QR code communique avec le serveur afin de récupérer les informations du compte (4) et, si le compte n'est pas bloqué, les informations du transfert à effectuer (8). Si aucune réponse de l'API ne lève d'erreur, le processus commence, les vérifications sont effectuées et si tout est OK, l'état du transfert en attente dans la base de données est changé à "confirmé" (13) et le transfer est effectué (14) (ce qui changera son état à "traité").

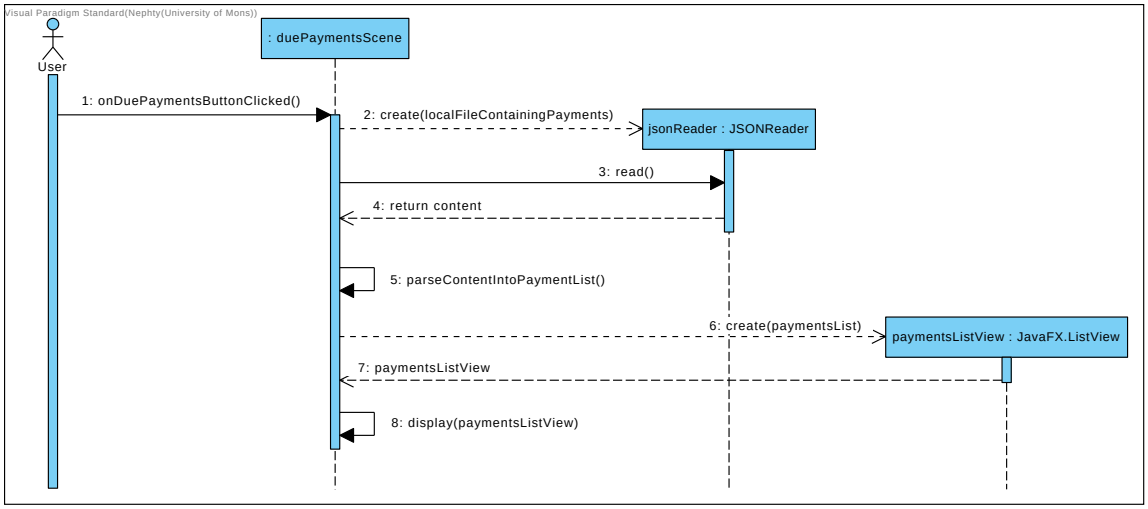


FIGURE 5.6 – Voir la liste des paiements à effectuer

Description Afin de visualiser la liste des paiements à effectuer, un lecteur de fichier JSON est utilisé et récupère les données stockées localement (3). Aucune communication serveur ne doit être effectuée. Le contenu du fichier local est ajouté à une *ListView* (objet *JavaFX*) (5) et l'utilisateur peut consulter les paiements.

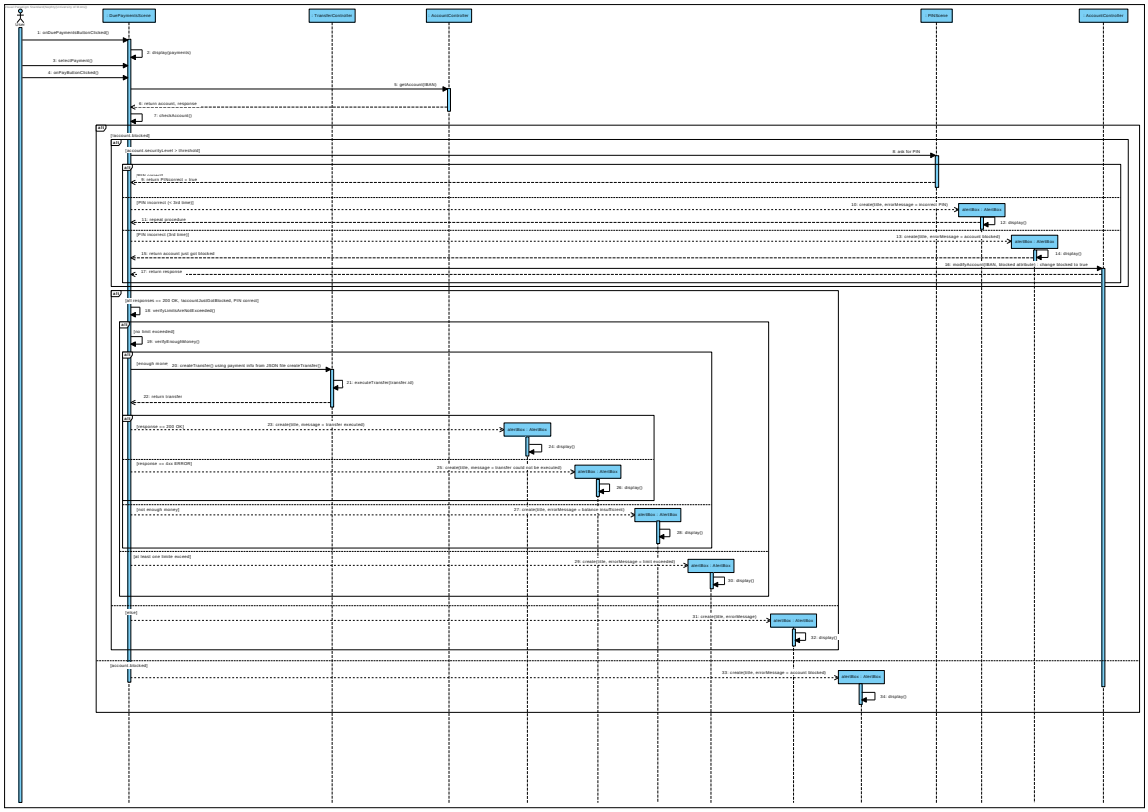


FIGURE 5.7 – Payer sans contact

Description Un paiement sans contact demande deux appels serveur : un pour obtenir le compte avec lequel payer (5) et un pour exécuter le transfert si toutes les conditions sont remplies (9). Le reste de l'exécution se déroule localement et est majoritairement composée de réponses appropriées au multiples erreurs que l'on pourrait rencontrer.

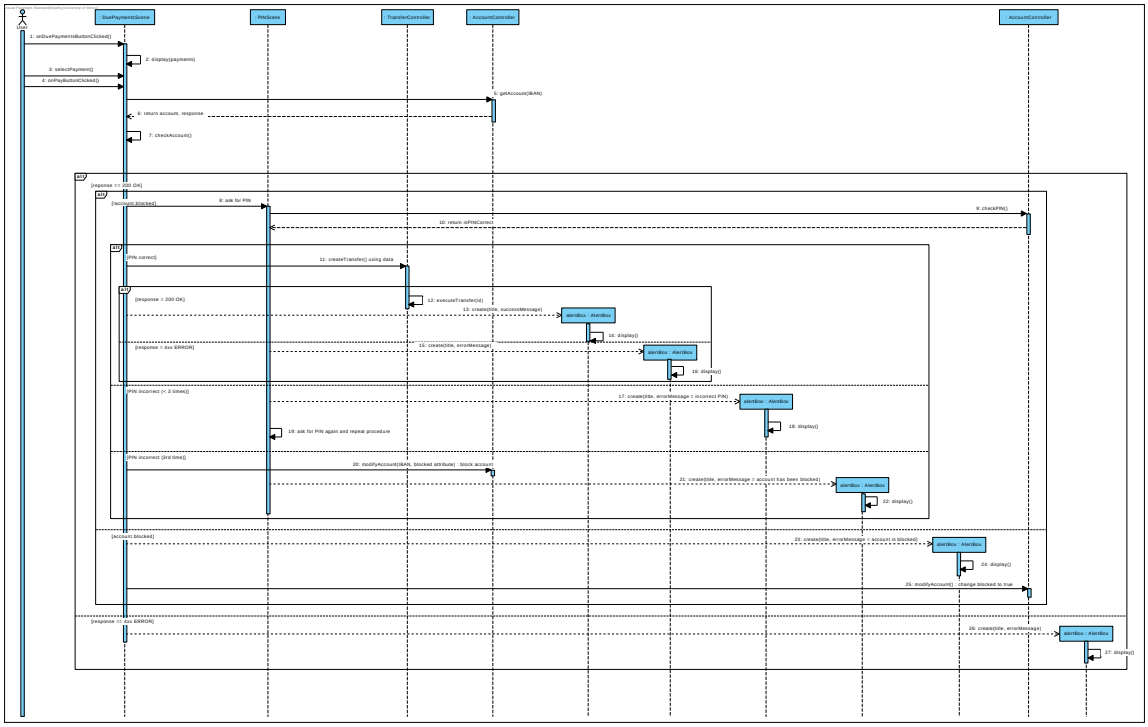


FIGURE 5.8 – Payer avec contact

Description Un paiement avec contact demande deux appels serveur : un pour obtenir le compte avec lequel payer (5) et un pour exécuter le transfert si toutes les conditions sont remplies (9). Le reste de l'exécution se déroule localement et est majoritairement orienté vers la gestion d'erreurs en proposant des réponses appropriées au multiples scénarios que l'on pourrait rencontrer. Le flot d'exécution ressemble beaucoup au paiement sans contact, sauf que l'on demande le code PIN dans le cas d'un paiement avec contact (8).

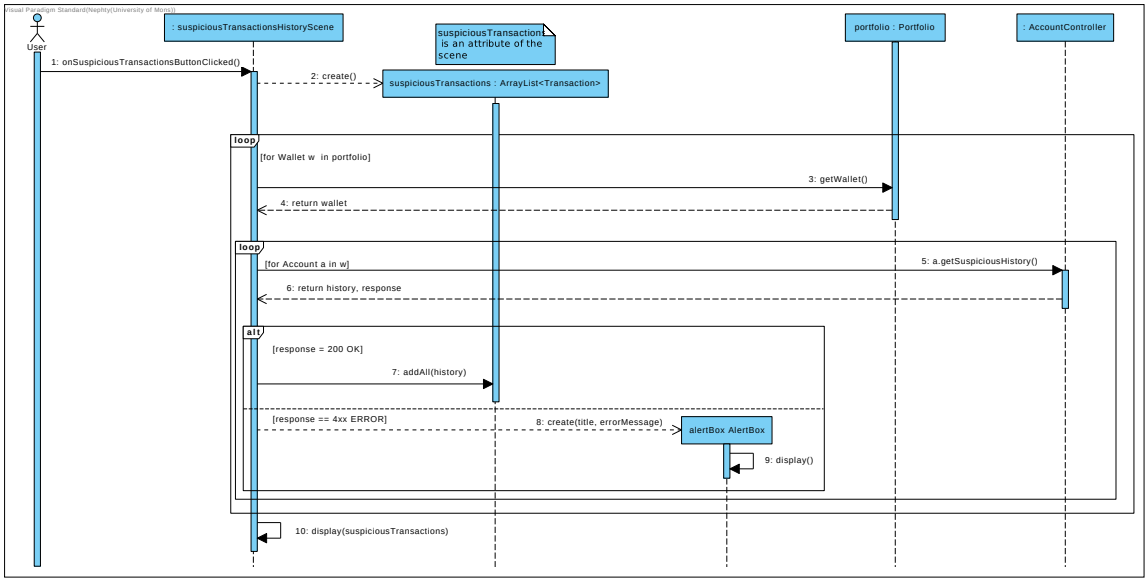


FIGURE 5.9 – Voir l'historique des transactions suspectes

Description L'historique des transactions suspectes est récupéré en utilisant la méthode *getSuspiciousHistory()* sur chaque compte (5). Avec des boucles, nous ajoutons les transactions à un objet *ListView* de *JavaFX* (7) si elles n'ont pas été *reviewed* par l'utilisateur et les transactions sont affichées à l'écran.

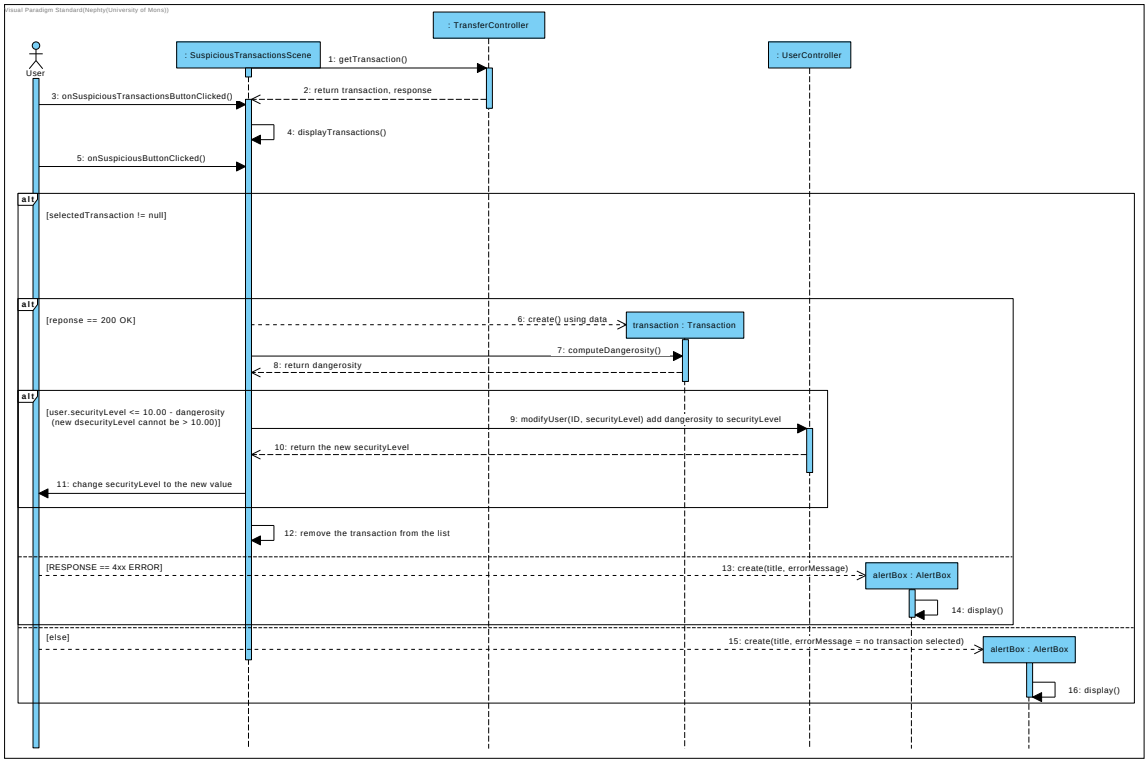


FIGURE 5.10 – Confirmer une suspicion de fraude

Description Afin de confirmer une suspicion de fraude, l'utilisateur doit sélectionner un paiement suspect et confirmer la suspicion (5). Une communication avec le serveur est alors établie afin de modifier le niveau de sécurité du compte (9), si l'augmentation de la valeur ne la fera pas dépasser le maxium autorisé.

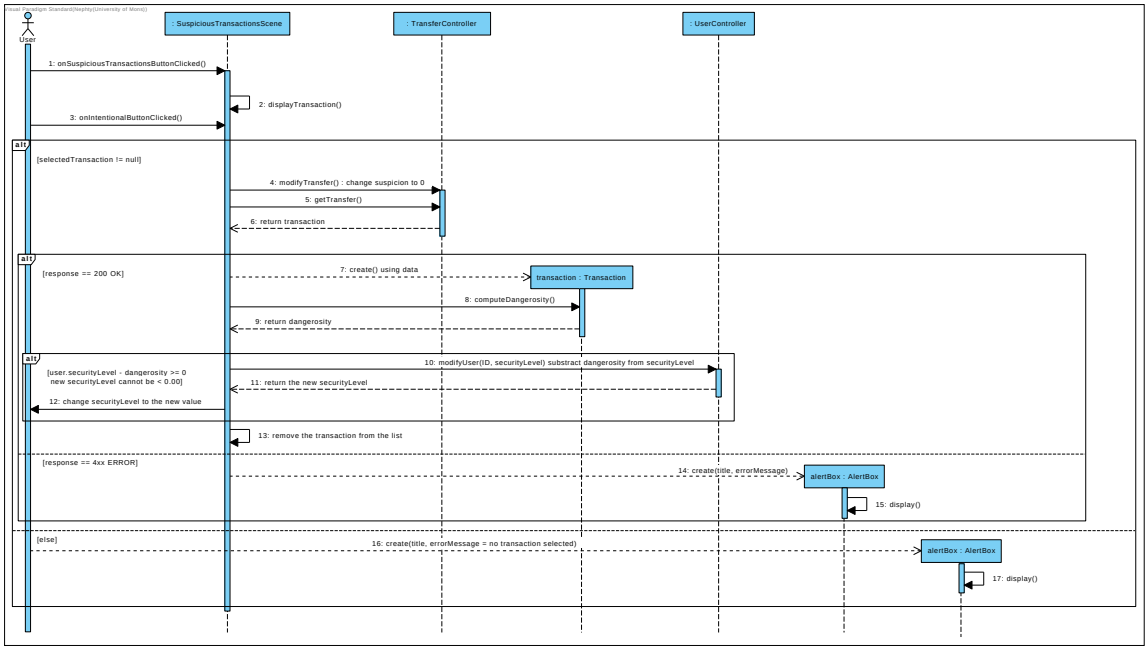


FIGURE 5.11 – Infirmer une suspicion de fraude

Description Afin d'infirmer une suspicion de fraude, l'utilisateur doit sélectionner un paiement suspect et infirmer la suspicion (3). Une communication avec le serveur est alors établie afin de modifier le niveau de sécurité du compte (10), si la réduction de la valeur ne la fera pas franchir le minimum autorisé.

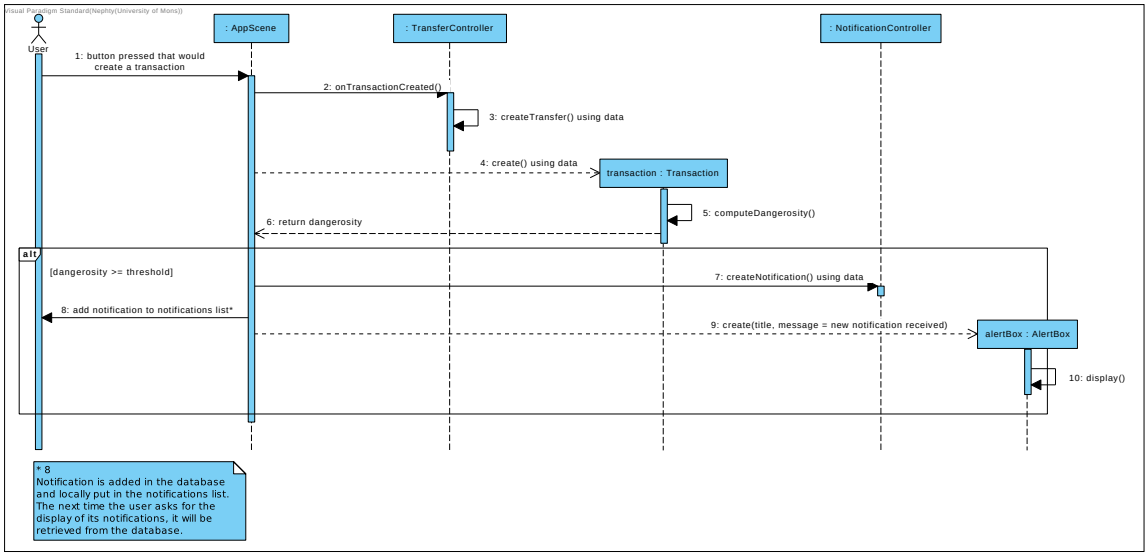
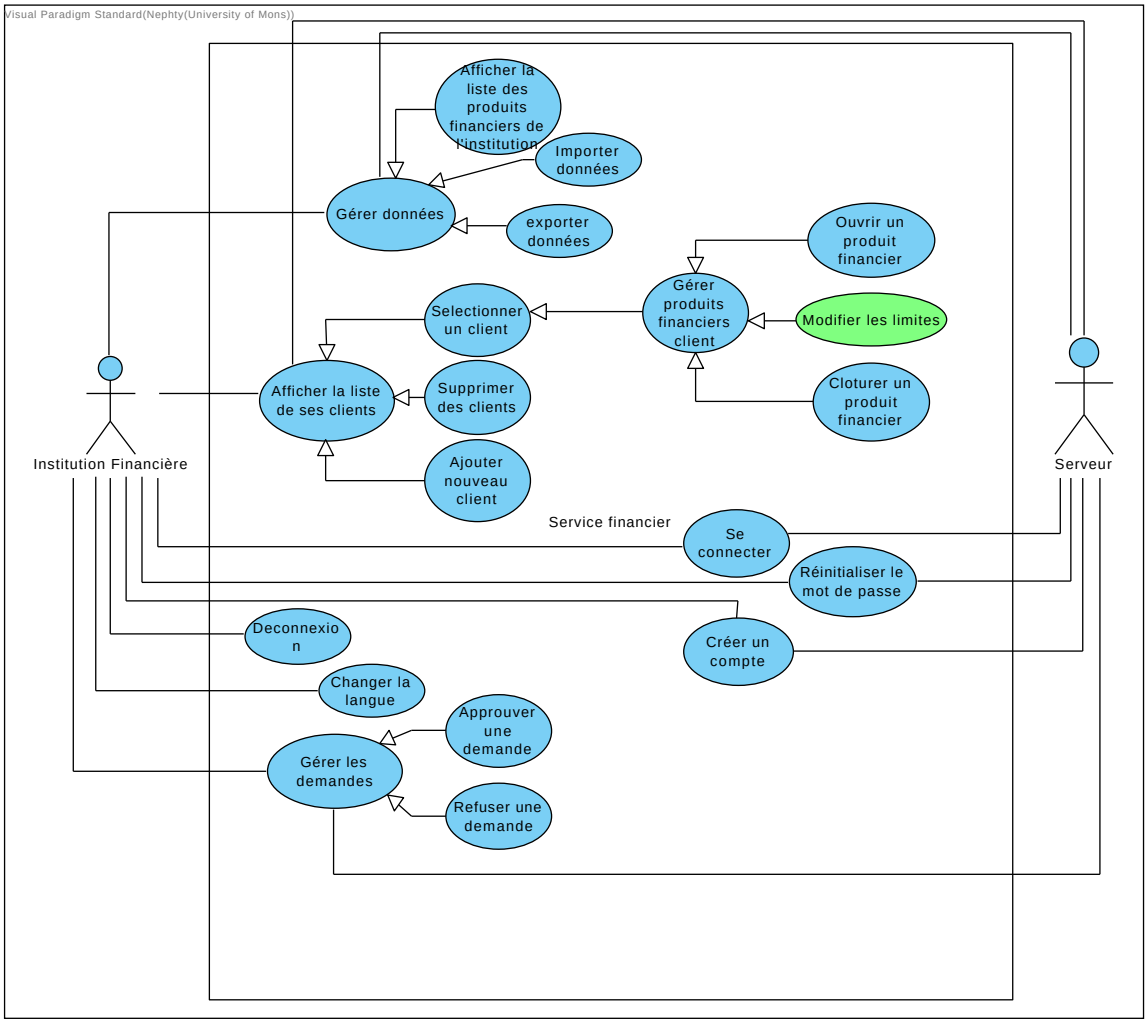


FIGURE 5.12 – Notifier le client d'une transaction suspecte

Description Chaque fois qu'une transaction est effectuée(1) (2) (3), son niveau de suspicion est calculée localement (4) (5), et si elle dépasse un certain seuil, une notification est créée (7) et envoyée à l'utilisateur (8). Cette notification possède un attribut qui l'empêche d'être supprimée pour éviter qu'un personne usurpant l'identité du client ne supprime la notification pour ne pas laisser de traces.

5.2 Diagrammes de conception UML : application institution

5.2.1 Use case diagram



Structure Un seul cas d'utilisation a été ajouté au diagramme de l'institution : *Modifier les limites*. Ce cas d'utilisation correspond à la modification des limites de la quantité d'argent qui peut être transférée depuis un compte, à plusieurs échelles de temps. Une description semi structurée peut être trouvée en annexe.

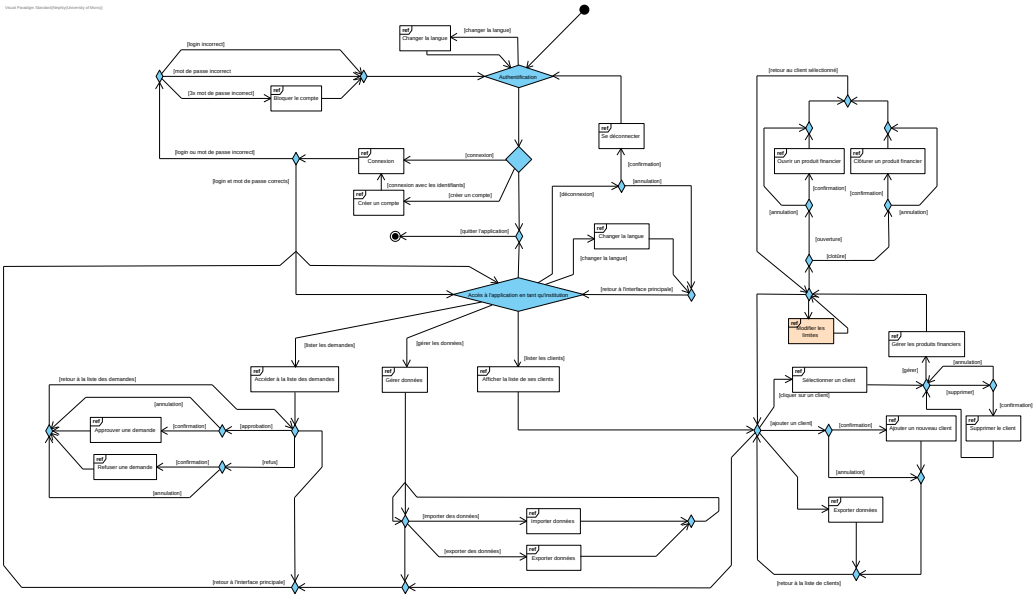
5.2.2 Interaction overview diagram

Nouvelle référence :

1. Modifier les limites.

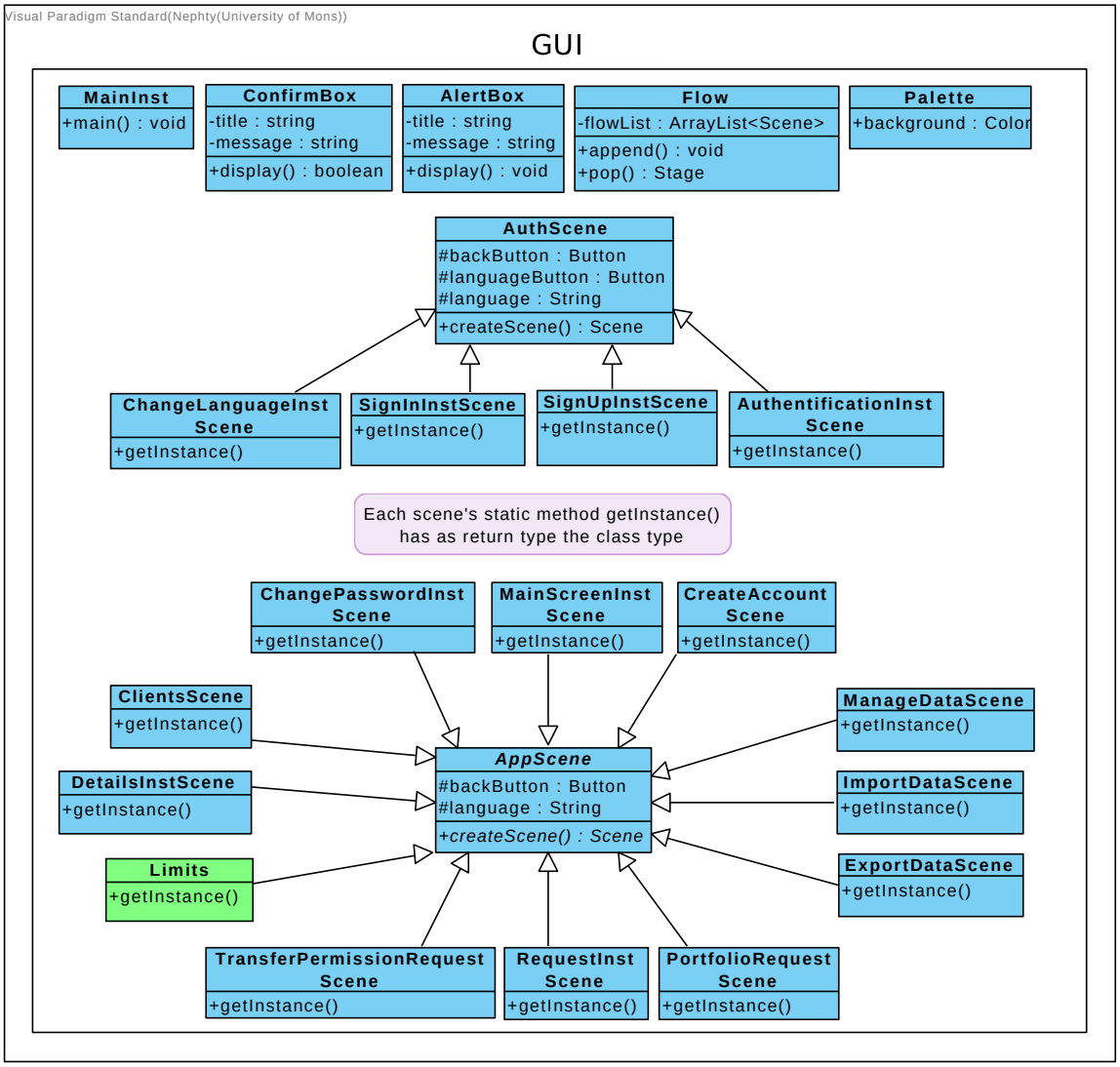
Modifier les limites L'insitution financière peut accéder aux limites de transactions (qui sont fixées par défaut) et les modifier. Ces limites s'appliquent sur les montants des transactions effectuées :

— en une fois ;
— tous les jours ;
— toutes les semaines ;
— tous les mois.



5.2.3 Class diagram

Structure Le diagramme de classe de l'institution est très légèrement modifié : une classe supplémentaire correspondant à une nouvelle fenêtre requise est ajoutée, toujours en respectant le design pattern Singleton. Pour rappel, la partie logique de l'application est commune avec celle de l'application client et peut être trouvée plus haut.



5.2.4 Sequence diagram

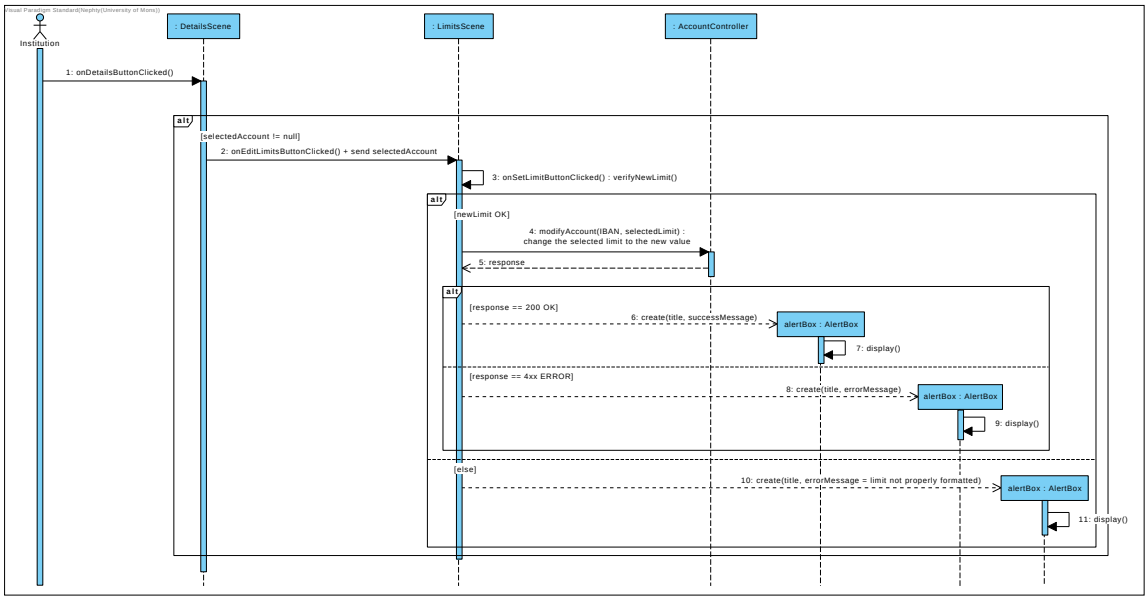


FIGURE 5.13 – Notifier le client d'une transaction suspecte

Description Afin de modifier les limites imposées à un compte, on vérifie d'abord que les entrées fournies sont correctes (notamment que la limite est un nombre à virgule flottante et non pas une chaîne de caractère) (3), et si les vérifications sont passées, un appel serveur est effectué afin de modifier la limite dans la base de données (4).

5.3 Modèle de données

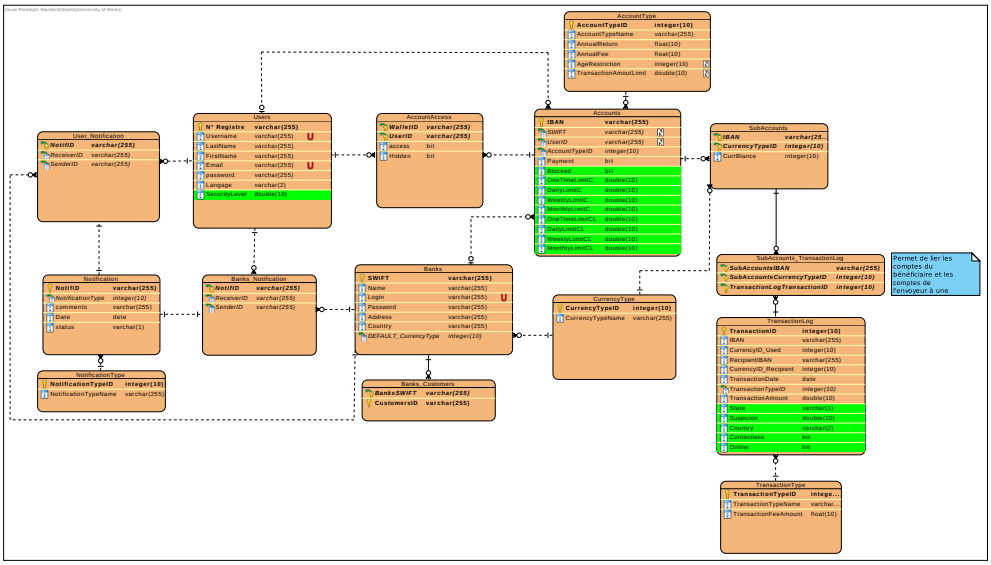
Nouveaux attributs La base de données possède certains attributs afin de contribuer au bon fonctionnement de cette extension.

Table *Users* Cette table possède un attribut *SecurityLevel* qui décrit le niveau de sécurité du compte de l'utilisateur. Typiquement, plus le niveau de sécurité sera élevé, plus l'utilisateur sera restreint ou devra confirmer son identité durant les transactions (par exemple, les limites de paiement sans contact seront plus basses, voire les paiements sans contact seront totalement bloqués).

Table *Account* Cette table possède un attribut *Blocked* qui indique si le compte est bloqué ou non, ainsi que huit attributs "limite" qui décrivent les limites instantanées, journalières, hebdomadaires et mensuelles des paiements avec et sans contact (limites distinctes).

Table *TransactionLog* Cette table possède cinq attributs : *State* qui décrit l'état d'une transaction (en attente, confirmée et traitée), *Suspicion* qui indique le niveau de suspicion de la transaction (une indication du taux de probabilité que cette transaction soit frauduleuse), *Country* qui indique le pays à partir duquel la transaction a été effectuée, *Contactless* qui indique si la transaction a été réalisée sans contact ou non et *Online* qui indique si la transaction a été réalisée dans un magasin en ligne.

Table *Notification* Cette table possède un attribut *Permanent* qui, s'il est à 1 (vrai), empêche de supprimer la notification de la liste des notifications. Grâce à cette stratégie, nous pouvons évaluer la dangerosité d'une transaction de manière locale (sans devoir évaluer la dangerosité d'une transaction chaque fois que le serveur en reçoit une), puis envoyer une notification au client à qui le compte à partir duquel le transaction a été réalisée appartenait. De cette manière, une personne usurpant le compte d'une autre personne et effectuant une transaction suspecte ne pourra pas supprimer la notification de la liste des notifications.



5.4 Maquette de l'interface utilisateur

Nouvelles fenêtres disponibles

- QR codes ;
- Generate QR code ;
- Read QR code ;
- Suspicious transactions history ;
- Due payments.

Fenêtres modifiées

- Main screen.

Fenêtre *QR codes*

Accès : en cliquant sur le bouton *QR codes* de la fenêtre *Main screen*.
Contenu : les boutons *Back*, *Generate* et *Read*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;
- Le bouton *Generate* : envoie l'utilisateur sur la fenêtre *Generate QR code* ;
- Le bouton *Read* : envoie l'utilisateur sur la fenêtre *Read QR code*.

Fenêtre *Generate QR code*

Accès : en cliquant sur le bouton *Generate* de la fenêtre QR code.
Contenu : les boutons *Back*, *Choose path...* et *Generate*, les labels *Amount*, *Message*, *Export location* et *Selected path* et les champs de texte *Amount* et *Message*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *QR codes* ;
- Le boutton *Choose path...* : ouvre l'explorateur de fichier afin de déterminer le chemin de destination ;
- Le bouton *Generate* : génère un QR code si le montant est un nombre supérieur à zéro, l'IBAN est correctement formatté et la destination est choisie ;
- Le champ de texte *Amount* : permet à l'utilisateur d'entrer le montant ;
- Le champ de texte *Message* : permet à l'utilisateur d'entrer la communication.

Fenêtre *Read QR code*

Accès : en cliquant sur le bouton *Read* de la fenêtre *QR codes*.
Contenu : les boutons *Back*, *Choose file...* et *Pay*, les labels *Choose file*, *Selected file*, *Account* et *QR code state* et le menu déroulant *Account*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *QR codes* ;
- Le bouton *Choose file...* : ouvre l'explorateur de fichier afin de déterminer le fichier à lire ;
- Le bouton *Pay* : effectue le paiement si le QR code est valide ;
- Le menu déroulant *Account* : permet à l'utilisateur de sélectionner le compte à partir duquel retirer l'argent.

Fenêtre *Suspicious transactions history*

Accès : en cliquant sur le bouton *Suspicious transactions* de la fenêtre *Main screen*.
Contenu : les boutons *Back*, *Intentional* et *Suspicious*, le label *Choose a suspicion* et la liste *Suspicious transaction history*.

- Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;

— Le bouton *Intentional* : permet à l'utilisateur de marquer une transaction comme intentionnelle et d'infirmer la suspicion ;
— Le bouton *Suspicious* : permet à l'utilisateur de marquer une transaction comme suspecte et de confirmer la suspicion ;
— La liste Suspicious transaction history : contient tout l'historique des transactions marquées comme suspectes.

Fenêtre *Due payments*

Accès : en cliquant sur le bouton *Due payments* de la fenêtre *Main screen*.
Contenu : les boutons *Back*, *Pay* et *Pay contactless*, les labels *Choose a payment* et *Choose an account*, le menu déroulant *Account* et la liste *Due payments*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Main screen* ;
— Le bouton *Pay* : permet à l'utilisateur de payer "avec contact" avec le compte sélectionné, c'est-à-dire entrer son compte PIN afin de valider la transaction ;
— Le bouton *Pay contactless* : permet à l'utilisateur de payer "sans contact" avec le compte sélectionné, c'est-à-dire valider la transaction sans entrer son code PIN, à condition que la limite ne soit pas dépassée ;
— Le menu déroulant *Account* : permet à l'utilisateur de sélectionner le compte à partir duquel effectuer le paiement ;
— La liste *Due payments* : affiche tous les paiements trouvés dans le fichier dédié et permet à l'utilisateur d'en sélecionner un à effectuer.

Fenêtre *Main screen*

Nouvel accès : en cliquant sur le bouton *Back* des fenêtre *QR codes*, *Suspicious transactions history* ou *Due payments*.
Nouveau contenu : les boutons *Due payments*, *Suspicious transactions* et *QR codes*.

— Le bouton *Due payments* : envoie l'utilisateur sur la fenêtre *Due payments* ;
— Le bouton *Suspicious transactions* : envoie l'utilisateur sur la fenêtre *Suspicious transactions history* ;
— Le bouton *QR codes* : envoie l'utilisateur sur la fenêtre *QR codes*.

Illustration du diagramme Par souci de clareté, ce diagramme est disponible en annexe du rapport.

5.5 Maquette de l'interface institution

Nouvelles fenêtres disponibles

— Limits.

Fenêtres modifiées

— Details.

Fenêtre *Limits*

Accès : en cliquant sur le bouton *Edit limits* de la fenêtre *Details*.
Contenu : les boutons *Back* et *Set limit*, les labels *Client name*, *Account*, *Time scope*, *Current limit* et *New limit*, le champ de texte *New limit...* et le menu déroulant *Time scope*.

— Le bouton *Back* : envoie l'utilisateur sur la fenêtre *Details* ;
— Le bouton *Set limit* : modifie la limite maximale d'argent qu'il est possible de transférer à l'échelle de temps choisie dans le menu déroulant ;
— Le menu déroulant *Time scope* : permet de choisir l'échelle de temps pour laquelle il faut modifier la limite (at once, daily, weekly ou monthly).

Fenêtre *Details*

Nouvel accès : en cliquant sur le bouton *Back* de la fenêtre *Limits*.
Nouveau contenu : le bouton *Edit limits*.

— Le bouton *Edit limits* : envoie l'utilisateur sur la fenêtre *Limits*.

Illustration du diagramme Par souci de clareté, ce diagramme est disponible en annexe du rapport.