



Date de mise à jour : 05/05/2021  
Version : 1.2

## Table des matières

1.	Objectif du document .....	3
1.1.	Eléments modifiés .....	3
1.2.	Eléments non traités .....	3
2.	Cas d'utilisation.....	4
2.1.	Sous-système « opérations de base » .....	4
2.1.1.	Demander une ouverture de compte.....	4
2.1.2.	Se connecter .....	6
2.2.	Sous-système « gérer les clients ».....	9
2.2.1.	Créer un client .....	9
2.3.	Sous-système « agréger un compte bancaire ».....	11
2.3.1.	Ajouter un compte bancaire externe.....	12
2.4.	Sous-système « gérer le compte bancaire » - Cas de la gestion d'un compte bancaire interne .....	15
2.4.1.	Consulter le solde du compte .....	15
2.4.2.	Effectuer un virement interne.....	17
2.4.3.	Effectuer un virement externe .....	19
2.4.4.	Mettre en place un virement automatique interne .....	22
3.	Regroupement des classes .....	25
3.1.	Groupe domaine .....	25
3.2.	Groupe domaine et cycle de vie .....	26
3.3.	Groupe Service .....	27
3.4.	Groupe interface utilisateur et système .....	27
4.	Annexes .....	28
4.1.	Terminologie .....	28
5.	Index des diagrammes .....	29

## 1. Objectif du document

Ce document constitue l'analyse du projet Mocha Bank, proposé pour l'unité d'enseignement GLG204 - Architecture logicielle du CNAM du second semestre 2019-2020.

Il a été élaboré à partir du document d'expression des besoins de ce projet et utilise le processus unifié simplifié décrit dans « Enterprise Java with UML » de C. T. Arrington.

Son objectif est de fournir un bref aperçu de la fonction du système et les raisons de son développement, sa portée et des références au contexte de développement (par exemple, référence à l'énoncé du problème écrit par le client, références aux systèmes existants, études de faisabilité). La granularité des informations du système pourra être affinée par la suite.

### 1.1. *Eléments modifiés*

Le cas d'utilisation « effectuer un virement » a été divisé en deux nouveaux cas d'utilisation pour plus lisibilité. En effet, effectuer un virement peut se faire vers un compte interne à Mocha Bank ou vers un compte externe à celle-ci.

Le cas d'utilisation « mettre en place un virement automatique » a également été divisé en deux nouveaux cas d'utilisation et suit le même principe que le cas d'utilisation « effectuer un virement ».

### 1.2. *Eléments non traités*

Le choix a été porté sur la pertinence des cas d'utilisation, dans l'idée d'essayer de représenter l'ensemble des objets du système. Ainsi, les éléments suivant n'ont pas été analysés dans ce document et pourront faire l'objet d'un traitement ultérieur.

- Sous-système « opérations de base » :
  - Changer de mot de passe
  - Se déconnecter
- Sous-système « gérer les clients » :
  - Rechercher un client
  - Mettre à jour les informations d'un client
  - Supprimer un client
  - Créer un compte bancaire
  - Rechercher un compte bancaire
  - Supprimer un compte
- Sous-système « agréger un compte bancaire » :
  - Supprimer un compte bancaire externe
- Sous-système « gérer le compte bancaire » - Cas de la gestion d'un compte bancaire interne :
  - Consulter l'historique des opérations du compte
  - Faire un dépôt
  - Mettre en place un virement automatique externe

- Réaliser automatiquement le virement
- Sous-système « gérer le compte bancaire » - Cas de la gestion d'un compte bancaire externe agrégé :
  - Consulter le solde d'un compte bancaire agrégé
  - Consulter l'historique des opérations d'un compte bancaire agrégé
  - Faire un dépôt sur un compte bancaire agrégé
  - Effectuer un virement depuis un compte bancaire agrégé
  - Mettre en place un virement automatique externe depuis un compte bancaire agrégé

## 2. Cas d'utilisation

### 2.1. Sous-système « opérations de base »

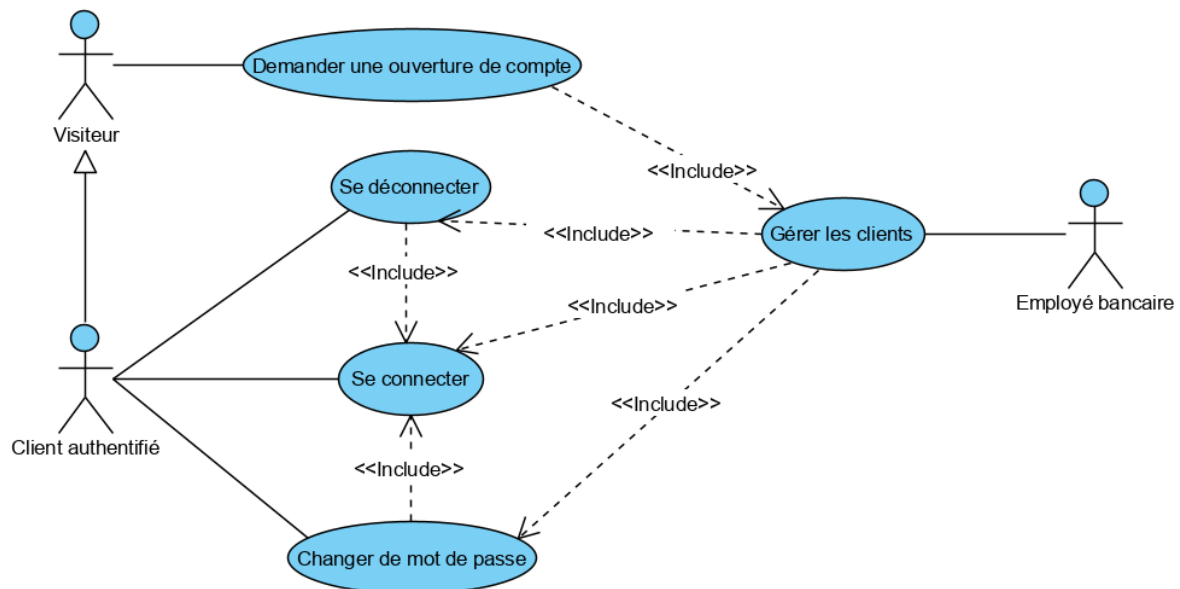


Fig. 1 : diagramme de cas d'utilisation sous-système « opérations de base »

#### 2.1.1. Demander une ouverture de compte

##### 2.1.1.1. Liste des objets candidats

- <<Entity>>

User : Objet représentant un utilisateur.

ProspectCustomer : Objet représentant un visiteur ayant demandé une ouverture de compte bancaire.

Address : Objet représentant une adresse postale.

OpenAccountRequest : Objet représentant une demande d'ouverture de compte bancaire.

- <<Boundary>>

OpenAccountRequestFormUI : Objet représentant l'interface utilisateur par laquelle le visiteur va interagir pour demander une ouverture de compte bancaire chez Mocha Bank.

- <<Control>>

OpenAccountRequestService : Objet chargé de la logique métier de demande d'ouverture de compte bancaire. Il gère l'affichage de l'objet OpenAccountRequestFormUI ainsi que la validation de la demande d'ouverture de compte bancaire.

- <<Life cycle>>

UserDAO : Objet chargé d'écrire ou de récupérer un utilisateur depuis le support de stockage.

ProspectCustomerDAO : Objet chargé d'écrire ou de récupérer un visiteur ayant demandé une ouverture de compte bancaire depuis le support de stockage.

OpenAccountRequestDAO : Objet chargé d'écrire ou de récupérer une demande d'ouverture de compte bancaire depuis le support de stockage.

### 2.1.1.2. Description des interactions entre objets

Le contenu du formulaire de demande d'ouverture de compte bancaire pourra éventuellement être modifié par la suite. Pour l'instant, il contient les informations client ainsi que le nom d'une offre de compte bancaire, en imaginant qu'il y a plusieurs offres disponibles. Ces différentes offres de comptes bancaires déterminent entre autres, le plafond de virement, les modalités de l'autorisation de découvert et les frais de tenue de compte.

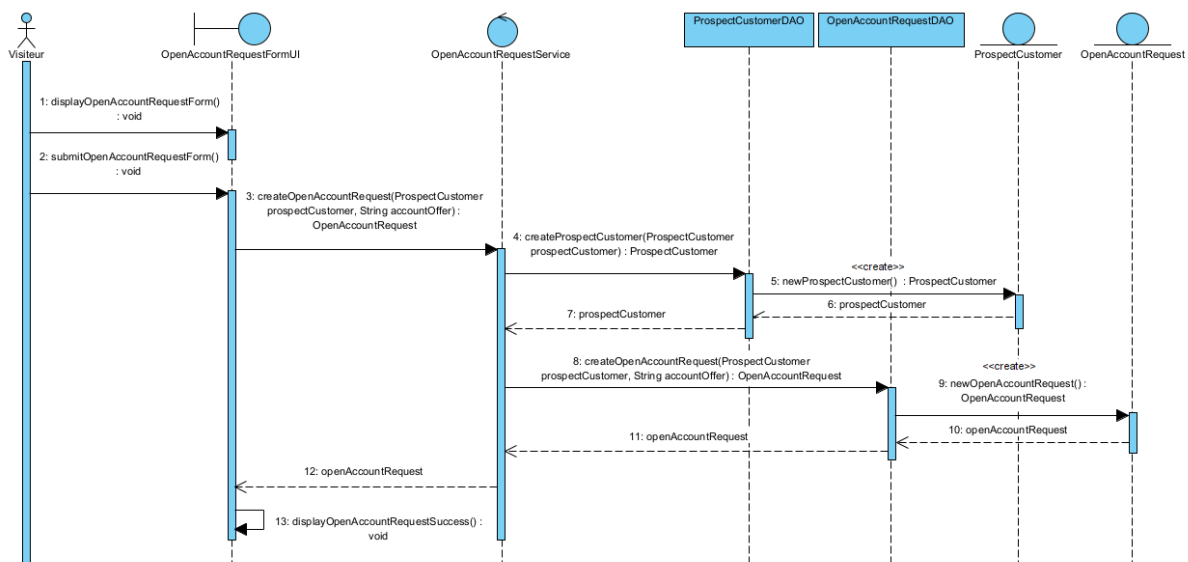


Fig. 2 : diagramme de séquences « demander une ouverture de compte »

### 2.1.1.3. Description des classes

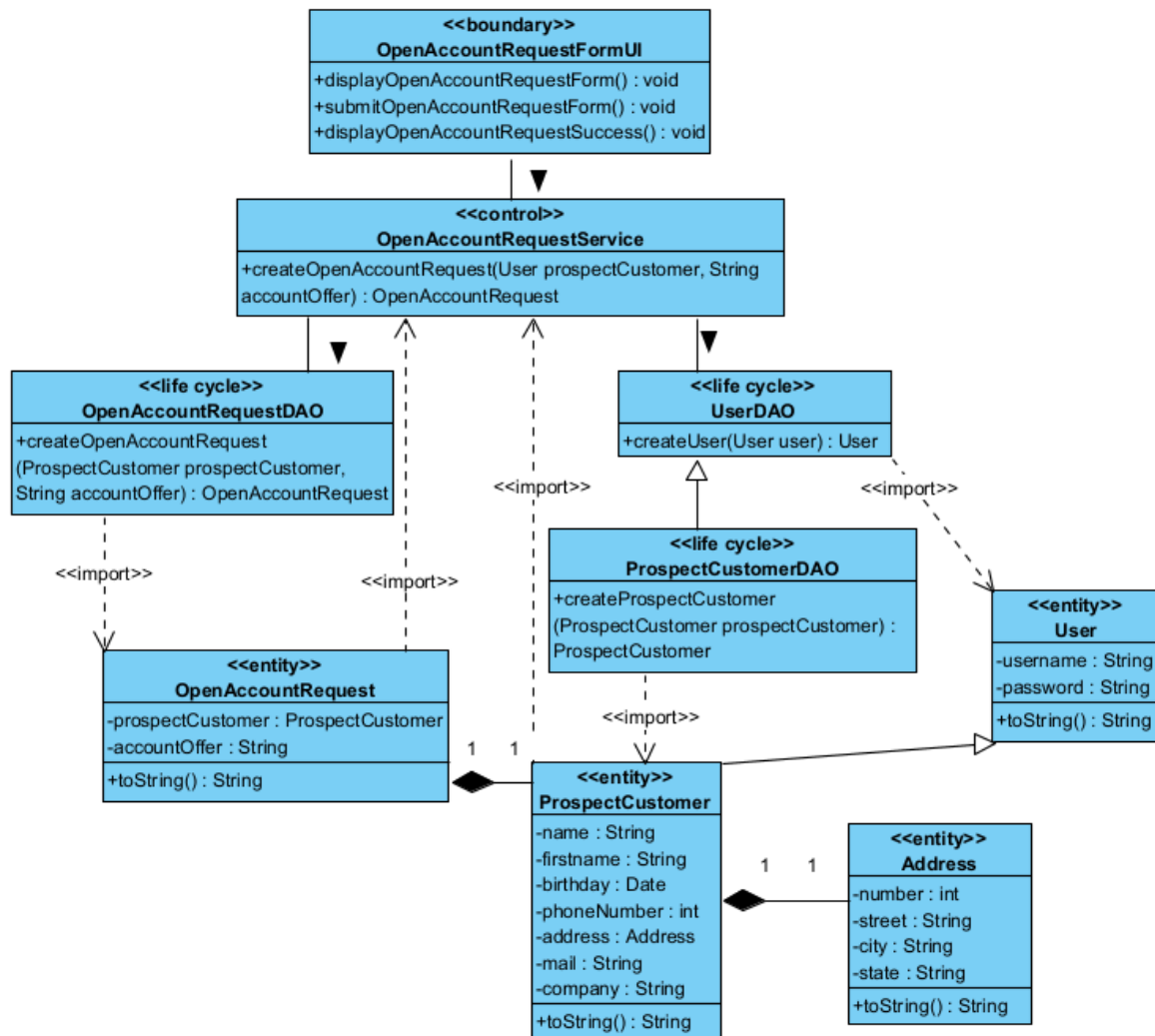


Fig. 3 : diagramme de classes « demander une ouverture de compte »

## 2.1.2. Se connecter

### 2.1.2.1. Liste des objets candidats

- <<Entity>>

User : Objet représentant un utilisateur.

Customer : Objet représentant un client.

BankClerk : Objet représentant un employé bancaire.

Address : Objet représentant une adresse postale.

- <<Boundary>>

LoginFormUI : Objet représentant l'interface utilisateur par laquelle l'utilisateur (le client ou l'employé bancaire) va interagir pour s'authentifier.

- <<Control>>

LoginService : Objet chargé de la logique métier d'authentification. Il gère l'affichage de l'objet LoginFormUI ainsi que la validation de l'authentification.

- <<Life cycle>>

UserDAO : Objet chargé d'écrire ou de récupérer un utilisateur depuis le support de stockage.

CustomerDAO : Objet chargé d'écrire ou de récupérer un client depuis le support de stockage.

BankClerkDAO : Objet chargé d'écrire ou de récupérer un employé bancaire depuis le support de stockage.

### 2.1.2.2. Description des interactions entre objets

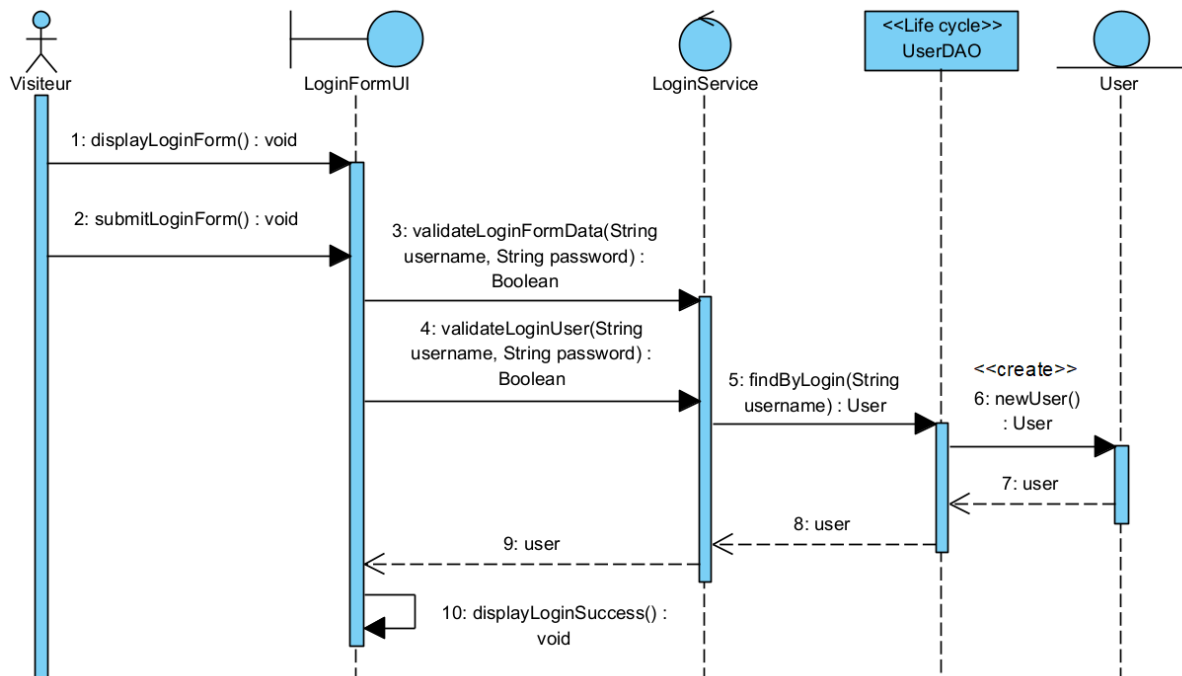


Fig. 4 : diagramme de séquences « se connecter »

## 2.1.2.3. Description des classes

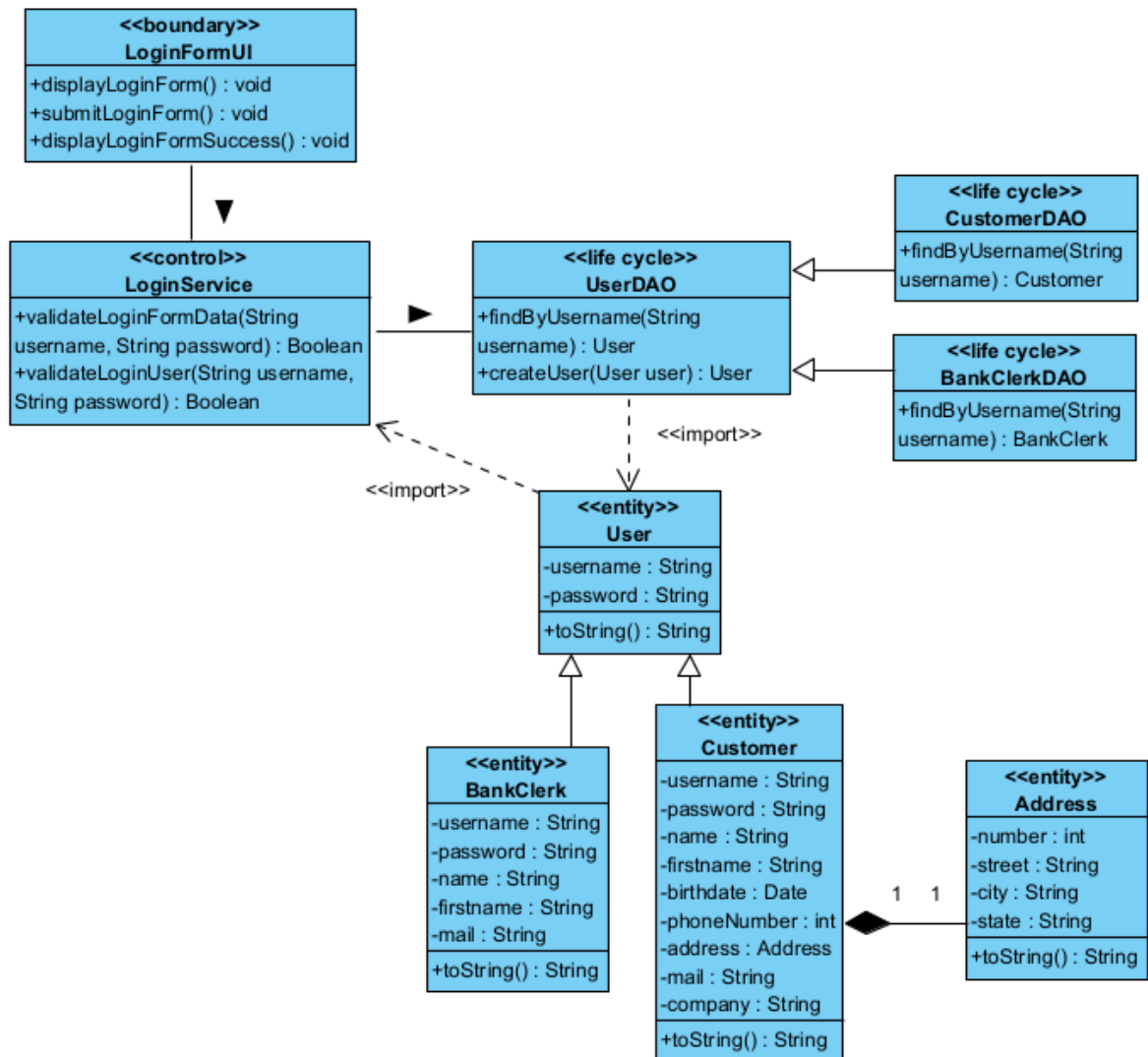


Fig. 5 : diagramme de classes « se connecter »



## 2.2. Sous-système « gérer les clients »

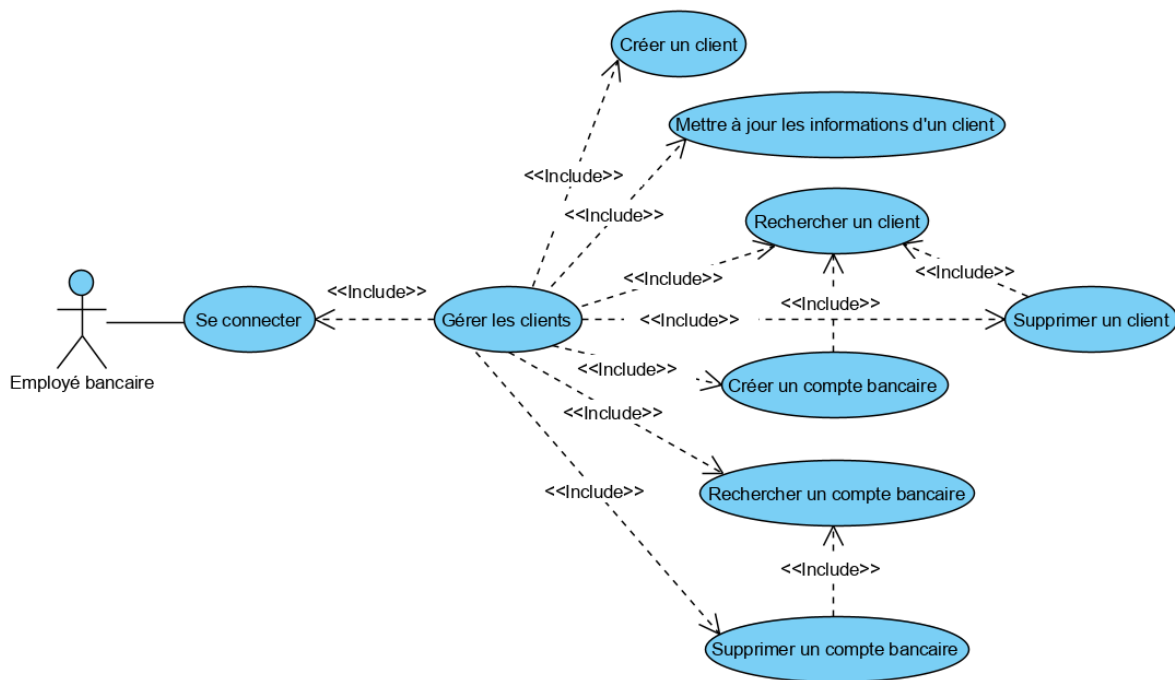


Fig. 6 : diagramme de cas d'utilisation sous-système « gérer les clients »

### 2.2.1. Créer un client

#### 2.2.1.1. Liste des objets candidats

- <<Entity>>

User : Objet représentant un utilisateur.

Customer : Objet représentant un client.

Address : Objet représentant une adresse postale.

BankClerk : Objet représentant un employé bancaire.

- <<Boundary>>

CustomerManagementUI : Objet représentant l'interface utilisateur par laquelle l'employé bancaire va interagir pour gérer les clients, ici, créer un client.

- <<Control>>

CustomerManagementService : Objet chargé de la logique métier de gestion de client. Il gère l'affichage de l'objet CustomerManagementUI ainsi que les opérations de gestion de client (créer un client, rechercher un client, mettre à jour les informations d'un client, supprimer un client).

- <<Life cycle>>

UserDAO : Objet chargé d'écrire ou de récupérer un utilisateur depuis le support de stockage.

CustomerDAO : Objet chargé d'écrire ou de récupérer un client depuis le support de stockage.

BankClerkDAO : Objet chargé d'écrire ou de récupérer un employé bancaire depuis le support de stockage.

### 2.2.1.2. Description des interactions entre objets

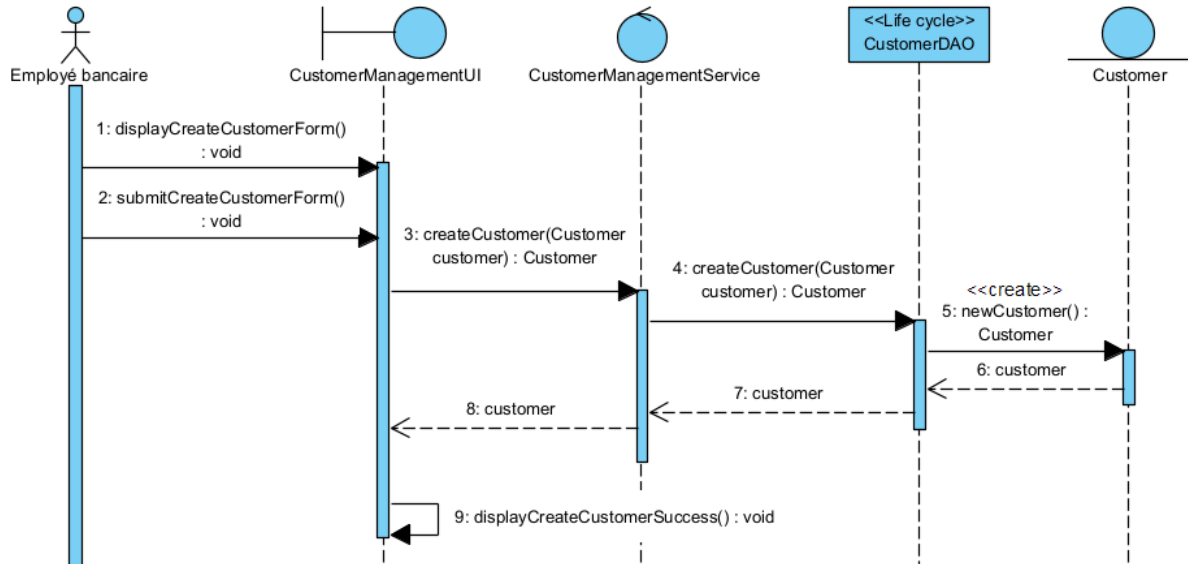


Fig. 7 : diagramme de séquences « créer un client »

### 2.2.1.1. Description des classes

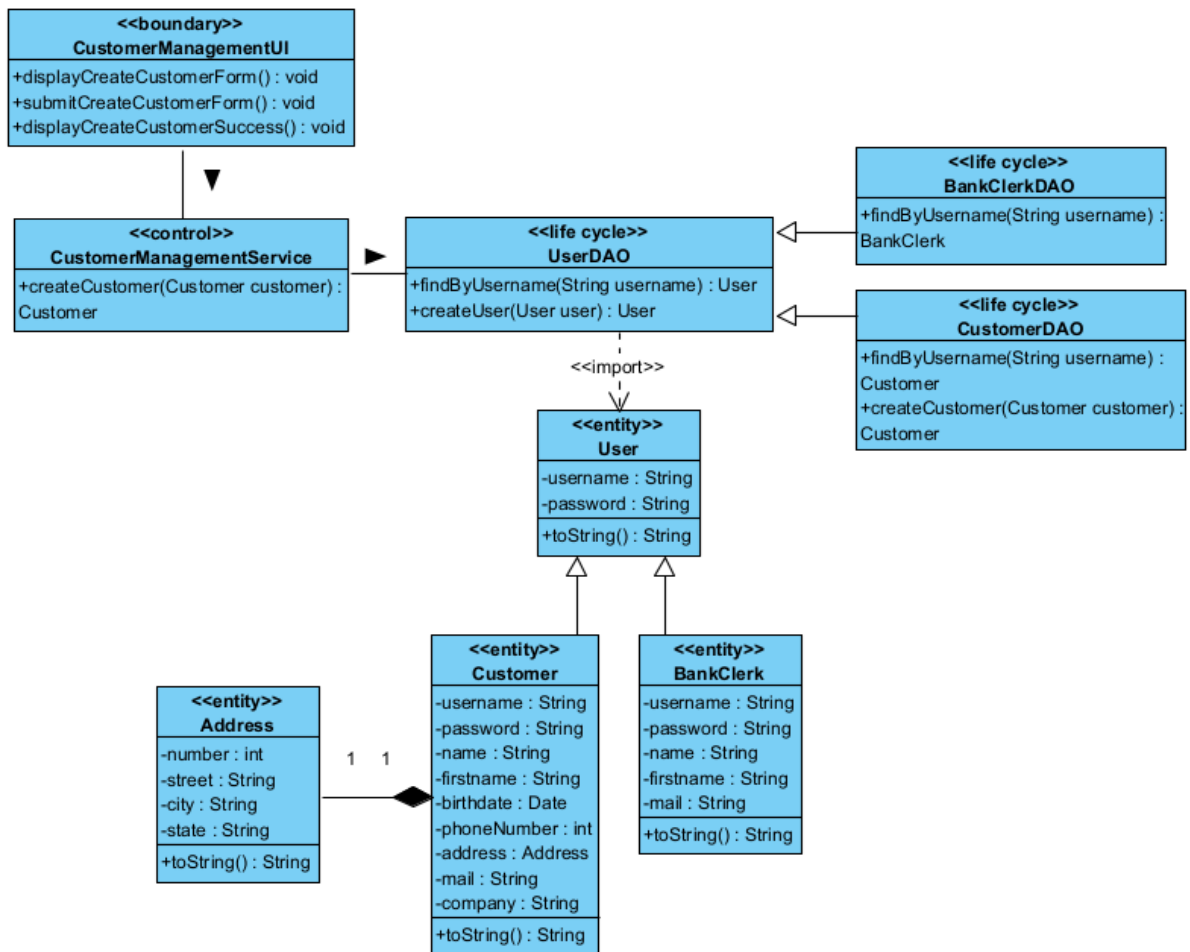


Fig. 8 : diagramme de classes « créer un client »

### 2.3. Sous-système « agréger un compte bancaire »

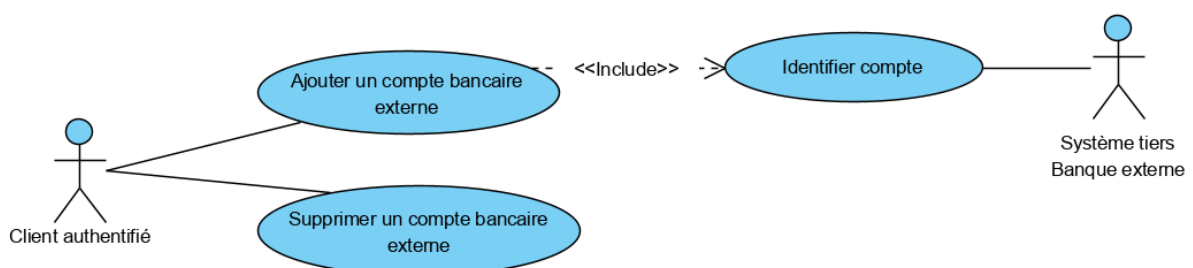


Fig. 9 : diagramme de cas d'utilisation sous-système « agréger un compte bancaire »

### **2.3.1. Ajouter un compte bancaire externe**

#### **2.3.1.1. Liste des objets candidats**

- <<Entity>>

Customer : Objet représentant un client.

Account : Objet représentant un compte bancaire.

- <<Boundary>>

AggregatedAccountAdditionUI : Objet représentant l'interface utilisateur par laquelle le client va interagir pour gérer l'agrégation de ses comptes bancaires externes à son espace client chez Mocha Bank, ici, ajouter un compte bancaire externe.

ExternalBank : Objet représentant une banque externe. Il interagit avec le système de Mocha Bank pour agréger un compte bancaire y étant domicilié.

- <<Control>>

AggregatedAccountAdditionService : Objet chargé de la logique métier de gestion d'agrégation de comptes bancaires. Il gère l'affichage de l'objet AggregatedAccountAdditionUI ainsi que les opérations de gestion d'agrégation de comptes bancaires (ajouter un compte bancaire externe, supprimer un compte bancaire externe).

HTTPSender : Objet chargé de la communication avec les banques externes via le protocole HTTP.

- <<Life cycle>>

AccountDAO : Objet chargé d'écrire ou de récupérer un compte bancaire depuis le support de stockage.

CustomerDAO : Objet chargé d'écrire ou de récupérer un client depuis le support de stockage.

#### **2.3.1.2. Description des interactions entre objets**

L'identifiant de la banque (bankId) et l'identifiant du compte (accountId) sont utilisés à la place du Numéro International de Compte Bancaire (IBAN) dans un souci de simplification. En effet, l'IBAN est constitué du code du pays (standard ISO 3166), de la clé de contrôle puis du Relevé d'Identité Bancaire (RIB(BBAN)) lui-même composé du code banque, du code guichet, du numéro de compte et de la clé du RIB.

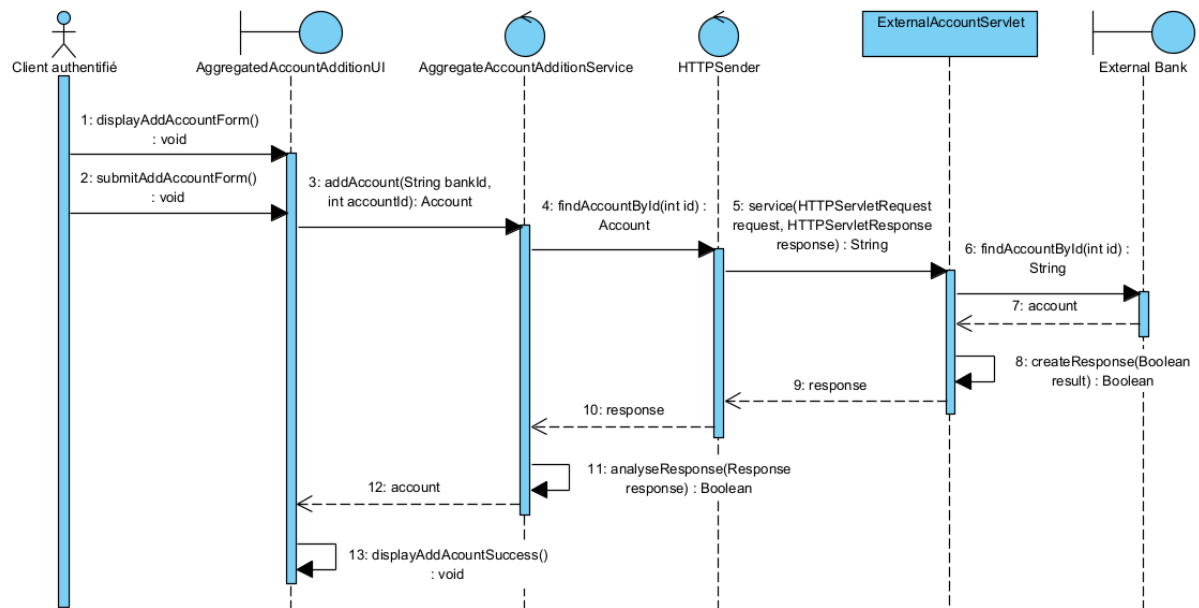


Fig. 10 : diagramme de séquences « ajouter un compte bancaire externe »

### 2.3.1.1. Description des classes

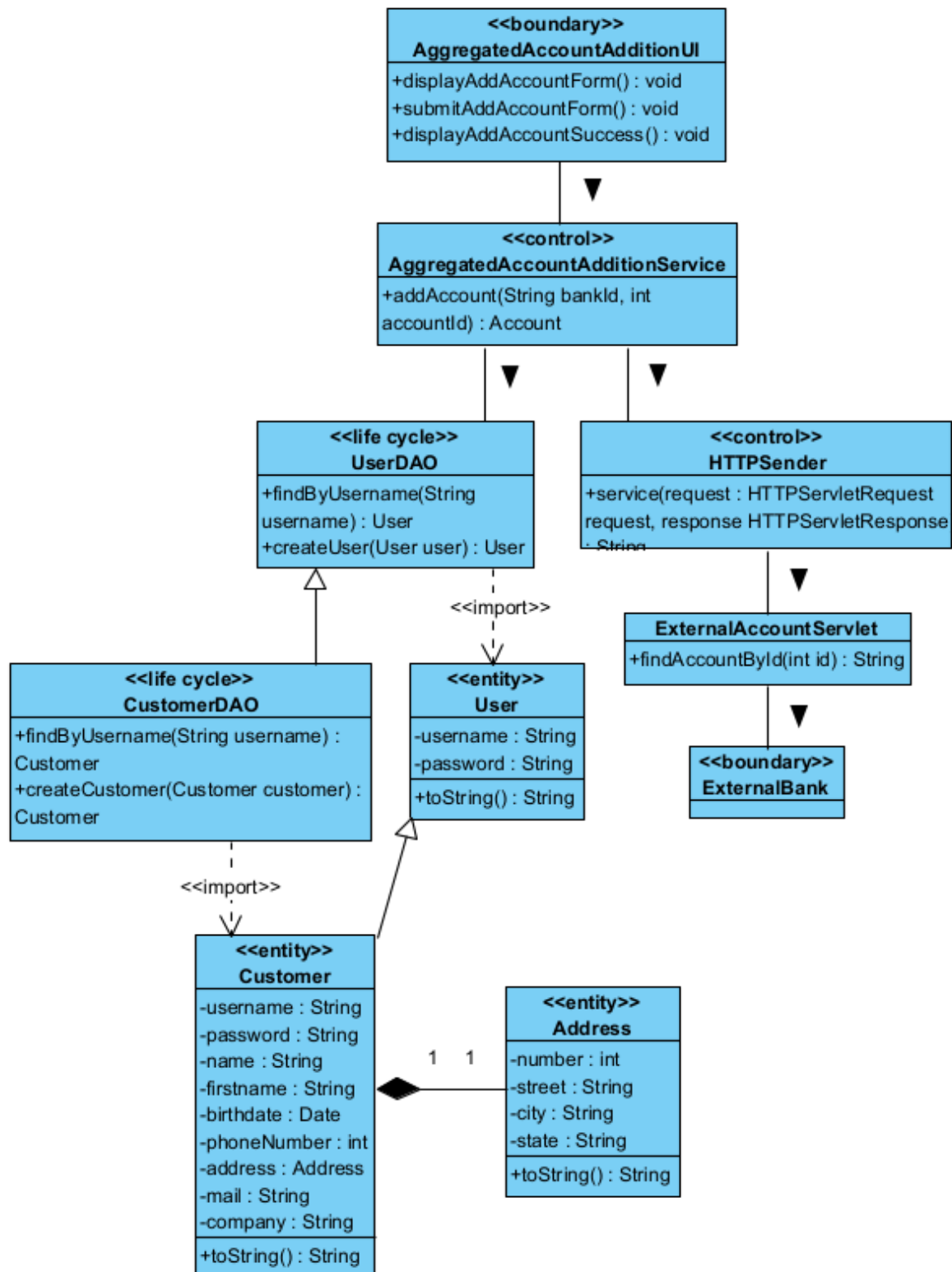


Fig. 11 : diagramme de classes «ajouter un compte bancaire externe »

## 2.4. Sous-système « gérer le compte bancaire » - Cas de la gestion d'un compte bancaire interne

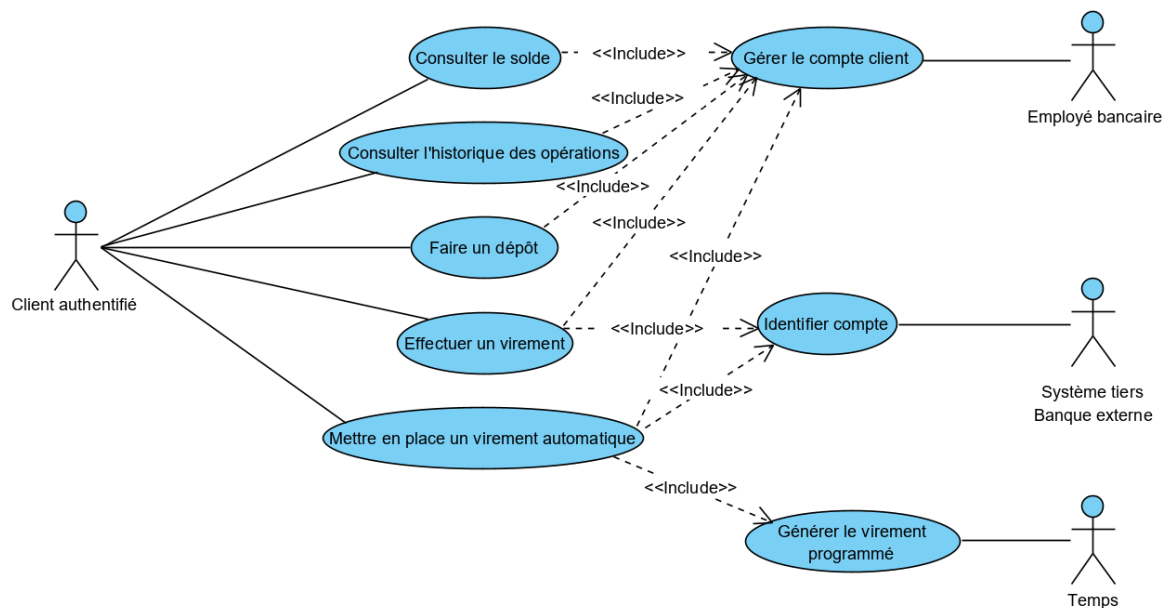


Fig. 12 : diagramme de cas d'utilisation sous-système « gérer le compte bancaire »

### 2.4.1. Consulter le solde du compte

#### 2.4.1.1. Liste des objets candidats

- **<<Entity>>**

Customer : Objet représentant un client.

Address : Objet représentant une adresse postale.

Account : Objet représentant un compte bancaire.

Transaction : Objet représentant une opération bancaire.

Operation : Objet représentant une opération bancaire de base.

ScheduledOperation : Objet représentant une opération bancaire programmée.

- **<<Boundary>>**

AccountManagementUI : Objet représentant l'interface utilisateur par laquelle le client va interagir pour gérer son compte bancaire, ici, consulter le solde du compte.

- **<<Control>>**

AccountManagementService : Objet chargé de la logique métier de gestion de compte bancaire. Il gère l'affichage de l'objet AccountManagementUI ainsi que les opérations de gestion de compte (consulter le solde, consulter l'historique des opérations, faire un dépôt, effectuer un virement interne/externe, mettre en place un virement automatique interne/externe).

- **<<Life cycle>>**

AccountDAO : Objet chargé d'écrire ou de récupérer un compte bancaire depuis le support de stockage.

CustomerDAO : Objet chargé d'écrire ou de récupérer un client depuis le support de stockage.

TransactionDAO : Objet chargé d'écrire ou de récupérer une opération bancaire depuis le support de stockage.

#### 2.4.1.2. Description des interactions entre objets

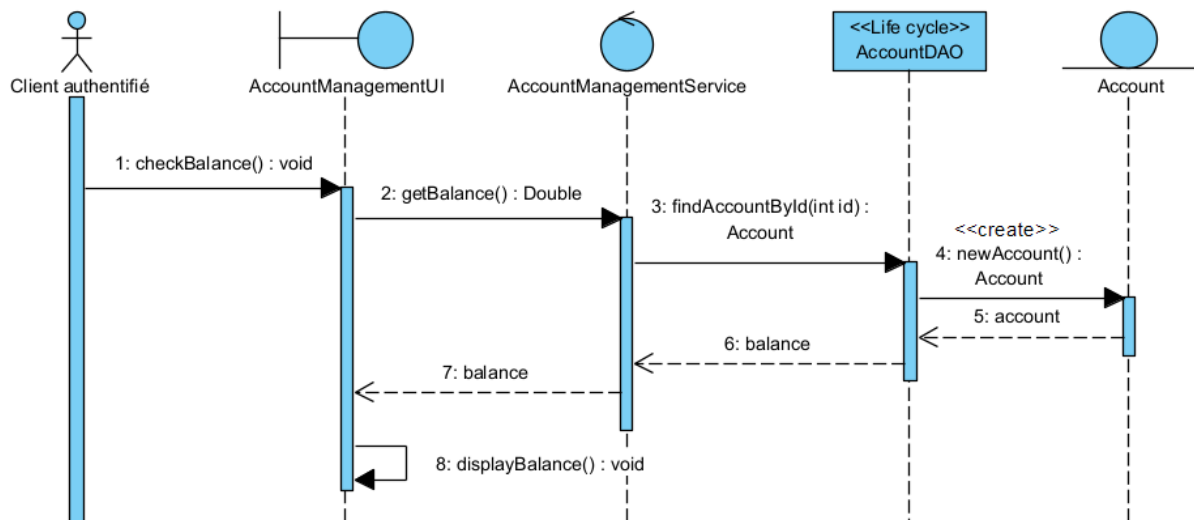


Fig. 13 : diagramme de séquences « consulter le solde du compte bancaire »

#### 2.4.1.3. Description des classes



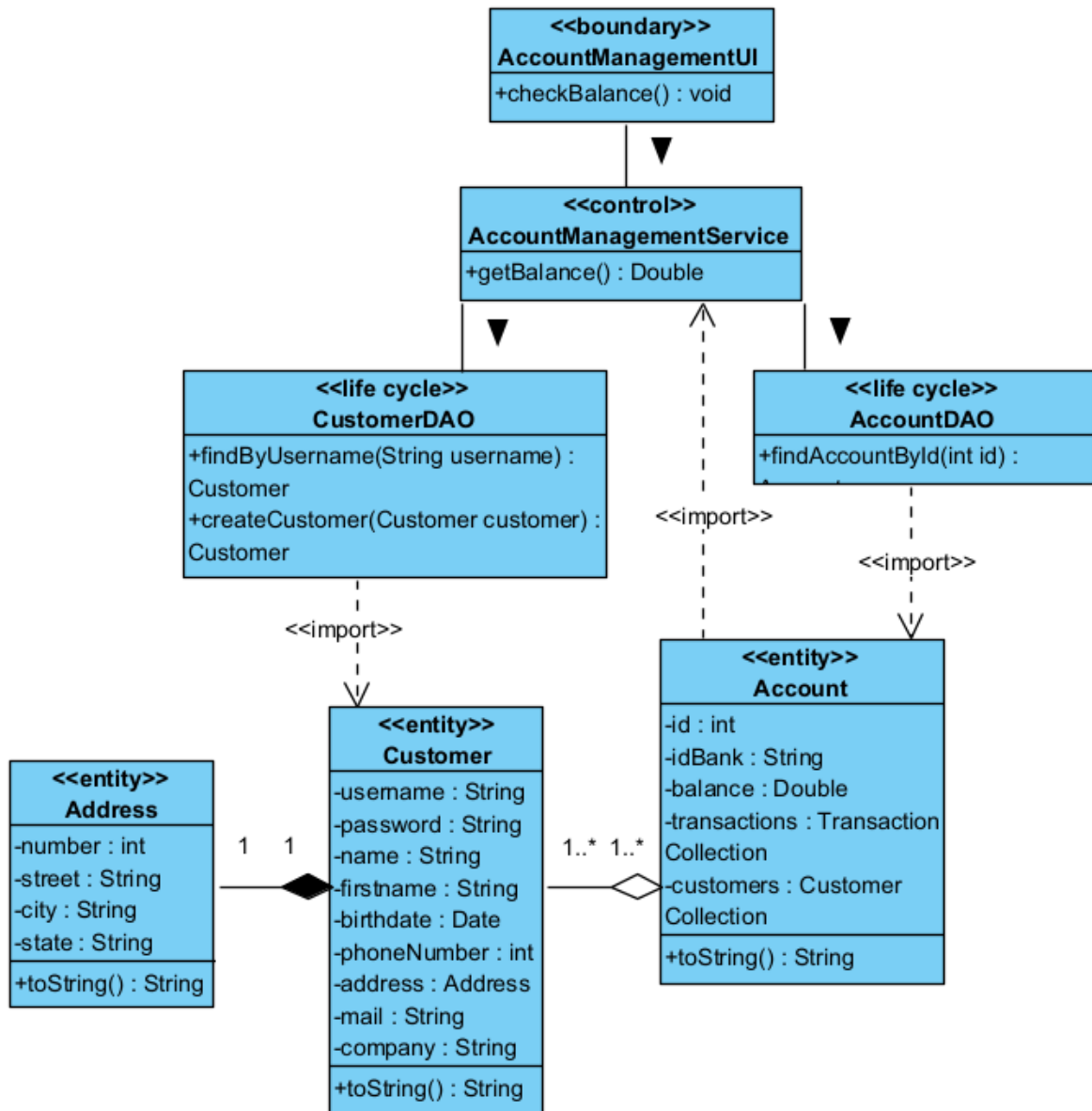


Fig. 14 : diagramme de classes « consulter le solde du compte bancaire »

## 2.4.2. Effectuer un virement interne

### 2.4.2.1. Liste des objets candidats

- `<<Entity>>`

Customer : Objet représentant un client.

Address : Objet représentant une adresse postale.

Account : Objet représentant un compte bancaire.

Transaction : Objet représentant une opération bancaire.

Operation : Objet représentant une opération bancaire de base.

- `<<Boundary>>`

AccountManagementUI : Objet représentant l'interface utilisateur par laquelle le client va interagir pour gérer son compte bancaire, ici, effectuer un virement interne.

- <<Control>>

AccountManagementService : Objet chargé de la logique métier de gestion de compte bancaire. Il gère l'affichage de l'objet AccountManagementUI ainsi que les opérations de gestion de compte (consulter le solde, consulter l'historique des opérations, faire un dépôt, effectuer un virement interne/externe, mettre en place un virement automatique interne/externe).

- <<Life cycle>>

AccountDAO : Objet chargé d'écrire ou de récupérer un compte bancaire depuis le support de stockage.

CustomerDAO : Objet chargé d'écrire ou de récupérer un client depuis le support de stockage.

TransactionDAO : Objet chargé d'écrire ou de récupérer une opération bancaire depuis le support de stockage.

### 2.4.2.2. Description des interactions entre objets

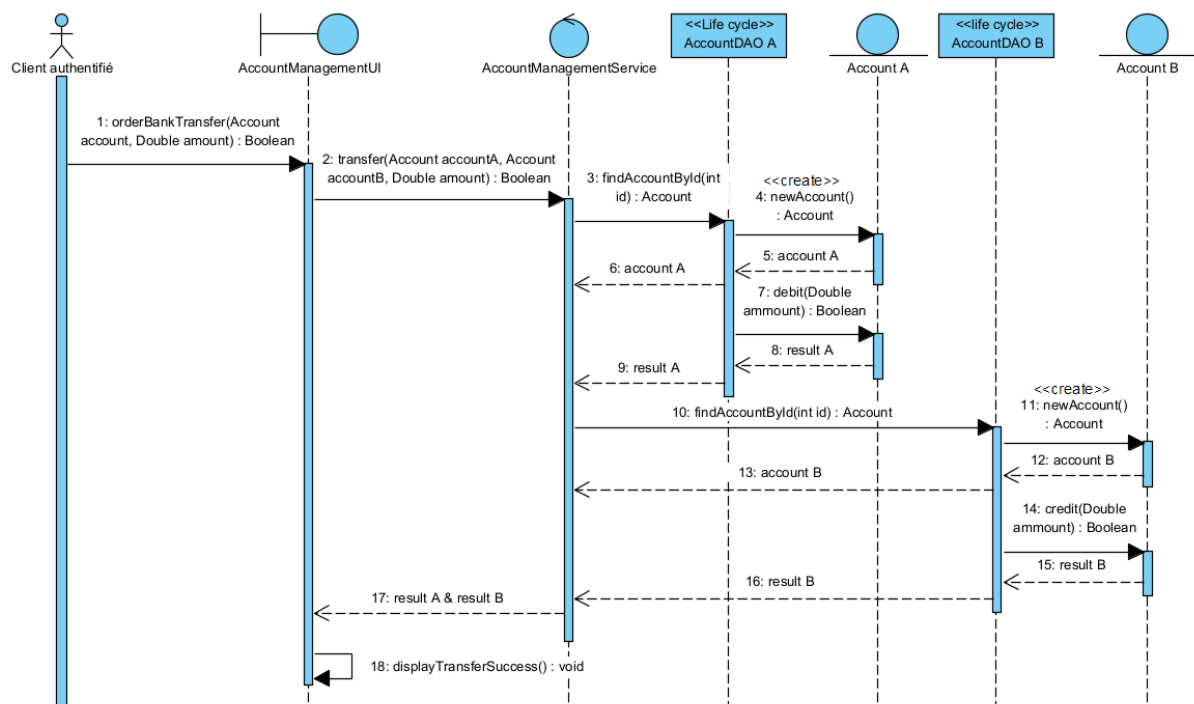


Fig. 15 : diagramme de séquences « effectuer un virement interne »

### 2.4.2.3. Description des classes

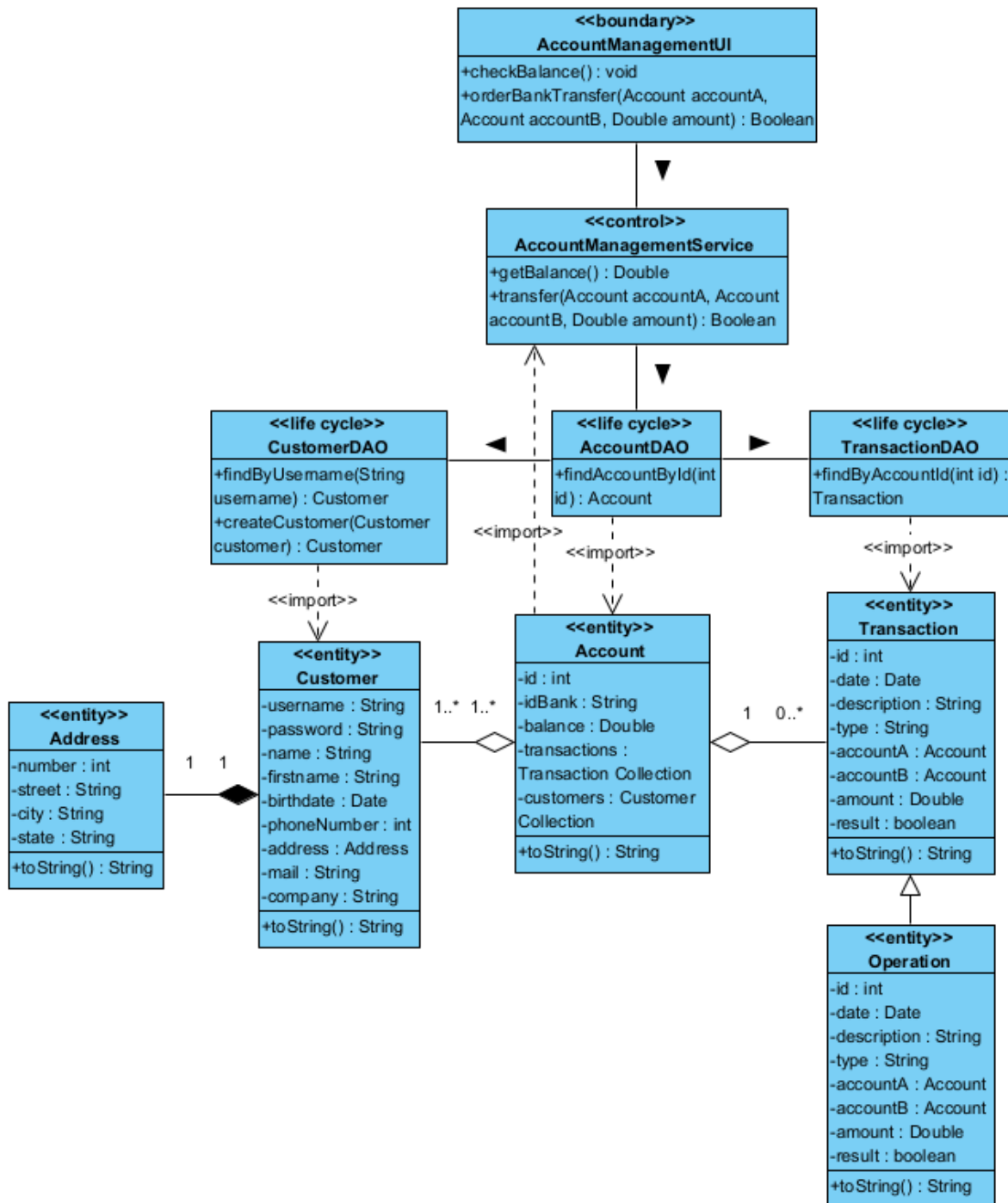


Fig. 16 : diagramme de classes « effectuer un virement interne »

### 2.4.3. Effectuer un virement externe

#### 2.4.3.1. Liste des objets candidats

- <<Entity>>

Customer : Objet représentant un client.

Address : Objet représentant une adresse postale.

Account : Objet représentant un compte bancaire chez Mocha Bank.

Transaction : Objet représentant une opération bancaire.

Operation : Objet représentant une opération bancaire de base.

- <<Boundary>>

AccountManagementUI : Objet représentant l'interface utilisateur par laquelle le client va interagir pour gérer son compte bancaire, ici, effectuer un virement externe.

ExternalBank : Objet représentant une banque externe. Il interagit avec le système de Mocha Bank pour réceptionner un virement externe (compte destinataire du virement).

- <<Control>>

AccountManagementService : Objet chargé de la logique métier de gestion de compte bancaire. Il gère l'affichage de l'objet AccountManagementUI ainsi que les opérations de gestion de compte (consulter le solde, consulter l'historique des opérations, faire un dépôt, effectuer un virement interne/externe, mettre en place un virement automatique interne/externe).

HTTPSender : Objet chargé de la communication avec les banques externes via le protocole HTTP.

- <<Life cycle>>

AccountDAO : Objet chargé d'écrire ou de récupérer un compte bancaire depuis le support de stockage chez Mocha Bank.

CustomerDAO : Objet chargé d'écrire ou de récupérer un client depuis le support de stockage.

TransactionDAO : Objet chargé d'écrire ou de récupérer une opération bancaire depuis le support de stockage.

#### **2.4.3.2. Description des interactions entre objets**

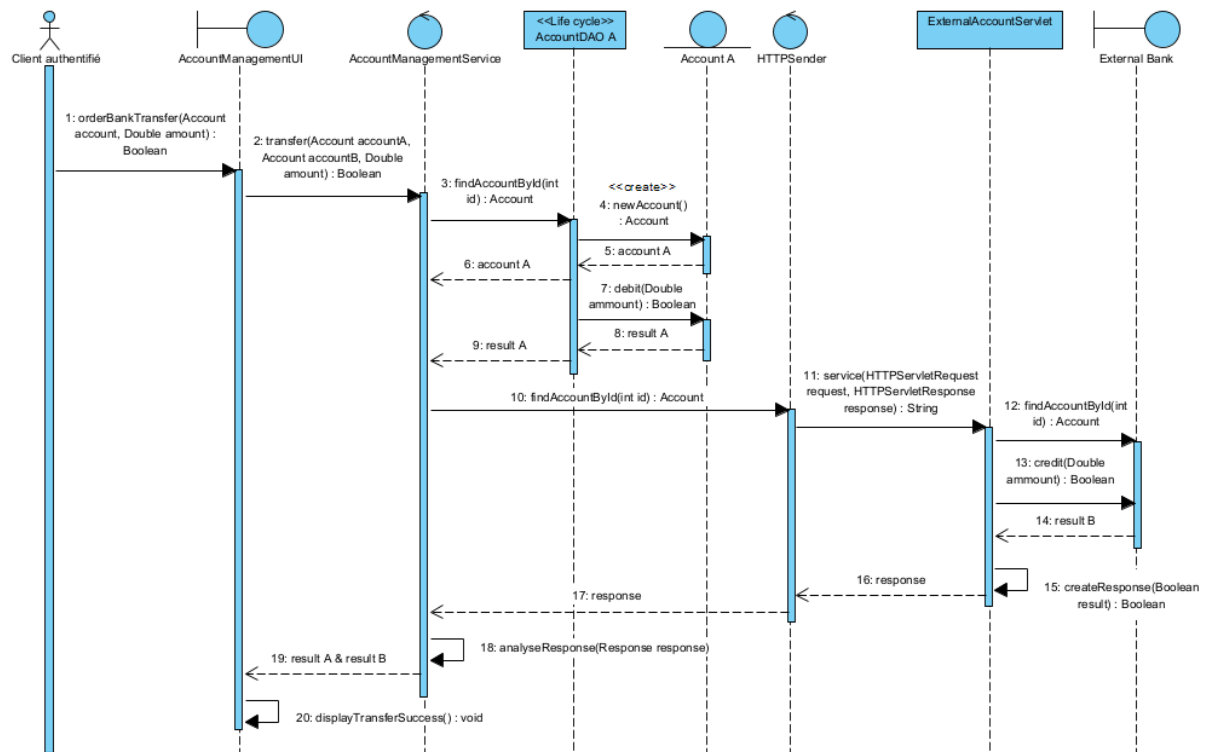


Fig.17 : diagramme de séquences « effectuer un virement externe »

### 2.4.3.3. Description des classes

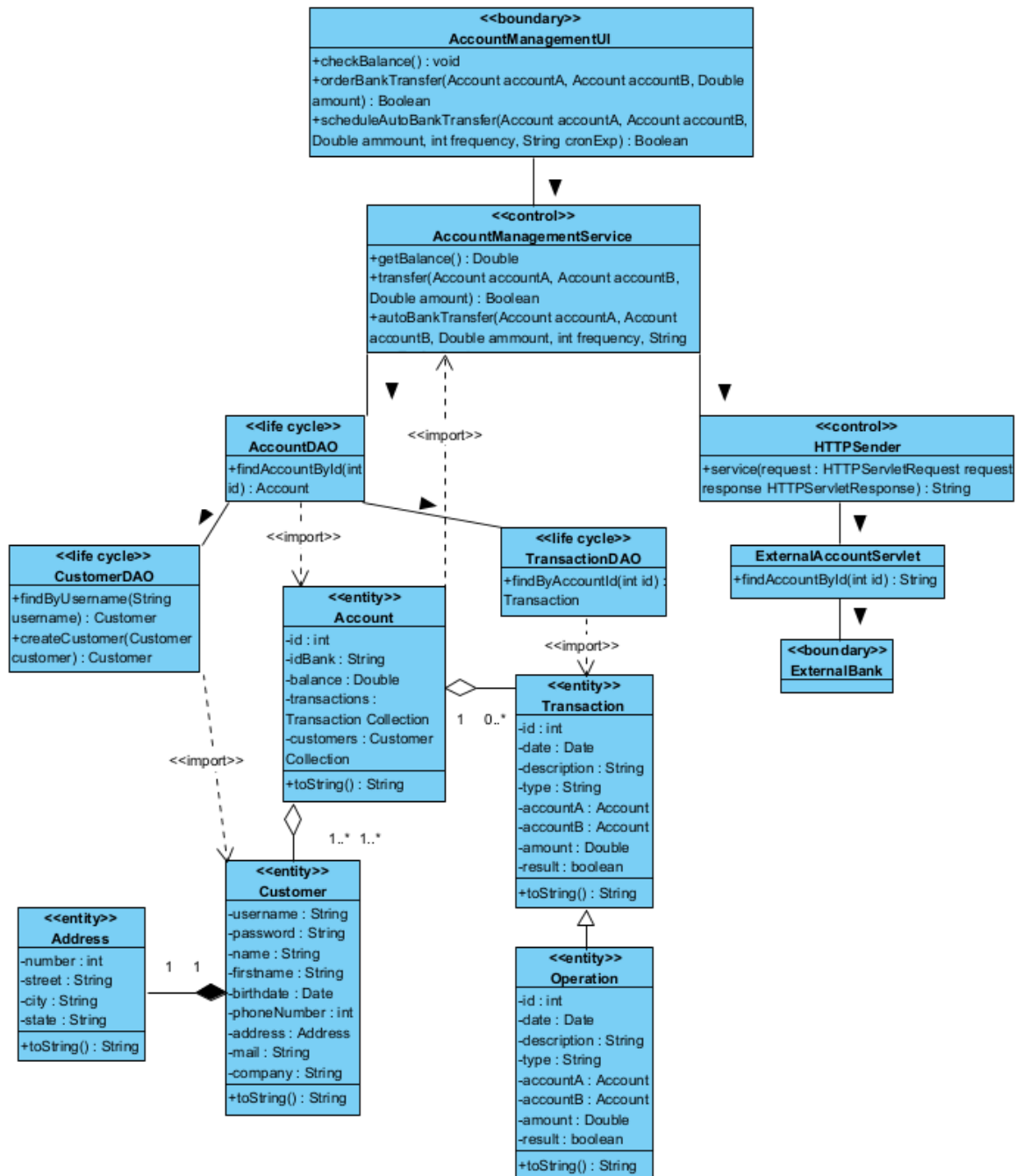


Fig.18 : diagramme de classes « effectuer un virement externe »

### 2.4.4. Mettre en place un virement automatique interne

#### 2.4.4.1. Liste des objets candidats

- <<Entity>>

Customer : Objet représentant un client.

Address : Objet représentant une adresse postale.

Account : Objet représentant un compte bancaire chez Mocha Bank.

Transaction : Objet représentant une opération bancaire.

Operation : Objet représentant une opération bancaire de base.

ScheduledOperation : Objet représentant une opération bancaire programmée.

- <<Boundary>>

AccountManagementUI : Objet représentant l'interface utilisateur par laquelle le client va interagir pour gérer son compte bancaire, ici, mettre en place un virement automatique interne.

- <<Control>>

AccountManagementService : Objet chargé de la logique métier de gestion de compte bancaire. Il gère l'affichage de l'objet AccountManagementUI ainsi que les opérations de gestion de compte (consulter le solde, consulter l'historique des opérations, faire un dépôt, effectuer un virement interne/externe, mettre en place un virement automatique interne/externe).

- <<Life cycle>>

AccountDAO : Objet chargé d'écrire ou de récupérer un compte bancaire depuis le support de stockage.

CustomerDAO : Objet chargé d'écrire ou de récupérer un client depuis le support de stockage.

TransactionDAO : Objet chargé d'écrire ou de récupérer une opération bancaire depuis le support de stockage.

ScheduledOperationDAO : Objet chargé d'écrire ou de récupérer les événements planifiés (scheduler) depuis le support de stockage

#### 2.4.4.2. Description des interactions entre objets

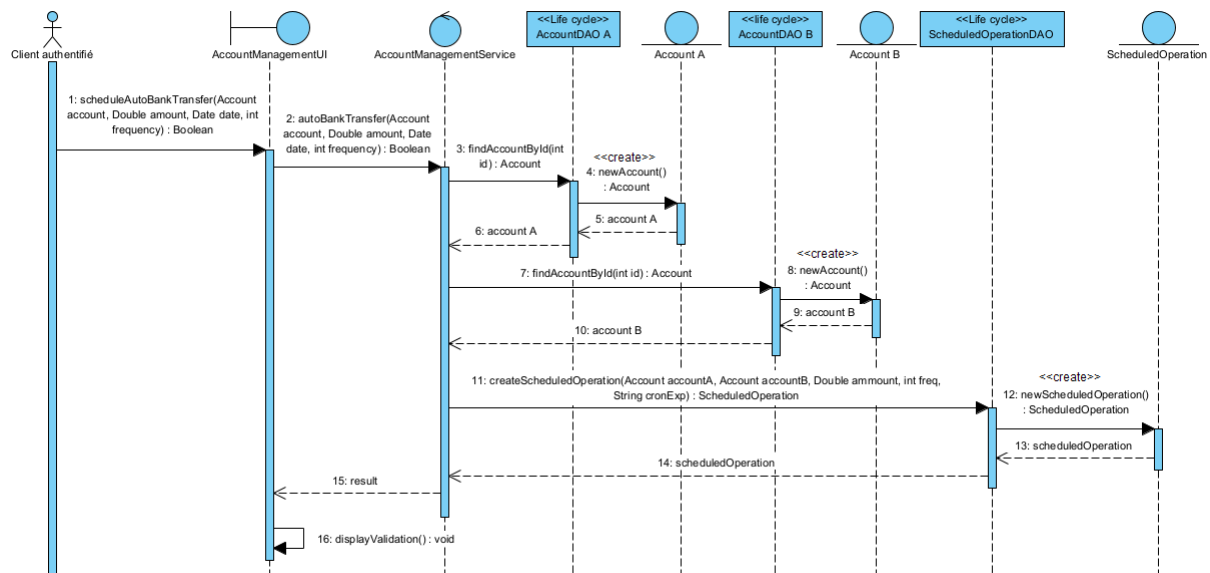


Fig. 19 : diagramme de séquences « mettre en place un virement automatique interne »

## 2.4.4.3. Description des classes

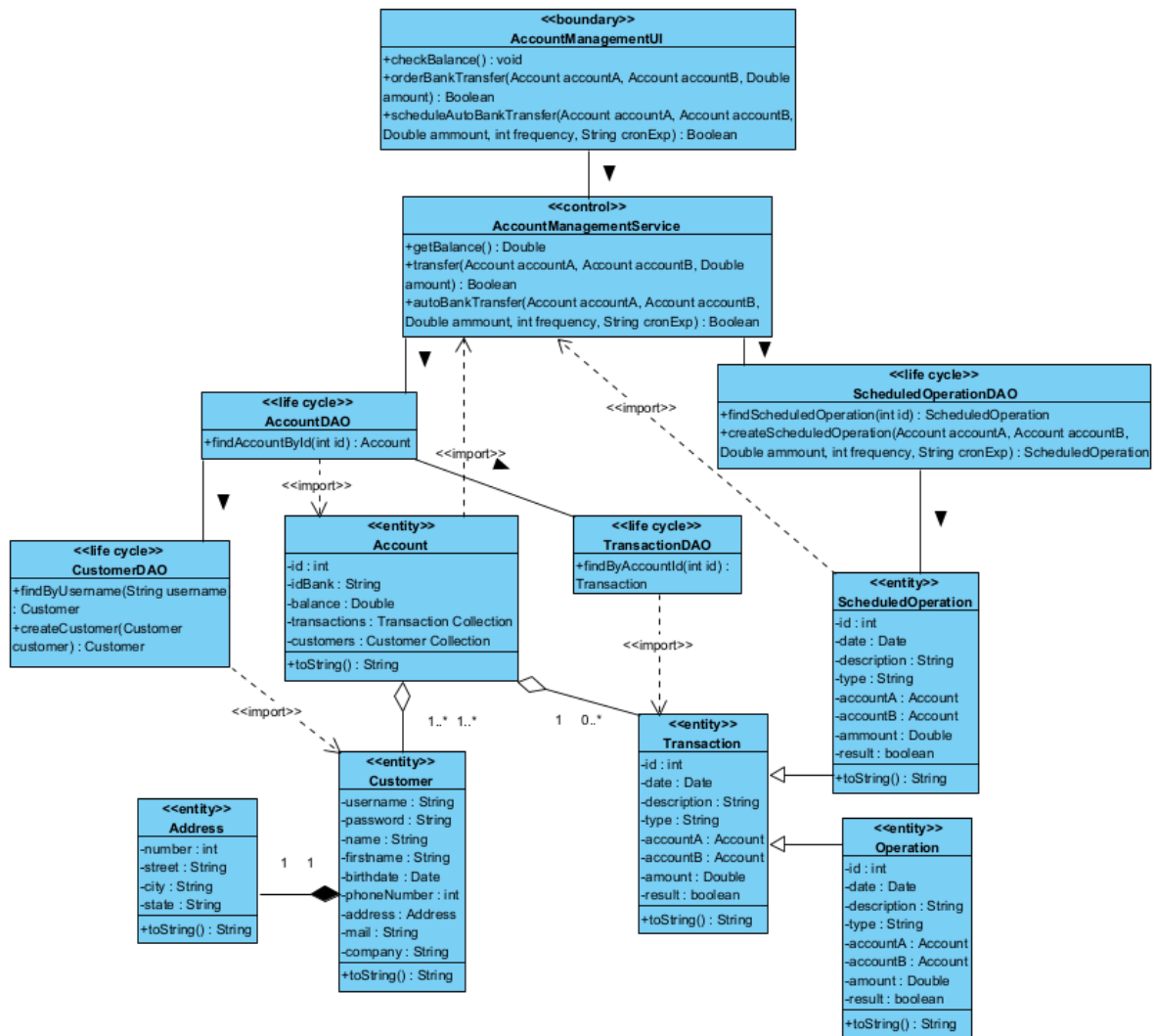


Fig. 20 : diagramme de classes « mettre en place un virement automatique interne »



### 3. Regroupement des classes

#### 3.1. Groupe domaine

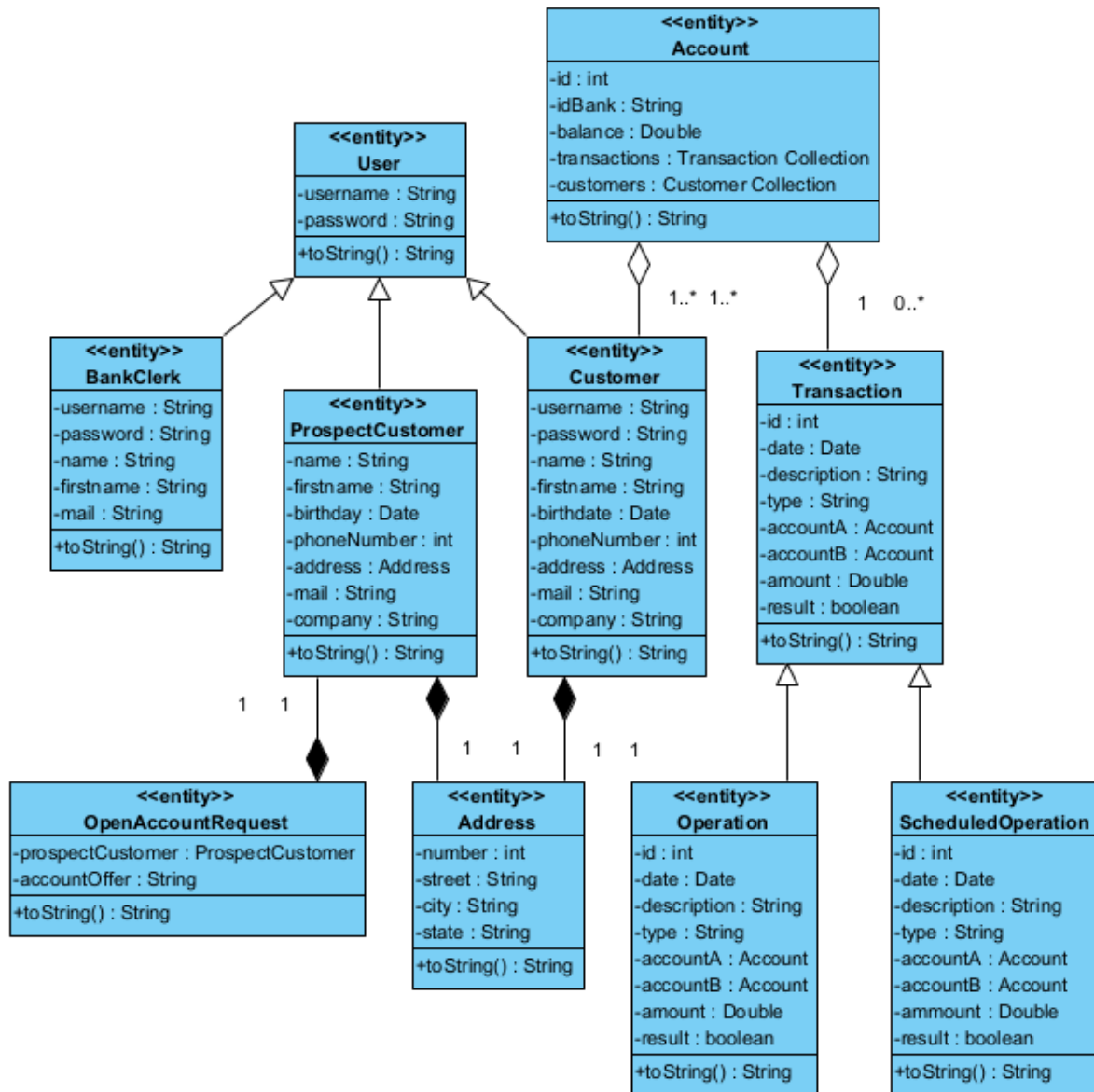


Fig. 21 : diagramme de classes « domain »

## 3.2. Groupe domaine et cycle de vie

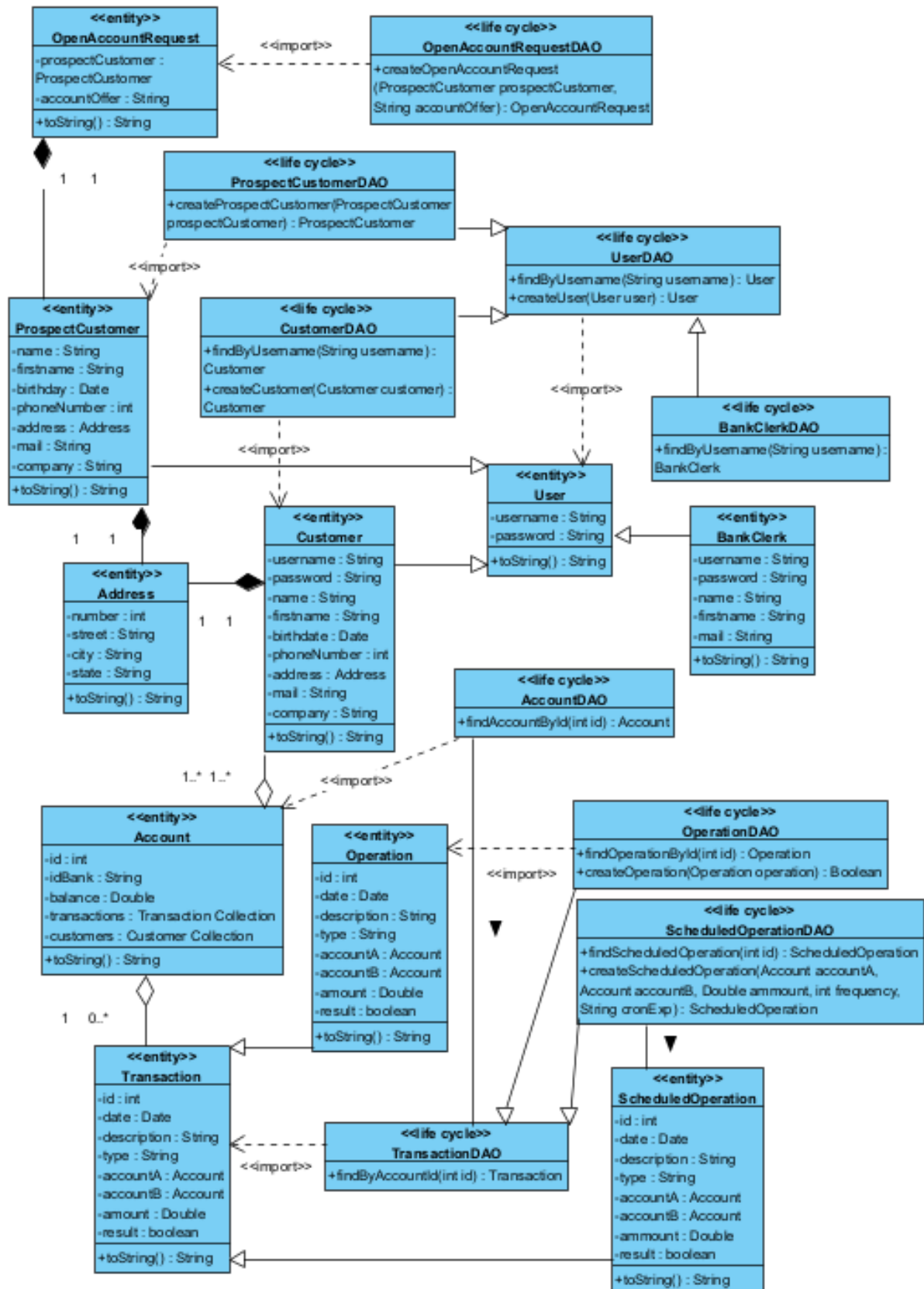


Fig. 22 : diagramme de classes « domain and life cycle »

### 3.3. Groupe Service

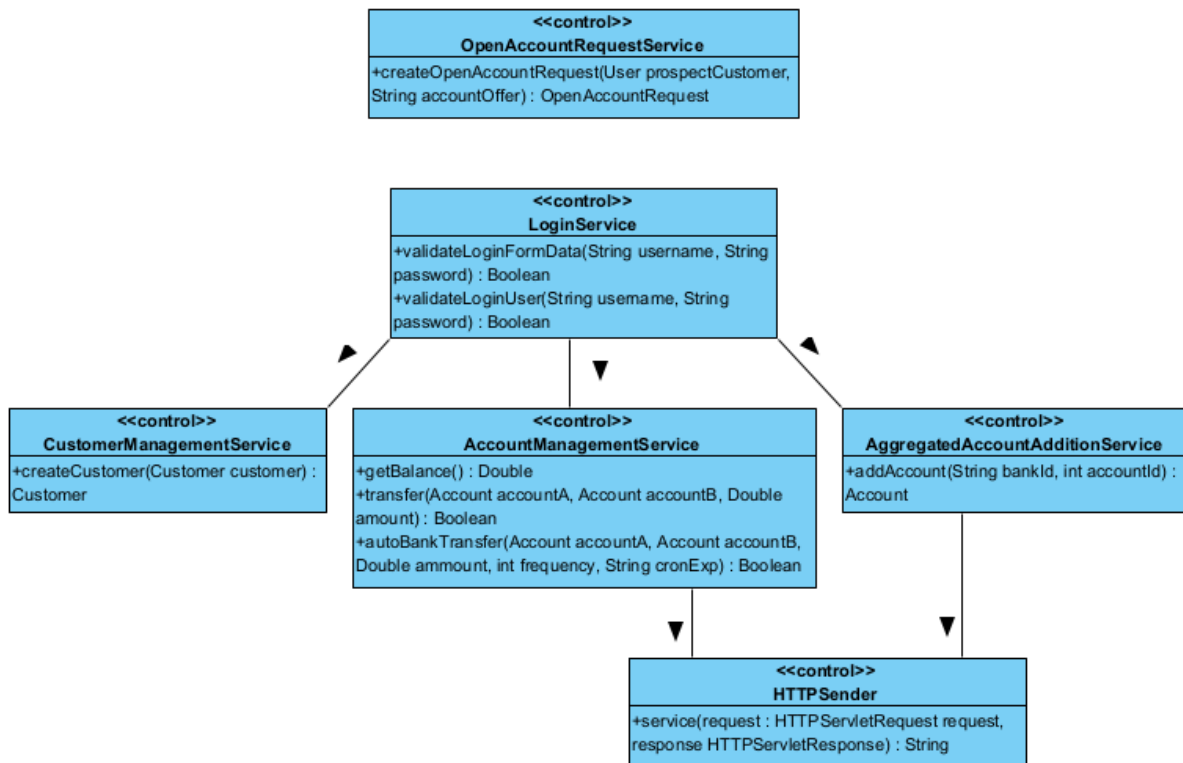


Fig. 23 : diagramme de classes « service »

### 3.4. Groupe interface utilisateur et système

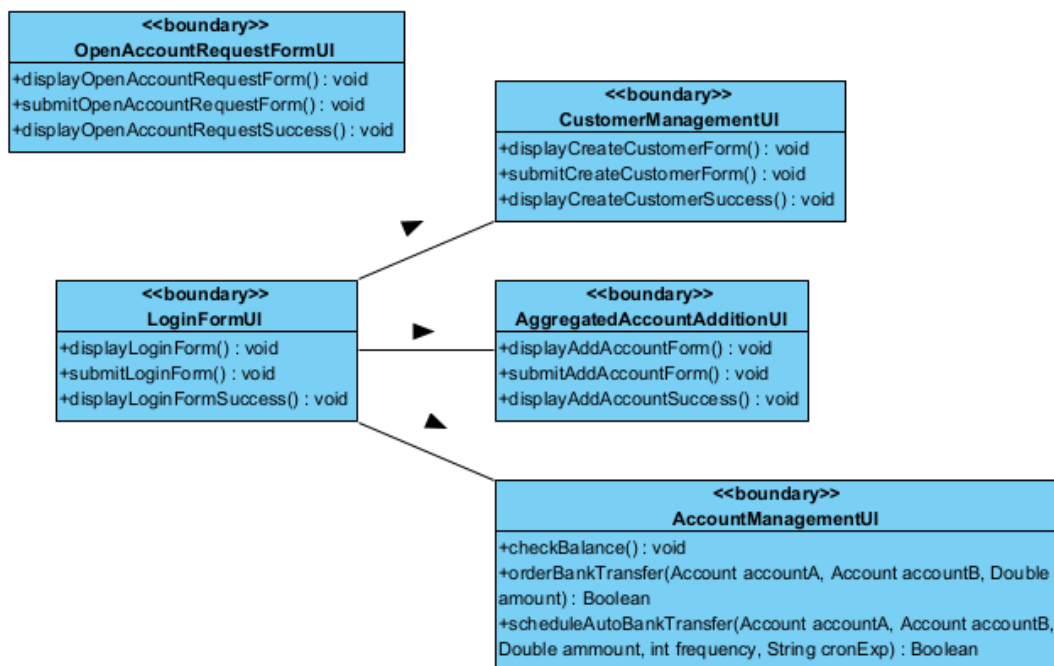





Fig. 24 : diagramme de classes « user interface »

## 4. Annexes

### 4.1. Terminologie

Entity 	Les objets « entity » encapsulent les données métier et la logique métier du système [Jacobson, et al. 1999]. (informations durables et persistantes)
Control 	Les objets « control » assurent une coordination d'autres objets. Il s'agit d'une façade entre objet « boundary » et objet « entity ».
Boundary 	Les objets « boundary » se trouvent à la frontière entre un système ou un sous-système et un acteur. Il s'agit des interfaces système.
Life cycle	Les objets « life cycle » assurent le suivi des objets « entity ». Les fonctions communes d'un objet « life cycle » incluent la création, la destruction et la localisation des objets « entity ».
User	Utilisateur de l'application Mocha Bank.
Prospect customer	Visiteur ayant effectué une demande création de compte chez Mocha Bank.
Customer	Client authentifié de Mocha Bank.
Bank clerk	Employé (authentifié) de Mocha Bank.
Open account request	Demande d'ouverture de compte bancaire.
Aggregated account addition	Ajout de compte agrégé.
Balance	Solde bancaire.
Transaction	Opération bancaire.
Deposit	Dépôt.
Bank transfer	Virement bancaire.

## 5. Index des diagrammes

Fig. 1 : diagramme de cas d'utilisation sous-système « opérations de base » .....	4
Fig. 2 : diagramme de séquences « demander une ouverture de compte » .....	5
Fig. 3 : diagramme de classes « demander une ouverture de compte » .....	6
Fig. 4 : diagramme de séquences « se connecter » .....	7
Fig. 5 : diagramme de classes « se connecter » .....	8
Fig. 6 : diagramme de cas d'utilisation sous-système « gérer les clients » .....	9
Fig. 7 : diagramme de séquences « créer un client » .....	10
Fig. 8 : diagramme de classes « créer un client » .....	11
Fig. 9 : diagramme de cas d'utilisation sous-système « agréger un compte bancaire » .....	11
Fig. 10 : diagramme de séquences « ajouter un compte bancaire externe » .....	12
Fig. 11 : diagramme de classes « ajouter un compte bancaire externe » .....	13
Fig. 12 : diagramme de cas d'utilisation sous-système « gérer le compte bancaire » .....	14
Fig. 13 : diagramme de séquences « consulter le solde du compte bancaire » .....	15
Fig. 14 : diagramme de classes « consulter le solde du compte bancaire » .....	16
Fig. 16 : diagramme de classes « effectuer un virement interne » .....	18
Fig. 16 : diagramme de séquences « effectuer un virement interne » .....	17
Fig. 17 : diagramme de séquences « effectuer un virement externe » .....	20
Fig. 18 : diagramme de classes « effectuer un virement externe » .....	21
Fig. 19 : diagramme de séquences « mettre en place un virement automatique interne » .....	22
Fig. 20 : diagramme de classes « mettre en place un virement automatique interne » .....	23
Fig. 21 : diagramme de classes « domain » .....	24
Fig. 22 : diagramme de classes « domain and life cycle » .....	25
Fig. 23 : diagramme de classes « service » .....	26
Fig. 24 : diagramme de classes « user interface » .....	26