



Université de Nouakchott

Faculté des Science et de techniques

Application

Rim banque

|  |  |
| --- | --- |
| Encadrer par:  Dr. Sidi Cheikh  Dr. Elbennay Mohamed Mahmoud  Dr. Vatimetou Mohamed Saleck  Dr. Abderrahman Sidi Mohamed | Realiser par :  Mohamedou Ahmed Mahmoud C14507  Elhancen Mohamed Souilim C14466  Lebatt Abdellahi Alloul C15849  Mohamed Abdellahi Sidi Mohamed C15623 |



Année Universitaire 2022 – 2023

Table des matières

[**Chapitre I. Présentation Générale** 2](#_Toc128175426)

[**1.** **Présentation du sujet** 2](#_Toc128175427)

[**2.** **Choix Méthodologique** 2](#_Toc128175428)

[**3.** **Le Langage de Modélisation Unifié UML** 2](#_Toc128175429)

[**Chapitre II. La Conception** 2](#_Toc128175430)

[**1.** **Diagramme de cas d’utilisation** 2](#_Toc128175431)

[A. cas d’utilisation general 3](#_Toc128175432)

[B. cas d’utilisation détails 4](#_Toc128175433)

[C. cas d’utilisation global 6](#_Toc128175434)

[**2.** **Diagramme de séquence** 6](#_Toc128175435)

[**3.** **Diagramme de classe** 6](#_Toc128175436)

[**Chapitre III. Réalisation** 6](#_Toc128175437)

[**1.** **Environnement du développement** 6](#_Toc128175438)

[Spring 8](#_Toc128175439)

[Hibernate 8](#_Toc128175440)

[Serveur Apache Tomcat 9](#_Toc128175441)

[SGBD MySQL 9](#_Toc128175442)

[StarUML 9](#_Toc128175443)

[Eclipse 9](#_Toc128175444)

[**Conclusion générale** 10](#_Toc128175445)

# **Chapitre I. Présentation Générale**

## **Présentation du sujet**

« RIMBanque » veut mettre en place une application qui gère les comptes de leurs clients (ouverture, consultation, modification et clôture d'un compte client ).

Un administrateur de l'application à le droit de réaliser toutes ces opérations, alors que le client peut consulter, déposer ou retirer de l'argent sur son compte.

Un client a la possibilité d’effectuer un virement sur le compte d'un autre client de la banque « RIMBanque » ou une autre banque.

Cette application doit offrir au client la possibilité de consulter l'historique de ses opérations validées (y compris les paiement en ligne) et/ou de commander un chéquier.

Le retrait, le versement sont immédiats alors que le virement prendra 24H.

Le paiement en ligne est réalisé à l'aide d'une carte bancaire, chaque carte est caractérisée par un numéro et un code secret et associée à un compte client. « RIMBanque » souhaite aussi connaître chaque fin du mois la situation globale des comptes clients.

## **Choix Méthodologique**

**Le Methode Scrum**

Selon la nature de notre projet, nous avons jugé que le methode scrum sera le plus adapté à sa réalisation.

**Scrum** : est une méthodologie de gestion de projet qui vise à offrir une approche souple et adaptable pour la réalisation de projets complexes. Elle est particulièrement utile dans les environnements où les exigences du projet sont incertaines ou susceptibles de changer fréquemment. La méthodologie Scrum se compose de plusieurs éléments clés, notamment :

les etapes de realisation d'un logiciel de gestion bancaire suivant la methode SCRUM

Voici les étapes de réalisation d'un logiciel de gestion bancaire suivant la méthode Scrum :

1. Étude de faisabilité et Définition des objectifs et des exigences métier : il est important de définir clairement les objectifs du projet et les exigences métier afin de s'assurer que le logiciel répond aux besoins du client. Cela peut impliquer une étude de marché et des entretiens avec les utilisateurs finaux pour comprendre leurs besoins et leurs attentes. :

Voici les objectifs et les exigences métier pour l'application de gestion de comptes bancaires "RIMBanque" d’après une étude faite sur l’existence et le sujet elle-même pour vérifier si le projet est réalisable sur le plan technique, et déterminer précisément ce que le projet doit accomplir et quels sont les besoins qui doivent être satisfaits pour atteindre ces objectifs.:

* **Objectifs :**
* Mettre en place une application permettant de gérer les comptes des clients de la banque "RIMBanque" (ouverture, consultation, modification et clôture de comptes)
* Offrir aux clients la possibilité de consulter l'historique de leurs opérations validées et de commander un chéquier
* Permettre aux clients de déposer, de retirer et de transférer de l'argent sur leur compte ou sur le compte d'un autre client de la banque ou d'une autre banque
* Proposer aux clients la possibilité de réaliser des paiements en ligne à l'aide d'une carte bancaire associée à leur compte
* Permettre à l'administrateur de l'application de réaliser toutes les opérations de gestion de comptes
* Permettre à l’administration de l’application de Connaître la situation globale des comptes clients chaque fin de mois
* **Exigences métier** :
* L'application doit être sécurisée et protégée contre les accès non autorisés
* Les clients doivent être en mesure de consulter leur compte en temps réel et de suivre l'historique de leurs opérations
* Les dépôts et retraits doivent être immédiats, tandis que les virements doivent être traités en 24 heures
* Les paiements en ligne doivent être sécurisés et protégés contre les fraudes
* L'administrateur de l'application doit avoir accès à toutes les fonctionnalités de gestion de comptes
* La banque doit être en mesure de connaître la situation globale des comptes clients chaque fin de mois

## **Le Langage de Modélisation Unifié UML**

Pour programmer une application, il ne convient pas de se lancer tête baissée

dans l’écriture du code : il faut d’abord organiser ses idées, les documenter, puis organiser la réalisation en définissant les modules et étapes de la réalisation. C’est cette démarche antérieure à l’écriture que l’on appelle modélisation ; son produit est un modèle.

Utilité des modèles

* Faciliter la compréhension d'un système

Permettre également la communication avec le client

Vision de communication, de documentation

* Définir l'architecture et le fonctionnement d'un système

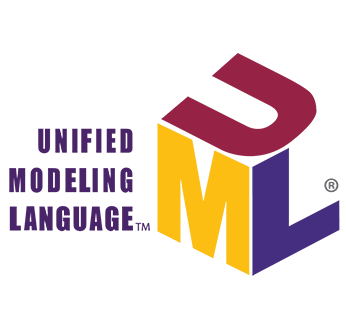
Dans ce cas, on se doit d'être le plus précis possible dans le contenu des modèles pour s'approcher du code.

Vision de développement, de production Cependant, pour ce qui concerne le langage de modélisation, nous en avons plusieurs, mais selon des points de vue techniques, le langage UML est le meilleur choix.

U.M.L est un langage qui contient les éléments constitutifs de tout langage, à savoir :

Des concepts, une syntaxe et une sémantique. Il est destiné aux phases amont de la réalisation d'un logiciel. U.M.L est une Technique de modélisation *UNIFIEE* .

De plus, UML a choisi une notation supplémentaire : il s’agit d’une forme visuelle Fondée sur des diagrammes.



UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. UML offre un standard de modélisation, pour représenter l'architecture logicielle. Les différents éléments représentables sont :

* Activité d'un objet/logiciel
* Acteurs
* Processus
* Schéma de base de données
* Composants logicie
* Réutilisation de composants.

# **Chapitre II. La Conception**

## **Diagramme de cas d’utilisation**

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs, ils interagissent avec les cas d'utilisation.

**• Cas d’utilisation**

Ils permettent de décrire l'interaction entre l'acteur et le système. L'idée forte est de dire que l'utilisateur d'un système logiciel a un objectif quand il l’utilise. Le cas d'utilisation est une description des interactions qui vont permettre à l'acteur d'atteindre son objectif en utilisant le système.

• Acteurs Ils sont des entités externes qui interagissent avec le système, comme une personne humaine ou un robot. Une même personne (ou robot) peut constituer plusieurs acteurs pour un système, c'est pourquoi les acteurs doivent surtout être décrits par leurs rôles, ces rôles décrivent les besoins et les capacités de l'acteur.

• Relation entre cas d’utilisation − Inclusions (includ) Dans ce type d'interaction, le premier cas d'utilisation inclut le second et son issue dépend souvent de la résolution du second.

**−** **Extensions (extend)**

Les extensions représentent des prolongements logiques de certaines tâches sous certaines conditions.

Autrement dit un cas d'utilisation A étend un cas d'utilisation B lorsque le cas d'utilisation A peut être appelé au cours de l'exécution du cas d'utilisation B.

− **Généralisations**

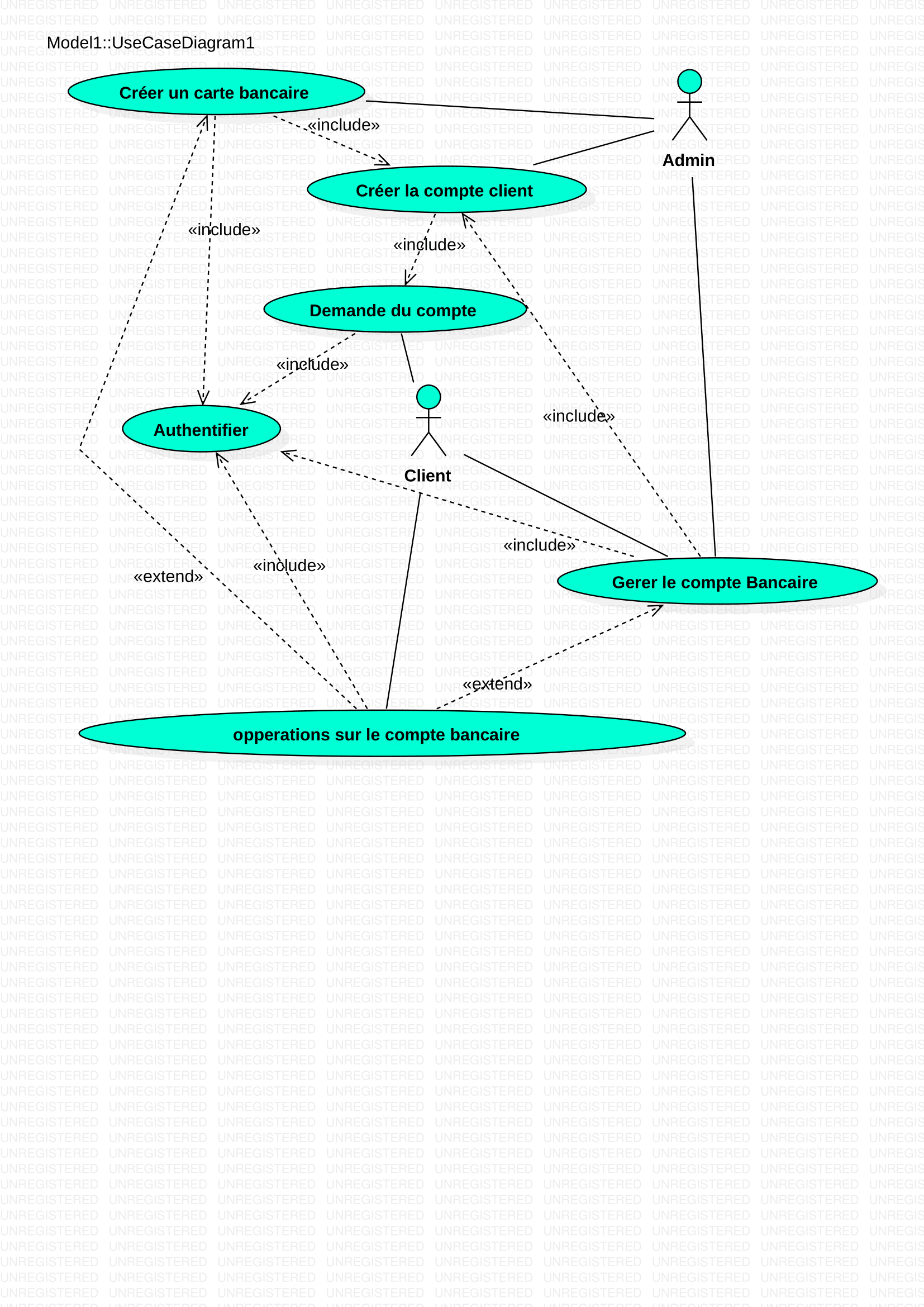
Le cas d'utilisation A est une généralisation de B, si B est un cas particulier de A c'est-à-dire lorsque A peut-être substitué par B pour un cas précis. Ces relations sont des traits pleins terminés par une flèche en triangle.

• Relation entre acteurs Il est également possible d'appliquer à un acteur la relation de généralisation.

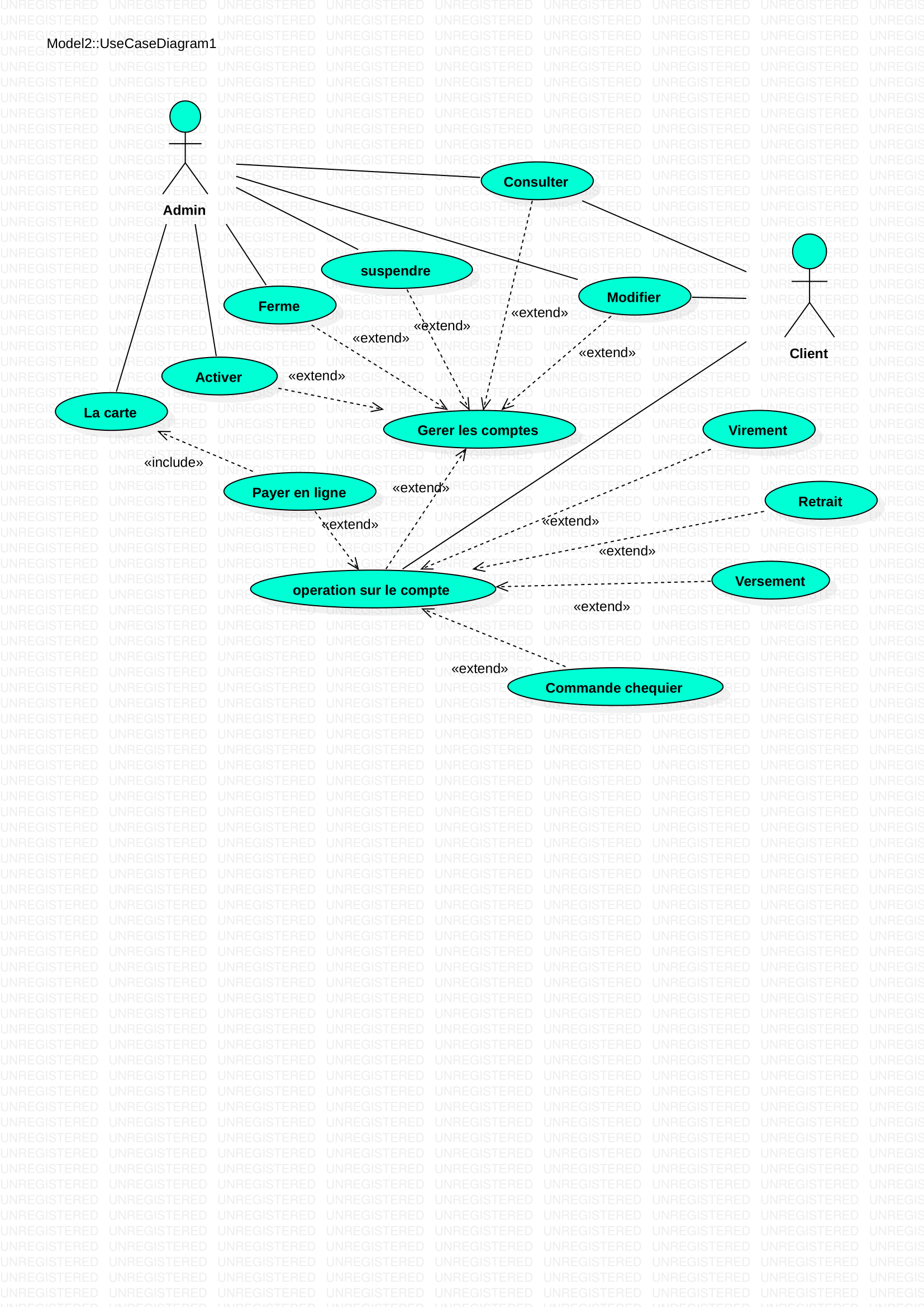
Cela se fait notamment lorsqu'un acteur est un sous-type d'une autre catégorie d'acteurs.

Un acteur lié à un autre par ce type de relation peut interagir avec le système avec plus de fonctionnalités que son parent.

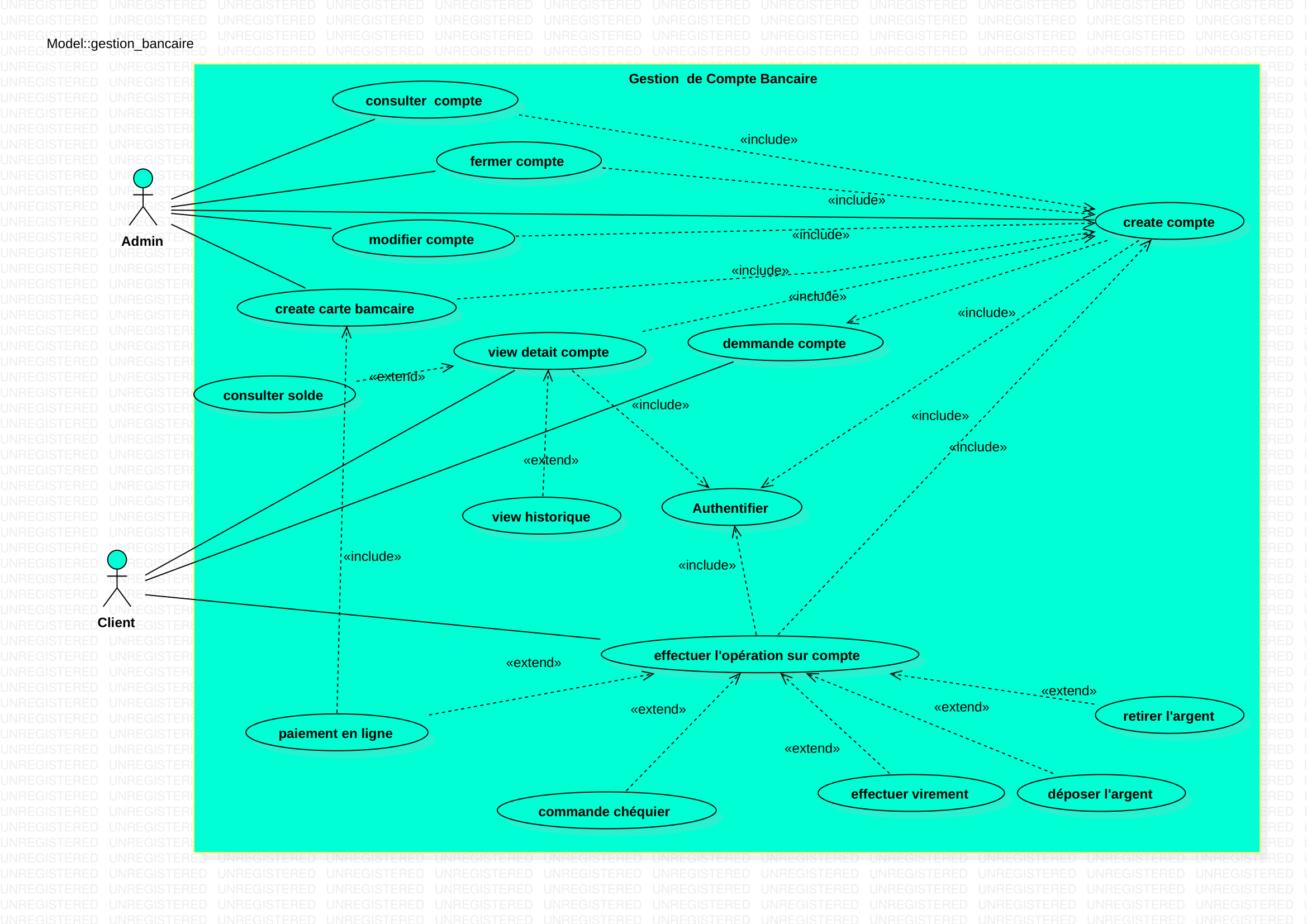
### cas d’utilisation general



### cas d’utilisation détails

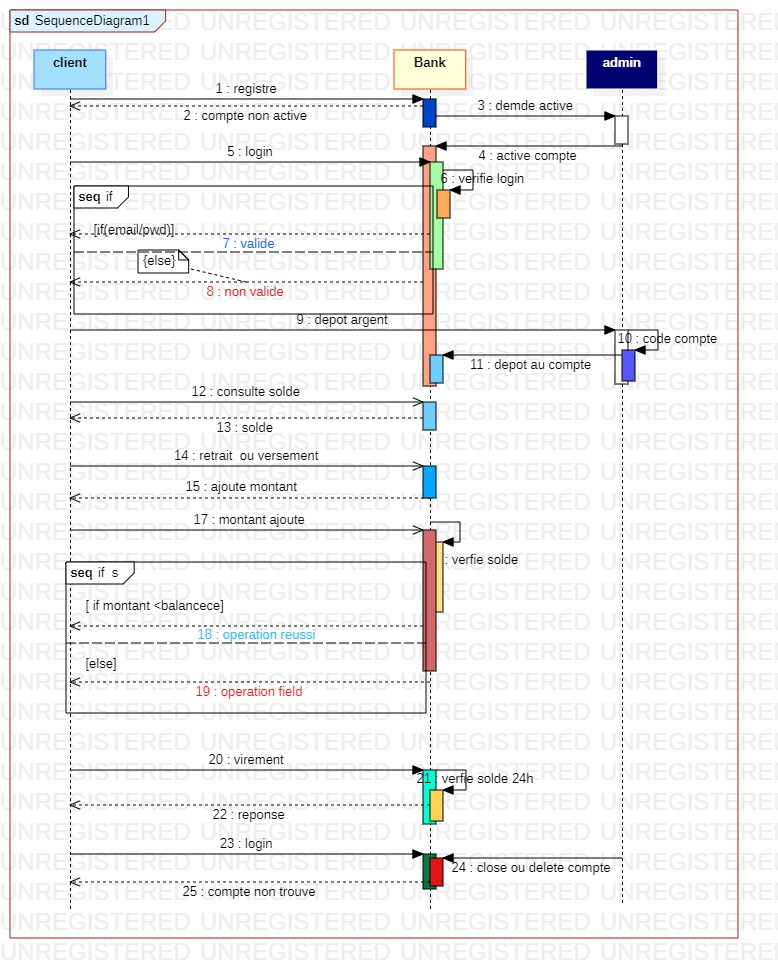


### cas d’utilisation global



## **Diagramme de séquence**

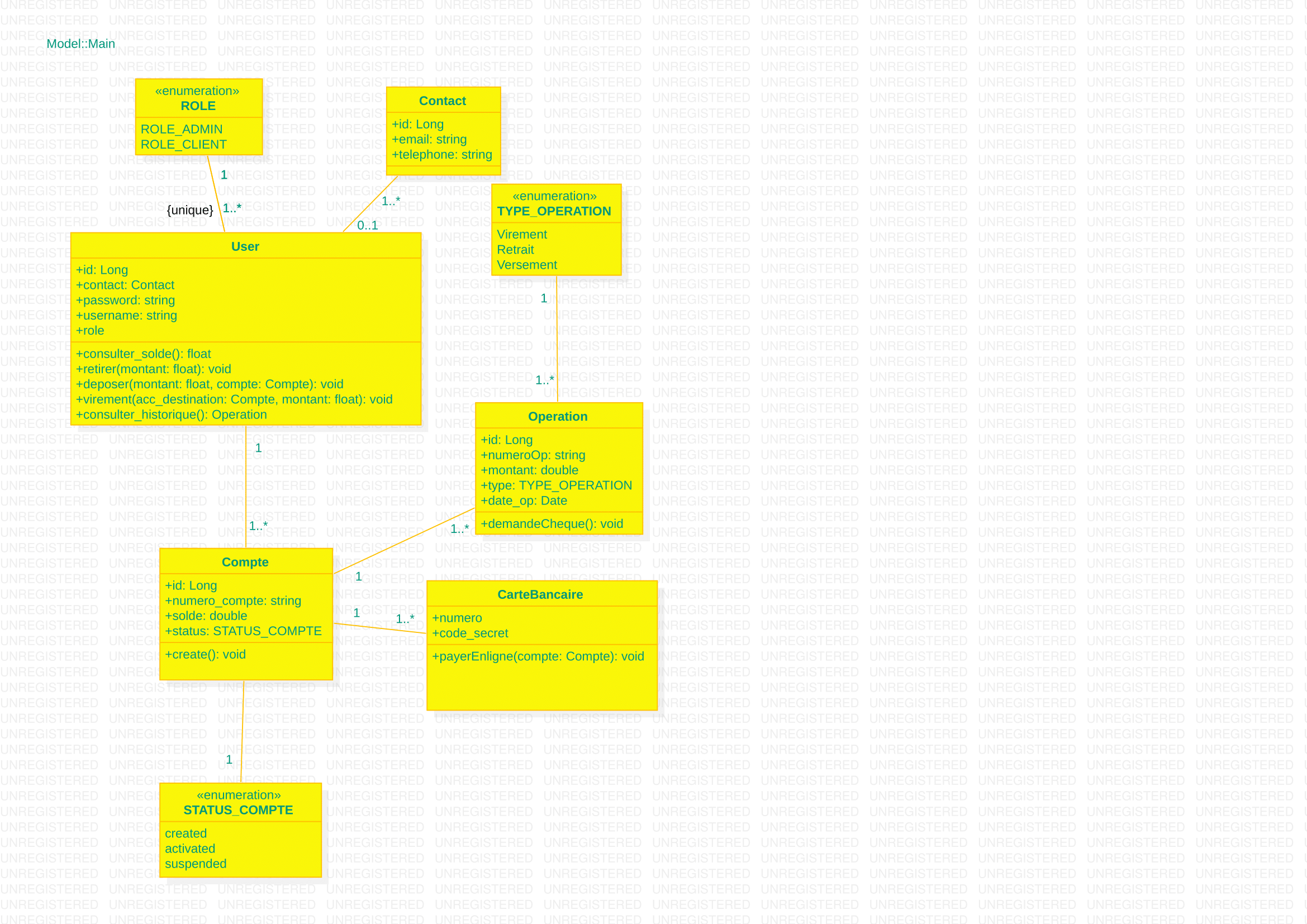
Le diagramme de séquence vise essentiellement à représenter des interactions entre les entités d’un système selon un point de vue temporel. Il possède une dimensiontemporelle mais ne représente pas explicitement les liens entre les objets. Il permet de visualiser les messages par une lecture de haut e bas. U diagramme de séquence possède deux axes : l’axe vertical qui représente le temps, et l’axe horizontal qui représente les objets qui se collaborent.



## **Diagramme de classe**

Le diagramme de classes est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci.

Ce diagramme fait partie de la partie statique d'UML car il fait abstraction des aspects temporels et dynamiques. Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe



# **Chapitre III. Réalisation**

Après avoir effectué la conception, nous allons présenter l’environnement du développement et les interfaces graphiques

de notre application.

## **Environnement du développement**

L'architecture d'une application web à page unique Spring MVC + Angular

Les applications d'entreprise intensives en formulaires sont idéalement adaptées pour être construites en tant qu'applications web à page unique. L'idée principale, par rapport aux architectures plus traditionnelles côté serveur, est de construire le serveur en tant qu'ensemble de services REST réutilisables et sans état, et du point de vue du modèle-vue-contrôleur, de retirer le contrôleur du backend et de le déplacer dans le navigateur.

Le client est capable de MVC et contient toute la logique de présentation qui est séparée en une couche de vue, une couche de contrôleur et une couche de services frontal. Après le démarrage initial de l'application, seules les données JSON sont échangées entre le client et le serveur.

Voici un exemple d'architecture à trois niveaux :

**Couche de présentation** : Il s'agit de la partie frontend de votre application. Elle peut être en HTML/CSS ou dans un framework web tel que Angular ou React. Vous voulez que cette couche affiche les données qu'elle récupère et qu'elle réfléchisse le moins possible.

**Couche d'application** : C'est le middleware de votre application. Elle peut être mise en œuvre dans n'importe quel langage tel que PHP, Java, C#, Python, etc. Cette couche est le "moteur" de votre logiciel et elle connecte la couche de présentation à la couche de données. Vous ne voulez pas de Vues ou d'interfaces utilisateur dans cette couche, car c'est le "grand penseur" de l'équipe.

**Couche de données** : Il s'agit de votre base de données ou système de stockage. Elle peut être en MySQL, PostgreSQL, MongoDB, etc. Les données sont accessibles par la couche d'application via des appels API (note : l'appel API lui-même est déclenché par la couche de présentation, mais celle-ci ne sait pas ce que la couche d'application fera avec cet appel, c'est une boîte noire entre les deux).

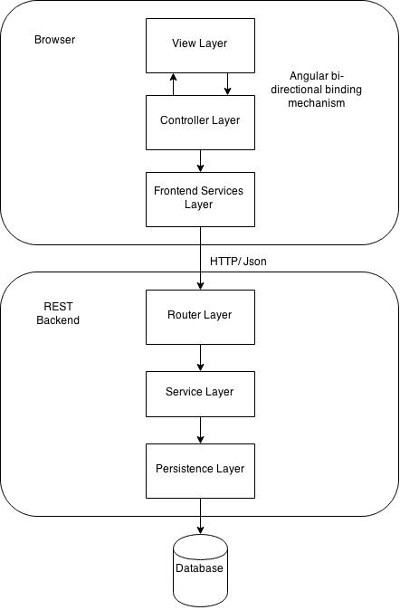
Pour des raisons sémantiques, je ne peux pas dire qu'Angular/Spring "est une architecture à trois niveaux", car cela dépend entièrement de votre implémentation. Les deux peuvent être à trois niveaux si vous le souhaitez. Dans votre cas, cependant, il est correct de dire qu'Angular devrait être la couche de présentation tandis que Spring Boot devrait être la couche d'application.

Utiliser Angular comme couche de présentation (et uniquement comme couche de présentation) a beaucoup de sens. Votre application Angular ne devrait pas se soucier des données qu'elle reçoit, elle devrait simplement les afficher. Spring Boot serait donc le composant qui livre les données de la couche de données à la couche de présentation.

Une bonne implémentation rend une architecture à trois niveaux interchangeable. Vous ne voulez plus utiliser Angular ? Il vous suffit de passer à un autre framework et les données que vos clients voient resteront les mêmes.

C'est la force d'une architecture bien conçue et ce que vous voulez accomplir avec elle

Un service Web REST (Representational State Transfer) est une interface de programmation d'applications qui permet aux applications de communiquer entre elles en utilisant le protocole HTTP et en échangeant des données au format JSON ou XML. Les services Web REST sont souvent utilisés pour exposer les données d'une application de manière accessible à d'autres applications.



### Spring



Le Framework Spring fournit un modèle de programmation et la configuration globale pour les applications d'entreprise modernes basées sur Java - sur n'importe quel type de plate-forme de déploiement. Un élément clé du printemps est le soutien de l'infrastructure au niveau de l'application: Printemps met l'accent sur la «plomberie» des applications d'entreprise afin que les équipes puissent se concentrer sur la logique métier de niveau de l'application, sans liens inutiles aux environnements de déploiement spécifiques

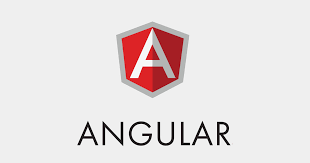
### Hibernate

**Hibernate** est un framework open source gérant la persistance des objets en base de données relationnelle.

Hibernate est adaptable en termes d'architecture, il peut donc être utilisé aussi bien dans un développement client lourd, que dans un environnement web léger de type Apache Tomcat ou dans un environnement Java EE complet : WebSphere, JBoss Application Server et Oracle WebLogic Server.

Hibernate apporte une solution aux problèmes d'adaptation entre le paradigme objet et les SGBD en remplaçant les accès à la base de données par des appels à des méthodes objet de haut niveau.

Angular



Angular est un framework de développement de front-end basé sur TypeScript qui permet de créer des applications côté client. Il fournit une structure de code et des outils pour la gestion de l'interface utilisateur et les interactions avec le serveur.

### Serveur Apache Tomcat



**Apache Tomcat** est un conteneur web libre de servlets et JSP Java EE. Issu du projet

Jakarta, c'est un des nombreux projets de l’Apache Software Foundation. Il implémente les spécifications des servlets et des JSP du Java Community Process.

### SGBD MySQL



En informatique, un **système de gestion de base de données** (*SGBD*) est un logiciel système servant à stocker, à manipuler ou gérer, et à partager des informations dans une base de données, en garantissant la qualité, la pérennité et la confidentialité des informations, tout en cachant la complexité des opérations.

### StarUML



StarUML est un logiciel de modélisation UML **open source**qui peut remplacer dans bien des situations des logiciels commerciaux et coûteux. Étant simple d’utilisation, nécessitant peu de ressources système, supportant UML2, ce logiciel constitue une excellente option pour une familiarisation à la modélisation.

### Eclipse



Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la fondation Eclipse visant à développer un environnement de production de logiciels libre qui soit extensible, universel et polyvalent, en s'appuyant principalement sur Java.

Son objectif est de produire et fournir des outils pour la réalisation de logiciels, englobant les activités de programmation (notamment environnement de développement intégré et frameworks).

Postman



Postman est un outil de développement de logiciels qui permet de tester et de déboguer les API (Application Programming Interface). Il offre une interface graphique pour envoyer des requêtes HTTP et visualiser les réponses obtenues.

Postman peut être utilisé pour tester les API de votre application, qu'elle soit déployée sur Tomcat ou sur tout autre serveur d'application. Vous pouvez utiliser Postman pour envoyer des requêtes HTTP à votre application déployée sur Tomcat et vérifier que les réponses sont conformes à ce qui est attendu. Cela peut être utile pour s'assurer que votre application fonctionne correctement et répond aux exigences de votre organisation.

**Git**



Git est un système de contrôle de version qui permet de suivre les modifications apportées aux fichiers d'un projet de manière efficace et de gérer les versions de ces fichiers. Git permet de créer des branches de développement, de fusionner les modifications et de travailler en collaboration avec d'autres personnes sur un projet.

GitHub



GitHub est une plateforme de gestion de projet en ligne qui permet aux équipes de développement de travailler sur des projets de manière collaborative. Elle offre un large éventail de fonctionnalités pour la gestion de projet, notamment :

Gestion de versions : GitHub utilise Git, un logiciel de gestion de versions, qui permet de suivre les modifications apportées au code et de revenir facilement à une version précédente en cas de besoin.

Collaboration : GitHub permet aux membres de l'équipe de travailler sur le même projet en même temps, en utilisant des "branches" pour chacun d'eux.

Les branches peuvent être fusionnées ensemble une fois que le travail est terminé.

# **Conclusion générale**