Rapport projet système distribués

Rémy Auloy

Sujet 7 : Gestion de comptes bancaire

Année 2022 - 2023

Doit composer :

* Une analyse du sujet, précisant, les structures de données les algorithmes, le paradigme avec sa justification, et une décomposition en sous problème éventuels.
* La spécification des classes principales ou des fonctions
* L’architecture logicielle détaillé
* Une évaluation / comparaison des performances
* Une documentation pour compiler et exécuter votre code
* Un jeu de test.

# Introduction

## Présentation

Pour ce projet, j’ai choisi de traiter le sujet 7, intitulé, « Gestion de comptes bancaires ». Ce sujet, doit permettre de mettre en place une simulation de gestion de comptes bancaires à l’aide de la technologie « Akka ». Nous devrons également mettre en place un système de gestion de base de données afin d’obtenir une persistance dans les données des clients.

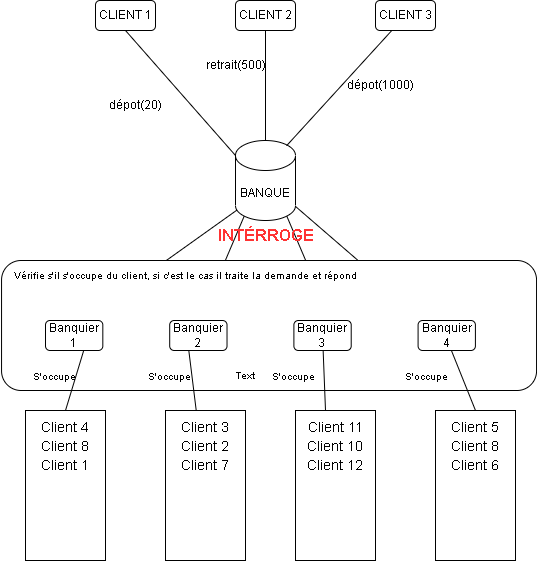
Pour informations : « Akka » est une bibliothèque de programmation concurrente pour Java qui utilise le modèle d'acteurs pour faciliter le développement d'applications distribuées. Le modèle d'acteurs est un modèle de programmation concurrente qui se concentre sur la communication entre objets logiciels plutôt que sur l'héritage. Akka permet également de gérer la concurrence et les transactions de manière transparente et offre une tolérance aux pannes et une résilience avancée pour les applications critiques qui doivent être disponibles en permanence. En résumé, Akka est une bibliothèque qui simplifie le développement d'applications distribuées en utilisant un modèle d'acteurs et offre une gestion avancée de la concurrence, des transactions et de la tolérance aux pannes.

Contextualisons la situation, nous avons une banque qui emploi des banquiers. Ces banquiers, eux, ont la charge de plusieurs compte client.

À présent, un client se présente à la banque pour déposer ou retirer de l’argent sur un de ces comptes. Il envoi donc une demande à la banque, la banque, elle, avant de demander au banquier, va chercher quelle est le banquier qui s’occupe du compte du client qui fait la demande. Une fois le banquier trouvé, la banque va interroger le banquier afin de savoir si la demande du client est possible.

Rajoutons, qu’il est totalement possible d’avoir plusieurs clients simultanément. Tout comme il peut y avoir plusieurs banquiers employés. En revanche, nous avons une seule banque, ce qui portera problème lorsque nous parlerons des problèmes éventuels.   
D’autres problèmes potentiels peuvent apparaitre mais comme je l’ai dit nous en reparlerons plus tard.

Afin de mieux visualiser ceci, voici un schéma :



## Objectifs du projet et justification de l’utilisation d’akka et de java

Pour réaliser ce projet, j'ai choisi d'utiliser la bibliothèque Akka pour gérer les acteurs et le langage de programmation Java. Les objectifs que je me suis fixés en réalisant ce projet étaient de mettre en place une simulation de gestionnaire de compte bancaire afin de comprendre comment fonctionne un tel système et d’essayer de me familiariser avec les concepts de concurrence et de gestion des acteurs.

J'ai choisi d'utiliser Akka car c'est une bibliothèque de gestion des acteurs très complète qui offre de nombreux outils pour gérer la concurrence et la distribution des tâches. En utilisant Akka, j'ai pu facilement mettre en place les différents acteurs du système et gérer leurs interactions de manière efficace. D’autant plus que nous avions découvert Akka avec java en TP

J'ai également choisi d'utiliser Java car c'est un langage de programmation populaire et largement utilisé dans l'industrie. En utilisant Java, j'ai pu bénéficier de la prise en charge de ce langage par de nombreux environnements de développement et me familiariser avec les bonnes pratiques de développement. De plus, Java offre une grande flexibilité et une grande richesse de bibliothèques pour réaliser des projets de ce type. J’ajouterais que je préfère Java aux deux autres langages possiblement choisissable C/C++ et Scala. Le C/C++ est un langage avec le quel j’ai du mal et ce qui concerne Scala, c’est un langage très puissant également et qui aurait pu parfaitement convenir a ce projet, en revanche je n’ai pas assez de connaissance dans ce langage pour me permettre de le choisir.

De plus, j’aimerais préciser que je n’ai pas développer d’interface graphique a mon application, tout se passe dans le terminal. Mais nous reviendrons sur cela plus tard.

# Analyse du sujet

## Problèmes rencontrés durant le projet

Avant de commencer à codé, je visualisais un problème majeur, problème que j’ai eu et que j’ai encore partiellement.

C’est un problème de concurrence des données, prenons un exemple :

* Le client « Rémy » possède le compte 1 géré par le banquier 1
* Le client « Tom » possède le compte 3 géré par le banquier 1

Si ces deux clients font plusieurs demandes simultanément imaginons 10 demandes. Ce qui dans la vraie vie est peu probable, mais passons. Cela signifie que plusieurs clients font de nombreuses demandes simultanées sur des comptes gérés par le même banquier, cela peut causer des problèmes de concurrence et entraîner des retards dans le traitement des demandes.   
C’est ce problème que je n’ai pas vraiment réussi à répondre. Mon, projet fonctionne parfaitement si j’ai, disons 3 clients, 3 banquiers et que je fais 5 demandes. Cependant dès que je vais avoir, disons, 50 demandes, comme il y a peu de banquiers, il y aura des retards dans le traitement des demandes et donc un « askTimeout ».

Il s ‘avère que j’ai fait des tests, et que le « askTimeout » se lève dès qu’un banquier doit traiter strictement plus de 4 demandes simultanément.

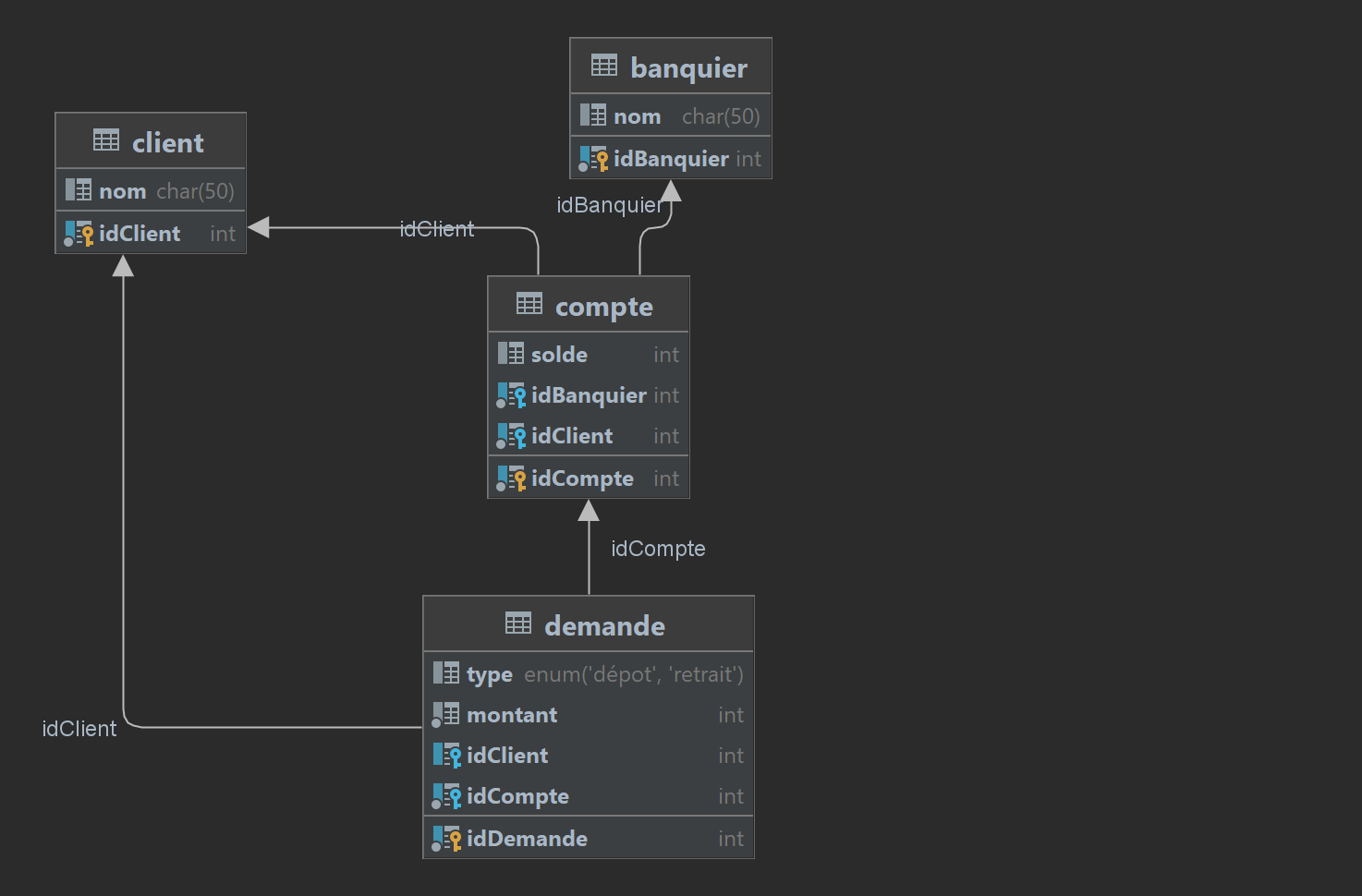
Les solutions pour résoudre sont multiples, par manque de temps et d’expérience sur Akka, je n’ai pas réussi à les mettre en place de manière propre, je préfère livrer quelque chose qui marche un minimum plutôt que quelque chose qui ne compile même pas.

Revenons aux solutions que j’aurais pu mettre en place afin de résoudre ce problème :

* Utiliser une file d’attente, un acteur qui gère une file d’attente des demandes des clients. Le banquier traitera alors qu’une demande à la fois et attendra que la demande soit traitée avant de passer à la suivante. Cela permettra gérer la concurrence de manière organisée et évite de surcharger le banquier. Cependant en faisait ça, les banquiers traiteront les choses de manière séquentielle et non parallèle. Donc cela serait contraire au principe d’akka.

Comme nous souhaitons, mettre en place un système de persistance des données, avant de vouloir coder, il faut concevoir notre base donnée.

Comme nous décidons d’avoir uniquement une seule banque, nous n’avons pas besoin de modéliser la table « Banque », si nous le faisions elle serait inutile.



1 : Diagramme de la base de données

J’ai pris le choix, le stocker les demandes des clients dans une classe demande. Afin de pouvoir avoir un historique des demandes si nous le souhaitons.

Détaillons ce diagramme afin de vous faire mieux comprendre la façon dont j’ai raisonné.

Je vais détailler les relations afin de mieux comprendre les clés étrangères qui existent.

* La relation client/compte :

Un client peut avoir 0 ou plusieurs comptes et un compte est possédé par 1 seul et même client.

D’où la clé étrangère idClient dans Compte. C’est car nous avons une relation de type (1,N) .

* La relation client/demande :

Un client fait 0 ou plusieurs demandes et une demande peut être faite par 1 seul et même client.  
Cela explique la clé étrangère idClient dans demande, pour la même raison que précédemment, nous avons une relation de type(1,N)

* La relation banquier/compte :

Un banquier gère 0 ou plusieurs demandes et une demande peut être géré par un seul banquier.

Encore une fois ici, nous avons une relation de type(1,N), ce qui explique la clé étrangère idBanquier dans la table compte

* La relation compte/demande :

Une demande concerne 1 seul compte et un compte peut être concerné par plusieurs demandes.

Encore une fois, on a une relation de type(1,N) ce qui explique la clé étrangère idCompte dans demande.

Ainsi, j’espère vous avoir bien fais compris le raisonnement que j’ai eu afin de mettre en place la base de données qui sera utilisé plus tard.

### État des données dans le code