POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

UNE BANQUE TOURNÉE VERS LE MARKETING NUMÉRIQUE

Rapport D'architecture

Auteurs :
Jules Prince - Théophile Yvars - Hamza Zoubair - Roméo David Amedomey

13 novembre 2023

Table des matières

1 Introduction

Dans le cadre du projet d'architecture logicielle, notre équipe travaille sur la mise en place d'un nouveau système bancaire. Notre approche choisie se concentre sur le marketing et l'analyse de données via un site web.

Nous avons consacré une attention particulière à l'architecture pour renforcer cette fonctionnalité unique. Notre objectif est de proposer une expérience personnalisée à nos clients en fonction de leurs habitudes d'utilisation de nos services bancaires.

Nous avons décidé de présenter aux clients une liste de produits offerts par notre banque, alignés sur leurs habitudes de consommation de nos services. Parmi ces produits, on retrouve des cartes bancaires offrant divers services, des options activables pour répondre aux besoins spécifiques des clients, ainsi que des assurances visant à les assister et les protéger dans leurs quotidiens.

Ces services et produits seront présentés aux clients par le biais de publicités directement sur leur compte. Ils pourront recevoir des notifications via notre application, des e-mails personnalisés ou des SMS, ainsi que des appels téléphoniques de nos conseillers.

Nos banquiers auront accès à un ensemble complet d'informations sur les habitudes de consommation de nos clients via leur espace dédié. Ils pourront également suivre en temps réel les transactions effectuées sur notre réseau. Cette visibilité leur permettra de comprendre quels services rencontrent le plus de succès auprès de la clientèle et quels sont ceux qui sont moins sollicités, ce qui nous aidera à adapter notre offre de manière plus efficace.

2 Besoins et Personna

Nous allons identifier les différent personna et pour chacun d'entre eux leurs besoins.

Suzanne Martin, avocate de 35 ans et cliente fidèle, cherche une expérience bancaire moderne. Elle souhaite effectuer des transactions en ligne, consulter rapidement son compte, et apprécier des publicités ciblées ainsi que des recommandations de produits bancaires personnalisées. Pour elle, une banque proactive et flexible répondrait parfaitement à son mode de vie actif.

Xavier Dupont, banquier expérimenté de 40 ans, souhaite une plateforme bancaire efficace. Il a besoin de consulter rapidement les comptes et les transactions de ses clients, visualiser les produits de la banque, obtenir des statistiques sur les transactions, et voir les produits spécifiquement proposés à ses clients. Pour Xavier, une solution bancaire intégrée et axée sur la performance est essentielle pour offrir un service de qualité à ses clients.

3 Itérations Hebdomadaires

3.1 Semaine 39

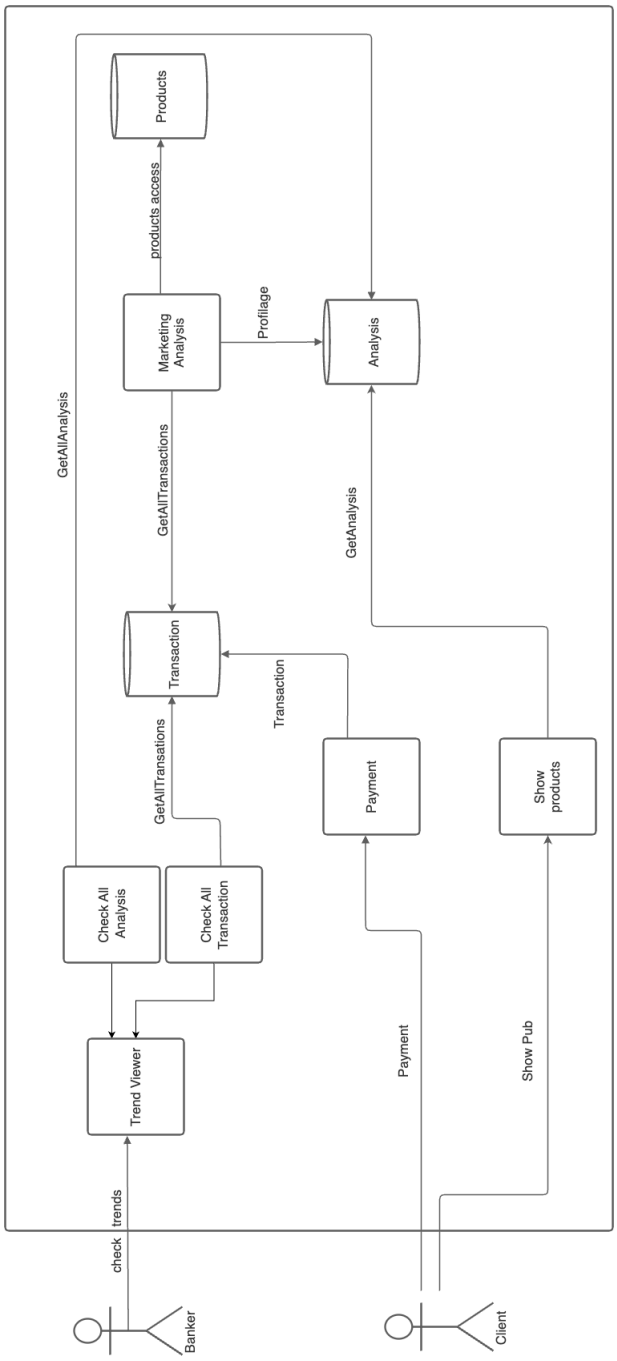

FIGURE 3.1.1 – Diagramme d'Architecture semaine 39

Notre architecture proposé en cette semaine 39 met l'accent sur la fonctionnalité de marketing et d'analyse de données en ligne. On ne retrouve que 2 acteurs, le client et le banquier.

On ne s'intéresse pas encore à la création de compte ni à la gestion de compte client. Le plus important dans notre système est la prise en compte des transactions et des habitudes du client pour définir des produits à lui présenter.

Idem pour le banquier car la création d'un compte du banquier n'est pas la priorité. Ce dernier dois pouvoir effectuer une consultation des statistiques courantes du système.

3.1.1 Point faible

- **Table Transaction reçoit trop d'appels** : Nous constatons que la table "Transaction" est fortement sollicitée en termes d'appels ou d'accès. Cela peut être un signe de surcharge du système, ce qui peut entraîner des problèmes de performances. Il est essentiel d'examiner comment nous pouvons optimiser l'accès à cette table pour réduire la pression sur la base de données et améliorer la réactivité du système.
- **Table Analysis recoit trop d'appel** : De manière similaire, la table "Analysis" subis un nombre élevé d'appels. Cela peut affecter la vitesse de traitement des requêtes liées à l'analyse des données, ce qui peut potentiellement ralentir le système. Nous devons envisager des moyens d'optimiser l'accès à cette table.
- **Gestion des comptes clients trop légères** : Une autre observation concerne la gestion des comptes clients, où les comptes sont actuellement représentés par de simples identifiant clients ("idClient") dans les transactions. Cette approche peut être insuffisante pour offrir une expérience client personnalisée et pour mener des analyses approfondies. Nous devrions envisager d'enrichir la gestion des comptes clients en stockant davantage d'informations pertinentes, ce qui nous permettra de mieux comprendre le comportement des clients et d'adapter nos services en conséquence.

3.1.2 Point fort

- **Peu d'acteurs :** La simplicité du système est un atout majeur. Le fait qu'il y ait peu d'acteurs impliqués facilite la gestion des flux d'informations et réduit la complexité de la communication entre les différents composants. Cela devrait contribuer à une mise en œuvre plus fluide et à des performances plus prévisibles.
- **Gestion des tendances par 2 services performants :** La gestion des tendances par deux services distincts et performants est une décision intelligente. Cela permet de répartir la charge de travail et de garantir une réactivité élevée pour l'analyse des tendances. Il est essentiel de maintenir la performance pour répondre aux besoins du marketing et de l'analyse de manière efficace.
- **MarketingAnalysis efficace :** L'efficacité de la fonction MarketingAnalysis est cruciale. Cela signifie que nous pouvons fournir des informations pertinentes et exploitables pour orienter nos stratégies marketing. Une analyse de haute qualité devrait nous permettre de prendre des décisions plus éclairées pour satisfaire nos clients et améliorer notre offre.
- **User story bien représentée :** La représentation précise des user stories (scénarios utilisateur) est un gage de réussite. Cela garantit que nous comprenons les besoins et les attentes de nos utilisateurs finaux. Une bonne représentation des user stories devrait nous aider à concevoir des fonctionnalités qui répondent à leurs besoins de manière efficace.
- **Plutôt simple d'implémentation :** La simplicité d'implémentation est un avantage majeur, car elle réduit les risques d'erreurs et facilite la maintenance à long terme. En minimisant la complexité, nous pouvons nous concentrer sur la mise en œuvre de manière plus efficace et éviter les écueils inutiles.

— **Mise en avant de l'essentiel dans notre variante :** Il est crucial de se concentrer sur les aspects les plus importants de notre variante, en mettant en avant les fonctionnalités clés qui apportent une réelle valeur ajoutée. Cela permettra de maintenir la clarté et la pertinence de notre solution.

3.2 Semaine 40

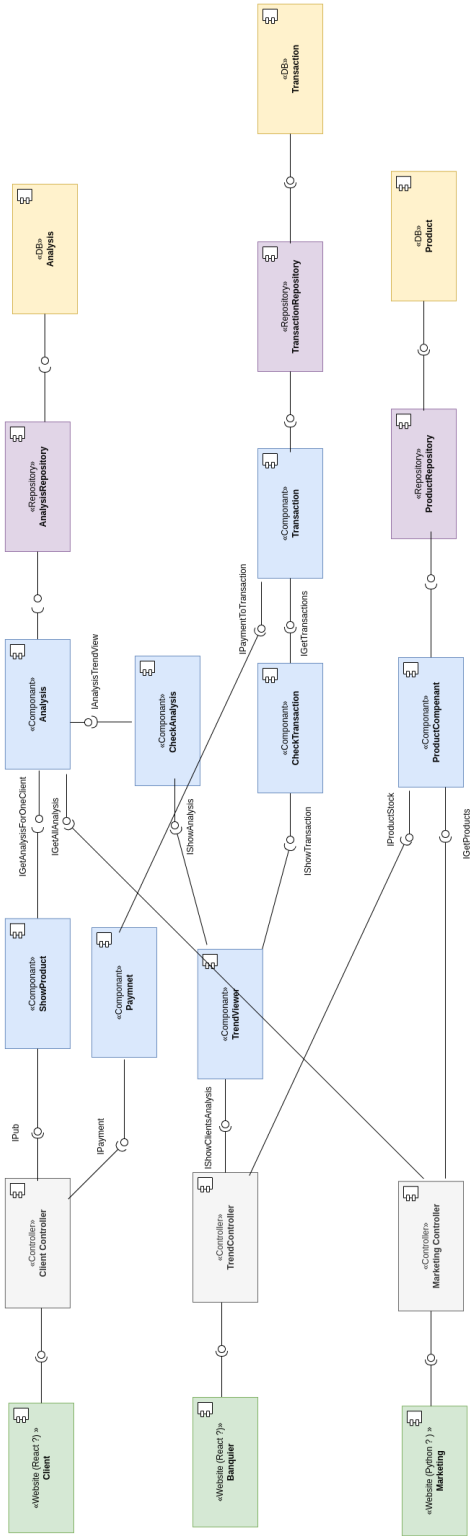

FIGURE 3.2.1 – Diagramme de composant

Durant cette semaine nous avons réaliser le diagramme de composant.

3.3 Semaine 41

Durant cette semaine nous avons décider de changer notre approche.

Après avoir réaliser notre diagramme de composant nous nous sommes rendu compte que il serait plus judicieux de casse notre monolithe.

Continuer avec notre architecture en monolithe nous aurait poser des problèmes de scabalité et des complexités dans le développement.

Pour surmonter ces défis, nous avons fait le choix de migrer vers une architecture plus moderne et modulaire, basée sur des microservices. Cette approche permet de découper l'application en petits services indépendants, chacun dédié à une fonctionnalité spécifique. Cette modularité facilite la scalabilité, permettant d'ajuster et de mettre à l'échelle chaque service individuellement en fonction des besoins. De plus, le déploiement et la mise à jour des microservices peuvent être effectués de manière indépendante, offrant une plus grande agilité dans le développement et la maintenance du système.

En adoptant une architecture basée sur des microservices, nous anticipons une amélioration significative de la flexibilité, de la scalabilité et de la facilité de développement, tout en assurant une meilleure gestion des défis liés à la croissance de notre application.

Durant cette semaine nous avons également décider d'ajouter un pattern CQRS sur le service de transaction. Nos clients vont effectuer des transactions à un rythme effrénné. Nous avons donc besoin de séparer la lecture de l'écriture pour des besoins d'optimisation.

3.4 Semaine 42

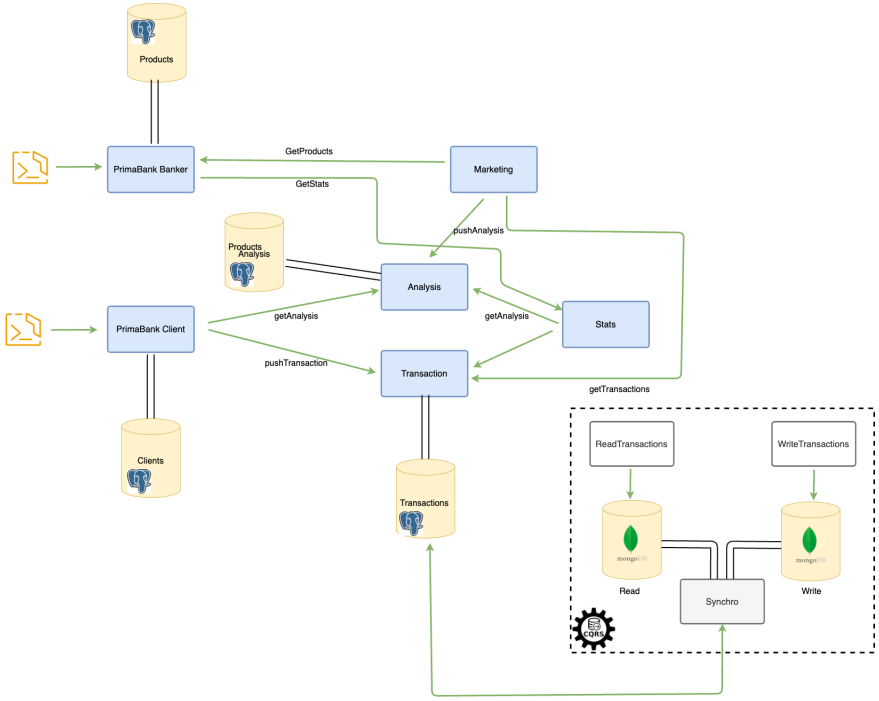

FIGURE 3.4.1 – Diagramme d'architecture semaine 42

Le service Marketing est capable désormais de renvoyer des offres au services d'analyse.

3.5 Semaine 43

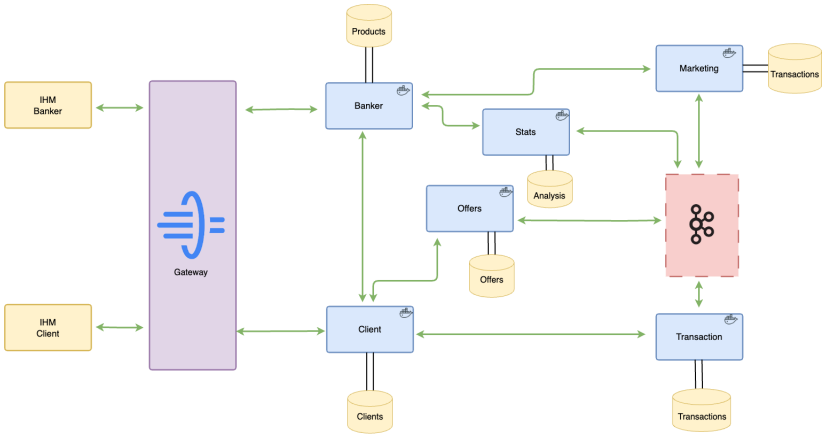

FIGURE 3.5.1 – Diagramme d'architecture semaine 43

Durant cette semaine nous avons décider de mettre en place un bus Kafka et ce pour faciliter l'échange de données entre nos services.

Le service de transaction continuera à acheminer les transactions vers les services de marketing et de statistiques en permanence. Ces deux services seront activés une fois par jour, car ce sont des services travaillant avec une grande quantité de données et pourraient donc potentiellement être gourmands en ressources, en particulier si le service marketing déploie un modèle d'intelligence artificielle.

De plus, lors de l'exécution de leurs processus, si ces services n'ont pas déjà l'ensemble des transactions dans leur base de données, ils devront effectuer un grand nombre de requêtes pour récupérer toutes les données nécessaires. Les services de marketing et de statistiques sont abonnés au topic de transaction et stockeront chacun de leur côté les transactions des clients.

Nous avons également déployer un service gateway permettant de centraliser les points d'entrées à nos services.

3.6 Semaine 46

La semaine 46 est la semaine avec la version la plus récente de notre code et de notre architecture.

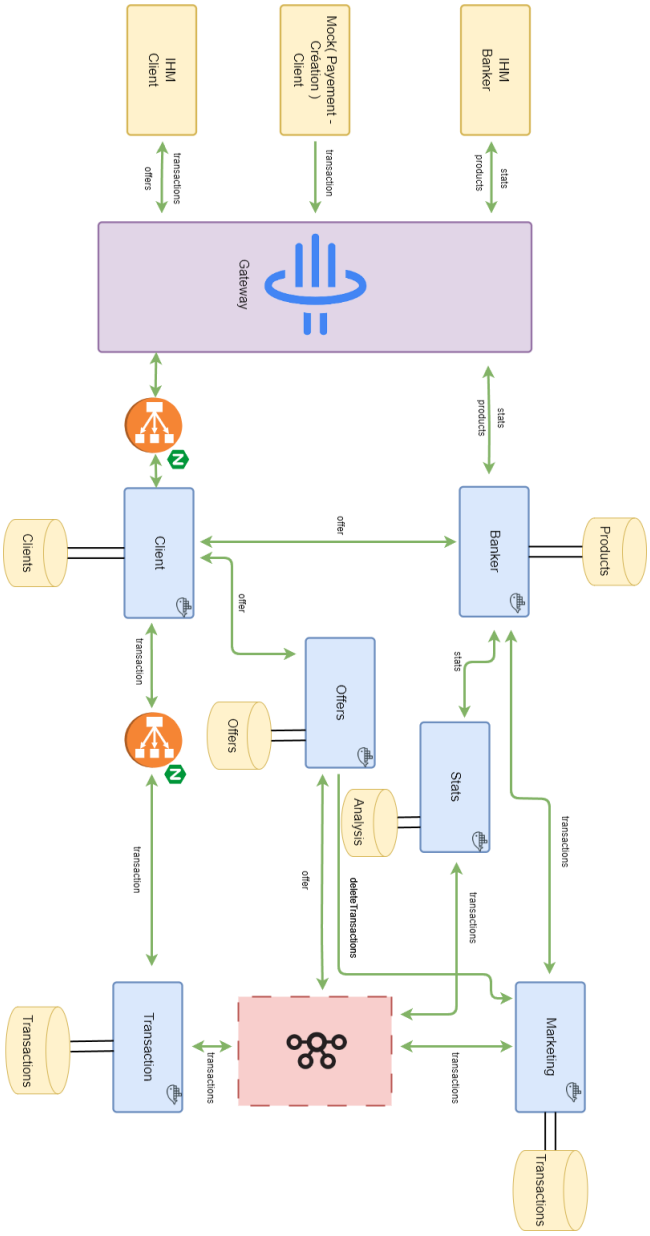

FIGURE 3.6.1 – Diagramme d'architecture semaine 43

Les fonctionnalités implémenté sont les suivantes :

— Le client peut consulter son compte et peut effectuer des transactions.
— Le banquier peut consulter tous les comptes des clients, les produits de la banque, les transactions effectués par chaque client.
— Le client peut consulter sur son compte des propositions de produits selon ses comportements.
— Le banquier peut voir des statistiques sur les transactions effectuées (nombre de transactions par pays, par type soit particulier ou entreprise, par origine soit internet ou physique, ...)

3.6.1 Explications de l'architecture

L'architecture est composée des services suivants :

- **Client** : Ce service permet aux clients de récupérer les informations de leurs comptes, effectuer des transactions et voir des offres (produits recommandés).
- **Banker** : Ce service permet au banquier de consulter les comptes des clients et leurs transactions, et consulter les statistiques.
- **Transactions** : Ce service est dédié pour effectuer des transactions et les stocker dans une base de données dédiée aux transactions. Chaque transaction effectuée par le client est envoyée dans le bus Kafka est envoyée aux deux services Marketing et Stats.
- **Marketing** : Ce service est chargé de récupérer les transactions et faire des analyses pour proposer pour chaque client un produit. Un job est lancé chaque jour à minuit pour effectuer des nouvelles analyses.
- **Stats** : Ce service est responsable de faire les calculs de statistiques à partir des transactions, comme Marketing un job est lancé quotidiennement pour calculer les statistiques et les stocker dans la base de données dédiée.
- **Offers** : Ce service renvoie les recommandations aux clients. Marketing quand il fait ses analyses il les envoie par le bus Kafka vers le service Offer qui les stocke dans sa base de données Analysis.
- **Gateway** qui constitue le point d'entrée de notre système pour tous les clients externes (client et banquier)

3.6.2 Argumentations de l'architecture

On a mis en place un pattern CQRS pour séparer les écritures des transactions dans la base de données et la lecture faite par les deux services Stats et Marketing. Chaque transaction écrite dans la base de données des transactions est persistée dans les bases de données des services Marketing et statistiques, ces derniers vont utiliser ces données pour faire leurs calculs. La synchronisation est faite à l'aide du bus Kafka, c'est le service Transaction qui produit chaque transaction crée et ensuite les 2 autres services consomment depuis le topic « primabanktransaction ».

Notre système contient également une Gateway, qui présente un point d'entrée unique vers les micro-services, cela est un point positif puisque nous avons deux fronts Client et Banker. Alors, les deux interfaces vont communiquer directement avec la Gateway. On a ajouté un pattern Circuit Breaker pour anticiper les pannes des services surtout Client et Transaction.

Ensuite, on a mis en place un load balancer avant les services clients et transactions, ce choix est fait parce que c'est la partie du système qui reçoit une charge importante. En effet, on peut avoir beaucoup de transactions au même instant. Puisque tous nos services sont stateless, nous avons pu faire de la scalabilité horizontale pour les 2 services transactions et clients, et le load balancer va permettre de distribuer la charge entre les instances.

4 Users Story

Nous avons construit 3 fronts : le premier front est le site du client, le site du banquier et une interface pour injecter les clients, les transactions avec des paramétrages (pourcentage des transactions en France, Étranger, type de transactions : entreprise ou particulier), ce front est important pour montrer notre POC pour recevoir des recommandations différentes selon les transactions de chaque client.

- **User Story 1** Le client effectue une transaction : Le client passe par son site qui va appeler directement la Gateway, qui communique avec le service client, ce dernier va faire les vérifications nécessaires sur le client avant d'envoyer la transaction au service des transactions qui va effectuer le paiement et persister la transaction dans la BD. Ensuite, cette transaction est envoyée dans le bus Kafka et consommé par les 2 services Marketing et Stats qui vont la stocker dans leurs BD.

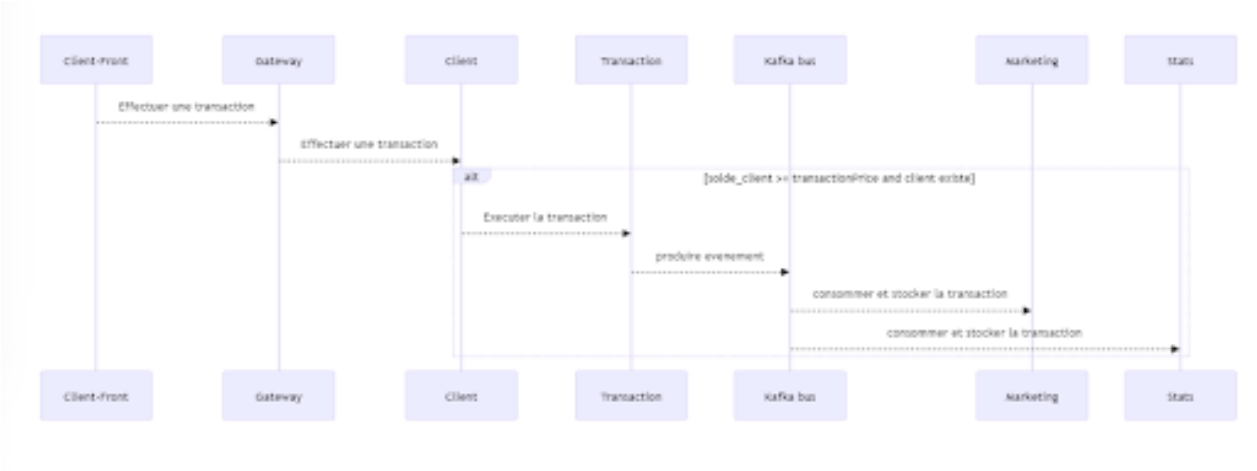

FIGURE 4.0.1 – Diagramme de séquence User Story 1

- **User Story 2** Effectuer des analyses sur les transactions : Le service Marketing est appelé par un Cron Job, ce service va récupérer toutes les transactions depuis sa BD locale, il fait son traitement sur les transactions de chaque client. Après que toutes les analyses sont prêtes, ce service les envoient dans le bus Kafka (chaque analyse référence un produit). Ils sont ensuite consommés et stockés par le service Offers.

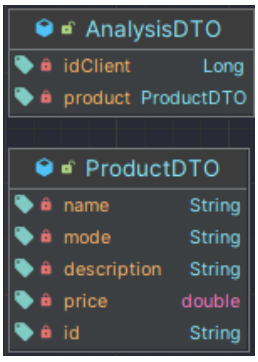

FIGURE 4.0.2 – Décomposition de classe Analysis et Product

- **User Story 3** Le client voit une offre : Le client se connecte sur son site, passe à travers le service Client qui va demander au service Offers les analyses du client et ensuite le client voit sur son site tous les produits proposés.

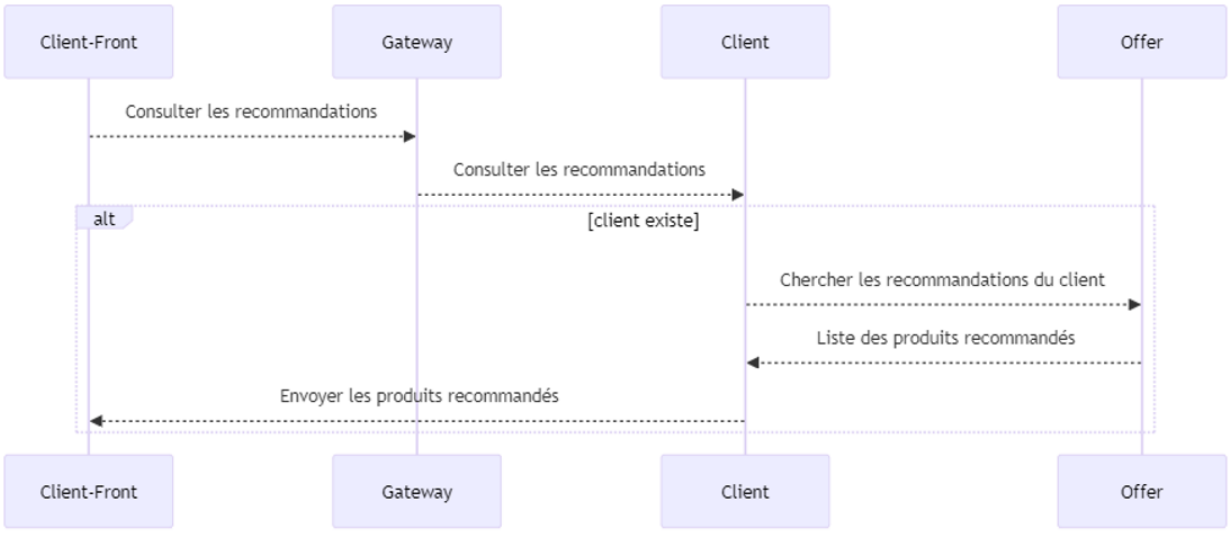

FIGURE 4.0.3 – Diagramme de séquence User Story 3

- **User Story 4** Statistiques calculés : Le service Stats est appelé par un Cron Job qui va récupérer toutes les transactions et faire les statistiques puis les stocker dans sa BD locale.
- **User Story 5** Le banquier voit les statistiques : Le banquier se connecte sur son site, passe par le service Banker qui va demander à Stats les derniers statistiques qui existent dans la base de données.

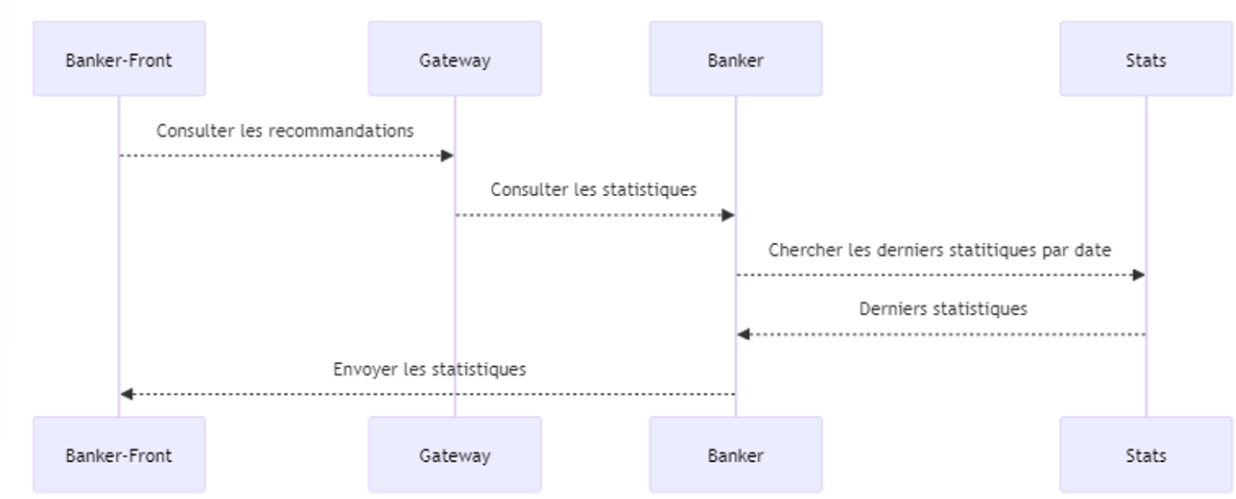

FIGURE 4.0.4 – Diagramme de séquence User Story 5

5 Points faibles de notre architecture

La base de données des transactions :Nous avons utilisés pour cette fonctionnalité une base de données de type PostgreSQL. Hors, une DB nosql serait plus approprié puisque qu'il n'y a aucune relation dans cette DB. On stocke uniquement les transactions. De plus, étant donné que c'est notre source pour faire fonctionner notre variant, il serait préférable de mettre en place un réplicat master-master sur cette base de données. De la sorte, nous restons fortement consistant, et nous avons une sécurité si une des DB tombe KO. Ainsi, nous serons toujours en mesure de fournir des analyses et de proposer des produits aux clients.

Gestion des comptes clients trop légères : Une autre observation concerne la gestion des comptes clients, où les comptes sont actuellement représentés par de simples identifiant clients ("idClient") dans les transactions. Cette approche peut être insuffisante pour offrir une expérience client personnalisée et pour mener des analyses approfondies. Nous devrions envisager d'enrichir la gestion des comptes clients en stockant davantage d'informations pertinentes, ce qui nous permettra de mieux comprendre le comportement des clients et d'adapter nos services en conséquence.

Suppression des offres depuis la BD Analysis : Puisque les transactions du client sont dynamiques, nous avons décidé de refaire les analyses chaque jour pour proposer des produits. Quand les analyses sont faites par Marketing, ce dernier il envoie un message à Offer pour supprimer les anciennes recommandations depuis la BD, ensuite Marketing il envoie les nouvelles analyses. Entre ces deux opérations, le client ne voit pas d'offre. Ce qu'on compte le travailler dans la deuxième étape.

6 Points forts de l'architecture

Peu d'acteurs : La simplicité du système est un atout majeur. Le fait qu'il y ait peu d'acteurs impliqués facilite la gestion des flux d'informations et réduit la complexité de la communication entre les différents composants. Cela devrait contribuer à une mise en œuvre plus fluide et à des performances plus prévisibles.

Séparation de la lecture et de l' écriture : Le pattern CQRS permet de séparer la lecture et l'écriture. C' est principalement la raison pour laquelle nous avons décidé de faire de l'événementiel avec kafka. Ce pattern est exploité lors de la création d'une transaction. Elle est sauvegardée dans la DB du service Transaction, et un événement est diffusé dans le bus. Marketing consomme cette événement pour le sauvegarder dans sa DB ainsi que le service Statistique. Marketing va se servir de ces transactions pour créer des offres appropriés et Statistique va fournir des informations de consommation client au banquier. Mais lorsque Marketing va produire des offres, il va à son tour produire des événements. Ces évènements seront consommés par le service Statistique et par le service Offers. Statistique consomme ces événements pour donner des informations sur les offres proposées, au banquier, et le service Offers consomme ces événements pour donner au client des offres lorsqu'il se connecte sur son compte. Le banquier a aussi accès aux offres proposées grâce au service Statistique, donc il est aussi en mesure de proposer au client des offres adaptées à leur profil. Kafka est la source de vérité de ce pattern. Si l'un des services consommateurs tombe KO, il pourra consommer ce qu'il a loupé lors de sa prochaine connexion.

MarketingAnalysis efficace : L'efficacité de la fonction MarketingAnalysis est cruciale. Cela signifie que nous pouvons fournir des informations pertinentes et exploitables pour orienter nos stratégies marketing. Une analyse de haute qualité devrait nous permettre de prendre des décisions plus éclairées pour satisfaire nos clients et améliorer notre offre.

Marketing facilement changeable Notre service marketing est une brique facilement interchangeable. l'intérêt d'avoir isolé l'analyse marketing dans notre architecture nous permet de pouvoir à notre guise et selon les retour clients modeler, voir changer l'algorithme rapidement. A l'instar d'application comme TikTok ou Instagram ou l'algorithme de recommandation est centrale à l'expérience utilisateur, notre variante à termes nous pousseras à constamment à revoir notre système de recommandation pour l'affiner.

Marketing et Statistique : Le cœur de notre variant sont ces services. Nous avons suivit la loi de Conway pour faire nos micro-services. Ainsi, nous avons une équipe IA dans Marketing pour faire des offres et une équipe de statisticiens pour produire des statistiques. Nous sommes donc en mesure de créer plusieurs micro-services correspondant à des algos différents et déployant le service souhaité. Les équipes d'IA et de Statistique peuvent expérimenter plusieurs algo en créant des micro services, et nous pouvons les déployer quand nous voulons. Ces services gardent le même id kafka donc quand ils sont déployés, il consomme les événements qu'ils ont manqué.

User story bien représentée : La représentation précise des user stories (scénarios utilisateur) est un gage de réussite. Cela garantit que nous comprenons les besoins et les attentes de nos utilisateurs finaux. Une bonne représentation des user stories devrait nous aider à concevoir des fonctionnalités qui répondent à leurs besoins de manière efficace.

7 Technologies utilisées

- **Composants** : Tous les composants sont implémentés en Spring boot.
- **Bus Kafka** : Envoie des transactions au services Marketing et Stats.
- **Load Balancing** : Nginx
- **Gateway** : Spring Cloud
- **Fronts** : React js

8 Auto Evaluation du travail et implication des membres d'équipes

8.1 Auto-Evaluation

Dans cette première partie du projet, nous avons essayé de construire un POC qui répond aux besoins de notre variante. Nous avons mis en place des choix pour avoir une architecture résiliente en introduisant de la scalabilité et de la réparation des charges, nous avons effectué des tests de performance avec plusieurs utilisateurs en parallèle, les résultats ont prouvé que notre système est capable de résister à une charge importante.

Nos stratégies de Marketing sont basiques jusqu'à maintenant puisqu'on pas des modèles d'apprentissage (Machine Learning). L'ajout des analyses basées sur un modèle l'apprentissage pourrait nous aider à avoir de bons résultats au niveau des recommandations.

Au niveau des méthodes Agile, nous somme basés sur la méthode Scrum en ajoutant des user stories pour chaque Milestone. On a utilisé l'outil GitHub pour créer les user stories. Chaque semaine avait un objectif spécifique.

8.2 Implication des membre d'équipe

Noms	Points
Roméo David Amedomey	100
Hamza Zoubair	100
Théophile Yvars	100
Jules Prince	100