Document d'architecture

NewBank - V8 Cashback

Team A

(comme argent)

Antoine BUQUET

-

Benoit GAUDET

-

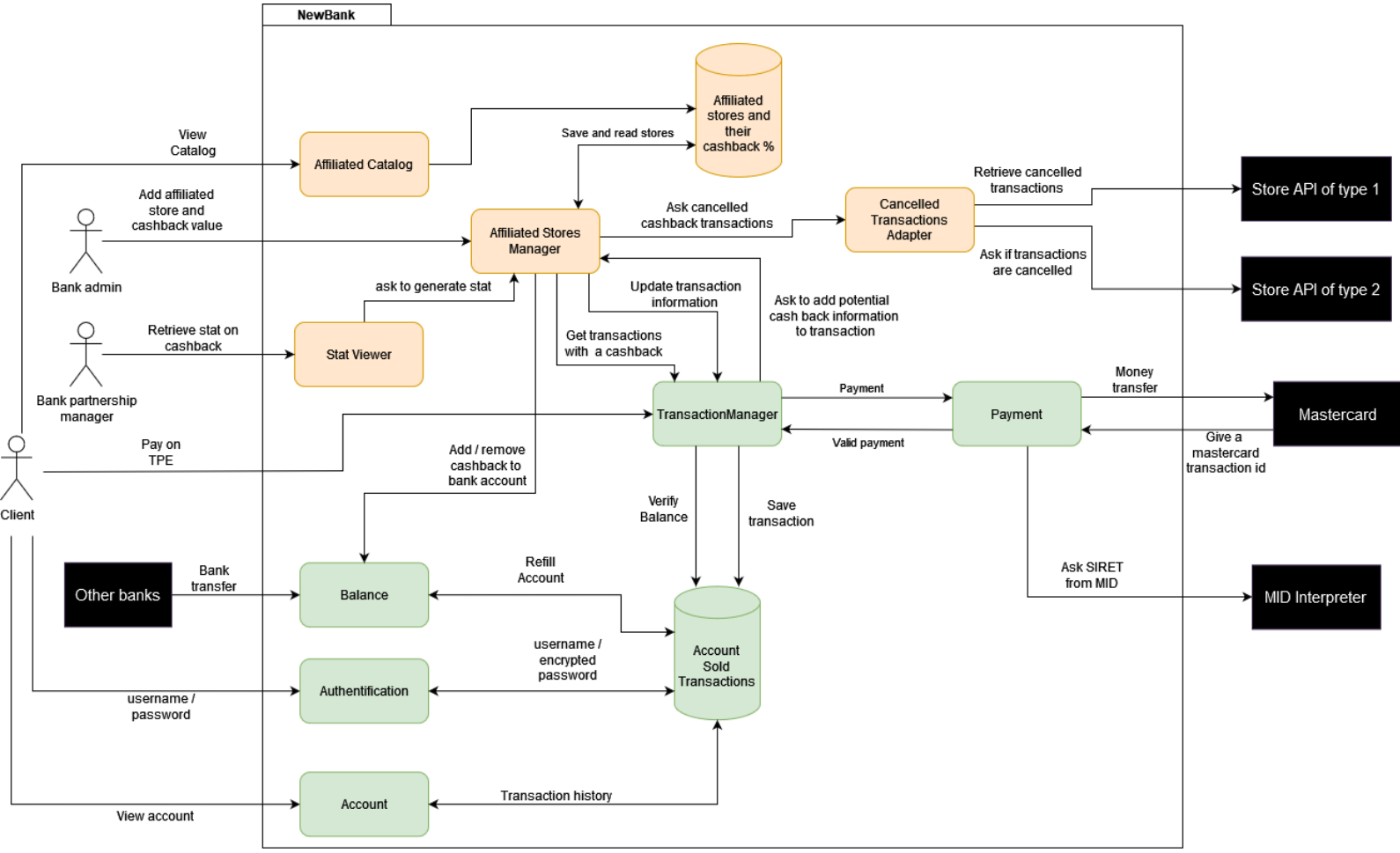
Ayoub IMAMI

-

Mourad KARRAKCHOU

I. État de l'architecture actuel

1. Diagramme d'architecture



Architecture composée de deux services :

- Bank Service
- Cashback Service

2. Justifications

L'architecture de notre banque est basée sur une architecture en services. Le premier service, représenté en vert dans le diagramme, aura pour objectif de gérer la partie des transactions et des comptes utilisateur. Le deuxième service, représenté en orange dans le diagramme, va quant à lui gérer la partie sur le cashback et la connexion avec les services fournis par les banques partenaires. Cela permet à la banque d'être évolutive et résiliente.

Nous avons choisi d'implémenter notre application de cette manière en séparant le côté "banque normale" de la partie cashback, car nous estimons qu'il s'agit simplement d'une extension supplémentaire venant se greffer au fonctionnement d'une banque normale.

De plus, il y a deux bases de données dans l'application, une par service, l'une pour permettre le stockage des données sur les magasins affiliés, avec leur offre de cashback, et l'autre pour le stockage des utilisateurs avec leur solde et leurs transactions. Cette séparation permet d'isoler les données métier relatives à leurs services et évite par ailleurs une centralisation du stockage qui pourrait être problématique si l'unique base de données venait à cesser de fonctionner. Dans le cas présent, si la base de données du service de cashback n'était plus accessible, alors l'utilisateur pourrait tout de même effectuer une transaction.

3. Descriptions des composants du diagramme

- **Balance** : ce composant permet aux utilisateurs de recharger leur compte bancaire pour effectuer des achats avec leur carte NewBank. Pour ce faire, il est accédé par les banques externes pour transférer les fonds.

- **Account** : ce composant est responsable de la gestion des comptes utilisateur. C'est par ce composant également que l'utilisateur peut consulter son historique de transactions ainsi que son solde et ses informations personnelles.

- **Transaction Manager** : ce composant est responsable de déclencher le paiement une fois des vérifications sur le solde ou la carte bancaire utilisée par exemple. Une fois le paiement effectué, il est responsable de transmettre l'information au service de cashback afin de récupérer le potentiel cashback générée par cette transaction. Cependant, ce n'est pas lui qui crédite le cashback sur le compte bancaire de l'utilisateur, il ne fait que récupérer l'information pour la sauvegarder dans la transaction créée au début. Enfin, lors de la vérification mensuelle des transactions annulée mais ayant produit du cashback, c'est à lui de mettre à jour les transactions que lui indique le service de cashback.

- **Payment** : il s'agit de notre système interne de paiement qui va devoir communiquer avec de nombreux services externes d'afin d'effectuer correctement la transaction.

- **Affiliated Stores Manager** : ce composant est responsable de la gestion des comptes magasins affiliée à notre banque et de la gestion des cashback offerts par les magasins affiliés. C'est lui qui déclenche chaque mois une vérification auprès des magasins partenaire dans le but de vérifier si une transaction ayant généré du cashback a été annulé par l'utilisateur. Il est également le seul responsable pour l'ajout et le retrait de cashback sur les comptes bancaires des utilisateurs.

- **Affiliated Catalog** : ce composant est responsable de fournir l'accès aux informations concernant tous les magasins affiliés, ainsi que leur taux de cashback.

- **Stat Viewer** : ce composant est responsable de fournir des statistiques sur les cashback au responsable des partenariats de NewBank à des fins de monitorage. Un exemple de statistique est disponible en annexe *(cf. Figure 1)*

- **Authentification :** ce composant est responsable de l'authentification des utilisateurs. Il est présent sur le schéma, mais il n'était pas la priorité en termes d'implémentation dans notre application, il est donc extrêmement simplifié.

- **Cancelled Transactions Adapter** : ce composant a pour but de s'adapter aux différentes API en fonction du type du magasin afin de récupérer l'information des transactions cashback ayant été annulée ou remboursée. Dans le cas d'un magasin de type 1, il récupère auprès du magasin la liste des transactions annulé le dernier mois et les compare avec toutes les transactions ayant généré du cashback au cours des 30 derniers jours. Dans le cas d'une API d'un magasin de type 2, il transmet chacune des transactions et obtient une réponse booléenne indiquant si la transaction a été annulée ou remboursée.

4. Descriptions des services externes

Nous avons différents services externes :

- ***external-bank-mock-service*** : il s'agit de la potentielle banque secondaire de l'utilisateur, il s'en sert pour faire des virements entre cette dernière et NewBank.

Nous avons pensé au fait qu'un article pouvait être rendu et le client remboursé. Il faut prendre cela en compte sur le cashback et retirer son montant du compte, sinon des clients pourrait abuser d'achat et de retour d'articles en conservant le cashback.

Pour cela, chaque mois, nous demandons aux magasins partenaire la liste des retours d'articles.

- ***external-carrefour-mock-service*** : il s'agit d'un simple mock permettant d'obtenir la liste des retours d'article.
- ***external-decathlon-mock-service*** : de même, mais la logique est différente, on donne une liste d'article et le service nous renvoie les éléments de cette liste qui ont été retournés au magasin.

Nous avons 2 exemples de logiques différentes, pour illustrer le fait que les magasins ne communique pas forcément de la même façon.

Afin d'implémenter un modèle proche de la réalité, nous avons également implémenté ces services externes. D'après nos recherches, lorsqu'un paiement est fait, le TPE du commerçant envoi son identifiant MID à la banque, qui utilise un ***service externe*** pour obtenir le numéro de SIRET du commerçant. Après traitement de la transaction, la banque retire le montant du compte du client, qu'il redirige vers un autre ***service externe*** afin que le commerçant soit payé.

- ***external-mid-interpreter-mock-service*** : il permet d'obtenir le numéro de SIRET d'un commerçant à partir de son identifiant MID.
- ***external-mastercard-mock-service*** : il permet à la banque de rediriger l'argent du client vers le commerçant. Il renvoie à notre application le numéro de la transaction mastercard que nous pourrons utiliser pour reconnaître la transaction dans le cas d'un remboursement client.

5. Fonctionnalités couvertes par notre PoC

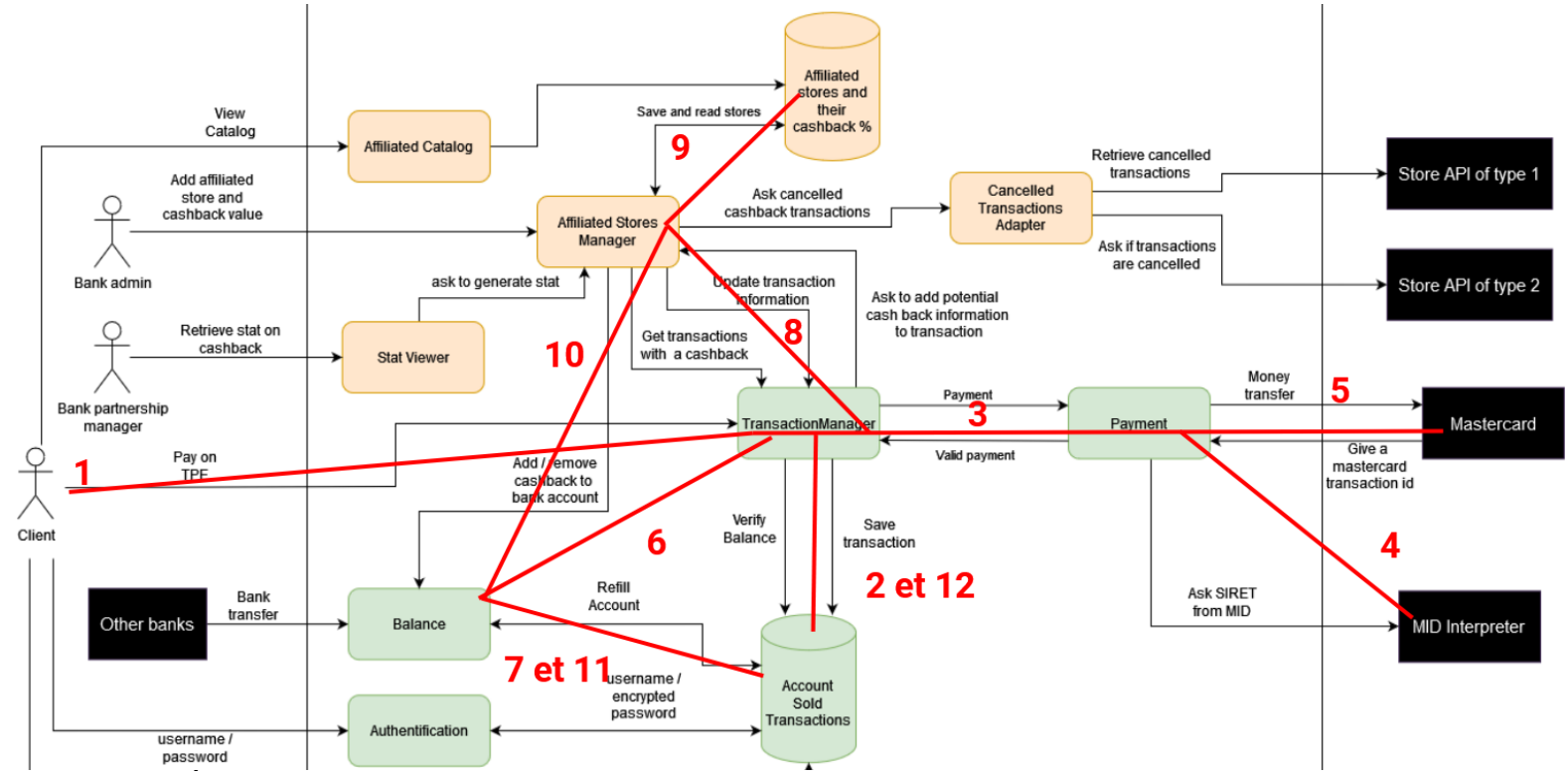
Notre architecture nous permet à l'heure d'aujourd'hui d'effectuer les différentes fonctionnalités suivantes :

- Pour l'utilisateur :
  - Créer un compte bancaire
  - Ajouter des fonds depuis une banque externe
  - Effectuer un paiement normal
  - Effectuer un paiement avec cashback
  - Visualiser des informations concernant son compte (solde, historique des transactions…)

- Pour la banque :
  - Enregistrer un magasin partenaire
  - Générer des statistiques sur les cashback pour le manager des partenariats
  - Retirer le cashback des utilisateurs ayant annulé ou remboursé une transaction que leur en avait octroyé

6. Scénario principal

Le scénario principal de notre application est celui d'un paiement auprès d'un magasin partenaire, ce qui aura pour effet de générer un cashback offert à l'utilisateur. Voici le chemin parcouru dans notre architecture lors de ce scénario :



Explication pas à pas :

1 : Le client fait un paiement sur un TPE
2 : On vérifie le solde de l'utilisateur
3 : On commence le paiement
4 : On contacte le service externe MID Interpreter pour obtenir le SIRET du magasin à l'aide du MID obtenu grâce au TPE
5 : On redirige l'argent en faisant appel au service externe Mastercard
6 : On débite le solde du compte bancaire de l'utilisateur
7 : On sauvegarde le nouvel état de son compte bancaire
8 : On demande au service de cashback le montant du cashback de la transaction effectuée
9 : Pour cela, le service de cashback recherche s'il dispose du numéro de SIRET (obtenu avec MID Interpreter) dans sa base de données des magasins partenaires.
10 : Quand le magasin est trouvé, la valeur du cashback est calculée et on incrémente le solde de l'utilisateur
11 : On sauvegarde le nouvel état de son compte bancaire
12 : On sauvegarde la transaction avec l'information du cashback renvoyé par le service de cashback

7. Forces et faiblesses

Comme évoqué précédemment, notre architecture permet de séparer la partie liée à la spécificité de notre application qui est le cashback du fonctionnement de la banque. Cette architecture avec deux services distincts permet au service de cashback de se greffer sur le système de paiement lorsque cela est nécessaire. De plus, lorsqu'il n'y a pas de cashback, la transaction s'effectue avec un seul service. Cependant, depuis le Cashback service, nous avons besoin de récupérer des informations depuis le Bank service pour gérer la suppression des cashbacks dans le cas d'un remboursement client. Pour cela, il pourrait être envisageable de copier certaines données afin de ne pas devoir faire de nombreux appels à Bank service

8. Auto-évaluation

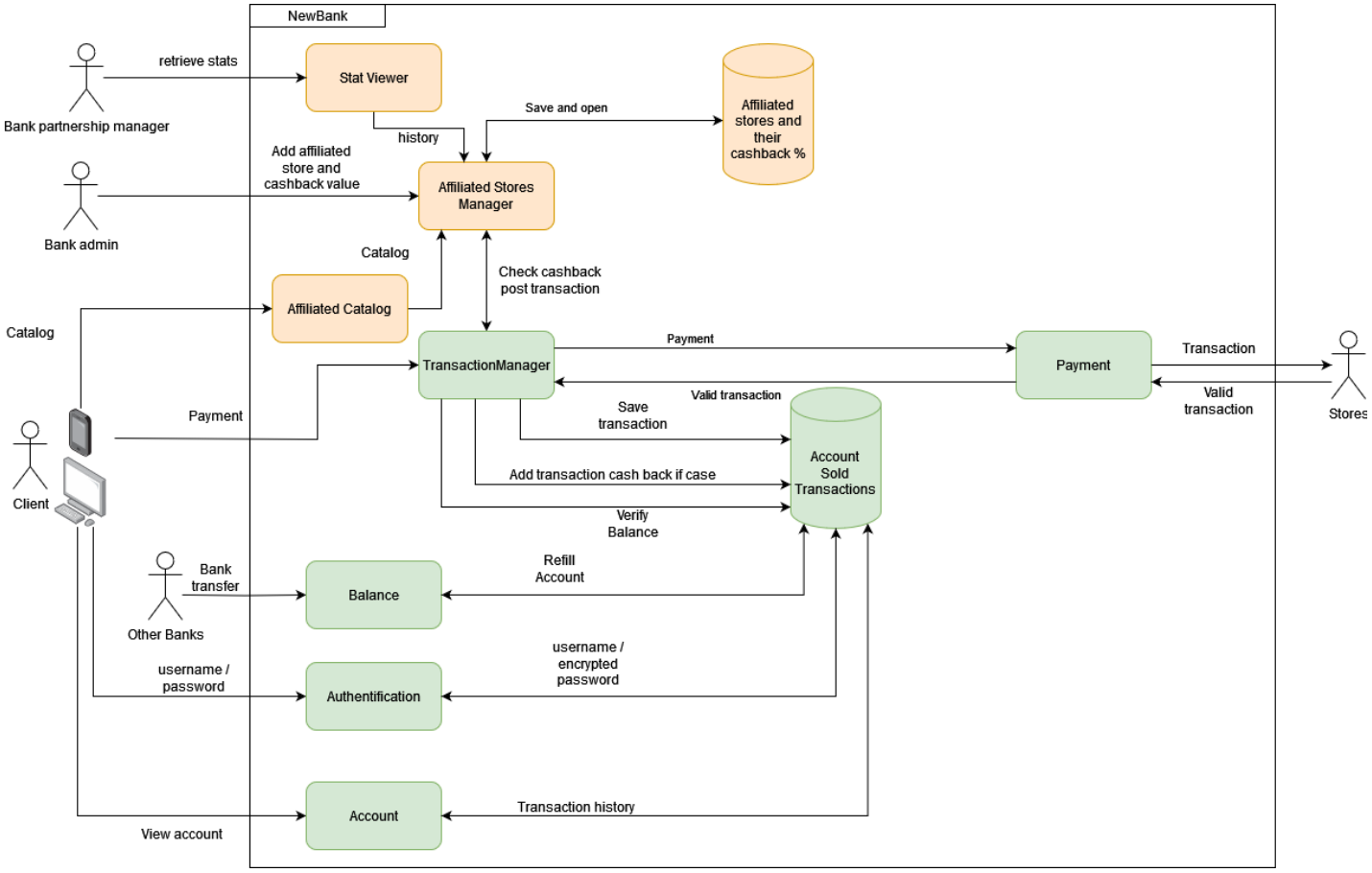
Nous trouvons que nous avons bien pris en compte les scénarios imposés par notre variante, à savoir le CashBack. De plus, nous avons voulu nous rapprocher le plus possible d'un modèle réel. Enfin, nous avons implémenté plusieurs fonctionnalités, les principales étant liées à notre sujet.

Toutefois, il y a des pistes d'améliorations. Par exemple, nous pourrions stocker une transaction avec son CashBack du côté du service de cashback afin de ne pas demander en permanence au service de la banque lorsqu'il vérifie si des paiements ont été annulés. S'il est annulé, alors seulement le service de cashback demandera au service de la banque d'enlever le cashback de la transaction, puis on le supprime de la base de données du service de cashback.

Membre	**Implication**
Ayoub IMAMI	100
Benoit GAUDET	100
Antoine BUQUET	100
Mourad KARRAKCHOU	100

II. Historique de l'architecture

I. 20 octobre 2023

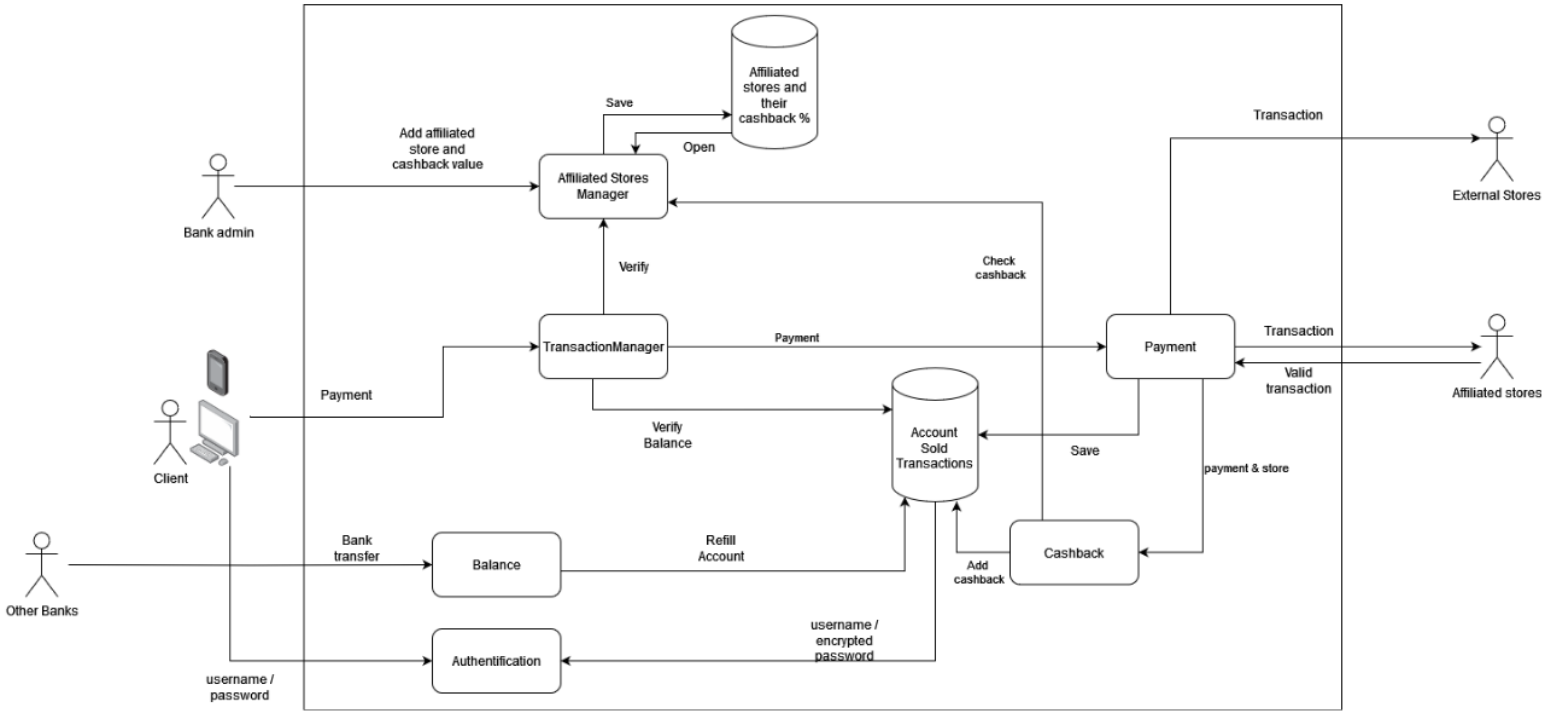


À cette date, notre application manquait cruellement de réalisme en ne faisant pratiquement aucun appel à des services externes. En effet, la transaction était directement transmise au magasin qui validait alors la transaction. Dans la réalité, le fonctionnement d'un paiement est bien plus complexe, en impliquant plusieurs acteurs externes pour effectuer le transfert des fonds.

L'application était donc fonctionnelle, mais certaines logiques n'étaient pas cohérentes. C'était le cas notamment avec le composant Transaction Manager qui avait la responsabilité d'ajouter le cashback dans le compte bancaire de l'utilisateur alors que théoriquement, il devrait s'agir d'une action effectuée depuis le service cashback.

Enfin, nous n'avions pas encore pensé à la fonctionnalité de vérification des transactions qui avaient du cashback mais qui ont été annulées ou remboursées par le magasin partenaire. Toute la partie avec les mocks simulant des API distante différentes afin de récupérer cette information n'était donc pas encore présente dans notre architecture.

II. 1 octobre 2023



Il s'agit du premier diagramme d'architecture élaboré dans le cadre de ce projet. Les briques les plus importantes de notre diagramme d'architecture actuel y sont déjà présentes telles que Affiliated Stores Manager, Transaction Manager ou bien Payment.

III. Annexes


```
{
  "generalStat": {
    "siret": "Overall",
    "numberOfTransactions": 8,
    "amountSpent": 641,
    "cashBackReturned": 106
  },
  "statsByStore": [
    {
      "siret": "12345678900010",
      "numberOfTransactions": 5,
      "amountSpent": 419,
      "cashBackReturned": 83.8
    },
    {
      "siret": "98765432100020",
      "numberOfTransactions": 3,
      "amountSpent": 222,
      "cashBackReturned": 22.200000000000003
    }
  ]
}
```


Figure 1 : Statistique de cashback disponible pour le responsable des partenariats