NewBank : Merchant webapp

Equipe B

Yahiaoui Imène - Gazzeh Sourour - Ben aissa Nadim

Al Achkar Badr

Sommaire

Sommaire..... 2

Mises à jour de l'incrément :.....3

Introduction.....4

Projet..... 4

Utilisateurs.....4

Risques Possibles..... 4

Cas d'Utilisation.....5

Hypothèses..... 8

Proposition d'architecture du système :.....9

Choix technologiques :..... 17

Limitation des risques identifié..... 18

SDK..... 19

Mises à jour de l'incrément :

Cet incrément contient des changements importants à l'architecture et quelques ajouts à celle-ci, ci-dessous une liste de ce qui est mis à jour :

- De nouvelles hypothèses, sur lesquelles nos réflexions s'appuyaient implicitement, ont été ajoutées.
- La base de données en mémoire qui avait été ajoutée pour "mettre en cache" les transactions a été supprimée, pour revenir à une base de données persistante sur disque, car les "cache-misses" qui avaient été négligés à l'origine se sont révélés être une cause d'incohérence.
- Le service "PaymentGateway" a été scindé une nouvelle fois, en séparant "BusinessIntegration" dans un microservice qui gère l'adhésion à notre système et en ne gardant que les interfaces "paiement" de base dans la passerelle de paiement.
- Suite à vos retours, le traitement des frais a fait l'objet d'un ajout d'hypothèses solides et de changements pour l'incorporer, car il nous a semblé un peu confus au début. Parallèlement au processus de "PaymentSettlement", qui a fait l'objet d'une mise au point plus fine.
- Un risque majeur a été identifié dans notre gestion des paiements : l'idempotence non traitée, qui est nécessaire en cas de défaillance du côté client (sdk). Un protocole d'échange a été conçu et mis en œuvre.

Les changements par rapport aux revendications précédentes sont écrits en bleu.
De nouvelles sections ont été ajoutées : "Protocole de paiement" pour aborder le protocole, "Scénario MVP" pour résumer le tout.

Introduction

Le présent rapport a pour but de mettre en lumière le sujet qui nous a été assigné, puis d'aborder la portée, les utilisateurs, les risques et les cas d'utilisation associés, l'architecture et les choix architecturaux et plus globalement tout ce qu'on a réalisé jusqu'à ce stade.

Projet

La portée de ce projet englobe le développement et d'une application spécialement conçue pour les commerçants. Il inclut la conception d'un kit de développement logiciel (SDK) pour faciliter les paiements par carte de débit et de crédit sur les sites web des commerçants. De plus, elle couvre la gestion des frais de transaction pour les transactions en ligne sans numéraire au sein de notre système bancaire.

Utilisateurs

1. Utilisateurs Principaux :
    - Commerçants : Ceux qui possèdent et exploitent des entreprises et qui utilisent l'application web pour gérer les transactions.
    - Clients : Individus effectuant des achats auprès des commerçants en utilisant des cartes de débit/crédit.
2. Utilisateurs Secondaires :
    - Administrateurs de la Banque : Responsables de la supervision du système bancaire en ligne sans numéraire.
    - Administrateurs Système : superviser la suppression de comptes et de cartes, de gérer les données sensibles et d'assurer la sécurité en cas de perte ou de vol.
    - Credit Card Network : ce réseau va nous solliciter afin d'autoriser une transaction chez nous (NewBank), notre système communiquera avec lui afin de lui demander de créer des cartes virtuelles.
    - Banques externes: Les banques externes nous sollicitent pour les transferts des fonds.

Risques Possibles

- Préoccupations en matière de sécurité : Risques liés à la gestion des transactions financières. Possibilité de violation de la sécurité des données sensibles des utilisateurs, ce qui pourrait entraîner des conséquences graves pour la confidentialité et la confiance des clients.
- Intégration : Difficultés pour intégrer de manière transparente le SDK dans divers sites web de commerçants.

- Scalabilité : S'assurer que le système peut gérer un grand nombre de clients (jusqu'à 1 million) sans problèmes de performance.
- Haute Disponibilité : De même, soutenir la gestion en temps réel d'un grand nombre de transactions.
- Extensibilité : Comme notre système interne sera étendu en premier lieu par un sdk fourni par nous.
- Robustesse : Comme le kit de développement est fourni par nous et permet l'utilisation de nos services par de nombreuses entreprises, c'est à nous de garantir la robustesse contre les entrées et les essais erronés ***et prévenir les comportements inattendus tels que les doubles paiements ou les paiements incomplets,*** ainsi que les tentatives d'effondrement de notre service.

Cas d'Utilisation

- Enregistrement et Intégration des Commerçants :

Description : Un commerçant s'inscrit sur l'application web, fournissant les détails nécessaires à l'intégration au sdk.

- Gestion des Comptes Clients :

Description : Les clients peuvent créer et gérer leurs comptes, y compris les comptes d'épargne. Ils ont la capacité d'effectuer des opérations telles que l'ouverture de nouveaux comptes, la consultation des comptes existants.

- Consultation de Solde et Historique des Transactions Client :

Description : Les clients peuvent consulter le solde et l'historique des transactions de leur compte. Cette fonctionnalité permet d'obtenir des informations détaillées sur les mouvements associés au compte client.

- Virement entre Comptes :

Description : Les clients ont la possibilité de réaliser des virements entre leur compte bancaire et le compte associé à la carte de notre service.

- Traitement des Transactions avec Cartes de Débit/Crédit :

Description : Les commerçants initient des transactions en utilisant les détails des cartes de débit/crédit, qui sont traités de manière sécurisée par le système.

- Gestion des Frais de Transaction :

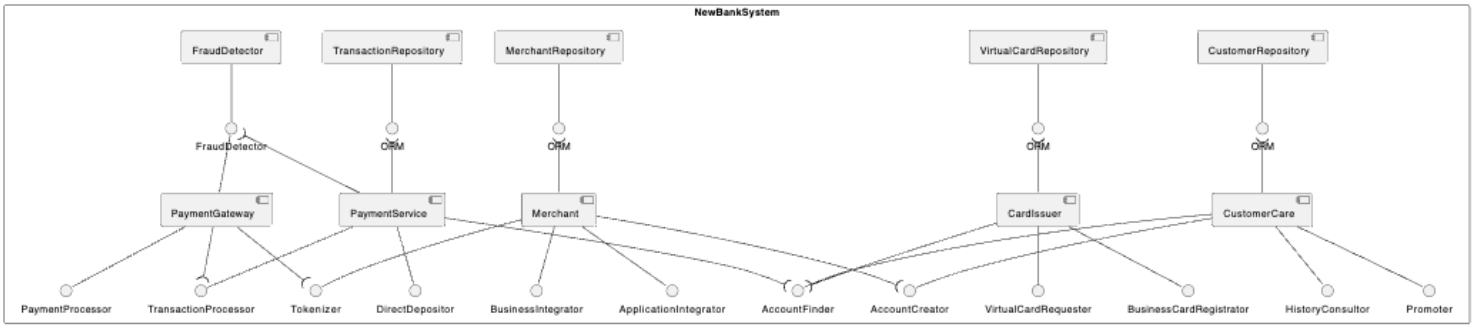
Description : Le système calcule et gère les frais de transaction associés à chaque transaction, assurant une facturation précise.

Diagramme de Composants :

Lien pour la version svg plus claire : diagramme de composants

Pour démarrer notre conception architecturale, nous avons commencé par un diagramme de composants dans lequel nous avons essayé d'identifier les principaux

composants métier qui entreront en jeu. Nous considérons ces composants comme les briques de notre compréhension de la logique métier que nous devrons mettre en œuvre.
C'est donc sur cette base que nos décisions en matière de conception architecturale émergeront.



Chaque composant est conçu pour jouer un rôle spécifique :

- CustomerCare : Ce composant gérera toute la logique liée à l'intégration des clients dans notre banque sans numéraire, de la création d'un compte à l'affichage de l'épargne et de l'historique des transactions.

- CardIssuer : Ce composant contiendra la logique d'émission de cartes virtuelles par la banque à ses clients, et permettra aux entreprises d'enregistrer différents types de cartes virtuelles qui pourraient leur servir à fidéliser leurs clients, par exemple des cartes-cadeaux, des cartes de restaurant, etc.

- Merchant : Ce composant assurera l'intégration des entreprises dans notre banque sans numéraire. Les entreprises qui souhaitent utiliser notre sdk, par exemple, devront avoir leur application enregistrée dans notre système. De même, les entreprises disposent d'une panoplie d'options diverses en ce qui concerne leurs comptes.

- FraudDetector : Comme son nom l'indique, ce composant sera chargé de détecter les activités suspectes et les tentatives de fraude. Ce composant sera utilisé par d'autres composants dans divers scénarios.

- PaymentService : Ce composant gère la logique du service transactionnel de base fourni par la banque. Il traite les demandes de paiement, qu'il s'agisse d'un débit ou d'un crédit d'un compte, gère l'autorisation et la logique de base des transactions et Il utilise le composant de détection des fraudes FraudDetector dans son processus..

- PaymentGateway : Cette composante est la partie exceptionnelle de notre système. C'est la variante sur laquelle nous allons travailler et elle gère toute la logique du traitement des paiements effectués à l'aide de notre sdk. Il sera chargé de contacter le réseau CreditCardNetwork pour la validation des cartes, de déduire les frais de transaction et d'utiliser le mécanisme de tokenisation fourni lors de l'intégration du merchant pour authentifier et crypter ce trafic. Il utilisera également le détecteur de fraude.

- Les repositories représentent des composants responsables de la gestion et du stockage des données.

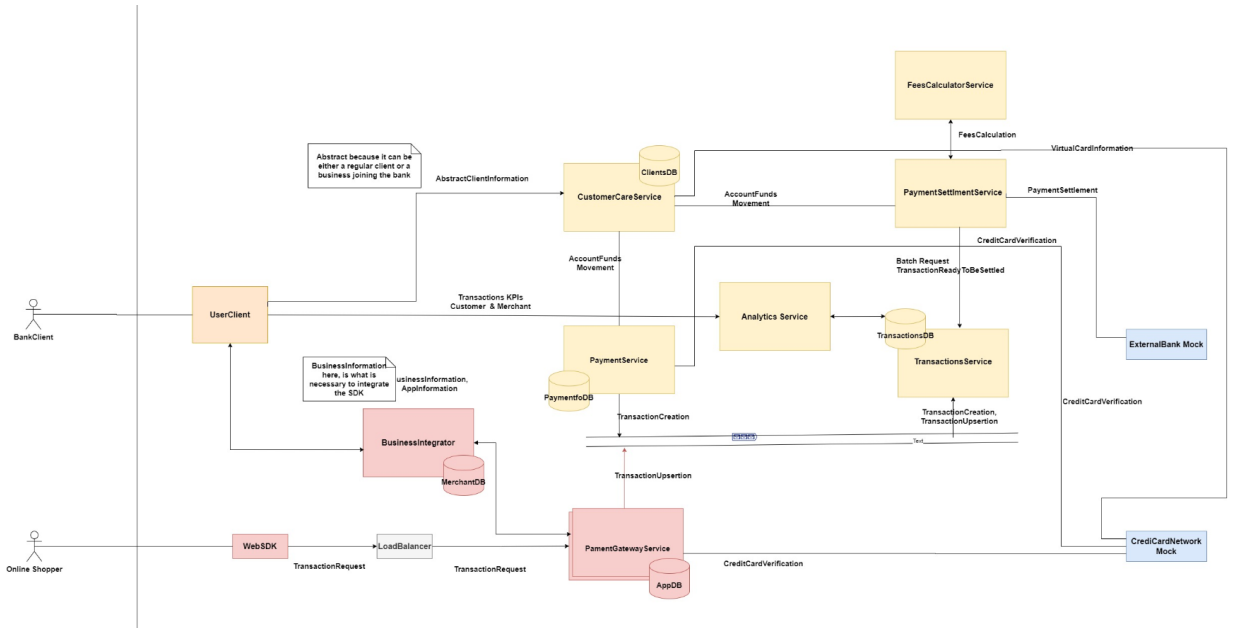
Hypothèses

- Les cartes virtuelles sont générées par les CCN, et par conséquent c'est le CCN qui se chargera de vérifier si un client possède une carte assignée à notre système NewBank pour nous appeler chez nous.
- Le SDK effectue le cryptage des données de la carte, la vérification de la validité du numéro de carte saisi, et la demande de paiement avec le token de l'application.
- La passerelle de paiement est chargée d'autoriser la transaction en effectuant les vérifications nécessaires sur l'appartenance du commerçant au système et en demandant l'autorisation externe le cas échéant pour les cartes de crédits des clients du marchand.
- Pour un paiement réussi, la passerelle de paiement effectue deux appels : un premier appel au CCN pour confirmer l'éligibilité et la disponibilité des fonds sur la carte du client, et un second appel directement à la banque du client pour retenir les fonds. Cela fait partie du protocole de paiement.
- Les frais appliqués à une transaction varient et dépendent du type de carte, de notre marge en tant que passerelle et de la marge bancaire du commerçant.
- Les frais sont déduits avant le règlement, et le type de carte (débit ou crédit) n'est pris en considération que par le service FeesCalculator lors de l'application des frais. D'autres services prennent la transaction telle quelle. Par exemple : la passerelle de paiement associe le type de carte à la transaction, mais ne l'utilise pour aucune autre action.
- L'autorisation d'une transaction génère un jeton d'autorisation de la banque cliente récupéré par le CCN et conservé par la passerelle de paiement pour être utilisé pendant les étapes suivantes et plus particulièrement le règlement de fonds.
- Les frais qui doivent nous être versés sont calculés après l'autorisation de la transaction par la passerelle de paiement et avant son règlement par le service de règlement des paiements.
- La demande de règlement des fonds est envoyée avec le jeton d'autorisation à la banque commerçant/acquéreur par le service de règlement des paiements.
- Si le commerçant est un client de notre banque, le service de règlement des paiements effectue lui-même la demande de règlement.
- Aucune demande de règlement n'est effectuée avant que les frais ne soient calculés et joints.
- Le règlement des transactions, qu'elles concernent notre passerelle ou notre banque interne, est effectué périodiquement et non pas toute de suite. Un cronJob trigger le règlement de toutes les transactions effectuées les 12 heures passées.

Proposition d'architecture du système :

Vous trouverez ci-dessous notre diagramme d'architecture qui a été initialement dérivé du diagramme de composants conçu pour un monolithe et puis amélioré au fur et à mesure. Vous trouverez ci-après une explication détaillée des composants architecturaux, de leurs interfaces internes et de leurs interfaces externes.

Lien svg plus clair : [diagramme-archi]



Nous avons choisi une architecture orientée services afin que chaque service de ce découpage représente une vue cohésive et claire du domaine du métier traité et ait une responsabilité claire qui reflète des fonctionnalités recherchées par la banque en ligne. Les fonctionnalités sont une réponse au parallèle que sont les besoins métiers. Notre découpage est axé sur le découpage du domaine lui-même.

Voici une description du rôle et des cas d'utilisation de chaque service ainsi que ses interfaces externes et internes :

Service principaux :

CustomerCareService :
Ce service sera chargé de tout ce qui concerne les relations avec les clients. Il créera des comptes d'utilisateurs, des comptes d'entreprises, gérera les mises à jour directes de leurs fonds et demandera des cartes de crédit

virtuelles pour ces clients à un service externe : le réseau de cartes de crédit (mock).

Composants et Interfaces Internes :

- AccountFinder : Tout ce qui concerne la recherche du compte d'un client est géré par cette interface
- AccountRegistration : En charge de l'intégration d'un client dans le système et de lui fournir son compte bancaire.
- ChequeAccountHandler : il déplace les fonds en cas de dépôt, de transfert ou de débit.
- SavingsAccountHandler : Différemment, il s'occupe de déplacer les fonds de l'épargne.
- VirtualCardRequester : Chargé de demander et d'émettre une carte bancaire virtuelle à un client.
- BusinessAccountHandler : permet de mettre à niveau le compte en business pour permettre d'augmenter la limite de paiement hebdomadaire.
- AccountLimitHandler : permet de réinstaller la limite de paiement chaque semaine

Interface externes :

- **/POST /api/customers**
  request body:
  - firstName : String
  - lastName : String
  - email :String
  - phoneNumber : String
  - Birthdate : String
  - FiscalCountry : String
  - address : String

  response body:
  - id : Long
  - customerProfile: CustomerProfileDTO
  - balance: BigDecimal
  - savingsAccount: SavingAccountDTO
  - creditCard : CreditCardDTO null before the creation of a virtual card)

  /GET /api/customers/:id
  response body:
  - id : Long
  - customerProfile: CustomerProfileDTO
  - balance: BigDecimal

- savingsAccount: SavingAccountDTO
- creditCard : CreditCardDTO null before the creation of a virtual card)

/POST /api/customers/:id/virtualcard
response body:

- id : Long
- customerProfile: CustomerProfileDTO
- balance: BigDecimal
- savingsAccount: SavingAccountDTO
- creditCard : CreditCardDTO
    - cardNumber: String
    - cardHolderName: String
    - expiryDate: String
    - cvv: String

/PUT /api/customers/:id/movetosavings
request body:

- amount: BigDecimal

/PUT /api/customers/:id/funds
request body:

- amount: BigDecimal
- operation : String (withdraw or deposit)

/PUT /api/customers/:id/upgrade

PaymentService :

Ce service est chargé d'autoriser la transaction réelle et la demande de paiement émanant d'une banque ou de notre réseau de cartes de crédit. Ce service effectue les contrôles de fraude nécessaires, par exemple, et les contrôles de fonds, car il demande les informations sur le client au service clientèle avant d'approuver la transaction. Il envoie également l'événement à une queue afin de créer la transaction et de la stocker dans la base de données par le service dédié qui écoute sur la queue.

Composants et Interfaces Internes :

- FundsHandler : application des contrôles nécessaires sur l'historique et le seuil des transactions, ainsi que vérification des fonds du compte.
- FraudDetector : Détecteur de fraude en cas, par exemple, de transactions multiples en même temps.

- TransactionLimitHandler : Dect
- TransactionProcessor : Vérifie l'existence du panier ou de l'IBAN et délivre une autorisation en cas d'approbation. Il envoie ensuite la transaction dans la queue interne.

Interface externes :

/POST /api/payments/process
request body :
- cardholderName : String
- cardNumber : String
- expirationDate : String
- cvv : String
- toAccountIBAN : String
- toAccountBic : String
- amount : BigDecimal

response body :
- success : boolean
- message : String
- authToken : String

/POST /api/payment/checkCreditCard :
request body :
- cardNumber : String
- expirationDate : String
- cvv : String

response body :
- response : boolean
- message : String
- authToken : String

/POST /api/transfer/process
request body :
- fromAccountIban :String
- fromAccountBic : String
- toAccountIban : String
- toAccountBic : String
- amount : bigDecimal

response body :
- success : boolean
- message : String
- authToken : String

TransactionsService :

Ce service s'occupera des insertions et des mises à jour de notre base de données de transactions. Il effectuera beaucoup d'écritures et sera purement utilisé pour nous permettre d'effectuer des autorisations plus rapides dans d'autres services sans se soucier de la latence d'écriture directe et concurrente dans la base de données par ses services (PaymentService et PaymentGateway).

Composants et Interfaces Internes :

- Persister : gère le stockage des transactions, qu'elles soient effectuées par l'intermédiaire du processeur de paiement ou de notre passerelle de paiement.

Interface externes :

Il n'aura pas d'interface externe mais il va plutôt écouter les transactions qui arrivent sur la queue pour manipuler la db des transactions.

Il écoute les transactions normales provenant du PaymentProcessor sur un topic 'internal-transactions', et les transactions en provenance du PaymentGateway sur un topic 'external-transactions'.

PaymentSettlementService :

Ce service s'occupera des mouvements de fonds et du règlement de toute transaction en cours qui a été approuvée. Il recevra les demandes de transfert de fonds de nos comptes clients vers d'autres comptes bancaires, et inversement, il procédera également à la demande de règlement de fonds à partir d'autres comptes bancaires si ce sont nos clients qui recevront les fonds.

Composants et Interfaces Internes :

- TransactionsDétenteur : Cette interface contient la logique de règlement des paiements : elle vérifie s'il s'agit d'un paiement entièrement interne ou externe et, si c'est le cas, rappelle l'organisme externe pour régler la transaction.
- FondsDéducateur : Cette interface est chargée de lancer l'application des mouvements de fonds liés aux comptes respectifs, ainsi que de prendre la réduction qui a été calculée précédemment lors de l'appel de notre calculateur de frais.

Interface externes :

/POST /api/settle

PaymentGatewayService :

Ce service a un rôle principal qui est le traitement des transactions effectuées par le SDK, l'appel au CCN pour l'autorisation de paiement du client et le déclenchement du règlement des fonds après l'autorisation.

Composants et Interfaces Internes :

- TransactionProcessor : Traite les transactions entrantes, délivre l'autorisation et demande une autorisation externe dans le cas où le client du commerçant utilisant notre sdk n'est pas un client de notre banque.
- IRSA : Décrypte les informations de carte de crédit dans le contexte d'une demande de paiement, ainsi que la génération ou la récupération de clés publiques RSA pour une application spécifique.

Interface externes :

/POST /api/gateway/authorize
request body :
- token : string
- amount : BigDecimal
- cryptedCreditCard : String

/GET /api/gateway/applications/{id}/publickey
response body :
- key: string

BusinessIntegratorService :

Ce service a pour rôle d'intégrer une entreprise et son application, de générer le jeton et de permettre l'utilisation de notre sdk en l'utilisant.

- BusinessIntegrator : Intègre le commerçant dans notre système, à condition qu'il nous communique ses coordonnées bancaires.
- BusinessFinder : Gère la recherche des entreprises qui nous ont déjà rejoints.
- ApplicationIntegrator : gère l'intégration de l'application proprement dite en lui associant un apitoken, une clé publique et un commerçant.
- ApplicationFinder : gère la recherche de l'application qui a été précédemment intégrée à notre passerelle.

/POST /api/integration/merchants
request body :
- name : String
- email : String
- bankAccount :
    - bic: String
    - iban: String

response body :
- id : String
- name : String
- email : String
- bankAccount :
    - iban: String
    - bic: String
- application : Application (null before application creation)

/POST /api/integration/applications
request body :
- name : String
- email : String
- url : String
- description : String
- merchantId : String

response body :
- id : Long
- name : String
- email : String
- url : String
- description : String
- apiKey : String
- merchant : MerchantDto

/POST /api/integration/applications/{id}/token
response body :
- token : String

/GET /api/integration/applications/{id}/token
response body :
- token : String

FeesCalculatorService :

Ce service gère le calcul des frais qui doivent être appliqués à un batch de transactions entrant appelé depuis le payment settlement et les rattache avant qu'elle ne soit réglée.

Composants et Interfaces Internes :

- FeesCalculator: gère les frais d'utilisation de notre passerelle et applique un forfait et un ratio en fonction du montant de la transaction.

Interface externes :

/POST /api/feescalculator/calculate

request body : une liste contenant les informations nécessaires des transactions pour le calcul des frais :

- id: String
- sender :
  - IBAN : String
  - BIC : String
- amount : BigDecimal

response body : une liste contenant l'id de la transaction et les frais associés :

- id : String
- fees : BigDecimal

AnalyticsService :

Ce service est responsable d'afficher les statistiques des activités financières des clients et des commerçants sur la plateforme, en fournissant des informations détaillées sur les transactions, les revenus, les dépenses et les soldes.

Composants et Interfaces Internes :

- AnalyticsCustomer: est dédié aux transactions d'un client sur un mois donné. Il permet de fournir des détails sur les revenus et les dépenses journalières, ainsi que le solde mensuel, pendant un mois.
- AnalyticsMerchant :permet d'avoir des statistiques sur les revenus quotidiens des commerçants grâce à l'utilisation du SDK, ainsi que des frais quotidiens associés à leurs activités. De plus, il permet de suivre la variation quotidienne des profits des commerçants.

Interface externes :

/POST /api/analytics/merchant

request body :

- IBAN : String
- BIC : String

response body : une liste contenant :

- date : Date
- totalAmountReceived : BigDecimal
- totalFees : BigDecimal
- percentageProfitVariation : BigDecimal

/POST /api/analytics/customer

request body :

- IBAN : String
- BIC : String

response body :

- monthlyBalance : BigDecimal
- income :
  - date : Date
  - amount : BigDecimal
- expense :
  - date : Date
  - amount : BigDecimal

Mocks :

ExternalBank Mock :

Ce service imitera un service bancaire typique : il autorise les paiements et demandera les règlements de fonds qui sont déclenchés.

Interface externes :

/POST /api/externalbank/authorize

request body :

- fromAccountIBAN : String
- toAccountIBAN : String
- amount : BigDecimal

response body :

- authorized : boolean

/POST /api/externalbank/add

request body :

- iban : String
- amount : BigDecimal

response body :

- settled : boolean

/POST /api/externalbank/deduce

request body :

- iban : String
- amount : BigDecimal

response body :

- settled : boolean

CreditCardNetwork Mock :
Ce service simulera un CCN, vérifiant la validité d'une carte et renvoyant un jeton d'autorisation si le client est autorisé dans le cas d'une transaction approuvée, et émettant des cartes de crédit virtuelles.

Interface externes :

/POST /api/payment/authorize

request body :

- cardNumber : String
- expirationDate : String
- cvv : String

response body :

- response : boolean
- message : String
- authToken : String
- AccountIBAN : String
- AccountBIC : String

Loadbalancer :
Ce composant supplémentaire fonctionne seul et fournit une sorte de proxy inverse pour accéder à la passerelle de paiement. Ce composant est placé devant la passerelle et répartit la charge sur ses différentes instances qui sont redondantes. Il sera chargé de limiter l'utilisation en cas de demandes élevées et étranges de la part d'un certain demandeur. Il réacheminera également les demandes vers une instance opérationnelle s'il détecte que l'autre instance en cours d'exécution a échoué.

Routes traitées :

/POST /api/gateway/authorize

/POST /api/gateway//applications/{id}/publickey

Next steps:Choix technologiques :

- Le développement des services se fera en Java Spring Boot pour deux raisons : la familiarité de la stack technologique pour l'équipe et la robustesse de java (multithreading, strong typing,..).
- La base de données de transactions sera en Cassandra pour permettre un flux d'écriture intensif.
- Les autres bases de données seront en Postgres vu qu’elles seront sollicitées autant en écriture qu'en lecture.
- La technologie de la file d'attente décidée est Kafka vu sa capacité de traiter un débit très haut de messages, dans notre cas cela sera les transactions.
- Le reverse proxy du load balancer est implémenté avec Nginx.

Limitation des risques identifié

En commençant par **l'extensibilité**, nous avons utilisé REST pour que la fonctionnalité de notre passerelle soit facilement exposée par le biais d'un kit de développement logiciel.

L'exposition d'interfaces externes simples, conformes aux normes REST et fournissant des points d'accès à la passerelle basés sur des cas d'utilisation simples, facilite l'extension et l'intégration de SDK à la passerelle.

Pour **l'intégration**, notre sdk est fourni comme une bibliothèque, qui sera publiée sur npm. Il est facile de l'extraire, de l'installer et de l'utiliser directement. Cela limite effectivement les marchands en termes de choix technologique, puisqu'ils sont obligés d'utiliser une pile technologique conforme à notre sdk. Mais c'est une contrainte inévitable.

En ce qui concerne la **robustesse**, notre approche a évolué. D'un point de vue fonctionnel, nous avons mis en place des mécanismes robustes de gestion des erreurs pour traiter à la fois les erreurs intentionnelles et non intentionnelles. Cependant, notre ***préoccupation architecturale principale*** porte désormais sur la garantie d'**idempotence des paiements**. Dans notre système de paiement, il est essentiel d'éviter la multiplication des paiements clients pour une seule transaction. Ce défi est efficacement atténué grâce au protocole de paiement discuté en détail ultérieurement dans cet axe.

Notre composant proxy inverse continue de jouer un rôle essentiel dans notre système, agissant comme gardien limiteur de débit et équilibreur de charge, en se servant de la redondance active-active de notre passerelle de paiement afin d'améliorer **la disponibilité**. La combinaison de ces mesures vise à garantir une haute disponibilité et une robustesse dans notre système.

De plus, il est important de noter que, bien que la partie "Sécurité du SDK" ait déjà été présentée, son objectif spécifique n'a pas été explicitement mentionné ici. Cette partie était consacrée à l'explication des mesures de sécurité mises en place pour mitiger le risque associé avec l'échange des données financiers du client, notamment l'introduction de l'authentification basée sur des jetons, ainsi que le chiffrement des détails des cartes de crédits/débits au moyen d'un algorithme de chiffrement asymétrique. Ces dispositions visaient à renforcer **la sécurité de notre système**.

Sécurité du SDK

Une clé api sera fournie à chaque client de l’SDK. Ce token JWT sera incorporé dans l'en-tête "Authorization" de chaque requête API envoyée au SDK. Si le token s'avère invalide, la gateway ne recevra aucune requête. En revanche, si le token JWT est valide, les transactions se déroulent sans accroc.

Concernant le cryptage des informations de carte, nous avons choisi d'employer l'algorithme RSA. Plus précisément, la gateway génère une paire de clés RSA, à savoir la clé privée et la clé publique. Le SDK utilise la clé publique pour chiffrer les données de carte avant de les transmettre à la passerelle. Ceci garantit une transmission sécurisée des informations sensibles.

Protocole de paiement :

Lors de la simulation de paiements au sein de notre système, nous avons identifié un scénario courant de défaillance qui nécessitait notre attention : les échecs de paiement. Les échecs de paiement peuvent survenir pour diverses raisons, notamment lorsque le SDK (côté marchand) tombe en panne avant de recevoir une confirmation de 'succès', en cas de problèmes de réseau, d'interruptions de notre passerelle, et bien d'autres encore. En réponse à ces défis potentiels, nous avons reconnu la nécessité de rendre la procédure de paiement idempotente afin d'assurer un processus de paiement robuste et fiable.

Pour parvenir à l'idempotence, nous avons conçu un protocole de paiement en deux phases : l'autorisation et la confirmation. Les diagrammes de séquence présentés ci-dessous illustrent à la fois le scénario de paiement standard et un scénario de défaillance potentiel.

Lien vers la version SVG plus claire : SVG

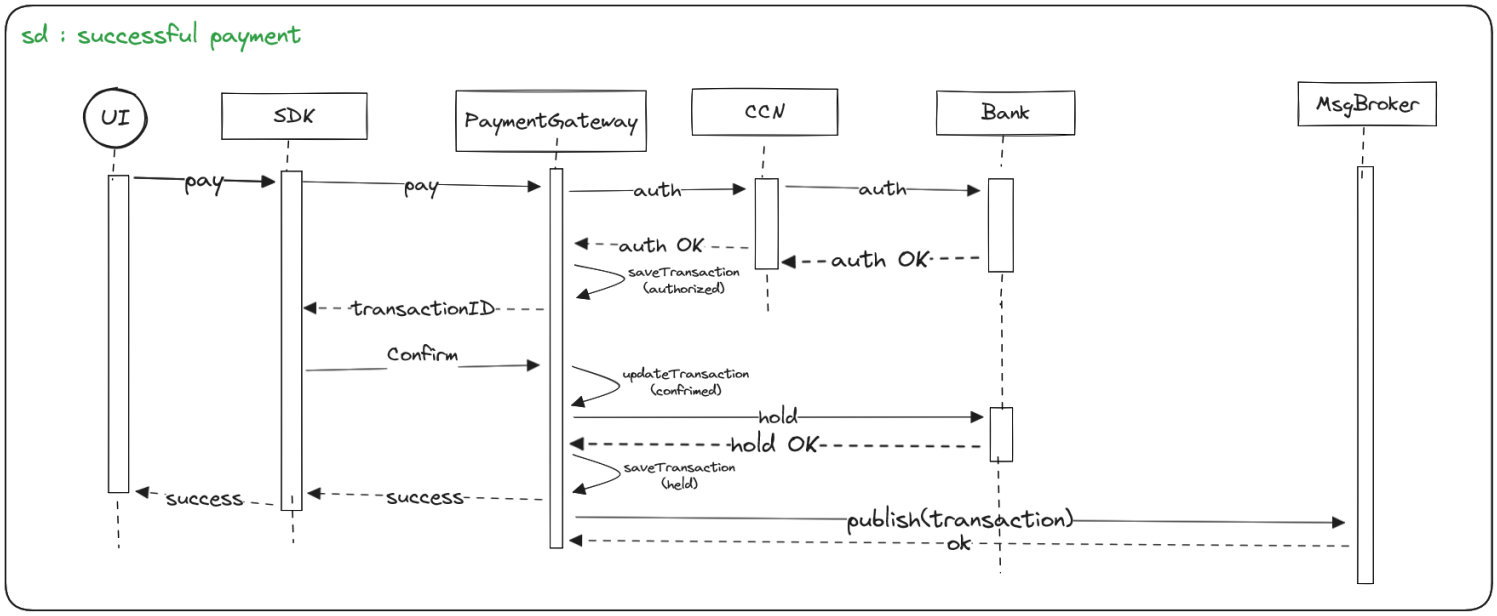


Figure 1 : Diagramme de séquence d'un paiement réussi

Lien vers la version SVG plus claire : SVG

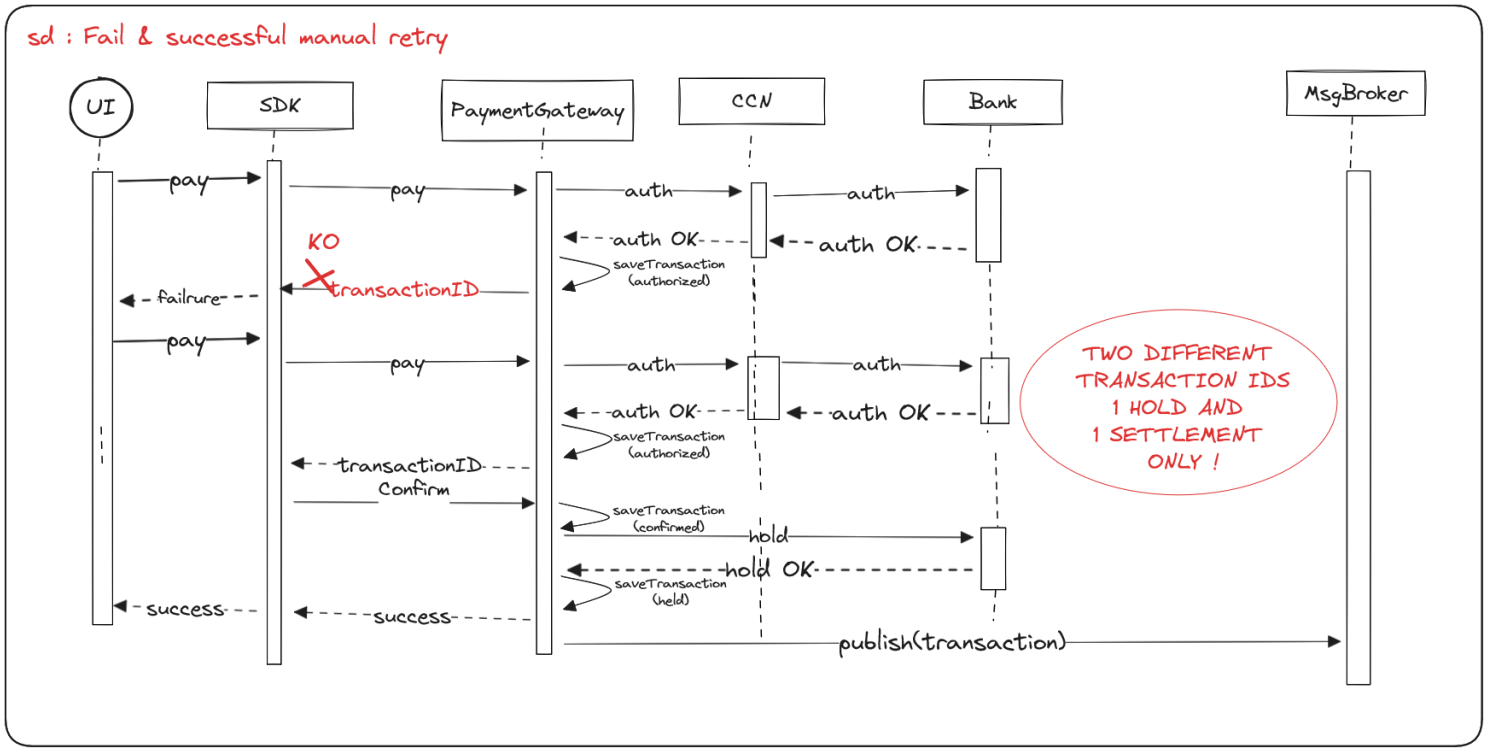


Figure 2 : Diagramme de séquence d'un paiement échoué et un retry réussi

Explications :

Dans la phase d'***Autorisation***, la passerelle traite la demande de paiement du client du marchand et effectue une vérification des fonds pour déterminer la solvabilité du client. Voici comment cela se déroule :

- Le client confirme le paiement et l'envoie via le SDK, qui en abstrait le processus.
- Le SDK initie le paiement en appelant la passerelle de paiement avec les informations de paiement.
- La passerelle de paiement effectue des vérifications de sécurité habituelles sur le jeton associé à la transaction.
- La passerelle de paiement demande l'autorisation de paiement au Réseau de Carte de Crédit (CCN).
- Le CCN contacte la banque émettrice de la carte de crédit pour vérifier la disponibilité des fonds et nous renvoie le jeton d'autorisation.
- Nous générons un identifiant de transaction, marquons la transaction comme 'APPROUVÉE' et la sauvegardons dans notre base de données avant de fournir à l'application SDK l'identifiant de transaction.

À la fin de la phase d'autorisation, aucun fonds ne sont retenus, et la transaction reste non confirmée.

Ensuite, dans la phase de ***'Confirmation'*** :

- Le SDK nous contacte à nouveau pour confirmer la transaction, en précisant l'identifiant de transaction.
- La passerelle de paiement effectue des vérifications de routine sur le jeton et d'autres détails.
- La passerelle de paiement met à jour le statut de la transaction en 'CONFIRMÉE'.
- La passerelle de paiement contacte directement la banque émettrice (la banque du client) pour bloquer les fonds à l'aide du jeton d'autorisation.
- Ensuite, la passerelle de paiement met à jour le statut de la transaction en 'RETENUE'.
- La passerelle de paiement répond au SDK avec une confirmation de succès.
- La passerelle de paiement place la transaction dans le message broker en vue d'une conservation et d'un règlement ultérieurs.

Ce protocole de paiement offre un certain degré d'idempotence en veillant à ce qu'en cas de défaillance, le client puisse réessayer et soit obtenir un nouvel identifiant de transaction ou bien résumer la procédure avec un identification de transaction déjà émis.

Si la défaillance survient après l'émission de l'identifiant de transaction, le marchand qui utilise notre SDK peut soit choisir de stocker et utiliser l'identifiant de transaction pour une nouvelle tentative, soit relancer le processus pour obtenir un nouvel identifiant de transaction. Notre SDK propose deux options pour mettre en œuvre ces procédures : l'utilisation directe de la méthode pour déclencher le protocole complet '.pay()' ou un contrôle à grains fins sur les phases avec les méthodes '.authorize()' et '.confirm()'.

En cas de défaillance survenant après la confirmation de la transaction, et si le marchand choisit de réessayer, notre passerelle peut vérifier le statut de la transaction.

Si le statut est 'RETENUE', la confirmation peut être effectuée directement. Si le statut est seulement 'CONFIRMÉE', le marchand peut réessayer avec le jeton d'autorisation. Cette hypothèse suppose que le jeton d'autorisation est un jeton à usage unique. Si la deuxième tentative de rétention échoue, cela signifie que la rétention précédente a réussi, et la transaction est considérée comme réussie et marquée 'retenue'."

Scénario MVP :

Lien vers la version SVG plus claire : SVG

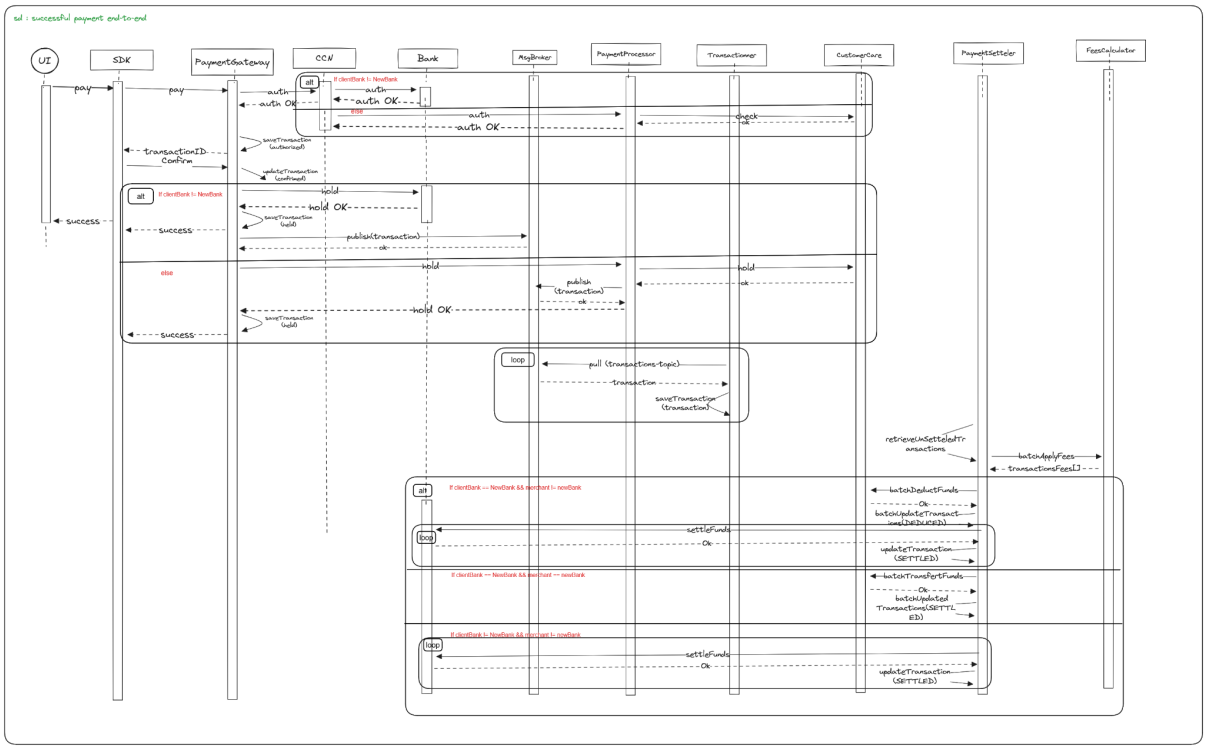


Figure 3 : Diagramme de séquence d'un paiement de bout-en-bout

Explication du scénario :

Pré-conditions : Ce scénario suppose que le commerçant s'est déjà joint à notre plateforme, a obtenu son jeton, installé le kit de développement logiciel (SDK) et l'utilise pour permettre à ses clients de finaliser leurs achats et effectuer des paiements.

Le paiement se déroule en deux phases : la réussite de la transaction et le règlement du paiement. Ces deux phases se déroulent de manière séquentielle mais asynchrone.

Dans la première phase :

1. Lorsque le client choisit de payer, une demande est envoyée par le SDK à la passerelle de paiement, et le protocole de paiement précédemment expliqué est exécuté.

Il est important de noter que le Réseau de Carte de Crédit (CCN) sait s'il doit appeler la banque Newbank (nous) ou la banque fictive (autres banques) en fonction de la carte utilisée par le client pour effectuer ses achats.

De plus, lorsque le client est le nôtre, lors de la phase de rétention des fonds, l'appel est effectué directement à notre processeur de paiement depuis notre passerelle de paiement, qui appelle ensuite le service client, également en interne, pour retenir les fonds.

2. Lorsque le paiement est réussi, une transaction est publiée dans notre file de messages (dans notre cas, il s'agit d'une file Kafka).
3. Un service de traitement de transactions écoutant la file extrait la transaction réussie et la sauvegarde.

Dans la deuxième phase, le règlement du paiement se produit de manière asynchrone, avec l'aide d'une tâche planifiée qui s'exécute périodiquement.

Voici ce qui se passe :

1. Le règleur de paiement récupère les transactions non réglées depuis la base de données (nous sommes conscients que ce n'est pas la meilleure approche, car cela lie le règleur à la structure de la base de données, mais nous avons choisi de le faire pour ne pas surcharger le service de traitement de transactions de manière occasionnelle).
2. Le règleur de paiement demande ensuite au calculateur de frais d'appliquer les frais par lots.
3. Le règleur de paiement reçoit ensuite les frais associés aux transactions et procède au règlement des fonds. À ce stade, trois scénarios se présentent :
    - Si à la fois le client et le commerçant sont les nôtres et possèdent un compte chez Newbank, nous transférons simplement les fonds et marquons ces transactions comme réglées.
    - Si le client est le nôtre mais le commerçant ne l'est pas, nous mettons à jour les fonds de notre côté, puis, en boucle, nous demandons aux banques des commerçants de procéder au règlement et mettons à jour le statut de la transaction.
    - Si ni le client ni le commerçant ne sont les nôtres, nous parcourons simplement les transactions et demandons à la banque fictive de les régler, puis mettons à jour le statut de nos transactions.