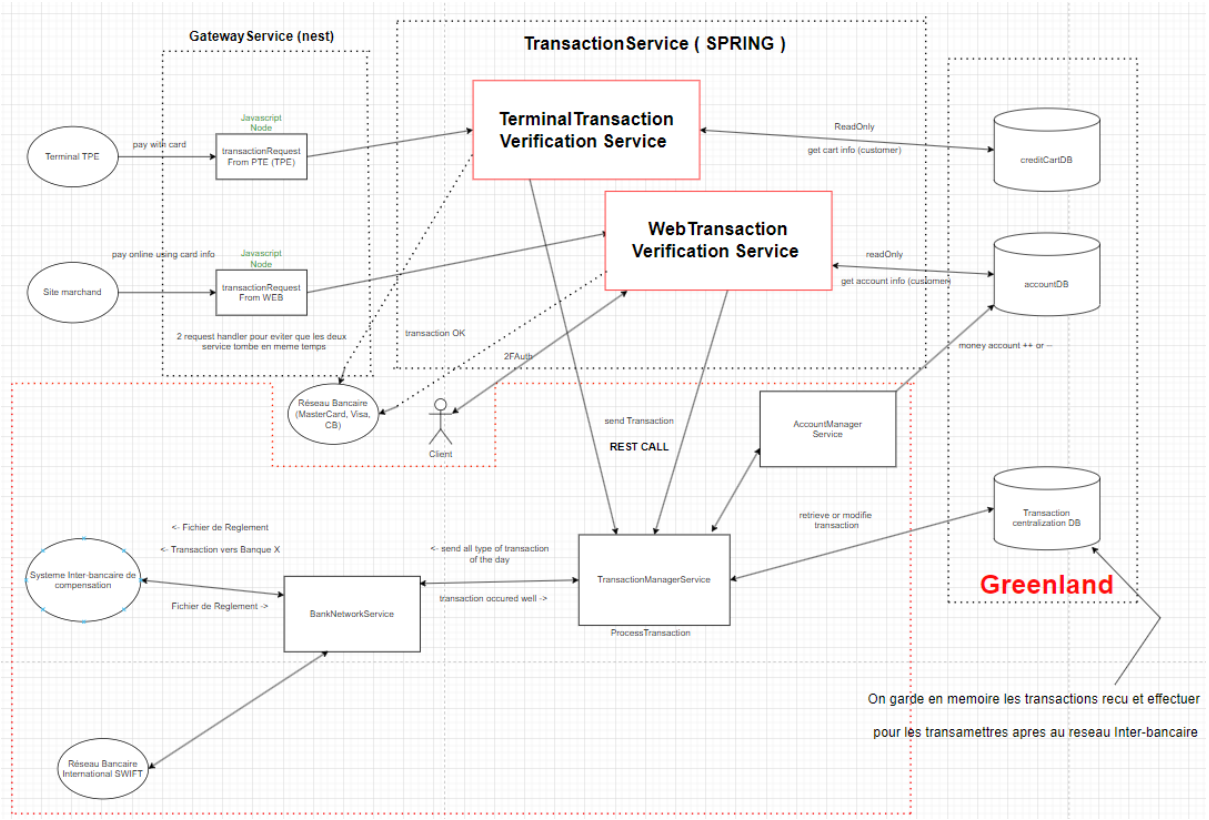
Follow-up Architecture Report
AL Néo-Bank

Week 40 :

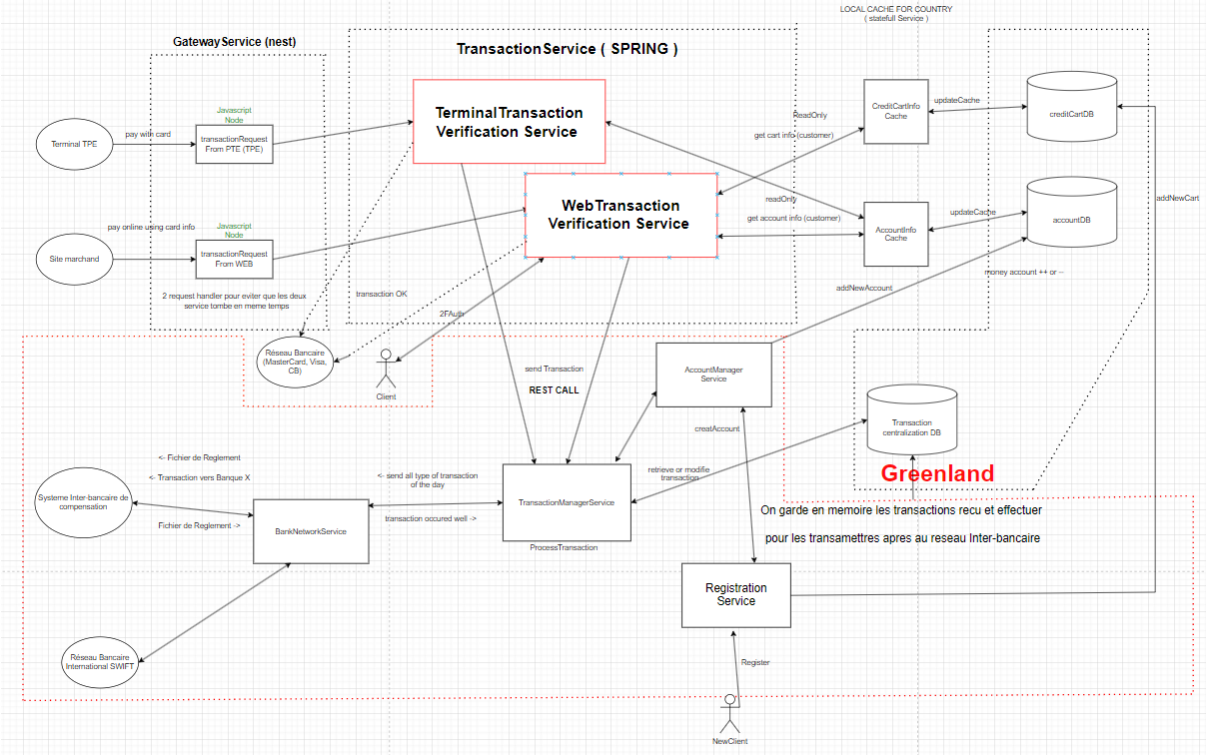
Current Structure of the services architecture :



Our architecture has been designed to meet a primary need: the scalability of our banking system. Indeed, the first type of service provided will be the banking transaction, so we decided to focus on this one.

In this diagram, we've separated the former large transactionService block into two services, one responsible for transactions issued from a TER and the other for transactions issued from another type of terminal, in our case a CB, Visa or other web-based banking transaction service. We have decided to leave bank transfers aside, as they are not a priority for the moment. This separation allows us to better manage a large flow of transaction requests. There will indeed be a certain amount of repetition in the verification process, but this will enable us to have a faster verification service in the TER case..

As far as the databases are concerned, at first we thought of a read-only solution. But given that we'll have to process a large number of requests in several countries in the future. With databases located in Greenland, this can be very slow. So we've come up with a kind of caching solution that would be set up on the servers in each country. Each time the databases undergo an update via a modification, they notify the caches that change their data. This solution would be even more feasible for credit cards, given that this information changes very little. Here's what the diagram would look like:



The parts circled in red are part of the transaction system (apart from customer registration, which is there to populate the databases). In fact, the most important thing is to create a simple transaction schema with the various rules and verifications linked to it. Then we'll add the system for notifying the banking network of successful transactions, and the processing of settlement files. These are logs of all the transactions carried out during the day, and it is on the basis of this file that we add or withdraw money from the various customers.

Note also that in the real world, a transaction comes from the credit card network and is then processed, so we've chosen to distinguish two types of transaction, rather than abstracting one from the credit card network.

Services :

TransactionRequestFromTPE:

- His role is to process the transaction for a TPE on client side, and to send only needed info to the service

TransactionRequestFromWEB:

- His role is to process the transaction coming from an NetworkCard API, such as Master, Visa or CB, and to send only needed info to the service

TransactionManagerService:

- Its role is to process a transaction when this one is validated by the validators services, it sends the transaction to the AccountManagerService so this service can process the impact of the recent transaction on account. He also add the transaction to the list of transaction of the day in the TransactionCentralisationDB. When the daily cicle has finished, he take the list of transaction that occurred during this time and send them to the BankNetwork. He can also be notified that transaction coming from other bank have arrived and he notifie the AccountManager (in futur implementation, will store those information so client can seed an historique of transaction)

AccountManagerService:

- Manage client's account. Create a new account for clients. Depending on the invoked action, can modify a client new balance depending on the transaction regarding him.
- Scaling point : this service is invoked only if needed, and for scale needs, can have just before a bus that content all the action that he have to archive (like add or retire money), those action can be done in the background as they dont have an immediat impact

BankNetworkService:

- Service that is linked to the outside, when a transaction comes from the systeme, it changes it into a reglementation file understandable by the banking network.
  When a file comes from the outside, I change it into a transactional JSON understandable by the TransactionManagerService.

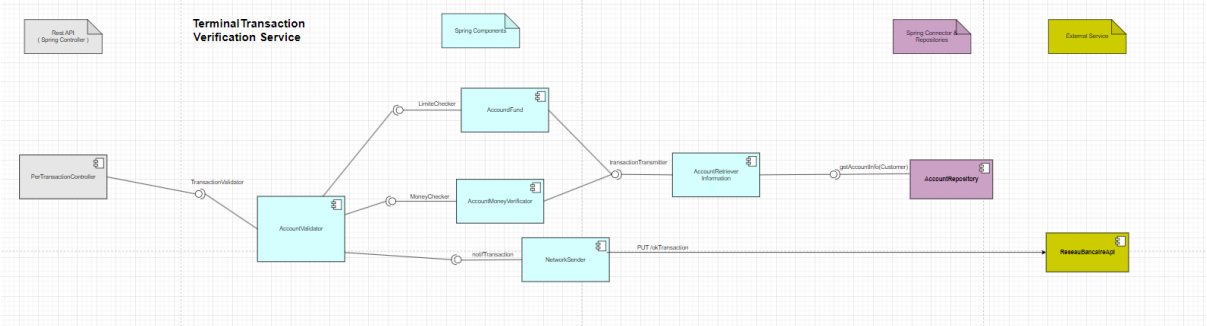
RegistrationService:

- Not importante service in this scope, here to fill DB. This service is here to manage the step in a new client registration. Treat a JSON with all client info and create his account and generate him a new neo-card.

TerminalTransactionVerificationService :

- Its role is to check if a wanted transaction can be done. It then send a REST Call to the transactionManagerService so it continue the process.

Component Diagram :



Components :

ComponentAccountValidator **:**

- receives a transaction and validates it or not

AccountFund **:**

- manages the verification linked to the account ceiling
  - checks if the account has reached the payment limit

AccountMoneyVerificator **:**

- handles verification related to account money, check if the account has enough money.If false: checks whether the account can or may still go into the red

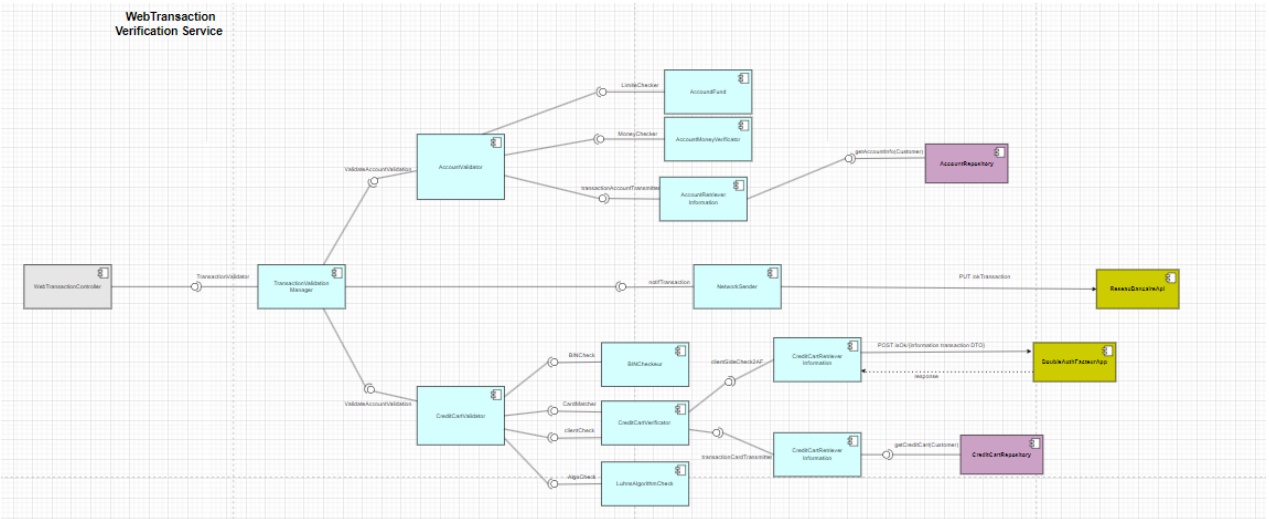
NetworkSender **:**

- if transaction is marked as successful, notifies the card network

TerminalTransactionVerificationService :

- its role is to check if a wanted transaction can be done. it then send a REST call to the transactionManagerService so it continue the process.

Component Diagram :



Components :

TransactionValidationManager:

- checks that all verifications have been carried out correctly, and is responsible for the order in which transactions are carried out.

CreditCartValidator:

- performs verification of credit card information

BINChecker:

- verifies regex format of card information submitted for verification

LuhnsAlgorithmCheck:

- checks the 16-digit code provided to verify the validity of the number

CreditCartVerificator:

- verifies that the information given by the customer corresponds to the object in BD

Persona :

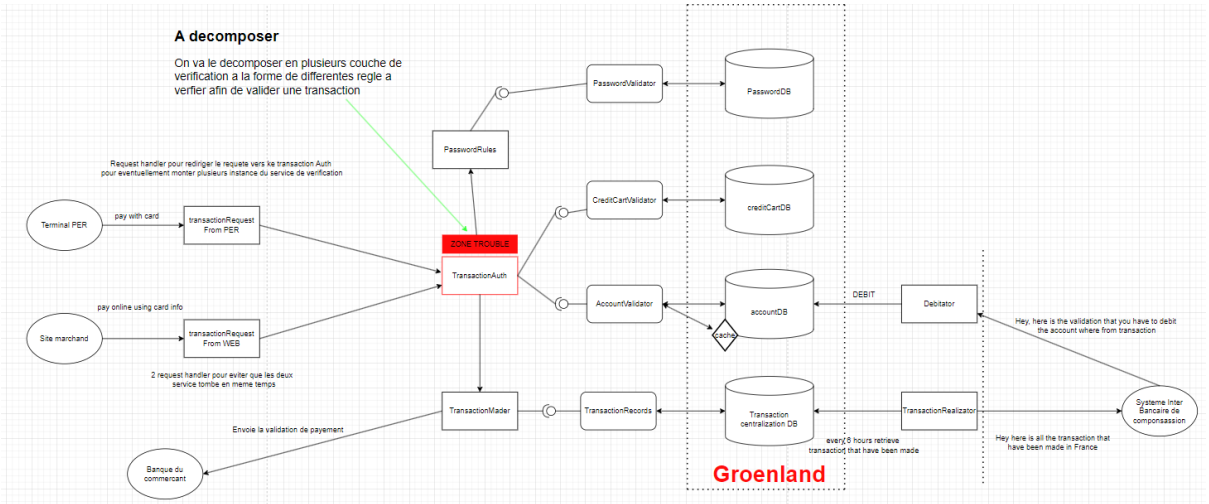
- Fabrice : technician at the new neo-bank "IgorBanque", his job is to keep BNP IgorBanque's servers up and running.
- Philippe : A merchant who recently opened a cookie-shop in Nice, he offers customers a means of paying by card, or on the internet via click and collect.

User Stories : user storie - explication - diagram de flow

- As Fabrice, i need to know from the cardNetwork which kind of transaction is coming
- As Phillipe, i need to know from my TPE terminal if the client transaction have succeeded
- As Phillipe, i need to be notified if a transaction have succeeded if the purchase of the click and collect have been made on my website
- As Fabrice, i need to know in the bank systeme if any kind of transaction have succeeded

Week 39 :

Current Structure of the architecture :



In our current architecture, we only support transactions for the moment, but we'd like to concentrate on them so that we can consider applications of version 7 of the bank, i.e. large-scale scaling.

In our schema, we have the following external services: the PER terminal, a merchant's site, his bank and the interbank clearing system, which manages money exchanges between banks.

We then have a verification layer which retransmits transaction requests to the central transaction verification system. We've imagined it this way in order to potentially isolate the transaction verification service, which could be the part most in demand during a scale-up, so we can raise several instances of this service.

In our verification service, we'd have a set of components that verify each part of the request, which must comply with a set of established rules (ceiling limit, sufficient balance, valid credit card, valid transmitted information, correct passwords, etc.). Most of the information will be stored in Greenland databases, with which we will interact to retrieve this information. A caching solution could also be deployed to speed up retrieval in the event of scaling.

A broadcast component will then be called if the transaction can go through. The money is then frozen on the account, and the new transaction is added to all those already processed. The customer's bank, if external, is notified of the pending transfer.

Every 6 hours (according to the Internet), the SIT will retrieve all the transactions carried out and, once all the transfers have been made, will tell the bank to withdraw the frozen money from the accounts. Given that the action of debiting the money happens every 6 hours in our conception of the thing, the debiting component may not be asynchronous in the immediate future, but in a scaling perspective, it probably will be.

In the end, we'll have 4 layers: transaction reception, verification, communication with external services and databases.

Ideally, what would be most feasible would be to group external management into a single service, which would also include functionality not required for transactions, and to output transaction verification processing in a single service.

Sequence Diagram Flow :