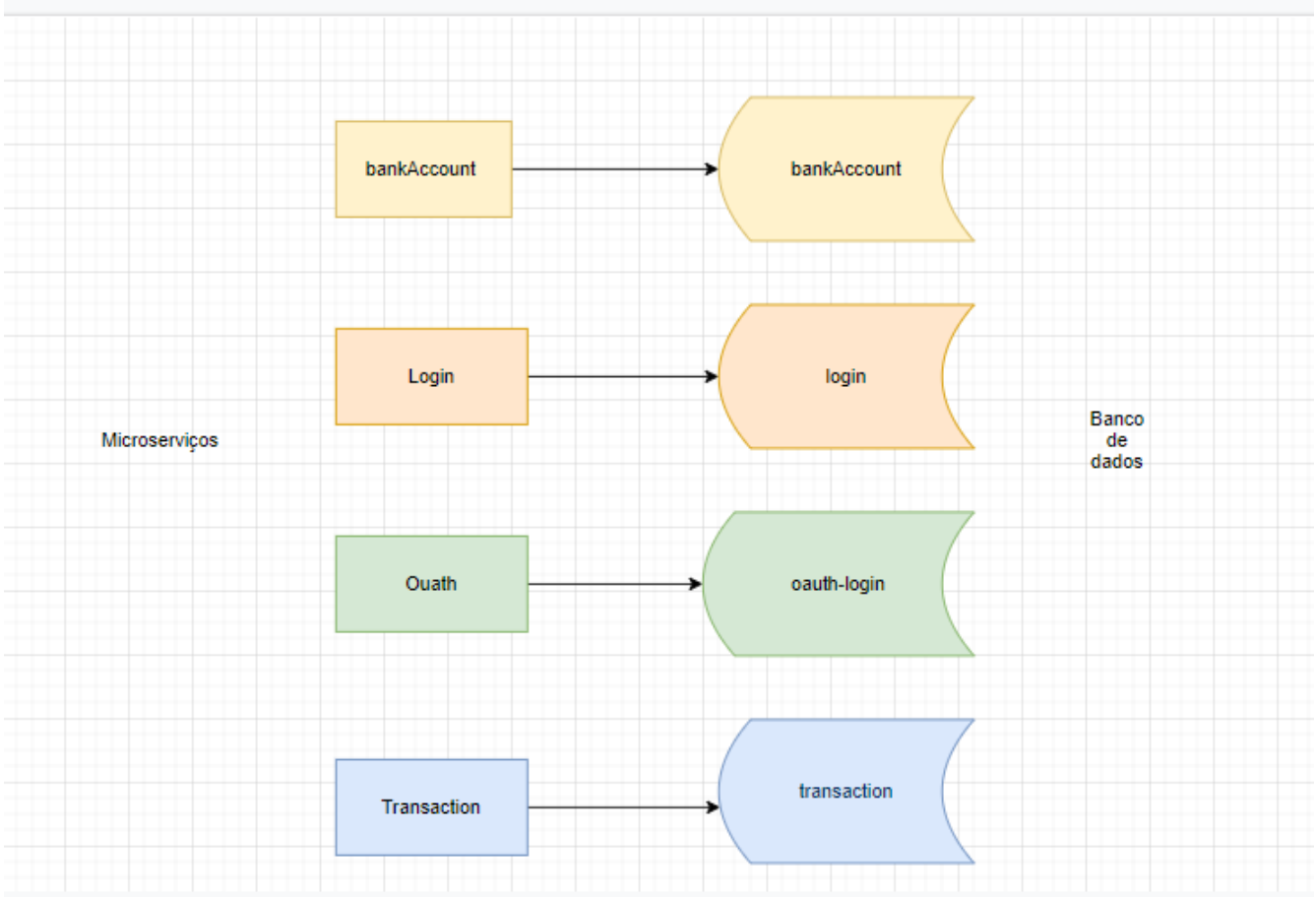
Requisitos não funcionais de nosso um internet banking

1)Disponibilidade 24 horas por dia
2) Segurança dos dados bancários , sendo que um usuário só deve acessar os dados de sua conta. As outras contas não devem ter acesso a conta de um usuário específico.
3)Rastreabilidade - Todas as transações feitas dentro do sistema devem ser de fácil rastreabilidade.
4) O tempo de resposta do sistema não deve ultrapassar 15 segundos.

Nossos microserviços

Veja abaixo os microservices e os bancos de dados que eles se conectam:



Bank account => Responsável pela criação de uma pessoa física , criação de conta corrente e conta poupança e também cartão de crédito.

Apis :

Post - /v1/customers - Cria uma pessoa física , uma conta corrente e uma conta poupança. Cada usuário vai ser criado com uma conta corrente e uma conta poupança. A conta corrente possui um cartão de crédito vinculado a ela. Na hora de criar uma conta, o nosso sistema cria o

login primeiro (chamando o serviço de login) e depois cria uma pessoa física , após a criação da pessoa física, ele vai criar as duas contas(corrente e poupança são criadas).
O usuário vai escolher qual conta deixar ativa na hora da criação , ele deve passar o typeAccount no body. Caso ele passe typeAccount = **SAVINGS_ACCOUNT**,duas contas serão criadas, mas somente a conta poupança será ativa. Se ele passar **CURRENT_ACCOUNT** , duas contas serão criadas, mas somente a conta corrente estará ativa.

Get => v1/customers/{document} - Responsável por buscar as informações dos usuários e de suas respectivas contas. Você utilizará as informações das contas para criar depósito , fazer retiradas e realizar ted . Temos o campo **AccountNumber** e o campo **checkDigit ,** eles irão o accountId do microserviço Transaction(accountId=AccountNumber+checkDigit)

Login => Criará um registro no banco de login e será responsável por fazer o login do usuário na aplicação. O usuário terá um login(cpf e senha) no banco login e esse serviço se comunicará com o serviço oauth para gerar o token do usuário. Esse token será usado pelos serviços de transaction. Somente usuários autenticados e tokenizados podem usar os serviços de transaction.

Apis :

Post - /login/authenticate => loga o usuário e retorna o token que será usado nas api de transações

Post - /login/register => Cria o registro do usuário no banco login

Oauth => responsável por criar o administrar a parte de tokenização do nosso sistema

Apis

/oauth/token => gera o token do usuário

/oauth/register => registra o usuário no servidor de autenticação

Transaction => Responsável pelas transações do sistema. O usuário deve estar logado para utilizar esse microserviço.
Temos uma api que pode criar uma transação genérica(addTransaction) , porém temos duas transações com modelos já prontos (depósito e retirada). Nas transações que precisarem informar o accountId do user (Account Number + dígito) , usar ele como:

Ex: **"accountNumber": 11223147,**
"checkDigit": 8,

Então :

AccountId=**112231478**

Usar o token gerado no corpo da resposta da api /login/authenticate no header das api do serviço de transaction. Para logar, você utiliza o cpf do usuário e senha criada no serviço de

***bankAccount(/v1/customers). O número da conta e o digíto pode ser obtido na hora da criação da conta , pois a geração retorna ele em seu body, ou usando api* v1/customers/{document}.**

Apis:

Post -> transaction/addTransaction => Adiciona uma transação bancária, de qualquer valor ou tipo(depósito ou retirada , por exemplo).

Abaixo os campos que compõem um transação:
id = Id da transação ;
accountId = Número da conta que será feita essa transação(account number + digito);
description_transaction: descricao da transacao , cada tipo de transacao com uma descricao
value= Valor da transação , podendo ser negativo ou positivo
createdDate = Data da criação da transação;
statusTransaction= Status da transação(1 para aprovado , 2 para reprovado e outros status podem ser criados;
descriptionStatus= Descricação do status da transação (Ex: Aprovado, Reprovado);
description_type_transaction = Descricação do tipo de transação(2 - Enter - Deposit 1 - Out - Withdrawal)
typeTransactionId =>1 - Withdrawal (saída de dinheiro) 2 - Depósito(Entrada)
money_name =Nome da moeda da transação (Ex: BRL - moeda real brasileira)
money_symbol = Símbolo da moeda da transação;
type_operation = Tipo de operação (Credit - C ou Debit - D);
description_extract = Descrição para o status;
guidId = Id para identificar a transação na hora de uma transferência;

Get - /transaction/balance : Traz o saldo da conta corrente e conta poupança usuário logado.

Get - /transaction/extract: Traz o saldo da conta corrente e conta poupança do usuário logado

Post - /transaction/deposit => Cria um depósito na conta do usuário. (Entrando dinheiro na conta do usuário). Informar o accountId do user (Account Number + digito) .
Ex: **"accountNumber": 11223147,**
"checkDigit": 8,
Então :
AccountId=**112231478**

Post - /transaction/withdrawal => Cria uma retirada de dinheiro da conta do usuário. Informar o accountId do user (Account Number + digito) .

Ex: **"accountNumber"**: **11223147**,
"checkDigit": **8**,

Então :

AccountId=**112231478**

Post - /transferByCPF => Transferência entre contas usando cpf . As contas serão buscadas pelo cpfs informados.

Campos para a transferência:

type_account_from : **Tipo da conta destino (1 para conta corrente e 2 para conta poupança)** ,

type_account_to:**Tipo da conta destino (1 para conta corrente e 2 para conta poupança),**

document_to : **documento do usuário da conta destino**,

"value":**Valor a ser transferencia**,

"description_extract":**"Descrição da transação no extrato"**