Vamos iniciar a construção de um projeto, para exercitar as habilidades de node.js e a capacidade de buscar novos conhecimentos e construir um projeto inteiro, abaixo os detalhes do projeto.

Cenário:

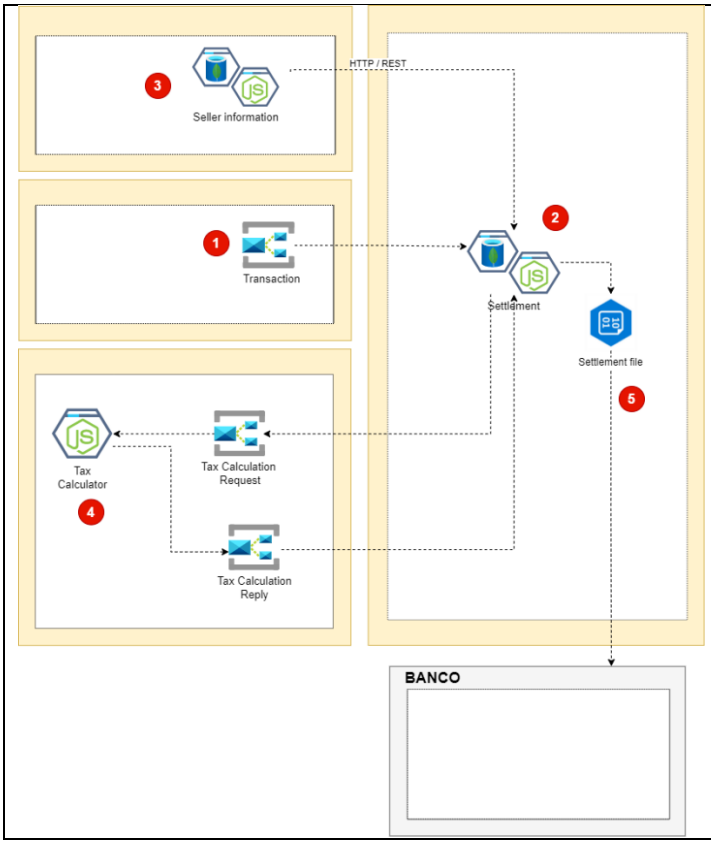
Quando um cliente faz uma compra em um estabelecimento (seller) ocorrem diversas operações, que de forma simplificada são:

1- Autorização – No momento que o cliente passa o cartão, é validado se o cartão está ativo, se tem saldo e se tudo estiver ok, a transação é autorizada e a maquininha de cartão imprime o comprovante.

2- Aprovação – Todas as transações autorizadas vão para um processo de aprovação, que confirma se não ocorreu nenhuma fraude, calcula as taxas que serão cobradas para aquela transação e se tudo tiver ok, disponibiliza a transação para liquidação.

3- Liquidação – Todas as transações aprovadas precisam ser enviadas para o banco, para realizar o pagamento do estabelecimento (seller). Ex: Se o cliente gastou 100,00 na padaria (seller), descontando as taxas de processamento, a padaria irá receber por exemplo R$ 98,00, então é necessário enviar um arquivo (settlement file) para o banco avisando que esse valor deverá ser enviado para a conta do dono da padaria.

Vamos desenvolver o projeto considerando somente o cenário de liquidação (settlement), abaixo um diagrama exemplificando:



1- Transações aprovadas são enviadas para o serviço via evento (EventBus);
2- O serviço de Settlement irá receber essas transações e gravar em um bd (Mongodb). Quando o processo de liquidação for solicitado irá agrupar todas as transações por seller_id e iniciar o processo de liquidação do movimento do dia;
3- Para cada transação o serviço Settlement irá obter os dados do Seller (conta bancária);
4- Para cada transação o serviço precisará enviar uma solicitação para cálculo de impostos, via evento;
5- Ao final do processo será gerado um arquivo que será enviado para o banco para liquidação;

Stack:

Node.js (Javascript), Mongodb, EventBus

Fase 1:

- Github

Crie um projeto “transaction_settlement” no seu github pessoal e a cada etapa, vamos adicionando os códigos para complementar o desafio. Preferencialmente, utilize nos commits mensagens

- Readme

Criar o Readme do projeto, descrevendo o objeto do processo e como executar o projeto.

- .gitignore

Também adicionar o arquivo .gitignore padrão do Nodejs

- Criar API seller_information

Vamos criar o projeto seller_information (item 3 do diagrama)

Criar a API utilizando padrão Restfull

Utilizar Node.js e Express

Deve ter os endpoints:

POST /v1/sellers

Inclui um novo Seller

request:

```
{
	"seller_id": 1,
	"name": "McDonalds",
	"cnpj": "90891366000190",
	"bankCode": 33,
	"bankAccount": 1000,
	"notes": ""
}
```

response:

HTTP 201

GET /v1/sellers/1

Obtém seller por id

request:

(vazio)

response:

```
{
```

```
	"seller_id": 1,
	"name": "McDonalds",
	"cnpj": "90891366000190",
	"bankCode": 033,
	"bankAccount": 1000,
	"notes": ""
}
```

PATH /v1/sellers/1

Altera informações de um seller

request:

```
{
	"notes": "anotações 1"
}
```

GET /v1/sellers?bankCode=033&page=1&pageSize=20

Retorna dados de forma paginada

request:

(vazio)

response:

```
[
	"page": 1,
	"pageSize": 20,
	"totalItems": 2,
	"totalPages": 1,
	"sellers": [
		{ "seller_id": 1, "name": "McDonalds", "cnpj": "90891366000190", "bankCode": 33, bankAccount:
1000, "notes": "anotações 1" },
		{ "seller_id": 2, "name": "BurguerKing", "cnpj": "61344649000114", "bankCode": 33,
"bankAccount": 2000 }
	],
]
```

- Swagger

A API deve ter documentação gerada de Swagger

- Banco de dados

A lista de Sellers deve ser gravada no MongoDb. Utilizar a lib mongose para conectar no mongo.

- Docker

Criar o arquivo de Docker compose para subir o Mongo para ser utilizado na API.