Projeto para Vaga de Analista Desenvolvedor Sênior

1. Avalara Brasil

A Avalara tem a mais completa solução fiscal do Brasil numa suíte única e em nuvem. A amplitude das nossas soluções elimina riscos e diminui custos, pois utiliza conteúdo em forma de tecnologia para validar os dados da empresa e da sua cadeia de clientes, fornecedores e dos produtos e serviços, calcular impostos, recuperar e emitir documentos fiscais e gerar, validar e monitorar as obrigações fiscais. Desde o nível transacional até o nível de conformidade fiscal (compliance).

Além de produtos também oferecemos serviços que tem a mesma amplitude para que a empresa possa se dedicar ao seu negócio, pois definitivamente seu negócio não é gestão fiscal, mas o nosso é.

`Make Tax Easy`

2. Projeto - Cálculo de Impostos

Nesse projeto o candidato terá de fazer uma API REST para calcular os impostos dos itens informados pelo ERP do `Cliente J`.

O contrato firmado entre esse `Cliente J` e a Avalara estabelece que o calculo de impostos não deva superar o tempo de 200 milissegundos.

> **Leia a parte de criação do Projeto no GitHub.**

3. Calculo de Imposto - Processos

O Projeto irá calcular três Impostos:

- IEC - Imposto para Encher o Cofre: definido pelo Diagrama do IEC.
- IST - Imposto Sobre Tudo: definido pelo Diagrama do IST.
- ISC - Imposto Sobre Consumo: definido pelo Diagrama do ISC.

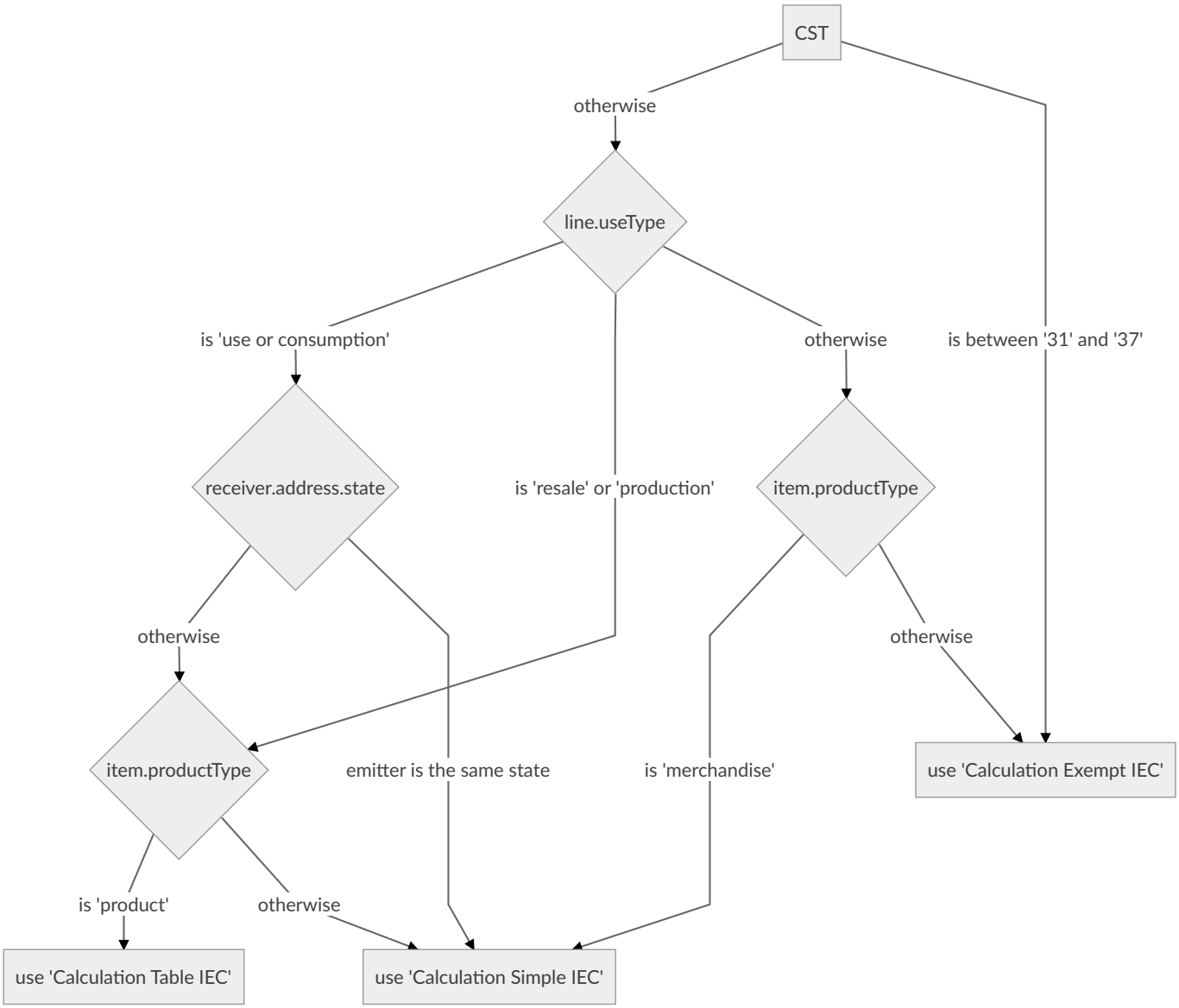
Os impostos irão usar o mesmo CST (Código da Situação Tributária), definido pelo Diagrama de CST.

3.1. Diagrama do IEC

O IEC é um imposto fictício de âmbito federal (`Jurisdiction = 'Country'`). Ele é cobrado sobre todos os tipos de transação que não envolva uma venda pra um Órgão Governamental Brasileiro.

É um imposto fictício que serve para que o Governo faça a manutenção dos buracos no seu orçamento, e assim, conseguir gerenciar a máquina pública.

Para calcular o IEC, a Avalara elaborou o seguinte diagrama:



Esse imposto não tem dependência de outro imposto. Mas ele é usado para compor a Base de Calculo do IST.

3.1.1. Calculation Simple IEC

O Cálculo do IEC Simples, deve seguir as seguintes regras:

- Para calcular a Base de Calculo do IEC Simples (`Base`), use a formula:

$$Base = Amount + OtherCosts - Discount$$

- A Alícota do IEC Simples é configurada por `Item` .

```
Rate = item.federalTax.IEC.rate
```

- O Imposto é calculado com a simples multiplicação da Base de Calculo com a Alícota.

$$Tax = Base * Rate$$

- Além dos valores acima, deve informar o tipo de cálculo que foi realizado:

```
Scenario = 'Calculation Simple'
```

3.1.2. Calculation Table IEC

O Cálculo do IEC Tabelado, deve seguir as seguintes regras:

- Antes de fazer o calculo da Base de Calculo, é necessário verificar o Fator de Redução de Base (`Fact`) a ser aplicado.

> Esse Fator de Redução de Base é configurado por `Item` e varia entre `0` e `1`. Por padrão o valor é `1`.

```
Fact = item.federalTax.IEC.fact
```

- O calculo da Base de Calculo do IEC por Tabela (`Base`) utiliza a seguinte formula:

$$Base = Amount + OtherCosts - Discount$$

- A Alícota do IEC definida pela Tabela de IEC que possui uma Alícota para cada Região do Brasil.

```
RecvAddr = receiver.address
EmtAddr = emitter.address
// IEC Table
Rate = findRateFromIECTable(RecvAddr.state, EmtAddr.state)
```

- O Imposto é calculado com a multiplicação da Base de Calculo (aplicando o Fator de Redução de Base, se existir) com a Alícota.

$$Tax = (Base * (1 - Fact)) * Rate$$

- Além dos valores acima, deve informar o tipo de cálculo que foi realizado:

```
Scenario = 'Calculation Table'
```

3.1.3. Calculation Exempt IEC

O Cálculo de Isenção do IEC, deve seguir as seguintes regras:

- O calculo da Base de Cálculo (`Base`) utiliza a seguinte formula:

$$Base = Amount + OtherCosts - Discount$$

- No Cálculo de Isenção do IEC os valores de Alícota e do Imposto devem ter seus valores ser zerados.

```
Rate = 0
Tax = 0
```

- Além dos valores acima, deve informar o tipo de cálculo que foi realizado:

```
Scenario = 'Calculation Exempt'
```

3.1.4. IEC Table

Tabela de Alicotas para calculo do IEC:

Region	Same State	Other State
CO	6.1%	4.26%
N	4.5%	3.93%
NE	2.73%	1.39%
S	8.41%	9.08%
SE	6.5%	5.84%

> Regiões:
>
> - CO : Centro-Oeste
> - N : Região Norte
> - NE : Região Nordeste
> - S : Região Sul
> - SE : Região Sul

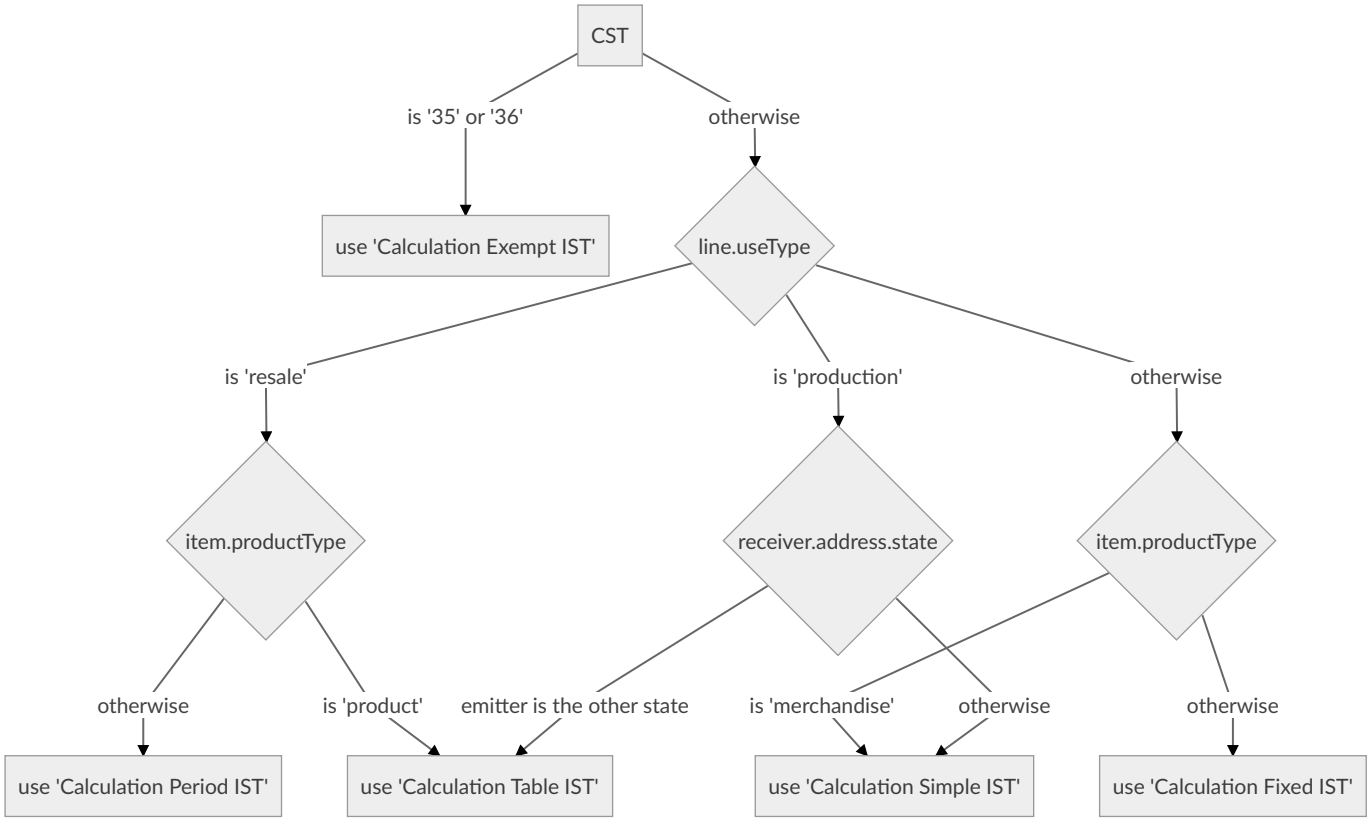
3.2. Diagrama do IST

O IST é um imposto fictício de âmbito Estadual (`Jurisdition = 'State'`). Que é cobrado de todas as transações no Brasil, com exceção de vendas para Órgãos Governamentais Federais e Estaduais.

Esse imposto fictício surgiu com a incrível solução de unificar os antigos impostos brasileiros.

A primeira idéia era a de simplificar a vida do contribuinte, mas na prática só somou as antigas alícotas e manteve suas complexidades.

Para calcular o IST, a Avalara elaborou o seguinte diagrama:



Dependendo do Cálculo utilizado, esse imposto pode ter a adição do IEC na Base de Cálculo.

3.2.1. Calculation Simple IST

O Cálculo do IST Simples, deve seguir as seguintes regras:

- Para calcular a Base de Calculo (`Base`), use a formula:

$$Base = Amount + OtherCosts - Discount$$

- A Alícota do IST Simples é configurada por `Item` .

```
Rate = item.federalTax.IST.rate
```

- O Imposto é calculado com a simples multiplicação da Base de Calculo com a Alícota.

$$Tax = Base * Rate$$

- Além dos valores acima, deve informar o tipo de cálculo que foi realizado:

```
Scenario = 'Calculation Simple'
```

3.2.2. Calculation Table IST

O Cálculo do IST Tabelado, deve seguir as seguintes regras:

- Antes de fazer o calculo da Base de Calculo, é necessário verificar o Fator de Redução de Base (`Fact`) a ser aplicado nessa base.

> Esse Fator de Redução de Base é configurado por `Item` e varia entre `0` e `1`. Por padrão o valor é `1`.

```
Fact = item.federalTax.IST.fact
```

- O calculo da Base de Calculo (`Base`), adiciona o valor do Imposto do IEC e utiliza a seguinte formula:

$$Base = Amount + OtherCosts - Discount + IEC$$

- A Alícota do IST definida pela Tabela de IST que possui uma Alícota para cada Região do Brasil.

```
RecvAddr = receiver.address
EmtAddr = emitter.address
// IST Table
Rate = findRateFromISTTable(RecvAddr.state, EmtAddr.state)
```

- O Imposto é calculado com a multiplicação da Base de Calculo (aplicando o Fator de Redução de Base, se existir) com a Alícota.

$$Tax = (Base * (1 - Fact)) * Rate$$

- Além dos valores acima, deve informar o tipo de cálculo que foi realizado:

```
Scenario = 'Calculation Table'
```

3.2.3. Calculation Exempt IST

O Cálculo de Isenção do IST, deve seguir as seguintes regras:

- O calculo da Base de Calculo (`Base`) utiliza a seguinte formula:

$$Base = Amount + OtherCosts - Discount$$

- No Cálculo de Isenção do IST os valores de Alícota e do Imposto devem ter seus valores ser zerados.

```
Rate = 0
Tax = 0
```

- Além dos valores acima, deve informar o tipo de cálculo que foi realizado:

```
Scenario = 'Calculation Exempt'
```

3.2.4. Calculation Period IST

O Cálculo por Período do IST, deve seguir as seguintes regras:

- Para os meses que não possuem `R` (que são: Maio, Junho, Julho e Agosto), possuem um Fator de Redução de Base de de `40%`.

> O Cálculo por Período do IST varia de acordo com o mês que a transação é feita.

```
Fact = 0.4
```

- Além de usarem a Alícota definida na Tabela de IST.

```
RecvAddr = receiver.address
EmtAddr = emitter.address
// IST Table
Rate = findRateFromISTTable(RecvAddr.state, EmtAddr.state)
```

- Para o calculo da Base de Calculo do IEC (`Base`), adiciona o Imposto do IEC e utiliza a seguinte formula:

$$Base = Amount - Discount + IEC$$

- Já nos meses de: Janeiro, Fevereiro, Março, Abril, Setembro, Outubro e Dezembro, o Fator de Redução é configurado por `Item` e varia entre `0` e `1`. Por padrão o valor é `1`.

```
Fact = item.federalTax.IST.fact
```

- Bem como a Alícota do IST é também configurada por `Item`.

```
Rate = item.federalTax.IST.rate
```

- O calculo da Base de Calculo do IST (`Base`) utiliza a seguinte formula:

$$Base = Amount - Discount$$

- O Imposto é calculado com a multiplicação da Base de Calculo (aplicando o Fator de Redução de Base, se existir) com a Alícota.

$$Tax = (Base * (1 - Fact)) * Rate$$

- Além dos valores acima, deve informar o tipo de cálculo que foi realizado:

```
Month = Date.format('MM')
Scenario = 'Calculation Period'
```

> O Mês deve ser retornado em formato numérico de 2 dígitos no padrão RFC 6350.

3.2.5. Calculation Fixed IST

O Cálculo do IST Fixado, deve seguir as seguintes regras:

- Para calcular a Base de Calculo (`Base`), use a formula:

$$Base = Amount + OtherCosts$$

- Sendo está Alícota do IST fixa de `14%` com Fator de Redução de Base de `8%`.

```
Rate = 0.14
Fact = 0.08
```

- O Imposto é calculado com a multiplicação da Base de Calculo (aplicando o Fator de Redução de Base, se existir) com a Alícota.

$$Tax = (Base * (1 - Fact)) * Rate$$

- Além dos valores acima, deve informar o tipo de cálculo que foi realizado:

```
Scenario = 'Calculation Fixed'
```

3.2.6. IST Table

Tabela de Alicotas para calculo do IST é usada para pegar a Alícota de um Estado de Origem para um Estado de Destino.

Nela estão presentes os seguintes Estados de Origem (topo da tabela):

- `AM`, `BA`, `CE`, `ES`, `GO`, `MA`, `MT`, `MG`, `PA`, `PB`, `PR`, `PE`, `RS`, `RJ`, `SC`, `SP`.

Os demais Estados de Origem possuem Alícotas pré-fixadas de `8%` para todos os destinos.

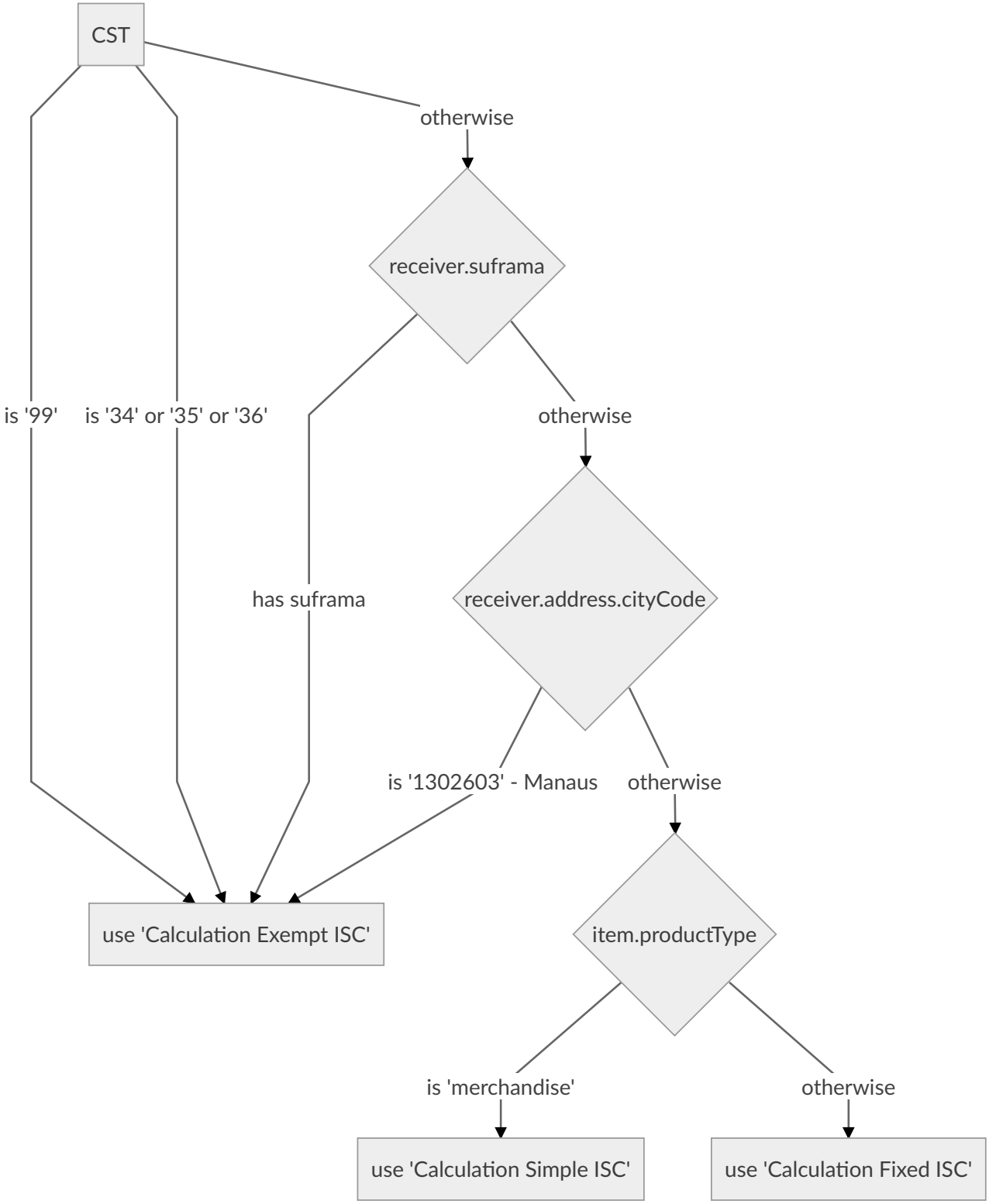
	AM	BA	CE	ES	GO	MA	MT	MG	PA	PB	PR	PE	RS	RJ	SC	SP
AC	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	8%	12%	9%	12%
AM	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	9%	12%	12%	12%	12%
AP	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	9%	12%
BA	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%
CE	12%	7%	7%	6%	12%	12%	12%	12%	12%	12%	12%	10%	12%	12%	11%	12%
DF	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%
ES	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	9%	12%	12%
GO	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%
MA	12%	4%	4%	12%	12%	12%	8%	12%	12%	12%	12%	12%	5%	12%	12%	12%
MT	12%	12%	12%	12%	5%	12%	12%	9%	12%	12%	12%	12%	12%	12%	12%	12%
MS	12%	10%	12%	12%	12%	12%	12%	12%	13%	12%	12%	12%	12%	11%	12%	12%
MG	12%	12%	12%	13%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%
PA	12%	16%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	16%	12%	12%
PB	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%
PR	11%	11%	11%	11%	12%	12%	12%	12%	12%	11%	11%	11%	12%	11%	12%	12%

	AM	BA	CE	ES	GO	MA	MT	MG	PA	PB	PR	PE	RS	RJ	SC	SP
PE	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	6%	12%	12%	12%	12%
PI	12%	5%	12%	12%	12%	12%	12%	12%	13%	12%	12%	12%	12%	12%	12%	12%
RN	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%
RS	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	5%	12%
RJ	12%	12%	12%	12%	12%	12%	12%	12%	13%	12%	12%	12%	13%	12%	12%	12%
RO	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	5%	12%	12%	12%	12%
RR	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	7%	12%	12%	12%
SC	12%	16%	12%	12%	12%	12%	16%	16%	12%	12%	16%	16%	12%	16%	16%	16%
SP	12%	7%	18%	12%	12%	18%	12%	17%	12%	7%	12%	7%	18%	12%	12%	12%
SE	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%
TO	12%	12%	12%	12%	12%	12%	12%	12%	12%	12%	18%	12%	12%	12%	12%	12%

3.3. Diagrama do ISC

O ISC é um imposto fictício de âmbito Municipal (`Jurisdition = 'City'`). Ele é cobrado sobre todos os tipos de produtos comercializados numa cidade, e que não envolva uma venda pra um Órgão Governamental Brasileiro.

É um imposto fictício que serve para punir o consumo desequilibrado de mercadorias nas cidades brasileiras.
Para calcular o ISC, a Avalara elaborou o seguinte diagrama:



O ISC não é cobrado para Empresas com `Código Suframa` ou que estejam na `Zona Franca de Manaus`.

Esse imposto não tem dependência de outro imposto.

3.3.1. Calculation Simple ISC

O Cálculo do ISC Simples, deve seguir as seguintes regras:

- Para calcular a Base de Calculo (`Base`), use a formula:

$$Base = Amount + OtherCosts - Discount$$

- A Alícota do ISC Simples é configurada por `Item`.

```
Rate = item.federalTax.ISC.rate
```

- O Imposto é calculado com a simples multiplicação da Base de Calculo com a Alícota.

$$Tax = Base * Rate$$

- Além dos valores acima, deve informar o tipo de cálculo que foi realizado:

```
Scenario = 'Calculation Simple'
```

3.3.2. Calculation Exempt ISC

O Cálculo de Isenção do ISC, deve seguir as seguintes regras:

- O calculo da Base de Calculo (`Base`) utiliza a seguinte formula:

$$Base = Amount + OtherCosts - Discount$$

- No Cálculo de Isenção do ISC os valores de Alícota e do Imposto devem ter seus valores ser zerados.

```
Rate = 0
Tax = 0
```

- Além dos valores acima, deve informar o tipo de cálculo que foi realizado:

```
Scenario = 'Calculation Exempt'
```

3.3.3. Calculation Fixed ISC

O Cálculo do ISC Fixado, deve seguir as seguintes regras:

- Para calcular a Base de Calculo (`Base`), use a formula:

$$Base = Amount + OtherCosts$$

- Sendo está Alícota do ISC fixa de `2%` com Fator de Redução de Base de `12%`.

```
Rate = 0.02
Fact = 0.12
```

- O Imposto é calculado com a multiplicação da Base de Calculo (aplicando o Fator de Redução de Base, se existir) com a Alícota.

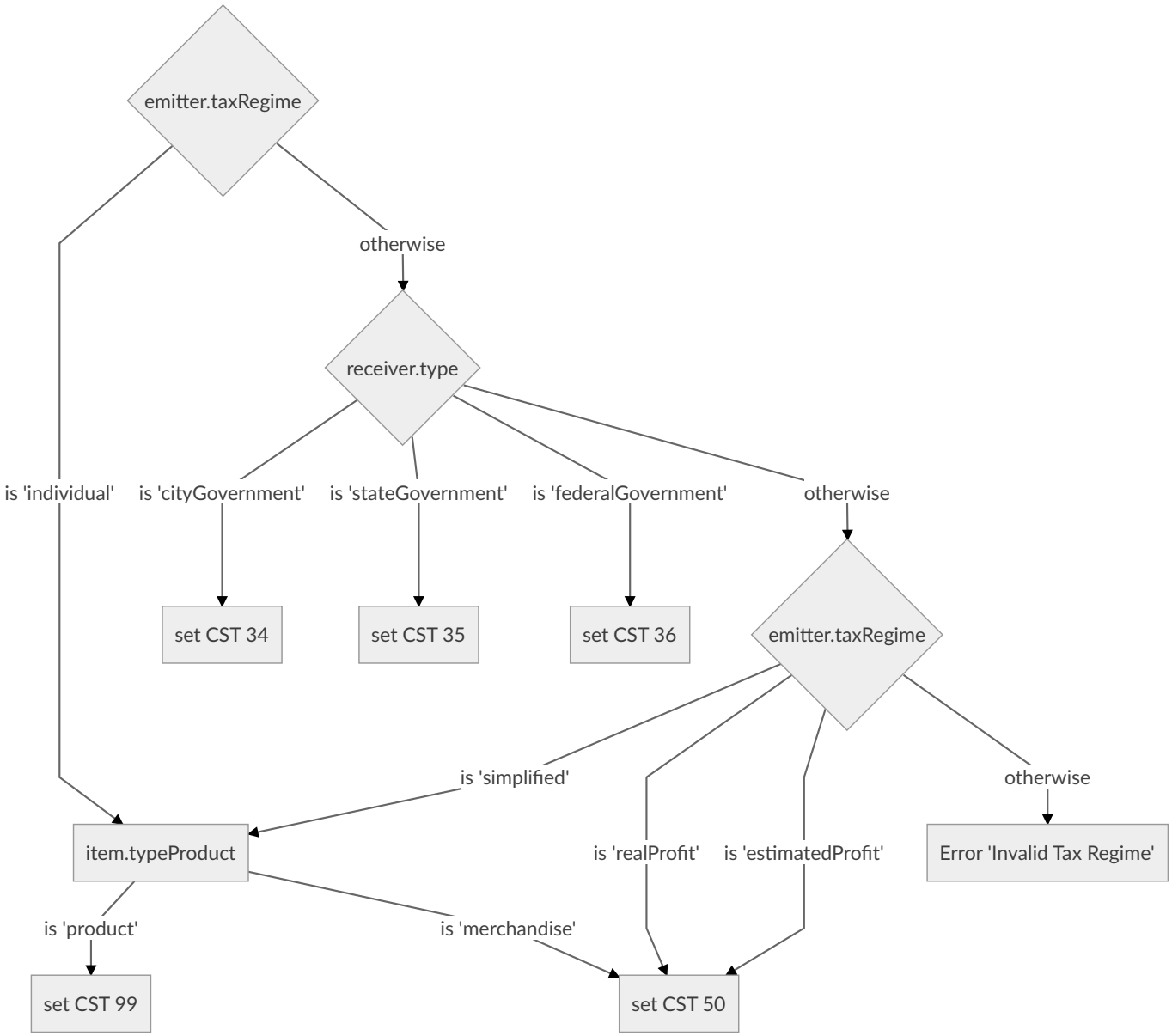
$$Tax = (Base * (1 - Fact)) * Rate$$

- Além dos valores acima, deve informar o tipo de cálculo que foi realizado:

```
Scenario = 'Calculation Fixed'
```

3.4. Diagrama de CST

Para identificar o CST, a Avalara elaborou o seguinte diagrama:



Todos os impostos utilizam o mesmo CST. Logo ele é pré-requisito para calcular um imposto.

- Sempre identifique o CST primeiro.

O CST deve ser retornado por linha de cálculo.

```
CST = '50'
```

> Ele é sempre numérico de 2 dígitos.

4. Padrões e Tecnologias

O Projeto será desclassificado caso os padrões não sejam respeitados.

4.1. GitHub

O Projeto deve ser feito no seu GitHub, utilizando os recursos para criar Issues, Pull Requests e Branchs.

Tudo no projeto será avaliado. Desde suas estruturas de pastas, nomes de arquivos e códigos fontes. Assim como cada recurso que facilite e melhore o processo Integração Continua (CI) será considerado um diferencial no processo de seleção.

Ao receber esta prova e lê-la. O candidato (para iniciar o processo) deve criar um **repositório no GitHub** e enviar o link para o email daniel.joppi@avalara.com, com o título `[TaxCalculation] [GitHub] - {nome do candidato}`.

> **Após o envio do endereço do projeto, não se mais deve usar o email para tirar dúvidas.**

O candidato poderá fazer perguntas utilizando **apenas** as Issues do projeto criado no GitHub.

Nessas Issues ele poderá perguntar ao usuário @danieljoppi, que agora será tratado como `Cliente J`.

> **No máximo duas perguntas serão respondidas por dia.**

O `Cliente J` responderá **apenas** dúvidas sobre o Cálculo de Impostos.

4.2. GitFlow

Seguir o GitFlow da Avalara, para criação e nomeação de Branches. Veja o artigo sobre o GitFlow escrito por Jeff Kreeftmeijer.

No Projeto, cada Issue deve estar contida numa Branch, e esta gerará um Pull Request.

A nomeção da Branch deve conter **type** e o **issue**. Que informa, respectivamente, o `tipo da branch` e o `número da issue` que está sendo referenciada.

```
git checkout -b <type>/AV-<issue>
```

Para isso deve se utilizar um dos dois tipos de Branches:

- **feature**: Branch de Melhoria.
- **fix**: Branch de Correção de Bug.

4.2.1. Exemplos de Criação de Branches

- Criando uma Branch de Melhorias para resolver a Issue `8`.

```
git checkout -b feature/AV-8
```

- Criando uma Branch para arrumar o Bug relatado na Issue `11`.

```
git checkout -b fix/AV-11
```

4.3. Commit Convention

Seguir o Commit Convetion da Avalara.

4.3.1. Commit Message Format

Cada commit deve Informar o que esta sendo feito.
E requer que a mensagem possua um **header** e um **body**.

O **header** é formado por um **type** e o **issue**. Que informa, respectivamente, o `tipo de alteção` e o `número da issue` que está sendo referenciada no commit.

> *Logo, todos os commits devem estar atrelados a uma issue.*

O **body** deve conter a descrição do que foi feito. Deve ser feito numa frase curta em inglês.

```
<type>: #<issue> - <body>
```

Para o **type** deve usar um dos seguintes tipos:

- **feat**: Nova funcionalidade.
- **fix**: Correção de Bug.
- **docs**: Alterações de Documentação.
- **style**: Formatação de Código, Remoção de Espaços, Adição/Remoção de ponto e virgula. Tudo que não afete a funcionalidade do programa.
- **refactor**: Re-fatoração de código.
- **perf**: Melhoria de performance.
- **test**: Adição/Correção ou Melhoria nos tests
- **ci**: Implementações no CI.

4.3.2. Exemplos de Commits

- Adicionando uma nova `Classe X`, referenciando a issue de número `1`.

```
feat: #1 - add new Class X
```

- Corrigindo um Bug da `Classe X`, referenciando a issue de número `1`.

```
fix: #1 - missing semi-colon in Class X
```

- Melhorando um Teste da Função `GetMyMoney`, referenciando a Issue de número `6`.

```
test: #6 - add test for function GetMyMoney
```

4.4. Semantic Versioning

Usar o padrão de numeração de versão Semantic Versioning 2.0.0.

> Inicie o projeto na versão **0.1.0**. De preferência atualizar no primeiro Pull Request.

4.5. Cobertura de Testes

O Projeto deve utilizar e integrar uma das seguintes ferramentas de análise de cobertura:

- [ ] Codecov
- [ ] Coveralls

> Se o Candidato preferir usar outra. Deve justificar numa Issue.

> Não será aceito o Projeto com menos de 91% de cobertura.

4.6. Banco de Dados

O Projeto deve utilizar o como base de dados os dois bancos de dados:

- **Cassandra**: Deve ser o principal banco de dados, onde será usado para armazenar os dados.
- **Redis**: É usado para cache de memória e replicação de dados do Cassandra.

> *Necessitando, o projeto poderá utilizar os bancos de outras formas. Mas essas mudanças devem estar documentadas em Issue.*

4.6.1. Modelos de Dados

- `Company` : É o modelo do Grupo da Empresa.

```
interface Company {
 /** * Company ID */ id: string; //> unique, format uuid
 /** * Company Code */ code: string;  //> unique
 /** * Company name */ name: string;
}
```

- `Location` : É o modelo da base do cadastro da Empresa. Todos os dados importantes da empresa estarão salvos neste modelo.

> Uma `Company` pode ter várias `Locations` , isto é, pode ter matriz e filiais (cada uma com seu CNPJ).

```
interface Location {
 /** * Company ID */ companyId: string; // format uuid
 /** * Location Code */ code: string;
 /** * Email */ email?: string;
 /** * Federal Tax ID, CNPJ or CPF */ federalTaxId?: string;
 /** * State Tax ID */ stateTaxId?: string;
```

```
 /** * City Tax ID */ cityTaxId?: string;
 /** * Suframa Code */ suframa?: string; // format ([0-9]{8,9})?
 /** * Main location activity */ mainActivity?: "commerce" | "industry" | "service";
 /** * Location Address */ address: Address;
}
```

- `Address` : É o modelo que define o endereço da Empresa.

> Note que tanto `Location` quando `Entity` usando a mesma estrutura de `Address` .

```
interface Address {
 /** * Street Name */  street?: string;
 /** * Neighborhood Name */  neighborhood?: string;
 /** * Zip Code */  zipcode?: string;
 /** * City Code */  cityCode?: number;
 /** * City Name */  cityName?: string;
 /** * State Code */  state: string;
 /** * Country Code */  countryCode?: number; // [0-9]{1,4}
 /** * Country Code - ISO 3166-1 alpha-3 */  country?: string; // ^([A-Z]{3})$
 /** * House number */  number?: string;
 /** * Complement */  complement?: string;
 /** * Phone number */  phone?: string; // ^(\d{6,14}|\(\d{2}\)\s*\d{4,5}-*\d{4})$
}
```

- `Item` : É o cadastro do produto vendido pela Empresa.

> Possui o código de referência do ERP e a configuração de impostos.

```
interface Item {
 /** * Company ID */  companyId: string; // format uuid
 /** * ERP Code */  code: string;
 /** * Item Description */  description?: string;
 /** * Product Type */ productType: "product" | "merchandise"
 /** * Federal Tax */  federalTax: federalTax;
}
```

- `FederalTax` : É o grupo de impostos configurados por `Item` .

```
interface FederalTax {
 /** * IEC Tax */  IEC: TaxType;
 /** * IEC Tax */  IST: TaxType;
 /** * IEC Tax */  ISC: TaxType;
}
```

- `TaxType` : É a configuração do Imposto dentro do `FederalTax` .

```
interface TaxType {
 /** * Rate */  rate?: number;
 /** * Fact */  fact?: number;
}
```

- `Transaction` : É a estrutura base que a API irá receber para calcular os impostos.

> O ERP enviará para API, apenas os dados de `Header` e `Lines`.

> A API deverá calcular inserir os dados de `CalculatedTaxSummary` e `ProcessingInfo` na transação. Bem como detalhar os cálculos de cada `Line`.

```
interface Transaction {
 /** * Header */ header: Header;
 /** * Lines */ lines: Line[];
 /** * Summary */  calculatedTaxSummary?: CalculatedTaxSummary;
 /** * Processing Info */ processingInfo?: ProcessingInfo;
}
```

- `Header` : Neste modelo se encontram os dados gerais para essa transação.

> No campo `companyLocation` é passado o código para localizar a `Location` :
>
> ```
> SELECT * FROM Location L WHERE L.code = {companyLocation}
> ```

> No campo `transactionType` é informado se a transação é de Venda (Sale) ou de Compra (Purchase).
> Isto serve para definir se o `Receiver` e o `Emitter` , para isso usa-se a regra:

	Sale	Purchase
Location	Emitter	Receiver
Entity	Receiver	Emitter

```
interface Header {
 /**  Transaction Type */ transactionType: "Sale" | "Purchase";
 /** Document Code */  documentCode: string;
 /** Currency */  currency: "BRL";
 /** Transaction Date */  transactionDate: string; // format ISO 8601
 /** Company Location Code */  companyLocation: string;
 /** Entity */ entity?: Entity;
}
```

- `Entity` : É o modelo semelhante ao `Location` , mas informa os dados da Empresa que está comprando ou vendendo um `Item` para a `Location` .

> Se for **`transactionType` = Sale**, significa que esta `Entity` está comprando um `Item` da `Location` .

- Neste caso, podemos dizer que o `Entity` é um Cliente da Empresa.

> Ou, se for **`transactionType` = Purchase**, significa que esta `Entity` está vendendo um `Item` para `Location` .

- Neste caso, podemos dizer que o `Entity` é um Fornecedor da Empresa.

```
interface Entity {
 /** * Email */ email?: string;
 /** * Federal Tax ID, CNPJ or CPF */ federalTaxId?: string;
 /** * State Tax ID */ stateTaxId?: string;
```

```
 /** * City Tax ID */ cityTaxId?: string;
 /** * Suframa Code */ suframa?: string; // format ([0-9]{8,9})?
 /** * Location Address */ address: Address;
}
```

- `Line` : É o modelo que informa o `Item` que está sendo comercializado, com seu valor negociado, quantidade.

> No campo `itemCode` é passado o código para localizar o `Item` :
>
> ```
> SELECT * FROM Item T WHERE T.code = {itemCode}
> ```

> A API irá inserir os dados de `CalculatedTax` ao calcular os impostos.

```
interface Line {
 /** * Line Code */  lineCode: number;
 /** * Item Code */  itemCode: string;
 /** * Number of Items, Quantity */  numberOfItems?: number; // default 1
 /** * Item Price */  itemPrice: number;
 /** * Amount */  lineAmount?: number; // default (numberOfItems * itemPrice)
 /** * Description */ itemDescription?: string;
 /** * Discount  */  lineDiscount?: number; // default 0
 /** * Other Cost  */  otherCostAmount?: number; // default 0
 /** * Calculated Tax */ calculatedTax?: CalculatedTax;
}
```

- `CalculatedTax` : É o modelo que lista os detalhes de cada cálculo de imposto e informa o total de impostos referente ao `Item` nesta linha (`Line`).

> É de responsabilidade da API preencher este modelo.

```
interface CalculatedTax {
 /** * Tax Details */ taxDetails: TaxDetails
 /** * Total Tax */ tax: number;
}
```

- `TaxDetails` : É o modelo que agrupo os detalhes dos três impostos.

```
interface TaxDetails {
 /** * IEC Detail */ iec: Detail
 /** * IST Detail */ ist: Detail
 /** * ISC Detail */ isc: Detail
}
```

- `Detail` : É o modelo que informa o que foi calculado pela API.

> Nele é possível rastrear o método de calculo utilizado, e os seus parâmetros.

> O campo `jurisdictionName` depende do `jurisdictionType` , para isso basta seguir a regra:
>
> - **City**: `Emitter.address.cityName` ;
> - **State**: `Receiver.address.state` ;

- **Country**: `"Brasil"`

```
interface Detail {
 /** * Jurisdiction Type */ jurisdictionType: "City" | "State" | "Country";
 /** * Jurisdiction Name */ jurisdictionName: string
 /** * Tax Type */ taxType: "IEC" | "IST" | "ISC"
 /** * Scemario */ scenario: string;
 /** * Rate */ rate: number;
 /** * Tax */ tax: number;
 /** * Calculation Base */ calcBase: number;
 /** * Fact */ fact?: number;
 /** * Month */ month?: string;
}
```

- `CalculatedTaxSummary` : É o resumo de tudo que foi calculado nos `CalculatedTax` de cada `Line` .

> Basicamente é reunir tudo que foi calculado e somar.

- `subtotal` : $\sum (Line.lineAmount - Line.lineDiscount)$
- `totalTax` : $\sum (Line.calculatedTax.tax)$
- `grandTotal` : $subtotal + totalTax$

```
interface CalculatedTaxSummary {
 /** Count of Lines */ numberOfLines: number;
 /** * Sub Total */ subtotal: number;
 /** * Total Tax */ totalTax: number;
 /** * Grand Total */ grandTotal: number;
 /** Tax By Type */ taxByType: TaxByTypes;
}
```

- `TaxByTypes` : É o modelo que agrupo os detalhes dos três impostos.

```
interface TaxByTypes {
 /** * IEC Summary */ iec: TaxSummary
 /** * IST Summary */ ist: TaxSummary
 /** * ISC Summary */ isc: TaxSummary
}
```

- `TaxSummary` : É o modelo que soma todos os impostos e agrupa as jurisdições.

```
interface TaxSummary {
 /** * Tax */ tax: number;
 /** * Jurisdictions */ jurisdictions: Jurisdiction[];
}
```

- `Jurisdiction` : É o modelo que agrupa jurisdições iguais e soma os impostos.

```
interface Jurisdiction {
 /** * Jurisdiction Type */ jurisdictionType: "City" | "State" | "Country";
 /** * Jurisdiction Name */ jurisdictionName: string
```

```
 /** * Tax by Jurisdiction */ tax: number;
}
```

- `ProcessingInfo` : É o modelo que informa o tempo total do processamento dos cálculos na API.

> Além de destacar a versão da API utilizada

> O tempo informado em `duration` deve ser feito em milissegundos, deixando 3 casas decimais para informar os microsegundo.

```
interface ProcessingInfo {
 /** * Version ID */ versionId: string;
 /** * Duration */ duration: number;
}
```

4.6.2. Exemplos de Dados

> Aqui listamos exemplos de dados do `Cliente J`.

```
company = {
 "id": "75106750-1ae4-4872-9d9b-562d94ea324f",
 "code": "CLIENT_J",
 "name": "Cliente J Corp."
}
location = {
 "companyId": "75106750-1ae4-4872-9d9b-562d94ea324f",
 "code": "27227668000122", // pattern: CNPJ only number
 "email": "client@clientj.com.br",
 "federalTaxId": "27.227.668/0001-22", // CNPJ
 "stateTaxId": "12462557",
 "cityTaxId": null,
 "suframa": null,
 "mainActivity": "industry",
 "address": {
   "street": "Rua Felipe Schmidt",
   "neighborhood": "Centro",
   "zipcode": "88010-001",
   "cityCode": 4205407,
   "cityName": "Florianópolis",
   "state": "SC",
   "countryCode": 1058,
   "country": "BRA",
   "number": "563",
   "complement": "sala 318",
   "phone": "4834566121"
 }
items = [{
 "companyId": "75106750-1ae4-4872-9d9b-562d94ea324f",
 "code": "VENTILADOR-DIGITAL-001",
 "description": "Ventilador Digital Valitana",
 "productType": "product",
 "federalTax": {
```

```
    "IEC": {"rate": 0.0321, "fact": 0.09}, // Rate 3.21% | Fact 9%
    "IST": {"rate": 0.0412, "fact": 0.10}, // Rate 4.12% | Fact 10%
    "ISC": {"rate": 0.0650, "fact": 0.11} // Rate 6.50% | Fact 11%
  }, {
  "companyId": "75106750-1ae4-4872-9d9b-562d94ea324f",
  "code": "JJJ-LEGAL-032",
  "description": "JJJ Tabajara",
  "productType": "merchandise",
  "federalTax": {
    "IEC": {"rate": 0.0200, "fact": 0.00}, // Rate 2.00% | Fact 0%
    "IST": {"rate": 0.1525, "fact": 0.08}, // Rate 15.25% | Fact 8%
    "ISC": {"rate": 0.0450, "fact": 0.01} // Rate 4.50% | Fact 1%
  }
}]
```

> Exemplos de **Request** e **Response** de transações, utilizando os dados acima, podem ser baixados pelo pelo Google Drive: https://drive.google.com/drive/folders/1dfK-dc3SElRRHaRdZVmF09YdvhWvht65?usp=sharing.

4.7. Ambiente de Desenvolvimento

O Projeto deve ter duas divisões:

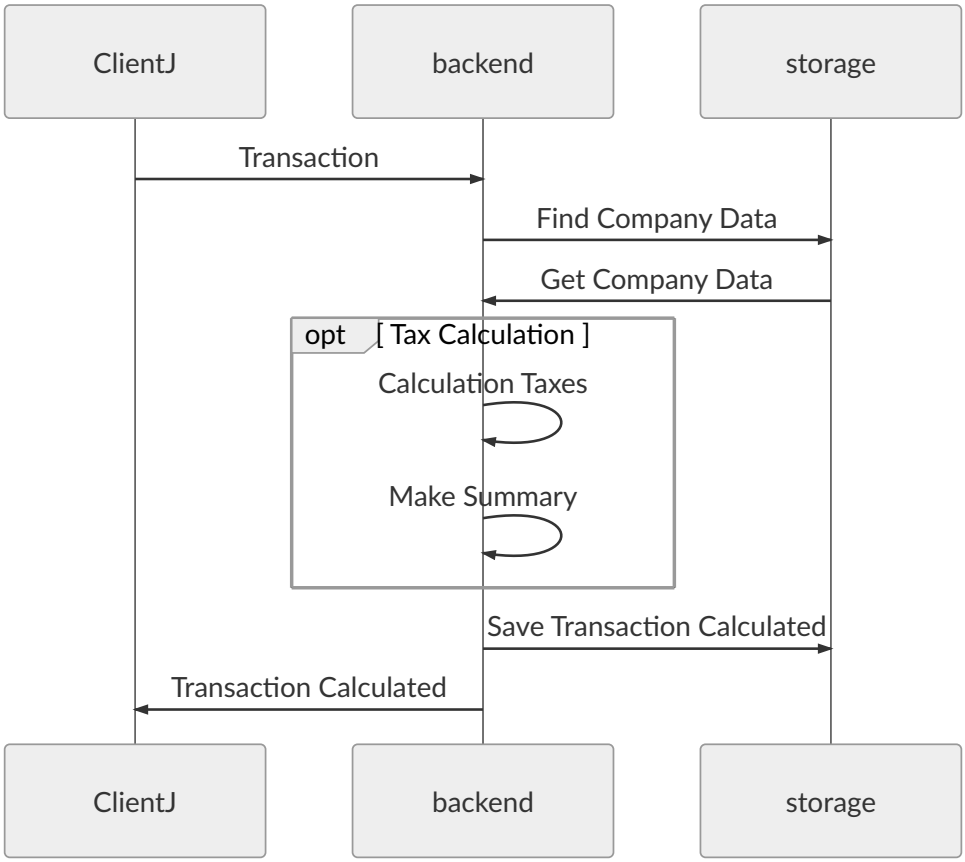
- **backend**: Deverá ser implementado usando `dotnet core` ou `nodejs`.
- **frontend**: Deverá ser implementado utilizando **React**.

O Projeto de Cálculo de Impostos será validado numa instância Linux t2.large do AWS.

4.7.1. backend

O **backend** é a parte principal do Projeto. É nele que é realizado os cálculos de impostos.

Para facilitar a vida do Candidato, criamos esse singelo diagrama de sequencia:



No diagrama é fácil de reparar que o **backend** vai ser uma API, e que será consumida pelo ERP do `Cliente J`.

Ela é síncrona. E a parte do `Tax Calculation` não deve superar 200 milissegundos.

> Dica: usar cache de dados com o Redis.

Sendo que todo o processo todo (até o `Cliente J` receber a transação calculada) não deve superar os 600 milissegundos.

> O `Cliente J` executará as chamadas da API do localmente. Então o candidato não precisa se preocupar com a latência da rede (ou da internet).

4.7.2. frontend

No **frontend**, o Candidato de implementar uma tela de listagem de transações, que foram enviadas e calculadas pelo **backend**.

Essa tela de listagem de ser construida em **React**, e nela deve ser possível:

- [ ] Paginar das transações calculadas;
- [ ] Ordenar as transações por data de envio;
- [ ] Filtrar as transações por Empresa;
- [ ] Visualizar as informações de uma transação;

> CRUD dos Modelos de Dados serão considerados diferenciais do Candidato, podendo serem usados para critério de desempate.

4.7.3. Pull Request Final

Ao finalizar o projeto o candidato deve:

- [ ] Criar um Pull Request para atualizar a versão para **1.0.0** (semver);
- [ ] Adicionar o `Cliente J` como revisor do Pull Request.
- [ ] Destacar um **Script Bash** para configurar o ambiente e instala-lo no Linux do AWS.

> Esse **Script Bash** vai executado no servidor do AWS. Os Bancos de Dados já estarão instalados localmente na máquina. Assim como as últimas versões do `dotnet core` e `nodejs`.

O **Script Bash** deve:

- [ ] Baixar o projeto do GitHub;
- [ ] Instalar as dependências;
- [ ] Criar um serviço para executar o projeto;
- [ ] Configurar o banco de dados;

O candidato deve informar no `readme.md`, os comando para:

- [ ] Iniciar o serviço
- [ ] Pausar o serviço
- [ ] Reiniciar o serviço
- [ ] Acompanhar o log do serviço

> Quanto mais completo o `readme.md` melhor para o `Cliente J` implantar o Projeto Cálculo no AWS.

5. Por Fim ...

O `Cliente J` poderá (se necessário) fazer três pedidos de modificações no Pull Request final.

> O Candidato terá um dia para arrumar cada pedido.

Se o `Cliente J` aprovar o Pull Request, significa que o Candidato está qualificado para a próxima etapa do processo de seleção.

Então, Boa Sorte!