# [EPD – Engineering Planning Document] Desenvolvimento de aplicação para tratamento de dados

**Autores:**

* Anderson Bolivar Nascimento
* Anselmo Berriel de Lira

**Links de Referência:**

* **DMOT Design Pattern for ETL (Data Model, Orchestrator, Transformer): https://towardsdatascience.com/dmot-a-design-pattern-for-etl-data-model-orchestrator-transformer-c0d7baacb8c7**

Links relacionados a este documento

# Linguagem Python

[O que é Python?](https://www.treinaweb.com.br/blog/o-que-e-python)

[Quais as diferenças entre tipagens](https://www.treinaweb.com.br/blog/quais-as-diferencas-entre-tipagens-estatica-ou-dinamica-e-forte-ou-fraca/)

O Python é uma linguagem de programação de alto nível, interpretada e multiparadigma, ou seja, suporta diversos paradigmas de desenvolvimento. Além do mais é uma linguagem de tipagem dinâmica, pois o tipo dos dados é definido de acordo com o valor atribuído à variável em tempo de execução dinamicamente.

[Benefícios](https://www.treinaweb.com.br/blog/quais-as-diferencas-entre-tipagens-estatica-ou-dinamica-e-forte-ou-fraca/) de se utilizar Python para a análise de dados

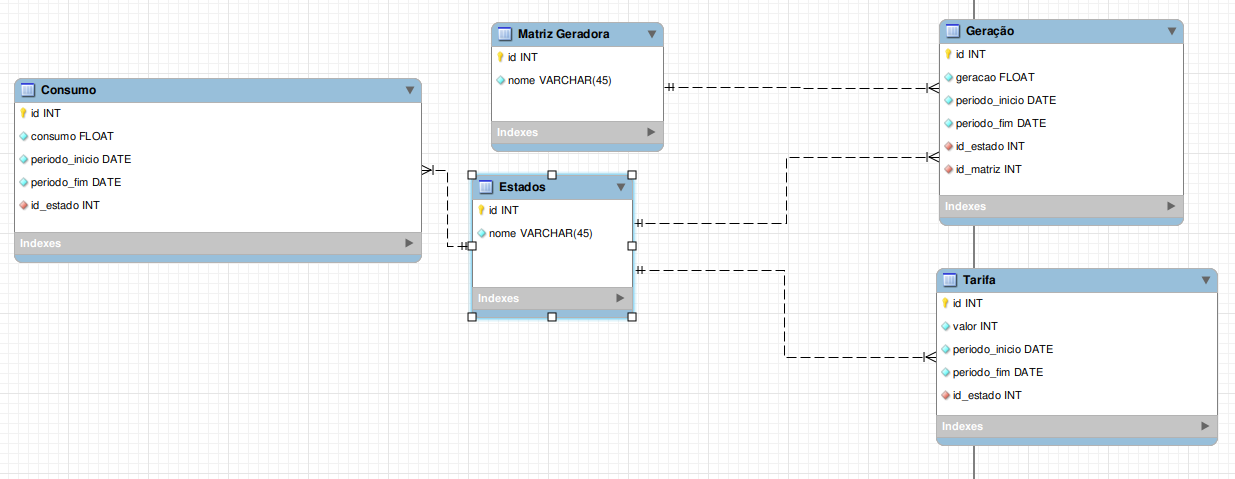
O Python, além de orientado a objetos (o que permite utilizar a modelagem baseada em negócios, possui uma ampla gama de bibliotecas e ferramentas para modelagem de dados, análise preditiva e estatística, desenvolvidas pela comunidade. Aliado a isto, o fato de ser uma linguagem interpretada possibilita desenvolver e corrigir erros enquanto se executa o código, diferente de Java e C++, onde o código deve ser compilado antes de ser executado e testado.

# Banco de Dados MySQL

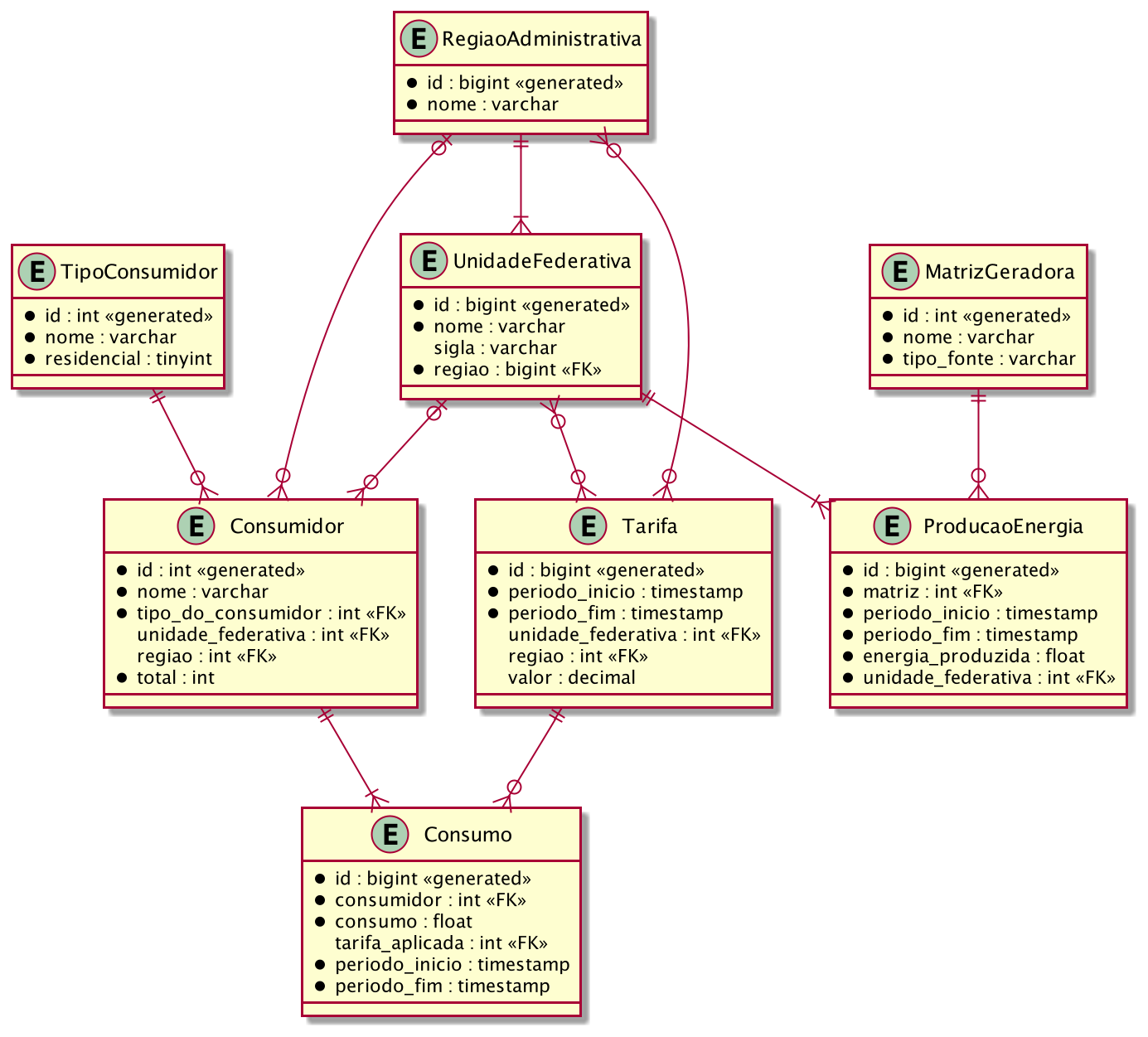
[Primeiros Passos no MySQL](https://www.devmedia.com.br/primeiros-passos-no-mysql/28438)

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (Structured Query Language ou Linguagem de Consulta Estruturada) como interface.

**Modelo de Dados:**



Outra sugestão pro modelo de dados com base no diagrama de classes mais abaixo:



# Arquitetura MVC

O MVC é um padrão de arquitetura de software utilizado em muitos projetos devido a arquitetura que possui, o que possibilita a divisão do projeto em camadas muito bem definidas. Cada uma delas, o **Model**, o **Controller** e a **View**, devem respeitar o princípio de modularidade, desacoplamento e encapsulamento entre camadas.

**Model:**

[**Connecting to MySQL Using Connector/Python**](https://dev.mysql.com/doc/connector-python/en/connector-python-example-connecting.html)

Camada responsável por gerenciar e controlar a forma como os dados se comportam por meio das funções, lógica e regras de negócios estabelecidas. Nessa cama comunicaremos com o banco e entregaremos os dados a partir de funções que farão consultas diretas no banco.

* 📁 model
  + 📁 database
    - 📄 connection.py
    - 📄 consumption.py
    - 📄 rate.py
    - 📄 generation.py
    - 📄 states.py
    - 📄 matriz.py
  + 📁 src
    - database\_a.py
    - database\_b.py

No diretório database, lidaremos com a montagem e consulta da nossa base de dados. O arquivo connection.py fará a instância do MySQL no projeto, utilizando a lib mysql. O restante dos arquivos possuirão funções que se comunicarão com o banco.

No diretório src, faremos a extração e limpeza das bases de dados externas escolhidas, utilizando a lib requests, teremos um arquivo para cada base de dados.

**Controller:**

O controller é a camada responsável por intermediar as requisições enviadas pelo View com as repostas fornecidas pelo Model. Nele isolaremos toda nossa regra de negócio, ou seja, as análises que faremos de transformação de dados.

* 📁 controller
  + 📄 index.py
  + 📄 reports.py
  + 📄 update.py

O arquivo index.py vai apenas exportar os outros dois arquivos da camada. O reports.py fica concentrado toda a lógica de análise de dados que faremos, o relacionamento entre as entidades, é interessante que essas funções tenha teste e comentários descrevendo os seus parâmetros e resultado. O update.py será o arquivo que vai popular o banco de dados com os dados brutos extraídos e limpados pela camada model, esse arquivo deve ser interpretado apenas uma vez durante o ciclo de vida da aplicação.

**View:**

Camada que irá apresentar as informações de forma visual ao usuário.

Existem várias bibliotecas do Python de visualização de dados, devemos escolher qual atende melhor os nossos critérios:

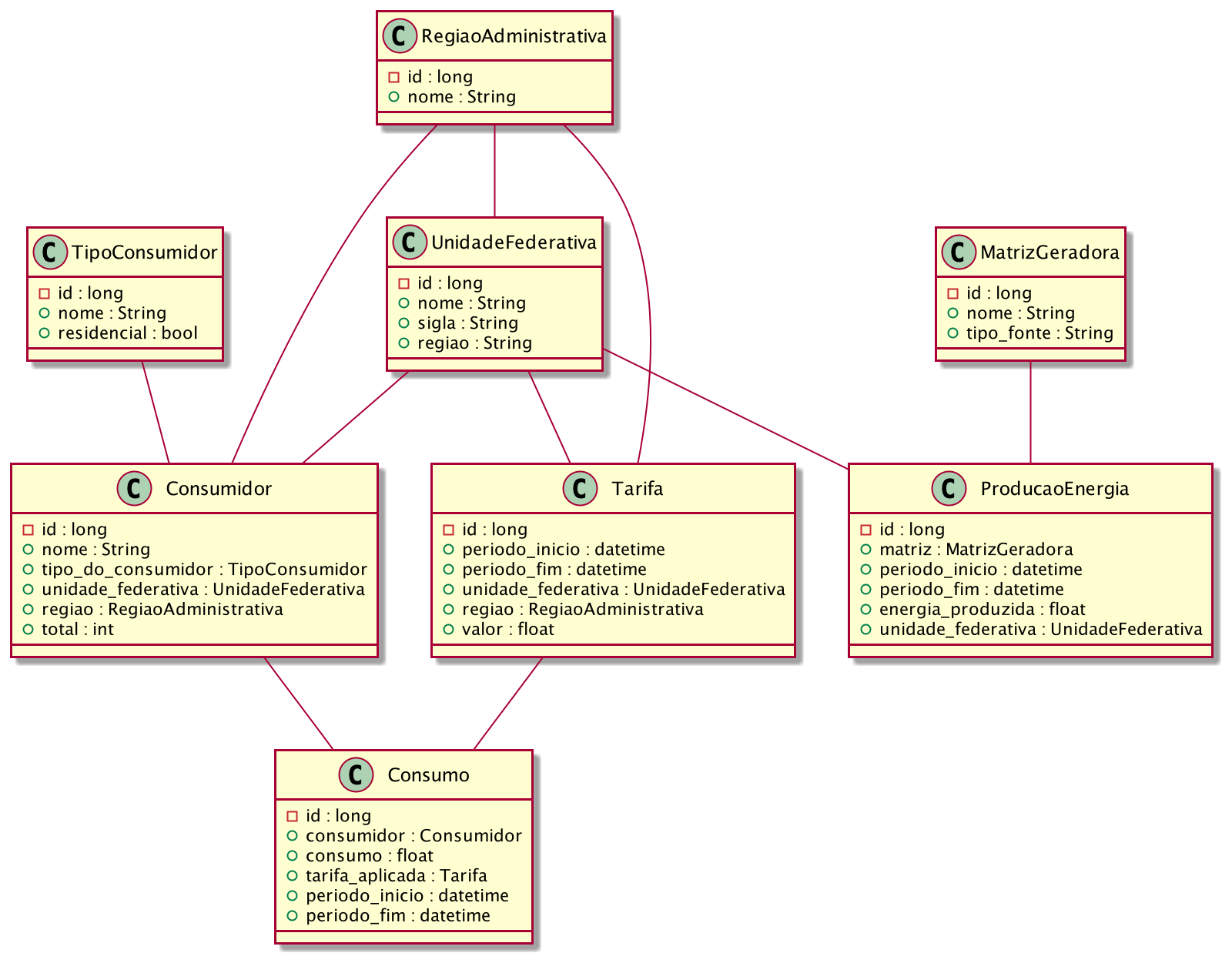
# [**Python Data Visualization Libraries for Business Analytics**](https://mode.com/blog/python-data-visualization-libraries/)

Caso necessário, podemos usar também o *inquirer* para interação com o usuário via CLI.

* 📁 view

Modelo de Classes

Abaixo, segue o modelo de classes da aplicação, o qual dará origem ao banco de dados com os dados agrupados e também tratados.



Segue o dicionário de dados:

* RegiaoAdministrativa
  + Definição: aqui temos o cadastro das regiões administrativas do Brasil (Norte, Nordeste, Sudeste, Centro-Oeste e Sul). Como alguns dados de consumo estão catalogados por região, esta classe/tabela receberá esses dados logo após o tratamento.

|  |  |  |
| --- | --- | --- |
| Coluna | Tipo | Definição |
| id | long | Identificador único do registro no sistema |
| nome | String | Nome da região administrativa |

* UnidadeFederativa
* TipoConsumidor
  + Definição: aqui temos o registro das categorias de consumidores por tipo de usuário (pessoa física - residencial – ou pessoa jurídica - setores produtivos ou comércio).
* Consumidor
  + Definição: aqui temos o registro dos tipos de consumidores por região e a quantidade presente. Esta segregação por tipo permite registrar o histórico por cada tipo e também para todos.
* Tarifa
  + Definição: aqui temos o histórico de tarifas aplicadas sobre energia consumida por período de tempo (dias, horas, meses, etc) e região.
* Consumo
  + Definição: aqui temos o histórico de consumo por tipo de consumidor, região e a tarifa aplicada na data/período do consumo registrado e região do consumidor.
* MatrizGeradora
  + Definição: aqui temos o registro das matrizes geradoras de energia do país e a definição do tipo da mesma.
* ProducaoEnergia
  + Definição: aqui temos o registro histórico de produção energética por matriz e região (pode ser unidade federativa ou região administrativa, pois nas fontes de dados podemos encontrar nos 2 formatos).

Diagrama de Atividades

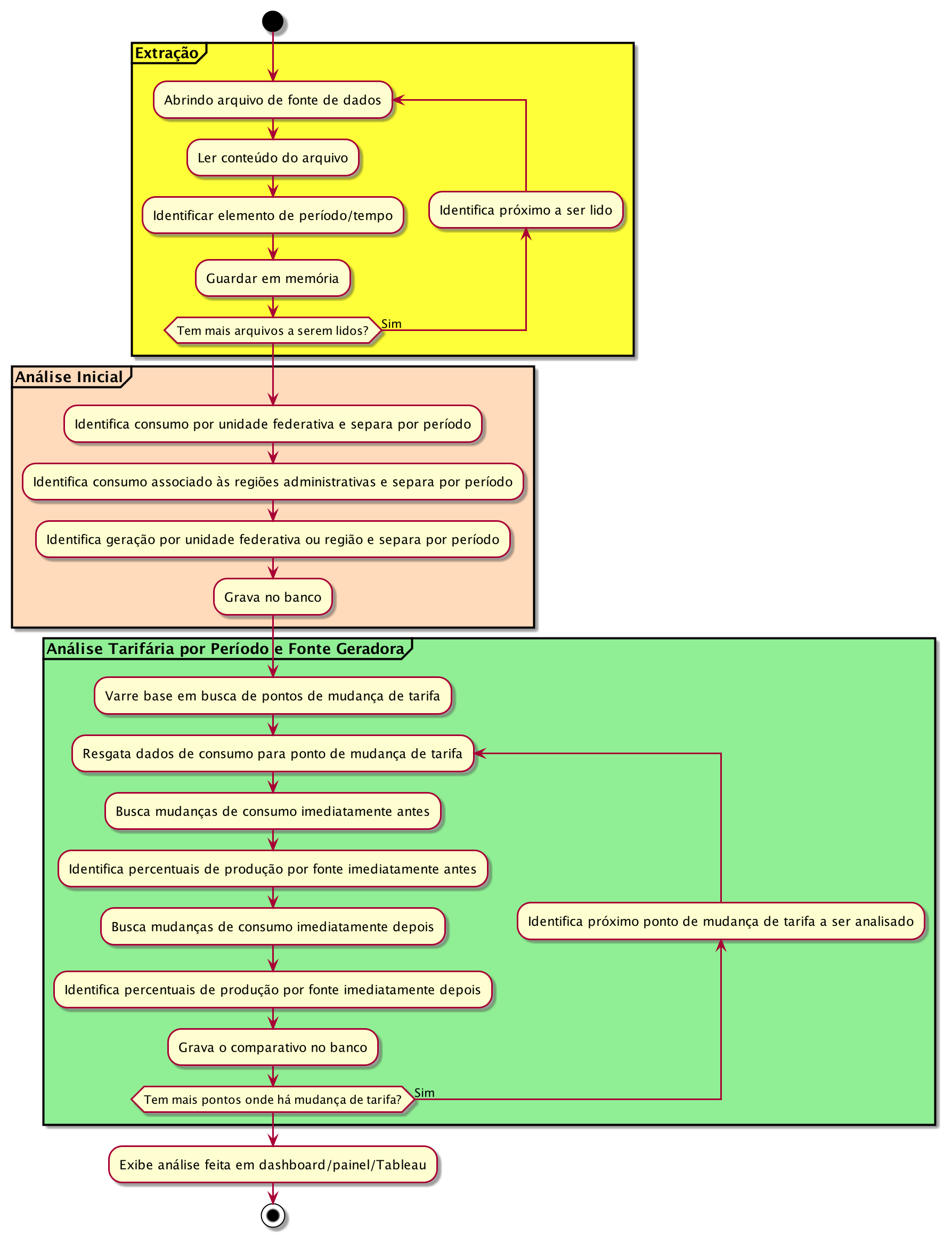


Diagrama de Infraestrutura

