**Documentação Desafio PubFuture**

**Nome:** Bruno Ferrari Vincensi

**Última mudança:** 13/01/2022

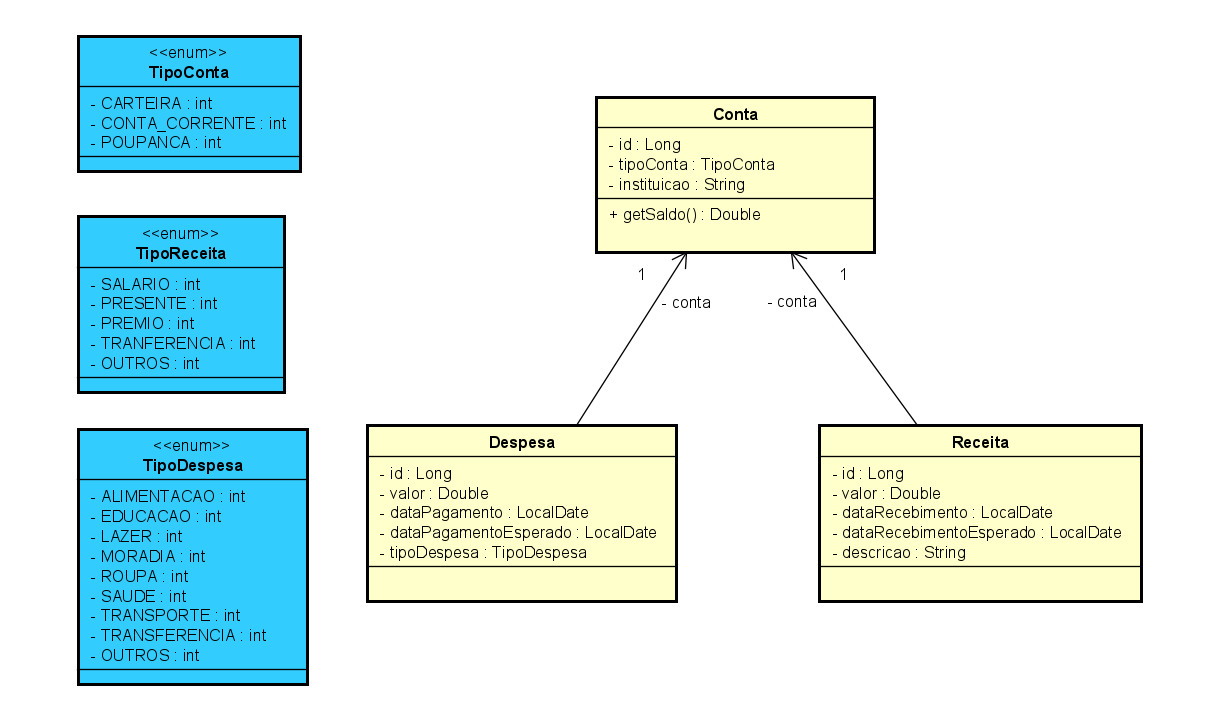
**Sobre**

Projeto com a função de gerir finanças pessoais, sendo uma Api Rest criada pelo Spring Boot(Java) na parte servidor e consumida pela tecnologia front-end ReactJS na parte do cliente

**Tecnologias utilizadas:**

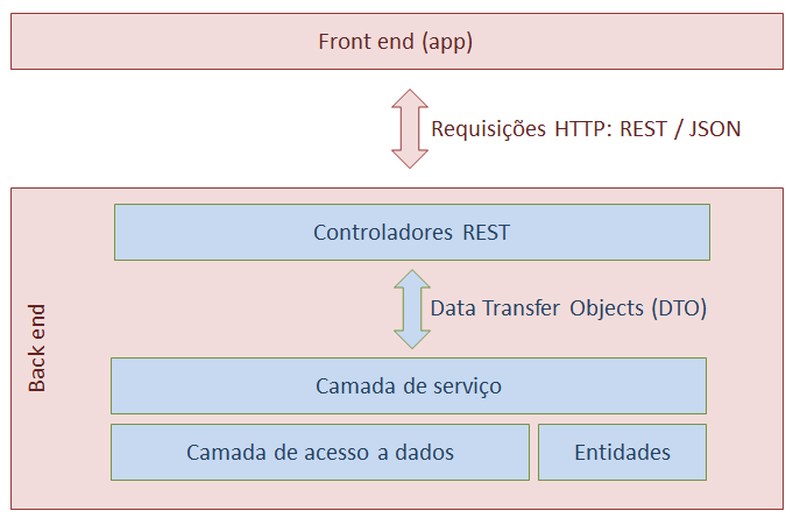
* Spring Boot (Java)
* MySQL
* ReactJS (Java Script, HTML, CSS)
* Bootstrap
* GIT | GitHub
* Intellij - IDE
* Postman | banco H2 | JUnit - para teste

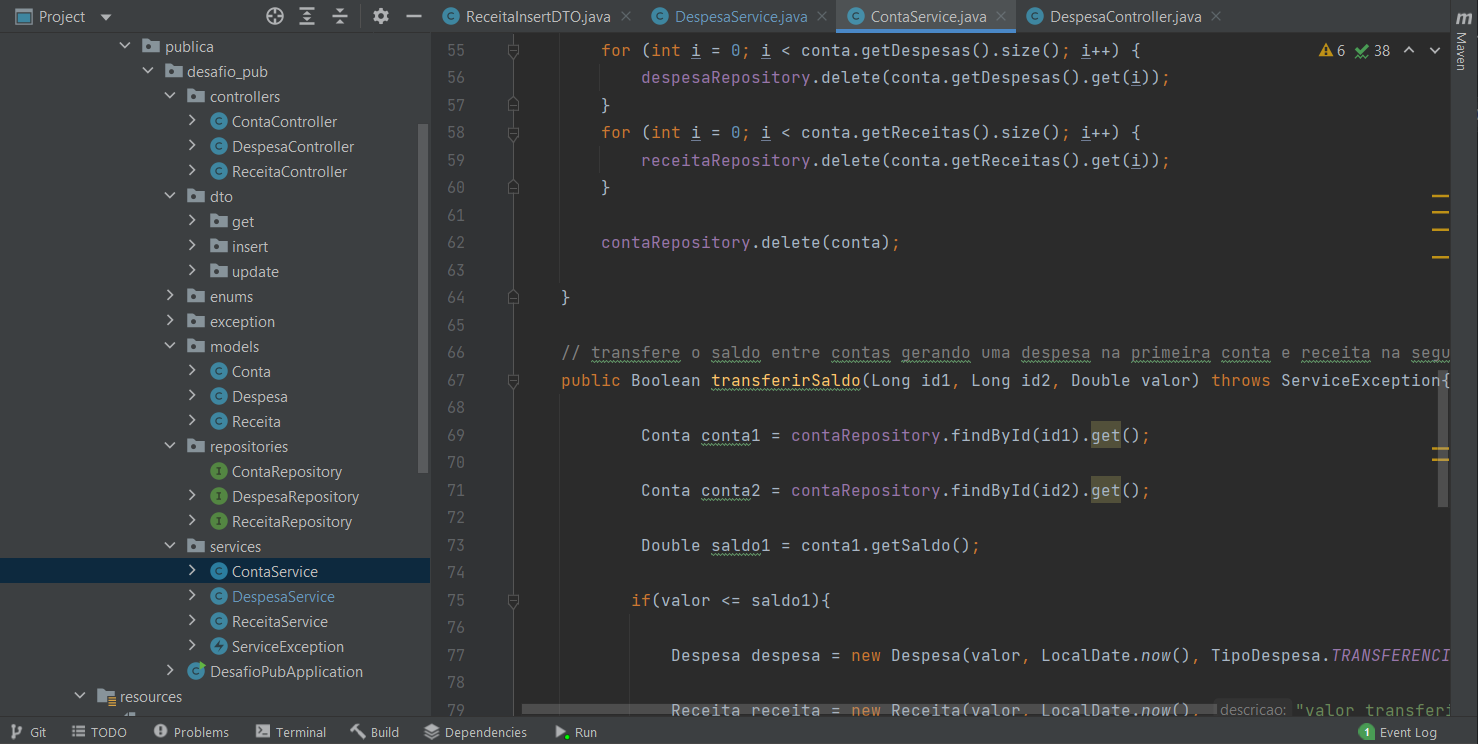
## **Diagrama UML (Linguagem Unificada de Modelagem)**



**Padrão de desenvolvimento**

* **Padrão camadas – se assemelha muito com o padrão MVC por conter a mesma ideia de ter model, view e controller**





Entidades (models): Refere-se às classes que fazem ligação com as tabelas do banco de dados relacional

Camada de acesso a dados (repositories): Faz requisições diretas para o banco de dados realizando as Querys

Camada de Serviço (services): Realiza os serviços e as regras de negócio de cada entidade

Data Transfer Ojects (dtos): Agrupa um conjunto de atributos de uma entidade numa classe simples de forma a otimizar a comunicação para as outras camadas

Controladores REST (controllers): São responsáveis por fazerem a chamada http e requisições dos endpoints (URI), realizando assim uma API Rest

**Funcionalidades**

Contas

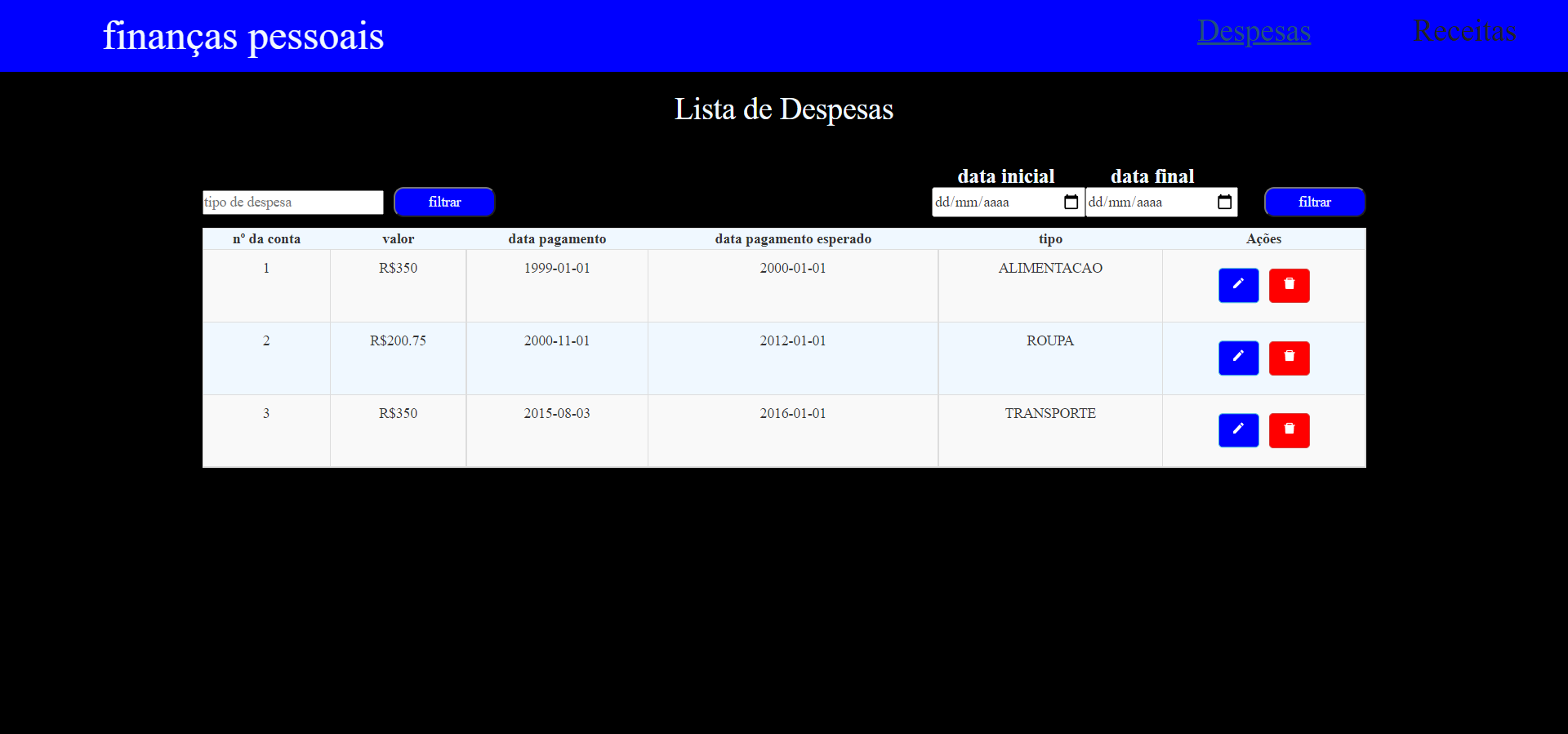
* Métodos CRUD
* Listar saldo total
* Transferir saldo entre contas





Despesas / Receitas

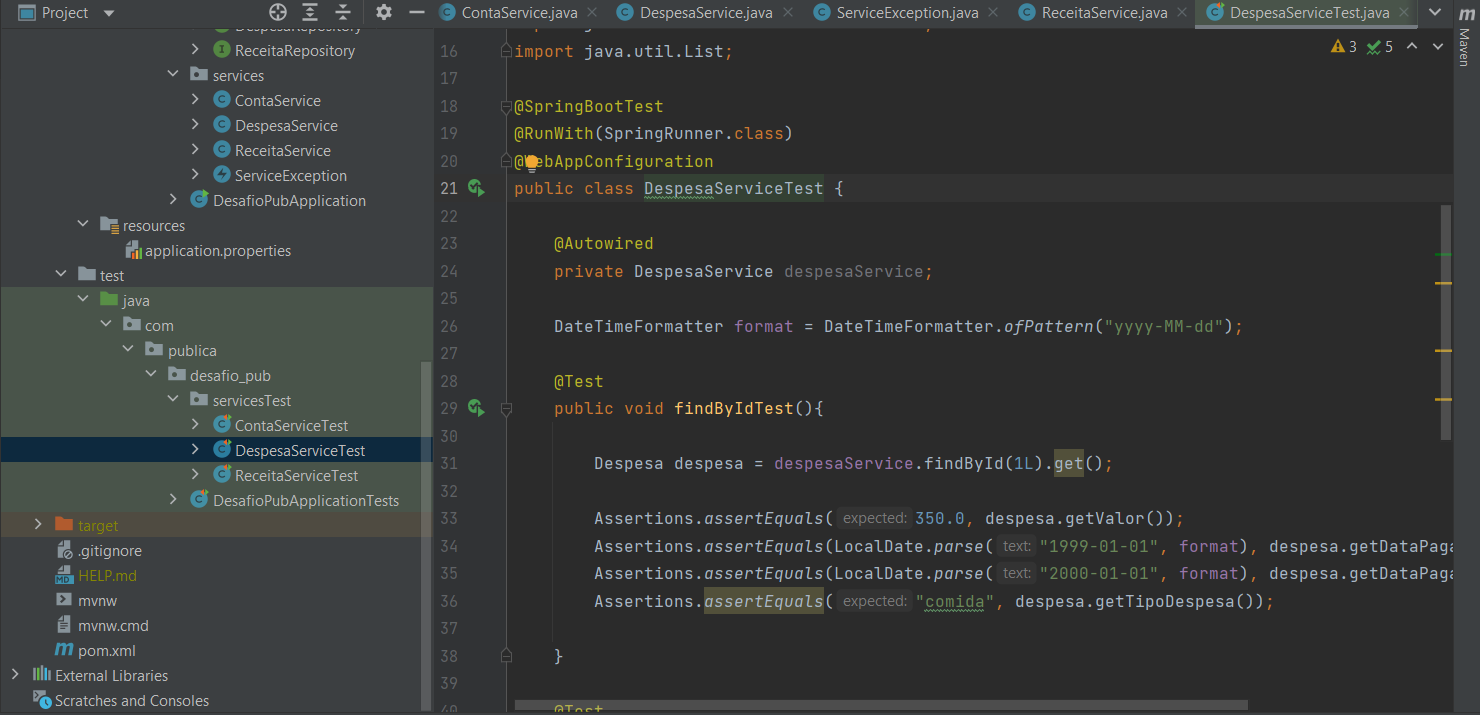
* Métodos CRUD
* Filtro por tipo
* Filtro por período (dataInicial - dataFinal)
* Listar despesa/saldo total

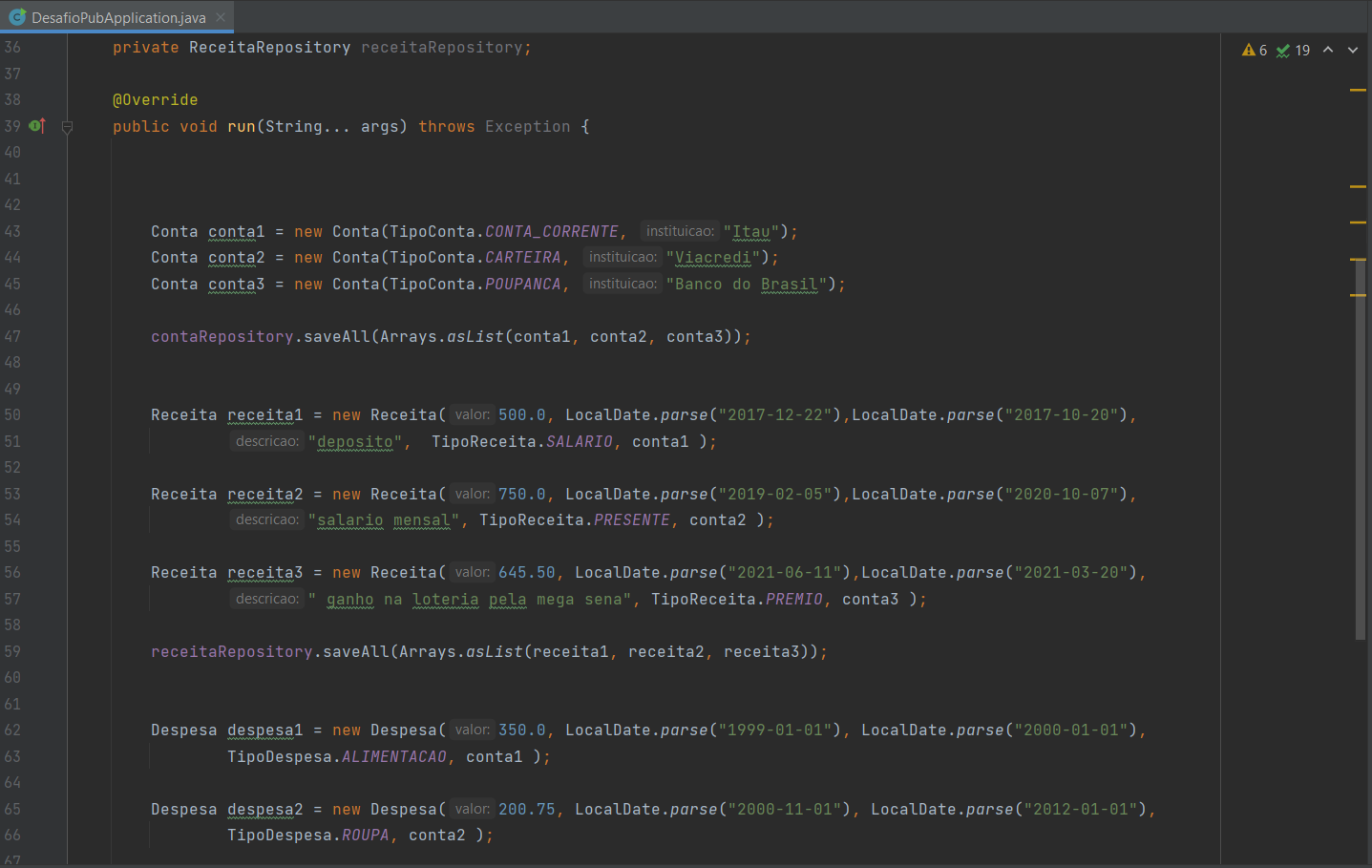




**Testes**

Foi utilizado o JUnit para realizar os testes unitários e para popular os dados é criada algumas entidades na hora de subir a aplicação

****

****

ReactJS

- Biblioteca do Java Script para desenvolver uma interface de usuário (IU) sólida e single-page.

- Para consumir a Api Rest foi utilizado o axios, cuja a função é servir como cliente recebendo as requisições das URIs do servidor

