Sistema de transferências financeiras

Introdução

Foi gerado uma api para atender a demanda de agendar e consultar transferências.

Ferramentas e Frameworks utilizados

Spring Boot

Como referência de facilidade e agilidade e vastamente utilizada no mercado, a api foi criado com o projeto psringboot

Spring Data

Também visando agilidade foi integrado ao projeto o mais uma ferramenta da Spring, a implementação de JPA do Spring.

H2 DataBase

Como banco de dados em memória foi escolhido o H2, além de ter uma gama diversificada de implementações de funções de vários bancos, possui um console que pode ser configurado e exposto para acesso direto sem precisar de uma IDE externa.

Swagger

Para facilitar o consumo e teste de desenvolvimento dos serviços, foi incrementado no projeto o Swagger, também uma ótima ferramenta para auxílio no desenvolvimento de API'S

-

Angular **

Como front end, foi escolhido Angular na versão 5, além de ser um framework em javascript poderosíssimo e amplamente utilizado no mercado, também podem ser incluídas ferramentas para aumentar a produtividade assim como o CLI do próprio Angular.

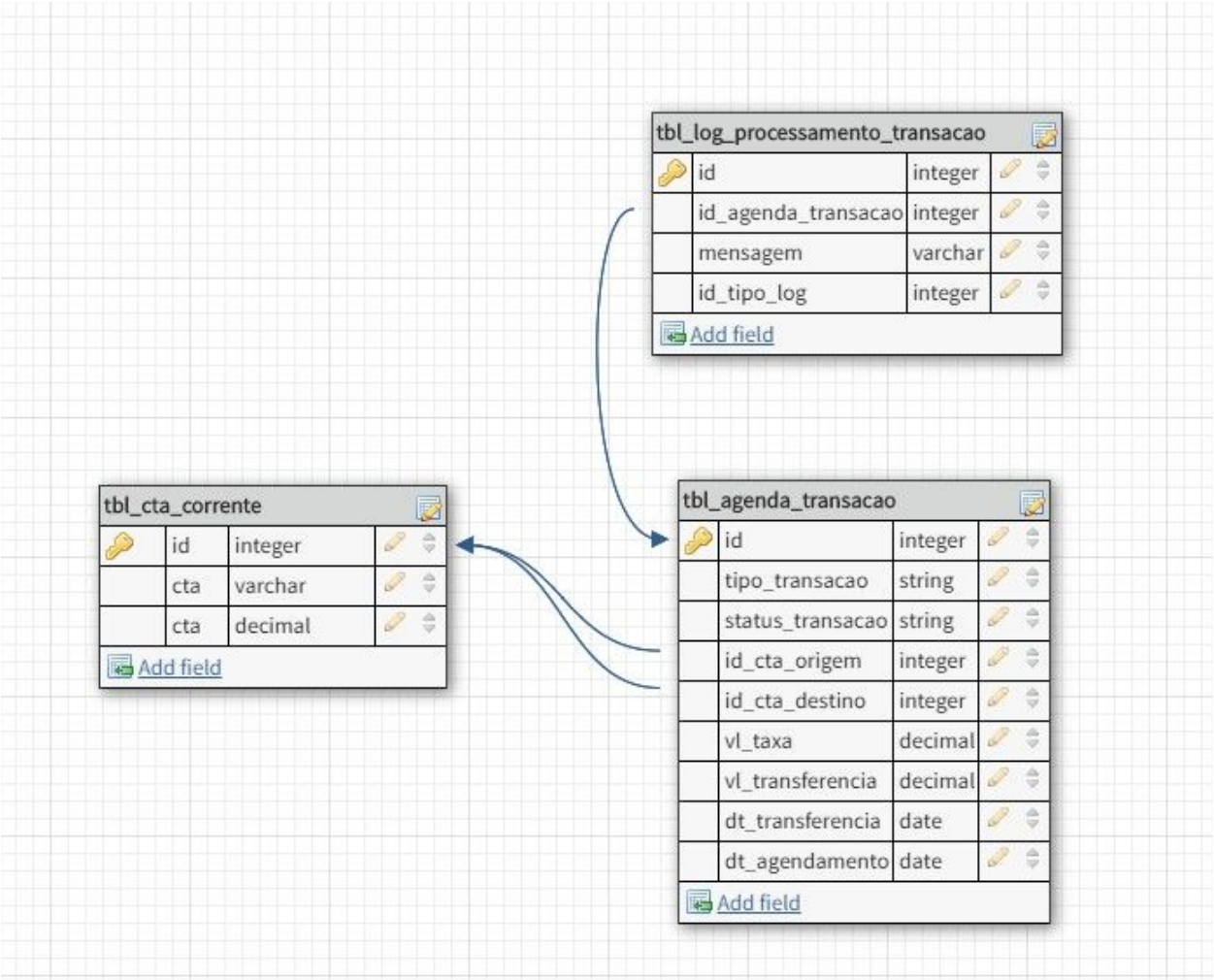
Arquitetura

Segurança

Não foram incluídos requisitos de segurança no projeto, porém como sugestão caso a segurança seja gerenciada pelas apis e não via barramento o qual pode abstrair esta camada, tenho como recomendação de autenticação statless, o par Spring Security + JWT.

Modelagem de Dados

A modelagem foi feita através da ferramenta dbdesigner.net, para esta solução com intuito de simplificar o processo, não foram criadas tabelas de domínio para manter status e classificação de alguns registros, porém nos cenários que foram necessários existir um campo deste tipo foi criado uma constraint com os possíveis valores.



img 01

A ferramenta utilizada não possui todas as tipagens de dados, alguns campos apresentados na figura, podem não representar a tipagem real.

A primeira tabela é “tbl_agenda_transacao” responsável por armazenar os dados de uma transferência agendada, para que posteriormente ela possa ser processada por um segundo processo, o qual pode ser um batch ou qualquer outro orquestrador. Nesta tabela como campos de controle temos status transação, que guarda qual é o status atual daquele registro (“Aguardando Transferência”, “Transferido com sucesso”, etc), logo na tabela01 podemos ver todas os possíveis status, neste mesmo formato guardamos o tipo da transação (“A”, “B”, “C”).

Valor	Descrição
AT	Aguardando Transação, status que identificar que este registro está aguardando seu processamento
EP	Em processamento, status que identifica um registro no instante de seu processamento
TS	Transação realizada com sucesso, status que identifica os registros já processados com sucesso
FT	Falha na transação, status que indica que o processamento dessa transação falhou
AR	Aguardando reprocessamento, status responsável por enfileirar registros para um novo processamento
FS	Falha por falta de saldo, status que indica falha de uma transação por falta de saldo na conta de origem.
...	Além destes, podem ser criando mais status com as mais diversas situações para facilitar uma análise mais critica do processamento de cada registro

tabela01

A segunda tabela é “tbl-cta_corrente”, esta tabela é responsável por guardar a conta corrente, para verificarmos se a mesma existe (Este cenário não contempla transação entre entidades externas, para abstrair pode-se adotar uma estratégia diferente) através do campo varchar(6) cta.

E por fim a terceira tbl_log_processamento_transacao, esta tabela tem o intuito de auxiliar o orquestrador a processar os registros, cada registro desta tabela guarda a data de início, data fim do processamento do registro de um único registro, a solução foi sugerida unitária ao invés de lotes, para que o serviço que processe cada registro de forma independente, assim podendo usufruir dos benefícios de escalabilidade. Nesta bela também guardamos e validamos em forma de constraint o status das transações (tabela02).

WebService

Como webservices, foi adotada a exposição de serviços REST
Temos um único serviço com trê operaçõe.,

O serviço tem o dominio “/apiagendamento-transfencia-financeira/” o mesmo é responsável por expor tarefas referente ao agendamento de transação.

Dentre as três operações temos:

- listaTodosAgendamentos (tipo GET)
  Exposto como domínio + “page/{pagina}/{quantidade}”, nesta operação temos uma listagem geral e paginada o qual retorna um object Response contendo um *org.springframework.data.domain.Pageable* para auxiliar na paginação dos registros quando consumido através de um front-end;
- buscaPorId (tipo GET)
  Exposto como domínio + “/id/{id}”, nesta operação passamos como filtro de busca o parâmetro id;
- agendarTransferenciaFinanceira (tipo POST)
- Exposto como domínio + “/agendar””, esta operação tem como funcionalidade realizar o agendamento através dos parâmetros informados pelo usuário.

Instruçoes de uso

A api foi criado como um projeto maven, para que a mesma seja executada temos duas opções
1 - Importar o código no eclipse e executar o projeto
2 - Garantindo que existe o maven instalado na máquina pode-se executar o comando “mvn clean install” na raiz do projeto, executar o comando java -jar no artefato gerado.

Para geração do banco de dados, está sendo criado a partir de uma opção do h2, o qual temos dois scripts (schema.sql e data.sql) que criam a estrutura e inserem alguns registros.

Considerações finais

Antes de tudo agradeço por participar deste processo seletivo, peço desculpas por somente conseguir enviar no último dia. Gostaria de ter implementados mais funcionalidade para que meu código pudesse ser avaliado de forma mais ampla, porém como retornei a minha rotina de trabalho e overtime tive somente algumas horas picadas para codificar o desafio, o qual ficou um pouco defasado em questão de testes unitários.