UFPR - TADS
Trabalho de DS152 - DAC
BANTADS - Internet Banking do TADS

Requisitos Funcionais

O objetivo deste trabalho é o desenvolvimento de um sistema de Internet Banking usando Angular e Java Spring, baseado na arquitetura de microsserviços.

O sistema possui 3 perfis de acesso:

- **Cliente**: usuários com esse perfil são os clientes do banco BANTADS;
- **Gerente**: usuários com esse perfil são gerentes de contas dos clientes;
- **Administrador**: usuários com esse perfil são administradores do sistema.

Os requisitos funcionais são apresentados a seguir (CRUD significa - Inserir, Remover, Atualizar e Listar todos).

PERFIL CLIENTE

- **R0: Autocadastro** - Uma pessoa só se torna cliente do BANTADS a partir do autocadastro, que é feito a partir da página inicial, sem necessitar login. Uma das informações é o salário mensal do cliente, que é usado para calcular seu limite. Se o cliente ganha R$ 2.000,00 por mês, ou mais, então tem direito ao limite em sua conta, que é exatamente igual a metade do seu salário. Ao se cadastrar, uma conta é automaticamente criada, que é identificada pelo número da conta, mas estará pendente de aprovação. O gerente responsável pela sua conta será o que possuir menos clientes e será atribuído automaticamente pelo sistema, visto que o gerente precisa aprovar a criação da conta. Cada cliente só pode ter uma conta no BANTADS;
- **R1: Login/Logout** - Login com e-mail/senha, todas as demais funcionalidades não podem ser acessadas sem um login com sucesso.
- **R2: Tela Inicial de Cliente** - Deve apresentar um menu, com as operações que podem ser efetuadas pelo cliente, e seu saldo atual. Se estiver negativo deve ser com sinal de "-" e em vermelho;
- **R3: Depositar** - Para simplificar, vamos assumir que ao fazer o depósito o dinheiro real é adicionado na conta e, portanto, as bases de dados podem ser atualizadas;
- **R4: Saque** - Para simplificar, vamos assumir que ao fazer um saque o dinheiro é retirado da conta e está disponível para o cliente e, portanto, as bases de dados podem ser atualizadas. Só pode haver saque se houver saldo suficiente na conta, contando com o limite do cliente;
- **R5: Transferência** - O cliente informa o número da conta corrente destino e saldo para transferência. A operação é automática e é registrada naquela data/hora em que foi solicitada;

- **R6: Consulta de saldo atual** - O cliente pode, a qualquer momento, consultar seu saldo atual, mostrando também seu limite;
- **R7: Consulta de extrato** - O cliente pode, a qualquer momento, consultar seu extrato, informando a data de início e a data de fim. Devem ser apresentados: data/hora da transação, operação (transferência, depósito, saque), cliente origem/destino (preencher em caso de transferência), valor. Se o valor for de saída (transferência ou saque) o registro deve ser mostrado todo em vermelho. Se a operação for de entrada (transferência ou depósito) o registro deve ser mostrado todo em azul. A cada dia, desde a data inicial, deve-se mostrar o saldo consolidado naquele dia, mesmo que não tenha movimentações;

PERFIL GERENTE

- **R8: Tela Inicial do Gerente** - Deve apresentar todos os pedidos de autocadastro para aprovação, como uma tabela, contendo o CPF, nome e salário do cliente, junto com dois botões "Aprovar" e "Recusar". Em caso de aprovação, um e-mail contendo a senha (aleatória) do cliente deve ser enviada para seu e-mail. Em caso de reprovação, deve-se cadastrar um motivo e um e-mail também deve ser enviado para o cliente, contendo o motivo da reprovação. A data/hora em que a aprovação/reprovação for efetuada também deve ser armazenada;
- **R9: Consultar Clientes** - Deve apresentar em uma tabela todos os seus clientes, contendo CPF, Nome, Cidade, Estado e Saldo da conta. Deve ser ordenado de forma crescente por Nome. Deve ser disponibilizado um campo de texto onde o gerente pode pesquisar o cliente por CPF (ou parte dele) e Nome (ou parte dele). Cada cliente deve possuir um link que, ao ser pressionado, vai para uma tela contendo todos os dados do cliente e de sua conta;
- **R10: Consultar Cliente** - Em uma tela em branco, o gerente deve informar em um campo de texto o CPF, o sistema deve mostrar todos os dados do cliente, incluindo os dados de sua conta;
- **R11: Consultar 5 melhores clientes** - Deve ser apresentada uma tela contendo somente os seus clientes que possuem os 5 maiores saldos em conta, mostrando CPF, Nome, Cidade, Estado e Saldo da conta. Deve ser ordenado de forma decrescente por Saldo;

PERFIL ADMINISTRADOR

- **R12: Tela Inicial** - Apresenta uma tela (pode ser em estilo dashboard) mostrando todos os gerentes do banco, para cada gerente apresenta: quantos clientes possui, a totalização (soma) de saldos positivo e a totalização (soma) de saldos negativos;
- **R13: CRUD de Gerentes** - Efetua o CRUD de gerentes.

Decomposição Por Subdomínio

Em uma análise preliminar, o sistema foi decomposto em poucos subdomínios e foram determinados os seguintes serviços a serem implementados, contendo seus dados (mínimos, podendo haver mais caso vocês detectem alguma necessidade):

- **Cliente**: responsável pela manutenção de clientes;
  - **Dados de Cliente**: Nome, e-mail, CPF, Endereço, Salário
  - **Endereço de Cliente**: Tipo (rua, avenida, etc), logradouro, número, complemento, CEP, Cidade, Estado
- **Conta**: responsável pela manutenção das contas dos clientes;
  - **Dados da Conta**: Cliente, Número da conta, data da criação, limite
  - **Histórico de Movimentações**: data/hora, tipo (depósito, saque, transferência), cliente origem/destino (quando for transferência), valor da movimentação

- **Gerente**: responsável pela manutenção dos dados de gerentes de contas;
    - **Dados do Gerente**: Nome, e-mail, CPF
- **Autenticação**: responsável pela autenticação no BANTADS.
    - **Dados de Usuário**: Cliente ou Gerente, Tipo (cliente/gerente/admin), login, senha

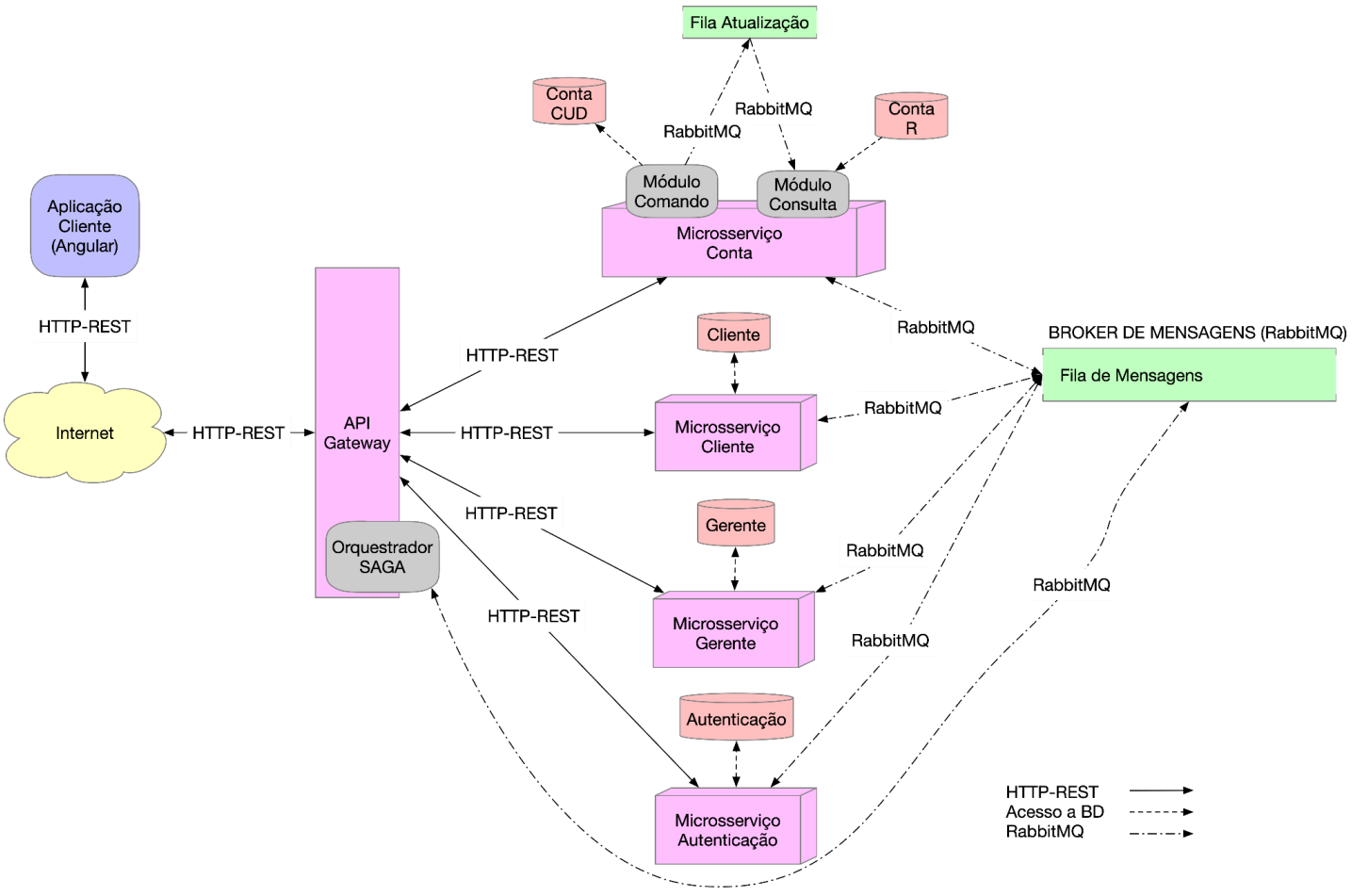
Arquitetura

O BANTADS deve ser implementado usando-se os seguintes padrões de projeto de microsserviços:

- **Arquitetura de Microsserviços**: para desenvolver o software;
- **Padrão API Gateway**: para expor a API;
- **Padrão Database per Service**: para manter os dados, sendo que cada serviço só tem acesso ao seu SGBD;
- **Padrão CQRS**: no microsserviço de Conta, para todas as consultas. A sincronização dos dois bancos de dados deve ser feita por mensageria;
- **Padrão SAGA Orquestrada**: para transações que abrangem vários serviços.

Cada microsserviço, incluindo o API Gateway e cada banco de dados, deve ser executado em uma imagem Docker separada.

A seguir uma Figura que ilustra o esboço da arquitetura que deve ser implementada.



Requisitos Não-Funcionais e Comentários

Toda e qualquer suposição, que não esteja definida aqui e que a equipe faça, deve ser devidamente documentada e entregue em um arquivo **.pdf** que acompanha o trabalho.

- Devem ser usadas as tecnologias vistas na disciplina: Angular 13 (mínimo), Node.js, Spring-boot, Spring Data JPA, PostgreSQL, Docker, e também as tecnologias apontadas como Pesquisa, RabbitMQ para mensageria;
- Os microsserviços são independentes e possuem bancos de dados separados, um microsserviço não pode acessar o BD de outro;
- Devem ser usados os padrões *Arquitetura de Microsserviços, API Gateway*, *Database Per Service, CQRS (Conta) e SAGA Orquestrada*;
- Cada microsserviço trata de um subdomínio específico: Cliente, Conta, Gerente, Autenticação. Pode ser criado mais algum microsserviço, desde que validado com o professor;
- Os microsserviços devem estar em conformidade com o **Modelo de Maturidade de Richardson Nível 2**.
- Todos os elementos dos sistemas devem ser conteinerizadas individualmente usando Docker: uma imagem para Front-end, uma imagem para o API Gateway, uma imagem para cada microsserviço e uma imagem para cada banco de dados;
- O Front-end só deve se comunicar com o API Gateway, via API HTTP-REST;
- O API Gateway deve se comunicar com seus microsserviços via API HTTP-REST;
- A implementação das SAGAs deve ser feita com orquestração usando filas assíncrona com RabbitMQ. **Esse conteúdo não será passado em sala e faz parte do conteúdo de PESQUISA que vocês devem aprender;**
- O microsserviço de Conta deve ser implementado com o padrão CQRS, usando fila assíncrona com RabbitMQ para atualização do banco de dados de consulta;
- O *build*, geração das imagens e execução deve ser feita a partir de um *shell script* automatizado;
- Senhas devem ser criptografadas;
- O leiaute deve ser agradável, usando Bootstrap ou Material no Angular;
- Todos os campos que precisarem devem ter validação;
- Todas as datas e valores monetários devem ser entrados e mostrados no formato brasileiro;
- Todos os campos que tiverem formatação devem possuir máscara;
- O banco de dados deve estar normalizado apropriadamente, exceto o banco de leitura do microsserviço Conta (CQRS) que pode estar denormalizado;
- O sistema será testado usando o navegador FIREFOX, versão mais recente.