Universidade Federal Fluminense

Mubank
Documento de Arquitetura de Software

Versão <0.3>

Histórico da Revisão

Data	Versão	Descrição	Autor
18/03/2022	0.1	Estado inicial	Grupo
15/06/2022	0.2	Ajustes da arquitetura	Grupo
12/07/2022	0.3	Ajustes finais na introdução, Metas e restrições da arquitetura, Requisitos funcionais, Camadas da Arquitetura e Visões da Arquitetura.	Daniel Maciel, Matheus Erthal, Ary Lopes e André Barata.

Índice Analítico

Documento de Arquitetura de Software

1. Introdução

1.1 Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões significativas que foram tomadas em relação ao sistema.

1.2 Escopo:

- Um usuário poderá fazer o login no sistema através do EMAIL e SENHA;
- Um usuário poderá se cadastrar no sistema através do EMAIL, SENHA, NOME e ACEITE DE TERMOS DE USO;
- Um usuário poderá retirar dinheiro da conta;
- Um usuário poderá inserir dinheiro na conta;
- Um usuário poderá visualizar um extrato com as transações realizadas em sua conta;

2. Metas e Restrições da Arquitetura

- **Estilos/Padrões Arquiteturais utilizados:**
  - O sistema deverá ser desenvolvido usando a linguagem Java.
  - Cliente servidor;
  - Baseado em componentes;
  - Pipes e filtros;
  - MVC;

Metas:

- O servidor, por sua vez, deverá responder as requisições de todos os clientes a uma taxa mínima de 5 requisições por segundo.
- O sistema deverá estar disponível na nuvem, de modo a atingir o requisito de confiabilidade de estar 99% do tempo disponível.

3. Suposições e Dependências

- O sistema deverá manter os dados das transações isolados e suas condições de corridas decorrentes do paralelismo tratadas, de modo que, uma transação, em hipótese alguma, possa gerar uma inconsistência dos valores armazenados no banco de dados

- Por se tratar de um sistema bancário, os valores serão tratados sempre como inteiros, ou seja, os valores em moeda sempre em sua unidade mais básica(por exemplo, os valores em reais serão sempre tratados em centavos), para evitarmos os problemas que são gerados com a manipulação de ponto flutuante.

- O banco de dados do sistema deverá ter uma rotina de backup diária, com objetivo de, no improvável caso de uma falha parar o banco completamente, ser possível retornar os dados a um estado anterior.

- O banco de dados a ser implementado deverá ser obrigatoriamente relacional e dentre uma das opções: PostgreSQL ou MySQL.

4. Requisitos Arquiteturalmente Significantes

4.1. Requisitos

4.1.1. Requisitos Funcionais

ID	Descrição	Teste de aceitação
RF01	O usuário poderá se cadastrar no sistema	O usuário irá inserir as suas informações de nome, email e senha, e após as validações necessárias, uma nova conta será criada.
RF02	O cliente poderá fazer login no sistema	O usuário irá inserir as suas credenciais, e caso estejam corretas, o acesso ao sistema será liberado. Se não estiverem corretas, será exibida uma mensagem de erro.
RF03	O usuário poderá retirar dinheiro da conta	O usuário logado poderá selecionar um valor, menor ou igual ao saldo de sua conta. Após confirmar, seu saldo deverá ser alterado para o saldo antigo menos o saldo retirado.
RF04	O usuário poderá fazer um depósito	O usuário logado poderá depositar uma quantia selecionada em sua conta. Após confirmar, seu saldo deverá ser alterado para o saldo antigo somado ao saldo adicionado.
RF05	O usuário poderá visualizar todas as suas transações	O usuário logado irá conseguir visualizar todas as transações que foram realizadas em sua conta. Caso não exista nenhuma movimentação, será apresentada a mensagem “Você ainda não possui nenhuma transação”.
RF06	O usuário poderá deslogar da	O usuário logado poderá clicar na opção "Sair", e com isso deslogar do

	sua conta	sistema.

4.1.2. Requisitos Não Funcionais

ID	Descrição	Classificação
RNF01	O sistema deverá ser desenvolvido usando a linguagem Java.	Requisito de desenvolvimento
RNF02	O sistema deverá ser testado nos principais navegadores (Chrome, Firefox, Safari e Edge).	Requisito de portabilidade
RNF03	O sistema deverá seguir a LGPD.	Requisito de segurança / proteção
RNF04	O sistema deverá ser responsivo, ou seja, terá que rodar em diferentes dispositivos e tamanhos de janela ou tela, como celulares, tablets e desktops	Requisito de usabilidade
RNF05	O sistema deverá fazer uso de linguagem orientada a objeto sob a arquitetura MVC.	Requisito de padrões
RNF06	Os usuários deverão operar o sistema após um treinamento de no máximo 2 horas.	Requisito de usabilidade
RNF07	O sistema deverá se comunicar com um banco de dados com base em SQL(MySQL, PostgreSQL).	Requisito de interoperabilidade
RNF08	O sistema deverá exigir a aceitação de um termo de uso para habilitar os usuários acessarem suas funcionalidades.	Requisito legal
RNF09	O sistema deve processar um mínimo de 5 requisições por segundo.	Requisito de eficiência

RNF10	O sistema deverá estar disponível durante 99% do tempo.	Requisito de confiabilidade

5. Decisões, Restrições e justificativas

- Decision or constraint and justification
- Decision or constraint and justification

6. Mecanismos Arquiteturais

[Liste os mecanismos arquiteturais, como mecanismos de persistência, comunicação e tratamento de erros, por exemplo, e descreva ocorrente estado de cada um. Inicialmente, cada mecanismo pode ser somente um nome e uma breve descrição. Eles evoluirão até que o mecanismo se torne um padrão ou uma colaboração de elementos de projeto que possam ser aplicados diretamente em algum aspecto do projeto.]

Mecanismo Arquitetural 1

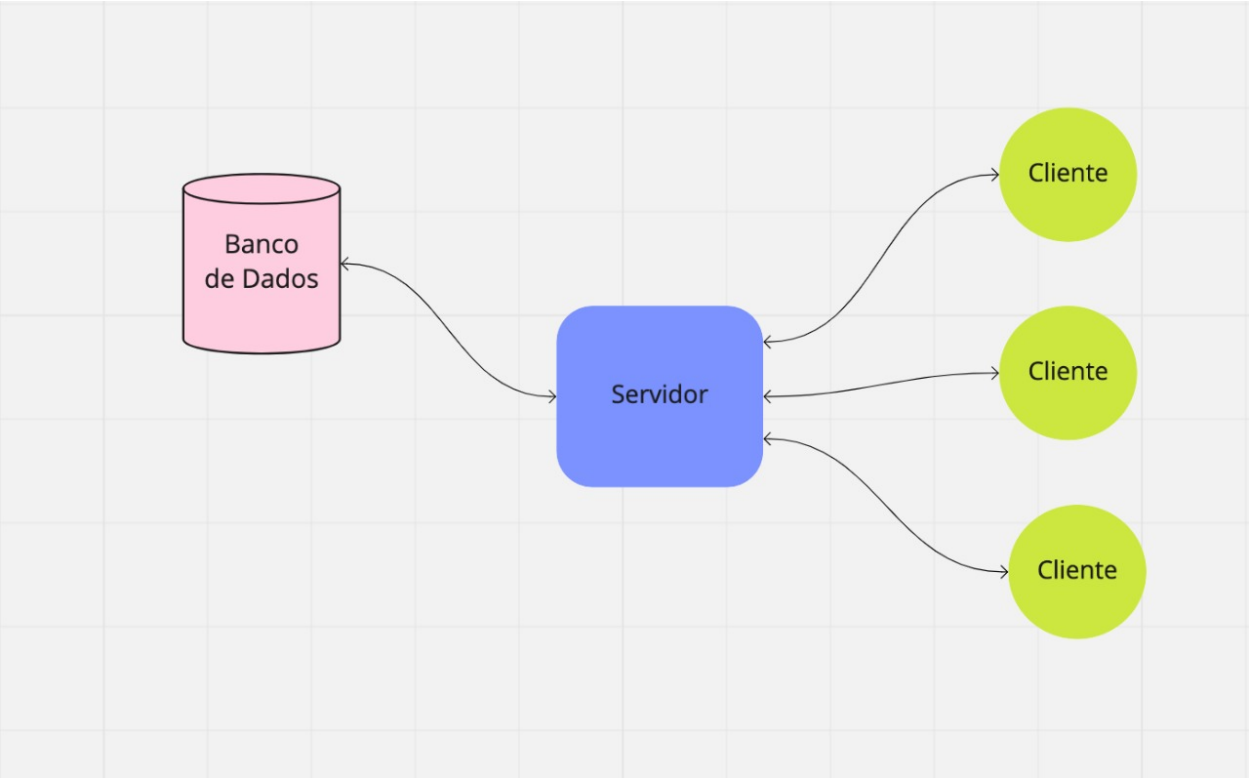
[Descreva a finalidade, os atributos e funções do mecanismo arquitetural.]

Mecanismo Arquitetural 2

[Descreva a finalidade, os atributos e funções do mecanismo arquitetural.]

7. Camadas da Arquitetura

O sistema foi construído utilizando SERVLET em Java, se comunicando com o banco de dados MYSQL por meio de um conector JDBC, seguindo o estilo arquitetural Cliente-Servidor no qual o servidor responde as requisições de diversos clientes.



8. Visões da Arquitetura

8.1. Casos de Uso

Nome do caso de uso	[UC1] Fazer o cadastro no sistema
Ator(es)	Usuário
Descrição	O usuário despertou um interesse na MuBank e deseja ser um cliente. Ele terá que preencher o formulário com seus dados reais para fazer o cadastro.
Referências	-
Gatilho	Selecionar a opção “Quero me cadastrar!”.
Pré-condições	- O usuário deverá concordar com os termos de uso da MuBank;

	- O usuário deverá preencher o formulário com os seus dados reais.
Pós-condições	-
Fluxo principal (cenário típico)	1. O usuário irá selecionar a opção “Quero me cadastrar!”; 2. O usuário irá preencher o formulário com os seus dados reais; 3. O usuário irá concordar com os termos de uso da MuBank; 4. O usuário irá selecionar a opção “Cadastrar”.
Fluxo alternativo (cenário alternativo)	1.1. O usuário irá selecionar a opção “Login”; 1.2. O usuário irá selecionar a opção “Ainda não tem cadastro? Clique aqui!”; 1.3. O usuário irá preencher o formulário com os seus dados reais; 1.4. O usuário irá concordar com os termos de uso da MuBank; 1.5. O usuário irá selecionar a opção “Cadastrar”.

Nome do caso de uso	[UC2] Fazer o login no sistema
Ator(es)	Cliente
Descrição	O cliente deseja acessar a plataforma e suas funcionalidades, ele terá que fazer o login no sistema com suas credenciais.
Referências	-
Gatilho	Selecionar a opção “Login”.

Pré-condições	- O cliente deverá estar cadastrado no sistema; - O cliente deverá preencher o formulário de Login com seu email e a sua senha.
Pós-condições	-
Fluxo principal (cenário típico)	1. O cliente irá selecionar a opção de login; 2. O cliente deverá preencher o formulário de Login com seu email e a sua senha; 3. O cliente irá selecionar a opção "Fazer Login“.
Fluxo alternativo (cenário alternativo)	-

Nome do caso de uso	[UC3] Retirar dinheiro da conta
Ator(es)	Cliente
Descrição	O cliente deseja retirar dinheiro da sua conta MuBank;
Referências	-
Gatilho	Clicar no botão "Sacar";
Pré-condições	- O cliente deverá possuir uma conta cadastrada no sistema; - O cliente deverá estar logado; - O cliente deverá possuir um valor na conta maior do que deseja sacar.
Pós-condições	- O valor do saque será retirado da conta do cliente; - O saque entrará na lista de transações.

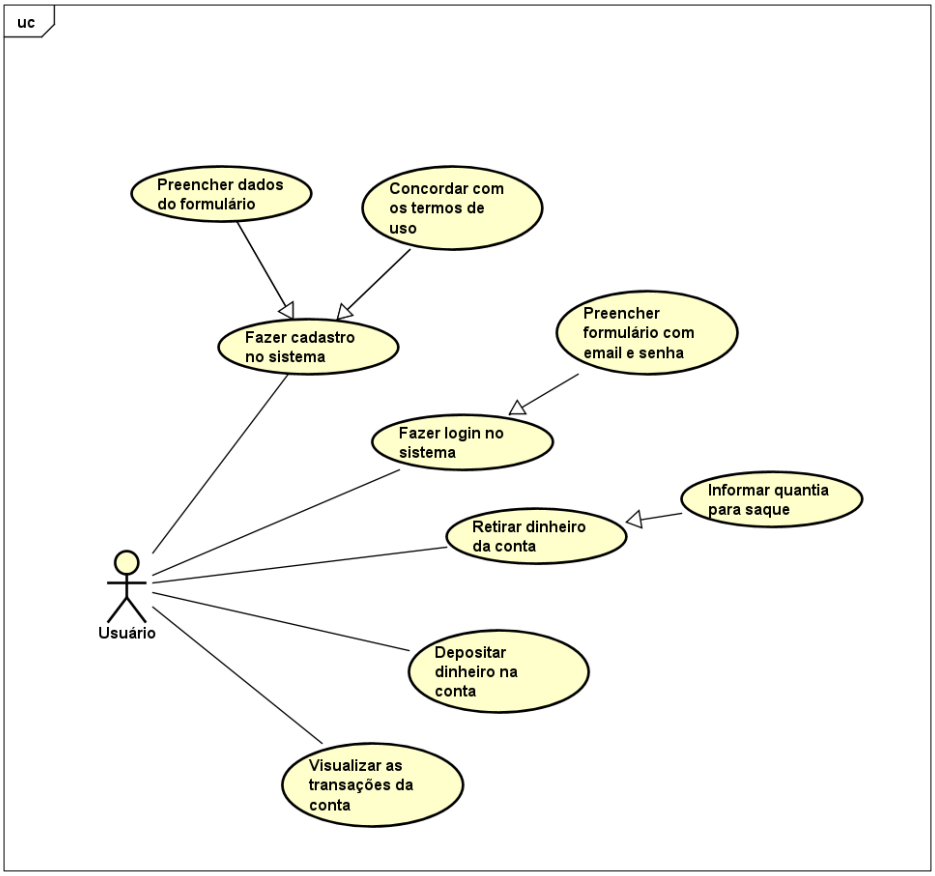
Fluxo principal (cenário típico)	1. O cliente, logado, irá clicar no botão “Fazer saque”; 2. O cliente irá informar o valor do saque; 3. O cliente irá clicar no botão "Sacar".
Fluxo alternativo (cenário alternativo)	-

Nome do caso de uso	[UC4] Depositar dinheiro na conta
Ator(es)	Cliente
Descrição	O cliente deseja depositar dinheiro na sua conta MuBank;
Referências	-
Gatilho	Clicar no botão “Depositar”;
Pré-condições	- A conta de destino deverá existir no sistema;
Pós-condições	- O valor do depósito será adicionado à conta de destino; - O depósito irá entrar no extrato com as transações.
Fluxo principal (cenário típico)	1. O cliente, logado, irá clicar no botão “Fazer depósito”; 2. O cliente irá informar o valor do depósito; 3. O cliente irá clicar no botão "Depositar".
Fluxo alternativo (cenário alternativo)	-

Nome do caso de uso	[UC5] Visualizar as transações da conta
Ator(es)	Cliente

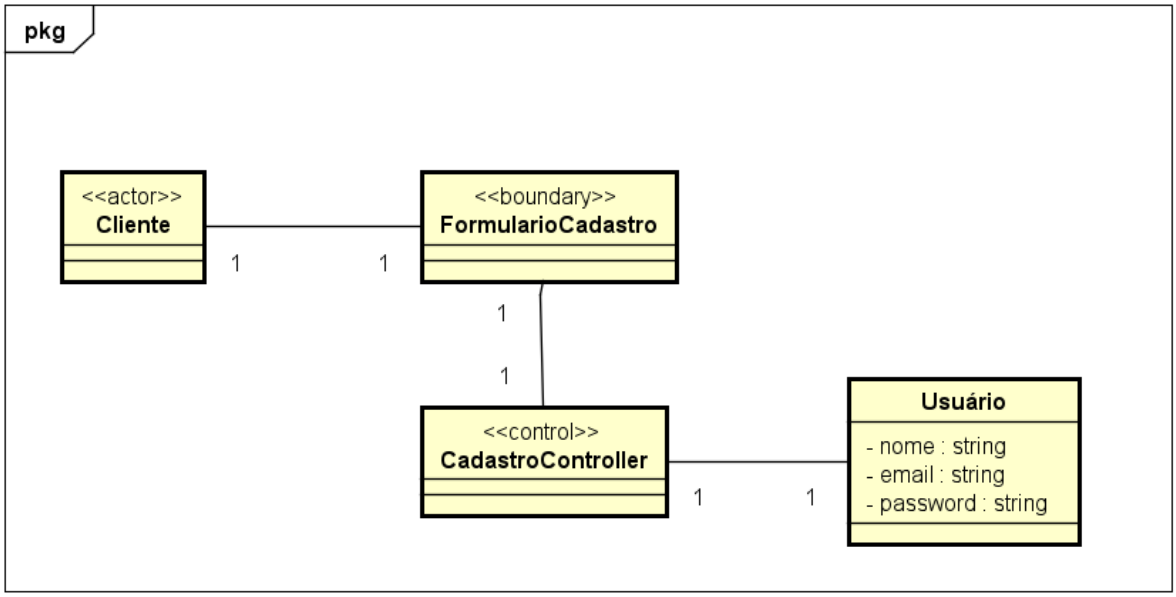
Descrição	O cliente deseja visualizar todas as transações(retiradas e depósitos) realizadas.
Referências	-
Gatilho	Clicar no botão "Ver Transações";
Pré-condições	- Ter alguma transação já realizada; - Estar logado no sistema.
Pós-condições	-
Fluxo principal (cenário típico)	1. O cliente, logado, irá clicar no botão “Extrato”; 2. Serão apresentadas todas as transações realizadas pelo cliente.
Fluxo alternativo (cenário alternativo)	-

8.2. Diagrama de Casos de Uso

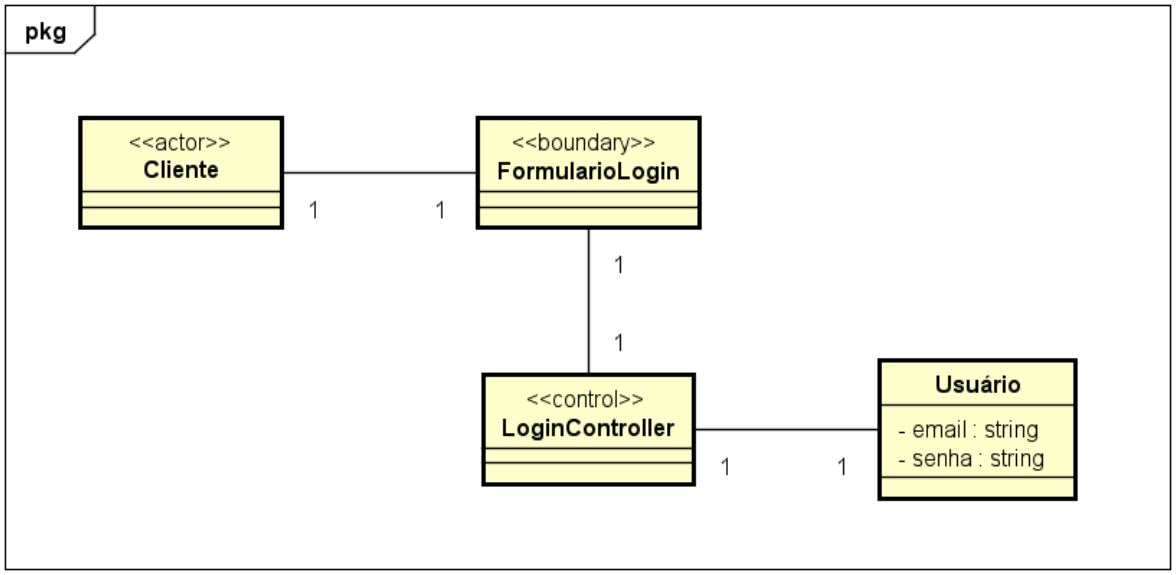


8.3. Visão de Classes Participantes (VCP)

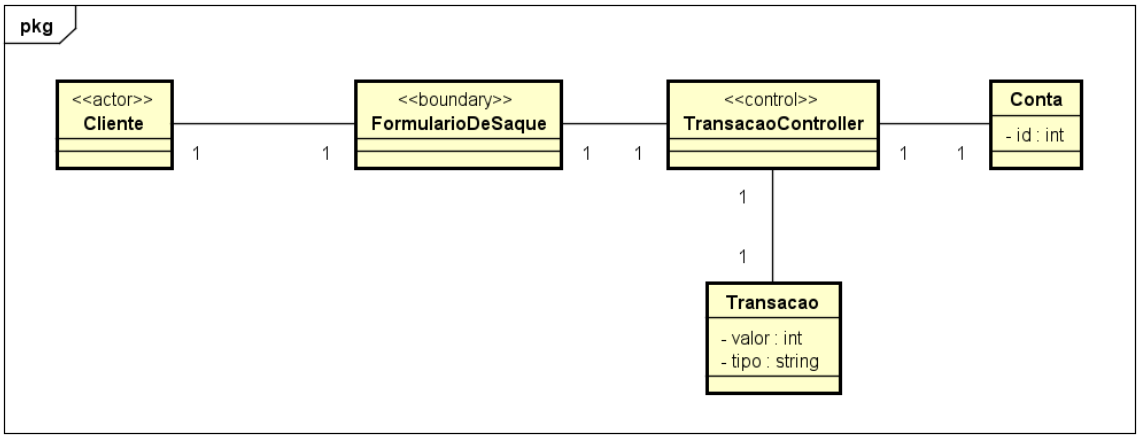
8.3.1 Fazer cadastro no sistema



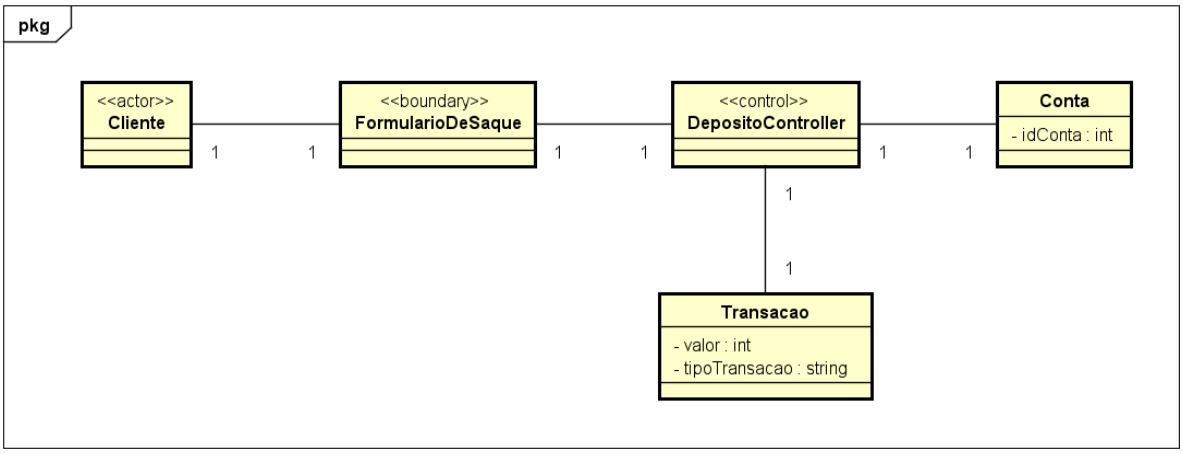
8.3.2 Fazer login



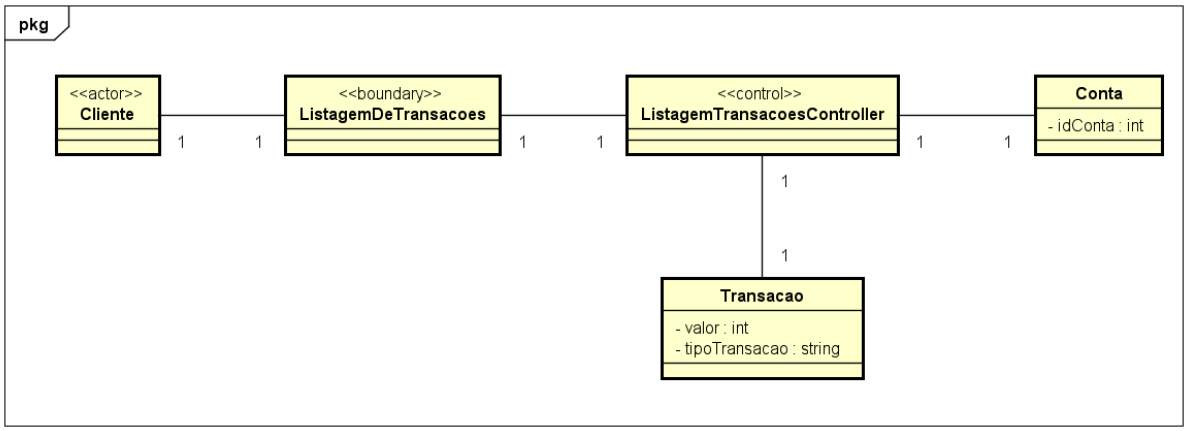
8.3.3 Realizar um saque(retirada)



8.3.4 Fazer depósito

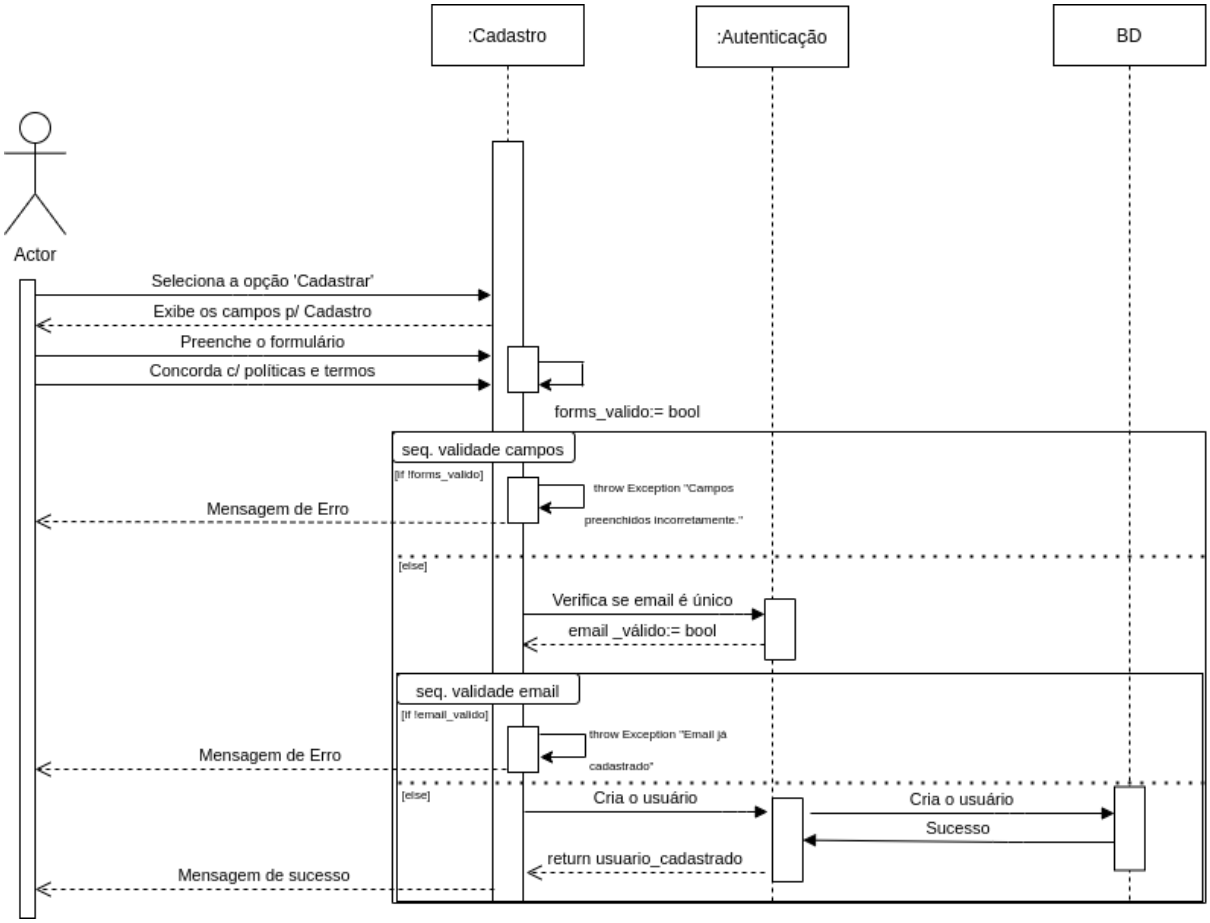


8.3.5 Transações(Extrato)



8.4. Diagrama de Sequências

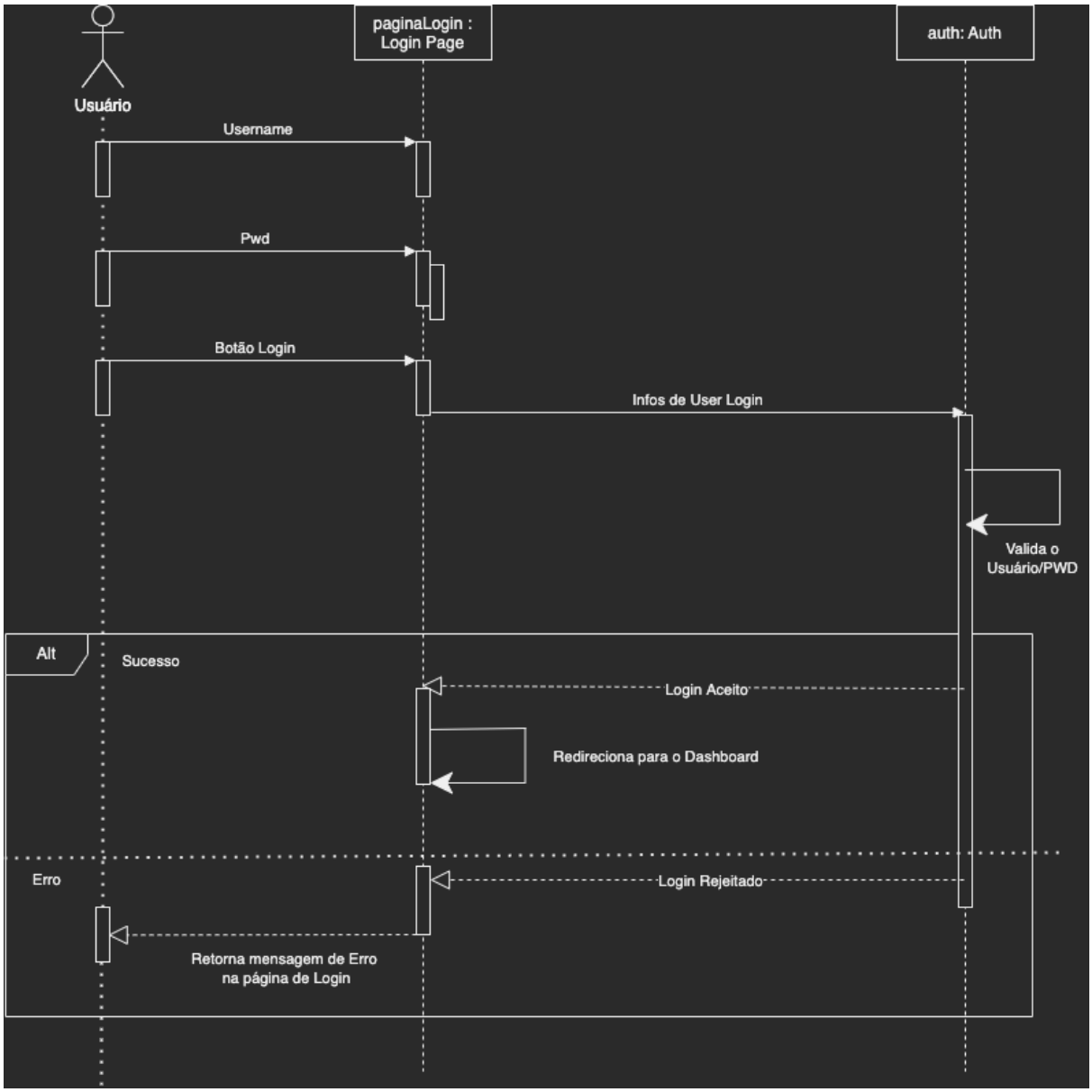
8.4.1 Cadastro



Métodos:

1. **criaUsuario(Object usuario);**

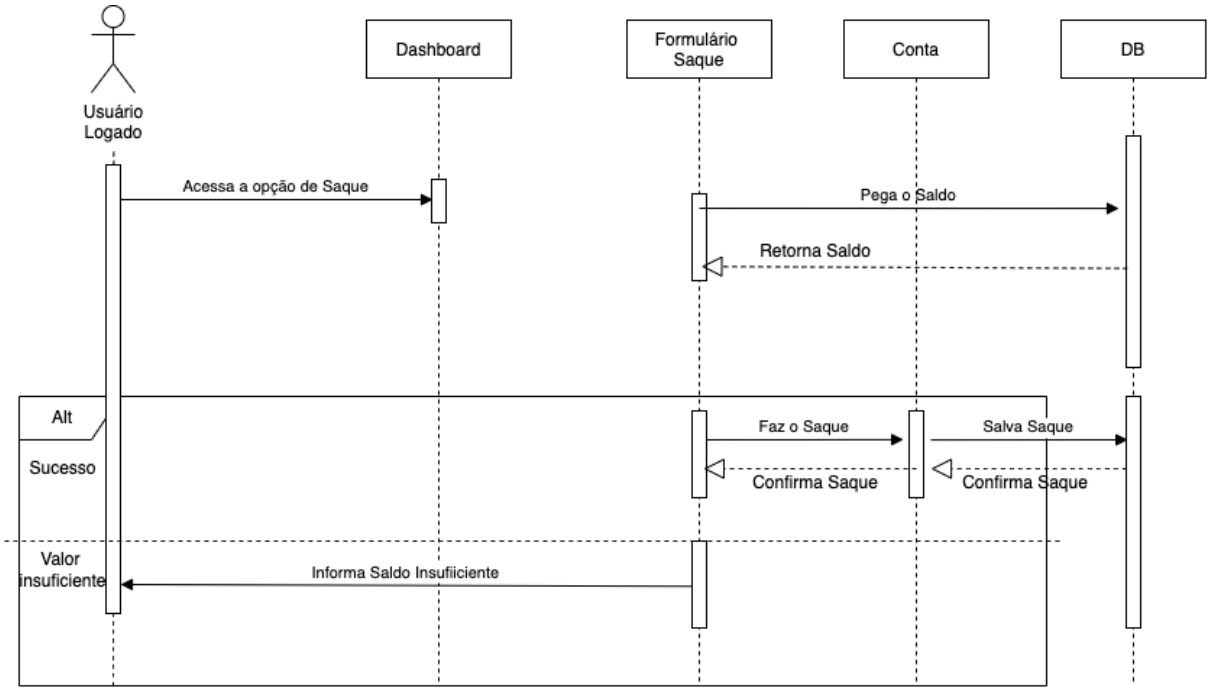
8.4.2 Login



Métodos:

1. **LogaUsuario(Object usuario);**

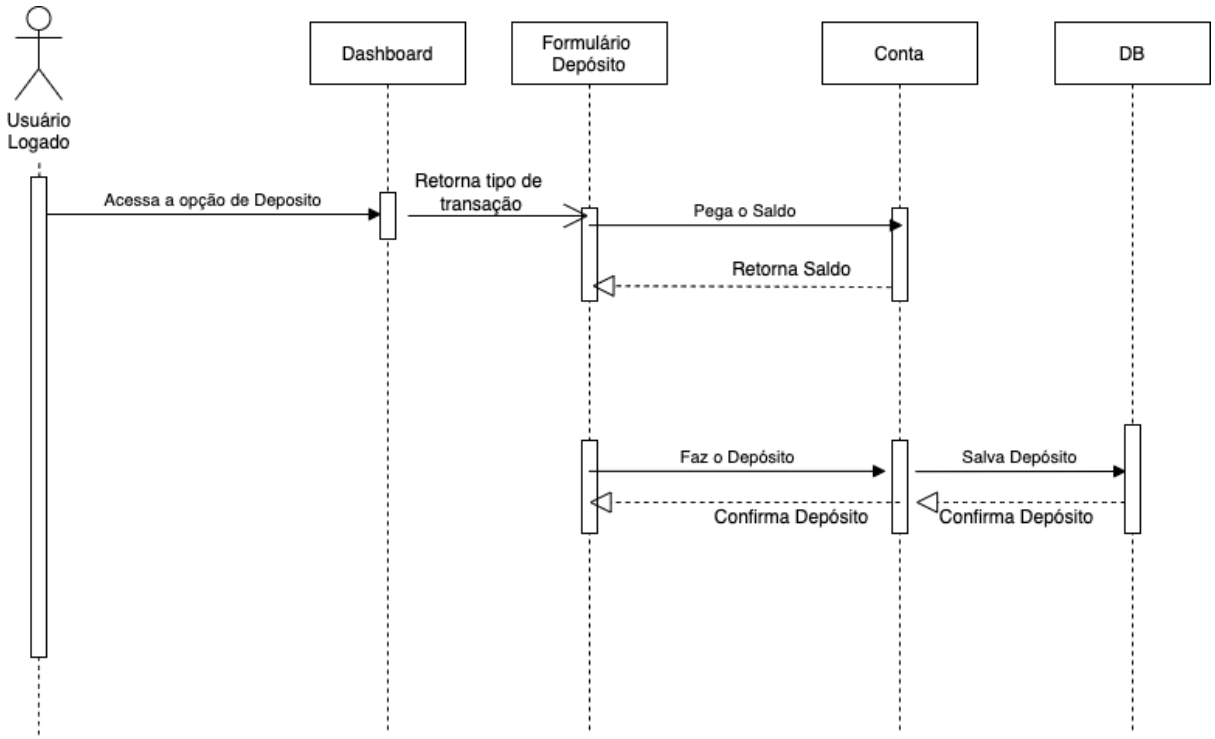
8.4.3 Saque(Retirada)



Métodos:

1. **getSaldo(Object usuario);**
2. **Saque(double valor);**

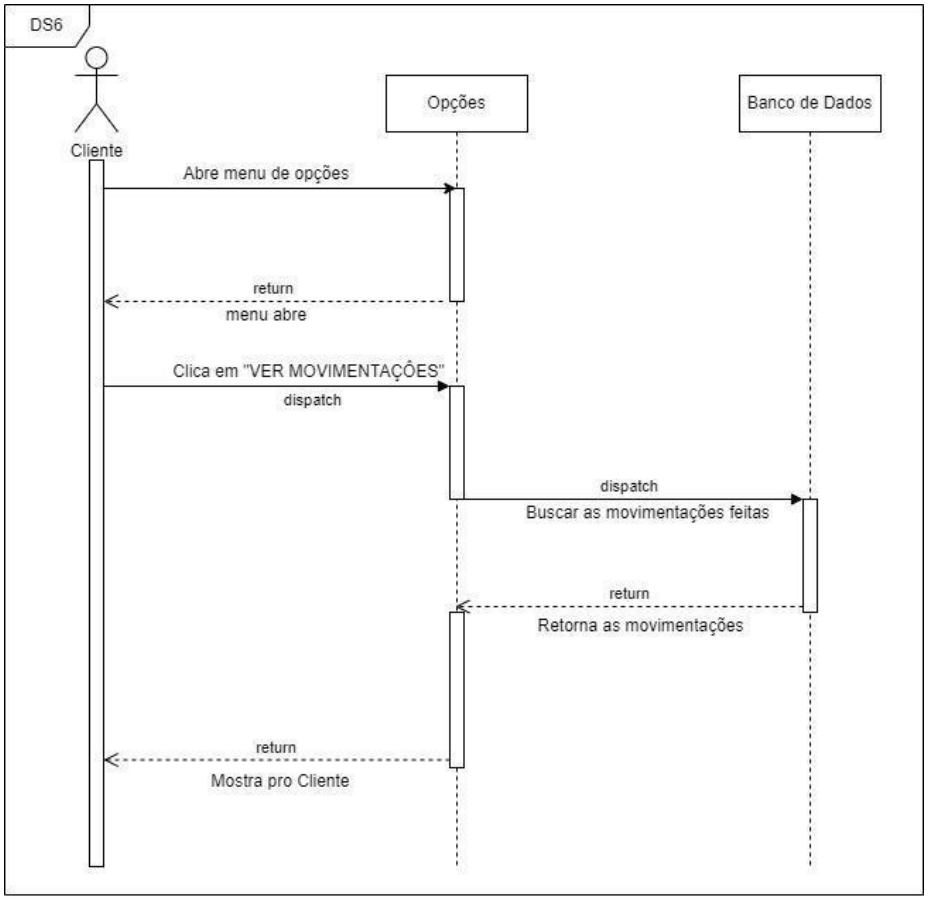
8.4.4 Depósito



Métodos:

1. **getValor(double valor);**
2. **getSaldo(Object usuario);**
3. **deposito(double valor, Object usuario);**

8.4.5 - Transações da conta(Extrato)



Métodos:

1. **getExtrato(DateTime data);**

8.5 Diagrama de Classes:

---

9. Qualidade

[Uma descrição de como a arquitetura do software contribui para todos os recursos (exceto a funcionalidade) do sistema: extensibilidade, confiabilidade, portabilidade e assim por diante. Se essas características possuírem significado especial, como implicações de segurança, garantia ou privacidade, elas deverão ser delineadas claramente.]

10. Problemas de Projeto e Soluções encontradas

Seguindo os padrões GRASP, no contexto de Alta Coesão, analisamos que na modelagem da classe Transação seria interessante dividir os tipos de Transação em subclasses diferentes, sendo essas Saque, Transferência e Depósito. O objetivo foi para que a coesão se mantivesse alta, fosse atribuído responsabilidades diferentes às classes e não houvesse sobrecarrega da classe Transação