UFF - UNIVERSIDADE FEDERAL FLUMINENSE
CIÊNCIA DA COMPUTAÇÃO
PROJETO DE SOFTWARE

IGOR QUINTES DOS SANTOS
JOÃO VICTOR LAGOS DE AGUIAR
MATHEUS LOPES CARAPINA DO NASCIMENTO
PEDRO RIBEIRO FILHO
VITOR VIEIRA GOIS

DOCUMENTO DE ARQUITETURA:

Gerenciador de Finanças Pessoais

Niterói
2024

Universidade Federal Fluminense - UFF
Ciência da Computação
Projeto de Software 2024.1

METODOLOGIA DE DESENVOLVIMENTO DE SOFTWARE

Documento de Arquitetura de Software

Objetivo deste Documento

Este documento tem como objetivo descrever as principais decisões de projeto tomadas pela equipe de desenvolvimento e os critérios considerados durante a tomada destas decisões.

Histórico de Produção

Autor	Atribuições
Igor Quintes dos Santos	• Produção do presente documento (.docx) • Definição dos Requisitos Não Funcionais (RNF1 ao RNF11) • Algumas visões arquiteturais: DSS (Manter Metas Financeiras), Comunicação (Adicionar Metas)
João Victor Lagos de Aguiar	• Produção do presente documento (.docx) • Introdução • Escopo do Sistema • Definição dos Requisitos Funcionais (RF1 ao RF9) • Alguma visões arquiteturais: Caso de Uso, DSS (Login, Receber Notificação e Visualizar Extrato), Comunicação (Inserir Credenciais e Adicionar Transação), Atividades (Login) e Estado (Manter Conta Bancária)
Matheus Lopes Carapina do Nascimento	• Produção do presente documento (.docx) • Definição e produção do Diagrama de Classe Detalhado
Pedro Ribeiro Filho	• Ajudou e auxiliou na validação do Caso de Uso do Sistema • Definiu a linguagem de programação e frameworks ideais para a criação do sistema, antecipadamente
Vitor Vieira Gois	• Produção deste documento (.docx) • Objetivos Arquiteturais • Restrições Arquiteturais • Escolha e explanação dos Padrões Arquiteturais (Microsserviços e Arquitetura Orientada por Mensagens) • Algumas visões arquiteturais: Modelo Conceitual, DSS (Manter e Integrar Contas Bancárias e Manter Despesas/Receitas), Comunicação (Integrar/Manter Conta e Visualizar Extrato)

SUMÁRIO

1. INTRODUÇÃO

O presente documento visa apresentar uma visão abrangente do sistema Gerenciador de Finanças Pessoais desenvolvido pelo grupo *VIPJM Moedinhas Malucas*, antiga [EM DEFINIÇÃO…], da disciplina de Projeto de Software, cursada no semestre 2024.1 do curso de Ciência da Computação, na Universidade Federal Fluminense (UFF). O referido grupo é composto por cinco integrantes, são eles: Igor Quintes dos Santos, João Victor Lagos de Aguiar, Matheus Lopes Carapina do Nascimento, Pedro Ribeiro Filho e Vitor Vieira Gois.

Este documento também visa delinear principais funcionalidades, escopo do sistema, requisitos arquiteturais do sistema, padrões arquiteturais, objetivos e restrições da arquitetura, bem como diagramas de casos de usos, modelo conceitual, diagramas de sequência do sistema e entre outros, conferindo assim uma qualidade maior na produção do software.

Por meio deste documento, espera-se que os integrantes do grupo possam transmitir claramente o escopo e os objetivos do projeto. Ademais, este documento servirá como um guia de referência durante as etapas de desenvolvimento e criação do sistema em questão e dos diagramas.

O link referente a este trabalho (projeto), no repositório do GitHub, é: https://github.com/vaniacourses/trabalho-pr-tico-vipjm-moedinhas-malucas.
**Observação: Este repositório é privado, portanto pode haver peculiaridades no acesso.*

1.2. CONTEXTUALIZAÇÃO

No mundo contemporâneo, o gerenciamento eficiente das finanças pessoais é fundamental para garantir a estabilidade financeira e o alcance de objetivos financeiros de curto, médio e longo prazo. Com a crescente complexidade das transações financeiras e a diversidade de fontes de renda e despesas, torna-se imperativo contar com ferramentas tecnológicas capazes de oferecer suporte na organização e controle das finanças individuais. Nesse contexto, surge a necessidade de desenvolvimento de um sistema Gerenciador de Finanças Pessoais, uma aplicação que visa facilitar o acompanhamento e a gestão das receitas, despesas e investimentos de um indivíduo.

2. DESCRIÇÃO DO ESCOPO DO SISTEMA

O sistema Gerenciador de Finanças Pessoais é uma aplicação de software projetada para ajudar/auxiliar os usuários na gestão eficaz e correta de suas finanças pessoais. O sistema permitirá aos usuários registrar e monitorar suas receitas e despesas, categorizando cada transação de acordo com sua natureza (ex: alimentação, transporte, moradia, lazer, etc.). Além disso, possibilitará o acompanhamento do saldo disponível, a definição de metas financeiras e a análise do histórico de transações.

Com isso, suas principais funções incluem, em um rol exemplificativo:

- **Registro de Receitas e Despesas**: Os usuários terão a capacidade de registrar diversas transações recorrentes, abrangendo tanto receitas quanto despesas. Este recurso visa fornecer uma visão abrangente das finanças pessoais do usuário, permitindo um acompanhamento detalhado de todas as fontes de entrada e saída de recursos financeiros.
  Algumas das transações que podem ser registradas incluem:
  - **Receitas:**
    - Salário mensal
    - Renda extra (por exemplo, freelancers, vendas, investimentos)
    - Entre outras fontes de renda
  - **Despesas:**
    - Contas de serviços públicos (água, luz, gás, etc)
    - Aluguel ou prestação da casa
    - Alimentação (supermercado)
    - Transporte (combustível, transporte público)
    - Educação (mensalidade)
    - Plano de saúde
    - Entre outras despesas

- **Categorização dos Registros:** O usuário poderá categorizar os registros, tanto de despesas como de receitas com base em padrões previamente estabelecidos, facilitando o processo de organização financeira. Essas categorias podem abranger uma variedade de áreas, como visto acima. Com isso, o usuário poderá acompanhar e analisar os seus gastos e ganhos.

- **Análise de Desempenho Financeiro:** O sistema fornecerá gráficos e relatórios detalhados sobre fluxo de caixa, hábitos de consumo, receitas e despesas, a fim de que os usuários possam identificar áreas de melhoria e tomar decisões financeiras mais informadas.

- **Integração com Contas Bancárias:** O sistema poderá integrar as contas

bancárias dos usuários, permitindo a importação automática de informações financeiras e facilitando a reconciliação de dados. Um exemplo da utilidade dessa função é que caso o usuário tenha sua conta bancária vinculada ao aplicativo, ele não precisará colocar a informação do seu salário, rendimentos de aplicações financeiras, mensalidade de escola ou plano de saúde e entre outros, pois o sistema irá identificar esses dados e importará para o aplicativo.

- **Notificações Financeiras:** Os usuários poderão ser informados pelo sistema sobre transações importantes, vencimentos de contas e outras informações relevantes para sua saúde financeira.

3. INFORMAÇÕES ARQUITETURAIS

Nesta seção será detalhado todos os aspectos relacionados à arquitetura do sistema. Serão abordados os requisitos arquiteturais (requisitos funcionais e requisitos não funcionais), objetivo e restrições da arquitetura.

Ao identificar e documentar os requisitos arquiteturais do sistema, podemos entender melhor as necessidades específicas que influenciam a arquitetura. Os objetivos e restrições da arquitetura fornecem uma estrutura para avaliar e justificar as decisões arquiteturais, garantindo que o sistema atenda aos requisitos funcionais e não funcionais, enquanto cumpre limitações e considerações importantes.

3.1. REQUISITOS ARQUITETURAIS

Os requisitos arquiteturais representam as necessidades específicas que moldam a arquitetura do referido sistema. Estes requisitos abrangem os aspectos funcionais e os não funcionais e desempenham um papel importante na estrutura e no comportamento do software.

Os requisitos funcionais delineiam as capacidades que o sistema deve ter, descrevendo as funcionalidades específicas que devem ser implementadas para atender às necessidades dos usuários e às exigências do domínio do problema. Já os requisitos não funcionais referem-se a características ou qualidades do sistema. No contexto deste software, consideramos aspectos como segurança, eficiência, portabilidade e confiabilidade, entre outros, a serem considerados.

Requisitos Funcionais:

- **RF1:** O sistema deverá permitir que o cliente cadastre e faça o respectivo login da sua conta no aplicativo.
- **RF2:** O sistema deverá permitir ao usuário gerenciar/manter o seu perfil, podendo configurá-lo da maneira que desejar.
- **RF3:** O sistema deverá permitir que o cliente crie várias contas e gerencie/mantenha as respectivas contas (CRUD das contas).
- **RF4:** O sistema deverá permitir ao usuário cadastrar as despesas e receitas em sua conta.
- **RF5:** O sistema deverá permitir que o usuário categorize os respectivos registros (receitas e despesas) por especialização, como alimentação, educação, transporte e entre outros.
- **RF6:** O sistema deverá permitir ao usuário visualizar o seu extrato e, a partir dele, a possibilidade de gerar um relatório financeiro, que será disponibilizado pelo sistema, referente ao respectivo extrato.

- **RF7:** O sistema deverá permitir ao usuário a possibilidade de definir metas financeiras.
- **RF8:** O sistema deverá permitir que o usuário possa receber notificações financeiras a respeito das suas contas.
- **RF9:** O sistema deverá permitir a integração de contas bancárias por parte dos usuários, permitindo assim a importação financeira automática e facilitando a reconciliação de dados.

Requisitos Não Funcionais:

Segurança

- **RNF1:** O sistema deve assegurar que as contas dos usuários sejam completamente protegidas contra acessos não autorizados.
- **RNF2:** O sistema deve incluir protocolos de criptografía para proteger todas as comunicações entre o servidor e o usuário.
- **RNF3:** O sistema deve ser capaz de registrar e rastrear todas as ações realizadas pelos usuários dentro da plataforma.

Eficiência

- **RNF4:** O sistema deve otimizar consultas ao banco de dados para garantir tempos de resposta rápidos, especialmente durante períodos de alto tráfego.
- **RNF5:** O sistema deve minimizar o consumo de recursos, como uso de CPU e memória, para garantir que o aplicativo seja executado suavemente em dispositivos com diferentes especificações.
- **RNF6:** O sistema deve garantir que o tempo de resposta para as interações do usuário seja mantido dentro de um limite máximo de 2 segundos.

Portabilidade

- **RNF7:** O sistema deve ser programado na linguagem Ruby.
- **RNF8**: O sistema deve garantir que a interface do usuário seja responsiva e se adapte automaticamente a diferentes tamanhos de tela e orientações de dispositivo.

Confiabilidade

- **RNF9:** O sistema deve implementar backups regulares dos dados dos usuários para evitar perda de informações importantes.
- **RNF10:** O sistema deve estabelecer um sistema de monitoramento contínuo para detectar e corrigir rapidamente problemas de desempenho ou falhas.

Manutenibilidade

- **RNF11:** O sistema deve ser desenvolvido com código limpo e bem documentado para facilitar a manutenção e aprimoramentos futuros.

3.2. OBJETIVOS ARQUITETURAIS

Os objetivos da arquitetura representam as metas que são buscadas alcançar por meio da implementação da arquitetura do sistema. Com isso em mente, segue os objetivos desejados ao término da implementação.

O primeiro deles é um sistema que consiga atender às necessidades dos usuários de forma abrangente e que consiga atender pessoas de diferentes níveis financeiros ou de conhecimento técnico de forma que se consiga garantir um ambiente financeiro mais saudável em suas vidas pessoais.

Outro objetivo seria permitir de forma mais eficiente a visualização do estado financeiro do usuário, sem precisar acessar diversos programas e aplicativos bancários que muitas vezes demoram e não são eficientes devido à alta demanda de usuários acessando o sistema ou por questões de hardware do próprio usuário.

Além disso, o sistema deve conseguir unificar dados de diferentes contas financeiras de forma segura que evite invasões e, em caso de tentativas dessas invasões, tenha a capacidade de impedir o vazamento de dados desses usuários.

E para terminar, o sistema deve ser escalável de forma que se possa inserir mais funcionalidades, usuários e transações no sistema sem comprometer o desempenho e a confiabilidade.

3.3. RESTRIÇÕES ARQUITETURAIS

As restrições da arquitetura envolvem os limites impostos ao projeto e influenciam nas escolhas arquiteturais. Os mais fáceis a serem observados são o prazo para a implementação do sistema e os recursos limitados que se tem em hardware e equipe e que criam como consequência a busca por maneiras mais rápidas e fáceis de se adicionar as funcionalidades e relacioná-las umas às outras.

Outra restrição seria a capacidade de se obter as informações bancárias de um usuário sem acesso aos dados de forma direta aos dados dentro da agência financeira e de forma que o sistema não possa ser utilizado como um auxílio para um possível invasor consiga acessar e manipular o dinheiro do usuário guardado nas contas bancárias inseridas no sistema.

A maturidade do grupo também é um ponto a se pensar. Em discussões percebe-se que a experiência em se produzir um gerenciador de finanças não é forte e que possivelmente novos conhecimentos técnicos de linguagens e formas de como se organizar o código vão ser aprendidos no momento da implementação, desacelerando o progresso de criação do sistema.

E por fim, como esse sistema poderia ser utilizado em diferentes dispositivos com desempenhos e sistemas operacionais diferentes. Há de se pensar então na implementação do sistema com formas de ser facilmente eficiente e portável caso um novo ambiente ou atualizações de ambientes apareçam.

3.4. PADRÕES ARQUITETURAIS

Padrões arquiteturais são organizações estruturais de sistemas com algum grau de semelhança e que podem ser usados para delimitar restrições, organizar módulos e definir suas relações em contextos parecidos. Ou seja, orientam a solução de uma família de problemas. Dentro desse sistema os padrões escolhidos são definidos abaixo:

- **Microsserviços:** As aplicações são estruturadas em serviços simples e independentes, com cada um deles tendo funções bem definidas. Eles podem ser desenvolvidos, testados e implantados independente dos outros, com diferentes linguagens de programação e frameworks se for da escolha dos desenvolvedores . Dessa forma, se um microsserviço falhar, outros ainda podem continuar sendo executados sem erros.
  Possuem baixo acoplamento já que os microsserviços se comunicam visando minimizar independências, e para que isso possa ocorrer, é recomendado o outro padrão arquitetural abaixo.

- **Arquitetura orientada por mensagens:** Nesse padrão, a comunicação entre os componentes do sistema ocorre por meio de mensagens assíncronas em uma fila de mensagens até que o componente que irá consumir esta mensagem esteja disponível . Os componentes não precisam estar funcionando ao mesmo tempo para a mensagem ser passada e assim os componentes não precisam esperar a resposta de outro para continuarem funcionando ou pararem de funcionar em caso de erro de um outro componente. Além de que o fluxo de mensagens pode ser monitorado e gerenciado de forma a direcionar essas mensagens para um determinado processo e melhorar o desempenho da comunicação.

Estas escolhas na arquitetura contribuem para os atributos de qualidade desejados. Por exemplo, existe um aumento da segurança já que a decomposição do sistema diminui o impacto de possíveis ataques, pois com cada um dos serviços tendo uma funcionalidade própria, os pontos de entrada para invasores são reduzidos. Além disso, a comunicação entre esses módulos pode ser protegida por mecanismos de autenticação e autorização, controlando o acesso aos dados financeiros do usuário.

O sistema também vai ter a capacidade de escalar adicionando servidores e banco de dados para cada microsserviço, conseguindo atender um possível

aumento de demanda de armazenamento e processamento e também agilizando a execução de requisições de forma eficiente. Tarefas complexas podem ser processadas de forma assíncrona, por consequência isso diminui o tempo de resposta entre o usuário e o sistema.

Como já dito anteriormente, os microsserviços podem podem ser desenvolvidos em diferentes linguagens de programação e frameworks e tem uma natureza modular. Isso facilita a migração e execução do sistema para diferentes plataformas e ambientes.

Cada microsserviço pode ter uma documentação individual e detalhada, o que facilita o funcionamento de cada parte do código e facilita a as manutenções e atualizações do sistema no futuro. Os testes também podem ser feitos em cada módulo individualmente, o que ajuda em perceber falhas e erros.

E para terminar, a falha de um microsserviço não afeta o funcionamento geral do sistema, pois os outros microsserviços continuam operando e a comunicação entre eles ocorre por mensagens assíncronas. Além disso as mensagens podem ser facilmente reprocessadas e retransmitidas, tornando o funcionamento do sistema mais confiável.

4. VISÕES ARQUITETURAIS

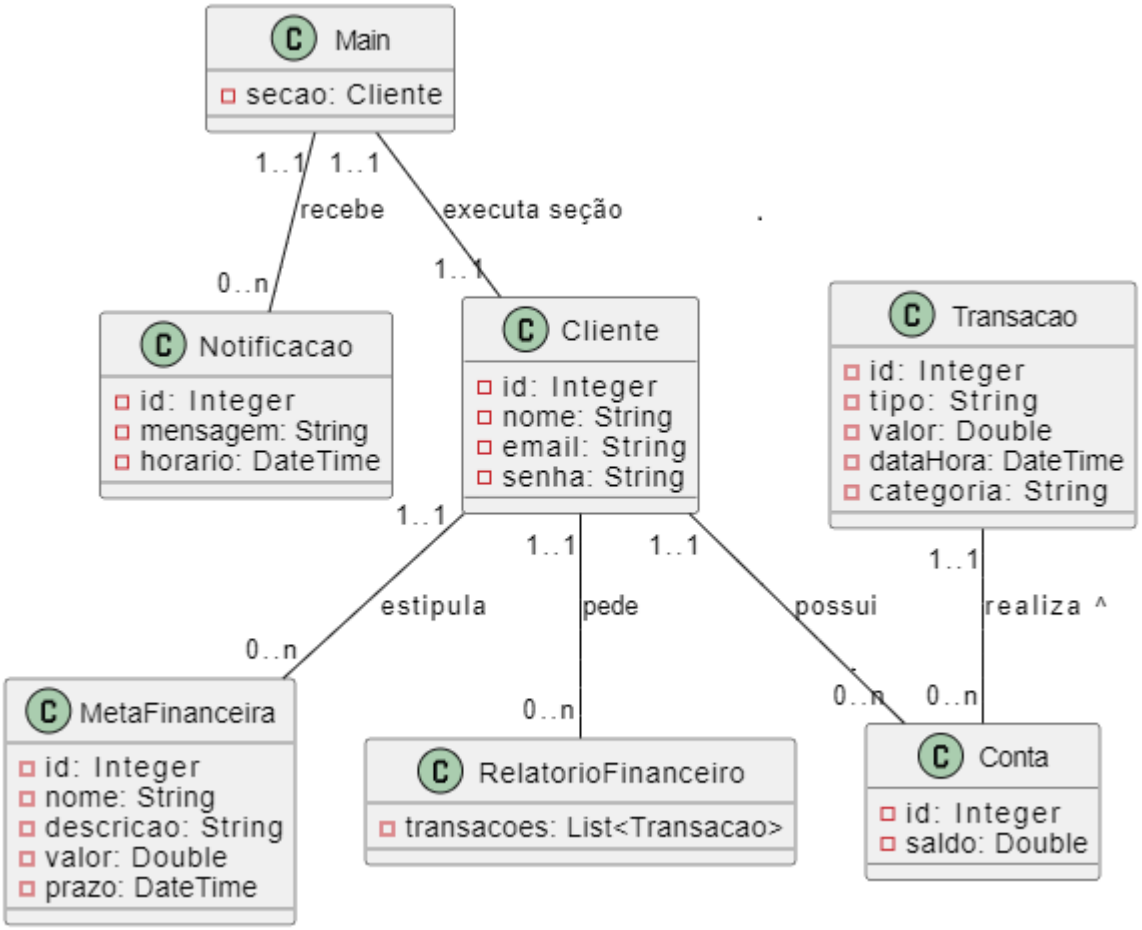
Em projeto de software, as visões arquiteturais consistem em representações abstratas e estruturadas de um sistema de software, que capturam diferentes perspectivas arquiteturais relevantes de forma que se observe a mesma coisa, ressaltando características e propriedades que sejam consideradas interessantes e omitindo as não relevantes. Uma visão arquitetural descreve como o sistema é organizado/esquematizado e como seus elementos interagem para atender aos requisitos funcionais e não funcionais observados.

As visões arquiteturais são de suma importância para o desenvolvimento do software, pois ajudam a decompor o complexo sistema em partes menores e gerenciáveis, além de comunicar, com clareza, as decisões para as partes interessadas. Cada visão arquitetural dá ênfase em aspectos específicos do sistema e fornece informações detalhadas sobre esses aspectos.

Existem diversas visões arquiteturais no mundo do desenvolvimento de software. Para o referente projeto, as visões arquiteturais disponíveis estão representadas abaixo:

4.1. CASO DE USO

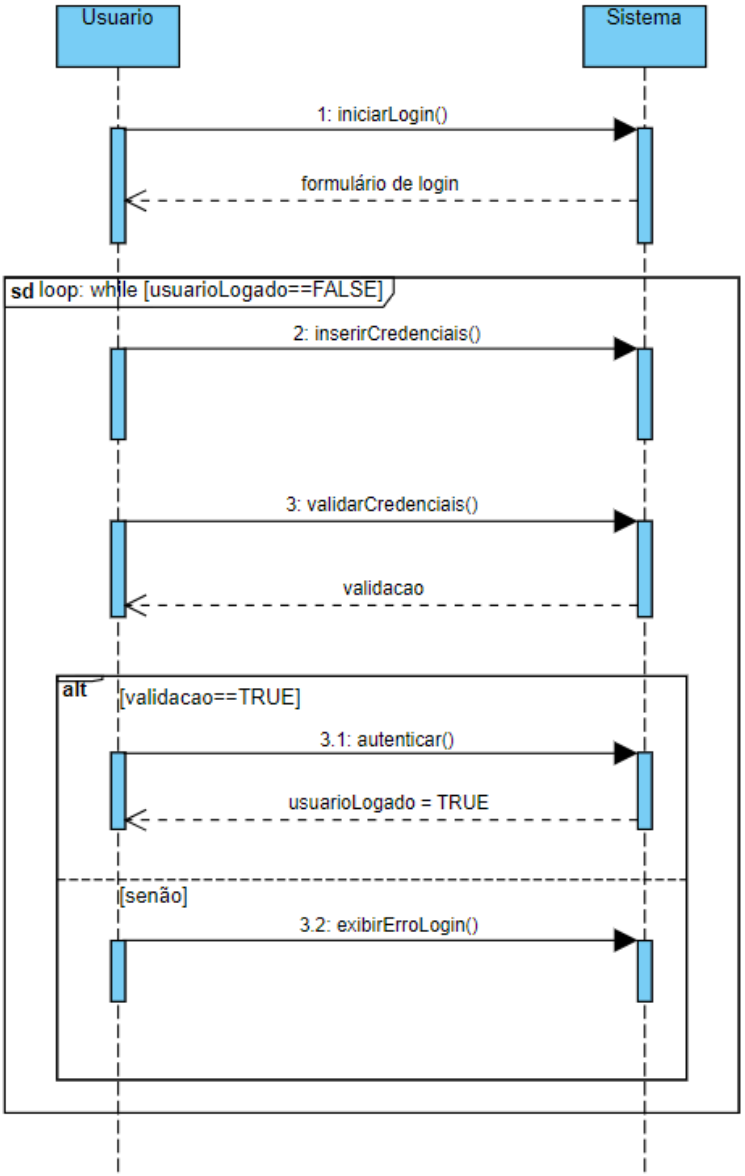
4.2. MODELO CONCEITUAL



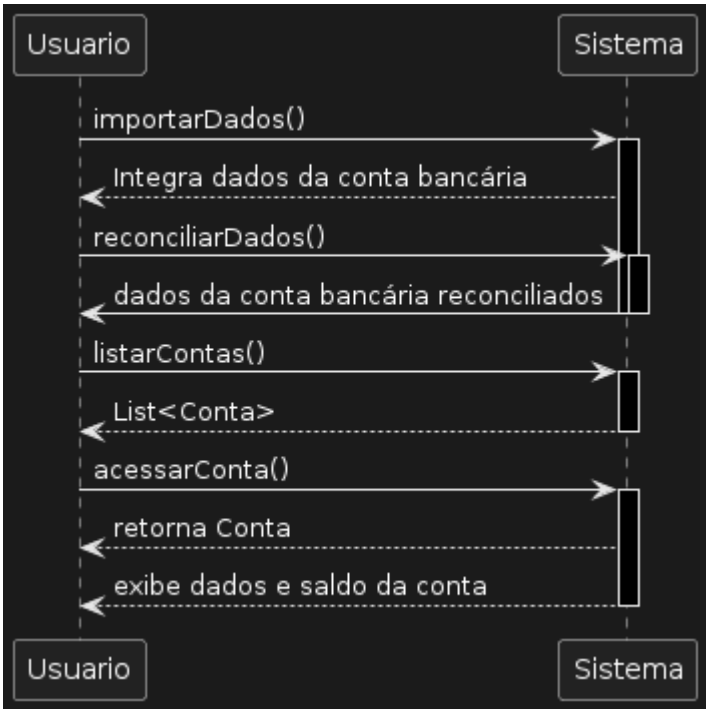
4.3. DIAGRAMAS DE SEQUÊNCIA DE SISTEMA

O Diagrama de Sequência é uma ferramenta de modelagem de sistemas. Ela representa a interação entre objetos em uma determinada sequência temporal, dando enfoque principalmente na troca de mensagens entre esses objetos ao longo do tempo. Para o software a ser desenvolvido, temos os seguintes diagramas de sequência de sistema representados abaixo:

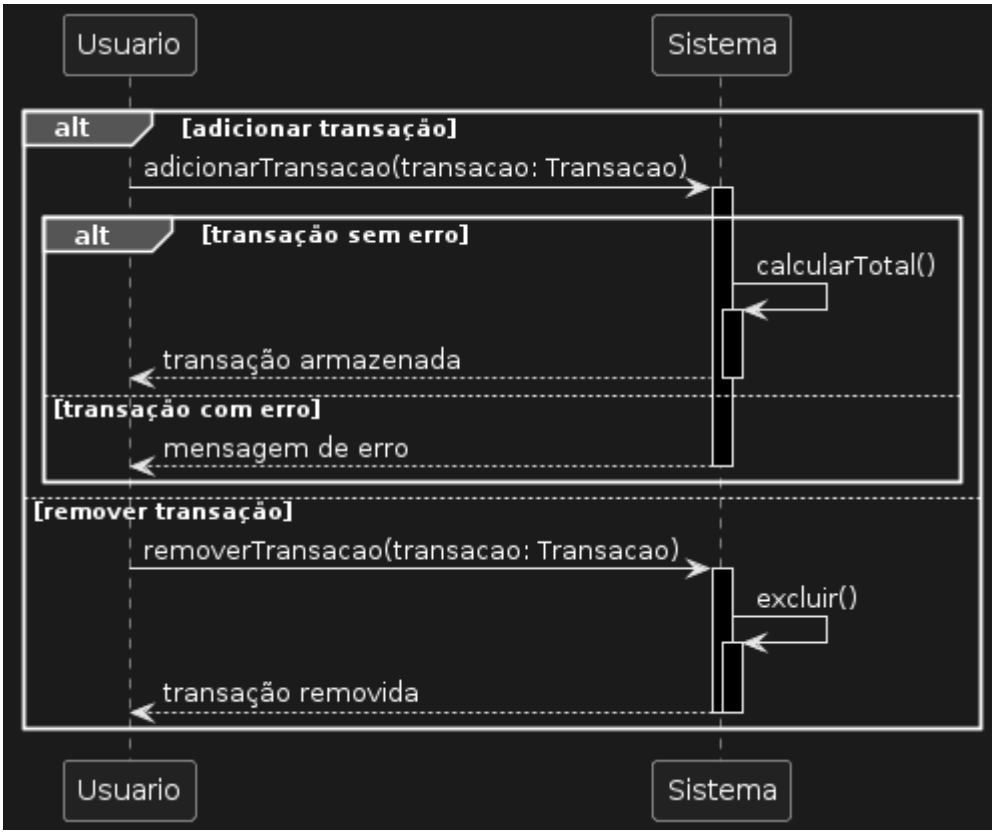
4.3.1. DSS: LOGIN



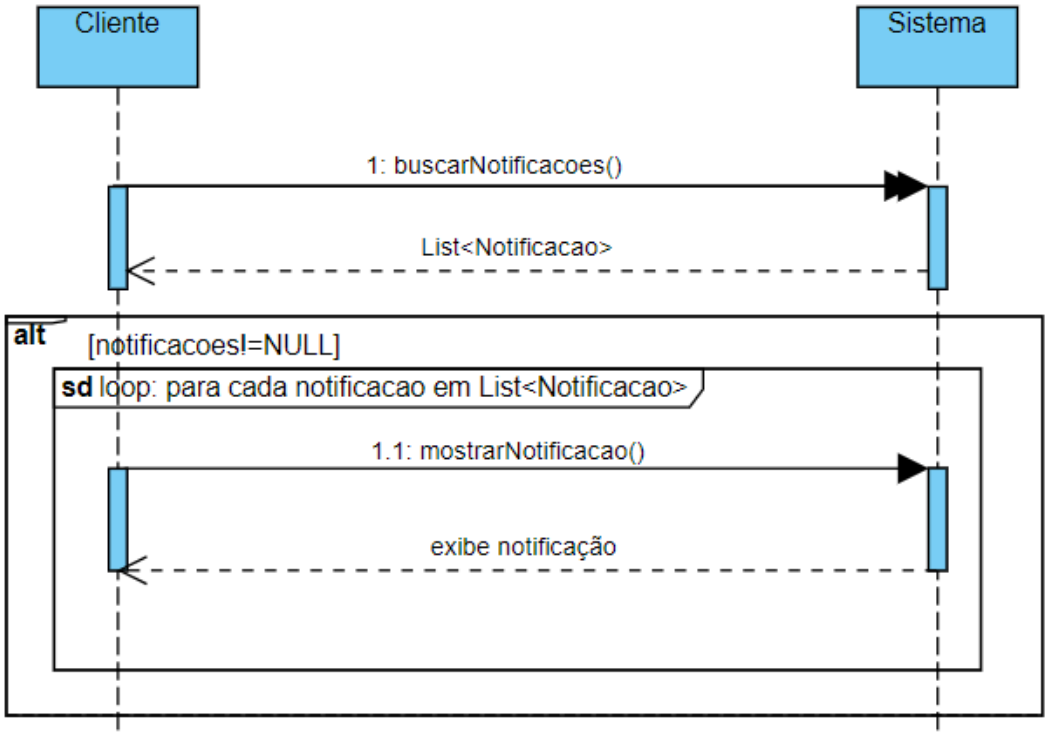
4.3.2. DSS: MANTER E INTEGRAR CONTAS BANCÁRIAS



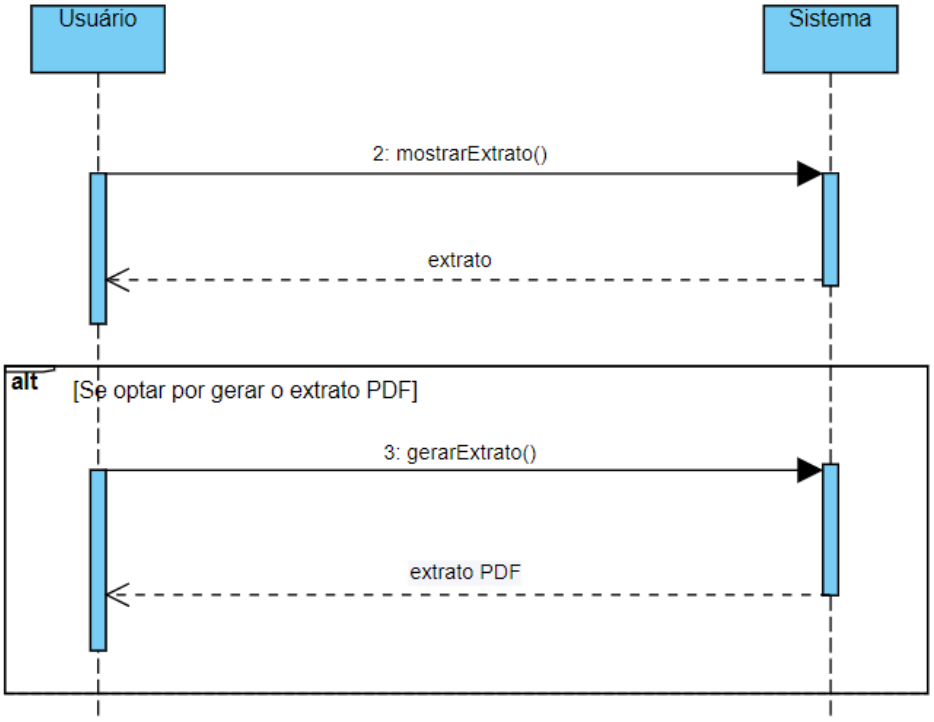
4.3.3. DSS: MANTER DESPESAS/RECEITAS



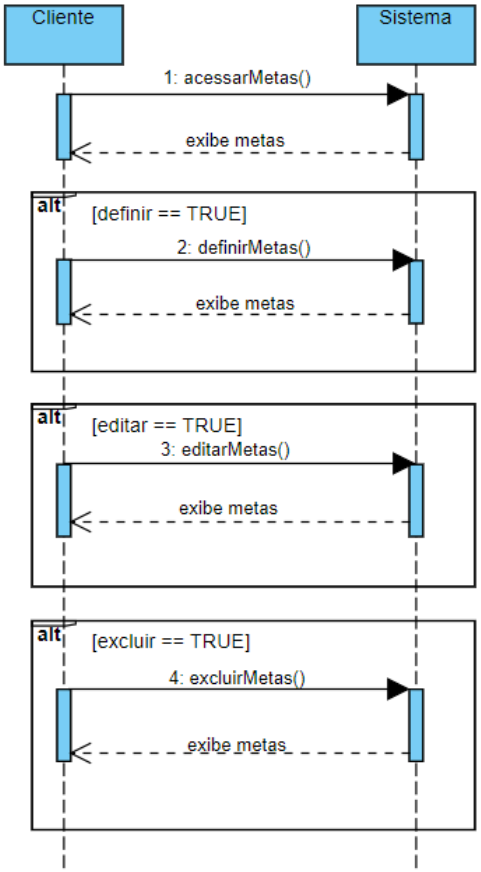
4.3.4. DSS: RECEBER NOTIFICAÇÃO



4.3.5. DSS: VISUALIZAR EXTRATO



4.3.6. DSS: MANTER METAS FINANCEIRAS



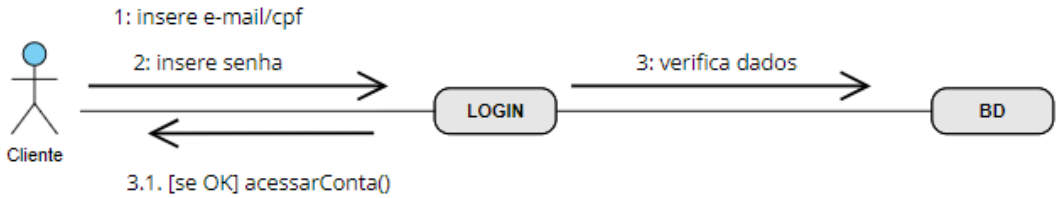
4.4. DIAGRAMAS DE INTERAÇÃO (COMUNICAÇÃO)

Os Diagramas de Interação são ferramentas utilizadas na modelagem de sistemas que proporcionam uma visão detalhada das interações dos objetos ao longo do tempo. Os principais diagramas de interação são os Diagramas de Sequência e os Diagramas de Comunicação.

Por opção, já que já temos o Diagrama de Sequência de Sistemas (DSS) acima, escolhemos representar essa modelagem através, então, do Diagrama de Comunicação.

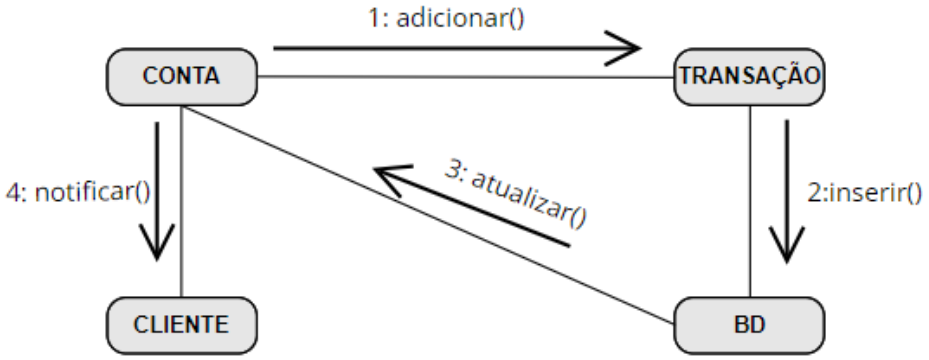
O diagrama de Comunicação oferece uma representação mais simplificada da interação entre objetos, enfatizando as relações estruturais e a troca de mensagens essenciais entre os objetos, sendo uma alternativa das mesmas informações dos diagramas de sequência. Nesse diagrama, a preocupação não está na ordem temporal em que as mensagens são trocadas, mas sim na organização estrutural dos objetos.

4.4.1. INSERIR CREDENCIAIS



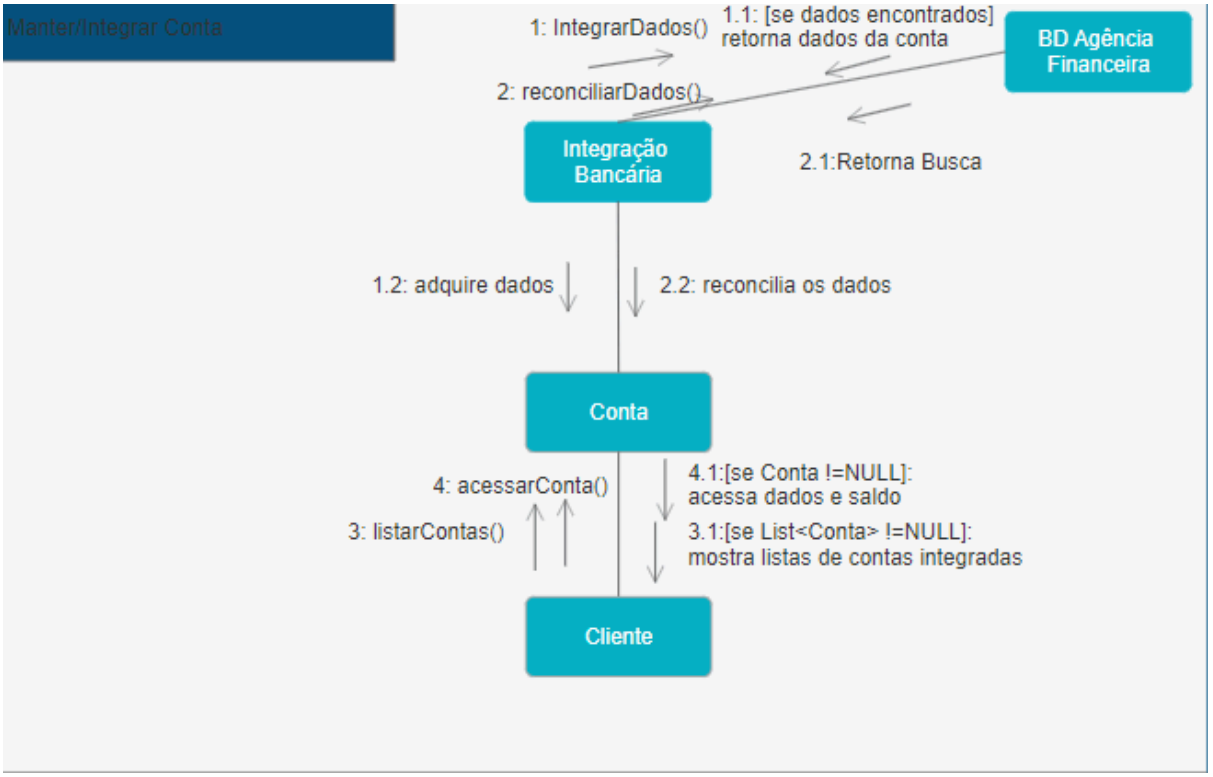
CONTRATO DA OPERAÇÃO	
OPERAÇÃO:	inserirCredenciais()
REFERÊNCIAS CRUZADAS:	Caso de Uso: “Login”
PRÉ-CONDIÇÕES:	• O cliente existe e sabe seu “ID”. • O cliente possui uma conta cadastrada no sistema.
PÓS-CONDIÇÕES:	• As credenciais foram inseridas pelo usuário. • O sistema autenticou o usuário e o associou a uma sessão.

4.4.2. ADICIONAR TRANSAÇÃO

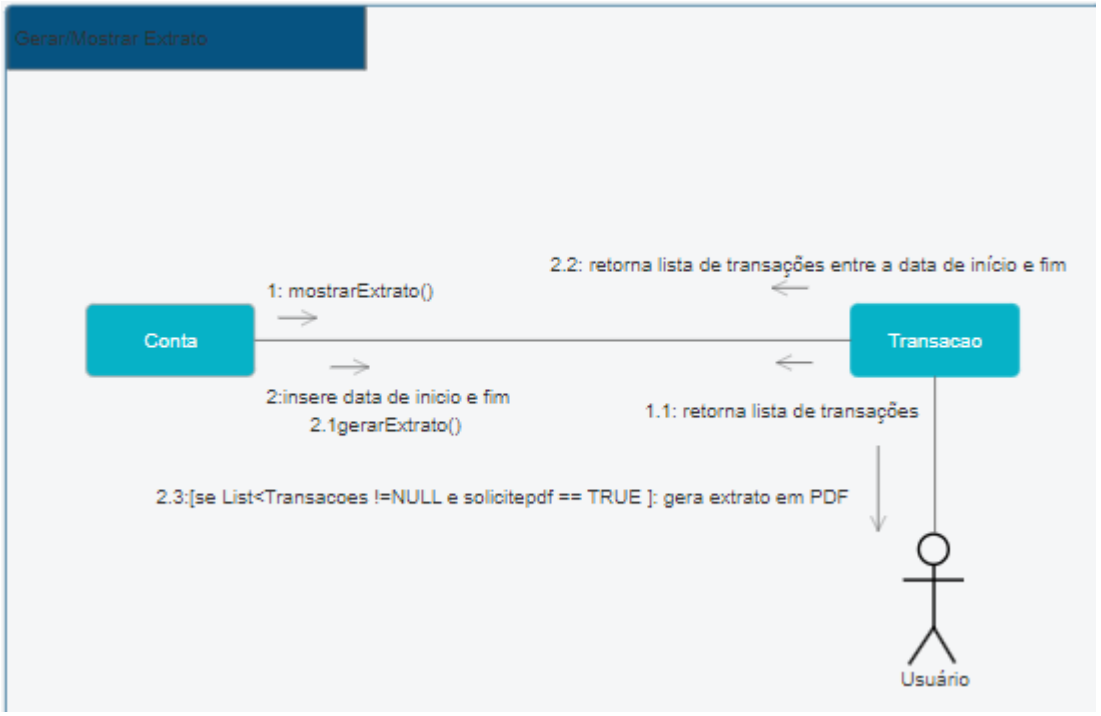


CONTRATO DA OPERAÇÃO	
OPERAÇÃO:	adicionarTransacao()
REFERÊNCIAS CRUZADAS:	Caso de Uso: “Manter Despesas/Receitas”
PRÉ-CONDIÇÕES:	• O cliente está logado no sistema. • O tipo de transação é especificada, receita ou despesa. • A descrição da transação é fornecida. • O valor da transação é fornecido.
PÓS-CONDIÇÕES:	• A transação é adicionada com sucesso ao sistema. • O saldo do usuário é atualizado no banco de dados do sistema.

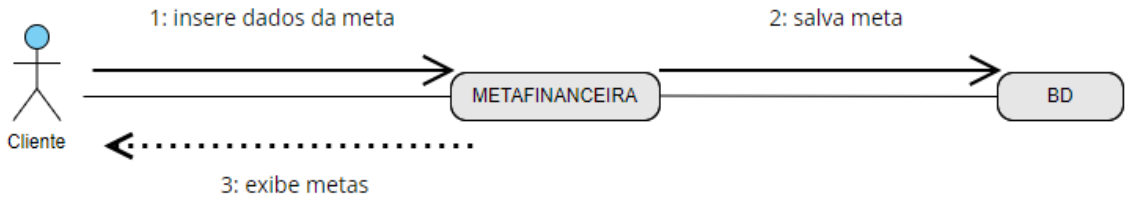
4.4.3. INTEGRAR/MANTER CONTA



4.4.4. VISUALIZAR EXTRATO



4.4.5. ADICIONAR METAS



CONTRATO DA OPERAÇÃO	
OPERAÇÃO:	adicionarMetas()
REFERÊNCIAS CRUZADAS:	Caso de Uso: “Manter Metas”
PRÉ-CONDIÇÕES:	• O cliente está logado no sistema. • Os dados da meta são válidos.
PÓS-CONDIÇÕES:	• A nova meta é adicionada à lista de metas.

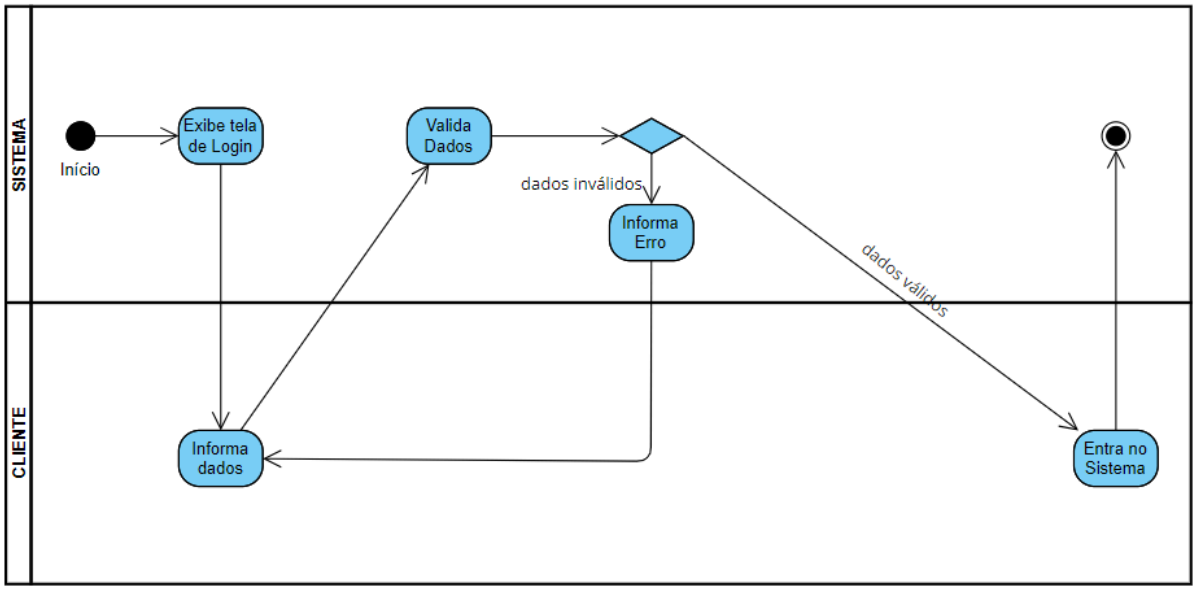
4.5. DIAGRAMAS DE ESTADO OU ATIVIDADES

O Diagrama de Estados, também conhecido como máquina de estados, se concentra em descrever o comportamento de um sistema em resposta a eventos que ocorrem, sejam eles internos ou externos. Com isso, esse diagrama serve para representar os diferentes estados que o sistema pode assumir ao longo do tempo e as transições desses estados.

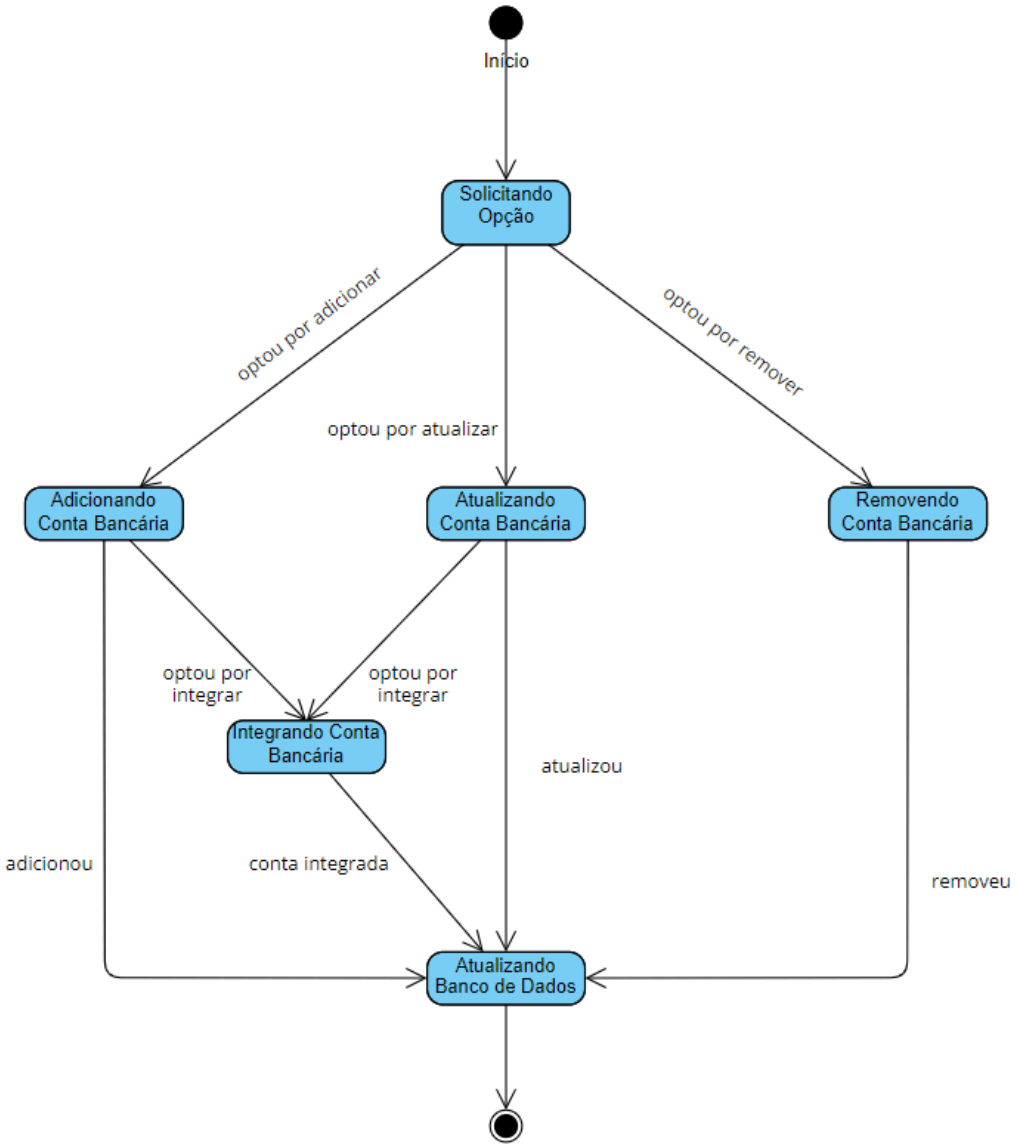
Já o Diagrama de Atividades se concentra em descrever o fluxo de trabalho do sistema ao mostrar atividades realizadas durante o processo, ou seja, fornece uma visualização do comportamento de um sistema descrevendo a sequência de ações em um processo.

Para o presente software em desenvolvimento, optamos por alguns diagramas de estado e atividades.

4.5.1. DIAGRAMA DE ATIVIDADES: LOGIN



4.5.2. DIAGRAMA DE ESTADO: MANTER CONTA BANCÁRIA



4.6. DIAGRAMA DE CLASSE DETALHADO

O Diagrama de Classes Detalhado, é uma representação visual da estrutura do sistema em uma perspectiva orientada a objetos. Ele se concentra em descrever as classes do sistema, bem como seus atributos e métodos, além do relacionamento entre elas. Segue abaixo o referido diagrama deste projeto: