Universidade Federal Rural do Semi-Árido
Campus Pau dos Ferros
Departamento de Engenharias e Tecnologia

Arthur Kellyson Pinheiro De Negreiros - 2020022215
Bruno Victor Paiva Da Silva - 2020011045
Pedro Makson Fontes Da Costa - 2021010616
Tiago Amaro Nunes - 2020011115

Sistema Bancário

Pau dos Ferros – RN
Outubro de 2023

Arthur Kellyson Pinheiro De Negreiros - 2020022215
Bruno Victor Paiva Da Silva - 2020011045
Pedro Makson Fontes Da Costa - 2021010616
Tiago Amaro Nunes - 2020011115

Sistema Bancário

Relatório apresentado à Universidade Federal Rural do Semi-Árido, na disciplina de Programação Orientada a Objetos, como requisito para avaliação na 3ª unidade, solicitado pelo professor Ítalo Augusto Sousa de Assis.

Pau dos Ferros – RN
Outubro de 2023

SUMÁRIO

1. Introdução

O projeto consiste no desenvolvimento de um sistema bancário, escrito na linguagem de programação Java, utilizando alguns dos principais conceitos do paradigma de orientação a objetos. O principal objetivo do sistema é fazer com que usuários do banco, Imperial Bank, possam se cadastrar no sistema a partir de suas credenciais informadas na etapa de cadastro e que após essa etapa e da de efetuar login, o usuário irá poder realizar os serviços do banco, tais como depositar, sacar, realizar uma transferência bancária com outro usuário, além de poder também solicitar empréstimos, esses que terão a aprovação do gerente antes de serem aceitos.

Etapas do projeto:

1. A primeira etapa ou no caso a primeira sprint do projeto além de simplesmente definir o tema, foi o levantamento de requisitos do sistema, onde foi definido os requisitos funcionais e os não funcionais que iriam compor o sistema.
2. A segunda etapa foi a elaboração de um diagrama de classe, que teve como objetivo facilitar o entendimento da estrutura do projeto, onde foi definido as entidades seguido das classes e respectivamente os atributos e métodos de cada classe.
3. Implementação do projeto foi a etapa seguinte, onde o que foi elaborado nas etapas anteriores seria colocado em prática.
4. Conexão com o banco de dados, que foi uma etapa que ocorreu basicamente de forma simultânea com a etapa anterior, o banco de dados que foi bastante útil para armazenar diversas informações e/ou dados importantes.
5. A etapa de alguns testes para revisão das funcionalidades do sistema, para verificar algumas correções de erros de código.

1.1. Equipe de desenvolvimento

Arthur Kellyson Pinheiro De Negreiros - 2020022215
Bruno Victor Paiva Da Silva - 2020011045
Pedro Makson Fontes Da Costa - 2021010616
Tiago Amaro Nunes - 2020011115

2. Desenvolvimento

2.1. Funcionalidades implementadas

Pode-se dizer que o início do projeto a partir da etapa de levantamento dos requisitos foi bastante satisfatório, devido a já experiência da equipe neste passo, o que impulsionou a agilizar em um primeiro momento a etapa de desenvolvimento do diagrama de classe, que foi em se tratando de maneira mais cautelosa pela equipe, pois era algo basicamente um pouco complicado de elaborá-lo no início, devido a algumas dúvidas sobre algumas funcionalidades de alguns métodos, mas a partir de uma das aulas de dúvidas, mas precisamente a aula sobre o próprio desenvolvimento do diagrama, ficou mais auto explicativo e a equipe conseguiu ter uma crescente evolução do projeto.

No desenvolver do projeto no quesito de implementar o código, houve algumas oscilações, muito na questão de modular algumas ideias definidas inicialmente que precisavam necessariamente serem revistas e subsequentemente implementar uma nova solução para aquela devida mudança e reformular alguns conceitos previstos na ideia inicial.

Na questão da etapa definida como conexão do banco de dados, foi algo implementado com mais tranquilidade tendo em vista que os discentes da equipe estão ou já pagaram a disciplina que facilitou a evolução desta etapa.

Na etapa final, houve alguns testes para verificar se as funcionalidades do programa estavam satisfazendo o que foi planejado e o que foi modificado com o decorrer principalmente do que foi modificado na parte prática do projeto.

Os maiores obstáculos encontrados foram mais por conta da pouca disponibilidade da equipe, pois alguns trabalham e outros estão envolvidos em projetos de pesquisa o que acaba consumindo parte do tempo, outro fator bastante importante que deve ser destacado como obstáculo é a questão dos horários de aulas dos discentes que são um pouco de fato diferentes o que dificultava a organização de reuniões tanto que era planejado no inicio realizar a implementação de uma interface gráfica, porém devido a estes fatores relacionados foi imediatamente descartado.

Por fim, algumas dúvidas que foram surgindo na implementação de alguns métodos e etapas a serem desenvolvidas do projeto, foram bem esclarecidas nas

aulas de orientação ao projeto e em determinado momentos pelo monitor da disciplina nos horários de monitoria.

Diagrama de Classe:

Na criação do diagrama de classe pela equipe, foi constituída a ferramenta Astah UML, que teve como objetivo trazer um esboço da estrutura do projeto, foi determinado que haveria quatro entidades para o projeto que seriam: Conta, Empréstimo, Gerente e Usuário.

Segue abaixo o modelo do diagrama de classe:

- **Classes auxiliares (de controle):**

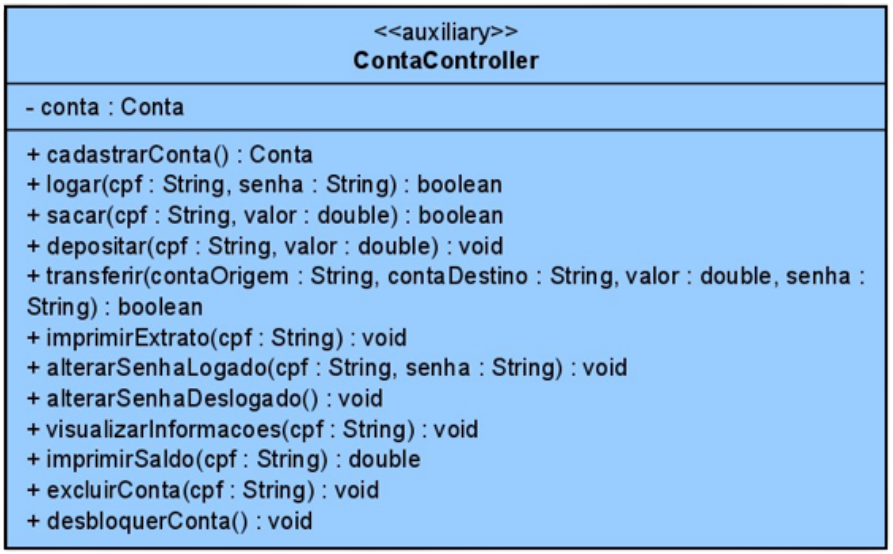

Figura 01: Diagrama de classe *ContaController*

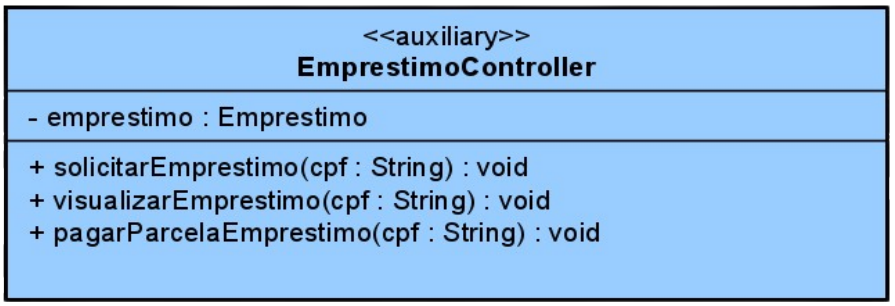

Figura 02: Diagrama de classe *EmprestimoController*

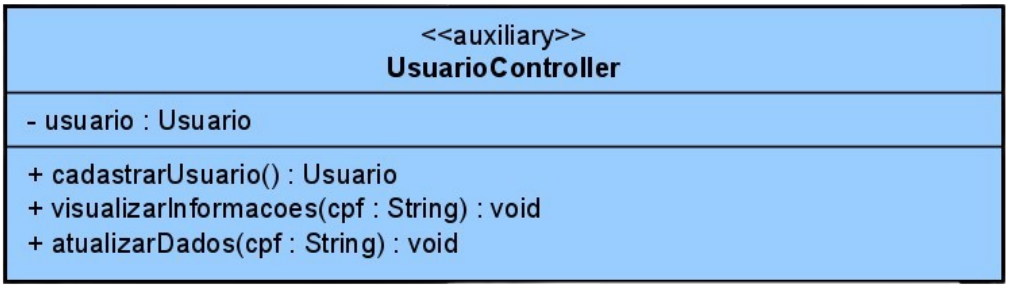

Figura 03: Diagrama de classe *UsuarioController*

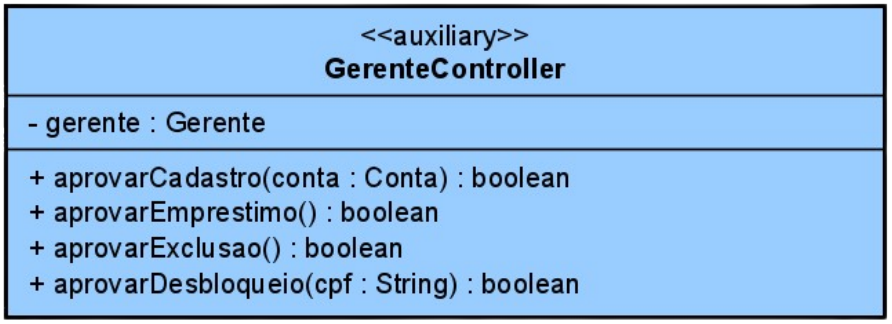

Figura 04: Diagrama de classe *GerenteController*

- **Classes entidades:**

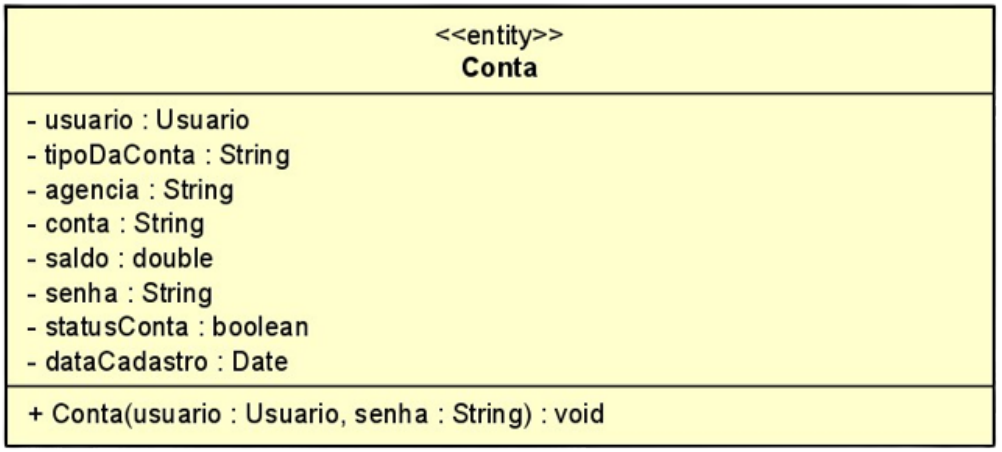

Figura 05: Diagrama de classe entidade *Conta*

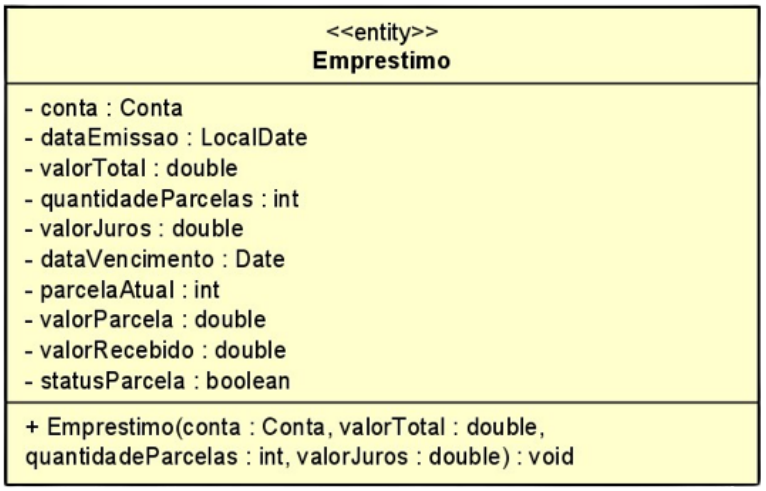

Figura 06: Diagrama de classe entidade *Emprestimo*

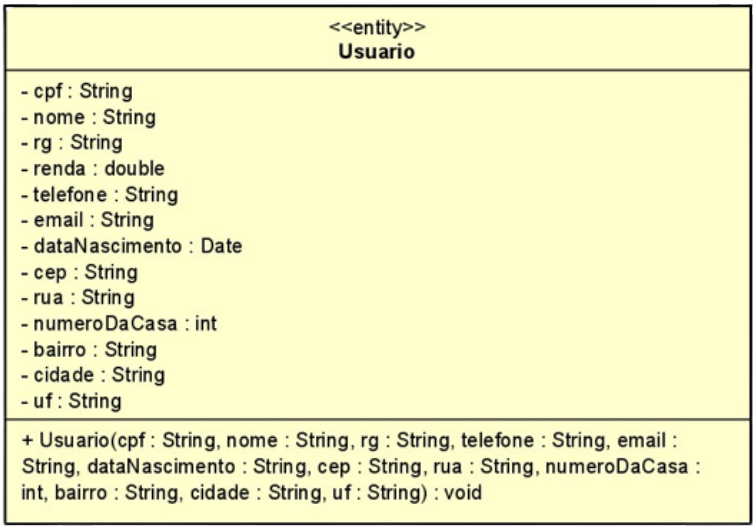

Figura 07: Diagrama de classe entidade *Usuario*

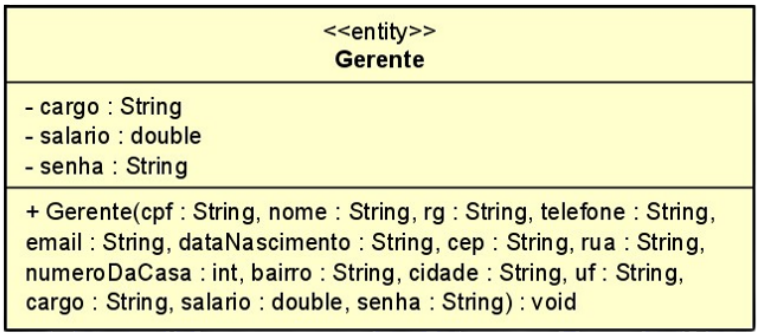

Figura 08: Diagrama de classe entidade *Gerente*

- **Diagrama de classe completo:**

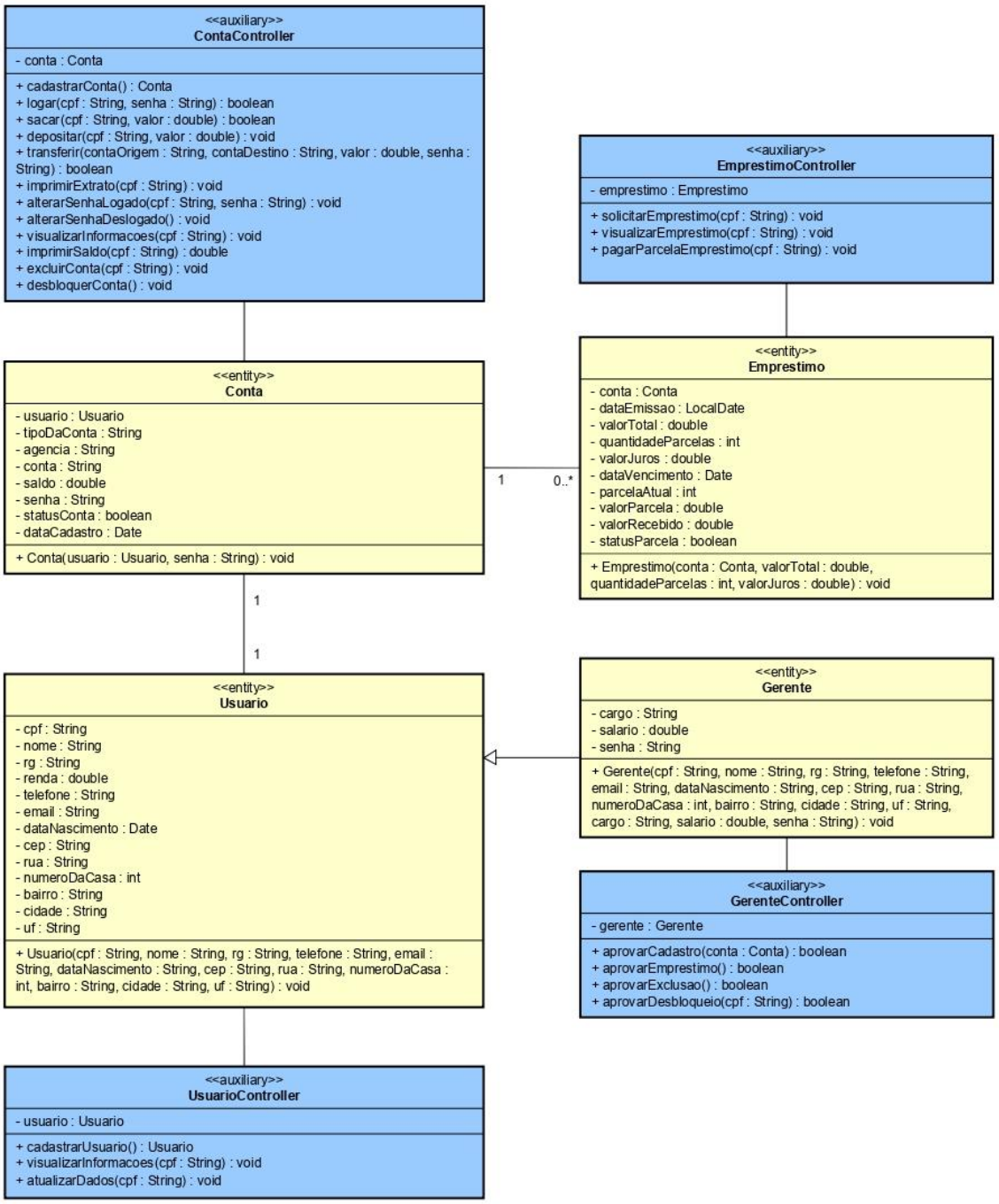

Figura 09: Diagrama de classe

Banco de dados:

Na criação do banco de dados foi utilizado o sistema de gerenciamento de banco de dados pgAdmin 4 ou no caso o PostgreSQL como ferramenta, e software Apache Maven para tratar as devidas dependências do java e padronização dos arquivos referente ao projeto.

No banco de dados utilizando a linguagem SQL, foi criado as tabelas a seguir:

```sql
CREATE TABLE Usuarios (
    cpf CHAR(11) NOT NULL,
    nome VARCHAR(100) NOT NULL,
    rg CHAR(9) NOT NULL,
    renda NUMERIC(10, 2) NOT NULL,
    telefone CHAR(11) NOT NULL,
    email VARCHAR(100) NOT NULL,
    dataNascimento DATE NOT NULL,
    cep CHAR(9) NOT NULL,
    rua VARCHAR(100) NOT NULL,
    numeroDaCasa INT NOT NULL,
    bairro VARCHAR(100) NOT NULL,
    cidade VARCHAR(100) NOT NULL,
    uf CHAR(2) NOT NULL,
        PRIMARY KEY (cpf)
);
```

Figura 10: Criação da tabela *Usuarios*

Tabela referente ao armazenamento das credenciais do usuário, com alguns campos sendo especificados o tipo de dados que devem ser digitados e a quantidade limite de caracteres que podem ser usados. como por exemplo “cpf CHAR(11) NOT NULL” onde é pedido que o campo cpf seja do tipo char e seja validado se tiver 11 dígitos.

```sql
CREATE TABLE Gerente (
    cpf CHAR(11) NOT NULL,
    nome VARCHAR(100) NOT NULL,
    rg CHAR(8) NOT NULL,
    telefone CHAR(11) NOT NULL,
    email VARCHAR(100) NOT NULL,
    dataNascimento DATE NOT NULL,
    cargo VARCHAR(50) NOT NULL,
    salario NUMERIC(10, 2) NOT NULL,
    senha CHAR(6) NOT NULL,
    cep CHAR(9) NOT NULL,
    rua VARCHAR(100) NOT NULL,
    numeroDaCasa INT NOT NULL,
    bairro VARCHAR(100) NOT NULL,
    cidade VARCHAR(100) NOT NULL,
    uf CHAR(2) NOT NULL,
    PRIMARY KEY (cpf)
);
```

Figura 11: Criação da tabela *Gerente*

Na tabela gerente vai funcionar na prática semelhante a tabela Usuarios, onde será pega as credenciais do gerente que está sendo cadastrado, mudando apenas alguns campos no momento do cadastro.

```sql
CREATE TABLE Conta (
    cpf CHAR(11) NOT NULL,
    tipoconta VARCHAR(50) NOT NULL,
    agencia CHAR(4) NOT NULL,
    conta CHAR(6) NOT NULL,
    saldo NUMERIC(10,2) NOT NULL,
    senha CHAR(6) NOT NULL,
    statusconta BOOLEAN NOT NULL,
    datacadastro DATE DEFAULT CURRENT_DATE NOT NULL,
    FOREIGN KEY (cpf) REFERENCES Usuarios
);
```

Figura 12: Criação da tabela *Conta*

Na tabela conta irá conter as informações da conta do usuário, como pode-se ver na tabela acima, o seu cpf que será sua chave primária, dentre outros como o saldo, tipo de conta e sua senha.

```
CREATE TABLE Emprestimo (
        cpf CHAR(11) NOT NULL,
    documento VARCHAR NOT NULL,
    dataemissao DATE DEFAULT CURRENT_DATE NOT NULL,
    valortotal NUMERIC(10,2) NOT NULL,
    quantidadeparcelas INTEGER NOT NULL,
    valorjuros NUMERIC(4,2) NOT NULL,
    datavencimento DATE NOT NULL,
    parcelaatual INTEGER NOT NULL,
    valorparcela NUMERIC(10,2) NOT NULL,
    valorrecebido NUMERIC(10,2) NOT NULL,
    statusparcela BOOLEAN DEFAULT FALSE NOT NULL,
    FOREIGN KEY (cpf) REFERENCES Usuarios
);
```

Figura 13: Criação da tabela *Emprestimo*

A tabela Emprestimo irá receber as informações das atividades do usuário referente a realização de empréstimos em sua conta.

```
CREATE TABLE Transacoes (
        cpf CHAR(11) NOT NULL,
    descricao VARCHAR(150) NOT NULL,
    valor NUMERIC(10, 2) NOT NULL,
        data DATE DEFAULT CURRENT_DATE NOT NULL,
        FOREIGN KEY (cpf) REFERENCES Usuarios
);
```

Figura 14: Criação da tabela *Transacoes*

A tabela Trasacoes, vai armazenar as informações das transações realizadas pelo usuário da conta.

- **Requisitos do sistema:**

[RF001] Cadastrar usuário.

O sistema deve permitir que o usuário realize um cadastro a partir das informações de suas seguintes credenciais (cpf, nome, rg, renda, telefone, email, data de nascimento, cep, rua, numero da casa, bairro, cidade e uf).

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF002] Login.

O sistema deve permitir que o usuário realize login no sistema a partir das informações de suas credenciais CPF e senha.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF003]Menu.

Após efetuar o login, o sistema deve apresentar uma tela de menu com as seguintes opções disponíveis para o usuário: Ver Extrato, sacar, depositar, transferir, Empréstimo, atualizar dados, alterar senha, ver saldo, ver informações da conta, deletar conta, sair.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF004] Ver Extrato.

O sistema deve possibilitar que o usuário visualize suas informações bancárias.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF005] Sacar.

O sistema deve possibilitar que o usuário possa sacar um valor de <= ao saldo.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF006] Transferir.

O sistema deve possibilitar que o usuário possa realizar transferências entre contas do mesmo banco.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF007] Depositar.

O sistema deve possibilitar que o usuário deposite um valor.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF008] Empréstimo .

O sistema deve possibilitar que o usuário realize uma solicitação ao gerente um empréstimo de acordo com o valor de seu saldo e que ele possa pagar as parcelas de seu empréstimo, visualizar empréstimo e voltar ao menu.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF009] Ver saldo.

O sistema deve possibilitar que o usuário visualize o seu saldo bancário.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF0010] Logout.

O sistema deve possibilitar que o usuário possa sair de sua conta a partir da opção de Logout .

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF011] Deletar conta.

O sistema deve possibilitar que o usuário possa apagar sua conta a partir da opção Deletar conta.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF012] Aceitação para conta.

O sistema deve possibilitar que o gerente realize se sim ou não que o usuário possa ter acesso ao sistema a partir de seu cadastro.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF013] Aprovação de empréstimo.

O sistema deve possibilitar que o gerente possa aprovar ou recusar uma solicitação de empréstimo do usuário.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF014] Login gerente.

O sistema deve possibilitar que o gerente possa fazer login no sistema a partir de suas credenciais.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF015] Logout gerente.

O sistema deve possibilitar que o gerente possa realizar sua saída do próprio sistema pela opção de sair.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF016] Atualizar dados.

O sistema deve possibilitar que o usuário possa atualizar a suas credenciais.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF017] Alterar senha.

O sistema deve possibilitar que o usuário possa alterar sua senha.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF018] Ver informações da conta.

O sistema deve possibilitar que o usuário possa visualizar as informações de sua conta.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

1 Requisitos Não Funcionais

Exemplos:

[RNF001] Restrição de Cadastro.

O sistema só deverá permitir efetuar o cadastro se o cliente for maior que 18 anos.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RNF 002] Validação de cadastro..

O sistema só poderá permitir um cadastro no sistema, se todos os campos obrigatórios forem preenchidos corretamente a partir de suas validações, pois nome completo não pode ter número ou CPF não pode ter letras. *Exemplo: TIAGO35* - seria um nome Inválido.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RF003] Limite de Saque.

O sistema deve possibilitar que o usuário possa apenas valores que sejam <= ao seu saldo atual.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RNF 004] Transferir valor.

O sistema só deverá permitir efetuar uma transferência bancária caso o valor desejado a ser transferido seja <= ao saldo atual.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

[RNF 005] Restrição de Login.

O sistema só deverá permitir que o acesso só seja efetuado se houver as informações das credenciais CPF e senha, desde que as mesmas estejam válidas no sistema.

Prioridade: (X) Essencial ( ) Importante ( ) Desejável

- **Explicação sobre as classes do sistema:**

Classe: Usuario.java

Descrição: A classe Usuario.java representa um modelo de entidade para armazenar informações sobre um usuário em um sistema. Ela encapsula uma série de atributos que descrevem um usuário, incluindo detalhes pessoais e de endereço

Classe: Gerente.java

Descrição: A classe Gerente.java é uma subclasse da classe Usuario.java e representa um modelo de entidade específico para gerentes em um sistema. Ela herda características e atributos da classe Usuario.java e adiciona informações adicionais específicas para gerentes

Classe: Emprestimo.java

Descrição: A classe Emprestimo.java representa um modelo de entidade que é usado para gerenciar informações relacionadas a empréstimos em um sistema. Ela é útil para representar e gerenciar informações sobre empréstimos, permitindo que um sistema acompanhe o estado de cada empréstimo, a data de vencimento, o valor das parcelas e outros detalhes relacionados aos empréstimos feitos por clientes ou usuários.

Classe: Conta.java

Descrição: A classe Conta.java representa um modelo de entidade para representar informações relacionadas a contas bancárias no sistema. Ela armazena detalhes sobre as contas, incluindo informações sobre o titular da conta, tipo de conta, número da agência, número da conta, saldo, senha, status da conta e data de cadastro

Classe: Conexao.java

Descrição: A classe Conexao.java é responsável por criar e gerenciar uma conexão com um banco de dados PostgreSQL.

Classe: UsuarioDAO.java

Descrição: A classe UsuarioDAO.java é responsável por interagir com o banco de dados para realizar operações relacionadas a usuários, como inserir, buscar e obter informações de usuários. Ela utiliza o padrão DAO (Data Access Object) para encapsular a lógica de acesso ao banco de dados e oferece métodos para realizar tarefas específicas relacionadas a usuários

Classe: GerenteDAO.java

Descrição: A classe GerenteDAO.java é responsável por realizar operações de acesso a dados relacionados a gerentes do sistema. Ela encapsula a lógica de verificação de credenciais de um gerente, como o número de CPF e a senha, no banco de dados.

Classe: EmprestimoDAO.java

Descrição: A classe EmprestimoDAO.java é responsável por realizar operações de acesso a dados relacionados a empréstimos do sistema. Ela encapsula a lógica de inserção de empréstimos no banco de dados, consulta de empréstimos por CPF, pagamento de parcelas e outras operações relacionadas a empréstimos.

Classe: ContaDAO.java

Descrição: A classe ContaDAO.java é responsável por realizar operações de acesso a dados relacionados a contas de usuários do sistema. Ela encapsula a lógica de inserção de contas no banco de dados, validação de usuário, gerenciamento de saldo, registros de transações e outras operações relacionadas a contas.

E temos as classes de controle (UsuarioController.java, GerenteController.java, EmprestimoController.java, ContaController.java) do padrão arquitetura MVC que no contexto do Maven, que é uma ferramenta de automação de compilação e gerenciamento de dependências, as classes de controller são apenas um componente de um projeto Java mais amplo. O Maven ajuda a gerenciar as dependências e a construção do projeto, mas a implementação das classes de controller é específica do aplicativo e das estruturas que você está usando.

- **Alguns dos Resultados:**

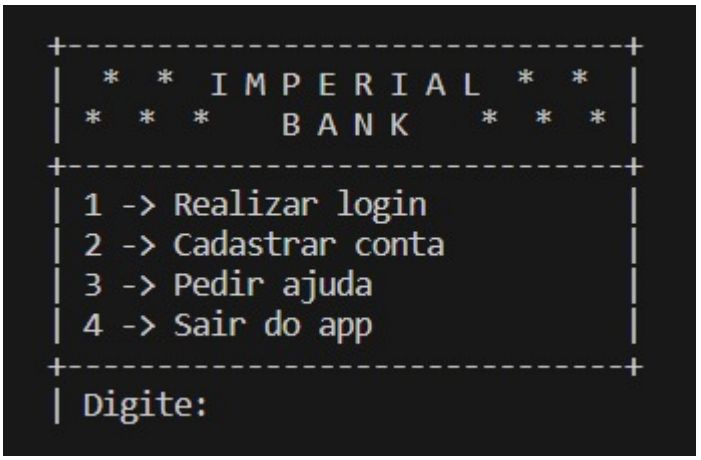

Figura 15: Tela inicial

Figura 16: Tela de menu

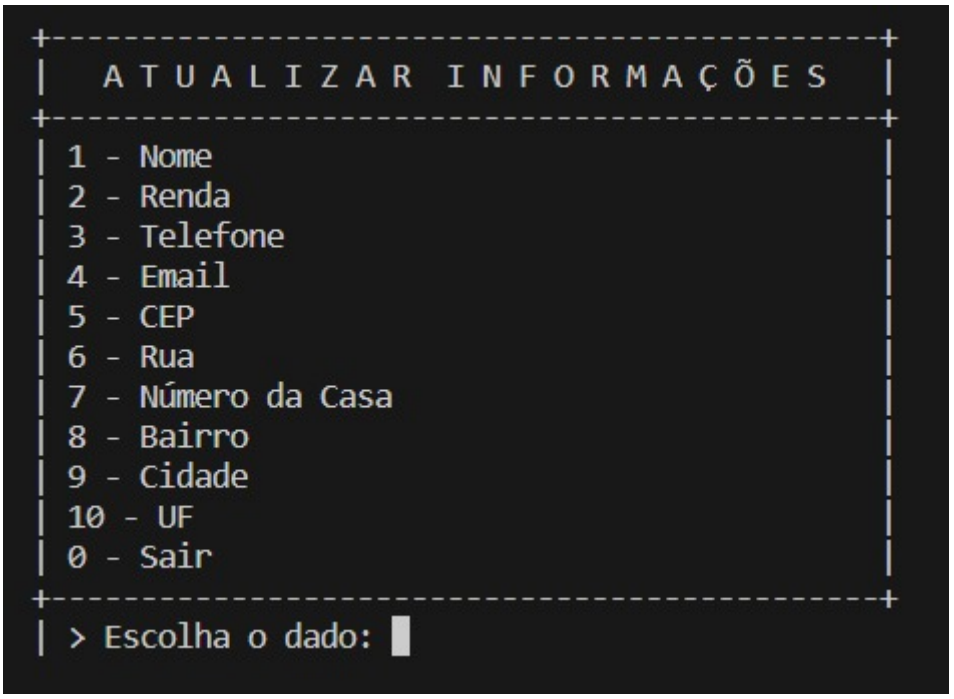

Figura 17: Tela de atualizar informações

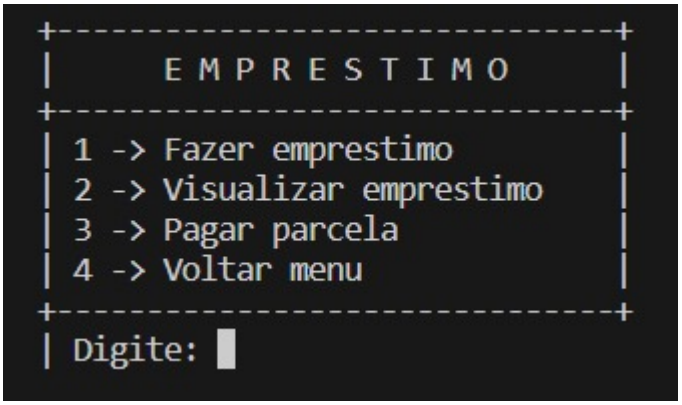

Figura 18: Tela de Emprestimos

2.2. Funcionalidades não concluídas

Perante ao que se foi planejado na etapa de requisitos e do diagrama de classe podemos dizer que não houve nenhuma funcionalidade que deixou de ser concluída, além do mais acabou que surgindo algumas outras a mais que precisassem ser implementadas no código.

2.3 Funcionalidades futuras:

É deixado em aberto para prováveis aperfeiçoamentos do sistema futuramente, como uma interface gráfica e outras funcionalidades (como por exemplo, o usuário escolher pagar o empréstimo com o saldo da conta, habilitar o débito automático para pagar parcelas futuras, fazer mais funcionalidades para o Gerente) ter outros tipos de conta (adicionar conta poupança) e trabalhar com clientes jurídicos (além do físico) no Imperial Bank.

2.4. Cronograma das atividades realizadas

Data	Autor	Descrição da atividade
25/09	Arthur, Bruno, Pedro e Tiago	Planejamento da ideia inicial, verificação do modelo de relatório e criação do repositório GitHub.
26/09	Arthur, Bruno, Pedro e Tiago	Planejamento inicial do diagrama de classe, coleta dos requisitos e discussão sobre algumas metodologias e ferramentas a serem utilizadas.
27/09	Arthur, Bruno, Pedro e Tiago	Modelo inicial do diagrama de classe finalizado, assim como a coleta de requisitos, e início da implementação do código.
28/09	Arthur, Bruno, Pedro e Tiago	Organização da estrutura do código utilizando uma das ferramentas da discussão do dia 26/09, o Maven, e iniciando o desenvolvimento das classes/entidades.
29/09	Arthur, Bruno, Pedro e Tiago	Desenvolvimento de classes que envolvem Usuário, Conta e Gerente (entidade, DAO, Controller).
30/09	Arthur, Bruno, Pedro e Tiago	Tentativas de organização do código e de conexão com o Banco de Dados.
01/10	Arthur, Bruno, Pedro e Tiago	Implementação da classe principal, e verificação das primeiras execuções do algoritmo.
02/10	Arthur, Bruno, Pedro e Tiago	Criação da classe empréstimo, e validação de alguns campus de cadastro de usuário que estavam dando problema.
03/10	Arthur, Bruno, Pedro e Tiago	Alguns ajustes na conexão com o banco de dados que estavam dando problema, as ferramentas que estão sendo utilizadas, Maven, PostgreSQL e VScode.
04/10	Arthur, Bruno, Pedro e Tiago	Atualização das classes Usuario.java e Conta.java, implementando novos métodos.
05/10	Arthur, Bruno, Pedro e Tiago	Criação de métodos na classe Gerente.java para que se possa validar emprestimos da classe Emprestimo.java para o usuario do sistema.
06/10	Arthur, Bruno, Pedro e Tiago	Realizando o tratamento de alguns erros nas respectivas classes do projeto, consequentemente implementando algumas modificações que estavam faltando
07/10	Arthur, Bruno, Pedro e Tiago	Finalizando as modificações finais nas classes e atualizando o diagrama de classe.
08/10	Arthur, Bruno, Pedro e Tiago	Envio do Projeto.

3. Conclusão

O projeto tem o intuito de proporcionar a possíveis clientes do Banco, nomeado como Imperial Bank, um cadastro em seu sistema para que se possa realizar atividades necessárias que são realizadas em um banco. Isso foi proporcionado a partir de implementações em linha de código com a linguagem de programação Java fazendo uso dos conceitos do paradigma de orientação a objetos.

Além disso, a implementação realizada, deixa em aberto um possível aperfeiçoamento do projeto para que se tenha melhorias nas funcionalidades de um banco mais completo com uma maior variedade de tipo de contas a serem proporcionadas.

Portanto podemos concluir que apesar dos conhecimentos adquiridos com um projeto que é colocado na prática, houve algumas dificuldades, principalmente pelos motivos listados em tópicos anteriores, o que limitou em alguns aspectos o desenvolvimento do projeto, mas que trouxe uma boa oportunidade para praticar aquilo que foi absorvido perante ao longo do curso, e principalmente da disciplina de programação orientada a objetos.