

**PROJETO INTERDISCIPLINAR**

**JEEV**

**BANK**

## Soluções inteligentes para aplicações financeiras

**Alunos:**

|  |  |
| --- | --- |
| **RGM** | **Nome** |
| 31696031 | Emily Vitoria Ramos dos Santos |
| 31016006 | Erick Alan Oliveira Moura |
| 31235417 | Jonnathan Costa |
| 31236294 | Vinicius Peres Oliveira |
|  |  |

São Paulo

2023

**UNIVERSIDADE CRUZEIRO DO SUL**

**PROJETO INTERDISCIPLINAR**

**JEEV**

**BANK**

Soluções inteligentes para aplicações financeiras

## Trabalho apresentado como parte do requisito para aprovação na Disciplina de Projeto Interdisciplinar do curso de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Cruzeiro do Sul.

**Orientadores:** Prof. Agnaldo Silibert Mota e Prof. Fabio Pereira da Silva

São Paulo

2023**Sumário**

[1. Apresentação: 3](#_Toc48489109)

[1.1 Justificativa e Motivação 3](#_Toc48489110)

[1.2 Dados do Sistema. 3](#_Toc48489111)

[2 Análise e Projetos de Sistema 3](#_Toc48489112)

[3 Engenharia de Software 3](#_Toc48489113)

[4 Consideração finais 3](#_Toc48489114)

[5 Referencias 3](#_Toc48489115)

# 1. Apresentação:

## 1.1 Justificativa e Motivação

Este projeto oferece os desafios à altura de um curso de Análise e Desenvolvimento de Sistemas, pois reúne, de uma só vez, habilidades adquiridas em diversas áreas da computação, em especial àquelas voltadas ao Análise e Projetos de Sistemas I e Engenharia de Software, além de incentivar a pesquisa.

## 1.2 Dados do Sistema.

JEEV Bank é um software para aplicações bancárias que permite a utilização de operações rotineiras (depósito, saque, transferências, cartões etc.) de modo dinâmico, moderno e intuitivo. Para alcançar o objetivo desejado utilizamos de técnicas de programação e engenharia de software. Para a composição do software utilizamos das seguintes linguagens e tecnologias: HTML, CSS, Javascript, Node.js e MySQL.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Figura 1- Tela de Cadastro JEEV Bank.

Acima a figura 1 representa nossa tela de cadastro com as solicitações necessárias para a abertura de uma conta de pessoa física ou jurídica. Após obter todos os dados é feita uma verificação pelo sistema e, não havendo impedimentos, um pop-up aparecerá na tela com agência e conta do novo cliente para acesso ao sistema bancário.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamenteFigura 2- Tela de Login JEEV Bank

Acima a figura 2 representa a tela de login do JEEV Bank que poderá ser acessada através de agência, conta e senha previamente definida no sistema.

Tela de um aparelho eletrônico

Descrição gerada automaticamente

Figura 3- Tela de operações.

Acima a figura 3 representa a tela de operações do JEEV Bank, sendo possível executar transferências, pagamentos, consulta de débito e limite de crédito, transações com PIX, transações do tipo Ted e Doc, chat para contatar nossa rede de apoio ao cliente e consulta de extrato e movimentações.

JEEV Bank trás leveza às transações necessárias ao seu dia a dia, uma interface simples que te auxiliará no controle de suas operações em uma única tela, além de, obviamente, toda a credibilidade e segurança necessárias para trazer estabilidade financeira aos nossos clientes.

# 2 Requisitos de análise e projetos de sistema

Ao desenvolver um projeto de software, cabe aos analistas, a abstração de cenários que, a priori não fazem parte do cenário principal do projeto. Ou seja, deve-se analisar cenários que não assegurados pelo software.

A seguir segue-se prints de cenários acima citados que foram abstraídos das regras de negócio.

Texto

Descrição gerada automaticamente

Figura 4- Regras de negócio

Acima a figura 4 representa cenários possíveis dentro do segmento do qual o software propõe atuar, dentre eles: definição de limite para transações bancárias, definição de regras para saques e depósitos. Sendo os dois últimos citados, por necessidade de uma agência física para efetuá-los, possuidores de características únicas que transcendem o escopo atribuído ao software, portanto, faz-se necessário um conjunto de regras claras para que não haja nuances que possam trazer dúvidas, gerando descontentamento aos clientes.

# 3 Requisitos de engenharia de software

Os tópicos a seguir foram elaborados para atender a demanda de engenharia de software relacionadas ao projeto.

# 3.1 Diagrama de caso de uso

Diagrama de caso de uso descreve a funcionalidade proposta para um novo sistema. Exemplificando a interação do usuário com o sistema proposto e, auxiliando na composição dos requisitos funcionais. Abaixo segue-se o diagrama de caso de uso do JEEV Bank:

# 3.2 Requisitos FUNCIONAIS

Requisito funcional define uma função a um sistema de software ou seu componente. A seguir segue-se os requisitos funcionais atribuídos ao JEEV Bank:

pop-up: mostrando sua instituição agência e conta quando o usuário faz o login.

Cadastro da conta: Ao entrar no JEEV bank, o usuário deverá realizar o login com sua agência, logo após inserir sua conta e senha.

Validação da conta: Após alguns segundos o próprio sistema irá validar e autenticar sua conta.

Saques :O usuário deverá inserir seu cartão físico fazendo uma validação e tendo que inserir seu código de segurança e sua senha onde será mostrado, na tela, com todos os serviços disponíveis, onde deverá escolher a opção de saque para inserir o valor desejado e assim, sacar. Após isso será validado pelo sistema e sairá as cédulas de dinheiro.

PIX: Ao entrar na sua conta no JEEV bank e aparecerá a opção de pix, onde o usuário deve escolher para quem enviar o valor que desejar, o usuário deverá escolher entre as chaves pix disponíveis. Após escolher seu destinatário aparecerá a confirmação de sua senha e os dados pessoais, se tudo estiver correto, após uns segundos seu pix será efeituado.

Docs: Acessando o JEEV bank, na tela inicial terá a opção DOC, para ser feito uma transação de no máximo R$ 5.000.00, onde o usuário deve inserir o código do banco do destinatário, número da agência, tipo da conta, cpf ou cnpj, e o valor da transação desejada. Após haverá uma validação, se estiver tudo dentro dos requisitos, será processado e, após 2 dias úteis, será efeituada a transação.

Teds: Acessando o JEEV bank, na tela inicial terá a opção TED, onde o usuário deverá inserir os dados do número da agência, tipo da conta, cpf ou cnpj, e o valor da transação, onde diferente do DOC, o Ted será efetuado no mesmo dia. A transação será feita depois de validada pelo sistema.

Comprovante: O usuário recebe o comprovante toda vez que fizer uma transação, em questão segundos o JEEV bank verificará a transação e já aparecerá na tela o comprovante virtual.

Histórico: O histórico de todas as transações ficará na memória do sistema apto a ser acessado sempre que necessário.

Extratos: Os extratos serão gerados logo após a validação de sua transação.

Aplicação em conta: Após verificar seu saldo em conta, sua validação será direcionada para um fundo de investimento, o melhor para o seu saldo em conta.

Chat de atendimento: Nosso chat de atendimento será 24 horas, onde poderá tirar suas dúvidas no chat do JEEV bank.

Integração com outros bancos: Onde o usuário poderá realizar depósitos e transações com outros bancos.

Depósitos: O usuário pode ir até um banco 24 horas, e fazer seu depósito.

Agendamento :O usuário pode agendar suas transações para o dia e hora que quiser, mas deve aguardar a validação pelo sistema.

# 3.3 Requisitos não FUNCIONAIS

Requisitos não funcionais são os requisitos relacionados ao uso da aplicação em termos de desempenho. Usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas.

A seguir segue-se os requisitos não funcionais atribuídos ao JEEV Bank:

Segurança do aplicativo - Tudo dentro do nosso app JEEV Bank tem referências às medidas e técnicas utilizadas para proteger o aplicativo e os dados dos usuários contra possíveis ataques e vulnerabilidades.

Itoken - É um sistema de autenticação de dois fatores utilizado em transações bancárias e em outros serviços online para garantir a segurança das informações e evitar fraudes.

Atendimento 24 horas/ precisa de ajuda? - Em nosso app JEEV Bank tem disponibilidade de suporte ao cliente em tempo integral para esclarecer dúvidas, solucionar problemas e oferecer assistência técnica.

Consegue sacar em qualquer banco 24 horas - Na nossa JEEV Bank tem à capacidade do aplicativo de permitir que o usuário realize saques em caixas eletrônicos de qualquer banco a qualquer hora do dia.

Investimentos - Dentro do app JEEV Bank tem às opções de investimento disponíveis no aplicativo, como renda fixa, renda variável, fundos de investimento, entre outros.

Uso sobre cartão de créditos/débito - Tem um método referente a sua renda para que seja feita pesquisa nas suas compras dentro do aplicativo com cartões de crédito e débito, permitindo que o usuário faça pagamentos e transações por meio do aplicativo.

Descontos integrados ao uso do cartão - Refere-se a possibilidade de o usuário obter descontos e benefícios exclusivos ao usar o cartão de crédito ou débito vinculado ao aplicativo.

Portabilidade de salário - acessando essa opção no nosso JEEV Bank de transferir o salário diretamente para a conta do aplicativo, permitindo que o usuário gerencie seus recursos financeiros com mais facilidade.

Feedback do app - Acessando essa operação dentro do app JEEV Bank de feedback do usuário sobre o aplicativo, como avaliações, comentários e sugestões, que podem ser usadas para melhorar a qualidade e a usabilidade do aplicativo.

Auditabilidade: Todas as transações devem ser registradas em um log de auditoria.

Integridade: Os dados do usuário devem ser precisos e não devem ser alterados sem autorização

Confiabilidade- Seria garantir todos os direitos dos clientes.

Segurança de dados - No nosso JEEV bank temos segurança de dados é a proteção de informações contra acessos não autorizados, perda, roubo, danos ou outras ameaças que possam comprometer a integridade,

Credibilidade- No nosso JEEV temos uma característica onde temos conhecimento, habilidade, integridade, transparência, imparcialidade e consistência. Onde temos informação que é considerada confiável.

# 3.4 regras de negócio

Regras de negócio são declarações sobre a forma da empresa fazer negócio. Elas refletem políticas do negócio que buscam satisfazer os objetivos da empresa, satisfazer clientes, fazer bom uso dos recursos e obedecer às leis ou convenções gerais do negócio.

A seguir segue-se as regras de negócio do JEEV Bank:

O cliente deverá fornecer os documentos exigidos para cadastro no sistema bancário.

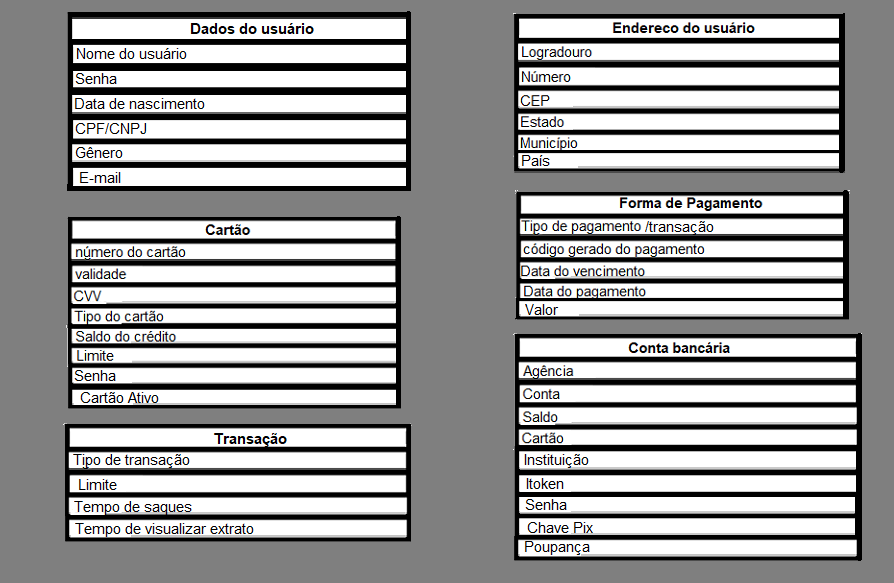


Figura 5 – Demonstração das regras relacionadas a cadastro de clientes

Deverá ser feita uma análise dos documentos junto aos órgãos competentes e, comprovada a veracidade, em até 24hs o cadastro do novo cliente deverá constar em nosso sistema.

Deverá ser elaborada as senhas de cartão de crédito/débito, uma senha eletrônica e validação por itoken para toda e qualquer movimentação dentro das aplicações.

Por questões relacionadas à segurança o cliente deverá cadastrar um número de celular para validação em duas etapas.

Ao realizar login em nossa instituição o cliente receberá um código de entrada por e-mail que deverá ser verificado após a inserção da senha, em um campo específico. Para realizar operações bancárias, um código itoken será enviado por sms. O código de entrada terá validade de 2 minutos e o itoken de 50 segundos.

Em caso de perda ou extravio dos cartões o cliente deverá comunicar ao banco o mais breve possível. O bloqueio dos cartões deve ser feito de imediato e, com a autorização do cliente, novos cartões serão emitidos. Caberá ao cliente a elaboração de novas senhas ou a permanência das senhas antigas, isentando a instituição de toda e quaisquer responsabilidades nelas referidas.

Em caso de perda ou esquecimento da senha eletrônica o cliente deverá constatar nosso suporte online. Para que haja êxito nesta solicitação é imprescindível a apresentação de documento com foto, com data de expedição de até 8 (oito) anos, e a aceitação de uma chamada de vídeo para verificação do cliente e os dados informados.

Movimentações do tipo TED devem ser realizadas em dias úteis até as 16hs59mnts horário de Brasília. O valor será creditado na conta do destinatário até as 16hs59mnts do próximo dia útil. Após o horário referido não aceitar este tipo de transferência.

Movimentações do tipo DOC devem ser realizadas em dias úteis às 21hs59mnts horário de Brasília. O valor será creditado em até 1 (um) dia útil. Após o horário referido o prazo será de 2 (dois) dias úteis.

O limite diário para transações TED/DOC é de: R$ 10.000,00. Considera-se limite total para ambas as transações. Para aumento deste limite deve-se consultar o gerente com tempo hábil de, no mínimo, 24hs.

Transações do tipo PIX não possuem limites de valores e horários estabelecidos pela instituição bancária. Entretanto, ao realizar o cadastro o cliente deve estabelecer um limite diurno (8hs às 18) e um limite noturno (19hs às 7hs). Para qualquer alteração do limite pré-estabelecido, deve-se contatar o setor financeiro, seja por telefone, seja por chat, com no mínimo, 24hs de antecedência.

Ao ser cadastrado um novo cliente, o referido deve estabelecer o limite de saques diários, sendo assim, quaisquer alterações relacionadas a valores de saques devem ser realizadas através de um dos nossos canais de suporte com, no mínimo, 24hs de antecedência.

Devido atuarmos apenas no mercado digital, para a realização de saques, o cliente deve procurar um caixa eletrônico 24hs e, devido a gastos operacionais relacionados aos caixas eletrônicos, será cobrado o valor de: R$5,99 por operação, que será debitado do valor em conta antes de qualquer transação.

Em relação a depósitos deve-se priorizar transferências bancárias, das quais não haverá taxa de utilização. Entretanto, caso seja necessário o depósito utilizando um caixa eletrônico segue-se a regra estipulada para os saques. Cabe esclarecer ao referido que a creditação do valor em conta, deverá passar pelo crivo da instituição, caso o valor depositado seja o mesmo valor descrito na declaração do envelope será creditado em conta em até 24hs. Caso haja divergência, será considerado o valor que consta no envelope.

Deve-se disponibilizar para o cliente o histórico de transações e ou extrato dos últimos 180 dias, automaticamente, através da solicitação disponível no menu de opções. Para solicitações acima do citado deve-se contratar nosso suporte.

Para disponibilidade de crédito deverá ser consultado se há restrições no nome do cliente através SPC SERASA. Caso não conste pendências o seguinte cálculo deve ser considerado para limite de crédito:

RM+40% = Lt, sendo RM renda mensal comprovada e lt limite total de crédito.

Para atrasos referentes a fatura de cartão deve-se considerar as regras e base de cálculo abaixo:

R$ 0,56 de juros de mora per dia.

Dez por cento do valor da fatura ao mês.

Para atrasos referentes a empréstimos deve-se considerar as regras e base de cálculo abaixo:

R$ 1,50 de juros de mora per dia.

20 por cento de multa por atraso ao mês.

# 3.5 Diagrama de classes

A modelagem para a compreensão de como o sistema deve operar é imprescindível para alcançar os objetivos propostos. Uma das principais ferramentas para esse gerenciamento são os diagramas de classe (UML) que, servem de modelo para a representação do sistema. A seguir segue-se do diagrama de classes do JEEV Bank:

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Figura 6 – Diagrama de classes do sistema JEEV Bank.

A seguir segue-se diagrama de classes do nosso banco de dados:

Tabela

Descrição gerada automaticamente

Figura 7 – Diagrama de classes do Banco de Dados JEEV Bank

# 3.6 fluxograma

Fluxograma é um diagrama que descreve um processo, sistema ou algoritmo. O fluxograma a seguir descreve a operação do sistema JEEV Bank.

Diagrama

Descrição gerada automaticamenteFigura 8 – Fluxograma Início e tela de cadastro.

A figura 8 demonstra o fluxo do início da aplicação e da tela de cadastro. Pegando como exemplo a tela de cadastro, o novo cliente irá inserir os dados necessários e após a verificação dos dados será levado a tela de login. A mesma lógica é aplicada as demais funcionalidades do sistema que serão abordadas na apresentação.

# 4 requisitos de programação

Após a elaboração dos requisitos de engenharia de software, seguimos com a elaboração da programação com base nos elementos anteriores acima descritos e relacionados. Para o desenvolvimento deste projeto utilizamos de Javascript para toda a parte “funcional” do software, HMTML5 e CSS3 para a elaboração visual da página e node.js para a implementação de rotas e integração com banco de dados. Para o banco de dados utilizamos o MySQL Workbench. Nos tópicos a seguir demonstraremos o cumprimento dos requisitos e técnicas utilizadas na implementação dos códigos.

# 4.1 implementação da tela de login

Demonstraremos a seguir, através de prints e comentários, as técnicas utilizadas para a implementação da tela de login do nosso sistema.

 <link rel="stylesheet" href="./login.css">

    <title>Login</title>

</head>

<body>

    <div class="container">

        <div class="card">

            <h1> Entrar</h1>

            <div class="label-float">

                <input type="text" id="agencia" placeholder="" required>

                <label id="agenciaLabel" for="usuario">Agência</label>

            </div>

            <div class="label-float">

                <input type="text" id="conta" placeholder="" required>

                <label id="contaLabel" for="usuario">Conta</label>

            </div>

            <div class=" label-float">

                <input type="password" maxlength="6" id="senha" placeholder="" required>

                <label for="usuario">Senha</label>

            </div>

            <div class="justify-center">

                <button onclick="entrar()">Entrar</button>

            </div>

            <div id="msgError"></div>

            <div class="justify-center">

                <hr>

            </div>

            <p>Deseja criar uma conta?

                <a href="http://127.0.0.1:5501/frontend/tela-cadastro/index.html" target="\_blank">Cadastre-se</a>

Figura 9 - HTML da página de login.

A figura 9 demonstra a construção da página de login em html. Podemos observar a criação do card para a implementação dos campos necessários para o login do cliente já existente em nosso banco de dados. Caso seja um novo cliente, há um link direto para a página de cadastro.

A seguir veremos a implementação do CSS da página de login.

.label-float input {/\*todos os inputs que estao dentro do 'label-float' sofrera essas alteracoes abaixo\*/

*width*: 100%;

*padding*: 5px 5px; /\*tamanho para baixo e para o lado direito\*/

*display*: inline-block; /\*Para fixar em uma linha, ou seja o componente que tiver essa tag 'input' so ele ocupara tal espaco\*/

*border*: 0;/\*sem bordas nos inputs, aquelas linhas de contorno\*/

*border-bottom*: 2px solid #006699;

*background-color*: transparent;

*outline*: none;/\*retira aquele contorno de quando clicamos no input\*/

*min-width*: 180px;

*font-size*: 16px;/\*tamanho da letra\*/

*transition*: all .3s ease-out;/\*meio que mostra o input se tranformando aos de acordo com a contagem no caso 0.3s\*/

*border-radius*: 0;

}

Figura 10 – Implementação do css na página de login.

A figura 9 demonstra a utilização de folhas de personalização da página. No exemplo demonstrado observamos os critérios utilizados para os inputs e labels utilizadas no html. Seguimos o mesmo princípio para as demais personalizações necessárias.

A seguir veremos a implementação do Javascript na tela de login.

*function* validationLogin(*dado*) {

    if(*dado*.length == 0)

        alert('Usuário não encontrado!')

    else {

*dado*.forEach(*element* *=>* {

            localStorage.setItem('chave\_pix', *element*.chave\_pix);

            localStorage.setItem('cpf\_cnpj', *element*.cpf\_cnpj);

            localStorage.setItem('dt\_nascimento', *element*.dt\_nascimento);

            localStorage.setItem('email', *element*.email);

            localStorage.setItem('id\_conta\_bancaria', *element*.id\_conta\_bancaria);

            localStorage.setItem('id\_usuario', *element*.id\_usuario);

            localStorage.setItem('instituicao', *element*.instituicao);

            localStorage.setItem('nome', *element*.nome);

            localStorage.setItem('nome\_social', *element*.nome\_social);

            localStorage.setItem('nr\_agencia', *element*.nr\_agencia);

            localStorage.setItem('nr\_conta', *element*.nr\_conta);

            localStorage.setItem('senha', *element*.senha);

            localStorage.setItem('sexo', *element*.sexo);

            localStorage.setItem('vl\_poupança', *element*.vl\_poupança);

            localStorage.setItem('vl\_saldo', *element*.vl\_saldo);

            localStorage.setItem('userLogin', true)

            isLogado()

        });

    }

}

*function* isLogado() {

    location.reload;

    if(localStorage.userLogin) {

        window.location.pathname = "../frontend/tela-principal/tela-principal.html"

    }

}

isLogado()

Figura 11 – Implementação do Javascript na tela de login.

A figura 11 demonstra uma função que valida os campos, salva no local storage e integra ao banco de dados.

Finalizamos, deste modo, as exemplificações das técnicas utilizadas na tela de login.

# 4.2 implementação da tela de cadastro

Demonstraremos a seguir, através de prints e comentários, as técnicas utilizadas para a implementação da tela de cadastro do nosso sistema.

 <div class="container">

        <div class="form-image">

            <img src="assets/undraw\_nakamoto\_-2-iv6.svg" alt="">

        </div>

        <div class="form" >

            <form action="/submit-form" method="post"></form>

                <div class="form-header">

                    <div class="title">

                        <h1>Cadastre-se</h1>

                    </div>

                </div>

                <div class="input-group">

                    <div class="input-box" oninput="nameValidate()">

                        <label for="firstname">Nome</label>

                        <input class="input" type="text" name="firstname" placeholder="Digite seu nome" required>

                        <span class="span-required"> Nome deve ter, no mínimo, 3 caracteres</span>

                    </div>

                    <div class="input-box" oninput="lastNameValidate()">

                        <label for="lastname">Sobrenome</label>

                        <input class="input"  type="text" name="lastname" placeholder="Digite seu sobrenome" required>

Figura 12 – Implementação do html da tela de cadastro

A figura 12 demonstra a construção da página de cadastro em html. Podemos observar a criação de um formulário responsivo para a implementação dos campos necessários para o cadastro de um novo cliente.

A seguir veremos a implementação do CSS da página de cadastro.

@media screen and (*max-width*: 1200px) {

    .form-image {

*display*: none;

    }

    .container {

*width*: 50%;

    }

    .form {

*width*: 100%;

    }

}

@media screen and (*max-width*: 1024px) {

    .container {

*width*: 90%;

*height*: auto;

    }

    .input-group {

*flex-direction*: column;

*overflow-y*: scroll;

*flex-wrap*: nowrap;

*max-height*: 10rem;

    }

    .gender-inputs {

*margin-top*: 2rem;

    }

    .gender-group {

*flex-direction*: column;

    }

    .gender-input {

*margin-top*: 0.5rem;

    }

}

Figura 13 – implementação da personalização da tela de cadastro.

A figura 3 demonstra a utilização de folhas de personalização da página. No exemplo demonstrado observamos os critérios utilizados para a criação de um formulário responsivo, ou seja, que se adapta as configurações de tela ali definidas. Seguimos o mesmo princípio para as demais personalizações necessárias.

A seguir veremos a implementação do Javascript na tela de cadastro.

btn.addEventListener('click', (*event*) *=>* {

*event*.preventDefault();

    nameValidate();

    lastNameValidate();

    socialNameValidate();

    cpfValidate();

    emailValidate();

    phoneValidate();

    mainPasswordValidate();

    comparePassword();

    if (campos[0].checked && campos[1].checked && campos[2].checked && campos[3].checked &&

        campos[4].checked && campos[5].checked && campos[6].checked && campos[7].checked) {

*let* aleatCont = gerarNumeroAleatorio();

*let* nomeS = campos[2].value;

        alert("Olá " + nomeS + ", \nSeja bem vindo(a) ao JEEV Bank! Abaixo segue sua a agência e conta:\n" + "\nAgência: 3788\n" + "\nConta: " + aleatCont);

        window.location.pathname = "../frontend/tela-login/login.html"

        }

});

Figura 14 – Implementação do Javascript da tela de cadastro

A figura 14 demonstra a descrição de um evento ligado a um “botão”. Ao clicar no botão, é verificado se todos os campos foram preenchidos através das funções, que foram definidas abaixo no script. Após a comprovação que os campos foram preenchidos, é feita a verificação dos critérios de validação de cada campo e, se os valores retornarem como verdadeiros, os valores serão salvos e um pop-up com agência e conta aparecerá na tela e, o cliente será redirecionado para a tela de login. Caso retorne falso, será demonstrado em tela os campos que necessitam de correção.

Finalizamos, deste modo, as exemplificações das técnicas utilizadas na tela de cadastro.

# 4.3 implementação da tela principal

Demonstraremos a seguir, através de prints e comentários, as técnicas utilizadas para a implementação da tela principal do nosso sistema.

<div class="logo-header">

            <img src="../assets/img/logo.jpg" class="logo" alt="logo do banco">

        </div>

        <div class="navigation-header" id="navigation-header">

            <br>

            <a href="#" class="active">Minha conta</a>

            <a href="#" class="active">Segurança</a>

            <a href="#" class="active">Sobre</a>

            <button class="btn-icon-signout" onclick="logout()">

                <svg xmlns="http://www.w3.org/2000/svg" width="25" height="25" fill="currentColor"

Figura 15 – implementação do html da página principal

A figura 15 demonstra a construção da página principal em html. Podemos observar a criação de “divs’” para a construção de elementos com ações definidas para obedecer a critérios ao “clicar” e “logar’

A seguir veremos a implementação do CSS da página principal.

@media screen and (*max-width*: 768px) {

    .btn-icon-header{

*display*: block;

    }

    .navigation-header {

*margin-top*: 55px;

*position*: absolute;

*flex-direction*: column;

*top*: 0;

*background-color*: var(--color-background-sidebar-esquerdo);

*height*: 100%;

*width*: 30vw;

*padding*: 1em;

*animation-duration*: 1s;/\*o sidebar vai aparecer da esquerda para direita durando 1s\*/

*margin-left*: -100vw;/\*views\*/

    }

Figura 16 – Implementação do CSS na tela principal

A figura 16 demonstra a utilização de folhas de personalização da página. No exemplo demonstrado observamos os critérios utilizados para a criação de uma página responsiva, ou seja, que se adapta as configurações de tela ali definidas. Seguimos o mesmo princípio para as demais personalizações necessárias.

*const* createRow = (*extrato*) *=>* {

*const* tr = createElement('tr');

    if (!*extrato*) {

*const* vazio = createElement('td', '')

*const* vazio2 = createElement('td', 'Extratos Vazio');

*const* vazio3 = createElement('td', '');

        tr.appendChild(vazio);

        tr.appendChild(vazio2).style.textAlign = 'center';

        tr.appendChild(vazio3);

        return tr

    }

*let* { id\_pagamento, id\_transacao, tipo\_pagamento, recebimento,

        valor, descricao, dt\_pagamento } = *extrato*;

    recebimento = recebimento.data[0];

    dt\_pagamento = formatData(dt\_pagamento)

    valor = formatValor(valor, recebimento);

*const* data = createElement('td', dt\_pagamento);

*const* descricaoPagamento = createElement('td', descricao);

*const* valorPagamento = createElement('td', valor);

    if (recebimento) {

        tr.appendChild(data).style.background = '#00cc0090';

        tr.appendChild(descricaoPagamento).style.background = '#00cc0090';

        tr.appendChild(valorPagamento).style.background = '#00cc0090';

    } else {

        tr.appendChild(data).style.background = '#ff330090';

        tr.appendChild(descricaoPagamento).style.background = '#ff330090';

        tr.appendChild(valorPagamento).style.background = '#ff330090';

    }

    return tr;

}

Figura 17 – Implementação do Javascript na tela principal

A figura 17 demonstra a execução de uma função para as operações relacionadas ao extrato bancário de uma conta. Sendo considerado se há ou não uma solicitação do extrato e, caso haja, todos os critérios relacionados dentre eles: tipo, descrição, data, valores, serão considerados.

Finalizamos, deste modo, as exemplificações das técnicas utilizadas na tela de cadastro.

# 5 Implementação do backend

Os prints a seguir demonstram a implementação do nosso backend.

async *function* executeQuery(*sql*) {

*const* conn = await connect();

*const* [rows] = await conn.query(*sql*);

    return rows;

    conn.query(*sql*,(*error*,*results*,*fields*)*=>*{

        if(*error*)

            return json(*error*);

        else

            return json(*results*);

        conn.end();

    });

}

Figura 18– Query no banco de dados

A figura 18 demonstra a execução de uma função que executa querys em nosso banco de dados.

*const* findUserByLogin = async(*body*) *=>* {

    return await db.executeQuery(

    `SELECT \* FROM tb\_conta\_bancaria as conta

        JOIN tb\_usuario usuario ON conta.id\_usuario = usuario.id\_usuario

        WHERE conta.nr\_agencia =${*body*.agencia} AND conta.nr\_conta=${*body*.conta} AND conta.senha= ${*body*.senha}`);

}

*const* findExtratoByIdContaBancaria = async(*id*) *=>* {

    return await db.executeQuery(`SELECT \* FROM tb\_pagamento WHERE id\_conta\_bancaria= ${*id*}`);

}

Figura 19 – Integração com banco de dados

A figura 19 demonstra a execução de uma função que adiciona o usuário as tabelas equivalentes em nosso banco de dados.

router.post('/find-conta-bancaria', userController.findUserByLogin);

router.post('/cadastrar', userController.createUser);

router.delete('/delete/:id', userController.deleteUser);

router.put('/alterar/:id', userValidation.validateBody, userController.updateUser);

router.get('/find-extrato/:idContaBancaria', userController.findExtratoByIdContaBancaria);

module.exports = router;

Figura 20 – Criação de rotas

A figura 20 demonstra a criação de rotas no servidor para cada operação necessária para o funcionamento do software.

*const* app = require('./app')

require('dotenv').config();

//servidor

*const* PORT = process.env.PORT || 8081;

app.listen(PORT,()*=>*{

    console.log(`Servidor iniciado em http://localhost:${PORT}`);

})

Figura 21 - Criação do servidor

A figura 21 demonstra as técnicas utilizadas para a criação de um servidor.

Finalizamos, deste modo, as exemplificações das técnicas utilizadas no backend.

# 5 Consideração finais

O processo de formação de um desenvolvedor é tão árduo quanto recompensador, poucas sensações, de fato, são melhores que elaborar todo um projeto a partir de uma ideia e após algum tempo, muita massa cinzenta gasta e momentos de frustração, somos premiados pela execução perfeita daquela ideia em um sistema funcional da forma que projetamos na nossa mente, ou até melhor.

É fato que em nossa trajetória sempre lidaremos com dificuldades, mas é óbvio que no início elas serão demasiadamente pertinentes, desta forma comentaremos a seguir algumas dessas dificuldades.

Um dos principais obstáculos, não só relacionado a carreira de desenvolvedor, mas toda área que necessite de uma ou mais pessoas interagindo para chegar a um único objetivo, é o trabalho em equipe. Isto porque, obviamente, as pessoas possuem diferentes visões, bagagem de conhecimento, modus operandi e assim por diante.

Concatenar essas diferenças entre os integrantes é tão importante quanto dominar as técnicas necessárias para a composição do projeto, visto que, por mais profissional que seja uma equipe se não há uma linha de horizonte bem definida, não se faz possível chegar a lugar algum. É fato que, por se tratar de uma equipe flexível e comprometida com o projeto, adquirimos naturalmente a fluidez necessária para a implementação a sua implementação.

Para a formação de um projeto faz-se imprescindível a compreensão de como modelar e estruturar os seus componentes. Ou seja, uma boa compreensão da engenharia de software resulta na implementação das melhores tecnologias, técnicas de gerência e tecnologias para aquele projeto específico. Devido a falta de experiência com o tema e o mercado propriamente dito, alcançar a excelência necessária para o domínio das técnicas foi um processo hercúleo, entretanto, os resultados alcançados demonstram que a equipe superou os obstáculos impostos até aqui.

Outro ponto de dificuldade que vale ressaltar foi a integração com o banco de dados, bem como, a implementação do node.js e criação de rotas no backend demandaram grande energia para melhor compreensão e implementação junto ao projeto.

Nos demais aspectos do projeto não encontramos dificuldade e a fluidez necessária ocorreu naturalmente.

É imprescindível para um desenvolvedor que busca uma carreira de sucesso dominar as principais técnicas de programação existentes e além de desenvolver uma análise eficiente das demandas do mercado e um bom trabalho em equipe.

Acredito que esse pensamento foi o ponto inicial para a elaboração do projeto por nossos instrutores, e posso afirmar com propriedade que essas características foram bem desenvolvidas no decorrer do processo, encaminhando a todos para um bom desenvolvimento profissional e, consequentemente, uma carreira de sucesso.

‘

# 6 BIBLIOGRAFIA

Disponível em:

<https://www.freecodecamp.org/portuguese/news/como-comecar-com-o-node-js-guia-para-iniciantes-em-node/>

Acessado em: 10/05/2023