INSTITUTO FEDERAL GOIANO - CAMPUS MORRINHOS

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

Ludmylla Alves Fagundes

RELATÓRIO TÉCNICO: Desenvolvimento de Sistema Web de Cobranças com Vue.js e PHP

MORRINHOS - GO

2023

Ludmylla Alves Fagundes

RELATÓRIO TÉCNICO: Desenvolvimento de Sistema Web de Cobranças com Vue.js e PHP

Relatório Técnico apresentado ao Curso Superior de Tecnologia de Sistemas para Internet do Instituto Federal Goiano - Campus Morrinhos como requisito parcial para obtenção de título de Tecnólogo em Sistemas para Internet.

Área de concentração: **Desenvolvimento de Sistemas.**

Orientador: **Rodrigo Elias Francisco**

MORRINHOS - GO

2023

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas – SIBI/IF Goiano Campus Morrinhos

F151r Fagundes, Ludmylla Alves.
Relatório técnico: desenvolvimento de Sistema Web de Cobranças com Vue.js e PHP. / Ludmylla Alves Fagundes. – Morrinhos, GO: IF Goiano, 2023.
56 f. : il. color.

Orientador: Me. Rodrigo Elias Francisco.

Trabalho de conclusão de curso (graduação) – Instituto Federal Goiano Campus Morrinhos, Tecnologia em Sistemas para a Internet, 2023.

1. Sistemas de cobrança de documentos financeiros. 2. PHP (Linguagem de programação de computador). 3. Software - Compatibilidade. I. Francisco, Rodrigo Elias. II. Instituto Federal Goiano. III. Título.

CDU 004.4'2:004.43*PHP

Fonte: Elaborado pela Bibliotecária-documentalista Morgana Guimarães, CRB1/2837

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO

PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610, de 19 de fevereiro de 1998, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano a disponibilizar gratuitamente o documento em formato digital no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

IDENTIFICAÇÃO DA PRODUÇÃO TÉCNICO-CIENTÍFICA

- ☐ Tese (doutorado)
- ☐ Dissertação (mestrado)
- ☐ Monografia (especialização)
- ☑ TCC (graduação)
- ☐ Artigo científico
- ☐ Capítulo de livro
- ☐ Livro
- ☐ Trabalho apresentado em evento
- ☐ Produto técnico e educacional - Tipo:

Nome completo do autor:
Ludmylla Alves Fagundes

Matrícula:
2019104211710243

Título do trabalho:
RELATÓRIO TÉCNICO: Desenvolvimento de Sistema Web de Cobranças com Vue.js e PHP

RESTRIÇÕES DE ACESSO AO DOCUMENTO

Documento confidencial: ☑ Não ☐ Sim, justifique:

Informe a data que poderá ser disponibilizado no RIIF Goiano: 29 / 02 / 2024

O documento está sujeito a registro de patente? ☐ Sim ☑ Não

O documento pode vir a ser publicado como livro? ☐ Sim ☑ Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara:

• Que o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;

• Que obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autoria, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

• Que cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Morrinhos (Local) 12 / 03 / 2024 (Data)

gov.br Documento assinado digitalmente
LUDMYLLA ALVES FAGUNDES
Data: 27/02/2024 20:11:14-0300
Verifique em https://validar.iti.gov.br

Assinat ; autorais

Ciente e de acordo:

gov.br Documento assinado digitalmente
RODRIGO ELIAS FRANCISCO
Data: 12/03/2024 20:38:17-0300
Verifique em https://validar.iti.gov.br

Assinatura do(a) orientador(a)

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

Ata nº 33/2023 - CCEPTNM-MO/CEPTNM-MO/DE-MO/CMPMHOS/IFGOIANO

ATA DE DEFESA DE TRABALHO DE CURSO

Ao(s) 14 dia(s) do mês de dezembro de 2023, às 16:00 horas, reuniu-se a banca examinadora composta pelos docentes: Rodrigo Elias Francisco (orientador), Felipe Nunes Gaia (membro), Norton Coelho Guimarães (membro), para examinar o Trabalho de Curso intitulado “RELATÓRIO TÉCNICO: Desenvolvimento de Sistema Web de Cobranças com Vue.js e PHP” da estudante Ludmylla Alves Fagundes, Matrícula nº 2019104211710243 do Curso de Tecnologia em Sistemas para Internet do IF Goiano – Campus Morrinhos. A palavra foi concedida a estudante para a apresentação oral do TC, houve arguição do(a) candidato pelos membros da banca examinadora. Após tal etapa, a banca examinadora decidiu pela **APROVAÇÃO** da estudante. Ao final da sessão pública de defesa foi lavrada a presente ata que segue assinada pelos membros da Banca Examinadora.

(Assinado Eletronicamente)

Rodrigo Elias Francisco

Orientador

(Assinado Eletronicamente)

Felipe Nunes Gaia

Membro

(Assinado Eletronicamente)

Norton Coelho Guimarães

Membro

Observação:

( ) O(a) estudante não compareceu à defesa do TC.

Documento assinado eletronicamente por:

- Rodrigo Elias Francisco, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 18/12/2023 14:09:49.
- Norton Coelho Guimaraes, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 18/12/2023 13:08:39.
- Felipe Nunes Gaia, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 18/12/2023 13:05:27.

Este documento foi emitido pelo SUAP em 15/12/2023. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 558441
Código de Autenticação: f7f24a9423

INSTITUTO FEDERAL GOIANO
Campus Morrinhos
Rodovia BR-153, Km 633, Zona Rural, SN, Zona Rural, MORRINHOS / GO, CEP 75650-000
(64) 3413-7900

RESUMO

Este trabalho consiste no desenvolvimento de um sistema projetado para facilitar a realização de cobranças em estabelecimentos que adotam o método de vendas a prazo. O sistema foi construído utilizando tecnologias de componentes web, com uma abordagem de front-end baseada no framework Vue.js. Além disso, faz uso da dependência Axios para realizar requisições Hypertext Transfer Protocol (HTML). Para armazenar e gerenciar os registros de forma eficiente, o sistema utiliza um banco de dados MySQL. As funcionalidades do sistema são inovadoras por meio da linguagem PHP:Hypertext Preprocessor (PHP).

Palavras-chaves: PHP, Vue.js, sistema, cobrança

ABSTRACT

This work consists of developing a system designed to facilitate billing in establishments that adopt the installment sales method. The system was built using web component technologies, with a front-end approach based on the Vue.js framework. Furthermore, it makes use of the Axios dependency to make Hypertext Transfer Protocol (HTML) requests. To store and manage records efficiently, the system uses a MySQL database. The system's functionalities are innovative using the PHP: Hypertext Preprocessor (PHP) language.

Keywords: PHP, Vue.js, system, billing

LISTA DE TABELAS

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIAÇÕES

Cascading Style Sheets (CSS)

HyperText Markup Language (HTML)

PHP: Hypertext Preprocessor (PHP)

JavaScript Object Notation (JSON)

Application Programming Interface (API)

Node Package Manager (NPM)

SUMÁRIO

1 INTRODUÇÃO

Conforme Fernandes (2003) software é um produto do trabalho humano em computadores, celulares ou outros dispositivos. Um software é um conjunto de instruções a serem seguidas. As três principais categorias são: software de programação, que são as ferramentas utilizadas pelo programador para desenvolver novos softwares e programas; software de sistemas, encarregado da comunicação entre o computador e o usuário; software de aplicação, que tem como função executar tarefas a calculadora, players de vídeo e música, jogos, entre outros.

O desenvolvimento web é responsável por codificar sites, páginas, portais e aplicativos para a web. Nas características do desenvolvimento visual: HyperText Markup Language (HTML): Responsável pela estrutura da página e interação com o usuário; Cascading Style Sheets (CSS): Define a aparência da página, incluindo núcleos, desenhos e organização; JavaScript: Possibilidade de interação do usuário com a página, implementando funcionalidades interativas.

No desenvolvimento web, a funcionalidade, as requisições e a interatividade com o servidor ou a máquina são frequentemente inovadoras com linguagens de programação, como C#, Python, PHP:Hypertext Preprocessor (PHP), JavaScript, Ruby, entre outras.

Os dados gerados por esses sistemas são armazenados em bases de dados, que podem ser relacionais ou não relacionais. Exemplos de bases de dados relacionadas incluem MariaDB, MySQL, PostgreSQL, Microsoft SQL Server, Oracle, entre outros, enquanto bases de dados não relacionadas incluem Hbase, Amazon DynamoDB, MongoDB, entre outros. Esses sistemas de gerenciamento de bancos de dados são essenciais para armazenar e recuperar informações de maneira eficaz.

1.1 HISTÓRICO

Conforme Cromwell (2016), o Vue.js é um framework JavaScript progressivo e de código aberto, destinado à criação de interfaces de usuário. Foi idealizado por Evan You em 2014, quando ainda estava envolvido com o Angular.js. A motivação

era extrair os elementos que ele apreciava no framework e construir algo mais leve, sem conceitos excessivos. Comparado ao React, o Vue.js direciona um pouco mais de atenção à acessibilidade.

De acordo com Cromwell (2016), em 2015 o Vue.js era inicialmente um projeto de código aberto sem um foco específico, porém a comunidade Laravel começou a colaborar com o Vue.js.

Como indicado por Cromwell (2016), o Vue.js passou por dois processos de reescrita completa. Isso ocorreu porque a correção original apresentou problemas que não puderam ser corrigidos gradualmente por meio de refatorações simples.

Dall'Oglio (2015) menciona que a linguagem de programação PHP originalmente era "Personal Home Page Tool" e foi criada no outono de 1994 por Rasmus Lerdorf. Durante um breve período, o PHP foi denominado FI (Forms Interpreter). Essa linguagem era composta por um conjunto de scripts escritos em linguagem C, destinados à criação de páginas dinâmicas. Com o tempo, Rasmus adicionou várias funcionalidades, incluindo a capacidade de interação com bancos de dados. Em 1995, o código fonte do PHP foi disponibilizado, permitindo que mais desenvolvedores se envolvessem no projeto.

1.2 PROBLEMA

Na concepção inicial, uma farmácia vendia remédios e registrava cada venda em um banco de dados. Propõe-se um sistema online integrado ao sistema de vendas para garantir precisão e confiabilidade no cálculo das dívidas do cliente, evitando problemas de pagamento incorreto.

Uma empresa frequentemente realiza vendas em um sistema e registra essas transações em um banco de dados. Essas vendas podem permanecer registradas por meses ou até anos. Geralmente, essas vendas só são verificadas quando um cliente decide liquidar suas dívidas ou fazer novas compras. No entanto, é possível que o vendedor não perceba a existência de uma venda atrasada.

No caso de uma empresa empregar cobradores ambulantes, esses cobradores terão a tarefa de registrar manualmente as cobranças em papel. No entanto, essa abordagem poderia resultar em erros, fazendo com que o cliente

pagasse um valor incorreto. Para evitar tais problemas, os cobradores precisam de um sistema online que lhes permita controlar as cobranças. Esse sistema deveria ser integrado ao sistema de vendas, garantindo que o cálculo do valor exato da dívida do cliente fosse preciso e confiável.

1.3 OBJETIVOS GERAIS

O sistema foi desenvolvido para proporcionar organização e praticidade no planejamento e na realização de cobranças. Com o sistema, os cobradores terão capacidade de sair para as ruas e conduzir as cobranças de forma online e direta, diretamente pelo sistema. Isso permite que as operações de cobrança sejam mais eficientes e atualizadas em tempo real, otimizando a forma como as cobranças são gerenciadas.

1.4 OBJETIVOS ESPECÍFICOS

- Desenvolver o front-end do sistema com o framework progressivo Vue.js.
- Armazenar os dados em um banco relacional MySql.
- Desenvolver o back-end do sistema com PHP.
- Testar o sistema com dados fictícios.

1.5 JUSTIFICATIVA

Com base no problema apresentado, o desenvolvimento deste sistema se mostra altamente relevante, oferecendo maior praticidade aos administradores na gestão de suas empresas e redução de constrangimentos no processo de cobrança com os clientes.

Ao possibilitar que os cobradores realizem cobranças em campo e registrem-nas online para a supervisão dos administradores, o sistema permite que as empresas organizem suas finanças de maneira mais eficiente e mantenham um controle mais preciso das transações financeiras.

1.6 METODOLOGIA

Etapas para o desenvolvimento

1. Planejamento do desenvolvimento do sistema considerando linguagens, frameworks e bibliotecas.
2. Criação de telas e suas funcionalidades no front-end.
3. Criação de métodos e funções no back-end.
4. Adição de dados fictícios no banco.
5. Testes de verificação de funcionamento.

2 FUNDAMENTAÇÃO

Neste capítulo, são abordadas as arquiteturas e tecnologias utilizadas no sistema.

2.1 DESENVOLVIMENTO WEB

Segundo Costa(2007) a história do desenvolvimento web começou em 1969 quando o departamento de defesa dos EUA criou uma rede experimental designada ARPA. Três anos depois foi criada a ARPNET que já permitia o correio electrónico e comunicações em tempo real.

2.2 SISTEMA DA INFORMAÇÃO

De acordo com Mattos (2017) um sistema é constituído de dois elementos: uma coleção de objetos, por uma lado, e uma relação lógica entre eles, por outro. Esses elementos físicos e lógicos fazem com que o sistema se comporte como um organismo. Um sistema de informações não necessita de computadores como ocorria antes de 1945.

As empresas criam sistemas de informações para resolver problemas organizacionais e para reagir às mudanças que ocorrem no ambiente. Wakulicz(2016) explica que estes sistemas são criados para reagir aos concorrentes, clientes, fornecedores e as mudanças sociais e tecnológicas. Sistemas empresariais de informações dão suporte a diversos níveis e setores como: fabricação e produção; vendas e marketing; finanças e contabilidade .

2.3 INTEGRAÇÃO DE SISTEMAS WEB

Athos, Diego e Rodrigo (2016) Web service é um meio utilizado para integração de sistemas e de comunicação entre aplicações heterogêneas.Em outros termos, essa tecnologia permite que novas aplicações possam interagir com aquelas existentes em um ambiente de trabalho e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Para Kalin (2010) um web service é, então, uma aplicação distribuída, cujos componentes podem ser aplicados e executados em dispositivos distintos.

2.4 SOFTWARES DE COBRANÇAS

Como afirma Kosh (2009) a falta de controle da inadimplência possibilita o fim de instituições pelo fato de as disponibilidades serem fator imprescindível para sua operacionalidade, isso evidencia a importância de um processo de cobrança eficaz.

Kosh (2009) também descreve que para um processo é necessário analisar as variáveis e a partir disso traçar um objetivo. Este conceito pode ser utilizado em processos administrativos. A preocupação com prazos e atendimentos, os colegas de trabalho cada parte influencia no resultado e no sucesso da instituição.

2.5 FRAMEWORKS

Wohlgethan (2018) em comparação com bibliotecas, os frameworks realmente oferecem uma pilha completa de funções úteis para assumir a responsabilidade por decisões que, de outra forma, o desenvolvedor teria que fazer antes de realmente escrever o código do aplicativo. Isso inclui estratégias para

roteamento de caminhos de URL no aplicativo, gerenciamento de estado, empacotamento e outros.

De acordo com Wohlgethan (2018) a lógica por trás do conceito de componentes é importante porque somente eles descrevem a interface do usuário. Quando ocorrem mudanças no dados subjacentes, a estrutura renderiza novamente o componente de UI completo.

2.6 LINGUAGEM DE PROGRAMAÇÃO PHP

Tavares e Cremasco (2010) o PHP como é conhecido hoje, é na verdade o sucessor para um produto chamado PHP/FI. Criado em 1994 por Rasmus Lerdof, a primeira encarnação do PHP foi um simples conjunto de binários Common Gateway Interface (CGI) escrito em linguagem de programação C.

Tavares e Cremasco (2010) ao longo do tempo, mais funcionalidades foram desejadas, e Rasmus reescreveu o PHP Tools, produzindo uma maior e rica implementação. Este novo modelo foi capaz de interações com Banco de Dados e mais, fornecendo uma estrutura no qual os usuários poderiam desenvolver simples e dinâmicas aplicações web, como um livros de visitas.

.De acordo com Tavares e Cremasco(2010) a linguagem foi desenvolvida para, deliberadamente, ser parecida com C, tornando-a fácil para ser adotada por desenvolvedores habituados com C, Perl e linguagens similares. Em junho de 1998, com muitos novos desenvolvedores ao redor do mundo unindo esforços, PHP 3.0 foi anunciado pelo novo time de desenvolvimento do PHP. Essa foi a primeira versão que se assemelha com o PHP como conhecemos hoje.

De acordo Php.net (2023) o suporte de sessões consiste em uma maneira de preservar dados através de acessos subsequentes. Para a manipulação de sessões é possível realizar a chamada de funções como: session_start inicia a função; session_abort que descarta as alterações feitas no array e encerra a sessão; session_id obtém e/ou define o id de sessão atual; entre outras.

2.7 FRAMEWORK VUE.JS

De acordo com Didier e Robles (2021), o Vue.js é um conjunto de regras, procedimentos e critérios usados para construir uma visualização de aplicações web. O Vue.js foi criado com o conceito de ser uma biblioteca muito completa e foi lançado em 2014. Os componentes do Vue.js são objetos que podem ser reutilizados e invocados através de botões, sendo organizados de acordo com sua situação.

Conforme Incau (2017), os componentes são fragmentos pequenos que compõem uma página. Isso facilita e fornece reutilização de código em páginas separadas.

Incau (2017) também destaca as convenções, que são identificadas pelo prefixo v- e são atributos especiais fornecidos pelo Vue.js. De acordo com Vuejs.org, espera-se que os valores de atributos de cláusulas sejam expressões JavaScript exclusivas, exceto para v-for, v-slot e v-on. A função das disposições é aplicar atualizações de forma reativa ao DOM quando o valor de sua expressão muda.

Segundo Capucço (2021), Props são essencialmente atributos personalizados que transmitem dados para um componente. De acordo com Vuejs.org (2023), os componentes Vue exigem uma declaração explícita de props para que o Vue saiba quais props externos passados para o componente devem ser tratados como atributos "fallthrough".

Como afirma a Vuejs.org(2023) cada instância de um componentes passa por uma série de etapas de inicialização quando é criada, por exemplo: precisa configurar a observação de dados; compilar o modelo; montar a instância no DOM e atualizar o DOM quando os dados forem alterados. Deste modo ele executa funções como os ganchos de ciclos de vida dando aos usuários oportunidades de adicionar o próprio código em estágios específicos.

Em uma pesquisa realizada por Clack (2023), foram identificadas as 10 melhores startups que utilizam o framework Vue.js. São elas:

1. Bitpanda: plataforma de investimento austríaca.

2. Trivago: empresa alemã conhecida pelo mecanismo de metabusca de viagens que compara preços das acomodações oferecidas por vários sites de reservas.
3. Plaid: empresa de serviços financeiros com sede na Califórnia, fornecendo ferramentas e uma rede de transferência de dados para empresas criarem soluções fintech e produtos financeiros.
4. GitLab: software de DevOps de código aberto e uma ferramenta de gerenciamento de hospedagem de repositório.
5. Quero Education: marketplace brasileiro online que conecta estudantes com as instituições de ensino.

Essas são apenas algumas das startups mencionadas na pesquisa (Clack, 2023).

3 PROJETO DE SISTEMA WEB PARA COBRANÇA

No desenvolvimento do backend, utilizamos a linguagem de programação PHP em conjunto com o MySQL, que é um sistema de gerenciamento de banco de dados relacional. O acesso e administração do MySQL geralmente são feitos por meio de ferramentas como o PHPMyAdmin.

No front-end, a linguagem de marcação HTML desempenha um papel fundamental como base para a construção do sistema, e a principal estrutura utilizada é o framework Vue.js. Para a comunicação entre o front-end e o backend, o sistema faz uso da biblioteca Axios, que é responsável por criar e gerenciar as requisições HTTP direcionadas ao PHP.

Este tutorial apresenta a sequência de etapas para a construção do sistema de cobrança onde cada tópico é um passo a ser seguido.

3.1 ESPECIFICAÇÕES DO SISTEMA

No sistema construído todas as funções principais são ilustradas na Figura 1 iniciando com o login. Algumas funções são executadas de acordo com suas dependências. Os funcionários cobradores têm o acesso para realizar cobranças avulsas, planejadas e ver a lista de cobranças executadas apenas pelo cobrador ativo. Os funcionários administradores executam as funções de planejar cobranças, realizar as importações de bases de dados de clientes e compras e visualizar a lista com todas as cobranças efetuadas por todos os funcionários cobradores.

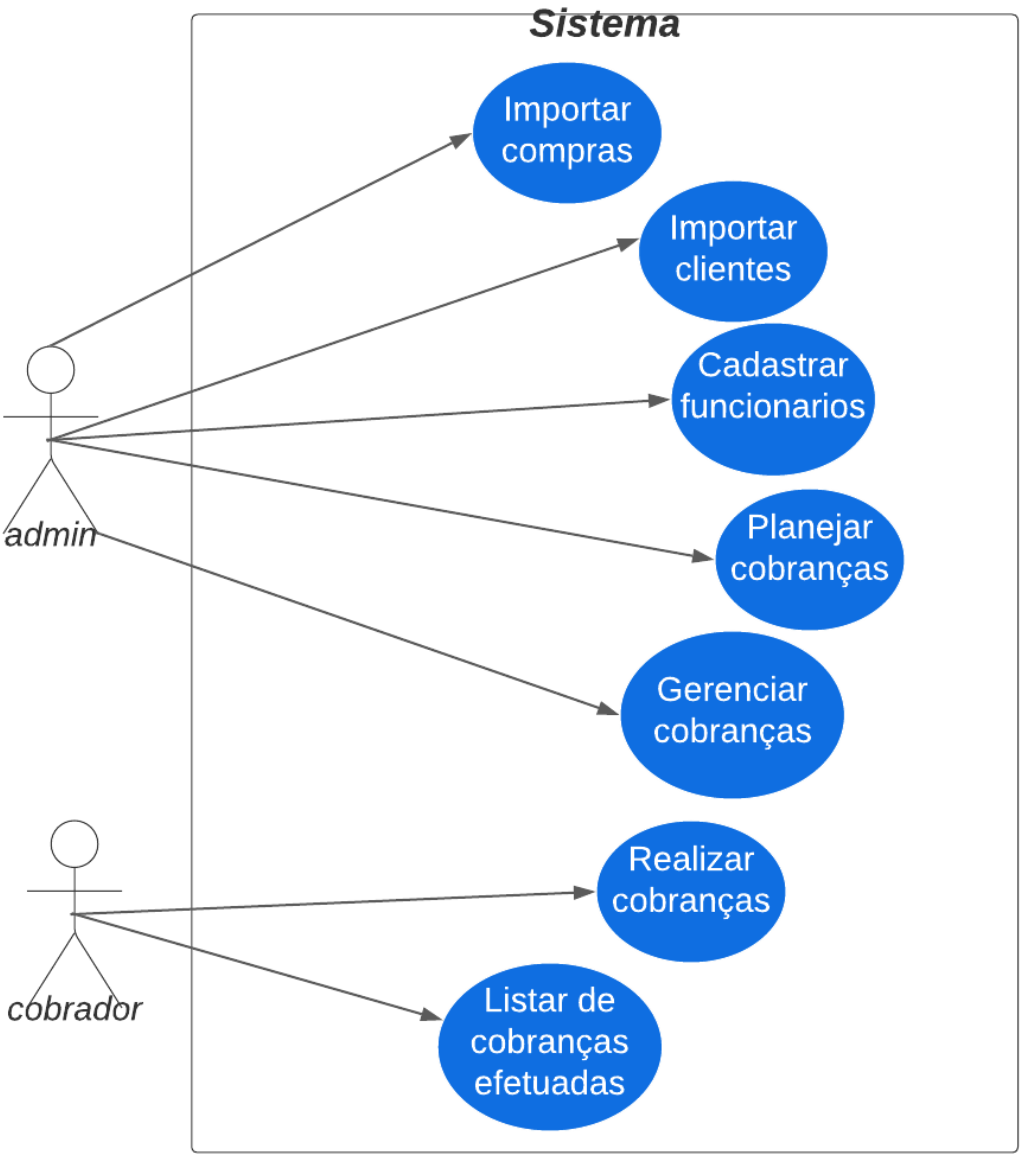


Figura 1 - Diagrama de caso de uso

3.1.1 Requisitos Funcionais:

O sistema deve ser capaz de:

1. Permitir o login por meio do uso de um nome de usuário e senha.
2. Apresentar uma lista completa de cobranças pendentes.
3. Mostra uma lista detalhada das compras registradas.

4. Efetuar o cadastro de novos funcionários no sistema.
5. Realizar o planejamento de cobranças, permitindo o agendamento futuro.
6. Efetuar cobranças conforme o planejamento estabelecido.
7. Realizar cobrança referente às compras efetuadas.
8. Facilitar a comunicação entre administradores e cobradores.
9. Permitir a remoção de cobranças que foram planejadas, mas não são mais necessárias.
10. Vue.js 3.0
11. PHP 8.2.4
12. Vusual Studio code 1.85
13. XAMPP 3.3.0
14. Axios 1.2.0

3.1.2 Requisitos não funcionais:

O sistema deve atender aos seguintes requisitos não funcionais:

1. Desempenho Eficiente: O sistema deve processar as operações de forma rápida e eficiente, garantindo tempos de resposta mínimos.
2. Armazenamento de dados: Os dados serão armazenados em um banco de dados relacional MySQL, garantindo confiabilidade e consistência.
3. Respostas Rápidas: O sistema deve fornecer respostas rápidas aos usuários, garantindo uma experiência ágil e sem atrasos.
4. Atualizações Regulares: O sistema será atualizado automaticamente ao final de cada processo, garantindo que as informações sejam sempre precisas e atualizadas.
5. Facilidade de Manutenção: O sistema será projetado para permitir uma manutenção fácil e eficaz, facilitando a correção de problemas e a adição de novas funcionalidades.

3.2 CRIAÇÃO DO BANCO DE DADOS

Para iniciar sobre o armazenamento de dados ilustramos um diagrama de entidades de relacionamento (DER) como é mostrado na figura 2.

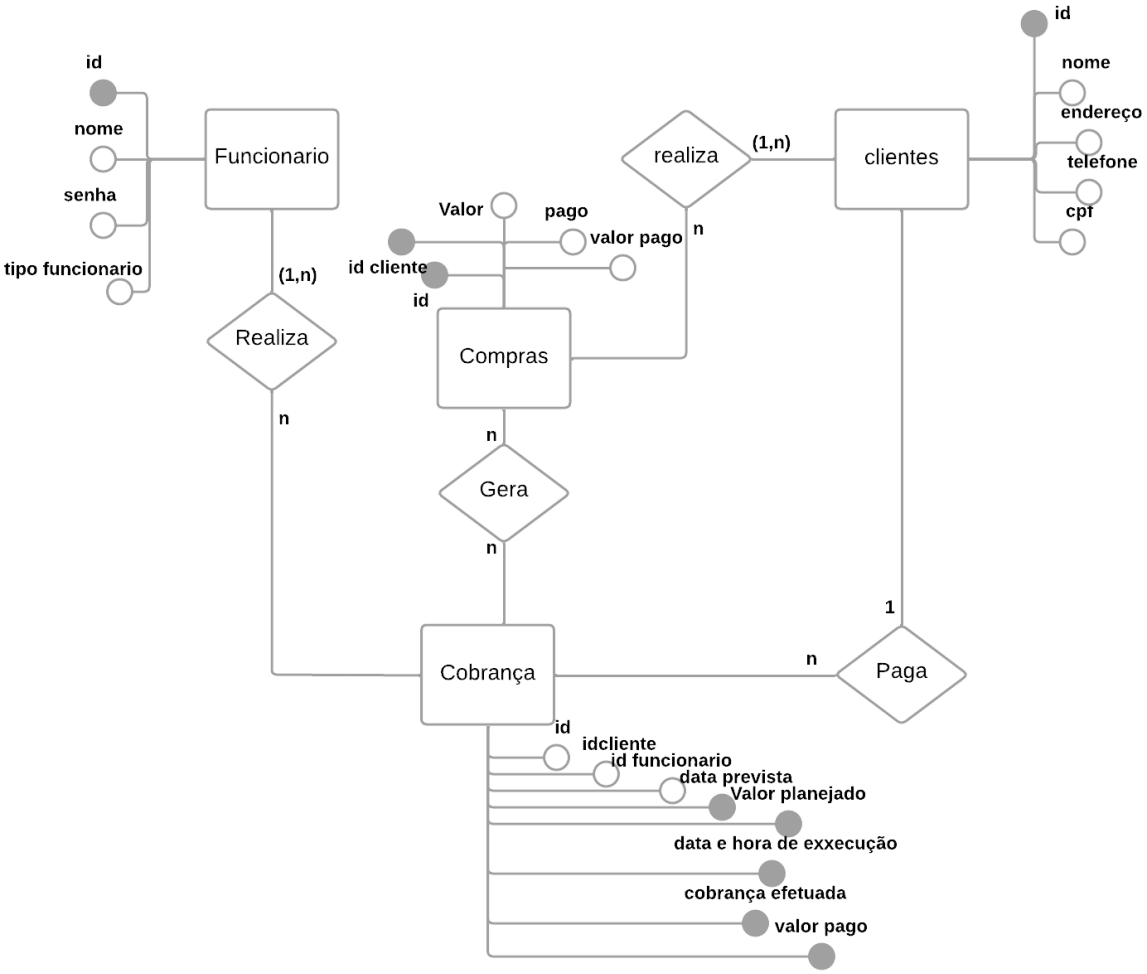


Figura 2: Diagrama de entidades de relacionamentos

O banco de dados desenvolvido é um banco relacional onde as tabelas são construídas conforme a necessidade. Com o uso do PHPMyAdmin foi criado uma base de dados com 6 tabelas, sendo elas uma para os clientes, compras, funcionários, tipo de funcionários, funcionário tipo de funcionário que uma tabela intermediária para juntar os dados dos funcionários e sejam especificados quais são seus cargos e a tabela cobrança.

Os dados utilizados para a realização de cobranças são enviados a partir de um upload feito pelo administrador. Primeiro ele envia a base de cliente e em seguida a base de compras assim as tabelas do banco se autocompletam, após os uploads o sistema está pronto para o uso.

3.3 ESCOLHA DAS FERRAMENTAS

Para iniciar o desenvolvimento, foram realizados estudos visando a escolha das ferramentas adaptadas, combinando o que era necessário com o que seria útil.

3.3.1 Uso do framework Vuejs:

A escolha do framework Vue.js foi feita devido à sua simplicidade de uso. A cada página desenvolvida, é possível adicionar a estilização com CSS e as funções JavaScript em um único arquivo. Isso proporciona agilidade para o desenvolvedor e interatividade para o usuário. Além disso, o Vue.js pode ser facilmente implementado em projetos já existentes, uma vez que sua base é em HTML, o que o torna compatível com diversos navegadores da web.

No Vue.js, existem ciclos de vida. Neste projeto, o método "montado" é o que mais se destaca nas páginas. Além disso, existem outros exemplos a serem citados, como "created", "beforeCreate", "beforeMount", entre outros. Além dos ciclos de vida, algumas diretivas como "v-if", "v-else", "v-model", "v-show", entre outras.

3.3.2 Dependencia Axios

O Axios é uma biblioteca que possibilita a realização de requisições HTTP garantidas em promisses (promessas). É ele que permite a busca de dados

provenientes da API e os traz para o sistema, tornando-os utilizáveis. Para instalá-lo através do Node Package Manager (NPM), utilize-se o comando "npm install axios", conforme exemplificado na Figura 3.

A requisição do tipo GET é a mais frequente, uma vez que os dados são solicitados em quase todas as páginas para a execução de funções.

Figura 3 - Instalação do axios.

3.3.3 Uso do PHP

O desenvolvimento backend foi realizado utilizando a linguagem de programação PHP. Essa linguagem é amplamente reconhecida por sua popularidade, atribuída à sua natureza de código aberto e às suas funcionalidades versáteis direcionadas ao desenvolvimento web. Isso promove a facilitação da conexão entre os servidores e a interface do usuário, permitindo uma interação eficiente entre esses componentes.

3.3.4 Importação de dados com JSON

Para a importação dos dados já existentes, é essencial utilizar arquivos no formato .json. Para a importação de clientes, os arquivos devem conter as seguintes informações: Id, Nome, Endereço, Telefone e CPF como mostra a Tabela 1. Recomenda-se realizar a importação de clientes antes da importação de compras, pois esta última depende dos seguintes dados: Id, id_cliente (que deve constar na

importação de dados do cliente), Valor, Valor pago e se a compra consta como Paga como mostra a Tabela 2.

Atributo	**Tipo de dados**	**Descrição**
id_cliente	int	Chave de Identificação
nome	String	Nome pessoal
endereco	String	Endereço pessoal
telefone	String	Telefone pessoal exemplo: “64981818181”
cpf	String	Cpf pessoal exemplo: “123.123.123-12”

Tabela 1: Tabela de descrição de dados dos clientes.

Atributo	**Tipo de dados**	**Descrição**
id_compras	int	Chave de identificação
id_clientes	int	Chave estrangeira
valor	float	Valor da compra. exemplo: 49.99
valor_pago	float	Valor pago pela compra. Exemplo: 39.90
pago	boolean	Compra paga? Sim/Não

Tabela 2: Tabela de descrição de dados das compras.

3.4 DESENVOLVIMENTO DO SISTEMA

No desenvolvimento deste sistema vários passos foram seguidos para que ele ficasse de uma forma simples e de fácil utilização, a cada passo um novo teste foi realizado até que se concluísse cada função corretamente. No seguinte link tem uma breve apresentação de todo o sistema em funcionamento: “https://youtu.be/lgJN2J-EYEo ”.

3.4.1 Login do funcionário

A tela inicial consiste em uma página de login, na qual é possível acessar o sistema. Essa tela apresenta um formulário que inclui duas etiquetas (labels), dois campos de entrada (inputs) e um botão. Um dos campos de entrada destina-se ao nome do usuário, enquanto o outro é destinado à senha. Após a inserção das informações nos campos, há um botão que, quando clicado, envia o formulário para validação. A Figura 4 ilustra o layout previsto para essa tela.

Figura 4: Tela de login

3.4.2 Validando o login

Após receber os dados vindos do front-end, o PHP executa uma função que consiste em verificar, por meio de uma consulta query na tabela "funcionário", se o nome de usuário e a senha fornecida correspondem a informações válidas. Caso sejam autênticos, o PHP retorna o ID desse funcionário dentro do arquivo PHP, para que ele possa ser utilizado em etapas subsequentes do processo.

```php
public function validarLogin($nome, $senha){
    //usado em: cadastrarFuncionario, Login
    $query =
    " SELECT * ".
    "   FROM funcionario ".
    "  WHERE nome= '".$nome."'".
    "    AND senha= '".$senha."'";

    $dao = new DAO();
    $resultado = $dao->selecionar1LinhaSQL($query);

    $id =0;
    try {
        if (isset($resultado["id_funcionario"]) ){
            $id = $resultado["id_funcionario"];
        }
    } catch(Exception $e) {
        $id =0;
    }

    $retorno = -1;
    if ($id>=1){
        $retorno = $id;
    }
    return $retorno;
}
```

Figura 5: Função para verificação de Login

3.4.3 Menu inicial

As Figuras 6 e 7 fornecem uma interface simples e explicativa que ilustra o modo de utilização do sistema. Por meio dessas imagens, é possível distinguir as variações entre a interface de um funcionário cobrador e de um funcionário administrador. As diferenças entre essas interfaces são aprimoradas nas figuras, oferecendo uma compreensão clara das características distintas entre os dois tipos de usuário.

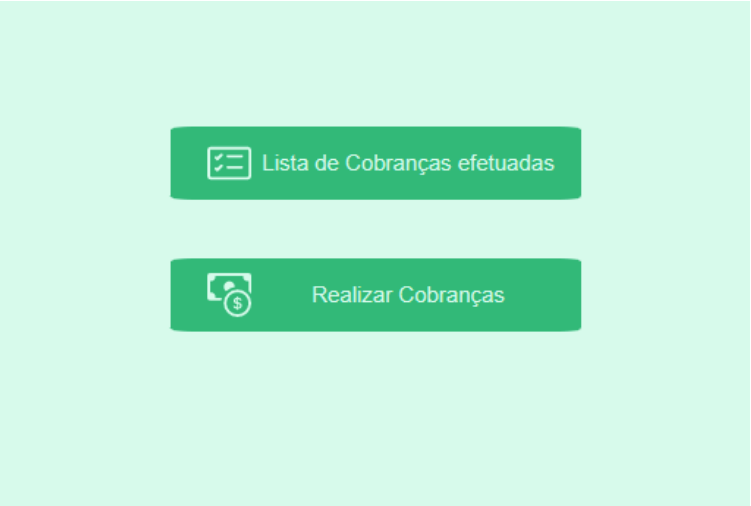


Figura 6 - Tela home de um cobrador

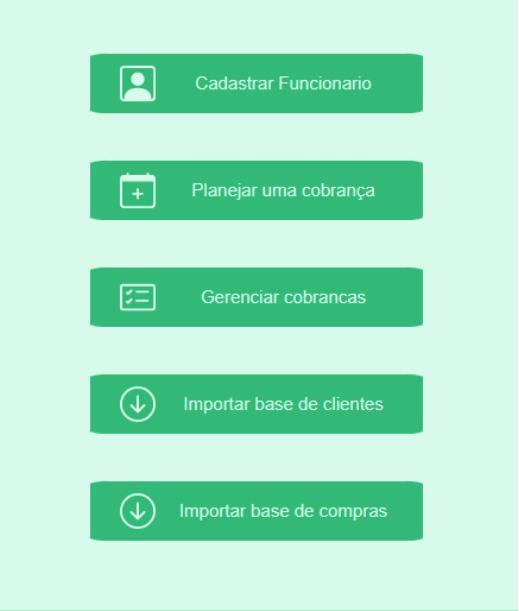


Figura 7 - Tela home de um Administrador

3.4.4 Definindo menu do usuário

Após receber os dados, o PHP realiza uma verificação no banco de dados para confirmar se o nome de usuário e a senha correspondem a um usuário existente. Uma vez que a identidade seja confirmada, o sistema recupera o ID e o tipo de funcionário associado a partir das sessões criadas no login. A função

representada na Figura 8 tem a responsabilidade de extrair o tipo de funcionário e redirecioná-lo para a página inicial, seja ela destinada a um cobrador ou a um administrador.

```php
if ($_SESSION['id_funcionario'] >= 1)
{

    //echo "usuario logado";

    if($_SESSION['tipo_funcionario'] != 1){
        echo("<script>
        window.location.href = 'http://localhost:8080/homecobrador'
        alert('Bem-vindo   " ) ;
        echo($_SESSION['nome']);
        echo ("');</script>");
    }else {
        echo("<script>
        window.location.href = 'http://localhost:8080/homeadmin'
        alert('Bem-vindo  " ) ;
        echo($_SESSION['nome']);
        echo ("');</script>");
    }

```

Seguir o link (ctrl + clique)

Figura 8: Função para encaminhar o funcionário para uma home

3.4.6 Cadastro de funcionários

A Figura 9 apresenta a página dedicada à criação de novos cadastros de funcionários. Nessa página, são disponibilizados campos de entrada (inputs) nos quais é possível inserir um nome de usuário e uma senha. Além disso, há um menu suspenso (selecionar) que permite escolher entre as opções "cobrador" e "administrador". É importante ressaltar que somente um funcionário com perfil administrador tem permissão para executar o processo de criação de novos cadastros.

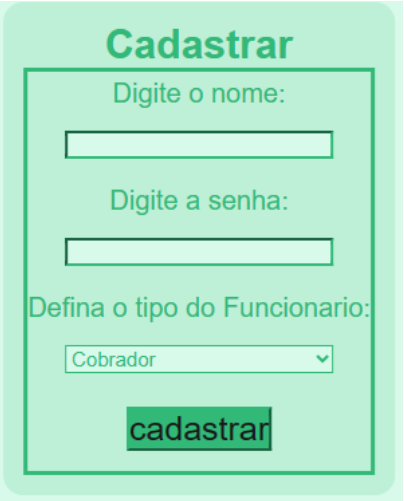


Figura 9 - Cadastro de um funcionário

3.4.7 Verificações para cadastrar

Para realizar o cadastro de um novo funcionário, o processo segue os seguintes passos: primeiro se verifica com na sessão se o usuário logado possui o status de administrador. Em seguida, ocorre uma análise dos dados fornecidos pelo front-end para garantir que sejam diferentes dos dados já presentes nos registros de funcionários cadastrados.

Após essas verificações, os dados deste novo funcionário são enviados por meio de uma consulta (query) para serem inseridos no banco de dados. Logo após é feito outra consulta (query) onde se resgata o id do funcionário que foi cadastrado e o id do tipo de funcionário e adiciona o id com os dados do funcionário e o id do tipo de funcionário que ele foi registrado. Um exemplo detalhado desse procedimento encontra-se ilustrado na Figura 10.

```php
if($_SESSION['tipo_funcionario'] == 1){
    $nome =$_POST["nome"];
    $senha=$_POST["senha"];
    $tipofuncionario= $_POST["tipofuncionario"];
    $configDB = new ConfigBD;
    $conn = $configDB->getConfig();

    $funcionarioDAO = new FuncionariosDAO();
    $validar = $funcionarioDAO->ValidarLogin($nome, $senha);
    $tipofunc =  $funcionarioDAO->getTipoFuncionarios($tipofuncionario);


    if(isset($_POST['cadastrar'])){
        if($validar < 1){
            //Cria um novo funcionario
            $query ="INSERT INTO `funcionario` (`nome`, `senha`)
            VALUES ('$nome', '$senha')";
            $funcionario = mysqli_query($conn,$query);
            if($funcionario){
                $idfuncionario = $funcionarioDAO->getIdFuncionario();
                    if($idfuncionario != 0){
                        if($tipofunc != 0){
                            //cria um novo funcionario com tipo e id;
                                $query1 = "INSERT INTO `funcionario_tipofun` (`id_funcionario`, `id_tipofuncionario`)
                                VALUES ('$idfuncionario', '$tipofunc')";
                                $inserir = mysqli_query($conn, $query1);
```

Figura 10: Função para cadastro de novos funcionários.

3.4.8 Importação de base de dados

Cada comércio terá sua própria base de clientes e histórico de compras, que serão gerenciados pelo administrador. No momento do envio das compras importadas, o backend executa uma verificação rigorosa, garantindo que apenas o administrador tenha permissão para realizar essa função. Essa restrição é aplicada aos cobradores, os quais não possuem autorização para realizar tal tarefa.

Para o desenvolvimento desta pagina é disponibilizado um campo de entrada (input) type ”file” para que possa ser enviado o arquivo com os dados seja dos clientes ou das compras e um botão para enviar esses dados para o back-end.

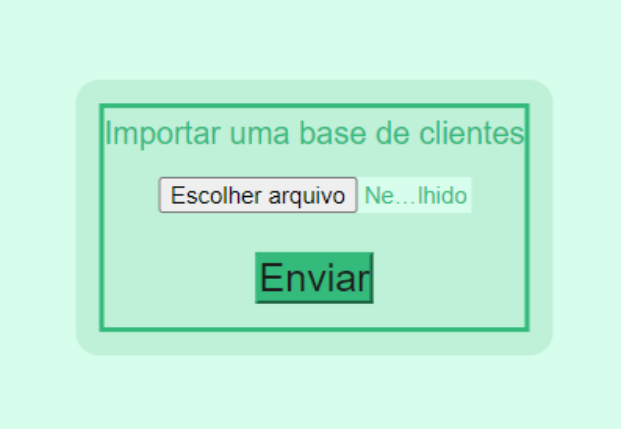


Figura 11: Tela de importação de dados

3.4.9 Inicializando sessões

Para iniciar uma session/sessão em uma página, na primeira linha do código é necessário colocar o seguinte código representado na Figura 12:

Figura 12- Iniciando a sessão

No exemplo sugerido na Figura 13 para se criar a sessão é utilizado o id_funcionario, que é encaminhado a partir do login feito pelo mesmo. As sessões são empregadas para transferir dados entre diferentes páginas. A sessão "tipo_funcionario" determina se um funcionário é um cobrador ou um administrador, direcionando-os para as respectivas páginas e concedendo acesso às funções correspondentes a cada perfil.

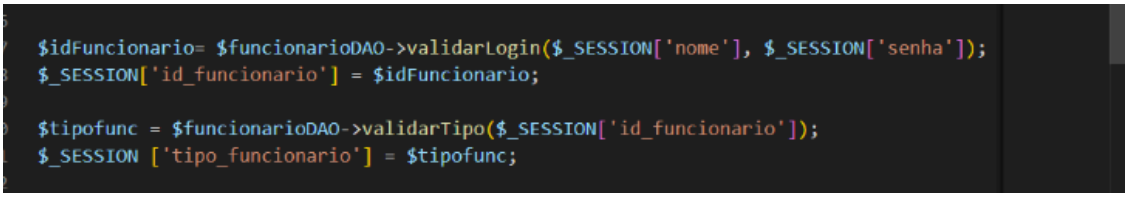


Figura 13- Criando sessões

3.4.10 Importação da base de dados

Antes de iniciar a importação dos dados, uma sessão do tipo de funcionário é recuperada. Após a verificação do usuário, o processo é concluído através do envio para uma pasta interna chamada "upload", onde um arquivo no formato .json é armazenado. O sistema, então, percorre toda a tabela no banco de dados e verifica se há ocorrência de dados repetidos. Somente os dados distintos são armazenados, enquanto os restantes são ignorados.

```php
if($_SESSION['tipo_funcionario'] == 1){
    if(isset($_FILES['arquivos'])){
        $obUpload = new Upload($_FILES['arquivos']);
        $sucesso = $obUpload->upload(__DIR__.'/upload');
        if($sucesso){
            $arquivoClientes = $obUpload->getRota(__DIR__.'/upload');
            $json= file_get_contents($arquivoClientes);
            $decodificado = json_decode($json,true);

            foreach($decodificado as $cliente){
            $query = "SELECT * FROM `cliente` WHERE id_cliente = '".$cliente['id_cliente']."' ";
            $verifique = mysqli_query($conn, $query);

                if(mysqli_num_rows($verifique)==0){
                $query= "INSERT INTO `cliente` ( `id_cliente`,`nome`, `endereco`, `telefone`, `cpf`)
                VALUES ('".$cliente['id_cliente']."','".$cliente['nome']."', '".$cliente['endereco']."', '".$cl
                $inserir = mysqli_query($conn, $query);
                }
```

Figura 14 - Código de importação da base de dados

3.4.11 Planejamento de cobranças.

Primeiramente, é necessário obter uma lista de clientes devedores. Para isso, é criado um método na instância "methods" que realiza uma requisição GET utilizando o axios, conforme exemplificado na Figura 15. Esse método é responsável por trazer os dados dos clientes devedores para a página.

```javascript
methods: {
    async getClientes() {
    const response = await axios.get('http://localhost/validandosc/php/controller/listadeclientes.php');
    if(response.status == 200){
        this.clientes = response.data;
    }else{
        console.log("erro de api")
    }
},
```

Figura 15: Função para pegar a lista de clientes

Para economizar uma cobrança, é fundamental salvar a lista de clientes devedores. Nesse processo, a utilização de componentes, como o "ListaClientes", desempenha um papel crucial.

O componente apresentado na Figura 16 inclui um formulário que realiza uma requisição POST, enviando o nome selecionado da lista. Esse formulário

cumpre um papel fundamental no planejamento da cobrança, permitindo segurança direta com os dados dos clientes devedores.

```
<template>
    <div name="form">
        <tr>
            <td>
                <div class="titulo">
                <label for="cliente">Selecione o cliente:</label>
                <form @change="postCliente()" id="inputform" name="inputform" method="POST">
                    <select required name="nomecliente" id="nomecliente" v-model="form.nomecliente"
                    :key="nomecliente"  >
                        <option v-for="cliente in clientes" :key="cliente.id"
                        :value="cliente.nome"  >{{cliente.nome}}
                        </option>
```

Figura 16: Componente ListaClientes

Para utilizar esse componente, siga os passos a seguir:

1. Importe o arquivo do componente para o arquivo em que ele será usado, dentro da seção de script.
2. Após a importação, adicione o nome do componente à seção "components" na instância Vue, conforme exemplificado na Figura 17.

Dessa forma, o componente estará pronto para ser utilizado e renderizado na página de acordo com sua implementação.

```
<script>
import ListaClientes from "../components/ListaClientes.vue";
import ListaFuncionarios from "../components/ListaFuncionarios.vue";
import axios from 'axios';
import Menu from '@/components/Menu.vue';

export default{
    name:'PlanejarCobranca',

    components:{
        ListaClientes,
        ListaFuncionarios,
        Menu
    },
```

Figura 17: Importando componentes.

Para usar estes componentes dentro do corpo o que se pode fazer é colocar o nome do componente como uma tag como se mostra na Figura 18.

```
6           <div class="cadastro">
7               <h1>Planejar uma cobrança</h1>
8               <form action="http://localhost/validandosc/php/controller/planejarCobranca.php"
    method="POST">
9                   <div class="fieldset">
10                      <fieldset>
11                          <ListaClientes required  name="nomecliente" :clientes ="clientes"/>
```

Figura 18: Adicionando um componente ao corpo

A Figura 19 apresenta uma interface visual para o planejamento de uma cobrança. O processo é detalhado da seguinte forma:

1. Inicialmente, o cliente a ser cobrado é selecionado. Ao fazer essa seleção, o saldo devedor do cliente é exibido.
2. posteriormente, o cobrador para essa cobrança é escolhido.
3. Em seguida, o valor a ser cobrado é inserido.
4. Por último, um dado para a conclusão da cobrança é especificado.
5. Para finalizar o processo, basta clicar no botão designado para enviar a cobrança para o banco de dados.

Dessa maneira, a interface fornece um fluxo intuitivo e direto para planejar e executar cobranças de maneira eficiente.

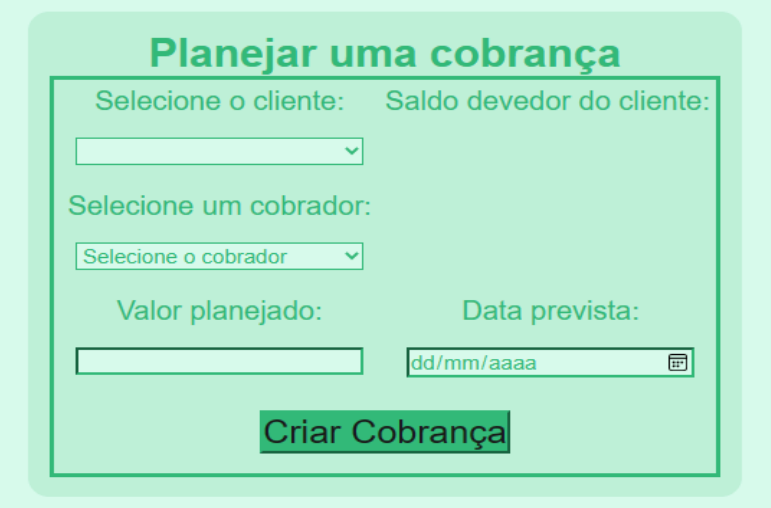


Figura 19 - Tela para planejar uma cobrança

3.4.12 Lista para exibir os clientes com compras pendentes

Para criar uma lista contendo todos os clientes devedores, o processo se inicia com o envio de uma consulta (query) ao banco de dados. Essa consulta seleciona todos os clientes cujas compras estão marcadas como não pagas. Dessa forma, o banco de dados retorna todos os clientes devedores, além de outros dados relevantes. A função que executa essa consulta pode ser visualizada na Figura 20.

```
4       public function selecionarClientes(){
5           //usado em: ListaDeCliente
6           $query= "SELECT DISTINCT compras.id_cliente, cliente.nome, cliente.id_cliente, cliente.cpf FROM
    compras INNER JOIN cliente
7           ON compras.id_cliente = cliente.id_cliente WHERE pago = 0";
8
9           $dao = new DAO();
10          $resultado = $dao->selecionarNLinhasSQL($query);
11          $clientes=array();
12          while ($linha = mysqli_fetch_assoc($resultado)) {
13
14              $cliente = new Cliente();
15              $cliente->idcliente = $linha["id_cliente"];
16              $cliente->nome = $linha["nome"];
17              $cliente->cpf = $linha["cpf"];
18
19              array_push($clientes, $cliente);
20          }
21          return $clientes;
22      }
```

Figura 20: Função que retorna a lista de clientes devedores.

3.4.13 Função para planejar uma compra

Para planejar uma cobrança, o administrador deve fornecer os seguintes dados: o cliente alvo da cobrança, o funcionário responsável pela execução da cobrança, o valor planejado para a cobrança e os dados previstos para sua execução. Após a inserção desses dados, uma consulta (query) é criada para salvar essas informações no banco de dados. A função responsável por executar essa operação está definida na Figura 21.

```
                1 reference | 0 overrides
201             public function PlanejarCobranca($idCliente, $idCobrador, $valorPlanejado, $dataPrevista){
202                 $query = "INSERT INTO `cobranca` (`id_cliente`, `id_funcionario`, `valor_planejado`,
    `data_prevista`)
203                 VALUES ('$idCliente', '$idCobrador', '$valorPlanejado', '$dataPrevista')";
204                 $configDB = new ConfigBD;
205                 $conn = $configDB->getConfig();
206                 $executar = mysqli_query($conn, $query);
207                 return $executar;
208             }
209
210
211     }
```

Figura 21: Função para guardas a cobrança planejada.

3.4.14 Exibição de lista de cobrança do cobrador

Para exibir uma lista de cobranças destinadas ao cobrador, é utilizado um formato de tabela. Para criar essa tabela, inicie-se com uma tag <table>. Dentro dela, uma tag <thead>é usada para inserir os títulos dos dados, e uma tag <tbody>é utilizada para mostrar os dados recebidos do back-end.

O código-fonte desenvolvido para essa finalidade pode ser encontrado na Figura 22, e a tabela resultante é ilustrada na Figura 23. Esse arranjo permite uma organização clara e estruturada dos dados de cobrança, tornando mais eficaz o trabalho do cobrador.

```
<template>
<div v-if="tableVisivel">
    <h1>Realizar cobrança</h1>
    <div class="tabela">
        <table>
            <thead>
                <tr>
                    <th>Id:</th>
                    <th>Cliente:</th>
                    <th>Endereço</th>
                    <th>Valor planejado:</th>
                    <th>Data prevista:</th>
                    <th>Cobrar</th>
                </tr>
            </thead>
            <tbody>
                <tr  v-for="(cobranca,index) in cobrancas" :key="cobranca.id">
                    <td class="valorTabela">{{ cobranca.idCobranca}}</td>
                    <td>{{ cobranca.nome}}</td>
                    <td>{{ cobranca.endereco }}</td>
                    <td class="valorPlanejado">{{ cobranca.valorPlanejado }}</td>
                    <td>{{ cobranca.dataPrevista }}</td>
                    <td>
                        <button  id="botao" type="submit" value="post" @click="pegarIdCobranca(index),
pegarValorPlanejado(index), DeixaInvisivel()">Cobrar</button>
```

Figura 22: Desenvolvendo a tabela.

Id:	Cliente:	Endereço	Valor planejado:	Data prevista:	Cobrar
226	Romeu	Rua dez	200,50	2023-08-24	Cobrar
227	João	Rua sete	418,00	2023-09-01	Cobrar
228	Paulo	Rua Dois	50,00	2023-08-31	Cobrar

Figura 23: Tabela cobranças planejadas de um cobrador.

3.4.15 Lista com as cobranças planejadas para um cobrador.

Para criar as listas de cobrança eletrônica, o procedimento é o seguinte:

1. Inicialmente, verifique-se a partir de uma sessão qual é o ID do funcionário logado.
2. Com o ID do funcionário, o PHP executa uma consulta (query) que recupera a lista de todas as cobranças iniciadas associadas a esse cobrador pelo nome.
3. O exemplo desse processo é apresentado na Figura 24.

Isso permite que o sistema apresente as cobranças automatizadas específicas para cada cobrador, facilitando o acompanhamento das tarefas de cobrança atribuídas a eles.

```
27          $query =
28          " SELECT  cobranca.*, cliente.nome, cliente.id_cliente, cliente.endereco FROM cobranca INNER JOIN
    cliente
29          ON cobranca.id_cliente = cliente.id_cliente WHERE cobranca_efetuada = 0  AND id_funcionario =
    '".$idfuncionario."'";
30          $dao = new DAO();
31          $resultado = $dao->selecionarNLinhasSQL($query);
32          $cobrancas=array();
33          $cliente = new Cliente();
34
35          while ($linha = mysqli_fetch_assoc($resultado)) {
36              $cobranca = new Cobrancas();
37              $cobranca->idCobranca = $linha["id_cobranca"];
38              $cobranca->idCliente = $linha["id_cliente"];
39              $cobranca->endereco = $cliente->endereco= $linha["endereco"];
40              $cobranca->idFuncionario = $linha["id_funcionario"];
41              $cobranca->valorPlanejado =number_format($linha["valor_planejado"],2, ',', '.' );
42              $cobranca->valorPago = $linha["valor_pago"];
43              $cobranca->dataPrevista = $linha["data_prevista"];
44              $cobranca->dataHoraExec = $linha["data_hora_exec"];
45              $cobranca->cobrancaEfetuada = $linha["cobranca_efetuada"];
46              $cobranca->nome = $cliente->nome = $linha['nome'];
47              array_push($cobrancas, $cobranca);
48              }
49              return $cobrancas;
50          }
```

Figura 24: Lista planejada para cobradores

3.4.16 Criação de um componente para realizar uma cobrança

Para executar uma cobrança, ao simplificar de criar uma página separada com um novo formulário, foi desenvolvido um componente contendo esse formulário. Isso permite a reutilização do código e uma melhor organização. Um exemplo desse componente é mostrado na Figura 25. Dessa forma, o processo de execução de cobrança é simplificado e integrado ao sistema de forma mais eficiente.

```
<label>Id Cobrança:</label>
<label v-for="codigo in codigos" :key="codigo.id">{{ codigo }}
    <input  required type="hidden" name="idCobranca" :value="codigo">
</label>
<br>
<label>Valor Planejado:</label>
<label v-for="valor in valorPlan" :key="valor.id">{{ valor }}
    <input type="hidden" name="valorPlanejado" :value="valor">
</label>
<br>
<div class="cobrarplan">
    <label>Valor Pago:</label><br>
    <input type="text" required name="valorPago"> <br>
</div>
<div class="cobrarplan">
    <label>Data e hora de Execução:</label><br>
    <input type="text" required name="dataHoraExec" :value="fulldatetime"> <br>
</div>
<button name="cobrar">Cobrar</button>
```

Figura 25: Componente para cobrança

Esse componente fica visível quando o usuário clica no botão com o rótulo "Cobrar" na tabela de cobranças eletrônicas. O exemplo desse componente é apresentado na Figura 26. Esse comportamento possibilita que o formulário de cobrança seja exibido somente quando necessário, mantendo a interface mais limpa e intuitiva para o usuário.

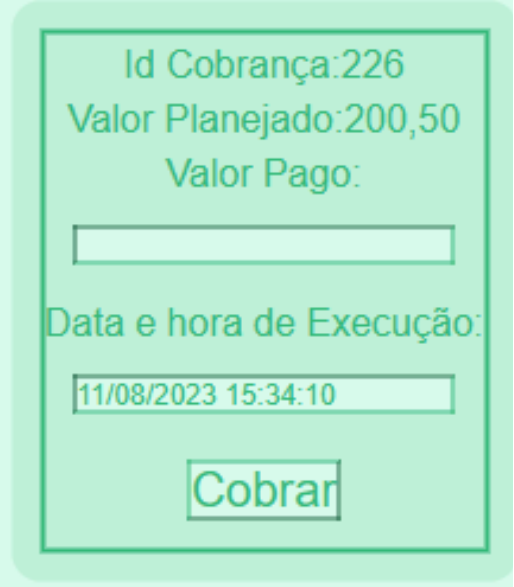


Figura 26: Formulário de cobrança

Para tornar esse componente visível ou invisível de acordo com o clique, utilize-se a diretiva v-if em conjunto com uma função que determina quando o componente deve ser exibido. A Figura 27 ilustra como essa diretiva pode ser utilizada em uma chamada de componente.

Ao aplicar a diretiva v-if com uma condição apropriada, o componente será renderizado somente quando a condição for verdadeira, permitindo que ele emocionalmente ou desapareça dinamicamente com base nas extremidades do usuário. Isso proporciona uma experiência mais interativa e responsiva para os usuários.

```
39      <Cobrar v-if="visivel" :codigos="codigos" :identidades="identidades" :valorcompras="valorcompras">
40          <button>Tabela</button>
41      </Cobrar>
```

Figura 27: Diretiva para deixar visível

Para evitar que o cobrador insira a data e hora incorretamente, a função representada na Figura 28 captura a data e a hora no momento exato em que o botão para chamar o componente é clicado.

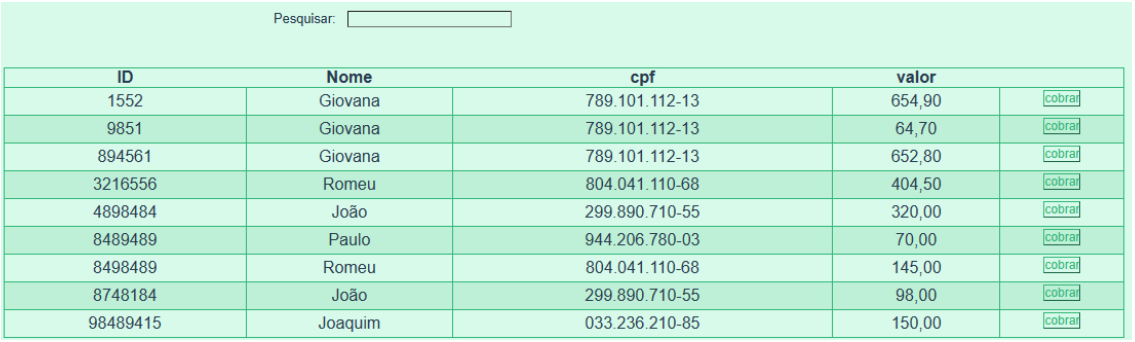
Essa abordagem assegura que os dados e a hora registrada sejam precisos e correspondam ao momento exato em que o usuário iniciou o processo de cobrança. Dessa forma, a integridade dos registros de cobrança é mantida e o risco de inserção de informações errôneas é minimizado.

```
60      methods:{
61          printFullDate: function(){
62
63          let date =  new Date().toLocaleDateString();
64          let time = new Date().toLocaleTimeString();
65          return date + ' ' + time;
66          },
67          getNomeCliente(){
68              this.clientes.push({
69              nome:this.nomeCliente
70          })
71          },
72          },
73      mounted(){
74          this.fulldatetime = this.printFullDate();
75      }
76          }
```

Figura 28: Função de data e hora

3.4.17 Compras avulsa com um input de pesquisa

Para possibilitar que um cliente possa realizar um pagamento antes de ser cobrado, foi criada uma tabela específica que registra as compras que ainda não foram pagas, como demonstrado na Figura 29. Essa tabela permite que os clientes visualizem e identifiquem quais compras estão pendentes de pagamento e ter a opção de fazer o pagamento de maneira proativa, antes mesmo de serem contatados para cobrança. Isso proporciona aos clientes maior autonomia e controle sobre suas transações


Pesquisar:

ID	Nome	cpf	valor	
1552	Giovana	789.101.112-13	654,90	cobrar
9851	Giovana	789.101.112-13	64,70	cobrar
894561	Giovana	789.101.112-13	652,80	cobrar
3216556	Romeu	804.041.110-68	404,50	cobrar
4898484	João	299.890.710-55	320,00	cobrar
8489489	Paulo	944.206.780-03	70,00	cobrar
8498489	Romeu	804.041.110-68	145,00	cobrar
8748184	João	299.890.710-55	98,00	cobrar
98489415	Joaquim	033.236.210-85	150,00	cobrar

Figura 29: Tabela para cobranças avulsas.

Para habilitar a pesquisa por um cliente, é inserido um campo de entrada (input) onde o usuário pode digitar os critérios de pesquisa. Para realizar essa pesquisa, é necessário computar os dados digitados e filtrar uma lista de clientes de acordo com esses critérios. A Figura 30 demonstra a instância computed que executa a função de filtragem dos clientes.

Ao digitar informações no campo de pesquisa, essa função filtra a lista de clientes para exibir somente aqueles que atendem aos critérios definidos pelo usuário, tornando a experiência de pesquisa mais eficaz e conveniente.

Figura 30: Instância computed para filtros

3.4.18 Lista completa com as cobranças criadas por um administrador

Para obter uma lista completa de todas as cobranças efetuadas, é necessário realizar uma eficiência (JOIN) entre duas tabelas do banco de dados: a tabela de clientes e a tabela de cobranças. Nesse caso, é utilizada a cláusula INNER JOIN para unir essas tabelas. Para determinar quais clientes devem ser incluídos na lista, é necessário selecionar o ID do cliente da tabela de clientes que corresponde ao ID do cliente na tabela de cobranças, e ainda verificar se a cobrança foi efetuada (valor igual a 1).

Um exemplo completo dessa função é apresentado na Figura 31. Essa consulta possibilita a obtenção de uma visão abrangente das cobranças já realizadas, com informações relevantes dos clientes associados a cada cobrança.

```php
1 reference | 0 overrides
public function cobrancasEfetuadas(){
    //usado em :ListaDeCobrancasEfetuadas
    $query = "SELECT  cobranca.*, cliente.nome, cliente.id_cliente, cliente.cpf FROM cobranca INNER JOIN cliente
    ON cobranca.id_cliente = cliente.id_cliente WHERE cobranca_efetuada = 1 ";
    $dao = new DAO();
    $resultado = $dao->selecionarNLinhasSQL($query);
    $cobrancas=array();
    $cliente = new Cliente();
    while ($linha = mysqli_fetch_assoc($resultado)) {
        $cobranca = new Cobrancas();

        $cobranca->idCobranca = $linha["id_cobranca"];
        $cobranca->idCliente = $linha["id_cliente"];
        $cobranca->idFuncionario = $linha["id_funcionario"];
        $cobranca->valorPlanejado = number_format($linha["valor_planejado"],2, ',', '.' );
        $cobranca->valorPago = number_format($linha["valor_pago"],2, ',', '.' );
        $cobranca->dataPrevista = $linha["data_prevista"];
        $cobranca->dataHoraExec = $linha["data_hora_exec"];
        $cobranca->cobrancaEfetuada = $linha["cobranca_efetuada"];
        $cobranca->nome = $cliente->nome = $linha['nome'];

        array_push($cobrancas, $cobranca);
        }
        return $cobrancas;
```

Figura 31: Função com cobrança realizadas

Além das cobranças já efetuadas, também existem as cobranças que foram iniciadas, mas ainda não foram concluídas. Para obter essa lista, é necessário executar uma nova consulta (query) que traga as cobranças que ainda não foram efetuadas. Diferentemente da lista anterior, nesse caso é importante que a coluna "cobranca_efetuada" esteja com o valor igual a 0, indicando que a cobrança ainda não foi realizada.

A Figura 32 ilustra como realizar essa consulta, selecionando as cobranças que atendem a esses critérios específicos. Isso proporciona uma visão completa das cobranças iniciadas, incluindo aquelas que ainda estão pendentes de execução.

```php
public function listaExcluir(){
    //Usado em: ListaDeCobrancasEfetuadas
    $query = $query =
    " SELECT  cobranca.*, cliente.nome, cliente.id_cliente, cliente.cpf FROM cobranca INNER JOIN
cliente
    ON cobranca.id_cliente = cliente.id_cliente WHERE cobranca_efetuada = 0";
    $dao = new DAO();
    $resultado = $dao->selecionarNLinhasSQL($query);
    $cobrancas=array();
    $cliente = new Cliente();

    while ($linha = mysqli_fetch_assoc($resultado)) {
        $cobranca = new Cobrancas();
        $cobranca->idCobranca = $linha["id_cobranca"];
        $cobranca->idCliente = $linha["id_cliente"];
        $cobranca->idFuncionario = $linha["id_funcionario"];
        $cobranca->valorPlanejado =number_format( $linha["valor_planejado"],2, ',', '.' );
        $cobranca->valorPago =number_format($linha["valor_pago"],2, ',', '.' );
        $cobranca->dataPrevista = $linha["data_prevista"];
        $cobranca->dataHoraExec = $linha["data_hora_exec"];
        $cobranca->cobrancaEfetuada = $linha["cobranca_efetuada"];
        $cobranca->nome = $cliente->nome = $linha['nome'];

        array_push($cobrancas, $cobranca);
        }
        return $cobrancas;

    }
```

Figura 32: Função que lista as cobranças não efetuadas.

3.4.19 Gerenciamento de cobranças

Para permitir que um administrador tenha controle sobre as cobranças realizadas e pendentes, a página de gerenciamento é projetada com uma lista que abrange todas as cobranças automatizadas e as cobranças que foram efetuadas avulso por esse administrador. A tabela apresentada ao usuário divide-se em duas partes: primeiro, são exibidas todas as cobranças pendentes, e em seguida, são mostradas as cobranças que já foram efetuadas.

Conforme ilustrado na Figura 33, as cobranças efetuadas são marcadas com um ícone de aprovação, enquanto as pendências são identificadas por um ícone de lixeira. Ao clicar no ícone de lixeira, um componente é acionado para permitir que a cobrança seja excluída. Essa abordagem oferece um método intuitivo para que o administrador possa gerenciar as cobranças de forma eficaz, visualizando claramente o status de cada uma e tendo a opção de tomar medidas específicas conforme necessário.

Id:	Cliente:	Funcionario:	Valor Planejado:	Valor Pago:	Data Prevista:	Data e Hora Execução:	Excluir
220	Giovana	85	1.000,00	0,00	2023-04-23		
226	Romeu	3	200,50	0,00	2023-08-24		
227	João	3	418,00	0,00	2023-09-01		
228	Paulo	3	50,00	0,00	2023-08-31		
209	João	4	500,00	100,00	2023-04-08	14/03/2023 13:34:24	
210	Giovana	4	300,00	500,00	2023-04-07	15/03/2023 16:13:17	
211	Giovana	4	0,00	64,70		15/03/2023 16:13:54	
212	Joaquim	4	0,00	150,00		15/03/2023 16:26:30	
215	Romeu	4	25,00	25,00	2023-04-06	15/03/2023 16:50:27	
219	Romeu	4	300,00	200,00	2023-04-06	23/04/2023 20:11:01	
221	João	4	320,00	200,00	2023-05-27	15/05/2023 19:24:28	
222	Joana	4	70,90	70,90	2023-06-09	15/05/2023 19:28:32	
223	Paulo	4	100,00	50,00	2023-06-09	15/05/2023 19:31:57	
224	Joana	4	100,00	100,00	2023-06-09	15/05/2023 19:34:37	
225	Romeu	3	100,00	50,00	2023-08-05	07/07/2023 21:55:03	

Figura 33: Lista de cobranças

3.4.20 Função para excluir uma cobrança

Para excluir uma cobrança, é necessário realizar uma consulta do tipo DELETE, em que o ID da cobrança corresponde ao ID recebido a partir do front-end. A Figura 34 apresenta um breve exemplo de como essa consulta pode ser integrada.

Ao utilizar essa consulta, a cobrança selecionada é removida do banco de dados, proporcionando um meio de exclusão direto e eficaz para o administrador.

```
1 reference | 0 overrides
171  public function excluir($idCobranca){
172      $query = "DELETE FROM cobranca WHERE `id_cobranca` = $idCobranca";
173      $configDB = new ConfigBD;
174      $conn = $configDB->getConfig();
175      $executar = mysqli_query($conn, $query);
176      return $executar;
177  }
```

Figura 34: Função para excluir uma cobrança

4 CONCLUSÃO

Espera-se que, num futuro próximo, esse sistema possa ser adotado por várias empresas, beneficiando-se com uma gestão financeira aprimorada e melhorando a experiência tanto dos administradores quanto dos clientes no processo de cobrança. Isso contribuirá para a eficiência e a transparência nas operações financeiras das empresas.

Ao concluir todo o processo, o objetivo deste trabalho foi criar um sistema para oferecer suporte a pequenos estabelecimentos, permitindo-lhes delegar aos seus funcionários a responsabilidade de realizar cobranças.

A escolha do Vue.js foi mostrada de forma mais vantajosa, uma vez que o uso de componentes disponibilizados pelo Vue.js facilitou a criação de elementos que podem ser alternados entre visíveis e invisíveis. Isso possibilitou a criação de componentes em vez de várias páginas separadas, criando a sensação de redirecionamento para diferentes páginas, sem sobrecarregar o sistema com múltiplas requisições de páginas a cada clique.

A integração do PHP, uma linguagem que trabalha bem em conjunto com o front-end, trouxe facilidade nas transferências de dados, tanto da parte do front-end para serem armazenados no banco de dados, quanto do banco de dados para serem usados no front-end. Isso contribuiu para uma experiência fluida e eficiente no sistema.

Compreendendo o propósito final do sistema, fica evidente que os funcionários e cobradores devem ser capazes de analisar e desempenhar suas respectivas funções de maneira prática e específica, com maior objetividade.

A abordagem de codificação do sistema, onde cada parte é separada e introduzida de acordo com sua necessidade, é uma prática de desenvolvimento excelente. Isso ajuda a garantir que o código permaneça organizado e não se torne repetitivo. Além disso, essa estrutura modular torna a manutenção do sistema mais fácil, permitindo que qualquer atualização ou modificação seja feita de forma eficiente. Basta ir diretamente à parte específica que requer alterações e realizar as devidas modificações, sem afetar outras partes do sistema. Isso contribui para a escalabilidade e a robustez do software, facilitando o gerenciamento e a evolução contínua do sistema.

No início do projeto, uma das principais dificuldades consiste na necessidade de utilizar uma dependência para realizar requisições a APIs. Nesse contexto, a ferramenta Axios, embora houvesse outras alternativas disponíveis. A escolha pela Axios implicou em adquirir novos conhecimentos e exigiu esforços adicionais para aprofundar na compreensão sobre seus métodos de requisição dinâmica.

Uma outra dificuldade significativa foi a prática de importação de dados. Este desafio é mostrado como um obstáculo específico, dado que implicava que o PHP precisasse processar extensos arquivos JSON para verificar a presença de duplicatas em todos os campos do banco de dados e , em seguida, adicionar apenas os dados diferentes. Nesse contexto, a maior complexidade reside em garantir que o PHP seja capaz de receber e processar eficientemente todos esses dados.

Um desenvolvimento que enriqueceu a experiência foi a incorporação do banco de dados PHPMyAdmin. A facilidade proporcionada por esta ferramenta na criação e preenchimento de tabelas declaradas é de grande valia, permitindo-nos conduzir testes de maneira versátil e dinâmica.

Para diagnosticar e compreender as falhas em algumas funcionalidades, foi necessário criar páginas adicionais em PHP com o propósito de localizar a origem do erro. Erros frequentes ocorrem durante o processo de envio de dados, o que exigia a leitura inicial dos dados pelo PHP antes de averiguar se o problema reside na requisição ou na implementação em Vue.js.

O design no início era escuro e altamente didático, o que foi concebido de forma a incorporar variáveis para controlar as cores. Essa abordagem se revelou particularmente benéfica ao realizar uma alteração abrangente na paleta de cores do projeto. Com uma simples modificação nas variáveis criadas, conseguimos fazer uma mudança abrangente em todo o projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

CAPUCÇO, V. D. O. (2021). Vantagens e desvantagens no desenvolvimento de aplicações com frameworks e bibliotecas: uso de Vue, Bulma e Buefy. Disponivel em: <http://ric.cps.sp.gov.br/bitstream/123456789/8826/1/ads_2021_2_victoriodeoliveiracapucco_vantagensedesvantagensnodesenvolvimentodeaplica%c3%a7%c3%b5es.pdf >

Clark, M. Top 10 startups usando Vue, 2023, disponível em: <https://blog.back4app.com/pt/startups-usando-vue/#Quais_sao_as_dez_melhores_startups_usando_VueJS>

Costa, C. J. (2007). Desenvolvimento para web. ITML press/Lusocredito. disponível em:<https://books.google.com.br/books?hl=pt-BR&lr=&id=Jn6dTDF-wcsC&oi=fnd&pg=PT5&dq=desenvolvimento+web&ots=wNgLNNY52f&sig=rHEpaB_q1Lk08pUoMhb9gastreM#v=onepage&q=desenvolvimento%20 web &f=false >

CROMWELL, V. Evan You. Between the Wires, 2016. Disponível em: < https://medium.com/free-code-camp/between-the-wires-an-interview-with-vue-js-creator-evan-you-e383cbf57cc4 >

Dall'Oglio, P. (2015). PHP Programando com Orientação a Objetos 3ª Edição. Novatec Editora. disponível em: <https://scholar.google.com.br/scholar?hl=pt-BR&as_sdt=0%2C5&q=o+que+%C3%A9+o+php&btnG=#d=gs_cit&t=1686075231674&u=%2Fscholar%3Fq%3Dinfo%3A40yjGaMs-bcJ%3Ascholar.google.com%2F%26output%3Dcite%26scirp%3D4%26hl%3Dpt-BR >

Robles Herrera, Didier Emmanuel. Estudio de factibilidad de uso del Framework Vue.js como herramienta, para el desarrollo del front end de un aplicativo web, para la gestión académica en la Unidad Educativa Miguel Ángel Samaniego Jiménez en

la Parroquia La Unión, 2021, disponível em: < http://dspace.utb.edu.ec/handle/49000/9488>

Fernandes, J. H. C. (2003). Qual a prática do desenvolvimento de software?. *Ciência e Cultura*, *55*(2), 29-33.disponivel em:<http://cienciaecultura.bvs.br/scielo.php?pid=S0009-67252003000200021&script=sci_arttext >

Incau, C. (2017). *Vue. js: Construa aplicações incríveis*. Editora Casa do Código. Disponível em:<https://books.google.com.br/books?hl=pt-BR&lr=&id=Ft-8DgAAQBAJ&oi=fnd&pg=PT2&dq=diretivas+vuejs&ots=_acXc4L3cw&sig=5CBmY18dApTsPnohSh_4hSavfqc#v=onepage&q=diretivas%20vuejs&f=false >

Koch, M. (2009). Critérios para implantação de um sistema de gestão por processos aplicados a empresas de cobranças. disponivel em:<https://repositorio.ufsc.br/bitstream/handle/123456789/126924/Contabeis291356.pdf?sequence=1 >

Mattos, A. C. M. (2017). Sistemas de informação. Saraiva Educação SA <https://books.google.com.br/books?hl=pt-BR&lr=&id=SYNnDwAAQBAJ&oi=fnd&pg=PT8&dq=sistema+da+informa%C3%A7%C3%A3o&ots=aDE6cMbs1b&sig=F1HqZkbYbRfw908Is6hUt9Mo2Pc#v=onepage&q=sistema%20da%20informa%C3%A7%C3%A3o&f=false >

php.net (2023). Documentação PHP. DIsponível em: <https://www.php.net/manual/pt_BR/intro.session.php >

Tavares, D. F., & Cremasco, E. F. (2010). PHP. Disponível em:<http://www.inf.ufes.br/~vitorsouza/archive/2020/wp-content/uploads/teaching-lp-20162-seminario-php.pdf>

Vuejs.org (2023). Documentação Vue.js. Disponivel em: <https://vuejs.org/guide/introduction.html >

Wakulicz, G. J. (2016). Sistemas de Informação gerenciais. Disponível em: <https://central3.to.gov.br/arquivo/453437/ >

Wohlgethan, E. (2018). Apoiar decisões de desenvolvimento web comparando três principais frameworks javascript: Angular, react e vue. js (dissertação de doutorado, Hochschule für Angewandte Wissenschaften Hamburgo). Disponivel em: <https://reposit.haw-hamburg.de/bitstream/20.500.12738/8417/1/BA_Wohlgethan_2176410.pdf >

KALIN, M.; Java web services: Implementando. Uma introdução rápida, prática e completa. São Paulo: O'Reilly, 2010.

Eulalio, A. D., Cordeiro, D., & de Souza, R. (2016). WEB SERVICES: Integração De Sistemas Orientado a Serviços com uma Proposta de Aplicação na EAD. Revista de Informática Aplicada, 12(2). Disponível em: <https://www.seer.uscs.edu.br/index.php/revista_informatica_aplicada/article/view/6918 >