Sistema gerenciador de crediário para vendedor ambulante

Ruan Hasselmann Da Silva, Ana Paula Canal
Curso de Sistemas de Informação
Universidade Franciscana
CEP 97010-032 – Santa Maria - RS - Brasil
ruan.hasselmann@ufn.edu.br, apc@ufn.edu.br

***Resumo*—A venda ambulante é uma prática comercial com vantagens e desvantagens. Este tipo de comércio geralmente oferece preços mais baixos, mas pode ter menos segurança e garantia de qualidade. Nas vendas a prazo, há uma facilidade ao acesso dos bens, mas pode representar riscos financeiros à quem vende. O objetivo deste trabalho foi desenvolver um software para gerenciamento de vendas, em ambiente Web e móvel, utilizando Flutter, Java e HTML/CSS/React. As duas ferramentas, tanto web quanto mobile auxiliam no comércio de ambulantes, provendo um maior e mais seguro controle de pagamentos, diminuindo os riscos para ambos os lados, vendedor e cliente.**

Palavras-chave: **Sistema de gerenciamento; Pagamento a prazo; Tecnologia; Desenvolvimento Web; Mobile;**

I. Introdução

A venda ambulante é uma prática comercial que se desenvolve há muito tempo e que ainda hoje se mantém como uma forma de comércio presente em muitas cidades e regiões. Consiste na venda de produtos ou serviços em locais públicos, como ruas, praças, feiras e mercados, geralmente realizada por vendedores que não possuem estabelecimentos fixos.

A venda à vista é uma modalidade de transação comercial na qual o pagamento é realizado imediatamente, sem o uso de crédito ou parcelamento. É comumente utilizada por vendedores ambulantes e pequenos estabelecimentos, exigindo que o cliente disponha do valor integral no momento da compra. Essa forma de pagamento pode oferecer descontos ou vantagens, mas pode ser desafiadora para consumidores que não possuem o montante necessário imediatamente disponível.

A venda a prazo, por sua vez, é uma modalidade de venda em que o cliente adquire um produto ou serviço, mas não efetua o pagamento integral no momento da compra. Em vez disso, o pagamento é parcelado em prestações, que podem ser mensais, semanais ou quinzenais, dependendo do acordo entre as partes.

Ambas as práticas têm suas vantagens e desvantagens. A venda ambulante, por exemplo, pode oferecer preços mais baixos em comparação com as lojas tradicionais, uma vez que os vendedores não têm os mesmos custos fixos, como aluguel e manutenção de um estabelecimento comercial. Por outro lado, pode apresentar desafios em relação à segurança e à garantia da qualidade dos produtos, uma vez que, é difícil verificar a procedência dos itens comercializados.

Já a venda a prazo pode facilitar o acesso de consumidores de baixa renda a bens e serviços que eles não teriam condições de adquirir à vista. No entanto, o crédito pode representar um risco tanto para o vendedor quanto para o comprador, já que há a possibilidade de inadimplência e prejuízos financeiros, principalmente pela perca dos registros físicos da venda, inclusive pelo esquecimento de alguma das partes da existência de alguma parcela ainda não paga.

Para lidar com essas questões, é importante que tanto vendedores ambulantes quanto lojistas que oferecem vendas a prazo estejam atentos às regulamentações legais que envolvem essas práticas comerciais, como a necessidade de autorização para a venda ambulante e a observância do Código de Defesa do Consumidor para as vendas a prazo.

Além disso, é essencial que haja uma relação de confiança entre as partes envolvidas. Vendedores devem oferecer produtos de qualidade e preços justos, e cumprir com as condições de pagamento acordadas. Já os consumidores devem honrar seus compromissos financeiros, evitando o endividamento excessivo e o não pagamento das prestações acordadas.

Assim, a venda ambulante e as vendas a prazo podem ser formas válidas e importantes de comércio, desde que realizadas de forma consciente e responsável por ambas as partes.

Esta forma de comércio ainda realiza os registros de forma física em papel, isso tornando o registro sujeito a degradação do tempo, assim como danificações por conta de água, rasgos, entre outros eventos que podem ocorrer com papeis. Os registros digitais irão possibilitar uma maior garantia que aquelas informações não serão perdidas por conta de efeitos naturais, tornando assim o comércio mais preciso nas informações sobre o pagamento dos clientes, possibilitando um fácil acesso a informação desejada, sem a necessidade de procurar algum registro específico em uma pilha de arquivos.

A. Objetivos

O objetivo deste trabalho é desenvolver o aplicativo para smartphone, um sistema web/desktop, que efetue o gerenciamento dos clientes e seus crediários para vendedores ambulantes. Assim, pode-se elencar os objetivos específicos deste trabalho:

- Desenvolver um aplicativo em Flutter [1] para ser utilizado em IOS e Android [2].
- Desenvolver uma ferramenta *Back-end* [3] em Java [4], para ser executada no servidor [5].
- Desenvolver uma aplicação *Front-end* [3] em HTML [6], CSS [7] e React [8].
- Melhorar a efetividade dos registros, removendo os fatores de degradação que podem ocorrer com registros físicos.
- Aumentar a eficiência do processo de gerenciamento do crediário removendo a necessidade de possuir registros físicos.

II. Referencial Teórico

Esta seção visa fornecer uma base teórica para o entendimento das temáticas abordadas nesta pesquisa.

A. ERP (Enterprise Resource Planning)

É um sistema de gestão empresarial que integra todas as áreas de uma empresa, como finanças, vendas, produção, logística, recursos humanos, entre outras, em uma única plataforma [9]. O objetivo do ERP é centralizar e otimizar o gerenciamento de todas as atividades da empresa, tornando os processos mais eficientes, reduzindo custos e aumentando a produtividade. O sistema ERP permite a tomada de decisões mais rápidas e precisas, uma vez que as informações estão integradas e disponíveis em tempo real. Além disso, o ERP ajuda a empresa a atender às exigências legais, como a emissão de notas fiscais eletrônicas, e a se manter competitiva no mercado.

B. Marketplace

É uma plataforma online que conecta vendedores e compradores, permitindo a transação de produtos ou serviços [10]. É um local virtual onde múltiplos vendedores podem listar seus produtos ou serviços e os compradores podem pesquisar, comparar e fazer compras diretamente nessa plataforma.

Os *marketplaces* podem abranger uma ampla variedade de setores e produtos, como eletrônicos, moda, alimentos, serviços de hospedagem, entre outros. Eles oferecem uma experiência conveniente para os compradores, pois podem encontrar uma variedade de produtos ou serviços em um só lugar, além de comparar preços e avaliações antes de fazer uma compra.

C. E-commerce

E-commerce, abreviação de "comércio eletrônico", refere-se à compra e venda de produtos ou serviços realizados pela internet [11]. É um modelo de negócio que permite às empresas venderem seus produtos ou serviços online, alcançando um público global e facilitando transações entre compradores e vendedores através de plataformas digitais.

No e-commerce, os consumidores podem navegar pelos catálogos de produtos ou serviços, fazer compras online, efetuar pagamentos eletronicamente e receber os produtos ou serviços no local desejado. O comércio eletrônico pode ocorrer em diferentes formatos, como lojas online individuais, *marketplaces* [10] (onde múltiplos vendedores estão reunidos em uma única plataforma) ou mesmo através das redes sociais, onde as empresas podem vender diretamente aos usuários.

D. Flutter

Flutter é um framework de código aberto criado pela Google para desenvolvimento de aplicativos móveis para iOS, Android [2], web e desktop, utilizando uma única base de código [1]. Ele permite que os desenvolvedores criem aplicativos nativos de alta qualidade com uma experiência do usuário consistente em todas as plataformas. Flutter utiliza a linguagem de programação Dart, que foi desenvolvida pela própria Google, e possui *widgets* personalizados que podem ser facilmente combinados e personalizados para criar interfaces de usuário atraentes e interativas. Além disso, o Flutter possui um compilador *just-in-time* que permite a atualização em tempo real do código e um compilador *ahead-of-time* que gera código nativo altamente otimizado. Isso faz do Flutter uma ferramenta poderosa para desenvolvedores que buscam produtividade e eficiência no desenvolvimento de aplicativos móveis multiplataforma.

E. Diferenças entre Android e IOS

Android e iOS são dois sistemas operacionais para dispositivos móveis amplamente utilizados em todo o mundo [2]. Ambos têm suas vantagens e desvantagens em termos de usabilidade, recursos de usabilidade e recursos de programação. Algumas das principais diferenças entre Android e iOS incluem:

- Personalização: Android oferece maior flexibilidade de personalização do sistema operacional em comparação com o iOS. Os usuários do Android podem personalizar a aparência da tela inicial, *widgets*, temas e ícones, enquanto os usuários do iOS têm menos opções de personalização.
- Fragmentação do sistema: Android é um sistema operacional de código aberto e está disponível em vários dispositivos de diferentes fabricantes, o que leva à fragmentação do sistema. Isso significa que há muitas variações na versão do sistema operacional Android em diferentes dispositivos, o que pode tornar difícil para

os desenvolvedores criarem aplicativos que funcionem perfeitamente em todos os dispositivos Android. Por outro lado, o iOS é exclusivo da Apple e está disponível apenas em seus dispositivos, o que torna mais fácil para os desenvolvedores criar aplicativos que funcionem em todos os dispositivos iOS.

- Lojas de aplicativos: A loja de aplicativos do iOS é conhecida por sua alta qualidade e rigorosos critérios de seleção de aplicativos. A App Store da Apple é conhecida por ser mais segura e confiável em relação aos aplicativos que estão disponíveis para download. Em contrapartida, a loja de aplicativos do Android, a Google Play Store, é conhecida por ter um processo de seleção menos rigoroso e permitir mais aplicativos de terceiros em comparação com a App Store.
- Linguagens de programação: O iOS é geralmente programado em Objective-C ou Swift, enquanto o Android é geralmente programado em Java [4] ou Kotlin. Swift é considerada uma linguagem de programação mais fácil de aprender do que Objective-C, o que torna mais fácil para os desenvolvedores iniciantes criar aplicativos para iOS. No entanto, Kotlin tem ganhado popularidade como uma linguagem de programação mais moderna e fácil de usar para Android.
- Integração com outros dispositivos: A Apple é conhecida por integrar perfeitamente seus dispositivos em seu ecossistema. Por exemplo, é possível usar o AirDrop para compartilhar arquivos entre dispositivos Apple ou receber chamadas e mensagens em seu Mac a partir de um iPhone. No entanto, a integração do Android com outros dispositivos pode ser menos fluida.

Em resumo, ambas as plataformas têm suas próprias vantagens e desvantagens e a escolha entre Android e iOS depende das necessidades e preferências do usuário.

F. Back-end e Front-End

Back-end e *front-end* são termos utilizados em desenvolvimento de software para se referir às duas principais partes de uma aplicação [3]. O *back-end*, ou "parte de trás", é a parte responsável pela lógica da aplicação, processamento de dados e comunicação com bancos de dados e outros sistemas. Já o *front-end*, ou "parte da frente", é a interface com a qual o usuário interage, como a interface gráfica, elementos visuais e de interação.

De maneira geral, o *back-end* é responsável por tudo que acontece "atrás das câmeras", ou seja, o processamento de dados e a realização de operações que o usuário não vê diretamente, mas que são essenciais para o funcionamento da aplicação. O *front-end*, por outro lado, é responsável por tudo que o usuário vê e interage, incluindo a interface gráfica, botões, menus, caixas de diálogo e outros elementos de interação.

Para construir uma aplicação web completa, é necessário trabalhar tanto no *back-end* quanto no *front-end*, criando uma integração entre as duas partes. O *back-end* geralmente é desenvolvido utilizando linguagens de programação como Java[4], PHP, Python ou Ruby, e frameworks como Node.js, Django ou Ruby on Rails. Já o *front-end* é desenvolvido utilizando linguagens de marcação como HTML[6], CSS[7] e JavaScript, e frameworks como React[8], Vue.js ou Angular.

G. Java e Servidor

Java é uma linguagem de programação orientada a objetos criada pela Sun Microsystems em 1995 e atualmente mantida pela Oracle Corporation[4]. É uma das linguagens mais populares do mundo, utilizada para desenvolver uma ampla variedade de aplicações, desde aplicativos móveis até sistemas corporativos.

Um servidor é um computador ou sistema de computação que é responsável por fornecer serviços ou recursos a outros dispositivos ou computadores em uma rede[5]. Em termos de programação, o servidor geralmente se refere a um programa ou software que é executado em um computador servidor e é responsável por fornecer recursos ou serviços para outras aplicações ou dispositivos. Os servidores podem ser usados para hospedar sites, fornecer serviços de e-mail, gerenciar bancos de dados e muito mais. Eles geralmente são gerenciados por administradores de sistema e podem ser acessados por meio de redes locais ou pela Internet.

H. HTML, CSS e React

HTML (*Hypertext Markup Language*) é uma linguagem de marcação usada para criar páginas web[6]. Ele define a estrutura básica e os elementos de uma página web, como títulos, parágrafos, imagens, links, formulários, tabelas e muito mais.

CSS (*Cascading Style Sheets*) é uma linguagem usada para estilizar a aparência de uma página web[7]. Ele define o layout, as cores, as fontes e outros aspectos visuais de uma página, permitindo que os desenvolvedores criem páginas web visualmente atraentes e bem projetadas.

React é uma biblioteca de JavaScript usada para criar interfaces de usuário (UI) interativas e escaláveis[8]. Ele é mantido pelo Facebook e é amplamente utilizado para construir aplicativos web e móveis modernos. React permite que os desenvolvedores criem componentes reutilizáveis que podem ser combinados para criar aplicativos mais complexos.

Um componente React é uma peça autônoma e reutilizável de código que encapsula a lógica e a apresentação de uma parte da interface do usuário. Cada componente tem seu próprio estado e propriedades que podem ser atualizados para alterar a aparência ou o comportamento do componente.

I. SCRUM

Scrum é uma metodologia ágil de gestão de projetos que tem como objetivo aumentar a eficiência e a produtividade das equipes de desenvolvimento de software. Foi criada por

Ken Schwaber e Jeff Sutherland, no início da década de 1990, e vem sendo amplamente utilizada em empresas de tecnologia[12].

A metodologia Scrum é baseada em um ciclo iterativo e incremental, chamado de Sprint, que dura de duas a quatro semanas. Durante cada Sprint, a equipe se concentra em um conjunto específico de tarefas e trabalha para concluir essas tarefas até o final do Sprint.

Antes de começar cada Sprint, a equipe se reúne para realizar o Sprint Planning, no qual definem o *backlog* do Sprint, ou seja, as tarefas que serão realizadas. O *backlog* do Sprint é uma lista ordenada de tarefas que devem ser concluídas para que o Sprint seja considerado um sucesso.

Durante o Sprint, a equipe se reúne diariamente para realizar o Daily Scrum. O objetivo desta reunião é discutir o progresso feito até o momento, definir as tarefas que serão realizadas naquele dia e identificar possíveis obstáculos que possam impedir o avanço do trabalho.

Ao final de cada Sprint, a equipe realiza o Sprint Review, no qual apresenta o trabalho concluído e recebe feedbacks dos stakeholders. Após o Sprint Review, a equipe se reúne para realizar o Sprint Retrospective, onde discutem o que funcionou bem e o que poderia ser melhorado para a próxima Sprint.

O Scrum é uma metodologia flexível e adaptável, e permite que a equipe faça ajustes durante o projeto para se adaptar às mudanças no ambiente e nos requisitos do projeto. É uma metodologia eficiente para projetos que exigem um alto nível de colaboração e comunicação entre a equipe e os stakeholders, e que requerem entregas frequentes e iterativas.

J. Unifield Modeling Language - UML

A UML é uma linguagem visual utilizada para representar e modelar sistemas de software. A UML oferece um conjunto de diagramas e notações que permitem descrever a estrutura, o comportamento e as interações entre os componentes de um sistema. Esses diagramas incluem diagramas de classes, diagramas de sequência, diagramas de atividades[13], diagramas de casos de uso[14], entre outros. A UML é amplamente utilizada na engenharia de software como uma ferramenta para a análise, projeto e documentação de sistemas[15].

1) Diagrama Caso de uso: O diagrama de caso de uso é uma ferramenta de modelagem utilizada em engenharia de software para representar as interações entre um sistema e seus atores externos, mostrando como os usuários interagem com o sistema para realizar tarefas específicas[14].

Os casos de uso descrevem as funcionalidades e ações que o sistema pode executar para atender às necessidades do usuário. Cada caso de uso é representado no diagrama como uma elipse, com o nome do caso de uso dentro dela. Os atores externos do sistema são representados como caixas, com seus nomes escritos dentro.

2) Diagrama de Atividade: O diagrama de atividades é uma ferramenta de modelagem utilizada em engenharia de software para representar o fluxo de atividades de um processo ou de um caso de uso. Ele é usado para ilustrar a sequência de atividades que ocorrem dentro do sistema e como elas se relacionam umas com as outras[13].

O diagrama de atividades é usado para ilustrar o fluxo de trabalho de um processo de negócios, a lógica do comportamento do sistema ou as interações entre atores em um caso de uso. Ele é uma ferramenta útil para entender e comunicar os processos e fluxos de trabalho de um sistema, permitindo uma melhor compreensão dos requisitos e das interações do usuário com o sistema.

III. TRABALHOS CORRELATOS

Nesta seção, são descritos alguns trabalhos correlatos como Bling[16], Omie[17] e Nibo[18].

A. Bling

Bling é um sistema de gestão empresarial (ERP[9]) que oferece diversas funcionalidades para gerenciar as principais áreas de uma empresa, como financeiro, estoque, vendas, compras, produção, entre outras[16]. Essas são apenas algumas das principais funcionalidades do sistema Bling, que possui diversas outras ferramentas e recursos para gerenciar uma empresa de forma eficiente e simplificada.

O aplicativo do sistema Bling está disponível apenas para dispositivos Android e foi desenvolvido com o objetivo de oferecer facilidade de uso ao usuário. Ele disponibiliza diversas funcionalidades, como a emissão de notas fiscais eletrônicas de forma prática, o controle de estoque, a gestão de produtos e pedidos de compras, além da possibilidade de realizar cadastros de produtos, clientes, fornecedores e vendedores, imprimir relatórios detalhados e ter controle completo do negócio. É possível também realizar conferências de estoque por meio da câmera do celular como leitor de código de barras, comparando a quantidade dos produtos lidos com a quantidade do estoque atual e atualizando o saldo em estoque dos produtos com a quantidade conferida. O Bling também oferece a opção de gerar propostas comerciais, boletos e relatórios de vendas agrupados por clientes, produtos ou vendedores, e permite a organização das finanças do negócio, gerenciando pagamentos, criando relatórios de fluxo de caixa e balancete de entrada e saída por período de relatório de seus recebimentos. O Bling não possui um módulo offline, sendo necessário o acesso à conta somente através de um navegador com internet ativa.

B. Omie

O Omie é um sistema de gestão empresarial baseado em nuvem, fácil de usar e personalizável. Compatível com dispositivos móveis e com recursos avançados de segurança, o Omie é uma solução completa de gestão empresarial para empresas de todos os tamanhos e setores[17]. Além dessas

funcionalidades, o Omie também oferece uma variedade de recursos adicionais, como integração com outras ferramentas e plataformas, relatórios personalizados, suporte técnico e mais.

O aplicativo do sistema Omie está disponível tanto para Android[2] quanto para iOS[2] e permite o acesso remoto às funcionalidades do sistema de gestão Omie. Não há custo de instalação ou taxa de uso, mas é importante ressaltar que o sistema só funciona de forma online, tanto na versão web quanto no aplicativo móvel. O app Omie permite aprovar pagamentos e pedidos de compras, bem como visualizar resumos de cada módulo do ERP[9], embora nem todas as funcionalidades estejam disponíveis no aplicativo. O objetivo principal é facilitar a vida do empreendedor, permitindo que ele aprove ou reprove pagamentos e pedidos pendentes de qualquer lugar. Além disso, os *dashboards* fornecem acesso aos principais indicadores e informações que são necessários para monitorar o desempenho da empresa e atingir as metas. Vale lembrar que ele não substitui a plataforma, já que nem todas as funcionalidades estão disponíveis no aplicativo.

C. Nibo

O Nibo é um sistema de gestão financeira e contábil. O Nibo é fácil de usar, possui integração com outras ferramentas e plataformas, suporte técnico e acesso a uma rede de contadores para ajudar as empresas a gerenciarem suas finanças e contabilidade de forma eficiente[18]. O sistema Nibo não possui aplicativo para smartphones, apenas a versão Web do sistema, também não possui funcionalidades *offline*, apenas funcionando com conexão ativa de internet.

D. Considerações sobre os Trabalhos Correlatos

Os sistemas citados são todos pagos, com valores agradáveis para as funcionalidades oferecidas. O diferencial deste trabalho será um sistema simplificado, focado no que é necessário para o dia a dia na demanda das vendas ambulantes, já que os sistemas citados não possuem uma interface objetiva para a qual este sistema foi elaborado.

IV. Metodologia

O sistema foi desenvolvido em Java[4] para a parte de *back-end*, Flutter[1] para a parte *mobile* e React [8] para o *front-end*[3]. O desenvolvimento do sistema foi amparado pela metodologia SCRUM[12], que permite e incentiva a realização de encontros semanais para divulgações dos resultados obtidos pela pesquisa no decorrer da semana. Bem como deverão ser realizadas entregas mensais que agregam valor evidenciando o andamento do projeto. No início do projeto foi realizado o levantamento dos requisitos e estudo acerca da demanda do trabalho para atingir o seu objetivo, na segunda parte, foi então desenvolvido o sistema.

Para uma melhor compreensão do sistema foi utilizada uma representação visual das informações, dados ou conceitos. Esta representação se dará por meio da utilização de símbolos, formas, linhas, cores e outros elementos gráficos, como o Diagrama de Casos de Uso e o Diagrama de Atividades.

A. Diagrama de Caso de Uso

Para a ilustração da interação do usuário com o sistema e a representação das funcionalidades foi elaborado o Diagrama de Caso de uso, apresentado na Figura 1. Pode-se observar que o ator Vendedor, poderá interagir com o sistema realizando vendas, consultas, aprazando pagamentos e realizando as baixas dos mesmos quando pagos. O Ator Gerente herda as atividades do Vendedor, porém, também tem a capacidade de tornar inativos clientes que permaneçam por um período pré determinado, sem realizar compras.

1) Os casos de uso:

- Cadastrar cliente - É realizado o cadastro do cliente com as principais informações como CPF, RG, dados de endereço, data de pagamento.
- Registrar venda - É realizado o registro das mercadorias vendidas, valor da venda, data de pagamento.
- Registrar pagamento - É realizado o registro do pagamento referente a venda, sendo no ato da venda ou posteriormente conforme pagamento semanal, quinzenal, mensal.
- Registrar data próximo pagamento - Conforme sinalizado no registro do pagamento, realizar agendamento da data do próximo pagamento.
- Consultar pagamentos por data - É realizada a consulta de clientes com pagamento agendado para a data consultada.
- Consultar clientes - É realizado a consulta no sistema por cliente, sendo realizada por nome ou cpf.
- Desativar clientes - Clientes que não efetuem compras há mais de 6 meses poderão ser desativados do sistema, assim, não sendo retornados em buscas utilizadas no dia a dia.

B. Diagrama de Atividade

Pode-se verificar na Figura 2 o diagrama de atividade,em que são apresentadas as atividades realizadas no momento de uma venda. Começando pelo registro da mesma, que pode ou não ser acompanhado pelo cadastro do cliente, caso ainda não esteja no sistema. Após o cadastro do cliente e registro da venda, são definidas as condições que o pagamento será efetuado, como número de parcelas e datas de vencimento de cada uma. Então, encerra-se a venda com todos os processos registradas no sistema.

V. Desenvolvimento

A fase de desenvolvimento foi concebida com o propósito de adquirir novos conhecimentos tecnológicos e atender a uma necessidade identificada no cotidiano dos profissionais que realizam vendas ambulantes. Muitos desses profissionais

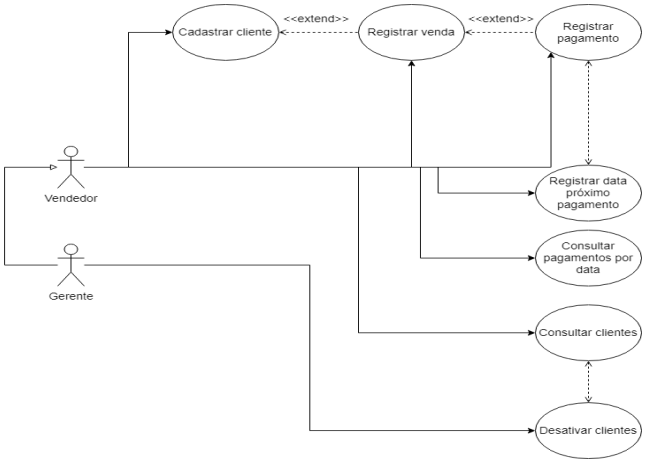

Figura 1. Diagrama de caso de uso

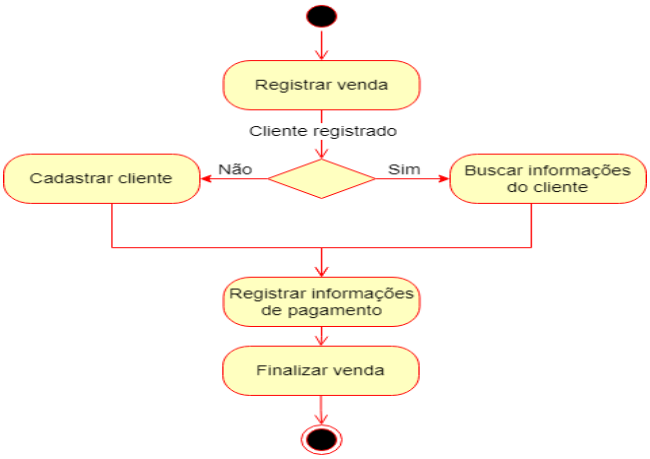

Figura 2. Diagrama de Atividade

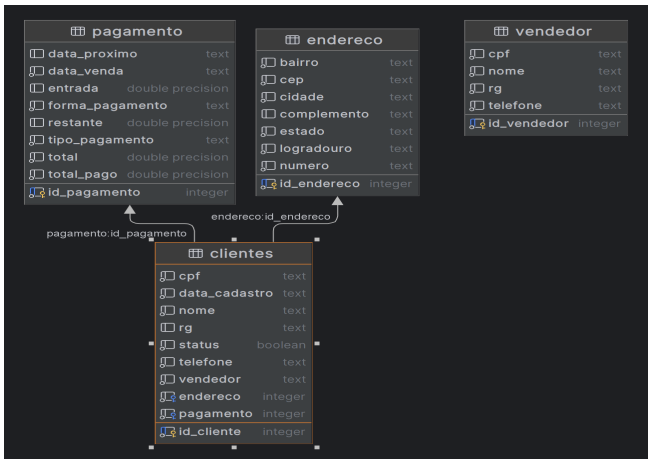

Figura 3. Diagrama de Tabelas

```
public Cliente createCliente(Cliente cliente) {
    if(clienteRepository.existCliente(cliente.getCpf())) {
        return new Cliente();
    }
    cliente.setStatus(true);
    cliente.setDataCadastro(getDateTime());
    enderecoService.createEndereco(cliente.getEndereco());
    pagamentoService.createPagamento(cliente.getPagamento());
    return clienteRepository.save(cliente);
}

1 usage  Ruan
public Integer updateCliente(Integer id, Cliente cliente) {
    return clienteRepository.updateCliente(cliente.getNome(), cliente.getCpf(), cliente.getRg(), cliente.getTelefone(), id);
}

4 usages  Ruan
public Integer updateStatusCliente(Integer id, Boolean status) {
    return clienteRepository.updateStatus(status, id);
}

1 usage  Ruan
public List<Cliente> findByData(String dataProximo) {
    return clienteRepository.findByData(formatDate(dataProximo));
}

1 usage  Ruan
public List<Cliente> findByCpf(String cpf) { return clienteRepository.findByCpf(cpf); }
```

Figura 4. Código Back-end

ainda utilizam registros em papel, e a meta é modernizar e tornar mais tecnológica essa forma de trabalho.

A. *Desenvolvimento Back-end*

O sistema foi desenvolvido na forma de uma API REST, com o propósito de executar operações CRUD (Create, Read, Update, Delete) do sistema. Ele age como uma ponte entre as interfaces do FrontEnd e do Mobile, conectando-se ao banco de dados.

Para o desenvolvimento, foi utilizada a linguagem Java, na versão 17, e teve-se o suporte de diversos frameworks e plugins, que facilitaram o processo. Alguns deles incluem:

- Spring Boot [19]
- Spring Data JPA [20]
- Spring Web [21]
- Spring Boot DevTools [22]
- PostgreSQL [23]
- Lombok [24]

Na Figura 3, é possível visualizar o diagrama das tabelas representadas. As tabelas contemplam o armazenamento dos dados dos Vendedores, Clientes, Endereços e Pagamentos.

Na Figura 4 pode-se ver parte da implementação da service de cliente. Implementa métodos para recuperar todos os clientes, obter cliente por ID, criar cliente com validação de CPF único, atualizar dados e status, buscar por data e CPF.

B. *Desenvolvimento Front-end*

O sistema foi desenvolvido utilizando React[8], baseado em TypeScript[25], e empregando um template Bootstrap[26] para criar uma interface de usuário mais agradável. No sistema, foi elaborado um formulário de cadastro de clientes que abrange todos os campos necessários para um registro completo como pode-se ver na Figura 6. Por outro lado, o formulário de cadastro de vendedores possui os dados essenciais para o controle de ambulantes, uma vez que serve para o controle dos funcionários.

Foi criada uma página de listagem de clientes que exibe somente os clientes ativos no banco de dados. Todos os

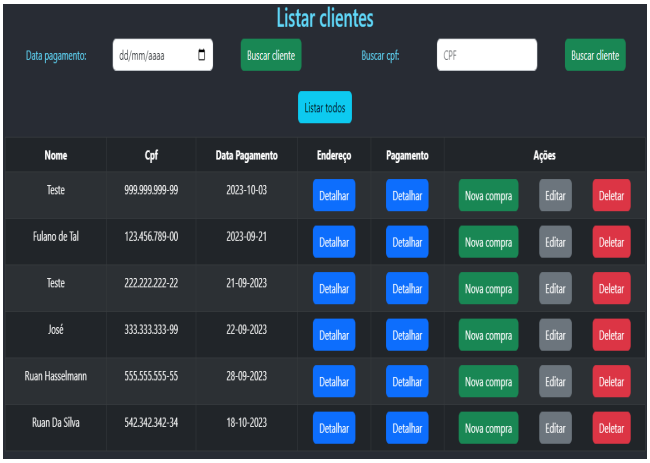

Figura 5. Lista de clientes no Front-end

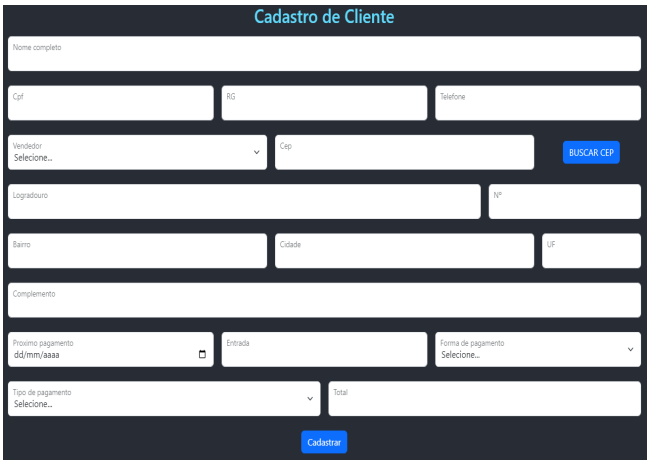

Figura 6. Formulário de cadastro

```
return (
  <div class="container">
    <h2>Listar clientes</h2>
    <div class="row gy-2 gx-3 align-items-center">
      <div class="col-2 form-floating-sm mb-3">
        <label for="floatingInput" className='labelbusca'>Data pagamento:</label>
      </div>
      <div class="col-2 form-floating-sm mb-3">
        <Form.Control type='date' class="form-control" id="autoSizingInput floatingInput"
          name="dataProximo"
          value={pagamento.dataProximo}
          onChange={handleChangePagamento} />
      </div>
      <div class="col-2 form-floating mb-3">
        <button type='button' class="btn btn-success" onClick={() => handlePag()}>Buscar cliente</button>
        <ModalFailSearch isOpen={isModalFailSearchOpen} onClose={() => setIsModalFailSearchOpen(false)} />
      </div>
      <div class="col-2 form-floating-sm mb-3">
        <label for="floatingInput" className='labelbusca'>Buscar cpf:</label>
      </div>
      <div class="col-2 form-floating-sm mb-3">
        <Form.Control as={InputMask} mask="999.999.999-99" type="text" class="form-control" id="autoSizingInput floatingInput"
          name="cpf"
          value={clientes.cpf}
          onChange={handleChangeCpf}
          placeholder='CPF'
        />
      </div>
      <div class="col-2 form-floating mb-3">
        <button type='button' class="btn btn-success" onClick={() => handleCpf()}>Buscar cliente</button>
        <ModalFailSearch isOpen={isModalFailSearchOpen} onClose={() => setIsModalFailSearchOpen(false)} />
      </div>
      <div class="col-2 form-floating mb-3">
        <button type='button' class="btn btn-info" onClick={() => refresh()}>Listar todos</button>
      </div>
    </div>
    {loading && <h2>Carregando dados ...</h2>}
    {error && <h2>{error}</h2>}
```

Figura 7. Código para listagem de clientes

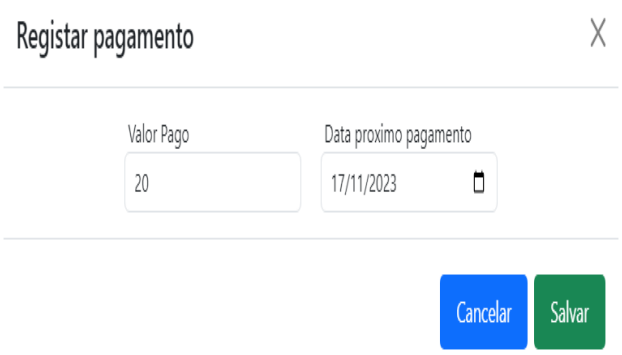

Figura 8. Interface de registro de pagamento

clientes possuem detalhes de endereço e informações de pagamento separadas na lista, evitando uma sobrecarga visual com muitas informações. É possível editar os detalhes do endereço caso um cliente mude de residência. Além disso, no detalhamento dos pagamentos, é possível registrar um pagamento vinculado ao cliente, especialmente se ele estiver em débito. Para completar, é possível editar e excluir clientes conforme necessário.

O sistema também permite a busca de um ou mais clientes por CPF ou pela data do próximo pagamento, exibindo uma lista dos clientes correspondentes. Nessa busca, o sistema retornará inclusive os clientes que porventura tenham sido excluídos, com a opção de ativá-los. Pode-se ver a lista de clientes com as funcionalidades da Figura 5.

Quanto aos vendedores, é possível cadastrar informações básicas sobre eles. Na lista de vendedores, é possível realizar edições e exclusões conforme necessário.

Pode-se ver na Figura 7 parte do código onde é realizada a busca dos clientes. Este trecho de código JSX representa um componente de interface de usuário para listar clientes. Ele inclui campos de busca por data de pagamento e CPF, botões de ação para buscar clientes, listar todos, e realizar diversas operações (detalhes, nova compra, edição, deleção, ativação/desativação). A interface também exibe uma tabela com detalhes dos clientes, e utiliza modais para mostrar informações adicionais e realizar edições. O código sugere o uso de tecnologias como React e Bootstrap.

Na Figura 8 pode-se ver a interface para o registro dos pagamentos para as vendas em aberto. Na Figura 9 pode-se ver a interface para que seja registrada uma nova compra para um cliente já cadastrado.

C. Desenvolvimento Mobile

O desenvolvimento da parte móvel do sistema foi conduzido utilizando o framework Flutter[1] e a linguagem Dart[27]. O objetivo principal da componente móvel é otimizar as operações diárias dos vendedores, possibilitando o cadastro de novos clientes, a edição de clientes existentes,

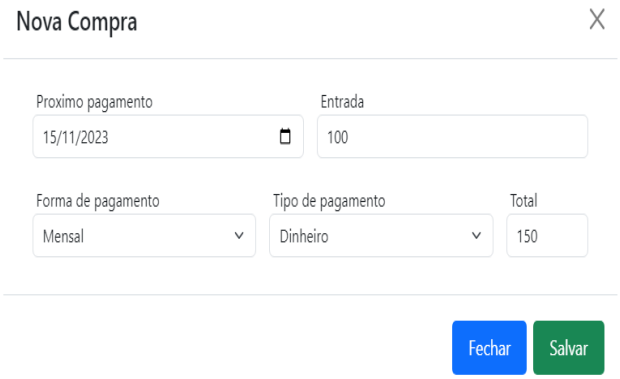

Figura 9. Interface de registro de nova compra

bem como o registro de pagamentos e novas vendas no sistema, promovendo maior mobilidade e eficiência operacional.

Para a criação da interface do usuário, foi adotada a biblioteca Material do Flutter[1], simplificando a elaboração de interfaces atrativas e coesas, em conformidade com as diretrizes de design do Material Design. Além disso, o Flutter é uma estrutura de desenvolvimento multiplataforma, permitindo a criação de aplicativos para Android, iOS, web e desktop, mantendo consistência na linguagem de design em todas as plataformas.

A Figura 10 apresenta a listagem dos clientes no aplicativo, exibindo apenas os dados essenciais para a identificação dos clientes. Na Figura 11, é possível identificar parte do código em Dart[27] responsável pela exibição da lista de clientes cadastrados.

A Figura 12 ilustra o código necessário para criar o formulário dos clientes, solicitando as informações indispensáveis para o cadastro. Já na Figura 13, é possível visualizar o formulário pronto para a inserção das informações a serem salvas.

VI. Conclusão

Ao final deste trabalho, percebe-se que os resultados obtidos poderão aprimorar significativamente a operação desse tipo de comércio, proporcionando maior autonomia e praticidade aos vendedores ambulantes. Isso permitirá um registro mais eficiente, mesmo em áreas onde o acesso à tecnologia e à internet pode ser precário. Essa abordagem visa evitar a degradação dos registros ao longo do tempo ou devido às condições de armazenamento. Portanto, visa melhorar a eficiência dos registros relacionados às vendas a prazo, otimizando os processos de pagamento e as datas de cobrança.

No decorrer deste projeto, deparamo-nos com desafios significativos, como a complexidade de lidar com novas

Figura 10. Listagem de clientes no aplicativo

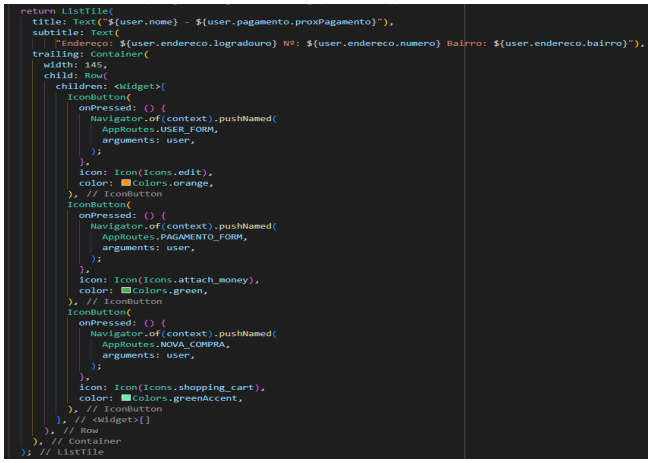

Figura 11. Código para listagem de clientes no aplicativo

linguagens e a restrição de tempo, uma vez que o sistema foi desenvolvido em três partes distintas: back-end, front-end e mobile. O desenvolvimento do front-end e do aplicativo móvel representou uma novidade, demandando a aquisição de novos conhecimentos ao longo do processo.

Durante o período de desenvolvimento, diversos problemas foram identificados em relação ao objetivo do trabalho. Em várias ocasiões, alguns pontos estavam em discordância com a ideia inicial, demandando uma pausa para análise e implementação de mudanças significativas visando otimizar o progresso do trabalho. Em razão do tempo e conhecimento limitados na área de front-end e mobile, foram necessários

```
body: Padding(
  padding: EdgeInsets.all(15),
  child: ListView(
    children: [
      Form(
        key: _form,
        child: Column(children: <Widget>[
          TextFormField(
            initialValue: _formData['nome'],
            decoration: InputDecoration(labelText: 'Nome'),
            validator: (value) {
              if (value == null || value.trim().isEmpty) {
                return 'Nome inválido';
              }
            },
            onSaved: (value) => _formData['nome'] = value!,
          ), // TextFormField
          TextFormField(
            initialValue: _formData['CPF'],
            decoration: InputDecoration(labelText: 'CPF'),
            validator: (value) {
              if (value == null || value.trim().isEmpty) {
                return 'CPF inválido';
              }
            },
            onSaved: (value) => _formData['CPF'] = value!,
          ), // TextFormField
          TextFormField(
            initialValue: _formData['RG'],
            decoration: InputDecoration(labelText: 'RG'),
            validator: (value) {
              if (value == null || value.trim().isEmpty) {
                return 'RG inválido';
              }
            },
            onSaved: (value) => _formData['RG'] = value!,
          ), // TextFormField
          TextFormField(
            initialValue: _formData['telefone'],
            decoration: InputDecoration(labelText: 'Telefone'),
            validator: (value) {
              if (value == null || value.trim().isEmpty) {
                return 'Telefone inválido';
              }
            },
            onSaved: (value) => _formData['telefone'] = value!,
          ), // TextFormField
```

Figura 12. Código para montagem do formulário de cadastro do cliente

Figura 13. Formulário de cadastro de cliente no aplicativo

cursos nas linguagens estabelecidas para o desenvolvimento, a fim de concluir o trabalho.

Devido a esses desafios, não foi possível finalizar todas as funcionalidades desejadas. O sistema, como um todo, foi subdividido em três projetos a serem realizados ao longo de um semestre. Com isso, o desenvolvimento foi direcionado para a entrega de um sistema funcional, contemplando as principais funcionalidades de maneira operacional.

Com o intuito de aprimorar a eficiência dos registros e eliminar a necessidade de papel, observamos que o objetivo foi alcançado. Agora, não é mais necessário carregar registros físicos ou canetas para realizar marcações. Basta um dispositivo celular capaz de utilizar tanto o aplicativo quanto a versão web no navegador do aparelho.

VII. Trabalhos futuros

Como trabalhos futuros o objetivo será atualizar os sistemas de forma a ficar mais eficiente e atendendo as demandas que virão a surgir com a entrada do sistema em produção e o início da utilização pelos usuários, ouvindo as demandas dos utilizadores.

Referências

[1] M. Alberto. “Flutter: O que é e tudo sobre o framework”. Em: (2023). Disponível: https://www.alura.com.br/artigos/flutter, (acessado em 2023-05-03).

[2] Shoptime. “IOS X Android: Quais as diferenças entre os sistemas operacionais”. Em: (2020). Disponível: https://www.techtudo.com.br/noticias/2020/12/ios-x-android-quais-as-diferencas-entre-os-sistemas-operacionais.ghtml, (acessado em 2023-04-20).

[3] Devmedia. “O que é front-end e back-end? Saiba a diferença!” Em: (2021). Disponível: https://www.devmedia.com.br/o-que-e-front-end-e-back-end/43216, (acessado em 2023-04-14).

[4] A. Bessa. “Java: O que é, linguagem e Guia para iniciar na tecnologia | Alura”. Em: (2023). Disponível: https://www.alura.com.br/artigos/java, (acessado em 2023-04-16).

[5] D. Kriger. “O que é um servidor, como ele funciona e principais vantagens”. Em: (2022). Disponível: https://kenzie.com.br/blog/servidor, (acessado em 2023-03-24).

[6] R. Marques. “O que é HTML? Entenda de forma descomplicada”. Em: (2019). Disponível: https://www.homehost.com.br/blog/tutoriais/o-que-e-html, (acessado em 2023-04-12).

[7] B. Kriger. “O que é CSS, como funciona, vantagens e onde aprender”. Em: (2022). Disponível: https://kenzie.com.br/blog/css, (acessado em 2023-05-04).

[8] U. Roveda. “React: O que é, como funciona e porque usar e como aprender”. Em: (2023). Disponível: https://kenzie.com.br/blog/react, (acessado em 2023-05-26).

[9] F. Ramos Braidotti. “Funcionalidades do ERP no padrão indústria 4.0”. Em: (2023). Disponível: https://doi.org/10.11606/D.3.2023.tde-04042023-081733 (acessado em 26/05/2023.)

[10] A. neto. “Marketplace: O que é, como funciona e como vender nessas vitrines virtuais”. Em: (2023). Disponível: https://www.escoladeecommerce.com/artigos/o-que-e-marketplace, (acessado em 2023-04-28).

[11] Redação Escola de E-commerce. “O que é E-commerce? Aprenda o conceito, funcionamento e como montar um”. Em: (2023). Disponível: https://www.escoladeecommerce.com/artigos/o-que-e-e-commerce, (acessado em 2023-05-24).

[12] C. Drumond. “Scrum — o que é, como funciona e por que é incrível”. Em: (). Disponível: https://www.atlassian.com/br/agile/scrum, (acessado em 2023-05-06).

[13] G. M. D. Gonçalves. “Diagramas de atividades - engenharia de software 31”. Em: (2010). Disponível: https://www.devmedia.com.br/diagramas-de-atividades-engenharia-de-software-31/18744, (acessado em 2023-04-23).

[14] L. Ribeiro. “Diagramas de caso de uso: O que é UML?” Em: (2012). Disponível: https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408, (acessado em 2023-04-13).

[15] P. Ventura. “O que é UML - o que é, para que serve, quando usar, e muito mais!” Em: (2019). Disponível: https://www.ateomomento.com.br/diagramas-uml, (acessado em 2023-06-02).

[16] Bling. “Bling ERP. Sistema ERP Online para pequenas empresas e lojas virtuais”. Em: (). Disponível: https://www.bling.com.br, (acessado em 2023-03-15).

[17] Omie. “Sistema de Gestão ERP Online - PMEs e grandes empresas”. Em: (). Disponível: https://www.omie.com.br, (acessado em 2023-03-16).

[18] Nibo. “Nibo - sistema de gestão online para empresários e contadores - nibo - controle financeiro e software de gestão empresaria”. Em: (). Disponível: https://www.nibo.com.br, (acessado em 2023-03-16).

[19] Spring Boot. “Spring Boot - Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run"”. Em: (). Disponível: https://spring.io/projects/spring-boot, (acessado em 2023-08-16).

[20] Spring Data JPA. “Spring Data JPA - Spring Data JPA, part of the larger Spring Data family, makes it easy to easily implement JPA based repositories.” Em: (). Disponível: https://spring.io/projects/spring-data-jpa, (acessado em 2023-08-16).

[21] Spring WEB. “Spring WEB - Spring Boot is well suited for web application development.” Em: (). Disponível: https://docs.spring.io/spring-boot/docs/current/reference/html/web.html, (acessado em 2023-08-16).

[22] DevTools. “Spring WEB - Spring Boot includes an additional set of tools that can make the application development experience a little more pleasant.” Em: (). Disponível: https://docs.spring.io/spring-boot/docs/1.5.16.RELEASE/reference/html/using-boot-devtools.html, (acessado em 2023-08-16).

[23] PostgreSQL. “PostgreSQL - PostgreSQL: The World’s Most Advanced Open Source Relational Database.” Em: (). Disponível: https://www.postgresql.org, (acessado em 2023-08-16).

[24] Lombok. “Lombok - Project Lombok is a java library that automatically plugs into your editor and build tools, spicing up your java.” Em: (). Disponível: https://projectlombok.org, (acessado em 2023-08-16).

[25] TypeScript. “TypeScript - is JavaScript with syntax for types.” Em: (). Disponível: https://www.typescriptlang.org, (acessado em 2023-08-16).

[26] Bootstrap. “Bootstrap - Build fast, responsive sites with Bootstrap.” Em: (). Disponível: https://getbootstrap.com, (acessado em 2023-08-16).

[27] Dart. “Dart - is a client-optimized language for fast apps on any platform.” Em: (). Disponível: https://dart.dev, (acessado em 2023-08-16).