CONCENTRAPAY: SISTEMA DE CENTRALIZAÇÃO DE PAGAMENTOS DIGITAIS

Mateus Rossi dos Santos, Rogério Xavier de Azambuja

[1] Instituto Federal do Rio Grande do Sul - Campus Farroupilha - Av. São Vicente, 785 CEP 95.174-274 - Bairro Cinquentenário - Farroupilha - RS - Brasil

mateus.rossisantos@gmail.com, rogerio.xavier@farroupilha.ifrs.edu.br

Abstract. *This article presents the development of a web system called ConcentraPay for splitting tab payments from bars and restaurants among multiple establishments. The goal is to optimize the payment process in multi-establishment environments, making it more efficient through the integration of web technologies to process physical tab payments. The developed application uses React, Java with Spring Boot, Firebase, and Firestore, leveraging the Efí Bank payment API. ConcentraPay allows customers to pay their tabs using a QR code, enabling secure and fast payments via Pix. The system proved to be efficient and feasible, allowing the distribution of amounts to multiple establishments from a single customer payment.*

Key-words: *Digital Payments, Pix, QR Code, Spring Boot, Firebase*

Resumo. *Este artigo apresenta o desenvolvimento de um sistema Web denominado ConcentraPay para a divisão do pagamento de comandas de bares e restaurantes entre diversos estabelecimentos. O objetivo é otimizar o processo de pagamento em empreendimentos multi-estabelecimentos, tornando-o mais eficiente, por meio da integração de tecnologias Web para efetivar o pagamento de comandas físicas. A aplicação desenvolvida utiliza React, Java com Spring Boot, FireBase e FireStore fazendo uso da API de pagamentos do Efí Bank. O ConcentraPay permite que os clientes efetuem o pagamento de suas comandas por meio de um código QR, realizando pagamentos de forma segura e rápida via Pix. O sistema desenvolvido mostrou-se eficiente e viável, permitindo a distribuição de valores a diversos estabelecimentos a partir de um único pagamento do cliente.*

Palavras-chave*: Pagamentos digitais, Pix, QR Code, Spring Boot, Firebase*

1. Introdução

Empreendimentos de complexos gastronômicos que reúnem diversos estabelecimentos, como restaurantes, bares e cafés, enfrentam desafios ao proporcionar uma experiência de pagamento prática e eficiente aos clientes. Em sistemas baseados em comandas, é comum a utilização de uma comanda única que deve ser quitada de forma fracionada em cada estabelecimento visitado.

Essa dinâmica gera desconforto ao cliente, que precisa realizar múltiplos pagamentos e, muitas vezes, enfrentar filas.

Diante desse cenário, este estudo propõe o desenvolvimento de uma aplicação web denominada ConcentraPay, projetada para acesso via dispositivos móveis e capaz de centralizar os pagamentos das comandas em uma única transação. A solução integra tecnologias modernas, como o Pix e a API (*Application Programming Interface*) de pagamentos do Efi Bank (Efi Bank, 2025) com o objetivo de assegurar operações seguras e eficientes. O objetivo deste trabalho é desenvolver uma solução tecnológica que permita ao cliente efetuar um único pagamento após a leitura do QR Code (*Quick Response Code* ou código para "resposta rápida") da comanda, reduzindo filas, otimizando o fluxo operacional dos estabelecimentos e aprimorando sua experiência de uso.

O público-alvo da solução é composto pelos frequentadores de complexos gastronômicos que utilizam sistemas de comandas e buscam uma experiência de pagamento mais simples, rápida e integrada. Esses usuários se beneficiam diretamente da centralização dos pagamentos, reduzindo o tempo gasto em filas e eliminando a necessidade de múltiplas transações ao final da visita.

Para alcançar isso, o estudo realizado analisou o cenário atual de pagamentos no Brasil, o domínio das tecnologias envolvidas, a integração de meios de pagamento modernos, a exploração do mercado de adquirentes, a documentação do processo, a avaliação do impacto e a apresentação dos resultados à comunidade acadêmica e aos empreendedores.

O restante deste artigo apresenta o referencial teórico, contendo os conceitos e trabalhos semelhantes à construção do ConcentraPay; os procedimentos metodológicos seguidos e as tecnologias adotadas no seu desenvolvimento são descritos detalhadamente por meio de sua modelagem e implementação, bem como os resultados obtidos e as considerações finais envolvendo a análise dos resultados e os possíveis trabalhos futuros. Por fim, as referências bibliográficas utilizadas são apresentadas.

2. Fundamentação Teórica

Esta seção apresenta uma breve descrição dos conceitos básicos para o entendimento deste trabalho, abordando os fundamentos essenciais, a situação do mercado atual de pagamentos no Brasil e descrevendo as tecnologias-chave que servirão como alicerces para o desenvolvimento da solução proposta neste estudo.

2.1. Sistemas de Pagamentos Digitais

Os pagamentos móveis são caracterizados como transações financeiras realizadas por meio de dispositivos móveis, utilizando redes de comunicação sem fio. Essa modalidade representa a convergência entre serviços bancários e recursos da tecnologia móvel, permitindo que o usuário efetue pagamentos diretamente pelo *smartphone*. (Braido, 2019).

No contexto brasileiro, observamos uma crescente adoção de pagamentos eletrônicos. Em 2017, as transações com cartões superaram o uso de dinheiro físico pela primeira vez na história, marcando uma mudança significativa nos padrões de pagamento (Braido & Klein, 2020). Entretanto, cerca de 45 milhões de brasileiros ainda não possuem uma conta bancária, o que indica a necessidade contínua de inovações para ampliar a inclusão financeira (Furini, 2020).

No campo das regulações do mercado, o Banco Central do Brasil (BACEN ou BCB)[1], a partir de 4 de novembro de 2013, abordou a prestação de serviços de pagamento no âmbito das disposições do Sistema de Pagamentos Brasileiro (SPB), estabelecendo diretrizes e padronizando os requisitos que os provedores de serviços precisam seguir.

O BACEN continuou suas ações, estabelecendo um grupo de trabalho com o objetivo de contribuir para a construção de um ecossistema de pagamentos instantâneos competitivo, eficiente, seguro e inclusivo. Neste contexto, cerca de 130 instituições, incluindo associações, instituições bancárias, proprietários de esquemas de pagamento, cooperativas de crédito, instituições governamentais, infraestruturas do mercado financeiro, fintechs, marketplaces, consultorias e escritórios de advocacia participaram das discussões (Braido & Klein, 2020).

A principal preocupação é que a regulamentação brasileira consiga equilibrar segurança e inovação, adotando como princípio a liberdade para inovar, sem exigir autorizações prévias que possam limitar o avanço tecnológico. Em setembro de 2019, foi lançada uma consulta pública sobre uma regulamentação preliminar que instituiu regras para o funcionamento de um ambiente de testes (*sandbox*) com autorizações temporárias para que empresas testem novos modelos de negócios no mercado de capitais brasileiro (Novaes & Hartmann, 2020).

Lançado em 2020 pelo Banco Central do Brasil, o Pix representa uma inovação nos pagamentos eletrônicos. Este sistema permite transferências instantâneas e virtuais de valores monetários, com destaque para sua velocidade, segurança e disponibilidade 24/7 (vinte e quatro horas por dia/sete dias por semana). Além disso, o Pix visa democratizar o acesso aos serviços financeiros, transformando gradualmente o dinheiro em papel em uma forma digital acessível a todos os usuários (Silva & Cruz, 2020).

Uma tecnologia essencial para a existência do Pix é a tecnologia do QR Code. O QR Code é um código de barras bidimensional que pode armazenar uma variedade de informações. Diferentemente dos códigos de barras tradicionais, o QR Code pode conter dados em formato de texto e em binário. Ele é amplamente utilizado devido à sua capacidade de armazenar informações em um espaço pequeno, à sua capacidade de correção de erros e à sua capacidade de ser lido em qualquer direção (Jesus, 2022).

Sobre o mercado de pagamentos via QR Code, vale constatar que:

> "O código QR é o novo cartão?" Pagamentos usando códigos QR dispensam cartões ou dinheiro, utilizando apenas um *smartphone*, por exemplo, e também

[1] autoridade monetária, reguladora e supervisora responsável por regulamentar instituições financeiras, dinheiro, crédito, pagamentos e câmbio

> podem ser associados a uma empresa que permite recargas por meio de boletos de pagamento, algo que afeta diretamente mais de 45 milhões de brasileiros (22% da população total) que não possuem ou utilizam contas bancárias. Em 2019, o estudo "Novos Meios de Pagamento" ouviu 500 pessoas de diferentes faixas etárias, rendas e regiões geográficas no Brasil, constatando que 17% dos consumidores estão utilizando códigos QR para pagamentos em lojas de varejo, enquanto a mesma quantidade era 0% em 2018. No entanto, dinheiro, cartão de crédito e cartão de débito ainda são os líderes de preferência para pagamentos em lojas de varejo: as porcentagens são 68%, 62% e 54%, respectivamente, enquanto pagamentos por aplicativos, como Samsung Pay e Apple Pay, representam 24%. Os usuários também afirmaram que códigos QR e aplicativos são os métodos de pagamento digital que mais gostariam de usar (17% e 20%, respectivamente), mas não o fazem porque a maioria dos vendedores comerciais não os aceita (Novaes & Hartmann, 2020).

Esses dados evidenciam uma evolução tecnológica significativa na área de meios de pagamento, impulsionada pela integração entre sistemas digitais e dispositivos móveis. Do ponto de vista da tecnologia da informação, o uso do QR Code no contexto do Pix demonstra a aplicação prática de conceitos como a interoperabilidade, a segurança da informação e a comunicação entre sistemas distribuídos. À medida que mais aplicações e APIs passam a oferecer suporte a esse formato de pagamento, observa-se um avanço no ecossistema de soluções financeiras digitais, consolidando o QR Code como uma tecnologia fundamental para a transformação digital do setor bancário e dos pagamentos digitais no Brasil.

2.2. Application Programming Interface (API)

Outra tecnologia essencial para a entrega da solução pretendida são as APIs, as quais são cruciais para a integração eficiente entre diferentes sistemas de software. Elas funcionam como uma ponte que permite que uma aplicação utilize os recursos ou dados de outra aplicação sem precisar conhecer os detalhes internos dessa aplicação. Em essência, uma API define as maneiras pelas quais os componentes de software devem interagir, especificando métodos de comunicação, formatos de dados e tipos de requisições e de respostas (Jorge, 2020).

As APIs tornaram-se ferramentas essenciais para o desenvolvimento de soluções modernas. Ao fornecer uma maneira eficiente e flexível de conectar sistemas de software, as APIs podem contribuir para aprimorar a eficiência, a produtividade e a inovação (Surwase, 2016).

As APIs podem ser usadas para uma variedade de propósitos, incluindo:

- Integração de sistemas: APIs podem ser usadas para conectar sistemas de software de diferentes fornecedores ou empresas. Isso permite que os dados e recursos sejam compartilhados entre os sistemas, o que pode melhorar a eficiência e a produtividade.
- Desenvolvimento de aplicativos: APIs podem ser usadas para simplificar o desenvolvimento de aplicativos. Ao fornecer acesso a recursos e dados existentes, as APIs podem reduzir a necessidade de desenvolver esses recursos internamente.

- Criação de novos serviços: APIs podem ser usadas para criar novos serviços baseados em dados e recursos existentes. Isso pode abrir novas oportunidades para empresas e organizações.

No contexto do trabalho proposto, uma API será desenvolvida para conectar o *back-end* na plataforma Firebase (Firebase, 2025) com a solução de pagamentos do Efi Bank. A adoção de APIs contribui para a padronização e a interoperabilidade entre sistemas, aspectos fundamentais para o sucesso de soluções distribuídas como a desenvolvida neste trabalho.

3. Desenvolvimento

O sistema desenvolvido consiste em uma aplicação Web com *layout* voltado para dispositivos móveis que tem o intuito de ser um facilitador de pagamentos de comandas de estabelecimentos comerciais, como bares e restaurantes. O site vai realizar a divisão do pagamento das comandas entre os estabelecimentos. O desenvolvimento utilizou o *framework* React (React, 2025), que utiliza a linguagem Typescript. Para o armazenamento dos dados será utilizada a plataforma Firebase e para o pagamento será utilizada a API Pix do Efi Bank, que permite o recebimento e principalmente o envio de transações Pix. A integração da API de pagamentos com o *front-end* será realizada por meio de um serviço desenvolvido com o *framework* Spring Boot, na linguagem Java (Java, 2025).

Nesta seção são descritas as tecnologias e ferramentas aplicadas no desenvolvimento do site, bem como suas funcionalidades.

3.1. Tecnologias

Durante a fase de produção, o React foi utilizado para o desenvolvimento da interface do usuário, devido à sua alta performance, reutilização de componentes e ampla adoção pela comunidade, o que garante maior produtividade e facilidade de manutenção ao longo do tempo. A plataforma Firebase será empregada como o sistema de autenticação e banco de dados, por ser robusta, escalável e facilmente integrada. Para a solução de pagamentos, a API do Efi Bank será utilizada, assegurando transações seguras e eficientes, bem como a possibilidade de envio de Pix da conta do titular para chaves Pix de fora do sistema do banco.

3.1.1. React

Para o desenvolvimento da interface com o usuário, foi adotado o uso do React, uma biblioteca JavaScript baseada em componentes, desenvolvida originalmente pela Meta (Meta, 2025). Sua principal proposta é permitir a criação de interfaces de usuário modulares e reutilizáveis, otimizando a performance por meio do uso do *virtual DOM* (VDOM), o que evita atualizações desnecessárias no DOM real e proporciona aplicações mais rápidas e responsivas. React também adota o conceito de fluxo de dados unidirecional, o que facilita o controle do estado da aplicação e a previsibilidade do comportamento da interface. Além disso, seu uso da linguagem JSX (Javascript XML-*Extensible Markup Language*) — uma extensão do JavaScript que permite a escrita de componentes com sintaxe semelhante ao HTML (*HyperText Markup Language*) —

simplifica a codificação e a associação de eventos à interface (Aggarwal et al., 2018). O React se mostra uma boa escolha para o desenvolvimento do *front-end* por possuir uma grande comunidade, o que implica em maior suporte, e por entregar uma integração facilitada com ferramentas como o Firebase.

3.1.2. Firebase

O Firebase, fornecido pelo Google (Google, 2025), é uma plataforma de desenvolvimento de aplicativos móveis e Web que oferece uma variedade de serviços prontos para uso. Desde bancos de dados em tempo real até autenticação de usuários e funções em nuvem, o Firebase simplifica tarefas complexas, permitindo que os desenvolvedores se concentrem na criação de experiências de usuário envolventes. Além disso, o Firebase oferece ferramentas robustas para análise de aplicativos, permitindo que os desenvolvedores compreendam o comportamento dos usuários e melhorem continuamente suas aplicações com base nos dados coletados (Khawas & Shah, 2018).

3.1.3. API Efi Bank

Para a integração com o sistema de pagamentos instantâneos, foi escolhida a API Pix do Efí Bank, uma solução robusta e certificada com nota A pelo Banco Central Brasileiro. Através desta API, é possível emitir cobranças imediatas (Pix Cob) ou com vencimento futuro (Pix CobV), gerar QR Codes dinâmicos e transações *Pix Cash-In* (recebimento automatizado) ou *Pisc Cash-Out* (envio de Pix diretamente via API). A integração exige a criação de credenciais, além de um certificado digital no formato .p12 para garantir segurança e conformidade com os padrões do Banco Central Brasileiro. A API ainda oferece suporte a diversos SDKs (*Software Development Kit*) tais como PHP, Node.js, Go, Java, .NET, entre outros, e um ambiente *sandbox* para testes, com mais de 6 bilhões de transações processadas apenas em 2024. Desta forma, uma opção eficiente, segura e escalável para projetos empresariais ou de *e-commerce*.

3.1.4. Spring Boot

Para o desenvolvimento da aplicação proposta, optou-se pela utilização do Spring Boot, uma extensão do *framework* Spring que tem como objetivo principal simplificar a configuração e a publicação de aplicações Java. O Spring Boot permite a criação de aplicações *stand-alone*, eliminando a necessidade de configurações manuais extensas, por meio de um conjunto de convenções predefinidas e de configurações automáticas baseadas nas dependências utilizadas. Essa abordagem possibilita maior agilidade no desenvolvimento e menor propensão a erros de configuração. O Spring Boot também se integra de forma eficiente com ferramentas de automação, como Maven e Gradle, fornecendo plugins específicos para empacotamento, execução e gerenciamento de dependências (Webb et al., 2018).

4. Visão Geral da Aplicação

A Figura 1 apresenta uma visão de alto nível do processo envolvido no uso da aplicação proposta neste projeto de software. O desenho abaixo demonstra o fluxo da solução de pagamentos empregada:

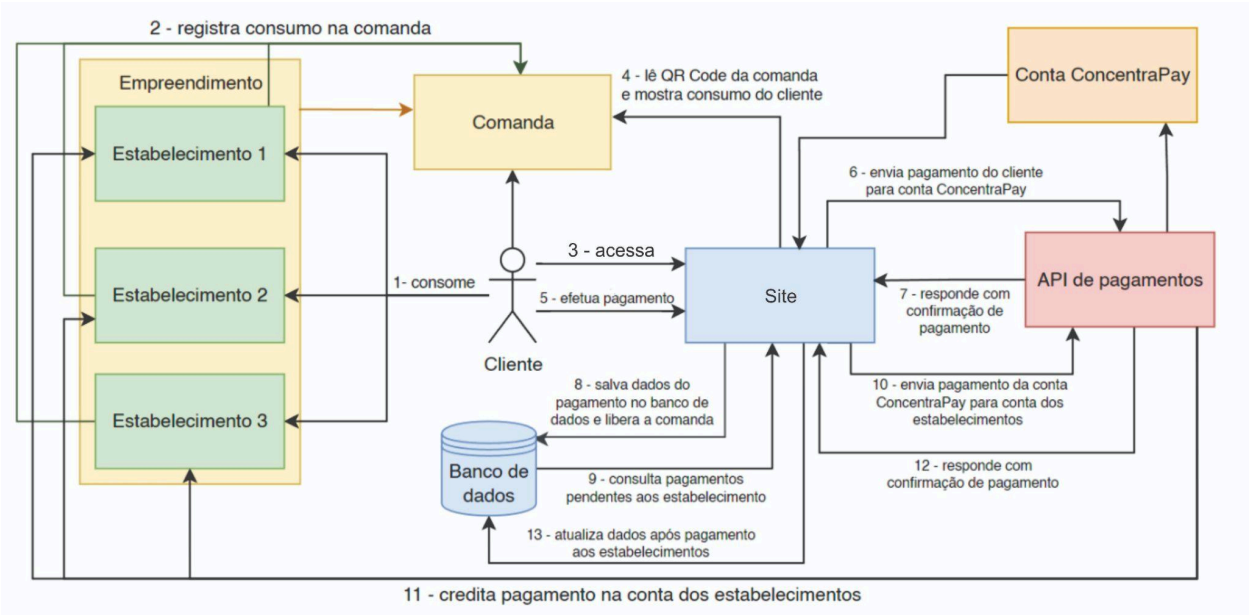


Figura 1. Fluxo da solução de pagamento

Fonte: Elaboração própria, 2023

A figura acima está dividida em 13 passos, explicados abaixo:

1 - O cliente, de posse de sua comanda física, realiza o consumo em três estabelecimentos diferentes dentro de um mesmo empreendimento.

2 - Os estabelecimentos registram os pedidos do cliente em suas comandas.

3 - O cliente acessa o ConcentraPay no seu *smartphone*.

4 - Após um breve cadastro, o cliente faz a leitura do QR Code da sua comanda e recebe no seu *smartphone* o valor de seu consumo.

5 - O cliente efetua um único pagamento no site, direcionado a uma conta de propriedade da ConcentraPay.

6 - O ConcentraPay envia o pagamento para a API de pagamentos pelo *back-end*.

7 - A API responde com a confirmação do pagamento e credita os valores na conta bancária do ConcentraPay.

8 - O sistema salva os dados no seu banco de dados (FireStore) com o detalhamento de qual valor é devido a qual estabelecimento do empreendimento. Logo após, o *status* da comanda é atualizado para liberar a saída do cliente.

A partir deste ponto, o cliente é liberado e a sua interação com o ConcentraPay termina. Porém, o sistema continua trabalhando na divisão dos pagamentos.

9 - Quando o usuário administrador do estabelecimento clicar no botão de Resgatar Pagamentos Pendentes no ConcentraPay, o sistema consulta seus pagamentos pendentes no banco de dados, confirmando que existem pagamentos pendentes, ele segue para o passo 10.

10 - O sistema envia a solicitação de pagamento à API de pagamentos, transferindo recursos da conta bancária do ConcentraPay para as contas de cada estabelecimento credor.

11 - A API de pagamentos credita os recursos nas contas dos estabelecimentos credores.

12 - A API de pagamentos responde ao site com a confirmação do pagamento.

13 - O ConcentraPay atualiza os dados de pagamentos pendentes no seu banco de dados.

4.1 Principais Funcionalidades

A Figura 2 ilustra o diagrama de casos de uso das principais funcionalidades do sistema. A seguir, uma breve descrição dos principais casos de uso.

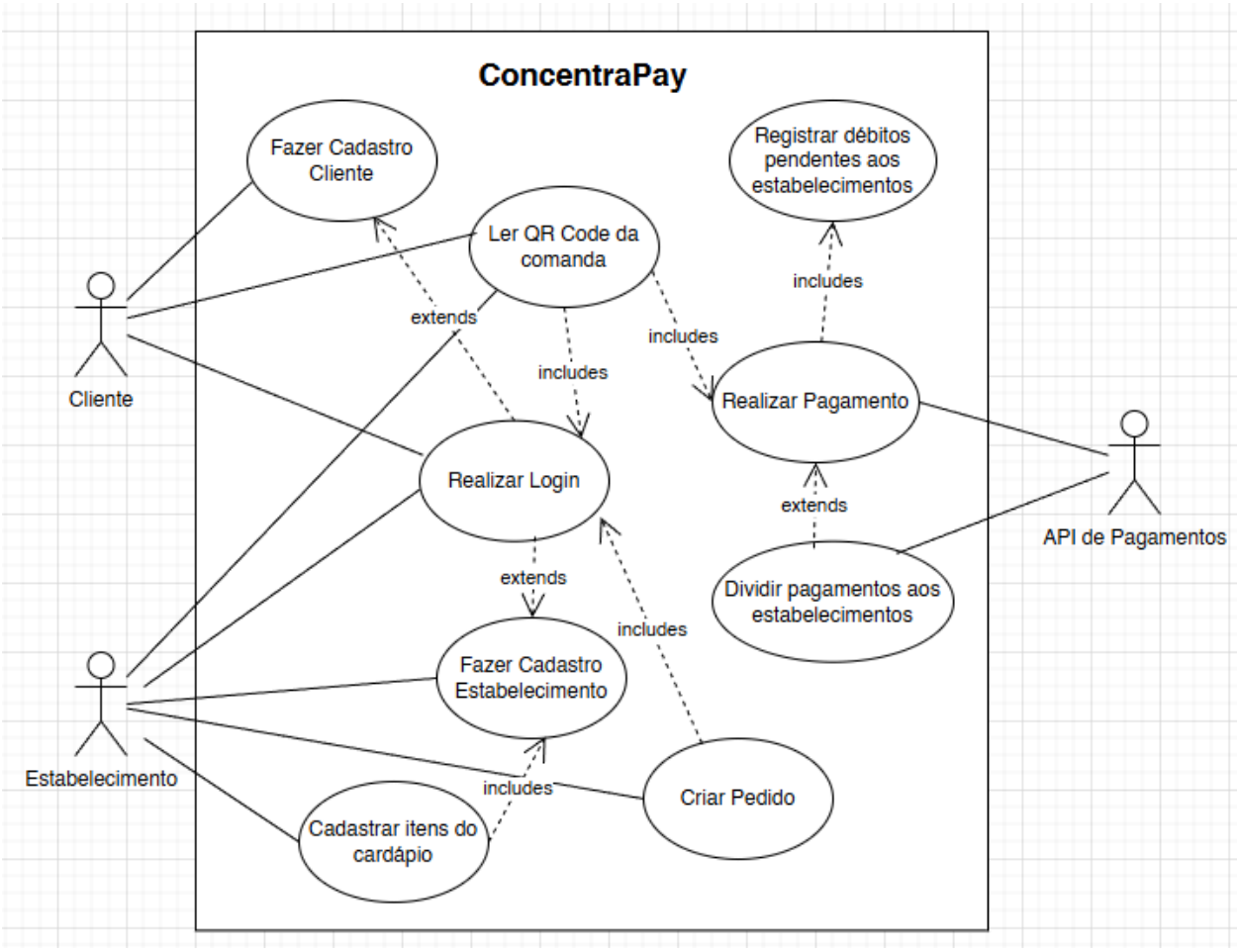


Figura 2. Casos de uso do software

Fonte: Elaboração própria, 2023

A figura acima apresenta os casos de uso principais do projeto desenvolvido. Foram identificados três atores principais no fluxo do sistema.

- O estabelecimento, que pode gerenciar os itens do cardápio e os pedidos, e também consultar a comanda para o usuário realizar o pagamento das comandas.
- O cliente, que pode visualizar seu consumo e realizar o pagamento das comandas.
- A API de pagamentos realiza a validação do pagamento da comanda e repassa os pagamentos aos estabelecimentos no momento oportuno.

Pode-se constatar que, mesmo que o foco do projeto seja no usuário final, a maior parte das interações são realizadas pelos estabelecimentos cadastrados. Abaixo são demonstrados através de uma tabela (Tabela 1), todos os casos de uso identificados:

Tabela 1. Casos de Uso

Nº	Como	Quero	Para
1	Cliente	Me cadastrar no site	Ter acesso ao sistema
2	Cliente	Logar no site com meu email e senha	Ter acesso à área logada e poder consultar minha comanda
3	Cliente	Ler o QR Code de minha comanda	Consultar meu consumo
4	Cliente	Realizar o pagamento de minha comanda	Quitar meu consumo e liberar minha saída do estabelecimento
5	Estabelecimento	Me cadastrar no site	Ter acesso ao sistema
6	Estabelecimento	Logar no site com meu email e senha	Ter acesso à área de gerenciamento das comandas do meu estabelecimento
7	Estabelecimento	Ler o QR Code da comanda do cliente	Consultar o consumo da comanda
8	Estabelecimento	Realizar o pagamento da comanda	Receber os valores e liberar a saída do cliente do estabelecimento
9	Estabelecimento	Abrir pedidos	Registrar o consumo do cliente
10	Estabelecimento	Cadastrar itens no cardápio	Dar opções de consumo ao cliente
14	ConcentraPay	Processar o pagamento da comanda	Quitar o consumo do cliente e liberar sua saída do estabelecimento.
15	ConcentraPay	Repassar os pagamentos divididos para os estabelecimentos corretos	Fazer com que os estabelecimentos recebam suas quantias devidas.

Fonte: Elaboração própria, 2023

Apenas com o desenvolvimento das funcionalidades acima já é possível validar a viabilidade deste projeto. A primeira tarefa foi a criação de um projeto React, seguida da criação de outro projeto na plataforma do Firebase. O Firebase oferece muitas soluções prontas para facilitar o processo de desenvolvimento, das quais utilizei:

- Authentication: Serviço que gerencia os usuários do lado do servidor, com autenticação e autorização. Utilizado no projeto para gerenciar tanto clientes quanto estabelecimentos
- Firestore Database: Armazenamento de dados em tempo real nos servidores do Firebase, no formato JSON (Javascript Object Notation), utilizado no projeto para salvar os dados de clientes, estabelecimentos e pagamentos.

Para a criação do banco de dados no Firestore Database, foram mapeados os dados necessários para cada coleção de dados. O resultado foi:

- Usuario:

```
{
    Id: String,
    email: String,
    isEc: boolean
}
```

- Estabelecimento:

```
{
    id: String,
    nome: String,
    email: String,
    cnpj: String,
    empreendimento: String,
    chavePix: String,
    pagamentoPendente: double,
    pagamentoAdiantado: double,
    endereco: String,
    cep: String,
    e2eIdUltimoPagamento: String
}
```

- Pedido:

```
{
    id: String,
    data: date,
    ec: String,
```

```
        numeroComanda: String,
        produtos: [
            {
                nome: String,
                quantidade: inteiro,
                precoUnitario: inteiro
            }
        ]
        valor: double,
        status: String,
        txId: String
}
```

- Produto:

```
{
        id: String,
        nome: String,
        descricao: String,
        valor: double,
        estabelecimento: String
}
```

4.2 Interfaces

No projeto React foram desenvolvidas 14 interfaces, São elas:

- interface de login de cliente. (Figura 3)
- interface de cadastro de cliente. (Figura 3)
- interface de cadastro de estabelecimento. (Figura 3)
- interface de recuperação de senha. (Figura 3)
- interface inicial de cliente logado com opção de leitura de QR Code.(Figura 4)
- interface inicial de estabelecimento logado, com opções de cadastro de itens, criação de pedido e resgate de pagamentos. (Figura 4)
- interface de criação de pedidos. (Figura 4)
- interface de cadastro de itens do cardápio. (Figura 4)
- interface de listagem de consumo. (Figura 5)
- interface para escolha do meio de pagamento. (Figura 5)
- interface de pagamento Pix com ‘QR Code Pix’ e ‘Pix copia e cola’. (Figura 5)
- interface de confirmação de pagamento. (Figura 5)
- interface de comanda liberada (Figura 6)

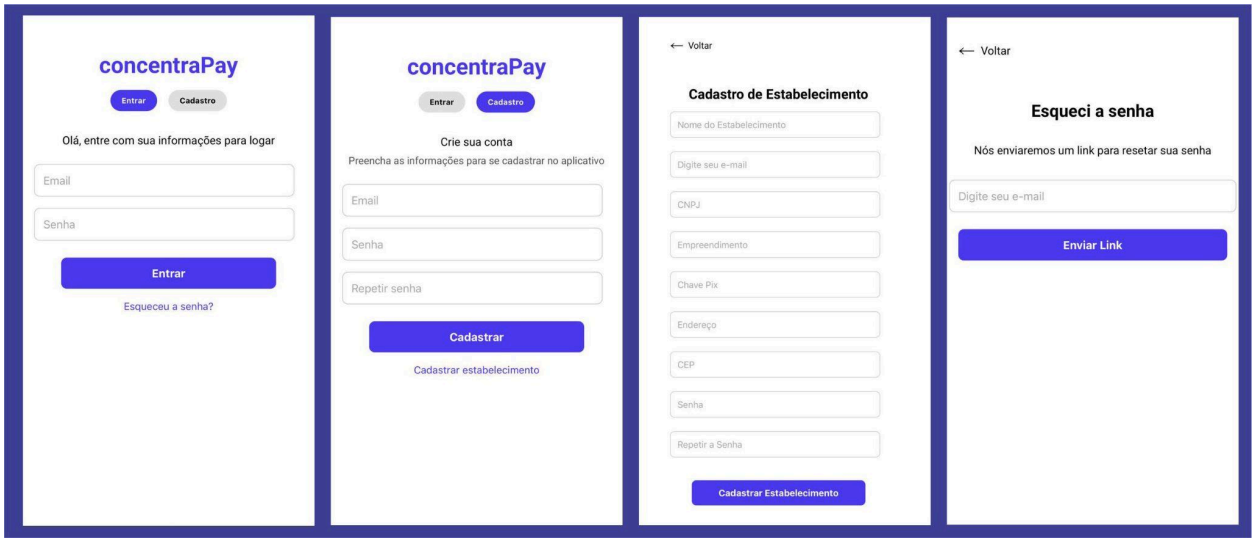


Figura 3. interface de login/cadastro de usuário e estabelecimento e recuperação de senha

Fonte: Elaboração própria, 2024

A interface de *login* se divide em duas abas. Na segunda aba, é possível fazer o cadastro caso o usuário não tenha cadastro ou ser direcionado ao cadastro de estabelecimento, que é a terceira interface mostrada na imagem acima. A quarta interface mostra a interface responsável por recuperar a senha do usuário em caso de esquecimento.

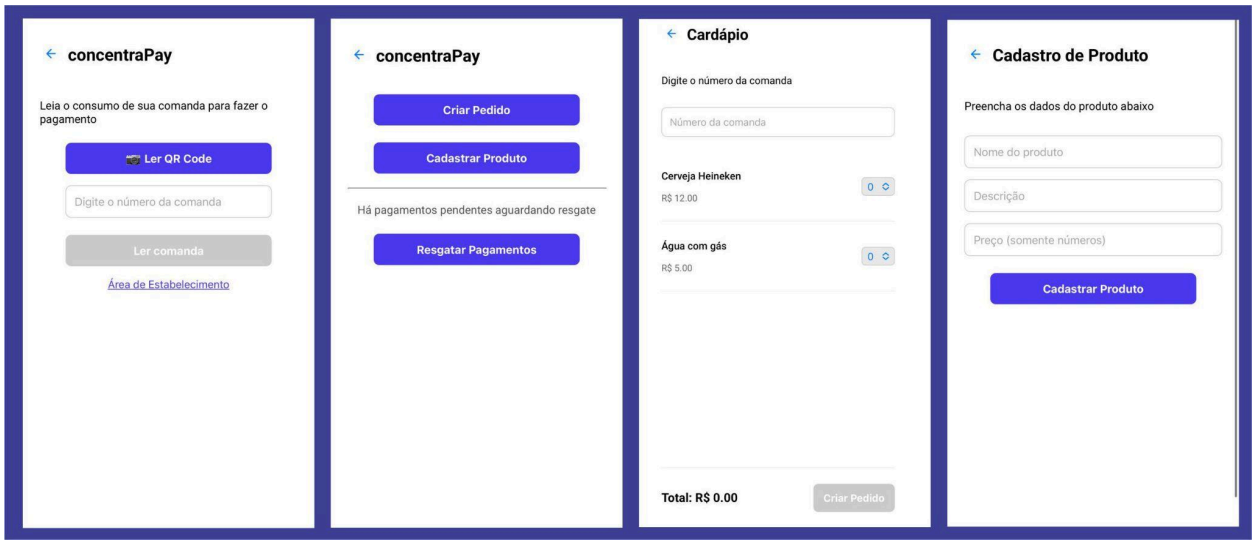


Figura 4. interface inicial após login, interface inicial de estabelecimento, interface de criação de pedido e interface de cadastro de produtos.

Fonte: Elaboração própria, 2024

A primeira interface acima é a interface inicial após o *login*, na qual o usuário poderá ler sua comanda. É feita a validação se o usuário logado é um estabelecimento. Caso seja, é mostrado o botão para se direcionar à área logada de estabelecimentos, onde são exibidas mais opções (segunda interface da imagem). Caso o estabelecimento tenha pagamentos pendentes a receber, o botão "Resgatar Pagamentos" é mostrado, para que o usuário receba valores que estejam pendentes. A terceira janela é para onde o usuário é direcionado ao clicar em "Criar Pedido", mostrando os itens do cardápio e permitindo a inserção do número da comanda para criar o pedido. A quarta interface mostra para onde o usuário é direcionado ao clicar em "Cadastrar Produto" na interface inicial do estabelecimento, permitindo o cadastro de novos produtos.

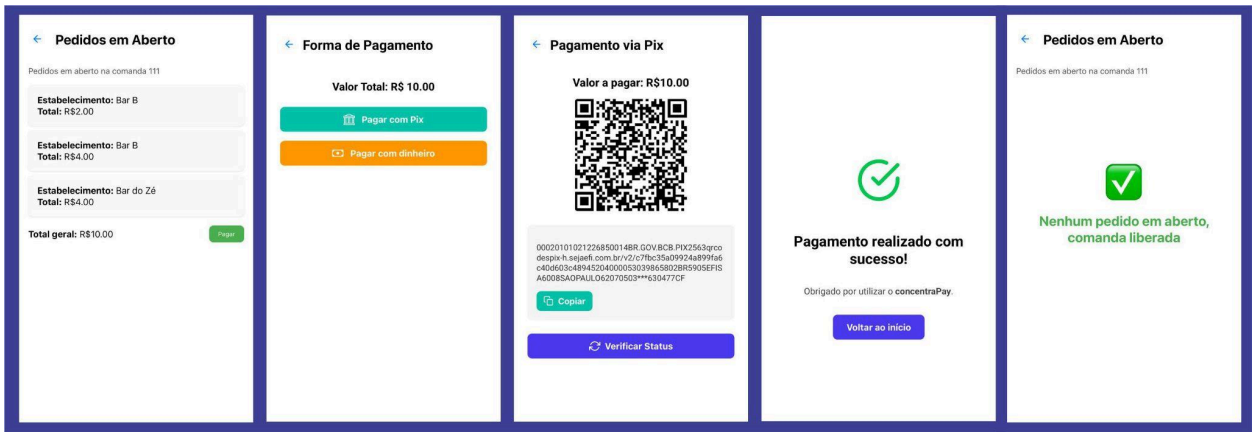


Figura 5. interface de pedidos abertos da comanda, formas de pagamento, interface de pagamento pix, interface de pagamento realizado com sucesso e interface de comanda liberada.

Fonte: Elaboração própria, 2024

Na figura 5, é mostrada, primeiramente, a interface de consumo da comanda lida na interface inicial (Figura 4). Ao clicar em "Pagar", a próxima interface exibe as opções de pagamento para o usuário. Mais uma vez, é feita uma validação se o usuário é um estabelecimento. Caso seja, mostrará a opção de pagamento com dinheiro; caso não, essa opção é ocultada, mostrando apenas a opção de pagamento com Pix. A terceira interface é a usada para o pagamento via Pix, com opção de pagamento via QR Code ou Pix Copia e Cola. Este pagamento é feito para uma conta do ConcentraPay. Com o pagamento concluído, é adicionado ao banco de dados um pagamento pendente do estabelecimento onde o usuário consumiu. No caso de pagamento em dinheiro, é adicionado ao banco um "pagamento adiantado" que é debitado dos Pix que ele possa ter pendentes. A quarta interface mostra a interface de "pagamento realizado com sucesso", exibida quando o Pix é confirmado. A quinta interface mostra uma nova leitura da comanda após o pagamento, indicando que a comanda está liberada, sem consumo pendente de pagamento.

4.3 Lógica

Abaixo seguem alguns trechos de código-fonte chave para o funcionamento do ConcentraPay.

4.3.1 - Método de criação de pedido:

```
const handleCreateOrder = async () => {
  const selectedItems = produtos
    .filter((produto) => quantities[produto.id] > 0)
    .map((produto) => ({
      nome: produto.nome,
      quantity: quantities[produto.id],
      unitPrice: produto.preco,
    }));

  if (selectedItems.length === 0 || !numeroComanda) {
    alert('Selecione pelo menos um item e preencha o número da comanda.');
    return;
  }

  try {
    await addDoc(collection(db, 'pedido'), {
      date: Timestamp.now(),
      ec: user.uid,
      valor: total,
      status: 'CREATED',
      numeroComanda,
      produtos: selectedItems,
    });

    navigate('/area-estabelecimento');
  } catch (error) {
    console.error('Erro ao criar pedido:', error);
  }
};
```

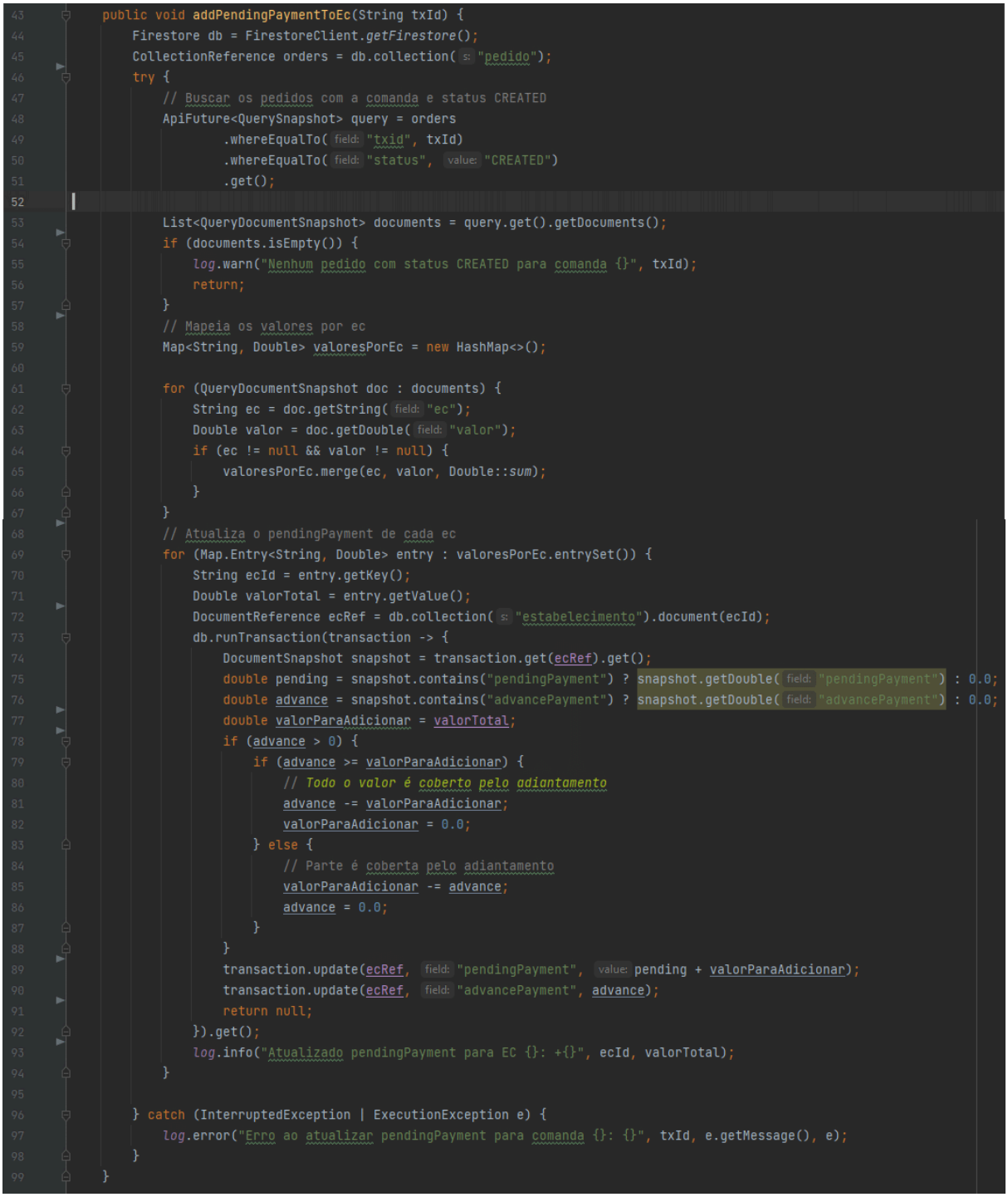
Figura 6. Método handleCreateOrder.

Fonte: Elaboração própria, 2025

A função *handleCreateOrder* cria um novo pedido no sistema com base nos produtos selecionados pelo usuário. Primeiro, ela verifica quais produtos têm quantidade maior que zero e monta uma lista com o nome, a quantidade e o preço de cada um. Em seguida, confere se há itens selecionados e se o número da comanda foi informado; caso contrário, exibe um alerta e interrompe o processo.

Se tudo estiver correto, a função registra o pedido no banco de dados do Firebase, salvando informações como a data, o identificador do estabelecimento, o valor total, o *status* “CREATED”, o número da comanda e os produtos selecionados. Por fim, no registro, o usuário é redirecionado para a área do estabelecimento. Se ocorrer algum erro, ele é exibido no console.

4.3.2 - Adição de pagamento pendente para o Estabelecimento:



```java
public void addPendingPaymentToEc(String txId) {
    Firestore db = FirestoreClient.getFirestore();
    CollectionReference orders = db.collection( s: "pedido");
    try {
        // Buscar os pedidos com a comanda e status CREATED
        ApiFuture<QuerySnapshot> query = orders
                .whereEqualTo( field: "txid", txId)
                .whereEqualTo( field: "status",  value: "CREATED")
                .get();

        List<QueryDocumentSnapshot> documents = query.get().getDocuments();
        if (documents.isEmpty()) {
            log.warn("Nenhum pedido com status CREATED para comanda {}", txId);
            return;
        }
        // Mapeia os valores por ec
        Map<String, Double> valoresPorEc = new HashMap<>();

        for (QueryDocumentSnapshot doc : documents) {
            String ec = doc.getString( field: "ec");
            Double valor = doc.getDouble( field: "valor");
            if (ec != null && valor != null) {
                valoresPorEc.merge(ec, valor, Double::sum);
            }
        }
        // Atualiza o pendingPayment de cada ec
        for (Map.Entry<String, Double> entry : valoresPorEc.entrySet()) {
            String ecId = entry.getKey();
            Double valorTotal = entry.getValue();
            DocumentReference ecRef = db.collection( s: "estabelecimento").document(ecId);
            db.runTransaction(transaction -> {
                DocumentSnapshot snapshot = transaction.get(ecRef).get();
                double pending = snapshot.contains("pendingPayment") ? snapshot.getDouble( field: "pendingPayment") : 0.0;
                double advance = snapshot.contains("advancePayment") ? snapshot.getDouble( field: "advancePayment") : 0.0;
                double valorParaAdicionar = valorTotal;
                if (advance > 0) {
                    if (advance >= valorParaAdicionar) {
                        // Todo o valor é coberto pelo adiantamento
                        advance -= valorParaAdicionar;
                        valorParaAdicionar = 0.0;
                    } else {
                        // Parte é coberta pelo adiantamento
                        valorParaAdicionar -= advance;
                        advance = 0.0;
                    }
                }
                transaction.update(ecRef,  field: "pendingPayment",  value: pending + valorParaAdicionar);
                transaction.update(ecRef,  field: "advancePayment", advance);
                return null;
            }).get();
            log.info("Atualizado pendingPayment para EC {}: +{}", ecId, valorTotal);
        }

    } catch (InterruptedException | ExecutionException e) {
        log.error("Erro ao atualizar pendingPayment para comanda {}: {}", txId, e.getMessage(), e);
    }
}
```

Figura 7. Método addPendingPaymentToEc.

Fonte: Elaboração própria, 2025

O método *addPendingPaymentToEc* tem como objetivo atualizar o campo pendingPayment (pagamento pendente) dos estabelecimentos que possuem pedidos com um determinado txId e status "CREATED". Ele também considera se o estabelecimento possui algum valor de adiantamento (*advancePayment*), que deve ser utilizado antes de adicionar novos valores pendentes.

Primeiro, o método recebe o identificador da transação (*txID*) e obtém uma instância do banco de dados do Firestore, acessando a coleção de pedidos chamada "pedido". Em seguida, ele consulta essa coleção, buscando todos os pedidos cujo campo txid corresponda ao valor recebido e cujo status seja "CREATED", ou seja, pedidos ainda não processados ou pagos.

Se nenhum pedido for encontrado, o método apenas registra um aviso no log e encerra a execução. Caso contrário, ele cria um mapa para agrupar os valores por estabelecimento. Para cada pedido retornado, o código obtém o identificador do estabelecimento (*ec*) e o valor do pedido, somando os valores dos pedidos que pertencem ao mesmo estabelecimento. Assim, ao final dessa etapa, o mapa contém o total de valores agrupados por EC.

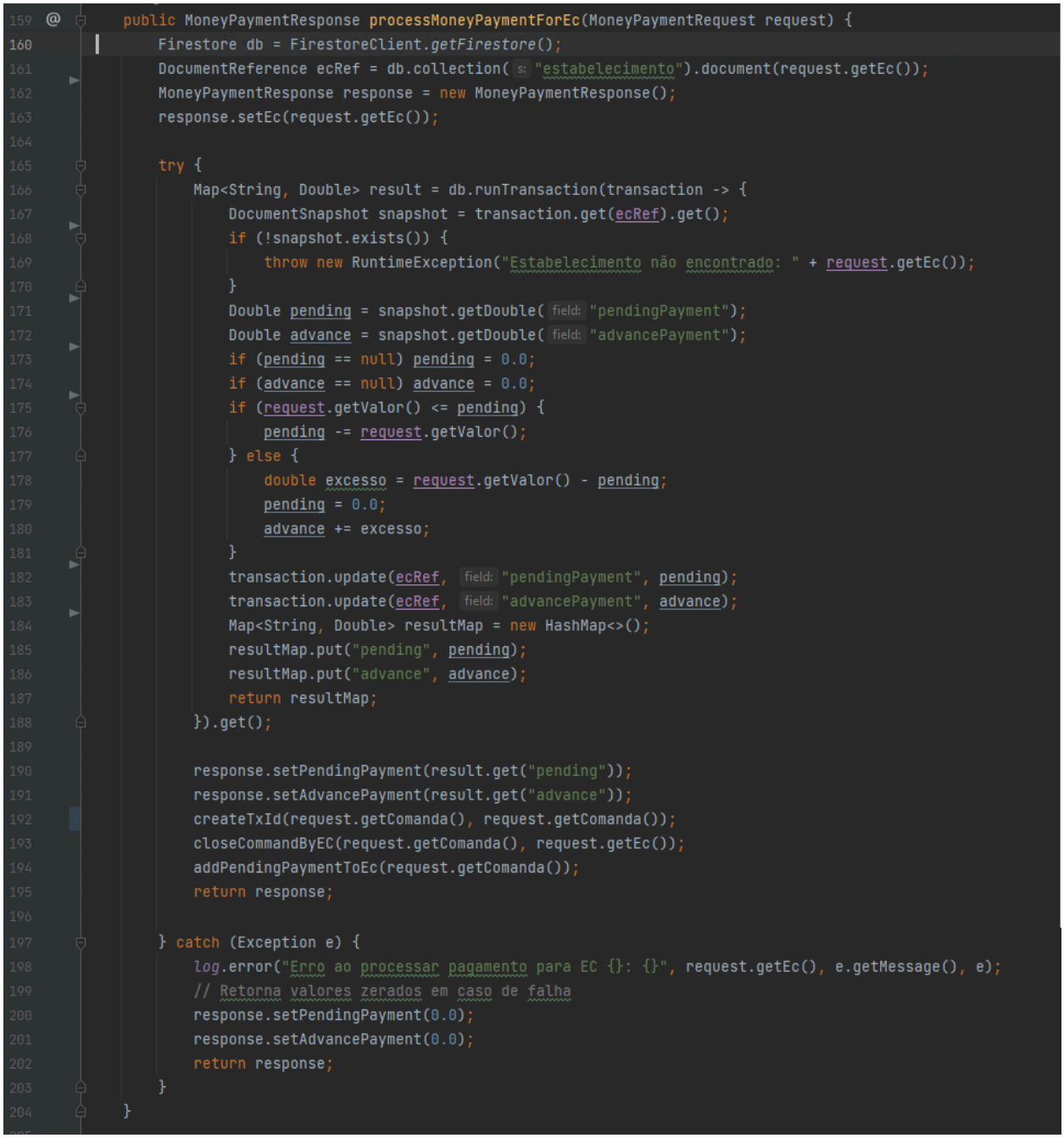
Depois disso, o método percorre cada entrada do mapa, representando um estabelecimento e o valor total associado. Para cada *EC*, ele obtém a referência ao documento correspondente na coleção "estabelecimento". A partir daí, é iniciada uma transação no Firestore, garantindo que as atualizações sejam atômicas e consistentes, mesmo que outros processos tentem modificar os mesmos dados simultaneamente.

Dentro da transação, o método lê os valores atuais de *pendingPayment* (o total pendente a receber) e de *advancePayment* (o valor já adiantado ao estabelecimento). Inicialmente, considera que o valor total a adicionar é igual ao valor total dos pedidos criados. Em seguida, aplica uma lógica de compensação: se o estabelecimento possui algum valor de adiantamento (*advancePayment*), esse valor é usado para cobrir parte ou até o valor total a ser adicionado. Caso o adiantamento seja suficiente, o valor pendente não aumenta e o adiantamento é reduzido. Se o adiantamento for menor que o total, ele é zerado e o restante é adicionado ao valor pendente.

Depois de calcular os novos valores, a transação atualiza os campos *pendingPayment* e *advancePayment* no documento do estabelecimento, aplicando as alterações de forma segura. Quando a transação é concluída com sucesso, o método registra uma mensagem de *log* informando o EC atualizado e o valor correspondente. Por fim, se ocorrer algum erro durante a execução ou na comunicação com o Firestore, uma mensagem de erro é registrada no *log* com o *txID* afetado e os detalhes da exceção.

4.3.3 Processamento de pagamento em dinheiro para o estabelecimento

No caso de um usuário preferir fazer o pagamento da comanda em dinheiro, ele pode se dirigir a um caixa do estabelecimento e pedir para que alguém leia sua comanda usando um sistema logado como estabelecimento.



```
public MoneyPaymentResponse processMoneyPaymentForEc(MoneyPaymentRequest request) {
    Firestore db = FirestoreClient.getFirestore();
    DocumentReference ecRef = db.collection( s: "estabelecimento").document(request.getEc());
    MoneyPaymentResponse response = new MoneyPaymentResponse();
    response.setEc(request.getEc());

    try {
        Map<String, Double> result = db.runTransaction(transaction -> {
            DocumentSnapshot snapshot = transaction.get(ecRef).get();
            if (!snapshot.exists()) {
                throw new RuntimeException("Estabelecimento não encontrado: " + request.getEc());
            }
            Double pending = snapshot.getDouble( field: "pendingPayment");
            Double advance = snapshot.getDouble( field: "advancePayment");
            if (pending == null) pending = 0.0;
            if (advance == null) advance = 0.0;
            if (request.getValor() <= pending) {
                pending -= request.getValor();
            } else {
                double excesso = request.getValor() - pending;
                pending = 0.0;
                advance += excesso;
            }
            transaction.update(ecRef, field: "pendingPayment", pending);
            transaction.update(ecRef, field: "advancePayment", advance);
            Map<String, Double> resultMap = new HashMap<>();
            resultMap.put("pending", pending);
            resultMap.put("advance", advance);
            return resultMap;
        }).get();

        response.setPendingPayment(result.get("pending"));
        response.setAdvancePayment(result.get("advance"));
        createTxId(request.getComanda(), request.getComanda());
        closeCommandByEC(request.getComanda(), request.getEc());
        addPendingPaymentToEc(request.getComanda());
        return response;

    } catch (Exception e) {
        log.error("Erro ao processar pagamento para EC {}: {}", request.getEc(), e.getMessage(), e);
        // Retorna valores zerados em caso de falha
        response.setPendingPayment(0.0);
        response.setAdvancePayment(0.0);
        return response;
    }
}
```

Figura 8. Método processMoneyPaymentForEc.

Fonte: Elaboração própria, 2025

Este passo habilita o pagamento em dinheiro. Ao clicar em pagamento em dinheiro (Figura 5), o método *processMoneyPayment* é invocado. Esse método é responsável por processar pagamentos em dinheiro. Ele começa a estabelecer uma conexão com o banco de dados Firestore e localiza o documento do estabelecimento a partir do identificador recebido na requisição. Em seguida, cria um objeto de resposta para armazenar o resultado da operação.

Dentro de uma transação, que garante que todas as alterações ocorram de forma segura e consistente, o código lê os valores atuais de pagamento pendente e pagamento adiantado do estabelecimento. Se esses valores não existirem, são inicializados com zero.

Depois, o método compara o valor pago com o valor pendente: se o pagamento for menor ou igual ao pendente, ele apenas reduz esse valor; se for maior, o excedente é somado ao pagamento adiantado, e o pendente é zerado. Esses novos valores são então gravados de volta no banco na mesma transação. Após a conclusão, o método atualiza o objeto de resposta com os valores calculados e executa algumas ações adicionais, como criar um identificador de transação, fechar a comanda e registrar o pagamento no sistema. Caso ocorra algum erro durante o processo, ele é registrado nos logs e a função retorna uma resposta com os valores zerados para evitar inconsistências.

4.3.4 Processamento de pagamento para o estabelecimento

```
136     public String processPendingPayments(String ecId) {
137         DocumentSnapshot doc = firebaseRepository.findEstablishmentById(ecId);
138 
139         if (doc == null || !doc.exists()) {
140             log.warn("Estabelecimento com ID {} não encontrado.", ecId);
141             throw new NotFoundException(String.format("Estabelecimento com ID {} não encontrado.", ecId));
142         }
143 
144         Double pendingPayment = doc.getDouble( field: "pendingPayment");
145         String chavePix = doc.getString( field: "chavePix");
146 
147         if (pendingPayment == null || pendingPayment <= 0.0) {
148             log.warn("Estabelecimento {} sem pagamento pendente.", ecId);
149             throw new BadRequestException(String.format("Estabelecimento {} sem pagamento pendente.", ecId));
150         }
151 
152         PixSentRequest pixSentRequest = PixSentRequest.builder()
153                 .chave(chavePix)
154                 .valor(String.format(Locale.US,  format: "%.2f", pendingPayment))
155                 .build();
156 
157         PixSentResponse response = this.sendPixPayment(pixSentRequest, ecId);
158         log.info("Pagamento enviado para EC {} no valor de R$ {}. Aguardando confirmação...", ecId, pendingPayment);
159         return response.getE2eId();
160     }
```

Figura 9. Método processPendingPayments.

Fonte: Elaboração própria, 2025

O método *processPendingPayments* é responsável por processar pagamentos pendentes de um estabelecimento específico, identificado pelo seu ecId. Ele segue uma sequência lógica de validação e ações até concluir o envio de um pagamento via Pix.

Primeiro, o método tenta localizar o documento do estabelecimento no banco de dados, por meio do *firebaseRepository*. Esse documento contém as informações financeiras e cadastrais do estabelecimento. Caso o documento não seja encontrado ou não exista, o sistema registra um aviso no log informando que o estabelecimento não foi localizado e lança uma exceção do tipo *NotFoundException*. Nesse ponto, a execução é interrompida.

Se o documento for encontrado, o método recupera dois dados principais: o valor pendente de pagamento (*pendingPayment*) e a chave Pix cadastrada (chavePix). Em seguida, ele verifica se o valor pendente é nulo ou menor ou igual a zero. Se for o caso, significa que o estabelecimento não tem valores a receber. Nesse caso, o método registra novamente um aviso no log e lança uma exceção do tipo *BadRequestException*, encerrando o processo sem tentar realizar qualquer pagamento.

Caso o valor pendente seja válido, o método prossegue para montar um objeto de requisição chamado *PixSentRequest*. Esse objeto contém as informações necessárias para o envio de um Pix — a chave Pix do estabelecimento e o valor a ser transferido. O valor é formatado com duas casas decimais e segue o padrão americano de notação numérica (com ponto decimal em vez de vírgula).

Depois de criar a requisição, o método chama outra função interna chamada sendPixPayment, que é responsável por efetivamente enviar o pagamento através do sistema de Pix da Efi Bank. Essa função deve se comunicar com uma API de pagamentos e retornar uma resposta do tipo PixSentResponse, contendo, entre outras informações, o identificador único da transação Pix, conhecido como e2eId.

Quando o pagamento é enviado com sucesso, o método registra no log uma mensagem informando o valor pago, o identificador do estabelecimento e o fato de que o sistema está aguardando a confirmação do pagamento. Por fim, ele retorna o e2eId, que pode ser usado posteriormente para rastrear ou confirmar o status do Pix enviado.

5. Conclusão

Este estudo delineou os fundamentos teóricos e as tecnologias-chave necessárias ao desenvolvimento bem-sucedido de uma aplicação web voltada a pagamentos móveis. Ao explorar conceitos essenciais como pagamentos digitais, QR Code e Pix, estabelecemos uma base sólida de compreensão para o cenário de pagamentos digitais, especialmente no contexto brasileiro em rápida evolução.

A seleção das tecnologias teve um papel fundamental para garantir que o sistema fosse seguro e eficiente. No front-end, optou-se por React, tecnologia que oferece flexibilidade e um *layout* de uso uniforme em diferentes dispositivos. Já para o back-end, a aplicação combina Spring Boot com o Firebase, tanto para o banco de dados até o serviço de autenticação, para entregar uma estrutura estável e confiável. Além disso, integramos a API do Efi Bank para facilitar transações seguras e eficientes, garantindo uma experiência de pagamento perfeita para os usuários finais.

Este estudo não apenas oferece uma visão aprofundada sobre os componentes vitais dos pagamentos móveis, mas também serve como um guia prático para desenvolvedores e pesquisadores interessados em criar soluções inovadoras e seguras no cenário de pagamentos digitais em constante mudança. A combinação de uma compreensão teórica sólida, escolhas tecnológicas criteriosas e uma metodologia de desenvolvimento ágil proporcionou uma

abordagem holística e eficaz para o desenvolvimento do sistema, oferecendo um modelo valioso para futuras iniciativas nesta área dinâmica e crucial da tecnologia.

Cabe destacar que o sistema desenvolvido possui caráter estritamente acadêmico, tendo sido implementado e avaliado em ambiente de homologação, sem utilização em produção, com o objetivo de validar os conceitos e tecnologias explorados ao longo deste trabalho.

Para trabalhos futuros, propõe-se a integração de novos métodos de pagamento, como transações com cartões de débito e crédito, a fim de ampliar as possibilidades de uso do sistema e atender a diferentes perfis de consumidores. Além disso, estudos complementares poderiam explorar aspectos de segurança e desempenho dessas integrações, assegurando a escalabilidade e a confiabilidade da aplicação em ambientes de produção. Outro trabalho a ser desenvolvido é a criação de uma ferramenta administrativa para que os estabelecimentos possam controlar seus fluxos de pagamentos de forma centralizada e eficiente.

6. Referências:

AGGARWAL, Sanchit et al. **Modern Web-Development Using ReactJS.** International Journal of Recent Research Aspects, v. 5, n. 1, p. 133-137, mar. 2018. Disponível em: https://irjaes.com/wp-content/uploads/2022/02/IRJAES-V7N1P162Y22.pdf. Acesso em: jul. 2025.

BRAIDO, Gabriel Machado; KLEIN, Amadolinda Zanela. **Análise Da Utilização De Pagamentos Móveis No Contexto Brasileiro:** Percepção De Usuários E Não Usuários. IPTEC: Revista Inovação, Projetos e Tecnologias, [*s. l.*], 2020. Disponível em: https://periodicos.uninove.br/iptec/article/view/17639/8656. Acesso em: set. 2023.

BRAIDO, Gabriel; KLEIN, Amaroinda; PAPALEO, Guilherme. **Facilitadores e Barreiras enfrentadas pelas Fintechs de Pagamentos Móveis no Contexto Brasileiro**. BBR: Brazilian Business Review, [*s. l.*], 14 dez. 2020. Disponível em: https://www.scielo.br/j/bbr/a/36khdWQsXtmDKgThWHTgzBP/?lang=pt. Acesso em: set. 2023.

BRAIDO, Gabriel Machado. **Mudanças Institucionais E Novos Ofertantes No Sistema De Pagamentos Brasileiros**: Análise A Partir Da Adoção De Pagamentos Móveis. Orientador: Prof.ª Dra. Amarolinda Zanela Klein. 2019. Tese (Doutorado em Administração) - UNIVERSIDADE DO VALE DOS SINOS, [*S. l.*], 2019. Disponível em: http://repositorio.jesuita.org.br/handle/UNISINOS/8690. Acesso em: set. 2023.

EFI BANK. **API Pix do Efí Bank:** integração, emissão de cobranças e pagamentos instantâneos via Pix. Seja Efí – Documentação técnica, 2025. Disponível em: https://dev.efipay.com.br/docs/api-pix . Acesso em: ago. 2025.

FURINI, Isabele Chaiben. **Mercado De Meios De Pagamento No Brasil**: Visão Histórica E Tendências Globais. Orientador: Prof. Dr. Glaison A. Guerrero. 2020. Trabalho de Conclusão de Curso (Graduação em Ciências Econômicas) - UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL, Porto Alegre, 2020. Disponível em: https://www.lume.ufrgs.br/handle/10183/216349. Acesso em: set. 2023.

FIREBASE. Disponível em: https://firebase.google.com/. Acesso em: set. 2023

GOOGLE**.** Disponível em: https://about.google/. Acesso em: set. 2023.

JESUS, Luan Ferreira Reis de. **Verificação Automática De Pagamento Instantâneo Via Leitura De Qr Code Dinâmico Gerenciado Por Sistema Embarcado Atuando No Atendimento Ao Cliente**. 2022. Trabalho Final de Curso (Pós-Graduação Lato Sensu em Engenharia Elétrica com ênfase em Sistemas Inteligentes Aplicados à Automação) - INSTITUTO FEDERAL DO ESPÍRITO SANTO, [*S. l.*], 2022. Disponível em: https://repositorio.ifes.edu.br/bitstream/handle/123456789/1883/TCC_Verifica%c3%a7%c3%a3o_Pagamento_QR_Code_Cliente.pdf?sequence=1&isAllowed=y. Acesso em: set. 2023.

JORGE, Paulo Alexandre Alcântara. **O papel das APIs na transformação digital.** Orientador: Prof. Dra. Winnie Picolo. 2020. Dissertação (Mestrado em Gestão de Sistemas de Informação) - Universidade de Lisboa, [*S. l.*], 2020. Disponível em: http://hdl.handle.net/10400.5/21135. Acesso em: set. 2023.

KHAWAS, Chunnu; SHAH, Pritam. **Application of Firebase in Android App Development:** A Study. International Journal of Computer Applications, [*s. l.*], v. 179, ed. 46, Junho 2018. Disponível em: https://www.researchgate.net/profile/Chunnu-Khawas/publication/325791990_Application_of_Firebase_in_Android_App_Development-A_Study/links/5bab55ed45851574f7e6801e/Application-of-Firebase-in-Android-App-Development-A-Study.pdf. Acesso em: set. 2023.

REACT. Disponível em: https://react.dev/. Acesso em: out. 2025.

META. Disponível em: https://about.meta.com/. Acesso em: nov. 2025.

NOVAES, André Luiz Faria; HARTMANN, Ivan. **Panorama sobre o Mercado Digital de Pagamentos Brasileiro:** Aspectos Legais, Business e Tecnológicos. Revista Brasileira de Direito, Passo Fundo, v. 16, p. 1-18, 30 dez. 2020. Disponível em: https://pdfs.semanticscholar.org/51df/8eb5889954f3629de907fa9953b7c19e7649.pdf. Acesso em: set. 2023.

JAVA. Disponível em: https://www.java.com/. Acesso em: out. 2025.

SILVA, Ricardo Antunes; CRUZ, Caroline Quaresma Piccinato da. **O Impacto Do Novo Ecossistema Democrático De Pagamento Instantâneo (Pix) No Sistema Financeiro Nacional.** Revista Jurídica da Universidade do Sul de Santa Catarina, [*s. l.*], v. 10, ed. 21, p. 195-208, 23 set. 2020. DOI https://doi.org/10.19177/ufd.v10e212020195-208. Disponível em: https://portaldeperiodicos.animaeducacao.com.br/index.php/U_Fato_Direito/article/view/19906. Acesso em: set. 2023.

SURWASE, Vijay. **REST API Modeling Languages** - A Developer's Perspective**.** 2016. - IJSTE - International Journal of Science Technology & Engineering, v. 2, Issue 10, abr. 2016. Disponível em: https://encurtador.com.br/dDHMV. Acesso em: dez. 2023.

WEBB, Phillip et al. **Spring Boot Reference Guide. 2.1.0.BUILD-SNAPSHOT.** [S. l.]: Pivotal Software, 2018. Disponível em: https://mackvord.github.io/PDF/spring-boot-reference.pdf. Acesso em: ago. 2025.