Documentação de Projeto

para o sistema

Tuscan - Compras em Eventos

Versão 5.0

Projeto de sistema elaborado pelo(s) aluno(s) Bernardo Cruz Rohlfs e Eric Guimarães Caldas Jardim e apresentado ao curso de **Engenharia de Software** da **PUC Minas** como parte do Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo da professora Aline Norberta de Brito, orientação acadêmica do professor Cleiton Silva Tavares e orientação de TCC II do professor (a ser definido no próximo semestre).

27/10/2024

Tabela de conteúdo

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão
Entrega 3	29/09/2024	Inclusão do documento, definindo seções 2.1, 2.2, 2.3, 4.1, 4.2, 4.3 e 4.4	1.0
Entrega 4	13/10/2024	Inclusão das seções 2.4, 3.1, 3.2 e 3.3	2.0

Entrega 5	27/10/2024	Adição de descrições e textos complementares. Inclusão das seções 3.4, 3.5, 3.6 e 5. Correção das entregas anteriores.	3.0
Entrega 6	17/11/2024	Adição dos planos de teste e cronogramas de desenvolvimento.	4.0
Entrega 7	01/12/2024	Revisão e correção de pendências.	5.0

1. Introdução

Este documento agrega: 1) a elaboração e revisão de modelos de domínio e 2) modelos de projeto para o sistema Tuscan. A referência principal para a descrição geral do problema, domínio e requisitos do sistema é o documento de especificação que descreve a visão de domínio do sistema. Tal especificação acompanha este documento. Anexo a este documento também se encontra o Glossário.

2. Modelos de Usuário e Requisitos

Esta seção tem como objetivo descrever os usuários e atores do sistema, assim como os requisitos aos quais esse deve atender. Para isso, é apresentada uma breve descrição de cada ator, assim como um modelo desse ator como usuário do sistema. Além disso, são apresentados o diagrama de caso de uso e as histórias de usuários relacionadas, ambos que servem de referência para desenvolvimento do sistema. Por último, são apresentados o diagrama de sequência do sistema e o contrato de comunicações, responsáveis por definir como a aplicação a ser projetada deve se comunicar com aplicações já existentes.

2.1. Descrição de Atores

Organizador do evento: este ator usa a aplicação Web do sistema Tuscan para realizar a gestão das vendas de comidas e bebidas oferecidas em cada um dos eventos que organiza, desde a edição de cardápios, cadastro de novos produtos (informando a quantidade em estoque e preço), até acesso a resumos e relatórios de valores brutos registrados, e aplicação de promoções.

Funcionário do bar: este ator usa o aplicativo Mobile Tuscan para escanear códigos de resposta rápida, apresentados pelos clientes consumidores, para validar as vendas de alimentos e bebidas durante eventos, com a finalidade de entregar os produtos comprados ao seus destinatários com praticidade e agilidade.

Cliente do evento: este ator usa a aplicação Web do sistema Tuscan para acessar o cardápio do evento que está frequentando (por meio do endereço disponibilizado pela equipe do evento em códigos de resposta rápida espalhados pelo estabelecimento) e comprar comidas e bebidas de forma prática, utilizando métodos de pagamento digitais. Após a compra, ele recebe o comprovante e o resumo da compra em sua caixa de e-mail.

2.2. Modelos de Usuários

Esta subseção tem como objetivo descrever os modelos de usuários desenvolvidos por meio da implementação de personas.

A **Tabela 1** descreve a persona do usuário Daniel, que é organizador de eventos, e usa o Sistema Tuscan para elevar a qualidade e agilidade dos processos de venda em seus eventos e ter acesso a informações importantes sobre os dados estatísticos obtidos durante cada evento.

Tabela 01	
Daniel	
Descrição	Daniel, aos 30 anos, é um organizador de eventos apaixonado pelo que faz. Com um perfil extrovertido e sociável, ele gosta de conhecer novas pessoas e tem uma mentalidade empreendedora, sempre disposto a explorar novos negócios e oportunidades. A música é sua grande paixão e hobby, o que, de certa forma, influencia também seu trabalho no mercado de entretenimento. Seu objetivo de vida é se tornar uma referência no mercado de produção de eventos, estabelecendo-se como um grande nome no setor.
Dores	● Perda de tempo com burocracia desnecessária. ● Frustração com processos organizacionais lentos e ineficazes.
Objetivos	● Otimizar processos organizacionais e aumentar as vendas de produtos nos eventos. ● Compreender melhor os clientes por meio de métricas e relatórios, e aumentar a receita dos eventos.

Tabela 1. Persona Daniel

A **Tabela 2** descreve a persona do usuário Roberta, que é funcionária de bares em eventos, e usa o Sistema Tuscan para escanear códigos de resposta rápida, apresentados pelos clientes consumidores, para validar as vendas de alimentos e bebidas durante eventos.

Tabela 02	
Roberta	
Descrição	Roberta tem 28 anos e trabalha como bartender, uma profissão que ela desempenha com muita energia e atenção. Extrovertida e organizada, ela gosta de interagir com pessoas e lida bem com a pressão diária de seu trabalho. Além disso, Roberta possui um ótimo senso de humor, o que torna suas interações com os clientes mais leves e agradáveis. Nos momentos de folga, ela se dedica a hobbies como dançar salsa e praticar yoga, atividades que refletem sua personalidade vibrante e disciplinada. Seu grande sonho é abrir um bar em uma cidade litorânea, onde possa combinar coquetéis artesanais com música ao vivo, proporcionando experiências únicas para seus clientes.

Dores	● Irritação com clientes rudes e arrogantes que desrespeitam seu trabalho. ● Falta de praticidade e agilidade durante o processo de aceite e entrega de pedidos.
Objetivos	● Atender clientes com eficiência e manter o ambiente organizado. ● Garantir uma experiência agradável e drinks de qualidade.

Tabela 2. Persona Roberta

A **Tabela 3** descreve a persona do usuário Iago, que é frequentador e cliente de eventos, e usa o Sistema Tuscan para acessar o cardápio do evento que está frequentando e comprar comidas e bebidas de forma prática, utilizando métodos de pagamento digitais.

Tabela 03	
Iago	
Descrição	Iago é um jovem de 22 anos que está no início de sua carreira jurídica, trabalhando como estagiário em um escritório de advocacia. Ele é uma pessoa analítica, determinada e muito focada em seus objetivos. Apesar de ser tímido em situações sociais, Iago se sente bastante confiante em ambientes acadêmicos e profissionais, onde seu conhecimento e dedicação o destacam. Fora do trabalho, ele tem o hobby de assistir séries de investigação criminal, o que o mantém sempre curioso e engajado com seu campo de estudo. Seu maior sonho é tornar-se um advogado especializado em Direito Penal, com o desejo de atuar em grandes casos de defesa criminal.
Dores	● Frustração quando não é levado a sério ou subestimado devido à inexperiência. ● Falta de organização e profissionalismo, especialmente em prazos e competência no trabalho.
Objetivos	● Estudar ao máximo para garantir boa performance nas provas e no estágio. ● Destacar-se academicamente e profissionalmente, com ferramentas que facilitem seu aprendizado.

Tabela 3. Persona Iago

2.3. Modelo de Casos de Uso e Histórias de Usuários

Esta subseção tem como objetivo descrever os casos de uso e histórias de usuário previstos para o projeto. Para isso é apresentado um diagrama de casos de uso onde todos são listados. Dessa mesma forma, também são apresentadas as histórias de usuário relacionadas às funcionalidades previstas para o sistema a ser desenvolvido.

2.3.1. Diagrama de Casos de Uso

A **Figura 1** representa o diagrama de casos de uso referente ao sistema proposto. No

No diagrama é possível ver 3 atores principais: organizador de eventos, funcionário do bar e cliente do eventos, que são os usuários do sistema. Também é possível visualizar 1 sistema externo, o Stripe API, que é responsável por realizar as transações financeiras para as vendas de comidas e bebidas realizadas em um evento. Para fins de organização, utiliza-se identificadores no formato US#ID, em que US refere-se a User Story. As histórias de usuário em questão são descritas na próxima seção.

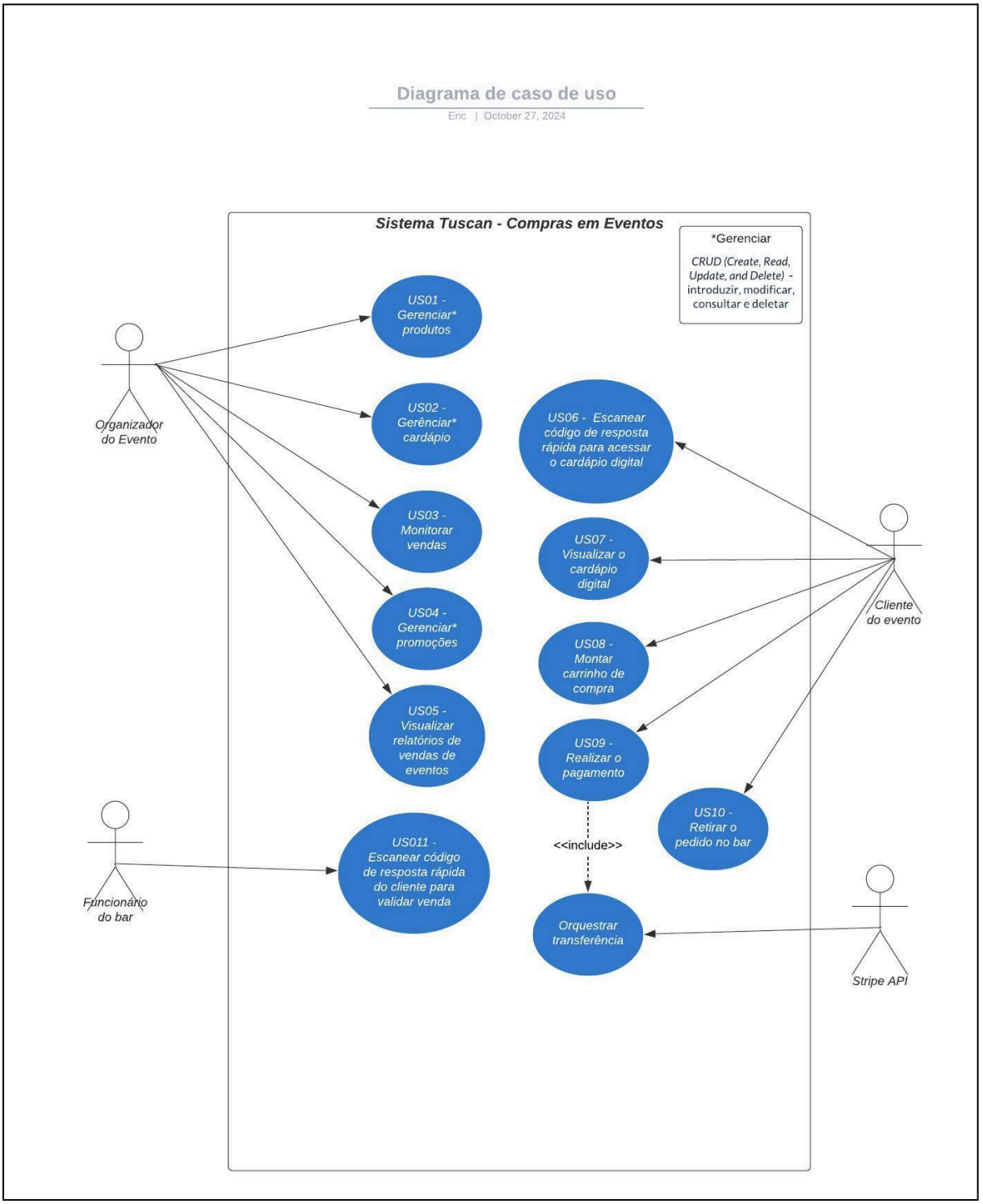

Figura 1. Casos de Uso

2.3.2. Histórias de Usuário

Nesta seção, a **Tabela 4** lista as histórias de usuários para o sistema proposto. Para fins de organização, utiliza-se identificadores no formato US#ID, em que US refere-se a User Story. Assim, as histórias de usuário identificadas para o sistema são:

Organizador do evento	
US 01	Como organizador do evento, gostaria de gerenciar* os produtos (bebidas, alimentos, etc.) que poderão ser adicionados aos cardápios dos meus eventos, para que os clientes possam visualizá-los e incluí-los em seus pedidos online.
US 02	Como organizador do evento, gostaria de gerenciar* a composição dos cardápios dos meus eventos e atualizar o estoque de cada produto em tempo real.
US 03	Como organizador do evento, gostaria de visualizar as vendas em tempo real, para monitorar o desempenho e identificar os produtos mais vendidos.
US 04	Como organizador do evento, gostaria de gerenciar* promoções, para oferecer descontos ou ofertas aos clientes durante o evento.
US 05	Como organizador do evento, gostaria de visualizar relatórios de vendas do evento, para analisar o comportamento dos clientes e melhorar a oferta de produtos em eventos futuros.
Cliente do evento	
US 06	Como cliente do evento, gostaria de acessar o cardápio digital do evento, por meio de códigos de resposta rápida espalhados pelo estabelecimento, sem ter que ir até o caixa.
US 07	Como cliente do evento, gostaria de visualizar o cardápio digital com todos os produtos e promoções disponíveis, para escolher o que vou comprar.
US 08	Como cliente do evento, gostaria de montar meu carrinho de compra, para efetuar a compra de vários produtos de uma só vez.
US 09	Como cliente do evento, gostaria de realizar o pagamento online usando Pix ou cartão de crédito, para evitar filas e agilizar a compra.
US 10	Como cliente do evento, gostaria de receber (na plataforma e na minha caixa de e-mail) um conjunto de códigos de resposta rápida para cada item comprado, para posteriormente retirá-los no bar.
Funcionário do bar	

US 11	Como funcionário do bar, gostaria de escanear os códigos de resposta rápida dos clientes do evento, para validar a venda e agilizar a entrega dos produtos comprados.
**Gerenciar entende-se as operações básicas de CRUD (Create, Read, Update, and Delete) ou seja: introduzir, modificar, consultar e deletar.*	

Tabela 4. Histórias de usuário

2.4. Diagrama de Sequência do Sistema e Contrato de Operações

Nesta seção, são apresentados os diagramas de sequência do sistema, assim como seus contratos de operações. O objetivo destes diagramas é descrever os fluxos de interação existentes entre os usuários e o sistema a ser desenvolvido.

A **Figura 2** é um diagrama de sequência de sistema que descreve como o organizador do evento gerencia os produtos disponíveis para seus eventos. Ele cobre as operações básicas de CRUD (criar, visualizar, atualizar e excluir produtos), permitindo que o organizador configure bebidas, alimentos e outros itens que comporão os cardápios digitais. Este fluxo se relaciona diretamente à história de usuário US01.

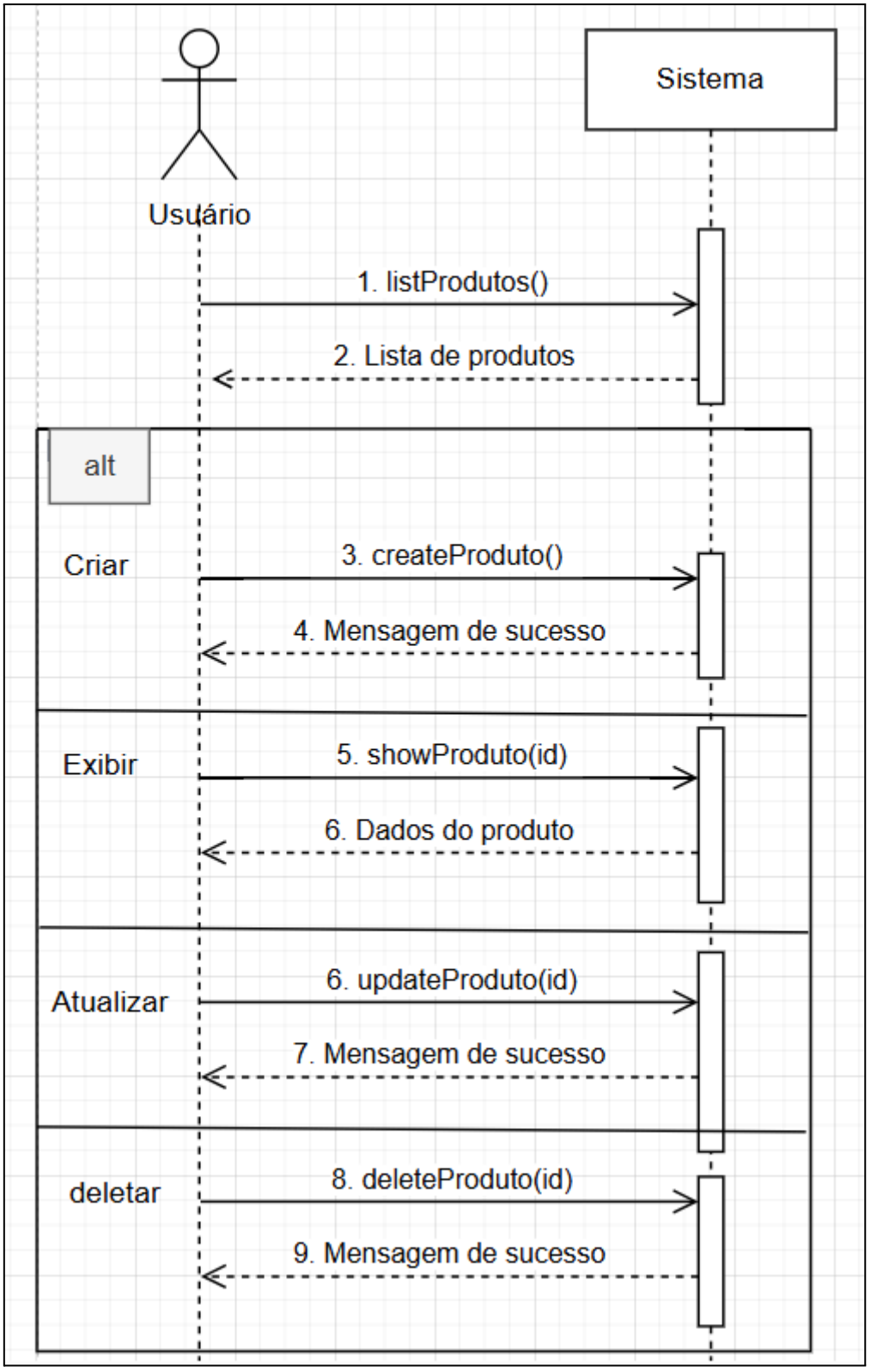

Figura 2. DSS - US 1 Gerenciar produtos

Contrato	Gerenciar Produtos
Operação	gerenciarProdutos()

Referências cruzadas	Histórias de usuário: US01
Pré-condições	O organizador deve estar autenticado no sistema
Pós-condições	O produto é adicionado, atualizado ou removido com sucesso no banco de dados

Tabela 5. Contrato Gerenciar Produtos

A **Figura 3** ilustra o processo pelo qual o organizador gerencia os cardápios dos eventos. O organizador pode criar, editar e excluir composições de cardápios, além de atualizar o estoque de produtos em tempo real para refletir a disponibilidade atualizada durante o evento. Este diagrama está vinculado à história de usuário US02.

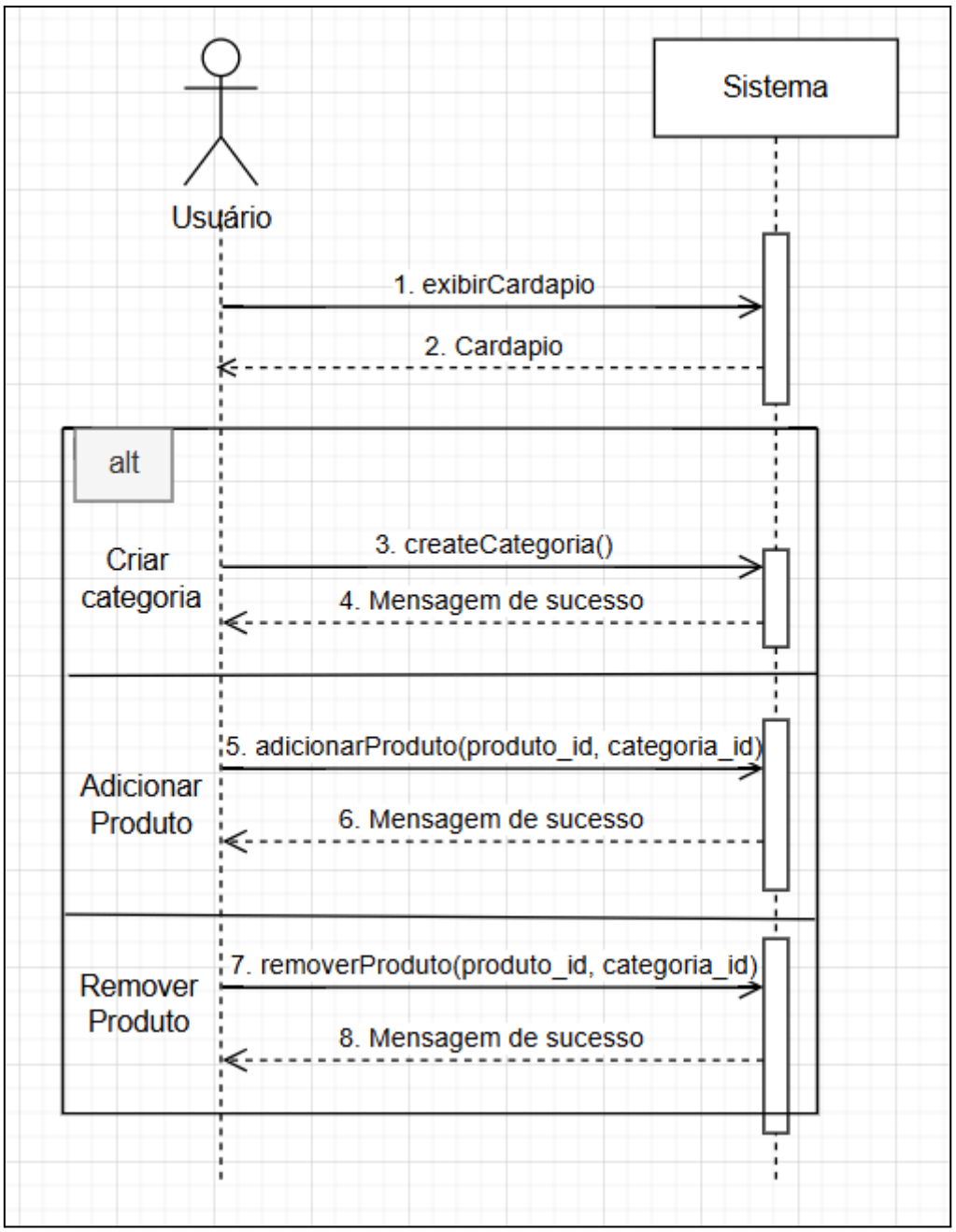

Figura 3. DSS - US 2 - Gerenciar cardápio

Contrato	Gerenciar Cardápio e Estoque
Operação	gerenciarMenu()
Referências cruzadas	Histórias de usuário: US02
Pré-condições	O organizador deve estar autenticado e o evento deve estar registrado no sistema
Pós-condições	O cardápio ou o estoque é atualizado no banco de dados

Tabela 6. Contrato Gerenciar Cardápio e Estoque

A **Figura 4** mostra como o organizador visualiza dados de vendas em tempo real. Ele consulta o sistema para acessar informações atualizadas sobre o desempenho das vendas e identificar os produtos mais vendidos, ajudando no monitoramento dinâmico do evento. Este fluxo atende à história de usuário US03.

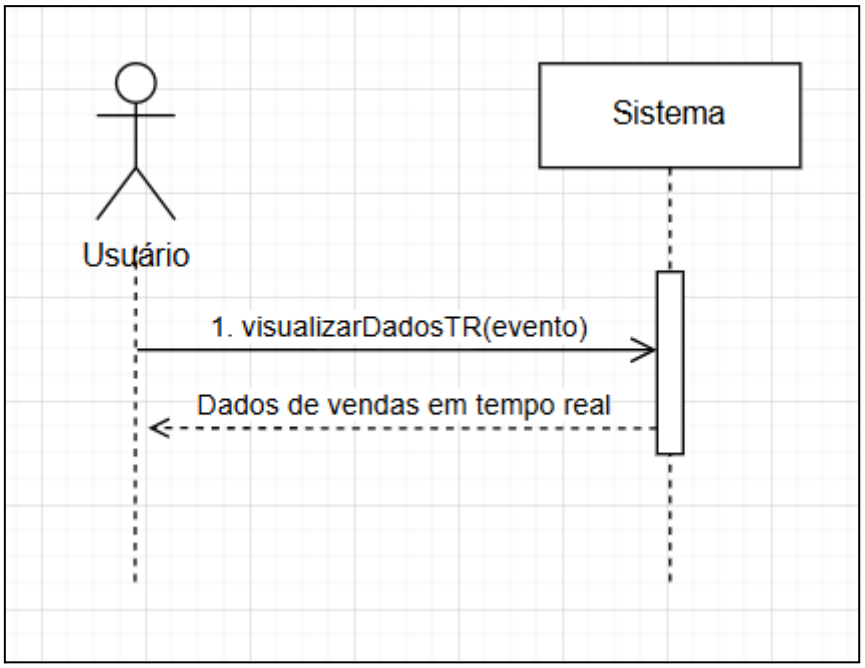

Figura 4. DSS - US 3 - Monitorar vendas

Contrato	Visualizar Vendas em Tempo Real
Operação	visualizarDadosTR(evento)
Referências cruzadas	Histórias de usuário: US03
Pré-condições	O organizador deve estar autenticado e vinculado ao evento solicitado
Pós-condições	Os dados de vendas em tempo real são exibidos na interface do usuário

Tabela 7. Contrato Visualizar Vendas em Tempo Real

A **Figura 5** descreve o processo de gerenciamento de promoções pelo organizador. Ele permite criar, atualizar e excluir promoções para produtos, aplicando descontos ou ofertas personalizadas durante o evento. O organizador também pode visualizar detalhes de promoções existentes. Este fluxo se refere à história de usuário US04.

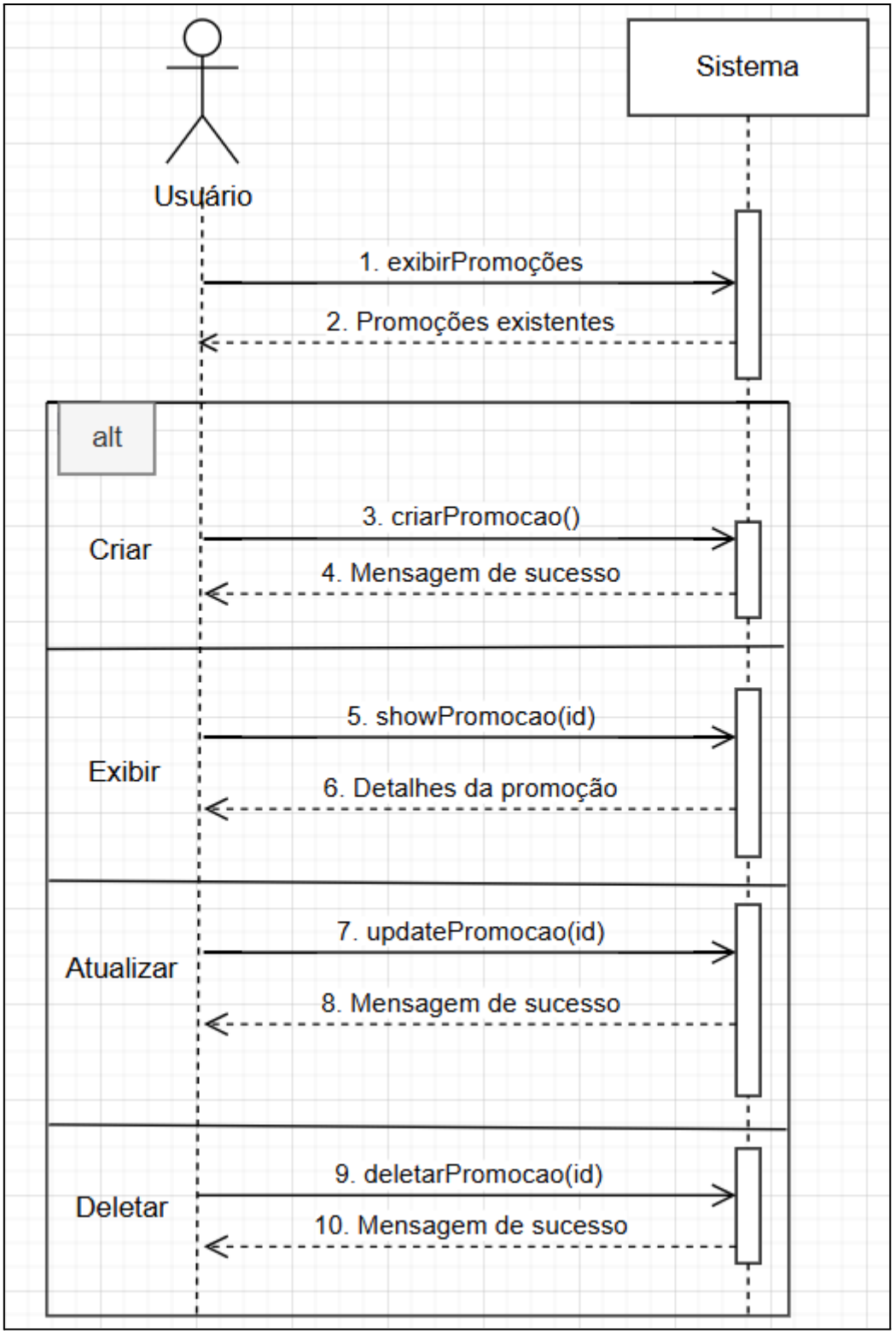

Figura 5. DSS - US 4 - Gerenciar promoções

Contrato	Gerenciar Promoções
Operação	gerenciarPromocoes()

Referências cruzadas	Histórias de usuário: US04
Pré-condições	O organizador deve estar autenticado e o evento deve estar registrado no sistema
Pós-condições	A promoção é criada, atualizada ou excluída com sucesso no banco de dados

Tabela 8. Contrato Gerenciar Promoções

A **Figura 6** detalha como o organizador visualiza relatórios de vendas do evento. O organizador solicita relatórios consolidados e o sistema retorna dados históricos sobre vendas e comportamento dos clientes, ajudando na análise e planejamento de eventos futuros. Este fluxo está associado à história de usuário US05.

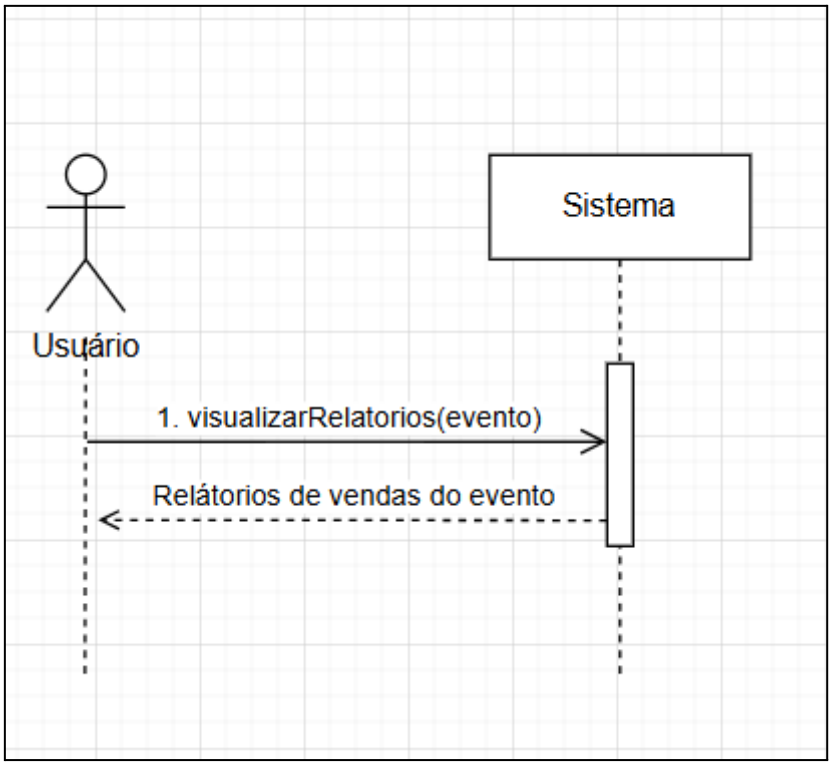

Figura 6. DSS - US 5 - Visualizar relatórios de vendas de eventos

Contrato	Visualizar Relatórios de Vendas
Operação	visualizarRelatorios(evento)
Referências cruzadas	Histórias de usuário: US05
Pré-condições	O organizador deve estar autenticado e vinculado ao evento solicitado

Pós-condições	O relatório é gerado e exibido na interface do usuário

Tabela 9. Contrato Visualizar Relatório de Vendas

O diagrama da **Figura 7** combina as histórias de usuário US06 e US07, mostrando como o cliente acessa o cardápio digital do evento por meio de QR Codes e navega pelos produtos disponíveis. O cliente pode visualizar itens com descrições e preços, além de verificar promoções aplicadas.

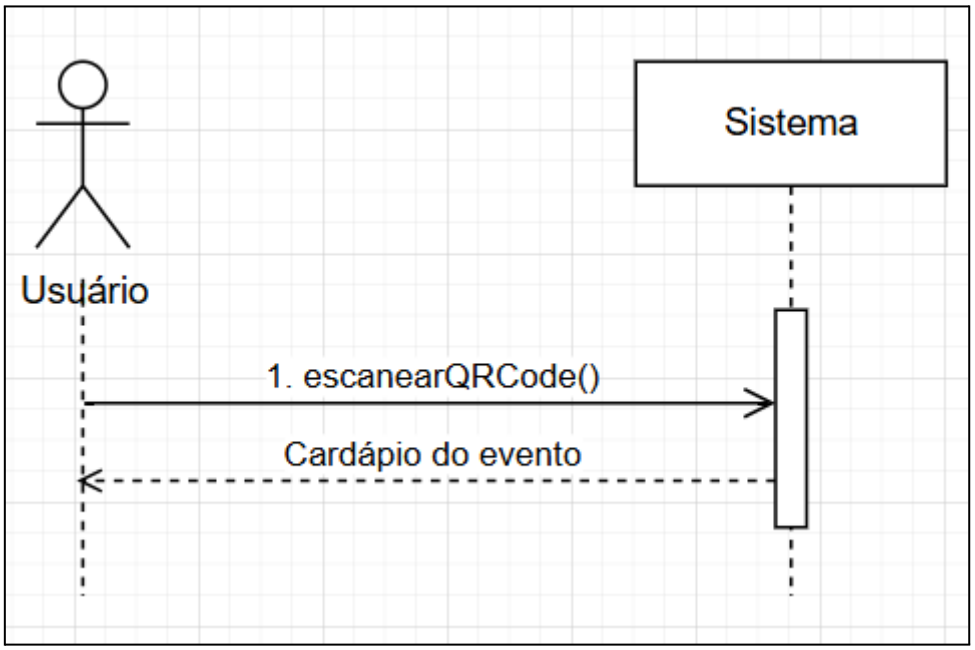

Figura 7. DSS - US 6 e 7 - Escanear código de resposta rápida e visualizar cardápio

Contrato	Acessar Cardápio Digital
Operação	escanearQRCode()
Referências cruzadas	Histórias de usuário: US06, US07
Pré-condições	O QR Code deve estar válido e o evento ativo no sistema
Pós-condições	O cardápio do evento é exibido ao cliente na interface

Tabela 10. Contrato Acessar Cardápio Digital

A **Figura 8** descreve o fluxo para montagem do carrinho de compras pelo cliente. O cliente pode adicionar produtos ao carrinho, verificar os itens selecionados e removê-los, se necessário, antes de prosseguir para o checkout. Este processo atende à história de usuário US08.

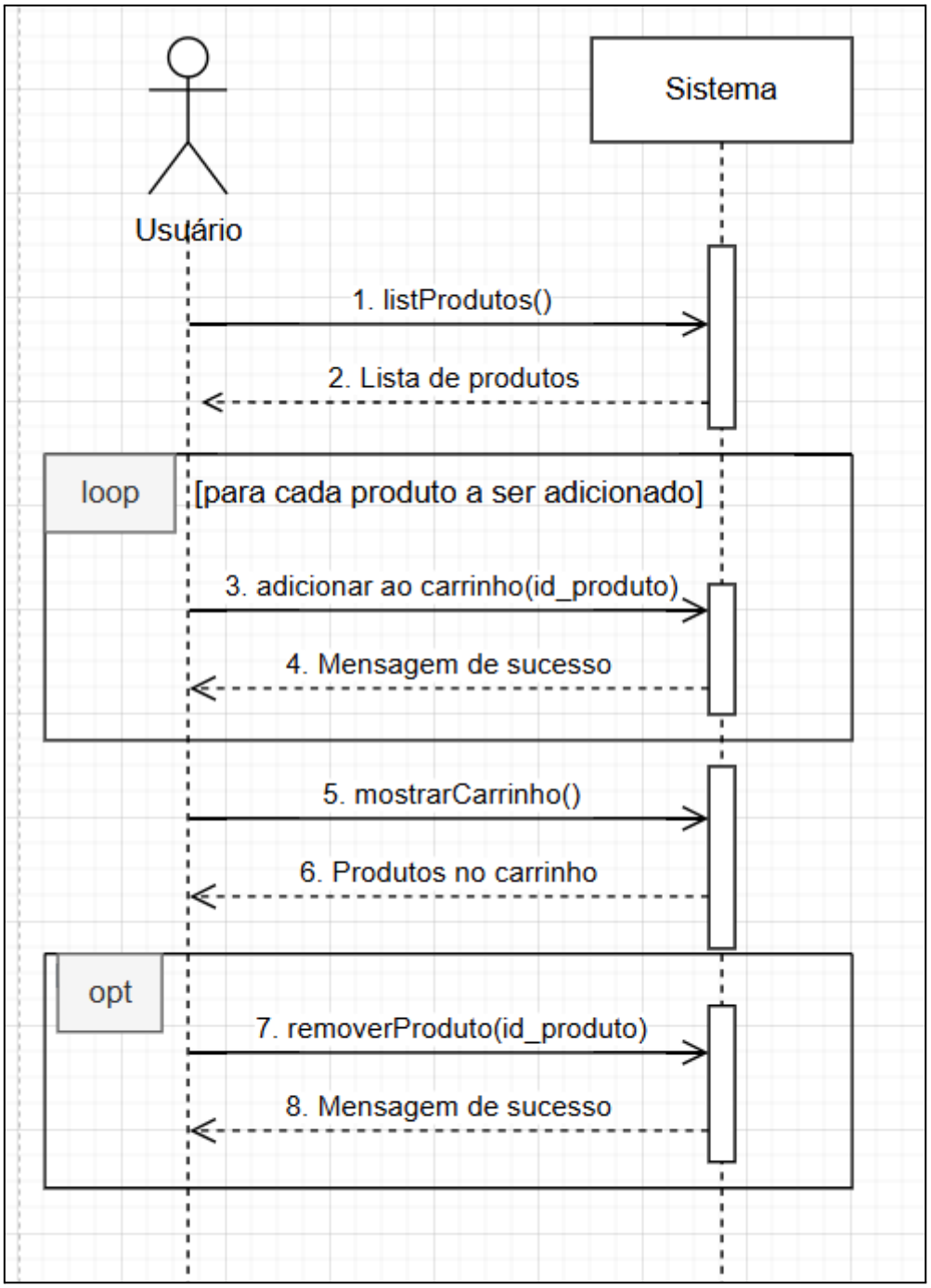

Figura 8. DSS - US 8 - Montar carrinho de compra

Contrato	Montar Carrinho
Operação	gerenciarCarrinho()
Referências cruzadas	Histórias de usuário: US08
Pré-condições	O cliente deve ter acessado o cardápio digital do evento

Pós-condições	O produto é adicionado ou removido do carrinho, e o carrinho atualizado é exibido

Tabela 11. Contrato Montar Carrinho

A **Figura 9** é um diagrama que ilustra o fluxo combinado das histórias US09 e US10, cobrindo o processo de finalização da compra e geração dos QR Codes. Após realizar o pagamento online via Pix ou cartão de crédito, o sistema gera um QR Code para cada item comprado e envia os códigos por e-mail para que o cliente possa utilizá-los posteriormente.

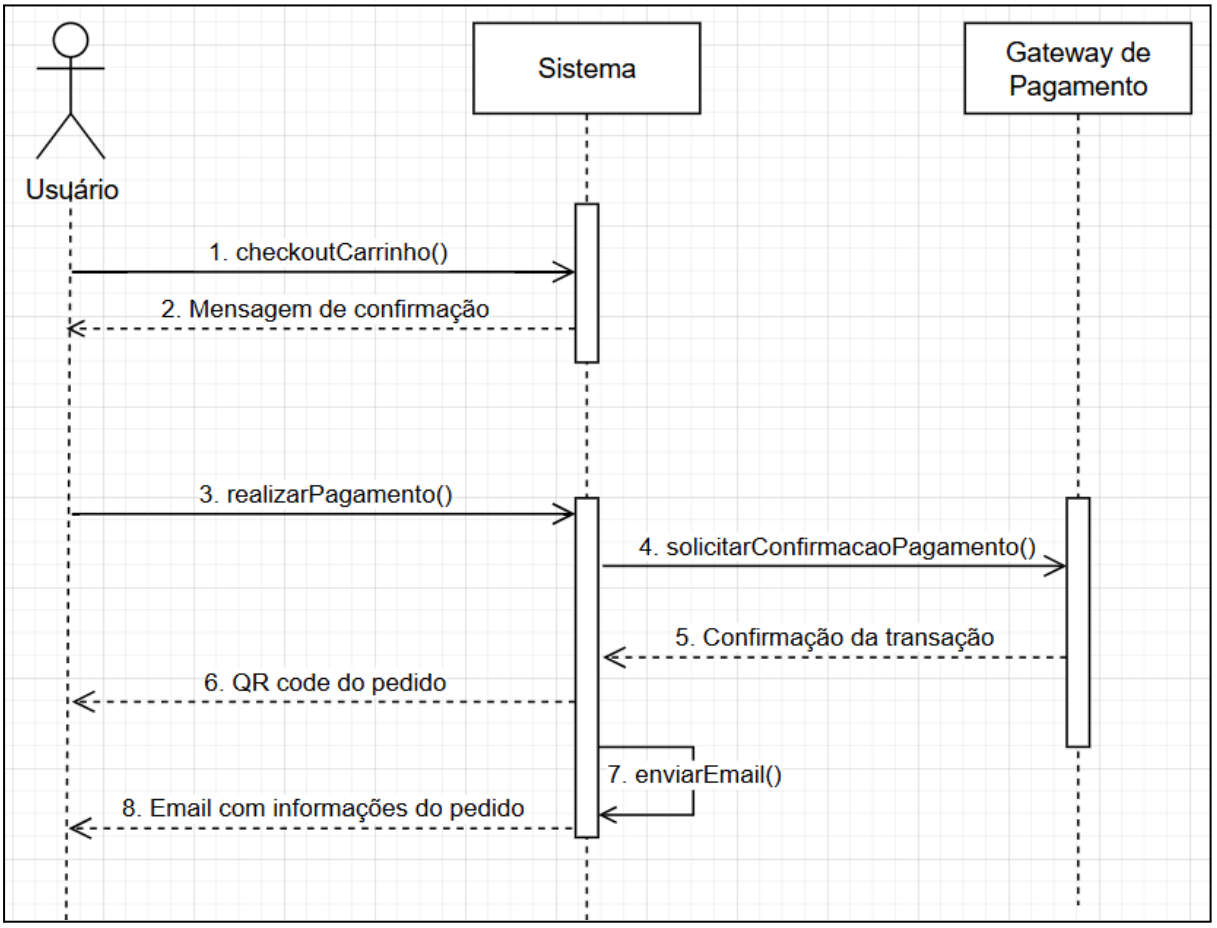

Figura 9. DSS - US 9 e 10 - Realizar pagamento e retirar pedido no bar

Contrato	Finalizar Compra e Gerar QR Codes
Operação	checkoutCarrinho()
Referências cruzadas	Histórias de usuário: US09, US10
Pré-condições	O cliente deve ter um carrinho preenchido e o pagamento deve ser aprovado

Pós-condições	Os QR Codes para os itens comprados são gerados e enviados por e-mail

Tabela 12. Contrato Finalizar Compra e Gerar QR Codes

A **Figura 10** mostra o diagrama que representa o fluxo pelo qual o funcionário do bar escaneia os QR Codes apresentados pelos clientes. O sistema valida os códigos e retorna as informações do pedido, permitindo que o funcionário agilize a entrega dos produtos comprados. Este fluxo está relacionado à história de usuário **US11**.

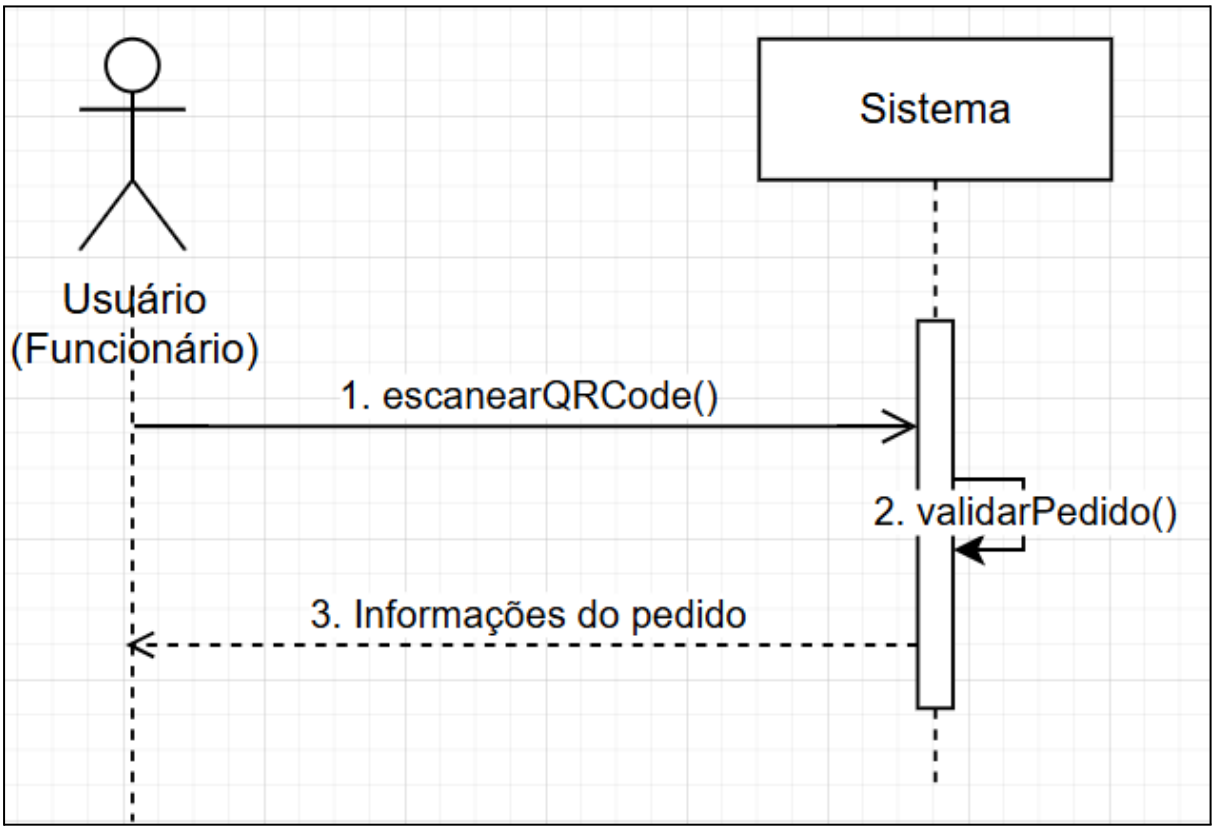

Figura 10. DSS - US 11 Escanear código de resposta rápida do cliente para validar venda

Contrato	Validar Pedido
Operação	validarPedido()
Referências cruzadas	Histórias de usuário: US11
Pré-condições	O funcionário deve estar autenticado e o QR Code deve ser válido
Pós-condições	O pedido é validado e os detalhes são exibidos ao funcionário

Tabela 13. Contrato Validar Pedido

3. Modelos de Projeto

3.1. Diagrama de Classes

A **Figura 11** mostra o diagrama de classes do sistema Tuscan, que representa a sua arquitetura lógica, detalhando as principais entidades, seus atributos, métodos e os relacionamentos entre elas. Ele foi projetado para fornecer uma visão de alto nível da estrutura do sistema, evidenciando como as diferentes partes interagem para suportar as funcionalidades principais, como gestão de eventos, produtos, promoções e pedidos.

O diagrama inclui classes centrais, como “AgenciaDeEventos”, responsável pela organização de eventos e gestão de produtos, e “Usuario”, que representa os diferentes tipos de usuários do sistema, incluindo organizadores e vendedores. Também estão presentes classes que modelam o processo de vendas, como Pedido, OrdemDeCompra e “Produto Evento”, as quais encapsulam informações sobre produtos, promoções e transações. Este diagrama serve como base para a implementação técnica e garante que a lógica do sistema esteja alinhada aos objetivos e necessidades descritos na documentação do projeto.

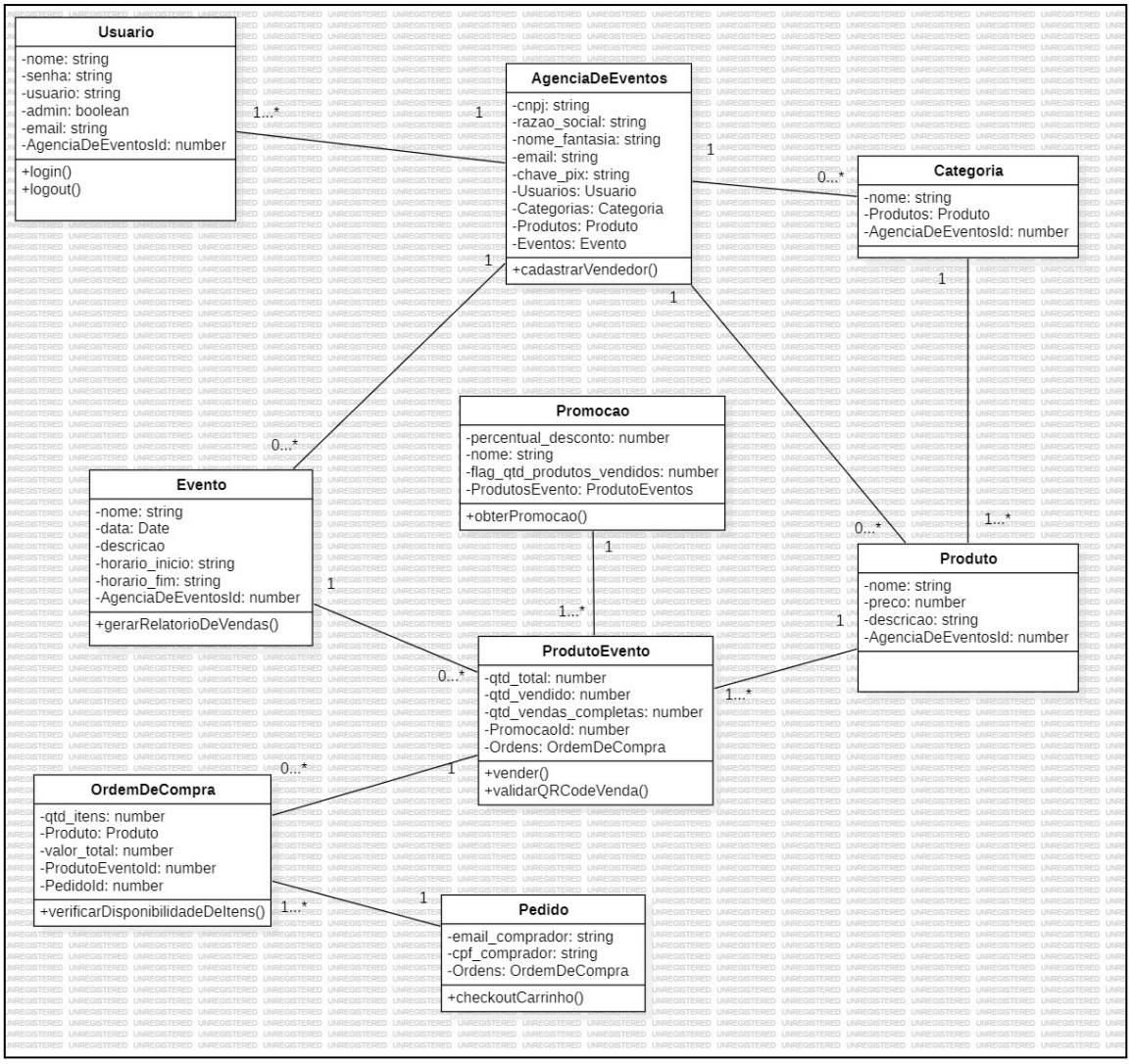

Figura 11. Diagrama de classes

3.2. Diagramas de Sequência

Nesta seção, são apresentados os diagramas de sequência que modelam o sistema a ser implementado. Esses diagramas têm como objetivo mapear os principais fluxo seguidos pelos usuários ao utilizar a aplicação. Para isso, as mensagens trocadas entre os diversos pacotes são representadas, para que a aplicação funcione da maneira correta.

A **Figura 12** ilustra o fluxo de interação entre o cliente, o cardápio digital, o Stripe, o backend do sistema e o banco de dados para o processo de realizar uma compra no Tuscan. Esse fluxo começa com o cliente acessando o cardápio via QR Code e adicionando produtos ao carrinho. Após a finalização do carrinho, o sistema gera o resumo da compra e direciona o cliente para a realização do pagamento via Pix, com integração ao Stripe para validação e registro do pedido no banco de dados. Este diagrama está diretamente relacionado às histórias de usuário **US6, US7, US8 e US9,** que cobrem o acesso ao cardápio digital, visualização de itens, montagem do carrinho e realização de pagamentos online.

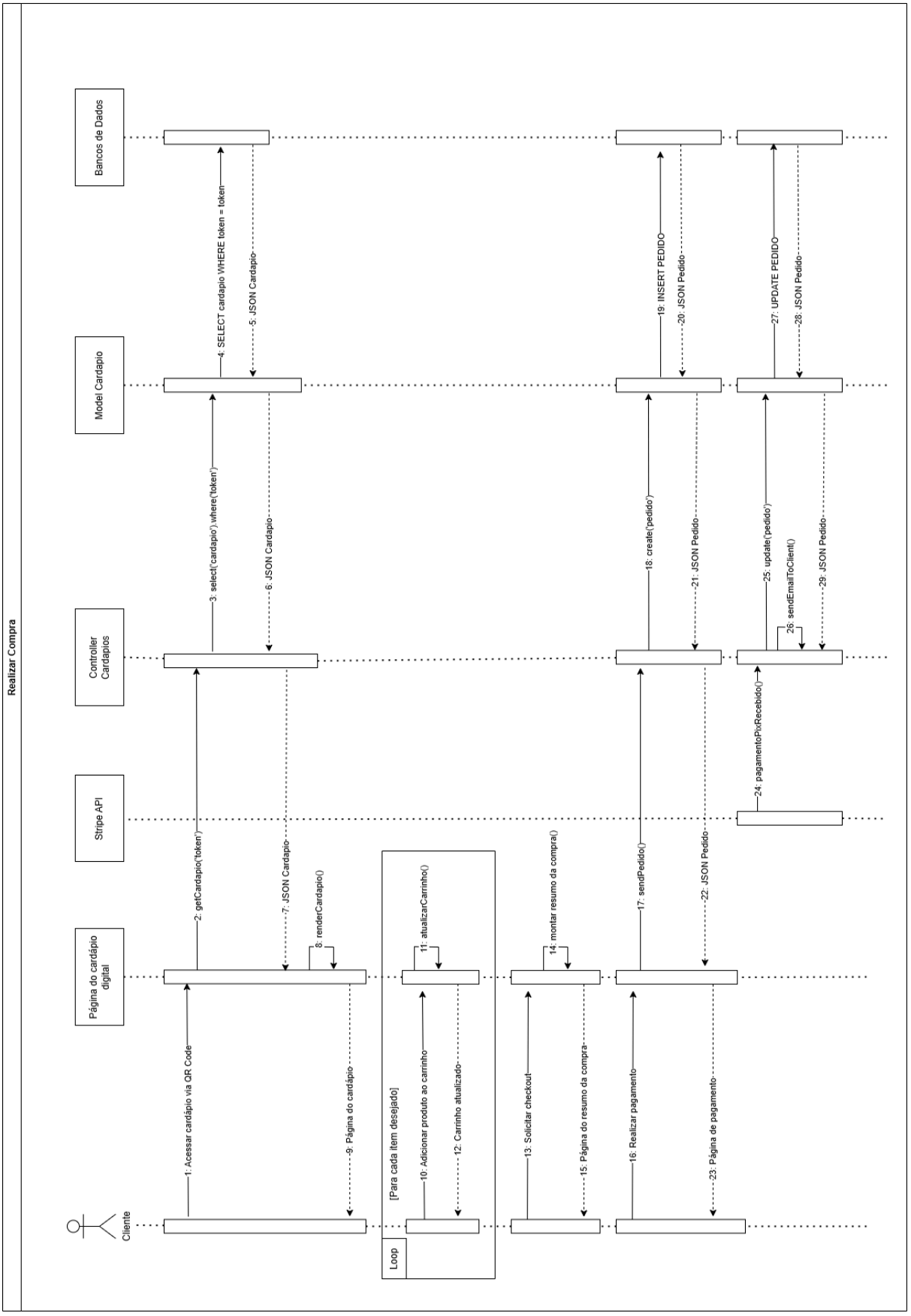

Figura 12. DS - Realizar Compra

A **Figura 13** apresenta o fluxo de interação para a entrega de um pedido no **Tuscan**, envolvendo o cliente, o funcionário, o aplicativo (tela de scanner), o backend e o banco de dados. O processo inicia com o cliente apresentando o QR Code ao funcionário, que o escaneia para validar o pedido. Após a confirmação da validade no backend e a atualização do status do pedido no banco de dados, o produto é entregue ao cliente. Em caso de erro ou QR Code inválido, o sistema exibe uma mensagem de falha. Este diagrama se relaciona às histórias de usuário **US10 e US11**, que tratam da validação de pedidos via QR Code e da entrega eficiente de produtos.

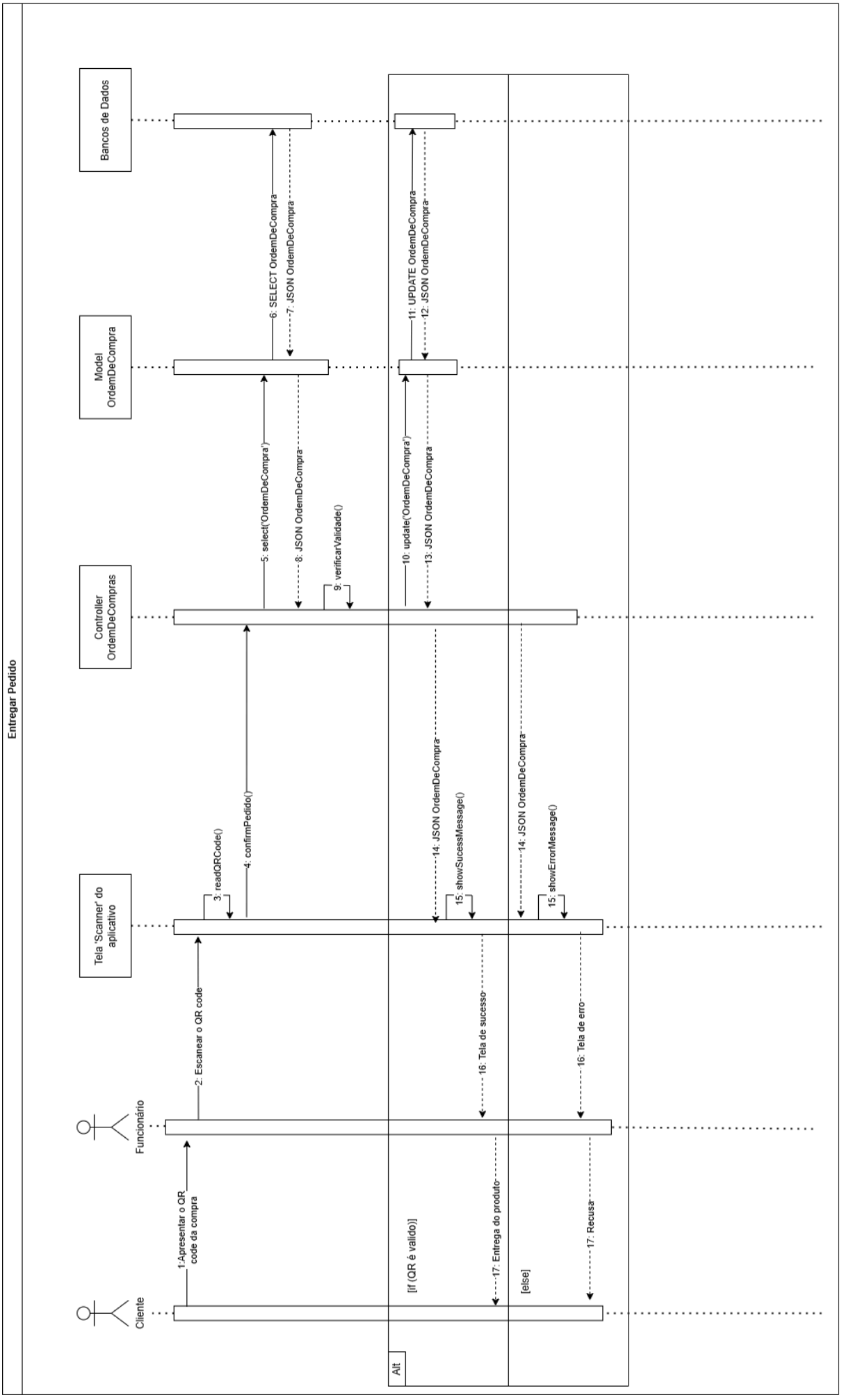
Entregar Pedido
Cliente
Funcionário
Tela 'Scanner' do aplicativo
Controller OrdemDeCompras
Model OrdemDeCompra
Bancos de Dados
1: Apresentar o QR code da compra
2: Escanear o QR code
3: readQRCode()
4: confirmPedido()
5: select('OrdemDeCompra')
6: SELECT OrdemDeCompra
7: JSON OrdemDeCompra
8: JSON OrdemDeCompra
9: verificarValidade()
10: update('OrdemDeCompra')
11: UPDATE OrdemDeCompra
12: JSON OrdemDeCompra
13: JSON OrdemDeCompra
14: JSON OrdemDeCompra
15: showSucessMessage()
14: JSON OrdemDeCompra
15: showErrorMessage()
16: Tela de sucesso
16: Tela de erro
17: Entrega do produto
17: Recusa
Alt
[if (QR é valido)]
[else]

Figura 13. DS - Entregar Pedido

3.3. Diagramas de Comunicação

Nesta seção, são apresentados os diagramas de comunicação que modelam as trocas de mensagens dos diversos pacotes do sistema. O principal objetivo desses diagramas é representar todos os agentes e as mensagens trocadas entre eles no sistema. Por conta disso, os principais fluxos da aplicação são representados como forma de documentar as comunicações que os compõem.

A **Figura 14** é um diagrama de comunicação que representa a interação entre os diferentes elementos do sistema durante o processo de Realizar Compra, cobrindo as histórias de usuário US6, US7, US8 e US9. Ele descreve como o cliente interage com a interface do cardápio digital para acessar produtos, adicionar itens ao carrinho e finalizar o pedido. A interação envolve a comunicação com o controlador de cardápios, a API Stripe para pagamento via Pix, e o controlador de pedidos, que registra a transação no banco de dados. O diagrama destaca o fluxo de mensagens entre os componentes necessários para concluir a compra de forma integrada.

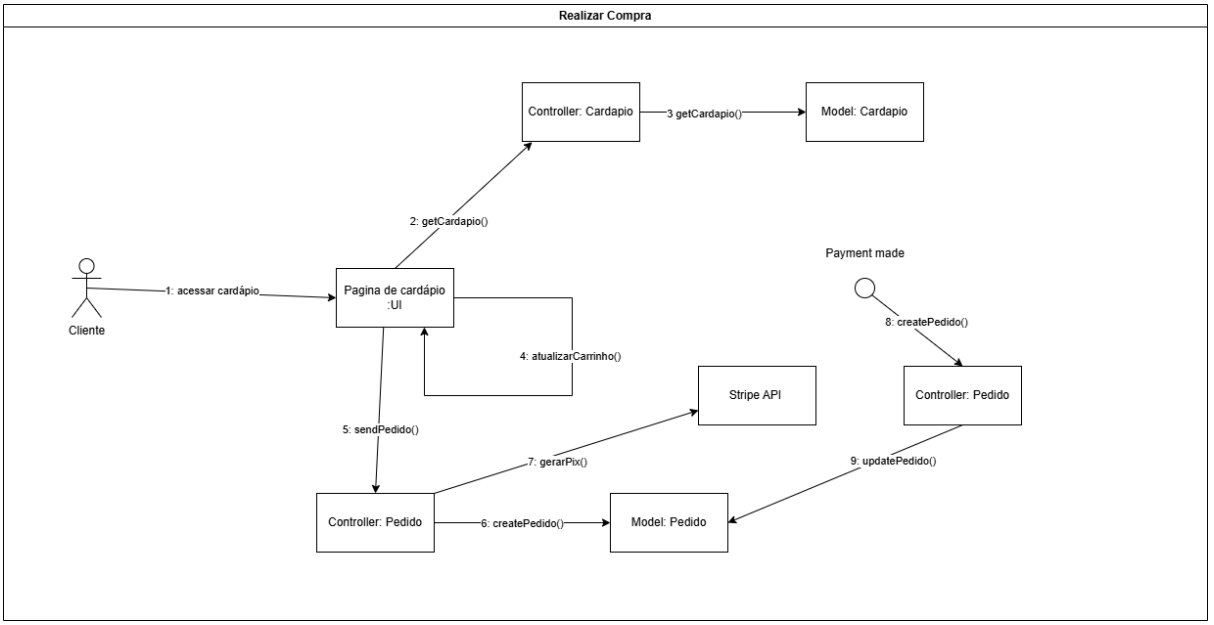

Figura 14. DC - Realizar Compra

A **Figura 15** é um diagrama de comunicação que ilustra as interações envolvidas no processo de Entregar Pedido, cobrindo as histórias de usuário US10 e US11. Ele detalha como o funcionário utiliza a tela de scanner para validar o QR Code do pedido apresentado pelo cliente. A validação envolve o controlador de ordens de compra, que consulta e atualiza as informações no modelo e banco de dados. O diagrama enfatiza a troca de mensagens para garantir a entrega correta e eficiente do produto ao cliente.

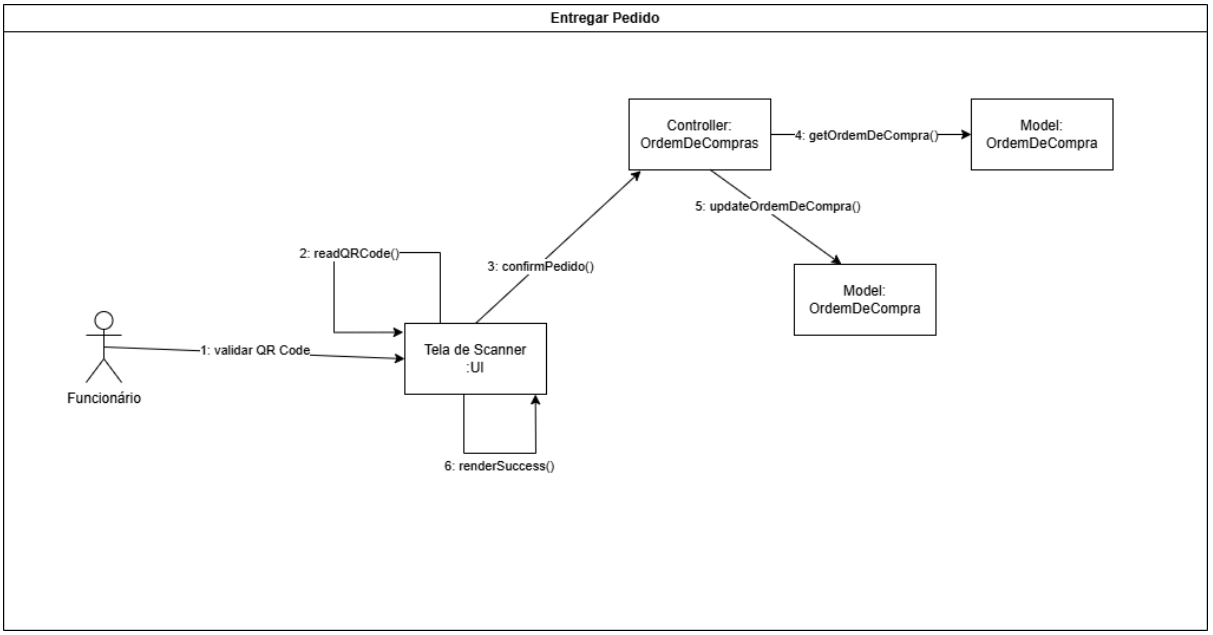

Figura 15. DC - Entregar Pedido

3.4. Arquitetura

A **Figura 16** representa o diagrama de pacotes, ou de arquitetura, do sistema. O diagrama representa uma aplicação com duas áreas principais: a Web App, que inclui serviços e interface do usuário, e a Backend API, que contém a lógica de negócios e acesso aos dados. A Web App se conecta à Backend API para realizar operações, enquanto o backend faz uso de modelos de dados para processar e persistir informações.

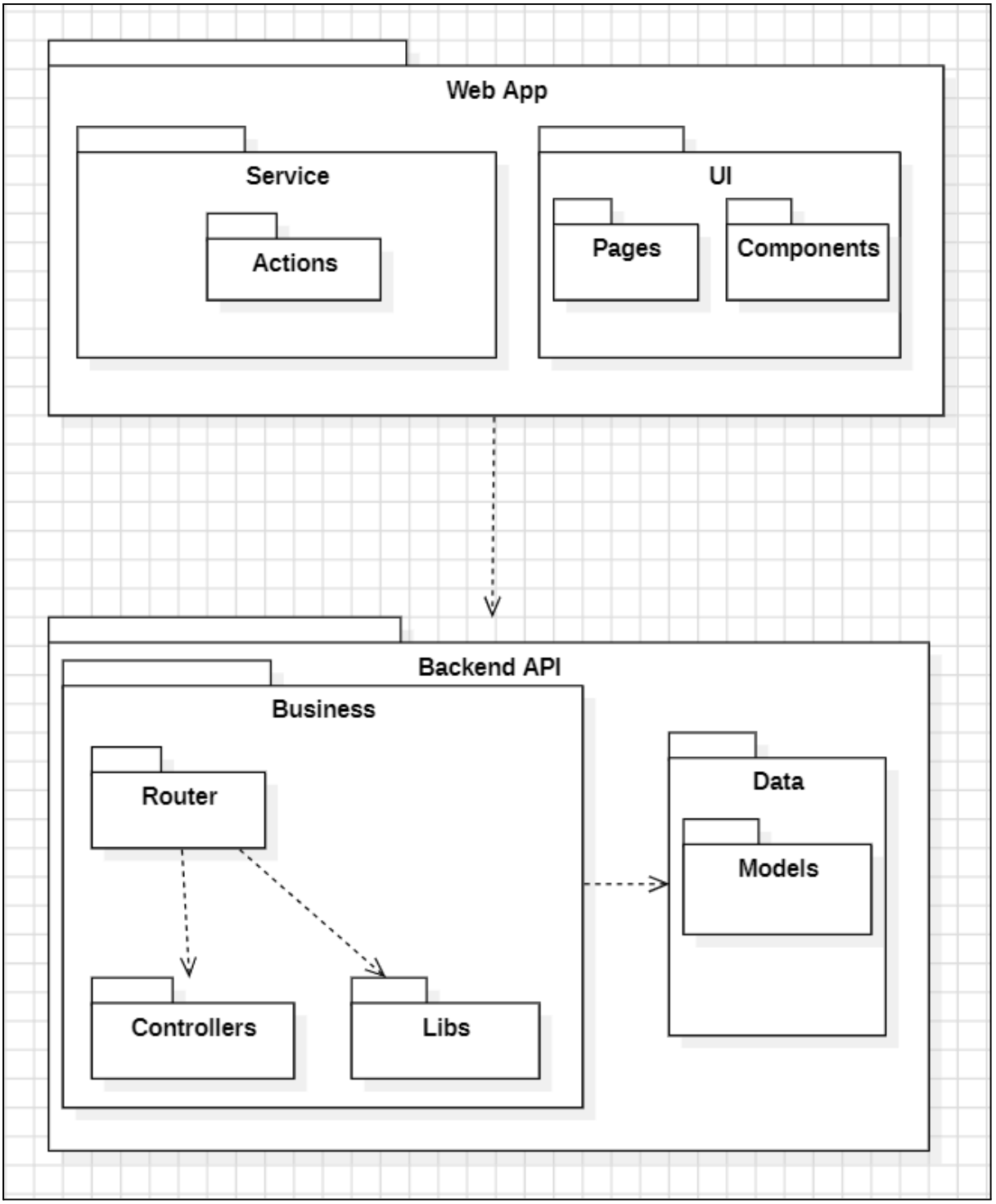

Figura 16. Diagrama de Pacotes

3.5. Diagramas de Estado

A **Figura 17** representa o diagrama de estado do sistema, e demonstra as fases do processo de venda de produtos, começando com a seleção dos itens pelo usuário. Após a seleção, o pedido é criado, e o sistema exibe o valor total. A partir desse ponto, o próximo estado é o de "Aguardando Pagamento", onde o sistema apresenta as opções de pagamento e verifica o status da transação. Quando o pagamento é confirmado, o pedido

muda para o estado de "Pago", momento em que um código QR é gerado e exibido para o cliente. Esse código será essencial para o processo de entrega. O pedido então fica no estado "Aguardando Entrega" até que o código QR seja escaneado pelo vendedor. Quando isso acontece, o pedido é marcado como "Entregue", e o sistema registra e exibe os dados da entrega. Em qualquer momento antes da entrega, o cliente pode solicitar o cancelamento do pedido, o que faz com que o sistema registre a solicitação e mova o pedido para o estado "Cancelado". Dessa forma, o diagrama retrata o ciclo completo do pedido, desde a criação até a entrega ou cancelamento, com transições claras entre cada fase, baseadas em eventos específicos, como a confirmação de pagamento ou o escaneamento do código QR.

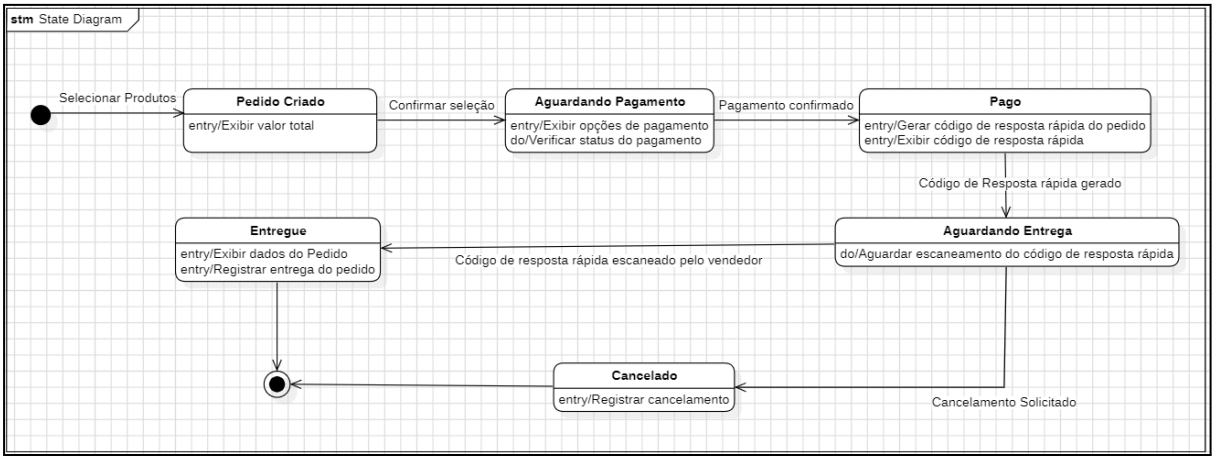

Figura 17. Diagrama de Estado

3.6. Diagramas de Componentes e Implantação

A **Figura 18** representa o diagrama de componentes do sistema, que divide os principais módulos entre a aplicação frontend e a API backend, e ilustrando suas interações.

A Front end App é composta por um App Router, que conecta as Pages e Components da interface de usuário. As Actions são responsáveis por realizar as operações que ocorrem no frontend e se comunicam com o backend via o protocolo HTTP. Essa estrutura sugere uma separação clara entre as rotas da aplicação, páginas e componentes visuais.

A Backend API é composta por um Router, que redireciona as requisições para os Controllers, que, por sua vez, utilizam Models e Libs para acessar e manipular dados. O backend também se comunica com o SGBD (Sistema de Gerenciamento de Banco de Dados) via o protocolo TCP/IP para armazenamento e recuperação de dados.

Além disso, a Backend API também tem uma integração com uma plataforma de processamento de pagamento, utilizando o protocolo HTTP, para lidar com transações financeiras.

Esse diagrama de componentes destaca como o frontend e o backend estão separados e organizados, ao mesmo tempo que interagem por meio de interfaces bem definidas, tanto com o banco de dados quanto com serviços externos, como a plataforma de pagamento.

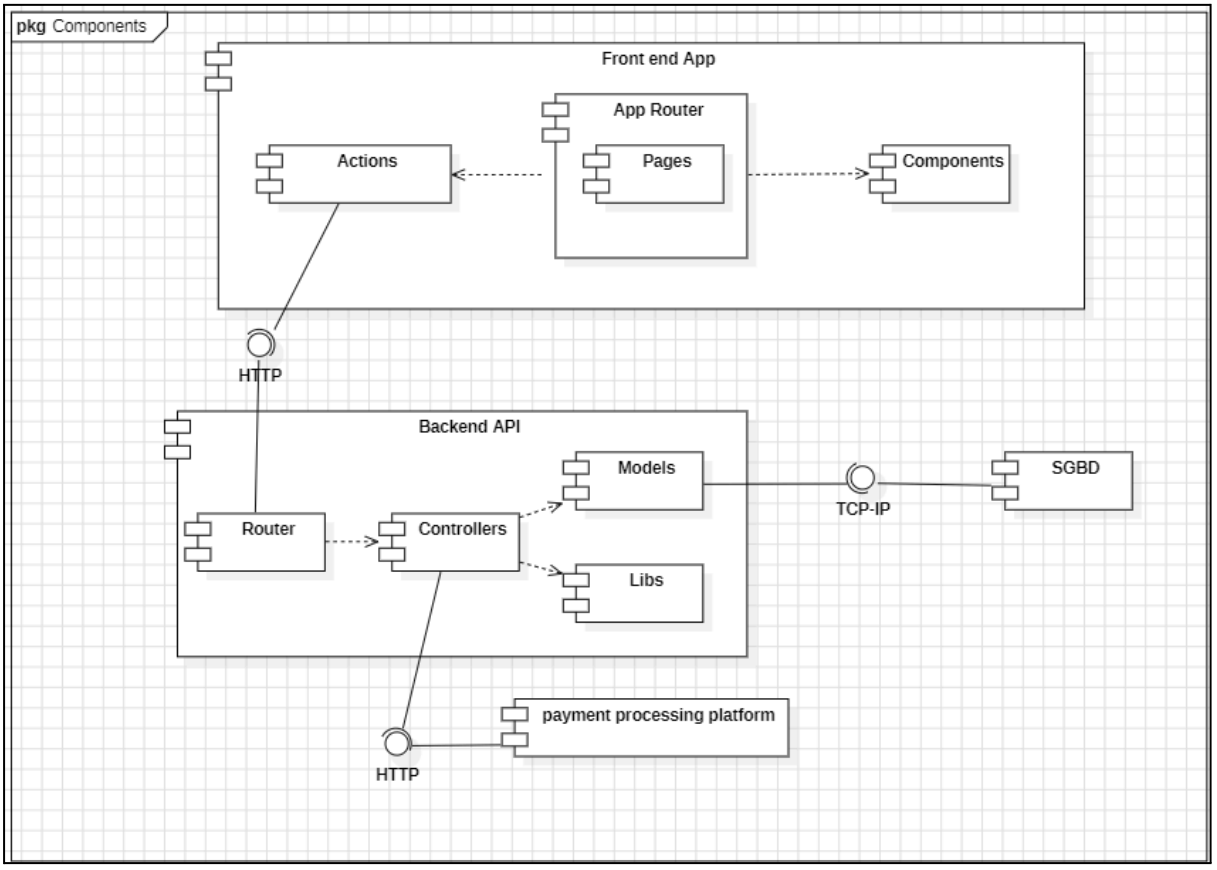

Figura 18. Diagrama de Componentes

A **Figura 19** representa o diagrama de implantação do sistema, que descreve a arquitetura física do sistema distribuído. Ele ilustra a interação entre os dispositivos de usuários, o servidor de aplicação e o banco de dados.

Os usuários acessam o sistema através de navegadores em dispositivos como desktops, Android ou iOS, conectando-se ao servidor de aplicação via protocolo TCP/IP. O servidor de aplicação está hospedado na infraestrutura AWS ECS (Elastic Container Service), onde a Backend API está empacotada e executada dentro de um container Docker.

O servidor de aplicação se comunica com o banco de dados MySQL, que está implantado no serviço de banco de dados AWS RDS, também utilizando o protocolo TCP/IP. Essa configuração garante a comunicação eficiente entre a API e o banco de dados, assim como entre os dispositivos dos usuários e a aplicação no backend.

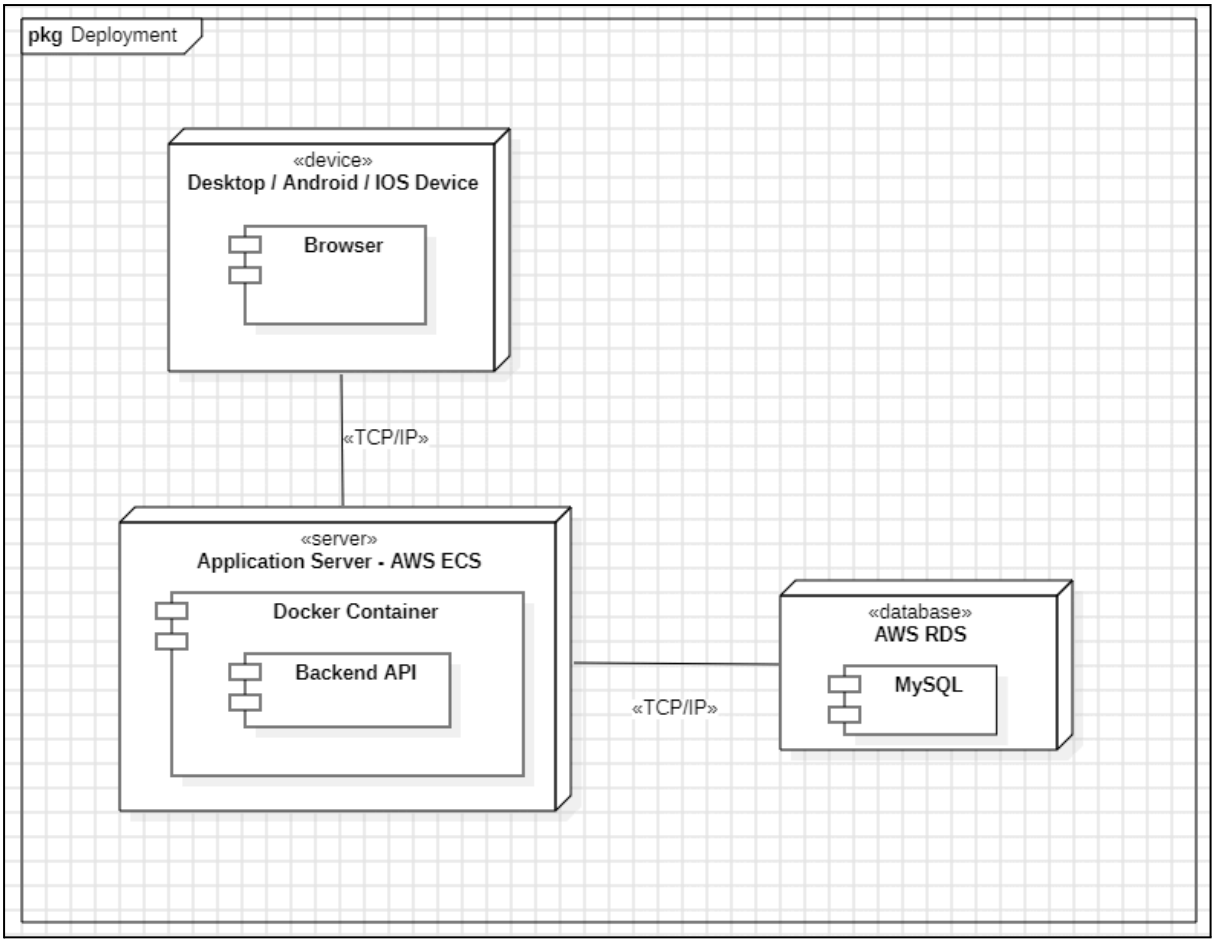

Figura 19. Diagrama de Implantação

4. Projetos de Interface com Usuário

Esta seção tem como objetivo mostrar e descrever as interfaces de interação com o usuário das quais a aplicação é composta. Para isso foi realizado um mockup de alta fidelidade usando a ferramenta Figma. Dessa mesma forma, as interfaces foram relacionadas com os casos de usos especificados na Seção 2.3.1 a fim de mapear todas as funcionalidades necessárias para cumprimento dos requisitos especificados.

4.1. Esboço das Interfaces Usadas Apenas pelo Organizador do Evento

A tela representada na **Figura 20** é usada pelo organizador do evento para se registrar na plataforma. Ela solicita informações como CNPJ, nome de usuário, senha e confirmação de senha. É a etapa inicial para que o organizador possa acessar as funcionalidades de gestão do sistema.

Figura 20. Tela de cadastro do organizador

A tela representada na **Figura 21** exibe e permite a edição de informações do organizador, como dados bancários e permissões de acesso para vendedores. Também serve como ponto de gerenciamento de configurações pessoais.

Figura 21. Tela de perfil do organizador

A tela representada na **Figura 22** oferece ao organizador uma visão geral das métricas do evento, como vendas e desempenho de produtos, “filtráveis” por data. É um painel estratégico para monitorar os resultados em tempo real e tomar decisões durante o evento.

Figura 22. Tela de dashboard do organizador

4.2. Esboço das Interfaces Usadas Apenas pelo Cliente do Evento

A tela representada na **Figura 23** exibe o cardápio do evento acessado pelo cliente através do QR Code. Nela, é possível visualizar os itens disponíveis, seus preços e descrições,

além de selecionar as quantidades desejadas. O cliente utiliza essa interface para montar seu pedido, que será finalizado na tela de checkout.

Figura 23. Tela de evento do cliente

A tela representada na **Figura 24** apresenta o resumo da compra feita pelo cliente, mostrando os itens adicionados ao carrinho, suas quantidades, valores individuais e o total. O cliente preenche informações obrigatórias, como nome, CPF e e-mail, antes de

prosseguir com o pagamento. Essa etapa é essencial no processo de finalização do pedido e antecede a geração do pagamento.

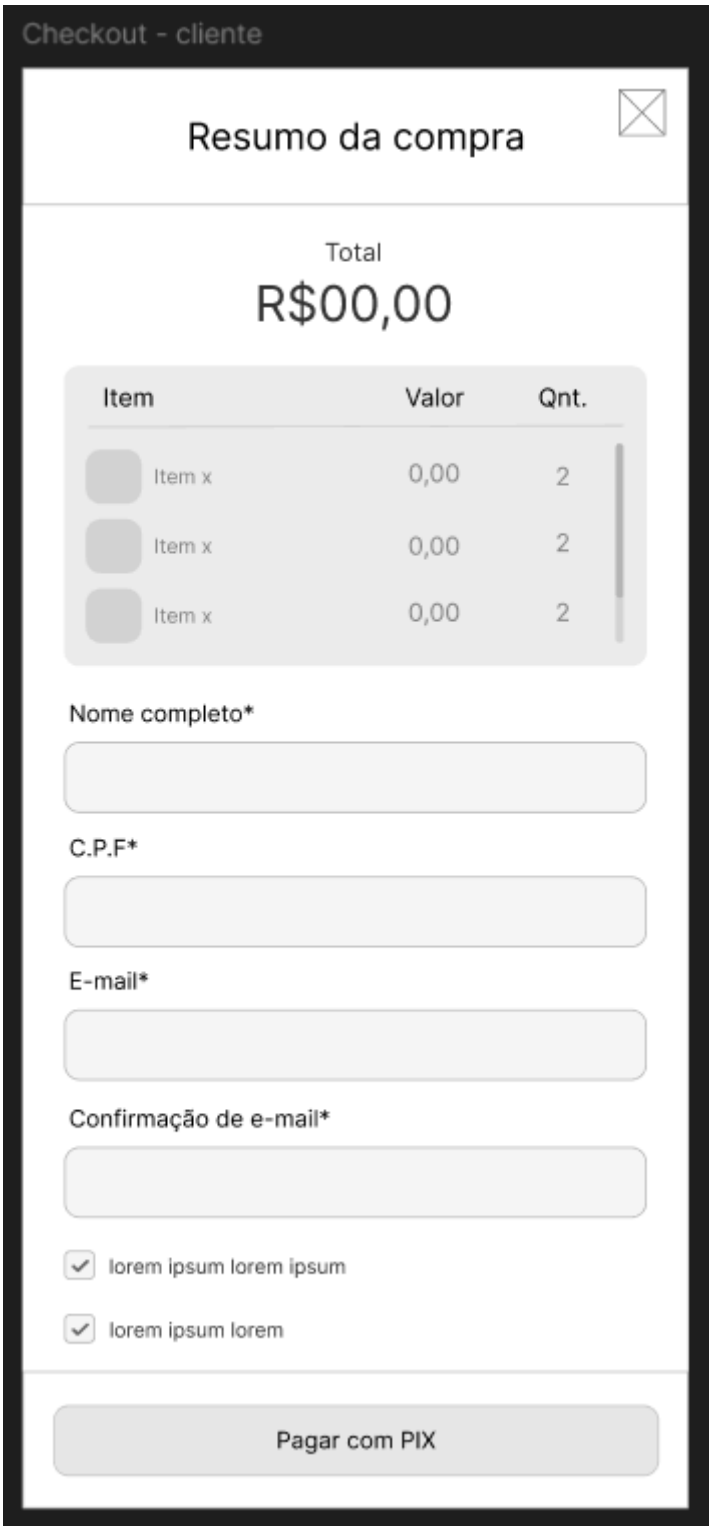

Figura 24. Tela de checkout do cliente

A tela representada na **Figura 25** aparece para o cliente após ele confirmar o pedido. Um QR Code e um código ‘copia e cola’ são gerados para que o cliente realize o pagamento

instantaneamente. Essa etapa é crucial para garantir a segurança e o registro do pagamento.

Figura 25. Tela de pagamento do cliente

A tela representada na **Figura 26** confirma o sucesso do pagamento realizado pelo cliente. Uma mensagem informa que as fichas ou comprovantes foram enviados para o e-mail

cadastrado, permitindo ao cliente acessar seus produtos posteriormente. É o encerramento do processo de compra.

Figura 26. Tela de pagamento processado do cliente

4.3. Esboço das Interfaces Usadas pelo Funcionário do Bar e Organizador do Evento

A tela representada na **Figura 27** é a tela inicial de acesso à plataforma para organizadores e funcionários. Nela, o usuário insere suas credenciais (nome de usuário e senha) para acessar as funcionalidades específicas do sistema.

Figura 27. Tela de login do funcionário e organizador

A tela representada na **Figura 28** permite que organizadores e funcionários visualizem e gerenciem os itens disponíveis no evento. Exibe informações sobre estoque, itens vendidos e restantes, com opções para editar ou adicionar novos produtos.

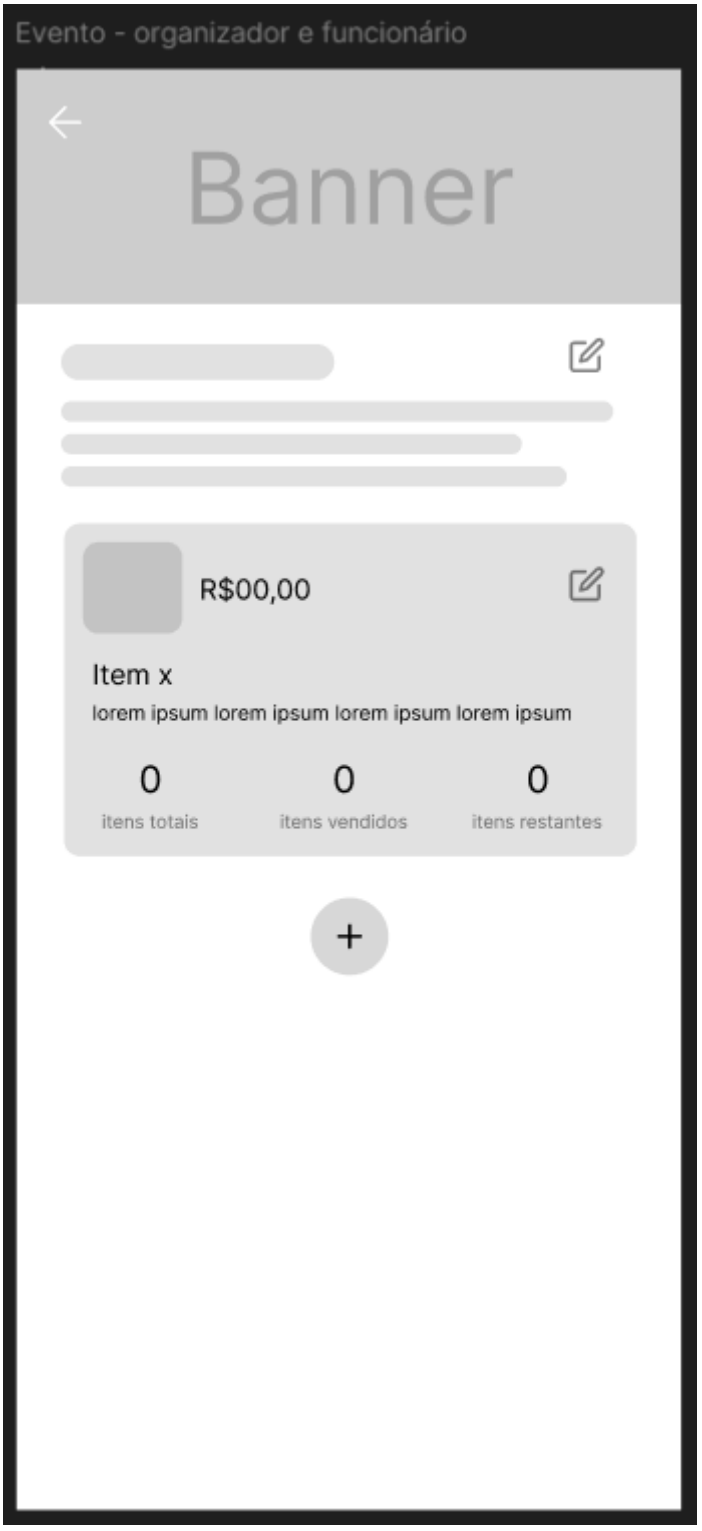

Figura 28. Tela de evento do funcionário e organizador

A tela representada na **Figura 29** lista todos os eventos associados ao organizador ou ao funcionário, com opções para filtrar por período. É o ponto de entrada para acessar e gerenciar os detalhes de cada evento.

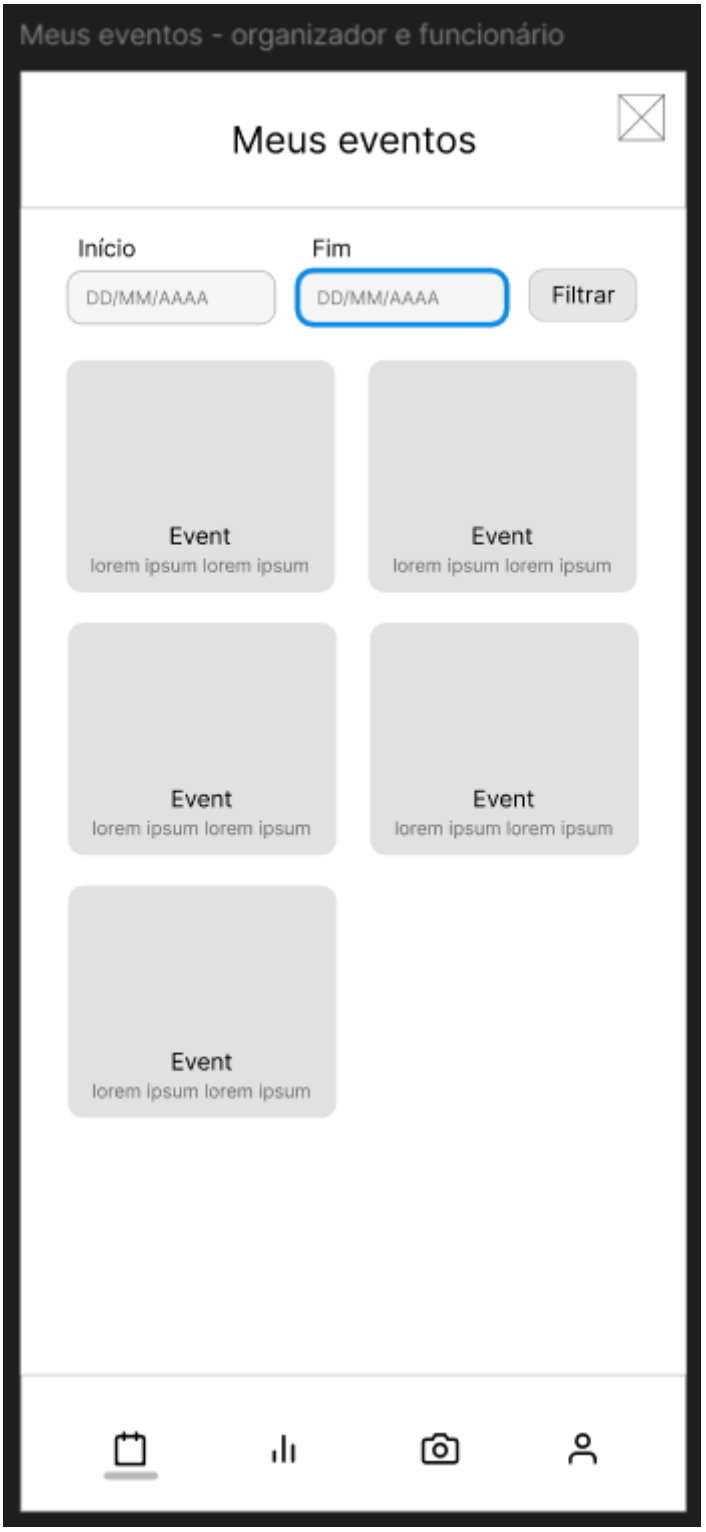

Figura 29. Tela de meus evento do funcionário e organizador

A tela representada na **Figura 30** é utilizada para validar pedidos escaneando os QR Codes apresentados pelos clientes. Após a leitura, os detalhes do pedido são exibidos, permitindo a confirmação rápida da venda.

Figura 30. Tela de scanner do funcionário e organizador

5. Glossário e Modelos de Dados

Esta Seção tem como objetivo descrever os modelos de dados que compõem a aplicação, assim como definir um glossário que permite interpretar diversos conceitos reservados a este projeto. Tendo isso em vista, a **Tabela 14** tem como objetivo definir os diversos atributos que servem de entrada ou saída para aplicação e que são específicos a este projeto. Dessa mesma forma, a **Figura 31** tem como objetivo representar a camada de banco de dados da aplicação por meio de um diagrama entidade relacionamento da aplicação. Nesse diagrama é possível identificar todas as tabelas implementadas, assim como a maneira pela qual elas se relacionam.

Entidade “Usuário”		
Atributo	**Formato**	**Descrição**
admin	booleano	Indica se o usuário é um organizador de eventos ou um funcionário do bar
Entidade “Categoria”		
Atributo	**Formato**	**Descrição**
nome	string	Nome da categoria que um produto pode se encaixar. Ex: energéticos, lanches, cervejas, refrigerantes.
Entidade “Promoção”		
Atributo	**Formato**	**Descrição**
flag_qtd_produtos	string	Indica a quantidade de produtos que poderão ser comprados até que a promoção deixe de existir.

Tabela 14. Glossário de definições do projeto

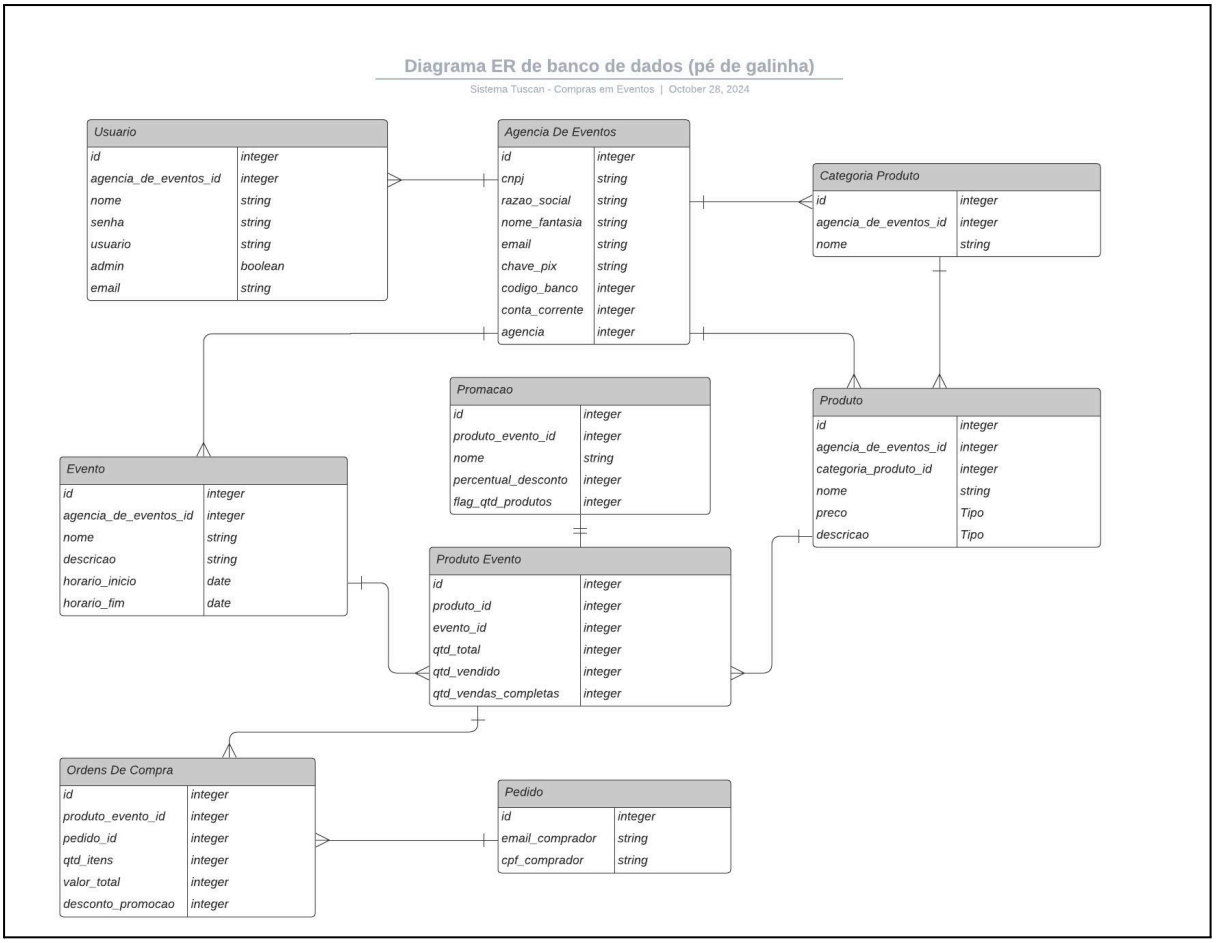

Figura 31. Diagrama de Entidade Relacionamento da Aplicação

6. Casos de Teste

Esta seção tem como objetivo descrever os casos de teste previstos para a aplicação. Na Seção 6.1 são apresentados os testes de aceitação. Esses testes têm como objetivo garantir que o sistema desenvolvido atende as necessidades básicas de seus usuários. Já na Seção 6.2 é apresentado o plano de teste de integração, que tem como objetivo descrever os testes a serem implementados para garantir que os diversos componentes do sistema funcionem da maneira correta juntos.

6.1. Testes de aceitação

Nesta Seção 6.1, são descritos os casos de testes relacionados a aceitação do sistema em relação às necessidades que ele busca suprir. Para isso, cada um das necessidades dos usuários é descrita por meio de casos de teste representados por meio de uma tabela semelhante à representada pela **Tabela 15**. Nela são apresentados quatro campos a serem preenchidos. O primeiro é o identificador único referente ao caso de teste proposto. Já o segundo, descreve uma pré-condição que deve ser atendida para que o caso de teste tenha início. Por fim, o terceiro e quarto campos representam as ações tomadas pelos usuários e quais são os resultados esperados como consequência dessas ações.

Identificador	Teste de aceitação X (TAX)
Pré-condição	Pré-condição para caso de teste X
Ações	1. Ação 1 2. Ação 2
Resultados	Resultados

Tabela 15. Exemplo de caso de teste de aceitação

6.1.1. Necessidade 1 - O cliente do evento deve ser capaz de acessar o cardápio digital

Essa necessidade garante que os clientes possam visualizar o cardápio do evento por meio de QR Codes espalhados pelo local, sem a necessidade de interação com caixas físicos. O objetivo é oferecer um acesso prático, atualizado e acessível às opções de produtos e promoções disponíveis no evento.

Identificador	Acesso ao cardápio via QR Code (TA1-01)
Pré-condição	O cliente deve ter acesso ao evento e um QR Code válido disponível.
Ações	1. O cliente escanear o QR Code usando o celular. 2. O sistema redireciona o cliente para o cardápio digital.
Resultados	O cliente visualiza o cardápio digital com os produtos disponíveis.

Tabela 16. Caso de teste de aceitação “Acesso ao cardápio via QR Code”

Identificador	Exibição de promoções no cardápio (TA1-02)
Pré-condição	O cardápio deve conter promoções cadastradas pelo organizador.
Ações	1. O cliente acessa o cardápio digital. 2. O cliente visualiza as promoções destacadas no topo da página.
Resultados	O cliente identifica os produtos em promoção com os respectivos descontos.

Tabela 17. Caso de teste de aceitação “Exibição de promoções no cardápio”

Identificador	Atualização automática do cardápio (TA1-03)
Pré-condição	O organizador altera o estoque ou preços de produtos.

Ações	1. O cliente acessa o cardápio digital. 2. O cliente confirma que os dados atualizados aparecem em tempo real.
Resultados	O cardápio exibe informações atualizadas sem precisar de uma nova requisição.

Tabela 18. Caso de teste de aceitação "Atualização automática do cardápio"

Identificador	Acessibilidade do cardápio para clientes sem login (TA1-04)
Pré-condição	O QR Code deve estar ativo e configurado no evento.
Ações	1. O cliente escanear o QR Code sem precisar estar logado. 2. O cliente visualiza o cardápio completo.
Resultados	O cliente acessa o cardápio sem restrições.

Tabela 19. Caso de teste de aceitação "Acessibilidade do cardápio para clientes sem login"

6.1.2. Necessidade 2 - O cliente deve ser capaz de realizar uma compra online

Essa necessidade assegura que os clientes possam adicionar produtos ao carrinho, remover itens, finalizar a compra e realizar o pagamento de maneira segura e prática, utilizando Pix ou cartão de crédito. Além disso, os QR Codes gerados para retirada de produtos são enviados por e-mail.

Identificador	Adicionar itens ao carrinho (TA2-01)
Pré-condição	O cliente deve ter acesso ao cardápio digital.
Ações	1. O cliente seleciona um item no cardápio. 2. O cliente adiciona o item ao carrinho.
Resultados	O item é adicionado com quantidade e preço corretos.

Tabela 20. Caso de teste de aceitação "Adicionar itens ao carrinho"

Identificador	Remover itens do carrinho (TA2-02)
Pré-condição	O carrinho deve conter itens adicionados.
Ações	1. O cliente acessa o carrinho. 2. O cliente remove um item previamente adicionado.
Resultados	O item é removido, e o total é atualizado.

Tabela 21. Caso de teste de aceitação "Remover itens do carrinho"

Identificador	Finalização de compra (TA2-03)
Pré-condição	O carrinho deve conter itens e um método de pagamento configurado.
Ações	1. O cliente seleciona “Finalizar compra”. 2. O cliente realiza o pagamento via Pix ou cartão de crédito.
Resultados	O pagamento é processado, e os QR Codes são gerados.

Tabela 22. Caso de teste de aceitação “Finalização de compra”

Identificador	Recebimento de QR Codes por e-mail (TA2-04)
Pré-condição	O pagamento deve ser confirmado.
Ações	1. O cliente finaliza a compra. 2. O cliente verifica o e-mail associado para recebimento dos QR Codes.
Resultados	Os QR Codes são entregues ao e-mail com informações dos produtos comprados.

Tabela 23. Caso de teste de aceitação “Recebimento de QR Codes por e-mail”

6.1.3. Necessidade 3 - O organizador deve ser capaz de gerenciar produtos

Essa necessidade foca na administração de produtos pelo organizador do evento. Ela inclui o cadastro, edição, exclusão e consulta de produtos disponíveis para o cardápio, garantindo que as informações estejam sempre atualizadas e corretas para os clientes.

Identificador	Cadastro de novos produtos (TA3-01)
Pré-condição	O organizador deve estar autenticado no sistema.
Ações	1. O organizador acessa a área de produtos. 2. O organizador insere informações sobre um novo produto (nome, preço, estoque).
Resultados	O produto é cadastrado com sucesso

Tabela 24. Caso de teste de aceitação “Cadastro de novos produtos”

Identificador	Edição de produtos existentes (TA3-02)
Pré-condição	Um produto deve estar previamente cadastrado.
Ações	1. O organizador acessa a área de produtos. 2. O organizador atualiza informações como preço ou

	quantidade de estoque.
Resultados	As alterações são salvas e refletidas no sistema.

Tabela 25. Caso de teste de aceitação "Edição de produtos existentes"

Identificador	Exclusão de produtos (TA3-03)
Pré-condição	Um produto deve estar cadastrado.
Ações	1. O organizador seleciona um produto existente. 2. O organizador remove o produto do sistema.
Resultados	O produto é excluído do banco de dados.

Tabela 26. Caso de teste de aceitação "Exclusão de produtos"

Identificador	Consulta de produtos cadastrados (TA3-04)
Pré-condição	O sistema deve conter produtos cadastrados.
Ações	1. O organizador acessa a área de consulta de produtos. 2. O organizador visualiza uma lista completa de todos os produtos.
Resultados	A lista de produtos é exibida com informações atualizadas.

Tabela 27. Caso de teste de aceitação "Consulta de produtos cadastrados"

6.1.4. Necessidade 4 - O organizador deve ser capaz de criar promoções

Essa necessidade garante que o organizador possa configurar promoções para atrair clientes, definir descontos em produtos específicos, editar regras promocionais e avaliar o impacto das promoções nas vendas por meio de relatórios.

Identificador	Cadastro de promoções (TA4-01)
Pré-condição	O organizador deve estar autenticado no sistema.
Ações	1. O organizador acessa a área de promoções. 2. O organizador define regras como produtos aplicáveis e descontos.
Resultados	A promoção é criada e aparece no cardápio.

Tabela 28. Caso de teste de aceitação "Cadastro de promoções"

Identificador	Edição de promoções (TA4-02)

Pré-condição	Deve existir uma promoção cadastrada.
Ações	1. O organizador acessa a área de promoções. 2. O organizador altera valores de desconto ou validade da promoção.
Resultados	A promoção é atualizada com sucesso.

Tabela 29. Caso de teste de aceitação "Edição de promoções"

Identificador	Exclusão de promoções (TA4-03)
Pré-condição	Deve existir uma promoção cadastrada.
Ações	1. O organizador acessa a área de promoções. 2. O organizador remove uma promoção existente.
Resultados	A promoção é excluída do sistema.

Tabela 30. Caso de teste de aceitação "Exclusão de promoções"

Identificador	Visualização de promoções aplicadas (TA4-04)
Pré-condição	Promoções devem estar cadastradas.
Ações	1. O organizador acessa o painel de relatórios. 2. O organizador verifica o impacto das promoções em vendas.
Resultados	O relatório exibe dados de vendas com promoções aplicadas.

Tabela 31. Caso de teste de aceitação "Visualização de promoções aplicadas"

6.1.5. Necessidade 5 - O funcionário do bar deve validar QR Codes

Essa necessidade assegura que os funcionários do bar possam escanear e validar QR Codes apresentados pelos clientes, confirmando a autenticidade e liberando os produtos comprados. Também inclui a rejeição de QR Codes inválidos e o registro de validações para controle e auditoria.

Identificador	Validação de QR Code (TA5-01)
Pré-condição	O cliente apresenta um QR Code válido.
Ações	1. O funcionário escaneia o QR Code. 2. O funcionário confirma a validade e o item associado.
Resultados	O QR Code é validado, e o produto é liberado.

Tabela 32. Caso de teste de aceitação "Validação de QR Code"

Identificador	Rejeição de QR Code inválido (TA5-02)
Pré-condição	O cliente apresenta um QR Code inválido ou expirado.
Ações	1. O funcionário escaneia o QR Code. 2. O funcionário recebe uma notificação de erro.
Resultados	O sistema alerta que o QR Code não é válido.

Tabela 33. Caso de teste de aceitação “Rejeição de QR Code inválido”

Identificador	Registro de validações (TA5-03)
Pré-condição	O sistema deve armazenar logs de validações de QR Codes.
Ações	1. O funcionário escaneia o QR Code. 2. O sistema registra a validação em logs.
Resultados	A validação é registrada para auditoria.

Tabela 34. Caso de teste de aceitação “Registro de validações”

Identificador	Consulta de QR Codes validados (TA5-04)
Pré-condição	QR Codes devem ter sido validados anteriormente.
Ações	1. O funcionário acessa o histórico de validações no sistema. 2. O funcionário consulta os detalhes dos QR Codes já validados, incluindo o horário da validação e os itens associados.
Resultados	O sistema exibe uma lista detalhada de QR Codes validados com todas as informações relevantes, permitindo controle e auditoria.

Tabela 35. Caso de teste de aceitação “Consulta de QR Codes validados”

6.2. Testes de integração

Nesta seção, são descritos os testes de integração propostos para o sistema implementado. Dentro desses testes são representadas interações com aplicações consideradas internas do sistema, essas sendo a Frontend App e a Backend API, e aplicações externas, como a plataforma para processamento de pagamentos. Para cada caso de teste é usado um modelo de representação tal como o descrito na **Tabela 36**, esse que representa quais são os sistemas envolvidos, de que maneira eles se comunicam e quais entradas e saídas são esperadas para o teste.

Identificador	Teste de aceitação X (TAX)

Sistemas Envolvidos	Sistema 1 e Sistema 2
Interface	Requisição HTTP
Pré-condição	Pré-condição para caso de teste X
Ações	3. Ação 1 4. Ação 2
Resultados	Resultados esperados

Tabela 36. Exemplo de caso de teste de integração

Como estratégia para realização dos testes de integração é proposta uma abordagem “bottom-up”, que é adequada especialmente devido à sua arquitetura modular. O backend e sua integração com o banco de dados e serviços externos (Stripe) são cruciais para a funcionalidade principal do sistema. Portanto, erros nesta camada podem afetar diretamente todas as outras. Testar primeiro as fundações (backend e banco de dados) irá garantir que as dependências mais importantes estejam funcionando antes de integrar o frontend, reduzindo o esforço de debugging na camada de interface visual, já que erros na base são corrigidos de antemão.

Estratégia:

1. Iniciar com testes unitários e de integração no backend.
2. Subir gradualmente para o frontend, onde serão realizados testes de integração com a API e, finalmente, testes end-to-end.
3. Validar interações com sistemas externos, como o Stripe, com testes de integração específicos.

Para automatização do processo de teste:

1. Backend API: Jest:
2. Frontend Mobile: React Testing Library e Jest.
3. Plataformas de Pagamento: Stripe CLI.

Para automatizar a execução dos testes, será utilizado o Github Actions, que permite a construção de uma pipeline de integração contínua no próprio Github. Para sua execução, serão utilizados mocks para a base de dados, de forma a evitar a necessidade de implantação de um segundo banco de dados e acelerar a execução dos testes.

Na **Tabela 37** é apresentado um caso de teste que tem como objetivo verificar a criação de eventos pelo organizador através da API. O teste valida se a requisição HTTP para o endpoint de criação retorna o status correto e se os dados do evento são armazenados corretamente no banco de dados.

Identificador	Teste de Integração 01 (TI-01)
Sistemas Envolvidos	Frontend App e Backend API
Interface	Requisição HTTP (POST)
Pré-condição	O usuário organizador deve estar autenticado no sistema.
Ações	1. Realizar uma requisição HTTP para o endpoint /events com os dados do evento (nome, data, local). 2. Validar a resposta do backend com status 201 (Created).
Resultados	1. Evento criado com sucesso no banco de dados. 2. Retorno do ID do evento criado no payload da resposta.

Tabela 37. Caso de teste de integração 1: Criação de evento pelo organizador via API.

Na **Tabela 38** é apresentado um caso de teste que tem como objetivo validar a funcionalidade de listagem de eventos cadastrados. O teste garante que o backend retorne todos os eventos existentes no banco de dados em formato JSON, com os atributos esperados, quando solicitado pelo frontend.

Identificador	Teste de Integração 02 (TI-02)
Sistemas Envolvidos	Frontend App e Backend API
Interface	Requisição HTTP (GET)
Pré-condição	Deve existir pelo menos um evento cadastrado no banco de dados.
Ações	1. Realizar uma requisição HTTP para o endpoint /events para listar todos os eventos. 2. Validar o retorno da lista de eventos com os atributos. esperados (nome, data, local, ID).
Resultados	1. Resposta com status 200 (OK). 2. Retorno do JSON contendo a lista de eventos cadastrados.

Tabela 38. Caso de teste de integração 2: Listagem de eventos cadastrados no sistema.

Na **Tabela 39** é apresentado um caso de teste que tem como objetivo garantir a integração correta entre a API do sistema e a plataforma Stripe para o processamento de pagamentos. O teste avalia que os dados do carrinho de compras são enviados adequadamente e que a confirmação do pagamento é recebida e processada.

Identificador	Teste de Integração 03 (TI-03)
Sistemas Envolvidos	Backend API e Plataforma de Pagamentos (Stripe)

Interface	Requisição HTTP (POST)
Pré-condição	O carrinho de compras deve estar preenchido com produtos e valores corretos no frontend.
Ações	1. Enviar uma requisição ao backend para o endpoint /checkout contendo os dados do carrinho e forma de pagamento (Pix ou cartão de crédito). 2. O backend deve encaminhar a requisição à API da plataforma de pagamentos (Stripe).
Resultados	1. Resposta com status 200 (OK) e confirmação do pagamento. 2. Retorno do comprovante de pagamento e QR Codes associados.

Tabela 39. Caso de teste de integração 3: Processamento de pagamento com a plataforma Stripe.

Na **Tabela 40** é apresentado um caso de teste que tem como objetivo verificar a atualização do estoque de produtos no banco de dados pelo organizador. O teste valida se a API recebe os dados de entrada ou saída corretamente e os reflete na base de dados após o processamento.

Identificador	Teste de Integração 04 (TI-04)
Sistemas Envolvidos	Backend API e Banco de Dados (SGBD)
Interface	Conexão TCP-IP
Pré-condição	Produto já cadastrado no banco de dados.
Ações	1. Atualizar o estoque de um produto pelo endpoint /products/{id}/stock enviando a quantidade de entrada ou saída. 2. Verificar se o backend realiza a atualização do estoque no banco de dados.
Resultados	1. Resposta com status 200 (OK). 2. Estoque atualizado corretamente no banco de dados.

Tabela 40. Caso de teste de integração 4: Atualização de estoque de produtos pelo organizador.

Na **Tabela 41** é apresentado um caso de teste que tem como objetivo validar a funcionalidade de obtenção de QR Codes associados a um pedido realizado. O teste garante que o backend retorne os QR Codes e detalhes dos produtos comprados em resposta a uma solicitação do frontend.

Identificador	Teste de Integração 05 (TI-05)
Sistemas Envolvidos	Backend API e Frontend App

Interface	Requisição HTTP (GET)
Pré-condição	QR Codes de produtos devem estar gerados após uma compra.
Ações	1. O frontend realiza uma requisição ao endpoint /orders/{id}/qr-codes para buscar os QR Codes da compra. 2. O backend retorna os QR Codes associados ao pedido.
Resultados	1. Resposta com status 200 (OK). 2. Retorno de um JSON com os QR Codes e detalhes dos produtos comprados.

Tabela 41. Caso de teste de integração 5: Obtenção de QR Codes associados a um pedido.

7. Cronograma e Processo de Implementação

Nesta seção, são apresentados o cronograma de desenvolvimento da aplicação e os detalhes do processo de implementação do projeto. A Seção 7.1 contém o cronograma de atividades, detalhando as tarefas a serem realizadas em cada quinzena entre fevereiro/2025 e junho/2025. Por se tratar de um trabalho realizado em dupla, são apresentados cronogramas separados para cada um dos alunos (Eric e Bernardo), com tarefas complementares. Já a Seção 7.2 descreve o processo de implementação adotado no desenvolvimento.

7.1. Cronograma

O desenvolvimento do projeto ocorre ao longo de aproximadamente 4 meses e meio, entre a primeira quinzena de fevereiro/2025 e a primeira quinzena de junho/2025, organizados em sprints quinzenais. Cada integrante da equipe é responsável por tarefas específicas. **As tabelas a seguir (42 e 43)** apresentam os cronogramas individuais de cada um dos alunos.

Cronograma - Eric Jardim	
Quinzena	**Atividades**
1ª Fevereiro (01-15)	● Configuração inicial do backend: estrutura de pastas, variáveis de ambiente e instalação de dependências. ● Implementação das tabelas no banco de dados e seus respectivos modelos na aplicação. ● Configuração inicial do repositório Github e Docker.
2ª Fevereiro (16-28)	● Implementação do fluxo de cadastro e autenticação de usuários. ● Desenvolvimento da API de CRUD (criar, ler, atualizar e remover) das entidades da aplicação. ● Correções e ajustes.

1ª Março (01-15)	● Implementação da lógica para geração de QR codes vinculados aos pedidos ● Correções e ajustes.
2ª Março (16-31)	● Desenvolvimento do fluxo de pedidos: APIs para criação, alteração de status e cancelamento. ● Integração inicial com o gateway de pagamento (Stripe) para processar pagamentos e associá-los aos pedidos. ● Correções e ajustes.
1ª Abril (01-15)	● Implementação de relatórios: API para relatórios de vendas diárias e totais. ● Correções e ajustes.
2ª Abril (16-30)	● Testes automatizados: criação de testes unitários e de integração para as APIs. ● Correções e ajustes.
1ª Maio (01-15)	● Implementação da funcionalidade de notificações em tempo real usando WebSockets. ● Correções e ajustes.
2ª Maio (16-31)	● Desenvolvimento de funcionalidades administrativas: gestão de usuários e permissões de acesso. ● Correções e ajustes.
1ª Junho (01-15)	● Configuração do ambiente de produção, deploy da API na AWS, e monitoramento inicial do sistema.

Tabela 42. Cronograma Eric Jardim

Cronograma - Bernardo Rohlfs	
Quinzena	**Atividades**
1ª Fevereiro (01-15)	● Configuração inicial do frontend: estrutura de pastas e instalação de dependências. ● Desenvolvimento das primeiras telas no frontend: Login e CRUD (criar, ler, atualizar e remover) de entidades.
2ª Fevereiro (16-28)	● Integração dos cadastros e listagens com o backend ● Implementação do fluxo de autenticação no frontend. ● Correções e ajustes.
1ª Março (01-15)	● Implementação da tela de QR code e desenvolvimento da lógica de validação de QR codes no frontend ● Correções e ajustes.

2ª Março (16-31)	● Desenvolvimento da interface do carrinho de compras. ● Implementação da página de resumo do pedido e pagamento. ● Correções e ajustes.
1ª Abril (01-15)	● Criação das telas de relatórios. ● Correções e ajustes.
2ª Abril (16-30)	● Validação de usabilidade e testes unitários para os principais componentes do frontend. ● Correções e ajustes.
1ª Maio (01-15)	● Integração da interface de pedidos com WebSockets para atualizações em tempo real. ● Correções e ajustes.
2ª Maio (16-31)	● Integração completa do frontend com as funcionalidades de administração (gestão de usuários e permissões) ● Correções e ajustes.
1ª Junho (01-15)	● Deploy do frontend em produção e validação do fluxo completo junto com o backend.

Tabela 43. Cronograma Bernardo Rohlfs

7.2. Processo de Implementação

Esta seção tem como objetivo detalhar o processo adotado para o desenvolvimento do sistema, explicando as etapas que serão seguidas ao longo do projeto. A abordagem escolhida é o desenvolvimento incremental, utilizando ciclos de trabalho com duração média de 15 dias, conforme descrito na Seção 7.1.

Cada ciclo, ou sprint, terá tarefas específicas e feedback contínuo para garantir a evolução do projeto. Na primeira Sprint, o foco estará na configuração do ambiente e criação da base do sistema, sem a implementação direta de funcionalidades finais. Após isso, as sprints seguintes se concentram na implementação de fluxos específicos e ajustes baseados nos retornos obtidos.

Práticas Adotadas

1. Controle de Tarefas: Uso do GitHub Projects para organizar e acompanhar o progresso das tarefas.
2. Controle de Versão: Uso do Git e GitHub para versionamento e colaboração.
3. CI/CD: Automação de testes e deploy usando GitHub Actions.
4. Testes Automatizados: Testes de unidade e integração para backend e frontend, garantindo a estabilidade do sistema.

Artefatos Produzidos

1. Código-fonte: Inclui backend e frontend com testes automatizados.
2. Documentação: README detalhado com instruções de instalação e uso.

Com a abordagem adotada, o frontend e backend serão desenvolvidos simultaneamente, garantindo a entrega de um sistema completo e funcional ao final do cronograma.

8. Post-Mortem

(…)