Documentação de Projeto

para o sistema

Tuscan - Compras em Eventos

Versão 9.0

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 João Paulo Carneiro Aramuni.

30/06/2025

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
Atualização 01/TCC 02	04/04/2025	Revisão e correção de diagramas e conceitos de arquitetura.	6.0
Atualização 02/TCC 02	20/04/2025	Revisão e correção dos casos de teste.	7.0
Atualização 03/TCC 02	18/05/2025	Complementação da seção ‘Arquitetura’.	8.0
Correções pós-banca	30/06/2025	Atualização do Post Mortem e inclusão de novos diagramas de sequência.	9.0

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.

1.1. Contextualização

A venda de alimentos e bebidas em eventos de médio e grande porte, como shows, festivais e festas universitárias, ainda depende amplamente de caixas físicos, totens ou máquinas de autoatendimento. Essas soluções, embora tenham trazido avanços em relação ao modelo tradicional, apresentam uma série de limitações operacionais: filas demoradas, alto custo de infraestrutura, necessidade de treinamento de equipes, risco de falhas humanas e baixa adaptabilidade a ambientes externos. Em paralelo, o público consumidor moderno exige mais autonomia, agilidade e meios de pagamento digitais (expectativas que os modelos atuais nem sempre conseguem atender de forma eficaz).

Por outro lado, os organizadores de eventos enfrentam desafios significativos na gestão de vendas e estoque em tempo real. A ausência de ferramentas flexíveis e digitais limita a capacidade de tomar decisões rápidas e de oferecer uma experiência mais fluida ao

público. Isso resulta em perda de vendas, insatisfação do consumidor e complexidade na operação do evento.

Nesse contexto, surge a proposta do Tuscan, um sistema web e mobile que visa simplificar e modernizar o processo de venda de produtos em eventos, eliminando a necessidade de caixas físicos e fichas de papel. A solução é baseada em cardápios digitais acessados por QR Code, pagamentos online via Pix ou cartão de crédito, e fichas digitais (em forma de QR Code) que são validadas por funcionários através de um aplicativo móvel. Além disso, o sistema fornece aos organizadores ferramentas para gestão de estoque e vendas em tempo real.

Com arquitetura baseada em tecnologias modernas, como Next.js, Supabase e Stripe, o Tuscan é escalável, seguro e de fácil adoção. Ele atende às exigências de segurança de dados da LGPD, evita o armazenamento de informações sensíveis e é ideal para eventos que buscam autonomia, eficiência e experiência digital para o consumidor final.

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 consumidores do evento, para validar as vendas de alimentos e bebidas durante eventos, com a finalidade de entregar os produtos comprados aos seus destinatários com praticidade e agilidade.

Consumidor 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 seu perfil e 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 consumidores 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 consumidores do evento, para validar as vendas de alimentos e bebidas durante os 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 consumidores 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 consumidores.
Dores	● Irritação com consumidores rudes e arrogantes que desrespeitam seu trabalho. ● Falta de praticidade e agilidade durante o processo de aceite e entrega de pedidos.
Objetivos	● Atender consumidores 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 de eventos, e usa o Sistema Tuscan para acessar o cardápio do evento que está frequentando e consumir 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 consumidor do evento, 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.

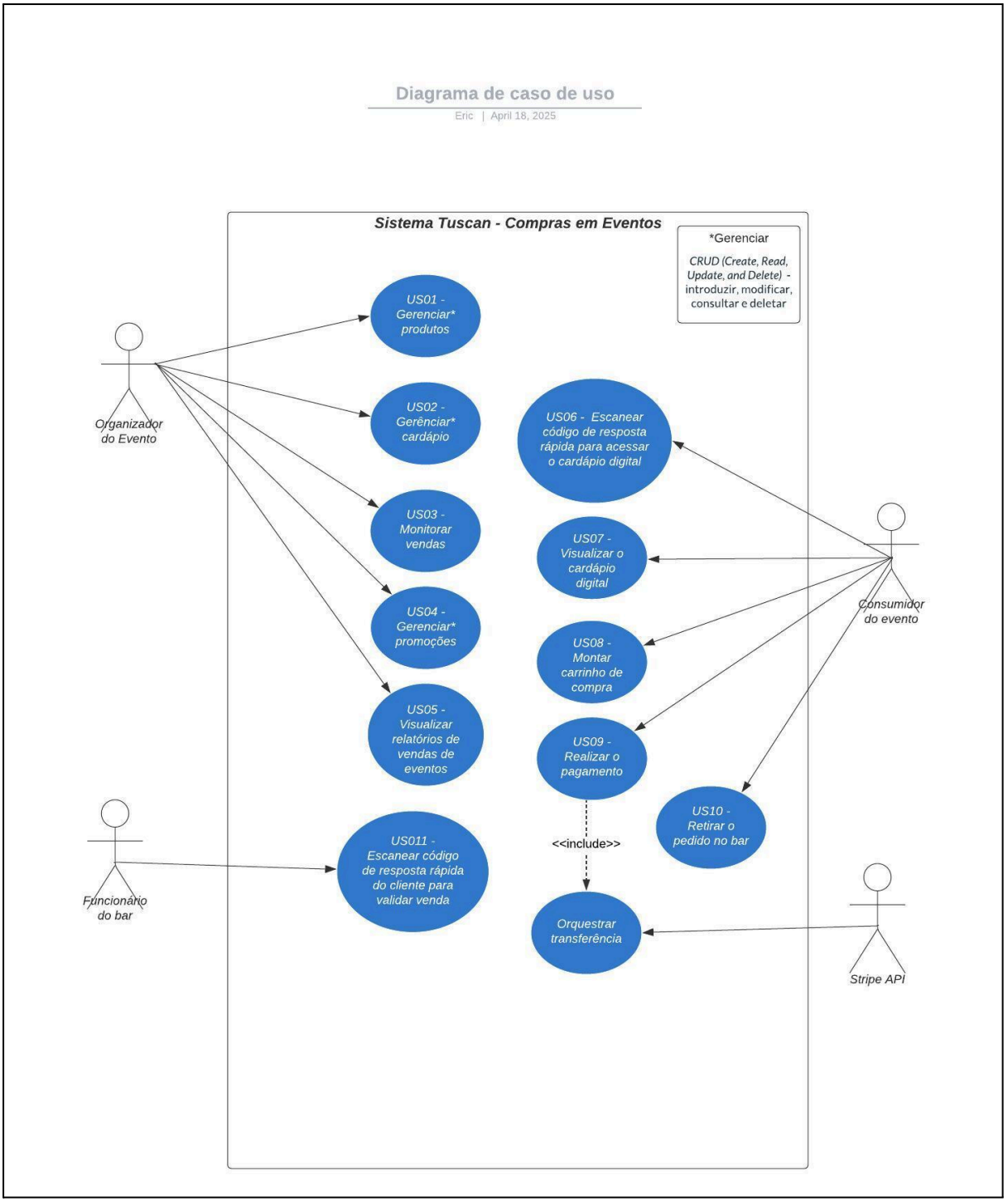

Figura 1. Casos de Uso

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 consumidores 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.
US 03	Como organizador do evento, gostaria de visualizar as vendas, 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 consumidores durante o evento.
US 05	Como organizador do evento, gostaria de visualizar relatórios de vendas do evento, para analisar o comportamento dos consumidores e melhorar a oferta de produtos em eventos futuros.
Consumidor do evento	
US 06	Como consumidor 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 consumidor do evento, gostaria de visualizar o cardápio digital com todos os produtos e promoções disponíveis, para escolher o que vou comprar.
US 08	Como consumidor do evento, gostaria de montar meu carrinho de compra, para efetuar a compra de vários produtos de uma só vez.
US 09	Como consumidor do evento, gostaria de realizar o pagamento online usando Pix ou cartão de crédito, para evitar filas e agilizar a compra.
US 10	Como consumidor 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 consumidores 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 irão compor os cardápios digitais. Este fluxo se relaciona diretamente à história de usuário US01.

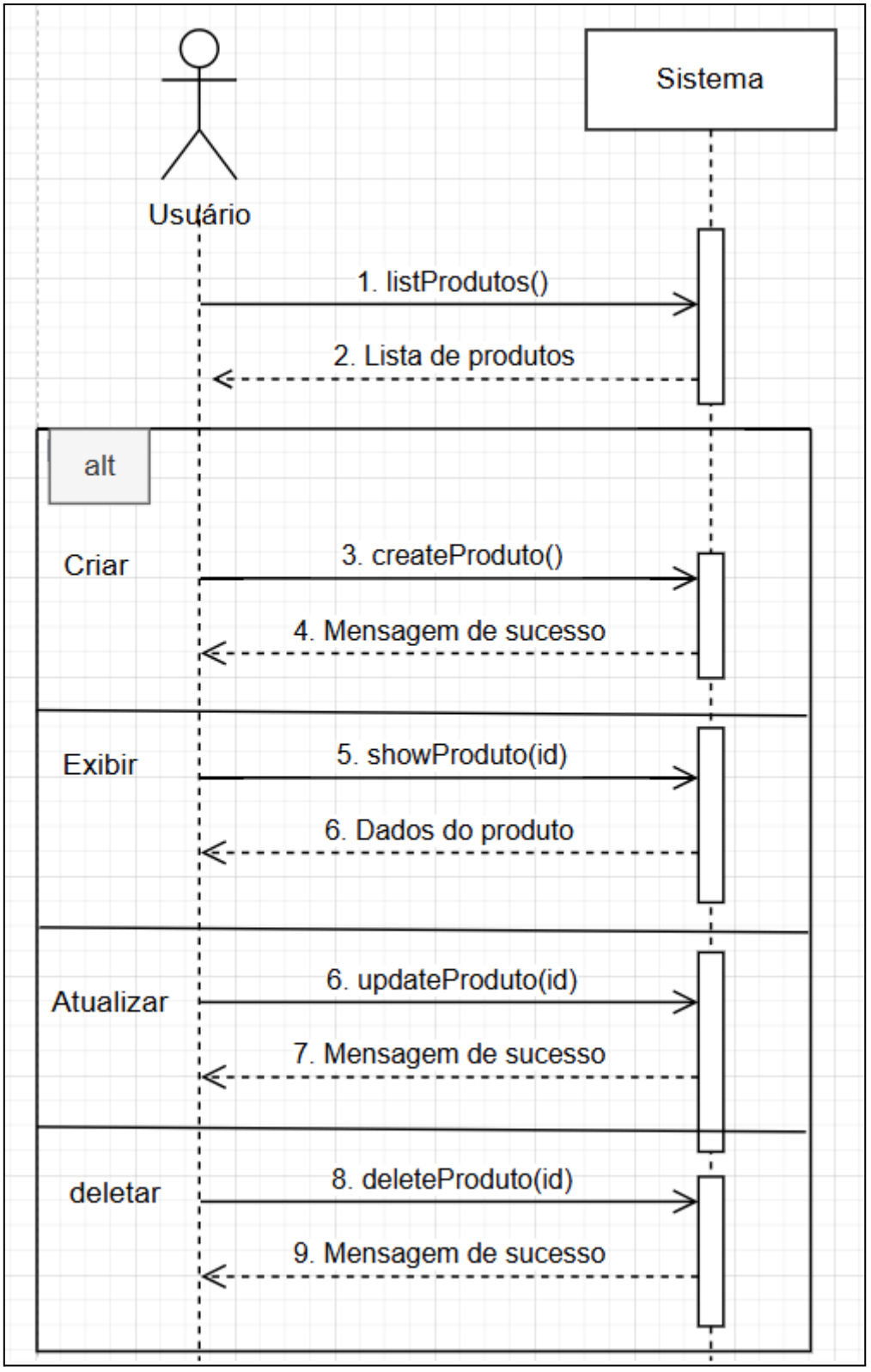

Figura 2. DSS - US 1 Gerenciar produtos

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 para refletir a disponibilidade atualizada durante o evento. Este diagrama está vinculado à história de usuário US02.

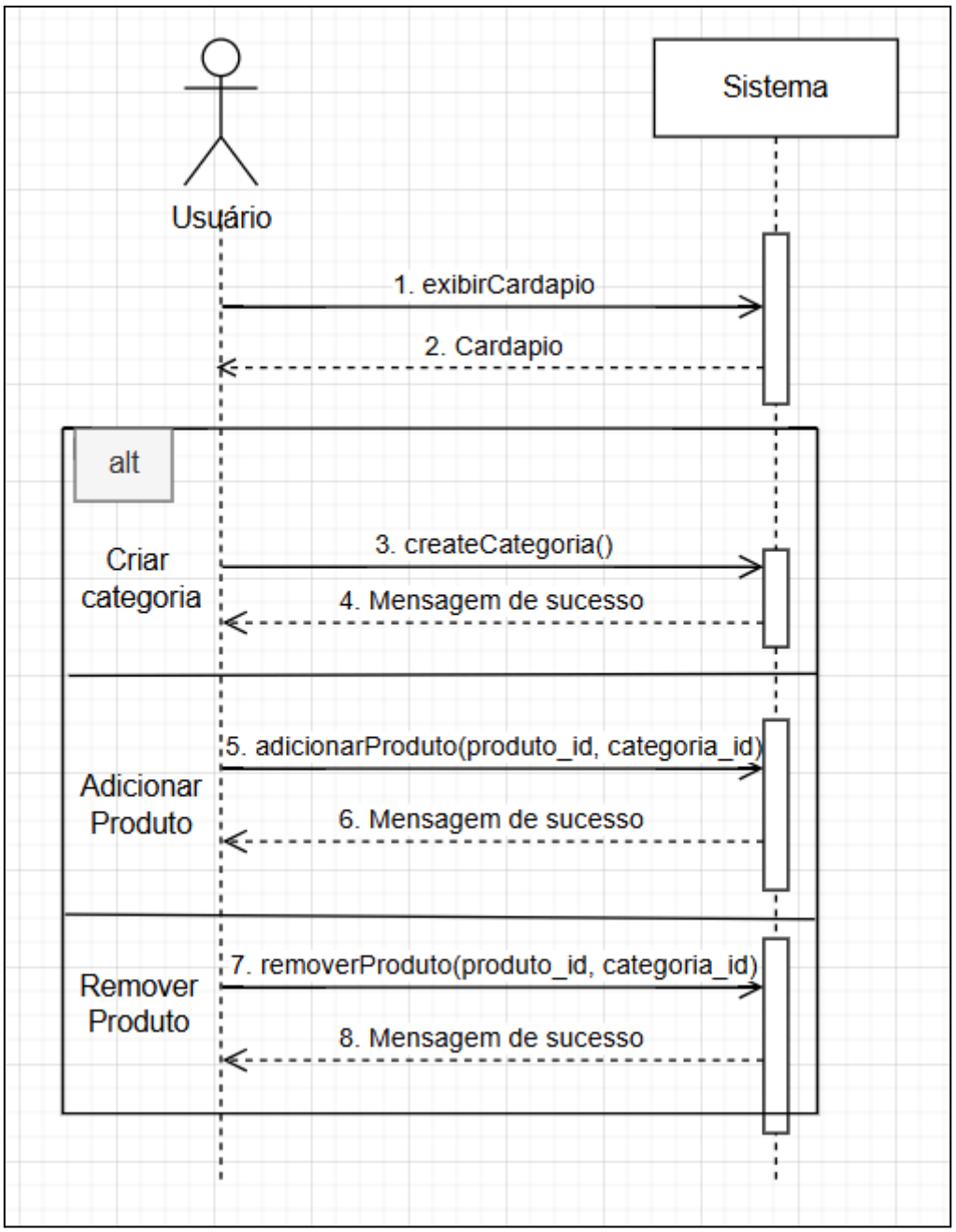

Figura 3. DSS - US 2 - Gerenciar cardápio

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. 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.

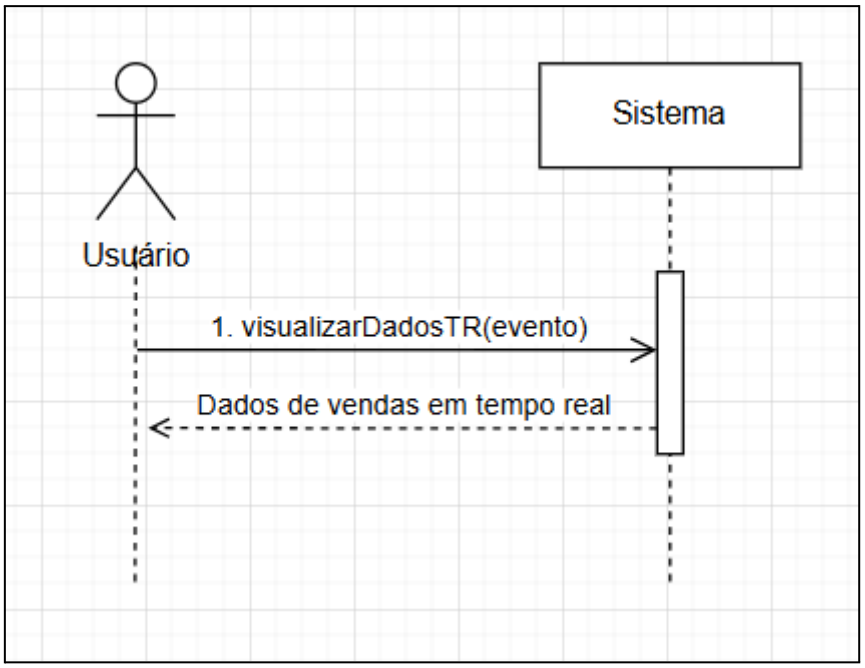

Figura 4. DSS - US 3 - Monitorar vendas

Contrato	Visualizar Vendas
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 são exibidos na interface do usuário

Tabela 7. Contrato Visualizar Vendas

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.

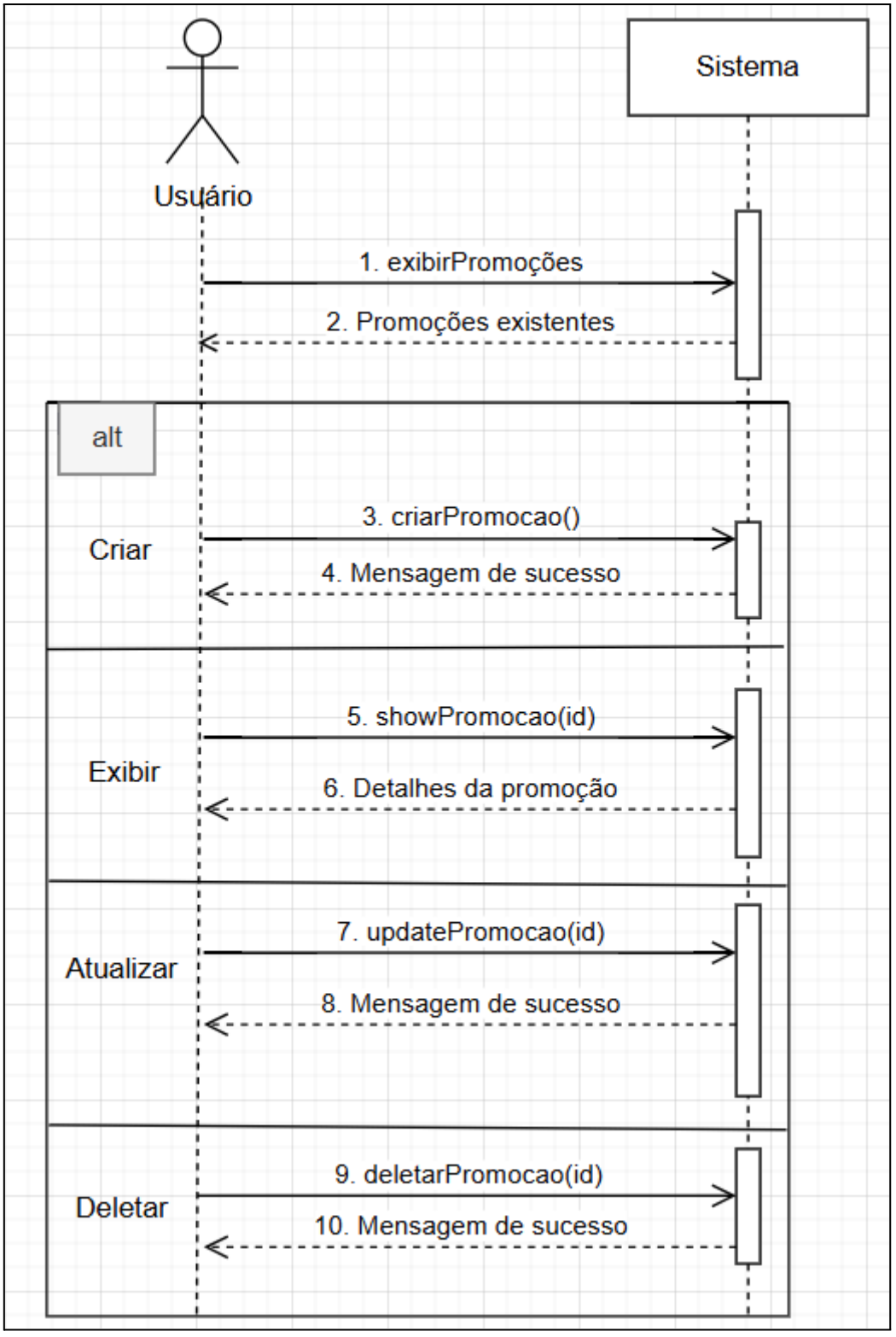

Figura 5. DSS - US 4 - Gerenciar promoções

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 consumidores, ajudando na análise e planejamento de eventos futuros. Este fluxo está associado à história de usuário US05.

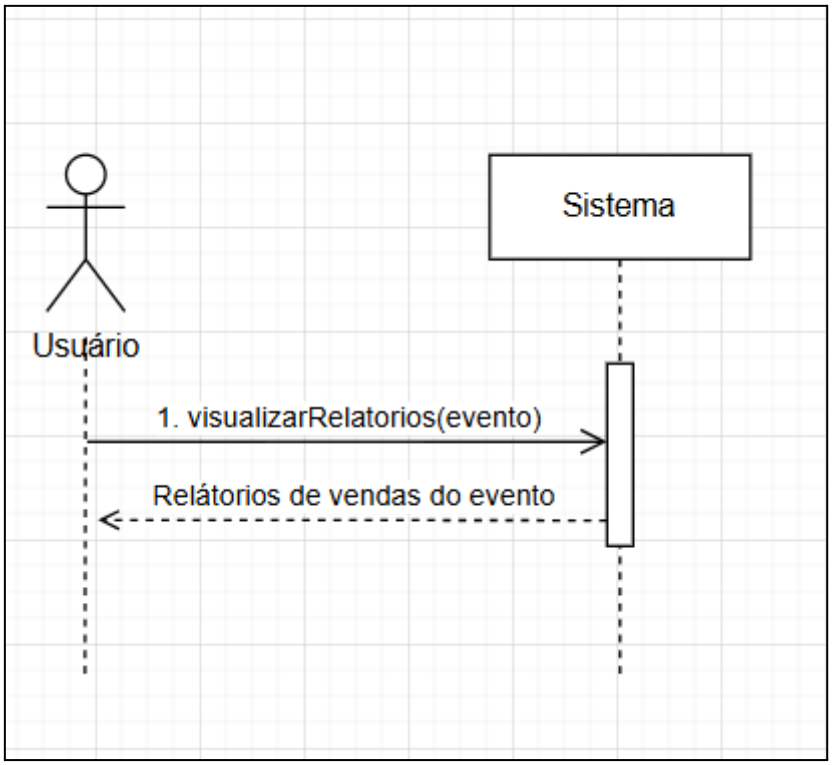

Figura 6. DSS - US 5 - Visualizar relatórios de vendas de eventos

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 consumidor acessa o cardápio digital do evento por meio de QR Codes e navega pelos produtos disponíveis. O consumidor pode visualizar itens com descrições e preços, além de verificar promoções aplicadas.

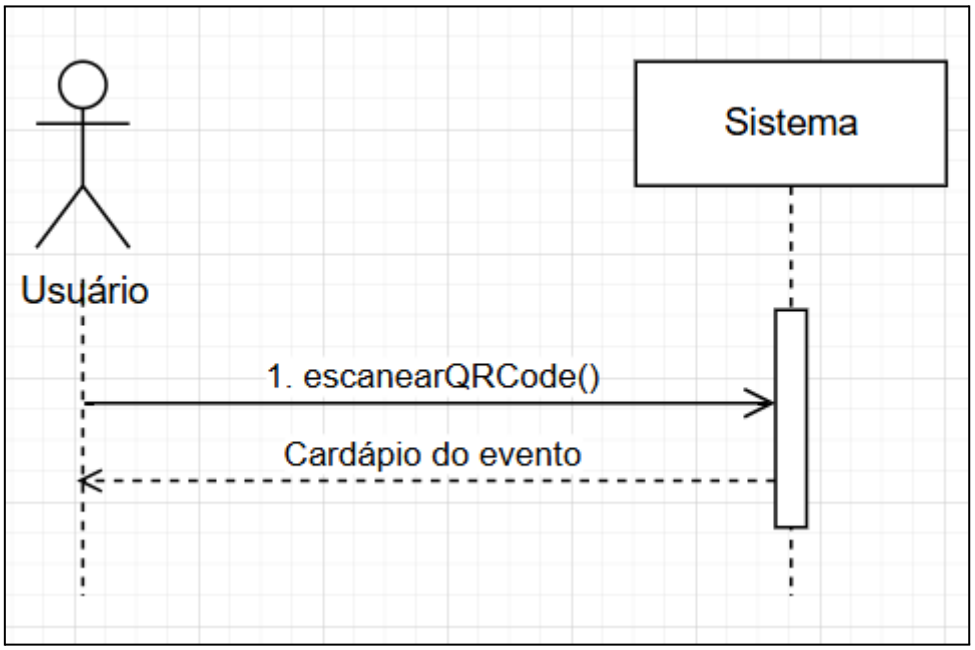

Figura 7. DSS - US 6 e 7 - Escanear código de resposta rápida e visualizar cardápio

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 consumidor na interface

Tabela 10. Contrato Acessar Cardápio Digital

A **Figura 8** descreve o fluxo para montagem do carrinho de compras pelo consumidor. O consumidor 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.

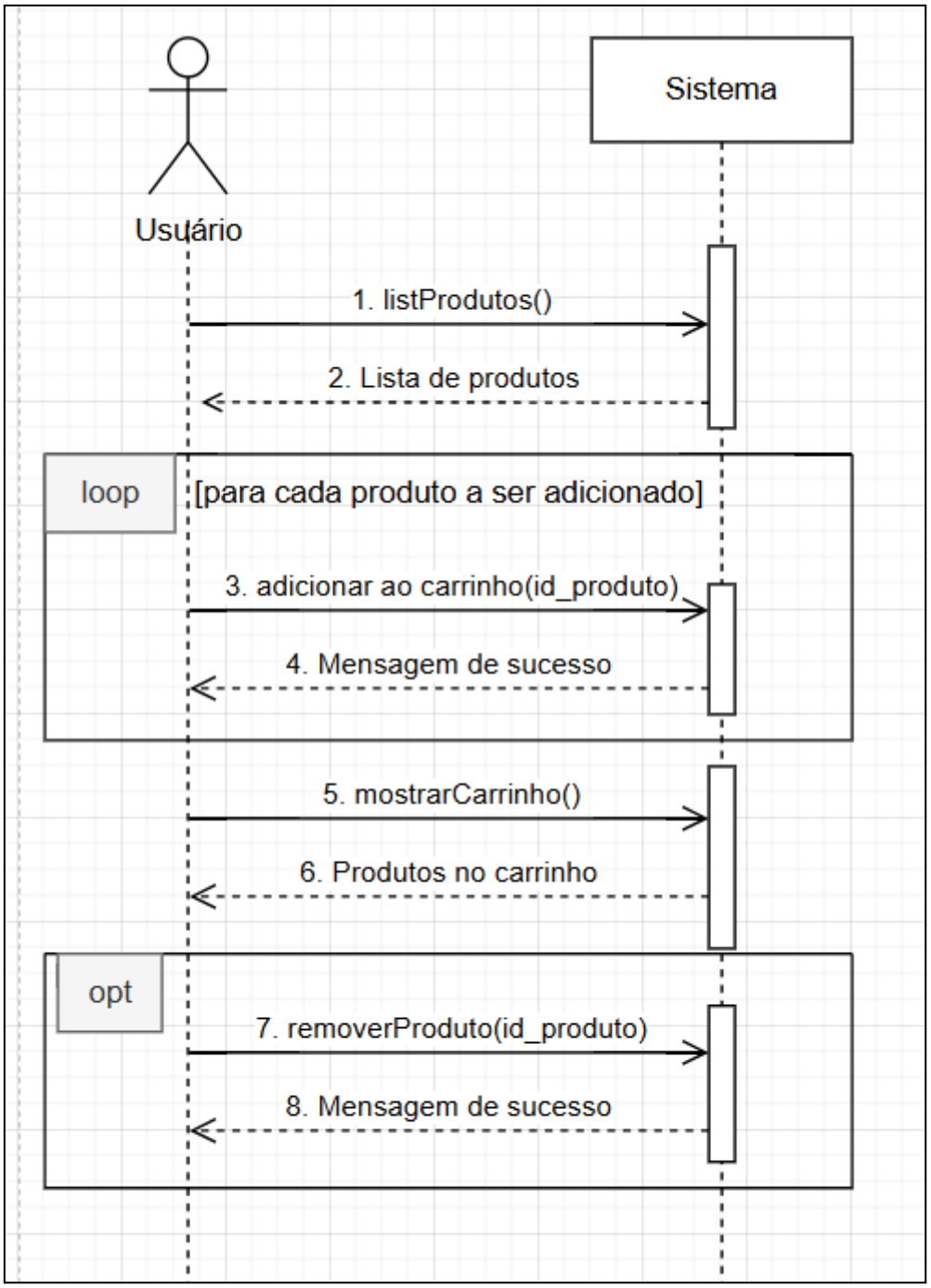

Figura 8. DSS - US 8 - Montar carrinho de compra

Contrato	Montar Carrinho
Operação	gerenciarCarrinho()
Referências cruzadas	Histórias de usuário: US08
Pré-condições	O consumidor 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 consumidor possa utilizá-los posteriormente.

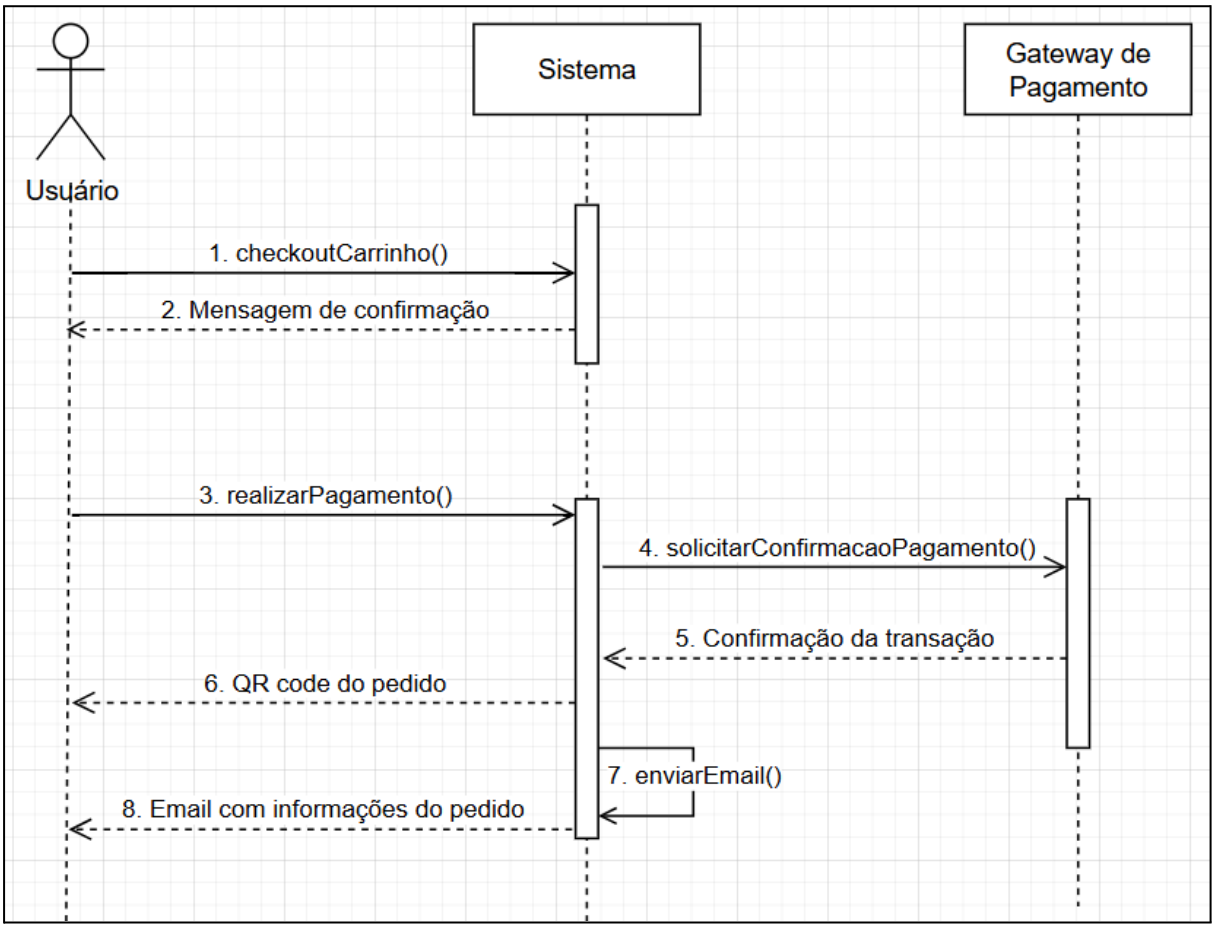

Figura 9. DSS - US 9 e 10 - Realizar pagamento e retirar pedido no bar

Contrato	Finalizar Compra e Gerar QR Codes
Operação	checkoutCarrinho()
Referências cruzadas	Histórias de usuário: US09, US10
Pré-condições	O consumidor 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 consumidores. 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**.

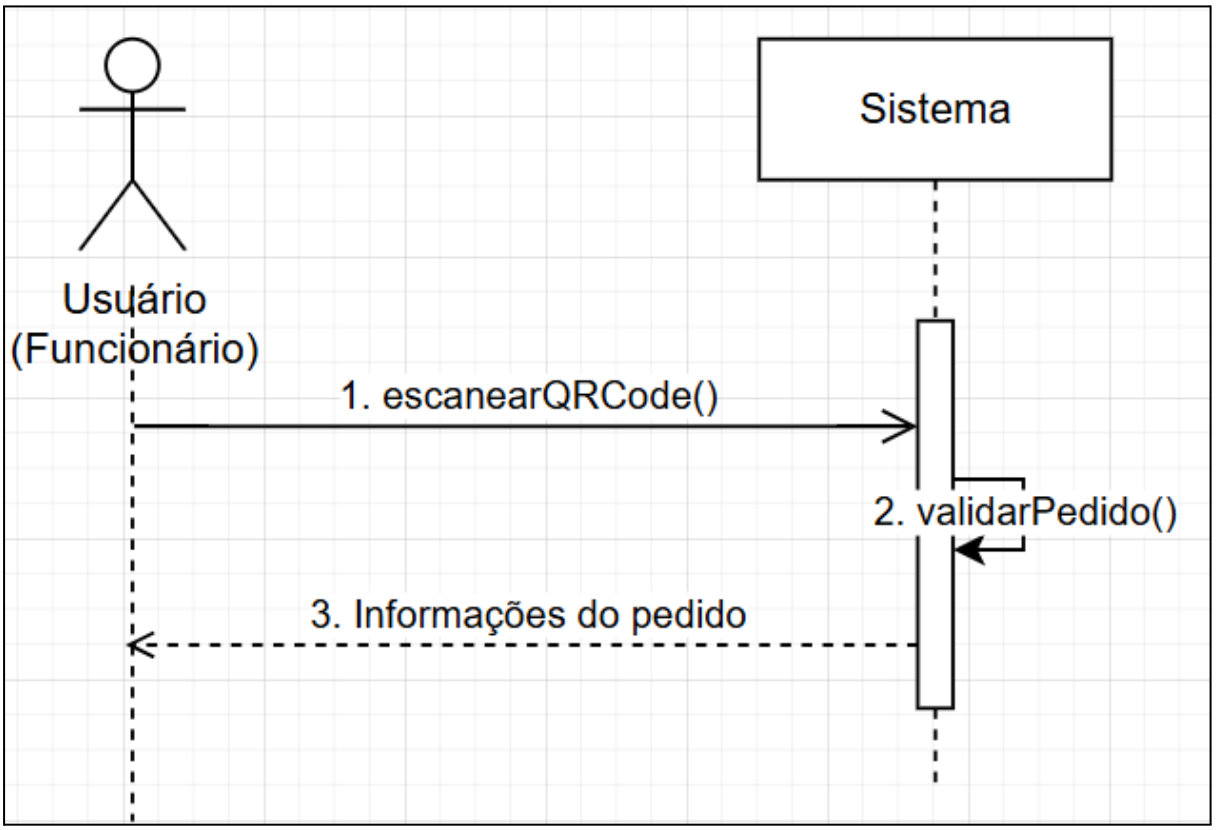

Figura 10. DSS - US 11 Escanear código de resposta rápida do consumidor para validar venda

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

Esta seção apresenta os modelos de projeto elaborados para o sistema Tuscan, com o objetivo de detalhar tecnicamente como os requisitos levantados foram transformados em componentes de software coesos e funcionais. Os diagramas aqui descritos — de classes, sequência, comunicação, estado, componentes e implantação — fornecem uma visão clara da estrutura interna da aplicação, dos fluxos de interação entre usuários e sistema, e da arquitetura adotada.

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 funcionários. 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.

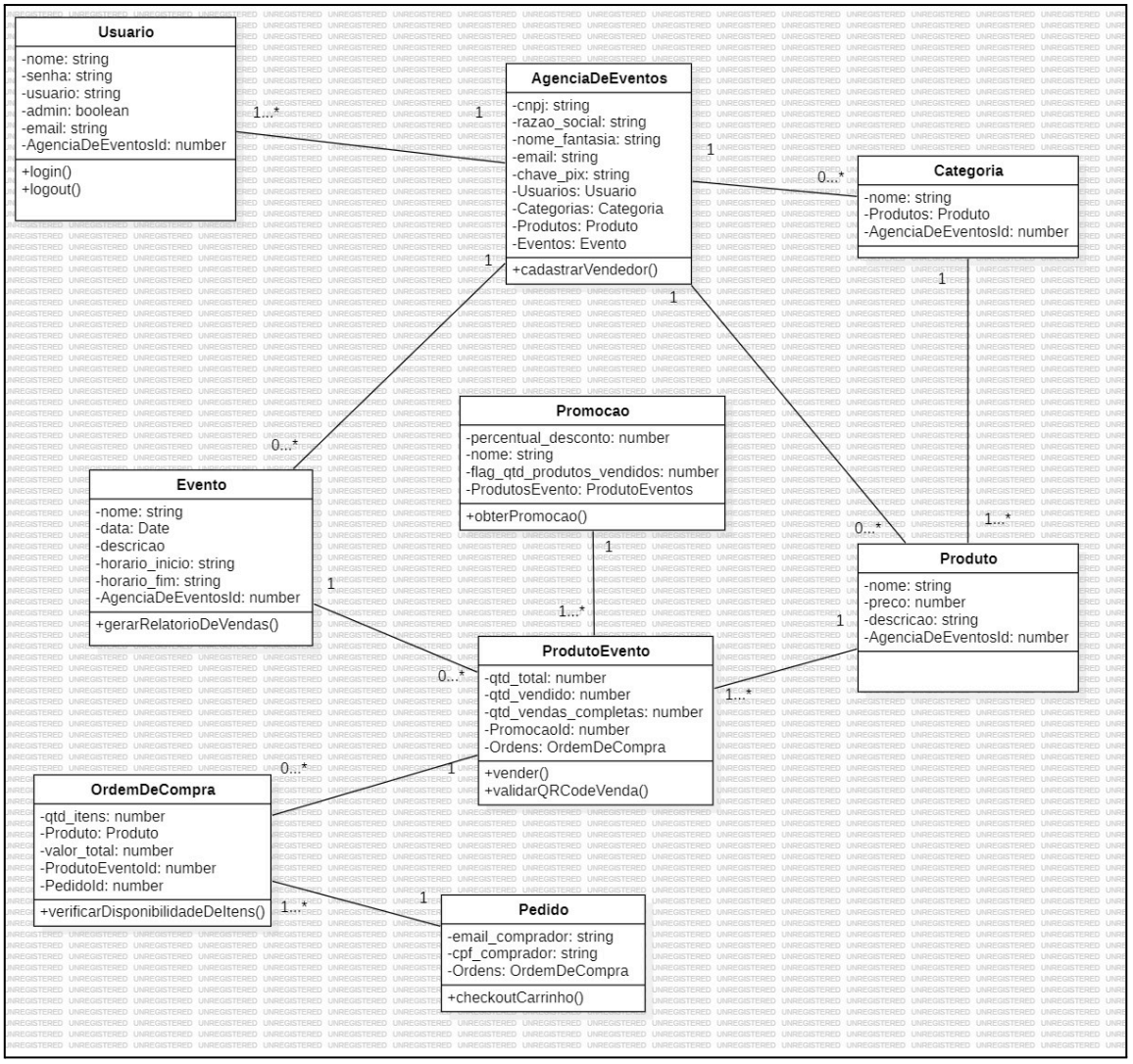

Figura 11. Diagrama de classes

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 componentes são representadas, para que a aplicação funcione da maneira correta.

A **Figura 12** ilustra o fluxo de interação entre o consumidor, o cardápio digital, o Stripe e o Supabase (Baas - Model e Banco de Dados) para o processo de realizar uma compra no Tuscan. Esse fluxo começa com o consumidor 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 consumidor 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.

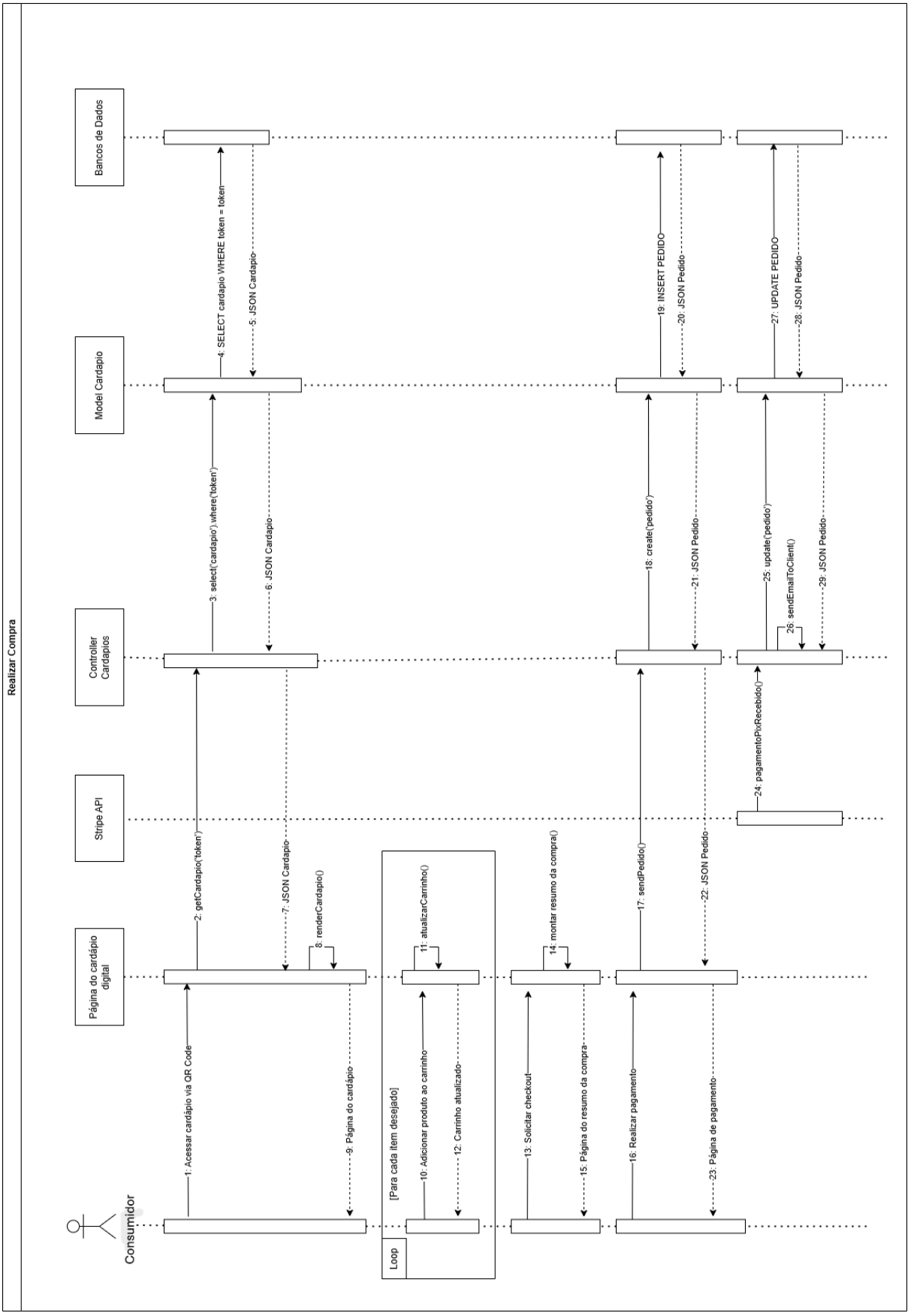

Figura 12. DS - Realizar Compra

A **Figura 13** apresenta o fluxo de interação para a entrega de um pedido no **Tuscan**, envolvendo o consumidor, o funcionário, o aplicativo (tela de scanner) e o Supabase (Baas - Model e Banco de Dados). O processo inicia com o consumidor 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 consumidor. 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.

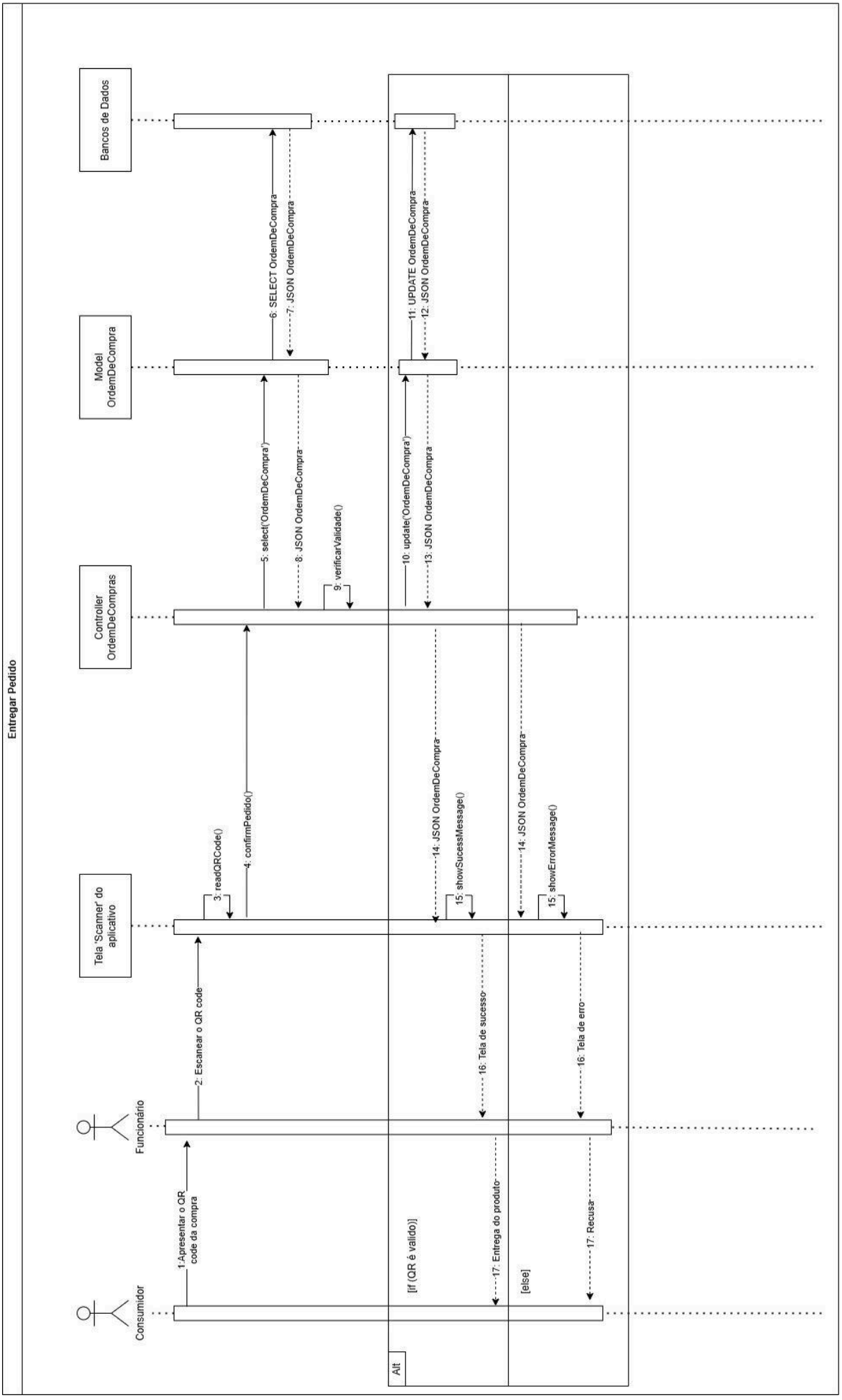
Entregar Pedido
Consumidor
Funcionário
Tela 'Scanner' do aplicativo
Controller OrdemDeCompras
Model OrdemDeCompra
Bancos de Dados
1: Apresentar o QR code da compra
2: Escanear o QR code
3: readQRCode()
4: confirmPedido()
5: select('OrdemDeCompra')
6: SELECT OrdemDeCompra
7: JSON OrdemDeCompra
8: JSON 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

A **Figura 14** é um diagrama que representa o fluxo de interação no processo de cadastro, edição e consulta de produtos realizados pelo organizador do evento, conforme descrito na história de usuário **US01**. O fluxo se inicia com o organizador acessando a tela de listagem de produtos cadastrados. A partir dessa interface, é possível abrir o formulário de cadastro, preencher os dados do novo produto e submetê-los ao sistema. Após a criação, uma mensagem de sucesso é exibida. O organizador também pode, posteriormente, editar as informações de um produto já existente, atualizando seus dados conforme necessário.

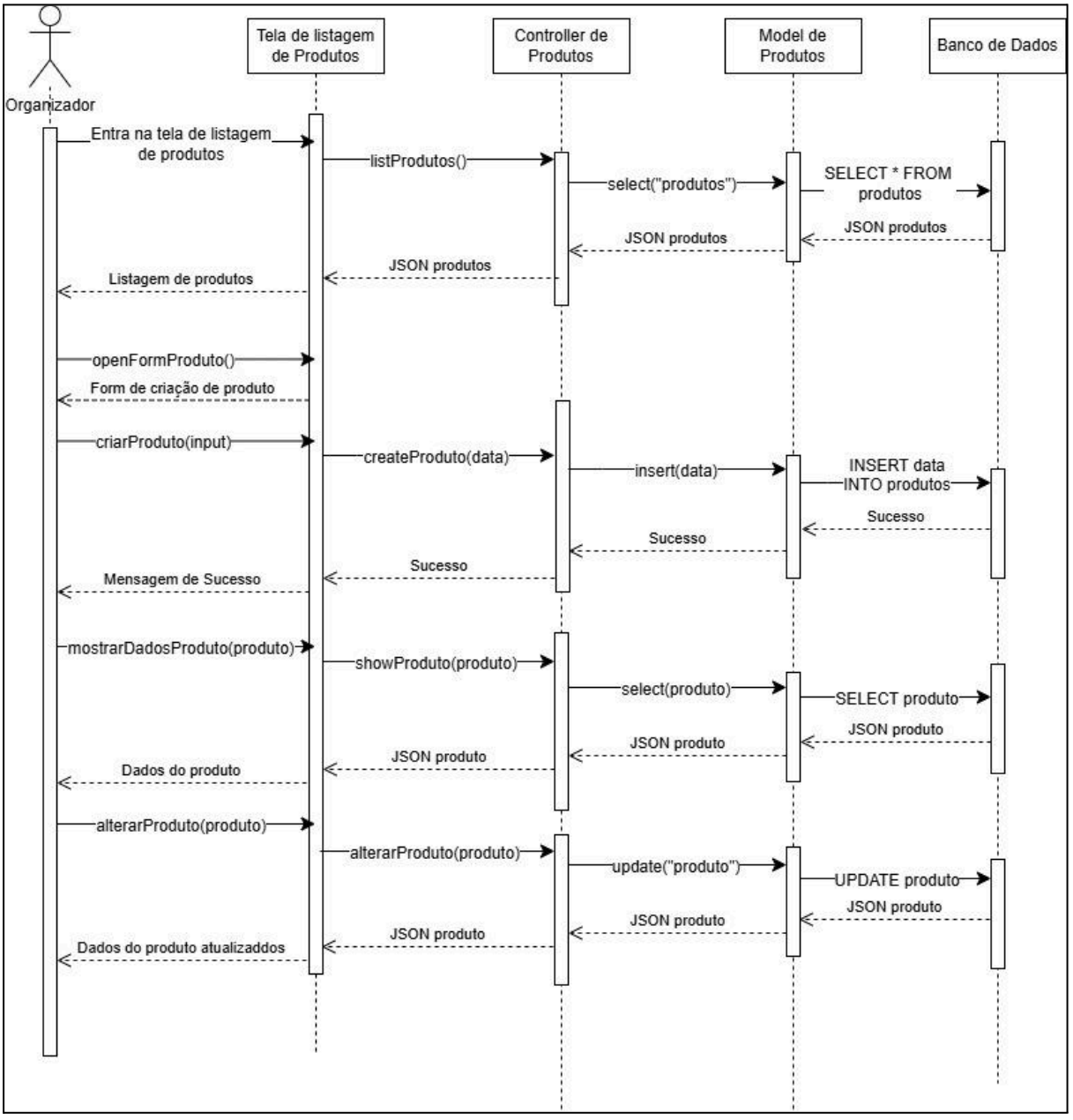

Figura 14. DS - CRUD de produtos*

A **Figura 15** é um diagrama de sequência que modela o processo de criação e edição de eventos, vinculado à história de usuário **US02**. O organizador acessa inicialmente a tela de

listagem de eventos disponíveis, visualizando os dados já cadastrados. A partir dessa tela, ele pode preencher o formulário de criação de um novo evento, enviando as informações para persistência no sistema. Também é possível retornar a essa interface posteriormente para editar os dados de um evento existente, garantindo flexibilidade e manutenção contínua da estrutura do sistema.

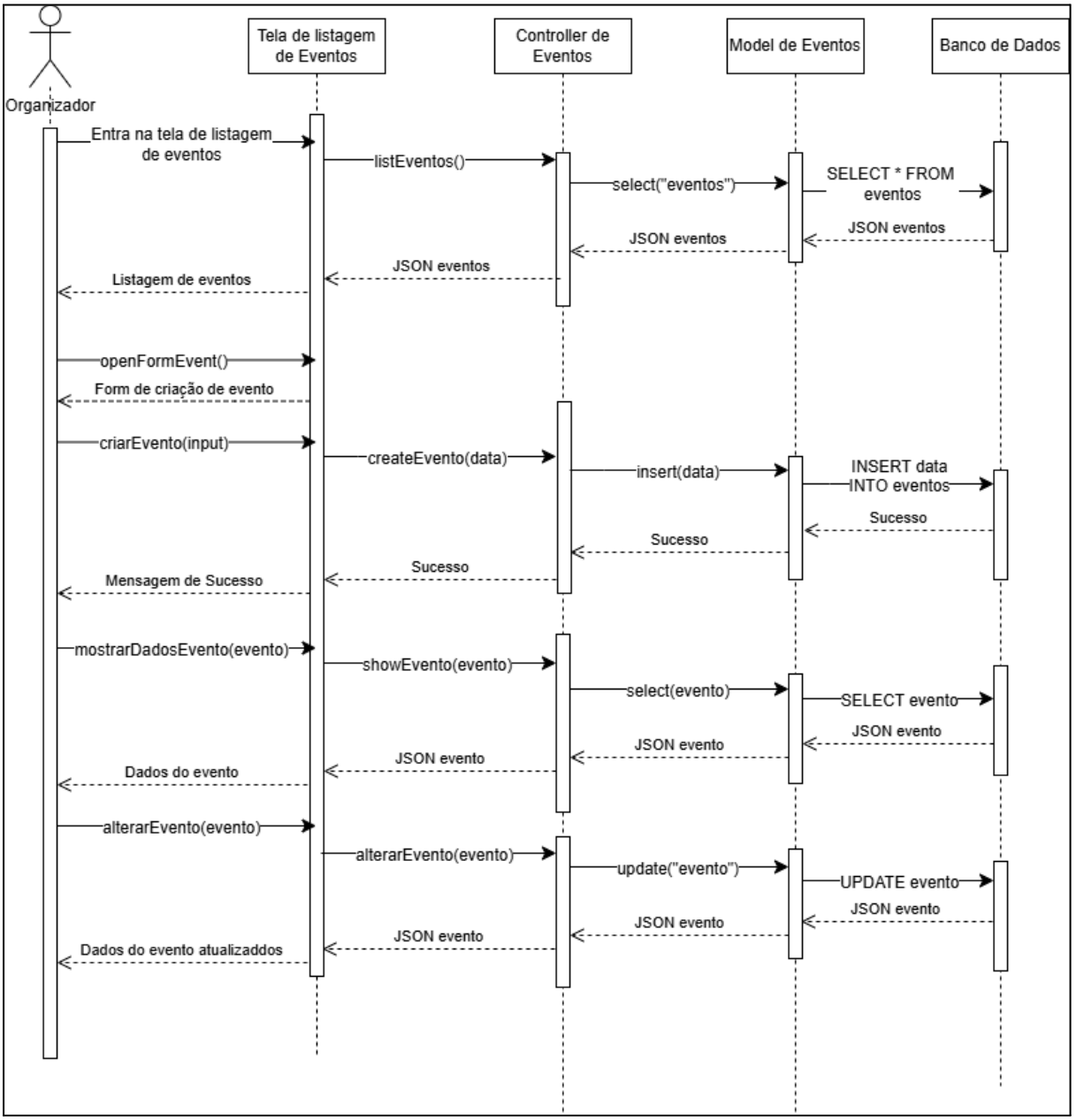

Figura 15. DS - CRUD de eventos*

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 16** 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 consumidor 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 Controller de Cardápios, a API Stripe para pagamento via Pix, e o Controller 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.

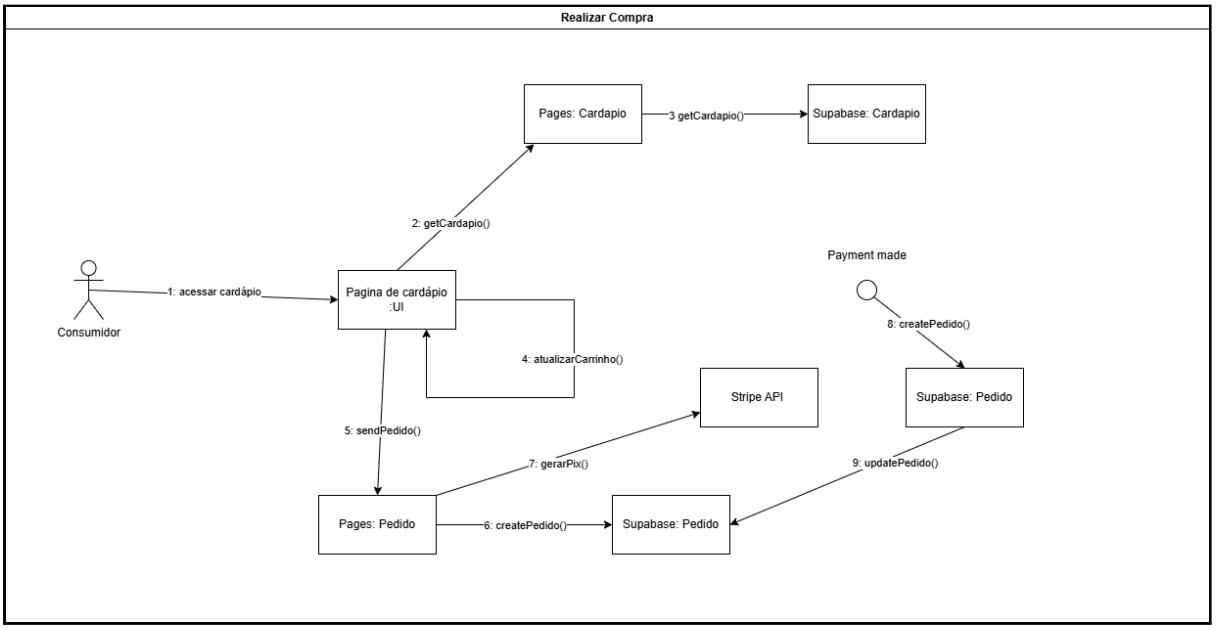

Figura 16. DC - Realizar Compra

A **Figura 17** 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 consumidor. A validação envolve o Controller de Ordens de Compra, que consulta e atualiza as informações no banco de dados. O diagrama enfatiza a troca de mensagens para garantir a entrega correta e eficiente do produto ao consumidor.

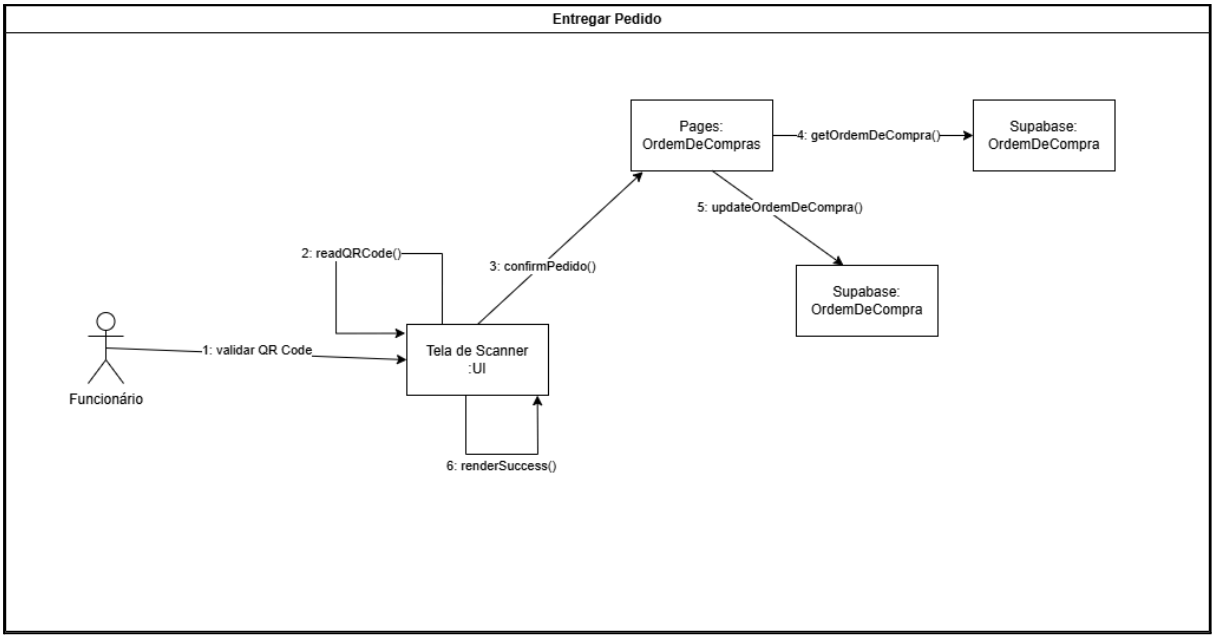

Figura 17. DC - Entregar Pedido

3.4. Arquitetura

A **Figura 18** apresenta o diagrama de pacotes do sistema Tuscan, evidenciando sua estrutura modular composta por três áreas principais:

- Web App: Utilizado por organizadores de eventos e consumidores finais.
- Mobile App para Funcionários: Dedicado exclusivamente à validação de QR Codes de pedidos.
- Infraestrutura Backend (Supabase): Responsável por persistência de dados, autenticação e integração com serviços externos.

Organização de Arquivos e Roteamento

[Web App (Next.js)]

O Web App foi desenvolvido utilizando o framework Next.js, que oferece uma estrutura baseada em roteamento por arquivos dentro da pasta app. Para definir uma rota, basta adicionar um arquivo page.tsx na respectiva pasta. Por exemplo, o arquivo ‘app/users/page.tsx’ corresponde à rota ‘/users’. Além disso, é possível definir layouts específicos para grupos de rotas utilizando o arquivo layout.tsx dentro das subpastas.

O Next.js permite especificar quais componentes são renderizados no servidor (Server Components) e quais são renderizados no cliente (Client Components). No contexto do Tuscan, todas as comunicações com o Supabase (banco de dados e storage) são realizadas através de Server Actions, organizadas na pasta actions. Essas funções são importadas diretamente nos componentes, funcionando como rotas de API, mas com integração facilitada, pois são tratadas como funções comuns em JavaScript.

[Mobile App para Funcionários (React Native com Expo)]

O aplicativo móvel foi desenvolvido com React Native utilizando o framework Expo. Assim como no Next.js, o Expo utiliza a pasta app para gerenciamento de rotas. No entanto, ao invés de ‘page.tsx’, utiliza-se ‘index.tsx’ para definir os componentes das rotas. Não é necessário criar pastas adicionais para novas rotas além da raiz /; basta nomear o arquivo com o nome da nova rota. Caso seja necessário criar um novo layout para um grupo de rotas, cria-se uma nova pasta.

Modelo de Arquitetura Adotado

[Web App: Modelo Híbrido / MVVM Adaptado]

Dentro do escopo da aplicação Web desenvolvida com Next.js, adotamos um modelo arquitetural híbrido, mais alinhado ao padrão MVVM (Model–View–ViewModel) adaptado, amplamente adotado em aplicações React modernas. Nesse modelo, os componentes e páginas React representam a View, sendo responsáveis pela renderização da interface e recepção de interações do usuário. A lógica de aplicação — como manipulação de estado, transformação de dados e controle do fluxo de navegação — é isolada em hooks personalizados, contexts, Server Actions e Server Components, que atuam como uma espécie de ViewModel: eles intermediam o acesso ao Model (dados) e fornecem uma camada de abstração entre os dados brutos e a interface.

O Model, por sua vez, está representado na forma dos dados persistidos no Supabase (PostgreSQL), acessados por meio do SDK do Supabase — que funciona como um super ORM — permitindo realizar queries e interações com a base de dados de maneira segura e reativa.

Diferente do padrão MVC clássico, em que as camadas de Controller e View são rigidamente separadas, no Next.js essas responsabilidades são divididas entre funções assíncronas (como Server Actions) e os próprios componentes que consomem e exibem os dados. Essa abordagem componente-centrada, combinada com a separação entre apresentação e lógica de estado, aproxima a arquitetura do padrão MVVM, promovendo modularidade, facilidade de testes e flexibilidade de desenvolvimento.

[Mobile App: Arquitetura Serverless com Backend-as-a-Service (BaaS)]

O aplicativo móvel segue uma arquitetura Serverless com Backend-as-a-Service (BaaS), onde o Supabase é responsável pela persistência de dados, autenticação e regras de acesso. O aplicativo em React Native (cliente) atua como camada de apresentação e lógica de controle leve, utilizando diretamente o Client do Supabase (SDK) para interações com o banco de dados PostgreSQL. A lógica de negócio é implementada diretamente no aplicativo ou nas Edge Functions do Supabase, conforme necessário.

Vantagens das Tecnologias Adotadas

[Next.js]

- Popularidade e Comunidade: O Next.js é atualmente um dos frameworks mais populares para aplicações React, sendo amplamente adotado por empresas como Spotify e Nike.
- Renderização Híbrida: Suporte a Server-Side Rendering (SSR), Static Site Generation (SSG) e Client-Side Rendering (CSR), proporcionando flexibilidade e otimização de desempenho.
- SEO e Performance: A renderização no servidor melhora significativamente o SEO e reduz o tempo de carregamento das páginas.
- Integração com Vercel: A Vercel, criadora do Next.js, oferece hospedagem gratuita com integração direta ao GitHub e fluxos de CI/CD automatizados.

[Supabase]

- Plataforma Completa: O Supabase fornece uma solução completa com banco de dados PostgreSQL, autenticação, storage, funções edge e funcionalidades em tempo real.
- Hospedagem Gratuita: Oferece um plano gratuito com 500MB de armazenamento de banco de dados e 1GB de armazenamento de arquivos, ideal para projetos acadêmicos e MVPs.
- Facilidade de Integração: Disponibiliza SDKs especializados para integração tanto com Next.js quanto com Expo, simplificando o desenvolvimento.

[React Native com Expo]

- Curva de Aprendizado Reduzida: A sintaxe do React Native é semelhante à do React para web, facilitando a transição para desenvolvedores familiarizados com React.
- Desenvolvimento Multiplataforma: Permite o desenvolvimento de aplicativos para iOS e Android a partir de uma única base de código.
- Ferramentas de Desenvolvimento: O Expo oferece ferramentas como o Expo Go para testes em dispositivos reais e o EAS (Expo Application Services) para criação de builds em nuvem, facilitando o processo de deployment.

Considerações Finais

A escolha das tecnologias para o sistema Tuscan foi guiada por critérios de escalabilidade, facilidade de desenvolvimento e manutenção, além de alinhamento com as práticas modernas de desenvolvimento de software. A combinação de Next.js, Supabase e React Native com Expo proporciona uma arquitetura robusta, eficiente e alinhada com as tendências atuais do mercado.

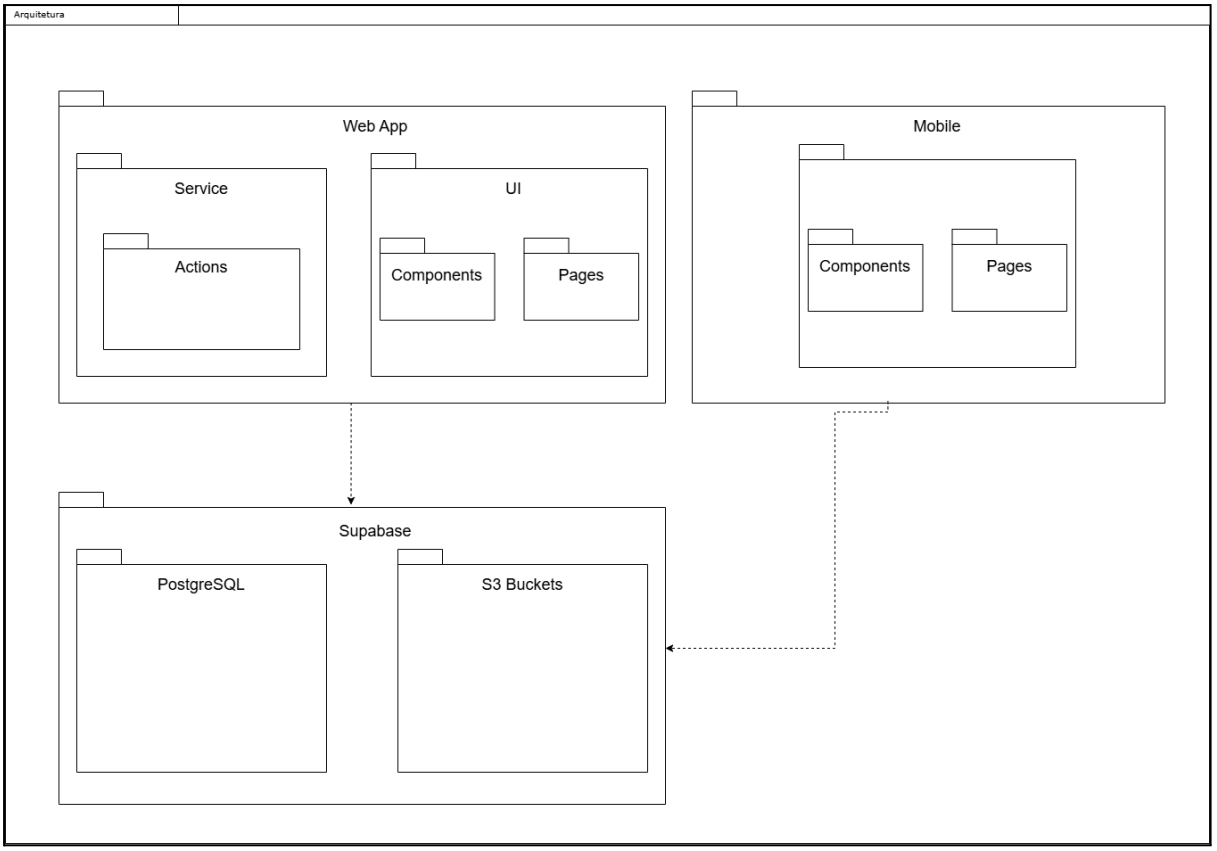

Figura 18. Diagrama de Pacotes

3.5. Diagramas de Estado

A **Figura 19** representa o diagrama de estados que descreve o ciclo de vida de um pedido realizado por um consumidor no sistema Tuscan. O fluxo inicia com a ação acessarCardapio(), que leva ao estado “Pedido criado”, no qual a interface exibe a tela de cardápio. Neste estado, o consumidor pode montar seu pedido e, ao solicitar a finalização, o sistema transita para o estado “Checkout pendente”, exibindo a tela de checkout.

A partir desse ponto, o consumidor escolhe uma forma de pagamento, o que desencadeia a transição para o estado “Pagamento solicitado”, onde é exibida a tela de pagamento. Após o recebimento da confirmação de pagamento, o estado muda para “Pagamento confirmado”, com a exibição da tela contendo os QR Codes das fichas adquiridas, que representam os produtos comprados.

Além disso, após a confirmação de pagamento, os QR Codes podem ser validados no ponto de retirada por um funcionário do evento. Essa ação, não altera o estado do pedido, mas garante a rastreabilidade das entregas de cada item.

Esse diagrama evidencia os principais estados e transições envolvidas na jornada de compra do consumidor dentro do sistema, cobrindo desde a navegação inicial pelo cardápio até o uso das fichas geradas após o pagamento.

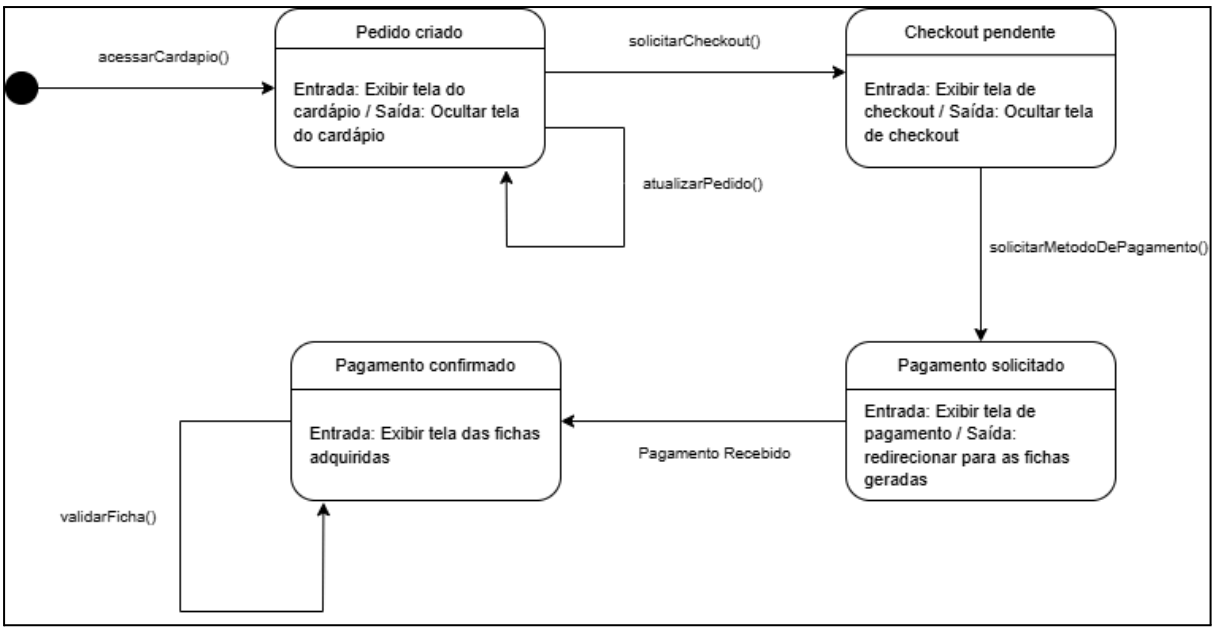

Figura 19. Diagrama de Estado

3.6. Diagramas de Componentes e Implantação

A **Figura 20** representa o diagrama de componentes do sistema, que define os principais módulos das aplicações Web, Mobile e Supabase, ilustrando suas interações.

O Web App, desenvolvido em Next.js, inclui tanto interfaces de usuário (ui/frontend - browser) quanto serviços (controllers/backend - server) que interagem com o Supabase para realizar operações como cadastro de eventos, gerenciamento de cardápios e processamento de pedidos. O Mobile App para Funcionários, desenvolvido com React Native (Expo), é uma aplicação frontend independente, que também se conecta diretamente ao Supabase via *Supabase Client* para validar os QR Codes apresentados pelos consumidores no momento da retirada dos produtos.

O Supabase atua como Backend-as-a-Service (BaaS), oferecendo um banco de dados relacional PostgreSQL, armazenamento de arquivos por meio de buckets AWS S3, autenticação integrada e extensões importantes, como a API do Stripe, usada para processar pagamentos. Dessa forma, tanto o Web App quanto o Mobile App consomem diretamente os recursos do Supabase, garantindo escalabilidade, segurança e uma arquitetura moderna baseada em serviços gerenciados.

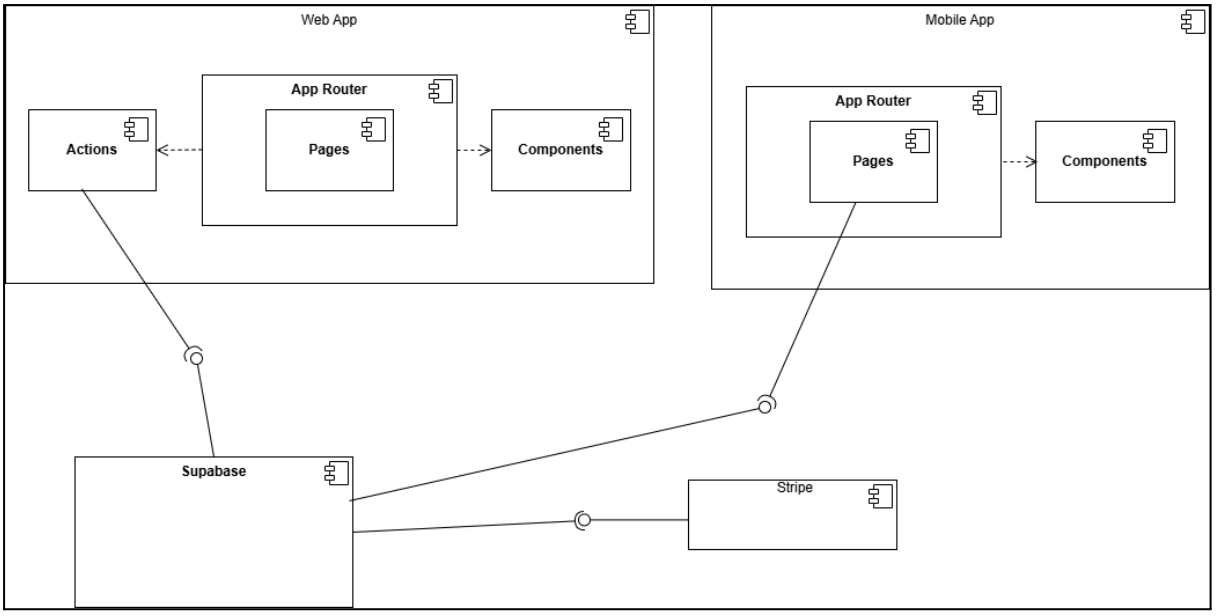

Figura 20. Diagrama de Componentes

A **Figura 21** representa o diagrama de implantação do sistema Tuscan, descrevendo sua arquitetura física baseada em uma estrutura distribuída. O diagrama ilustra como os diferentes dispositivos dos usuários se comunicam com os serviços hospedados na Vercel (WebApp) e na infraestrutura do Supabase.

Os consumidores e organizadores acessam o sistema por meio de navegadores em dispositivos desktop, enquanto os funcionários do bar utilizam um aplicativo mobile dedicado, instalado em dispositivos Android ou iOS. O WebApp, desenvolvido em Next.js, é hospedado na Vercel — uma plataforma de deploy focada em aplicações com suporte nativo ao Next.js, que oferece entrega otimizada de conteúdo com CDN global.

A arquitetura do Tuscan garante alta escalabilidade e menor complexidade operacional, favorecendo tanto a manutenção quanto a evolução da aplicação.

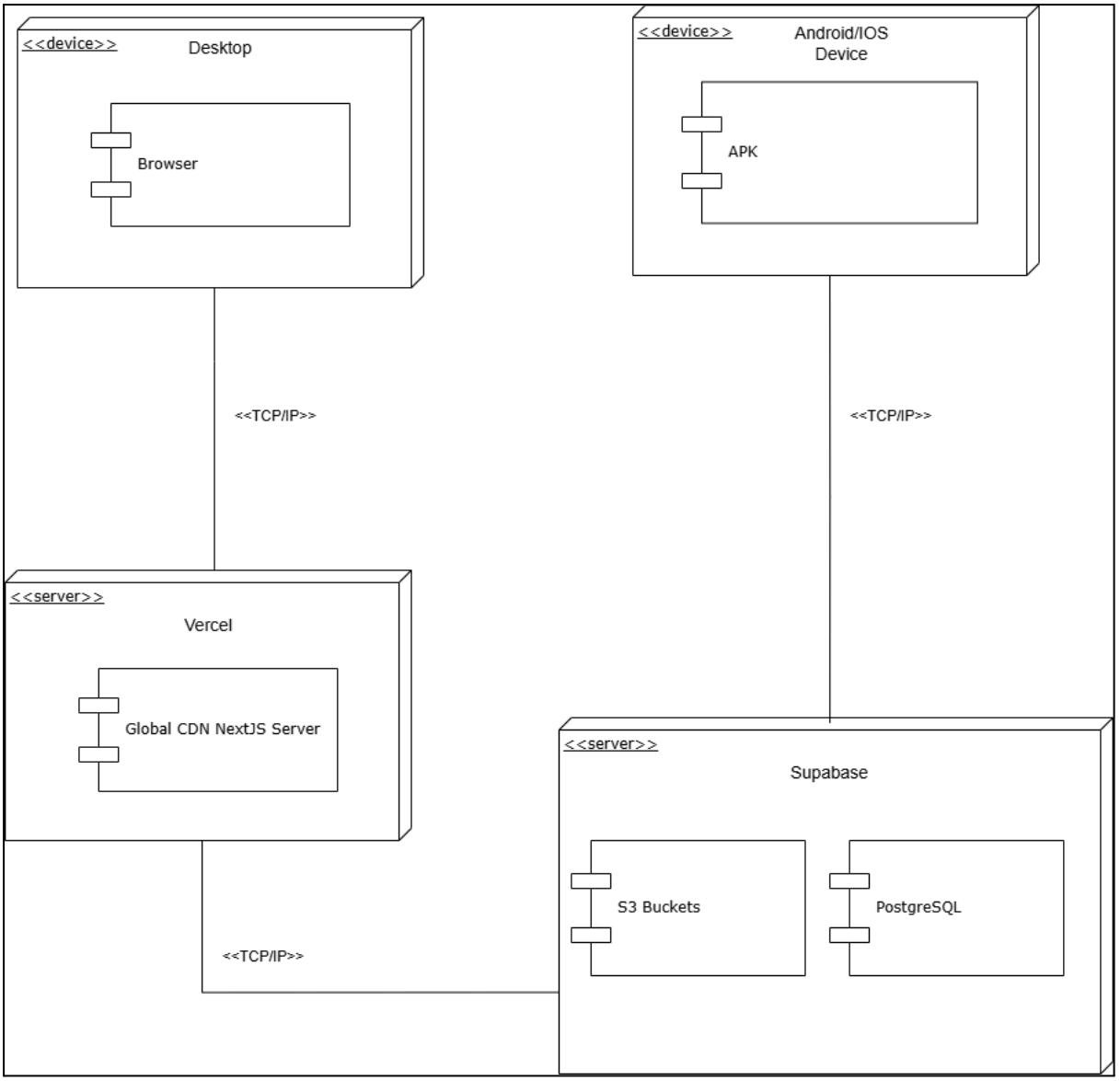

Figura 21. Diagrama de Implantação

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 22** lista todos os eventos criados pelo organizador, com opções para filtrar pelo nome do evento. É o ponto de entrada para acessar e gerenciar os detalhes de cada evento. Além disso, a tela permite a inclusão de novos eventos.

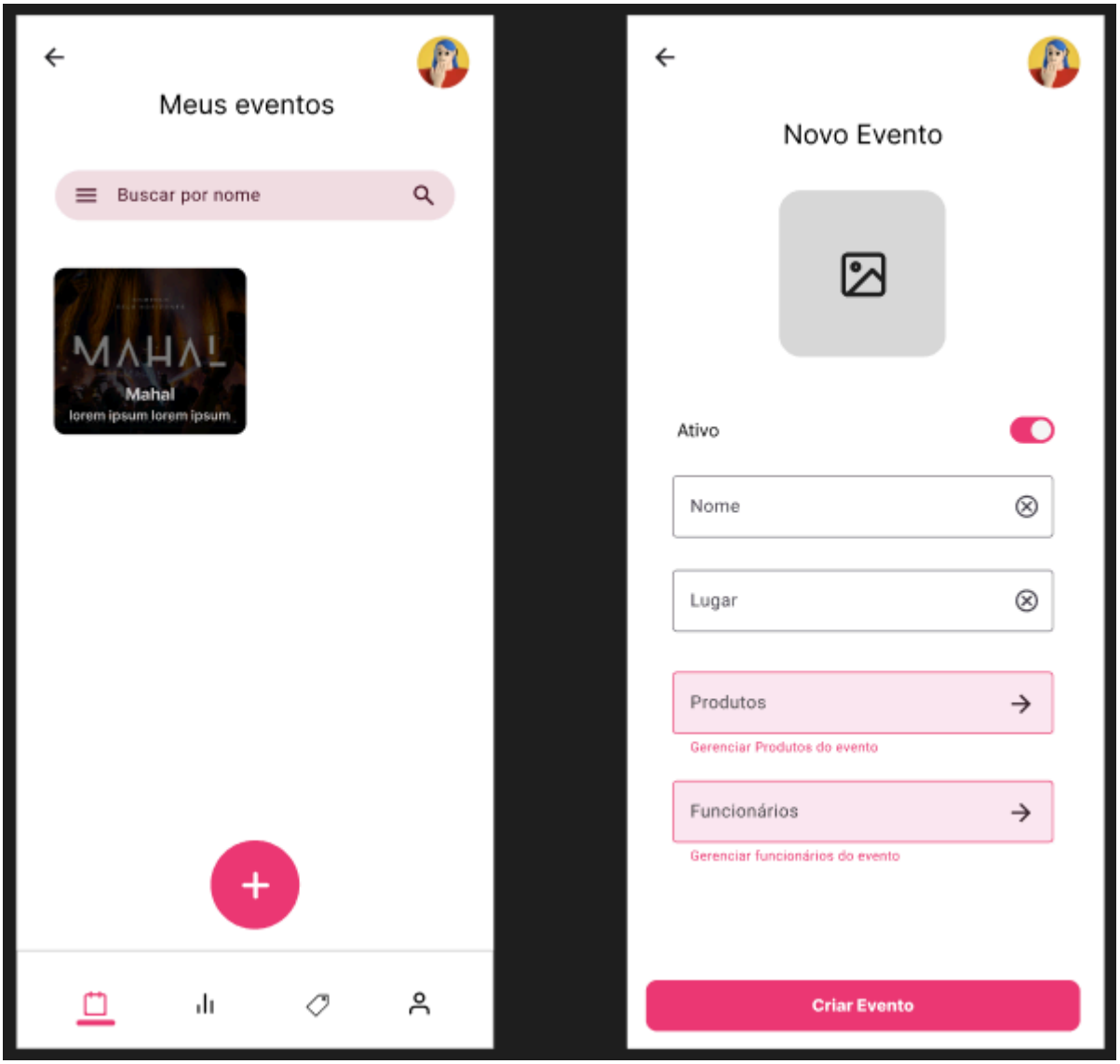

Figura 22. Tela “Meus Eventos” do organizador

A tela representada na **Figura 23** permite que organizadores gerenciem os itens do evento: adicionar novos produtos ou produtos já existentes, assim como vincular os usuários para acesso dos funcionários do evento.

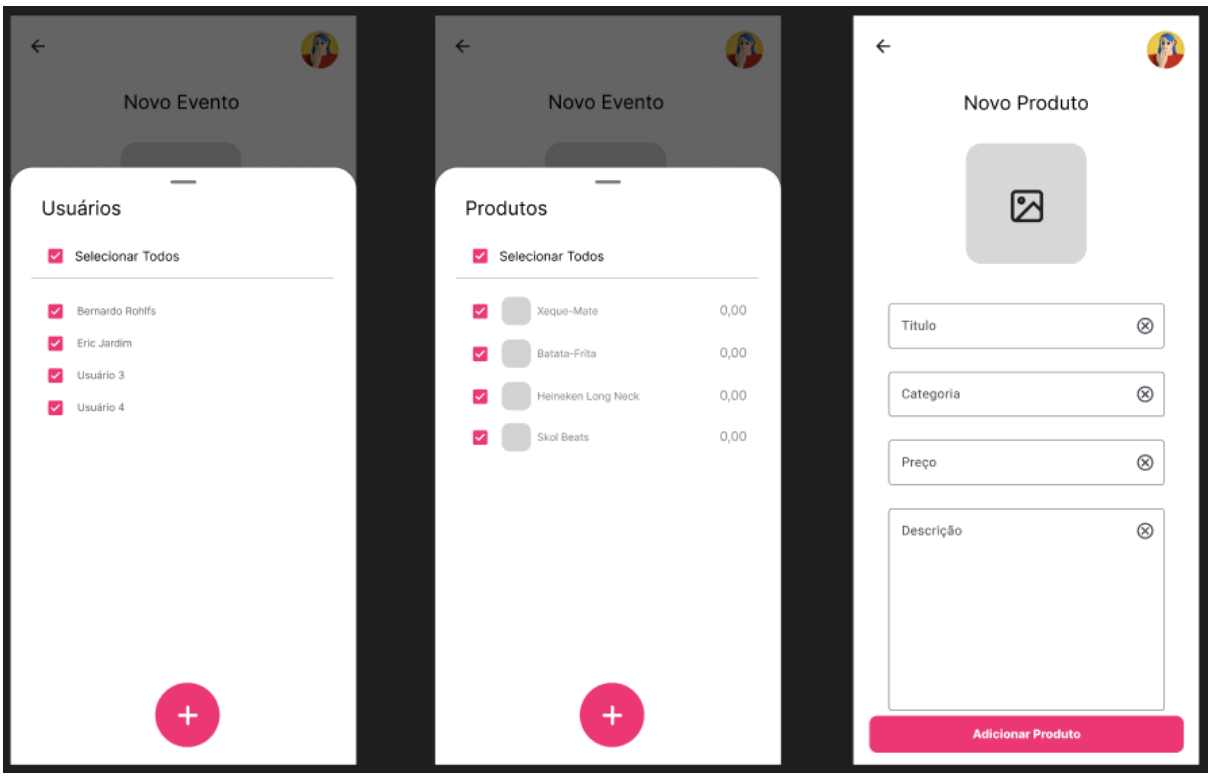

Figura 23. Tela de evento do organizador

A tela representada na **Figura 24** exibe e permite a edição de informações do organizador, como dados bancários e permissões de acessos para funcionários. Também serve como ponto de gerenciamento de configurações pessoais.

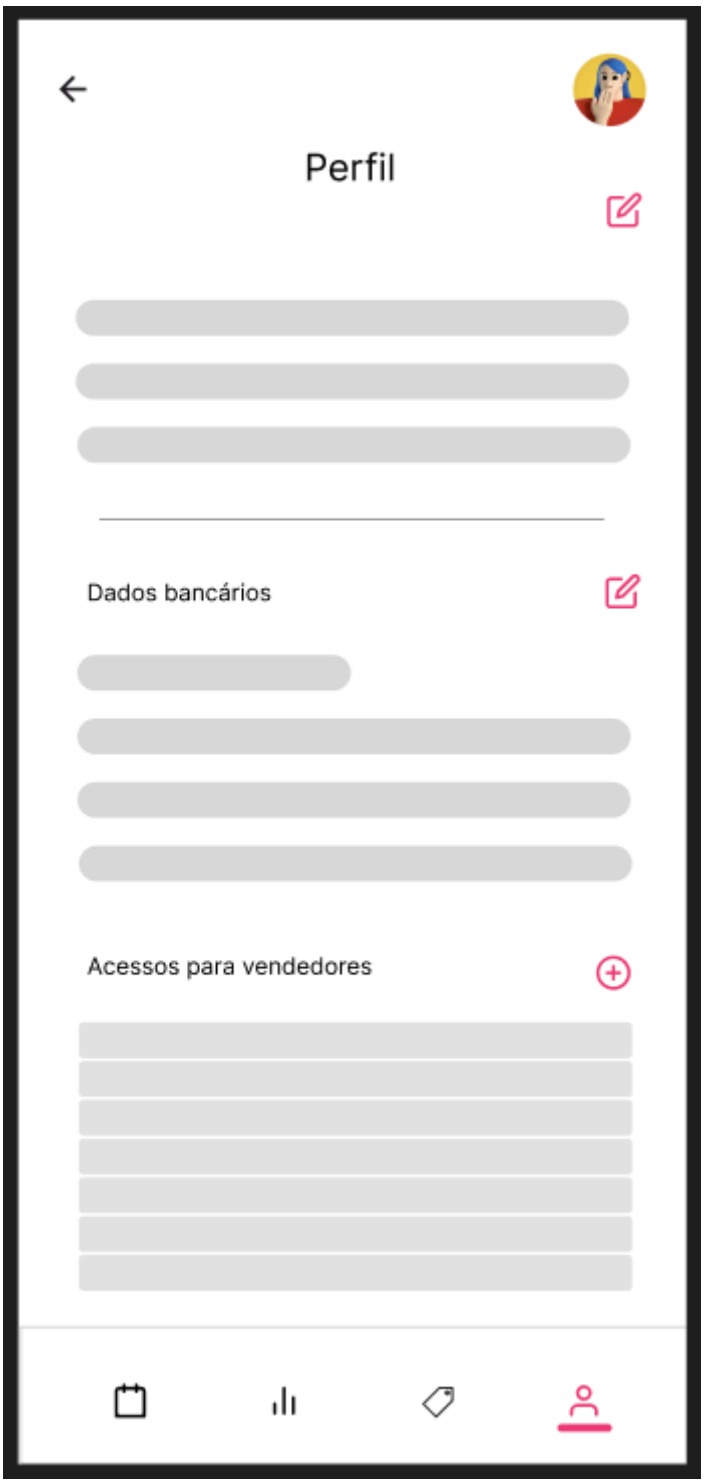

Figura 24. Tela de perfil do organizador

A tela representada na **Figura 25** oferece ao organizador uma visão geral das métricas dos eventos, como vendas e desempenho de produtos, “filtráveis” por data. É um painel estratégico para monitorar os resultados dos eventos.

Figura 25. Tela de dashboard do organizador

4.2. Esboço das Interfaces Usadas Apenas pelo Consumidor do Evento

A tela representada na **Figura 26** exibe o cardápio do evento acessado pelo consumidor através do QR Code. Nela, é possível visualizar os itens disponíveis, seus preços e descrições, além de selecionar as quantidades desejadas. O consumidor utiliza essa interface para montar seu pedido, que será finalizado na tela de checkout.

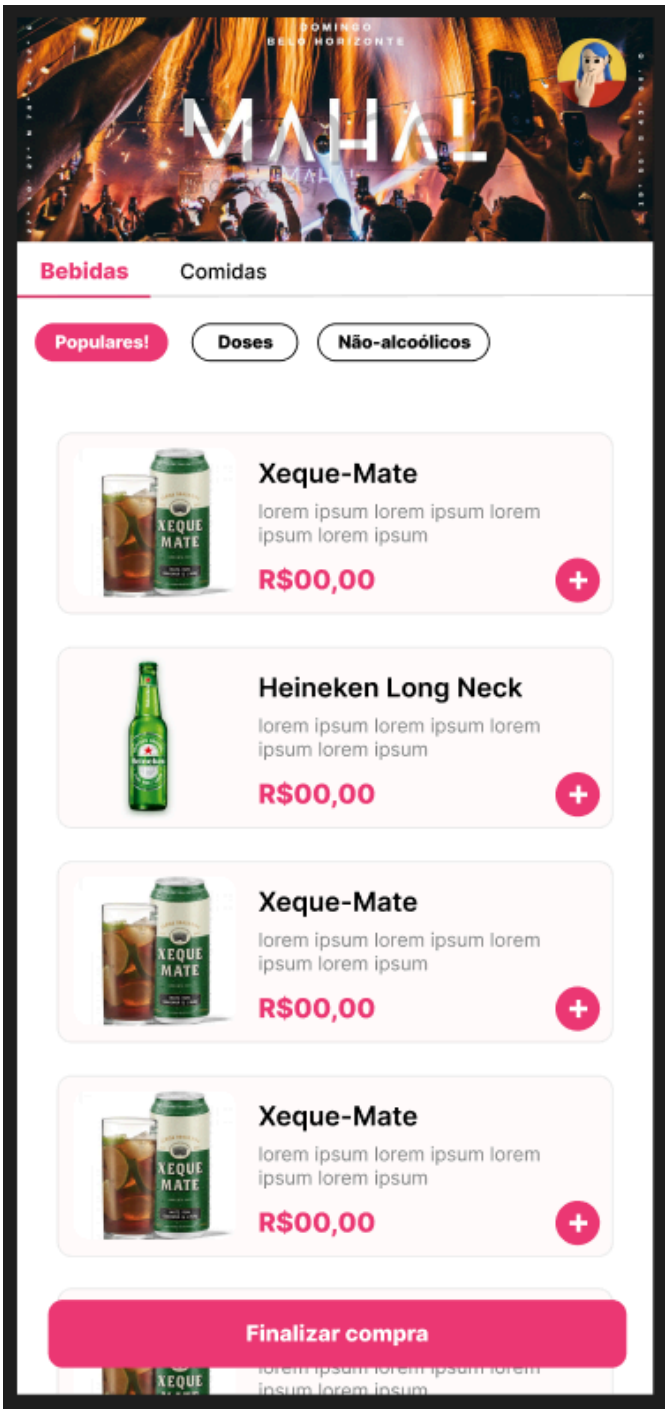

Figura 26. Tela de evento do consumidor

A tela representada na **Figura 27** apresenta o resumo da compra feita pelo consumidor, mostrando os itens adicionados ao carrinho, suas quantidades, valores individuais e o total. Essa etapa é essencial no processo de finalização do pedido e antecede a geração do pagamento.

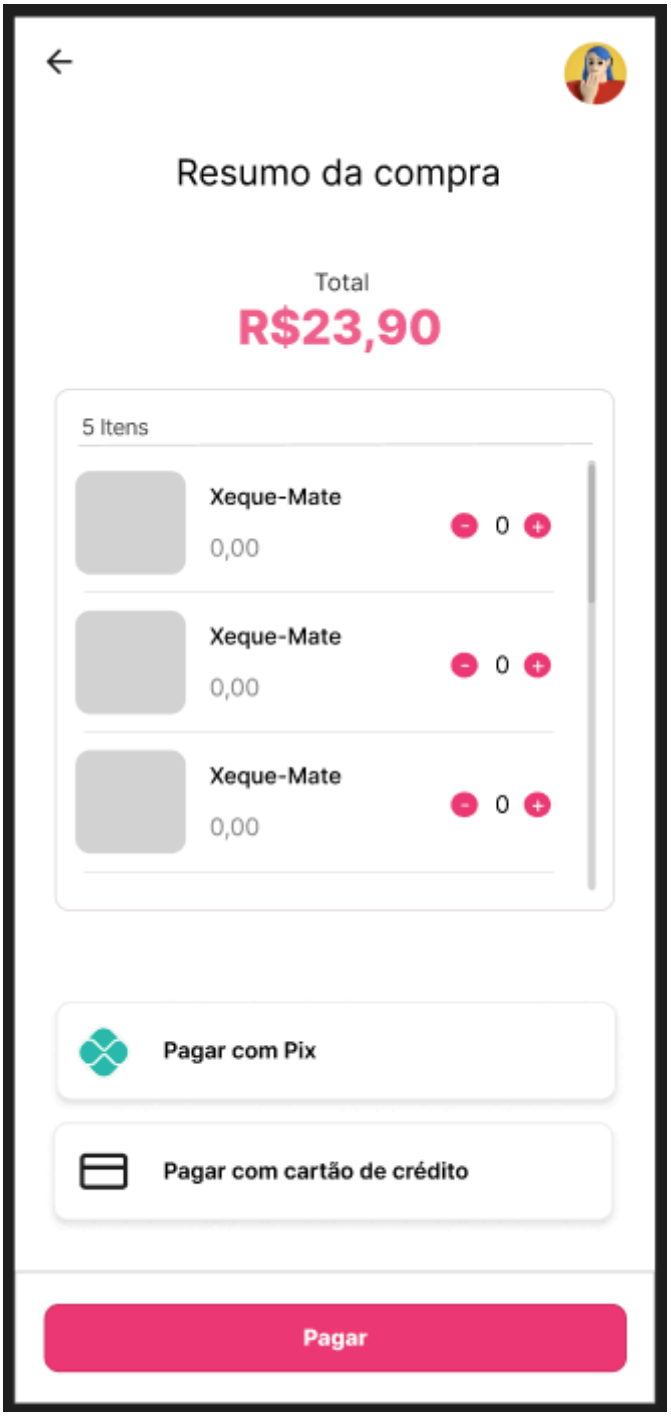

Figura 27. Tela de checkout do consumidor

A tela representada na **Figura 28** aparece para o consumidor após ele confirmar a forma de pagamento. Caso o usuário escolha pagar por Pix, por exemplo, um QR Code é gerado para que o consumidor realize o pagamento instantaneamente.

Figura 28. Tela de pagamento do consumidor

A tela representada na **Figura 29** confirma o sucesso do pagamento realizado pelo consumidor. É o encerramento do processo de compra.

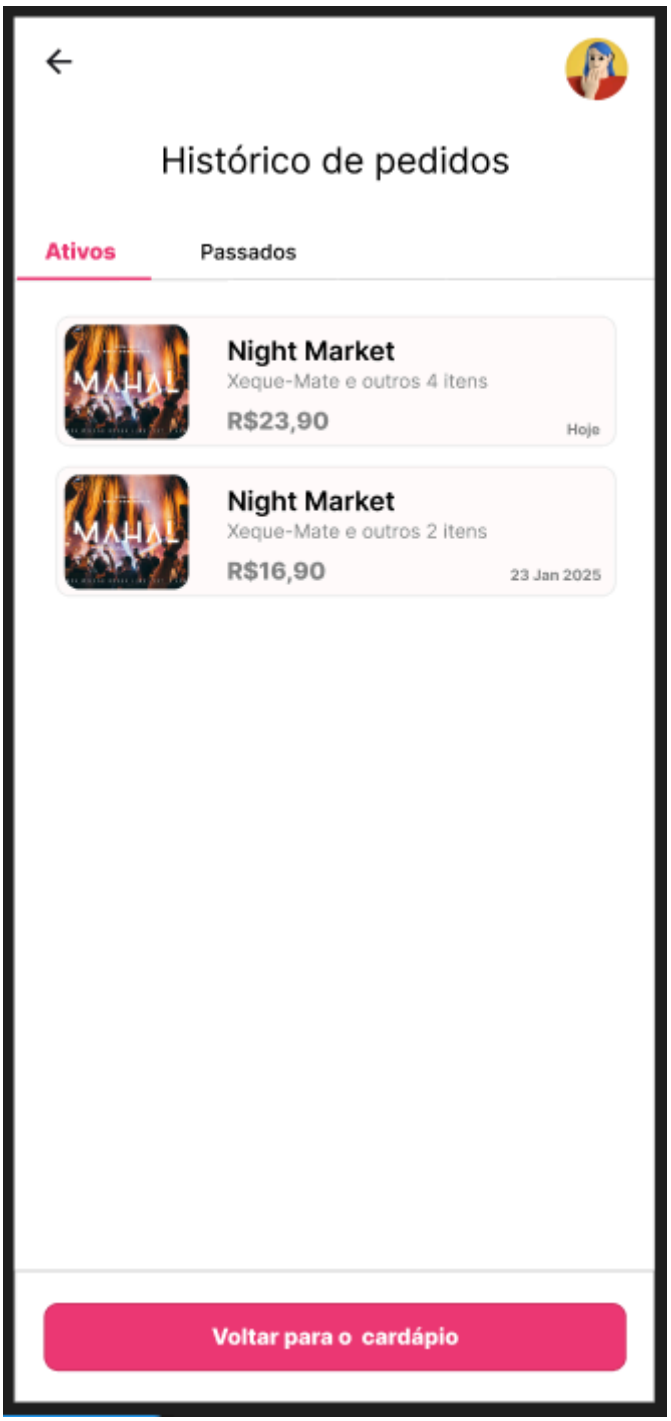

Figura 29. Tela de pagamento processado do consumidor

4.3. Esboço das Interfaces Usadas Apenas pelo Funcionário do Bar

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

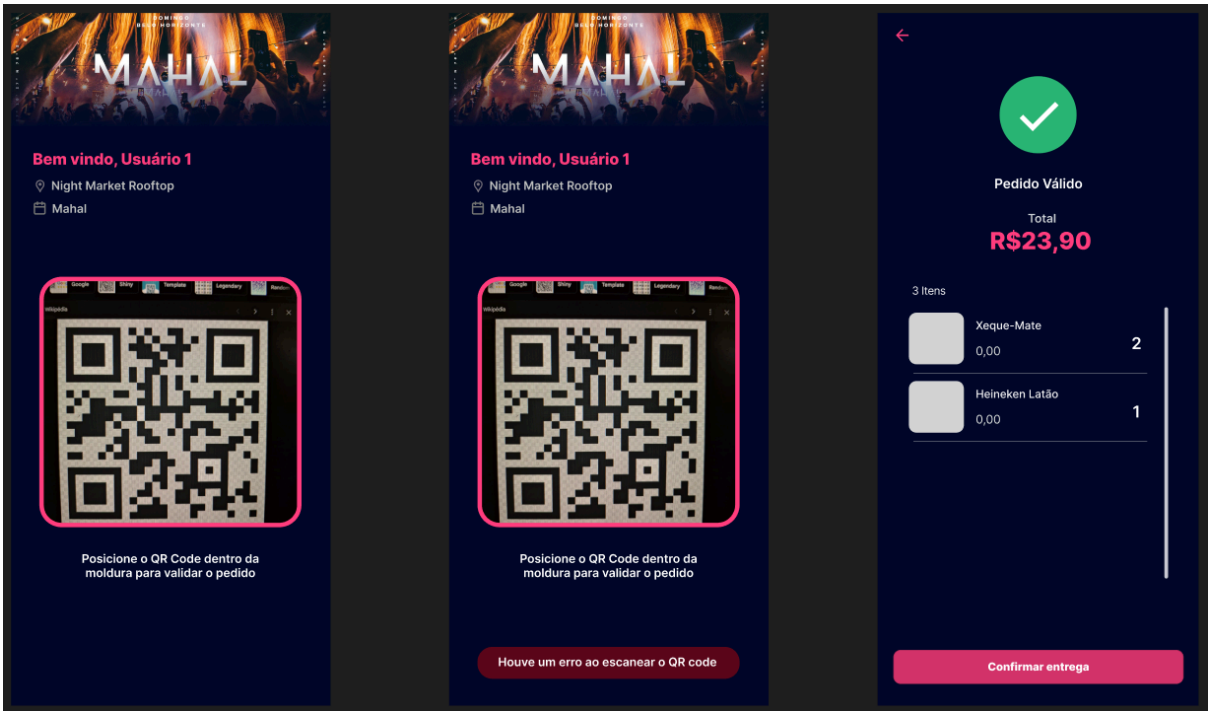

Figura 30. Tela de scanner do funcionário e organizador

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

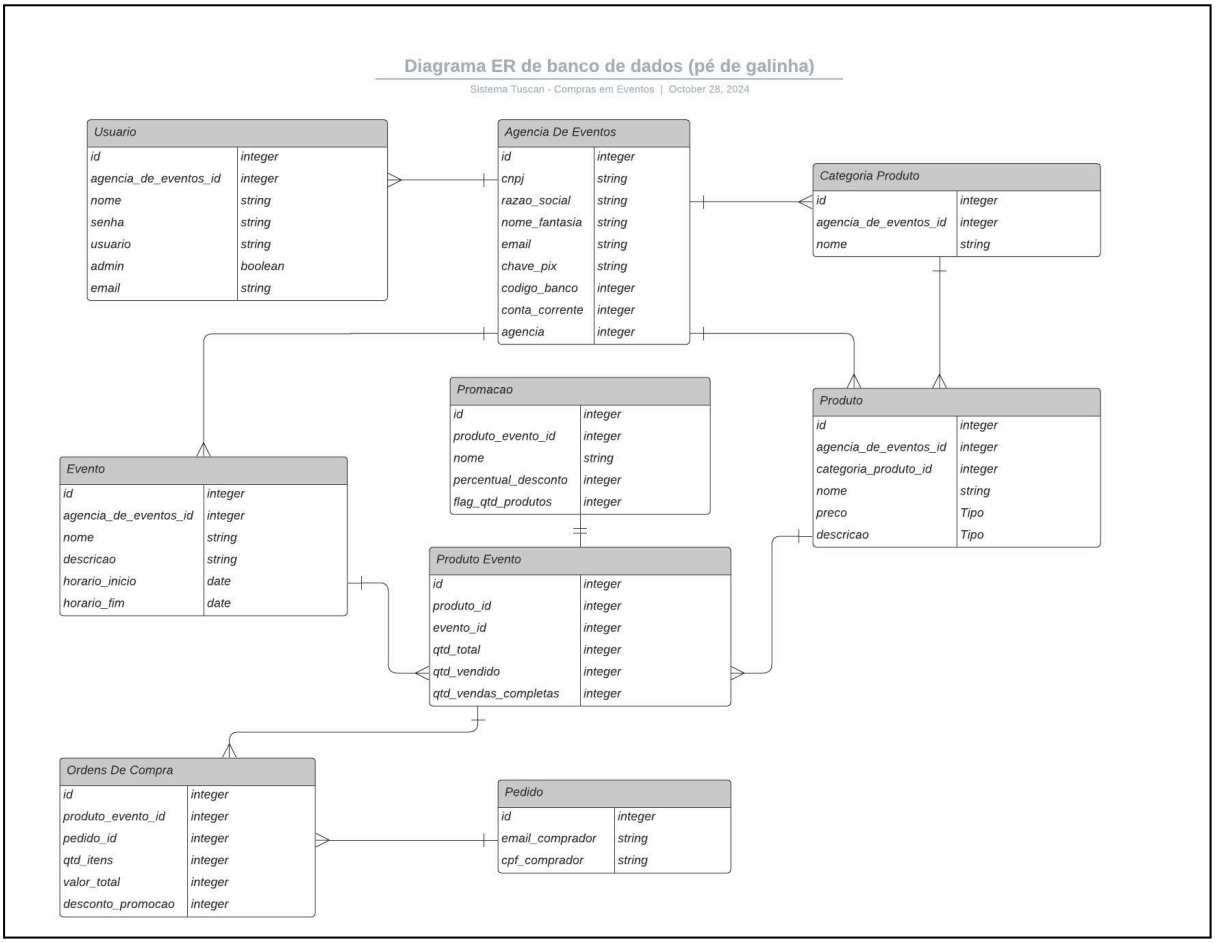

Figura 31. Diagrama de Entidade Relacionamento da Aplicação

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 testes unitários, que tem como objetivo validar o funcionamento das funções a serem implementadas, garantindo que os componentes que as utilizam funcionem da maneira correta.

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
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

Os casos de teste para o WebApp serão automatizados utilizando a biblioteca **PlayWright**, que simula as interações dos usuários finais por meio de **testes E2E**.

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

Essa necessidade garante que os consumidores 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 consumidor deve ter acesso ao evento e um QR Code válido disponível.
Ações	1. O consumidor escanear o QR Code usando o celular. 2. O sistema redireciona o consumidor para o cardápio digital.
Resultados	O consumidor visualiza o cardápio digital com os produtos disponíveis.

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

Identificador	Atualização automática do cardápio (TA1-02)
Pré-condição	O organizador altera o estoque ou preços de produtos.
Ações	1. O consumidor acessa o cardápio digital. 2. O consumidor confirma que os dados atualizados aparecem.

Resultados	O cardápio exibe informações atualizadas sem precisar de uma nova requisição.

Tabela 17. Caso de teste de aceitação “Atualização automática do cardápio”

6.1.2. Necessidade 2 - O consumidor do evento deve ser capaz de realizar uma compra online

Essa necessidade assegura que os consumidores 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 consumidor deve ter acesso ao cardápio digital.
Ações	1. O consumidor seleciona um item no cardápio. 2. O consumidor adiciona o item ao carrinho.
Resultados	O item é adicionado com quantidade e preço corretos.

Tabela 18. 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 consumidor acessa o carrinho. 2. O consumidor remove um item previamente adicionado.
Resultados	O item é removido, e o total é atualizado.

Tabela 19. 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 consumidor seleciona “Finalizar compra”. 2. O consumidor realiza o pagamento via Pix ou cartão de crédito.
Resultados	O pagamento é processado, e os QR Codes são gerados.

Tabela 20. 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 consumidor finaliza a compra. 2. O consumidor 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 21. 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 consumidores.

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 22. 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 23. Caso de teste de aceitação "Edição de produtos existentes"

Identificador	Consulta de produtos cadastrados (TA3-03)
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 24. Caso de teste de aceitação “Consulta de produtos cadastrados”

6.1.4. Necessidade 4 - O funcionário do bar deve validar QR Codes

Essa necessidade assegura que os funcionários do bar possam escanear e validar QR Codes apresentados pelos consumidores, 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 consumidor 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 25. Caso de teste de aceitação “Validação de QR Code”

Identificador	Rejeição de QR Code inválido (TA5-02)
Pré-condição	O consumidor 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 26. 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 27. 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 28. Caso de teste de aceitação "Consulta de QR Codes validados"

6.2. Testes Unitários

Nesta seção, são descritos os testes unitários propostos para o sistema implementado. Para cada caso de teste é usado um modelo de representação tal como o descrito na **Tabela 29**, esse que representa quais são as entradas e saídas esperadas para o teste.

A automatização dos testes unitários no WebApp será realizada por meio da biblioteca **Jest**.

Identificador	Teste unitário X.
Ações	1. Ação 1 2. Ação 2
Resultados	Resultados esperados.

Tabela 29. Exemplo de caso de teste unitário

Na **Tabela 30** é apresentado um caso de teste que tem como objetivo validar a adição de um novo item no carrinho.

Identificador	Teste Unitário 01 (TU-01)
Ações	1. Adicionar um novo item ao carrinho. 2. Atribuir a ID do evento ao carrinho
Resultados	1. Item adicionado. 2. Evento vinculado ao carrinho.

Tabela 30. Caso de teste unitário 1: AddItemCarrinho().

Na **Tabela 31** é apresentado um caso de teste que tem como objetivo validar o aumento da quantidade de um item dentro do carrinho.

Identificador	Teste Unitário 02 (TU-02)
Ações	1. Incrementar a quantidade de um item no carrinho.
Resultados	1. Quantidade incrementada.

Tabela 31. Caso de teste unitário 2: incrementItemAmount().

Na **Tabela 32** é apresentado um caso de teste que tem como objetivo validar a remoção de um item do carrinho.

Identificador	Teste Unitário 03 (TU-03)
Ações	1. Remover um item do carrinho.
Resultados	1. Item removido.

Tabela 32. Caso de teste unitário 3: removeItem().

Na **Tabela 33** é apresentado um caso de teste que tem como objetivo validar a diminuição da quantidade de um item dentro do carrinho.

Identificador	Teste Unitário 04 (TU-04)
Ações	1. Diminuir a quantidade de um item no carrinho.
Resultados	1. Quantidade diminuir.

Tabela 33. Caso de teste unitário 4: diminishItemAmount().

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 (34 e 35)** apresentam os cronogramas individuais de cada um dos alunos.

Cronograma - Bernardo Rohlfs	
Quinzena	**Atividades**
1ª Fevereiro (01-15)	● Configuração inicial do Supabase: estrutura do banco e “policies”.
2ª Fevereiro (16-28)	● Implementação do fluxo de cadastro e autenticação de usuários.

	● 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: 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: relatórios de vendas diárias e totais. ● Correções e ajustes.
2ª Abril (16-30)	● Testes automatizados: criação de testes unitários e de integração dentro do WebApp. ● Correções e ajustes.
1ª Maio (01-15)	● Correções e ajustes.
2ª Maio (16-31)	● Desenvolvimento de funcionalidades administrativas: gestão de usuários e permissões de acesso.
1ª Junho (01-15)	● Configuração do bundle de produção do Mobile App.

Tabela 34. Cronograma Bernardo Rohlfs

Cronograma - Eric Jardim	
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.
1ª Maio (01-15)	● 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 WebApp em produção.

Tabela 35. Cronograma Eric Jardim

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 frontend, garantindo a estabilidade do sistema.

Artefatos Produzidos

1. Código-fonte: Inclui 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

O desenvolvimento do sistema Tuscan representou uma jornada de aprendizado prático, tomada de decisões técnicas e adaptação a desafios reais de projeto. Ao longo do processo, a equipe precisou equilibrar a ambição de entregar uma solução inovadora para vendas em eventos com restrições de tempo, escopo e infraestrutura. Esta seção apresenta uma análise crítica e reflexiva sobre os principais acertos, dificuldades enfrentadas, decisões técnicas adotadas, e oportunidades de evolução para futuras versões do sistema.

- ❖ Repositório do Projeto no GitHub

8.1. Conquistas e Aprendizados

- Elaboração de uma plataforma robusta, eliminando caixas físicos e filas em eventos, utilizando QR Codes para as fichas digitais, pagamento (Pix, Google Wallet, Apple Pay e cartão) via Stripe e retirada via escaneamento.
- Aperfeiçoamento real de UX e fluxos de usuário para três perfis (consumidor, funcionário, organizador), reforçando a importância do design centrado no usuário.
- Aprofundamento prático em tecnologias modernas: React Native + Expo, Next.js, Supabase com políticas RLS, e integração com Stripe.
- Implementação de Row-Level Security (RLS) no Supabase para garantir o isolamento seguro dos dados conforme os perfis, atendendo à LGPD.
- Confiança nos pagamentos com Stripe, que oferece criptografia AES-256 e alta disponibilidade (~99.999% uptime).
- Estrutura clara de documentação técnica, incluindo README, diagramas UML e pipeline CI/CD, trazendo maturidade e transparência ao projeto.
- Implantação de uma arquitetura serverless, com hospedagens na Vercel e Supabase, que elimina a necessidade de manter ou escalar servidores manualmente, permitindo que o sistema lide com picos de acesso com eficiência e mantendo baixa complexidade de operação.

8.2. Experiências Negativas e Limitações

- Algumas funcionalidades do organizador (relatórios avançados, personalização de promoções) precisaram ser simplificadas por falta de tempo.
- Apesar da suíte de testes, faltaram testes automatizados ponta-a-ponta mais robustos, especialmente nos fluxos críticos. Implementamos testes unitários com Jest e testes E2E com Playwright baseados nos casos de aceitação, mas tivemos

dificuldades em simular o login via Google no fluxo do consumidor para testes E2E, o que impediu a cobertura completa desse fluxo.

- Limitação técnica: o sistema não é adequado para locais com conexão instável à Internet, como estádios, já que depende totalmente da conectividade em tempo real.

8.3. Segurança, Conformidade e Restrições

- O uso de RLS no PostgreSQL garantiu que cada perfil acessasse apenas os dados autorizados, reforçando a proteção de dados pessoais (LGPD) na camada de dados da aplicação.

- A integração com Stripe evita o armazenamento de dados sensíveis (como CPF/CNPJ ou números de cartão) na base de dados, transferindo isso para o ambiente PCI-DSS da Stripe. O projeto só guarda os tokens referentes às fichas digitais em QR Code.

- Planejamento futuro: conforme evoluir o fluxo de repasse financeiro ao organizador (via Stripe Connect), será necessário avaliar o registro e proteção de informações fiscais adicionais (CNPJ, dados bancários).

8.4. Integração com Serviços Externos: Pontos Importantes

- A alta disponibilidade (~99.999%) do Stripe deu suporte confiável às transações.

- Projetos futuros devem incluir mecanismos de tolerância, como filas locais ou cache, para mitigar falhas temporárias da Stripe ou Supabase.

8.5. Lições Aprendidas e Próximos Passos

- Arquitetura escalável desde o início evita retrabalho futuro — vale aplicar já em MVPs.

- Validação com usuários reais poderia ter sido mais frequente e formal (testes de usabilidade seriam bem-vindos).

- Testes E2E completos: soluções como seções de autenticação simuladas são recomendadas para melhorar a cobertura de testes.

- Recursos offline ou cache local são essenciais para eventos com internet instável.

8.6. Melhoria Contínua / Recomendações Futuras

- Desenvolver dashboards avançados para organizadores, com exportação de relatórios e métricas.
- Completar a cobertura de testes, incluindo regressão automática e rastreamento de cobertura.
- Avaliar e implementar modo offline / sincronização posterior para regiões com rede precária.
- Planejar o fluxo de repasses via Stripe Connect, adequar à LGPD e registrar dados fiscais de organizadores.

8.7. Conclusão

O projeto Tuscan cumpriu seu propósito de modernizar a experiência de consumo em eventos, com foco em segurança, UX e confiabilidade técnica. Apesar de limitações (tempo, testes, conectividade), o uso de RLS, criptografia e boas práticas com Stripe e CI/CD colocam o sistema em um ótimo nível de maturidade. As recomendações apresentadas alinham-se bem às restrições iniciais do Documento de Visão e demonstram potencial sólido para continuidade, expansão e profissionalização do sistema.