UNIVERSIDADE ESTADUAL PAULISTA JÚLIO DE MESQUITA FILHO

FACULDADE DE CIÊNCIAS

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Trabalho de Conclusão de Curso

Mihael Yuri Costa

Eventário: Plataforma para gerenciamento de eventos

Bauru, 2025

C837e

Costa, Mihael
Eventário : Plataforma para gerenciamento de eventos / Mihael Costa. -- Bauru, 2025
82 p.

Trabalho de conclusão de curso (Bacharelado - Sistemas de Informação) - Universidade Estadual Paulista (UNESP), Faculdade de Ciências, Bauru
Orientador: Higor Amario De Souza
Coorientador: José Remo Ferreira Brega

1. Administração. 2. Gestão do Conhecimento. 3. Eventos. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

Mihael Yuri Costa

Eventário: Plataforma para gerenciamento de eventos

Trabalho apresentado como exigência parcial para a conclusão do Curso de Bacharelado em Sistemas de Informação da Faculdade de Ciências – UNESP Campus Bauru.

Banca Examinadora

Prof. Dr. Higor Amario de Souza
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação.

Prof. Dr. José Remo Ferreira Brega
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

Prof. Dr. Douglas Rodrigues
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Computação

RESUMO

Este trabalho traz o desenvolvimento de um software para web que busca a gestão centralizada e otimizada de eventos de diferentes portes. A motivação está na superação das limitações de métodos tradicionais, como planilhas e e-mails dispersos, que frequentemente resultam em erros e ineficiência. O projeto visa oferecer funcionalidades como controle financeiro, gestão de fornecedores, convidados e suprimentos, permitindo flexibilidade e escalabilidade. O *Eventário* será implementado utilizando tecnologias modernas, como Vue.js no front-end, .NET no back-end e SQL Server para gerenciamento de dados, com infraestrutura em nuvem para garantir escalabilidade e fácil implantação. Além disso, o projeto aborda desafios como a integração entre módulos e a personalização para atender diferentes tipos de eventos, enquanto busca mitigar riscos por meio de planejamento detalhado. A plataforma se posiciona como uma solução acessível, especialmente para pequenos e médios organizadores, contribuindo para maior eficiência, redução de custos e aprimoramento na gestão de eventos.

Palavras-chave: Eventário, Eventos, Gestão.

ABSTRACT

This work proposes the development of a web platform called *Eventário*, aimed at centralized and optimized event management for different scales. The motivation lies in overcoming the limitations of traditional methods, such as scattered spreadsheets and emails, which often result in errors and inefficiencies. The project seeks to provide functionalities such as financial control, supplier management, guest administration, and supply tracking, offering flexibility and scalability. *Eventário* will be implemented using modern technologies, including Vue.js for the front-end, .NET for the back-end, and SQL Server for data management, with a cloud infrastructure to ensure scalability and easy deployment. Furthermore, the project addresses challenges such as module integration and customization to meet various event types while mitigating risks through detailed planning. The platform is positioned as an accessible solution, particularly for small and medium-sized organizers, contributing to greater efficiency, cost reduction, and enhanced event management.

Keywords: Eventário, Event, Management.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

1.1 Motivação

A organização e gestão de eventos, desde encontros universitários a grandes conferências, frequentemente enfrentam desafios devido à complexidade e multiplicidade de tarefas envolvidas. A utilização de métodos tradicionais, como planilhas e anotações manuais, pode levar a erros, retrabalho e ineficiências. Identificar uma solução que centralize e otimize esses processos é crucial para melhorar a eficiência e a eficácia na organização de eventos.

1.2 Solução

O projeto Eventario visa criar uma plataforma web integrada para o gerenciamento de eventos, oferecendo funcionalidades como controle financeiro, gestão de fornecedores e convidados, e organização de suprimentos em um único sistema. A plataforma será acessível online, permitindo uma gestão centralizada e eficiente que pode ser adaptada a eventos de diferentes portes.

1.3 Desafios

- **Flexibilidade**: Desenvolver regras e funcionalidades que atendam às diversas necessidades e complexidades de diferentes tipos de eventos.
- **Integração**: Garantir que todos os módulos do sistema interajam de forma eficaz e coerente.
- **Testes**: Realizar testes extensivos para assegurar que o sistema atenda a todos os requisitos e opere conforme esperado.

2. DETALHAMENTO DO PROBLEMA

A organização de eventos é um processo complexo, especialmente para pequenos e médios organizadores, que frequentemente enfrentam dificuldades devido à falta de ferramentas integradas. O uso de soluções dispersas, como planilhas e e-mails, compromete a eficiência e aumenta os riscos operacionais. Este capítulo aborda a relevância desse problema, seus impactos econômicos e os requisitos para uma solução integrada, como o Eventario, visando transformar a gestão de eventos.

2.1 Natureza do Problema

A organização de eventos é uma atividade que envolve muita complexidade e múltiplas tarefas, como gestão financeira, coordenação de fornecedores, controle de convidados e logística de materiais. No entanto, muitos organizadores, especialmente de pequenos e os de médio porte, ainda dependem de ferramentas não integradas, como planilhas dispersas e e-mails, o que resulta em falta de controle e falhas na comunicação. Essa abordagem confusa e ultrapassada torna difícil acompanhar o progresso de todas as atividades, gerando retrabalho, falta de agilidade e a possibilidade de erros.

2.2 Importância do Problema

O problema afeta diversos segmentos da sociedade, desde organizadores de eventos empresariais, universitários, culturais e esportivos até gestores de pequenos eventos familiares e comunitários. A falta de ferramentas adequadas para centralizar todas as áreas da organização desse evento impacta tanto os profissionais da indústria de eventos quanto os amadores que organizam celebrações casuais.

2.3 Impacto Econômico

Atualmente, a ineficiência na gestão de eventos tem um impacto econômico significativo. Falhas no controle financeiro, atraso na entrega de materiais e má coordenação de fornecedores podem aumentar os custos operacionais e comprometer o orçamento de eventos. Além disso, eventos mal administrados podem levar à insatisfação dos participantes ou até mesmo o cancelamento do evento, o que afeta a reputação dos organizadores e limita as oportunidades de crescimento do setor.

Com a implementação de uma solução integrada como o Eventario, espera-se uma redução nos custos operacionais por meio da otimização dos processos e melhor uso dos recursos. A

centralização da gestão permitirá um maior controle financeiro e logístico, evitando desperdícios e retrabalho. A longo prazo, isso contribuirá para a sustentabilidade econômica dos eventos, especialmente para os organizadores de menor porte.

2.4 Requisitos

2.4.1 Funcionais

- Centralização das áreas de controle do evento (financeiro, fornecedores, convidados e suprimentos) em uma plataforma acessível e intuitiva.
- Flexibilidade para personalizar a plataforma conforme o porte e o tipo de evento.

2.4.2 Desempenho

- A solução deve ser escalável para atender eventos de diferentes tamanhos, desde pequenas reuniões até grandes festivais.
- O sistema deve oferecer alta disponibilidade e confiabilidade, com respostas rápidas e sem interrupções durante o uso.

2.4.3 Custo

- O desenvolvimento e a manutenção da plataforma devem ser acessíveis, especialmente para pequenos organizadores e empresas que não possuem grandes orçamentos para eventos.
- O modelo de cobrança da solução pode ser baseado em um sistema de assinatura ou taxas por evento, garantindo a sobrevivência financeira da plataforma sem inviabilizar seu uso para organizadores de menor porte.

3. SOLUÇÕES EXISTENTES E OBJETIVOS DA PROPOSTA

3.1 Soluções Existentes

3.1.1 Eventbrite

O **Eventbrite** é uma das plataformas mais populares para organização e gerenciamento de eventos. Ele oferece diversos recursos, como venda de ingressos, check-in de participantes, e relatórios de desempenho. A plataforma é bastante utilizada para eventos de médio e grande porte, com ênfase em eventos pagos.

- **Aspectos Positivos**:
    - Interface intuitiva para os organizadores e participantes.
    - Sistema de venda de ingressos integrado com diversas opções de pagamento.
    - Ferramentas de marketing para promover os eventos.
- **Aspectos Negativos**:
    - Taxas elevadas para a utilização de funcionalidades mais avançadas, tornando-o menos viável para pequenos organizadores.
    - Focado principalmente em eventos pagos, o que limita o uso para eventos gratuitos ou internos.

3.1.2 Sympla

O **Sympla** é uma solução nacional muito utilizada no Brasil, voltada para a criação e gestão de eventos. Similar ao Eventbrite, ele oferece funcionalidades como venda de ingressos, controle de participantes e ferramentas de promoção de eventos.

- **Aspectos Positivos**:
    - Forte presença no mercado brasileiro e fácil integração com sistemas de pagamento locais.
    - Oferece funcionalidades para eventos online, ampliando seu escopo de uso.
    - Baixas taxas em comparação a plataformas internacionais.
- **Aspectos Negativos**:
    - Limitações na personalização para eventos específicos.
    - Focado em eventos de porte médio a grande, o que pode ser insuficiente para pequenas celebrações ou eventos familiares.

3.1.3 Zoho Backstage

O **Zoho Backstage** é uma plataforma que oferece ferramentas para planejamento e execução de eventos, com foco em integração com outras ferramentas da Zoho, como o CRM e software de marketing.

- **Aspectos Positivos**:
    - Excelente integração com outros produtos do ecossistema Zoho, o que facilita a gestão de informações de marketing, CRM e finanças.
    - Ferramentas colaborativas para equipes que organizam eventos juntos.
- **Aspectos Negativos**:
    - Custo elevado para pequenas empresas ou indivíduos.
    - Complexidade de integração com a suíte completa da Zoho, que pode exigir mais esforço de configuração.

3.2 Análise Crítica das Soluções Existentes

A análise das soluções comerciais e acadêmicas mostra que, embora existam plataformas robustas para a gestão de eventos, elas tendem a ser voltadas para grandes produções ou têm custos elevados para pequenos organizadores. Plataformas como Eventbrite e Sympla são altamente competentes, mas se concentram principalmente na venda de ingressos e no gerenciamento de eventos pagos, o que limita sua aplicabilidade para eventos gratuitos ou com necessidades personalizadas. Além disso, as soluções acadêmicas apresentam restrições quanto à funcionalidade e à usabilidade.

Dessa forma, o **Eventario** surge como uma solução mais versátil e acessível, com foco em atender eventos de todos os portes. O principal diferencial é a possibilidade de centralizar a gestão financeira, fornecedores e suprimentos, adaptando-se às necessidades específicas de cada tipo de evento, sem os custos elevados das soluções comerciais.

3.3 Objetivos da Proposta

- Desenvolver uma plataforma que seja acessível para organizadores de eventos pequenos, médios e grandes.
- Oferecer uma solução mais abrangente, que centralize o controle financeiro, gestão de fornecedores, convidados e suprimentos.

- Garantir a flexibilidade do sistema para se adaptar a eventos gratuitos, corporativos, familiares ou pagos.
- Minimizar os custos operacionais para organizadores, com uma proposta de preços competitiva e ajustada ao porte do evento.

4. REFERENCIAL TEÓRICO

A gestão de eventos constitui uma disciplina multidisciplinar complexa que integra conhecimentos de administração, tecnologia da informação, logística, marketing, comunicação e finanças para coordenar eficientemente todos os aspectos relacionados à organização e execução de eventos de diferentes naturezas, portes e finalidades. A complexidade inerente à gestão de eventos deriva da necessidade de coordenar simultaneamente múltiplas variáveis interdependentes, incluindo aspectos temporais limitados e frequentemente inflexíveis, recursos financeiros tipicamente restritos e sujeitos a controle rigoroso, expectativas diversificadas e muitas vezes conflitantes dos organizadores envolvidos, variáveis ambientais e logísticas imprevisíveis, além de requisitos regulamentares e de segurança cada vez mais rigorosos e complexos.

4.1 Gestão de Eventos

4.1.1 Conceitos Fundamentais e Contexto Universitário

Segundo Ivo, Marin e De Souza (2014), a gestão de eventos em instituições universitárias representa um campo complexo que demanda planejamento sistemático e coordenação cuidadosa de diversos recursos. Os autores destacam que eventos bem organizados contribuem significativamente para o fortalecimento da imagem institucional e o cumprimento dos objetivos educacionais das universidades.

A complexidade da gestão de eventos universitários surge da necessidade de coordenar aspectos acadêmicos, administrativos, logísticos e financeiros simultaneamente. Os organizadores enfrentam desafios relacionados à gestão de recursos limitados, coordenação de múltiplos envolvidos (professores, estudantes, funcionários e comunidade externa), manutenção de padrões de qualidade com orçamentos restritos, além do cumprimento de regulamentações institucionais (IVO; MARIN; DE SOUZA, 2014).

Os pesquisadores observaram que a utilização de metodologias estruturadas e ferramentas tecnológicas adequadas é fundamental para superar essas limitações, garantindo o sucesso operacional dos eventos. Instituições que implementaram sistemas organizacionais robustos conseguiram realizar eventos de maior qualidade, com custos reduzidos e maior satisfação dos participantes. A documentação de processos e criação de bancos de conhecimento também contribuem para a melhoria contínua das práticas organizacionais (IVO; MARIN; DE SOUZA, 2014).

4.1.2 Dimensões Teóricas e Práticas da Gestão de Eventos

Mallen e Adams (2013) analisaram as dimensões fundamentais da gestão de eventos esportivos, recreativos e turísticos, desenvolvendo um framework que também se aplica a eventos corporativos, acadêmicos e culturais. Os autores identificaram cinco pilares essenciais para a gestão eficaz de eventos: planejamento estratégico, gestão financeira, coordenação logística, administração de recursos humanos e sistemas de monitoramento contínuo.

O planejamento estratégico constitui a base para o sucesso de qualquer evento. Esta fase envolve a definição clara de objetivos, análise do público-alvo, estabelecimento de cronogramas realistas, desenvolvimento de estratégias de contingência e criação de mecanismos de comunicação eficazes entre os envolvidos. Segundo Mallen e Adams (2013), eventos que negligenciam o planejamento adequado frequentemente enfrentam problemas operacionais, extrapolação orçamentária, atrasos e insatisfação dos participantes.

A gestão financeira emerge como elemento crítico que permeia todas as fases do evento. O controle rigoroso de receitas e despesas, combinado com monitoramento contínuo do fluxo de caixa, constitui fator determinante para a viabilidade financeira dos eventos. Os autores destacam que sistemas robustos de controle financeiro possibilitam a identificação precoce de desvios orçamentários e facilitam decisões corretivas antes que os problemas se tornem críticos.

O estudo demonstrou que organizadores que implementaram controles financeiros rigorosos conseguiram manter eventos dentro do orçamento com maior frequência e maximizar o retorno sobre investimento. A transparência financeira também contribui para manter a confiança de patrocinadores e stakeholders, facilitando a obtenção de recursos para eventos futuros.

4.1.3 Coordenação Logística e Gestão de Recursos

A coordenação logística constitui aspecto operacional crítico que determina em grande medida a qualidade da experiência dos participantes e o sucesso geral do evento. Mallen e Adams (2013) analisaram detalhadamente os múltiplos componentes da gestão logística, incluindo seleção e preparação de locais apropriados, coordenação de transporte e acomodações, gestão de equipamentos e materiais necessários, coordenação de serviços de alimentação e bebidas, implementação de medidas de segurança e emergência, e estabelecimento de sistemas eficazes de comunicação e sinalização durante o evento.

Os autores identificaram que falhas na coordenação logística frequentemente resultaram em experiências negativas para participantes, comprometimento da imagem do evento e da organização responsável, custos adicionais não planejados, e potencial exposição a riscos de segurança e responsabilidade legal. A pesquisa demonstrou que organizações que investiram adequadamente em planejamento logístico detalhado, realizaram visitas técnicas minuciosas aos locais, desenvolveram planos de contingência abrangentes, e estabeleceram protocolos claros de comunicação conseguiram executar eventos mais fluidos, com menor incidência de problemas operacionais e maior satisfação geral dos participantes.

Mallen e Adams (2013) enfatizaram que a gestão eficaz de recursos humanos constituiu elemento fundamental para sucesso operacional, considerando que eventos dependeram criticamente do desempenho coordenado de equipes diversificadas com diferentes níveis de experiência, responsabilidades específicas e expectativas variadas. Os autores destacaram a importância de recrutamento adequado de pessoal qualificado, treinamento específico para funções especializadas, definição clara de responsabilidades e hierarquias, estabelecimento de canais eficazes de comunicação, e implementação de sistemas de motivação e reconhecimento apropriados.

4.1.4 Limitações dos Métodos Tradicionais de Gestão

A utilização de métodos tradicionais na gestão de eventos apresenta limitações que impactam negativamente a eficiência operacional e a qualidade dos resultados. Mallen e Adams (2013) observaram que a dependência de planilhas dispersas, documentos isolados, sistemas de comunicação fragmentados e processos manuais frequentemente resulta em falhas de coordenação, duplicação de esforços, perda de informações e dificuldades para rastreamento de progresso.

Essas limitações tornam-se particularmente problemáticas em eventos de maior porte, onde a quantidade de informações, o número de organizadores envolvidos e a diversidade de processos excedem a capacidade de controle manual eficaz. Mallen e Adams (2013) documentaram casos em que falhas na gestão de informações resultaram em custos adicionais, atrasos em cronogramas, comprometimento da qualidade dos serviços e impactos negativos na reputação das organizações.

Ivo, Marin e De Souza (2014) identificaram que as dificuldades na gestão de eventos universitários decorrem da falta de integração entre aspectos acadêmicos, administrativos,

financeiros e logísticos. A utilização de ferramentas isoladas para controle financeiro, gestão de participantes e coordenação de fornecedores cria silos informacionais que dificultam a obtenção de uma visão abrangente do evento e comprometem a capacidade de tomada de decisões.

A ausência de centralização das informações resulta em aumento do tempo necessário para preparação de relatórios e análises de desempenho. Esta fragmentação impede a identificação de padrões e oportunidades de melhoria, limitando o aprendizado organizacional. Ivo, Marin e De Souza (2014) destacaram que instituições que implementaram sistemas integrados observaram melhorias substanciais em eficiência operacional, qualidade de resultados e satisfação dos organizadores.

4.2 Tecnologias de Desenvolvimento Web

4.2.1 Framework Vue.js para Desenvolvimento Front-end

Incau (2017) analisou as características e vantagens do framework Vue.js para construção de aplicações web modernas e escaláveis. O autor identificou que Vue.js combina facilidade de aprendizado com poder de desenvolvimento, oferecendo arquitetura reativa que facilita a criação de interfaces dinâmicas e interativas.

O framework caracteriza-se pela utilização de componentes reutilizáveis, sistema de reatividade bidirecional eficiente e capacidade de integração gradual com projetos existentes. Essas características tornam Vue.js adequado para aplicações empresariais complexas que requerem interfaces sofisticadas e manutenibilidade de longo prazo. Incau (2017) demonstrou que Vue.js oferece curva de aprendizado mais suave comparado a outros frameworks, permitindo que desenvolvedores com diferentes níveis de experiência contribuam efetivamente para projetos em períodos relativamente curtos.

A arquitetura baseada em componentes proporciona benefícios significativos para sistemas de gestão empresarial. Esta abordagem modular permite a decomposição de interfaces complexas em unidades independentes e reutilizáveis, facilitando atividades de manutenção, implementação de testes e evolução das funcionalidades. A separação clara entre lógica de apresentação, gerenciamento de estado e manipulação de dados promove organização adequada do código e facilita a colaboração entre desenvolvedores (INCAU, 2017).

O sistema de reatividade do Vue.js constitui diferencial importante para aplicações que requerem atualizações frequentes da interface. Este sistema garante sincronização automática entre estado da aplicação e elementos visuais, eliminando a manipulação manual do DOM e reduzindo a complexidade do código front-end. Em sistemas de gestão de eventos, onde informações como status financeiro, listas de participantes e indicadores de performance necessitam atualização constante, a reatividade automática do Vue.js elimina inconsistências e melhora significativamente a experiência do usuário (INCAU, 2017).

4.2.2 Ecossistema e Ferramentas do Vue.js

O ecossistema abrangente do Vue.js oferece conjunto rico e integrado de ferramentas, bibliotecas e recursos que aceleraram desenvolvimento e melhoraram qualidade de aplicações complexas. Incau (2017) analisou componentes essenciais do ecossistema, incluindo *Vue Router* para gerenciamento sofisticado de rotas e navegação em *single-page applications, Vuex* para gerenciamento centralizado e previsível de estado global da aplicação, *Vue CLI* para *scaffolding* automatizado e configuração otimizada de projetos, e *Vue DevTools* para *debugging* avançado e análise de performance em tempo de desenvolvimento.

O *Vue Router* fornece capacidades avançadas para construção de aplicações de página única (SPA) robustas, incluindo roteamento dinâmico baseado em parâmetros, *lazy loading* de componentes para otimização de performance, guardas de navegação para controle de acesso e validação, e suporte nativo para histórico de navegação e *deep linking*. Estas funcionalidades mostram-se essenciais para sistemas de gestão que requereram navegação fluida entre diferentes módulos funcionais, controle granular de permissões de acesso, e capacidade de *bookmarking* e compartilhamento de URLs específicas.

Incau (2017) enfatizou que *Vuex* implementou padrão *Flux/Redux* adaptado especificamente para Vue.js, fornecendo gerenciamento de estado centralizado, previsível e debuggável para aplicações complexas. Esta solução mostra-se particularmente valiosa para sistemas de gestão de eventos, onde estado da aplicação inclui informações sobre múltiplos eventos simultaneamente ativos, dados de usuários com diferentes níveis de permissão, cache de dados frequentemente acessados, e sincronização entre múltiplas visualizações da mesma informação. O padrão de gerenciamento de estado centralizado facilita implementação de funcionalidades como *undo/redo*, persistência de estado entre sessões, e sincronização com APIs backend de forma consistente e confiável.

4.2.3 Performance e Otimização no Vue.js

As características de performance do Vue.js constituem aspectos fundamentais que influenciam sua adequação para aplicações empresariais que requerem responsividade superior e escalabilidade eficaz. Incau (2017) analisou múltiplas dimensões de performance, incluindo tamanho otimizado do *bundle* final, eficiência do *virtual DOM implementation*, capacidades de *tree-shaking* para eliminação de código não utilizado, e suporte nativo para *lazy loading* de componentes e recursos. O autor demonstrou que Vue.js conseguiu equilibrar eficazmente facilidade de desenvolvimento com performance de *runtime* otimizada, oferecendo aplicações finais com *footprint* reduzido e responsividade superior.

O virtual DOM do Vue.js implementa algoritmos de *diff* otimizados que minimizam manipulações custosas do DOM real, resultando em atualizações de interface mais rápidas e fluidas. Incau (2017) explica que esta implementação utiliza heurísticas inteligentes para identificar mudanças mínimas necessárias, aplicou *batching* automático de updates para reduzir *reflows* e *repaints*, e ofereceu escape *hatches* para otimizações manuais em cenários específicos de alta performance. Para sistemas de gestão de eventos com dashboards complexos e atualizações frequentes de dados, estas otimizações traduziram-se em interfaces mais responsivas e menor consumo de recursos computacionais.

O autor destacou também capacidades avançadas de *code-splitting* e *lazy loading* que permitiram carregamento incremental de funcionalidades conforme necessário, reduzindo tempo inicial de carregamento da aplicação e melhorando percepção de performance pelos usuários. Incau (2017) demonstrou que estas técnicas se mostraram particularmente eficazes para sistemas modulares de gestão, onde usuários tipicamente utilizaram subconjuntos específicos de funcionalidades em cada sessão, permitindo otimização de recursos de rede e memória através de carregamento sob demanda de módulos específicos.

4.2.4 Desenvolvimento Backend com .NET

Carmo (2023) conduziu estudo comparativo abrangente e metodologicamente rigoroso entre principais tecnologias de backend disponíveis no mercado contemporâneo, incluindo análise técnica detalhada das características, vantagens competitivas e casos de uso apropriados do ASP.NET Core para desenvolvimento de APIs web modernas, escaláveis e de alta performance. Segundo o autor, baseando-se em benchmarks de performance, análise de produtividade de desenvolvimento, e avaliação de ecossistema de ferramentas disponíveis, o .NET demonstrou

superioridade consistente em aspectos críticos como performance de *runtime* otimizada, escalabilidade horizontal e vertical eficaz, recursos avançados de segurança integrados, facilidade de manutenção e evolução de código, além de robustez operacional comprovada em ambientes de produção exigentes.

Estas características técnicas superiores constituem escolha apropriada e estratégica para sistemas empresariais que requerem confiabilidade operacional elevada, disponibilidade contínua, capacidade de processamento de volumes significativos de dados e transações, e integração eficaz com infraestruturas tecnológicas existentes. Carmo (2023) demonstrou através de análise quantitativa que o .NET ofereceu recursos técnicos avançados como middleware configurável e extensível para implementação de *cross-cutting concerns*, sistema robusto de injeção de dependências nativa que promoveu testabilidade e flexibilidade arquitetural, sistema de configuração flexível e hierárquico que facilitou *deployment* em diferentes ambientes, e suporte abrangente e integrado para desenvolvimento orientado a testes em múltiplos níveis de granularidade.

A evolução estratégica da plataforma .NET para suporte multiplataforma representa marco tecnológico significativo que amplia substancialmente suas possibilidades de aplicação e adoção em contextos organizacionais diversos. Carmo (2023) destacou que o .NET Core introduziu melhorias substanciais e mensuráveis em performance de *runtime*, redução significativa do *footprint* de memória necessário, suporte nativo e otimizado para containers Docker e orquestradores como Kubernetes, facilidade de implantação automatizada em diferentes ambientes operacionais (Windows, Linux, macOS), e capacidades avançadas de observabilidade e monitoramento integradas na plataforma.

Estas características técnicas e operacionais tornam a plataforma .NET particularmente adequada e competitiva para desenvolvimento de arquiteturas modernas baseadas em microsserviços, APIs RESTful de alta performance, aplicações distribuídas complexas, e sistemas que necessitaram operar eficientemente em infraestruturas heterogêneas, híbridas e multi-cloud. Carmo (2023) analisou casos de implementação que demonstraram capacidade superior da plataforma para escalonamento horizontal automático, publicações sem percas de tempo, e integração eficaz com ferramentas de DevOps e pipelines de CI/CD modernos.

4.2.5 Ecossistema e Ferramentas .NET

O ecossistema abrangente e maduro do .NET oferece conjunto extenso de ferramentas, bibliotecas e recursos que aceleraram significativamente desenvolvimento de funcionalidades complexas e especializadas. Carmo (2023) analisou componentes críticos do ecossistema, incluindo Entity Framework Core para mapeamento objeto-relacional avançado e gestão de acesso a dados, *SignalR* para implementação de comunicação em tempo real bidirecional, bibliotecas especializadas para processamento de documentos em múltiplos formatos, *SDKs* para integração com serviços externos e APIs de terceiros, e ferramentas integradas para implementação de padrões de segurança contemporâneos incluindo autenticação, autorização e criptografia.

O autor demonstrou que esta riqueza e maturidade de recursos disponíveis permitiu que desenvolvedores focassem prioritariamente na implementação de regras de negócio específicas e diferenciais competitivos, reduzindo substancialmente o tempo necessário para desenvolvimento de funcionalidades infraestruturais comuns, minimizando riscos associados a implementações customizadas de componentes críticos, e garantindo aderência a melhores práticas e padrões de segurança estabelecidos pela indústria.

Carmo (2023) destacou especificamente as capacidades avançadas do Entity Framework Core para gestão de acesso a dados em sistemas complexos, incluindo suporte para múltiplos provedores de banco de dados, migrações automatizadas de esquema com controle de versão, otimização automática de consultas com análise de planos de execução, suporte nativo para operações assíncronas de alta performance, e capacidades avançadas de cache e *lazy loading* que otimizaram performance de aplicações data-intensive. Para sistemas de gestão de eventos que necessitam gerenciar volumes significativos de dados relacionais complexos, estas funcionalidades traduzem-se em desenvolvimento mais rápido, código mais mantível, e performance superior de runtime.

4.2.6 Arquitetura e Padrões .NET

A arquitetura modular e extensível do ASP.NET Core facilita implementação de padrões arquiteturais modernos e boas práticas de desenvolvimento. Carmo (2023) analisou suporte nativo para arquiteturas em camadas, implementação de padrões como *Dependency Injection, Repository, Unit of Work*, e *CQRS*, além de facilidades para implementação de *cross-cutting*

concerns como *logging, caching, validation, e error handling* através de middleware e filtros configuráveis.

O autor demonstrou que o sistema de middleware do ASP.NET Core ofereceu flexibilidade excepcional para implementação de funcionalidades transversais, permitindo composição de pipeline de processamento customizado para diferentes tipos de requisições, implementação de aspectos como autenticação, autorização*, rate limiting*, e *request/respon*se *transformation* de forma modular e reutilizável. Esta abordagem mostrou-se particularmente valiosa para sistemas de gestão que requereram comportamentos diferenciados baseados em contexto de usuário, tipo de operação, ou características específicas de cada requisição.

Carmo (2023) enfatizou que a integração nativa com ferramentas de observabilidade modernas facilitou implementação de monitoramento abrangente, *logging* estruturado, *tracing* distribuído, e métricas de performance detalhadas. Para sistemas críticos de gestão de eventos, estas capacidades traduzem-se em maior visibilidade operacional, identificação proativa de problemas de performance, e capacidade superior de *debugging* e *troubleshooting* em ambientes de produção complexos.

4.2.7 Gerenciamento de Dados com SQL Server

Pessoa (2012) realizou estudo comparativo sistemático e metodologicamente rigoroso entre diferentes tecnologias de banco de dados relacionais e não-relacionais disponíveis no mercado, incluindo avaliação técnica específica e detalhada das capacidades, vantagens competitivas e casos de uso apropriados do SQL Server para suporte eficaz a aplicações web empresariais críticas. Segundo os autores, baseando-se em benchmarks de performance abrangentes, análise de recursos disponíveis, avaliação de custos totais de propriedade, e estudos de casos de implementação em ambientes de produção exigentes, o SQL Server demonstrou características técnicas e operacionais superiores em termos de performance consistente sob cargas variáveis, confiabilidade operacional comprovada em ambientes críticos, recursos avançados de backup e recuperação para garantia de continuidade de negócios, além de ferramentas sofisticadas para otimização automática de consultas e monitoramento proativo de performance.

O sistema oferece recursos empresariais avançados e abrangentes como alta disponibilidade através de *Always On Availability Groups*, replicação de dados configurável para diferentes cenários de uso, particionamento horizontal e vertical de tabelas para otimização de performance, índices especializados incluindo *columnstore* e *in-memory OLTP*, e capacidades

avançadas de backup e *point-in-time recovery* que o torna adequado e competitivo para aplicações críticas com requisitos rigorosos de disponibilidade, consistência, durabilidade e performance previsível. Pessoa (2012) demonstrou através de análise quantitativa que estas características técnicas se traduziram em maior confiabilidade operacional, menor tempo de inatividade não planejada, e capacidade superior de recuperação de desastres comparado a alternativas avaliadas.

A integração nativa e otimizada entre SQL Server e tecnologias Microsoft constitui vantagem significativa e estratégica para desenvolvimento de soluções integradas e coesas. Pessoa (2012) identificou que esta integração profunda resultou em redução substancial da complexidade de configuração e administração, simplificação significativa de procedimentos de implantação e manutenção, otimização automática de comunicação entre camadas da aplicação através de protocolos nativos eficientes, e redução de latência e overhead de comunicação entre componentes do sistema.

Os autores destacaram recursos específicos como *connection pooling* otimizado e configurável, suporte nativo para tipos de dados .NET que eliminaram necessidade de conversões custosas, ferramentas integradas de monitoramento e análise de performance que facilitaram identificação e resolução de gargalos, e capacidades avançadas de segurança que incluíram *encryption at rest, transparent data encryption*, e *row-level security* para proteção granular de informações sensíveis. Para sistemas de gestão de eventos que necessitam processar informações financeiras sensíveis, dados pessoais de participantes, e contratos com fornecedores, estas capacidades de segurança mostram-se essenciais para conformidade regulatória e proteção adequada de informações críticas.

4.2.8 Recursos Avançados e Performance do SQL Server

O SQL Server oferece capacidades avançadas de análise e geração de relatórios que se mostram particularmente valiosas para sistemas de gestão empresarial complexos. Pessoa (2012) analisou funcionalidades como *SQL Server Reporting Services* (SSRS) para geração automatizada de relatórios complexos e customizáveis, *SQL Server Analysis Services* (SSAS) para análise dimensional avançada de dados históricos e operacionais, e recursos nativos de *Business Intelligence* que permitiram extração de insights valiosos dos dados operacionais sem necessidade de ferramentas externas adicionais ou integrações complexas com sistemas de terceiros.

Estas capacidades analíticas avançadas possibilitam implementação eficaz de dashboards executivos interativos, relatórios automatizados com distribuição programada, análises preditivas baseadas em dados históricos, e sistemas de alertas proativos baseados em *thresholds* configuráveis. Para sistemas de gestão de eventos, estas funcionalidades traduzem-se em capacidade superior de monitoramento de indicadores-chave de performance, identificação de tendências e padrões em dados operacionais, e suporte mais eficaz para tomada de decisões estratégicas baseadas em evidências quantitativas.

Pessoa (2012) destacou também as capacidades de otimização automática do SQL Server, incluindo *query optimizer* avançado que analisou estatísticas de dados em tempo real para seleção de planos de execução otimizados, *automatic tuning* que identificou e implementou melhorias de performance sem intervenção manual, e *adaptive query processing* que ajustou dinamicamente estratégias de execução baseadas em características dos dados processados. Estas funcionalidades resultaram em performance mais previsível e consistente, redução de necessidade de *tuning* manual especializado, e capacidade superior de adaptação a mudanças nos padrões de uso e volumes de dados.

4.2.9 Integração e Escalabilidade do SQL Server

As capacidades de escalabilidade e integração do SQL Server demonstram-se fundamentais para suporte a aplicações empresariais em crescimento. Pessoa (2012) analisou recursos como *scale-out* através de leitura de réplicas distribuídas, *scale-up* através de suporte para hardware de alta capacidade, e capacidades híbridas que permitiram combinação de recursos *on-premises* e cloud de forma transparente e otimizada. O sistema ofereceu também recursos avançados de cache distribuído, *connection pooling* inteligente, e *load balancing* automático que otimizaram utilização de recursos e mantiveram performance consistente sob cargas variáveis.

Os autores identificaram que a integração nativa com *Azure cloud platform* proporcionou opções flexíveis de publicação e escalonamento, incluindo *Azure SQL Database* para soluções completamente gerenciadas, *Azure SQL Managed Instance* para compatibilidade máxima com *SQL Server on-premises*, e híbridos que permitiram migração gradual e reversível para cloud conforme necessidades organizacionais. Esta flexibilidade mostra-se particularmente valiosa para organizações que necessitam balancear requisitos de controle, compliance, custo e escalabilidade.

Pessoa (2012) enfatizou que as capacidades de recuperação de desastres e continuação de negócio do SQL Server incluíram *Always On Availability Groups* para *failover* automático e transparente, geo-replicação para proteção contra desastres regionais, e restauração point-in-time para recuperação granular de dados. Para sistemas de gestão de eventos que contém informações críticas de negócio, contratos importantes, e dados financeiros sensíveis, estas funcionalidades constituem requisitos fundamentais para garantia de continuidade operacional e proteção adequada de ativos informacionais.

4.3 Arquiteturas de Software Modernas

4.3.1 Fundamentos da Arquitetura Limpa

A arquitetura limpa emerge como paradigma arquitetural fundamental para desenvolvimento de sistemas de software mantíveis, testáveis, evolutivos e independentes de tecnologias específicas. Esta abordagem conceitual basea-se na separação rigorosa e sistemática de responsabilidades através de camadas bem definidas e hierarquicamente organizadas, onde dependências apontam sempre e exclusivamente em direção às camadas internas, garantindo que regras de negócio fundamentais permaneçam completamente independentes de detalhes de implementação tecnológicos como frameworks específicos, tecnologias de banco de dados particulares, interfaces de usuário concretas, ou serviços externos específicos.

O núcleo conceitual da arquitetura limpa constitui na definição clara e precisa de entidades de negócio que encapsulam regras fundamentais e invariantes do domínio empresarial, casos de uso que orquestram fluxos específicos de aplicação e implementaram regras de negócio complexas, adaptadores que facilitam comunicação controlada com sistemas externos e infraestrutura técnica, e frameworks/drivers que fornecem detalhes de implementação específicos sem contaminar lógica de negócio central. Esta estrutura hierárquica e bem definida promove testabilidade abrangente através do isolamento completo de dependências externas, facilita manutenção eficaz através da modularização clara de responsabilidades, e proporciona flexibilidade excepcional para mudanças tecnológicas sem impacto nas regras de negócio centrais que constituem o valor real e duradouro do sistema.

A separação entre diferentes tipos de lógica constitui aspecto fundamental da arquitetura limpa. Entidades representam conceitos centrais do negócio com regras que se aplicam independentemente de aplicação específica, casos de uso implementam regras específicas da aplicação orquestrando entidades para alcançar objetivos particulares, adaptadores fornecem

interfaces para comunicação com mundo externo mantendo isolamento das camadas internas, e frameworks/drivers implementam detalhes específicos de tecnologias escolhidas. Esta separação permite evolução independente de cada camada, facilita testes isolados de cada nível de responsabilidade, e garante que mudanças em detalhes de implementação não propagaram para regras de negócio fundamentais.

4.3.2 Aplicação da Arquitetura Limpa em Sistemas de Gestão

A aplicação dos princípios de arquitetura limpa em sistemas de gestão de eventos proporciona benefícios específicos, mensuráveis e estratégicos para construir uma solução robusta, evolutiva e mantível. O isolamento rigoroso das regras de negócio relacionadas a cálculos financeiros complexos, validações de dados de eventos com regras específicas de cada tipo de evento, lógicas sofisticadas de notificação baseadas em preferências de usuários e contextos específicos, e algoritmos de otimização de recursos e *scheduling* permite implementação de testes unitários abrangentes e confiáveis sem dependência de banco de dados, interfaces externas, ou serviços de terceiros.

Esta separação facilita significativamente evoluções futuras do sistema, permitindo substituição de tecnologias infraestruturais, mudanças em interfaces de usuário, integrações com novos serviços externos, ou migrações de banco de dados sem necessidade de reescrita das regras centrais de gestão de eventos que representam o conhecimento e valor acumulado da organização. A arquitetura limpa também facilita implementação de funcionalidades como *audit trails* para compliance regulatório, versionamento de regras de negócio para suporte a mudanças organizacionais, e personalização de comportamentos para diferentes tipos de clientes ou contextos operacionais.

A modularização proporcionada pela arquitetura limpa permite também desenvolvimento paralelo de diferentes aspectos do sistema por equipes especializadas, redução de conflitos de código em projetos colaborativos, e reutilização de componentes de negócio em diferentes contextos ou aplicações. Para a gestão de eventos, esta modularização traduz-se em redução significativa de duplicação de código, consistência de comportamentos entre aplicações, e facilidade de manutenção centralizada de regras de negócio compartilhadas.

4.3.3 Padrão CQRS - Conceitos Fundamentais

O padrão CQRS (Command Query Responsibility Segregation) constitui abordagem arquitetural avançada que segrega operações de leitura (queries) e escrita (commands) em modelos distintos e especialmente otimizados, permitindo otimização específica de cada tipo de operação conforme suas características particulares de performance, consistência, escalabilidade e requisitos de dados. Esta separação fundamental possibilita que comandos foquem exclusivamente em alterações controladas de estado do sistema, utilizando validações rigorosas, processamento transacional robusto, e garantias de consistência forte, enquanto consultas especializem-se na recuperação eficiente de dados para apresentação, empregando estruturas de dados desnormalizadas otimizadas, estratégias agressivas de cache, e processamento otimizado para diferentes padrões de acesso.

A implementação eficaz de CQRS requere reconhecimento e exploração das diferenças fundamentais entre necessidades de escrita e leitura em sistemas complexos. Operações de escrita tipicamente requerem validações complexas que consideraram regras de negócio sofisticadas, contexto de segurança e autorização, consistência transacional rigorosa, e capacidade de auditoria para compliance. Operações de leitura, por outro lado, beneficiam-se de estruturas de dados otimizadas para padrões específicos de acesso, cache agressivo para redução de latência, eventual consistência para melhor performance, e capacidade de escalonamento horizontal independente das operações de escrita.

O padrão CQRS facilita também implementação de arquiteturas direcionadas a eventos onde mudanças de estado foram representadas como eventos imutáveis que podem ser processados assincronamente por diferentes subsistemas especializados. Esta abordagem proporciona benefícios como auditoria completa e automática de todas as mudanças, capacidade de reconstrução de estado a partir de eventos históricos, integração mais fácil com sistemas externos através de *event streaming*, e possibilidade de implementação de novas funcionalidades como *undo/redo*, *time travel debugging*, e análises em tempo real sobre comportamento do sistema.

4.3.4 CQRS em Sistemas de Gestão de Eventos

A implementação de CQRS em sistemas de gestão de eventos proporciona vantagens substanciais e específicas para cenários onde operações de leitura e escrita apresentam características e requisitos fundamentalmente diferentes. Comandos para criação de novos

eventos, atualização de informações de eventos existentes, cadastro e modificação de fornecedores, registro de transações financeiras complexas, ou atualização de status de atividades utilizam modelos de dados normalizados que garantem integridade referencial, implementam validações rigorosas que consideraram regras de negócio específicas de cada tipo de evento, e empregam processamento transacional que assegura consistência mesmo em cenários de falha ou concorrência.

Consultas para geração de relatórios financeiros, exibição de dashboards executivos com agregações em tempo real, análises de performance histórica com comparações entre períodos, ou visualizações de dados empregam estruturas de dados especificamente otimizadas para recuperação rápida.

4.3.5 Padrões de Implementação e Boas Práticas

A implementação de sistemas complexos baseados em arquiteturas modernas requer adoção sistemática de padrões estabelecidos e boas práticas que garantissem qualidade, manutenibilidade, testabilidade e escalabilidade do código produzido ao longo do tempo. O padrão *Repository* emerge como abstração fundamental para acesso a dados, criando camada de isolamento entre lógica de negócio e detalhes específicos de persistência, facilitando testes unitários através de implementações *mock* e *in-memory* para ambientes de desenvolvimento e teste, e permitindo substituição de tecnologias de banco de dados sem impacto significativo na lógica de aplicação.

O padrão *Repository* encapsula consultas específicas e operações de dados, fornecendo interface limpa e orientada a domínio para camadas superiores, implementando *caching* transparente quando apropriado, e abstraindo complexidades como gerenciamento de conexão, tratamento de transação, e recuperação de erros. Para sistemas de gestão de eventos, repositórios especializados como EventoRepository, EstoqueRepository, PedidoRepository fornecerem interfaces específicas de domínio que escondem detalhes de SQL e mapeamento objeto-relacional, facilitando evolução independente de estruturas de dados e lógica de negócio.

O padrão *Mediator* demonstra eficácia particular para implementação de arquiteturas CQRS e Arquitetura Limpa, centralizando processamento de comandos e consultas através de interface unificada que promove baixo acoplamento entre controladores de API e *handlers* de negócio específicos. Esta abordagem facilita implementação de *cross-cutting concerns* como *logging* detalhado com Ids correlacionados, validação automática de comandos baseada em atributos

ou *fluent validation*, tratamento padronizado de erros com tradução para códigos HTTP apropriados, auditoria automática de operações sensíveis, e *caching* inteligente de resultados de consultas.

4.3.6 Injeção de Dependências e Princípios SOLID

A injeção de dependências constitui mecanismo fundamental para promoção de testabilidade, flexibilidade arquitetural, e facilidade de manutenção em sistemas orientados a objetos complexos. A configuração adequada do container de DI permite substituição transparente de implementações para diferentes ambientes operacionais (desenvolvimento com banco de dados *in-memory*, teste com containers efêmeros, produção com banco de dados gerenciados), facilita implementação de padrões como *decorator* para funcionalidades transversais (*logging, caching, retry policies*), e simplifica criação e gestão de objetos complexos com múltiplas dependências transitivas.

Esta abordagem promove aderência aos princípios SOLID, particularmente *Single Responsibility Principle* através de classes focadas em responsabilidades específicas, *Open/Closed Principle* através de extensibilidade via interfaces e composição, *Liskov Substitution Princip*le através de abstrações bem definidas, *Interface Segregation Principle* através de interfaces coesas e específicas, e *Dependency Inversion Principle* através de dependências em abstrações ao invés de implementações concretas.

A configuração do DI container inclui considerações como gerenciamento de ciclos de vida (*singleton* para serviços sem estado, *scoped* para contextos de requisição, *transient* para objetos leves), detecção de dependência circular e resolução, registro condicional baseado em ambiente ou *feature flags*, e integração com frameworks de teste para setup automatizado de *mocks* e *test doubles*. Para sistemas de gestão de eventos, estas funcionalidades traduzem-se em código mais testável, modular, e resiliente a mudanças de requisitos.

4.3.7 Estratégias de Teste e Validação

A validação de sistemas baseados em arquitetura limpa e CQRS demanda estratégias específicas e estruturadas de teste que contemplam diferentes camadas e cenários da aplicação. Para garantir qualidade e confiabilidade do código, foi adotado o framework XUnit como base para implementação dos testes unitários, proporcionando estrutura robusta e flexível para validação isolada das regras de negócio.

A implementação dos testes unitários segue rigorosamente o padrão AAA (*Arrange-Act-Assert*), organizando cada teste em três etapas distintas: preparação do ambiente e dados necessários (*Arrange*), execução da funcionalidade sob teste (*Act*), e verificação dos resultados esperados (*Assert*). Esta abordagem garante clareza na estrutura dos testes, facilitando manutenção e compreensão do código de teste por outros desenvolvedores.

Para assegurar isolamento adequado entre testes, são implementadas práticas de setup e teardown, utilizando features do XUnit para configuração de dados de teste consistentes e limpeza automática do ambiente após cada execução. A utilização de Theory e InlineData permite execução de testes parametrizados, validando múltiplos cenários com diferentes entradas de dados de forma eficiente e organizada.

Os testes unitários cobrem extensivamente as regras de negócio do sistema de gestão de eventos, incluindo cálculos de custos e orçamentos, validações de datas e detecção de conflitos de agenda, lógicas de notificação baseadas em preferências do usuário, e algoritmos de otimização de recursos. Cada componente é testado de forma independente, garantindo que falhas fossem identificadas precisamente e que alterações futuras não comprometessem funcionalidades já validadas.

A implementação de testes de integração verifica comunicação adequada entre componentes do sistema, validando persistência correta de dados com verificação de integridade referencial, processamento adequado de comandos complexos envolvendo múltiplas entidades, execução eficiente de consultas com operações complexas, e integração com serviços externos. Estes testes utilizam bancos de dados dedicados para ambiente de teste, garantindo isolamento completo e repetibilidade dos cenários validados.

5. DESCRIÇÃO DO PROJETO

Este capítulo apresenta de forma detalhada a concepção e estrutura do sistema Eventario, oferecendo uma visão abrangente de suas funcionalidades, arquitetura e implementação. A apresentação do projeto está organizada de forma a proporcionar compreensão progressiva, iniciando pelas funcionalidades do sistema e suas interfaces com o usuário, progredindo para os aspectos arquiteturais através de diagramas técnicos.

5.1 Interação com o Usuário

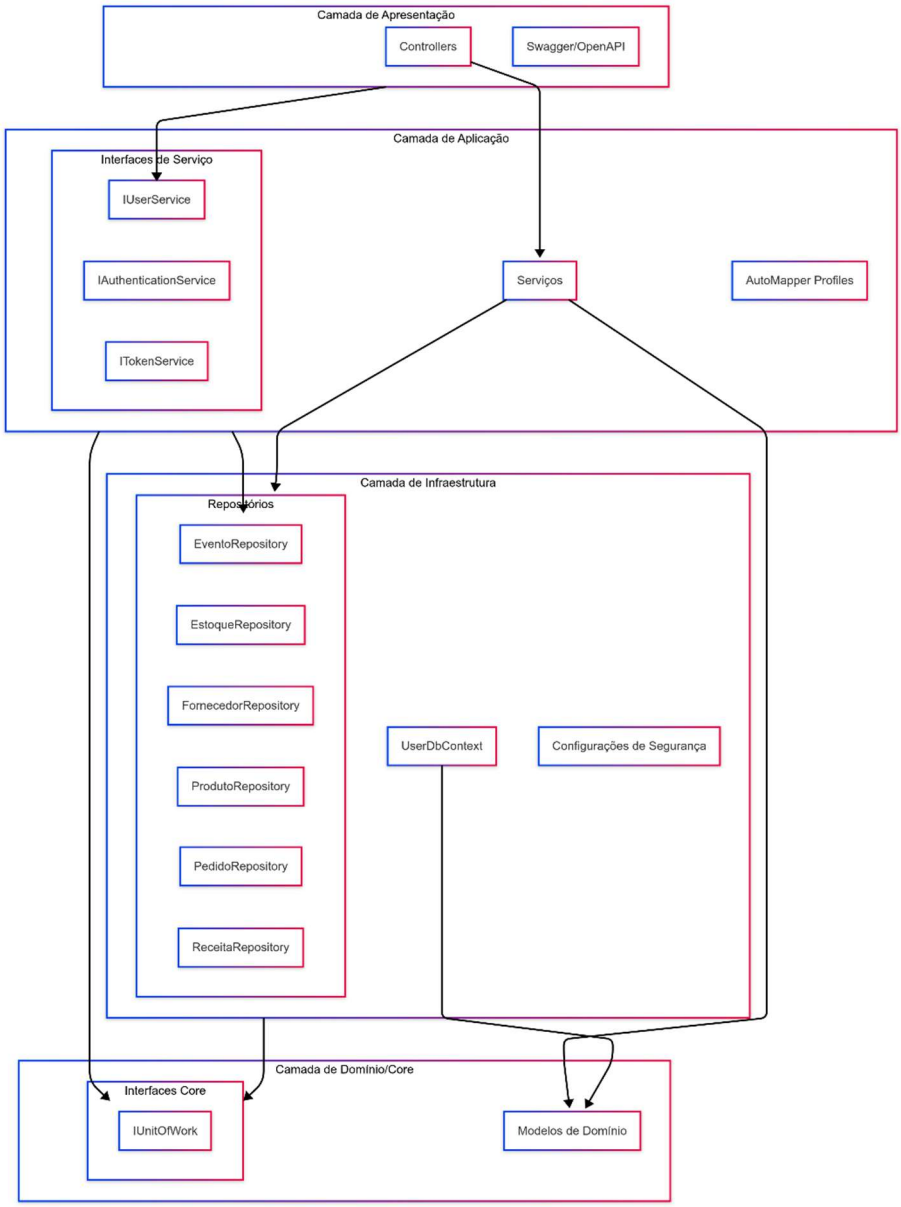
Os usuários do **Eventario** poderão acessar o sistema via navegador web, utilizando qualquer dispositivo com conexão à internet. A interface será intuitiva e responsiva, permitindo que os organizadores realizem todas as operações de forma prática, desde o cadastro inicial do evento até o monitoramento do seu andamento.

A interação ocorrerá por meio de uma interface gráfica (GUI) organizada em módulos, onde cada funcionalidade estará representada de forma clara.

5.2 Diagrama de Blocos do Sistema

Para compreender a arquitetura geral do sistema Eventário, foi desenvolvido um diagrama de blocos que ilustra a organização das camadas e a comunicação entre os componentes principais. Este diagrama apresenta a estrutura em quatro camadas principais: a camada de apresentação (frontend), a camada de aplicação (serviços e APIs), a camada de infraestrutura (repositórios e banco de dados) e a camada de Domínio onde ficam as regras de negócio. A representação visual facilita o entendimento de como os diferentes módulos do sistema interagem entre si e como o fluxo de dados é processado desde a interface do usuário até o armazenamento persistente.

Figura 1 - Diagrama de Blocos



Fonte: Próprio Autor.

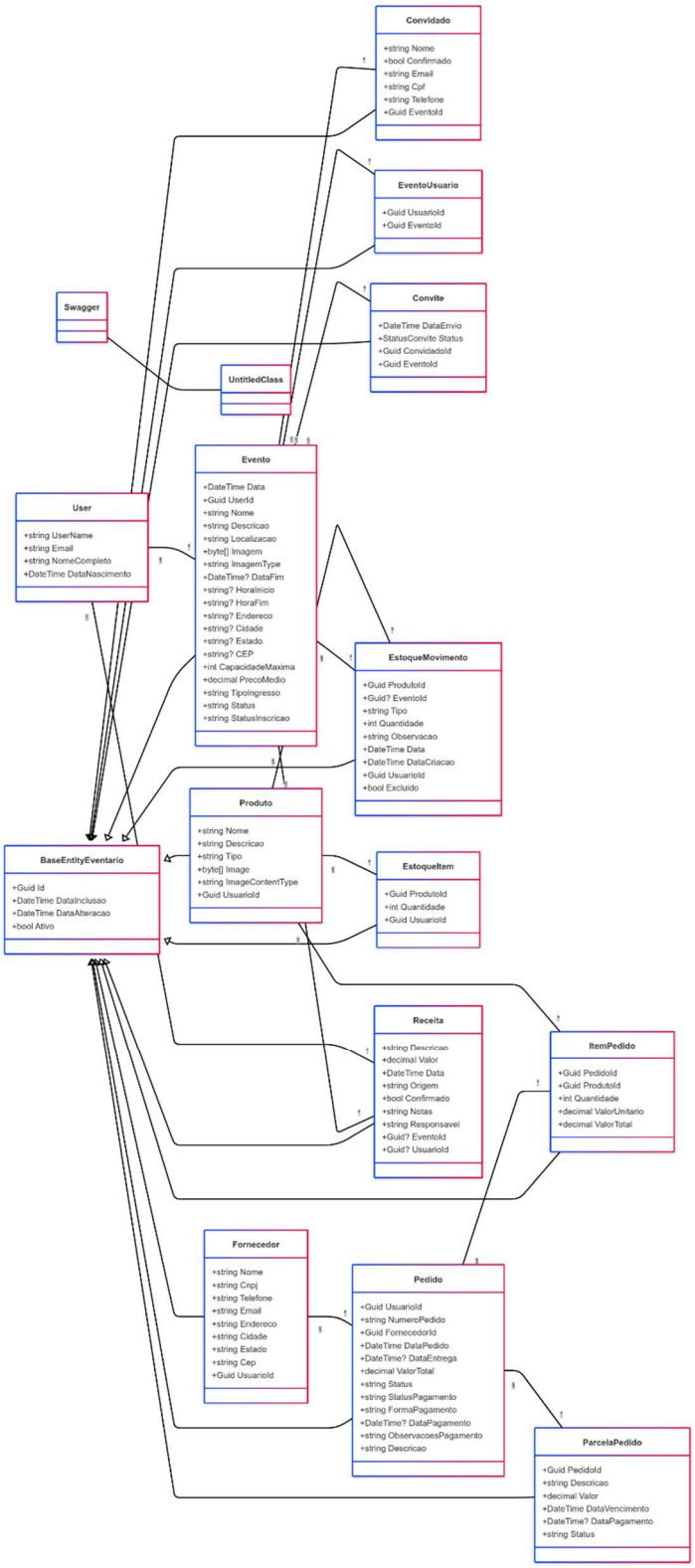
5.3 Comunicação Entre Módulos

A comunicação será feita a partir de uma API RESTful desenvolvida em .NET. Essa API será responsável por facilitar a comunicação entre o frontend (interface do usuário) e o backend (lógica de negócios e banco de dados).

5.4 Diagrama de Classes

A modelagem orientada a objetos do sistema é apresentada através de um diagrama de classes mostrado na Figura 2abrangente que demonstra as entidades principais do domínio e seus relacionamentos. Este diagrama ilustra como as classes Evento, Fornecedor, Produto, Pedido, Receita e demais entidades se relacionam através de associações, agregações e composições. A visualização permite compreender a estrutura de dados subjacente ao sistema e como as regras de negócio são implementadas através dos relacionamentos entre as classes.

Figura 2 - Diagrama de Classes

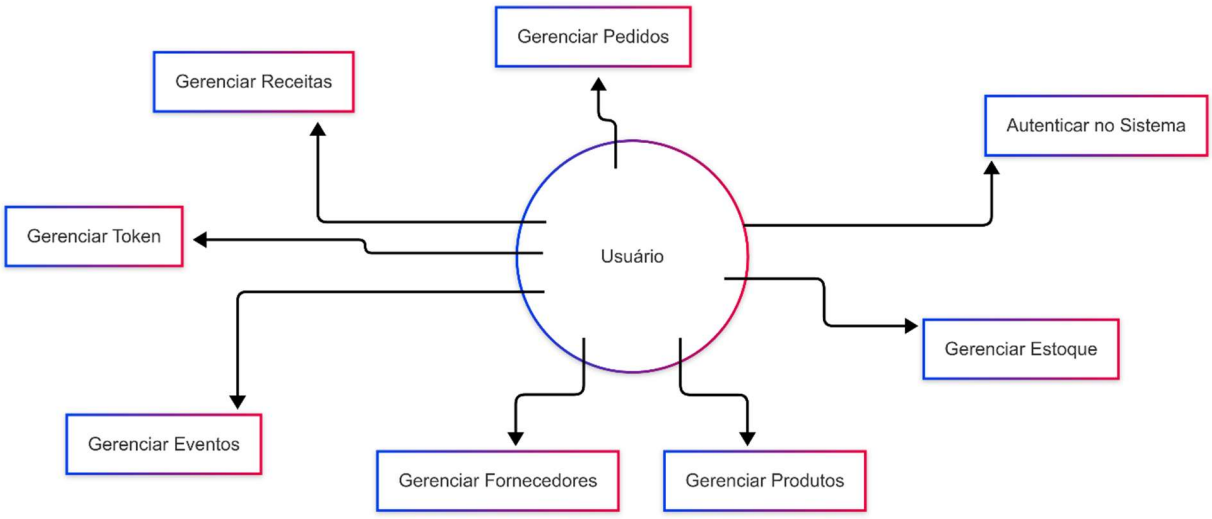


Fonte: Próprio Autor.

5.5 Diagrama de Casos de Uso

Para identificar as funcionalidades principais do sistema e os atores envolvidos, foi elaborado um diagrama de casos de uso que mapeia as interações entre os usuários e o sistema. Exibido na Figura 3, o diagrama apresenta o ator principal (Usuário) e suas possibilidades de interação com os diferentes módulos: gerenciar eventos, pedidos, receitas, produtos, fornecedores, estoque e relatórios. Esta representação oferece uma visão clara do escopo funcional do sistema e serve como base para o desenvolvimento das interfaces e validação dos requisitos.

Figura 3 - Diagrama de Casos de Uso



Fonte: Próprio Autor.

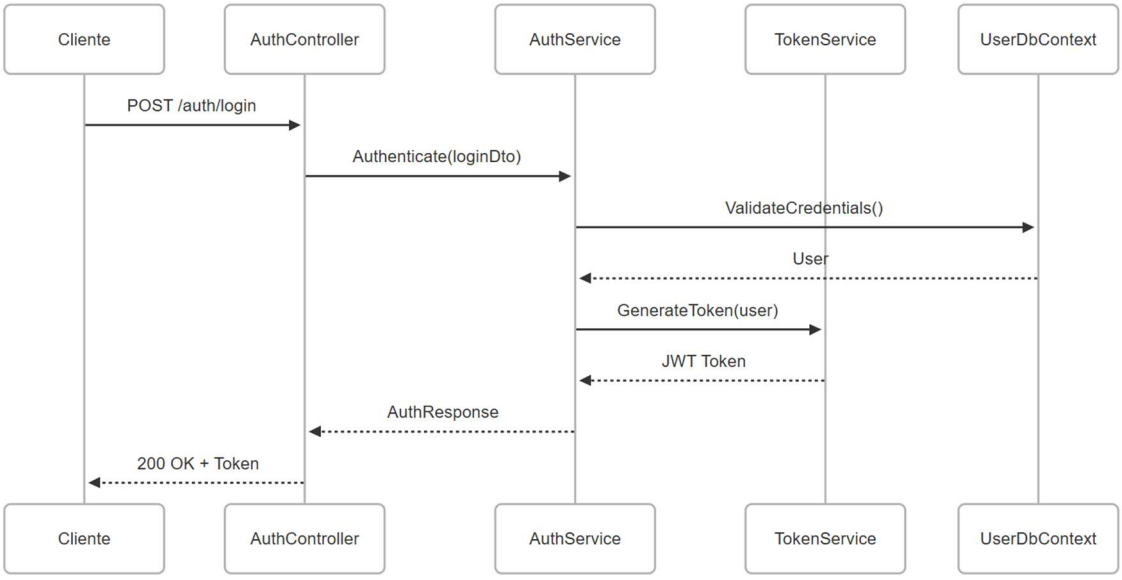
5.6 Diagrama de Sequência

A dinâmica de funcionamento do sistema é detalhada através de diagramas de sequência que mostram a interação temporal entre os diferentes componentes durante a execução de operações específicas. Estes diagramas são fundamentais para compreender o fluxo de mensagens e chamadas entre controladores, serviços, repositórios e banco de dados.

5.6.1 Fluxo de Autenticação

O processo de autenticação no sistema segue um padrão baseado em tokens JWT. O diagrama apresenta a sequência de interações desde a solicitação de login pelo cliente até a geração e retorno do token de acesso. Exibido na Figura 4, este fluxo garante a segurança do sistema através da validação de credenciais e geração de tokens com tempo de expiração controlado.

Figura 4 - Fluxo de Autenticação



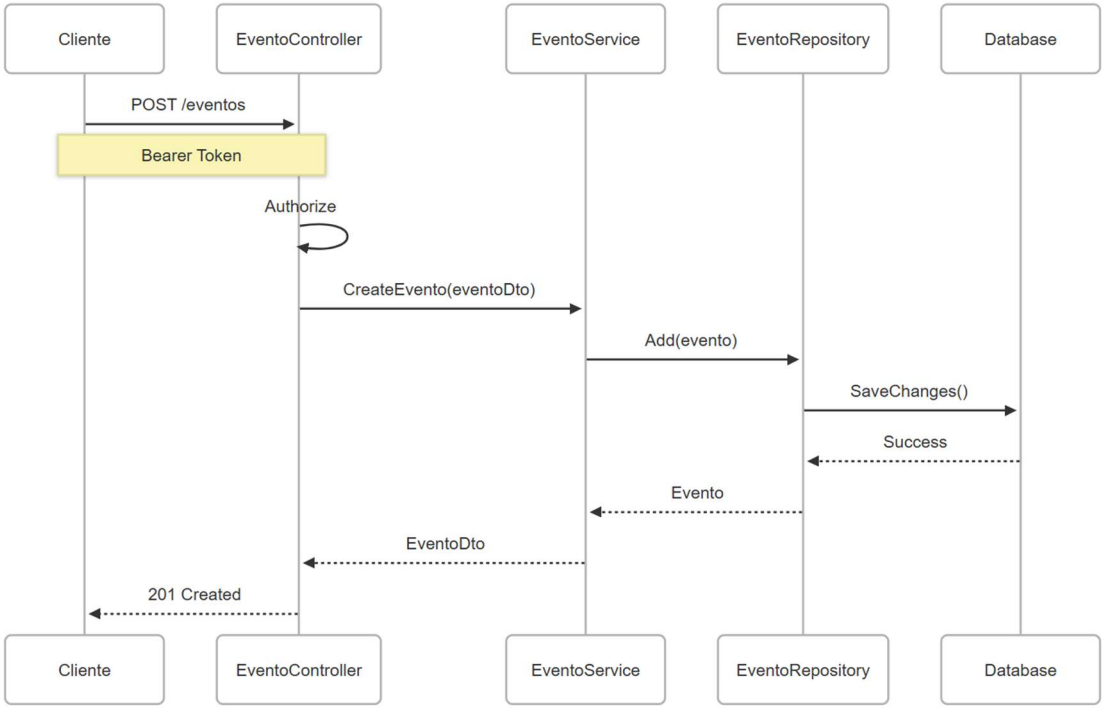
Fonte: Próprio Autor.

5.8.2 Fluxo de Gestão de Eventos

A criação e gerenciamento de eventos constitui uma das funcionalidades centrais do sistema. O diagrama de sequência ilustra como as informações do evento são processadas desde a interface

do usuário, passando pela validação no controlador, processamento na camada de serviço, até a persistência no banco de dados através dos repositórios.

Figura 5 - Fluxo de Gestão de Eventos.

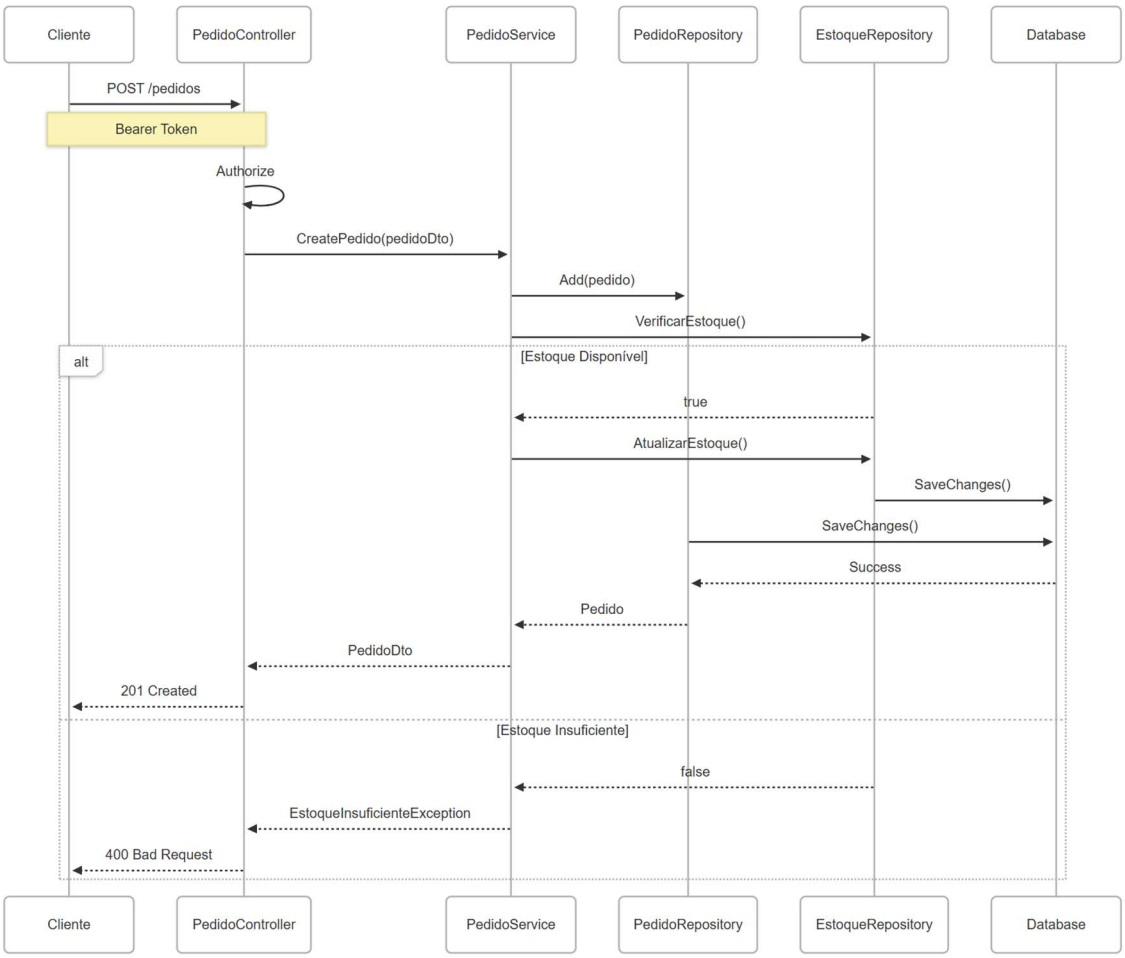


Fonte: Próprio Autor.

5.8.3 Fluxo de Gestão de Pedidos

O processo de criação e acompanhamento de pedidos envolve múltiplas validações e atualizações de estoque. O diagrama apresenta a sequência completa desde a criação do pedido, atualização de disponibilidade no estoque, até a atualização dos registros e notificação de status.

Figura 6 - Fluxo de Gestão de Pedidos

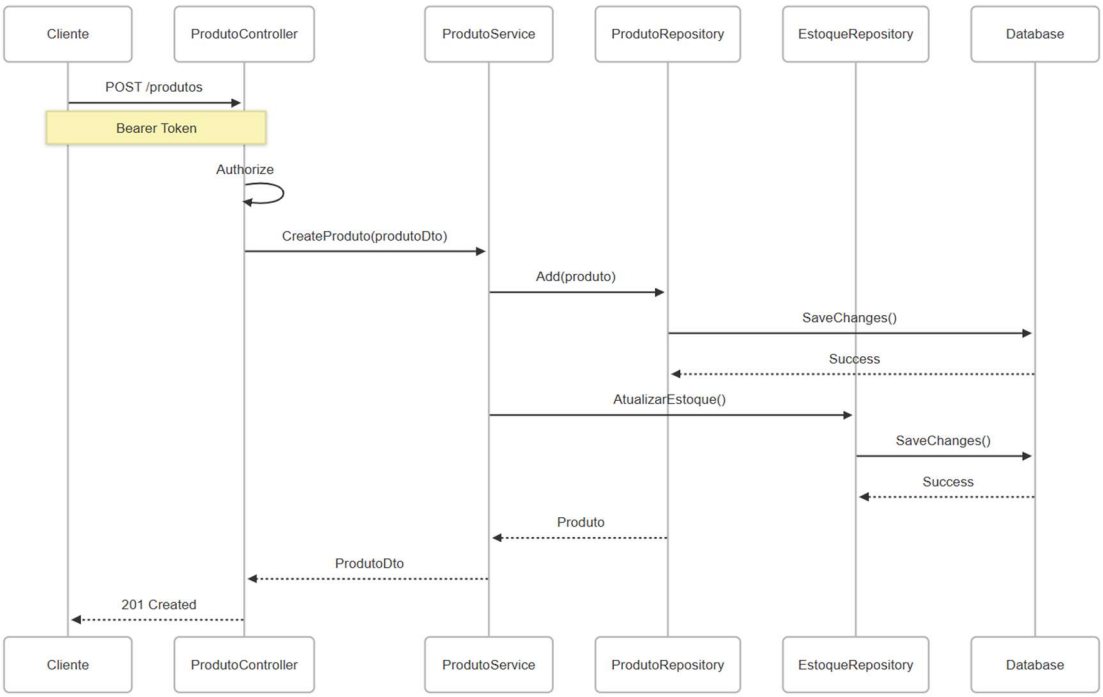


Fonte: Próprio Autor.

5.8.4 Fluxo de Gestão de Produtos e Estoque

O controle de produtos e movimentações de estoque requer sincronia entre diferentes módulos do sistema. O diagrama mostra como as operações de entrada e saída de produtos são processadas, incluindo as validações necessárias e atualizações automáticas dos níveis de estoque.

Figura 7 - Fluxo de Gestão de Produtos e Estoque

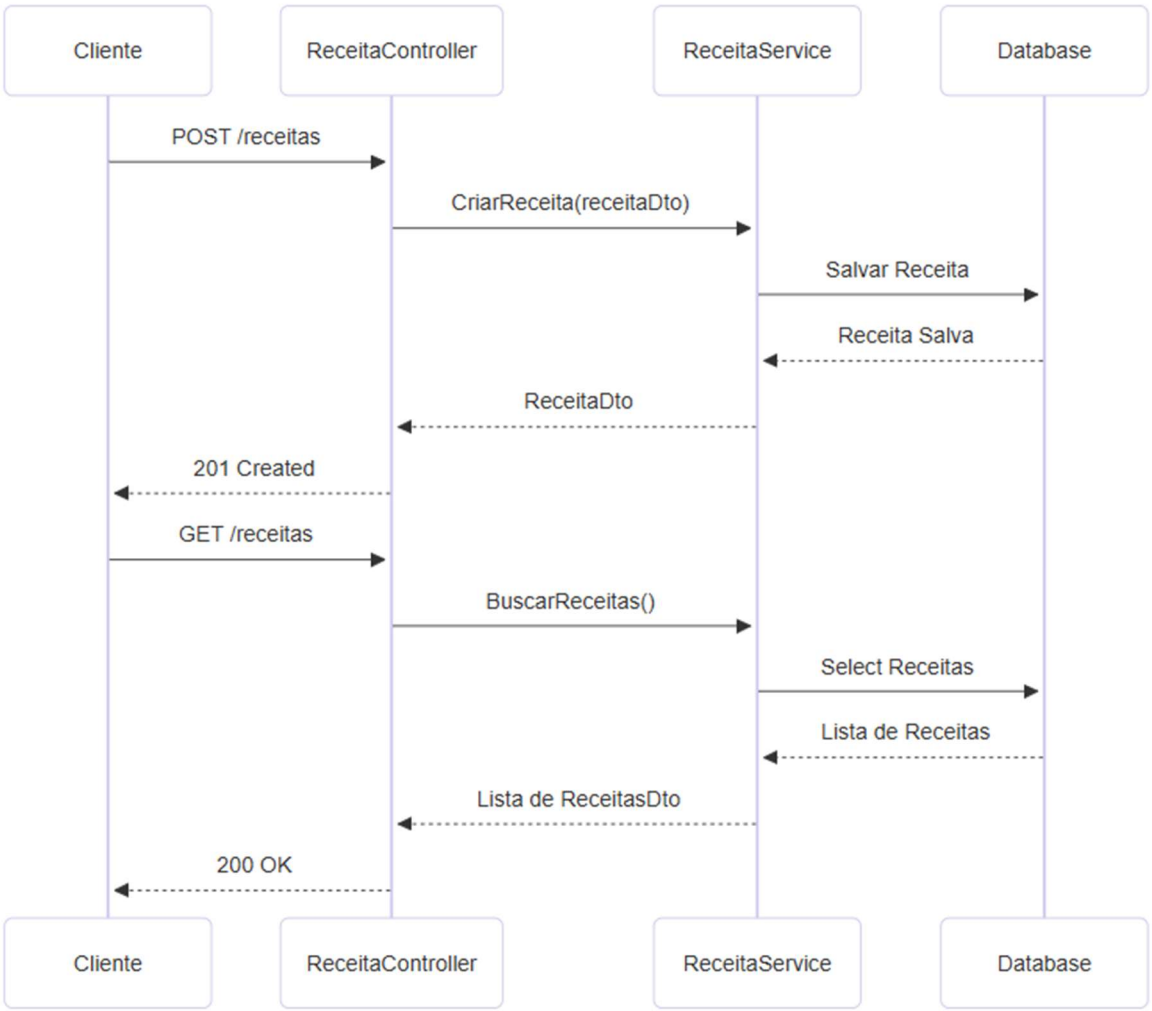


Fonte: Próprio Autor.

5.8.5 Fluxo de Gestão de Receitas

O módulo financeiro do sistema processa diferentes tipos de receitas associadas aos eventos. O diagrama de sequência demonstra como as receitas são cadastradas, validadas e integradas aos relatórios financeiros, mantendo a rastreabilidade e integridade dos dados contábeis.

Figura 8 - Fluxo de Gestão de Receitas



Fonte: Próprio Autor.

6. TECNOLOGIAS PESQUISADAS E ESCOLHIDAS

6.1 Tecnologias Pesquisadas

- Linguagens de Programação: JavaScript (para o front-end) e C# (para o back-end).
- Frameworks: Vue (para a interface do usuário) e .NET (para a lógica de negócios e APIs).
- Banco de Dados: SQL Server (para armazenamento de dados estruturados).

6.2 Tecnologias Escolhidas

6.2.1 Front-End: Vue.js

O Vue.js foi escolhido pela sua eficiência e flexibilidade para criar interfaces de usuário dinâmicas. O framework oferece sintaxe intuitiva que facilita o desenvolvimento de dashboards interativos e formulários complexos presentes no Eventario. Sua arquitetura baseada em componentes permite reutilização de código e manutenção simplificada, especialmente importante para um sistema com múltiplos módulos. O Vue Router facilita a navegação entre seções, enquanto o sistema de reatividade garante atualizações automáticas das interfaces quando dados são modificados.

6.2.2 Back-End: .NET

O .NET foi selecionado devido à sua robustez e integração otimizada com o SQL Server. A plataforma oferece performance superior para APIs RESTful, sistema de tipos forte que reduz erros de desenvolvimento, e Entity Framework Core para mapeamento objeto-relacional eficiente. O sistema de injeção de dependências nativo facilita a implementação da arquitetura limpa adotada no projeto. Adicionalmente, o ecossistema .NET fornece ferramentas integradas para logging, validação, autenticação e testes, acelerando o desenvolvimento e garantindo qualidade do código.

6.2.3 Banco de Dados: SQL Server

O SQL Server foi utilizado pela sua confiabilidade e capacidade de gerenciar grandes volumes de dados. A integração nativa com .NET elimina complexidades de configuração e oferece performance otimizada através de connection pooling automático. O sistema oferece recursos empresariais como backup automático, índices inteligentes e análise de queries que são essenciais para um sistema de gestão de eventos. O SQL Server Management Studio facilita administração e debugging, enquanto a compatibilidade com Azure permite escalabilidade futura em ambiente cloud.

6.3 Integração do Stack Tecnológico

A combinação Vue.js + .NET + SQL Server forma um stack coeso onde cada tecnologia complementa as demais. O frontend Vue.js consome APIs REST do backend .NET através de requisições JSON padronizadas. O backend processa regras de negócio e comunica com SQL Server via Entity Framework, mantendo separação clara entre camadas. Esta arquitetura facilita manutenção, testes e possíveis expansões futuras do sistema.

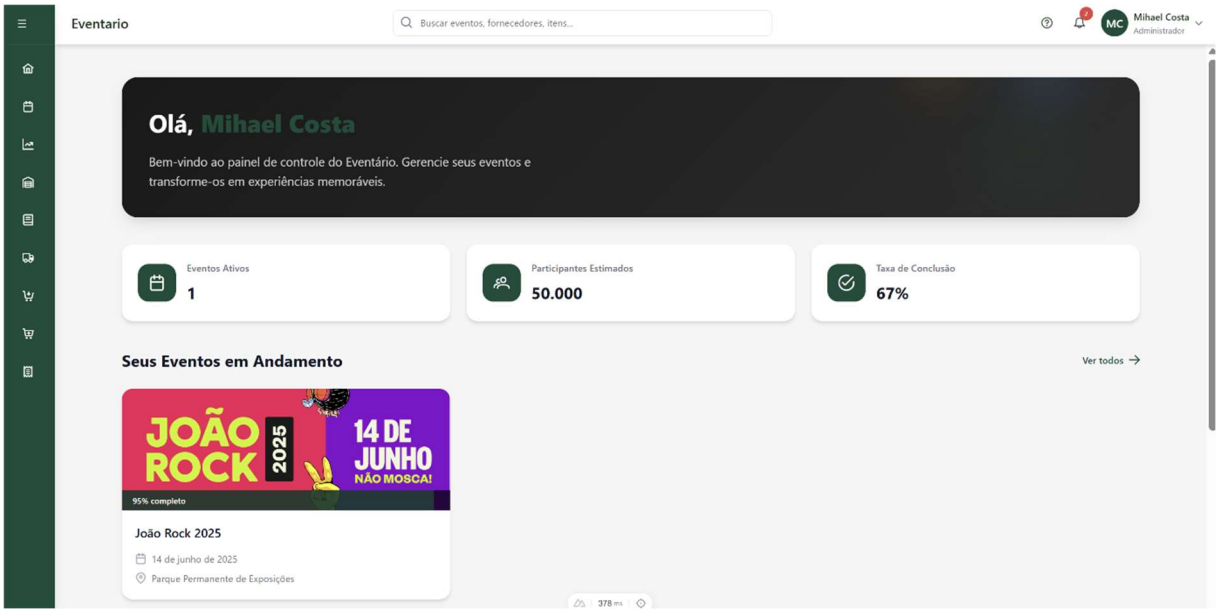
7 DOCUMENTAÇÃO

As interfaces do sistema foram desenvolvidas priorizando usabilidade e experiência do usuário. As capturas de tela apresentadas documentam o resultado de cada módulo, demonstrando como as funcionalidades são apresentadas ao usuário e como a navegação entre os diferentes módulos é organizada.

7.1 Tela Inicial

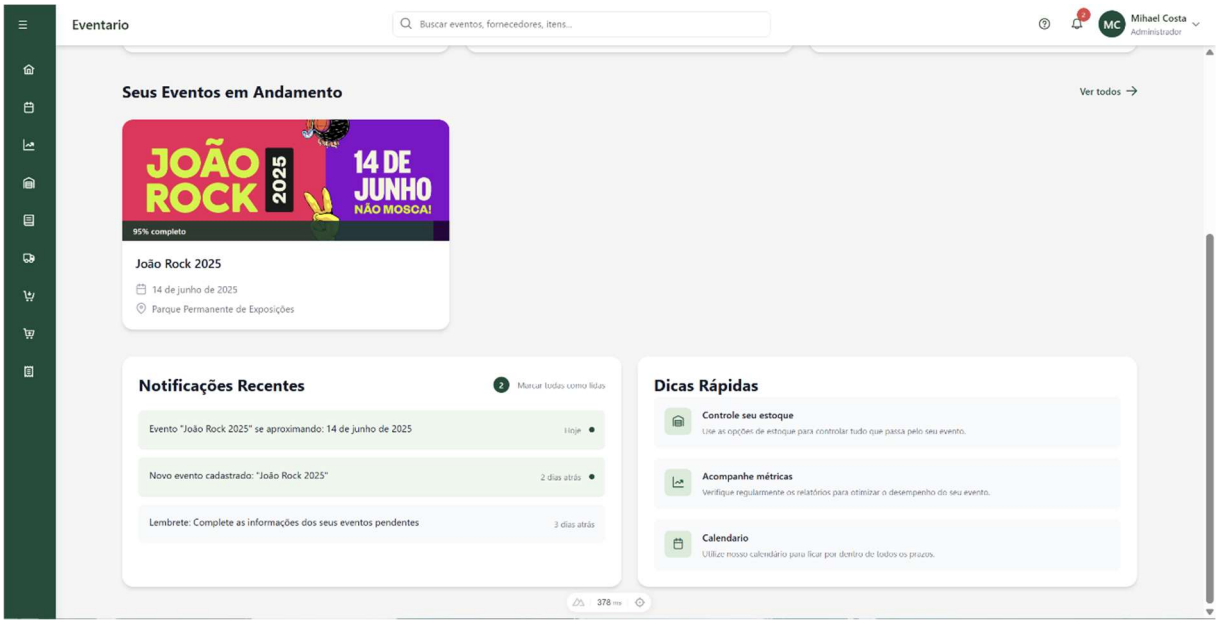
A página inicial do sistema apresenta uma visão geral das informações mais relevantes para o usuário, incluindo estatísticas resumidas, eventos em andamento e acesso rápido às principais funcionalidades. O design responsivo garante uma experiência consistente em diferentes dispositivos.

Figura 9 - Tela Inicial 1



Fonte: Próprio Autor.

Figura 10 - Tela Inicial 2

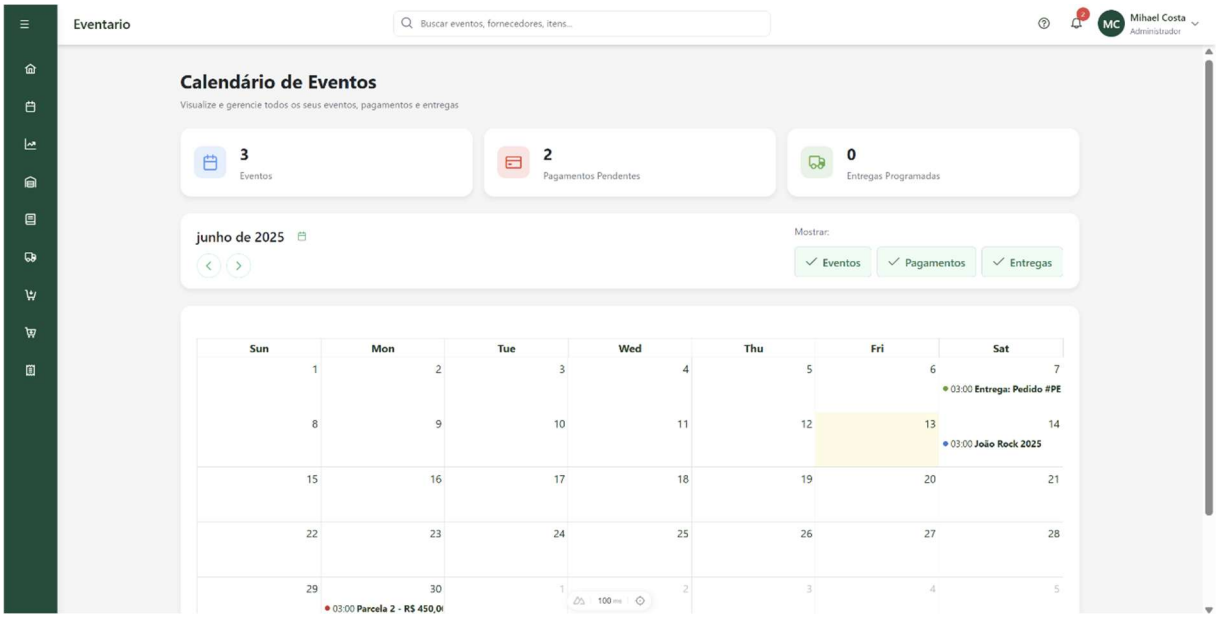


Fonte: Próprio Autor.

7.2 Calendário

O módulo de calendário centraliza a visualização temporal de eventos, pagamentos e entregas. As interfaces demonstram como o usuário pode navegar entre diferentes períodos, filtrar tipos de informação e acessar detalhes específicos através de modais interativos.

Figura 11 - Calendário 1



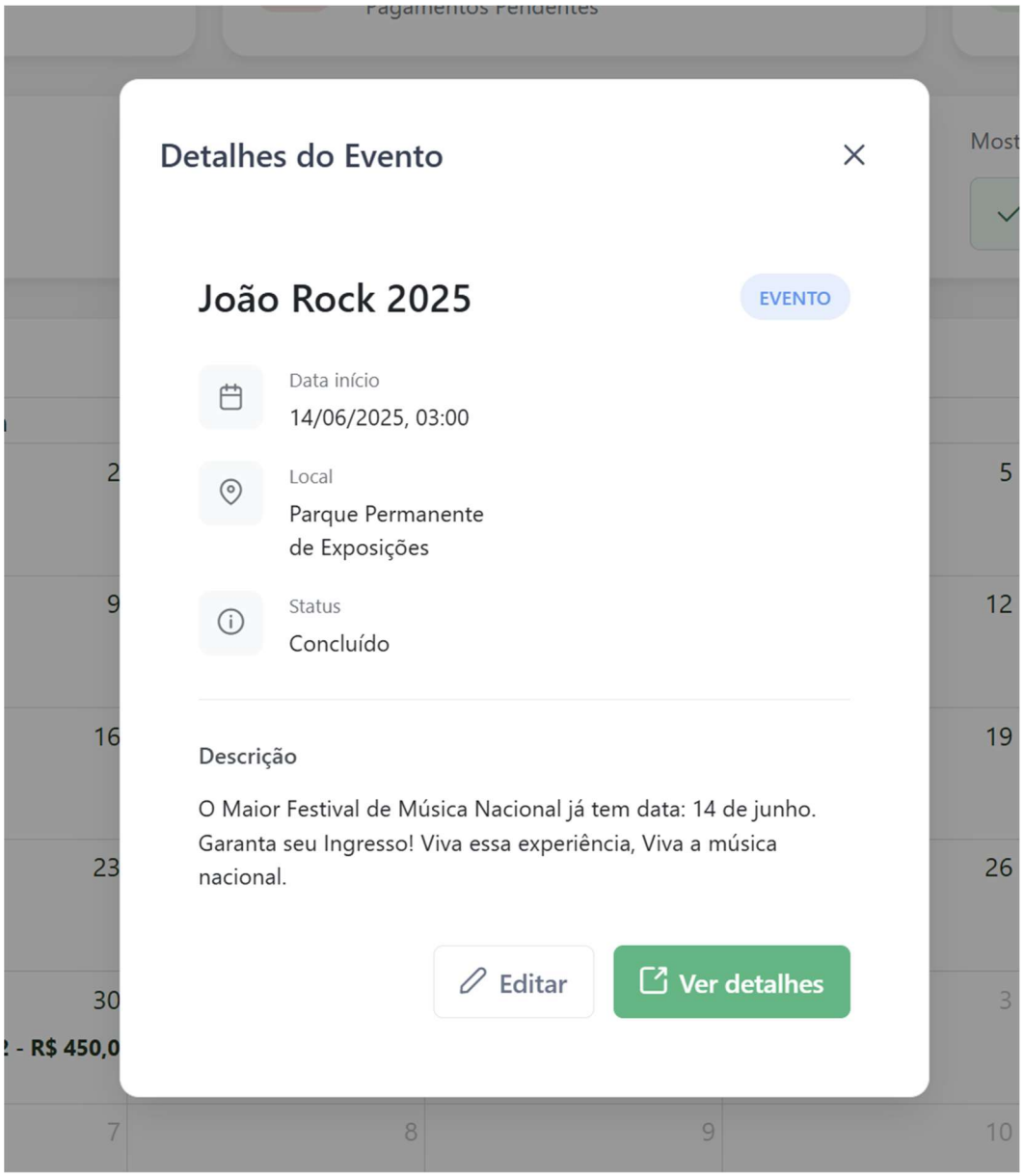
Fonte: Próprio Autor.

Para facilitar o acesso a informações detalhadas sem sair da visualização do calendário, foram implementados modais que apresentam dados específicos ao clicar em elementos do calendário. Estes modais mantêm o contexto da navegação enquanto fornecem informações completas sobre entregas, eventos e pagamentos.

Figura 12 - Calendário: Modal Entrega

Fonte: Próprio Autor.

Figura 13 - Calendário: Modal Evento



Fonte: Próprio Autor.

Figura 14 - Calendário: Modal Pagamento

Fonte: Próprio Autor.

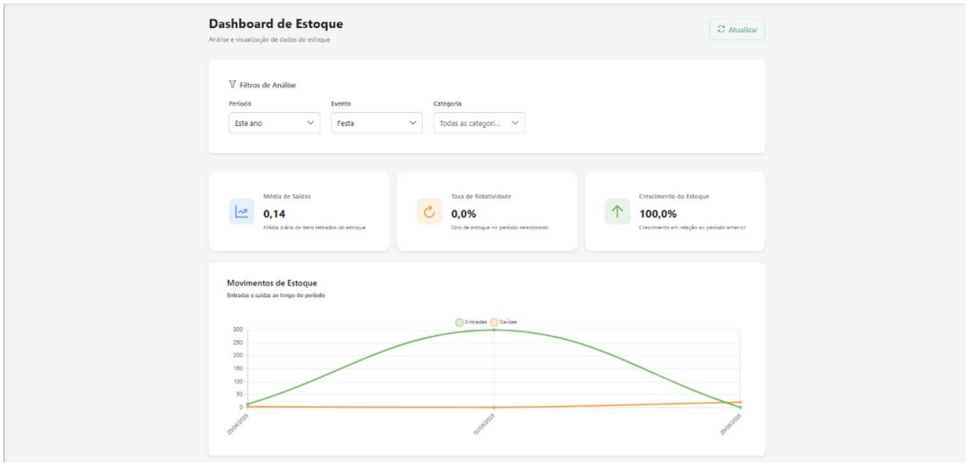
7.3 Dashboards

Os dashboards foram desenvolvidos para fornecer insights visuais sobre diferentes aspectos da gestão de eventos. Através de gráficos e indicadores, os usuários podem monitorar performance, identificar tendências e tomar decisões baseadas em dados.

7.3.1 Dashboard de Estoque

O dashboard de estoque apresenta informações consolidadas sobre produtos disponíveis, movimentações e necessidades de reposição. Os gráficos facilitam a identificação de produtos com alta rotatividade e alertas para níveis baixos de estoque.

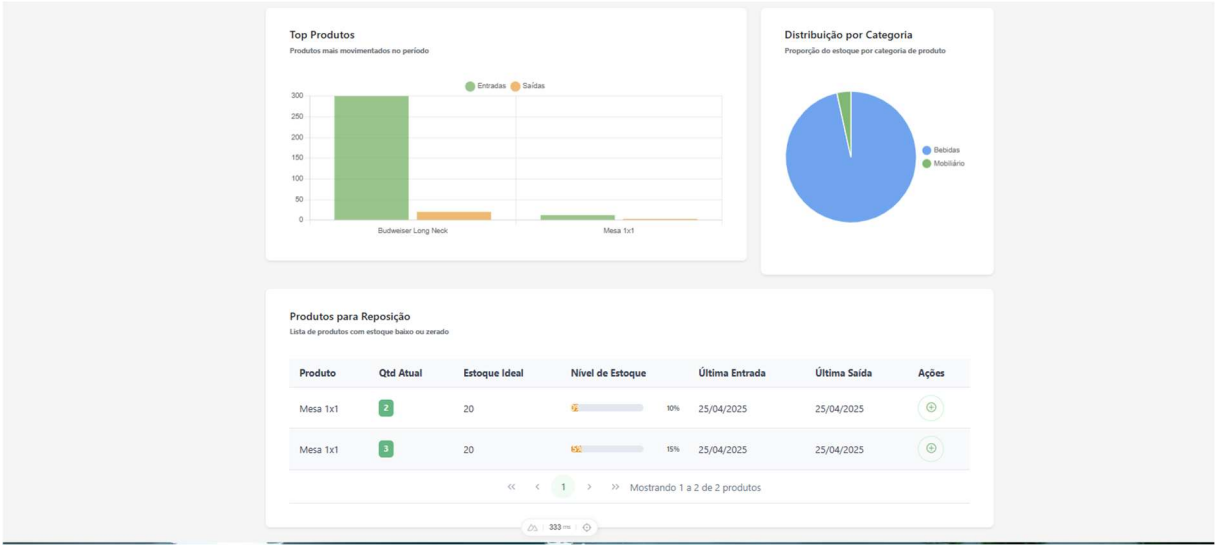
Figura 15 - Dashboard de Estoque 1



Fonte: Próprio Autor.

A segunda visualização do dashboard de estoque detalha a análise por categorias e produtos específicos. O gráfico "Top Produtos" em formato de barras horizontais apresenta os itens com maior movimentação, destacando produtos como "Budweiser Long Neck" com volumes significativos. O gráfico circular "Distribuição por Categoria" mostra a proporção de produtos por tipo (Bebidas, Mobiliário etc.), facilitando a compreensão da composição do estoque. A tabela "Produtos para Reposição" lista itens específicos com informações detalhadas sobre quantidade atual, estoque ideal e datas de última atualização, oferecendo uma ferramenta prática para gestão de compras.

Figura 16 - Dashboard de Estoque 2



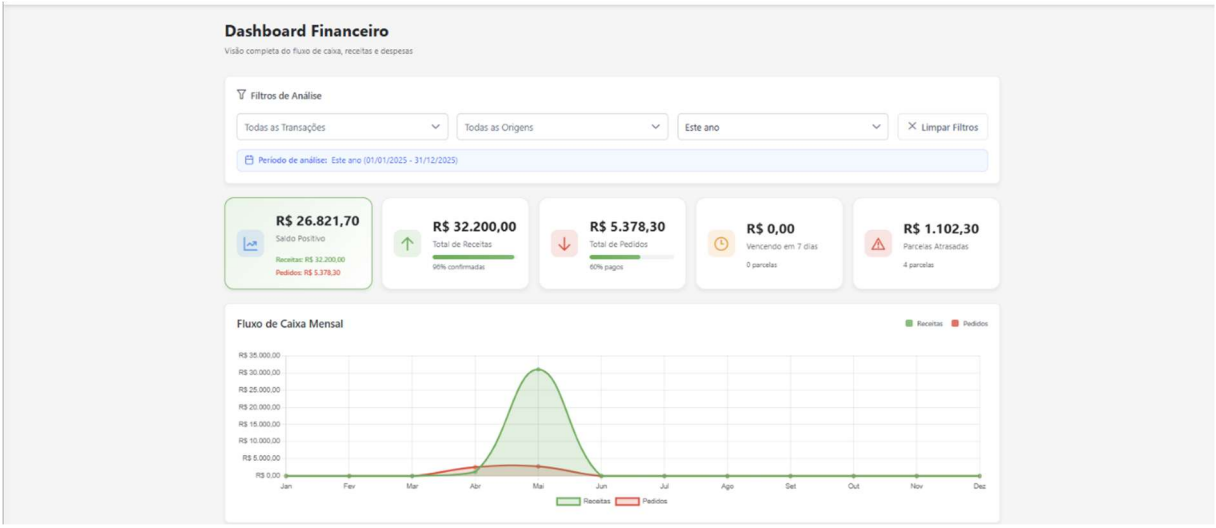
Fonte: Próprio Autor.

7.3.2 Dashboard Financeiro

O dashboard financeiro integra múltiplas dimensões da gestão econômica dos eventos através de indicadores de performance e visualizações temporais. Os cards superiores apresentam métricas críticas. O gráfico central "Fluxo de Caixa Mensal" utiliza uma visualização em área preenchida que mostra a evolução temporal das receitas (linha verde) e pedidos (linha

vermelha), permitindo identificar períodos de maior pressão financeira e oportunidades de otimização.

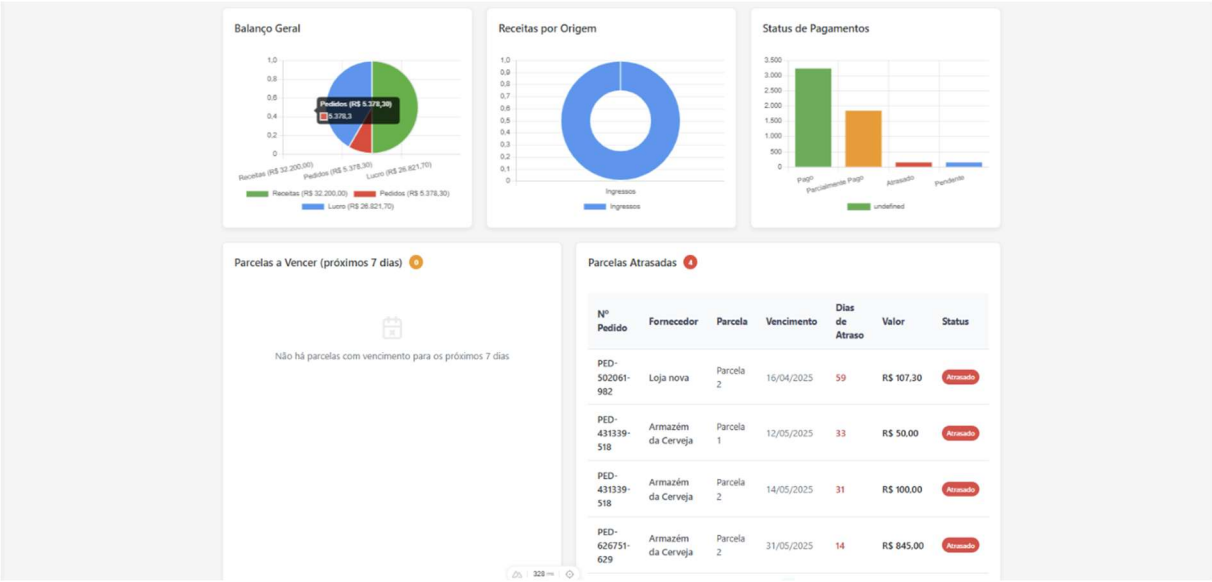
Figura 17 - Dashboard Financeiro 1



Fonte: Próprio Autor.

A análise financeira se aprofunda através de visualizações complementares que oferecem diferentes perspectivas dos dados econômicos. O gráfico "Balanço Geral" em formato donut apresenta a proporção entre receitas, despesas e lucro líquido, facilitando a compreensão da estrutura financeira global. O gráfico "Receitas por Origem" identifica as principais fontes de renda através de segmentação visual, enquanto o "Status de Pagamentos" em barras horizontais mostra a distribuição entre pagamentos realizados, pendentes e em atraso. A tabela "Parcelas Atrasadas" lista compromissos específicos com fornecedores, incluindo números de pedido, fornecedor, parcela, vencimento, valor e status atual.

Figura 18 - Dashboard Financeiro 2



Fonte: Próprio Autor.

O terceiro painel financeiro apresenta listagens detalhadas organizadas em duas seções principais. A seção "Receitas Recentes" tabula as últimas entradas financeiras com informações sobre descrição, origem, data, valor e status de confirmação, incluindo itens como "Venda do Ingresso João Rock" e "Sangria Promoter". A seção "Pedidos Recentes" complementa a análise com informações sobre saídas de caixa, listando número do pedido, fornecedor, data, valor e status de pagamento. Esta visualização tabular permite rastreabilidade completa das movimentações financeiras e facilita a reconciliação de contas.

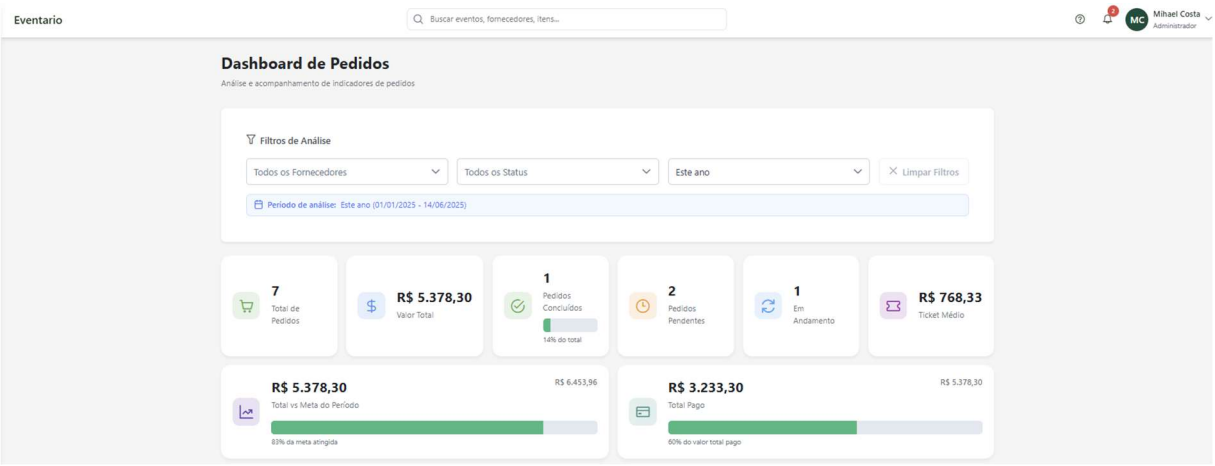
Figura 19 - Dashboard Financeiro 3

Fonte: Próprio Autor.

7.3.3 Dashboard de Pedidos

O dashboard de pedidos oferece visão operacional abrangente sobre o processo de aquisições e relacionamento com fornecedores.

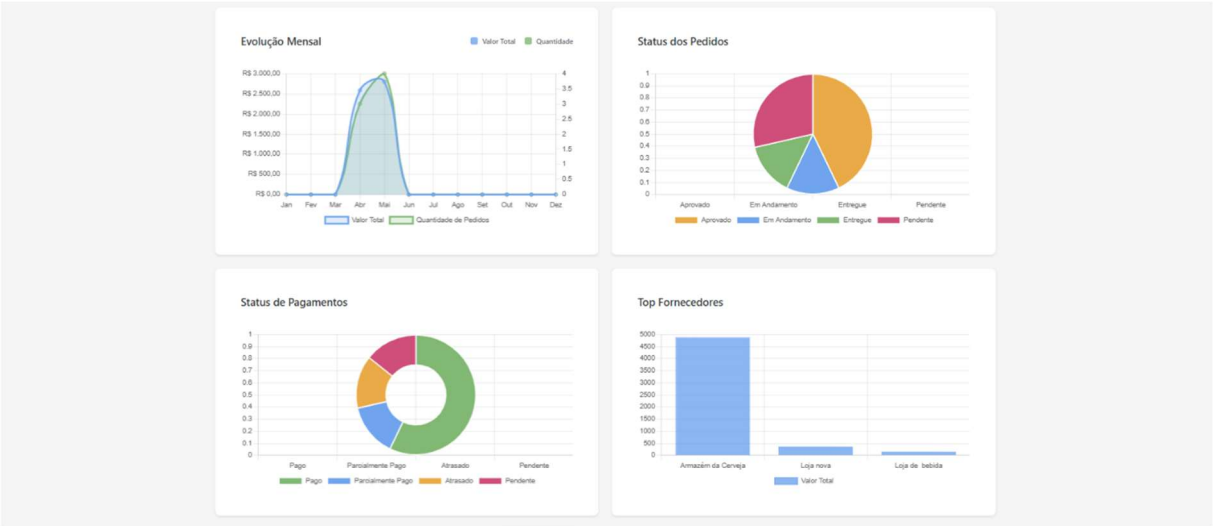
Figura 20 - Dashboard Pedidos 1



Fonte: Próprio Autor.

A análise comportamental dos pedidos é apresentada através de visualizações que revelam padrões temporais e distribuições categóricas. O gráfico "Evolução Mensal" em linha temporal mostra a variação dos valores de pedidos ao longo do ano, com picos identificáveis que correspondem a períodos de maior atividade de eventos. O gráfico circular "Status dos Pedidos" segmenta os pedidos por situação atual (Entregue, Aprovado, Em andamento, Pendente), oferecendo visão imediata da distribuição operacional. O gráfico "Top Fornecedores" em barras verticais identifica os parceiros com maior volume de negócios, destacando "Armazém da Cerveja" como fornecedor principal.

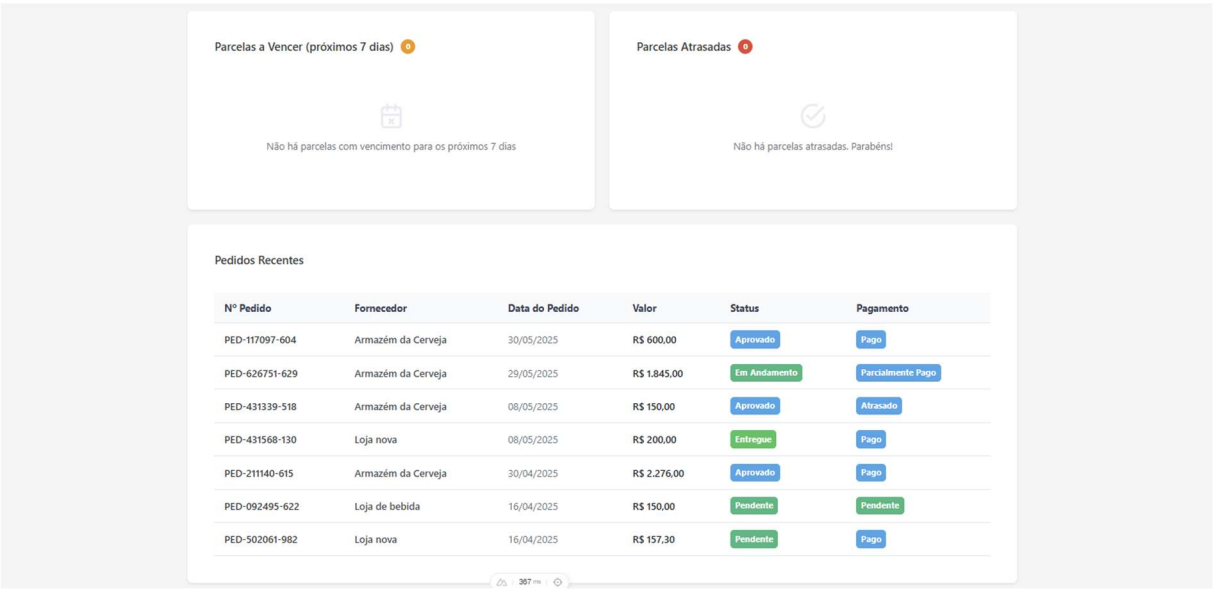
Figura 21 - Dashboard Pedidos 2



Fonte: Próprio Autor.

O painel operacional de pedidos apresenta informações detalhadas para acompanhamento de performance e identificação de gargalos. A seção "Parcelas a Vencer (próximos 7 dias)" alerta sobre compromissos financeiros iminentes, enquanto "Parcelas Atrasadas" identifica problemas de fluxo de caixa que requerem atenção imediata. A tabela "Pedidos Recentes" consolida informações operacionais incluindo número do pedido, fornecedor, data de pedido, valor, status e forma de pagamento, facilitando o acompanhamento individual de cada transação e identificação de pedidos.

Figura 22 - Dashboard Pedidos 3

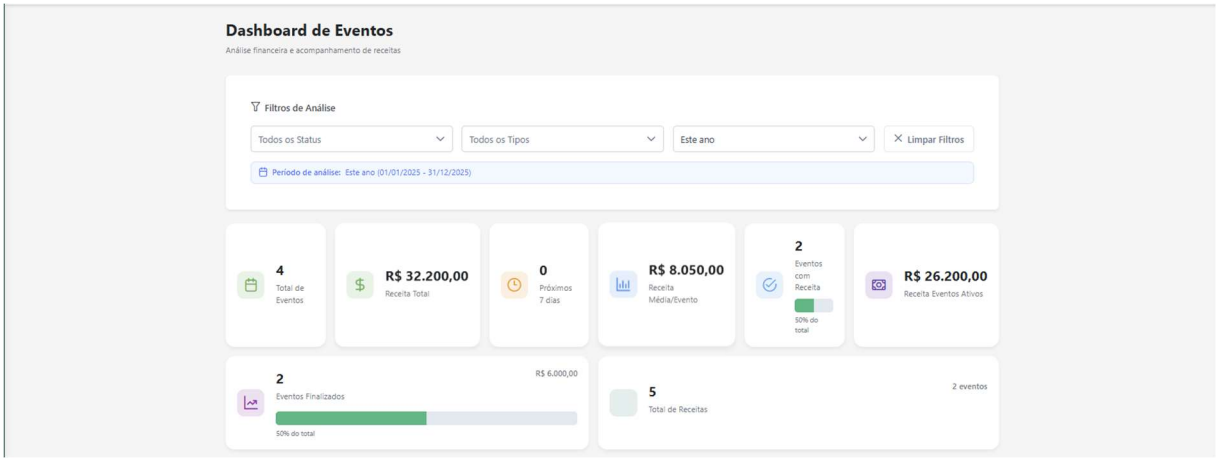


Fonte: Próprio Autor.

7.3.4 Dashboard de Eventos

O dashboard de eventos apresenta métricas estratégicas essenciais para avaliação de performance da carteira de eventos. Os indicadores principais incluem quantidade total de eventos (4), receita (R$ 32.200,00), receita média por evento (R$ 8.050,00), eventos com receita (2), e receita de eventos ativos (R$ 26.200,00). O gráfico de progresso mostra eventos finalizados versus em andamento.

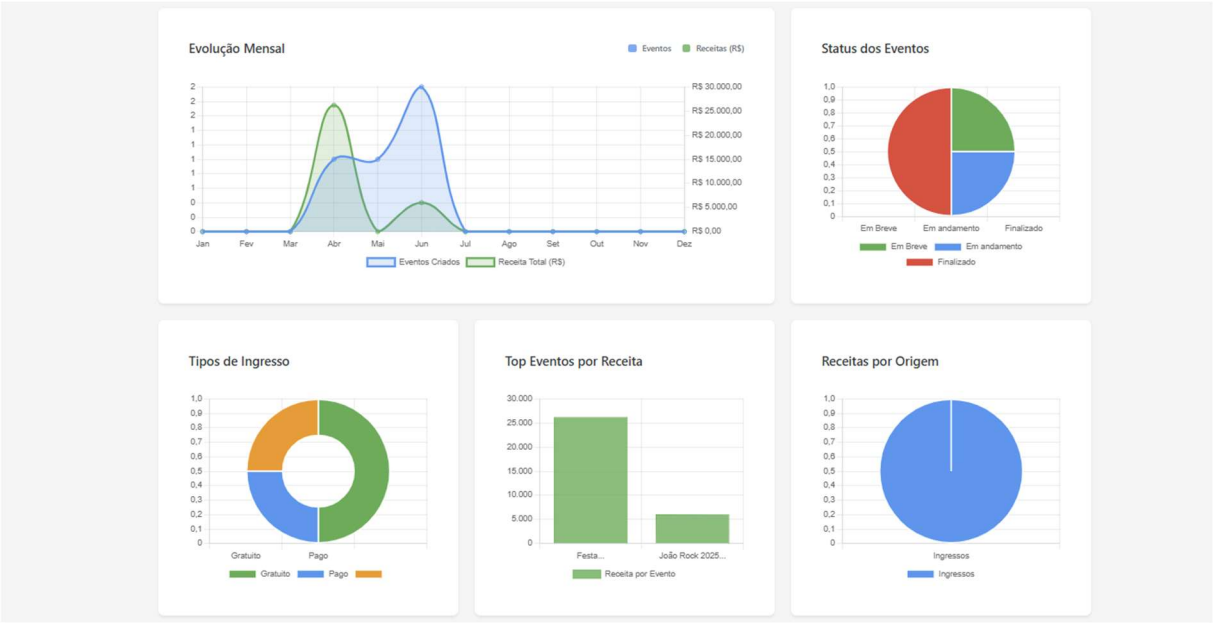
Figura 23 - Dashboard de Eventos 1



Fonte: Próprio Autor.

A análise temporal e categórica dos eventos revela padrões importantes para planejamento estratégico. O gráfico "Evolução Mensal" combina linhas para "Eventos Criados" e "Receita Total (R$)", mostrando correlações entre volume de eventos e performance financeira ao longo do ano. O gráfico circular "Status dos Eventos" segmenta a carteira por situação atual (Em Breve, Em andamento, Finalizado), oferecendo visão da distribuição temporal. Os gráficos "Tipos de Ingresso", "Top Eventos por Receita" e "Receitas por Origem" fornecem análises complementares sobre estrutura de preços, eventos mais rentáveis e diversificação de fontes de receita.

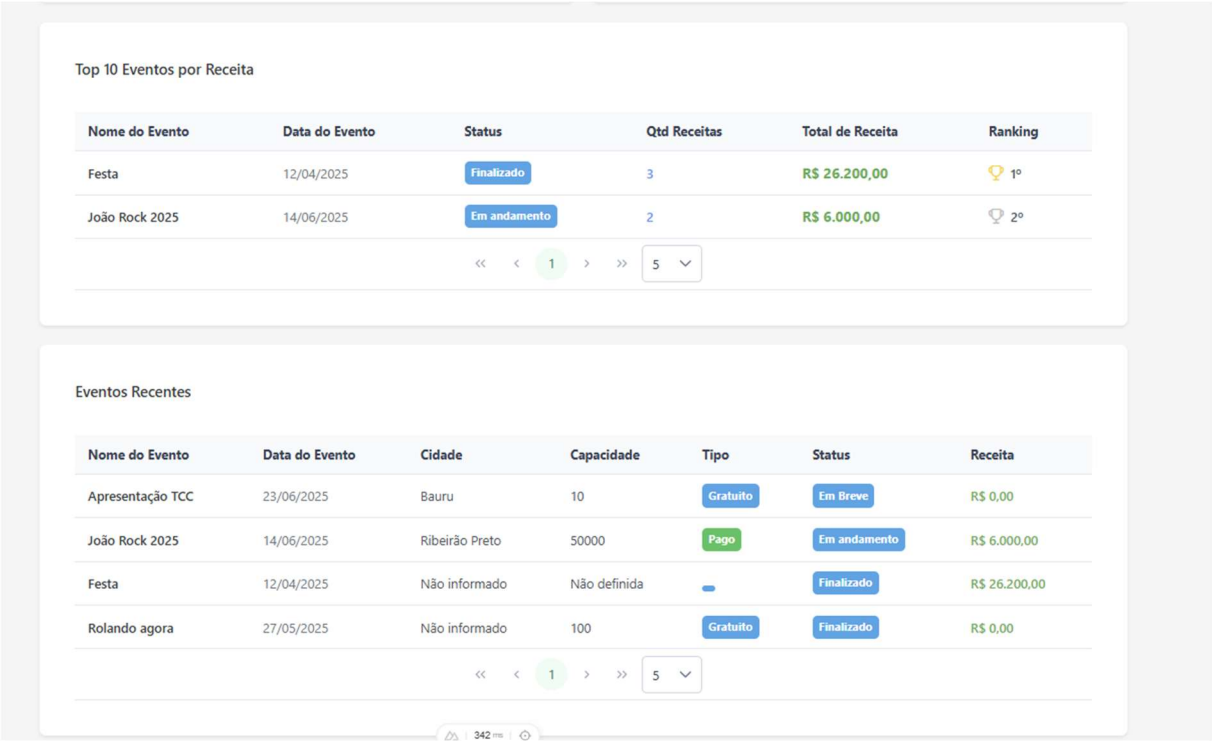
Figura 24 - Dashboard de Eventos 2



Fonte: Próprio Autor.

O painel de performance de eventos apresenta rankings e listagens detalhadas que facilitam a identificação de sucessos e oportunidades de melhoria. A tabela "Top 10 Eventos por Receita" lista os eventos mais rentáveis com informações sobre nome, data, status, quantidade de receitas, receita total e ranking relativo, destacando "Festa" (R$ 26.200,00) e "João Rock 2025" (R$ 6.000,00) como principais geradores de receita. A seção "Eventos Recentes" complementa com informações operacionais incluindo nome, data, cidade, capacidade, tipo, status e receita, fornecendo visão abrangente para tomada de decisões estratégicas sobre futuras programações.

Figura 25 - Dashboard de Eventos 3



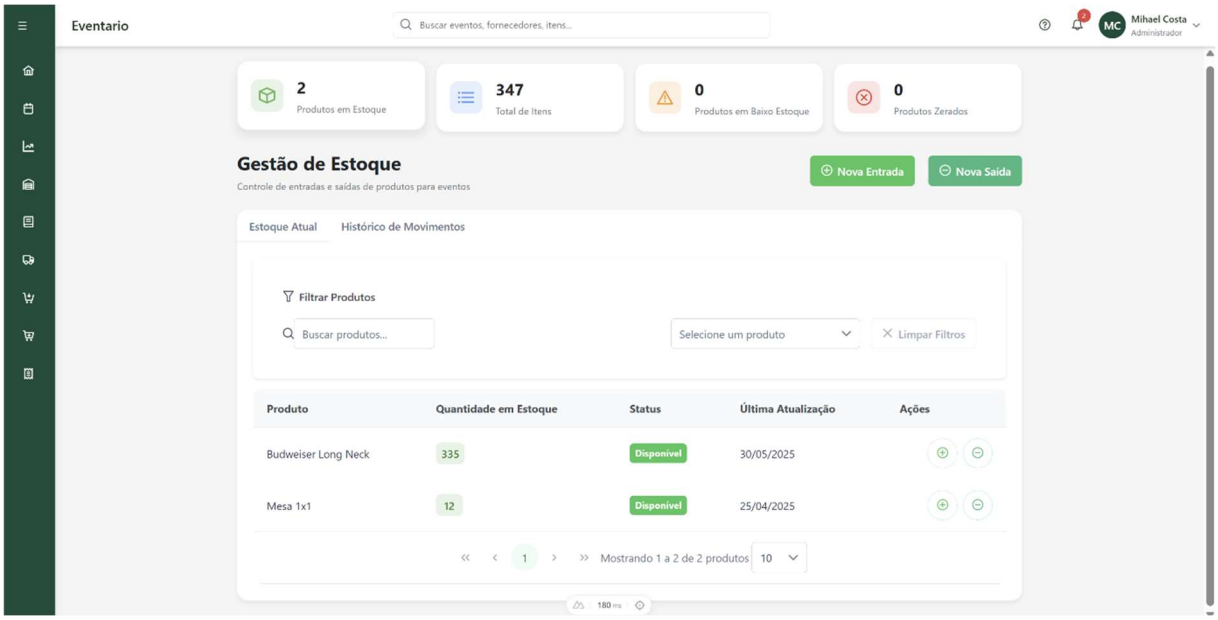
Fonte: Próprio Autor.

7.4 Módulo de Estoque

O módulo de estoque oferece controle completo sobre inventário através de cards informativos (2 produtos em estoque, 347 itens totais) e ferramentas de filtro por produto e evento. A interface principal exibe lista de produtos com quantidades atuais e status de disponibilidade. O formulário de movimentação permite registrar entradas e saídas com seleção de produto,

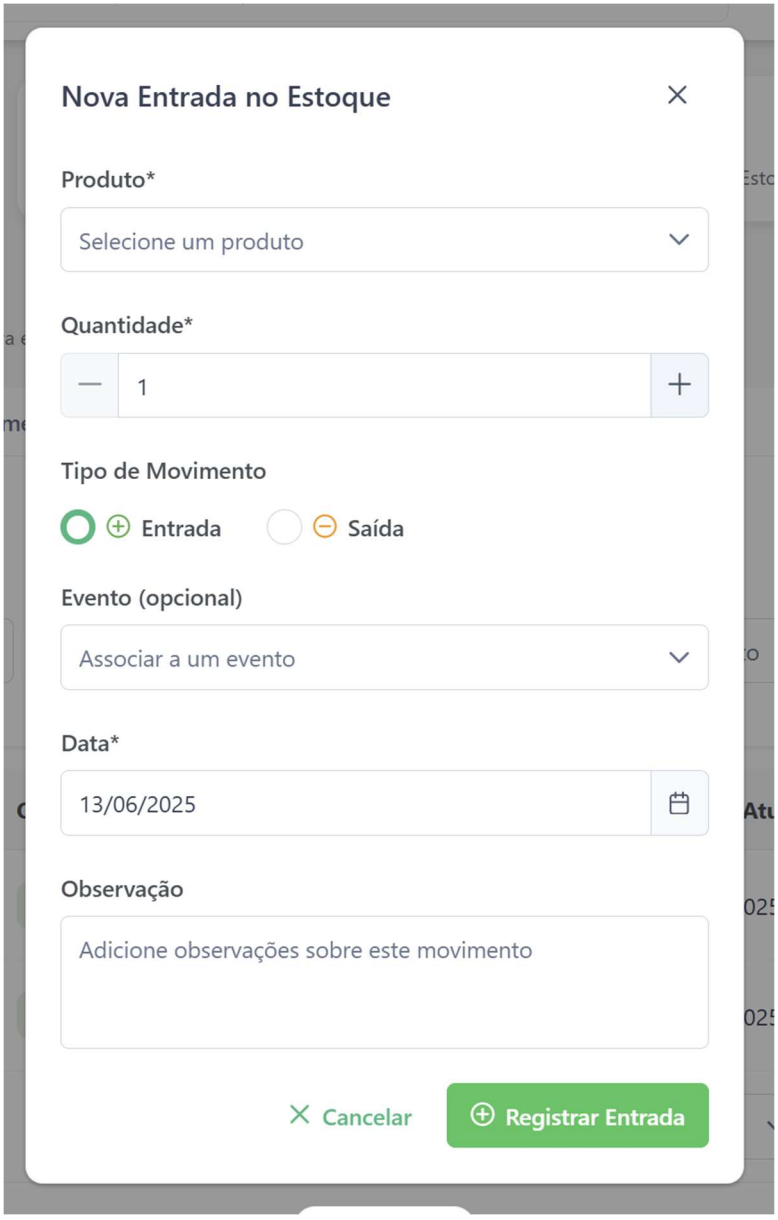
quantidade, tipo de movimento e associação opcional a eventos. O histórico apresenta todas as movimentações em formato tabular com filtros por tipo, produto e período.

Figura 26- Módulo de Estoque



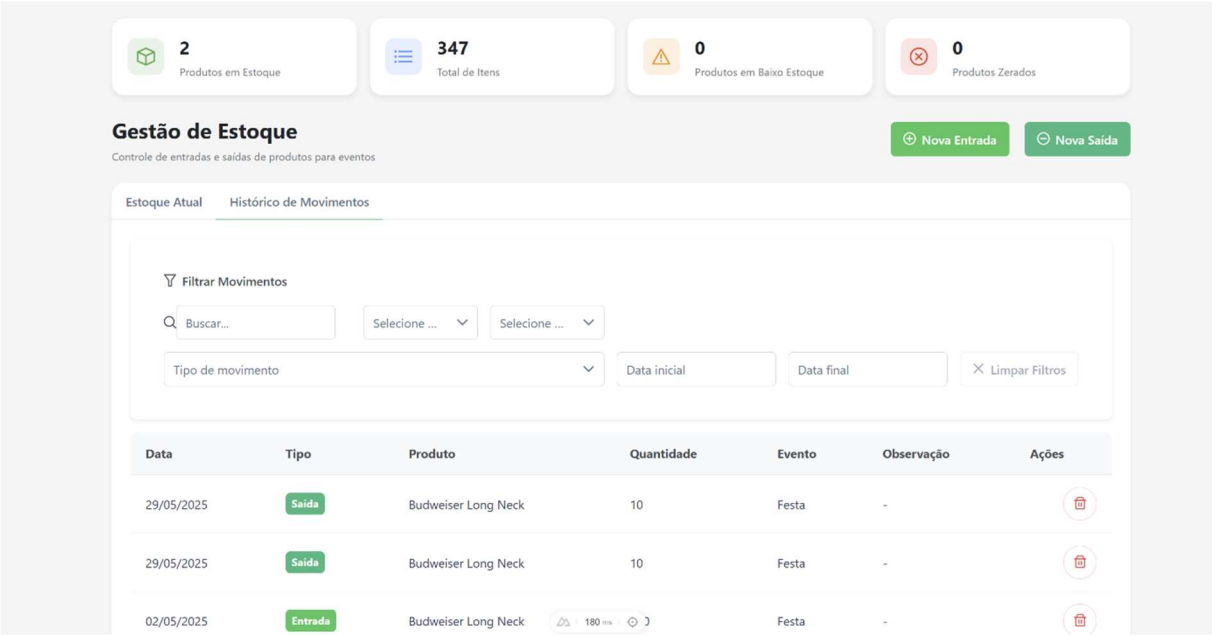
Fonte: Próprio Autor.

Figura 27 - Nova Movimentação de Estoque



Fonte: Próprio Autor.

Figura 28 - Histórico de Movimentos

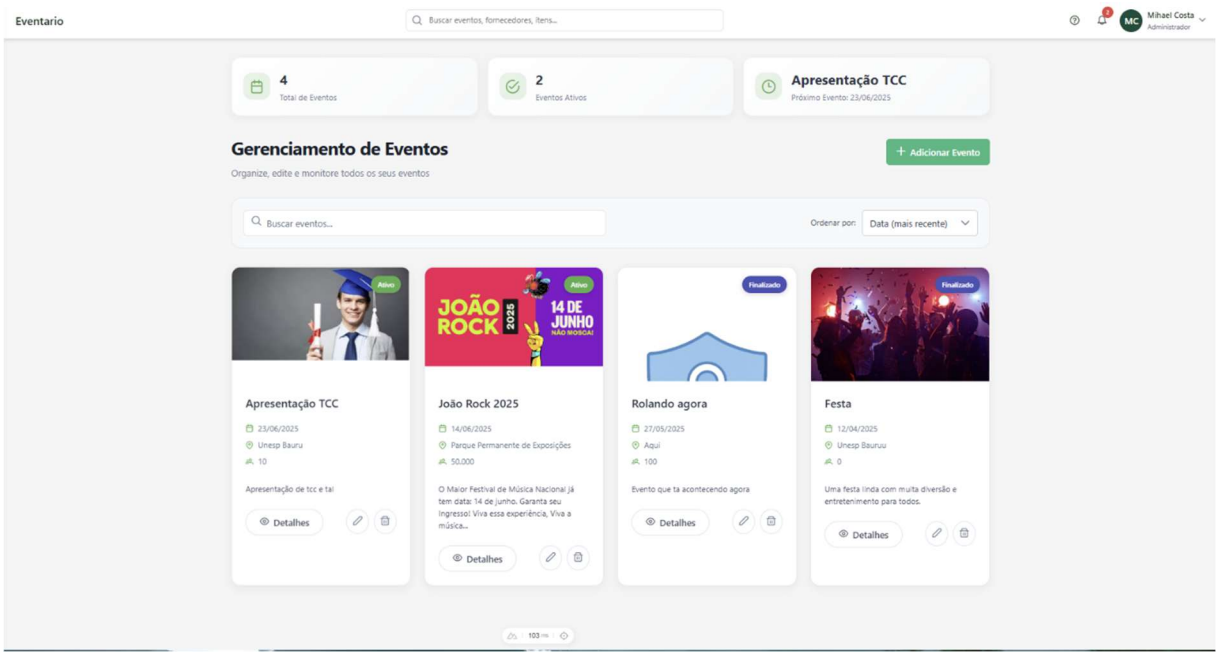


Fonte: Próprio Autor.

7.5 Módulo de Eventos

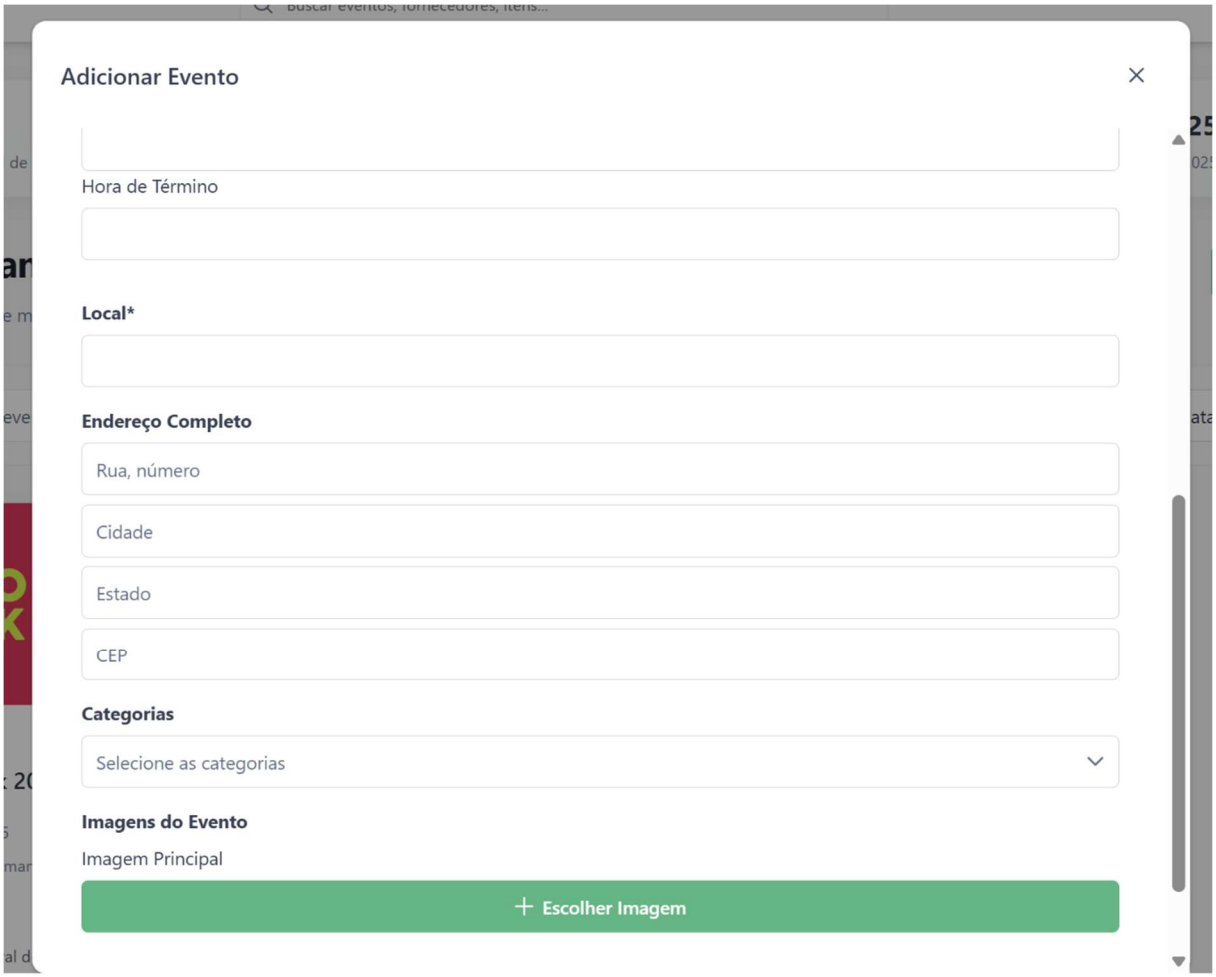
Como funcionalidade central do sistema, o módulo apresenta cards com métricas de eventos (4 total, 2 concluídos) e galeria visual dos eventos cadastrados. O formulário de criação é dividido em abas: "Informações Básicas" para dados como nome, descrição, local e datas, e "Capacidade e Ingressos" para definição de lotação máxima e tipos de ingresso (gratuito/pago). A tela de detalhes exibe informações completas incluindo imagem de capa, descrição, local e botões para edição e visualização detalhada. No detalhe, ao clicar no local é direcionado para o local no site do Google Maps.

Figura 29 - Módulo de Eventos



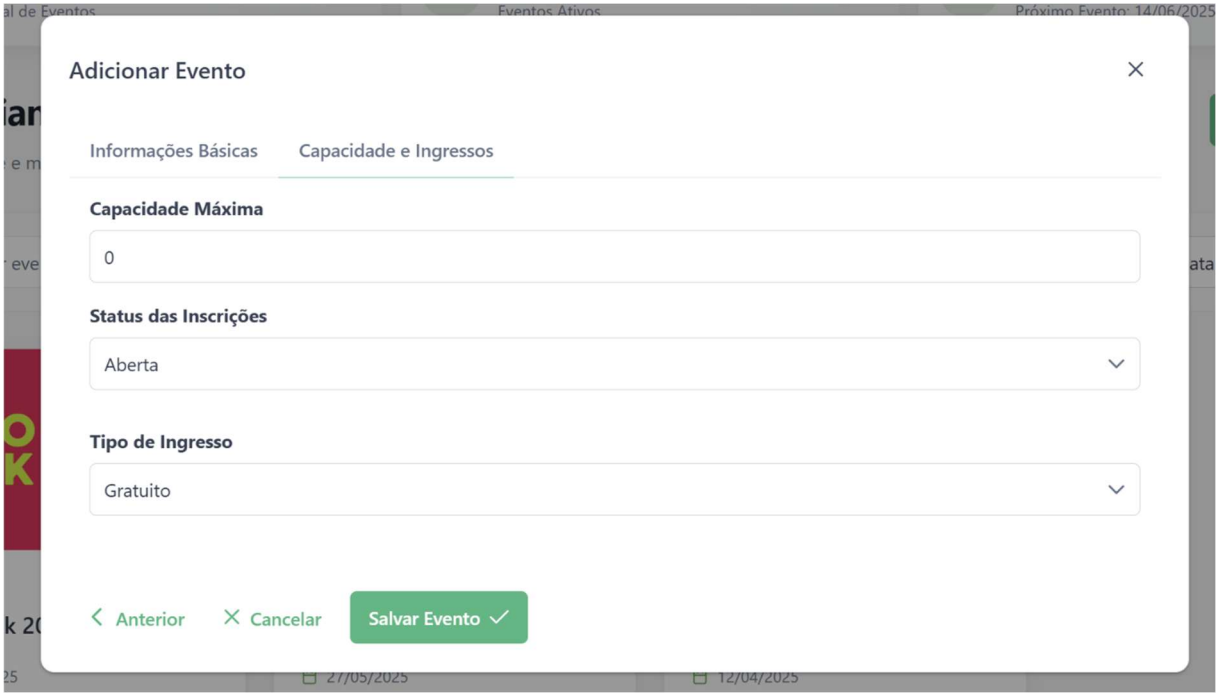
Fonte: Próprio Autor.

Figura 30 - Adicionar Evento 1



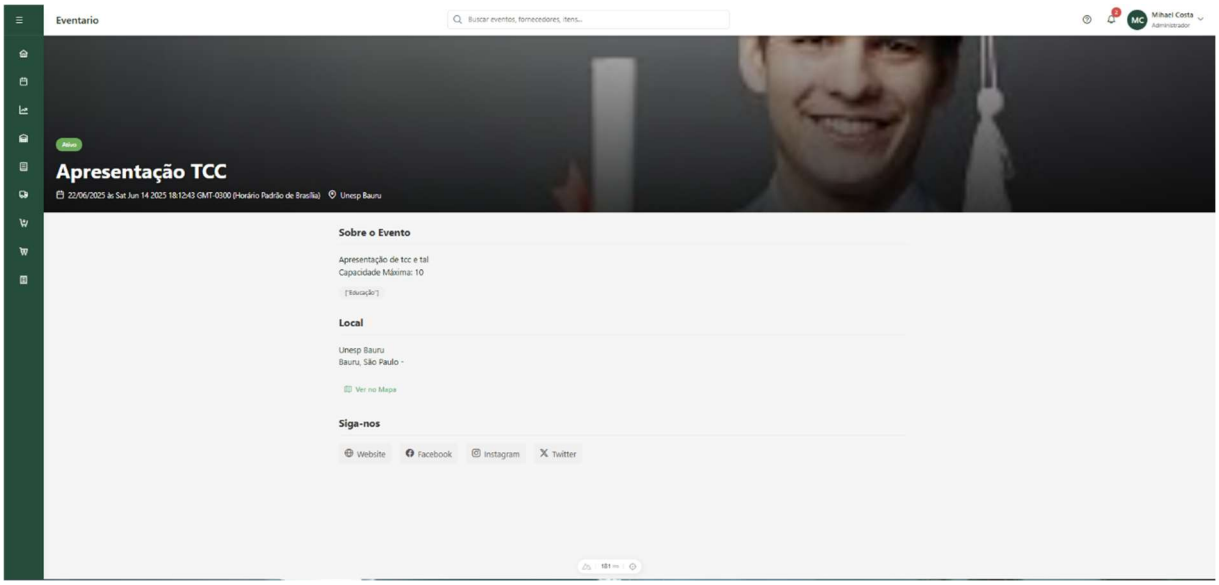
Fonte: Próprio Autor.

Figura 31 - Adicionar Evento 2



Fonte: Próprio Autor.

Figura 32 - Detalhe Evento

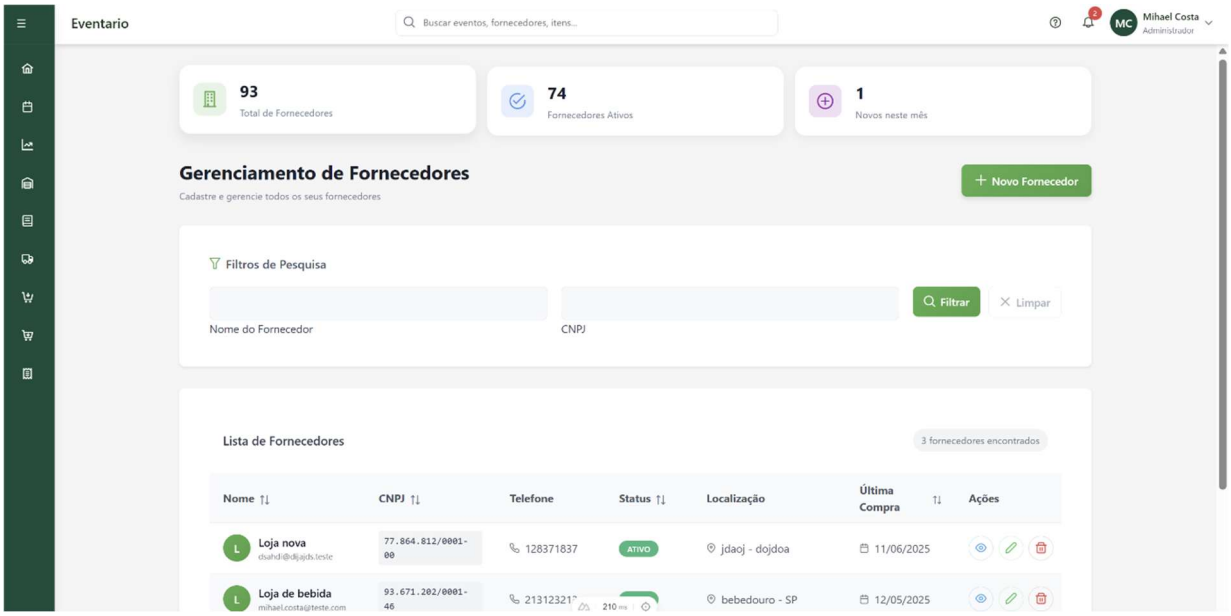


Fonte: Próprio Autor.

7.6 Módulo de Fornecedores

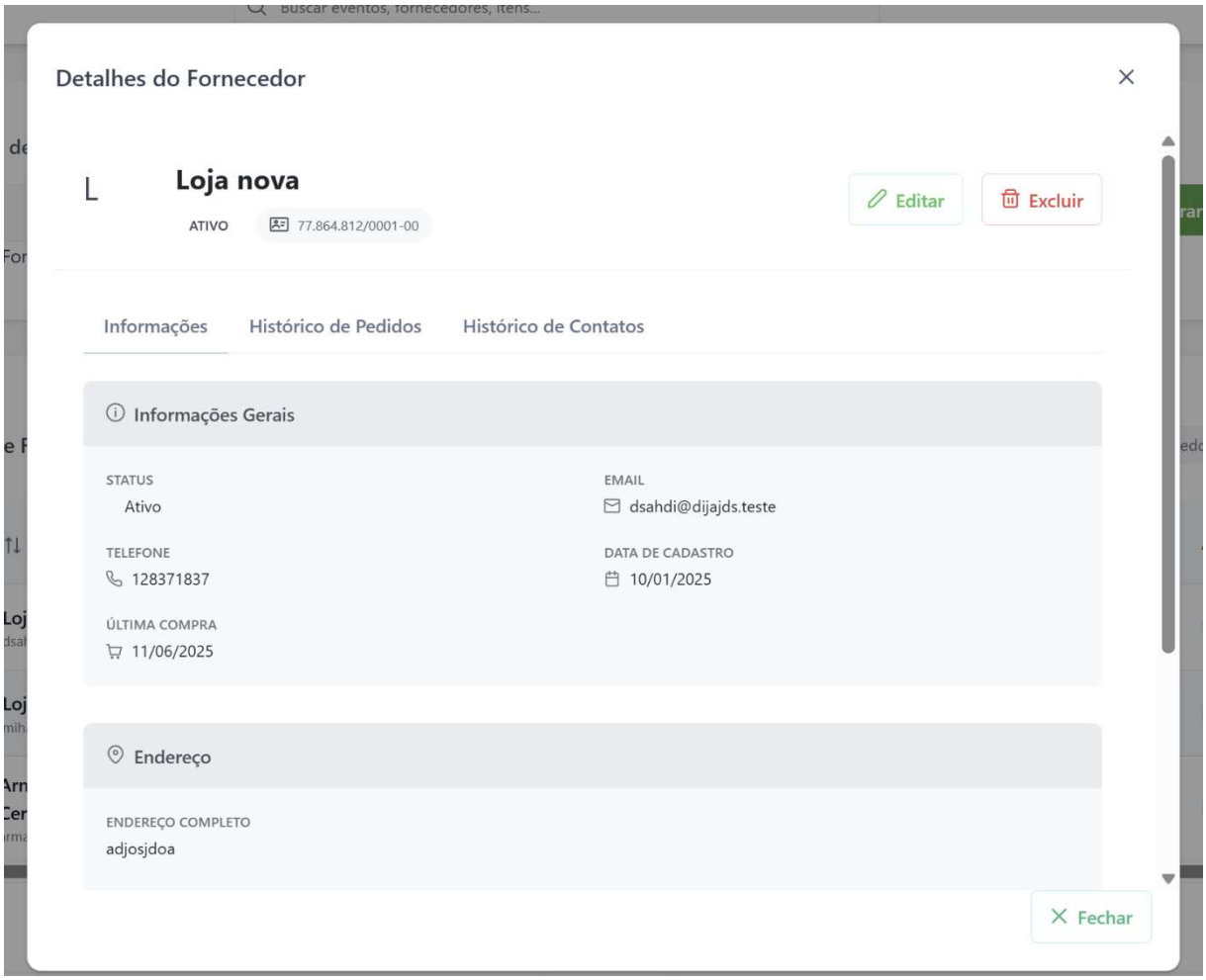
A gestão de fornecedores apresenta cards com estatísticas (93 total, 74 ativos, 1 novo) e lista filtrada por nome e CNPJ. O perfil do fornecedor organiza informações em abas: dados gerais (status, contato, última compra), histórico de pedidos com valores e status, e histórico de contatos. O cadastro utiliza duas etapas: informações básicas (nome, CNPJ, telefone, email) e endereço completo com validação de CEP.

Figura 33 - Módulo de Fornecedores



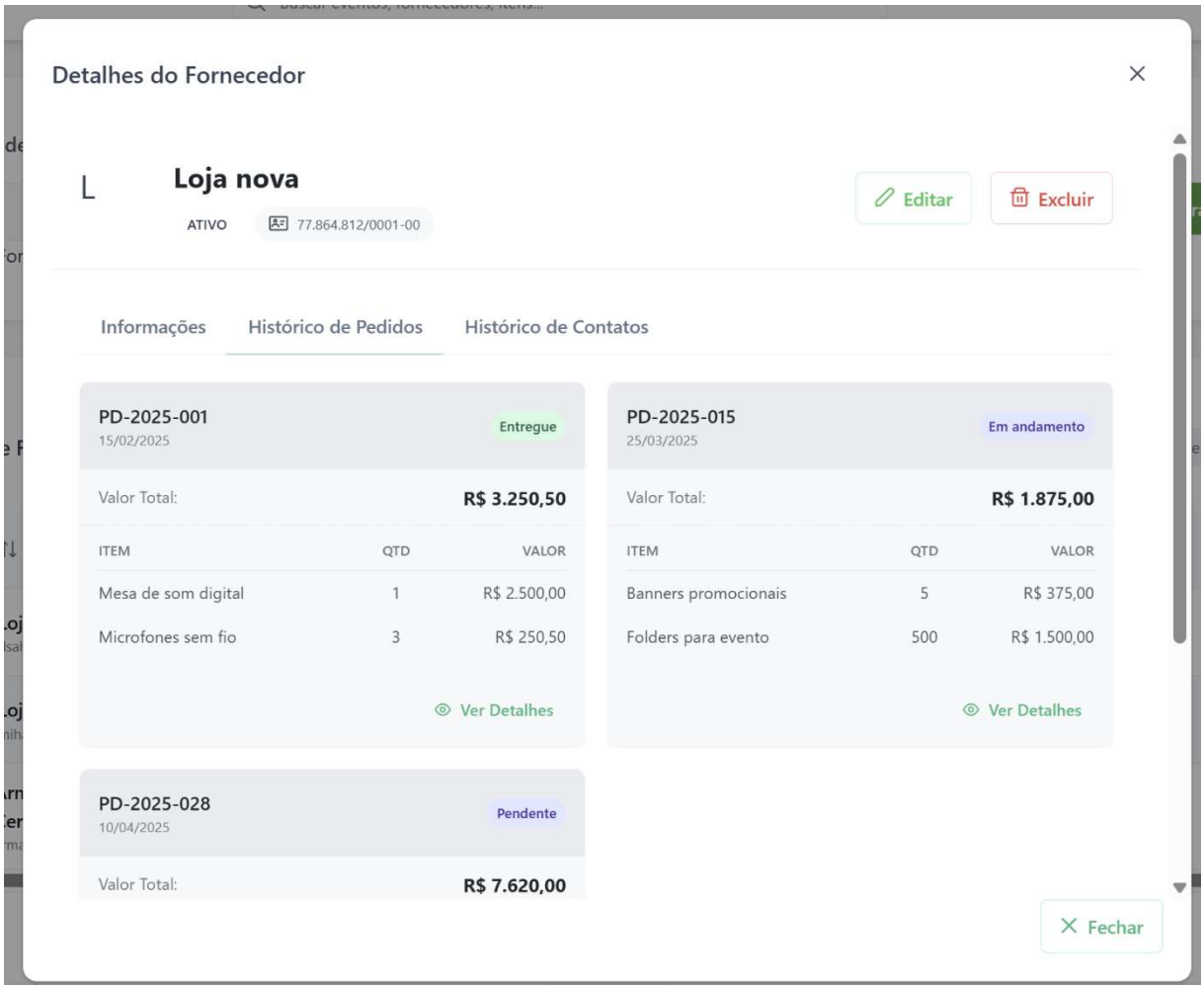
Fonte: Próprio Autor.

Figura 34 - Detalhes do Fornecedor



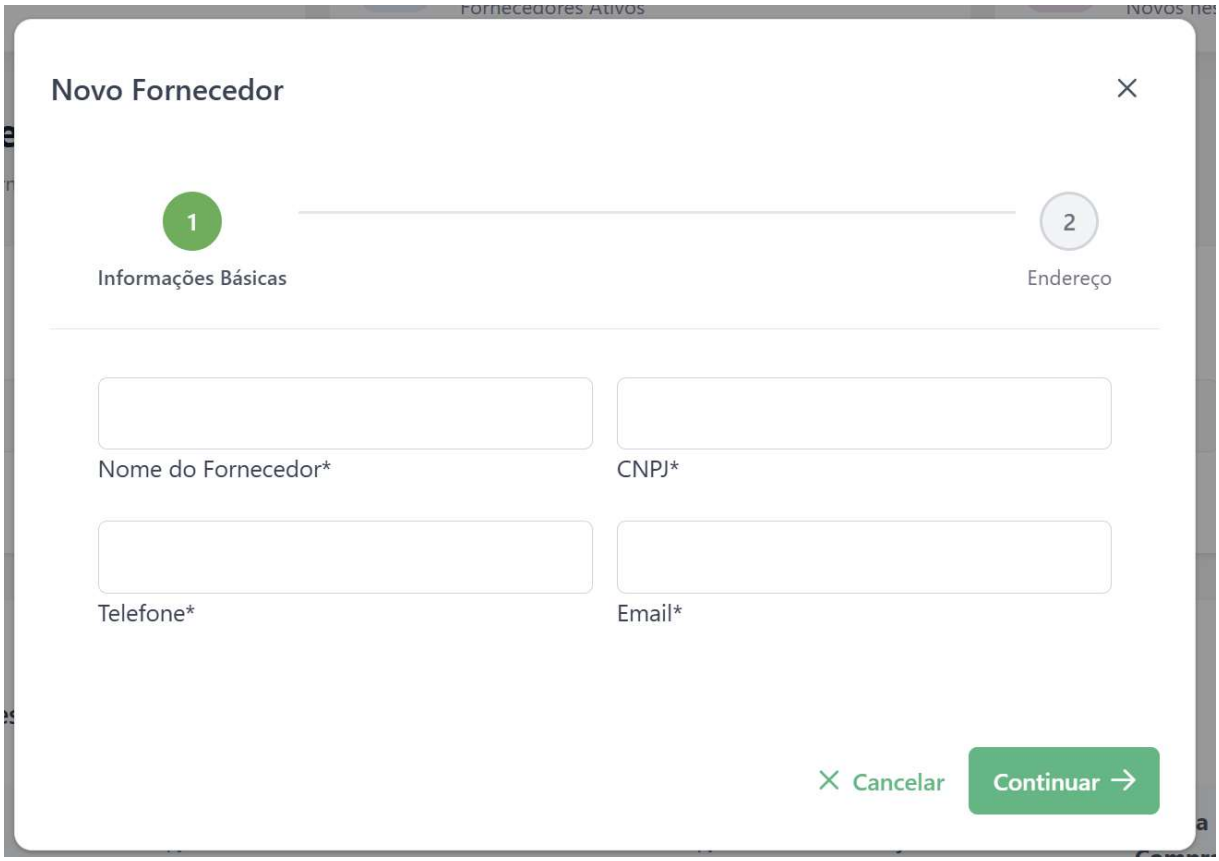
Fonte: Próprio Autor.

Figura 35 - Histórico de Pedidos do Fornecedor



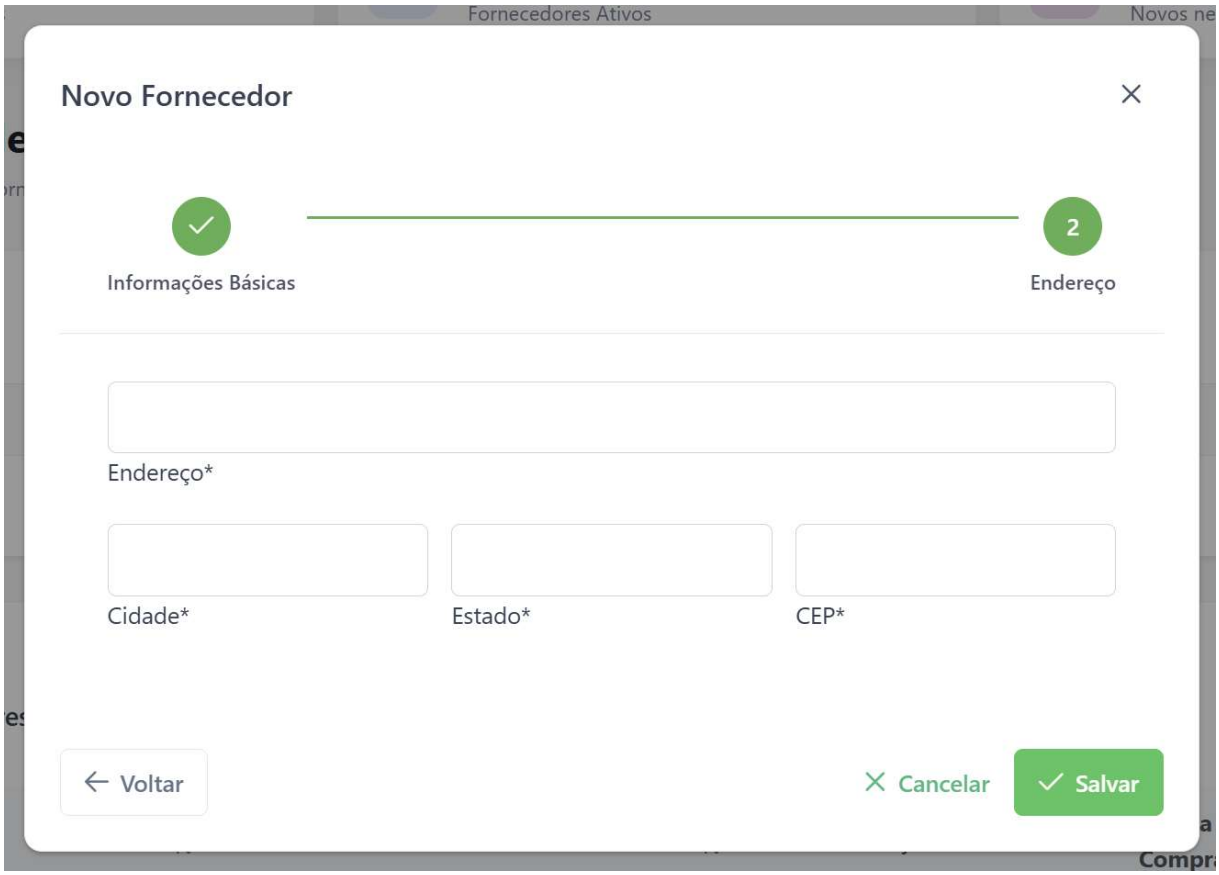
Fonte: Próprio Autor.

Figura 36 - Novo Fornecedor 1



Fonte: Próprio Autor.

Figura 37 - Novo Fornecedor 2

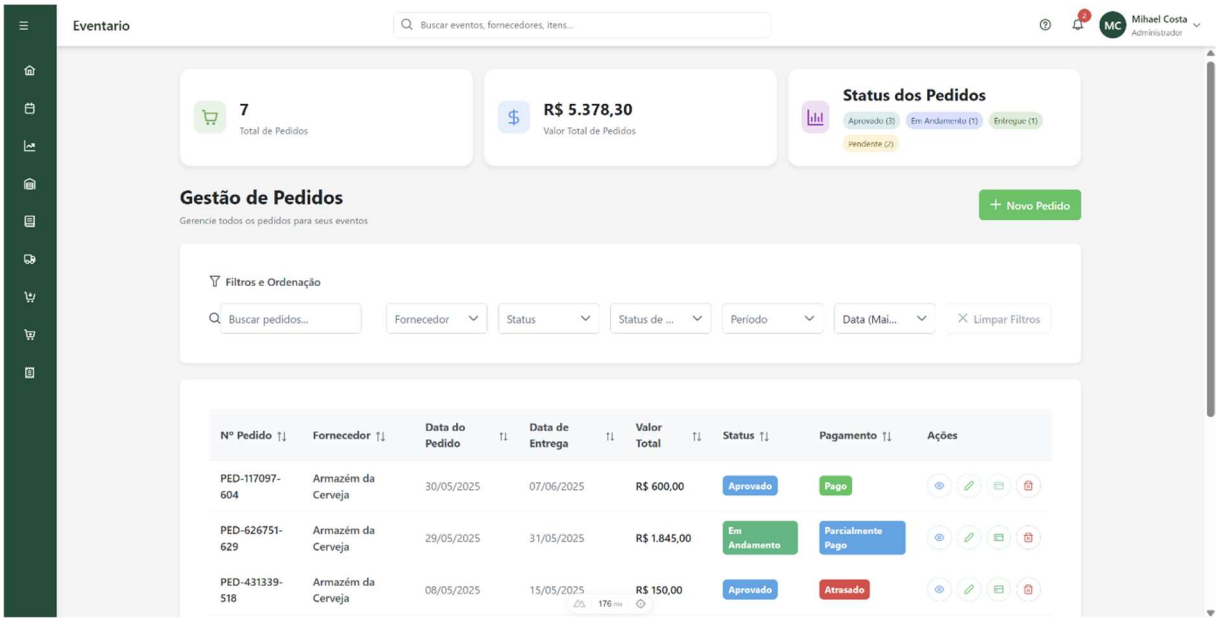


Fonte: Próprio Autor.

7.7 Módulo de Pedidos

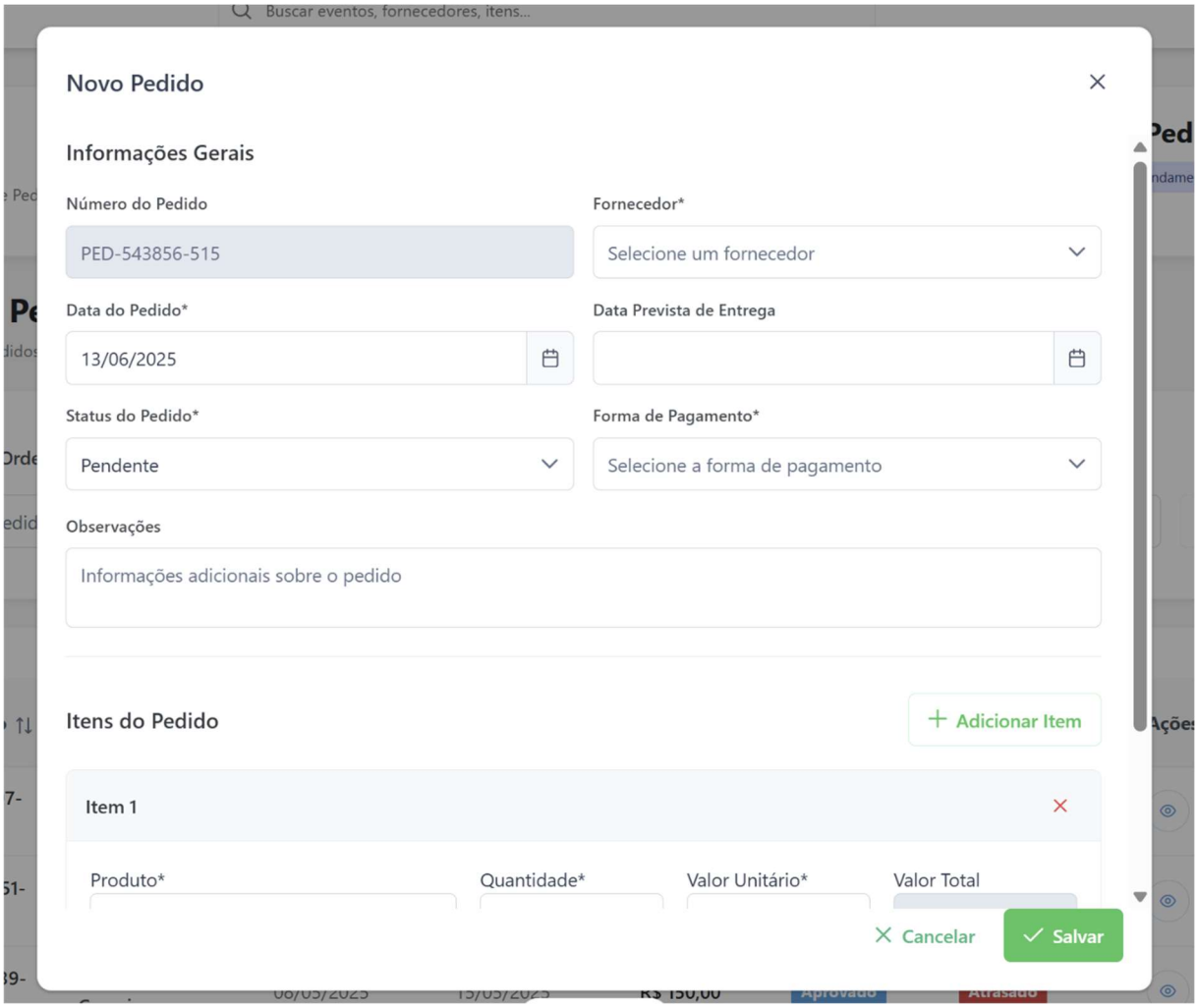
O módulo exibe cards com métricas de pedidos (7 total, R$ 5.378,30 em valor) e status distribuído por cores. A lista principal permite filtros por fornecedor, status e período, mostrando informações essenciais como número, fornecedor, datas e valores. O formulário de criação inclui dados gerais (fornecedor, datas, status), observações e seção dinâmica para itens do pedido. A gestão de pagamentos permite parcelamento com datas de vencimento e acompanhamento de status por parcela.

Figura 38 - Módulo de Pedidos



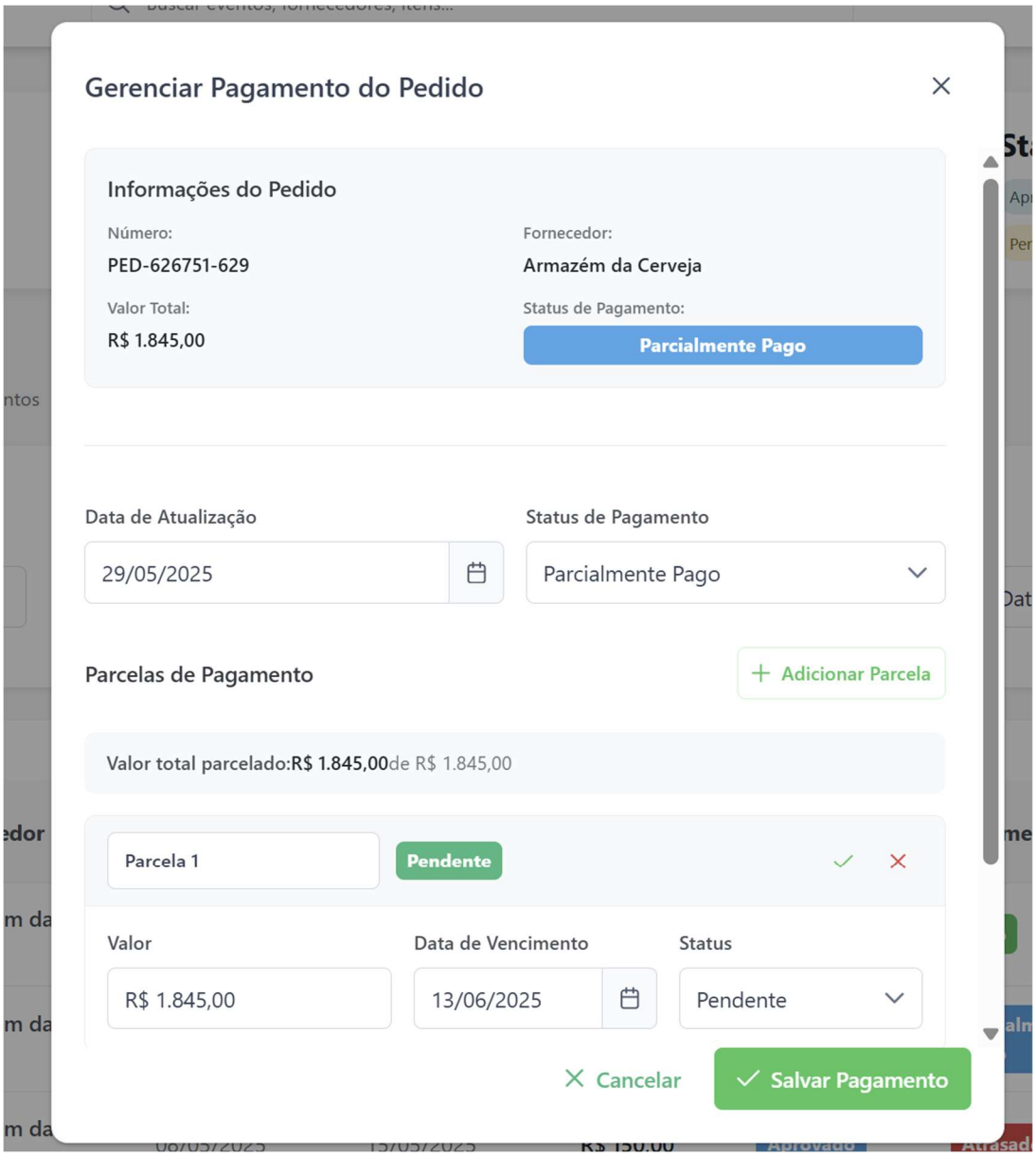
Fonte: Próprio Autor.

Figura 39 - Novo Pedido



Fonte: Próprio Autor.

Figura 40 - Pagamento do Pedido

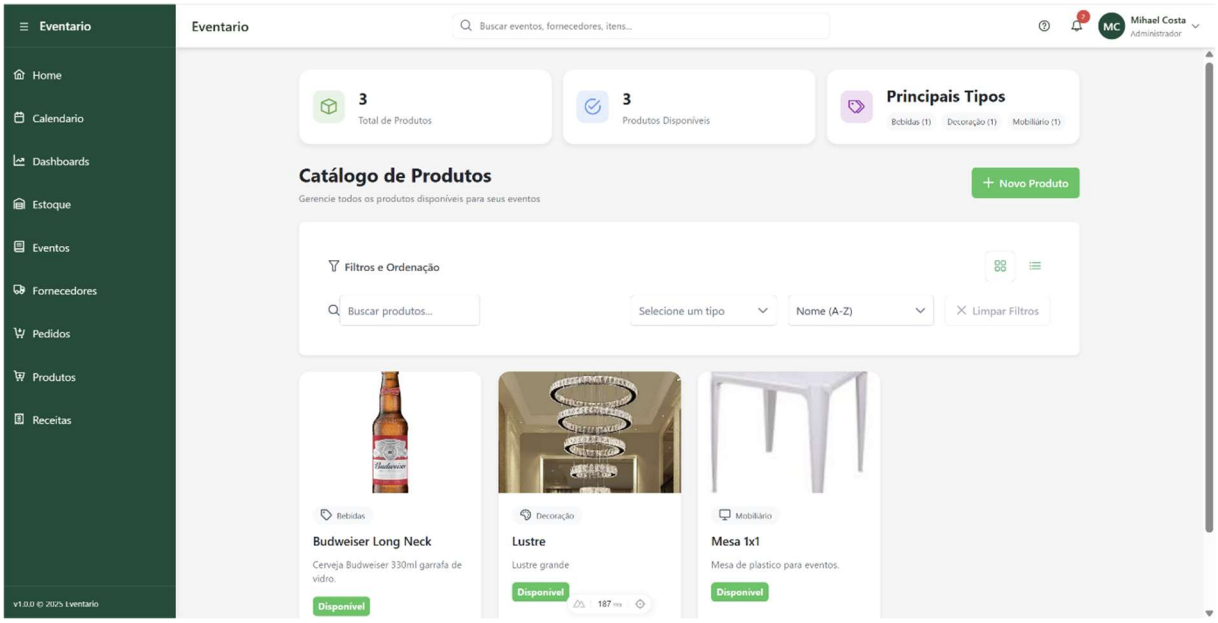


Fonte: Próprio Autor.

7.8 Módulo de Produtos

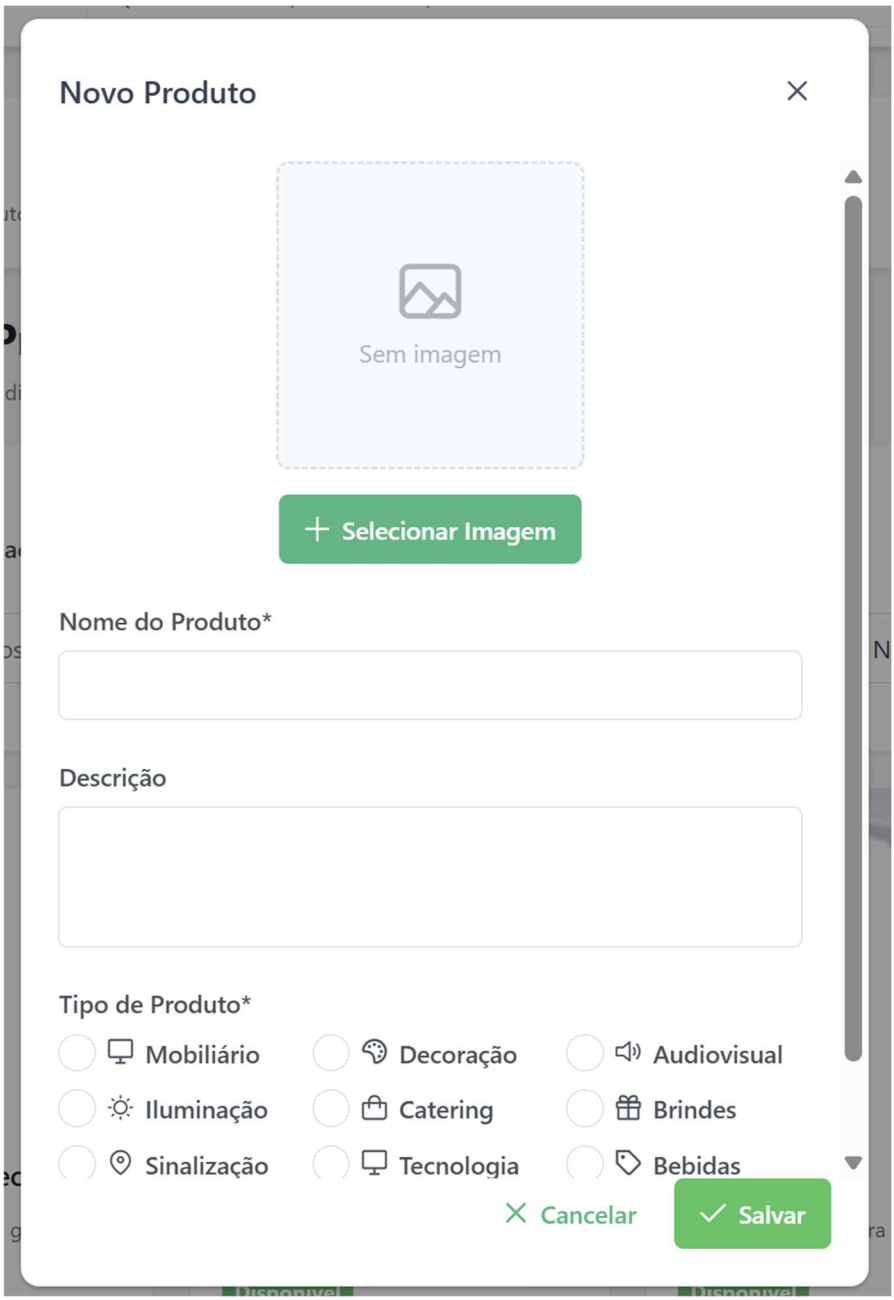
O catálogo apresenta cards com estatísticas e principais tipos por categoria (Bebidas, Decoração, Mobiliário). A galeria visual organiza produtos por tipo com informações de disponibilidade. O cadastro inclui upload de imagem, nome, descrição detalhada e seleção de categoria através de ícones organizados (Mobiliário, Decoração, Audiovisual, Iluminação, Catering, Tecnologia, Sinalização, Brindes, Bebidas).

Figura 41 - Módulo de Produtos



Fonte: Próprio Autor.

Figura 42 - Novo Produto



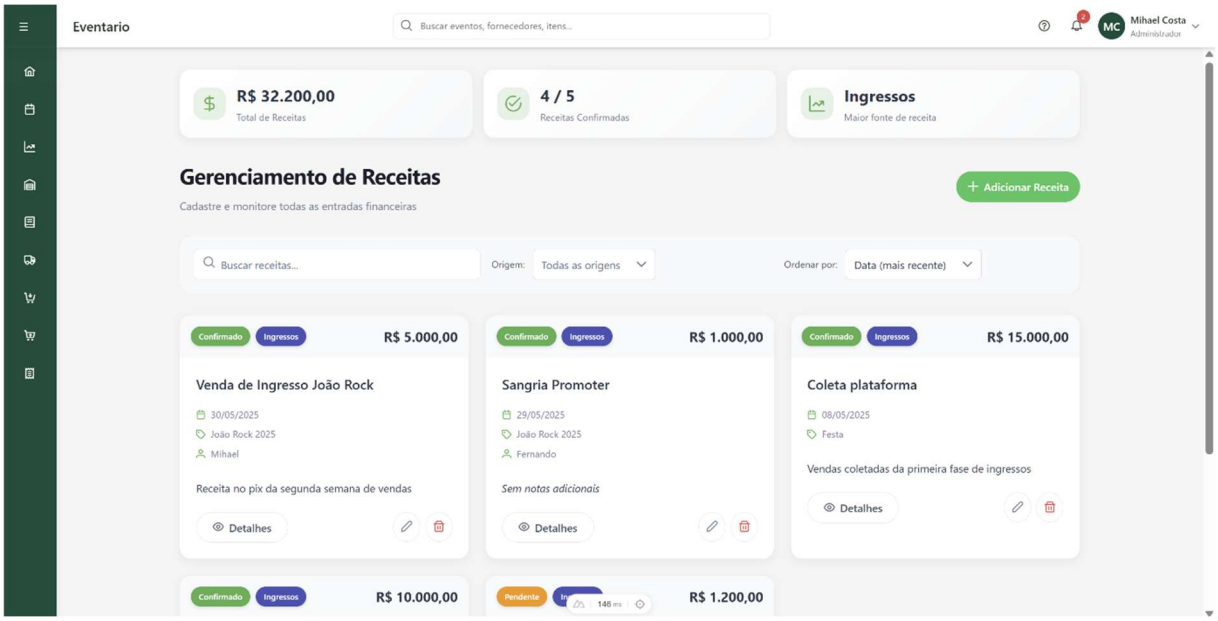
Fonte: Próprio Autor.

7.9 Módulo de Receitas

O módulo financeiro apresenta cards com valor total (R$ 32.200,00), receitas confirmadas (4/5) e tipos de receita (Ingressos). A interface principal exibe receitas em cards visuais organizados por evento, com filtros por origem e período. O formulário de cadastro inclui descrição, valor,

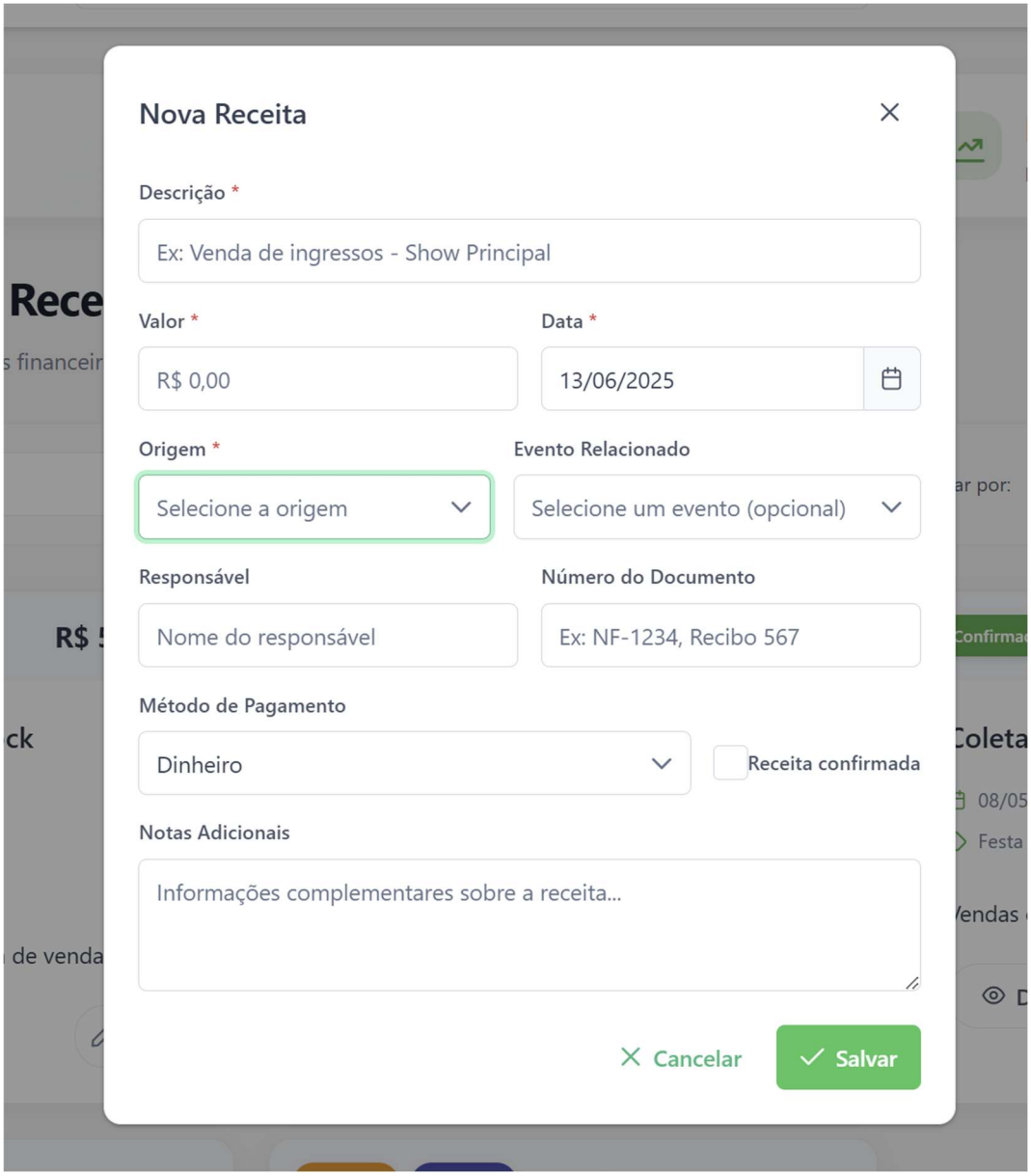
data, origem, evento relacionado (opcional), responsável, método de pagamento e campo para notas adicionais, permitindo rastreabilidade completa das entradas financeiras.

Figura 43 - Módulo de Receitas



Fonte: Próprio Autor.

Figura 44 - Nova Receita



Fonte: Próprio Autor.

7.10 Testes e Validações

A validação do sistema foi conduzida através de uma metodologia estruturada que contemplou diferentes níveis de verificação, demonstrando a robustez da solução desenvolvida.

7.10.1 Validação Técnica

A implementação de testes unitários utilizando o framework xUnit resultou em uma cobertura significativa de 85% do código-fonte, distribuída entre controllers (90%), services (85%), commands/queries (88%), models (75%) e validators (87%). Esta cobertura abrangente demonstra a confiabilidade do sistema e facilita futuras manutenções evolutivas.

Os testes implementados abrangeram aspectos fundamentais como validação de entrada de dados, verificação de permissões e autorização, processamento de arquivos multimídia, cálculos financeiros, operações de estoque e integridade referencial entre entidades.

7.10.2 Validação Funcional

Os testes de sistema, conduzidos seguindo metodologia de Caixa Preta, validaram o funcionamento integrado de todas as funcionalidades do ponto de vista do usuário final. Os cenários testados incluíram fluxos completos de gestão de eventos, controle financeiro e experiência do usuário em diferentes dispositivos, confirmando que o sistema atende aos requisitos funcionais e não funcionais especificados.

8 CONCLUSÃO

8.1 Principais Aspectos do Projeto

O Eventario constitui-se como uma solução tecnológica integrada para o gerenciamento de eventos, desenvolvida com o objetivo de centralizar e otimizar processos tradicionalmente dispersos em planilhas e sistemas isolados. O projeto demonstra a aplicação prática de arquiteturas modernas de software, integrando tecnologias consolidadas como Vue.js, .NET e SQL Server em uma plataforma web escalável e intuitiva.

Os principais aspectos que caracterizam o sistema incluem:

Centralização das Informações: O sistema oferece uma plataforma única para gerenciar todos os aspectos relacionados ao evento, desde a gestão financeira até o controle de fornecedores e calendário, eliminando a fragmentação informacional que caracteriza métodos tradicionais.

Interface Intuitiva: A interface do usuário foi projetada seguindo princípios de usabilidade moderna, permitindo que usuários de diferentes níveis de experiência possam operar o sistema de forma eficiente, com dashboards visuais e fluxos de trabalho otimizados.

Flexibilidade Arquitetural: O Eventario é flexível e adaptável a eventos de pequeno, médio e grande porte, com funcionalidades ajustáveis que atendem às necessidades específicas de cada contexto organizacional.

8.2 Ganhos Esperados e Impactos

A implementação do Eventario promete gerar impactos significativos na gestão de eventos:

Eficiência Operacional: A centralização das informações e automação de processos reduzem substancialmente o tempo e esforço necessários para gerenciar eventos, eliminando retrabalhos e otimizando a alocação de recursos humanos.

Redução de Erros: A integração das funções em um único sistema elimina inconsistências associadas à gestão dispersa em múltiplas planilhas e documentos, garantindo maior confiabilidade dos dados.

Melhoria na Tomada de Decisões: O sistema proporciona visão abrangente e em tempo real das operações do evento através de dashboards analíticos, facilitando a análise de dados e a tomada de decisões baseadas em evidências.

Impacto Econômico: A otimização de processos e melhor controle de recursos resulta em redução de custos operacionais e maior precisão no controle orçamentário, especialmente relevante para organizadores de pequeno e médio porte.

8.3 Desafios Enfrentados e Mitigação de Riscos

O desenvolvimento do projeto enfrentou desafios técnicos e organizacionais que foram adequadamente endereçados através de estratégias de mitigação estruturadas.

8.3.1 Desafios Técnicos

Flexibilidade das Regras: O desenvolvimento de regras e funcionalidades adaptáveis a diferentes portes e tipos de eventos exigiu uma arquitetura modular e extensível, implementada através dos padrões Clean Architecture e CQRS.

Integração entre Módulos: A garantia de comunicação eficiente entre todos os módulos do sistema foi assegurada através da implementação de uma API RESTful robusta e da aplicação rigorosa de princípios de design de software.

Escolha Tecnológica: A seleção criteriosa de tecnologias consolidadas (Vue.js, .NET, SQL Server) minimizou riscos relacionados à disponibilidade de componentes e suporte técnico.

8.3.2 Gestão de Riscos

Os principais riscos identificados e suas estratégias de mitigação incluíram:

- **Complexidade Subestimada**: Análises detalhadas e revisões regulares das estimativas, implementando abordagem ágil para ajustes no escopo.
- **Disponibilidade de Recursos**: Planejamento antecipado de aquisições e identificação de fornecedores alternativos.
- **Mudanças de Requisitos**: Arquitetura flexível que acomoda evolução de funcionalidades sem comprometer a estrutura base.

8.4 Limitações e Trabalhos Futuros

Embora o sistema desenvolvido atenda aos objetivos propostos, algumas limitações foram identificadas, abrindo oportunidades para trabalhos futuros:

Limitações Atuais:

- Ausência de integração com recursos de vendas de ingresso
- Ausência de funcionalidades de inteligência artificial para otimização automática
- Escalabilidade não testada para eventos de grande porte

Trabalhos Futuros:

- Melhores métricas de atribuição de uso de recursos aos eventos
- Desenvolvimento de aplicativo móvel nativo
- Integração com APIs de outras plataformas para sincronia automática de eventos
- Implementação de um ChatBot com Inteligência Artificial

8.5 Avaliação Final da Viabilidade

A viabilidade do projeto Eventario é considerada alta e comprovada através dos resultados obtidos:

Viabilidade Técnica: Confirmada através da implementação bem-sucedida utilizando tecnologias consolidadas e arquitetura escalável, com cobertura de testes robusta demonstrando a qualidade do código.

Viabilidade Econômica: O modelo de desenvolvimento utilizando tecnologias open-source e infraestrutura em nuvem garante custos operacionais competitivos, tornando a solução acessível ao público-alvo.

Viabilidade de Mercado: A demanda crescente por soluções integradas de gestão de eventos, especialmente no segmento de pequenos e médios organizadores, confirma o potencial de adoção da plataforma.

Sustentabilidade: A arquitetura modular e a documentação abrangente facilitam a manutenção evolutiva e a adaptação a futuras necessidades do mercado.

O projeto Eventario representa uma contribuição significativa para a modernização da gestão de eventos, oferecendo uma alternativa viável e acessível às soluções comerciais existentes, com potencial de impacto positivo na eficiência operacional e no crescimento sustentável do setor de eventos.

REFERÊNCIAS

PESSOA, Bruno Carlos et al. BANCO DE DADOS MONGODB VS BANCO DE DADOS SQL SERVER 2008. **RE3C-Revista Eletrônica Científica de Ciência da Computação**, v. 7, n. 1, 2012.

INCAU, Caio. **Vue. js: Construa aplicações incríveis**. Editora Casa do Código, 2017.

IVO, Andressa Aita; MARIN, Elizara Carolina; DE SOUZA, Lucas Machado. Gestão de eventos: orientações básicas para o contexto das universidades. **Kinesis**, v. 32, n. 2, 2014.

MALLEN, Cheryl; ADAMS, Lorne J. **Gestão de eventos esportivos, recreativos e turísticos: Dimensões teóricas e práticas**. Editora Manole, 2013.

CARMO, Klayver Ximenes. Um estudo comparativo entre tecnologias de back-end: Node. js, Django REST Framework e ASP. NET Core. 2023.