Faculdade de Tecnologia de Americana "Ministro Ralph Biasi"

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Bruno Ramos Negri

DESENVOLVIMENTO DE UM SISTEMA WEB PARA GESTÃO DE FESTAS EM UM BUFFET INFANTIL

Americana, SP

2025

Bruno Ramos Negri

DESENVOLVIMENTO DE UM SISTEMA WEB PARA GESTÃO DE FESTAS EM UM BUFFET INFANTIL

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas na área de concentração em laboratório de engenharia de software.

Orientador: Prof. Me. Jonas Bodê

Este trabalho corresponde à versão final do Trabalho de Conclusão de Curso apresentado por Bruno Ramos Negri e orientado Prof. Me. Jonas Bodê.

Americana, SP

2025

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana Ministro Ralph Biasi- CEETEPS Dados Internacionais de Catalogação-na-fonte

NEGRI, Bruno Ramos

Desenvolvimento de um sistema web para gestão de festas em um buffet infantil. / Bruno Ramos Negri – Americana, 2025.

59f.

Monografia (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana Ministro Ralph Biasi – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Ms. Jonas Bodê

1. Desenvolvimento de software 2. Java – linguagem de programação 3. Scrum – software de projetos. I. NEGRI, Bruno Ramos II. BODÊ, Jonas III. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana Ministro Ralph Biasi

CDU: 681.3.05
681.3.061Java
681.3.077Scrum

Elaborada pelo autor por meio de sistema automático gerador de ficha catalográfica da Fatec de Americana Ministro Ralph Biasi.

Bruno Ramos Negri

Desenvolvimento de um Sistema Web para Gestão de Festas em um Buffet Infantil

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas pelo Centro Paula Souza – FATEC Faculdade de Tecnologia de Americana Ministro Ralph Biasi.
Área de concentração: Análise e Desenvolvimento de Sistemas.

Americana, 2 de dezembro de 2025.

Banca Examinadora:

Jonas Bodê
Mestre
Fatec Americana "Ministro Ralph Biasi"

Antonio Alfredo Lacerda
Especialista
Fatec Americana "Ministro Ralph Biasi"

Paula da Fonte Sanches
Mestre
Fatec Americana "Ministro Ralph Biasi"

RESUMO

O presente texto demonstra o desenvolvimento de uma aplicação web para gestão de festas para um *buffet* infantil. Tendo em vista a forma defasada e ineficiente que *buffets* utilizam para administrar seus eventos, em muitos casos utilizando planilhas e papel para fazer o gerenciamento, esse trabalho foi criado como uma proposta de solução a essa demanda. Possui como objetivo acadêmico demonstrar a criação da aplicação seguindo a metodologia Scrum de forma adaptada para o uso individual. É possível encontrar o uso de tecnologias como *Spring Boot*, *Angular*, ambientes de implementação que a AWS possui, dentre eles serviços como EC2, S3 e RDS, ferramentas de testes unitários como *JUnit* 5, *JaCoCo*, *Mockito*, além de ferramentas de versionamento como o *Git*. A arquitetura *REST* foi escolhida para fazer a comunicação do *front-end* e *back-end*. O resultado adquirido foi uma aplicação robusta, com interface simples e feedbacks contínuos, possibilitando cadastro, pesquisa e exclusão de festas.

Palavras Chave: *Angular*; AWS; *Spring Boot*.

ABSTRACT

This text demonstrates the development of a web application for managing parties for a children's catering service. Given the outdated and inefficient way that catering services manage their events, in many cases using spreadsheets and paper, this work was created as a proposed solution to this demand. Its academic objective is to demonstrate the creation of the application following the Scrum methodology adapted for individual use. It uses technologies such as Spring Boot, Angular, AWS implementation environments, including services such as EC2, S3, and RDS, unit testing tools such as JUnit 5, JaCoCo, Mockito, and versioning tools such as Git. The REST architecture was chosen for front-end and back-end communication. The result was a robust application with a simple interface and continuous feedback, enabling registration, search, and deletion of parties.

Keywords: *Angular;* AWS; *Spring Boot.*

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API - Application Programming Interface
AWS - Amazon Web Services
DER - Diagrama Entidade-Relacionamento
DoD - Definition of Done
DTO - Data Transfer Object
EC2 - Elastic Compute Cloud
H2 - H2 Database Engine
HTTP - HyperText Transfer Protocol
IDE - Integrated Development Environment
JSON - JavaScript Object Notation
JPA - Java Persistence API
JWT - JSON Web Token
ORM - Object-Relational Mapping
REST - Representational State Transfer
RF - Requisito Funcional
RNF - Requisito Não Funcional
RDS - Relational Database Service
S3 - Simple Storage Service
SIGA - Sistema Integrado de Gestão Acadêmica
SQL - Structured Query Language

SUMÁRIO

1 INTRODUÇÃO

Primeiramente, serão apresentados o contexto em que o trabalho está inserido e as causas que impulsionaram o desenvolvimento de uma solução tecnológica. Em seguida, serão abordados, respectivamente, os principais objetivos do trabalho e, de forma sucinta, a metodologia utilizada para o seu desenvolvimento.

1.1 Contextualização do tema

O avanço tecnológico tem influenciado áreas de diversos setores, dentre eles, a área de eventos tem sido modificada e melhorada em seu aspecto de gestão. Em um nicho mais específico de eventos, estão presentes os *buffets* de festas infantis que muitas vezes não possuem uma metodologia segura e eficiente para a organização de seus eventos, utilizando, assim, planilhas de Excel e, nos casos mais limitados, o papel.

1.2 Problema da pesquisa

Identificar meios de auxiliar *buffets* infantis no gerenciamento de festas, possibilitando o desuso de planilhas e papel e, com isso, propondo uma opção mais confiável, persistente e de fácil uso para a organização os eventos.

1.3 Justificativa

A escolha desse tema ocorreu pela falta de soluções eficazes voltadas para gestão de eventos em *buffets* infantis. Um sistema web customizado seria uma opção viável, pois oferece pesquisas ágeis, persistência de dados em nuvem, e um padrão estruturado para a organização.

Além disso, esse projeto contribui academicamente por demonstrar a criação de um sistema web *full stack*, utilizando tecnologias atuais no mercado, bem como metodologias ágeis, testes unitários que asseguram a qualidade do sistema e, por último, sua execução em produção, utilizando serviços web para realizar o *deploy*.

1.4 Objetivo Geral

Desenvolver uma aplicação web *full stack* para auxiliar *buffets* de festas infantis em seu gerenciamento de eventos, contemplando metodologias de produção de software, análises e o *deploy* da aplicação.

1.5 Objetivos específicos

- Analisar as necessidades de gerenciamento de eventos na área de festas infantis;
- Modelar a base de dados para suportar o sistema;
- Desenvolver o *back-end* usando *Spring Boot* 3.5.0, *Java* 21 e JWT para a segurança;
- Desenvolver o *front-end* com *Angular* 19 e *TailwindCSS* para a interface gráfica;
- Realizar testes unitários e de integração no *back-end* com *JUnit* 5, *Mockito* e *JaCoCo*;
- Utilizar banco de dados *MySQL* para persistência dos dados e banco H2 para a fase de testes;
- Aplicar a metodologia ágil Scrum para o gerenciamento do desenvolvimento;
- Utilizar versionamento de código com a ferramenta *Git*;
- Efetuar o *deploy* da aplicação em um ambiente da AWS
-

1.6 Metodologia de desenvolvimento

O desenvolvimento será conduzido seguindo a metodologia Scrum, de forma adaptada ao cenário de apenas uma pessoa responsável por todas as funções, proporcionando a visualização das entregas parciais da aplicação, resultando, assim, em uma visão mais precisa do rumo que o projeto, como um todo, está tomando. As ferramentas que serão utilizadas foram escolhidas por sua alta relevância no mercado atualmente, desde o uso de frameworks como *Angular* e *Spring Boot* até os testes unitários e a plataforma de serviços web, neste caso a AWS.

Esse projeto será documentado em todas as suas partes, contemplando análises de requisitos, desenvolvimento do *back-end* e *front-end*, implementação de testes unitários e de integração e, finalizando, o *deploy* em ambiente de nuvem da AWS.

2 REFERENCIAL TEÓRICO

Esta seção apresenta os pilares sobre os quais o trabalho foi desenvolvido. O uso de autores, sites e metodologias poderá ser facilmente observado.

2.1 Sistemas de Informação aplicados à gestão de eventos

Os sistemas de informação são mecanismos que nos proporcionam melhor organização de dados, além de sua alta capacidade de processamento e da maestria em substituir tarefas feitas manualmente. Segundo Laudon e Laudon (2014), eles contribuem para o controle das informações e, consequentemente, afetam de maneira positiva a eficiência da organização ou empresa. Desse modo, ao centralizar dados de clientes e eventos, é possível reduzir a taxa de erros que seriam cometidos em um processo manual.

2.2 Importância da informatização em buffets e pequenas empresas

A informatização é um recurso de extrema relevância não só para grandes empresas, mas também para pequenas, como é o caso de *buffets* de festas. Muitas dessas pequenas empresas recorrem ao uso de planilhas ou anotações em papel, o que pode ocasionar possíveis erros e perdas de informações que poderiam ser evitadas. Nesse caso, implementar uma tecnologia, como, por exemplo, um sistema web, pode possibilitar melhor desempenho, além de acesso às informações de forma remota.

2.3 Metodologia de Desenvolvimento de Software

Ao se desenvolver um software, é de suma importância seguir um padrão em sua gestão, seja para alcançar resultados com maior eficiência e rapidez ou simplesmente para garantir organização. A seguir, será apresentada a metodologia escolhida para o respectivo trabalho.

2.3.1 Metodologia Ágil e o Scrum

As metodologias ágeis foram criadas com o intuito de acompanhar a velocidade e a alta demanda que o mercado exigia, pois, as metodologias tradicionais de desenvolvimento não se mostravam eficazes o suficiente. Uma delas é o Scrum. Segundo Schwaber e Sutherland (2020), “o Scrum é um framework leve que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos”.

2.4 Arquitetura de Sistemas Web

Sendo a metodologia um padrão a ser seguido na gestão, a arquitetura padronizada adequa-se à produção do software.

2.4.1 Conceito de aplicação web

A aplicação web pode ser definida como um sistema acessado por meio de navegadores, funcionando em um modelo cliente-servidor, onde o navegador (cliente) faz requisições ao servidor para obter acesso aos *endpoints*. Assim, possibilita o acesso remoto, sem a necessidade de instalação local.

2.4.2 Arquitetura cliente-servidor

Um dos modelos de arquitetura mais utilizados para aplicações web é o modelo cliente-servidor, no qual o cliente faz requisições ao servidor, que captura a solicitação e retorna o valor requisitado. O estilo arquitetural Representational State Transfer (REST), proposto por Fielding (2000), é amplamente implementado para a comunicação entre *front-end* e *back-end*.

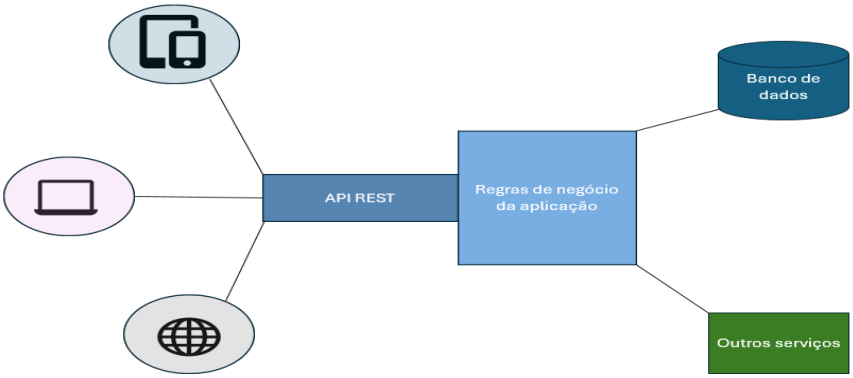
2.4.3 Arquitetura REST

A arquitetura REST pode ser visualizada na Figura 1. Ela foi proposta por Roy Fielding em sua tese de doutorado. Segundo Fielding:

> O REST fornece um conjunto de restrições arquitetônicas que, quando aplicadas como um todo, enfatizam a escalabilidade do componente, a generalidade das interfaces, a implantação de interações independentes dos

> componentes e componentes intermediários para reduzir a latência da interação, reforçar a segurança e encapsular sistemas legados (2000, p. 75, tradução do autor).

Figura 1 - Fluxograma do modelo REST



Fonte: Produção própria (2025).

2.4.4 Modelos full Stack

No modelo de desenvolvimento *full stack*, inclui tanto o *front-end* quanto o *back-end*. Esse tipo de desenvolvimento permite maior flexibilidade de integração entre as regras de negócio e a interface gráfica, possibilitando que as entregas parciais sejam visuais e funcionais, pois são desenvolvidas em conjunto.

2.5 Tecnologias Utilizadas

Esta seção abordará os softwares e as linguagens de programação utilizadas para a criação do projeto.

2.5.1 Java

O *Java* é uma linguagem de programação orientada a objetos, muito utilizada em softwares corporativos e aplicações web por conta de sua portabilidade e robustez. A Oracle disponibiliza documentação oficial sobre o *Java* (ORACLE, 2025).

2.5.2 Spring Boot

O *Spring Boot* é um framework em *Java* que tem como objetivo acelerar o desenvolvimento e reduzir complexidade de configurações, de acordo com sua documentação (SPRING, 2025). É amplamente utilizada para a criação de microsserviços e *back-end* de sistemas web.

2.5.3 Angular

O *Angular* é um framework baseado em TypeScript. De acordo com sua documentação (*ANGULAR*, 2025), ele permite a criação de interfaces escaláveis, organizadas em módulos e componentes, possibilitando assim a reutilização deles com maior facilidade.

2.5.4 Banco de Dados MySQL e H2

O *MySQL* é um sistema de gerenciamento de banco de dados relacional, enquanto o H2 Database Engine de acordo com sua documentação possui as seguintes características: API JDBC de código aberto e muito rápida; modos incorporado e de servidor; bancos de dados em memória; aplicativo de console baseado em navegador (H2, 2025).

2.5.5 JUnit e Testes Automatizados

O *JUnit* 5 é um framework que tem como objetivo fornecer ferramentas para testes unitários em aplicações *Java* (*JUNIT*, 2025). Combinando essa ferramenta com o *Mockito*, para simular dependências, e o *JaCoCo*, para medir a cobertura dos testes feitos na aplicação, resulta em um robusto arsenal para deixar a aplicação menos propensa a erros.

2.5.6 AWS para deploy em nuvem

A Amazon Web Service (AWS) fornece uma variedade enorme de serviços web, facilitando o desenvolvimento e a escalabilidade de aplicações. Dentre esses serviços, destacam-se o Amazon *Elastic Compute Cloud* (EC2), responsável por

permitir hospedar aplicações em máquinas virtuais configuráveis, o Amazon *Relational Database Service* (RDS), que oferece um banco de dados relacional gerenciado, permitindo a automatização de processos. Amazon Simple Storage Service (S3), utilizado para o armazenamento e disponibilização de arquivos estáticos, como o *front-end* da aplicação (AMAZON WEB SERVICES, 2025).

2.5.7 Postman

O Postman é uma ferramenta com presença marcante no ambiente de desenvolvimento de APIs. Ele permite o envio de requisições HyperText Transfer Protocol (HTTP), possibilitando testar o funcionamento dos *endpoints* e validar as respostas do servidor.

2.5.8 Git

Git é uma ferramenta de que permite o controle de versionamento de código, acompanhar o histórico de desenvolvimento e integrar diferentes *branchs* (*GIT*, 2025).

2.6 Observações sobre o referencial teórico

O referencial teórico apresentado permitiu estabelecer as bases conceituais necessárias para o desenvolvimento deste projeto, abordando desde os fundamentos de sistemas de informação até modelos arquiteturais modernos, metodologias ágeis e tecnologias utilizadas na construção da solução.

A revisão demonstrou que a informatização de processos em buffets infantis é uma demanda concreta e que frameworks, ferramentas e boas práticas consolidadas no mercado fornecem suporte adequado para a criação de aplicações robustas, seguras e escaláveis.

Dessa forma, os conceitos explorados neste capítulo servem como alicerce para as etapas metodológicas e práticas apresentadas no capítulo seguinte, orientando decisões de desenvolvimento e garantindo alinhamento com padrões técnicos atuais.

3 METODOLOGIA

A presente seção tem o intuito de detalhar o uso das metodologias escolhidas para a documentação do trabalho e para o desenvolvimento da aplicação.

3.1 Tipo de pesquisa

Esse trabalho é uma monografia com foco no desenvolvimento de uma aplicação web, voltada para o planejamento, desenvolvimento e *deploy* da aplicação. Aborda processos como análise de requisitos, modelagem do banco, modelagem de classes, testes para integridade do produto e implementação na nuvem.

3.2 Metodologia de desenvolvimento (Scrum adaptado)

O desenvolvimento seguiu princípios da metodologia ágil Scrum, aplicada no contexto em que se encontra apenas um indivíduo construindo o projeto, visando sempre os pilares do framework, que são, de acordo com o Scrum Guide (SCRUM GUIDES, 2020): transparência, inspeção e adaptação.

A transparência possibilita acompanhar claramente o direcionamento do projeto. As inspeções permitem um olhar atento, analisando e avaliando os resultados obtidos. Por fim, a adaptação possibilita fazer reajustes para a evolução do projeto.

Neste projeto foram adotadas quatro *sprints*, com duração de quatro semanas cada. Ou seja, o desenvolvimento completo durou cerca de quatro meses. Cada *sprint* foi planejada para que cada parte da estrutura fosse desenvolvida em sua respectiva ordem de importância, que são representadas nos Quadros 1, 2, 3, 4.

Quadro 1 - Atividades sprint um

ID	Atividades
1	Definir escopo e personas
2	Levantar requisitos funcionais (RF)
3	Levantar requisitos não funcionais (RNF)
4	Priorizar backlog inicial
5	Diagramar Casos de Uso
6	Revisar/validar Casos de Uso
7	Diagramar Classes
8	Revisar/validar Classes
9	Modelar DER
10	Revisar/validar DER
11	Configurar repositório Git e convenções
12	Planejar sprints e critérios de pronto (DoD)

Fonte: Produção própria (2025)

Quadro 2 - Atividades sprint dois

ID	**Atividades**
13	Criar projeto *Spring Boot* 3.5 / *Java* 21
14	Configurar dependências (JPA, Security, JWT, Validação)
15	Mapear entidades (Aniversariante, Festa, Convidado, Admin)
16	Criar DTOs iniciais
17	Criar repositories JPA
18	Implementar services base
19	Implementar endpoints de Festa (CRUD básico)
20	Configurar H2 e properties
21	Implementar JWT (geração/validação)
22	Implementar login do admin
23	Políticas de validação (Bean Validation)
24	Refatorações e revisão de regras

Fonte: Produção própria (2025)

Quadro 3 - Atividades sprint três

ID	Atividades
25	Criar projeto *Angular* 19
26	Montar layout (header/footer fixos)
27	Configurar TailwindCSS
28	Serviço HTTP e interceptors
29	Página Login + integração JWT
30	Página Cadastro de Festa (form + validações)
31	Integração Cadastro
32	Página Pesquisa de Festas (filtros e data)
33	Integração Pesquisa
34	Página Atualização de Festa
35	Integração Atualização
36	Página Perfil Admin (alterar nome e senha)

Fonte: Produção própria (2025)

Quadro 4 - Atividades sprint quatro

ID	Atividades
37	Migrar de H2 para *MySQL*
38	Ajustar datas,horas e scripts
39	Criar testes unitários (*JUnit* 5)
40	Implementar dependências com *Mockito*
41	Configurar *JaCoCo*
42	Gerar relatório de cobertura
43	Testes de integração essenciais
44	Preparar AWS EC2 (*back-end*)
45	Preparar AWS RDS (*MySQL*)
46	Preparar AWS S3 (*front-end*)

Fonte: Produção própria (2025)

A Tabela 1 exibe o acompanhamento das entregas semanais. A coluna “Linha Base de Entrega” representa o cenário uniforme para as entregas, enquanto a coluna “Atividades Restantes” indica a quantidade de atividades que ainda restam entregar.

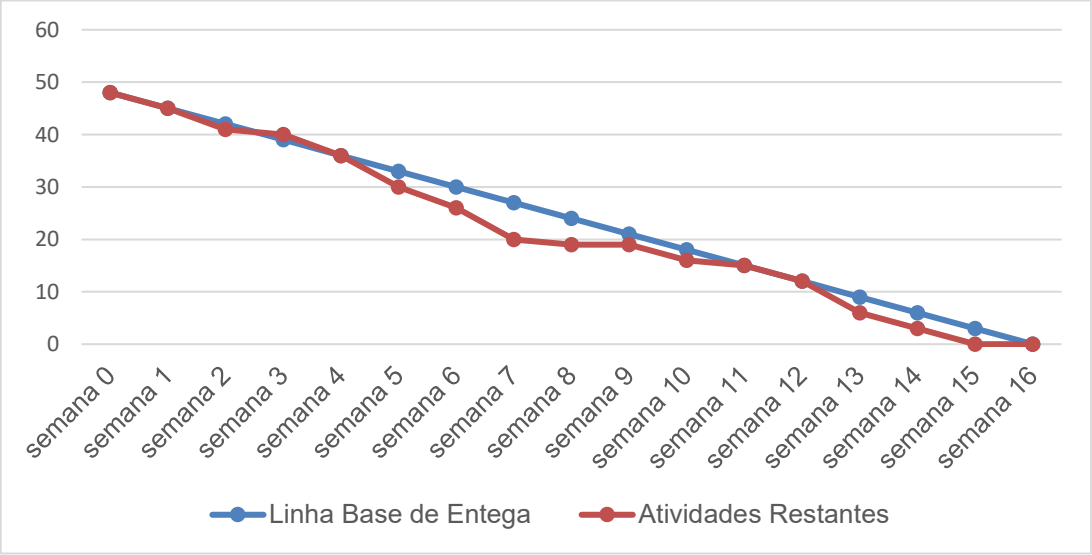
Tabela 1 - Acompanhamento de entregas

Atividades Propostas	Semana	Linha Base de Entrega	Atividades Restantes
-	semana 0	48	48
1 até 3	semana 1	45	45
4 até 6	semana 2	42	41
7 até 9	semana 3	39	40
10 até12	semana 4	36	36
13 até 15	semana 5	33	30
16 até 18	semana 6	30	26
19 até21	semana 7	27	20
22 até 24	semana 8	24	19
25 até 27	semana 9	21	19
28 até 30	semana 10	18	16
31 até 33	semana 11	15	15
34 até 36	semana 12	12	12
37 até 39	semana 13	9	6
40 até 42	semana 14	6	3
43 até 45	semana 15	3	0
46	semana 16	0	0

Fonte: Produção própria (2025)

Conforme ilustrado na Figura 2, o gráfico utiliza como base os dados da Tabela 1, permitindo, assim, uma melhor visualização das entregas das atividades.

Figura 2 - Gráfico Burndown das entregas de atividades em todo ciclo

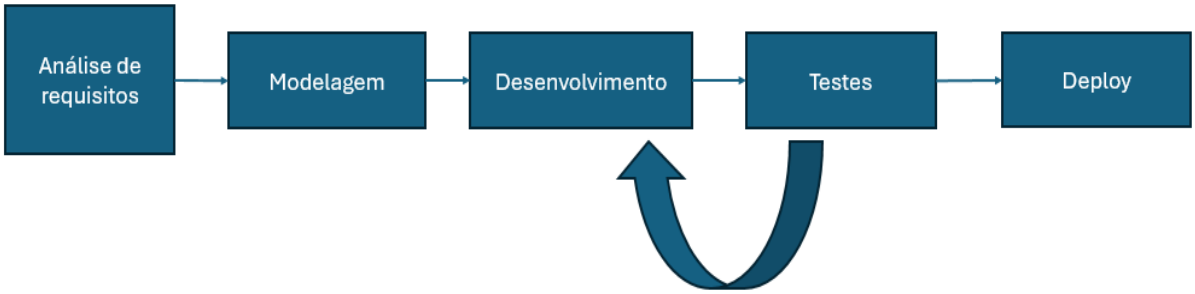


Fonte: Produção própria (2025)

3.3 Etapas de desenvolvimento

As etapas de desenvolvimento serão apresentadas em forma de fluxograma na Figura 3 para melhor visualização.

Figura 3 - Fluxograma das etapas de desenvolvimento



Fonte: Produção própria (2025)

3.3.1 Levantamento de requisitos

Para a construção de um sistema, o desenvolvedor, antes mesmo de começar a produção, necessita de uma base teórica sobre a aplicação. Um dos métodos mais utilizados na engenharia de software é o levantamento de requisitos, por meio do qual é possível ter um panorama das necessidades relevantes e superficiais que a aplicação deve suprir.

A tabela foi desenvolvida utilizando o Word e será apresentada no Capítulo 4, na seção 4.1 *Levantamento de requisitos*.

3.3.2 Modelagem do sistema

A modelagem do sistema é outra etapa essencial para o desenvolvimento de uma aplicação. Nela, é possível visualizar a interação entre o usuário e a aplicação, o comportamento entre as classes e seus respectivos métodos, além da estruturação do banco de dados em sua forma lógica.

Para atender a essa demanda, foram utilizados neste projeto o Diagrama de Casos de Uso, o Diagrama de Classes e o Diagrama Entidade-Relacionamento (DER).

O Diagrama de Casos de Uso demonstra a interação do usuário (ator) com o sistema, mostrando visualmente os processos disponíveis para cada tipo de ator. Além disso, evidencia o comportamento esperado para cada requisição feita ao sistema.

O Diagrama de Classes representa a estrutura interna da aplicação, mostrando suas classes, métodos, atributos e relacionamentos. Esse modelo é essencial para a construção de um projeto orientado a objetos, servindo como base para a implementação do *back-end*.

O Diagrama Entidade-Relacionamento (DER) representa a arquitetura lógica do banco de dados em sua representação gráfica, mostrando o relacionamento entre as tabelas e os atributos de cada entidade.

Os diagramas foram criados utilizando a ferramenta gratuita *Lucidchart*. e, para o modelo, foi utilizado o *MySQL Workbench*. Serão apresentados no Capítulo 4, na seção 4.2 *Modelagem do sistema.*

3.3.3 Implementação

A implementação do sistema foi dividida em duas partes: *back-end* e *front-end*.

Para o *back-end*, foi utilizado *Spring Boot* 3.5.0 com *Java* 21, além da implementação de segurança com JSON Web Token (JWT) para autenticação do usuário administrador. Foi utilizada a arquitetura REST para fazer a comunicação com o *front-end*, desenvolvido com *Angular* 19 e *TailwindCSS* para o design responsivo.

3.3.4 Testes e validações

Os testes foram executados com o auxílio do *framework JUnit* 5, garantindo um funcionamento robusto dos métodos em suas respectivas execuções.

Para o isolamento das camadas de serviço, foi utilizado o *Mockito*, permitindo simular as dependências que os métodos exigiam. Finalizando o fluxo de testes, o

JaCoCo foi responsável pela cobertura, gerando relatórios automáticos de desempenho.

3.3.5 Deploy

A implementação do software foi executada com os serviços gratuitos que a AWS fornece nos seis primeiros meses de uso de uma conta. Para o *back-end* desenvolvido com *Spring Boot*, foi feito o *deploy* usando o serviço Amazon Elastic Compute Cloud (EC2), viabilizando o gerenciamento de instâncias de máquinas virtuais.

O banco de dados *MySQL* foi implementado usando o serviço Amazon Relational Database Service (RDS), permitindo o gerenciamento de *backups* e oferecendo uma segurança robusta.

Já o *front-end* desenvolvido em *Angular* foi implementado no serviço Amazon Simple Storage Service (S3), cuja função é fornecer um ambiente para arquivos estáticos e prover baixa latência e alta disponibilidade.

3.4 Ferramentas de apoio

O Quadro 5 demonstra as ferramentas que foram utilizadas na construção de modelos, planilhas e diagramas, além de recursos essenciais para versionamento, testes e codificação do sistema.

Essas ferramentas desempenharam um papel fundamental ao longo do desenvolvimento, pois proporcionaram organização, controle das atividades, modelagem visual das estruturas do sistema e suporte técnico para a implementação do back-end e front-end.

A utilização integrada desses recursos permitiu maior produtividade, rastreabilidade das alterações e melhor visualização do progresso do projeto.

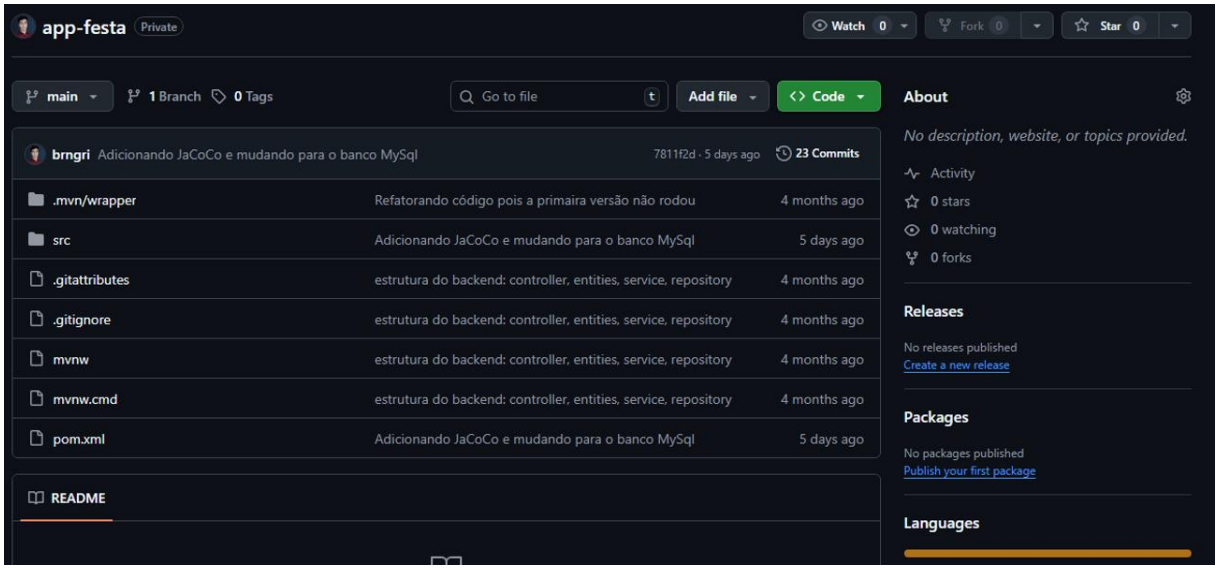
Quadro 5 - Ferramentas utilizadas no desenvolvimento

Ferramenta	Finalidade
VS Code	IDE utilizada para o desenvolvimento do *back-end* e *front-end.*
Git	Controle de versionamento do código-fonte, garantindo o rastreamento e gerenciar alterações no projeto.
GitHub	Repositório remoto para armazenamento seguro e sincronização do código desenvolvido.
Postman	Testes e validação das requisições HTTP e *endpoints* da aplicação web.
MySQL Workbench	Modelagem e gerenciamento do banco de dados relacional *MySQL*. Criação do modelo entidade-relacionamento (DER).
Lucidchart	Criação dos diagramas de caso de uso e classes.
Excel	Organização das tarefas, cronogramas e acompanhamento do progresso das sprints.

Fonte: Produção própria (2025)

Conforme pode ser observado na Figura 4, trata-se de uma captura de tela do repositório onde está armazenado os versionamentos do *back-end.*

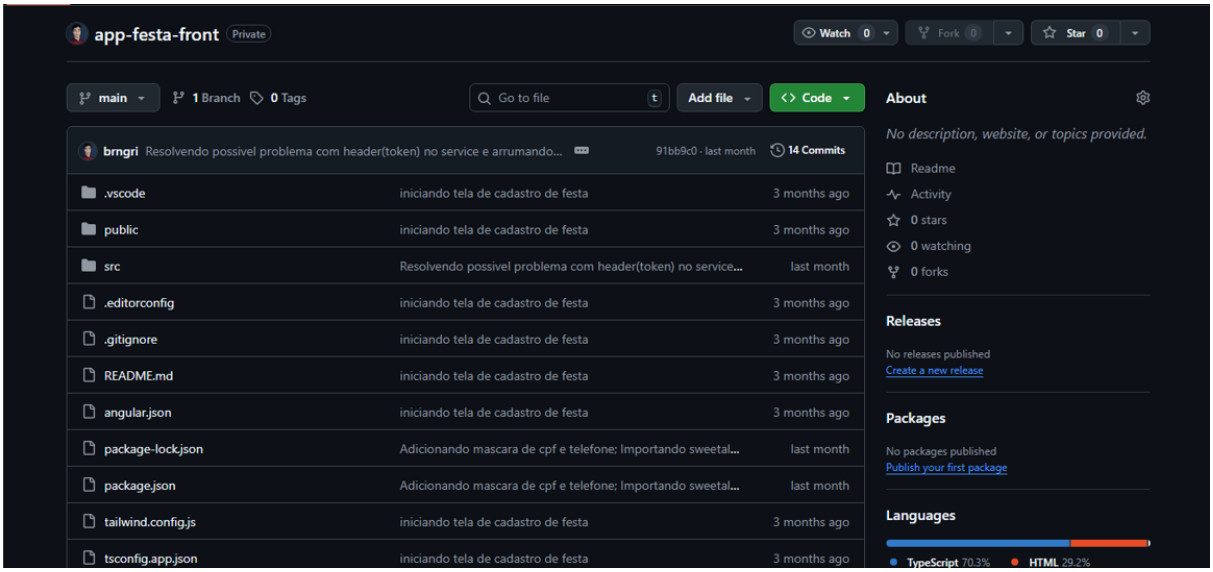
Figura 4 - Repositório GitHub back-end



Fonte: Produção própria (2025)

Como ilustrado na Figura 5, a imagem corresponde a uma captura de tela do repositório onde está armazenado os versionamentos *front-end.*

Figura 5 - Repositório GitHub front-end



Fonte: Produção própria (2025)

3.5 Observações sobre as metodologias

A metodologia apresentada neste capítulo permitiu estruturar de forma clara e organizada todas as etapas do desenvolvimento da aplicação, desde a definição dos requisitos até a implementação e validação final do sistema.

A utilização de um Scrum adaptado possibilitou um acompanhamento contínuo da evolução do projeto, garantindo entregas incrementais, inspeções recorrentes e ajustes necessários ao longo das sprints.

Além disso, o conjunto de ferramentas adotadas forneceu suporte técnico e estrutural indispensável para o desenvolvimento eficiente e alinhado às boas práticas de engenharia de software.

Dessa forma, os procedimentos metodológicos aqui descritos servem como base para a compreensão das decisões tomadas e fundamentam o capítulo seguinte, no qual são apresentados os detalhes da construção do sistema.

4 DESENVOLVIMENTO DO SISTEMA

O desenvolvimento da aplicação será abordado nesta seção de forma mais visual, contendo análises baseadas em conceitos de engenharia de software, diagramas e ferramentas disponibilizadas pelas linguagens escolhidas.

4.1 Requisitos Funcionais

O Quadro 6 apresenta os requisitos funcionais propostos para a aplicação.

Quadro 6 - Requisitos funcionais

ID	Requisito	Descrição Detalhada	Prioridade
RF01	Cadastro de Festa	Permitir cadastrar nova festa com data, dados do responsável e horas de início e fim	Alta
RF02	Validação de Data	A data informada não pode ser anterior à data atual.	Alta
RF03	Limite de Festas por Dia	Máximo de 2 festas por dia.	Alta
RF04	Horário de Início	Festas devem iniciar a partir das 12:00.	Alta
RF05	Horário de Término	Festas devem terminar até 23:30.	Alta
RF06	Duração da Festa	Duração entre 3 e 4 horas.	Alta
RF07	Intervalo entre Festas	Nova festa deve ter pelo menos 1h de intervalo da anterior.	Alta
RF08	Cadastro de Admin Padrão	Criação automática de usuário admin padrão com senha criptografada.	Alta
RF09	Login Seguro	Autenticação via *endpoint* /auth/login com JWT.	Alta
RF10	Listagem de Festas	Listar todas as festas e permitir filtrar por data ou CPF.	Alta
RF11	Atualização de Admin	Permitir alterar informações do administrador autenticado.	Média
RF12	Exclusão de Festas	Permitir excluir festas cadastradas, respeitando regras.	Média

Fonte: Produção própria (2025)

4.2 Requisitos Não Funcionais

O Quadro 7 apresenta os requisitos não funcionais propostos para a aplicação.

Quadro 7 - Requisitos não funcionais

ID	Requisito	Descrição Detalhada	Prioridade
RNF01	Banco de Dados	Armazenar informações no *MySQL* (RDS AWS) e suportar H2 para testes.	Baixa
RNF02	Segurança	Rotas protegidas com JWT, exceto login.	Baixa
RNF03	Interface	Usar *TailWindCss* para interface gráfica	Baixa
RNF04	Disponibilidade	*Back-end* hospedado no EC2 AWS e *front-end* no S3 AWS.	Baixa
RNF05	Usabilidade	Mensagens de erro claras e humanizadas.	Baixa
RNF06	Portabilidade	Executável em ambiente *Linux* (EC2) com *Java* 21.	Baixa

Fonte: Produção própria (2025)

4.3 Diagrama de Casos de Uso

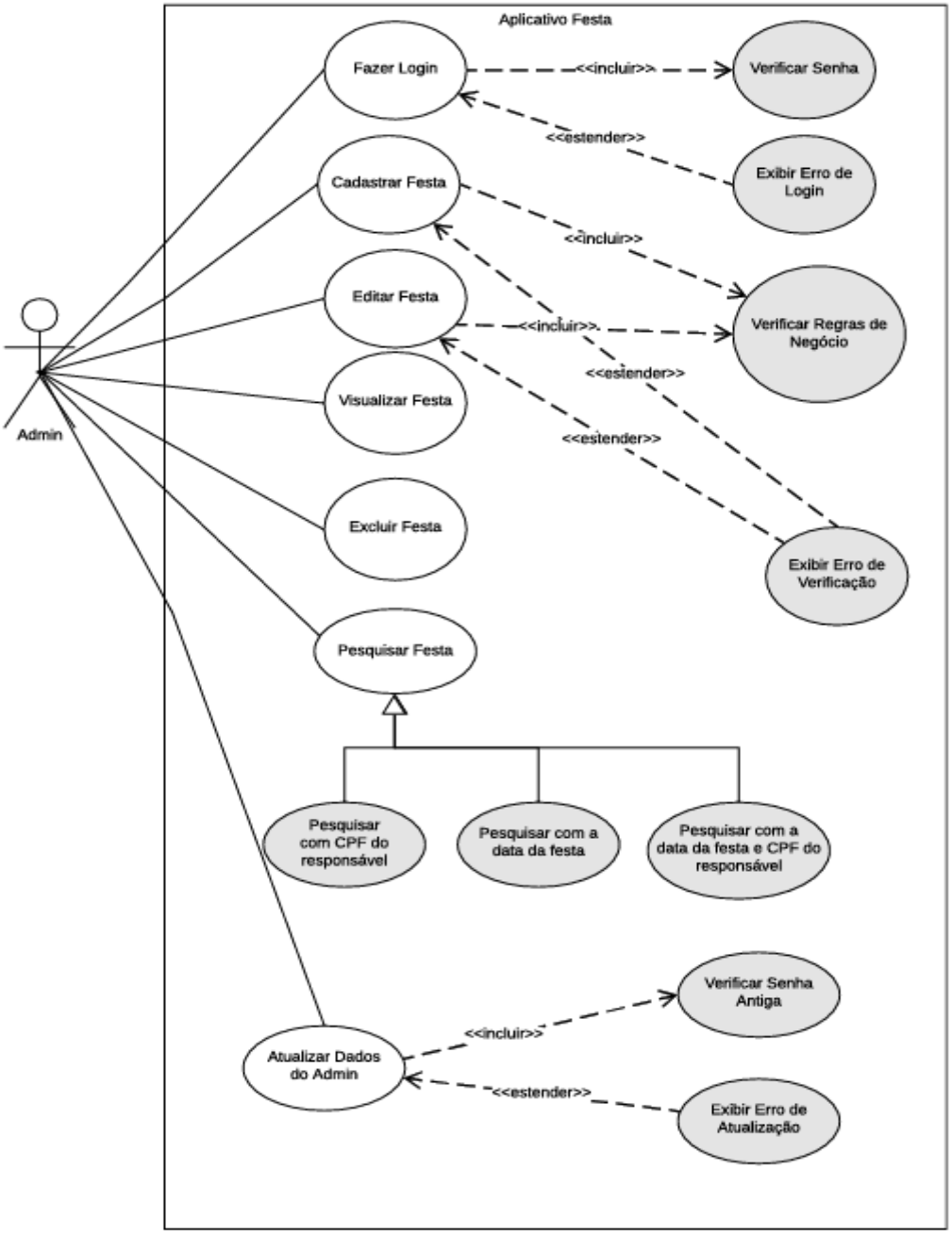
Observando a Figura 6, é possível identificar de forma estruturada as funcionalidades disponibilizadas ao usuário e suas respectivas interações com o sistema.

O diagrama de caso de uso permite visualizar, de maneira abstrata e sistemática, os comportamentos esperados para cada ator, evidenciando limites operacionais, responsabilidades e fluxos importantes para o funcionamento da aplicação.

Essa representação contribui para o alinhamento entre requisitos funcionais e modelagem, assegurando que as ações descritas na documentação estejam coerentes com as necessidades levantadas no processo de engenharia de software.

Assim, o modelo apresentado desempenha papel fundamental na validação conceitual do sistema, servindo como base para etapas posteriores de implementação e testes.

Figura 6 - Diagrama de caso de uso

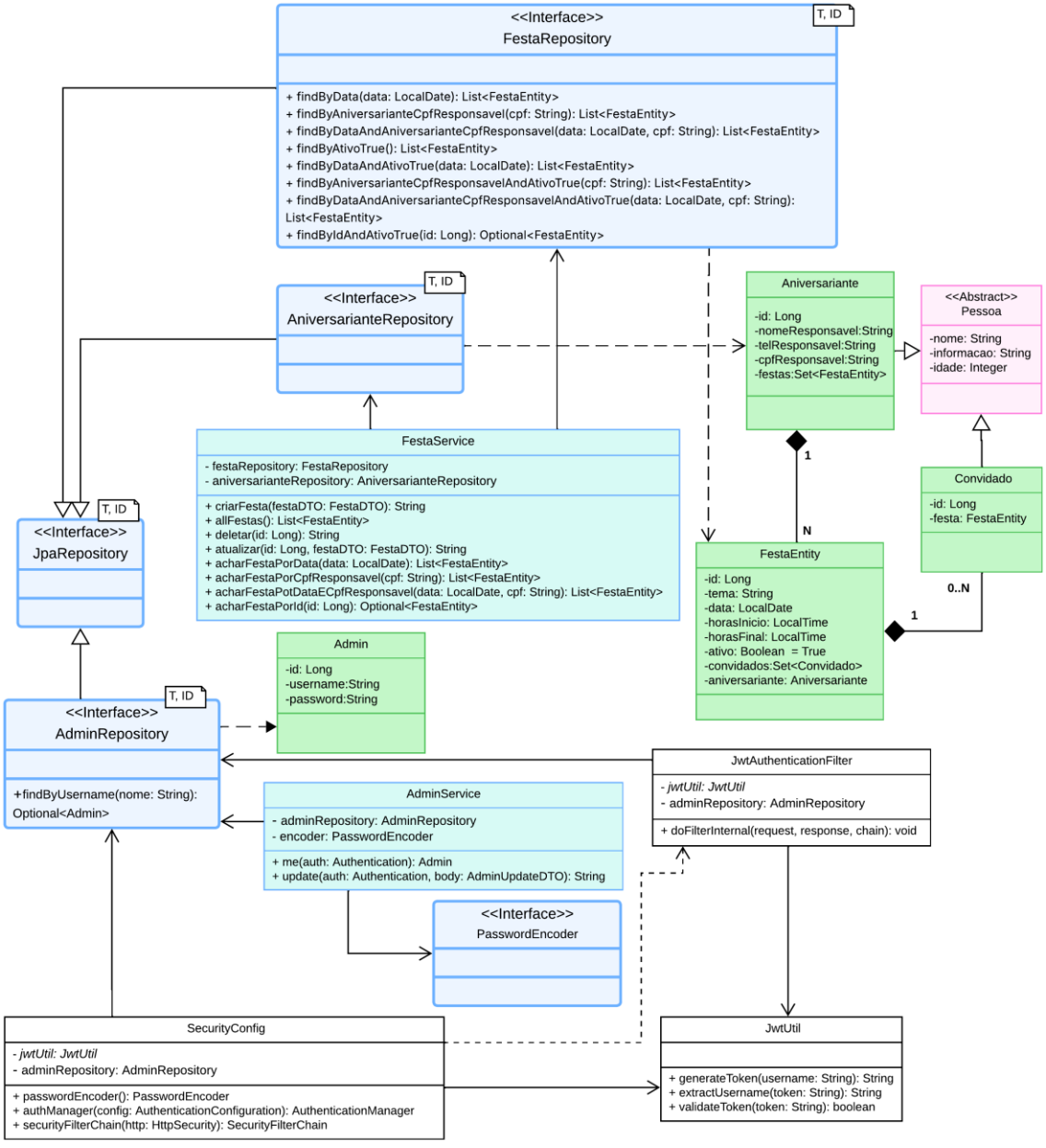


Fonte: Produção própria (2025)

4.4 Diagrama de Classe

Pode-se observar na Figura 7 as classes com seus atributos e métodos, cada um possuindo seu tipo de privacidade. As setas sólidas representam associação, já as pontilhadas representam dependência. Optou-se por não incluir os DTOs no diagrama para facilitar a leitura da estrutura principal do projeto.

Figura 7 - Diagrama de classe



Fonte: Produção própria (2025)

É de grande importância observar os relacionamentos de composição apresentados entre “FestaEntity”, “Aniversariante” e “Convidados”. Na qual a exclusão de um componente resulta, obrigatoriamente, na exclusão do outro.

4.5 DER

O banco de dados foi criado de forma automática pelo *Hibernate,* esse assunto será tratado com mais cautela adiante. Para a criação do DER foi utilizado a ferramenta de engenharia reversa disponibilizada pelo *MySQL Workbench.*

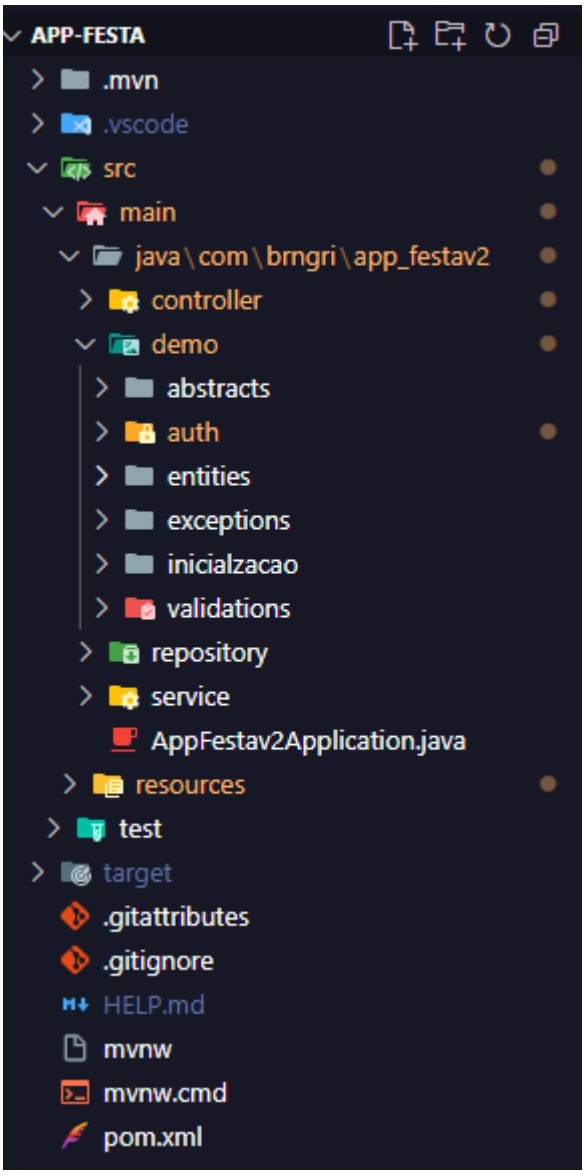
Figura 8 - Diagrama de entidade-relacionamento

Fonte: Produção própria (2025)

4.6 Estrutura de pastas

Conforme apresentado na Figura 9, observa-se a estrutura de pastas em que o *back-end* foi organizado

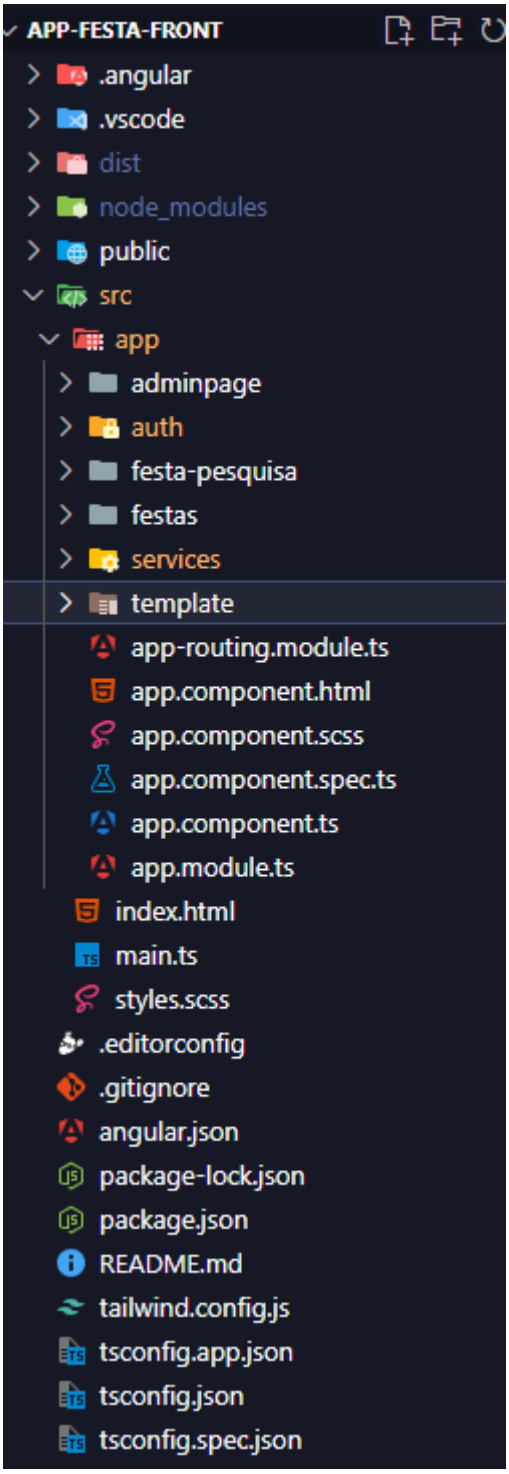
Figura 9 - Estrutura de pastas back-end



Fonte: Produção própria (2025)

A Figura 10 exibe a estrutura de pasta em que o *front-end* foi organizado.

Figura 10 - Estrutura de pastas front-end

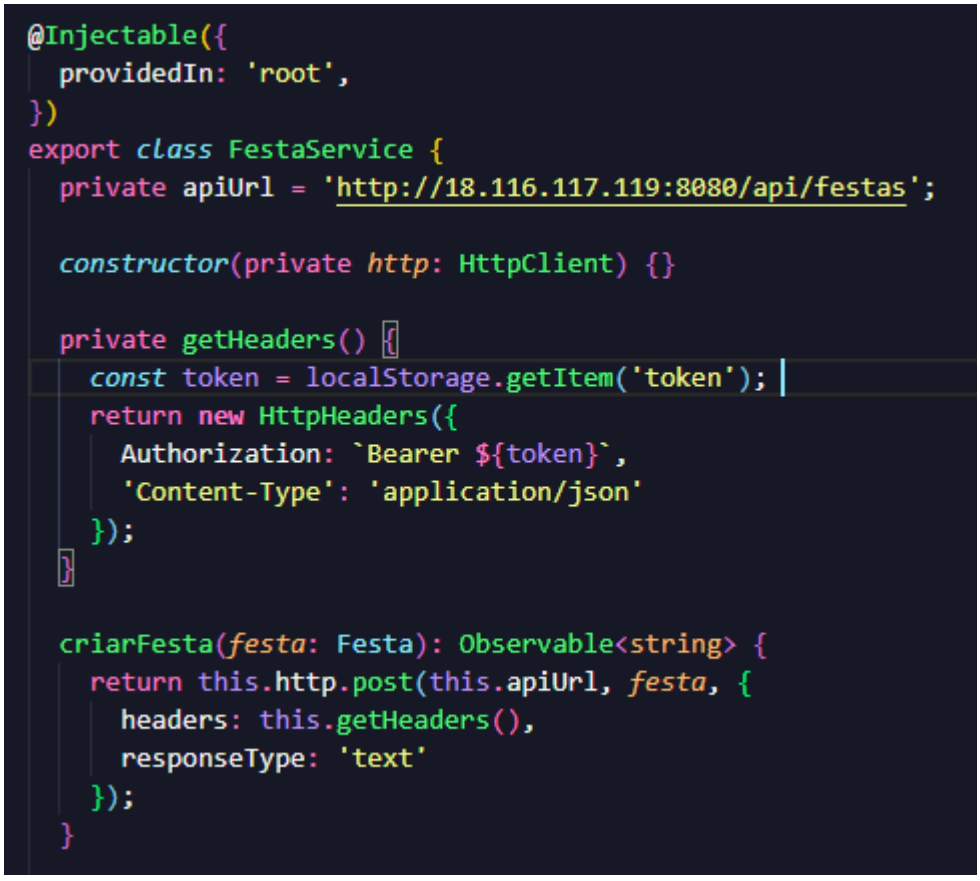


Fonte: Produção própria (2025)

4.7 Conexão com o back-end

A Figura 11 exibe como foi feita a conexão do serviço de criar festa no *front-end* com o serviço de criar festa do *back-end*, fazendo uma requisição para o *endpoint* encontrado no caminho ‘http://18.116.117.119:8080/api/festas’.

Figura 11 - Conexão do front-end com o back-end



```
@Injectable({
  providedIn: 'root',
})
export class FestaService {
  private apiUrl = 'http://18.116.117.119:8080/api/festas';

  constructor(private http: HttpClient) {}

  private getHeaders() {
    const token = localStorage.getItem('token');
    return new HttpHeaders({
      Authorization: `Bearer ${token}`,
      'Content-Type': 'application/json'
    });
  }

  criarFesta(festa: Festa): Observable<string> {
    return this.http.post(this.apiUrl, festa, {
      headers: this.getHeaders(),
      responseType: 'text'
    });
  }
```

Fonte: Produção própria (2025)

4.8 Banco de dados

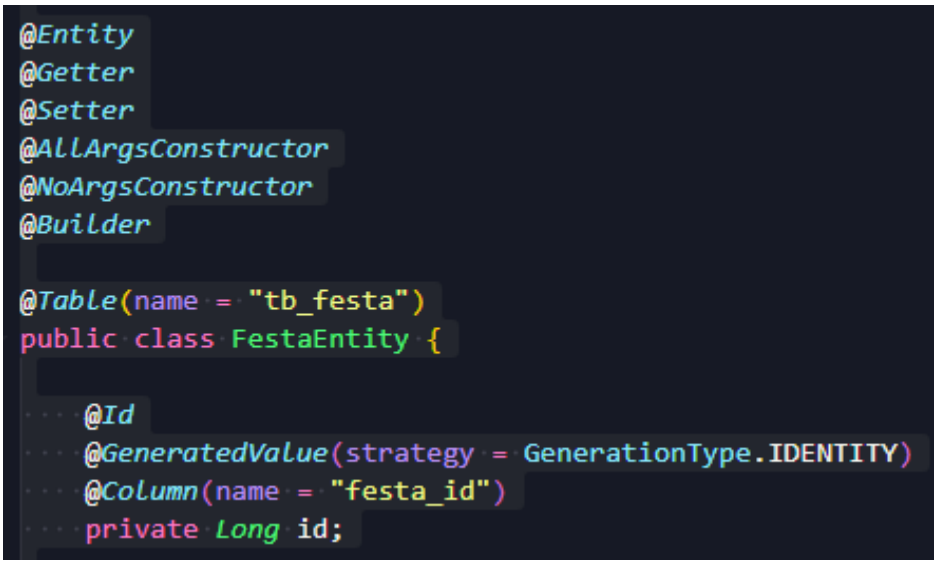
A criação do banco de dados foi feita de forma automatizada através das assinaturas disponibilizadas pelo *Java* Persistence API (JPA) e da execução do *Hibernate*.

O JPA disponibiliza um conjunto de assinaturas para o *Java* mapear as classes em tabelas no banco de dados. Ele, sozinho, não executa nada. Dentre as assinaturas usadas estão: @Entity, @Id, @GeneratedValue, @OneToMany e @ManyToOne.

Conforme mostrado na Figura 12 é possível observar essas assinaturas sendo usadas.

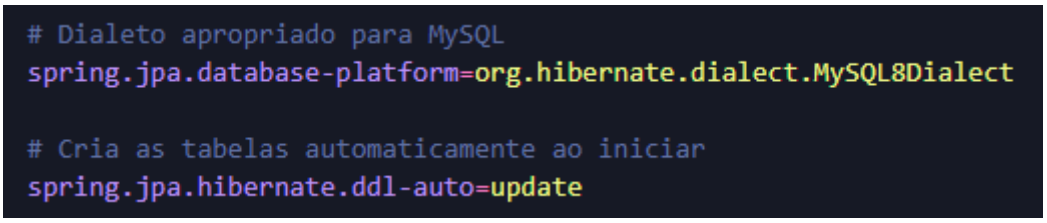
O *Hibernate* é o responsável por analisar essas assinaturas e transformá-las em comandos SQL, além de fazer o Object-Relational Mapping (ORM), que seria a conversão de objetos *Java* para tabelas no banco de dados. A Figura 13 mostra as configurações feitas para o uso do *Hibernate.*

Figura 12 - Assinaturas do JPA



Fonte: Produção própria (2025)

Figura 13 - Configuração do Hibernate



Fonte: Produção própria (2025)

Na Figura 14, é possível visualizar o modo que foi feito a conexão do *back-end* com o serviço RDS da AWS

Figura 14 - Configuração de conexão com o serviço RDS

```
#MySql AWS
spring.datasource.url=jdbc:mysql://festadb.cji40qyqg9eg.us-east-2.rds.amazonaws.com:3306/festadb
spring.datasource.username=[redacted]
spring.datasource.password=[redacted]
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

Fonte: Produção própria (2025)

4.9 Testes

A Figura 15 ilustra o relatório feito pelo *JaCoCo* com o foco nas classes de serviços. É possível notar que foi alcançada uma média de 87% na cobertura das classes.

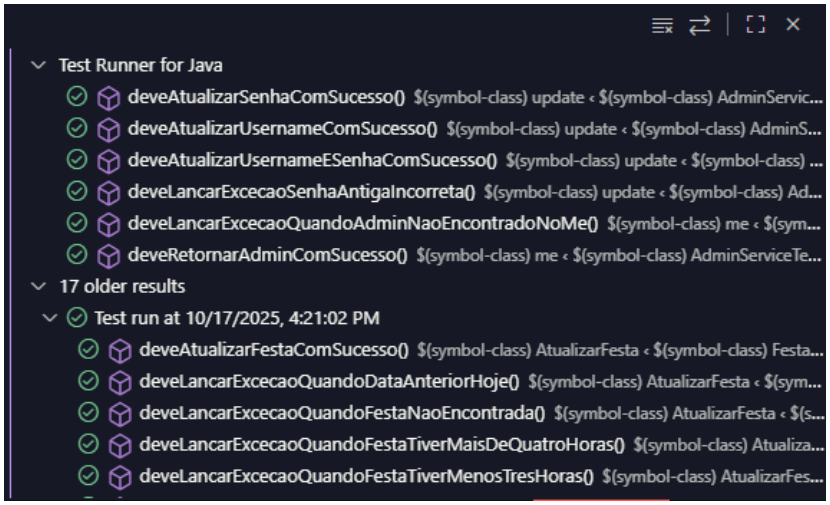
Figura 15 - Cobertura de testes na classe Festaservice e AdminService

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
FestaService		85%		71%	13	31	14	110	5	15	0	1
AdminService		100%		83%	2	10	0	12	0	4	0	1
Total	66 of 539	87%	11 of 44	75%	15	41	14	122	5	19	0	2

Fonte: Produção própria (2025)

A Figura 16 demonstra uma amostra dos resultados do *JUnit*5 relatados no *prompt*.

Figura 16 - Amostra de testes aprovados

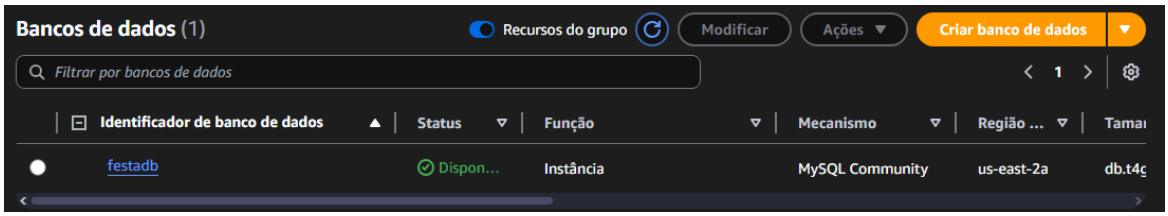


Fonte: Produção própria (2025)

4.10 Implementação

A Figura 17 apresenta uma captura de tela do console do Relational Database Service (RDS) na AWS.

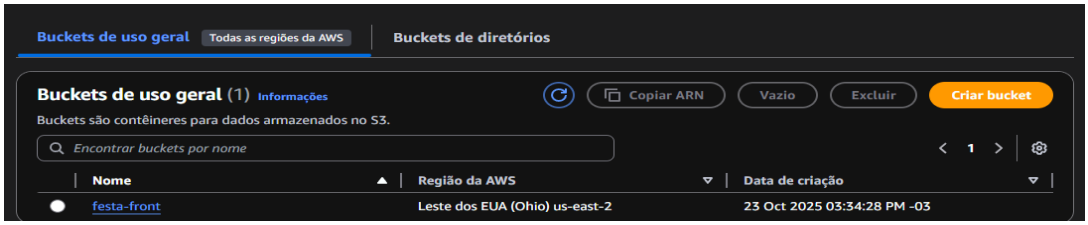
Figura 17 - Painel de controle do serviço RDS



Fonte: Produção própria (2025)

A Figura 18 exibe uma captura de tela do console do Simple Storage Service (S3) na AWS.

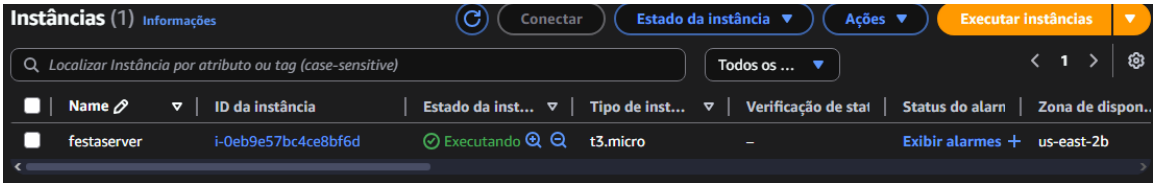
Figura 18 - Painel de controle do serviço S3



Fonte: Produção própria (2025)

A Figura 19 ilustra uma captura de tela do console do Elastic Compute Cloud (EC2) na AWS.

Figura 19 - Painel de controle do serviço EC2

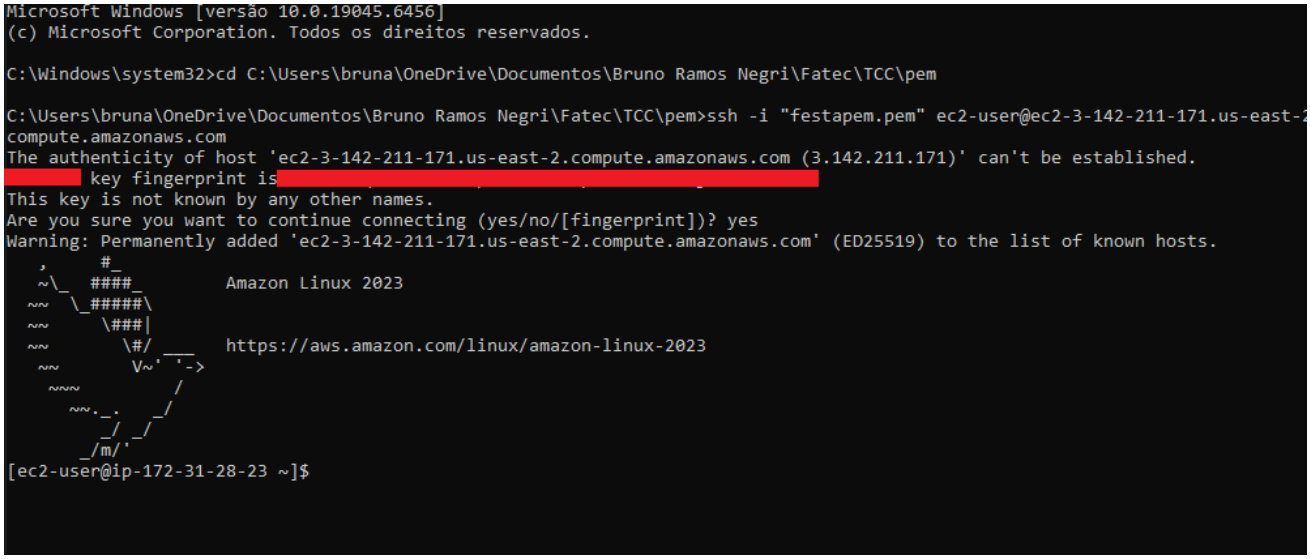


Fonte: Produção própria (2025)

As Figuras 20 e 21 demonstram o acesso à máquina virtual disponibilizada pelo serviço EC2 via terminal e a inicialização do *back-end*.

Utilizando o respectivo comando no terminal do Windows ‘ssh -i "festapem.pem" ec2-user@ec2-3-142-211-171.us-east-2.compute.amazonaws.com’ foi possível acessar a instância. O arquivo ‘.pem’ é fornecido pela própria plataforma no momento da criação da máquina virtual.”

Figura 20 - Entrando na instância EC2



Fonte: Produção própria (2025)

Foi possível enviar o arquivo .jar da minha máquina para a instância utilizando o seguinte comando: ‘scp -i "C:\aws\festapem.pem" "C:\Users\TCC\app-festa\target\app-festav2-0.0.1-SNAPSHOT.jar" ec2-user@ec2-3-142-211-171.us-east-2.compute.amazonaws.com:/home/ec2-user/projects’. Basicamente, informa-se primeiro a chave de acesso, depois o arquivo alvo e, por fim, o caminho onde ele deve ser salvo.

A execução do arquivo ‘.jar’ ocorreu após o uso do seguinte comando: ‘java --enable-preview -jar app-festav2-0.0.1-SNAPSHOT.jar’

Figura 21 - Rodando arquivo Java na instância EC2

Fonte: Produção própria (2025)

A Figura 22 apresenta a URL onde se pode encontrar o *front-end* que está rodando através do serviço S3.

Figura 22 - Caminho URL front-end

Fonte: Produção própria (2025)

O desenvolvimento do sistema é apresentado nesta seção de forma estruturada, combinando descrições técnicas, artefatos de engenharia de software e representações visuais que ilustram as decisões tomadas ao longo do processo de construção da aplicação.

Assim, os diagramas, quadros e exemplos apresentados ao longo do capítulo não apenas demonstram as funcionalidades desenvolvidas, mas também refletem a aplicação prática dos conceitos teóricos discutidos anteriormente, permitindo uma compreensão clara da arquitetura, da lógica de negócio e dos mecanismos que sustentam o funcionamento da solução proposta.

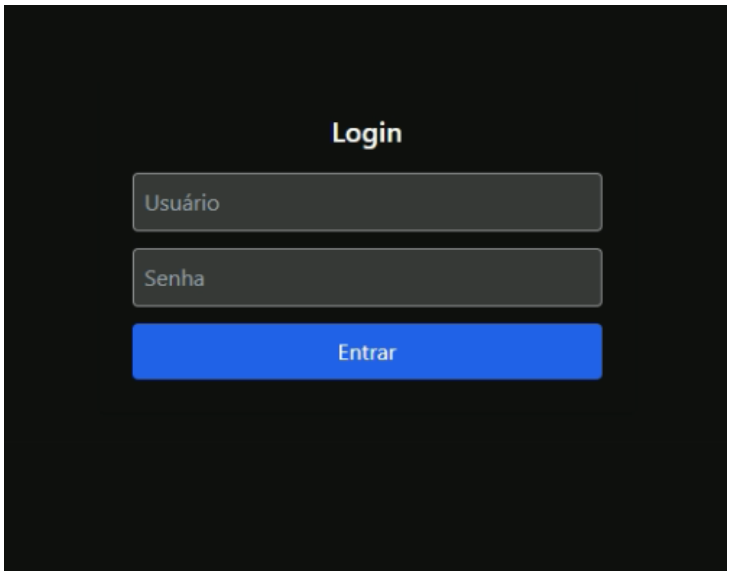
5 RESULTADOS

A seguir, serão exibidas capturas de tela da aplicação, mostrando suas principais funcionalidades e interfaces.

5.1 Demonstração das telas do sistema

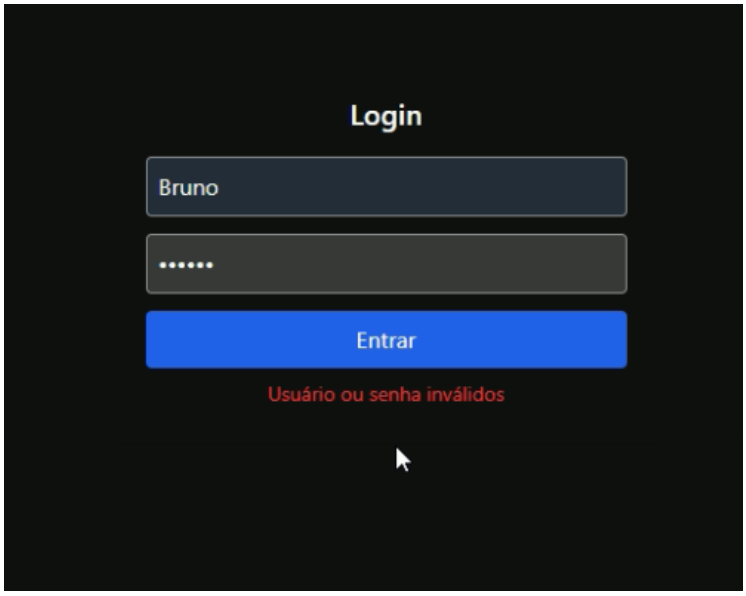
As Figuras 23 e 24 mostram a tela de login para acessar a aplicação. Ao inserir um usuário ou senha inválida ele retorna uma mensagem de erro

Figura 23 - Tela login



Fonte: Produção própria (2025)

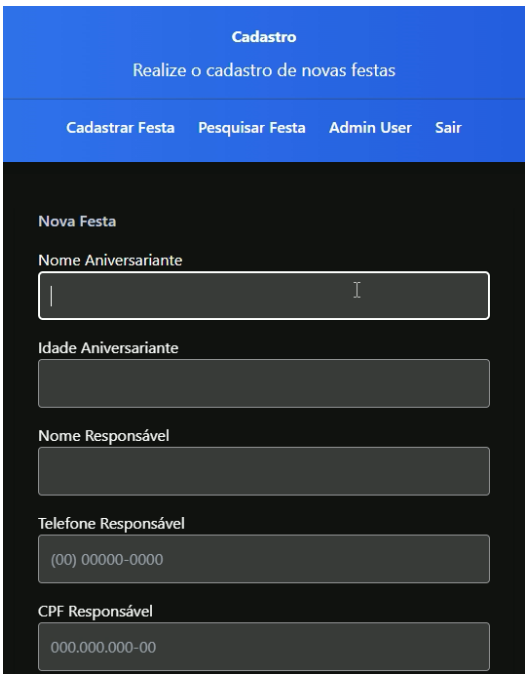
Figura 24 - Erro tela de login



Fonte: Produção própria (2025)

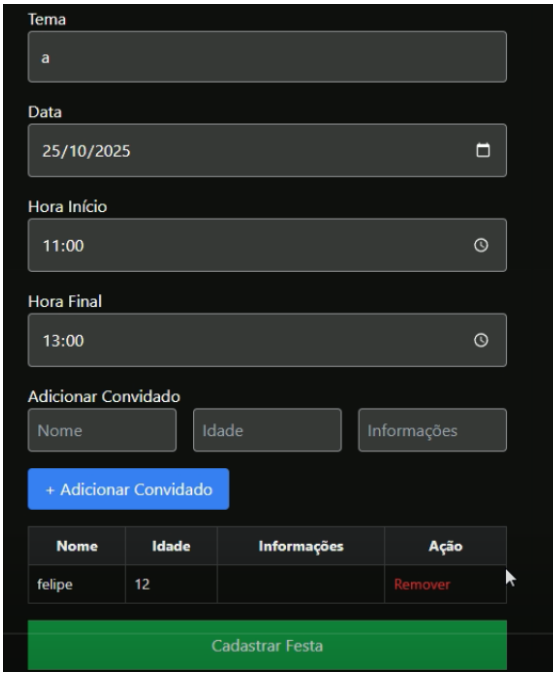
As Figuras 25 e 26 exibem a tela de cadastro de festa. Vale ressaltar que os campos de telefone e CPF possuem máscaras para melhor usabilidade. É possível perceber que na Figura 27 e 28 o sistema de verificação de campo nulo foi ativado.

Figura 25 - Tela cadastro e atualização de festa parte um



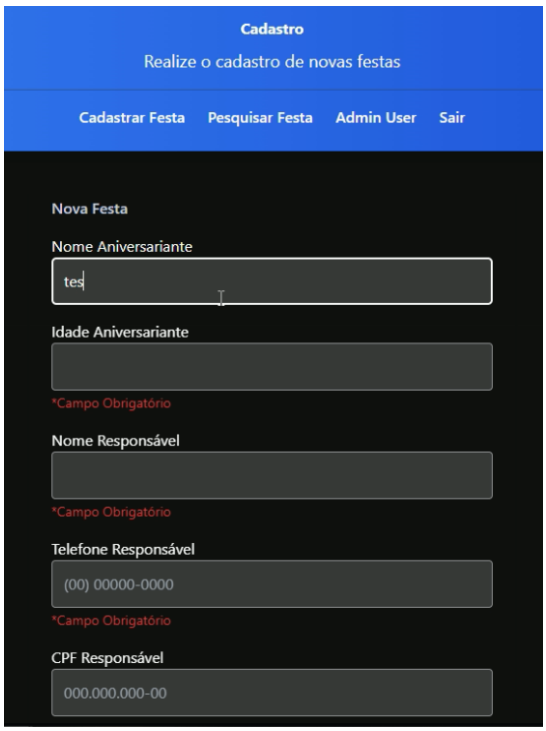
Fonte: Produção própria (2025)

Figura 26 - Tela cadastro e atualização de festa parte dois



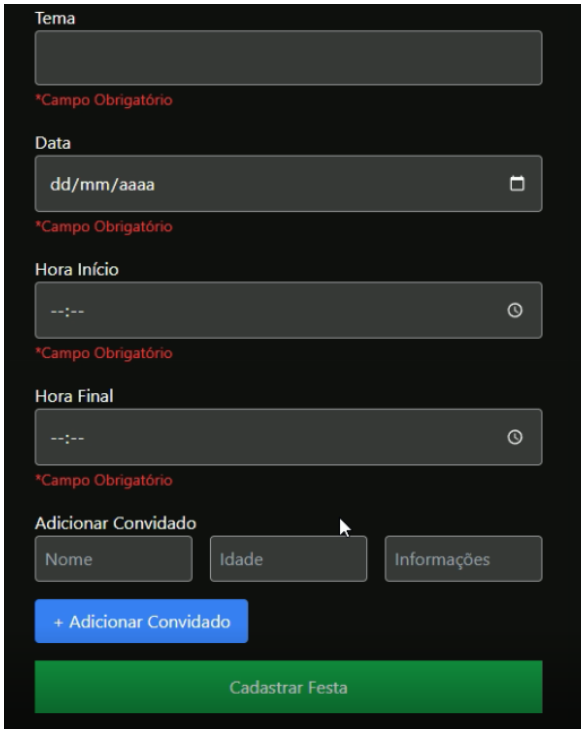
Fonte: Produção própria (2025)

Figura 27 - Erro campo vazio tela cadastro e atualização de festa parte um



Fonte: Produção própria (2025)

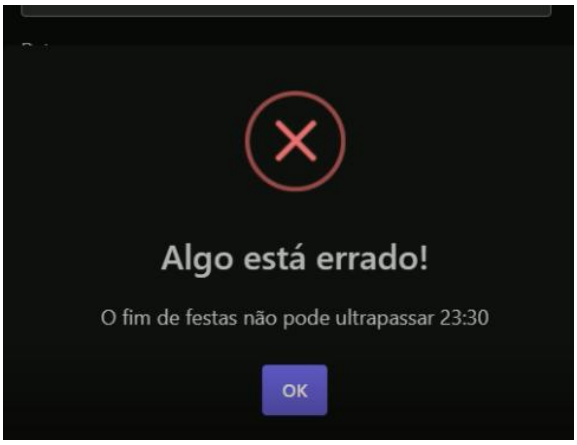
Figura 28 - Erro campo vazio tela cadastro e atualização de festa parte dois



Fonte: Produção própria (2025)

A Figura 29 apresenta uma mensagem de erro exibida caso o usuário tente criar uma festa cujo horário de término ultrapasse 23 horas e 30 minutos.

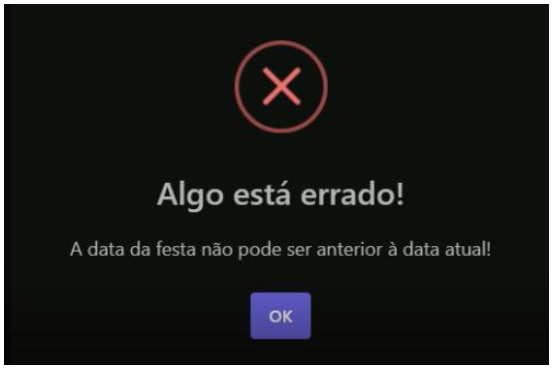
Figura 29 - Erro de cadastro e atualização um



Fonte: Produção própria (2025)

A Figura 30 apresenta uma mensagem de erro exibida caso o usuário tente criar uma festa cuja data seja anterior à data atual.

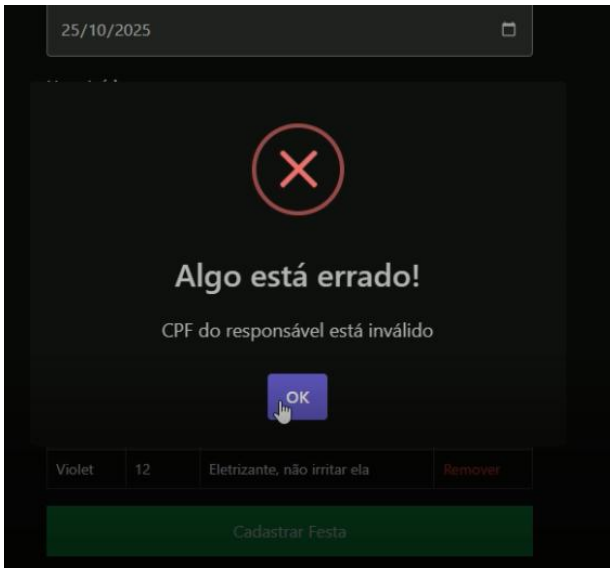
Figura 30 - Erro de cadastro e atualização dois



Fonte: Produção própria (2025)

A Figura 31 apresenta uma mensagem de erro exibida caso o usuário tente criar uma festa com um CPF inválido.

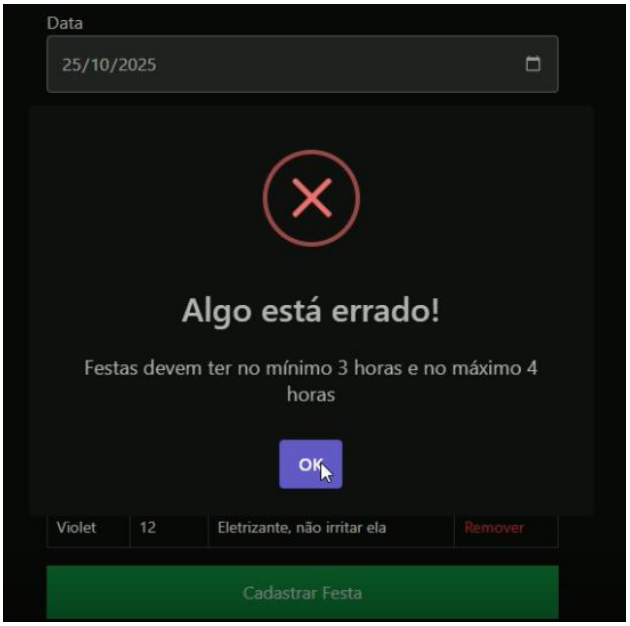
Figura 31 - Erro de cadastro e atualização três



Fonte: Produção própria (2025)

A Figura 32 apresenta uma mensagem de erro exibida caso o usuário tente criar uma festa cujo horário seja inferior a 3 horas.

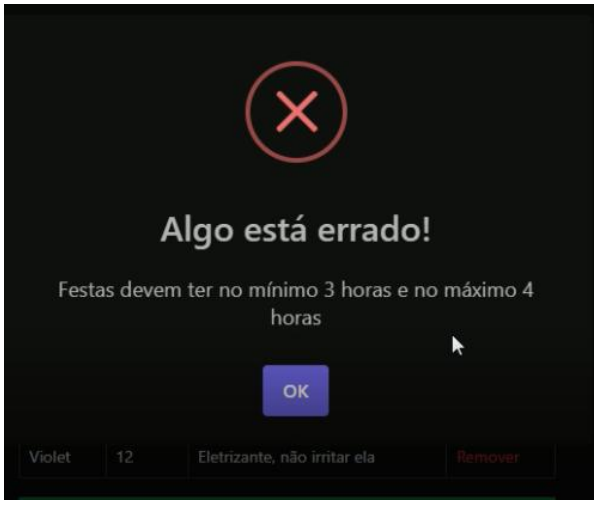
Figura 32 - Erro de cadastro e atualização quatro



Fonte: Produção própria (2025)

A Figura 33 apresenta uma mensagem de erro exibida caso o usuário tente criar uma festa cujo horário seja superior a 4 horas.

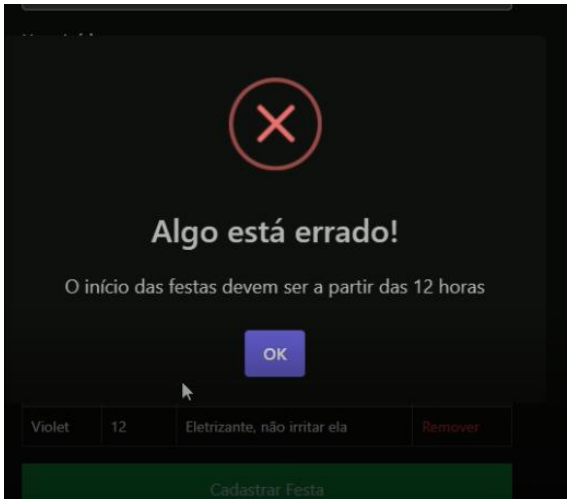
Figura 33 - Erro de cadastro e atualização cinco



Fonte: Produção própria (2025)

A Figura 34 apresenta uma mensagem de erro exibida caso o usuário tente criar uma festa cujo horário de início seja inferior as 12 horas.

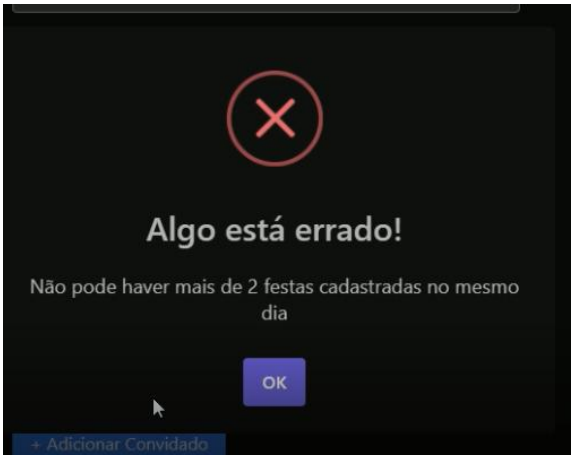
Figura 34 - Erro de cadastro e atualização seis



Fonte: Produção própria (2025)

A Figura 35 apresenta uma mensagem de erro exibida caso o usuário tente criar mais de duas festas no mesmo dia.

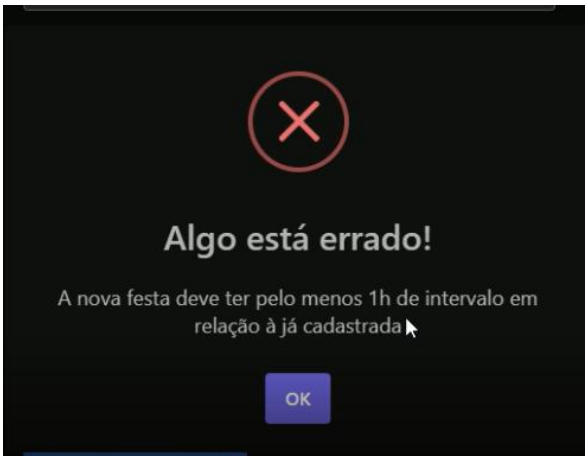
Figura 35 - Erro de cadastro e atualização sete



Fonte: Produção própria (2025)

A Figura 36 apresenta uma mensagem de erro exibida caso o usuário tente criar mais de duas festas com horários conflitantes ou sem o intervalo mínimo de 1 hora entre elas.

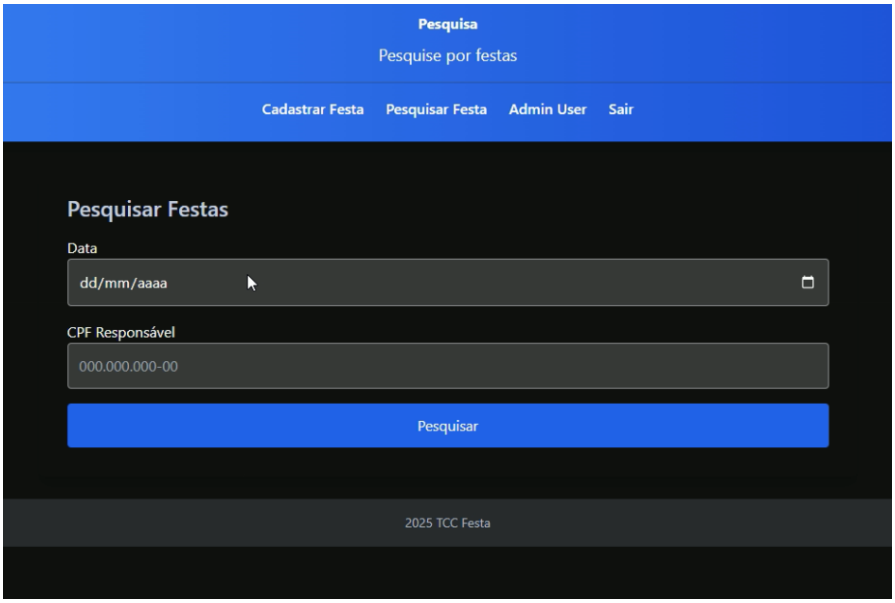
Figura 36 - Erro de cadastro e atualização oito



Fonte: Produção própria (2025)

A Figura 37 exibe a tela de pesquisa, onde é possível pesquisar com o CPF do responsável, data da festa ou com ambos ao mesmo tempo.

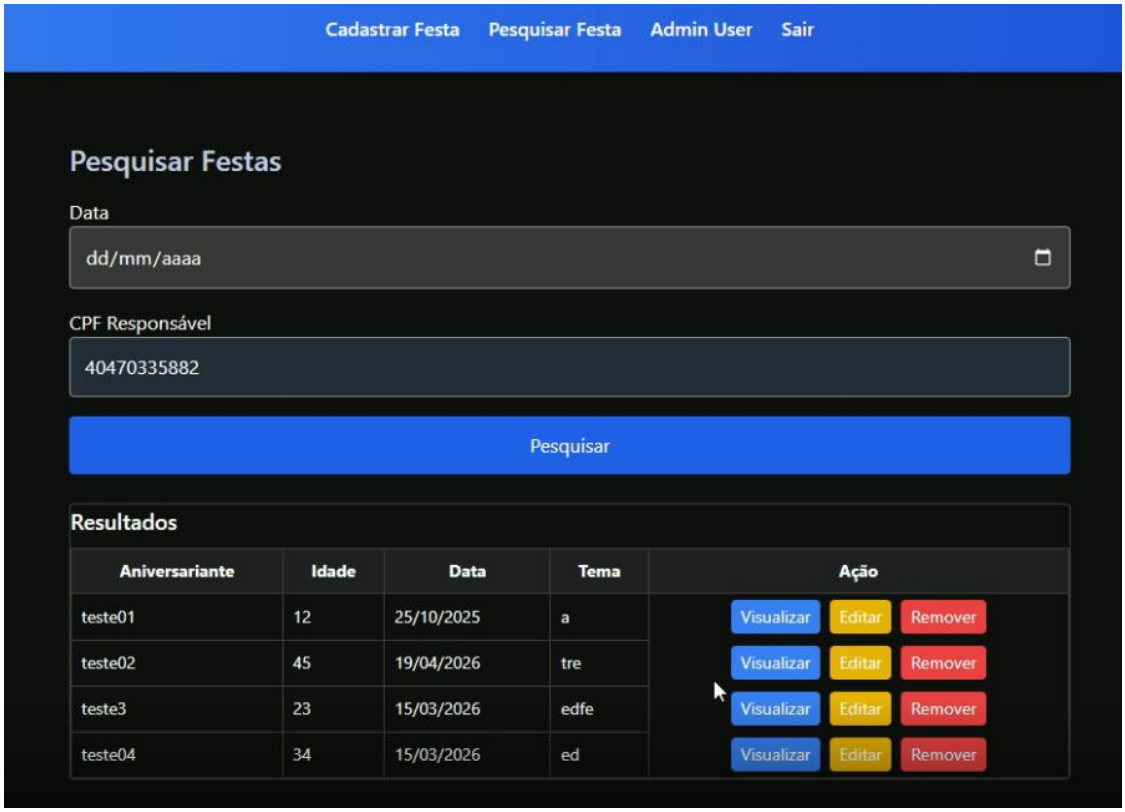
Figura 37 - Tela de pesquisa



Fonte: Produção própria (2025)

É possível observar na Figura 38 o retorno da pesquisa e as possibilidades que o software permite, dentre elas está: visualizar, editar e excluir (soft delete).

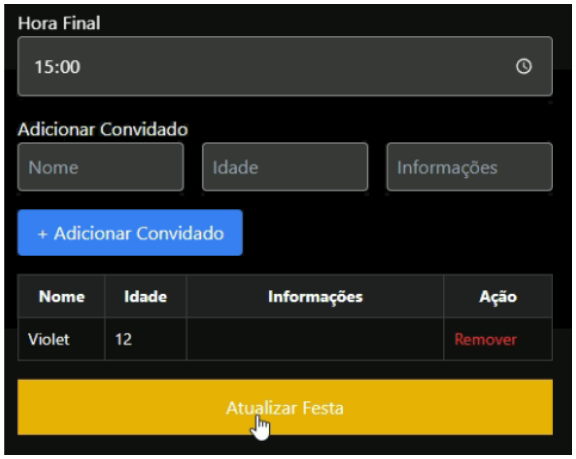
Figura 38 - Tela de pesquisa com dados encontrados



Fonte: Produção própria (2025)

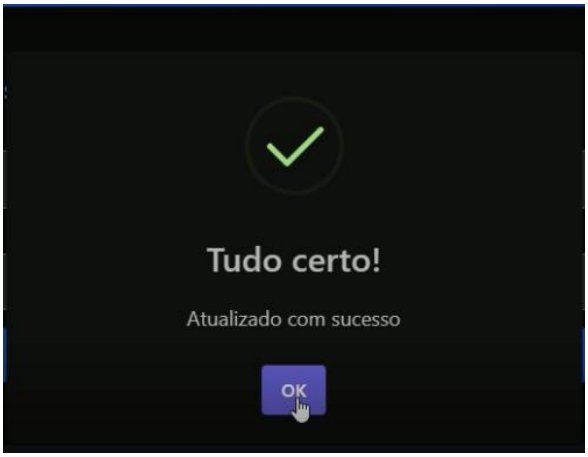
As Figuras 39 e 40 mostram a ação de atualizar. Ela reaproveita a interface da tela de cadastro e apenas muda o botão de *submit* e o feedback. Caso ocorra algum erro, os feedbacks serão os mesmos da ação de cadastro.

Figura 39 - Botão de atualizar



Fonte: Produção própria (2025)

Figura 40 - Feedback de atualização feita com sucesso



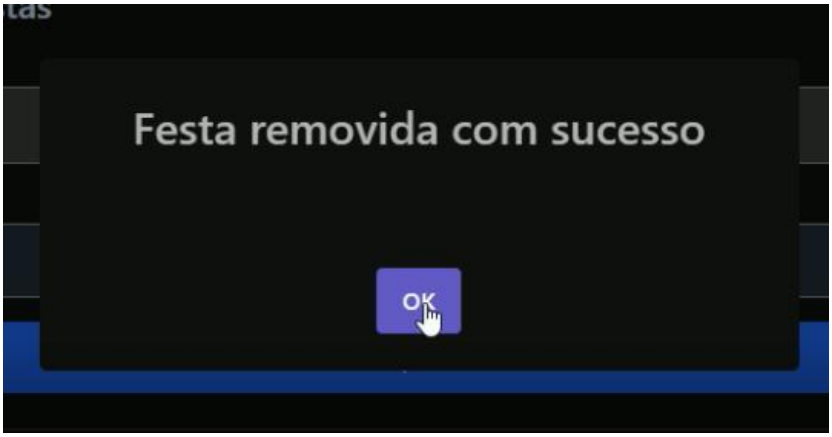
Fonte: Produção própria (2025)

As Figuras 41 e 42 ilustram ação de remover uma festa, no banco de dados apenas atualizando a variável “ativo” para false.

Figura 41 - Confirmação de remoção da festa

Fonte: Produção própria (2025)

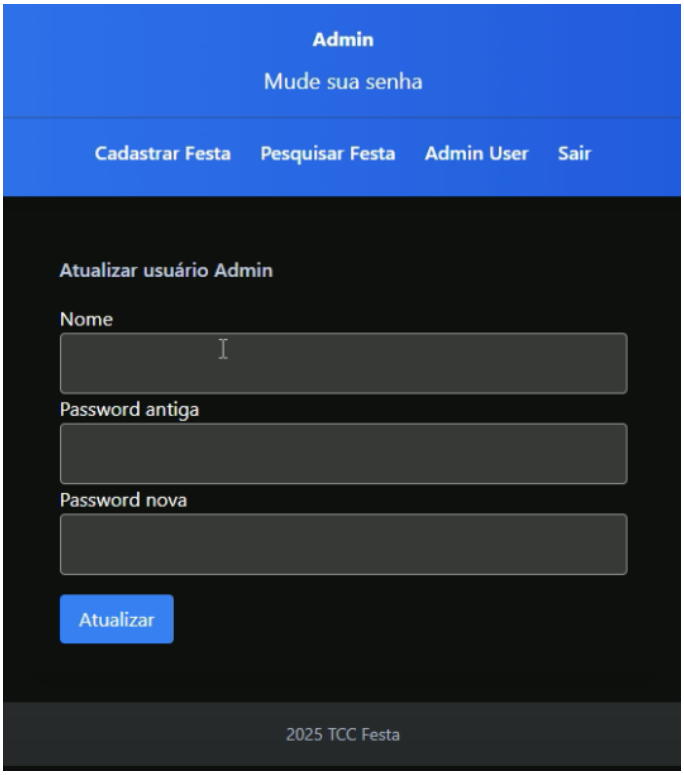
Figura 42 - Feedback festa removida com sucesso



Fonte: Produção própria (2025)

A Figura 43 mostra a tela da alteração de nome e senha do admin.

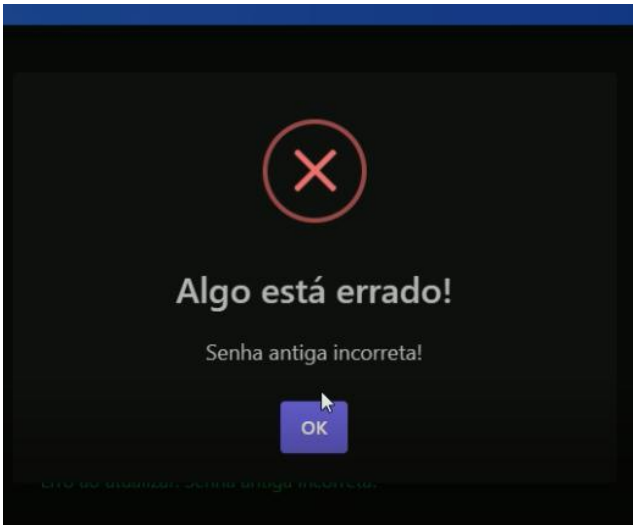
Figura 43 - Tela de configuração conta admin



Fonte: Produção própria (2025)

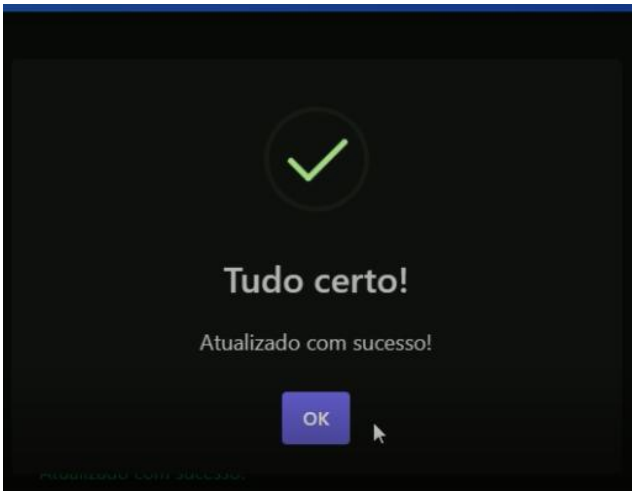
As Figuras 44 e 45 apresentam seus respectivos feedbacks. Por uma questão de segurança, as informações de login do administrador só podem ser alteradas mediante um valor-chave, que corresponde à senha atual do usuário admin.

Figura 44 - Erro atualização nome ou senha admin



Fonte: Produção própria (2025)

Figura 45 - Feedback atualização admin com sucesso



Fonte: Produção própria (2025)

5.2 Comparação entre objetivos e resultados alcançados

Em um escopo geral, o projeto seguiu estritamente os pontos essenciais para o bom funcionamento da aplicação para o uso de um *buffet*. Dentre esses pontos, destacam-se as regras de negócio e as mensagens de erro humanizadas.

A primeira, sendo um requisito funcional, foi implementada com êxito; já a segunda, mesmo sendo um requisito não funcional, foi desenvolvida com a devida

atenção necessária para uma melhor experiência do usuário, sempre visando o provável nível de conhecimento do cliente na área da computação como básico.

5.3 Benefícios para a gestão do buffet infantil

Com a aplicação em mãos, a questão sobre como melhorar a gestão de festas em um *buffet* infantil ganhou uma opção robusta, simples de usar e com uma interface amigável.

Robusta pelo fato de possuir autenticação com JWT no *back-end*, além de estar em produção nos ambientes da AWS, possibilitando, assim, ambientes escaláveis, backups de dados e separação dos componentes (*back-end*, *front-end* e banco de dados), o que permite modificações unitárias sem alterar necessariamente o andamento de outro componente.

Simples de usar, pois a aplicação segue um padrão, não sendo alterado ao longo das páginas. Seus inputs e botões possuem textos, tornando autoexplicativo qual tipo de dado deve ser inserido ou qual funcionalidade o botão exerce.

Sua interface amigável é notória por um padrão simples de objetos presentes na tela, mantendo uma paleta de cores não enjoativa, além de apresentar feedback a cada interação que o usuário faz com a aplicação.

6 CONCLUSÃO

A presente seção apresentará o resumo dos resultados esperados e alcançados, bem como os pontos de limitação e as possíveis melhorias.

6.1 Síntese do trabalho desenvolvido

Ao fazer uma síntese analítica da criação da aplicação, observando desde seu planejamento até sua execução em um ambiente de produção, é possível observar técnicas de engenharia de software para a construção da base do projeto. Para o acompanhamento do projeto, a implementação de uma metodologia ágil, criando assim objetivos concretos de entrega para não haver nenhum tipo de atraso.

O projeto contempla o uso de tecnologias amplamente utilizadas no mercado atualmente, seja com o objetivo de versionamento de código, repositório e *deploy*, ou também para a criação do projeto em seu *front-end* e *back-end*.

Como relatado no Capítulo 5, Seção 5.2 “Comparação entre objetivos e resultados alcançados”, a aplicação, com êxito, consegue oferecer uma solução para melhorar a gestão de festas em um *buffet* infantil, possibilitando o desuso de planilhas e armazenamento local sem um padrão implementado.

Vale ressaltar que o uso da aplicação, embora traga uma solução robusta e eficiente, não está pronta para o uso imediato em um cenário real, pois as regras de negócio podem variar de acordo com os *buffets*. É de suma importância relembrar que os testes unitários foram feitos com foco nas classes *services*, ou seja, em uma aplicação real os testes devem cobrir muito mais do que foi apresentado, para garantir a segurança completa do usuário. Entretanto, esses pontos não desqualificam, em nenhuma hipótese, a riqueza de conhecimento presente e sua funcionalidade.

6.2 Limitações encontradas

A limitação mais vigente apresentada durante a produção da aplicação é o uso dos serviços da AWS.

Conforme informações da plataforma, novos clientes recebem 100 USD para testes dos produtos disponíveis, ou seja, de forma gratuita.

O usuário pode usufruir da possibilidade de conseguir mais 100 USD ao cumprir os objetivos propostos pela plataforma, explorando mais seus serviços; todavia, terá 6 meses para usar todos os seus créditos, e após esse período, perde o valor concedido (AWS BILLING, 2025).

6.3 Sugestões para trabalhos futuros

Sempre há algo a ser melhorado quando se trata do universo da tecnologia, e para o projeto apresentado não é diferente.

Possíveis modificações que podem ser feitas em sprints futuras são: possibilitar o cadastro de funcionários e deixar a conta administrativa apenas com o dono do *buffet*; adicionar gráficos de análise de dados, permitindo ao usuário observar métricas como média de idades ou quantidade de pessoas por festa; e acrescentar uma página administrativa para gerenciar gastos e lucros das festas, o que seria extremamente útil.

REFERÊNCIAS

AMAZON WEB SERVICES. *AWS Documentation*. Disponível em: https://docs.aws.amazon.com/. Acesso em: 4 nov. 2025.

ANGULAR. *Angular Documentation*. Disponível em: https://angular.dev/. Acesso em: 3 out. 2025.

AWS BILLING. *Explore AWS Services with AWS Free Tier*. Disponível em: https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/free-tier.html. Acesso em: 1 nov. 2025.

FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. 2000. 162 f. Tese (Doutorado em Ciência da Computação) – University of California, Irvine. Disponível em: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf. Acesso em: 4 out. 2025.

GIT. *Pro Git Book: Documentation*. Disponível em: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control. Acesso em: 4 out. 2025.

H2 DATABASE. *H2 Database Engine*. Disponível em: https://www.h2database.com/. Acesso em: 3 out. 2025.

JUNIT. *JUnit 5 User Guide*. Disponível em: https://junit.org/junit5/. Acesso em: 3 out. 2025.

LAUDON, Kenneth C.; LAUDON, Jane P. *Sistemas de informação gerenciais*. 12. ed. São Paulo: Pearson, 2014.

ORACLE. *Java Documentation*. Disponível em: https://docs.oracle.com/en/java/. Acesso em: 4 out. 2025.

SCHWABER, Ken; SUTHERLAND, Jeff. *Guia do Scrum: O guia definitivo para o Scrum – As regras do jogo*. 2020. Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf. Acesso em: 3 out. 2025.

SCHWABER, Ken; SUTHERLAND, Jeff. *The Scrum Guide*. Disponível em: https://scrumguides.org/scrum-guide.html. Acesso em: 18 out. 2025.

SPRING. *Spring Boot Reference Documentation*. Disponível em: https://spring.io/projects/spring-boot. Acesso em: 3 out. 2025.

APÊNDICE A – ARQUIVO COMPACTADO DO *BACK-END*

O presente apêndice contém o arquivo compactado referente ao código-fonte completo do back-end desenvolvido para este trabalho. O arquivo inclui a estrutura de diretórios, classes Java, serviços, controladores, configuração de segurança com JWT, entidades mapeadas pelo JPA e demais componentes utilizados na implementação da API construída com Spring Boot 3.5.0.
O arquivo compactado está disponível para consulta, permitindo análise detalhada da arquitetura e das regras de negócio empregadas na solução.

app-festa.rar

APÊNDICE B – ARQUIVO COMPACTADO DO *FRONT-END*

Este apêndice apresenta o arquivo compactado contendo o código-fonte integral do front-end do sistema. Nele estão incluídos os componentes Angular 19, serviços de comunicação com o back-end, modelos de dados, folhas de estilo, além de templates e elementos visuais utilizados para a composição da interface gráfica. O conteúdo disponibilizado permite compreender a estrutura modular da aplicação e acompanhar o processo de construção e integração da camada visual.

app-festa-front.rar