

**izabelly cristina silva brito**

**kétlyn cristine tonholi**

**maysa lopes de sousa**

# HavenNanny

## Plataforma Inteligente para Conexão entre Pais e Babás

**Projeto Integrador**

Jales 2025

# HavenNanny

## Plataforma Inteligente para Conexão entre Pais e Babás

Projeto Integrador apresentado à Faculdade de Tecnologia Professor José Camargo – Fatec Jales, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistema.

Orientador: Prof. Jefferson Antônio Ribeiro Passarini.

Jales 2025

# HavenNanny

## Plataforma Inteligente para Conexão entre Pais e Babás

Projeto Integrador apresentado à Faculdade de Tecnologia Professor José Camargo – Fatec Jales, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistema.

Orientador: Prof. Jefferson Antônio Ribeiro Passarini.

## Banca Examinadora:

Prof. (Orientador) Instituição

Prof. Instituição

Prof. (Orientador) Instituição

Jales, de de .

# RESUMO

O HavenNanny é uma plataforma inovadora que visa conectar pais a babás de forma prática, eficiente e segura, oferecendo uma solução completa para o gerenciamento de cuidados infantis. Através de um sistema inteligente de matchmaking, o software sugere babás baseadas nas necessidades e preferências dos pais, levando em conta a experiência e compatibilidade com o perfil da criança. Além disso, a plataforma proporciona funcionalidades essenciais como agendamento de horários, monitoramento de atividades infantis, checklists personalizados, e sistemas de avaliação, garantindo uma experiência transparente e confiável para todas as partes envolvidas. O projeto visa não só facilitar a contratação de babás, mas também promover uma comunicação constante e o acompanhamento de cuidados em tempo real, oferecendo uma abordagem diferenciada no mercado brasileiro. A solução foi desenvolvida com o objetivo de proporcionar aos pais maior segurança e controle sobre o bem-estar de seus filhos, enquanto oferece às babás as ferramentas necessárias para otimizar sua gestão de tempo e carreira. Com um modelo de monetização baseado no freemium, o HavenNanny oferece recursos exclusivos para aprimorar a experiência do usuário e apoiar a qualidade do atendimento.

**Palavras-chave:** Palavras-chave: Gestão infantil. Plataforma de Conexão. Cuidado infantil. Tecnologia para pais e babás.

***ABSTRACT***

*HavenNanny is an innovative platform designed to connect parents with nannies in a practical, efficient, and secure way, offering a comprehensive solution for child care management. Through an intelligent matchmaking system, the software suggests nannies based on parents' needs and preferences, considering experience and compatibility with the child's profile. Additionally, the platform provides essential features such as schedule management, activity monitoring, personalized checklists, and rating systems, ensuring a transparent and reliable experience for all parties involved.*

*The project aims not only to facilitate the hiring of nannies but also to promote constant communication and real-time child care monitoring, offering a differentiated approach in the Brazilian market. The solution was developed to provide parents with greater security and control over their children's well-being while equipping nannies with the necessary tools to optimize their time management and career. With a freemium-based monetization model, HavenNanny offers exclusive features to enhance user experience and support the quality of service.*

*.*

***Keywords:*** *Child care management. Connection platform. Child care. Technology for parents and nannies.*

# LISTA DE FIGURAS

**L+ISTA DE QUADROS**

Quadro 1 – Requisitos Funcionais........................................................................15

Quadro 2 – Requisitos Não Funcionais................................................................16

**SUMÁRIO**

1. [**INTRODUÇÃO 10**](#_bookmark0)
2. [**LEVANTAMENTO DE REQUISITOS DE SOFTWARE 11**](#_bookmark1)
   1. [Descrição dos objetivos do sistema 11](#_bookmark2)
   2. [Descrição do sistema atual 12](#_bookmark3)
   3. [Análise De Sistemas Existentes 14](#_bookmark9)
   4. [Descrição dos principais problemas 17](#_bookmark13)
   5. [Descrição dos requisitos funcionais 17](#_bookmark14)
   6. [Descrição dos requisitos não funcionais 18](#_bookmark15)

# [VISÃO DE CASO DE USO – UML 20](#_bookmark16)

* 1. [Diagrama de Classes 20](#_bookmark17)
  2. [Dicionário de Classes 22](#_bookmark19)
  3. [Definição dos Atores 31](#_bookmark41)
  4. [Lista de Casos de Uso 32](#_bookmark43)
  5. [Diagrama de Casos de Uso 40](#_bookmark46)
     1. [Diagrama de Casos de uso individuais 46](#_bookmark51)
     2. [Diagrama de Casos de uso individuais – Listar tipo atração 51](#_bookmark58)
     3. [Diagrama de Casos de uso individuais – Listar atração 56](#_bookmark65)
     4. [Diagrama de Casos de uso individuais – Listar Turismo 61](#_bookmark72)
  6. [Diagrama de Sequência 66](#_bookmark79)
     1. [Diagrama de Sequência – Cadastrar Tipo Turismo 66](#_bookmark80)
     2. [Diagrama de Sequência – Alterar Tipo Atração 67](#_bookmark82)
     3. [Diagrama de Sequência – Listar Usuário 69](#_bookmark84)
     4. [Diagrama de Sequência – Excluir Empresa 70](#_bookmark86)
     5. [Diagrama de Sequência – Consultar Atração 71](#_bookmark88)

# [DEFINIÇÃO DA INTERFACE COM O USUÁRIO (UX) 72](#_bookmark90)

* 1. [Descrição de Cenário 72](#_bookmark91)
  2. [Descrição de Personas 73](#_bookmark94)
  3. [Esboços de tela (*WIREFRAMES*) 75](#_bookmark97)
  4. [Protótipos de tela 79](#_bookmark102)

# [BANCO DE DADOS 82](#_bookmark107)

* 1. [Modelo Entidade Relacionamento 82](#_bookmark108)
  2. [Script das Tabelas 83](#_bookmark110)
  3. [Mapeamento Objeto Relacional – ORM 90](#_bookmark129)

# [ARQUITETURA DE SOFTWARE 91](#_bookmark130)

* 1. [Arquitetura de Desenvolvimento 91](#_bookmark131)
     1. [BACK-END 92](#_bookmark132)
     2. [FRONT-END 93](#_bookmark133)
  2. [Segurança da Informação 93](#_bookmark134)
  3. [Implantação do Sistema 94](#_bookmark135)

1. [**CONCLUSÃO 98**](#_bookmark137)
2. [**REFERÊNCIAS 99**](#_bookmark138)
3. **INTRODUÇÃO**

A tecnologia tem desempenhado um papel fundamental na transformação de diversos setores, e a gestão do cuidado infantil não é exceção. O crescimento da demanda por soluções que conectem pais a profissionais qualificados reflete a necessidade de serviços eficientes, seguros e acessíveis. Segundo pesquisas recentes, muitos pais enfrentam desafios para encontrar babás confiáveis, enquanto profissionais do setor buscam melhores oportunidades de trabalho e organização de sua rotina. Nesse contexto, o HavenNanny surge como uma plataforma inovadora que facilita a intermediação entre pais e babás, promovendo um ambiente mais transparente, seguro e acessível para ambas as partes.

Atualmente, a busca por babás ainda ocorre, em muitos casos, por meio de redes informais e recomendações pessoais, o que pode dificultar a verificação da experiência e da compatibilidade dos profissionais com as necessidades específicas das famílias. A proposta do HavenNanny é utilizar a tecnologia para aprimorar esse processo, oferecendo um sistema inteligente de matchmaking que sugere profissionais qualificados com base em critérios como experiência, preferências dos pais e perfil da criança. Além disso, a plataforma proporciona funcionalidades essenciais, como agendamento de horários e um sistema de avaliação, garantindo mais segurança e confiabilidade no serviço prestado.

O HavenNanny não se limita à simples conexão entre pais e babás, mas busca também promover uma comunicação constante e um acompanhamento detalhado do cuidado infantil, fortalecendo a relação de confiança entre as partes envolvidas. O modelo de monetização freemium permite que os usuários acessem funcionalidades básicas gratuitamente, enquanto recursos avançados são disponibilizados por meio de planos pagos, garantindo acessibilidade e qualidade.

Dessa forma, este projeto visa desenvolver e validar o HavenNanny como uma solução inovadora no mercado brasileiro, analisando seu impacto na gestão do cuidado infantil e explorando as tecnologias utilizadas para sua implementação.

# LEVANTAMENTO DE REQUISITOS DE SOFTWARE

O levantamento de requisitos é um estágio crucial no ciclo de desenvolvimento de software, desempenhando um papel fundamental na compreensão das necessidades dos usuários e na definição clara do escopo do projeto. Para Roger S. Pressman:

A engenharia de requisitos fornece o mecanismo apropriado para entender o que o cliente deseja, analisando as necessidades, avaliando a exequibilidade, negociando uma condição razoável, especificando e gerindo os requisitos à medida que eles são transformados em um sistema operacional (PRESSMAN, 2021, p.118).

Conforme observado por Sommerville (2018), o levantamento de requisitos não apenas estabelece a base fundamental, mas também se configura como um estágio essencial no ciclo de vida do desenvolvimento de software. Sua importância reside na capacidade de construir um software que atenda às demandas dos usuários e agregue valor ao negócio. Esse processo crítico assegura o sucesso do projeto ao garantir uma compreensão clara e abrangente das expectativas e requisitos do cliente, delineando diretrizes cruciais para a construção de um sistema que verdadeiramente satisfaça as necessidades e objetivos estabelecidos.

* 1. Descrição dos objetivos do sistema

Os conceitos apresentados são materializados por meio do desenvolvimento do HavenNanny, um software projetado para tornar a busca e a contratação de babás mais acessível e eficiente para os pais, ao mesmo tempo em que oferece aos profissionais do setor uma plataforma estruturada para otimizar sua gestão de tempo e carreira. A proposta central do software é proporcionar um ambiente confiável e seguro, no qual pais possam encontrar profissionais qualificados de acordo com suas necessidades, e babás possam gerenciar suas oportunidades de trabalho de maneira organizada.

A plataforma promove um sistema de avaliações que contribui para a construção de uma rede de profissionais qualificados, garantindo um serviço de qualidade.

Ao analisar as implicações desse aprimoramento na gestão do cuidado infantil, torna-se evidente o impacto positivo tanto para as famílias quanto para os profissionais. O software tem o potencial de proporcionar maior segurança e tranquilidade aos pais, ampliar oportunidades de trabalho para babás e fortalecer o setor de serviços infantis, impulsionando a economia local e melhorando a qualidade de vida das famílias que necessitam desse suporte.

* 1. Descrição do sistema atual

Atualmente, não existe uma plataforma centralizada no Brasil que ofereça todas as funcionalidades integradas do HavenNanny. Pais e babás recorrem a redes sociais, grupos de mensagens e indicações para encontrar profissionais ou oportunidades, o que pode gerar insegurança e falta de critérios padronizados de avaliação. Algumas plataformas de recrutamento geral incluem anúncios de babás, mas sem oferecer recursos específicos para acompanhamento das atividades infantis, gestão de horários ou avaliações detalhadas. O HavenNanny surge como uma solução inovadora ao integrar um sistema de match inteligente, controle de agenda, notificações de tarefas e feedbacks detalhados, proporcionando um serviço mais confiável e organizado.

* 1. Análise De Sistemas Existentes

Após uma análise detalhada das plataformas existentes para contratação de babás, identificamos que muitas oferecem funcionalidades básicas de recrutamento, mas carecem de ferramentas avançadas de monitoramento e gestão do cuidado infantil. Por exemplo, o Babysits facilita a conexão entre famílias e babás, permitindo pesquisa, comunicação e agendamento de serviços de babá de forma imediata.

No contexto brasileiro, plataformas como a Sitly oferecem recursos para encontrar babás próximas, com a possibilidade de filtrar resultados por localização e outras características. Entretanto, essas plataformas geralmente não incluem mecanismos integrados de verificação de antecedentes, avaliações detalhadas ou sistemas estruturados de organização de horários e tarefas.

O HavenNanny diferencia-se ao integrar todas essas funcionalidades em uma única plataforma. Além de facilitar a conexão entre pais e babás, o HavenNanny oferece ferramentas completas de monitoramento e gestão do cuidado infantil, garantindo mais segurança e eficiência para ambas as partes.

* 1. Descrição dos principais problemas

Os principais desafios enfrentados pelo HavenNanny estão relacionados à segurança, confiabilidade e adesão dos usuários à plataforma. Um dos problemas centrais é garantir que os pais encontrem babás qualificadas e confiáveis, minimizando riscos e proporcionando um ambiente seguro para as crianças. A falta de verificação rigorosa de antecedentes e experiências pode gerar desconfiança e dificultar a adoção do sistema por novos usuários. Para resolver essa questão, é essencial implementar um processo robusto de validação de perfis, incluindo avaliações, certificações e recomendações.

Outro desafio relevante está na adaptação da plataforma às necessidades tanto dos pais quanto das babás. Para os pais, é fundamental que o sistema seja intuitivo, permitindo uma busca eficiente e segura por profissionais compatíveis. Já para as babás, a plataforma deve oferecer ferramentas que facilitem a gestão de suas atividades e oportunidades de trabalho. Se essas demandas não forem bem atendidas, há o risco de baixa adesão ao software.

Além disso, a monetização do serviço deve ser equilibrada para garantir que os recursos premium sejam atrativos, sem prejudicar o acesso às funcionalidades essenciais. O modelo freemium precisa ser estruturado de forma que agregue valor tanto para usuários gratuitos quanto para assinantes, incentivando a sustentabilidade da plataforma a longo prazo.

* 1. Descrição dos requisitos funcionais

O HavenNanny destaca-se por conectar pais a babás de maneira prática e confiável, oferecendo funcionalidades que facilitam a comunicação, o agendamento e o monitoramento dos cuidados infantis. A partir desse princípio, seus requisitos funcionais definem os serviços fundamentais que a plataforma deve disponibilizar.

Os botões de ação, como cadastrar, editar e excluir, desempenham um papel essencial na organização das informações, permitindo o gerenciamento de perfis de pais e babás, a atualização de dados e a exclusão de registros inativos. O sistema de pesquisa também é uma característica fundamental, permitindo que os pais filtrem babás com base em experiência, localização e disponibilidade, garantindo uma busca eficiente e precisa.

A acessibilidade é uma prioridade no desenvolvimento do HavenNanny, garantindo que a plataforma seja intuitiva e adaptável a diferentes dispositivos.

Além disso, o software possibilita a integração com sistemas externos, como serviços de geolocalização e notificações em tempo real, ampliando sua funcionalidade e alcance. Para tornar a experiência mais interativa e confiável, o HavenNanny conta com um sistema de avaliações, onde pais e babás podem fornecer feedback sobre suas experiências, promovendo um ambiente seguro e transparente para todos os usuários.

Quadro 1 – Requisitos Funcionais

|  |  |  |
| --- | --- | --- |
| **ID** | **Requisitos Funcionais** | **Descrição** |
| RF01 | Cadastro de Pais | Os pais devem poder se cadastrar fornecendo informações pessoais, município e dados dos filhos (idade, necessidades especiais, preferências alimentares, horários de rotina, alergias). |
| RF02 | Cadastro de Babás | As babás devem poder se cadastrar com informações pessoais, experiência, formação, disponibilidade e referências. |
| RF04 | Agendamento de Serviços | Os pais devem poder agendar horários de cuidados com a babá diretamente pela plataforma. |
| RF05 | Confirmação de Presença | O sistema deve registrar a confirmação de presença da babá e o tempo de cuidado. |
| RF06 | Avaliação dos Serviços | Os pais devem poder avaliar a performance da babá após o atendimento. |
| RF07 | Avaliação das Famílias | As babás devem poder avaliar a comunicação e organização dos pais. |
| RF10 | Criação de Checklists | Os pais devem poder criar checklists diários ou semanais para a babá seguir (alimentação, trocas de fraldas, sonecas, etc). |
| RF12 | Plano Freemium | O sistema deve oferecer recursos básicos gratuitos e recursos avançados pagos. |
| RF13 | Assinaturas para Babás | As babás devem poder pagar uma taxa para se destacar no sistema e obter mais visibilidade. |
| RF14 | Gerenciamento de Agenda | As babás devem poder gerenciar sua agenda e disponibilidade. |
| RF15 | Comunicação em Tempo Real | O sistema deve permitir comunicação entre pais e babás para atualizações durante o atendimento. |
| RF16 | Histórico de Atendimentos | O sistema deve manter um histórico de todos os atendimentos realizados pela babá. |

Fonte: Elaborada pelos autores.

* 1. Descrição dos requisitos não funcionais

Além dos requisitos funcionais, Sommerville (2018) também aborda os requisitos não funcionais do sistema. Esses requisitos abrangem aspectos gerais do sistema, em vez de se concentrarem em serviços e características individuais como os requisitos funcionais. Alguns exemplos de requisitos não funcionais são “restrições de tempo, restrições sobre o processo de desenvolvimento e restrições impostas por padrões” (Sommerviller, 2018, p. 89).

Os requisitos não funcionais surgem das necessidades dos usuários, que se devem a restrições orçamentárias, políticas organizacionais, necessidade de interoperabilidade com outros sistemas de software ou hardware, ou fatores externos, como normas de segurança (safety) ou legislação relativa à privacidade (Sommerville, 2018, p. 91).

Os Requisitos Não Funcionais do Sistema HavenNany podem ser diversos e devem ser considerados para garantir a eficiência, segurança e adequação do sistema. Alguns exemplos de Requisitos Não Funcionais que podem ser relevantes (Quadro 2).

Quadro 2 – Requisitos Não Funcionais

|  |  |  |
| --- | --- | --- |
| **ID** | **Requisitos Não Funcionais** | **Descrição** |
| RNF01 | Segurança de Dados | O sistema deve garantir a proteção dos dados pessoais dos usuários e crianças. |
| RNF02 | Usabilidade | A interface do sistema deve ser intuitiva e de fácil navegação para todos os tipos de usuários. |
| RNF03 | Responsividade | O sistema deve funcionar adequadamente em dispositivos móveis, tablets e desktops. |
| RNF04 | Desempenho | O sistema deve garantir tempo de resposta inferior a 3 segundos para operações comuns. |
| RNF05 | Escalabilidade | O sistema deve ser capaz de suportar um número crescente de usuários sem degradação de desempenho. |
| RNF06 | Compatibilidade | O sistema deve ser compatível com os principais navegadores e sistemas operacionais. |
| RNF07 | Acessibilidade | O sistema deve seguir diretrizes de acessibilidade para atender a usuários com necessidades especiais. |

Fonte: Elaborada pelos autores.

# VISÃO DE CASO DE USO – UML

* 1. Diagrama de Classes

O diagrama de classes é um dos componentes mais importantes da UML, exercendo um papel central ao dar suporte à maioria dos outros diagramas. Ele é responsável por definir a estrutura das classes utilizadas no sistema, especificando seus atributos, métodos e as inter- relações entre elas, além de determinar como as classes se conectam e trocam informações entre si (GUEDES, 2018).

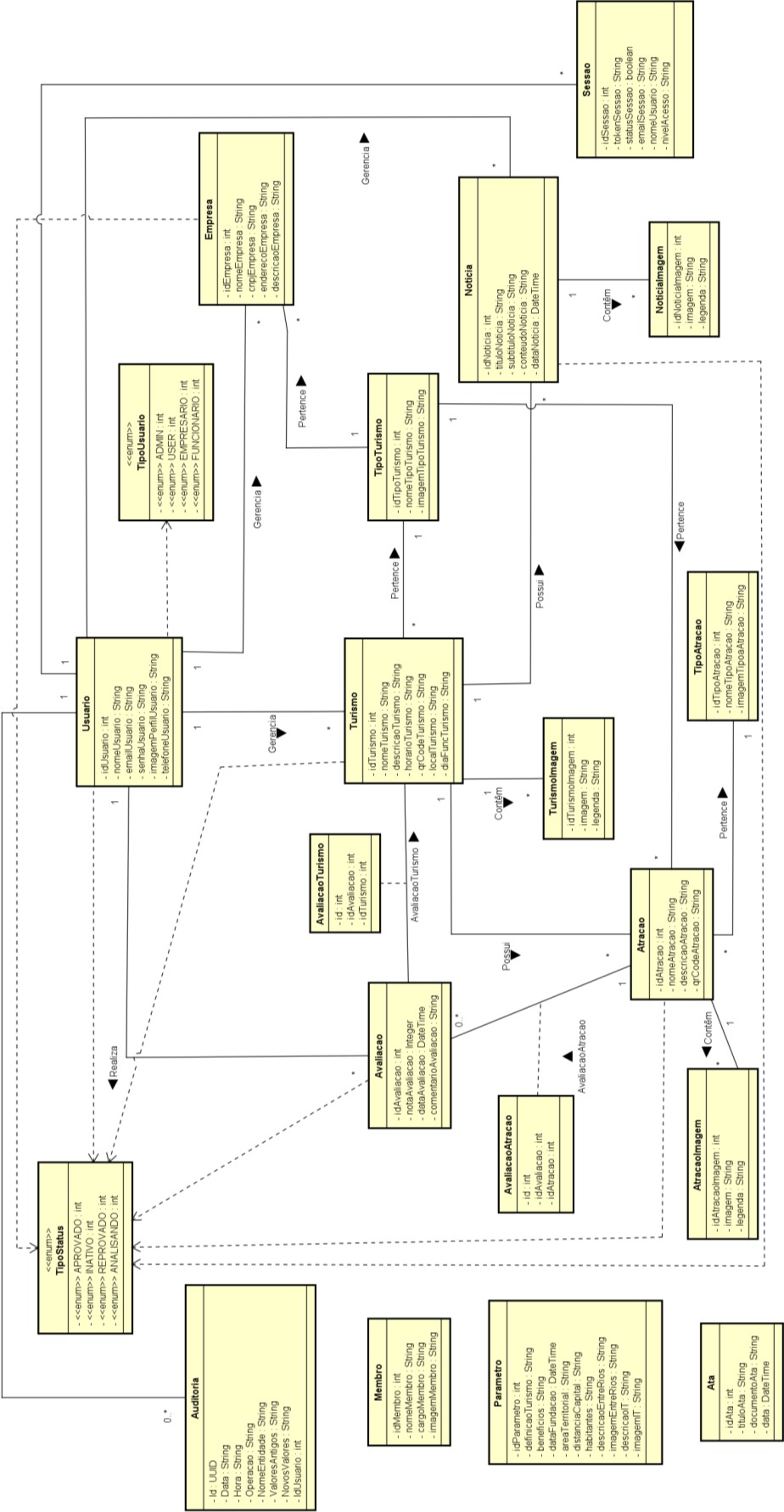
O diagrama de classes desenvolvido para o Comtur, sistema de turismo do município de Jales, define a estrutura lógica das principais entidades e suas interações. Esse modelo representa a regra de negócios do sistema, organizando classes como Usuário, Empresa, Turismo, Atração e Notícia, cada uma com seus respectivos atributos. As associações entre essas classes indicam a forma como os dados se relacionam, como no caso de um Usuário que gerencia uma Empresa, ou o vínculo entre Turismo e TipoTurismo (SOMMERVILLE, 2018). Além das classes principais, o diagrama inclui elementos auxiliares, como Avaliação,

que registra o feedback de usuários, e as classes de imagem vinculadas a empresas, turismos e atrações, como AtracaoImagem, EmpresaImagem e TurismoImagem. A multiplicidade dos relacionamentos define quantas instâncias de uma classe podem estar associadas a outra, oferecendo uma visão clara da dinâmica entre os objetos no sistema (GUEDES, 2018).

Outro componente importante no diagrama são as enumerações, como TipoUsuario e TipoStatus, que delimitam um conjunto específico de valores a serem utilizados no sistema, como funções de usuário (administrador, empresário, funcionário) ou status de aprovação. Conforme Guedes (2018), uma enumeração é um tipo de dado que estabelece um conjunto limitado de valores, representados por literais específicos. Nos metamodelos, sua principal função é listar todos os valores permitidos para determinado tipo de dado, assegurando que somente as opções previamente definidas sejam aplicadas. Embora as enumerações não possuam associações diretas, costumam ser posicionadas próximas às classes que utilizam esses valores, o que contribui para manter a consistência e integridade das informações no sistema.

Demostra-se na Figura 9 a representação gráfica do diagrama de classes, que ilustra a regra de negócio proposta para o sistema Comtur. Esse diagrama visa organizar e visualizar as principais entidades, suas interações e os relacionamentos entre elas, proporcionando uma compreensão clara da estrutura lógica do sistema.

**Figura 9 —** Diagrama de Classes



Fonte: Elaborado pelos autores.

* 1. Dicionário de Classes

Os dicionários de classes são um modelo favorável no processo de análise de requisitos, pois oferecem uma visão abrangente do processamento do sistema de ponta a ponta. Eles contribuem para a identificação dos objetos e das operações realizadas sobre eles, fornecendo informações detalhadas sobre as entidades envolvidas na modelagem e projeção do sistema. Dessa forma, desempenham um papel crucial na compreensão e organização das estruturas e comportamentos necessários para o desenvolvimento da aplicação. (SOMMERVILLE, 2018)

Serão apresentados os dicionários de classes que compõem o sistema Comtur, detalhando os atributos e definindo suas funções dentro da regra de negócio do sistema desenvolvido.

A classe “Turismo” (Quadro 1) tem como objetivo controlar os pontos turísticos inseridos no software, bem como verificar e atualizar as informações que serão apresentadas aos usuários da plataforma.

**Quadro 1 —** Descrição Classe Turismo

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idTurismo | Integer | Código que irá identificar o ponto turismo. |
| nomeTurismo | String | Nome de identificação ao local |
| descricaoTurismo | String | Descrição e guia ao local |
| horarioTurismo | String | Horário de funcionamento |
| qrCodeTurismo | String | Qrcode para projeção e descrição do ponto  turístico visitado. |
| localTurismo | String | Endereço e localização |
| diaFuncTurismo | String | Dias de funcionamento |

Fonte: Elaborado pelos autores.

A classe "Usuário" (Quadro 2) é responsável pelo gerenciamento dos acessos ao sistema. Ela é projetada para facilitar a interação dos usuários com o site, permitindo que utilizem suas diversas ferramentas e funcionalidades. A presente classe abrange a autenticação, o controle de permissões, o registro de atividades e a personalização da experiência do usuário, assegurando um uso seguro e eficiente da plataforma.

**Quadro 2** — Descrição Classe Usuário

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idUsuario | Integer | Código que irá identificar o usuário. |
| nomeUsuario | String | Nome de identificação |
| emailUsuario | String | Endereço de e-mail do usuário |
| senhaUsuario | String | Senha de acesso a plataforma |
| imagemPerfilUsuario | String | Imagem de identificação do usuário com  foto |
| telefoneUsuario | String | Número de contato |

Fonte: Elaborado pelos autores.

A classe "Avaliação" (Quadro 3) é responsável pelo armazenamento e gerenciamento das avaliações e índices de satisfação dos usuários na plataforma. Ela coleta, organiza e exibe feedback, permitindo uma análise detalhada das experiências dos usuários.

**Quadro 3 —** Descrição Classe Avaliação

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idAvaliacao | Integer | Código que irá identificar avaliação do  usuário |
| notaAvaliacao | Integer | Nota em numeração |
| dataAvaliacao | DateTime | Data da avalição |
| comentarioAvaliacao | String | Comentário do usuário |

Fonte: Elaborado pelos autores.

A classe "Notícia" (Quadro 4) é responsável pelo gerenciamento e organização de notícias relacionadas à cidade, eventos e pontos turísticos. Ela coleta, armazena e exibe informações atualizadas, garantindo que os usuários tenham acesso a conteúdos relevantes e recentes.

**Quadro 4** — Descrição Classe Notícia

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idNoticia | Integer | Código que irá identificar notícia |
| tituloNoticia | String | Título de referenciamento de assunto |
| subtituloNoticia | String | Subtítulo de referenciamento de assunto |
| conteudoNoticia | String | Assunto da notícia |
| dataNoticia | DataTime | Data de publicação |

Fonte: Elaborado pelos autores.

A classe "Tipo Turismo" (Quadro 5) é responsável por identificar e categorizar a natureza dos pontos turísticos adicionados ao site, como pontes, praças e outros. Ela classifica os diferentes tipos de atrações turísticas, facilitando a organização e a busca por informações específicas.

**Quadro 5** — Descrição Classe Tipo Turismo

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idTipoTurismo | Integer | Código que irá identificar tipo turismo |
| nomeTipoTurismo | String | Nome de identificação |
| imagemTipoTurismo | String | Imagem de identificação |

Fonte: Elaborado pelos autores.

A classe "Atração" (Quadro 6) é responsável pelo gerenciamento das atrações associadas aos eventos ou pontos turísticos do site. Ela organiza e mantém informações detalhadas sobre cada atração, facilitando a administração e a apresentação desses dados aos usuários.

**Quadro 6 —** Descrição Classe Atração

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idAtracao | Integer | Código que irá identificar a atração |
| nomeAtracao | String | Nome de identificação |
| descricaoAtracao | String | Descrição da atração |
| qrCodeAtracao | String | Qr Code utilizado para projetar e fornecer  informações sobre a atração visitada. |

Fonte: Elaborado pelos autores.

A classe "Tipo Atração" (Quadro 7) é responsável por identificar e categorizar a natureza das atrações adicionadas ao site, como circos, shows, e outros tipos de entretenimento. Ela classifica cada atração de acordo com suas características específicas, facilitando a organização e a busca por informações detalhadas.

**Quadro 7 —** Descrição Classe Tipo Atração

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idTipoAtracao | Integer | Código que irá identificar tipo atração |
| nomeTipoAtracao | String | Nome de identificação |
| imagemTipoAtracao | String | Imagem de identificação |

Fonte: Elaborado pelos autores.

A classe “Empresa” (Quadro 8) é responsável pelo gerenciamento das empresas parceiras, facilitando a interação destas com o site e a divulgação de seus estabelecimentos. Ela organiza informações detalhadas sobre cada empresa, mantendo-as atualizadas e acessíveis aos usuários. Essa classe garante que as parcerias sejam bem administradas, permitindo que os serviços e produtos fornecidos pelos parceiros sejam divulgados eficientemente na plataforma.

**Quadro 8 —** Descrição Classe Empresa

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idEmpresa | Integer | Código que irá identificar empresa |
| nomeEmpresa | String | Razão social da empresa |
| cnpjEmpresa | String | Cadastro Nacional da Pessoa Jurídica |
| enderecoEmpresa | String | Endereço da empresa |
| descricaoEmpresa | String | Atuação da empresa |

Fonte: Elaborado pelos autores.

A classe "Atração Imagem" (Quadro 9) é responsável por gerenciar as imagens associadas a cada atração no site. Ela armazena e organiza as imagens relevantes para cada atração, permitindo uma representação visual atraente e informativa para os usuários.

**Quadro 9 —** Descrição Classe Atração Imagem

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idAtracaoImagem | Integer | Código que irá identificar imagem |
| imagem | String | Imagem da atração exibida |

|  |  |  |
| --- | --- | --- |
| legando | String | Legenda referente a imagem exibida |

Fonte: Elaborado pelos autores.

A classe "Turismo Imagem" (Quadro 10) tem a responsabilidade de gerenciar as imagens associadas aos pontos turísticos presentes no site. Seu objetivo é armazenar e organizar imagens relevantes para cada local turístico, proporcionando aos usuários uma representação visual completa e atrativa.

**Quadro 10 —** Descrição Classe Turismo Imagem

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idTurismoImagem | Integer | Código que irá identificar imagem |
| imagem | String | Imagem do ponto turístico exibido |
| legenda | String | Legenda referente a imagem exibida |

Fonte: Elaborado pelos autores.

A classe “Notícia Imagem” (Quadro 11) assegura que as imagens estejam corretamente associadas às notícias correspondentes, contribuindo para uma experiência informativa e visualmente atraente ao explorar as notícias disponíveis na plataforma.

**Quadro 11 —** Descrição Classe Notícia Imagem

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idNoticiaImagem | Integer | Código que irá identificar imagem |
| imagem | String | Imagem de referência da notícia exibida |
| legenda | String | Legenda referente a imagem exibida |

Fonte: Elaborado pelos autores.

A classe “Empresa Imagem” (Quadro 12) visa gerenciar as imagens que representam visualmente as empresas parceiras no site. Ela armazena e organiza as imagens relevantes para cada organização, garantindo uma apresentação visual adequada de cada negócio.

**Quadro 12 —** Descrição Classe Empresa Imagem

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idEmpresaImagem | Integer | Código que irá identificar imagem |
| imagem | String | Imagem da empresa parceira |
| leganda | String | Legenda referente a imagem exibida |

Fonte: Elaborado pelos autores.

A classe “Tipo Usuário” (Quadro 13) é uma enumeração que define os diferentes papéis que os usuários podem ter no sistema, permitindo o controle de acesso e a aplicação de funcionalidades específicas com base no tipo de usuário que está acessando.

**Quadro 13 —** Descrição Enum Tipo Usuário — Enumeração dos tipos de usuário

|  |  |  |
| --- | --- | --- |
| Enumerações | Tipo | Descrição |
| ADMIN | Integer | Representa um usuário com privilégios  administrativos no sistema. (4) |
| USER | Integer | Representa um usuário comum do sistema,  com acesso básico às funcionalidades. (1) |
| EMPRESARIO | Integer | Representa um empresário que possui uma  empresa parceira cadastrada no sistema. (3) |
| FUNCIONARIO | Integer | Representa um funcionário associado a uma  empresa parceira cadastrada no sistema. (2) |

Fonte: Elaborado pelos autores.

A classe "Tipo Status" (Quadro 14) é uma enumeração que oferece uma forma de representar os diferentes estados que uma entidade pode assumir no sistema, facilitando o acompanhamento e o gerenciamento de seu status ao longo do tempo.

**Quadro 14 —** Descrição Enum Tipo Status — Enumeração de status

|  |  |  |
| --- | --- | --- |
| Enumerações | Tipo | Descrição |
| APROVADO | Integer | Representa o status de aprovação de uma  entidade no sistema. (2) |
| DESATIVADO | Integer | Representa o status de inatividade de uma  entidade no sistema. (4) |
| REPROVADO | Integer | Representa o status de reprovação de uma  entidade no sistema. (3) |
| ANALISANDO | Integer | Representa o status de análise em  andamento de uma entidade no sistema. (1) |

Fonte: Elaborado pelos autores.

A classe "Auditoria" (Quadro 15), é fundamental para o registro e controle das operações realizadas no sistema. Cada atributo especificado fornece detalhes cruciais sobre as

alterações efetuadas, incluindo a data e hora da auditoria, a operação realizada, a entidade afetada, os valores antigos e novos, além da identificação do usuário que autorizou a atualização.

**Quadro 15 —** Descrição Classe Empresa Auditoria

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| id | Integer | Código que irá identificar a auditoria |
| data | String | Data da auditoria |
| hora | String | Horário da auditoria |
| operacao | String | Operação que foi realizada |
| nomeEntidade | String | Entidade que foi sujeita a inserção, remoção  ou edição |
| valorAntigo | String | Informação substituída |
| novosValores | String | Novo valor atualizado |
| idUsuario |  | Autor da que autorizou a atulização |

Fonte: Elaborado pelos autores.

A classe “Membro” (Quadro 16), representa os integrantes da prefeitura que participam do conselho. Os atributos listados fornecem informações essenciais sobre cada membro, incluindo o nome, o cargo ocupado e sua foto de identificação.

**Quadro 16 —** Descrição da Classe Membros

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idMembro | Integer | Código que irá identificar o Membro |
| nomeMembro | String | Nome do membro da prefeitura que  participa do conselho |
| cargoMembro | String | Cargo que o membro atua |
| imagemMembro | String | Imagem de identificação do membro |

Fonte: Elaborado pelos autores.

A classe “Ata” (Quadro 17) representa os registros formais das reuniões do Conselho Municipal de Turismo (Comtur). Os atributos listados fornecem informações essenciais sobre cada ata, incluindo um identificador único, o título da ata, o anexo do documento correspondente e a data em que a ata foi apresentada.

**Quadro 17 —** Descrição da Classe Ata

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idAta | Integer | Código que irá identificar a Ata |
| tituloAta | String | Título da Ata |
| documentoAta | String | Anexo do documento de Ata |
| data | DataTime | Data da Ata apresentada |

Fonte: Elaborado pelos autores.

A classe “Parâmetros” (Quadro 18) é responsável pelos atributos da tela inicial do site, permitindo que o administrador edite os dados apresentados. Os atributos listados incluem informações essenciais, benefícios, data da fundação da cidade, área territorial, distância até a capital, população e uma descrição da região turística do Entre Rios, esses dados serão apresentados em cards na tela “home”.

**Quadro 18 —** Descrição da Classe Parâmetros

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idParametro | Integer | Código que irá identificar o parâmetro |
| definicaoTurismo | String | Definição do turismo apresentado |
| beneficios | String | Apresentação de benefícios (componente  tela home do projeto) |
| dataFundacao | DataTime | Data da fundação da cidade (componente  tela home do projeto) |
| areaTerritorial | String | Área territorial da cidade |
| distanciaCapital | String | Distância entre a cidade de Jales até a São  Paulo |
| habitantes | String | Quantidade de habitantes da cidade |
| descricaoEntreRios | String | Descrição da região turísticas que se  encontram no projeto do Entre Rios |
| imagemEntreRios | String | Imagem de identificação |
| descricaoIT | String | Descrição Interesse Turístico |
| imagemIT | String | Imagem Interesse Turístico |

Fonte: Elaborado pelos autores.

A classe “Sessão” (Quadro 19) contém os atributos relacionados à gestão das sessões de usuário no sistema. Os atributos listados incluem informações essenciais, como o código identificador da sessão, o token utilizado para validação de acesso, o status da sessão, além de dados do utente que está acessando, como e-mail e nome. O nível de acesso é um componente importante, pois determina as permissões e funcionalidades disponíveis ao usuário dentro da plataforma.

**Quadro 19 —** Descrição da Classe Sessão

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| idSessao | Integer | Código que irá identificar a sessão |
| tokenSessao | String | Token para fazer a validação de acesso |
| statusSessao | Boolean | Status da sessão |
| emailSessao | String | E-mail do usuário que está acessando |
| nomeUsuario | String | Nome do usuário que está acessando |
| nivelAcesso | String | Que determina suas permissões e  funcionalidades disponíveis dentro do sistema. |

Fonte: Elaborado pelos autores.

A classe relacional “Avaliação Atração” (Quadro 20) é responsável pelos atributos que definem a relação entre as avaliações dos usuários e as atrações disponíveis no sistema.

**Quadro 20 —** Descrição da Classe Relacional Avaliação Atração

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |
| id | Integer | Código que irá identificar Avaliação  Atração |
| idAvaliação | Integer | Recebe o id de Avaliação |
| idAtracao | Integer | Recebe o id de Atração |

Fonte: Elaborado pelos autores.

A classe relacional “Avaliação Turismo” (Quadro 21) é responsável pelos atributos que definem a relação entre as avaliações dos usuários e os turismos disponíveis no sistema.

**Quadro 21 —** Descrição da Classe Relacional Avaliação Turismo

|  |  |  |
| --- | --- | --- |
| Atributo | Tipo | Descrição |

|  |  |  |
| --- | --- | --- |
| id | Integer | Código que irá identificar Avaliação  Turismo |
| idAvaliação | Integer | Recebe o id de Avaliação |
| idTurismo | Integer | Recebe o id do Turismo |

Fonte: Elaborado pelos autores.

* 1. Definição dos Atores

Os atores podem ser representados por usuários, dispositivos de hardware ou sistemas externos que interagem diretamente com as funcionalidades de um sistema. Na notação UML, sua representação gráfica utiliza ícones que simbolizam pessoas, acompanhados de breves descrições que explicam seus papéis. Esse formato visual simplifica a compreensão das interações e funções desempenhadas pelos diferentes elementos envolvidos no sistema (GUEDES, 2018).

Neste trabalho, a definição de cada ator destaca suas funções no escopo da aplicação, o que torna o processo de análise mais simples para o desenvolvimento de modelagem. Com base nisso, a definição dos autores assegura uma representação das interações entre os elementos do sistema.

No âmbito da aplicação, o usuário exerce a função principal de interagir com a plataforma, acessando e navegando por listagens de empresas parceiras, atrações, tipos de turismo, eventos, notícias e anúncios, além de poder, realizar avaliações.

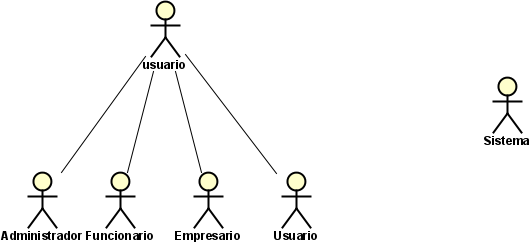
Por sua vez, o empresário tem a responsabilidade de inserir e manter atualizadas as informações relativas à sua empresa e das atividades turísticas que oferece, assegurando a correta apresentação desses dados.

O funcionário, vinculado a uma empresa parceira ou à prefeitura, dedica-se à gestão e atualização das informações e acessos pertinentes à sua função.

Já o administrador desempenha um papel fundamental na supervisão e manutenção da integridade do sistema, garantindo que as informações apresentadas sejam precisas e atualizadas, além de zelar pela segurança dos dados, monitorando e prevenindo atividades maliciosas.

Demostra-se na Figura 10 a representação gráfica do diagrama de atores, que ilustra as interações dos diferentes usuários e sistemas com a plataforma. Essa visualização facilita a compreensão das funções e responsabilidades de cada ator no contexto da aplicação.

**Figura 10 —** Diagrama de Atores



Fonte: Elaborado pelos autores.

* 1. Lista de Casos de Uso

Guedes (2018) destaca que os casos de uso são fundamentais para a identificação dos requisitos de um sistema. Eles se referem aos serviços, tarefas ou funcionalidades que o software deve oferecer e que podem ser utilizados pelos atores que interagem com ele. Essas representações ajudam a expressar e documentar os comportamentos esperados para as diversas funções do software. Além disso, os requisitos podem ser registrados por meio de diferentes diagramas, como os de sequência, máquina de estados e atividades, entre outros. Para assegurar a padronização das informações, foram elaboradas duas listas: uma com as mensagens do sistema (Quadro 22) e outra com os casos de uso do software (Quadro 23).

A lista de mensagens do sistema tem como finalidade detalhar as interações entre os componentes do software e os usuários durante a execução dos casos de uso, garantindo clareza e consistência nas comunicações.

**Quadro 22 —** Lista de Mensagens

|  |  |
| --- | --- |
| **Nº da Mensagem** | **Conteúdo da Mensagem** |
| Msg 1 | Cadastrado com sucesso |
| Msg 2 | Desativado com sucesso |
| Msg 3 | Excluído com sucesso |
| Msg 4 | Alterado com sucesso |
| Msg 5 | Login realizado com sucesso |

|  |  |
| --- | --- |
| Msg 6 | Campos em branco |
| Msg 7 | Erro ao conectar com o banco |
| Msg 8 | Esse tipo de turismo já existe |
| Msg 9 | Boas-vindas |
| Msg 10 | Email ou senha incorretos |

Fonte: Elaborado pelos autores.

A lista de casos de uso é uma ferramenta essencial no desenvolvimento de software, pois descreve como os atores interagem com o sistema para alcançar objetivos específicos. Cada caso de uso detalha um conjunto de ações e ajuda a definir os requisitos funcionais do software. Essa lista oferece uma visão estruturada do funcionamento do sistema, orienta os desenvolvedores na implementação e facilita a comunicação com os stakeholders. O Quadro 23 apresenta a lista completa de casos de uso do sistema.

**Quadro 23** — Lista de casos de uso do software

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nº** | **Descrição do Caso de Uso** | **Entrada** | **Caso de Uso** | **Resposta** |
| 1 | Administrador realiza login | Informa dados de acesso | Efetuar login | Msg 5 |
| 2 | Administrador altera administrador | Informa dados administrador | Alterar administrador | Msg 4 |
| 3 | Administrador altera empresário | Id empresário | Alterar empresário | Msg 4 |
| 4 | Administrador altera usuário | Informa dados usuário | Alterar usuário | Msg 4 |
| 5 | Administrador altera funcionário | Informa dados funcionário | Alterar funcionário | Msg 4 |
| 6 | Administrador altera avaliação | Informa dados de avaliação | Alterar avaliação | Msg 4 |
| 7 | Administrador altera empresa | Informa dados de empresa | Alterar empresa | Msg 4 |
| 8 | Administrador altera turismo | Informa dados de turismo | Alterar turismo | Msg 4 |
| 9 | Administrador altera atração | Informa dados de atração | Alterar atração | Msg 4 |
| 10 | Administrador altera notícia | Informa dados de notícia | Alterar notícia | Msg 4 |
| 11 | Administrador altera tipo turismo | Informa dados tipo turismo | Altera tipo turismo | Msg 4 |
| 12 | Administrador altera tipo atração | Informa dados tipo atração | Altera tipo atração | Msg 4 |
| 13 | Administrador cadastra avaliação | Informa dados de avaliação | Cadastrar avaliação | Msg 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nº** | **Descrição do Caso de Uso** | **Entrada** | **Caso de Uso** | **Resposta** |
| 14 | Administrador cadastra turismo | Informa dados de turismo | Cadastrar turismo | Msg 1 |
| 15 | Administrador cadastra atração | Informa dados de atração | Cadastrar atração | Msg 1 |
| 16 | Administrador cadastra administrador | Informa dados de administrador | Cadastrar administrador | Msg 1 |
| 17 | Administrador cadastra usuário | Informa dados usuário | Cadastrar usuário | Msg 1 |
| 18 | Administrador cadastra funcionário | Informa dados de funcionário | Cadastrar funcionário | Msg 1 |
| 19 | Administrador cadastra empresário | Informa dados de empresário | Cadastrar empresário | Msg 1 |
| 20 | Administrador cadastra notícia | Informa dados de notícia | Cadastrar notícia | Msg 1 |
| 21 | Administrador cadastra empresa | Informa dados de empresa | Cadastrar empresa | Msg 1 |
| 22 | Administrador desativa administrador | Id administrador | Desativa administrador | Msg 2 |
| 23 | Administrador desativa tipo atração | Id tipo atração | Desativa tipo atração | Msg 2 |
| 24 | Administrador desativa turismo | Id turismo | Desativa turismo | Msg 2 |
| 25 | Administrador desativa atração | Id atração | Desativa atração | Msg 2 |
| 26 | Administrador desativa empresa | Id empresa | Desativa empresa | Msg 2 |
| 27 | Administrador desativa tipo turismo | Id tipo turismo | Desativa tipo turismo | Msg 2 |
| 28 | Administrador desativa avaliação | Id avaliação | Desativa avaliação | Msg 2 |
| 29 | Administrador desativa usuário | Informa dados usuário | Desativa usuário | Msg 2 |
| 30 | Administrador desativa empresário | Id empresário | Desativa empresário | Msg 2 |
| 31 | Administrador desativa funcionário | Id funcionário | Desativa funcionário | Msg 2 |
| 32 | Administrador desativa notícia | Id notícia | Desativar notícia | Msg 2 |
| 33 | Administrador exclui empresa | Id empresa | Excluir empresa | Msg 3 |
| 34 | Administrador exclui administrador | Id administrador | Excluir administrador | Msg 3 |
| 35 | Administrador exclui usuário | Id usuário | Excluir usuário | Msg 3 |
| 36 | Administrador exclui empresário | Id empresário | Excluir empresário | Msg 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nº** | **Descrição do Caso de Uso** | **Entrada** | **Caso de Uso** | **Resposta** |
| 37 | Administrador exclui funcionário | Id funcionário | Excluir funcionário | Msg 3 |
| 38 | Administrador exclui avaliação | Id avaliação | Excluir avaliação | Msg 3 |
| 39 | Administrador exclui atração | Id atração | Excluir atração | Msg 3 |
| 40 | Administrador exclui turismo | Id turismo | Excluir turismo | Msg 3 |
| 41 | Administrador exclui notícia | Id tipo notícia | Excluir notícia | Msg 3 |
| 42 | Administrador exclui tipo turismo | Id tipo turismo | Excluir tipo turismo | Msg 3 |
| 43 | Administrador exclui tipo atração | Id tipo atração | Excluir tipo atração | Msg 3 |
| 44 | Administrador consulta notícia | Id notícia | Consultar notícia | Retorna dados notícia |
| 45 | Administrador consulta avaliação | Id avaliação | Consultar avaliação | Retorna dados avaliação |
| 46 | Administrador consulta atração | Id atração | Consultar atração | Retorna dados atração |
| 47 | Administrador consulta turismo | Id turismo | Consultar turismo | Retorna dados turismo |
| 48 | Administrador consulta empresa | Id empresa | Consultar empresa | Retorna dados empresa |
| 49 | Administrador consulta administrador | Id administrador | Consultar administrador | Retorna dados administrador |
| 50 | Administrador consulta empresário | Id empresário | Consultar empresário | Retorna dados  empresário |
| 51 | Administrador consulta funcionário | Id funcionário | Consultar funcionário | Retorna dados funcionário |
| 52 | Administrador consulta usuário | Id usuário | Consultar usuário | Retorna dados usuário |
| 53 | Administrador consulta tipo turismo | Id tipo turismo | Consultar tipo turismo | Retorna dados tipo turismo |
| 54 | Administrador consulta tipo atração | Id tipo atração | Consultar tipo atração | Retorna dados tipo atração |
| 55 | Administrador solicita lista de turismo | Id turismo | Listar turismo | Retorna lista turismo |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nº** | **Descrição do Caso de Uso** | **Entrada** | **Caso de Uso** | **Resposta** |
| 56 | Administrador solicita lista de usuário | Id tipo usuário | Listar tipo usuário | Retorna lista tipo usuário |
| 57 | Administrador solicita lista de notícia | Id tipo notícia | Listar tipo notícia | Retorna lista notícia |
| 58 | Administrador solicita lista de tipo atração | Id tipo atração | Listar tipo atração | Retorna lista tipo atração |
| 59 | Administrador solicita lista de tipo turismo | Id tipo turismo | Listar tipo turismo | Retorna lista tipo turismo |
| 60 | Administrador solicita lista de atração | Id atração | Listar atração | Retorna lista atração |
| 61 | Administrador solicita lista de empresa | Id empresa | Listar empresa | Retorna lista empresa |
| 62 | Administrador solicita lista de avaliação | Id avaliação | Listar avaliação | Retorna lista avaliação |
| 63 | Funcionário realiza login | Informa dados de acesso | Efetuar login | Msg 5 |
| 64 | Funcionário altera funcionário | Informa dados funcionário | Alterar funcionário | Msg 4 |
| 65 | Funcionário altera usuário comum | Informa dados usuário comum | Alterar usuário comum | Msg 4 |
| 66 | Funcionário altera empresário | Informa dados empresário | Alterar empresário | Msg 4 |
| 67 | Funcionário altera empresa | Informa dados de empresa | Alterar empresa | Msg 4 |
| 68 | Funcionário altera turismo | Informa dados de turismo | Alterar turismo | Msg 4 |
| 69 | Funcionário altera atração | Informa dados de atração | Alterar atração | Msg 4 |
| 70 | Funcionário altera notícia | Informa dados de notícia | Alterar notícia | Msg 4 |
| 71 | Funcionário altera tipo turismo | Informa dados tipo turismo | Altera tipo turismo | Msg 4 |
| 72 | Funcionário altera tipo atração | Informa dados tipo atração | Altera tipo atração | Msg 4 |
| 73 | Funcionário cadastra avaliação | Informa dados de avaliação | Cadastrar avaliação | Msg 1 |
| 74 | Funcionário cadastra turismo | Informa dados de turismo | Cadastrar turismo | Msg 1 |
| 75 | Funcionário cadastra atração | Informa dados de atração | Cadastrar atração | Msg 1 |
| 76 | Funcionário cadastra funcionário | Informa dados de funcionário | Cadastrar funcionário | Msg 1 |
| 77 | Funcionário cadastra usuário comum | Informa dados usuário comum | Cadastrar usuário comum | Msg 1 |
| 78 | Funcionário cadastra empresário | Informa dados empresário | Cadastrar empresário | Msg 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nº** | **Descrição do Caso de Uso** | **Entrada** | **Caso de Uso** | **Resposta** |
| 79 | Funcionário cadastra notícia | Informa dados de notícia | Cadastrar notícia | Msg 1 |
| 80 | Funcionário cadastra empresa | Informa dados de empresa | Cadastrar empresa | Msg 1 |
| 81 | Funcionário desativa funcionário | Id funcionário | Desativa funcionário | Msg 2 |
| 82 | Funcionário desativa avaliação | Id avaliação | Desativa avaliação | Msg 2 |
| 83 | Funcionário desativa usuário | Informa dados usuário | Desativa usuário | Msg 2 |
| 84 | Funcionário desativa empresário | Informa dados empresário | Desativa empresário | Msg 2 |
| 85 | Funcionário desativa administrador | Informa dados administrador | Desativa administrador | Msg 2 |
| 86 | Funcionário desativa tipo turismo | Informa dados tipo turismo | Desativa tipo turismo | Msg 2 |
| 87 | Funcionário desativa turismo | Informa dados turismo | Desativa turismo | Msg 2 |
| 88 | Funcionário desativa atração | Informa dados atração | Desativa atração | Msg 2 |
| 89 | Funcionário desativa tipo atração | Informa dados tipo atração | Desativa tipo atração | Msg 2 |
| 90 | Funcionário desativa empresa | Informa dados empresa | Desativa empresa | Msg 2 |
| 91 | Funcionário desativa notícia | Id notícia | Desativar notícia | Msg 2 |
| 92 | Funcionário consulta notícia | Id notícia | Consultar notícia | Retorna dados notícia |
| 93 | Funcionário consulta avaliação | Id avaliação | Consultar avaliação | Retorna dados avaliação |
| 94 | Funcionário consulta atração | Id atração | Consultar atração | Retorna dados atração |
| 95 | Funcionário consulta turismo | Id turismo | Consultar turismo | Retorna dados turismo |
| 96 | Funcionário consulta empresa | Id empresa | Consultar empresa | Retorna dados empresa |
| 97 | Funcionário consulta funcionário | Id funcionário | Consultar funcionário | Retorna dados funcionário |
| 98 | Funcionário consulta usuário | Id usuário | Consultar usuário | Retorna dados usuário |
| 99 | Funcionário consulta empresário | Id empresário | Consultar empresário | Retorna  dados empresário |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nº** | **Descrição do Caso de Uso** | **Entrada** | **Caso de Uso** | **Resposta** |
| 100 | Funcionário consulta tipo turismo | Id tipo turismo | Consultar tipo turismo | Retorna dados tipo turismo |
| 101 | Funcionário consulta tipo atração | Id tipo atração | Consultar tipo atração | Retorna dados tipo atração |
| 102 | Funcionário solicita lista de turismo | Id turismo | Listar turismo | Retorna lista turismo |
| 103 | Funcionário solicita lista de usuário | Id tipo usuário | Listar tipo usuário | Retorna lista tipo usuário |
| 104 | Funcionário solicita lista de notícia | Id tipo notícia | Listar tipo notícia | Retorna lista notícia |
| 105 | Funcionário solicita lista de tipo atração | Id tipo atração | Listar tipo atração | Retorna lista tipo atração |
| 106 | Funcionário solicita lista de tipo turismo | Id tipo turismo | Listar tipo turismo | Retorna lista tipo turismo |
| 107 | Funcionário solicita lista de atração | Id atração | Listar atração | Retorna lista atração |
| 108 | Funcionário solicita lista de empresa | Id empresa | Listar empresa | Retorna lista empresa |
| 109 | Funcionário solicita lista de avaliação | Id avaliação | Listar avaliação | Retorna lista avaliação |
| 110 | Empresário realiza login | Informa dados de acesso | Efetuar login | Msg 5 |
| 111 | Empresário cadastra empresário | Informa dados do empresário | Cadastrar empresário | Msg 1 |
| 112 | Empresário cadastra avaliação | Informa dados de avaliação | Cadastrar avaliação | Msg 1 |
| 113 | Empresário cadastra empresa | Informa dados da empresa | Cadastrar empresa | Msg 1 |
| 114 | Empresário exclui avaliação | Id avaliação | Excluir avaliação | Msg 3 |
| 115 | Empresário exclui empresa | Id empresa | Excluir empresa | Msg 3 |
| 116 | Empresário exclui empresário | Id empresário | Excluir empresário | Msg 3 |
| 117 | Empresário altera empresário | Informa dados do empresário | Altera empresário | Msg 4 |
| 118 | Empresário altera empresa | Informa dados da empresa | Alterar empresa | Msg 4 |
| 119 | Empresário altera avaliação | Informa dados da avaliação | Alterar avaliação | Msg 4 |
| 120 | Empresário consultar empresa | Id empresa | Consultar empresa | Retorna dados de empresa |
| 121 | Empresário consultar avaliação | Id avaliação | Consultar avaliação | Retorna  dados de avaliação |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nº** | **Descrição do Caso de Uso** | **Entrada** | **Caso de Uso** | **Resposta** |
| 122 | Empresário consultar turismo | Id turismo | Consultar turismo | Retorna dados de turismo |
| 123 | Empresário consultar tipo turismo | Id tipo turismo | Consultar tipo turismo | Retorna dados de tipo turismo |
| 124 | Empresário consultar usuário | Id usuário | Consultar usuário | Retorna dados de usuário |
| 125 | Empresário consultar notícia | Id notícia | Consultar notícia | Retorna dados de notícia |
| 126 | Empresário consultar atração | Id atração | Consultar atração | Retorna dados de atração |
| 127 | Empresário solicita lista de avaliação | Id avaliação | Listar avaliação | Retorna lista avaliação |
| 128 | Empresário solicita lista de empresa | Id empresa | Listar empresa | Retorna lista empresa |
| 129 | Empresário solicita lista de atração | Id atração | Listar atração | Retorna lista atração |
| 130 | Empresário solicita lista de notícia | Id tipo notícia | Listar tipo notícia | Retorna lista notícia |
| 131 | Empresário solicita lista de turismo | Id turismo | Listar turismo | Retorna lista turismo |
| 132 | Usuário cadastra avaliação | Informa dados da avaliação | Cadastrar avaliação | Msg 1 |
| 133 | Usuário cadastra usuário | Informa dados usuário | Cadastrar usuário | Msg 1 |
| 134 | Usuário exclui usuário | Id usuário | Excluir usuário | Msg 4 |
| 135 | Usuário consultar empresa | Id empresa | Consultar empresa | Retorna dados de  empresa |
| 136 | Usuário consultar notícia | Id notícia | Consultar notícia | Retorna dados da notícia |
| 137 | Usuário consultar atração | Id atração | Consultar atração | Retorna dados de atração |
| 138 | Usuário consultar turismo | Id turismo | Consultar turismo | Retorna dados de turismo |
| 139 | Usuário consultar usuário | Id usuário | Consultar usuário | Retorna dados do usuário |
| 140 | Usuário realiza login | Informa dados de acesso | Efetuar login | Msg 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nº** | **Descrição do Caso de Uso** | **Entrada** | **Caso de Uso** | **Resposta** |
| 141 | Usuário altera avaliação | Informa dados avaliação | Alterar avaliação | Msg 4 |
| 142 | Usuário altera usuário | Informa dados do usuário | Alterar usuário | Msg 4 |
| 143 | Usuário solicita lista de avaliação | Id avaliação | Listar avaliação | Retorna lista avaliação |
| 144 | Usuário solicita lista de empresa | Id empresa | Listar empresa | Retorna lista empresa |
| 145 | Usuário solicita lista de atração | Id atração | Listar atração | Retorna lista atração |
| 146 | Usuário solicita lista de notícia | Id tipo notícia | Listar tipo notícia | Retorna lista notícia |
| 147 | Usuário solicita lista de turismo | Id turismo | Listar turismo | Retorna lista turismo |

Fonte: Elaborado pelos autores.

* 1. Diagrama de Casos de Uso

O diagrama de Caso de Uso é o mais abstrato, portanto, o mais flexível e informal, utilizado para representar uma visão abrangente do sistema. Seu propósito é identificar os atores que interagirão com o software, juntamente com suas respectivas funcionalidades. Este diagrama é comumente empregado durante as fases de levantamento e análise de requisitos, embora sua aplicação se estenda ao longo de todo o processo de modelagem, podendo inclusive servir como fundamento para a elaboração de outros diagramas (GUEDES, 2018).

Nas subseções seguintes, serão apresentados os diagramas de caso de uso, baseados nos diferentes atores que interagem com o sistema. Cada diagrama ilustra as interações e responsabilidades de um ator específico, detalhando como cada um contribui para o funcionamento geral do sistema.

O Administrador é o ator responsável por gerenciar o sistema de forma global, abrangendo funcionalidades e dados. Suas atribuições incluem a criação, manutenção, definição de permissões e monitoramento das atividades.

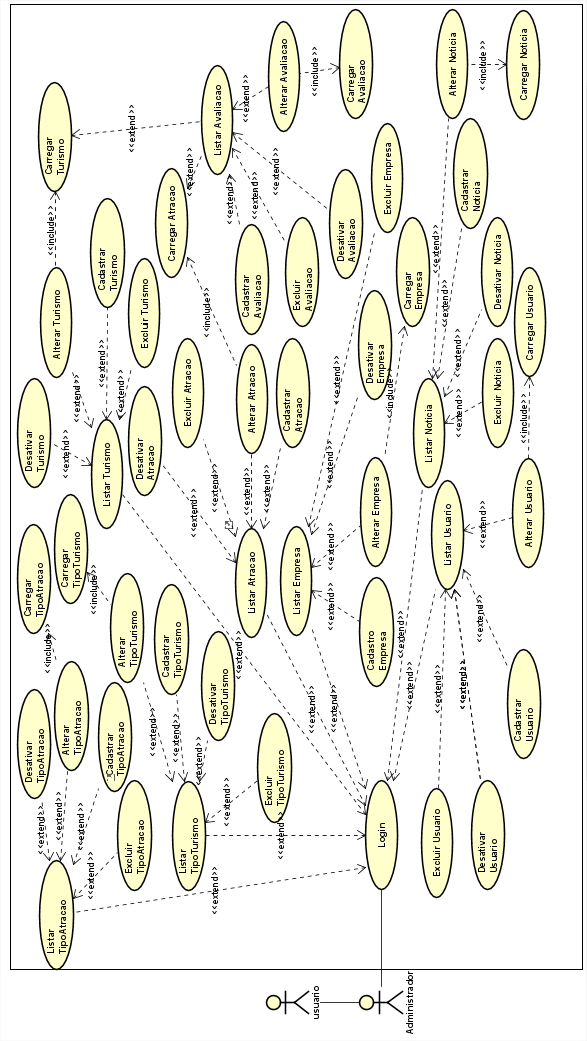
O Funcionário é o ator que realiza tarefas operacionais. Dentre suas funcionalidades, estão a inserção e manutenção dos dados gerais do sistema. Além disso, o funcionário é responsável por modificar registros existentes, realizando ações como alterar ou desativar dados de usuários, empresas e avaliações, conforme necessário.

O Empresário é o ator responsável por cadastrar e gerenciar todas as informações relacionadas à sua empresa dentro do sistema. Isso inclui o registro e a atualização de dados

empresariais, bem como a gestão de atrações e turismo associadas à empresa. Essas funções permitem ao empresário manter a precisão das informações.

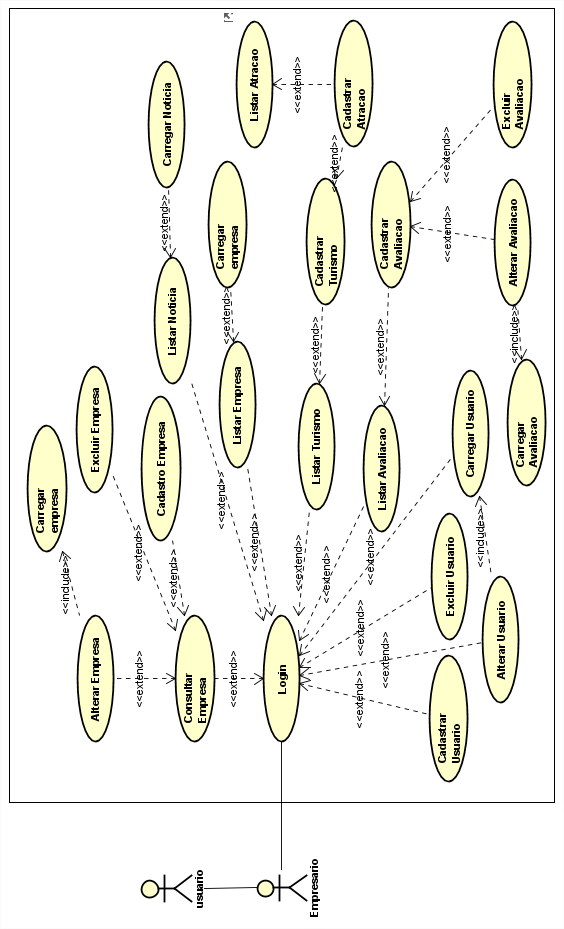
O Usuário é o ator que interage diretamente com o sistema para usufruir dos serviços e funcionalidades oferecidas. Suas interações estão focadas na utilização das funções disponíveis, como fazer login, acessar seu perfil, ou utilizar os serviços específicos do sistema. O diagrama de caso de uso do usuário destaca essas interações típicas, evidenciando como ele acessa e navega pelas funcionalidades.

**Figura 11 —** Diagrama de caso de uso - Visão Administrador



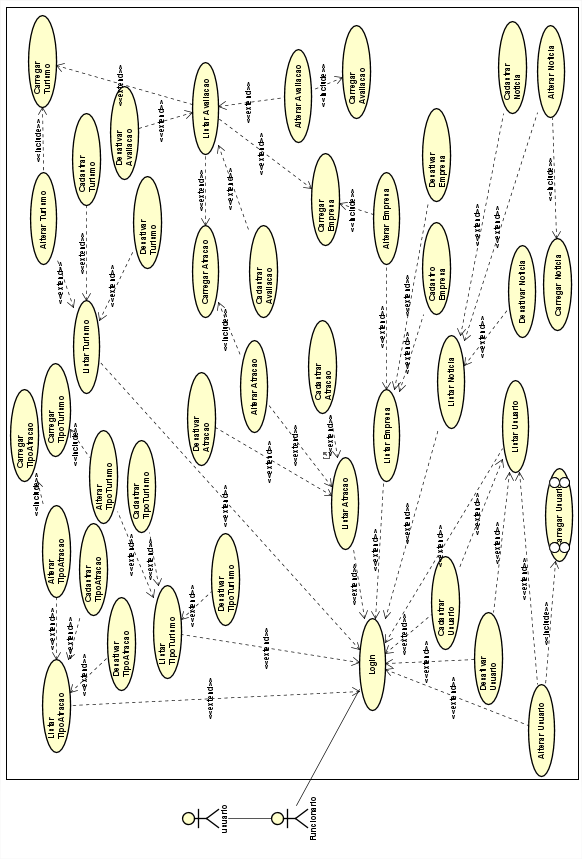
Fonte: Elaborado pelos autores.

**Figura 12 —** Diagrama de caso de uso - Visão do Empresário



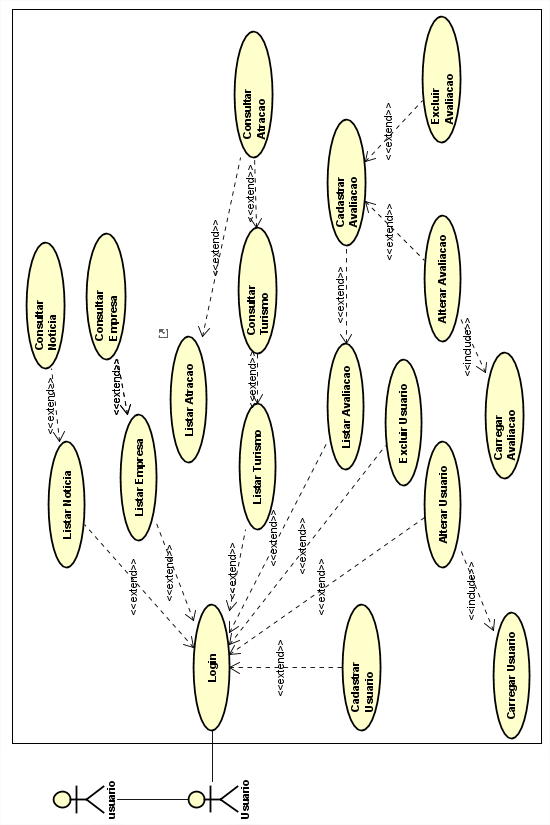
Fonte: Elaborado pelos autores.

**Figura 13 —** Diagrama de caso de uso - Visão do Funcionário



Fonte: Elaborado pelos autores.

**Figura 14 —** Diagrama de caso de uso - Visão do Usuario

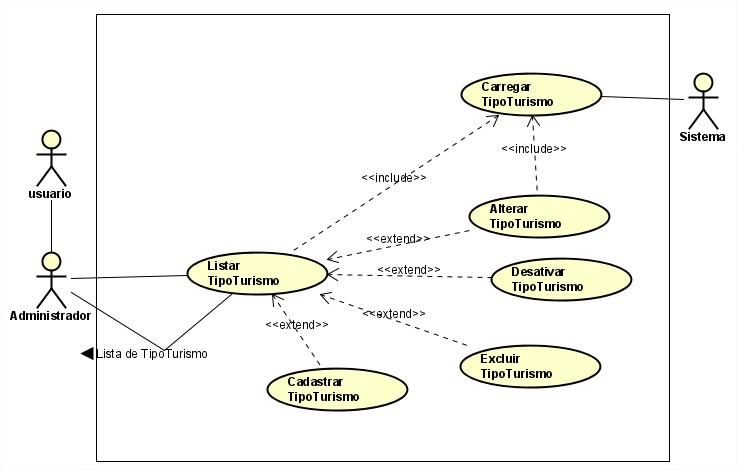


Fonte: Elaborado pelos autores.

* + 1. Diagrama de Casos de uso individuais

Na Figura 15, o diagrama de caso de uso mostra como o administrador interage com o sistema para gerenciar os tipos de turismo. O diagrama demonstra que ao listar os tipos de turismo, o administrador pode realizar outras ações, sendo elas, cadastrar, excluir, alterar e desativar.

**Figura 15 —** Diagrama de casos de uso específico – Listar Tipo Turismo



Fonte: Elaborado pelos autores.

O Quadro 24 apresenta a documentação do caso de uso "Listar Tipo Turismo". Nesse cenário, o ator administrador deseja visualizar uma lista de tipos de turismo disponíveis no sistema. Para isso, ele deve estar previamente logado no sistema. O fluxo normal inicia-se quando o administrador seleciona a opção "Listar Tipo Turismo" na página dedicada ao turismo. Em seguida, o sistema executa o caso de uso correspondente, processando a solicitação e exibindo os dados na página específica de listagem. O processo é concluído com a apresentação da lista de tipos de turismo ao administrador. Não há fluxo alternativo previsto para esse caso de uso.

**Quadro 24 —** Documentação do caso de uso Listar Tipo Turismo

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja listar os tipos turismo, seleciona a opção Listar na  página de Turismo. O sistema executa o Caso de uso Listar Tipo Turismo e exibe os dados na Página Listar Tipo Turismo. |
| Pré-Requisito: O Administrador estar Logado no Sistema. |
| Fluxo Normal:   1. – Administrador deseja listar Tipo Turismo. 2. – Administrador seleciona a opção Listar Tipo Turismo na página Turismo. 3. – Sistema executa o Caso de Uso Listar Tipo Turismo. 4. – Sistema exibe a página Listar Tipo Turismo. 5. – Fim. |
| Fluxo alternativo: Não Contém fluxo alternativo. |
| Dados: Lista Tipo Turismo. |

Fonte: Elaborado pelos autores.

De acordo com o Quadro 25, o caso de uso "Cadastrar Tipo Turismo" descreve o processo pelo qual o administrador insere novos tipos de turismo no sistema. Ao acessar a página de listagem de tipos de turismo, o administrador seleciona a opção de cadastro, e o sistema exibe um formulário de inserção que deve ser completamente preenchido. Ao submeter o formulário, o sistema valida os dados e, se não houver inconsistências, adiciona o novo tipo de turismo, confirmando a operação com uma mensagem de sucesso. Caso ocorra algum erro, como campos não preenchidos, o sistema notifica o administrador e permite a correção antes de finalizar o cadastro.

**Quadro 25 —** Documentação do caso de uso Cadastrar Tipo Turismo

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja adicionar tipos turismo, seleciona a opção cadastrar  na página de Listar Tipo Turismo. O sistema executa o Caso de uso Cadastrar Tipo Turismo e exibe a tela de cadastro Tipo Turismo. |
| Pré-Requisito: Preencher todos os campos do formulário de inserção. |
| Fluxo Normal:   1. – Usuário solicita a inserção de um novo tipo turismo. 2. – Sistema recebe solicitação e mostra tela com formulário de inserção. |

|  |
| --- |
| 1. – Usuário informa dados do tipo turismo ao formulário e clica em “Salvar”. 2. – Sistema valida campos e dados adicionados. 3. – Sistema retorna mensagem “Tipo Turismo adicionado!”. |
| Fluxo alternativo:   1. – Sistema valida campos e dados. 2. – Sistema identifica inconsistência de dados informados ou campos em branco. 3. – Sistema envia mensagem “Preencha todos os campos”. 4. – Sistema retorna ao fluxo normal. |
| Dados: Lista Tipo Turismo. |

Fonte: Elaborado pelos autores.

De acordo com o que está exposto no Quadro 26, a documentação do caso de uso "Alterar Tipo Turismo" descreve o processo em que o administrador, já logado no sistema, deseja modificar as informações de um tipo de turismo previamente cadastrado. O fluxo normal começa quando o administrador seleciona a opção "Alterar" na página de listagem dos tipos de turismo. O sistema exibe os dados na tela de alteração. Após realizar as modificações necessárias, o administrador clica em "Salvar", momento em que o sistema valida as alterações e atualiza as informações, confirmando a operação com a mensagem "Alteração realizada com sucesso". Caso ocorra uma falha na conexão com o banco de dados durante a validação, o sistema seguirá um fluxo alternativo, retornando a mensagem "Erro de conexão com o banco de dados" antes de retomar o fluxo normal.

**Quadro 26 —** Documentação do caso de uso Alterar Tipo Turismo

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja alterar tipos turismo, seleciona a opção alterar na página de Listar Tipo Turismo. O sistema executa o Caso de uso Alterar Tipo Turismo e  exibe os dados de Tipo Turismo na tela de Alterar Tipo Turismo. |
| Pré-Requisito: Tipo turismo cadastrado. |
| Fluxo Normal:   1. – Usuário solicita a alteração de um tipo turismo. 2. – Sistema recebe solicitação e mostra tela com os dados do tipo turismo. 3. – Usuário altera os dados do tipo turismo e seleciona o botão “Salvar”. 4. – Sistema valida solicitação e altera dados. 5. – Sistema envia mensagem “Alteração realizada com sucesso”. |

|  |
| --- |
| Fluxo alternativo:   1. – Sistema valida solicitação e altera dados. 2. – Sistema falha ao conectar-se com o banco de dados. 3. – Sistema envia mensagem “Erro de conexão com o banco de dados”. 4. – Sistema retorna ao fluxo normal. |
| Dados: Lista Tipo Turismo. |

Fonte: Elaborado pelos autores.

Conforme apresentado no Quadro 27, o caso de uso "Excluir Tipo Turismo", apresenta o processo a qual um administrador pode remover tipos de turismo cadastrados no sistema. Para que a exclusão ocorra, é necessário que o tipo de turismo esteja previamente cadastrado. O fluxo normal começa quando o administrador solicita a exclusão de um tipo de turismo na página de listagem. O sistema exibe uma tela com os tipos de turismo disponíveis, permitindo selecionar o tipo desejado para exclusão. Após a validação da solicitação, o sistema direciona o administrador para o módulo específico onde ele pode confirmar a exclusão. O sistema, então, exibe a tela de confirmação da exclusão. Existe um fluxo alternativo que trata de possíveis falhas de conexão com o banco de dados; neste caso, o sistema exibe uma mensagem de erro e retorna ao fluxo normal.

**Quadro 27** — Documentação do caso de uso Excluir Tipo Turismo

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja excluir tipos turismo, seleciona a opção excluir na página de Listar Tipo Turismo. O sistema executa o Caso de uso excluir Tipo Turismo e  exclui o Tipo Turismo. |
| Pré-Requisito: Tipo turismo cadastrado. |
| Fluxo Normal:   1. – Usuário solicita exclusão dos tipos turismo cadastrados. 2. – Sistema recebe solicitação e mostra tela com os tipos turismo. 3. – Usuário seleciona tipo turismo que deseja ter acesso às informações. 4. – Sistema valida solicitação. 5. – Sistema direciona usuário ao módulo correspondente de operações. 6. - Usuário seleciona botão “Excluir”. 7. - Sistema carrega tela de exclusão. |
| Fluxo alternativo: |

|  |
| --- |
| 1. – Sistema lista solicitação. 2. – Sistema falha ao conectar-se com o banco de dados. 3. – Sistema envia mensagem “Erro de conexão com o banco de dados”. 4. – Sistema retorna ao fluxo normal. |
| Dados: Lista Tipo Turismo. |

Fonte: Elaborado pelos autores.

O Quadro 28 descreve o caso de uso "Desativar Tipo Turismo", detalhando como um administrador pode desativar tipos de turismo do sistema. O pré-requisito para essa operação é que o tipo de turismo esteja previamente cadastrado. O fluxo normal inicia-se quando o administrador opta por desativar um tipo de turismo na página de listagem. O sistema então apresenta uma tela com todos os tipos de turismo cadastrados, permitindo selecionar o tipo específico para desativação. Após validar a solicitação, o sistema direciona o administrador para o módulo de operações, para confirmar a ação. A tela de desativação é então carregada para completar o processo. Existe um fluxo alternativo que lida com falhas de conexão com o banco de dados: se ocorrer um erro, o sistema exibe uma mensagem de “Erro de conexão com o banco de dados” e retorna ao fluxo normal.

**Quadro 28 —** Documentação do caso de uso Desativar Tipo Turismo

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja desativar tipos turismo, seleciona a opção desativar na página de Listar Tipo Turismo. O sistema executa o Caso de uso desativar Tipo Turismo  e desativa o Tipo Turismo. |
| Pré-Requisito: Tipo turismo cadastrado. |
| Fluxo Normal:   1. – Usuário solicita desativação dos tipos turismo cadastrados. 2. – Sistema recebe solicitação e mostra tela com os tipos turismo. 3. – Usuário seleciona tipo turismo que deseja ter acesso às informações. 4. – Sistema valida solicitação. 5. – Sistema direciona usuário ao módulo correspondente de operações. 6. - Usuário seleciona botão “Desativar”. 7. - Sistema carrega tela de exclusão. |
| Fluxo alternativo:   1. – Sistema lista solicitação. |

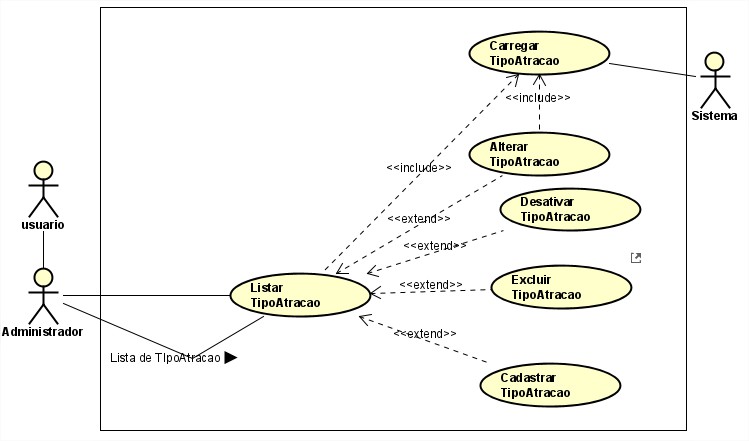
|  |
| --- |
| 1. – Sistema falha ao conectar-se com o banco de dados. 2. – Sistema envia mensagem “Erro de conexão com o banco de dados”. 3. – Sistema retorna ao fluxo normal. |
| Dados: Lista Tipo Turismo. |

Fonte: Elaborado pelos autores.

* + 1. Diagrama de Casos de uso individuais – Listar tipo atração

Na Figura 16, o diagrama de caso de uso ilustra a interação do administrador com o sistema para a gestão dos tipos de atração. O diagrama revela que, ao listar os tipos de atração, o administrador tem a capacidade de executar várias ações adicionais, incluindo cadastrar, excluir, alterar e desativar tipos de atração.

**Figura 16** - Diagrama de casos de uso específico – Listar Tipo Atração



Fonte: Elaborado pelos autores.

O Quadro 29 detalha o caso de uso "Listar Tipo Atração". Neste caso, o administrador, que deve estar previamente logado no sistema, deseja visualizar os tipos de atração disponíveis. O fluxo normal começa quando o administrador seleciona a opção "Listar Tipo Atração" na página destinada a atração. O sistema então processa essa solicitação e exibe os dados na página de listagem específica. O processo é concluído com a apresentação da lista de tipos de atração ao administrador. Não há fluxos alternativos definidos para este caso de uso.

## Quadro 29 — Documentação do caso de uso Listar Tipo Atração

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja listar os tipos atração, seleciona a opção listar na  página de Atração. O sistema executa o Caso de uso Listar Tipo Atração e exibe os dados na Página Listar Tipo Atração. |
| Pré-Requisito: O Administrador estar Logado no Sistema. |
| Fluxo Normal:   1. – Administrador deseja listar Tipo Atração. 2. – Administrador seleciona a opção Listar Tipo Atração na página Atração. 3. – Sistema executa o Caso de Uso Listar Tipo Atração. 4. – Sistema exibe a página Listar Tipo Atração. 5. – Fim. |
| Fluxo alternativo: Não Contém fluxo alternativo. |
| Dados: Lista Tipo Atração. |

Fonte: Elaborado pelos autores.

Como descrito no Quadro 30, o caso de uso "Cadastrar Tipo Atração" descreve o processo pelo qual o administrador adiciona novos tipos de atração ao sistema. Quando o administrador acessa a página de listagem de tipos de atração e seleciona a opção de cadastro,

o sistema apresenta um formulário de inserção que deve ser preenchido completamente. Após

o envio do formulário, o sistema valida os dados e, se não houver erros, registra o novo tipo de atração e confirma a operação com uma mensagem de sucesso. Se houver erros, como campos não preenchidos, o sistema notifica o administrador para que as correções necessárias sejam feitas antes de concluir o cadastro.

**Quadro 30** — Documentação do caso de uso Cadastrar Tipo Atração

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja adicionar tipos atração, seleciona a opção cadastrar  na página de Listar Tipo Atração. O sistema executa o Caso de uso Cadastrar Tipo Atração e exibe a tela de cadastro Tipo Atração. |
| Pré-Requisito: Preencher todos os campos do formulário de inserção. |
| Fluxo Normal:   1. – Usuário solicita a inserção de um novo tipo atração. 2. – Sistema recebe solicitação e mostra tela com formulário de inserção. |

|  |
| --- |
| 1. – Usuário informa dados do tipo atração ao formulário e clica em “Salvar”. 2. – Sistema valida campos e dados adicionados. 3. – Sistema retorna mensagem “Tipo Atração adicionado!”. |
| Fluxo alternativo:   1. – Sistema valida campos e dados. 2. – Sistema identifica inconsistência de dados informados ou campos em branco. 3. – Sistema envia mensagem “Preencha todos os campos”. 4. – Sistema retorna ao fluxo normal. |
| Dados: Lista Tipo Atração. |

Fonte: Elaborado pelos autores.

Como indicado no Quadro 31, o caso de uso "Alterar Tipo Atração" descreve o processo pelo qual um administrador, já logado no sistema, modifica as informações de um tipo de atração previamente cadastrado. O fluxo normal começa quando o administrador seleciona a opção "Alterar" na página de listagem de tipos de atração. O sistema então exibe os dados atuais na tela de alteração. Após realizar as modificações desejadas, o administrador clica em "Salvar". Nesse momento, o sistema valida as alterações e atualiza as informações, confirmando a operação com a mensagem "Alteração realizada com sucesso". Se ocorrer uma falha na conexão com o banco de dados durante a validação, o sistema exibirá a mensagem "Erro de conexão com o banco de dados" e, em seguida, retornará ao fluxo normal para nova tentativa.

**Quadro 31** — Documentação do caso de uso Alterar Tipo Atração

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja alterar tipos atração, seleciona a opção alterar na  página de Listar Tipo Atração. O sistema executa o Caso de uso Alterar Tipo Atração e exibe os dados de Tipo Atração na tela de Alterar Tipo Atração. |
| Pré-Requisito: Tipo atração cadastrado. |
| Fluxo Normal:   1. – Usuário solicita a alteração de um tipo atração. 2. – Sistema recebe solicitação e mostra tela com os dados do tipo atração. 3. – Usuário altera os dados do tipo atração e seleciona o botão “Salvar”. 4. – Sistema valida solicitação e altera dados. 5. – Sistema envia mensagem “Alteração realizada com sucesso”. |
| Fluxo alternativo: |

|  |
| --- |
| 1. – Sistema valida solicitação e altera dados. 2. – Sistema falha ao conectar-se com o banco de dados. 3. – Sistema envia mensagem “Erro de conexão com o banco de dados”. 4. – Sistema retorna ao fluxo normal. |
| Dados: Lista Tipo Atração. |

Fonte: Elaborado pelos autores.

Como demonstrado no Quadro 32, o caso de uso "Excluir Tipo Atração" detalha o processo pelo qual um administrador pode remover tipos de atração cadastrados no sistema. Para que a exclusão seja possível, o tipo de atração deve estar previamente cadastrado. O fluxo normal inicia-se quando o administrador solicita a exclusão de um tipo de atração na página de listagem. O sistema então apresenta uma tela com todos os tipos de atração disponíveis, permitindo ao administrador selecionar o tipo desejado para exclusão. Após validar a solicitação, o sistema direciona o administrador para o módulo específico para confirmar a exclusão. O sistema, então, exibe a tela de confirmação da exclusão. Caso ocorra uma falha de conexão com o banco de dados, um fluxo alternativo entra em ação: o sistema exibe uma mensagem de erro e retorna ao fluxo normal para uma nova tentativa.

**Quadro 32** — Documentação do caso de uso Excluir Tipo Atração

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja excluir tipos atração, seleciona a opção excluir na  página de Listar Tipo Atração. O sistema executa o Caso de uso excluir Tipo Atração e exclui o Tipo Atração. |
| Pré-Requisito: Tipo atração cadastrado. |
| Fluxo Normal:   1. – Usuário solicita exclusão dos tipos atração cadastrados. 2. – Sistema recebe solicitação e mostra tela com os tipos atração. 3. – Usuário seleciona tipo atração que deseja ter acesso às informações. 4. – Sistema valida solicitação. 5. – Sistema direciona usuário ao módulo correspondente de operações. 6. - Usuário seleciona botão “Excluir”. 7. - Sistema carrega tela de exclusão. |
| Fluxo alternativo:   1. – Sistema lista solicitação. |

|  |
| --- |
| 1. – Sistema falha ao conectar-se com o banco de dados. 2. – Sistema envia mensagem “Erro de conexão com o banco de dados”. 3. – Sistema retorna ao fluxo normal. |
| Dados: Lista Tipo Atração. |

Fonte: Elaborado pelos autores.

O Quadro 33 detalha o caso de uso "Desativar Tipo Atração", que descreve o processo pelo qual um administrador pode desativar tipos de atração no sistema. Para realizar essa operação, o tipo de atração deve estar previamente cadastrado. O fluxo normal começa quando o administrador escolhe a opção de desativar um tipo de aração na página de listagem. O sistema então exibe uma tela com todos os tipos de atração cadastrados, permitindo que o administrador selecione o tipo a ser desativado. Após a validação da solicitação, o sistema direciona o administrador para o módulo de operações para confirmar a desativação. Em seguida, a tela de desativação é carregada para concluir o processo. Caso ocorra uma falha na conexão com o banco de dados, o sistema exibirá uma mensagem de "Erro de conexão com o banco de dados" e retornará ao fluxo normal para nova tentativa.

**Quadro 33** — Documentação do caso de uso Desativar Tipo Atração

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja desativar tipos atração, seleciona a opção desativar na página de Listar Tipo Atração. O sistema executa o Caso de uso desativar Tipo Atração e  desativa o Tipo Atração. |
| Pré-Requisito: Tipo Atração cadastrado. |
| Fluxo Normal:   1. – Usuário solicita desativação dos tipos atração cadastrados. 2. – Sistema recebe solicitação e mostra tela com os tipos atração. 3. – Usuário seleciona tipo atração que deseja ter acesso às informações. 4. – Sistema valida solicitação. 5. – Sistema direciona usuário ao módulo correspondente de operações. 6. - Usuário seleciona botão “Desativar”. 7. - Sistema carrega tela de exclusão. |
| Fluxo alternativo:   1. – Sistema lista solicitação. 2. – Sistema falha ao conectar-se com o banco de dados. |

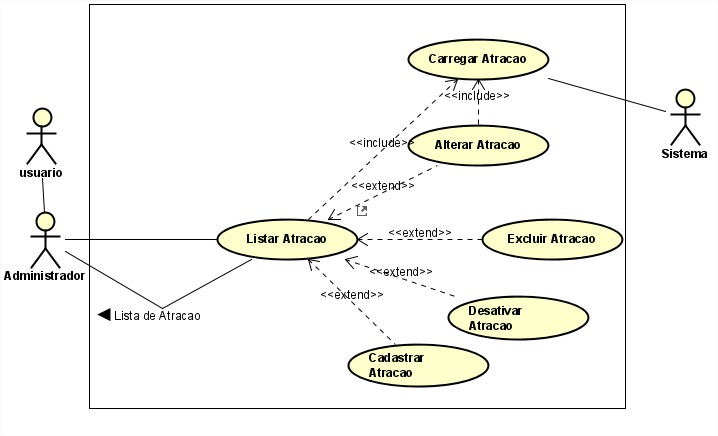
|  |
| --- |
| 1. – Sistema envia mensagem “Erro de conexão com o banco de dados”. 2. – Sistema retorna ao fluxo normal. |
| Dados: Lista Tipo Atração. |

Fonte: Elaborado pelos autores.

* + 1. Diagrama de Casos de uso individuais – Listar atração

O diagrama de caso de uso na Figura 17 mostra a interação do administrador com o sistema para gerenciar as atrações. O administrador pode listar as atrações e realizar ações como cadastrar, excluir, alterar e desativar.

**Figura 17** — Diagrama de casos de uso específico – Listar Atração



Fonte: Elaborado pelos autores.

O Quadro 34 descreve o caso de uso "Listar Atração", no qual o administrador, após fazer login no sistema, deseja visualizar as atrações disponíveis. O fluxo principal se inicia quando o administrador seleciona a opção "Listar Atração" na página de atrações. O sistema processa a solicitação e exibe a lista de atração na página correspondente. O processo se encerra com a apresentação da lista ao administrador, sem a existência de fluxos alternativos definidos para esse caso de uso.

**Quadro 34** — Documentação do caso de uso Listar Atração

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja listar os atração, seleciona a opção listar na página  principal do sistema. O sistema executa o Caso de uso Listar Atração e exibe os dados na Listar Atração. |
| Pré-Requisito: O Administrador estar Logado no Sistema. |
| Fluxo Normal:   1. – Administrador deseja listar Atração. 2. – Administrador seleciona a opção Listar Atração na página Atração. 3. – Sistema executa o Caso de Uso Listar Atração. 4. – Sistema exibe a página Listar Atração. 5. – Fim. |
| Fluxo alternativo: Não Contém fluxo alternativo. |
| Dados: Lista Atração. |

Fonte: Elaborado pelos autores.

Conforme ilustrado no Quadro 35, o caso de uso "Cadastrar Atração" detalha o processo de inserção de novas atrações no sistema pelo administrador. Ao acessar a página de listagem e escolher a opção de cadastro, o sistema apresenta um formulário que deve ser devidamente preenchido. Após o envio, o sistema realiza a validação dos dados e, se estiverem corretos, registra a nova atração, exibindo uma mensagem de sucesso. Caso ocorram erros, como campos obrigatórios não preenchidos, o sistema alerta o administrador para que as correções sejam feitas antes de finalizar o cadastro.

**Quadro 35** — Documentação do caso de uso Cadastrar Atração

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja adicionar atração, seleciona a opção cadastrar na página de Listar Atração. O sistema executa o Caso de uso Cadastrar Atração e exibe a tela  de cadastro Atração. |
| Pré-Requisito: Preencher todos os campos do formulário de inserção. |
| Fluxo Normal:   1. – Usuário solicita a inserção de uma nova atração. 2. – Sistema recebe solicitação e mostra tela com formulário de inserção. 3. – Usuário informa dados da atração ao formulário e clica em “Salvar”. |

|  |
| --- |
| 1. – Sistema valida campos e dados adicionados. 2. – Sistema retorna mensagem “Atração adicionada!”. |
| Fluxo alternativo:   1. – Sistema valida campos e dados. 2. – Sistema identifica inconsistência de dados informados ou campos em branco. 3. – Sistema envia mensagem “Preencha todos os campos”. 4. – Sistema retorna ao fluxo normal. |
| Dados: Lista Atração. |

Fonte: Elaborado pelos autores.

Conforme exibido no Quadro 36, o caso de uso "Alterar Atração" aborda o processo no qual o administrador, já autenticado no sistema, modifica as informações da atração previamente registrado. O fluxo principal se inicia quando o administrador seleciona a opção "Alterar" na página de listagem de tipos de atração. O sistema, então, exibe os dados atuais para edição. Após realizar as mudanças necessárias, o administrador clica em "Salvar". O sistema valida as alterações e, se estiverem corretas, atualiza as informações, confirmando o sucesso da operação com a mensagem "Alteração realizada com sucesso". Caso ocorra uma falha na conexão com o banco de dados durante a validação, o sistema exibe a mensagem "Erro de conexão com o banco de dados" e permite que o administrador tente novamente.

**Quadro 36** — Documentação do caso de uso Alterar Atração

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja alterar atração, seleciona a opção alterar na página de Listar Atração. O sistema executa o Caso de uso Alterar Atração e exibe os dados de Atração  na tela de Alterar Atração. |
| Pré-Requisito: Atração cadastrado. |
| Fluxo Normal:   1. – Usuário solicita a alteração de uma atração. 2. – Sistema recebe solicitação e mostra tela com os dados da atração. 3. – Usuário altera os dados da atração e seleciona o botão “Salvar”. 4. – Sistema valida solicitação e altera dados. 5. – Sistema envia mensagem “Alteração realizada com sucesso”. |
| Fluxo alternativo:   1. – Sistema valida solicitação e altera dados. |

|  |
| --- |
| 1. – Sistema falha ao conectar-se com o banco de dados. 2. – Sistema envia mensagem “Erro de conexão com o banco de dados”. 3. – Sistema retorna ao fluxo normal. |
| Dados: Lista Atração. |

Fonte: Elaborado pelos autores.

Conforme descrito no Quadro 37, o caso de uso "Excluir Atração" descreve o processo em que o administrador remove uma atração previamente cadastrada no sistema. O fluxo principal começa quando o administrador solicita a exclusão de uma atração na página de listagem. O sistema exibe todas as atrações disponíveis para que o administrador selecione o item a ser excluído. Após a validação da solicitação, o sistema direciona o administrador para

o módulo de confirmação, onde a exclusão é confirmada. Caso haja uma falha de conexão com

o banco de dados, o sistema ativa um fluxo alternativo, exibindo uma mensagem de erro e retornando ao fluxo principal para nova tentativa.

**Quadro 37** — Documentação do caso de uso Excluir Atração

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja excluir atração, seleciona a opção excluir na página  de Listar Atração. O sistema executa o Caso de uso excluir Atração e exclui a Atração. |
| Pré-Requisito: Atração cadastrado. |
| Fluxo Normal:   1. – Usuário solicita exclusão das atrações cadastrados. 2. – Sistema recebe solicitação e mostra tela com as atrações. 3. – Usuário seleciona atração que deseja ter acesso às informações. 4. – Sistema valida solicitação. 5. – Sistema direciona usuário ao módulo correspondente de operações. 6. - Usuário seleciona botão “Excluir”. 7. - Sistema carrega tela de exclusão. |
| Fluxo alternativo:   1. – Sistema lista solicitação. 2. – Sistema falha ao conectar-se com o banco de dados. 3. – Sistema envia mensagem “Erro de conexão com o banco de dados”. |

|  |
| --- |
| 4 – Sistema retorna ao fluxo normal. |
| Dados: Lista Atração. |

Fonte: Elaborado pelos autores.

O Quadro 38 apresenta o caso de uso "Desativar Atração", que envolve o processo no qual o administrador desativa as atrações previamente cadastrados no sistema. O fluxo principal inicia-se quando o administrador seleciona a opção de desativar uma atração na página de listagem. O sistema apresenta uma tela com todos os tipos cadastrados, permitindo que o administrador escolha o item a ser desativado. Após validar a solicitação, o sistema redireciona para o módulo de confirmação da desativação. O processo é concluído com a tela de desativação. Se houver uma falha de conexão com o banco de dados, o sistema exibirá a mensagem "Erro de conexão com o banco de dados" e retornará ao fluxo principal para uma nova tentativa.

**Quadro 38 —** Documentação do caso de uso Desativar Atração

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja desativar atração, seleciona a opção desativar na página de Listar Atração. O sistema executa o Caso de uso desativar Atração e desativa a  Atração. |
| Pré-Requisito: Atração cadastrado. |
| Fluxo Normal:   1. – Usuário solicita desativação das atrações cadastrados. 2. – Sistema recebe solicitação e mostra tela com as atrações. 3. – Usuário seleciona atração que deseja ter acesso às informações. 4. – Sistema valida solicitação. 5. – Sistema direciona usuário ao módulo correspondente de operações. 6. - Usuário seleciona botão “Desativar”. 7. - Sistema carrega tela de exclusão. |
| Fluxo alternativo:   1. – Sistema lista solicitação. 2. – Sistema falha ao conectar-se com o banco de dados. 3. – Sistema envia mensagem “Erro de conexão com o banco de dados”. 4. – Sistema retorna ao fluxo normal. |

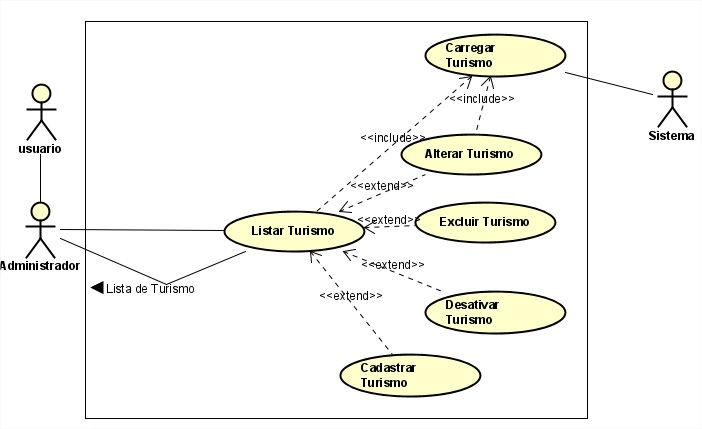
|  |
| --- |
| Dados: Lista Atração. |

Fonte: Elaborado pelos autores.

* + 1. Diagrama de Casos de uso individuais – Listar Turismo

Na Figura 18, o diagrama de caso de uso ilustra a interação do administrador com o sistema para a gestão dos turismos. O diagrama revela que, ao listar os turismos, o administrador pode executar várias ações adicionais, incluindo cadastrar, excluir, alterar e desativar turismo.

**Figura 18** — Diagrama de casos de uso específico – Listar Turismo



Fonte: Elaborado pelos autores.

O Quadro 39 retrata o caso de uso "Listar Turismo", que detalha o processo pelo qual o administrador, após fazer login no sistema, visualiza os turismos disponíveis. O fluxo principal se inicia quando o administrador seleciona a opção "Listar Turismo" na página de gerenciamento de turismo. O sistema processa a solicitação e exibe a lista de atrações na página correspondente. O processo se encerra com a apresentação da lista ao administrador, sem a existência de fluxos alternativos definidos para este caso de uso.

**Quadro 39** — Documentação do caso de uso Listar Turismo

|  |
| --- |
| Ator principal: Usuário (Administrador) |

|  |
| --- |
| Descrição: O ator administrador deseja listar os turismos, seleciona a opção listar na página principal do sistema. O sistema executa o Caso de uso Listar Turismo e exibe os dados na  Página Listar Turismo. |
| Pré-Requisito: O Administrador estar Logado no Sistema. |
| Fluxo Normal:   1. – Administrador deseja listar Turismo. 2. – Administrador seleciona a opção Listar Turismo na página Turismo. 3. – Sistema executa o Caso de Uso Listar Turismo. 4. – Sistema exibe a página Listar Turismo. 5. – Fim. |
| Fluxo alternativo: Não Contém fluxo alternativo. |
| Dados: Lista Turismo. |

Fonte: Elaborado pelos autores.

Conforme descrito no Quadro 40, o caso de uso "Cadastrar Turismo" detalha o processo de inserção de novas atrações turísticas no sistema pelo administrador. Ao acessar a página de listagem e escolher a opção de cadastro, o sistema apresenta um formulário que deve ser preenchido corretamente. Após o envio, o sistema valida os dados e, se estiverem corretos, registra a nova atração, exibindo uma mensagem de sucesso. Caso ocorram erros, como campos obrigatórios não preenchidos, o sistema alerta o administrador para que as correções sejam feitas antes de concluir o cadastro.

**Quadro 40** — Documentação do caso de uso Cadastrar Turismo

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja adicionar turismo, seleciona a opção cadastrar na página de Listar Turismo. O sistema executa o Caso de uso Cadastrar Turismo e exibe a tela  de cadastro Turismo. |
| Pré-Requisito: Preencher todos os campos do formulário de inserção. |
| Fluxo Normal:   1. – Usuário solicita a inserção de um novo turismo. 2. – Sistema recebe solicitação e mostra tela com formulário de inserção. 3. – Usuário informa dados do turismo ao formulário e clica em “Salvar”. 4. – Sistema valida campos e dados adicionados. 5. – Sistema retorna mensagem “Turismo adicionado!”. |

|  |
| --- |
| Fluxo alternativo:   1. – Sistema valida campos e dados. 2. – Sistema identifica inconsistência de dados informados ou campos em branco. 3. – Sistema envia mensagem “Preencha todos os campos”. 4. – Sistema retorna ao fluxo normal. |
| Dados: Lista Turismo. |

Fonte: Elaborado pelos autores.

Conforme descrito no Quadro 41, o caso de uso "Alterar Turismo" aborda o processo pelo qual o administrador, já autenticado no sistema, modifica as informações de um turismo previamente registrado. O fluxo principal inicia-se quando o administrador seleciona a opção "Alterar" na página de listagem de turismo. O sistema então exibe os dados atuais para edição. Após realizar as modificações necessárias, o administrador clica em "Salvar". O sistema valida as alterações e, se estiverem corretas, atualiza as informações, confirmando o sucesso da operação com a mensagem "Alteração realizada com sucesso". Caso ocorra uma falha na conexão com o banco de dados durante a validação, o sistema exibe a mensagem "Erro de conexão com o banco de dados" e permite que o administrador tente novamente.

**Quadro 41** — Documentação do caso de uso Alterar Turismo

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja alterar turismo, seleciona a opção alterar na página  de Listar Turismo. O sistema executa o Caso de uso Alterar Turismo e exibe os dados de Turismo na tela de Alterar Turismo. |
| Pré-Requisito: Turismo cadastrado. |
| Fluxo Normal:   1. – Usuário solicita a alteração de um turismo. 2. – Sistema recebe solicitação e mostra tela com os dados do turismo. 3. – Usuário altera os dados do turismo e seleciona o botão “Salvar”. 4. – Sistema valida solicitação e altera dados. 5. – Sistema envia mensagem “Alteração realizada com sucesso”. |
| Fluxo alternativo:   1. – Sistema valida solicitação e altera dados. 2. – Sistema falha ao conectar-se com o banco de dados. 3. – Sistema envia mensagem “Erro de conexão com o banco de dados”. |

|  |
| --- |
| 4 – Sistema retorna ao fluxo normal. |
| Dados: Lista Turismo. |

Fonte: Elaborado pelos autores.

Como ilustrado no Quadro 42, o caso de uso "Excluir Turismo" descreve o processo em que o administrador remove um turismo previamente cadastrado no sistema. O fluxo principal começa quando o administrador solicita a exclusão na página de listagem. O sistema exibe todos os turismos disponíveis para que o administrador selecione o item a ser excluído. Após validar a solicitação, o sistema direciona o administrador para o módulo de confirmação, onde a exclusão é confirmada. Caso haja uma falha de conexão com o banco de dados, o sistema ativa um fluxo alternativo, exibindo uma mensagem de erro e retornando ao fluxo principal para uma nova tentativa.

**Quadro 42** — Documentação do caso de uso Excluir Turismo

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja excluir turismo, seleciona a opção excluir na página  de Listar Turismo. O sistema executa o Caso de uso excluir Turismo e exclui o Turismo. |
| Pré-Requisito: Turismo cadastrado. |
| Fluxo Normal:   1. – Usuário solicita exclusão dos turismos cadastrados. 2. – Sistema recebe solicitação e mostra tela com os turismos. 3. – Usuário seleciona turismo que deseja ter acesso às informações. 4. – Sistema valida solicitação. 5. – Sistema direciona usuário ao módulo correspondente de operações. 6. - Usuário seleciona botão “Excluir”. 7. - Sistema carrega tela de exclusão. |
| Fluxo alternativo:   1. – Sistema lista solicitação. 2. – Sistema falha ao conectar-se com o banco de dados. 3. – Sistema envia mensagem “Erro de conexão com o banco de dados”. 4. – Sistema retorna ao fluxo normal. |
| Dados: Lista Turismo. |

Fonte: Elaborado pelos autores.

O Quadro 43 representa o caso de uso "Desativar Turismo", que envolve o processo no qual o administrador desativa turismos previamente cadastrados no sistema. O fluxo principal inicia-se quando o administrador seleciona a opção de desativar um turismo na página de listagem. O sistema apresenta uma tela com todos os turismos cadastrados, permitindo que o administrador escolha a que deve ser desativada. Após validar a solicitação, o sistema redireciona para o módulo de confirmação da desativação. O processo é concluído com a tela de confirmação da desativação. Caso ocorra uma falha de conexão com o banco de dados, o sistema exibirá a mensagem "Erro de conexão com o banco de dados" e retornará ao fluxo principal para uma nova tentativa.

**Quadro 43** — Documentação do caso de uso Desativar Turismo

|  |
| --- |
| Ator principal: Usuário (Administrador) |
| Descrição: O ator administrador deseja desativar turismo, seleciona a opção desativar na  página de Listar Turismo. O sistema executa o Caso de uso desativar Turismo e desativa o Turismo. |
| Pré-Requisito: Turismo cadastrado. |
| Fluxo Normal:   1. – Usuário solicita desativação dos turismos cadastrados. 2. – Sistema recebe solicitação e mostra tela com os turismos. 3. – Usuário seleciona turismo que deseja ter acesso às informações. 4. – Sistema valida solicitação. 5. – Sistema direciona usuário ao módulo correspondente de operações. 6. - Usuário seleciona botão “Desativar”. 7. - Sistema carrega tela de exclusão. |
| Fluxo alternativo:   1. – Sistema lista solicitação. 2. – Sistema falha ao conectar-se com o banco de dados. 3. – Sistema envia mensagem “Erro de conexão com o banco de dados”. 4. – Sistema retorna ao fluxo normal. |
| Dados: Lista Turismo. |

Fonte: Elaborado pelos autores.

* 1. Diagrama de Sequência

O diagrama de sequência adota um modelo que representa o comportamento do usuário ao realizar ações dentro da aplicação, atentando para a ordem temporal das ações no desenvolvimento do processo. Este diagrama visa identificar o evento gerador do processo, o ator responsável por esse evento e a maneira como o desenvolvimento do mesmo deve ocorrer. Em suma, ele oferece uma representação visual que destaca a interação entre diferentes elementos da aplicação, facilitando a compreensão da lógica temporal e das responsabilidades associadas às ações realizadas (GUEDES, 2018).

Nas subseções seguintes, serão apresentados os diagramas de sequência para algumas ações dentro do sistema, visto que, dentre as camadas dos processos existem:

* **Ator:** Possui a responsabilidade de iniciar a solicitação, quando almeja realizar requisições obtendo a resposta do sistema.
* ***Interface*:** Destinada a conectar ações do usuário, direcionando suas solicitações ao sistema e trazendo suas respostas para o usuário.
* ***Controller*:** Camada responsável por obter requisições da *Interface* e transmiti- las para a *Repository*.
* ***Repository*:** Responsável por preparar as requisições recebidas antes de enviá- las para o Banco de Dados.
* **Banco de Dados:** Refere-se a onde os dados são armazenados de forma organizada e estruturada, representados nos diagramas como “DBComtur”.
  + 1. Diagrama de Sequência – Cadastrar Tipo Turismo

Demonstra-se na Figura 19, o processo de cadastro de Tipo Turismo. O fluxo começa com a solicitação de cadastro pelo funcionário, passa pela validação e armazenamento de dados na camada *Controller*, *Repository* e Banco de Dados, finalizando com a recuperação e envio da lista de tipos de turismo cadastrados pela *Interface*.

**Figura 19** — Diagrama de Sequência – Cadastrar Tipo Turismo



Fonte: Elaborado pelos autores.

* + 1. Diagrama de Sequência – Alterar Tipo Atração

Na Figura 20, observa-se o fluxo para alterar um Tipo de Atração. O processo começa com o funcionário solicitando a alteração, seguida pela validação e atualização dos dados na camada *Controller* e no *Repository*. No *Repository*, verifica-se a existência dos dados, envia-se

os novos dados para o Banco de Dados, e retorna-se com a confirmação da atualização. Finalmente, a *Interface* recupera e envia a lista atualizada de tipos de atração.

**Figura 20 —** Diagrama de Sequência – Alterar Tipo Atração

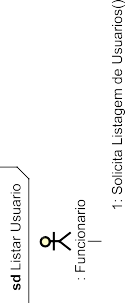


Fonte: Elaborado pelos autores.

* + 1. Diagrama de Sequência – Listar Usuário

Ilustra-se na Figura 21, o fluxo para listar usuários. Inicialmente, o funcionário solicita a listagem através da *Interface*, que encaminha a solicitação ao *Controller*. A *Controller* então repassa a solicitação ao *Repository*, que consulta o Banco de Dados. Após a consulta, o Banco de Dados retorna a lista de usuários ao *Repository*, que encaminha essa lista de volta ao *Controller*, e por fim, a Interface entrega a lista de usuários ao funcionário.

**Figura 21** — Diagrama de Sequência – Listar Usuário



Fonte: Elaborado pelos autores.

* + 1. Diagrama de Sequência – Excluir Empresa

Pode-se visualizar na Figura 22, o processo para excluir empresa. Inicializa com o funcionário solicitando a exclusão através da *Interface*, que encaminha a solicitação ao *Controller*. A *Controller* então repassa a solicitação ao *Repository*, que por sua vez verifica se o item existe, e então é repassado ao Banco de Dados. Após a remoção, o Banco de Dados, o *Repository*, e a *Controller*, retorna verdadeiro para a operação, e a *Interface* informa a lista atualizada.

**Figura 22 —** Diagrama de Sequência – Excluir Empresa



Fonte: Elaborado pelos autores.

* + 1. Diagrama de Sequência – Consultar Atração

Delineia-se na Figura 23, o processo de consulta de atração. O fluxo começa com a solicitação de consulta pela *Interface*, que pede os dados necessários ao funcionário e, após recebê-los, os envia ao *Controller*. Este, por sua vez, encaminha a solicitação ao *Repository*, que a envia ao Banco de Dados. Após o processamento, o Banco de Dados retorna os dados ao *Repository*, que os leva a *Controller* e retorna os dados à *Interface* para consulta.

**Figura 23 —** Diagrama de Sequência – Consultar Atração



Fonte: Elaborado pelos autores.

# DEFINIÇÃO DA INTERFACE COM O USUÁRIO (UX)

*User Experience* (UX), traduzido para o português como Experiência do Usuário, é o estudo de práticas e técnicas para garantir que as necessidades e expectativas do usuário sejam atendidas de maneira satisfatória e intuitiva. O objetivo é proporcionar uma experiência agradável para melhorar a usabilidade e satisfação, apresentando valor real ao serviço ou produto oferecido (Unger e Chandler, 2009).

* 1. Descrição de Cenário

Conforme destacado por Unger e Chandler (2009) os cenários são narrativas detalhadas que capturam as interações entre um usuário e um sistema. Essas histórias fornecem uma compreensão abrangente ao abordar o contexto no qual a persona está inserida, os objetivos que ela busca alcançar, as ações específicas que realiza para atingir esses objetivos, os desafios que pode encontrar no processo e, por fim, os resultados dessas interações. Os cenários de persona é uma das ferramentas fundamentais para orientar o *design* de experiência do usuário, permitindo uma avaliação detalhada da eficácia do sistema sob diversas circunstâncias.

O primeiro cenário mostra como um comerciante local, ao ver sobre o site de turismo da cidade, pode usar a plataforma para aumentar a visibilidade de seu negócio (Figura 24).

**Figura 24** — Imagem do primeiro cenário



Fonte: Elaborado pelos autores.

O segundo cenário demonstra como uma funcionária do Conselho pode utilizar o site para promover eventos e facilitar o acesso às informações turísticas (Figura 25).

**Figura 25** — Imagem do segundo cenário



Fonte: Elaborado pelos autores.

* 1. Descrição de Personas

Unger e Chandler (2009) define que personas são representações fictícias de usuários que interagem com um sistema específico. Elas são desenvolvidas com foco no usuário para garantir uma compreensão aprofundada das suas necessidades, comportamentos e motivações. Ao criar personas, os *designers* e desenvolvedores buscam uma visão mais humanizada e centrada no usuário, permitindo a criação de soluções que atendam de maneira satisfatória às expectativas e requisitos de quem usará o produto final.

Para o Comtur, foi desenvolvido duas personas de acordo com as funcionalidades e objetivos do sistema. A primeira persona (Figura 26) é um empresário dono de uma loja de

sapatos que representa os comerciantes locais que precisam do turismo para impulsionar suas vendas e querem uma plataforma eficaz para se promover.

**Figura 26** — Imagem da primeira persona



Fonte: Elaborado pelos autores.

A segunda persona é uma funcionária do Conselho Municipal de Turismo (Figura 27) escolhida para representar os interesses e desafios dos servidores públicos que buscam divulgar eventos e atrair mais turistas para a cidade, garantindo que o site atenda às necessidades de campanha cultural e de informação.

**Figura 27** — Imagem da segunda persona

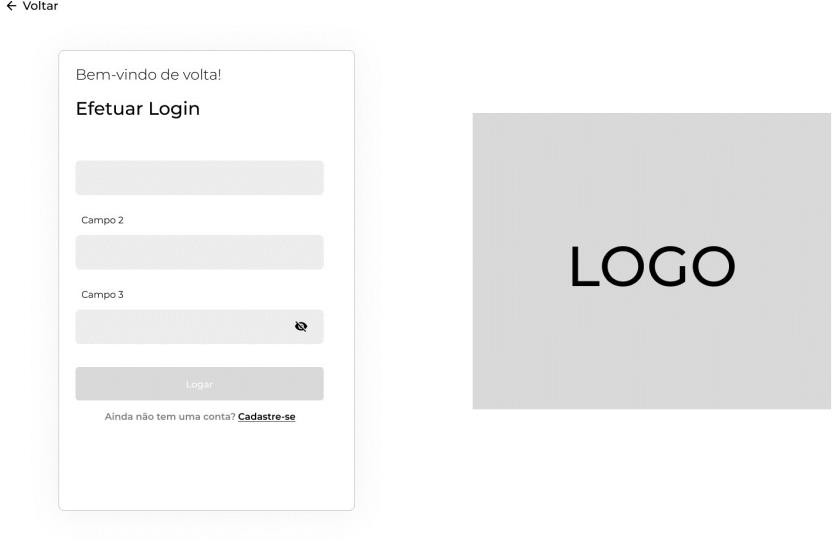


Fonte: Elaborado pelos autores.

* 1. Esboços de tela (*WIREFRAMES*)

Os *wireframes* são representações visuais esquemáticas e simplificadas de interfaces de usuário, desempenhando um papel crucial no processo de *design* de experiência do usuário. Nielsen (2006) define *wireframes* como: "*wireframes* são esboços visuais que representam a estrutura e o layout básico de uma página da web ou aplicativo".

**Figura 28** — Imagem do *Wireframe* da Tela de *Login*

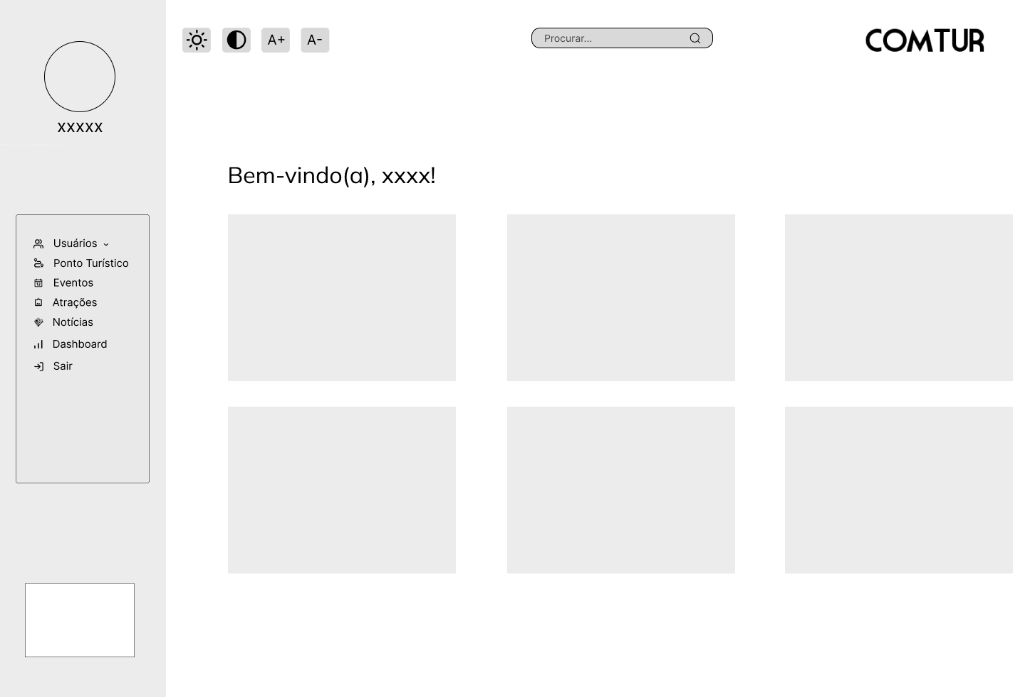
**

Fonte: Elaborado pelos autores.

Demonstra-se na figura 28, o *wireframe* de *login* no Comtur que possui uma disposição simplificada e estruturada para garantir uma experiência de usuário eficiente. Na parte esquerda da página, são claramente identificados os campos essenciais, refletindo a abordagem minimalista típica de w*ireframes*.

Além dos campos tradicionais de *login*, o *wireframe* inclui uma opção de "Cadastrar- se", estrategicamente posicionada para oferecer uma alternativa imediata a usuários que ainda não possuem uma conta. Essa abordagem, inspirada nas melhores práticas de usabilidade, segue os princípios de Nielsen (2006) sobre a clareza na apresentação de informações importantes em interfaces de usuário.

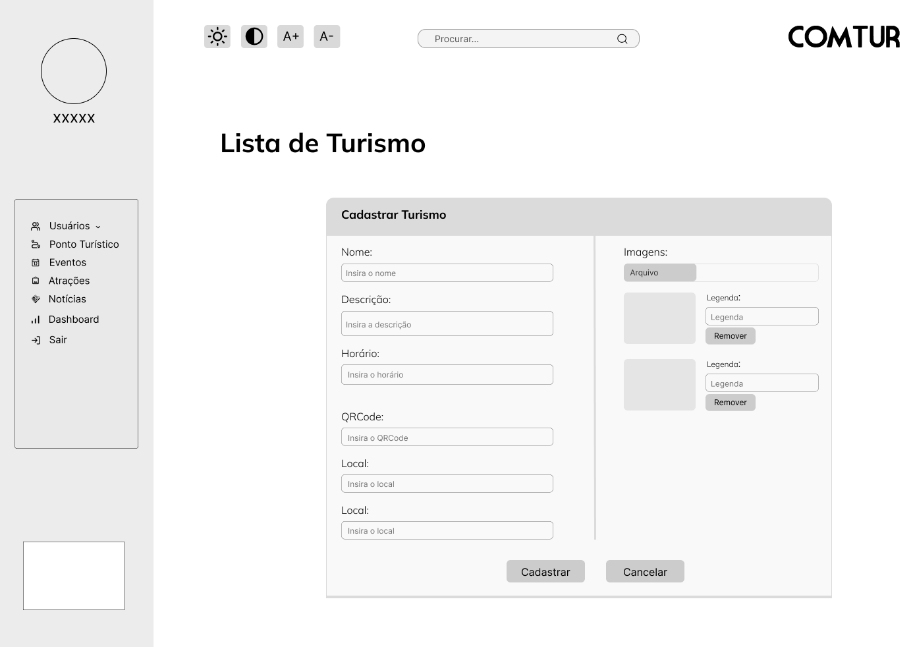
**Figura 29 —** Imagem do *wireframe* da Tela Inicial do Administrador



Fonte: Elaborado pelos autores.

Exibe-se na Figura 29, *wireframe* da tela inicial para o administrador no Comtur, ele é estruturado para garantir uma navegação intuitiva e eficiente. Inclui um *Navbar* na parte superior para acesso rápido às principais seções, um *Sidebar* à esquerda para uma navegação mais detalhada e um botão "*Logout*" para encerrar sessões facilmente. Seis *cards* no corpo principal oferecem atalhos visuais para informações e funcionalidades importantes.

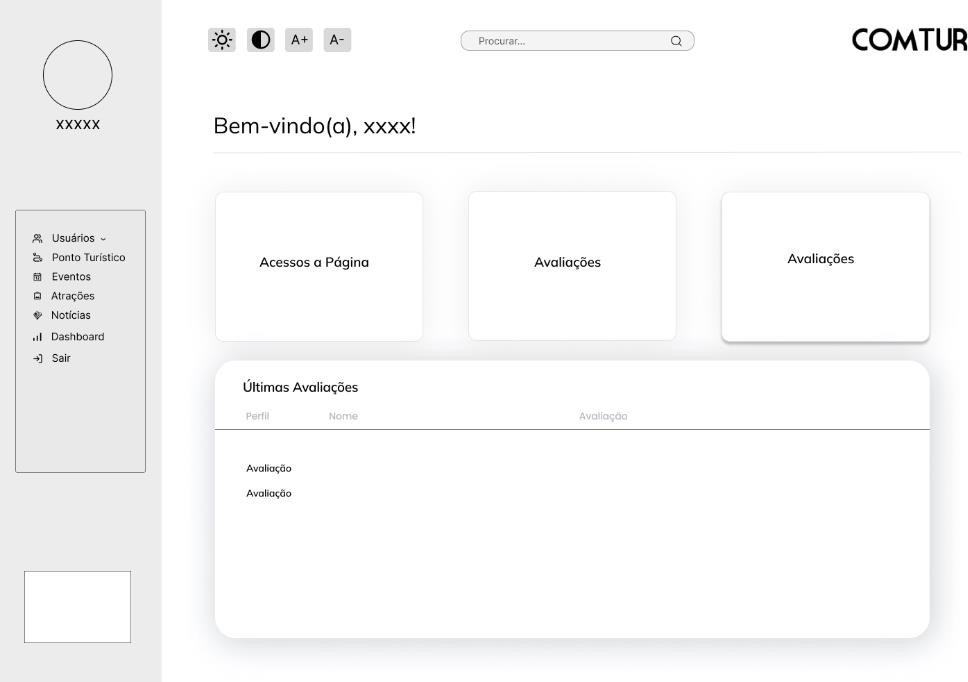
**Figura 30 —** Imagem do *wireframe* da Tela de Cadastro de Turismo



Fonte: Elaborado pelos autores.

Expõe-se na Figura 30, o *wireframe* para a tela de cadastro de turismos, onde mantém a consistência com o *Navbar* e *Sidebar*, proporcionando uma navegação intuitiva. O *Navbar* superior continua oferecendo acesso rápido a diferentes seções, enquanto o *Sidebar* à esquerda organiza *links* específicos para o cadastro de notícias. O corpo principal da tela destaca áreas para preenchimento dos detalhes do turismo, mantendo um *design* claro e simplificado para facilitar a entrada de dados pelo administrador. Essa abordagem assegura uma experiência coesa e eficiente durante o processo de cadastro.

**Figura 31 —** Imagem do w*ireframe* da Tela Inicial do Empresario



Fonte: Elaborado pelos autores.

Demonstra-se na Figura 31 o *wireframe* da tela inicial para o empresário, que possui uma navegação intuitiva e eficiente. Inclui um *Navbar* na parte superior para acesso rápido às principais seções, um *Sidebar* à esquerda para uma navegação mais detalhada. Além disso, possui três *cards* com intuito de trazer dados sobre a empresa, e uma tabela com as últimas avaliações.

Em síntese, os *wireframes* desenvolvidos para as telas fo Comtur refletem uma abordagem consistente e intuitiva, mantendo a usabilidade como prioridade. A estrutura organizada, aliada ao *Navbar* e *Sidebar*, proporciona uma experiência de navegação clara e eficiente para os administradores. A continuidade no *design*, seja durante o cadastro de notícias ou atrações, contribui para a familiaridade do usuário, simplificando os processos de entrada de dados. Assim, os *wireframes* reforçam a importância da consistência e usabilidade no

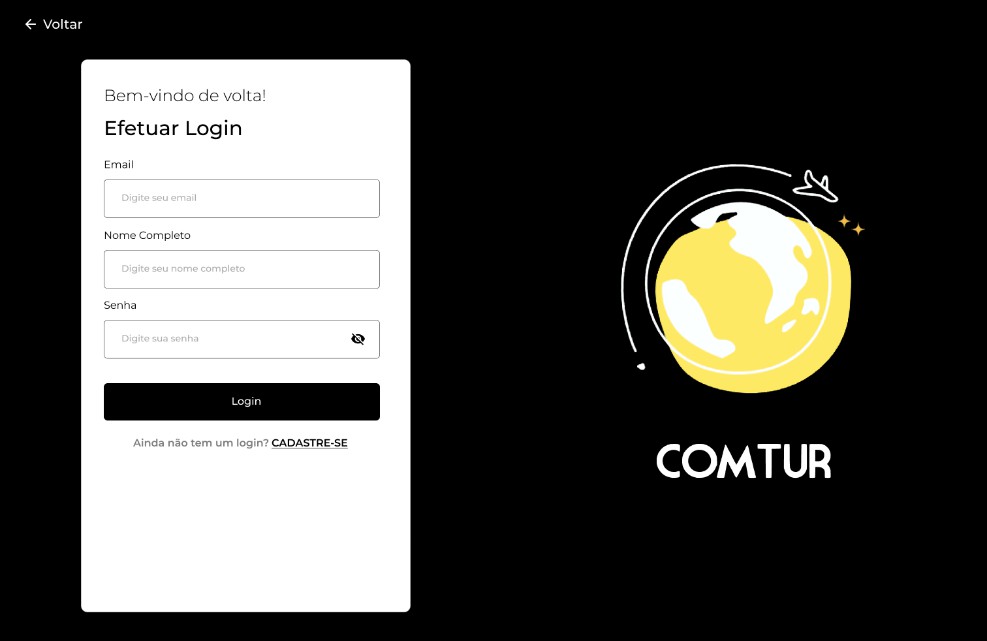
desenvolvimento de interfaces, promovendo uma experiência de usuário coesa e eficaz no Comtur.

* 1. Protótipos de tela

Os protótipos de tela, conforme destacado por Buxton (2007), servem como representações tangíveis e interativas da proposta de *design*, permitindo a exploração, comunicação e refinamento de ideias. No contexto do Comtur, essas representações desempenham um papel essencial ao fornecer uma visão visual antecipada das interfaces planejadas, garantindo a implementação de funcionalidades de forma mais assertiva. Krug (2010) também enfatiza que os protótipos são ferramentas fundamentais para explorar possíveis soluções de *design* antes da implementação completa, sem perder o foco na usabilidade real do produto.

A importância dos protótipos no Comtur transcende a mera visualização. Como destacado por Nielsen (2006), a iteração rápida e o feedback contínuo dos protótipos são cruciais para a criação de uma interface de usuário eficaz. Dessa forma, os protótipos de tela não apenas facilitam a comunicação entre as equipes de *design* e desenvolvimento, mas também desempenham um papel fundamental na identificação precoce de ajustes necessários, contribuindo para economia de tempo e recursos. No âmbito do Comtur, os protótipos de tela foram utilizados como ferramentas indispensáveis para a validação de conceitos, o refinamento de funcionalidades e a garantia de uma experiência de usuário otimizada desde as fases iniciais do desenvolvimento.

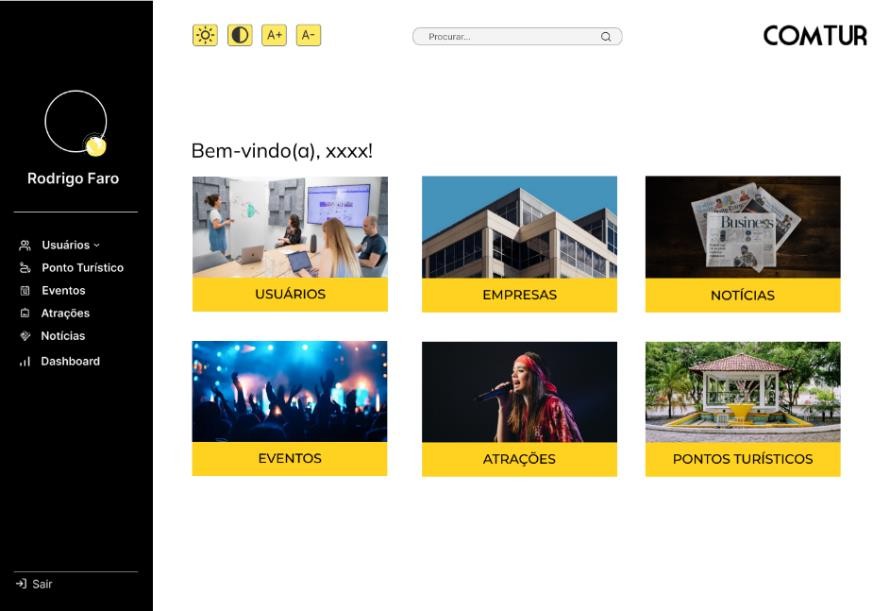
**Figura 32 —** Imagem do Protótipo da Tela de Login



Fonte: Elaborado pelos autores.

Exibe-se na Figura 32, o protótipo de tela para realizar um Login no Comtur, criado com base nas diretrizes de *design* de Norman (2002), que defende a flexibilidade e eficiência de uso. Com isso em mente, o objetivo é oferecer uma experiência de usuário eficaz e intuitiva. A inclusão de elementos como botões de "Entrar" e "Esqueceu a Senha" contribui para uma experiência de login acessível e orientada pelo usuário, refletindo o compromisso do Comtur com a usabilidade.

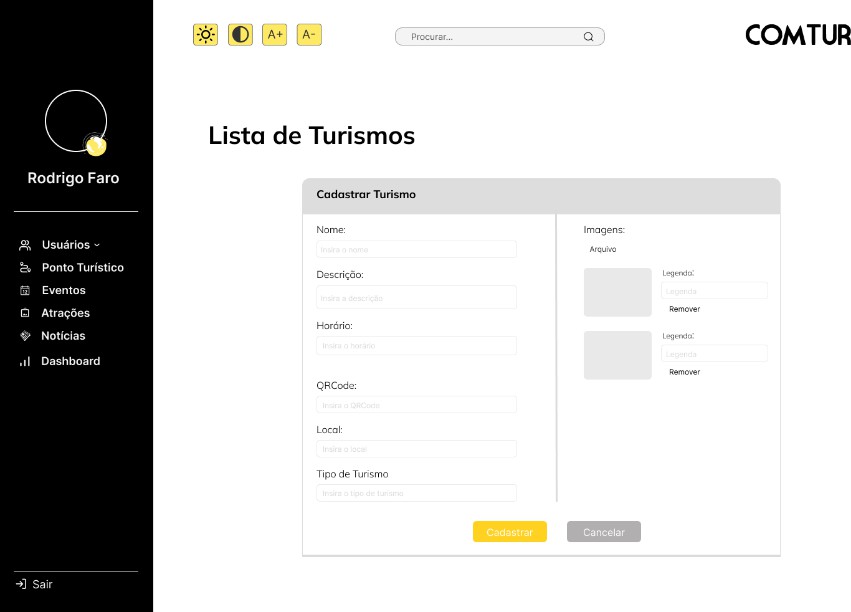
**Figura 33 —** Imagem do Prototipo da Tela Inicial do Administrador



Fonte: Elaborado pelos autores.

Ilustra-se na Figura 33, o protótipo de tela para a Tela Inicial do Administrador. A tela destaca um *Navbar* e um *Sidebar*. Seis *cards* que oferecem atalhos visuais para áreas importantes.

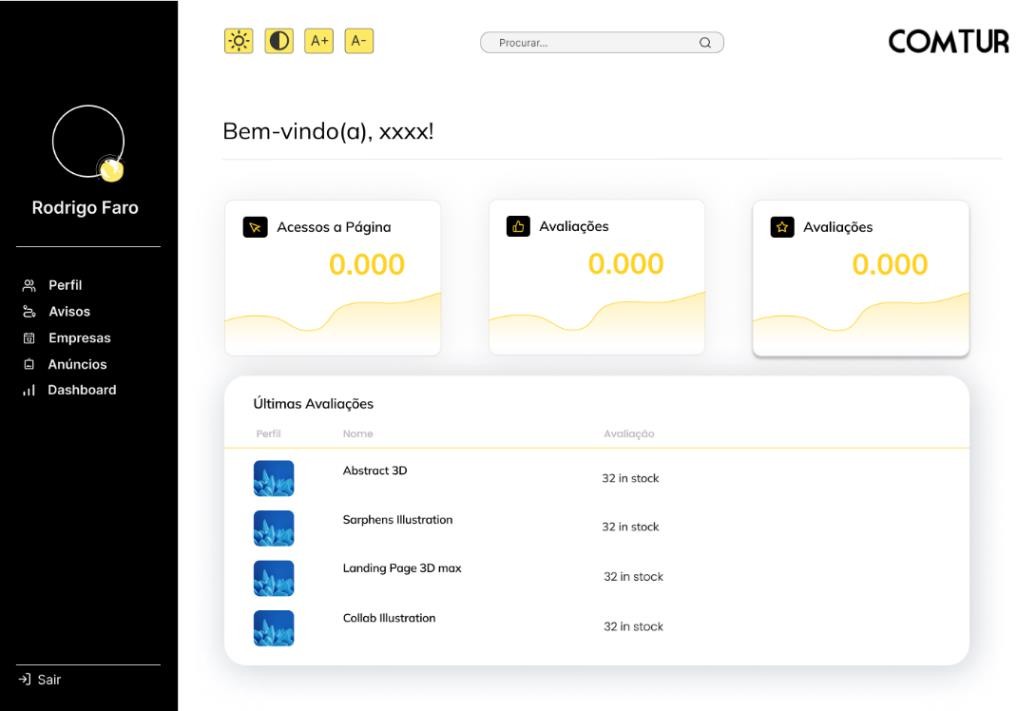
**Figura 34 —** Imagem do Prototipo da Tela de Cadastro de Turismos



Fonte: Elaborado pelos autores.

Representa-se na Figura 34, a tela de cadastro de turismos, mantendo a coesão através da continuidade do *Navbar* e *Sidebar*, garantindo uma navegação intuitiva. O *Navbar* superior continua proporcionando acesso rápido a diferentes utilidades, enquanto o *Sidebar* à esquerda organiza links específicos demais funcionalidades. O corpo principal da tela destaca áreas para preenchimento dos detalhes do turismo, mantendo um *design* claro e simplificado para facilitar a entrada de dados pelo administrador. Essa abordagem garante uma experiência fluida e eficiente durante o processo de cadastro.

**Figura 35 —** Imagem do Protótipo da Tela Inicial de Empresário



Fonte: Elaborado pelos autores.

Demonstra-se na Figura 35 o *layout* inicial para o empresário no Comtur, que apresenta uma navegação intuitiva e eficiente. Inclui um *Navbar* na parte superior para acesso rápido às principais seções, um *Sidebar* à esquerda para uma navegação mais detalhada e um botão "*Logout*" para encerrar sessões facilmente. Além disso, exibe três *cards* com o intuito de fornecer dados sobre a empresa e uma tabela com as últimas avaliações.

# BANCO DE DADOS

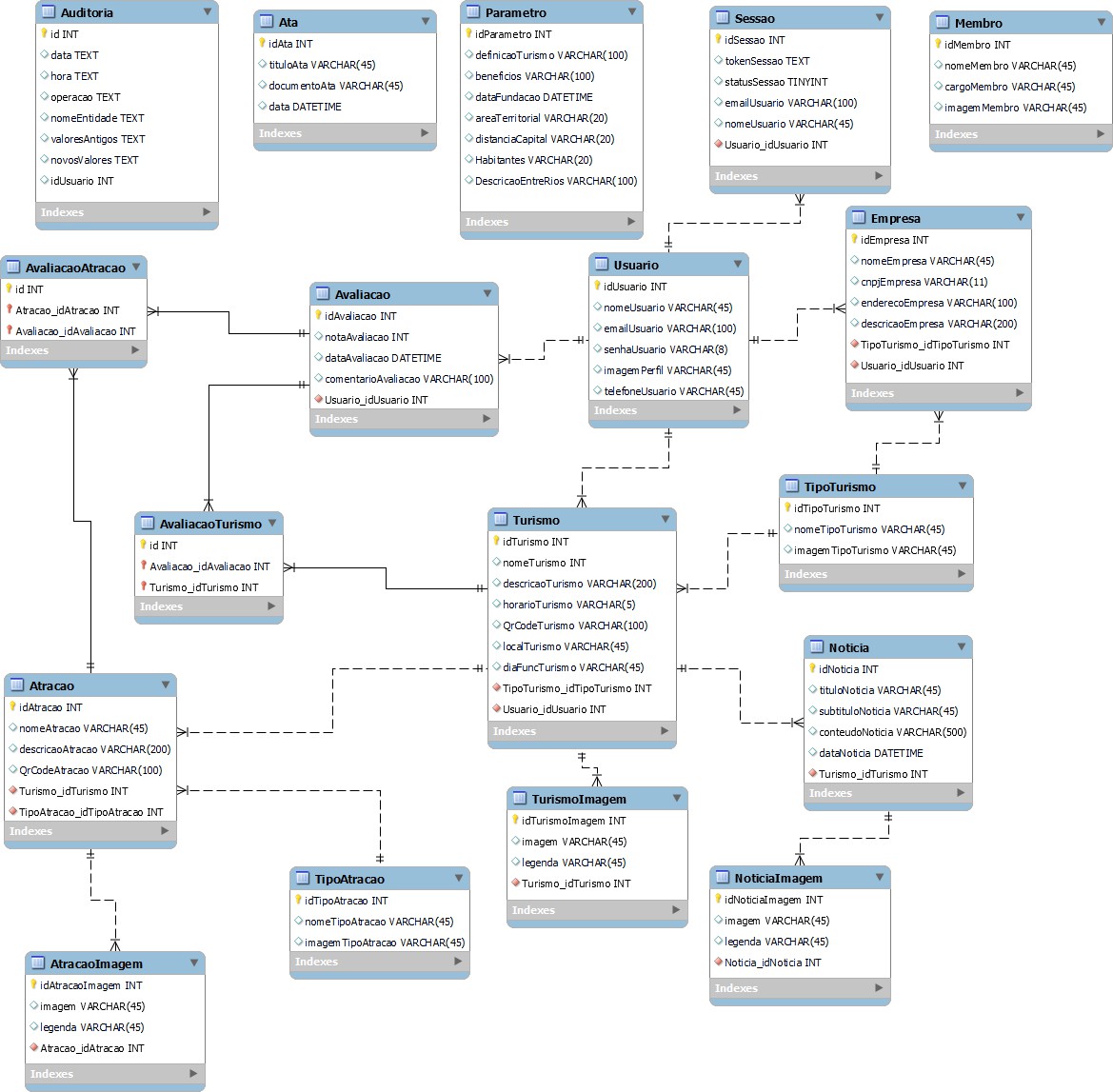
Conforme os conceitos apresentados por Machado (2020), um banco de dados é uma coleção de dados que permite o armazenamento, gerenciamento e recuperação das informações. Ele deve atender às demandas particulares de uma aplicação, assegurando a segurança e integridade, podendo ser categorizado como relacional e não relacional. O modelo relacional organiza os dados em tabelas, compostas por linhas e colunas. Cada tabela representa uma entidade, sendo seus relacionamentos determinados por chaves primárias e estrangeiras.

O PostgreSQL é um Sistema Gerenciador de Banco de Dados (SGBD) relacional, muito utilizado por sua capacidade de armazenamento, conformidade com padrões SQL e segurança. Além de oferecer recursos avançados como replicação e *multithreads*, popularizando seu uso no meio empresarial e projetos de grande escala (Milani, 2008).

* 1. Modelo Entidade Relacionamento

O Modelo Entidade-Relacionamento (MER), de acordo com Machado (2020), oferece uma representação visual das entidades (representação de objetos do mundo real), atributos (características de uma entidade) e relacionamentos (conexão entre diferentes entidades) que constituem o sistema de armazenamento de dados. É uma ferramenta que auxilia na captura de relações complexas e na estruturação dos dados de maneira compreensível (Figura 36).

**Figura 36** — Mapeamento do Objeto Relacional



Fonte: Elaborado pelos autores.

* 1. Script das Tabelas

Script das tabelas, conforme abordado por Machado (2020), são as instruções ou comandos escritos em uma linguagem de consulta estruturada (SQL) para criar, modificar ou excluir tabelas em um banco de dados. Esses scripts são utilizados para definir a estrutura de um banco de dados relacional, incluindo as tabelas, campo e restrições (Quadro 44 - 61).

**Quadro 44 —** Script SQL – Tabela TipoTurismo

CREATE TABLE tipoturismo (

tipoturismoid integer GENERATED BY DEFAULT AS IDENTITY, nome character varying(50) NOT NULL,

imagemtipoturismo text NULL, usuarioid integer NOT NULL,

CONSTRAINT "PK\_tipoturismo" PRIMARY KEY (tipoturismoid),

CONSTRAINT "FK\_tipoturismo\_usuario\_usuarioid" FOREIGN KEY (usuarioid) REFERENCES usuario (usuarioid) ON DELETE CASCADE);

Fonte: Elaborado pelos autores.

**Quadro 45 —** Script SQL – Tabela TipoAtracao

CREATE TABLE tipoatracao (

tipoatracaoid integer GENERATED BY DEFAULT AS IDENTITY, nome character varying(50) NOT NULL,

usuarioid integer NOT NULL,

CONSTRAINT "PK\_tipoatracao" PRIMARY KEY (tipoatracaoid),

CONSTRAINT "FK\_tipoatracao\_usuario\_usuarioid" FOREIGN KEY (usuarioid) REFERENCES usuario (usuarioid) ON DELETE CASCADE);

Fonte: Elaborado pelos autores.

**Quadro 46 —** Script SQL – Tabela Noticia

CREATE TABLE noticia (

noticiaid integer GENERATED BY DEFAULT AS IDENTITY, titulo text NOT NULL,

subtitulo text NOT NULL, conteudo text NOT NULL, datapublicacao date NOT NULL, horapublicacao text NOT NULL, usuarioid integer NOT NULL, idturismo integer NULL,

CONSTRAINT "PK\_noticia" PRIMARY KEY (noticiaid),

CONSTRAINT "FK\_noticia\_turismo\_idturismo" FOREIGN KEY (idturismo) REFERENCES turismo (turismoid) ON DELETE CASCADE,

CONSTRAINT "FK\_noticia\_usuario\_usuarioid" FOREIGN KEY (usuarioid) REFERENCES usuario (usuarioid) ON DELETE CASCADE);

Fonte: Elaborado pelos autores.

**Quadro 47 —** Script SQL – Tabela Usuario

CREATE TABLE usuario (

usuarioid integer GENERATED BY DEFAULT AS IDENTITY, nome character varying(50) NOT NULL,

telefone text NOT NULL, emailusuario text NOT NULL, senhausuario text NOT NULL, tipousuario integer NOT NULL, imagemperfilusuario text NULL,

CONSTRAINT "PK\_usuario" PRIMARY KEY (usuarioid));

Fonte: Elaborado pelos autores.

**Quadro 48 —** Script SQL – Tabela Avaliação

CREATE TABLE avaliacao (

avaliacaoid integer GENERATED BY DEFAULT AS IDENTITY, "notaAvaliacao" text NOT NULL,

"dataAvaliacao" text NOT NULL, "comentarioAvaliacao" text NOT NULL, statusavaliacao integer NOT NULL, idusuario integer NOT NULL,

CONSTRAINT "PK\_avaliacao" PRIMARY KEY (avaliacaoid),

CONSTRAINT "FK\_avaliacao\_usuario\_idusuario" FOREIGN KEY (idusuario) REFERENCES usuario (usuarioid) ON DELETE CASCADE);

Fonte: Elaborado pelos autores.

**Quadro 49 —** Script SQL – Tabela Turismo

CREATE TABLE turismo (

turismoid integer GENERATED BY DEFAULT AS IDENTITY, nome character varying(50) NOT NULL,

descricao text NOT NULL, horario text NOT NULL, qrcode text NOT NULL, local text NOT NULL,

diafuncionamento text NOT NULL, usuarioid integer NOT NULL, idtipoturismo integer NOT NULL, statustipoatracao integer NOT NULL, "TipoTurismoModelId" integer NULL,

CONSTRAINT "PK\_turismo" PRIMARY KEY (turismoid),

CONSTRAINT "FK\_turismo\_tipoturismo\_TipoTurismoModelId" FOREIGN KEY ("TipoTurismoModelId") REFERENCES tipoturismo (tipoturismoid),

CONSTRAINT "FK\_turismo\_tipoturismo\_idtipoturismo" FOREIGN KEY (idtipoturismo) REFERENCES tipoturismo (tipoturismoid) ON DELETE CASCADE, CONSTRAINT "FK\_turismo\_usuario\_usuarioid" FOREIGN KEY (usuarioid)

REFERENCES usuario (usuarioid) ON DELETE CASCADE);

Fonte: Elaborado pelos autores.

**Quadro 50 —** Script SQL – Tabela Atração

CREATE TABLE atracao (

atracaoid integer GENERATED BY DEFAULT AS IDENTITY, nome text NOT NULL,

descricao text NOT NULL, qrcode text NOT NULL, idtipoatracao integer NOT NULL, idturismo integer NOT NULL, usuarioid integer NOT NULL, statusatracao integer NOT NULL,

"TipoAtracaoModelId" integer NULL,

CONSTRAINT "PK\_atracao" PRIMARY KEY (atracaoid),

CONSTRAINT "FK\_atracao\_tipoatracao\_TipoAtracaoModelId" FOREIGN KEY ("TipoAtracaoModelId") REFERENCES tipoatracao (tipoatracaoid),

CONSTRAINT "FK\_atracao\_tipoatracao\_idtipoatracao" FOREIGN KEY (idtipoatracao) REFERENCES tipoatracao (tipoatracaoid) ON DELETE CASCADE,

CONSTRAINT "FK\_atracao\_turismo\_idturismo" FOREIGN KEY (idturismo) REFERENCES turismo (turismoid) ON DELETE CASCADE,

CONSTRAINT "FK\_atracao\_usuario\_usuarioid" FOREIGN KEY (usuarioid) REFERENCES usuario (usuarioid) ON DELETE CASCADE);

Fonte: Elaborado pelos autores.

**Quadro 51 —** Script SQL – Tabela Empresa

CREATE TABLE empresa (

empresaid integer GENERATED BY DEFAULT AS IDENTITY, nome character varying(50) NOT NULL,

cnpj text NOT NULL, endereco text NOT NULL,

tipoturismoid integer NOT NULL, usuarioid integer NOT NULL, statusatracao integer NOT NULL, "TipoTurismoModelId" integer NULL,

CONSTRAINT "PK\_empresa" PRIMARY KEY (empresaid),

CONSTRAINT "FK\_empresa\_tipoturismo\_TipoTurismoModelId" FOREIGN KEY ("TipoTurismoModelId") REFERENCES tipoturismo (tipoturismoid),

CONSTRAINT "FK\_empresa\_tipoturismo\_tipoturismoid" FOREIGN KEY (tipoturismoid) REFERENCES tipoturismo (tipoturismoid) ON DELETE CASCADE, CONSTRAINT "FK\_empresa\_usuario\_usuarioid" FOREIGN KEY (usuarioid)

REFERENCES usuario (usuarioid) ON DELETE CASCADE);

Fonte: Elaborado pelos autores.

**Quadro 52 —** Script SQL – Tabela AtracaoImagem

CREATE TABLE imagematracao (

imagematracaoid integer GENERATED BY DEFAULT AS IDENTITY, imagem text NOT NULL,

legendaimagem text NOT NULL, idatracao integer NOT NULL, usuarioid integer NOT NULL,

CONSTRAINT "PK\_imagematracao" PRIMARY KEY (imagematracaoid),

CONSTRAINT "FK\_imagematracao\_atracao\_idatracao" FOREIGN KEY (idatracao) REFERENCES atracao (atracaoid) ON DELETE CASCADE,

CONSTRAINT "FK\_imagematracao\_usuario\_usuarioid" FOREIGN KEY (usuarioid) REFERENCES usuario (usuarioid) ON DELETE CASCADE);

Fonte: Elaborado pelos autores.

**Quadro 53 —** Script SQL – Tabela NoticiaImagem

CREATE TABLE imagemnoticia (

imagemnoticiaid integer GENERATED BY DEFAULT AS IDENTITY, legendaimagem text NOT NULL,

imagem text NOT NULL, idnoticia integer NOT NULL, usuarioid integer NOT NULL,

CONSTRAINT "PK\_imagemnoticia" PRIMARY KEY (imagemnoticiaid), CONSTRAINT "FK\_imagemnoticia\_noticia\_idnoticia" FOREIGN KEY (idnoticia)

REFERENCES noticia (noticiaid) ON DELETE CASCADE,

CONSTRAINT "FK\_imagemnoticia\_usuario\_usuarioid" FOREIGN KEY (usuarioid) REFERENCES usuario (usuarioid) ON DELETE CASCADE);

Fonte: Elaborado pelos autores.

**Quadro 54 —** Script SQL – Tabela TurismoImagem

CREATE TABLE imagemturismo (

imagemturismoid integer GENERATED BY DEFAULT AS IDENTITY, imagem text NOT NULL,

legendaimagem text NOT NULL, idturismo integer NOT NULL, usuarioid integer NOT NULL,

CONSTRAINT "PK\_imagemturismo" PRIMARY KEY (imagemturismoid),

CONSTRAINT "FK\_imagemturismo\_turismo\_idturismo" FOREIGN KEY (idturismo) REFERENCES turismo (turismoid) ON DELETE CASCADE,

CONSTRAINT "FK\_imagemturismo\_usuario\_usuarioid" FOREIGN KEY (usuarioid) REFERENCES usuario (usuarioid) ON DELETE CASCADE);

Fonte: Elaborado pelos autores.

**Quadro 55 —** Script SQL – Tabela Auditoria

CREATE TABLE "Auditoria" ( "Id" uuid NOT NULL, "Data" text NOT NULL, "Hora" text NOT NULL, "Operacao" text NOT NULL,

"NomeEntidade" text NOT NULL, "ValoresAntigos" text NULL, "NovosValores" text NULL, "IdUsuario" integer NOT NULL,

CONSTRAINT "PK\_Auditoria" PRIMARY KEY ("Id"));

Fonte: Elaborado pelos autores.

**Quadro 56 —** Script SQL – Tabela Parametro

CREATE TABLE parametro (

parametroid integer GENERATED BY DEFAULT AS IDENTITY, "definicaoTurismo" text NOT NULL,

beneficios text NOT NULL, datafundacao date NOT NULL, "areaTerritorial" text NOT NULL, "distanciaCapital" text NOT NULL, habitantes integer NOT NULL, descricaoentrerios text NOT NULL, imagementrerios text NOT NULL, descricaoit text NOT NULL, imagemit text NOT NULL,

CONSTRAINT "PK\_parametro" PRIMARY KEY (parametroid));

Fonte: Elaborado pelos autores.

**Quadro 57 —** Script SQL – Tabela Ata

CREATE TABLE ata (

idata uuid NOT NULL,

tituloata character varying(100) NOT NULL, documentoata bytea NOT NULL,

dataata date NOT NULL,

CONSTRAINT "PK\_ata" PRIMARY KEY (idata));

Fonte: Elaborado pelos autores.

**Quadro 58 —** Script SQL – Tabela Membros

CREATE TABLE membros (

membroid integer GENERATED BY DEFAULT AS IDENTITY, nome character varying(50) NOT NULL,

cargo text NOT NULL, imagem text NOT NULL,

CONSTRAINT "PK\_membros" PRIMARY KEY (membroid));

Fonte: Elaborado pelos autores.

**Quadro 59 —** Script SQL – Tabela Sessao

CREATE TABLE sessao (

idssessao integer GENERATED BY DEFAULT AS IDENTITY, tokensessao text NOT NULL,

statussessao boolean NOT NULL, emailusuario text NOT NULL, nomeusuario text NOT NULL, nivelacesso text NOT NULL, "UsuarioModelId" integer NULL, idusuario integer NOT NULL,

CONSTRAINT "PK\_sessao" PRIMARY KEY (idssessao), CONSTRAINT "FK\_sessao\_usuario\_UsuarioModelId" FOREIGN

KEY

("UsuarioModelId") REFERENCES usuario (usuarioid));

Fonte: Elaborado pelos autores.

**Quadro 60 —** Script SQL – Tabela AvaliacaoTurismo

CREATE TABLE "avaliacaoTurismo" (

avaliacaoturismoid integer GENERATED BY DEFAULT AS IDENTITY, statusavaliacaoturismo integer NOT NULL,

idavaliacao integer NOT NULL, idturismo integer NOT NULL,

CONSTRAINT "PK\_avaliacaoTurismo" PRIMARY KEY (avaliacaoturismoid),

CONSTRAINT "FK\_avaliacaoTurismo\_avaliacao\_idturismo" FOREIGN KEY (idturismo) REFERENCES avaliacao (avaliacaoid) ON DELETE CASCADE,

CONSTRAINT "FK\_avaliacaoTurismo\_turismo\_idturismo" FOREIGN KEY (idturismo) REFERENCES turismo (turismoid) ON DELETE CASCADE);

Fonte: Elaborado pelos autores.

**Quadro 61 —** Script SQL – Tabela AvaliacaoAtracao

CREATE TABLE "avaliacaoAtracao" (

avaliacaoatracaoid integer GENERATED BY DEFAULT AS IDENTITY, statusavaliacaoatracao integer NOT NULL,

idavaliacao integer NOT NULL, idatracao integer NOT NULL,

CONSTRAINT "PK\_avaliacaoAtracao" PRIMARY KEY (avaliacaoatracaoid),

CONSTRAINT "FK\_avaliacaoAtracao\_atracao\_idatracao" FOREIGN KEY (idatracao) REFERENCES atracao (atracaoid) ON DELETE CASCADE,

CONSTRAINT "FK\_avaliacaoAtracao\_avaliacao\_idatracao" FOREIGN KEY (idatracao) REFERENCES avaliacao (avaliacaoid) ON DELETE CASCADE);

Fonte: Elaborado pelos autores.

* 1. Mapeamento Objeto Relacional – ORM

De acordo com Silberschatz et al. (2012), O Mapeamento Objeto-Relacional (ORM) é uma técnica que facilita a interação entre sistemas orientados a objetos e bancos de dados relacionais. Com ele é possível ter a representação dos dados do banco como objetos no código da aplicação, eliminando a necessidade de escrever comandos SQL manualmente para as operações de inserção, atualização, visualização e exclusão dos dados. Com o ORM o desenvolvimento do sistema se torna mais simples, melhorando a legibilidade e facilitando a manutenção do código.

Para o desenvolvimento desse projeto, foi utilizado o Entity Framework, um dos frameworks ORM mais utilizados para linguagem C#. Ele permite que as classes da aplicação sejam mapeadas automaticamente para tabelas no banco de dado, sem precisar de configurações complexas, utilizando as configurações personalizadas pelo desenvolvedor (Silberschatz et al, 2012).

O processo de mapeamento com o Entity Framework, segundo Silberschatz et al. (2012), ocorre de forma transparente, deixando as preocupações relacionadas a criação de tabelas ou com a escrita de comandos SQL para manipular os dados para o framework. Além disso, ele permite gerenciar relações entre diferentes entidades, como relações de um-para- muitos ou muitos-para-muitos, facilitando também o controle de transações e validações de dados.

Um exemplo no contexto do projeto é a entidade que representa os pontos turísticos cadastrados, mapeada diretamente para uma tabela no banco de dados, sendo todas as operações de controle de dados tratadas pelo Entity Framework. Resultando no desenvolvimento agilizado, reduzindo a complexidade e facilitando a manutenção do código.

Em suma, a utilização do ORM como o Entity Framework possibilitou um trabalho mais eficiente, permitindo que o foco esteja na criação de funcionalidades, enquanto o framework lida com a camada de persistência de dados.

# ARQUITETURA DE SOFTWARE

A arquitetura de software constitui a estrutura essencial que orienta o desenvolvimento da aplicação, traduzindo requisitos do sistema em uma representação tangível (BASS; CLEMENTS; KAZMAN, 2016). No âmbito do projeto Comtur, a arquitetura de software representa a espinha dorsal do sistema, proporcionando uma visão organizada e coesa de como os componentes interagem entre si. Como ressaltado por KEELING (2019), a arquitetura de software é o "esqueleto" que sustenta a aplicação, delineando as relações entre os diversos módulos e garantindo uma estrutura robusta. A necessidade de uma arquitetura sólida no Comtur transcende uma simples organização do código. Conforme preconiza Fowler (2012) "a arquitetura é importante para lidar com a complexidade inerente ao desenvolvimento de software". No contexto do Comtur, a complexidade do sistema exige uma arquitetura bem definida para facilitar não apenas a implementação atual, mas também a manutenção futura e a expansão do projeto.

* 1. Arquitetura de Desenvolvimento

A arquitetura de desenvolvimento do Comtur é utilizada para definir a estrutura e a organização do código, bem como as interações entre os diferentes componentes do sistema. Como o Comtur é um sistema em Web *API* (Interface de Programação de Aplicações Web) e foi dividido entre back-end e front-end, essa arquitetura é importante para garantir que o sistema seja modular e de fácil manutenção. Web *API* permite a comunicação entre sistemas e aplicações pela web, utilizando protocolos como HTTP. Segundo Pressman (2021), tomar boas decisões durante a definição da arquitetura de um software é essencial para o sucesso do produto, pois ela define a estrutura do sistema e impacta diretamente sua qualidade.

Uma Web *API* é um conjunto de definições e protocolos utilizados para criar e integrar softwares de aplicações. De acordo com a Microsoft (2024), a Web *API* permite a criação de serviços que atendem uma grande diversidade de clientes, como navegadores e dispositivos móveis, utilizando o mesmo framework e padrões tanto para páginas da web quanto para serviços, tudo dentro de um único projeto.

Cada parte do projeto desempenha um papel específico no fluxo de dados e na regra de negócio. A estrutura do *back-end* é desenvolvida em C# e é composta por várias camadas, cada uma responsável por uma parte distinta da aplicação. Já no *front-end*, o Comtur foi desenvolvido utilizando React,, uma biblioteca que facilita a criação de componentes reutilizáveis para interfaces de usuário, permitindo uma atualização eficiente da interface sem

a necessidade de recarregar a página inteira, melhorando a performance e experiência do usuário (REACT, 2024).

* + 1. *Back-end*

O *back-end* do Comtur foi desenvolvido utilizando C#, que é amplamente reconhecida como uma das linguagens mais importantes para o desenvolvimento .NET. Schildt (2009) descreve essa linguagem como uma das principais da Microsoft, destacando sua relevância no contexto do desenvolvimento de sistemas modernos. Skeet (2019) também aponta que C# é uma linguagem orientada a objetos e moderna, amplamente empregada na criação de aplicações robustas e de alto desempenho. Essas características tornam a escolha do C# adequada para atender às demandas de um sistema complexo e baseado em dados, como o Comtur.

A arquitetura do *back-end* do Comtur foi estruturada em diversas camadas, cada uma desempenhando um papel específico. Fowler (2012) destaca que a separação de responsabilidades em camadas torna os sistemas mais modulares, facilitando sua manutenção e escalabilidade. No contexto do Comtur, a camada que gerencia as requisições recebidas da *API* atua como o ponto de entrada para as interações com o sistema, processando solicitações, validando dados e encaminhando ações para as camadas apropriadas. Freeman e Robson (2020) descrevem que essa abordagem, baseada no padrão MVC, organiza as responsabilidades de forma clara, promovendo eficiência no processamento de dados e na interação com o usuário.

A manipulação de dados no Comtur ocorre por meio de uma camada específica que implementa a lógica de acesso e persistência ao banco de dados, desempenhando o papel de interface entre a lógica de negócios e o armazenamento. De acordo com Evans (2004), o uso de repositórios proporciona uma abstração importante, que simplifica a lógica de negócios e facilita o teste de componentes. Na camada responsável pela lógica de negócios central do sistema, as operações mais complexas e as regras de negócio são aplicadas. Martin (2017) ressalta que a separação entre lógica de negócios e persistência é essencial para garantir flexibilidade e reduzir o acoplamento entre os componentes do sistema.

Outra camada fundamental é a que gerencia a conexão com o banco de dados, garantindo o mapeamento de dados entre as entidades do sistema e as tabelas do banco de dados. Lerman (2019) observa que o uso de contextos no *Entity Framework* simplifica a configuração das conexões e o mapeamento de dados, reduzindo a complexidade de operações e aumentando a eficiência do sistema. Além disso, os arquivos *Program.cs* e *Startup.cs* desempenham um papel crítico na configuração e inicialização da aplicação, definindo dependências e serviços necessários. Conforme Proise (2021), essa etapa inicial de configuração é essencial para

assegurar que os serviços sejam injetados e inicializados corretamente, prevenindo problemas em tempo de execução.

Com essa organização em camadas, o Comtur apresenta uma arquitetura modular que facilita a manutenção e a expansão do sistema, além de contribuir para o desenvolvimento contínuo. A estrutura assegura que cada componente tenha responsabilidades bem definidas, resultando em uma aplicação robusta e eficiente, conforme defendido por diversos autores especializados em arquitetura de *software*.

* + 1. FRONT-END

O *front-end* é a parte do desenvolvimento de uma solução responsável por se comunicar com o servidor e proporcionar interações com o usuário por meio de uma interface. Segundo Meloni (2018), o *front-end* é fundamental, pois é a área da aplicação web com a qual o usuário interage diretamente. Ele é responsável pela apresentação visual dos dados e pela garantia de uma interação eficiente e fácil para o usuário.

O *front-end* foi desenvolvido utilizando o React, uma biblioteca JavaScript para criação de interfaces de usuário. Esta biblioteca permite a construção de componentes que gerenciam seus próprios estados e se combinam formar telas complexas (React, 2024). Como interface gráfica da aplicação, o *front-end* reúne elementos como menus, botões, imagens e *layout*, com o objetivo de proporcionar uma experiência fluída e intuitiva ao usuário, facilitando sua navegação e interação com a aplicação (TOTVS, 2021).

Além de facilitar a criação de interfaces complexas, o React contribui para uma organização de código mais modular e escalável. Cada componente é independente e atualizado de forma isolada, permitindo uma manutenção mais ágil e reduz o risco de falhas ao realizar alterações em partes específicas do código. Essa modularidade facilita o desenvolvimento incremental, em que novos recursos podem ser adicionados de forma contínua, sem impacto negativo no restante do sistema. Ademais, o React é amplamente suportado pela comunidade de desenvolvedores, com uma documentação abrangente de funcionalidades especificas, o que aumenta ainda mais a eficiência no desenvolvimento e na escalabilidade do projeto (React, 2024).

* 1. Segurança da Informação

A Segurança da Informação, conforme descrito por Fontes (2008), refere-se a um conjunto de políticas e práticas que têm como objetivo proteger dados e informações sensíveis contra divulgações indevidas, acessos não autorizados, alterações e exclusões. Com o aumento

da conectividade, a proteção dos dados tornou-se essencial para empresas de diversos segmentos, uma vez que a violação de dados pode acarretar sérias consequências, incluindo prejuízos financeiros, danos à reputação e implicações legais. Portanto, é crucial implementar medidas eficazes de segurança que assegurem a confidencialidade, a disponibilidade e a integridade das informações, garantindo assim um ambiente confiável e seguro para os usuários.

Nesse contexto, o projeto utiliza o JSON Web Token (JWT), que é um padrão aberto descrito por Jones, Bradley e Sakimura (2015) na RFC 7519. Ele define um formato compacto e autossuficiente para a transmissão de informações entre partes na forma de um objeto JSON. A escolha do JWT é justificada por sua capacidade de proporcionar uma maneira segura e eficiente de autenticar e autorizar usuários, permitindo, assim, a troca de informações de forma segura entre cliente e servidor (JWT.IO, 2024).

O processo de geração do token JWT ocorre no momento do login do usuário, onde as credenciais são validadas e um *payload* com informações relevantes é criado, seguido pela assinatura do token com uma chave secreta. O token é composto por três partes: o header, que contém informações sobre o tipo do token e o algoritmo de assinatura; o *payload*, que armazena as informações a serem transmitidas; e a *signature*, que garante a autenticidade do emissor (JWT.IO, 2024).

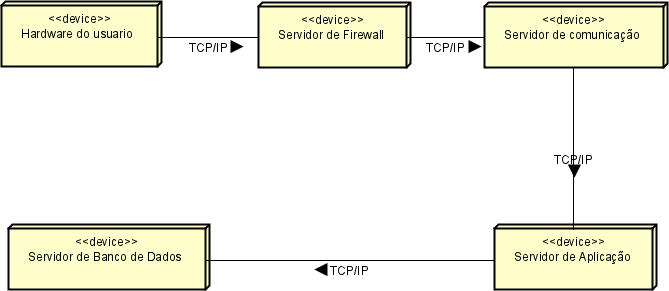
Em cada requisição que requer autenticação, o token é enviado no cabeçalho *Authorization*. O servidor, então, válida a presença do token, a assinatura e se o token ainda não expirou. A segurança do JWT é garantida pelo uso de algoritmos de assinatura robustos e pela seleção de uma chave secreta forte. Ademais, é fundamental implementar a expiração dos tokens e a possibilidade de revogação. Assim, a implementação do JWT no Comtur proporciona uma camada adicional de segurança, além de uma experiência de usuário fluida, permitindo uma autenticação eficiente em sistemas web (JWT.IO, 2024).

* 1. Implantação do Sistema

O diagrama de implantação é uma representação gráfica destinada a planejar e organizar a infraestrutura de hardware em que o software será executado. Ele descreve as máquinas que suportarão o sistema, como computadores e servidores, e define a configuração de conexões e os protocolos necessários para comunicação e a transmissão de dados entre esses dispositivos. Além disso, o diagrama também permite identificar a distribuição dos módulos em múltiplos servidores, um aspecto essencial em sistemas que operam em ambientes distribuídos. Sua aplicabilidade se justifica em sistemas que necessitam de múltiplas máquinas para execução de módulos ou armazenamento de arquivos essenciais (GUEDES, 2018).

Para ilustrar o planejamento e a organização dessa infraestrutura de hardware, o diagrama de implantação, representado na imagem a seguir, demonstra graficamente a distribuição dos módulos do sistema e a comunicação entre os diferentes servidores e dispositivos que sustentam o ambiente.

**Figura 37 -** Diagrama de Implantação



Fonte: Elaborado pelos autores.

Para suportar essa infraestrutura de forma ágil e modular, o docker é uma plataforma de virtualização leve que permite empacotar, distribuir e executar aplicações em contêineres, garantindo ambientes consistentes e padronizados para desenvolvimento, teste e produção. Cada contêiner inclui a aplicação e suas dependências, como bibliotecas e configurações, assegurando funcionamento uniforme em diferentes sistemas operacionais, reduzindo conflitos de compatibilidade e oferecendo eficiência e flexibilidade, otimizando recursos do sistema, simplificando implantação e escalonamento de aplicações em nuvem ou servidores locais, tornando-se fundamental para equipes de desenvolvimento que buscam agilidade e confiabilidade na entrega de software (DOCKER, 2024).

A implantação de sistemas distribuídos requer ferramentas que facilitem a configuração, automação e gerenciamento de múltiplas aplicações de forma eficiente e segura. Nesse contexto, o Docker se destaca como uma plataforma poderosa para criação e gerenciamento de contêineres, que são unidades de software leves e independentes que empacotam o código e todas as suas dependências. Com o Docker, é possível configurar ambientes de desenvolvimento e produção que garantem a consistência, independência de

infraestrutura e portabilidade entre diferentes ambientes, como máquinas de desenvolvimento e servidores em nuvem. Neste trabalho, abordaremos a implantação de um sistema completo utilizando Docker para integrar o *front-end*, o *back-end* e o banco de dados, demonstrando as configurações necessárias para que o sistema funcione de maneira integrada e eficiente (DOCKER, 2024).

Para a configuração do ambiente de *back-end* no Docker, é necessário criar um arquivo denominado *Dockerfile*, o qual deve ser posicionado no diretório onde se encontra o arquivo

.sln do projeto. Esse *Dockerfile* conterá as instruções fundamentais para a construção do contêiner que executará a aplicação. Posteriormente, é essencial compilar a aplicação juntamente com todas as suas dependências, o que pode ser feito por meio do comando dotnet publish -c Release, que gera o build e prepara o projeto para execução.

No *front-end*, dentro do diretório correspondente, é necessário criar um arquivo .env que contenha a URL da API. Esse arquivo deve seguir a estrutura: VITE\_API\_URL=https://localhost:7256/api/. Essa URL deve corresponder à porta na qual a API do *back-end* está em execução. Além disso, um *Dockerfile* deve ser incluído no diretório do *front-end*, utilizado para compilar e executar a aplicação, incluindo instruções para instalação das dependências, cópia dos arquivos do projeto e definição de variáveis de ambiente. Este *Dockerfile* inicia com uma imagem node:18 para construir o front-end e, em seguida, usa o nginx:alpine para servir a aplicação compilada. Uma configuração de Nginx é necessária no arquivo nginx.conf, que especifica o servidor para redirecionar requisições para a aplicação front-end e, quando necessário, para o contêiner da API.

A configuração do banco de dados PostgreSQL no ambiente Docker requer a criação de uma pasta denominada *Database*, localizada fora dos diretórios do *front-end* e do *back-end*. Nessa pasta, o comando pg\_dump -U postgres -d sgedDB -h localhost -p 5432 -f ComturDB.sql deve ser executado para exportar o banco de dados, resultando em um arquivo ComturDB.sql que contém todos os dados necessários. Esse arquivo será então utilizado para inicializar o banco de dados no contêiner.

A integração entre os serviços de *back-end*, *front-end* e banco de dados ocorre por meio do arquivo docker-compose.yml, o qual define as configurações de cada serviço, incluindo portas, variáveis de ambiente e dependências entre os contêineres. No docker- compose.yml, o serviço PostgreSQL é iniciado primeiro, com variáveis de ambiente para usuário, senha e nome do banco, além de um volume para garantir a persistência de dados. O serviço da API depende do PostgreSQL e só é inicializado quando o banco de dados estiver em estado saudável. Por fim, o serviço de *front-end* depende da API, e é iniciado após a

configuração dos demais serviços. Esse processo garante que todos os contêineres sejam inicializados na ordem correta e que todas as dependências estejam estabelecidas.

Para manter o ambiente Docker organizado e evitar resíduos de configurações anteriores, recomenda-se a execução do comando docker-compose down --rmi all -v, o qual remove todos os contêineres e imagens previamente criados. Em seguida, para construir e inicializar os contêineres, deve-se executar o comando docker-compose up --build. Esse procedimento cria e inicia o ambiente completo, integrando o banco de dados, a API e o *front- end*, e permitindo o acesso a cada serviço conforme as portas especificadas. Essa estruturação do ambiente com Docker proporciona um ambiente de desenvolvimento e produção coeso e modular, ideal para aplicações distribuídas.

# CONCLUSÃO

À medida que a internet se consolida como uma ferramenta crucial no cenário do turismo, torna-se fundamental que as cidades adotem estratégias inovadoras para promover seus destinos. Nesse contexto, o projeto Comtur é apresentado como uma solução para impulsionar o turismo em Jales, preenchendo lacunas existentes e oferecendo uma experiência rica em conteúdo e interatividade.

A análise detalhada dos desafios atuais revelou a ausência de uma presença online específica para o turismo na cidade. Fatores como a concentração do turismo na área de saúde, a falta de divulgação de locais turísticos menos conhecidos e a inexistência de um sistema eficaz para conectar pontos culturais foram identificados como obstáculos importantes. O Comtur busca abordar esses desafios e convertê-los em oportunidades para o crescimento e desenvolvimento local.

Os requisitos funcionais foram delineados para incluir funcionalidades abrangentes, como botões de ação para atualização de registros e um sistema de pesquisa avançado. Essas características visam não apenas destacar pontos turísticos, mas também proporcionar uma experiência interativa para os usuários, conectando-os de maneira significativa à riqueza cultural de Jales.

O Comtur vai além de ser um guia turístico online, buscando integrar-se à comunidade local por meio de futuras funcionalidades. Entre elas, destaca-se a inclusão de um leitor de QR Code, que permitirá aos usuários acessar informações detalhadas sobre possíveis atrações turísticas ao escanear códigos disponíveis em locais específicos. Essa funcionalidade visa enriquecer a experiência dos visitantes, facilitando o acesso a conteúdos interativos e atualizados sobre pontos de interesse. Essas implementações têm como objetivo promover um ambiente online mais informativo, interativo e seguro.

Por fim, a ferramenta tem o potencial de impactar o setor de turismo em Jales, contribuindo para a geração de empregos, a criação de oportunidades de negócios e a melhoria da qualidade de vida de residentes e visitantes. Com sua abordagem inovadora e o fortalecimento da conexão comunitária, o Comtur se posiciona como um projeto que pode impulsionar o desenvolvimento sustentável e transformar a experiência turística da cidade.

# REFERÊNCIAS

BASS, Len; CLEMENTS, Paul; KAZMAN, Rick. **Arquitetura de software: padrões, práticas e estilos.** Tradução de Luiz Souza. 1. ed. São Paulo: Bookman, 2016.

BRASIL. **A importância da internet para o turismo.** Disponível em: <https://www.gov.br/turismo/pt-br/assuntos/noticias/a-importancia-da-internet-para-o-turismo>. Acesso em: 11 jun. 2024.

BOOCH, G. **Object-Oriented Analysis and Design with Applications.** Addison-Wesley, 1991.

BUXTON, B. **Sketching User Experiences: Getting the Design Right and the Right Design.** Morgan Kaufmann, 2007.

DESTINOS INTELIGENTES. **Destinos Inteligentes.** Disponível em: [https://www.destinosinteligentes.tur.br.](https://www.destinosinteligentes.tur.br/) Acesso em: 4 jun. 2024.

DOCKER. **Docker Overview.** Disponível em: <https://www.docker.com/>. Acesso em: 2 nov. 2024.

EVANS, E. **Domain-driven design: tackling complexity in the heart of software.** Boston: Addison-Wesley, 2004.

FONTES, E. **Praticando a Segurança da Informação.** [s.l.]: BRASPORT, 2008.

FOWLER, M. **Patterns of enterprise application architecture.** Boston: Addison-Wesley, 2012.

FREEMAN, A.; ROBSON, S. **Head First Design Patterns: building extensible and maintainable object-oriented software.** 2. ed. Sebastopol: O'Reilly Media, 2020.

GUARIZO, M. A.; SCHWARTZ, G. M. Usabilidade de Sites nos Contextos do Lazer, Turismo e Hospitalidade: Uma Análise da Produção Científica. **LICERE - Revista do Programa de Pós-graduação Interdisciplinar em Estudos do Lazer**, Belo Horizonte, v. 23,

n. 2, p. 362–385, 2020. Disponível em: <https://www.periodicos.ufmg.br/index.php/licere/article/view/24050>. Acesso em: 17 jun. 2024.

GUEDES, Gilleanes T. A**. UML: uma abordagem prática.** 3. ed. São Paulo: Novatec, 2018. JWT. **JSON Web Tokens - JWT.** Disponível em: <https://jwt.io/>. Acesso em: 2 nov. 2024.

KEELING, Michael. **Design de software: como construir aplicações de alta qualidade.**

Tradução de Pedro Silva. Rio de Janeiro: Alta Books, 2019.

KRUG, Steve. **Não me faça pensar: a abordagem do bom senso para a usabilidade na web.** Tradução de Fernanda Lazzarini. 2. ed. Rio de Janeiro: Elsevier, 2010.

LERMAN, J. **Programming Entity Framework: Code First.** Sebastopol: O'Reilly Media, 2019.

MACHADO, F. N. R. **Banco de dados: projeto e implementação.** 4. ed. São Paulo: Érica, 2020.

MARTIN, R. C. **Clean Architecture: a craftsman’s guide to software structure and design.** Boston: Prentice Hall, 2017.

MELONI, Julie C. **HTML, CSS, and JavaScript All in One.** 4. ed. Indianapolis: Sams Publishing, 2018.

MICROSOFT. **Microsoft, 2024. APIs com o ASP.NET Core.** Disponível em: <https://dotnet.microsoft.com/pt-br/apps/aspnet/apis>. Acesso em: 12 jul. 2024.

MILANI, A. **PostgreSQL: guia do programador.** 1. ed. São Paulo: Novatec, 2008. NIELSEN, J. **Ten Usability Heuristics.** 2006.

NORMAN, D. A. **The Design of Everyday Things.** Basic Books, 2002.

PAULA FILHO, W. P. **Engenharia de software: fundamentos, métodos e padrões.** 3. ed. Rio de Janeiro: LTC, 2015.

PREFEITURA MUNICIPAL DE ARAÇATUBA. **Prefeitura Municipal de Araçatuba.**

Disponível em: <https://aracatuba.sp.gov.br/turismo>. Acesso em: 4 jun. 2024.

PREFEITURA DE JALES. **Instagram da Prefeitura de Jales.** Disponível em: [https://www.instagram.com/prefeituradejales.](https://www.instagram.com/prefeituradejales) Acesso em: 20 nov. 2023.

PREFEITURA MUNICIPAL DE JALES. **Prefeitura Municipal de Jales.** Disponível em: <https://jales.sp.gov.br/cidade/turismo>. Acesso em: 09 ago. 2024.

PRESSMAN, R. S. **Engenharia de Software: uma abordagem profissional.** 9. ed. Porto Alegre: AMGH, 2021.

PROISE, J. **Programming Microsoft ASP.NET Core.** Redmond: Microsoft Press, 2021.

REACT. **Uma biblioteca JavaScript para criar interfaces de usuário.** Disponível em: [https://pt-br.legacy.reactjs.org.](https://pt-br.legacy.reactjs.org/) Acesso em: 3 nov. 2024.

SANTOS, F; LUNARDI, G. L; MAIA, C. R. AÑAÑA, E. S. **Fatores que influenciam a participação dos consumidores no Turismo Eletrônico.** Disponível em: [https://www.scielo.br/j/rbtur/a/LRSwCR7ZmjGRkgChr7gC6xf/#ModalTutors](https://www.scielo.br/j/rbtur/a/LRSwCR7ZmjGRkgChr7gC6xf/" \l "ModalTutors). Acesso em: 18 jun. 2024.

SCHILDT, H. **C#: The complete reference.** 4. ed. New York: McGraw-Hill Education, 2009.

SILBERSCHATZ, A; KORTH, F. H; SUDARSHAN, S. **Sistema de Banco de Dados.** 6. ed. Rio de Janeiro: Elsevier, 2012.

SKEET, J. **C# in depth.** 4. ed. New York: Manning Publications, 2019.

SMART TOUR BRASIL. **Smart Tour.** Disponível em: [https://smarttourbrasil.com.br](https://smarttourbrasil.com.br/). Acesso em: 4 jun. 2024.

SOMMERVILLE, I. **Engenharia de Software.** 10. ed. São Paulo: Pearson Education do Brasil, 2018.

TOTVS. **Front end: O que é, como funciona e qual a importância.** 14 jul. 2021. Disponível em: <https://www.totvs.com/blog/developers/front-end/>. Acesso em: 3 nov. 2024.

UNGER, Russ; CHANDLER, Carolyn**. O guia para projetar UX.** 1. ed. São Paulo: Alta Books, 2009.

VIEIRA, Aline. **Planejamento e políticas públicas de turismo: análise dos módulos operacionais do programa de regionalização do turismo no pólo São Luis – MA.** 2011.