PROJETO INTERDISCIPLINAR 1 – INSTITUTO CRIATIVO

Requisitos da disciplina Modelagem de Software e Arquitetura de Sistemas

INTEGRANTES DO PROJETO e RA'S

Vitor Santos	RA: 23025502
Gustavo Henrique	RA: 24026874
Gustavo Roberto	RA: 24026770
Lucas Alves Bernardo	RA: 24026768
Eriane S O Dias	RA: 24026678

São Paulo

2025

Sumário

1 INTRODUÇÃO

Este projeto busca otimizar a gestão do site Instituto Criativo por meio de um Dashboard de publicação e reserva de eventos. O objetivo é centralizar informações sobre eventos, atividades e projetos educacionais, permitindo que o usuário possa reservar eventos e que pelo login do administrador seja possível fazer um monitoramento mais preciso e eficiente de engajamento. A plataforma contará com funcionalidades como visualização de cronogramas, rastreamento de progresso, divulgação de projetos, geração de relatórios e notificações.

2. DOCUMENTO DE ABERTURA DO PROJETOS

Prefácio

Este documento é destinado a gestores, desenvolvedores, educadores e parceiros envolvidos no desenvolvimento e utilização do Dashboard de Gestão do Instituto Criativo. Ele apresenta as especificações detalhadas do sistema, servindo como um guia para a compreensão de suas funcionalidades e requisitos.

Histórico de Versões:

Versão 1.0: Documento inicial contendo os requisitos fundamentais do sistema.

Versão 1.1: Revisão para inclusão de integração com redes sociais e ajustes nos requisitos de usabilidade.

Introdução

O Dashboard de Gestão do Instituto Criativo é uma ferramenta inovadora desenvolvida para otimizar a administração e acompanhamento das iniciativas da instituição. Com o objetivo de centralizar informações sobre atividades, desafios e projetos educacionais, a aplicação permitirá um monitoramento mais preciso e eficiente, auxiliando gestores, educadores e parceiros na tomada de decisões estratégicas.

Atualmente, a gestão desses projetos ocorre de forma descentralizada, dificultando a avaliação de impacto e a comunicação entre as partes interessadas. Para resolver esse problema, o dashboard oferecerá uma visão consolidada das iniciativas, proporcionando maior transparência e colaboração.

A plataforma será desenvolvida com tecnologias modernas, como .NET para o back-end, React para a interface de usuário e MySQL para o armazenamento de dados.

Entre suas funcionalidades, destacam-se a visualização de cronogramas, rastreamento de progresso, geração de relatórios automáticos e notificações sobre prazos e atualizações.

Com uma interface intuitiva e personalizável, o dashboard permitirá a filtragem de informações por projeto, período ou metas específicas, além de incluir indicadores-chave de desempenho (KPIs) para medir o impacto das iniciativas. Essa solução tecnológica visa aprimorar a eficiência do Instituto Criativo, fortalecendo sua missão de oferecer uma educação inovadora e transformadora.

Glossário

- Dashboard: Painel de controle com informações consolidadas;
- API: Interface para integração com outros sistemas;
- KPI: Indicador-chave de desempenho;
- Back-end: Parte do sistema responsável pelo processamento de dados;
- Front-end: Interface visual do sistema acessada pelos usuários.

Definição de requisitos de usuário

- Visualização de cronogramas e progresso das atividades;
- Geração automática de relatórios;
- Notificações sobre prazos e atualizações;
- Integração com redes sociais.

Arquitetura do sistema

- Front-end: Desenvolvido em React;
- Back-end: Utilizando .NET;
- Banco de Dados: MySQL;
- Integrações: APIs de redes sociais e ferramentas de relatórios.

Especificação de requisitos do sistema

Funcionais:

- Cadastro e gestão de atividades.
- Geração de relatórios personalizados.
- Notificações automáticas.
- Não Funcionais:
- Acessibilidade para diferentes perfis de usuários.
- Tempo de resposta inferior a 2 segundos para consultas padrão.

Modelos do sistema

O sistema contará com modelos gráficos, incluindo:

- Modelo de fluxo de dados: Exibindo interações entre módulos.
- Modelo de entidade-relacionamento: Definindo a estrutura do banco de dados.

Evolução do sistema

O sistema que estamos criando pressupõe que o usuário administrador irá conseguir imputar no site, eventos a serem promovidos criando assim um banco de dados que também será usado pelo usuário cliente para selecionar o evento de sua preferência e reservar. O administrador contará com uma interface que lhe permitirá acrescentar a data do evento, o local em que irá ser sediado, uma breve descrição do que acontecerá e um espaço para incluir ao menos uma foto da chamada para o evento. O cliente poderá navegar pelo site, escolher o evento de sua preferência por meio de clique direto na foto do evento, ou utilizando um filtro no dashboard do site.

Apêndices

Requisitos de Hardware:

- Servidor com suporte a .NET e MySQL.
- Mínimo de 8GB de RAM e processador Quad-Core.

Requisitos de Banco de Dados:

- Estrutura otimizada para alto volume de acessos.
- Políticas de backup e segurança implementadas.

Este documento servirá de base para o desenvolvimento e evolução do Dashboard de Gestão do Instituto Criativo.

3. REQUISITOS DE SISTEMA

3.1 REQUISITOS FUNCIONAIS DE SOFTWARE

RFS01	
Função	LogIn e SignIn
Descrição	Função Logar e Criar uma conta.
Entradas	Informações do Usuário
Fonte	Banco de dados
Saídas	Registro do Usuário.
Ação	O usuário escolherá entre as opções LogIn e SignIn. Caso ele escolha o LogIn, o usuário terá de colocar suas informações, sendo elas o Email do usuário e a Senha.

RFS02	
Função	Participar de um evento

Descrição	Função de poder fazer parte de um evento
Entradas	Informação do usuário mais preço pelo evento
Fonte	Banco de Dados
Saídas	Adesão da participação em um evento
Ação	Na aba “eventos”, caso a pessoa já esteja logada no site, a pessoa tem a opção de participar de um evento. O evento pode ou não pode ter um valor monetário, ou seja, ser pago ou gratuito. Após preencher suas informações pessoais e pagar o evento (0 reais caso seja gratuito), a pessoa estará participando de um evento.
RFS03	
Função	Criar um evento
Descrição	Ação de um criar um evento
Entradas	Informações dadas pelo criador do evento
Fonte	Banco de Dados
Saídas	Evento
Ação	Caso a pessoa possua conta de administrador, ela poderá criar um evento. Informações incluem nome do evento, local do evento, descrição do evento e valor do evento.

RFS04	
Função	Apagar Evento
Descrição	Donos de eventos poderão apagar o evento
Entradas	Pedido Para Apagar Evento
Fonte	Banco de dados
Saídas	Evento Apagado do Banco de dados
Ação	O Dono do evento poderá apagar o Evento com um pedido que será enviado para a Gestão com as informações do motivo pelo qual estará apagando o evento, a gestão se certificará do pedido do motivo da exclusão para realizar o pedido.

RFS05	
Função	Geração de Relatórios
Descrição	Administradores podem gerar relatórios sobre eventos e participação.
Entradas	Filtros de pesquisa.
Fonte	Banco de dados
Saídas	Fazer Relatórios exportáveis.
Ação	O usuário escolhe os parâmetros do relatório e gera um documento.

RFS06	
Função	Notificações Automáticas
Descrição	Envio de notificações sobre eventos e prazos importantes.
Entradas	Eventos e prazos cadastrados.
Fonte	Banco de dados
Saídas	Alertas para usuários.
Ação	O sistema dispara notificações com base em eventos programados.

3.2 REQUISITOS NÃO FUNCIONAIS DE SOFTWARE

RNFS01	
Função	Flexibilidade
Descrição	Maneira que o site de adapta a diferentes telas
Entradas	Script de flexibilidade para HTML
Fonte	Código fonte
Saídas	Site Auto Ajustável
Ação	Por meio de ajustes do Script no Código fonte do Site; Janelas, textos e imagens se ajustaram de maneira automática ao tamanho da tela atual, tornando o site mais agradável a diferentes tipos de dispositivos.

RNFS02

Função	Otimização
Descrição	Maneiras de como deixar o site mais otimizado
Entradas	Script de otimização em JavaScript
Fonte	Código Fonte
Saídas	Site mais ajustável
Ação	O site possui alguns erros de otimização, como o uso de imagens desnecessárias ou código fonte confuso, que podem tornar o site mais pesado ou repetitivo. Para evitar possíveis problemas no futuro, a otimização do código fonte é essencial, eliminando repetições ou adotando métodos mais ágeis.

RNFS03	
Função	Segurança de dados
Descrição	O Sistema precisa ser seguro a ponto de não permitir vazamento de dados dos clientes e nem dos eventos registrados
Entradas	sistema de banco de dados, criptografia
Fonte	-
Saídas	-
Ação	O sistema terá autenticação com login e senha criptografada, garantindo acesso seguro. Apenas usuários autorizados poderão visualizar ou modificar dados, com criptografia aplicada tanto em trânsito quanto em repouso para evitar acessos não autorizados.

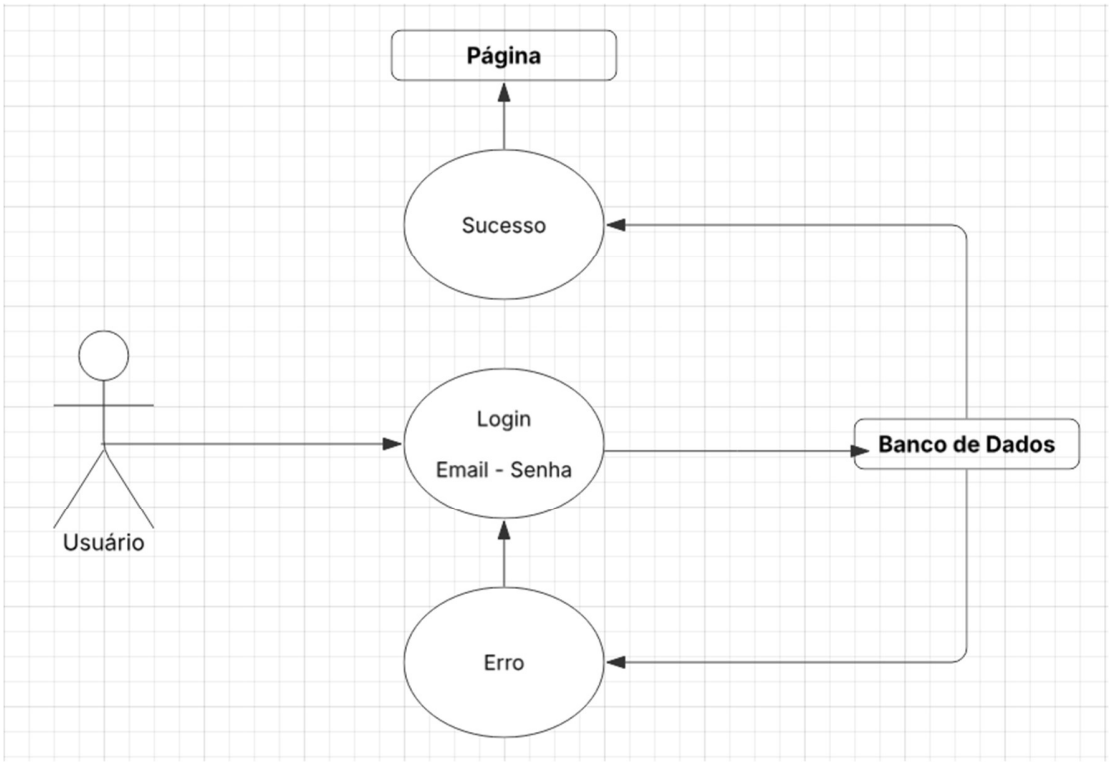
RNFS04	
Função	Velocidade de navegação
Descrição	O sistema deverá ser rápido a ponto de manter-se interessante para o usuário cliente e otimizar o tempo do usuário administrador
Entradas	Tecnologias de cache, balanceamento de carga, otimização de consultas no banco de dados, compressão de dados, e estratégias de redução de latência.
Fonte	-
Saídas	-
Ação	O sistema será otimizado para garantir tempos de resposta rápidos, utilizando técnicas de cache, redução de latência e otimização de código. Serão realizados testes de carga para assegurar uma navegação fluida para o usuário cliente e agilidade nas tarefas administrativas, mesmo em alta demanda.

RNFS05	
Função	Escalabilidade
Descrição	Capacidade de suportar aumento de usuários sem degradação de desempenho.
Entradas	Dados de usuários e eventos
Fonte	-
Saídas	-
Ação	Adoção de infraestrutura que permita crescimento sem perda de qualidade.

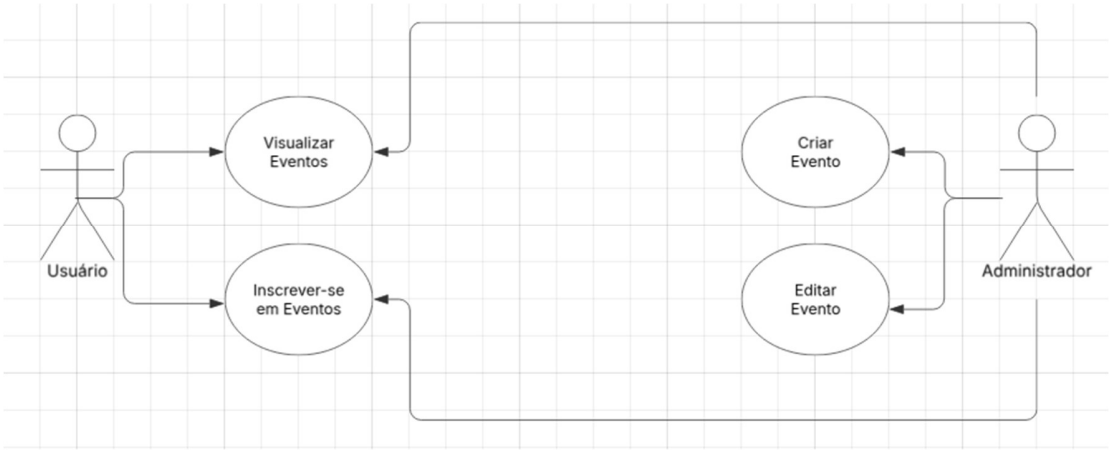
RNFS06	
Função	Compatibilidade com APIs
Descrição	Integração com redes sociais e ferramentas externas.
Entradas	Dados provenientes de redes sociais e ferramentas externas.
Fonte	APIs externas integradas ao sistema
Saídas	-
Ação	Implementação de conectores para permitir integração fluida.

4. CASOS DE USO

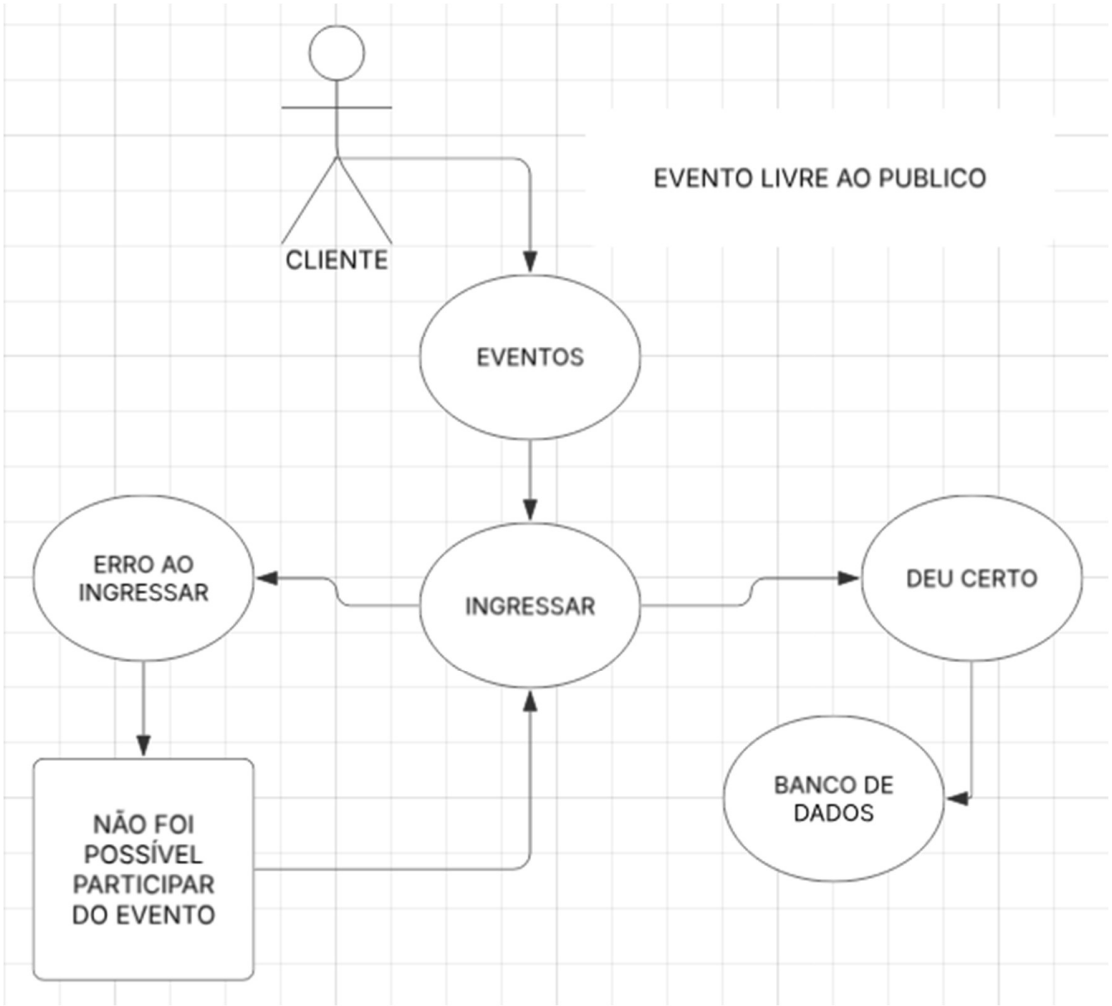
4.1 Login



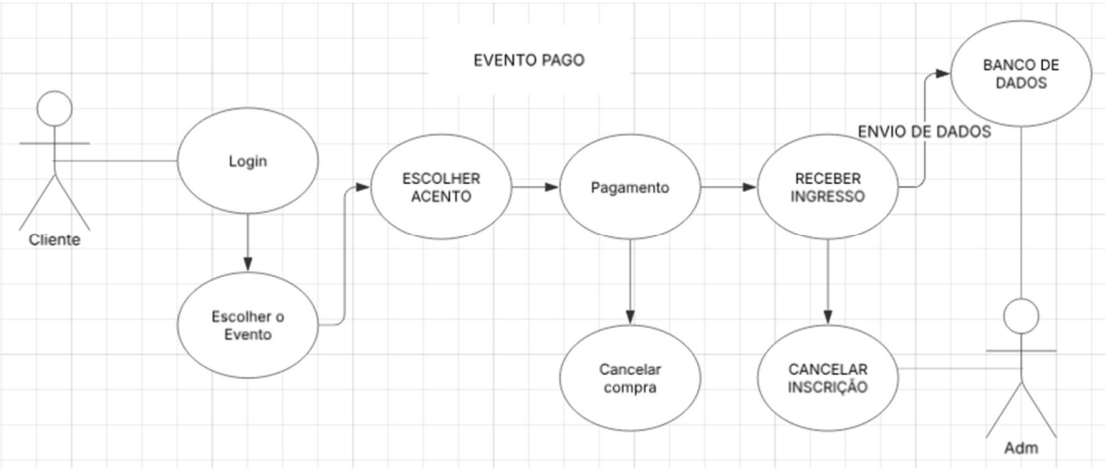
4.2 Permissões



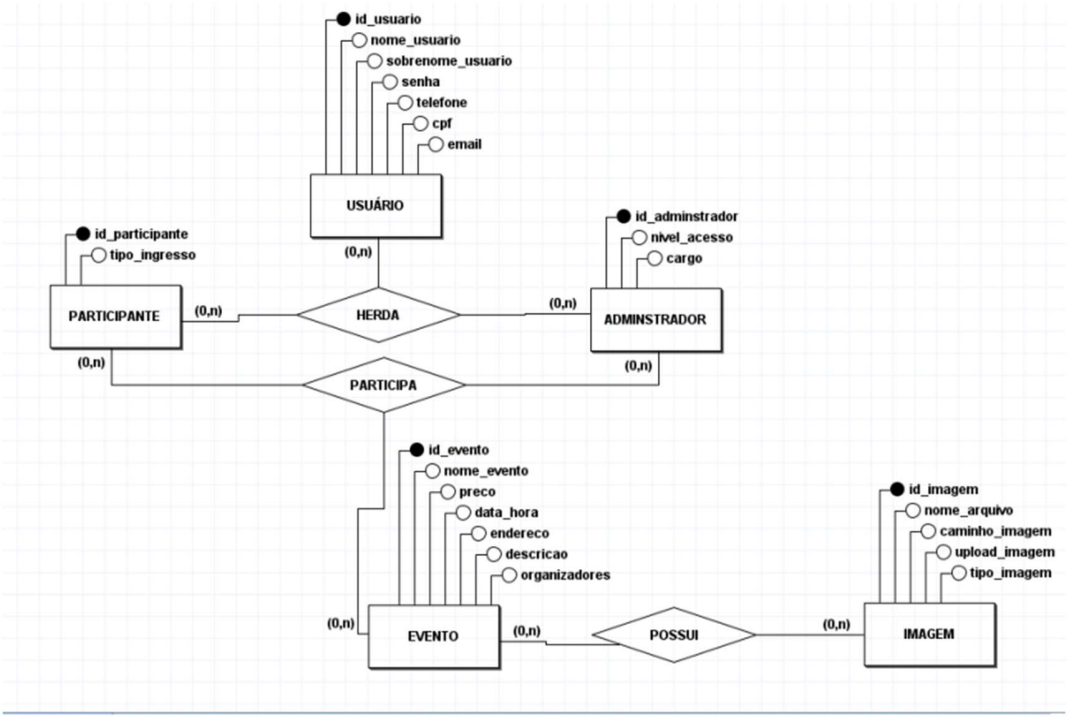
4.3 Evento Livre



4.4 Evento Pago



5. DIAGRAMA DE CLASSE



6. ARQUITETURA DO SISTEMA

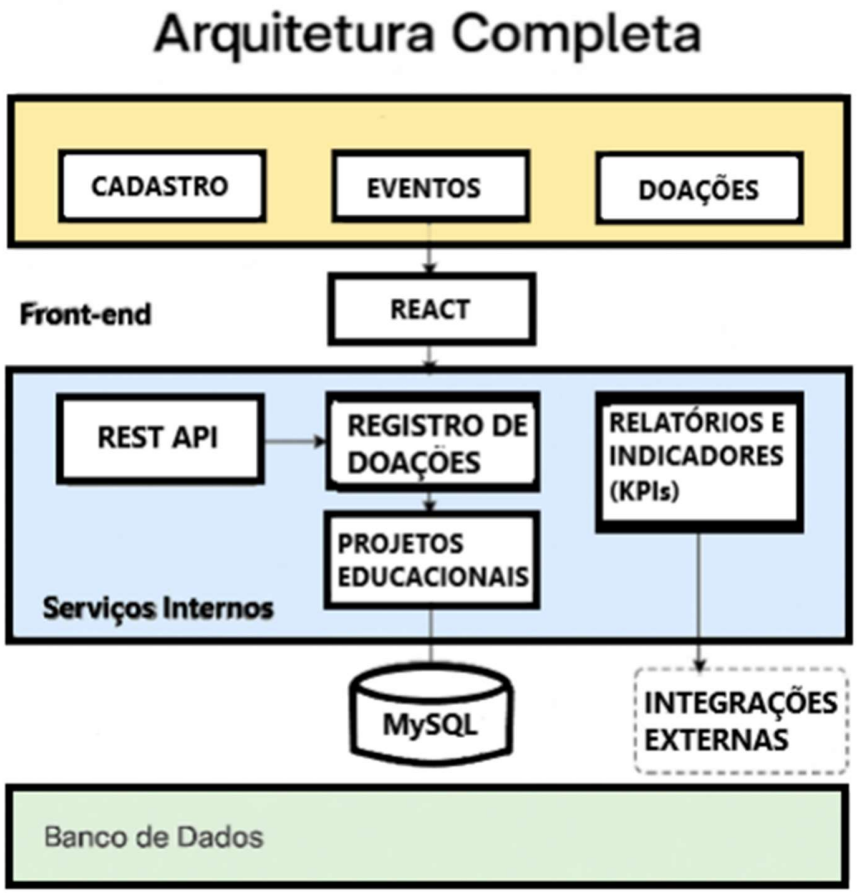
Para a arquitetura do sistema, considere o template abaixo:

Template (esquema de arquitetura) para orientar a elaboração de um **Projeto de Arquitetura de Sistema Web**, cobrindo desde a camada de apresentação até a persistência de dados, com foco em modularidade, escalabilidade e eficiência.

6.1. Visão Geral do Sistema

- Nome do Sistema: TechTeam
- Objetivo do Sistema: permitir que qualquer usuário visualize eventos disponíveis, realize cadastro, login e se inscreva em eventos.
- Público-Alvo: Gestores e Administradores do Instituto Criativo
- Principais Funcionalidades: Acompanhar desenvolvimento dos Eventos, Medir a quantidade de Vendas e Apurar Faturamento.

6.2. Arquitetura Geral



6.3. Componentes Arquiteturais e Camadas

6.3.1. Camada de Apresentação (Frontend)

A camada de apresentação é responsável pela interface com o usuário. Todas as interações visuais, formulários, botões e páginas fazem parte desta camada. As funcionalidades presentes incluem:

- Visualização de eventos sem necessidade de login
- Cadastro e autenticação de usuários

- Preenchimento de formulários
- Inscrição em eventos
- Acesso a áreas internas por usuários autenticados

A aplicação foi desenvolvida com React, utilizando JavaScript, CSS e ferramentas de build como Vite.js. A estrutura está organizada com base em diretórios que seguem a lógica de páginas e componentes, promovendo reutilização e clareza no desenvolvimento.

Pastas principais na camada de apresentação:
- components: componentes reutilizáveis de UI
- pages: páginas do site (Homepage, Login, Signup, Eventos, etc.)
- App.jsx: ponto central da aplicação com configuração de rotas
- index.jsx: ponto de entrada que renderiza o app

Esta estrutura reflete uma clara separação entre interface, layout e lógica, facilitando a evolução futura da aplicação.

Camada de Negócio (Lógica de Aplicação)

6.3.2. Camada de Aplicação (Backend/API)

A camada de negócio é responsável por centralizar as regras da aplicação, garantindo que as operações sigam a lógica correta antes de serem executadas ou enviadas para a persistência de dados.

Responsabilidades da camada de negócio:
- Validação de dados de entrada
- Controle de fluxos, como inscrição em eventos
- Processamento de regras específicas do domínio
- Comunicação com a API (camada de persistência)
- Controle de autenticação (login, acesso autorizado)
- Gerenciamento de estado intermediário da aplicação

A arquitetura sugere que esta lógica seja isolada em arquivos ou módulos específicos (como serviços), promovendo o desacoplamento entre a lógica e a interface. Isso facilita a manutenção, testes e reutilização da lógica em diferentes partes da aplicação.

6.3.3. Camada de Persistência (Banco de Dados)

A camada de persistência do sistema é responsável por armazenar, consultar e manter os dados de forma estruturada e segura. O projeto utiliza o banco de dados relacional MySQL, amplamente adotado em aplicações web, em conjunto com a biblioteca Sequelize, que atua como ORM

(Object-Relational Mapping).

O back-end é implementado em Node.js com o framework Express, e utiliza o driver mysql2 para se conectar diretamente ao banco. As rotas de comunicação são desenvolvidas em JavaScript e organizadas no servidor, permitindo a integração entre o front-end (React) e a

camada de dados.

Durante o processo de desenvolvimento, também são utilizadas bibliotecas complementares como:

- bcryptjs: para criptografia de senhas
- jsonwebtoken: para autenticação via tokens JWT
- multer: para upload de arquivos
- dotenv (via nodemon): para carregar variáveis de ambiente de forma segura

6.4. Diagrama da Arquitetura

Banco de dados MySQL

6.5. Organização de Pastas e Módulos

Descrever como o código será organizado no repositório:

```
PROJETO1/
├── code/
│   ├── client/
│   │   └── src/
│   │       ├── .gitignore
│   │       ├── index.html
│   │       ├── package-lock.json
│   │       ├── package.json
│   │       ├── README.md
│   │       ├── styleguide.css
│   │       └── vite.config.js
│   ├── server/
│   │   ├── middleware/
│   │   ├── public/
│   │   ├── db.js
│   │   ├── package-lock.json
│   │   ├── package.json
│   │   ├── routes.js
│   │   └── server.js
│   ├── documents/
│   │   ├── Entrega 1/
│   │   ├── Entrega 2/
│   │   └── Entrega 3/
│   ├── .gitignore
│   ├── banco_instituto.sql
│   └── readme.md
```

6.6. Integrações Externas (se houver)

Em desenvolvimento.

6.7. Ambiente e Implantação

- **Ambientes**: Vscode React, Figma, lucidchart
- **CI/CD**: GitHub Actions
- **Hospedagem**: Azure

7. REFERÊNCIAS BIBLIOGRÁFICAS

SOMMERVILLE, I. **Engenharia de Software.** 11ª Edição. São Paulo: Pearson Addison-Wesley, 2017.