# FECAP

## CulturaHub

**Requisitos da disciplina Modelagem de Software e Arquitetura de Sistemas**

São Paulo

2024

## INTEGRANTES DO PROJETO e RA'S

| | | |
|---|---|---|
| Felipe Oluwaseun Santos Ojo | - | 24026245 |
| Gustavo de Souza Castro | - | 20021558 |
| Marcella Santana Gonçalves Diniz Rocha | - | 24025750 |
| Thays Helyda da Silva Pontes | - | 24026610 |

**Sumário**

# 1 INTRODUÇÃO

A cultura desempenha um papel fundamental na construção da identidade de comunidades e na promoção do diálogo entre diferentes grupos sociais. Em um mundo cada vez mais globalizado, o acesso a eventos culturais se torna essencial para fortalecer laços, fomentar a criatividade e enriquecer a experiência humana. Este projeto visa desenvolver uma plataforma que não apenas facilite o gerenciamento de eventos, mas também amplifique a acessibilidade à cultura, tornando-a uma parte integral da vida cotidiana das pessoas.

Com funcionalidades que permitem a criação, busca e inscrição em eventos, o sistema se propõe a ser uma vitrine para diversas manifestações culturais, desde shows e exposições até workshops e palestras. Ao conectar organizadores e participantes, a plataforma cria um espaço dinâmico onde a cultura pode ser celebrada e compartilhada, contribuindo para a educação e o desenvolvimento social.

Além de proporcionar uma experiência intuitiva, o sistema almeja inspirar o envolvimento da comunidade, promovendo a diversidade cultural e incentivando a participação ativa dos cidadãos. Assim, o projeto se alinha com a missão de tornar a cultura acessível e relevante, valorizando o papel transformador que ela exerce na sociedade.

# 2. DOCUMENTO DE ABERTURA DO PROJETOS

## Prefácio

Este documento é destinado a desenvolvedores de software, analistas de sistemas, gerentes de projeto e stakeholders que buscam compreender os requisitos e a arquitetura do CulturaHub. Ele é uma fonte essencial para a equipe envolvida na implementação e manutenção do software, além de servir como referência para futuras atualizações e melhorias.

## Introdução

A necessidade de um sistema eficiente de gerenciamento de eventos surge da demanda crescente por plataformas que facilitam a organização e a participação em eventos de diversas naturezas, desde culturais a educacionais. Este sistema foi concebido para atender a essa necessidade, permitindo que usuários se cadastrem, realizem login, busquem eventos e se inscrevam em atividades de interesse.

As principais funções do sistema incluem o cadastro de usuários, login e autenticação, criação e pesquisa de eventos, além de permitir que os usuários se inscrevam em eventos. O sistema funcionará em sinergia com outros sistemas através de APIs, garantindo que as informações sejam acessíveis e integradas de forma segura e eficiente.

Além disso, o sistema visa proporcionar uma plataforma robusta e amigável que incentive a participação em eventos, potencializando o engajamento do público e a visibilidade dos organizadores.

## Glossário

**Cadastro de Usuários**: Processo onde um usuário fornece suas informações para criar uma conta no sistema.

**Login**: Acessar uma conta no sistema utilizando credenciais previamente cadastradas.

**Evento**: Atividade organizada, como concertos, workshops ou palestras.

**Inscrição**: Registro de interesse do usuário em participar de um evento específico.

**API**: Interface de Programação de Aplicações, que permite a comunicação entre diferentes sistemas.

### Definição de requisitos de usuário

**Cadastro e Autenticação:** Permitir que os usuários se cadastrem com informações básicas e realizem login para acessar suas contas.

**Criação de Eventos:** Usuários autorizados poderão criar eventos, especificando título, data, descrição e imagem.

**Pesquisa de Eventos:** Usuários poderão buscar eventos por nome, categoria ou localização.

**Inscrição em Eventos:** Usuários poderão salvar eventos de interesse em suas páginas pessoais.

Os requisitos não funcionais incluem:

**Desempenho:** O sistema deve carregar em até 3 segundos.

**Segurança:** Implementação de criptografia para proteção de dados.

**Responsividade:** O sistema deve ser acessível em dispositivos móveis e desktops.

**Arquitetura do sistema**

A arquitetura do sistema é composta por várias camadas que interagem entre si:

1. **Camada de Apresentação**: Utiliza HTML, CSS e JavaScript para exibir a interface do usuário.
2. **Camada de Lógica de Negócios**: Implementada em Node.js, onde as regras de negócio são processadas.
3. **Camada de Acesso a Dados**: Realiza operações de CRUD no banco de dados.
4. **Banco de Dados**: Utiliza SQLiteStudio para armazenamento de dados.
5. **Camada de Segurança**: Implementa protocolos de segurança como HTTPS.
6. **Camada de API**: Facilita a comunicação entre o frontend e o backend.

## Especificação de requisitos do sistema

Os requisitos funcionais detalhados incluem funcionalidades como cadastro de usuários, login e autenticação, criação e exibição de eventos, pesquisa de eventos, e inscrição em eventos. Os requisitos não funcionais, como desempenho, escalabilidade e segurança, são igualmente cruciais e devem ser atendidos.

## Evolução do sistema

O sistema foi projetado com a flexibilidade em mente, permitindo futuras modificações conforme as necessidades dos usuários, escalabilidade e aplicação de novas features. A evolução de hardware e software, juntamente com novas demandas do mercado, serão consideradas durante a manutenção do sistema.

## 3. REQUISITOS DE SISTEMA

### 3.1 REQUISITOS FUNCIONAIS DE SOFTWARE

| **RFS01** | |
|---|---|
| **Função** | Cadastro de usuários. |
| **Descrição** | O sistema deve permitir que usuários se cadastrem com e-mail válido e senha + nome e sobrenome para acessar a plataforma. |
| **Entradas** | Primeiro nome, sobrenome, e-mail e senha |
| **Fonte** | Usuário. |
| **Saídas** | “Cadastrado realizado com sucesso!” |
| **Ação** | Registro de usuário no banco de dados. |

| RFS02 | |
|---|---|
| **Função** | Login e Autenticação |
| **Descrição** | O sistema deve permitir que os usuários façam login com suas credenciais, verificando se estão registrados. |
| **Entradas** | E-mail e senha. |
| **Fonte** | Banco de dados. |
| **Saídas** | Login realizado, informação incorreta ou usuário não encontrado + direcionamento para página de cadastro. |
| **Ação** | Verificar registro no banco de dados, caso e-mail corresponda com senha o login deve ser realizado, caso o e-mail exista mas a senha esteja incorreta, o usuário deve ser informado que alguma informação está incorreta e por fim, se o e-mail não for encontrado o usuário deve ser instruído para realizar o cadastro. |

| RFS03 | |
|---|---|
| **Função** | Criação de Eventos |
| **Descrição** | O sistema deve possibilitar que usuários específicos registrem novos eventos com informações de título, imagem, descrição, categoria, data e local. |
| **Entradas** | Título do evento, imagem, descrição, categoria, data e local. |
| **Fonte** | Parceiros. |
| **Saídas** | Criação do card do evento. |
| **Ação** | Criar página do evento e card na plataforma. |

| RFS04 | |
|---|---|
| **Função** | Pesquisa de Eventos |
| **Descrição** | O sistema deve permitir que os usuários pesquisem eventos por nome, categoria ou localização através de um campo de busca com filtros. |
| **Entradas** | Nome do evento, categoria ou localização. |
| **Fonte** | Usuário. |
| **Saídas** | Eventos correspondentes aos filtros utilizados. |

| | |
|---|---|
| **Ação** | Buscar no banco de dados os eventos que ainda não aconteceram que correspondem aos filtros utilizados na busca. |

| **RFS05** | |
|---|---|
| **Função** | Exibição de Eventos |
| **Descrição** | O sistema deve exibir os eventos cadastrados em carrosséis e cards, separados por categorias como "Para Crianças", "Música" ou "Cultura". |
| **Entradas** | Dados do evento. |
| **Fonte** | Banco de dados. |
| **Saídas** | Card com informações do evento. |
| **Ação** | Receber informações dos eventos e separá-los por categorias na interface do usuário. |

| **RFS06** | |
|---|---|
| **Função** | Inscrição em Eventos |
| **Descrição** | O sistema deve permitir que os usuários salvem eventos, registrando seu interesse no banco de dados. |
| **Entradas** | Curtida/botão salvar. |
| **Fonte** | Usuário. |
| **Saídas** | Salvamento do evento na página do usuário para que ele consulte os próximos eventos que tem interesse de participar. |
| **Ação** | Registrar interesse no banco de dados e reunir seus interesses em um carrossel na página do usuário. |

## 3.2 REQUISITOS NÃO FUNCIONAIS DE SOFTWARE

| **RFS01** | |
|---|---|
| **Função** | Desempenho |
| **Descrição** | O site deve carregar as páginas de eventos em até 3 segundos, mesmo em conexões lentas. |
| **Entradas** | Requisições de páginas. |
| **Fonte** | Navegador do usuário. |
| **Saídas** | Páginas carregadas. |
| **Ação** | Otimizar tempo de carregamento. |

| RFS02 | |
|---|---|
| **Função** | Escalabilidade |
| **Descrição** | O sistema deve ser escalável para suportar um grande número de eventos e usuários simultâneos. |
| **Entradas** | Aumento de usuários e eventos. |
| **Fonte** | Acesso simultâneo ao sistema. |
| **Saídas** | Manutenção de performance. |
| **Ação** | Implementar soluções escaláveis. |

| RFS03 | |
|---|---|
| **Função** | Segurança |
| **Descrição** | As informações de login dos usuários devem ser criptografadas e o site deve seguir padrões de segurança (SSL, proteção contra SQL Injection). |
| **Entradas** | Dados do usuário (login, inscrição). |
| **Fonte** | Interações do usuário. |
| **Saídas** | Dados protegidos. |
| **Ação** | Implementar criptografia e padrões de segurança. |

| RFS04 | |
|---|---|
| **Função** | Compatibilidade |
| **Descrição** | O sistema deve ser compatível com todos os principais navegadores e dispositivos móveis. |
| **Entradas** | Navegadores e dispositivos. |
| **Fonte** | Variados (Chrome, Firefox, mobile). |
| **Saídas** | Interface funcional. |
| **Ação** | Testar e ajustar para compatibilidade. |

| RFS05 | |
|---|---|
| **Função** | Responsividade |
| **Descrição** | O layout do site deve ser responsivo, adaptando-se bem a diferentes tamanhos de tela, especialmente em dispositivos móveis. |
| **Entradas** | Tamanhos de tela variados. |
| **Fonte** | Dispositivos móveis e desktops. |
| **Saídas** | Layout adaptável. |
| **Ação** | Implementar design responsivo. |

| RFS06 | |
|---|---|
| **Função** | Usabilidade |
| **Descrição** | O sistema deve ser fácil de usar e intuitivo, com interfaces claras para navegação entre os eventos e ações como cadastro e login. |
| **Entradas** | Ações do usuário. |
| **Fonte** | Interface do sistema. |
| **Saídas** | Navegação intuitiva. |
| **Ação** | Testes de usabilidade e melhorias de interface. |

## 4. CASOS DE USO

### 1. Cadastro de Evento:

**Ator:** Usuário autenticado.

**Descrição:** O usuário logado acessa a página de criação de evento, preenche os campos necessários e confirma o cadastro. O sistema registra o evento e o disponibiliza na página principal.

**Fluxo Alternativo:** Se o usuário não estiver logado, o sistema redireciona para a página de login.

### 2. Busca de Evento:

**Ator:** Usuário autenticado ou visitante.

**Descrição:** O usuário utiliza o campo de busca na página principal para localizar eventos. O sistema exibe os resultados filtrados conforme os parâmetros de busca.

**Fluxo Alternativo:** Se nenhum evento for encontrado, o sistema exibe uma mensagem de “Nenhum evento encontrado”.

### 3. Inscrição em Evento:

**Ator:** Usuário autenticado.

**Descrição:** O usuário seleciona um evento da lista e clica em "Inscrever-se". O sistema registra a inscrição do usuário no evento e confirma a ação.

**Fluxo Alternativo:** Se o usuário não estiver logado, o sistema solicita que faça login antes de prosseguir com a inscrição.

## 5. ARQUITETURA DO SISTEMA

### 1. Camada de Apresentação (Frontend)

**Tecnologias:** HTML, CSS, JavaScript.

**Responsabilidade:** Interface do usuário, onde os usuários podem interagir com o sistema. Exibe formulários de cadastro, login, criação de eventos e exibição de eventos.

### 2. Camada de Lógica de Negócios (Backend)

**Tecnologias:** Node.js.

**Responsabilidade:** Processamento das regras de negócio, validação de dados, autenticação de usuários, manipulação de eventos e gerenciamento de inscrições.

### 3. Camada de Acesso a Dados

**Responsabilidade:** Interação com o banco de dados. Realiza operações de CRUD (Create, Read, Update, Delete) para usuários e eventos.

### 4. Banco de Dados

**Tecnologias:** SQLiteStudio.

**Responsabilidade:** Armazenamento das informações de usuários, eventos e inscrições. Deve ser otimizado para consultas rápidas.

### 5. Camada de Segurança

**Tecnologias:** HTTPS para segurança de comunicação.

**Responsabilidade:** Gerenciar autenticação e autorização, proteger dados sensíveis e evitar vulnerabilidades.

### 6. Camada de API

**Responsabilidade:** Comunicação entre o frontend e o backend. Fornece endpoints para operações como cadastro de usuários, criação de eventos, pesquisa de eventos e inscrição.

### 7. Infraestrutura

**Responsabilidade:** Gerenciar a implantação e a escalabilidade do sistema. Prover um ambiente seguro e de alto desempenho.

### Fluxo Geral

1. **Usuário interage com a interface** (cadastro, login, criação de eventos).
2. **Frontend faz requisições para o backend** através da API.
3. **Backend processa as requisições**, aplicando as regras de negócio.
4. **Backend acessa o banco de dados** para armazenar ou recuperar informações.
5. **Retorna a resposta ao frontend**, que atualiza a interface.

## 6. REFERÊNCIAS BIBLIOGRÁFICAS

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