Documentação do Código: Plataforma de Gerenciamento de Eventos Culturais

Introdução

Esta documentação abrange o código-fonte do backend de uma plataforma de gerenciamento de eventos culturais, escrito em TypeScript usando o framework Express. A plataforma permite que organizadores criem e listem eventos, enquanto participantes podem explorar, pesquisar e filtrar eventos com base em categorias, locais e datas. O código utiliza o Prisma como ORM para interação com o banco de dados.

Estrutura do Projeto

O projeto é dividido em duas classes principais: AdminController e ParticipantController. Cada classe corresponde a um conjunto específico de funcionalidades, sendo AdminController responsável por operações CRUD relacionadas a eventos, categorias e locais, enquanto ParticipantController oferece funcionalidades específicas para os participantes da plataforma.

Dependências

O código utiliza as seguintes dependências:

- express: Framework web para Node.js.
- prismaClient: ORM para interação com o banco de dados.
- typescript: Linguagem de programação.
- Outras dependências listadas no arquivo de configuração package.json.

Métodos Comuns

Ambas as classes compartilham alguns métodos comuns para obtenção de informações gerais.

getAllEvents
Descrição: Obtém todos os eventos cadastrados.
Rota (Admin): GET **/organizer/get-events**
Rota (Participant): GET **/participant/get-events**
Parâmetros: Nenhum.

getAllCategories
Descrição: Obtém todas as categorias cadastradas.
Rota (Admin): GET **/organizer/get-categories**
Rota (Participant): Não aplicável.
Parâmetros: Nenhum.

getAllLocals
Descrição: Obtém todos os locais cadastrados.
Rota (Admin): GET **/organizer/get-locals**
Rota (Participant): Não aplicável.
Parâmetros: Nenhum.

Métodos - AdminController

A classe AdminController fornece operações CRUD relacionadas a eventos, categorias e locais.

Métodos de Eventos

createEvent
Descrição: Cria um novo evento.
Rota: POST **/organizer/create-events**
Parâmetros:

- name: Nome do evento.
- date: Data do evento. (no formato 'YYYY-MM-DDT00:00:00Z')
- categoryId: ID da categoria do evento.
- locationId: ID do local do evento.

updateEvent
Descrição: Atualiza um evento existente.
Rota: PUT **/organizer/update-events/:id**
Parâmetros:

- id: ID do evento a ser atualizado.
- name: Novo nome do evento.
- date: Nova data do evento.
- categoryId: Novo ID da categoria do evento.
- locationId: Novo ID do local do evento.

deleteEvent
Descrição: Exclui um evento.
Rota: DELETE **/organizer/delete-events/:id**
Parâmetros:
id: ID do evento a ser excluído.

Métodos de Categorias

createCategory
Descrição: Cria uma nova categoria.
Rota: POST **/organizer/create-categories**
Parâmetros:
name: Nome da categoria.

updateCategory
Descrição: Atualiza uma categoria existente.
Rota: PUT **/organizer/update-categories/:id**
Parâmetros:

- id: ID da categoria a ser atualizada.
- name: Novo nome da categoria.

deleteCategory
Descrição: Exclui uma categoria.
Rota: DELETE **/organizer/delete-categories/:id**
Parâmetros:
id: ID da categoria a ser excluída.

Métodos de Locais

createLocal
Descrição: Cria um novo local.
Rota: POST **/organizer/create-locals**
Parâmetros:
name: Nome do local.

updateLocal
Descrição: Atualiza um local existente.
Rota: PUT **/organizer/update-locals**
Parâmetros:

- id: ID do local a ser atualizado.
- name: Novo nome do local.

deleteLocal
Descrição: Exclui um local.
Rota: DELETE **/organizer/delete-locals**
Parâmetros:
id: ID do local a ser excluído.

Métodos - ParticipantController

A classe ParticipantController fornece funcionalidades específicas para os participantes da plataforma.

Métodos de Eventos para Participantes

getEventByLocal
Descrição: Obtém eventos com base no local solicitado pelo participante.
Rota: GET : **/participant/get-events/:local**
Parâmetros:
local: ID do local desejado.

getEventByDate
Descrição: Obtém eventos com base na data solicitada pelo participante.
Rota: GET : **/participant/get-events/:date**
Parâmetros:
date: Data desejada (no formato 'YYYY-MM-DDT00:00:00Z').

getEventByCategory
Descrição: Obtém eventos com base na categoria solicitada pelo participante.
Rota: GET :**/participant/get-events/:category**
Parâmetros:
category: ID da categoria desejada.

Tratamento de Erros
Em caso de erro durante a execução de qualquer operação, as rotas respondem com um código de status 500 e um objeto JSON contendo a mensagem de erro correspondente.

Autenticação

Cadastro de Usuários:
Endpoint: /admin/signup ou /participant/signup
Método: POST
Corpo da Requisição: JSON
```
{
   "email": "seuemail@example.com",
   "password": "sua_senha"
}
```
Resposta: O ID do usuário criado.

Login de Usuários:
Endpoint: /admin/login ou /participant/login
Método: POST
Corpo da Requisição: JSON
```
{
   "email": "seuemail@example.com",
   "password": "sua_senha"
}
```
Resposta: Um token JWT que deve ser usado para autenticar solicitações subsequentes.

Acessar Rotas Protegidas:
Para acessar as rotas protegidas, você precisa incluir o token JWT no cabeçalho de autorização de suas solicitações. No Postman, você pode fazer isso na seção “Headers” da sua solicitação. Adicione uma chave chamada “Authorization” e para o valor use “Bearer seu_token”, onde “seu_token” é o token que você recebeu ao fazer login.

Conclusão
Esta documentação fornece uma visão abrangente do código-fonte do backend da plataforma de gerenciamento de eventos culturais. As classes AdminController e ParticipantController oferecem funcionalidades específicas para organizadores e participantes, implementamos também a autenticação para usuários Organizadores e Participantes, oferecendo mais confiabilidade e segurança na plataforma. Respectivamente, proporcionando uma experiência completa e personalizada na exploração e gestão de eventos culturais.