**Match**

O controlador **Match**é responsável por lidar com as operações relacionadas às partidas em uma API.

**Rotas**

As seguintes rotas estão disponíveis neste controlador:

* **GET** **/api/Match/GetAll** - Obtém todas as partidas.
* **GET** **/api/Match/GetById/{id}** - Obtém uma partida pelo ID.
* **POST** **/api/Match/Add** - Adiciona uma nova partida.
* **PUT** **/api/Match/Update** - Atualiza uma partida existente.
* **DELETE** **/api/Match/Delete/{id}** - Exclui uma partida pelo ID.

**Métodos**

**GetAll**

Obtém todas as partidas.

* **Rota:** **GET /api/Match/GetAll**
* **Retorno de sucesso:** Retorna um objeto JSON contendo uma propriedade **matches**, que é uma matriz de objetos contendo detalhes das partidas.
* **Retorno de erro:** Retorna um código de status 404 se não houver partidas encontradas ou um código de status 400 se ocorrer um erro.

**GetById**

Obtém uma partida pelo ID.

* **Rota:** **GET /api/Match/GetById/{id}**
* **Parâmetros:**
  + **id** - O ID da partida a ser obtida.
* **Retorno de sucesso:** Retorna um objeto JSON contendo uma propriedade **match**, que é um objeto contendo detalhes da partida.
* **Retorno de erro:** Retorna um código de status 404 se a partida não for encontrada ou um código de status 400 se ocorrer um erro.

**Add**

Adiciona uma nova partida.

* **Rota:** **POST /api/Match/Add**
* **Corpo da solicitação:** Um objeto JSON contendo os detalhes da partida a ser adicionada.
* **Retorno de sucesso:** Retorna um objeto JSON contendo uma propriedade **message** com uma mensagem de sucesso e uma propriedade **status** com o valor "success".
* **Retorno de erro:** Retorna um código de status 400 se ocorrer um erro de validação nos campos obrigatórios da partida ou um código de status 400 se ocorrer um erro ao adicionar a partida.

**Update**

Atualiza uma partida existente.

* **Rota:** **PUT /api/Match/Update**
* **Corpo da solicitação:** Um objeto JSON contendo os detalhes atualizados da partida.
* **Retorno de sucesso:** Retorna um objeto JSON contendo uma propriedade **message** com uma mensagem de sucesso e uma propriedade **status** com o valor "success".
* **Retorno de erro:** Retorna um código de status 400 se ocorrer um erro de validação nos campos obrigatórios da partida, um código de status 400 se ocorrer um erro ao atualizar a partida ou um código de status 400 se for fornecido um ID de log de partida inválido.

**Delete**

Exclui uma partida pelo ID.

* **Rota:** **DELETE /api/Match/Delete/{id}**
* **Parâmetros:**
  + **id** - O ID da partida a ser excluída.
* **Retorno de sucesso:** Retorna um objeto JSON contendo uma propriedade **message** com uma mensagem de sucesso e uma propriedade **status** com o valor "danger".
* **Retorno de erro:** Retorna um código de status 404 se a partida não

**Model**

**Match**

O modelo **Match** representa uma partida.

Propriedades:

* **MatchId** (int) - O ID da partida.
* **StadiumId** (int) - O ID do estádio onde a partida será realizada.
* **RefereeId** (int) - O ID do árbitro da partida.
* **MasterTeamId** (int) - O ID da equipe mandante.
* **AwayTeamId** (int) - O ID da equipe visitante.
* **LogMatchId** (int?) - O ID do registro de log da partida.
* **Ativo** (bool) - Indica se a partida está ativa.
* **Date** (DateTime) - A data da partida.

**Erros**

Ao retornar um erro, a resposta JSON terá a seguinte estrutura:

* **message** (string) - A mensagem de erro.
* **status** (string) - O status do erro.

É importante fornecer valores adequados para os campos obrigatórios da partida e tratar possíveis erros de validação ou exceções durante as operações.

**LogMatchController**

O controlador **LogMatchController** é responsável por lidar com as operações relacionadas aos registros de log das partidas em uma API.

**Rotas**

As seguintes rotas estão disponíveis neste controlador:

* **GET** **/api/LogMatch/GetAll** - Obtém todos os registros de log das partidas.
* **GET** **/api/LogMatch/GetById/{id}** - Obtém um registro de log da partida pelo ID.
* **POST** **/api/LogMatch/Add** - Adiciona um novo registro de log da partida.
* **PUT** **/api/LogMatch/Update** - Atualiza um registro de log da partida existente.
* **DELETE** **/api/LogMatch/Delete/{id}** - Exclui um registro de log da partida pelo ID.

**Métodos**

**GetAll**

Obtém todos os registros de log das partidas.

* **Rota:** **GET /api/LogMatch/GetAll**
* **Retorno de sucesso:** Retorna um objeto JSON contendo uma propriedade **LogMatches**, que é uma matriz de objetos contendo detalhes dos registros de log das partidas.
* **Retorno de erro:** Retorna um código de status 404 se não houver registros de log das partidas encontrados ou um código de status 400 se ocorrer um erro.

**GetById**

Obtém um registro de log da partida pelo ID.

* **Rota:** **GET /api/LogMatch/GetById/{id}**
* **Parâmetros:**
  + **id** - O ID do registro de log da partida a ser obtido.
* **Retorno de sucesso:** Retorna um objeto JSON contendo uma propriedade **LogMatches**, que é um objeto contendo detalhes do registro de log da partida.
* **Retorno de erro:** Retorna um código de status 404 se o registro de log da partida não for encontrado ou um código de status 400 se ocorrer um erro.

**Add**

Adiciona um novo registro de log da partida.

* **Rota:** **POST /api/LogMatch/Add**
* **Corpo da solicitação:** Um objeto JSON contendo os detalhes do registro de log da partida a ser adicionado.
* **Retorno de sucesso:** Retorna um objeto JSON contendo uma propriedade **message** com uma mensagem de sucesso e uma propriedade **status** com o valor "success".
* **Retorno de erro:** Retorna um código de status 400 se ocorrer um erro de validação nos campos obrigatórios do registro de log da partida ou um código de status 400 se ocorrer um erro ao adicionar o registro.

**Update**

Atualiza um registro de log da partida existente.

* **Rota:** **PUT /api/LogMatch/Update**
* **Corpo da solicitação:** Um objeto JSON contendo os detalhes atualizados do registro de log da partida.
* **Retorno de sucesso:** Retorna um objeto JSON contendo uma propriedade **message** com uma mensagem de sucesso e uma propriedade **status** com o valor "success".
* **Retorno de erro:** Retorna um código de status 400 se ocorrer um erro de validação nos campos obrigatórios do registro de log da partida, um código de status 400 se ocorrer um erro ao atualizar o registro ou um código de status 400 se for fornecido um ID de partida inválido.

**Delete**

Exclui um registro de log da partida pelo ID.

* **Rota:** **DELETE /api/LogMatch/Delete/{id}**
* **Parâmetros:**
  + **id** - O ID do registro de log da partida a ser excluído.
* **Retorno de sucesso:** Não há retorno de dados. O registro de log da partida é excluído com sucesso.
* **Retorno de erro:** Não há retorno de dados. Retorna um código de status 404 se o registro de log da partida não for encontrado ou um código de status 400 se ocorrer um erro ao excluir o registro.

**Modelos**

**LogMatch**

O modelo **LogMatch** representa um registro de log de uma partida.

Propriedades:

* **LogMatchId** (int) - O ID do registro de log da partida.
* **MatchId** (int) - O ID da partida relacionada.
* **WinnerTeamId** (int) - O ID da equipe vencedora da partida.
* **MatchDescription** (string) - A descrição do registro de log da partida.

**Erros**

Ao retornar um erro, a resposta JSON terá a seguinte estrutura:

* **message** (string) - A mensagem de erro.
* **status** (string) - O status do erro.

É importante fornecer valores adequados para os campos obrigatórios do registro de log da partida e tratar possíveis erros de validação ou exceções durante as operações.