CENTRO UNIVERSITÁRIO DE BRASÍLIA

GRADUAÇÃO CIÊNCIA DA COMPUTAÇÃO

Camila Nascimento Cardoso - RA: 21653528

José Mateus Pereira Santos - RA: 21650651

SERENA

SISTEMA GERENCIADOR DE CAMPEONATOS DE TÊNIS

BRASÍLIA

2018

HISTÓRICO DE REVISÃO

Data	Versão	Descrição	Autor
05/09/18	1.0	Desenvolvimento da introdução	Camila e José Mateus
27/09/18	2.0	Modificações da introdução	Camila e José Mateus
31/10/18	3.0	Desenvolvimento da descrição geral	Camila e José Mateus
12/11/18	4.0	Modificações e complemento da descrição geral	Camila e José Mateus
15/11/18	5.0	Desenvolvimento da seção 3	Camila e José Mateus
18/11/18	6.0	Finalização do documento	Camila e José Mateus

SUMÁRIO

1. INTRODUÇÃO

1.1. Propósito

Desde a revolução industrial, um marco muito importante para o início da evolução tecnológica, as necessidades humanas, que antes eram feitas manualmente e exigiam tempo, tornam-se cada vez mais automáticas e rápidas. A tecnologia não só agiliza um processo, mas também contribui na organização, no planejamento e na praticidade. Eventos esportivos, em geral, precisam de uma boa organização e um bom planejamento para evitar falhas que afetem diretamente os competidores, rankings e saldos de pontos.

Para ajudar um grupo de jogadores amadores de tênis, os quais residem em um condomínio localizado em Brasília-DF, viu-se a necessidade de desenvolver um *software* para facilitar o gerenciamento de campeonatos de tênis. Para entender melhor a finalidade, discursamos sobre o contexto do problema.

Atualmente, no condomínio *Park Sul Prime*, localizado em uma região administrativa de Brasília, há um grupo de jogadores amadores de tênis que durante o ano realizam uma série de campeonatos internos. Eles administram todos os eventos relacionados ao esporte em planilhas no *software* Microsoft Excel.

Utilizar planilhas Excel para administrar eventos de campeonatos pode ser muito difícil e pouco confiável. Algumas desvantagens são a dificuldade para manter a integridade das planilhas, a possibilidade de, por um acaso, duplicar as informações e a necessidade de atualização dos dados dependendo do tipo de campeonato.

Outro ponto importante a ser considerado é a contagem de pontos, rankings e saldos de pontos que podem gerar problemas uma vez que algum cálculo errado pode influenciar nos resultados dos campeonatos. É necessário ter uma grande precisão para que nenhum competidor seja injustiçado por um erro que pode ser evitado.

Com isso, foi pensado em desenvolver uma ferramenta que auxilie pessoas no gerenciamento de campeonatos de tênis. Será uma aplicação que possibilitará que o planejamento de campeonatos seja mais rápido e mais confiável, tendo em

vista que será uma aplicação *web* na qual o acesso às informações será fácil e atualizadas a todo minuto. Isso permite uma transparência de informações a todos os envolvidos e soluciona o problema falta de confiabilidade que o Excel gera.

1.2. Convenções do documento

Microsoft Office Excel - Excel
Golang - Go
JavaScript - JS
HyperText Transfer Protocol - HTTP
HyperText Markup Language - HTML
Cascading Style Sheets - CSS
Not Only SQL - NoSQL
MongoDB - Mongo
International Organization Standardization - ISO
International Electrotechnical Comission - IEC

1.3. Público-alvo

É voltado para todos aqueles que necessitam gerar e gerenciar campeonatos de tênis, praticam o esporte e necessitam de uma ferramenta facilitadora para conseguir gerar campeonatos de forma eficiente, rápida e segura.

1.4. Escopo do projeto

Esse projeto está sendo desenvolvido para ser uma ferramenta facilitadora de gerenciamento de campeonatos de tênis. Será entregue um *software* em plataforma *web* desenvolvido por dois protocolos utilizando a linguagem Go e JavaScript.

O sucesso desse projeto acarreta em um sistema que ajudará um grupo de jogadores a criarem campeonatos de tênis, de maneira mais prática e segura, agilizando a organização dos campeonatos efetuados durante o ano e dando mais segurança nos rankings de cada um para descobrir o vencedor. O prazo de entrega está previsto para o fim deste ano de 2018.

O *software* contará com cadastro de jogadores, cadastro de organizadores, gerenciamento de diferentes tipos campeonatos, visualização de ranking para cada campeonato, ranking geral, tabela de competidores, cadastro de equipe, criação e exclusão de campeonatos.

1.5. Referências

[1] PERON, Fernando**. Desenvolvimento de um Software de Gerenciamento de Campeonatos de Futebol.** Florianópolis, 2005.

[2] FERREIRA, João Luiz. **Aplicação Web para Gerenciamento de Campeonatos de Futebol.** Florianópolis, 2017.

[3] POLLI, Vinícius. **FUTCHAMPS: Sistema Gerenciador de Competições Futebolísticas**. Curitiba, 2015.

[4] KUNZ, Marcelo; FOLETTO, Antônio Augusto; DA SILVA, Joel. **MICUIM: Uma Proposta de Sistema de Gerenciamento de Atividades Desportivas**. Rio Grande do Sul, 2014.

[5] DESIRÓ, Fábio. **Sistema de Controle de Jogos**. São Paulo, 2010.

[6] MENA, Gean; COMIN, Eduardo. **Sistema de Gerenciamento de Competições Esportivas Voltado ao Futebol**. Santa Catarina, 2013.

[7] SILVA, Maritza; WEINERT, Wagner. **Software para o Gerenciamento de Atividades Relacionadas à Prática do Karatê**. Paraná.

[8] TANEMBAUM, Andrew S. **Redes de Computadores**. 7ª Edição, Editora Campus, Rio de Janeiro – RJ, 2003.

[9] SILVA. M. S. **JavaScript Guia do Programador**. São Paulo: Novatec, 2010.

[10] FREITAS, Luiz Alexandre; MATOS, Fernando; NOGUEIRA, Paulo Eduardo. **Estudo Experimental sobre Paralelismo na Linguagem Go usando Goroutines**. XIII Encontro Anual de Computação - UFG, 2017.

[11] CARVALHO, Suelen. **Golang a Linguagem do Google**. Universidade de São Paulo, 2015.

[12] BANKER, Kyle. **MongoDB in Action**. Shelter Island: Manning, 2012.

[13] BARASUOL ÉRION, Ricardo. **MongoDB uma base de dados orientada a documentos que utiliza orientação a objetos**. Educacional do Município de Assis - Assis, 2012. 35.

[14] JACYNTHO, Mark Douglas. **Processos para Desenvolvimento de Aplicações Web**. PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO, 2008.
[15] PESSOA, Bruno Carlos; SILVA, Wilson; SANTOS, Isley Tadeu; ALVES, Túlio;
[16] DIAS, Alexandre; LUZ, Francisco. **Banco de Dados MongoDB vs Bando de Dado SQL Server 2008**. Universidade José do Rosário Vellano.
[17]PRESMANN, R. **Engenharia de Software: Uma abordagem Profissional**. 7º edição. Editora Bookman.
[18] MOSIMANN, Clara Pellegrinello. ALVES, José Osmar de Carvalho. FISCH, Silvio. **Controladoria: seu papel na administração de empresas**. Florianópolis: ed. da UFSC, Fundação ESAG, 1993.
[19] KROENKE, David. **Sistemas de informação gerenciais**. São Paulo: Saraiva, 2012.
[20] WAKULICZ, Gilmar Jorge. **Sistemas de informações gerenciais.** Universidade Federal de Santa Maria, Colégio Politécnico, Rede e-Tec Brasil, 2016.
[21] CAMARGO, A. A. B.; KHOURI, L. H. E.; GIAROLA, P. C. **O Uso de Sistemas Colaborativos na Gestão de Projetos: Fatores Relevantes para o Sucesso**. Trabalho de Conclusão de Curso, Fundação Instituto de Administração – FIA, 2005.
[22] ARAÚJO, Luciana; FILHO, Edelvino. **Os Sistemas de Informação como Suporte à Tomada de Decisão Estratégica**. Revista Competitividade e Sustentabilidade – ComSus, Paraná, v. 4, n. 2, p. 66-75, 2017.
[23] *Atributos de Qualidade: Sobre a importância e influência dos atributos de qualidade no projeto da Arquitetura de Software.* Disponível em: <https://archive.cnx.org/contents/67b8bc5e-7e5e-4885-aee4-e57105b00f73@5/atributos-de-qualidade>. Acesso em: 15 nov. 2018.

2. DESCRIÇÃO GERAL

2.1. Perspectiva da solução

O software Serena tem como objetivo ajudar a realizar o planejamento de competições de tênis. Esse sistema visa ser um facilitador tanto para os administradores dos campeonatos como para os jogadores.

A solução permite que o decorrer de cada torneio seja realizado de maneira fácil, objetiva, transparente e segura. O lançamento dessa ferramenta permitirá uma maior organização de eventos de competições, possibilitando a realização de vários campeonatos por ano e assim, quem sabe, popularizando cada vez mais esse esporte.

2.2. Características da solução

A solução possui funcionalidades como cadastro de jogadores, cadastro de equipes e cadastro de organizadores. Para obter uma hierarquia, existe um controle de permissões entre essas três entidades. Os organizadores irão ter acesso total ao sistema e as equipes e os tenistas terão algumas funcionalidades a menos.

O cadastro de torneios e os agendamentos poderão ser feitos por qualquer uma das entidades, mas alteração e exclusão de jogadores e equipes fica por responsabilidade dos organizadores. As outras funcionalidades como visualização de ranking, tabela de jogos, tabela de competidores e monitoramento de jogos ficarão disponíveis para todas as entidades.

2.3. Modelagem de dados e características

O sistema é, basicamente, composto pelas entidades*: Jogador, Equipe, Ranking, Partida e Campeonato*. Para acessar a ferramenta, primeiramente, deve-se criar uma conta utilizando a entidade Jogador. É preciso preencher um formulário que solicita informações pessoais para o cadastro, juntamente com a criação de um login e senha para poder ter acesso ao software.

A entidade Equipe poderia ter a mesma função que a do Jogador, que é a criação de usuário para ter acesso ao sistema, mas aqui apenas relaciona-se um jogador com outro jogador, ambos já criados, para a formação de uma dupla. Ou seja, essa entidade não funciona como uma criadora de usuário. Porém, ela também possui a permissão de criar Campeonatos.

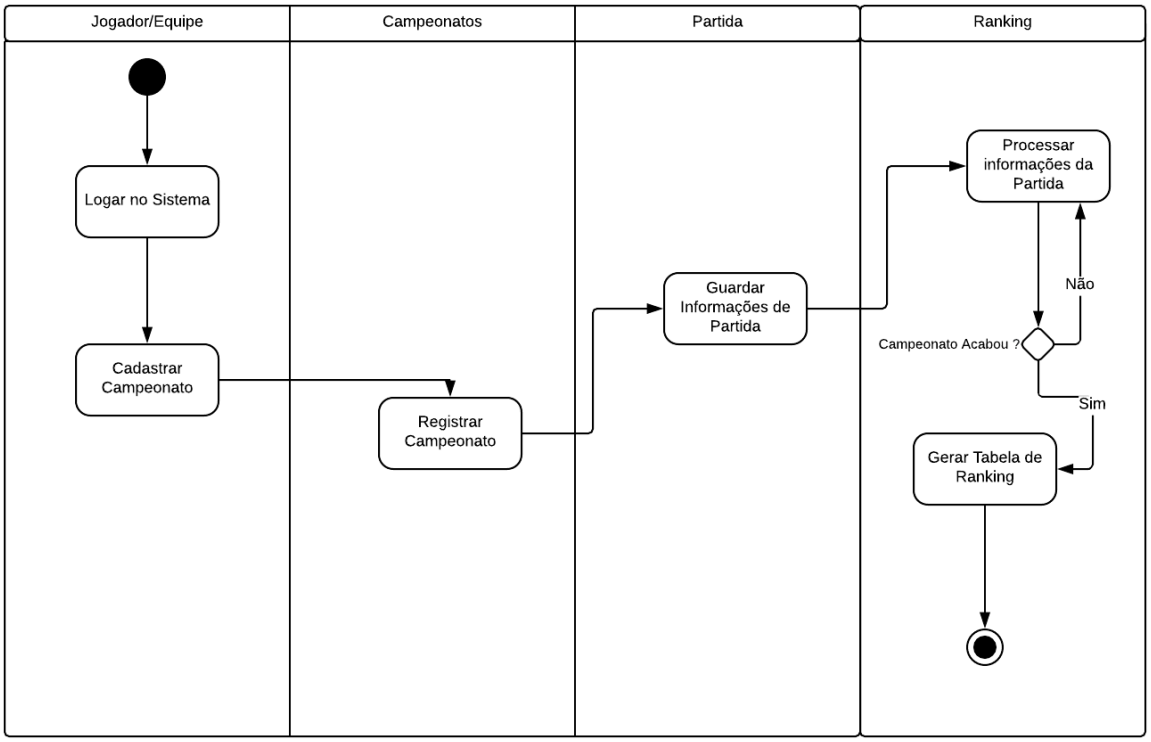
Se a criação de uma conta for efetuada com sucesso, o usuário poderá ter acesso às demais funcionalidades do sistema que é composto pelas entidades Campeonato, Ranking e Partida. A entidade Campeonato é gerada a partir dos jogadores participantes, data de início, organizador e nome. Para criar basta que o usuário logado preencha essas informações, que o sistema guardará a informação para que ela seja recuperada e atualizada até o término do campeonato gerado.

Como cada campeonato de tênis é composto por partidas, a entidade Partida foi criada para auxiliar na contagem de pontos de cada campeonato pedindo informações como número de *points, sets e games*. Para cada partida que for realizada no campeonato, o sistema pedirá que o usuário preencha as informações da partida para cada jogador envolvido.

Para obter o vencedor do campeonato a entidade Ranking se responsabiliza a contabilizar os pontos de cada Partida e processar as informações retornando uma

tabela que tem como informação o nome do jogador, quantidade de pontos e sua colocação.

Com isso, o sistema é feito para facilitar o gerenciamento de campeonatos de tênis para os amantes e praticantes desse esporte.



2.4. Ambiente operacional (Protocolos e Tecnologias)

O protocolo utilizado para desenvolver esse sistema foi o *HyperText Transfer Protocol* (HTTP). Ele tem como característica permitir a transferência de documentos hipertextos. Segundo (KUROSE, 2006), o HTTP é dividido entre cliente e servidor, eles conversam entre si por meio de mensagens HTTP, que, por sua vez, tem como função definir a estrutura das mensagens e como elas irão ser trocadas ou transferidas entre cliente e servidor.

O HTTP possui um conjunto de métodos diferentes para as solicitações de uma página Web. Segundo (TANEBAUM, 2003) o método GET solicita ao servidor que envie a página. Já o HEAD, solicita apenas o cabeçalho da mensagem, sem a página propriamente dita. O método PUT é o inverso do GET: em vez de ler, ele

grava a página. Esse método possibilita a criação de um conjunto de páginas da Web em um servidor remoto. O POST é semelhante ao PUT, porém em vez de substituir os dados existentes, os novos dados são "anexados" a ele, em um sentido mais genérico. Existem outros métodos, que não serão explicados aqui porque não foram muito utilizados, eles são: DELETE, TRACE, CONNECT e OPTIONS.

Como utilizamos HTTP, é imprescindível falar de *HyperText Markup Language* (HTML), a linguagem padrão de programação de páginas web. O HTML é feito e interpretado por um browser que em HTTP é visto como cliente.

A construção das nossas páginas foram feitas em HTML utilizando *Cascading Style Sheets* (CSS), *JavaScript* (JS) e o *framework* Bootstrap. O CSS foi criado para formatar a informação entregue pelo HTML. A maioria das vezes a formatação é para melhorar questões visuais, porém isso não quer dizer que não ocorra para outros fins também. Com o CSS é possível formatar características como margem, fontes, tamanhos, posições, cores, estruturas, etc.

O JS possui a finalidade de trazer mais interatividade entre as páginas web. Segundo (SILVA, 2010) o JS faz parte de uma camada de comportamento do HTML que pode definir ou alterar de forma dinâmica os documentos e também controlar o comportamento do navegador de diversos aspectos como criar janelas pop-up e apresentar mensagens ao usuário.

O Bootstrap é um framework front-end para desenvolvimento web. Ele tem sido um dos frameworks mais utilizados hoje em dia por ter padrões que seguem princípios de usabilidade e tendências para design de interfaces. Isso quer dizer que os sistemas webs ficam com um aspecto visual mais bonito, o que é ótimo em questão de satisfação do usuário. O Bootstrap foi escolhido para ser utilizado nesta aplicação por conta com uma série de classes em CSS prontas, plugins em JavaScript (jQuery), uma biblioteca de componentes, reuso de códigos e muito mais.

Juntando o HTML, JS e CSS estamos falando sobre o front-end da nossa aplicação. Para desenvolver o backend nós utilizamos a linguagem Go, criada pela Google em 2007, desenvolvida para solucionar alguns problemas enfrentados em relação à infraestrutura. Então, os engenheiros da Google desenvolveram essa linguagem para ser utilizada para e por pessoas que escrevem, leem, debugam e mantém grandes sistemas de software. Go tem como característica ser uma

linguagem de programação compilada, concorrente, forte e estaticamente tipada [CARVALHO, 2015].

As vantagens de usar Go nesta aplicação são muitas. Em termos de uso de memória é uma linguagem “leve”. Como utiliza a concorrência, Go implementa as goroutines caso haja um grande número de processos simultâneos que, possivelmente, irão sobrecarregar o backend. Possui uma enorme biblioteca que possuem ferramentas para comunicação em redes, como em servidor HTTP implementado no nosso projeto.

Para finalizar as tecnologias do software, discursamos sobre o MongoDB, ferramenta utilizada para modelagem do banco de dados. O Mongo é um banco de dados que trabalha com Not Only SQL (NoSQL), termo utilizado para definir uma classe de banco de dados que não seguem o tradicional modelo normalizado dos bancos de dados relacionais.

Segundo (BANKER, 2012) o Mongo possui um modelo de dados construído para suportar grande volume de leitura e escrita, ser facilmente escalável e com tolerância a falhas. Por isso, é indicado para sistemas e aplicações desenvolvidas e que utilizam a infraestrutura da internet.

O Mongo é utilizado na nossa ferramenta por possuir uma grande flexibilidade. Nos proporciona um modelo rico que mapeia os tipos de linguagem de programação nativa por armazenar em formato JSON.

2.5. Restrições de projeto e implementação

O desenvolvimento do nosso projeto foi focado para dar suporte a pequenos grupos de jogadores de tênis, portanto não está preparado, a princípio, para grandes demandas de “solicitações”.

Temos a funcionalidade de cadastro de campeonatos, onde o organizador do campeonato seleciona a data de início e os jogadores que participarão. A partir desses dados o sistema, automaticamente, monta o chaveamento dos jogos. Não tem data fixa para cada jogo, o sistema deixa em aberto o campeonato até que chegue na etapa final e ele possa saber o vencedor e, assim, finalizar o torneio.

O esporte tênis possui campeonatos oficiais e o nosso sistema não os reproduzem por não ser voltado a torneios de grande porte. E, por ser assim, nosso sistema de rankeamento também ocorre de uma maneira diferente.

Outra característica restrita do sistema, nesse primeiro momento, é no cadastro de jogadores. Ao realizar o cadastro no sistema para ter acesso é necessário preencher os campos: Nome Completo, CPF, Telefone, Email, Senha, Data de Nascimento, Categoria, Grupo e Classe. As categorias são limitadas de A até E. Os grupos são limitados de 1 a 5. As classes são limitadas de primeira até quinta. Pra essa nossa primeira versão deixaremos assim, porém, se for necessário, adicionaremos a opção de aumentar alguns desses atributos ou até mesmo personalizá-los.

Como está sendo desenvolvido em Go, o software estará disponível para uso no endereço http:/localhost:8080, determinado durante o desenvolvimento. Não será necessário utilizar Apache, Tomcat ou ISS. Isso só é possível porque estamos utilizando o próprio servidor HTTP do Go. Futuramente, pretende-se disponibilizar a aplicação em algum domínio para não ficar restrito a uma série de comandos para conseguir acesso a ferramenta.

2.6. Documentação para usuários

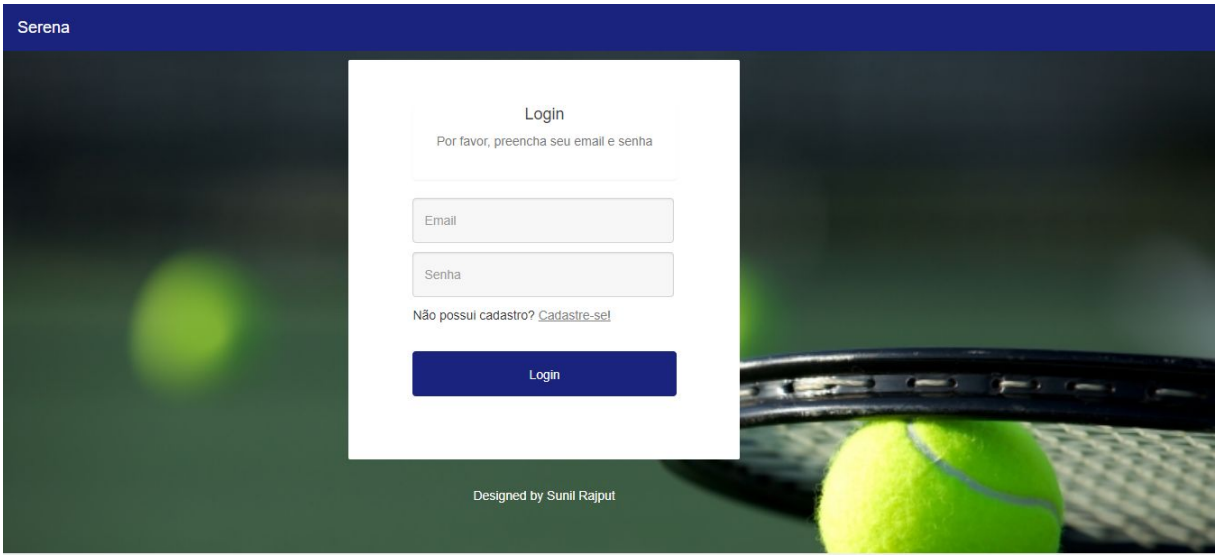


Figura 1: Tela de Login

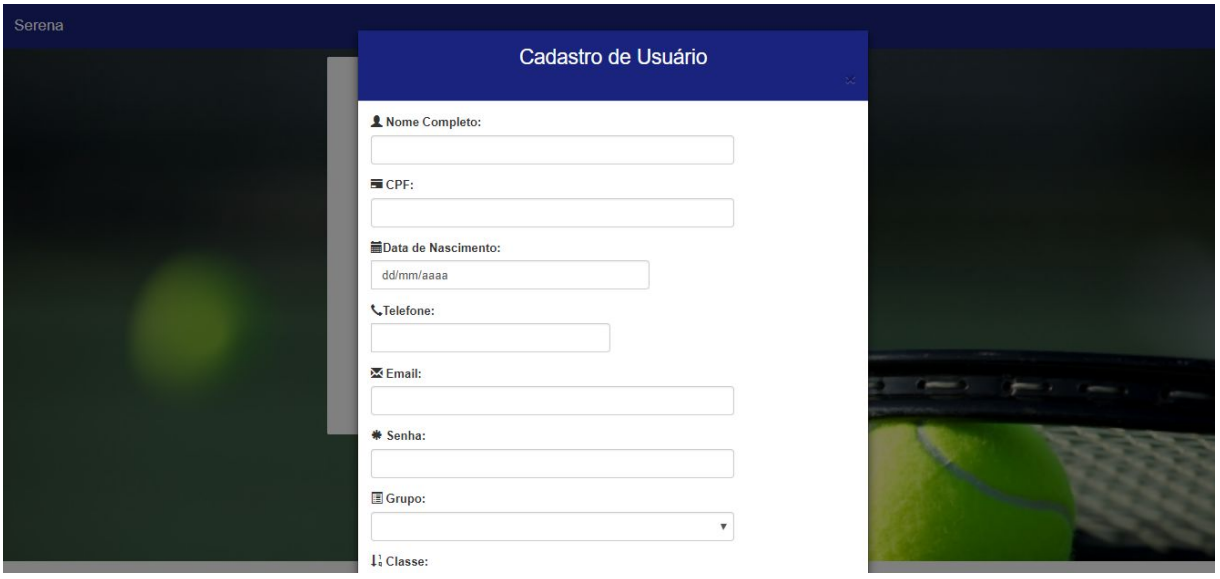


Figura 2: Tela de Registro de Usuário na Página Inicial

Figura 3: Tela de Dashboard

Figura 4: Tela de Jogadores

Figura 5: Tela de Cadastro de Jogador

Figura 6: Tela de Equipes

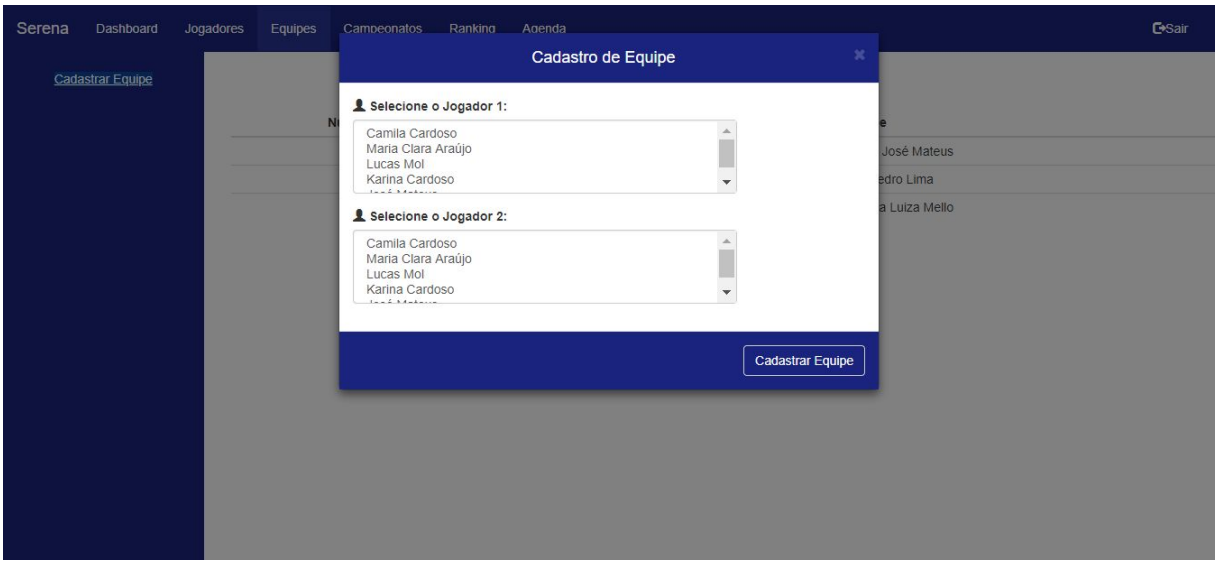


Figura 7: Tela de Cadastro de Equipes

Figura 8: Tela de Campeonatos

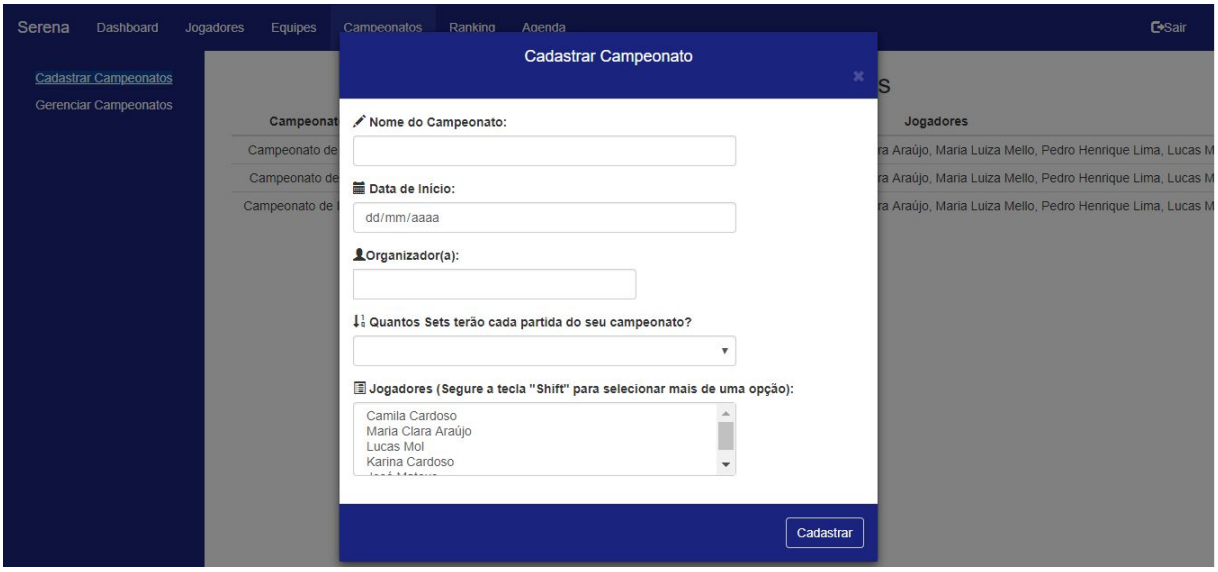


Figura 9: Tela de Cadastro de Campeonatos

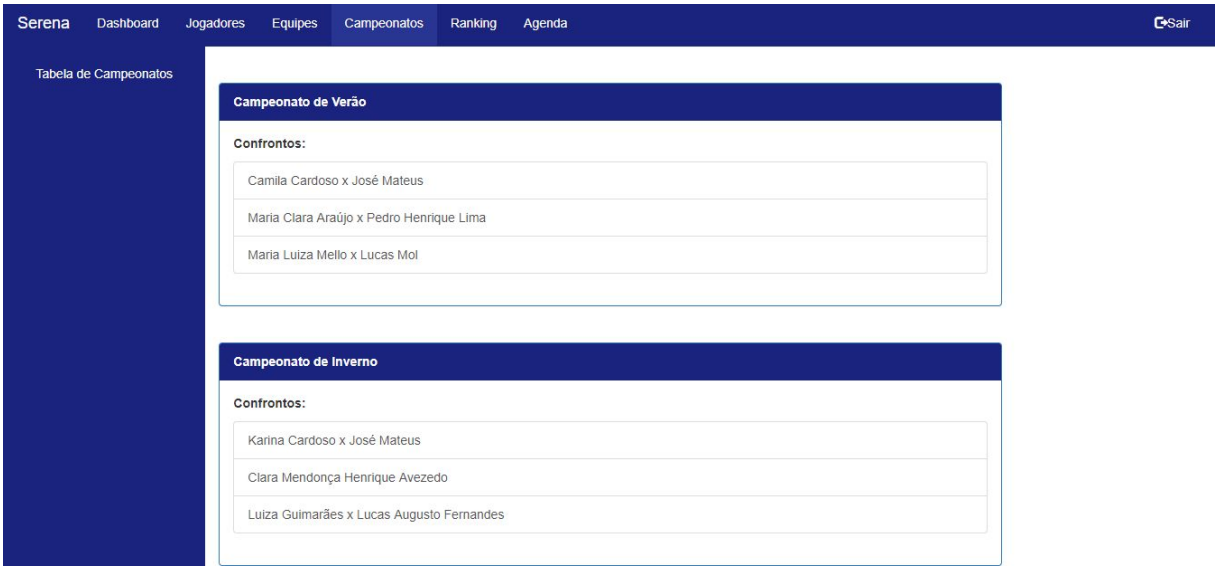


Figura 10: Tela de Gerenciamento de Campeonatos

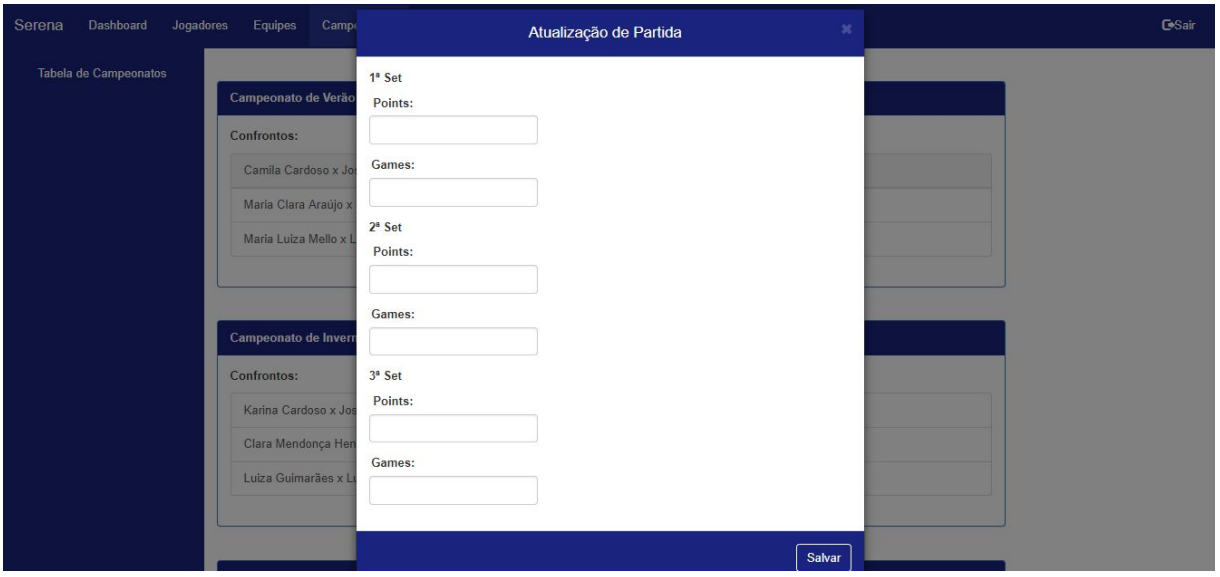


Figura 11: Tela de Gerenciamento de Campeonatos - Atualização de Partida

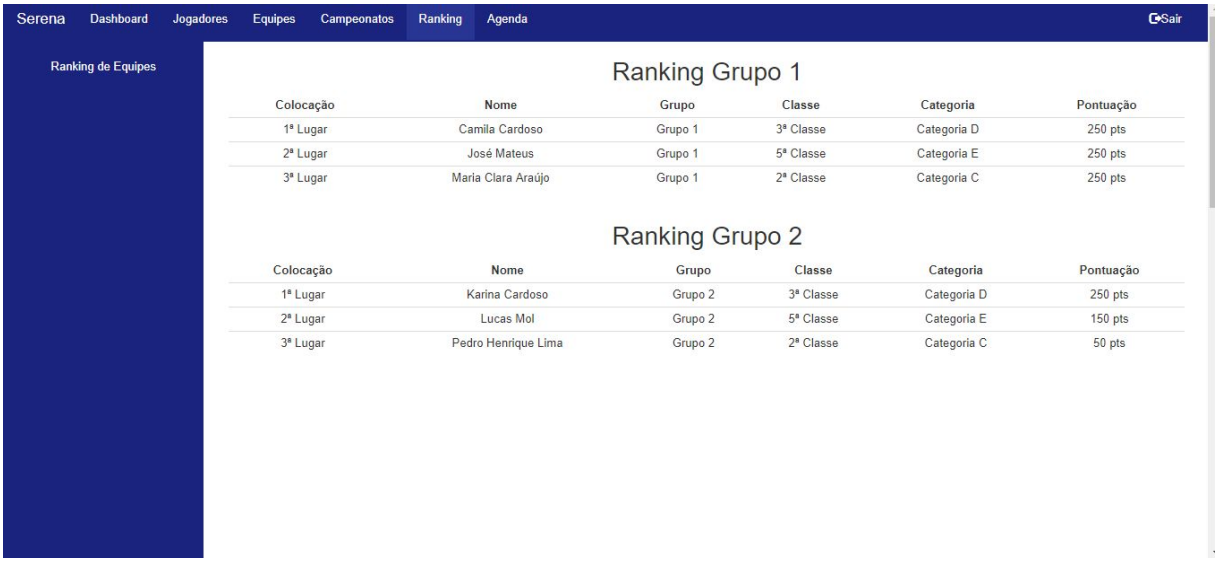


Figura 12: Tela de Ranking de Jogadores

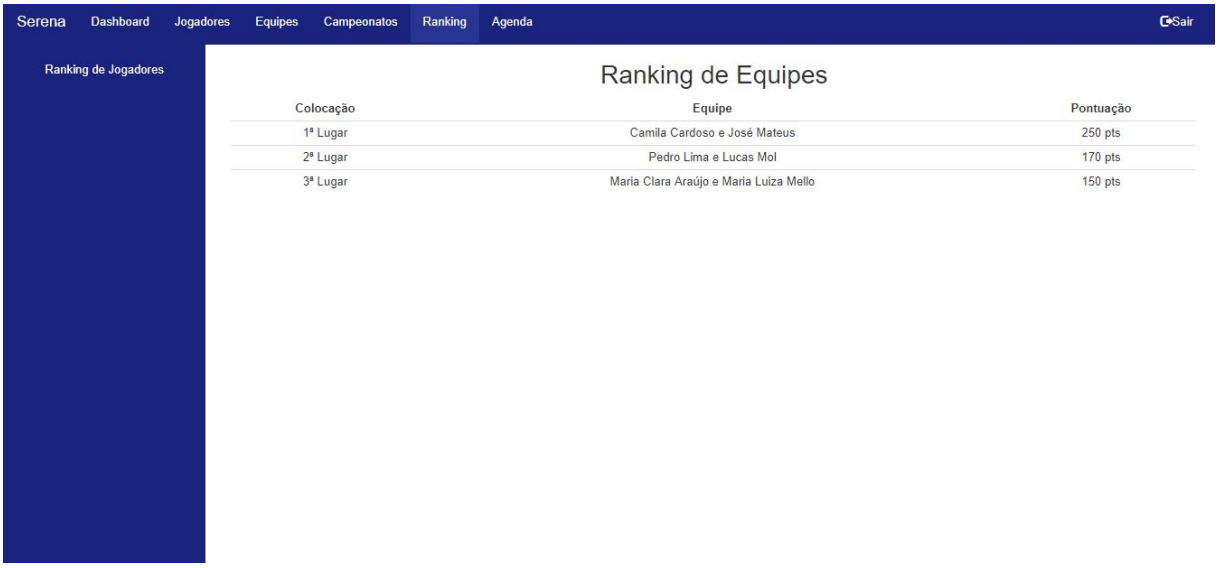
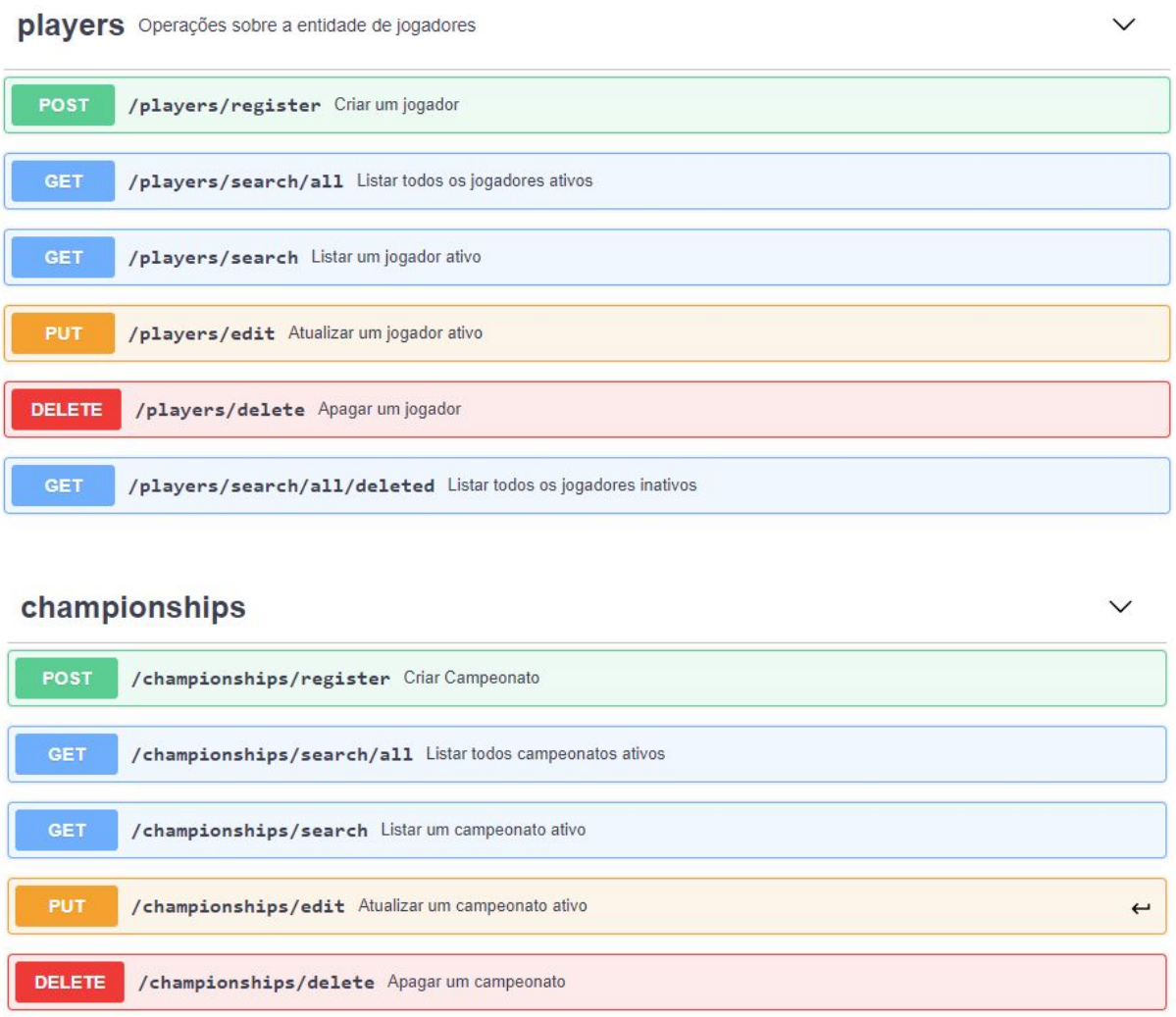
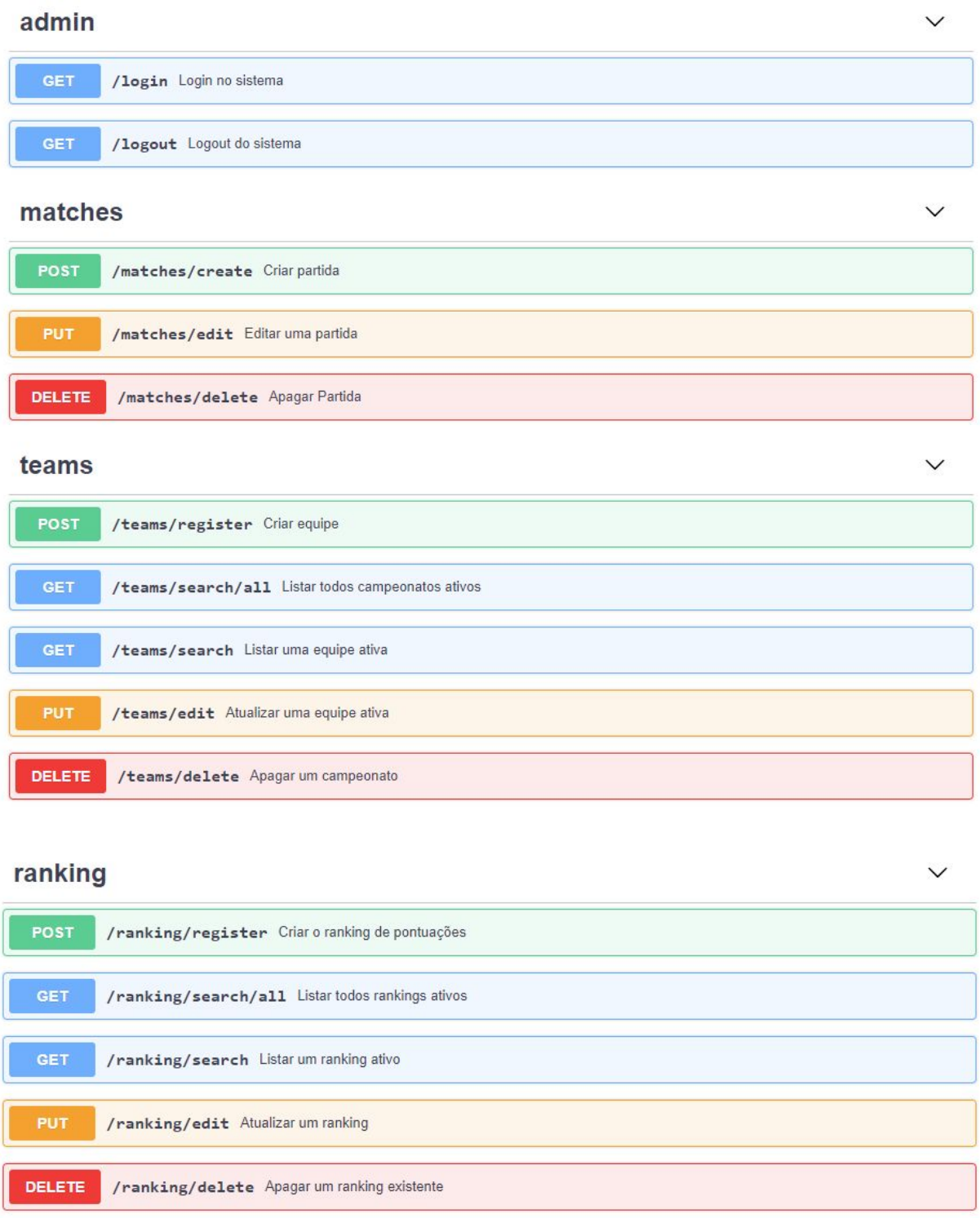


Figura 13: Tela de Ranking de Equipes

Figura 14: Tela de Agenda

2.6.1. Documentação da API





Caso você queira ver a documentação completa, vá até o tópico 5 os anexos estão lá a sua disposição.

2.7. Hipóteses e dependências

O software Serena foi desenvolvido para facilitar o gerenciamento de campeonatos de tênis. A nossa primeira página possui um formulário com email e senha para ser preenchido. Você só terá acesso ao sistema se estiver cadastrado. Caso não esteja, no formulário tem um link “Cadastre-se” que ao clicar abre um modal com o formulário de cadastro para ser para ser preenchido. Se o cadastro for efetuado com sucesso, o modal se fecha e você digita o email e a senha que preencheu para acessar o sistema.

A segunda tela é o dashboard. Como já dito em cima, você só tem acesso a essa e as outras páginas se tiver cadastro no sistema. O dashboard possui uma tabela com a visão geral do software que nela contém informações dos campeonatos, jogadores participantes, data de início, data de término, ganhador(a).

Nesta aplicação é possível cadastrar jogadores clicando na aba “Jogadores” e depois no link “Cadastrar Jogador”. Nesta aba é possível ver uma tabela que possui todos os jogadores cadastrados no sistema indicando seu grupo, sua classe e sua categoria.

A aba de Equipes possui a funcionalidade de formar equipes clicando no link “Cadastrar Equipes” e, também, é possível ver em uma tabela todas as equipes já cadastradas. Para cadastrar uma equipe, necessariamente, ambos os jogadores precisam estar já cadastrados no sistema, porque é feito a partir de uma busca de todos os jogadores relacionando-os.

A parte de campeonatos está localizada na aba “Campeonatos”, lá é possível visualizar todos os campeonatos que estão cadastrados e não foram finalizados. Existe o link “Cadastrar Campeonatos” e você pode criar um a qualquer momento. No link “Gerenciar Campeonatos” abre-se uma página com os campeonatos que ainda estão em andamentos e os próximos confrontos entre jogadores ou equipes. Nesta seção é possível atualizar informações em relações a cada partida, como, por exemplo, números de points e games de cada set.

A agenda ela é formada por uma tabela onde possui os jogos agendados referentes a cada mês. Se não existir jogo nenhum em algum determinado mês logo, não aparecerá nenhuma tabela referente aquele mês. A aba de ranking possui

o rankeamento de cada grupo e o de equipes. Por último, o dashboard possui uma tabela com a visão geral do sistema que nela contém informações dos campeonatos, jogadores participantes, data de início, data de término, ganhador(a).

3. CARACTERÍSTICAS DO SISTEMA

3.1. Características do sistema 1

Segundo (JACYNTHO, 2008) o desenvolvimento de aplicações web é realizado por equipes multidisciplinares, com diferentes habilidades, em prazos curtíssimos ditados pela voraz concorrência e em um contexto extremamente volúvel, marcado por incertezas. Para completar, sistemas web são tecnologicamente muito abstrusos, reunindo padrões, protocolos e tecnologias diversas na definição de uma arquitetura que encapsule em um front-end amigável um back-end que pode ser por demasiado complexo e heterogêneo.

A aplicação web é orientada a conteúdo. Entende-se por conteúdo dados estruturados (banco de dados, por exemplo) e não estruturados (arquivos textos, vídeos, etc.). Precisa ter uma interface limpa, intuitiva e autoexplicativa para que o usuário não o abandone por não conseguir utilizar. Jacyntho declara no seu estudo a importância para a diferença entre aplicação web e aplicação na web.

> É importante ressaltar a diferença entre aplicação na web e aplicação web. Aplicação na web é qualquer tipo de aplicação que utiliza a web como ambiente de execução. Um simples repositório de arquivos ou uma aplicação com estilo tradicional desktop, composta apenas por buscas e formulários, são exemplos de aplicação na web. Já aplicações web são aquelas que, necessariamente, exploram o paradigma hipermídia. Em outras palavras, no contexto deste trabalho, somente são consideradas aplicações web aquelas que possuem uma estrutura navegacional bem definida, fazendo jus ao elemento fundamental da web que é a noção de hiperlink. O grande desafio das aplicações web modernas é integrar, elegantemente, dois paradigmas capitais: hipermídia e transação. (Jacyntho, 2008).

Nosso sistema encaixa em uma aplicação web porque reúne características citadas acima. O protocolo que utilizamos, como já dito, foi o HTTP. Criamos um back-end na linguagem de programação Go e o nosso front-end foi criado a partir de

do framework Bootstrap e as linguagens CSS, JS e HTML para dar uma interface mais bonita e mais instantânea para o usuário. Nosso conteúdo compreende dados estruturados com o uso do MongoDB.

3.2. Características do sistema 2

Segundo (Wakulicz, 2016) sistema é um grupo de componentes que estão inter-relacionados e que visam uma meta comum a partir do recebimento de informações produzindo resultados em um processo organizado de transformação. Possui três componentes ou funções básicas em interação:

- ➔ Inputs – envolve a captação e reunião de elementos que ingressam no sistema para serem processados (dados, instruções).
- ➔ Processamento – envolve processos de transformação que convertem insumos (entradas) em produto (programas, equipamentos).
- ➔ Outputs – envolve a transferência de elementos produzidos por um processo de transformação até seu destino final (relatórios, gráficos, cálculos).

A partir dessa definição é possível entender o conceito de Sistema de Informação (SI). Um SI é uma rede de informações cujos fluxos alimentam o processo de tomada de decisões, não apenas da empresa como um todo, mas, também, de cada área de responsabilidade [MOSIMANN; ALVES; FISH, 1993]. Kroenke (2012) diz que a estrutura de um sistema de informação é composta por cinco componentes: hardware, software, dados, redes e pessoas.

Ficou bem claro que o nosso software tem característica de ser um Sistema de Informação. Para desenvolver o sistema, foi necessário coletar os requisitos, entender sua definição e quais etapas estariam envolvidas no desenvolvimento da ferramenta. Para chegarmos no resultado final, foram tomadas uma série de decisões baseadas nas informações que usuário nos contava, como o motivo da necessidade do software, requisitos funcionais e não-funcionais do sistema, regras de negócio, etc.

Porém, um SI possui vários tipos como, por exemplo, sistemas de processamento de transações, sistemas de controle de processos, sistemas colaborativos, sistemas de informações gerenciais, sistemas de informação

executivas. Entrando mais a fundo a definição do software, podemos afirmar que ele encaixa melhor na definição de Sistemas Colaborativos.

Os Sistemas Colaborativos são ferramentas de software utilizadas em redes de computadores para facilitar a execução de trabalhos em grupos. Essas ferramentas devem ser especializadas o bastante, a fim de oferecer aos seus usuários formas de interação, facilitando o controle, a coordenação, a colaboração e a comunicação entre as partes envolvidas que compõe o grupo, tanto no mesmo local, como em locais geograficamente diferentes e que as formas de interação aconteçam tanto ao mesmo tempo ou em tempos diferentes. Percebe-se com isso que o objetivo dos Sistemas Colaborativos é diminuir as barreiras impostas pelo espaço físico e o tempo [Camargo, Khouri, & Giarola, 2005].

Nosso software foi desenvolvido, justamente, para facilitar o gerenciamento de campeonatos de tênis de um pequeno grupo de tenistas de um condomínio de Brasília - DF. Ele oferece informações como, jogadores cadastrados, campeonatos cadastrados, dias de jogos, equipes formadas e tabela de ranking. Ou seja, é possível obter as características de um sistema colaborativo porque nossa ferramenta facilita o controle, a comunicação, a colaboração entre os usuários do sistema. Então, é possível concluir que o nosso software tem característica de ser um sistema de informação colaborativo.

4. OUTROS REQUISITOS NÃO FUNCIONAIS

4.1. Necessidades de desempenho

O requisito de desempenho afeta a usabilidade de um sistema. Então, se o sistema não corresponder com a necessidade do usuário, o software ficará ruim de ser utilizado. O nosso sistema é web e possui um servidor HTTP desenvolvido em Go. Escolhemos essa linguagem para desenvolver porque ela possui vantagens por aguentar uma grande quantidade de requisições sem afetar o sistema.

Em relação ao tempo de resposta, o software responde rápido. As informações e o acesso são em tempos reais. O processamento das informações

também. Um fator que pode influenciar no nosso sistema é se a conexão de internet estiver lenta, devido ao fato de ser uma aplicação web.

4.2. Necessidades de proteção

Os requisitos de proteção do nosso software não foram implementados, não foi possível atender esses aspectos devido ao curto prazo de desenvolvimento do projeto.

4.3. Necessidades de segurança

Os requisitos de segurança tem como objetivo manter a integridade do sistema, por isso o software apenas dá acesso a aqueles que tenham sido autenticadas por um componente de controle acesso, ou seja, possuem um login e senha cadastrados. O sistema deverá garantir o sigilo das informações, bem como o controle de acesso a perfis de usuários.

4.4. Atributos de qualidade de software

Os atributos de qualidade do software ou de seu ciclo de desenvolvimento são características, capacidades ou restrições de uma função específica ou de um conjunto de funções do software.

A *International Organization Standardization* (ISO) e a *International Electrotechnical Comission* (IEC) se uniram para provar normas internacionais para produtos, processos, serviços e procedimentos. A norma que nos interessa entender é a ISO 9000 que atua na gestão da qualidade.

A ISO/IEC 9126 possui um modelo de qualidade de software que propõe que os atributos de qualidade são divididos em seis características: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.

A funcionalidade é a capacidade do software de realizar as funções que foram especificadas. Em questão de adequação nosso software possui as funções que atendem o objetivo dos usuários. Um dos grandes problemas que nos levaram a desenvolver essa solução foi a dificuldade do cliente em manter o gerenciamento dos campeonatos em planilhas excel. Nosso sistema faz tudo o que já era feito em

planilhas excel, porém com muito mais segurança, mantendo a integridade dos dados, com facilidade de manuseio, com praticidade e com muita mais rapidez.

A usabilidade é a medida da facilidade de o usuário executar alguma funcionalidade do sistema. O nosso software foi desenvolvido utilizando um framework de front-end, justamente, para obter uma interface mais amigável e intuitiva para o usuário. A compreensão do sistema está fácil de aprender porque o sistema foi dividido em entidades pensando em facilitar o manuseio e o entendimento, ou seja, temos a entidade Jogador e todas as funções relacionadas estarão nesse ambiente, como, por exemplo, cadastro de jogadores, edição e deleção, visualização de todos os jogadores cadastrados.

A eficiência é a qualidade relacionada ao uso de recursos do sistema quando esse provê funcionalidade e é também a com que os desenvolvedores mais se preocupam. O software foi desenvolvido na linguagem Go utilizando MongoDB, isso quer dizer que é o servidor aguenta um grande número de requisições. Já o Mongo é conhecido por ser um banco de dados escalável e com tolerância a falhas. Por isso, é indicado para sistemas e aplicações desenvolvidas e que utilizam a infraestrutura da internet. Essas são características que ajudam muito na eficiência do software.

A portabilidade é a medida de adaptações necessárias para que o sistema tenha seus requisitos ou ambientes de execução modificados, podendo ser o ambiente de software, de hardware ou organizacional. O software é uma aplicação web, ou seja, ele funciona em qualquer sistema operacional tanto que tenha internet e um browser. Ele é compatível com Mozilla, Chrome, Microsoft Edge, etc. Porém, não foi desenvolvido para aplicativo mobile, se fosse existir uma portabilidade para esse tipo de plataforma teria que existir alterações que dariam trabalho.

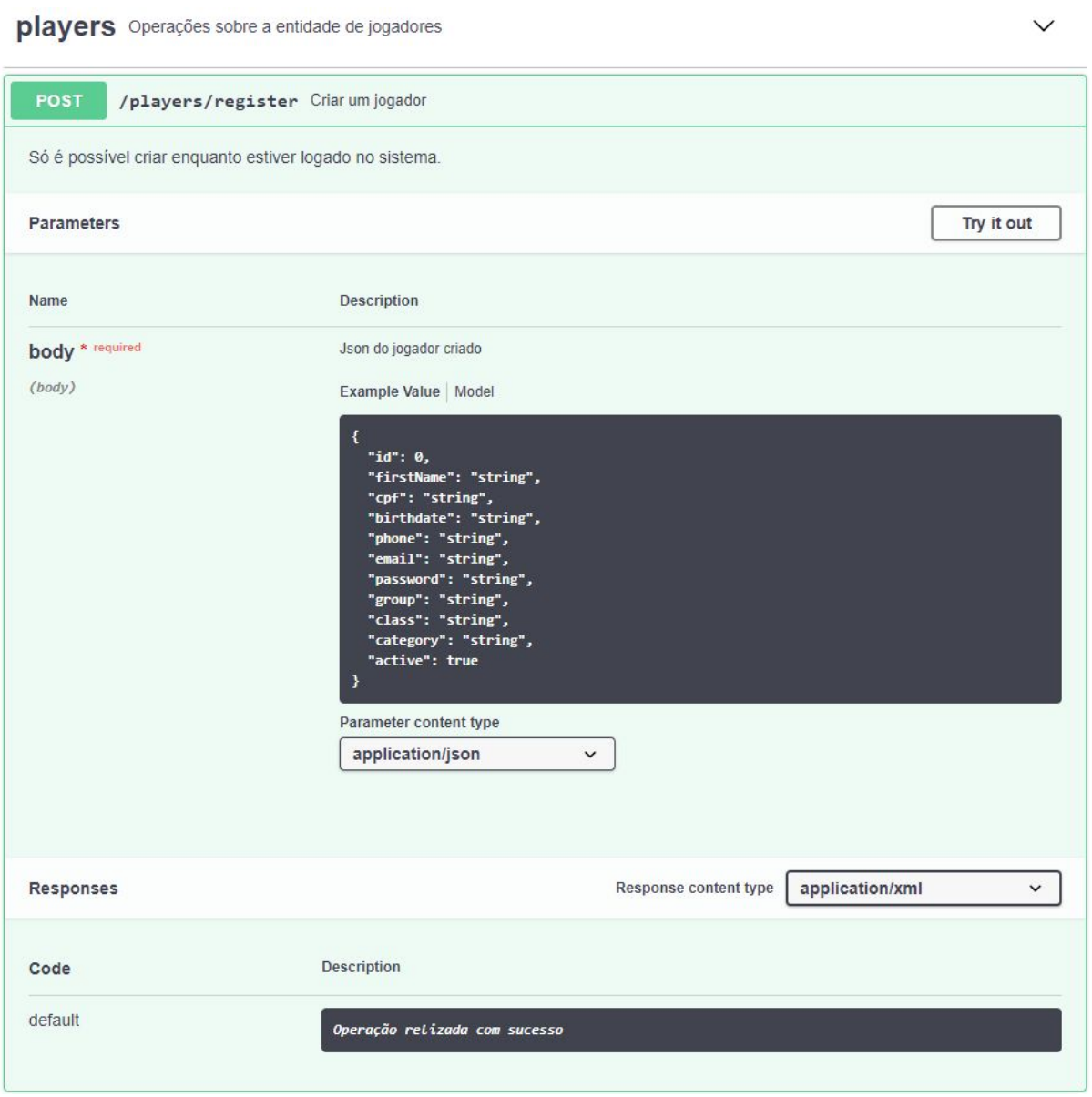
A manutenibilidade é a capacidade de o software ser modificado em seu processo de evolução. Pensando na questão de testabilidade, que envolve a manutenibilidade, o software possui os requisitos bem definidos e por isso foram feitos vários testes de integração para testar todos os componentes. Isso indica que é fácil encontrar uma falha quando ela existe e, logo, é fácil corrigi-la.

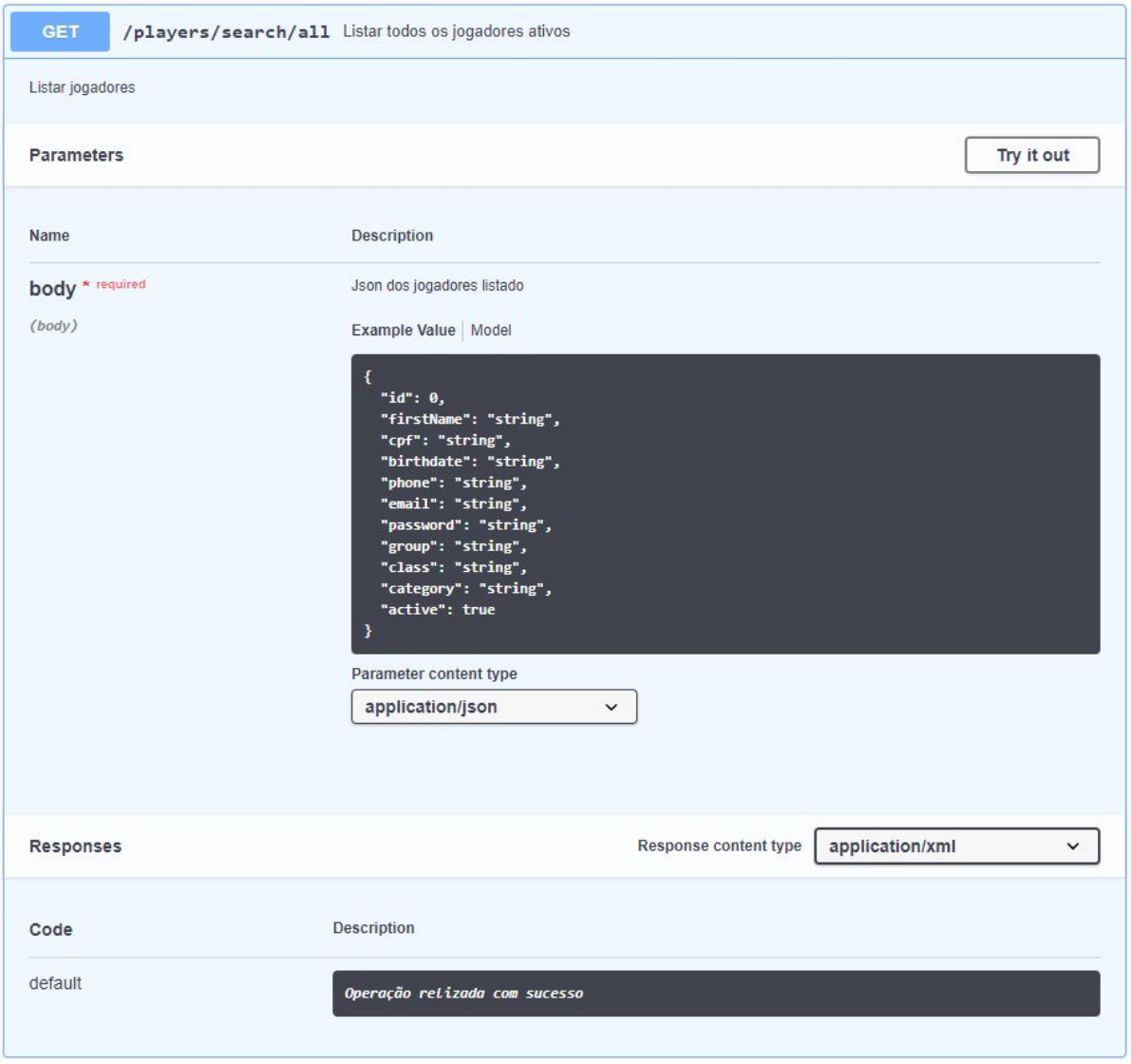
A confiabilidade é a capacidade do sistema de manter algum nível de desempenho quando funcionando sob circunstâncias determinadas. Por ter uma pilha de testes de integração feitos, o sistema possui uma grande maturidade.

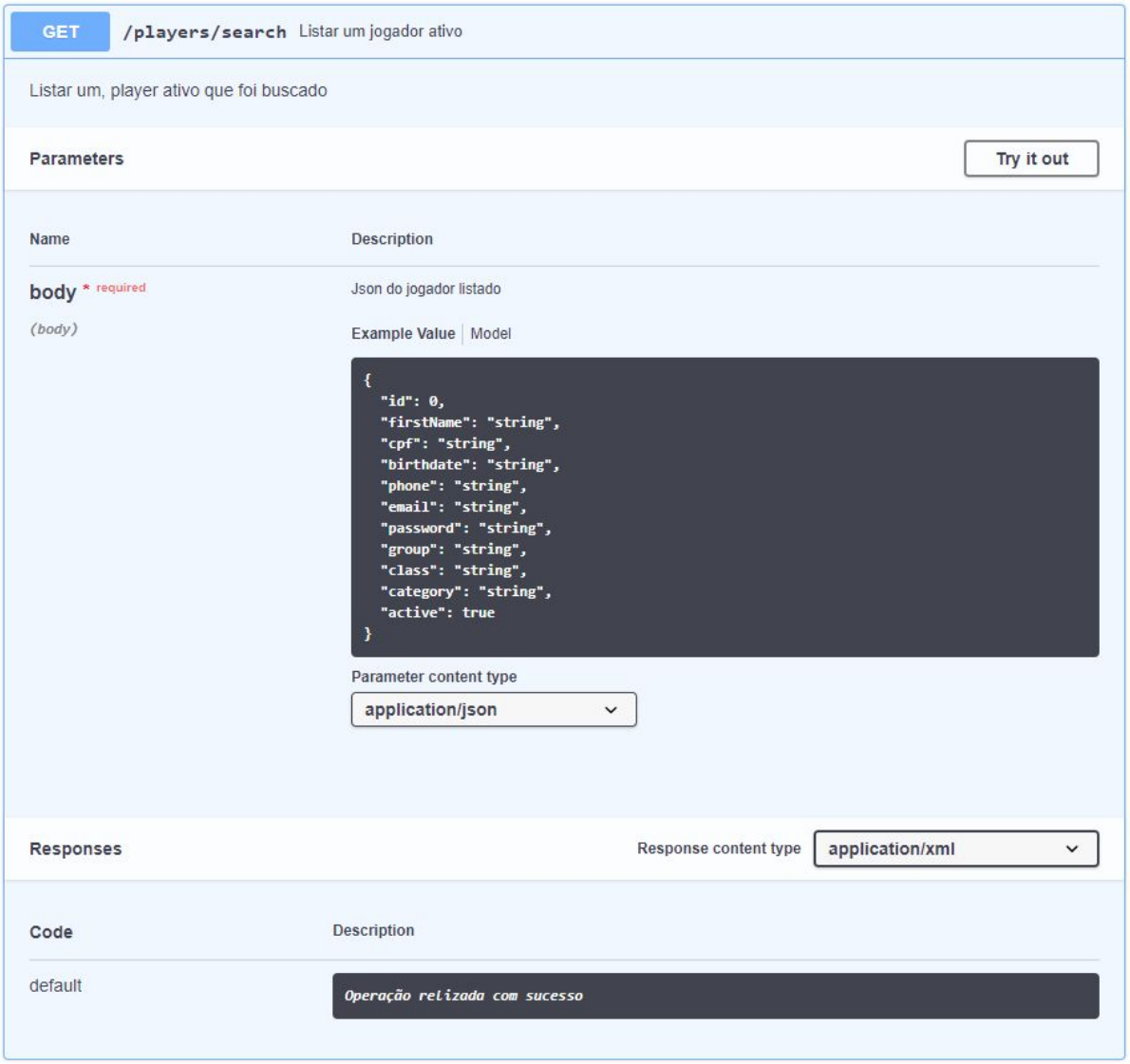
5. OUTROS REQUISITOS

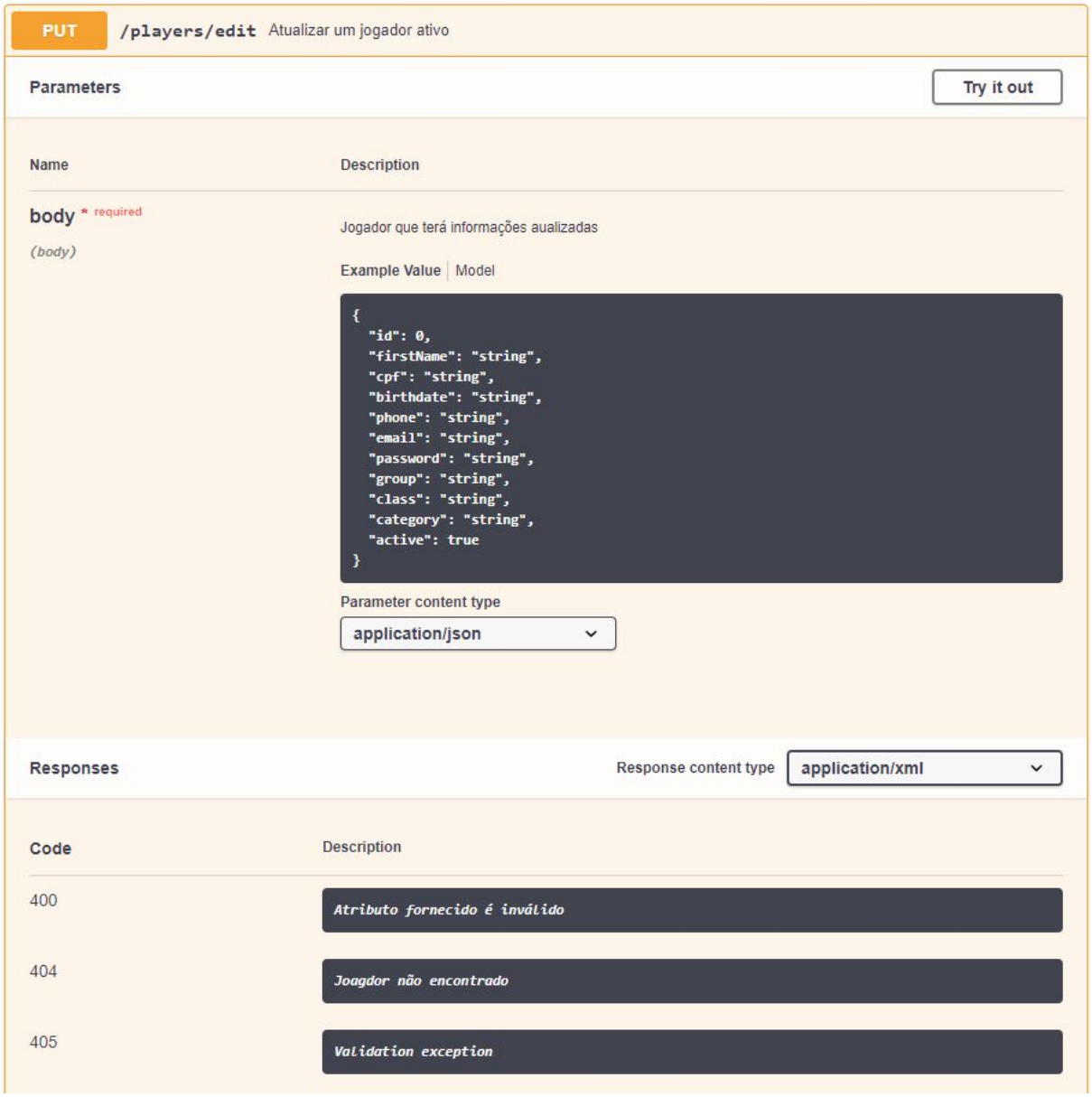
5.1. Documentação do Swagger completa

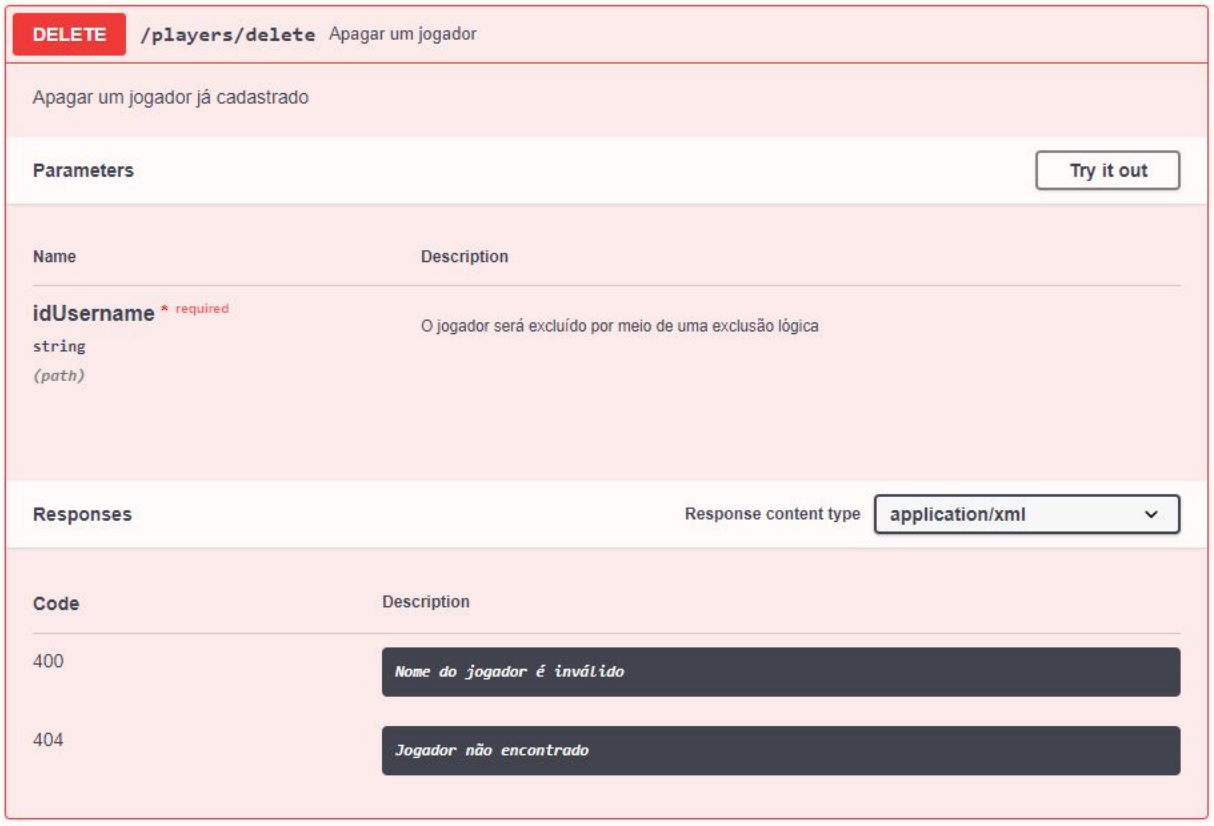
5.1.1. Entidade Players

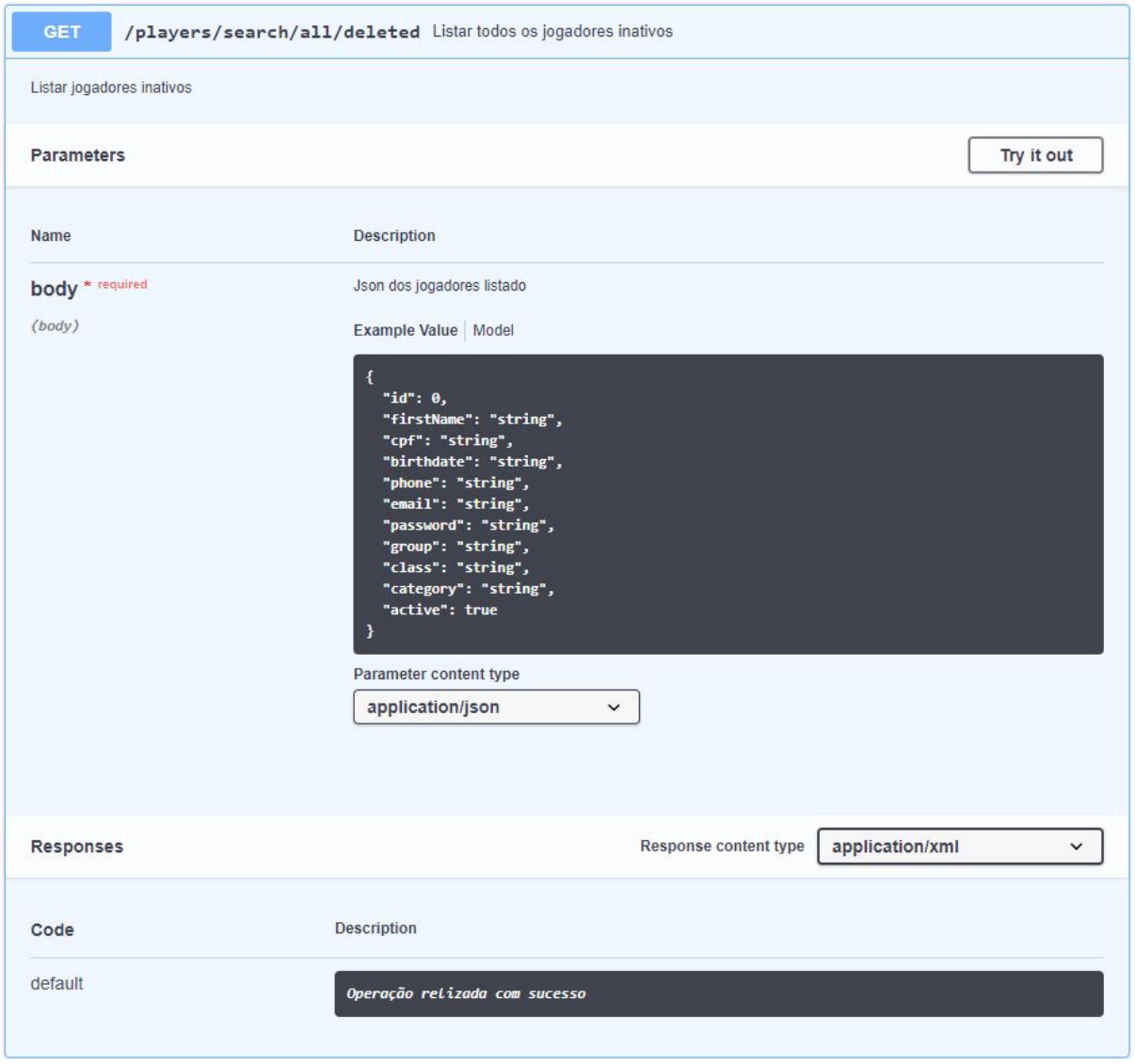



GET
/players/search/all
Listar todos os jogadores ativos
Listar jogadores
Parameters
Try it out
Name
Description
body * required
(body)
Json dos jogadores listado
Example Value | Model
{
"id": 0,
"firstName": "string",
"cpf": "string",
"birthdate": "string",
"phone": "string",
"email": "string",
"password": "string",
"group": "string",
"class": "string",
"category": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
default
Operação relizada com sucesso

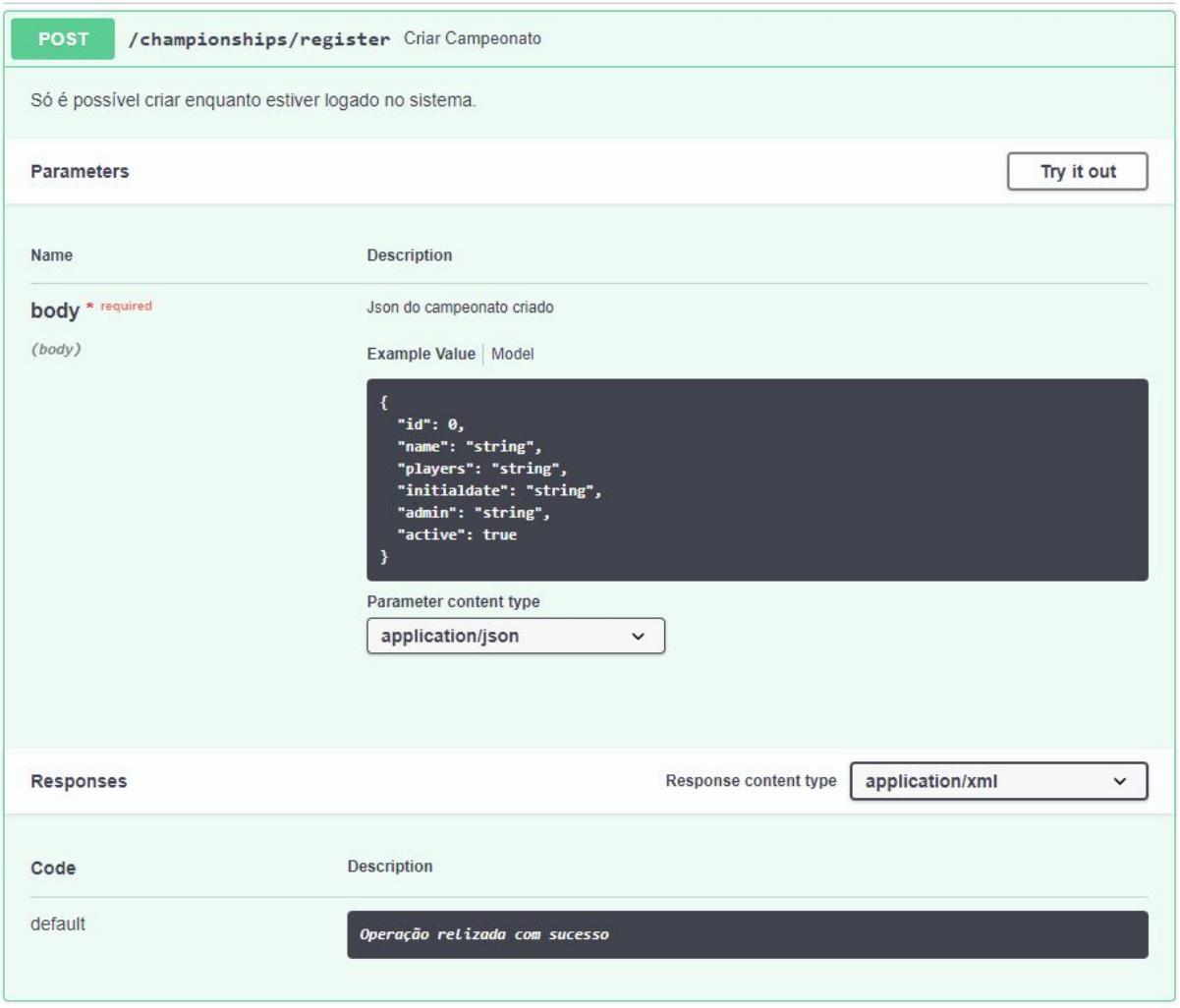

GET
/players/search
Listar um jogador ativo
Listar um, player ativo que foi buscado
Parameters
Try it out
Name
Description
body * required
(body)
Json do jogador listado
Example Value | Model
{
"id": 0,
"firstName": "string",
"cpf": "string",
"birthdate": "string",
"phone": "string",
"email": "string",
"password": "string",
"group": "string",
"class": "string",
"category": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
default
Operação relizada com sucesso

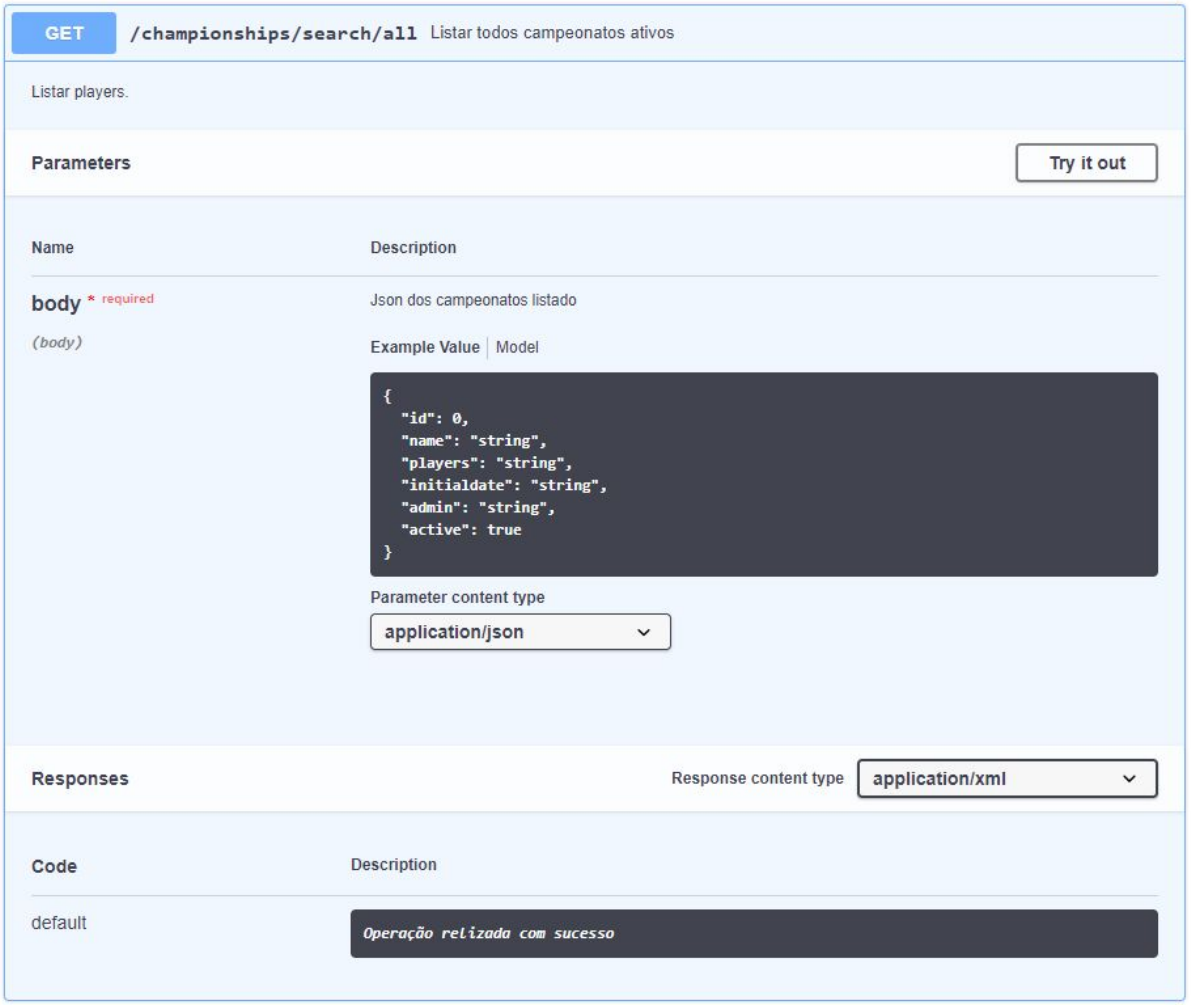

PUT
/players/edit
Atualizar um jogador ativo
Parameters
Try it out
Name
Description
body * required
(body)
Jogador que terá informações aualizadas
Example Value | Model
{
"id": 0,
"firstName": "string",
"cpf": "string",
"birthdate": "string",
"phone": "string",
"email": "string",
"password": "string",
"group": "string",
"class": "string",
"category": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
400
Atributo fornecido é inválido
404
Joagdor não encontrado
405
Validation exception

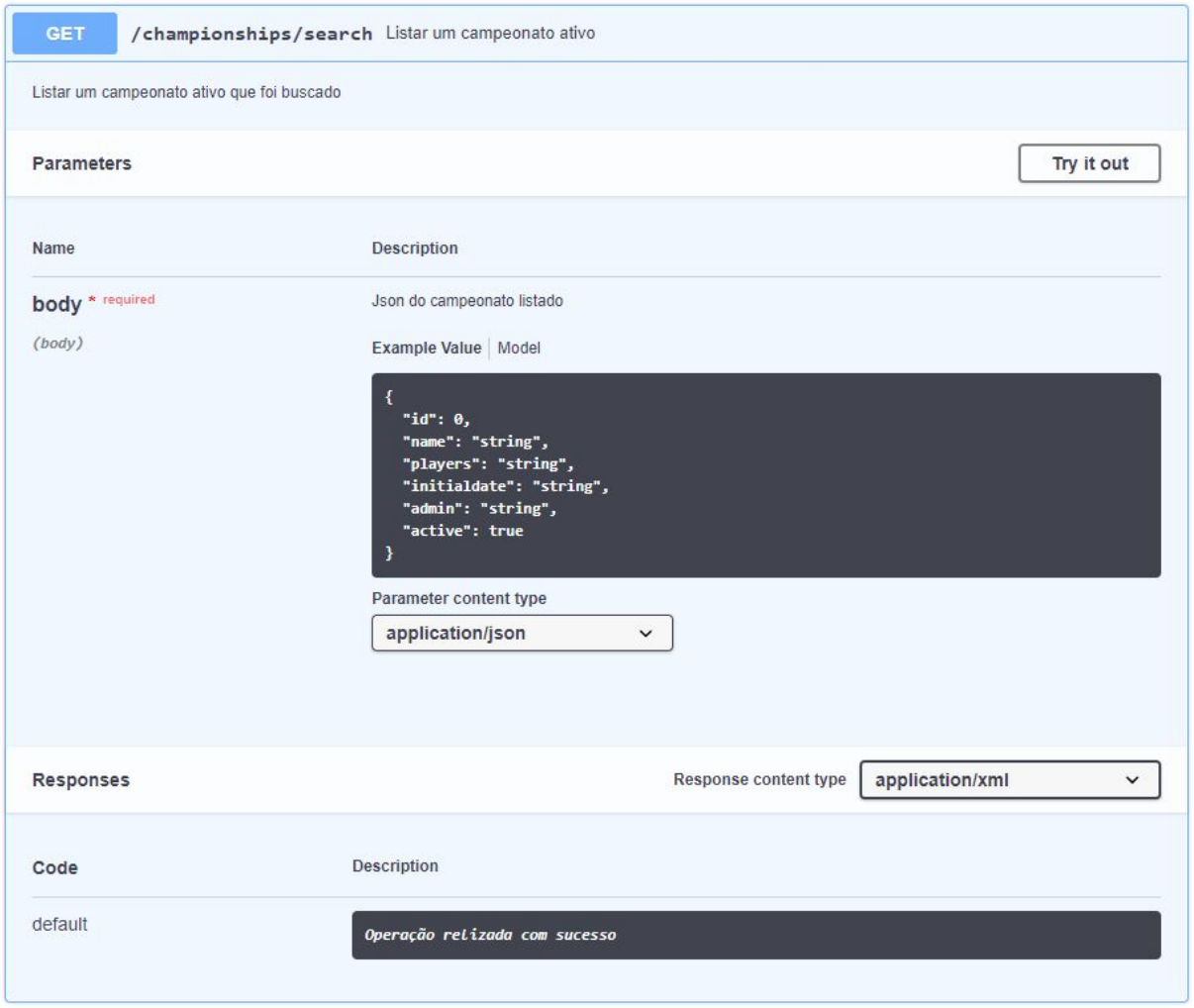

DELETE
/players/delete
Apagar um jogador
Apagar um jogador já cadastrado
Parameters
Try it out
Name
Description
idUsername
* required
string
(path)
O jogador será excluído por meio de uma exclusão lógica
Responses
Response content type
application/xml
Code
Description
400
Nome do jogador é inválido
404
Jogador não encontrado

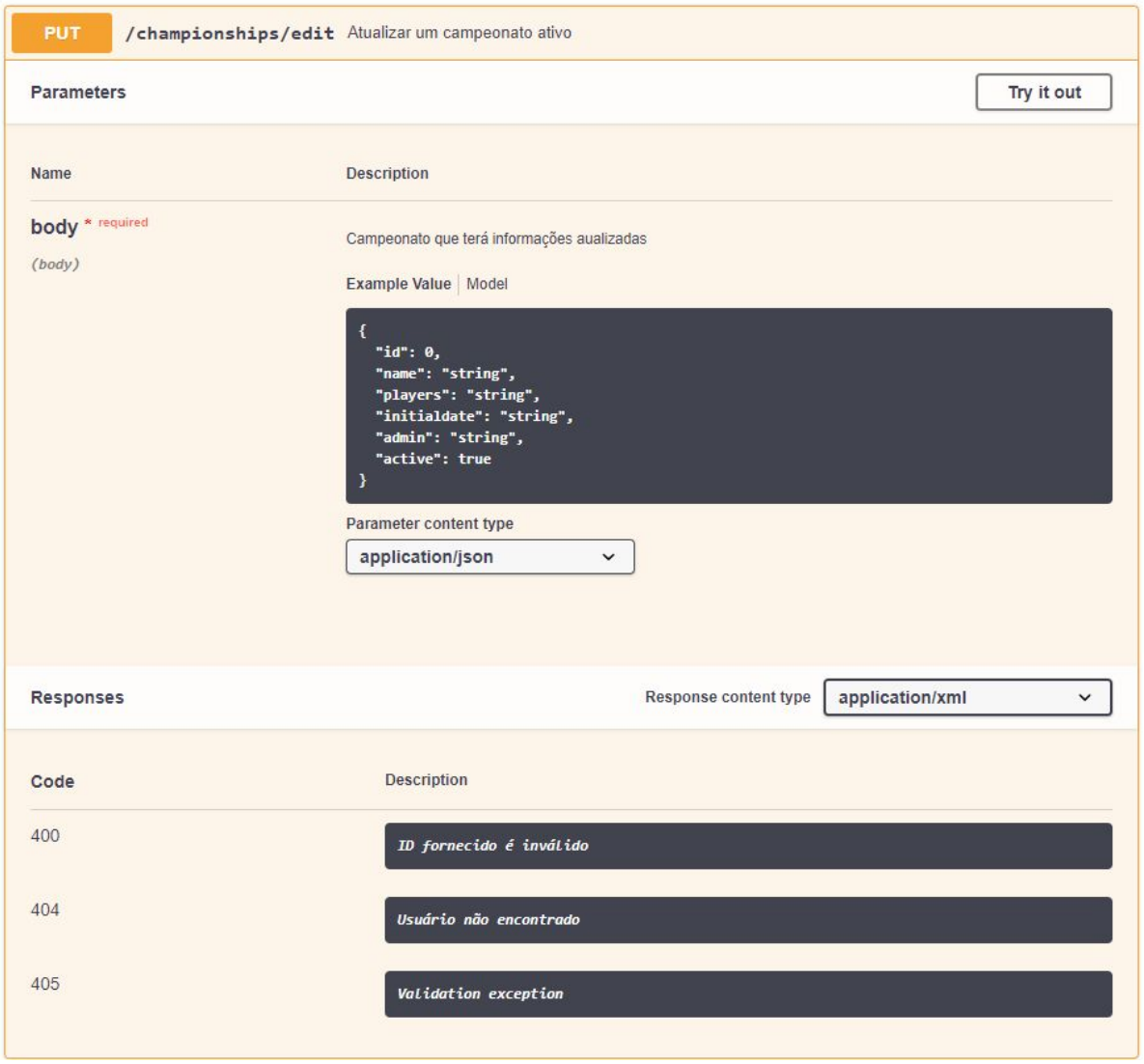

GET
/players/search/all/deleted
Listar todos os jogadores inativos
Listar jogadores inativos
Parameters
Try it out
Name
Description
body * required
(body)
Json dos jogadores listado
Example Value | Model
{
"id": 0,
"firstName": "string",
"cpf": "string",
"birthdate": "string",
"phone": "string",
"email": "string",
"password": "string",
"group": "string",
"class": "string",
"category": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
default
Operação relizada com sucesso

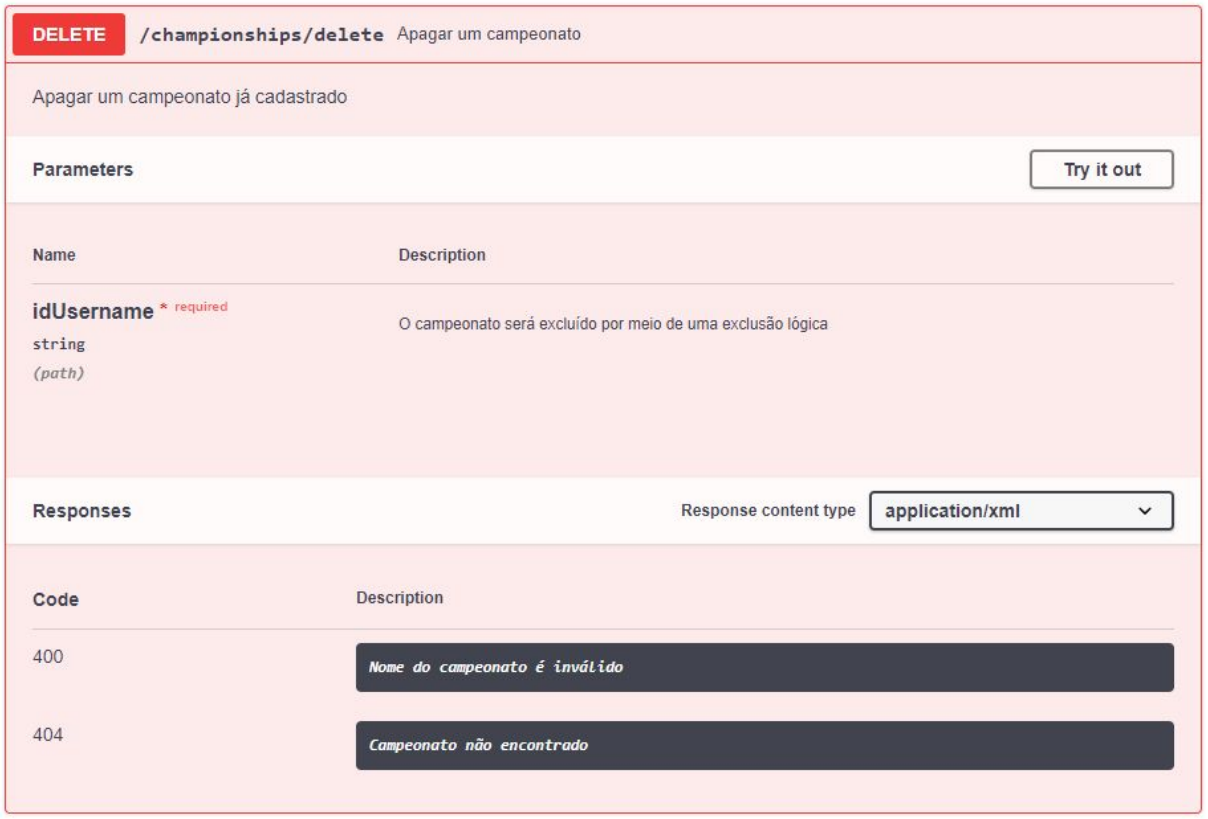
5.1.2. Entidade Championship




GET
/championships/search/all
Listar todos campeonatos ativos
Listar players.
Parameters
Try it out
Name
Description
body * required
(body)
Json dos campeonatos listado
Example Value
Model
{
"id": 0,
"name": "string",
"players": "string",
"initialdate": "string",
"admin": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
default
Operação relizada com sucesso


GET
/championships/search
Listar um campeonato ativo
Listar um campeonato ativo que foi buscado
Parameters
Try it out
Name
Description
body * required
(body)
Json do campeonato listado
Example Value | Model
{
"id": 0,
"name": "string",
"players": "string",
"initialdate": "string",
"admin": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
default
Operação relizada com sucesso


PUT
/championships/edit
Atualizar um campeonato ativo
Parameters
Try it out
Name
Description
body * required
(body)
Campeonato que terá informações aualizadas
Example Value | Model
{
"id": 0,
"name": "string",
"players": "string",
"initialdate": "string",
"admin": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
400
ID fornecido é inválido
404
Usuário não encontrado
405
Validation exception


DELETE
/championships/delete
Apagar um campeonato
Apagar um campeonato já cadastrado
Parameters
Try it out
Name
Description
idUsername * required
string
(path)
O campeonato será excluído por meio de uma exclusão lógica
Responses
Response content type
application/xml
Code
Description
400
Nome do campeonato é inválido
404
Campeonato não encontrado

admin

GET `/login` Login no sistema

Parameters Try it out

Name	Description
username * required string *(query)*	Nome de usuário para o login
password * required string *(query)*	Senha do usuário para validação do login

Responses Response content type application/xml

Code	Description

200	*Operação realizada com sucesso*
	
	Example Value | Model
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<!-- XML example cannot be generated -->`
	
	Headers:
	Name / Description / Type
	X-Rate-Limit — Chamadas permitidas por hora — integer
	X-Expires-After — Data em UTC quando sessão expira — string

400	*Usuário ou Senha inválido(s)*

GET `/logout` Logout do sistema

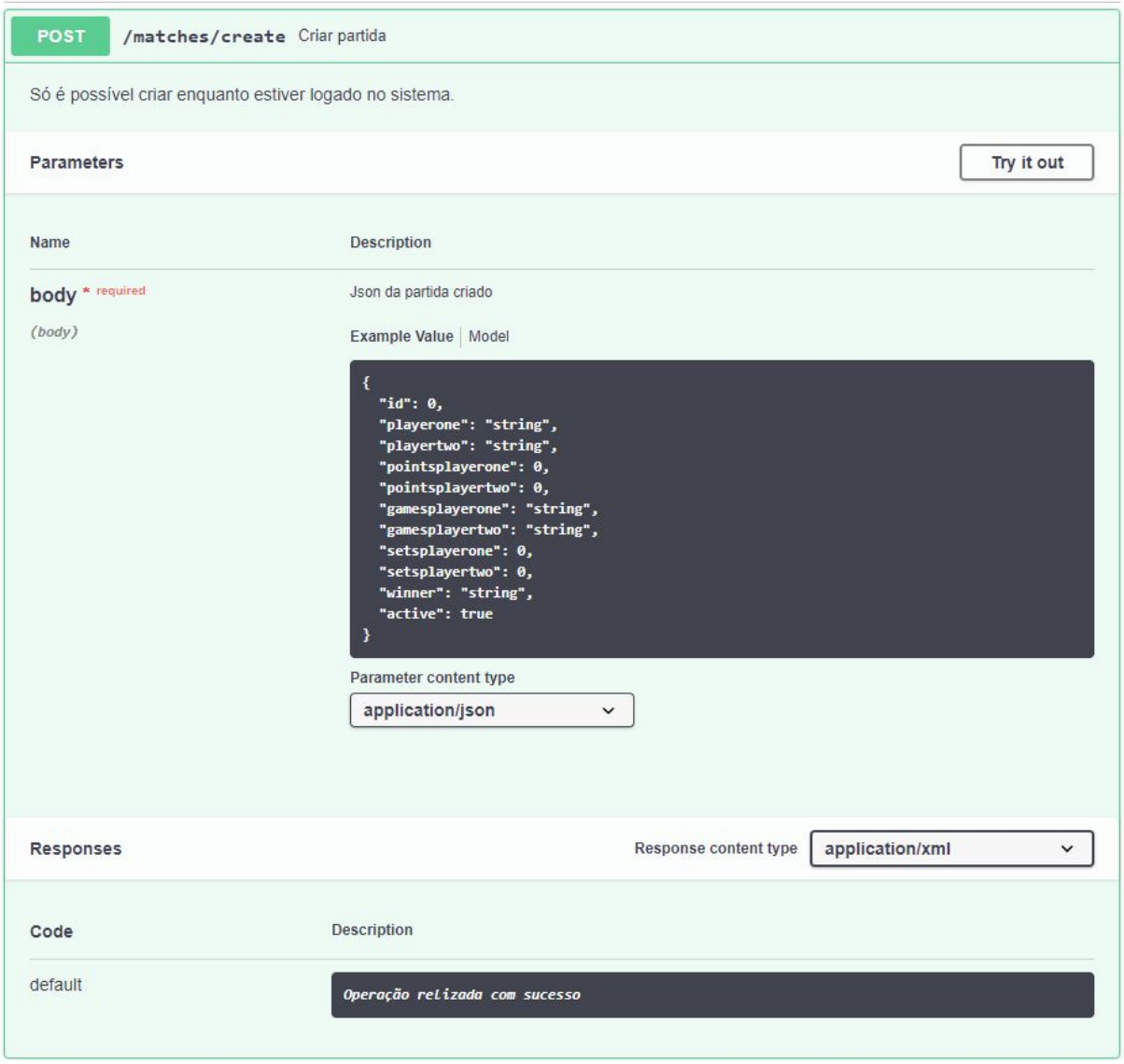
Parameters Try it out

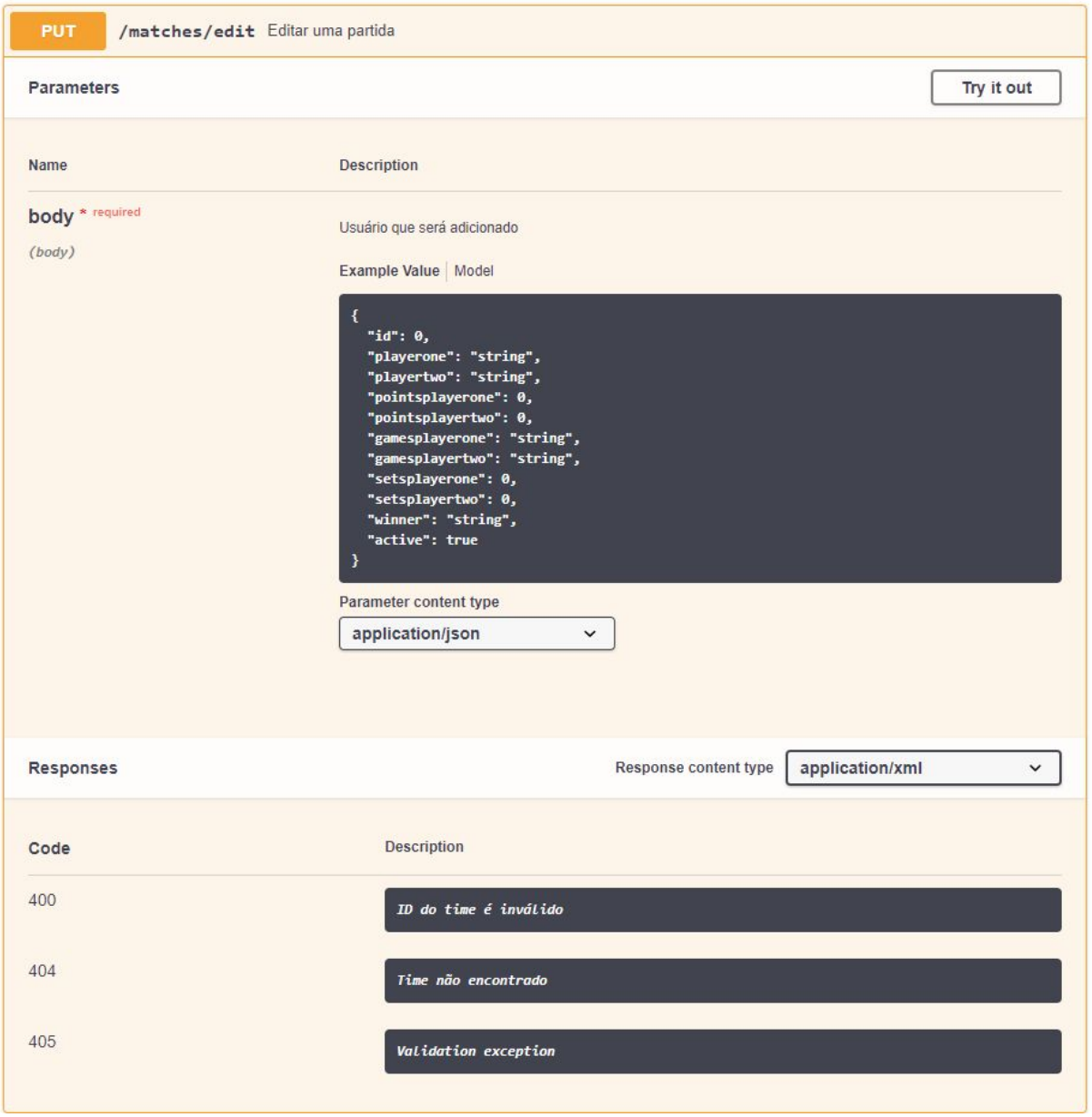
No parameters

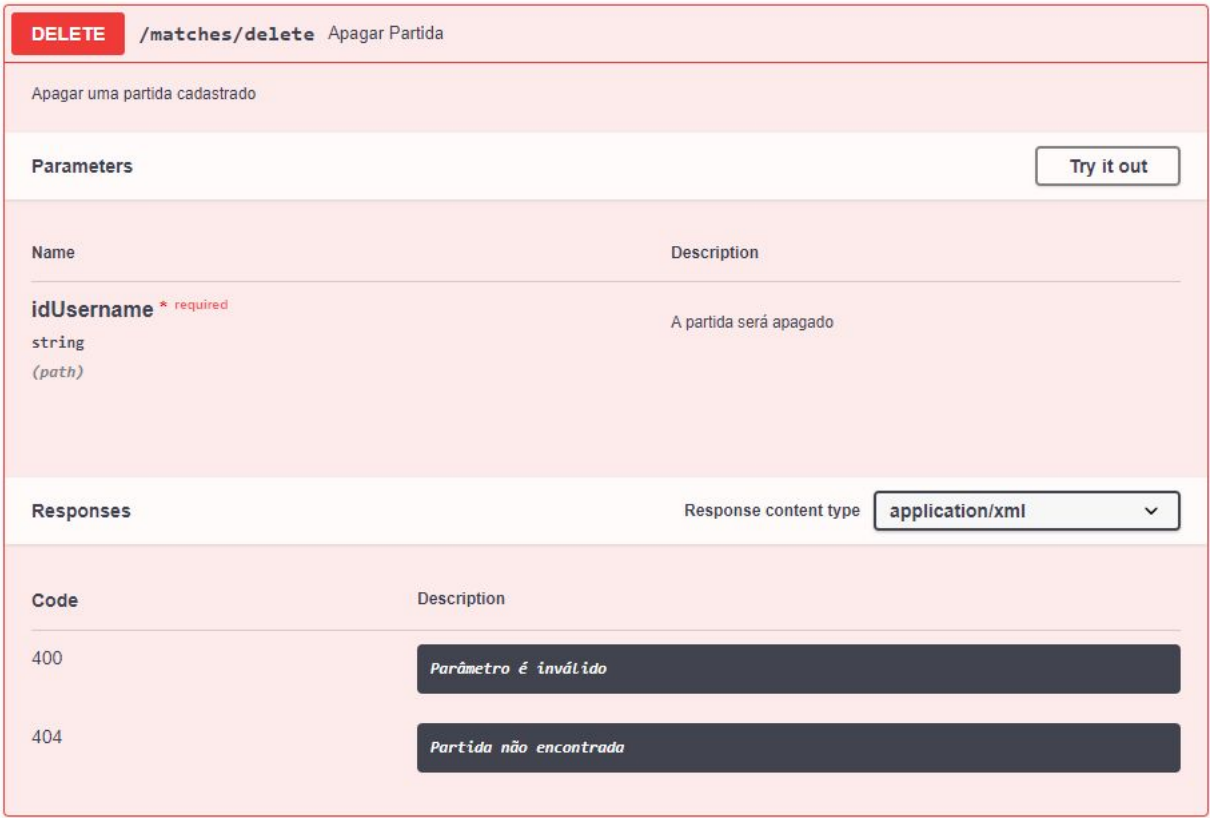
Responses Response content type application/xml

Code	Description
default	*Operação realizada com sucesso*

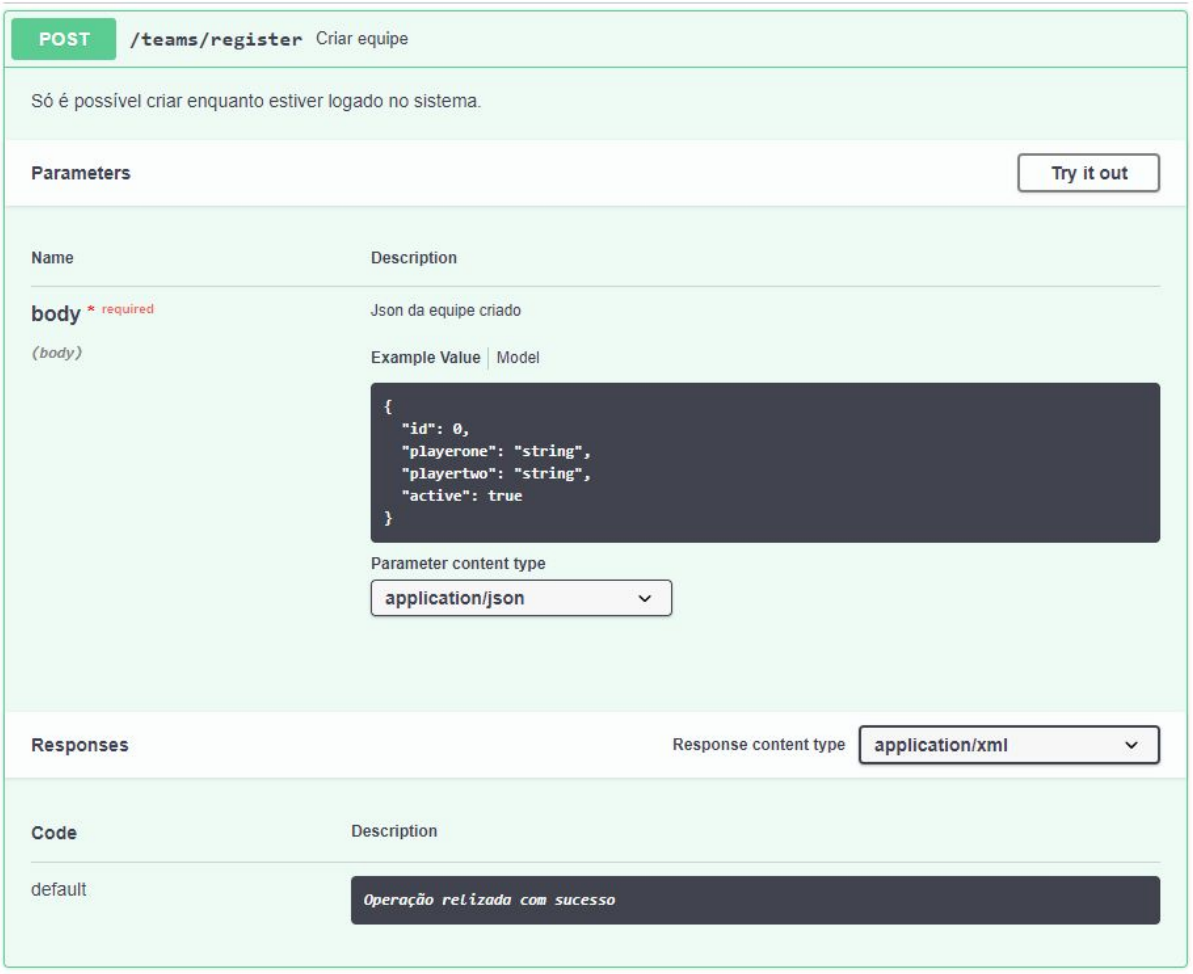
5.1.3. Entidade Match

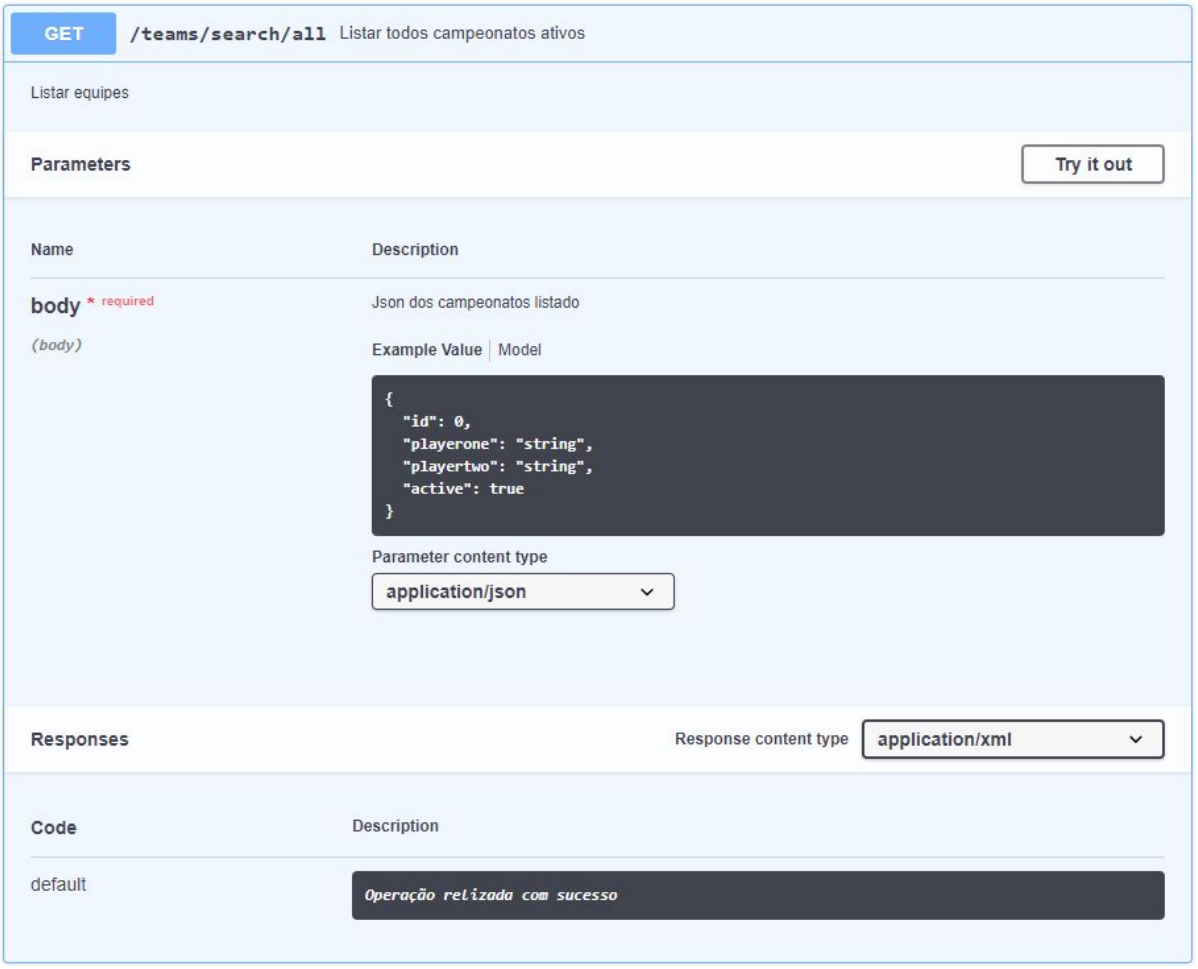

POST
/matches/create
Criar partida
Só é possível criar enquanto estiver logado no sistema.
Parameters
Try it out
Name
Description
body * required
(body)
Json da partida criado
Example Value | Model
{
"id": 0,
"playerone": "string",
"playertwo": "string",
"pointsplayerone": 0,
"pointsplayertwo": 0,
"gamesplayerone": "string",
"gamesplayertwo": "string",
"setsplayerone": 0,
"setsplayertwo": 0,
"winner": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
default
Operação relizada com sucesso

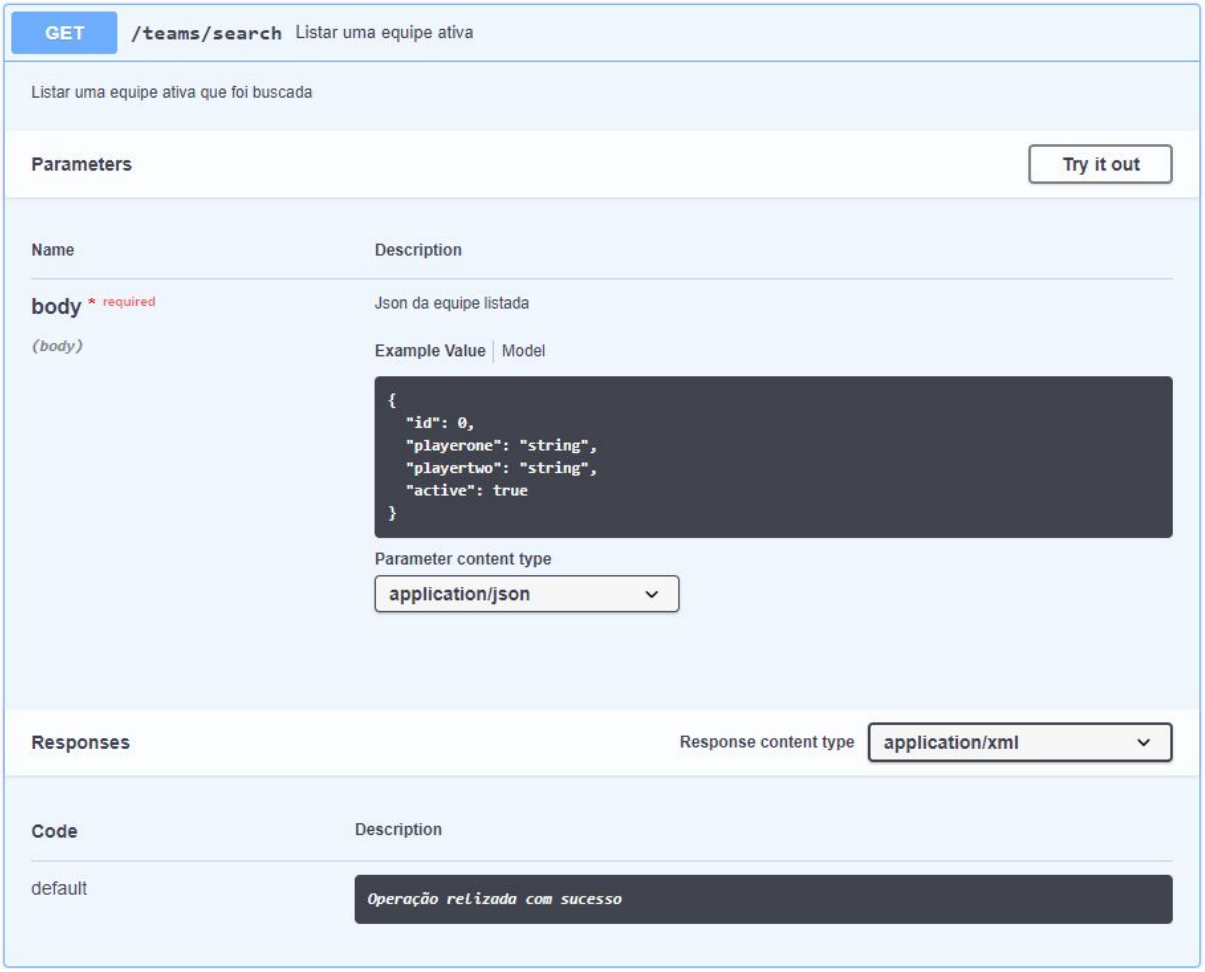

PUT
/matches/edit
Editar uma partida
Parameters
Try it out
Name
Description
body * required
(body)
Usuário que será adicionado
Example Value | Model
{
"id": 0,
"playerone": "string",
"playertwo": "string",
"pointsplayerone": 0,
"pointsplayertwo": 0,
"gamesplayerone": "string",
"gamesplayertwo": "string",
"setsplayerone": 0,
"setsplayertwo": 0,
"winner": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
400
ID do time é inválido
404
Time não encontrado
405
Validation exception

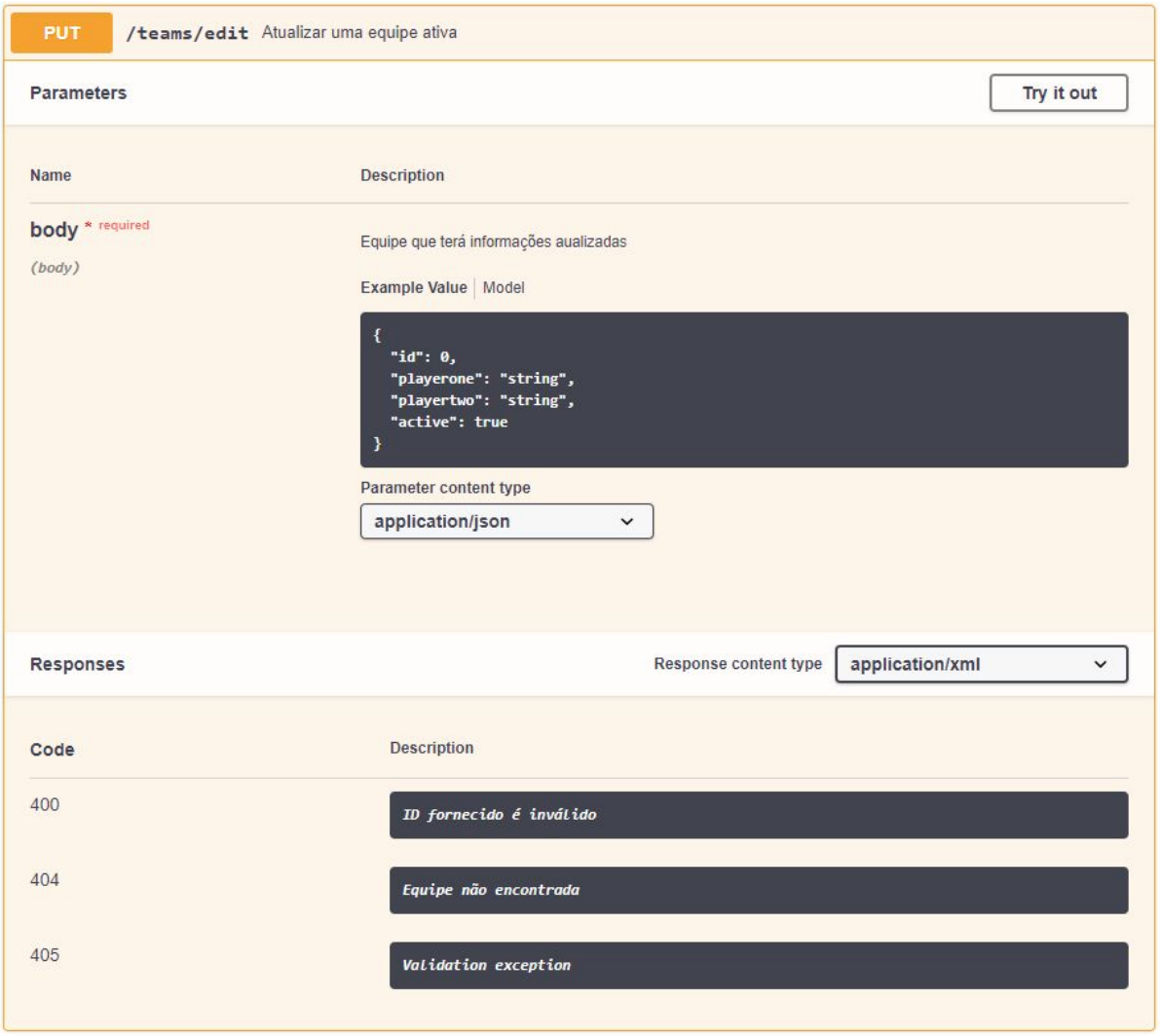


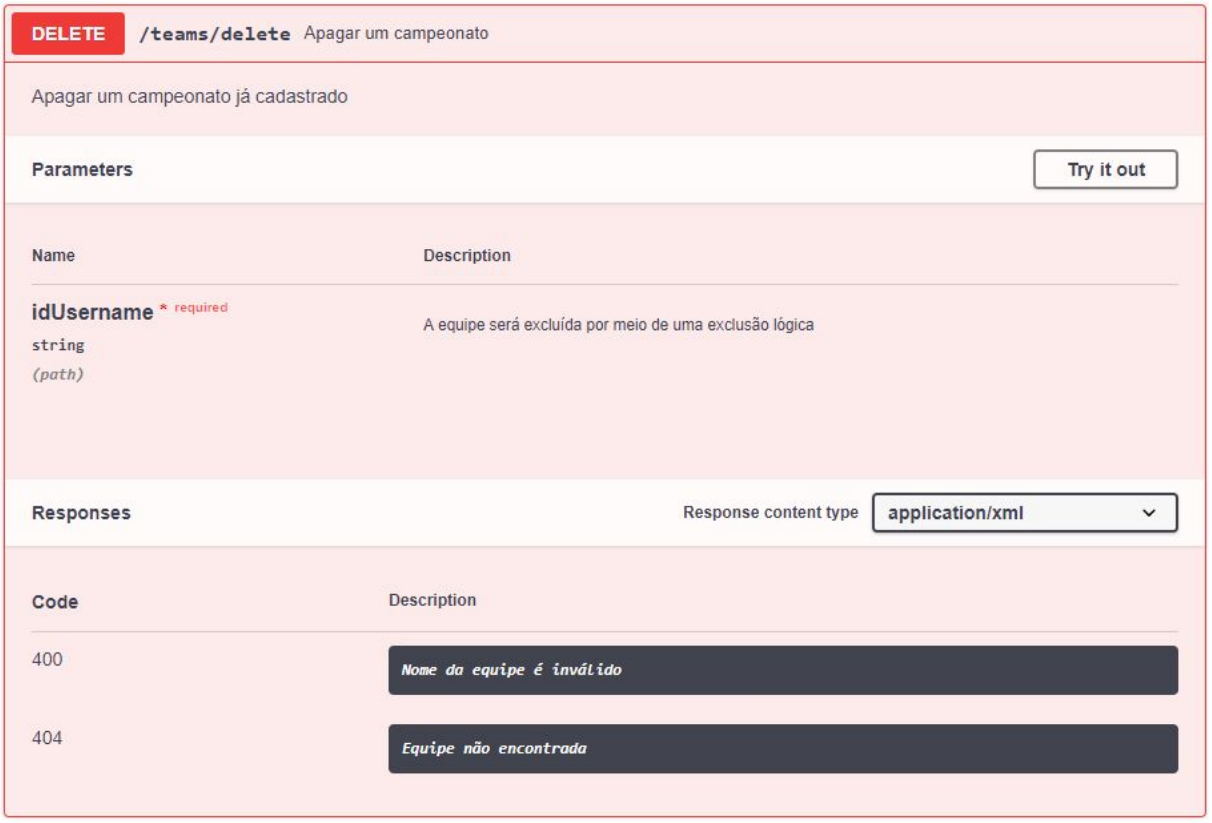
5.1.4. Entidade Ranking


POST
/teams/register
Criar equipe
Só é possível criar enquanto estiver logado no sistema.
Parameters
Try it out
Name
Description
body * required
(body)
Json da equipe criado
Example Value | Model
{
"id": 0,
"playerone": "string",
"playertwo": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
default
Operação relizada com sucesso


GET
/teams/search/all
Listar todos campeonatos ativos
Listar equipes
Parameters
Try it out
Name
Description
body * required
(body)
Json dos campeonatos listado
Example Value
Model
{
"id": 0,
"playerone": "string",
"playertwo": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
default
Operação relizada com sucesso


GET
/teams/search
Listar uma equipe ativa
Listar uma equipe ativa que foi buscada
Parameters
Try it out
Name
Description
body * required
(body)
Json da equipe listada
Example Value
Model
{
"id": 0,
"playerone": "string",
"playertwo": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
default
Operação relizada com sucesso


PUT
/teams/edit
Atualizar uma equipe ativa
Parameters
Try it out
Name
Description
body * required
(body)
Equipe que terá informações aualizadas
Example Value | Model
{
"id": 0,
"playerone": "string",
"playertwo": "string",
"active": true
}
Parameter content type
application/json
Responses
Response content type
application/xml
Code
Description
400
ID fornecido é inválido
404
Equipe não encontrada
405
Validation exception


DELETE
/teams/delete
Apagar um campeonato
Apagar um campeonato já cadastrado
Parameters
Try it out
Name
Description
idUsername * required
string
(path)
A equipe será excluída por meio de uma exclusão lógica
Responses
Response content type
application/xml
Code
Description
400
Nome da equipe é inválido
404
Equipe não encontrada