Engenharia e Gestão de Serviços

TaskMaster

Documentação: PointSystemAPI

Eduardo Alves: nºmec 104179

---

eduardoalves@ua.pt

Universidade de Aveiro
2025

universidade de aveiro

1 Introdução

A **PointSystemAPI** é um dos micro-serviços do **TaskMaster**, sendo responsável pela gestão dinâmica de pontuação dos utilizadores com base nas suas interações e desempenho. Este micro-serviço oferece funcionalidades essenciais, incluindo a **criação**, **atualização** e **remoção de utilizadores**, bem como a **atribuição** e **remoção de pontos** de forma flexível e transparente. Além disso, mantém um **ranking** dinâmico, ordenando os utilizadores de acordo com a sua pontuação total, permitindo uma visão clara do desempenho relativo entre eles. Além disto, uma característica interessante da API é o seu **histórico de alterações de pontos** para cada utilizador, que regista todas as transações, incluindo a data e a mensagem descritiva justificando a atribuição ou remoção de pontos. Esta API é também capaz de gerar a sua **chave da API**, tanto como verificar se a mesma é válida para dado utilizador.

2 Contexto

A API tem acesso a uma base de dados em **postgresql** que é estruturada para armazenar e gerenciar informações relacionadas com os utilizadores e os seus pontos. Esta base de dados contêm duas tabelas: **users** e **points**. A tabela users armazena dados essenciais sobre os utilizadores do micro-serviço, como o **nome**, **e-mail**, o **total de pontos** acumulados e a **chave da API**. A tabela points, por sua vez, regista todas as alterações no saldo de pontos dos utilizadores, incluindo o **valor da alteração**, a **data** e uma **mensagem associada**.

A relação entre essas duas tabelas é estabelecida através de uma chave estrangeira na tabela points que faz referência ao user_id na tabela users, que permite obter informações do histórico de pontos de cada utilizador. A nível do código, essa relação é definida pelas instruções "**relationship**" do SQLAlchemy, que configura uma conexão bidirecional entre as duas tabelas.

3 Endpoints

3.1 Criar Utilizador

POST /v1/users/

- **Descrição:** Cria um novo utilizador na plataforma.
- **Parâmetros:**
  - `name` (string, obrigatório) - Nome do utilizador.
  - `email` (string, obrigatório) - Email do utilizador.
- **Resposta:**

```
{
  "message": "Utilizador criado com sucesso!"
}
```

3.2 Atualizar Utilizador

PATCH /v1/users/{user_id}/

- **Descrição:** Atualiza o nome e email de um utilizador existente.
- **Parâmetros:**
  - `user_id` (int, obrigatório) - ID do utilizador a ser atualizado.
  - `name` (string, opcional) - Novo nome do utilizador.
  - `email` (string, opcional) - Novo email do utilizador.
- **Resposta:**

```
{
  "message": "Utilizador actualizado com sucesso!"
}
```

3.3 Remover Utilizador

DELETE /v1/users/{user_id}

- **Descrição:** Remove um utilizador da plataforma com base no seu ID.
- **Parâmetros:**
  - `user_id` (int, obrigatório) - ID do utilizador a ser removido.
- **Resposta:**

```
{
  "message": "Utilizador removido com sucesso!"
}
```

3.4 Obter Ranking de Utilizadores

GET /v1/users/

- **Descrição:** Retorna a lista de utilizadores ordenada por total de pontos (ranking).
- **Resposta:**

```
{
  "ranking": [
    {
      "rank": 1,
      "user_id": 4,
      "name": "Diogo Gomes",
      "Email": "DG@ua.pt",
      "total_points": 9999
    },
```

```
    {
      "rank": 2,
      "user_id": 1,
      "name": "eduardo",
      "Email": "eduardo@ua.pt",
      "total_points": 1900
    }
  ]
}
```


3.5 Adicionar Pontos

POST /v1/users/{user_id}/points/

- **Descrição:** Adiciona pontos a um utilizador específico.
- **Parâmetros:**
  - `user_id` (int, obrigatório) - ID do utilizador.
  - `points` (int, obrigatório) - Quantidade de pontos a adicionar.
  - `message` (string, obrigatório) - Motivo da atribuição dos pontos.
- **Resposta:**


```
{
  "message": "Pontos atribuidos com sucesso!",
  "total_points": 1762
}
```


3.6 Remover Pontos

DELETE /v1/users/{user_id}/points/

- **Descrição:** Remove pontos a um utilizador específico.
- **Parâmetros:**
  - `user_id` (int, obrigatório) - ID do utilizador.
  - `points` (int, obrigatório) - Quantidade de pontos a remover.
  - `message` (string, obrigatório) - Motivo da atribuição dos pontos.
- **Resposta:**


```
{
  "message": "Pontos removidos com sucesso!",
  "total_points": 1693
}
```

3.7 Histórico de Pontos

GET /v1/points/history/{user_id}

- **Descrição:** Retorna um histórico das mudanças de pontos do utilizador.
- **Parâmetros:**
  - `user_id` (int, obrigatório) - ID do utilizador.
  - `skip` (int, opcional) - Número de entradas a ignorar (para paginação).
  - `limit` (int, opcional) - Número máximo de resultados a retornar.
- **Resposta:**


```
{
  "history": [
    {
      "points_change": +50,
      "change_date": "2025-03-14T12:00:00",
      "message": "Completed bonus task"
    },
    {
      "points_change": -10,
      "change_date": "2025-03-13T15:45:00",
      "message": "Penalty for late submission"
    }
  ]
}
```


3.8 Gerar chave API

POST /v1/generate-api-key

- **Descrição:** Gera uma chave API para um dado utilizador.
- **Parâmetros:**
  - `user_id` (int, obrigatório) - ID do utilizador.
- **Resposta:**


```
{
  "api_key": "0923
     dc8c52c63e13da53166573a43f96d8b3800fac3c52a1102b4422b77e34f1
     "
}
```

3.9 Validar uma chave API

GET /v1/validate-api-key

- **Descrição:** Valida uma chave API para efeitos de teste.
- **Parâmetros:**
  - `api_key` (string, obrigatório) - chave API para testar se é válida ou não.
- **Resposta:**

```
{
  "valid": true,
  "user_id": 1
}
```

4 Conclusão

Esta documentação serve como base para entender o funcionamento do micro-serviço PointSystemAPI, com um foco especial na visão geral dos endpoints da API, explicando as suas funcionalidades essenciais, parâmetros e exemplos de resultados positivos esperados ao executar cada endpoint. É também de notar que o último endpoint para validar as chaves da API serve para efeitos de testagem, no produto final, em princípio, o mesmo não será utilizado.