# Performance Sports - Postman Collection Blueprint

Blueprint para construir e executar a colecao Postman do backend (v1). O objetivo e validar, no Postman, os fluxos criticos antes do frontend: autenticacao, tokens de 1 minuto, cooldown, escopo por professor, eventos e pagamentos. Este documento e propositalmente prescritivo para reduzir ambiguidade durante a implementacao.

## 1. Estrutura da colecao

Criar uma colecao chamada **PerformanceSports API - v1** com os seguintes folders (na ordem):
1) 00 - Health & Meta
2) 10 - Auth (Public)
3) 20 - Modalidades
4) 30 - Alunos
5) 40 - Professores
6) 50 - Eventos
7) 60 - Pagamentos
8) 90 - Negativos (Security/Scopes)
9) 99 - Regressao (Runs)

## 2. Environment (variaveis)

**Environment: variaveis recomendadas**

| Chave | Valor | Observacoes |
|---|---|---|
| baseUrl | http://localhost:8080 | API base (ajustar para Railway/Render em prod) |
| accessToken | (vazio) | Preenchido apos login; usado no header Authorization |
| tokenType | Bearer | Padrao |
| alunoEmail | aluno.teste+001@exemplo.com | Usar alias para reexecucoes |
| alunoPassword | SenhaForte@123 | Somente para testes locais |
| profEmail | prof.teste+001@exemplo.com | Professor pendente/aprovado conforme cenarios |
| profPassword | SenhaForte@123 | Somente para testes locais |
| superEmail | super.admin@exemplo.com | Conta seed (migracao/seed) |
| superPassword | SenhaForte@123 | Conta seed |
| lastAlunoId | (vazio) | Armazenar IDs retornados por respostas |
| lastProfessorId | (vazio) | Armazenar IDs retornados por respostas |
| lastEventoId | (vazio) | Armazenar IDs retornados por respostas |
| lastPagamentoId | (vazio) | Armazenar IDs retornados por respostas |

## 3. Headers padrao

Para requests autenticadas, definir o header:
`Authorization: {{tokenType}} {{accessToken}}`

e `Content-Type: application/json` quando houver body.

## 4. Scripts de apoio (recomendado)

Sugestao operacional (nao obrigatoria):
- Pre-request em nivel de folder para garantir que requests autenticadas falhem rapidamente caso accessToken esteja vazio.
- Tests para capturar IDs e salvar em variaveis (ex.: lastAlunoId, lastEventoId).
- Para tokens de 1 minuto: usar cenarios com delay manual ou execucao imediata; a expiracao deve ser validada via resposta 400/422/429 conforme regra.

# 00 - Health & Meta

## Requests

| Request | Metodo | Path | Auth | Observacoes (inclui asserts) |
|---|---|---|---|---|
| **Health** | GET | /api/actuator/health (ou /health) | Publico | Validar que a API esta no ar. **Asserts:** Status 200; body contem status=UP |
| **Versao/Meta (opcional)** | GET | /api/meta | Publico | Se implementado: retorna versao do build, commit, timestamp. **Asserts:** Status 200 |

# 10 - Auth (Public)

## Requests

| Request | Metodo | Path | Auth | Observacoes (inclui asserts) |
|---|---|---|---|---|
| **Register Aluno** | POST | /api/auth/register-aluno | Publico | Resposta neutra (nao revelar existencia do email). **Body:** `{"nome":"...","sobrenome":"...","email":"{{alunoEmail}}","password":"{{alunoPassword}}","dataNascimento":"2000-01-01","cpf":"12345678909","modalidadeIds":[1],"enderecos":[{...}] }` **Asserts:** Status 202; message presente |
| **Confirm Register (codigo)** | POST | /api/auth/confirm-register | Publico | Executar imediatamente apos receber codigo. Em producao o codigo vem por email. **Body:** `{"email":"{{alunoEmail}}","code":""}` **Asserts:** Status 200; message |
| **Login Aluno** | POST | /api/auth/login | Publico | Capturar accessToken. **Body:** `{"email":"{{alunoEmail}}","password":"{{alunoPassword}}"}` **Asserts:** Status 200; salvar accessToken; profileType=ALUNO |
| **Forgot Password** | POST | /api/auth/forgot-password | Publico | Resposta neutra; gera token 60s. **Body:** `{"email":"{{alunoEmail}}"}` **Asserts:** Status 200; message |

| | | | | |
|---|---|---|---|---|
| **Reset Password** | POST | /api/auth/reset-password | Publico | Token 60s; consumir apos sucesso.<br>**Body:** `{"email":"{{alunoEmail}}","code":"","newPassword":"NovaSenha@123"}`<br>**Asserts:** Status 200; message |
| **Cooldown (5 erros)** | POST | /api/auth/confirm-register (ou reset) | Publico | Repetir 5 vezes com code invalido.<br>**Body:** `{"email":"{{alunoEmail}}","code":"000000"}`<br>**Asserts:** Na 5a tentativa: Status 429; informar aguardar 10 min |

## 20 - Modalidades

### Requests

| Request | Metodo | Path | Auth | Observacoes (inclui asserts) |
|---|---|---|---|---|
| **List Modalidades** | GET | /api/modalidades | JWT | Leitura global.<br>**Asserts:** Status 200; lista; inclui professorDefaultId quando existir |
| **Create Modalidade** | POST | /api/modalidades | JWT (ADMIN/SUPER_ADMIN) | Usar conta SUPER_ADMIN para criar.<br>**Body:** `{"nome":"Bike","descricao":"...","valor":120.00,"professorDefaultId":}`<br>**Asserts:** Status 201; id retornado |

# 30 - Alunos

## Requests

| Request | Metodo | Path | Auth | Observacoes (inclui asserts) |
|---|---|---|---|---|
| **List Alunos** | GET | /api/alunos | JWT | Professor USER: so escopo. ADMIN/SUPER_ADMIN: todos. **Asserts:** Status 200; nao retornar cpf/dataNascimento em list |
| **Get Aluno Detail** | GET | /api/alunos/{{lastAlunoId}} | JWT | Detalhe pode retornar cpfMasked. **Asserts:** Status 200; cpfMasked presente; cpf puro ausente |
| **Approve Aluno** | POST | /api/alunos/{{lastAlunoId}}/approve | JWT (PROF/ADMIN/SUPER) | Aprovacao ativa o aluno. **Body:** `{"approved":true}` **Asserts:** Status 200; aluno passa para ACTIVE |

# 40 - Professores

## Requests

| Request | Metodo | Path | Auth | Observacoes (inclui asserts) |
|---|---|---|---|---|
| **Register Professor** | POST | /api/professores/register | Publico | Cria professor pendente de aprovacao SUPER_ADMIN. **Body:** `{"nome":"...","sobrenome":"...","email":"{{profEmail}}","password":"{{profPassword}}","dataNascimento":"1990-01-01","cpf":"12345678909","modalidadeIds":[1],"pagamentoFormato":"AULA","enderecos":[{...}] }` **Asserts:** Status 202; message |
| **Approve Professor** | POST | /api/professores/{{lastProfessorId}}/approve | JWT (SUPER_ADMIN) | Aprova e define role ADMIN/USER. **Body:** `{"approved":true,"role":"USER"}` **Asserts:** Status 200 |

# 50 - Eventos

## Requests

| Request | Metodo | Path | Auth | Observacoes (inclui asserts) |
|---|---|---|---|---|
| **Create Evento** | POST | /api/eventos | JWT (PROF/ADMIN/SUPER) | Cria evento pendente de validacao; alunoIds devem estar no escopo. **Body:** `{"nomeDescricao":"Treino Especial","dataInicio":"2026-02-01","dataFim":"2026-02-01","hora":"07:00:00","modalidadeId":1,"alunoIds":[{{lastAlunoId}}]}` **Asserts:** Status 201; status=PENDING_VALIDATION; salvar lastEventoId |
| **Approve Evento** | POST | /api/eventos/{{lastEventoId}}/approve | JWT (SUPER_ADMIN) | Aprova e dispara convites por e-mail. **Body:** `{"approved":true}` **Asserts:** Status 200; evento status=APPROVED |
| **Confirm Participacao** | POST | /api/eventos/{{lastEventoId}}/confirm | JWT (ALUNO) | Aluno confirma YES/NO. **Body:** `{"confirm":"YES"}` **Asserts:** Status 200 |

# 60 - Pagamentos

## Requests

| Request | Metodo | Path | Auth | Observacoes (inclui asserts) |
|---|---|---|---|---|
| **Create Payment Link** | POST | /api/pagamentos/links | JWT (PROF/ADMIN/SUPER) | Retorna paymentUrl; nao retornar gateway_reference_id. **Body:** `{"alunoId":{{lastAlunoId}},"valor":120.00,"dueDate":"2026-02-10","modalidadeId":1,"eventoId":null}` **Asserts:** Status 201; paymentUrl presente; gateway_reference_id ausente |
| **List Pagamentos (opcional)** | GET | /api/pagamentos?status=PENDING | JWT | Se implementado: listar por escopo. **Asserts:** Status 200 |
| **Webhook PagSeguro** | POST | /api/pagamentos/webhook/pagseguro | Publico (assinatura) | Testar com payload fake em dev; em prod validar assinatura. **Asserts:** Status 200; idempotente |

# 90 - Negativos (Security/Scopes)

## Requests

| Request | Metodo | Path | Auth | Observacoes (inclui asserts) |
|---|---|---|---|---|
| **Protected without token** | GET | /api/alunos | Sem JWT | Deve falhar.<br>**Asserts:** Status 401 |
| **Professor USER acessa aluno fora do escopo** | GET | /api/alunos/ | JWT (PROF USER) | Nao vazar informacao de ownership.<br>**Asserts:** Status 403 (ou 404 generico) |
| **ADMIN endpoint com USER** | POST | /api/modalidades | JWT (USER) | Criacao deve falhar.<br>**Asserts:** Status 403 |
| **Cooldown token (apos 5 erros)** | POST | /api/auth/confirm-register | Publico | Depois do cooldown: tentar novamente antes de 10 min.<br>**Asserts:** Status 429 |

# 99 - Regressao (Runs)

Sugestao de sequencia para Collection Runner (happy path):
Health -> (Seed SUPER_ADMIN se necessario) -> Create Modalidade -> Register Professor -> Approve Professor -> Register Aluno -> Confirm Register -> (Professor Approve Aluno) -> Login Aluno -> Create Evento -> Approve Evento -> Confirm Participacao -> Create Payment Link.

**Importante:** os passos com codigo (confirm-register/reset) dependem do recebimento do e-mail em ambiente real. Em ambiente de desenvolvimento, voce pode habilitar um provedor SMTP de teste para capturar os codigos.

## Anexos - Checklist rapido

- Verificar expiracao 60s (confirm/register e reset) - Verificar 5 erros -> 429 e cooldown 10 min - Verificar que listagens nao trazem CPF/data nasc completa - Verificar escopo do professor USER - Verificar que endpoints ADMIN falham para USER - Verificar que paymentUrl retorna, mas ids internos do gateway nao