Performance Sports - Contrato de API (v1)

Contrato de API para o sistema Performance Sports. Define endpoints, autenticacao, escopo por role, padroes de erro, schemas (alto nivel) e regras de privacidade. Este contrato e orientado ao uso no Postman e integracao com o frontend Angular.

1. Convencoes gerais

- Base path: /api - Autenticacao: Bearer JWT em todas as rotas privadas - Formato: JSON (request/response), exceto webhooks - Datas: ISO-8601 (DATE: YYYY-MM-DD; DATETIME: YYYY-MM-DDTHH:mm:ss) - Privacidade: nao retornar senha/hash/tokens; CPF sempre mascarado quando retornado - Erros: padrao {status, error, message, timestamp, fieldErrors?}

2. Autorizacao e escopo

Roles: SUPER_ADMIN, ADMIN, USER. **Perfis:** ALUNO, PROFESSOR. **Escopo:** - ALUNO: acesso apenas ao proprio perfil, pagamentos e inscricoes. - PROFESSOR USER: acesso apenas aos alunos/pagamentos/eventos do seu escopo. - PROFESSOR ADMIN: acesso global (todos alunos). - SUPER_ADMIN: acesso total + validacoes.

3. Padrao de respostas

Sucesso: JSON com payload minimo necessario. **Erros:** - 400/422: validacao (fieldErrors) - 401: nao autenticado - 403: acesso negado - 404: recurso inexistente (sem vazar escopo) - 429: cooldown/limite de tentativas

4. Endpoints - Autenticacao

POST /api/auth/register-aluno

Auth: Publico

Roles/Perfis: Publico

Request (schema):

```
{ "nome": "string", "sobrenome": "string", "email": "string", "password": "string", "dataNascimento": "YYYY-MM-DD", "cpf": "string(11 digitos)", "modalidadeIds": [1, 2], "enderecos": [ { ... }, { ... } ] }
```

Response (schema):

```
202 { "message": "Se aplicavel, enviamos um codigo para seu e-mail." }
```

Regras/Notas: Cria usuario em PENDING_EMAIL. Token REGISTER_VERIFY expira em 60s. Resposta sempre neutra.

POST /api/auth/confirm-register

Auth: Publico

Roles/Perfis: Publico

Request (schema):

```
{ "email": "string", "code": "string" }
```

Response (schema):

```
200 { "message": "E-mail confirmado. Aguarde aprovacao do professor responsavel." }
```

Regras/Notas: 5 codigos invalidos geram cooldown 10 min (HTTP 429). Token consumido apos sucesso.

POST /api/auth/login

Auth: Publico

Roles/Perfis: Publico

Request (schema):

```
{ "email": "string", "password": "string" }
```

Response (schema):

```
200 { "accessToken": "jwt", "tokenType": "Bearer", "expiresIn": 3600, "profile": {
"userId": 123, "displayName": "Nome Sobrenome", "roles": ["USER"], "profileType":
"ALUNO" } }
```

Regras/Notas: Bloquear login se PENDING_EMAIL/PENDING_APPROVAL/SUSPENDED com mensagem apropriada (sem detalhes internos).

POST /api/auth/forgot-password

Auth: Publico

Roles/Perfis: Publico

Request (schema):

```
{ "email": "string" }
```

Response (schema):

```
200 { "message": "Se aplicavel, enviamos um codigo para seu e-mail." }
```

Regras/Notas: Token PASSWORD_RESET expira em 60s. Resposta neutra. 5 erros => cooldown 10 min.

POST /api/auth/reset-password

Auth: Publico

Roles/Perfis: Publico

Request (schema):

```
{ "email": "string", "code": "string", "newPassword": "string" }
```

Response (schema):

```
200 { "message": "Senha alterada com sucesso." }
```

Regras/Notas: Consome token apos sucesso. Nunca retornar qualquer dado sensivel.

5. Endpoints - Modalidades

GET /api/modalidades

Auth: JWT

Roles/Perfis: ALUNO/PROFESSOR/ADMIN/SUPER_ADMIN

Request (schema):

```
(sem body)
```

Response (schema):

```
200 [ { "id": 1, "nome": "string", "descricao": "string", "valor": 120.00,
"professorDefaultId": 10 } ]
```

Regras/Notas: Leitura global. professorDefaultId usado para auto atribuicao do professor ao aluno.

POST /api/modalidades

Auth: JWT

Roles/Perfis: ADMIN/SUPER_ADMIN

Request (schema):

```
{ "nome": "string", "descricao": "string", "valor": 120.00, "professorDefaultId": 10 }
```

Response (schema):

```
201 { "id": 1, "nome": "string", "descricao": "string", "valor": 120.00,
"professorDefaultId": 10 }
```

Regras/Notas: Somente ADMIN/SUPER_ADMIN. Validar existencia do professorDefaultId se informado.

6. Endpoints - Alunos

GET /api/alunos

Auth: JWT

Roles/Perfis: PROFESSOR USER (escopo) | ADMIN | SUPER_ADMIN

Request (schema):

```
(sem body)
```

Response (schema):

```
200 [ { "id": 1, "nome": "string", "sobrenome": "string", "cidade": "string",
"statusAtivacao": "PENDING/ACTIVE", "modalidades": [ { "id": 1, "nome": "..." } ] } ]
```

Regras/Notas: Nao retornar CPF nem dataNascimento em listagens.

GET /api/alunos/{id}

Auth: JWT

Roles/Perfis: ALUNO (proprio) | PROFESSOR (escopo) | ADMIN | SUPER_ADMIN

Request (schema):

```
(sem body)
```

Response (schema):

```
200 { "id": 1, "nome": "string", "sobrenome": "string", "cpfMasked": "***.***.***-09",
"dataNascimento": "YYYY-MM-DD", "modalidades": [ { "id": 1, "nome": "...", "valor":
120.00 } ], "enderecos": [ { ... } ], "professorResponsavel": { "id": 10, "nome": "..."
}, "statusAtivacao": "PENDING/ACTIVE" }
```

Regras/Notas: CPF sempre mascarado. Aplicar escopo: professor USER so acessa seus alunos.

POST /api/alunos/{id}/approve

Auth: JWT

Roles/Perfis: PROFESSOR (escopo) | ADMIN | SUPER_ADMIN

Request (schema):

```
{ "approved": true }
```

Response (schema):

```
200 { "message": "Aluno aprovado." }
```

Regras/Notas: Quando aprovado: aluno ACTIVE e user.enabled=true.

7. Endpoints - Professores

POST /api/professores/register

Auth: Publico

Roles/Perfis: Publico

Request (schema):

```
{ "nome": "string", "sobrenome": "string", "email": "string", "password": "string", "dataNascimento": "YYYY-MM-DD", "cpf": "string(11 digitos)", "modalidadeIds": [1,2], "pagamentoFormato": "AULA|PERCENT", "pagamentoPercentual": 15.0, "enderecos": [ { ... }, { ... } ] }
```

Response (schema):

```
202 { "message": "Cadastro recebido. Aguarde aprovacao do super admin." }
```

Regras/Notas: Professor fica PENDING_SUPERADMIN_APPROVAL. Resposta neutra.

POST /api/professores/{id}/approve

Auth: JWT

Roles/Perfis: SUPER_ADMIN

Request (schema):

```
{ "approved": true, "role": "ADMIN|USER" }
```

Response (schema):

```
200 { "message": "Professor aprovado." }
```

Regras/Notas: Somente SUPER_ADMIN. Pode promover a ADMIN.

8. Endpoints - Eventos

POST /api/eventos

Auth: JWT

Roles/Perfis: PROFESSOR (escopo) | ADMIN | SUPER_ADMIN

Request (schema):

```
{ "nomeDescricao": "string", "dataInicio": "YYYY-MM-DD", "dataFim": "YYYY-MM-DD", "hora": "HH:mm:ss", "modalidadeId": 1, "alunoIds": [1,2,3] }
```

Response (schema):

```
201 { "id": 1, "status": "PENDING_VALIDATION" }
```

Regras/Notas: Professor USER: alunoIds devem pertencer ao seu escopo. Evento aguarda aprovacao SUPER_ADMIN.

POST /api/eventos/{id}/approve

Auth: JWT

Roles/Perfis: SUPER_ADMIN

Request (schema):

```
{ "approved": true }
```

Response (schema):

```
200 { "message": "Evento aprovado." }
```

Regras/Notas: Ao aprovar: status APPROVED e disparar convites por e-mail.

POST /api/eventos/{id}/confirm

Auth: JWT

Roles/Perfis: ALUNO (proprio)

Request (schema):

```
{ "confirm": "YES|NO" }
```

Response (schema):

```
200 { "message": "Confirmacao registrada." }
```

Regras/Notas: Aluno somente confirma a propria inscricao.

9. Endpoints - Pagamentos

POST /api/pagamentos/links

Auth: JWT

Roles/Perfis: PROFESSOR (escopo) | ADMIN | SUPER_ADMIN

Request (schema):

```
{ "alunoId": 1, "valor": 120.00, "dueDate": "YYYY-MM-DD", "modalidadeId": 1,
"eventoId": null }
```

Response (schema):

```
201 { "id": 10, "status": "PENDING", "dueDate": "YYYY-MM-DD", "paymentUrl":
"https://..." }
```

Regras/Notas: Somente URL necessaria ao pagamento deve ser retornada. Nunca retornar gateway_reference_id.

POST /api/pagamentos/webhook/pagseguro

Auth: Publico

Roles/Perfis: Gateway

Request (schema):

```
(payload do PagSeguro - validar assinatura)
```

Response (schema):

```
200 { "status": "ok" }
```

Regras/Notas: Endpoint publico com validacao forte. Implementar idempotencia por gateway_reference_id + tipo de evento.

10. Observabilidade e hardening

- Logs: mascarar CPF e nunca logar tokens/JWT/payment payload completo. - CORS: permitir somente dominios do frontend. - Rate limit: recomendado para endpoints publicos de token. - Scheduler: lembretes de vencimento e marcacao OVERDUE.