**Relatório de Teste da API REST de Jogadores de Futebol**

**Testador:** Viviane Barbosa  
**Ferramenta de automação:** Playwright + Node.js  
**Data:** 18/08/2025  
**Objetivo:** Validar todos os endpoints da REST API conforme os requisitos do desafio.

**Sumário Executivo**

Este relatório apresenta os resultados da automação de testes realizada na API REST de jogadores de futebol, conforme os requisitos do desafio de QA. A automação foi desenvolvida utilizando **Playwright** com **JavaScript**. A suíte de testes abrange todos os 14 requisitos funcionais da API, com foco em endpoints de listagem, busca, detalhes, top-k e montagem de times.

Os testes foram executados com sucesso, e os resultados indicam que a API está funcionando conforme o esperado em todos os cenários testados. Nenhuma falha funcional crítica foi identificada, o que sugere que a aplicação atende aos requisitos de forma satisfatória.

**Escopo e Cobertura**

Os testes automatizados foram criados para cobrir todas as 14 funcionalidades da API REST, incluindo:

* **Listagem de jogadores (com e sem paginação):** Validação da estrutura da resposta e da paginação.
* **Busca por atributos:** Testes de busca por nome, time, liga, nacionalidade e posição.
* **Detalhes do jogador:** Verificação do endpoint para obter detalhes e a imagem de um jogador específico.
* **Listagem de Top-K jogadores:** Testes para obter os melhores jogadores (por overall) em listas gerais, por posição, nacionalidade e liga.
* **Montagem do melhor time:** Testes para os endpoints que retornam o melhor time (squad) em cenários global, por liga e por nacionalidade.

Cada teste valida não apenas o status code HTTP (esperando 200 OK), mas também a estrutura da resposta (JSON válido, array de dados), o conteúdo retornado (por exemplo, se o nome do jogador corresponde à busca), e a ordem dos dados (para as listas top-K).

**Metodologia de Teste e Ferramentas**

A abordagem de teste utilizada foi a **black-box**, focando no comportamento externo da API sem acesso ao seu código-fonte. A automação foi implementada utilizando as seguintes ferramentas:

* **Playwright Test Runner:** Framework de teste para Node.js, escolhido por sua simplicidade e robustez para testes de API e e2e.
* **JavaScript:** Linguagem de programação utilizada para escrever os scripts de teste.
* **Requisições HTTP:** As requisições à API foram feitas usando o cliente request do Playwright.

O ambiente de teste configurado permitiu a execução dos testes de forma automatizada, garantindo repetibilidade e consistência nos resultados.

**Resultados dos Testes**

Todos os testes automatizados executados foram aprovados. Isso demonstra que a aplicação se comporta conforme as especificações do desafio. Os resultados detalhados por requisito são apresentados a seguir:

**✅ R1: Listar todos com paginação**

* **Resultado:** Aprovado. A API retorna uma lista de jogadores, e a paginação com os parâmetros items e page funciona corretamente, retornando a quantidade esperada de itens.

**✅ R2: Listar por nome**

* **Resultado:** Aprovado. A busca por nome (/players/search?name=Neymar) retorna jogadores que correspondem ao nome fornecido.

**✅ R3: Listar por time**

* **Resultado:** Aprovado. A busca por time (/players/search?club=Barcelona) retorna corretamente os jogadores do time especificado.

**✅ R4: Listar por liga**

* **Resultado:** Aprovado. A busca por liga retorna jogadores que fazem parte da liga informada. O teste valida que a URL foi codificada corretamente para lidar com caracteres especiais (Campeonato Brasileiro Série A).

**✅ R5: Listar por nacionalidade**

* **Resultado:** Aprovado. A busca por nacionalidade (/players/search?nationality=Brazil) retorna apenas jogadores da nacionalidade especificada.

**✅ R6: Listar por posição**

* **Resultado:** Aprovado. A busca por posição (/players/search?position=GK) retorna apenas jogadores que atuam na posição de goleiro. A validação considera posições isoladas ou entre vírgulas.

**✅ R7: Detalhes por ID + Imagem**

* **Resultado:** Aprovado. O endpoint de detalhes (/players/{id}) retorna um JSON com as informações corretas de um jogador. O endpoint de imagem (/players/{id}/img) retorna uma imagem válida com conteúdo.

**✅ R8: Top-K geral**

* **Resultado:** Aprovado. O endpoint (/players/top/10/overall) retorna os 10 melhores jogadores, e a lista está corretamente ordenada por overall em ordem decrescente.

**✅ R9: Top-K por posição**

* **Resultado:** Aprovado. A busca por top-K por posição (/players/top/5/overall?position=GK) retorna os melhores jogadores da posição especificada, ordenados corretamente.

**✅ R10: Top-K por nacionalidade**

* **Resultado:** Aprovado. O endpoint (/players/top/5/overall?nationality=Brazil) retorna os melhores jogadores brasileiros, ordenados por overall.

**✅ R11: Top-K por liga**

* **Resultado:** Aprovado. A busca por top-K por liga retorna os melhores jogadores da liga especificada, ordenados por overall.

**✅ R12: Melhor time (global)**

* **Resultado:** Aprovado. O endpoint (/team/best) retorna um time com pelo menos 11 jogadores, validando a formação de uma equipe completa.

**✅ R13: Melhor time por liga**

* **Resultado:** Aprovado. O endpoint de melhor time por liga retorna um time de pelo menos 11 jogadores que pertencem à liga especificada.

**✅ R14: Melhor time por nacionalidade**

* **Resultado:** Aprovado. O endpoint (/team/best?nationality=Brazil) retorna um time de pelo menos 11 jogadores da nacionalidade especificada.

**Análise de Bugs e Sugestões**

Nenhum bug funcional foi encontrado durante a execução dos testes automatizados. A API demonstrou ser estável e confiável em relação aos requisitos definidos.

**Bugs encontrados:** Nenhum. **Severidade:** N/A.

**Sugestões e Melhorias**

Embora a aplicação tenha passado em todos os testes, algumas sugestões de melhoria podem ser consideradas para o futuro do projeto:

* **Validação de Parâmetros:** Adicionar validações robustas para os parâmetros de entrada (ex: K negativo ou league inexistente). No momento, os testes focam em cenários de sucesso (happy path), mas cenários de erro (ex: 404 Not Found ou 400 Bad Request) para entradas inválidas seriam cruciais.
* **Documentação da API:** A ausência de uma documentação interativa (como Swagger/OpenAPI) dificultou o mapeamento inicial dos endpoints. Uma documentação clara e atualizada seria um artefato de grande valor para o desenvolvimento e QA.
* **Testes de Performance:** Implementar testes de carga e estresse para avaliar o comportamento da API sob um grande volume de requisições. Isso garantiria que a aplicação seja escalável.
* **Testes de Segurança:** Adicionar testes que verifiquem a autenticação, autorização e vulnerabilidades de segurança (ex: injeção de SQL ou sanitização de dados).

**Declaração Técnica**

Com base nos resultados dos testes automatizados, declaro que a aplicação **atende com sucesso aos requisitos** do desafio. A suíte de testes desenvolvida cobriu de forma abrangente as funcionalidades da API, e a ausência de bugs funcionais encontrados é um forte indicativo de sua qualidade. A utilização de automação garante que futuras alterações no código possam ser validadas rapidamente, prevenindo regressões.