AAUAv

universidade de aveiro

deti

Taça UA

M4: Validação de Componentes

Bernardo Borges 103592
Alexandre Regalado 124574
Mateus Rocha 122949
Diego Aguilar 117122
Gonçalo Simões 119412

Índice

Performance

Performance - Testes

Uso de K6

Testes de performance realizados usando o K6, com o *Grafana* como uma dashboard para visualização.

Testes Feitos

Latência, Débito, Peak, Smoke, Soak, Stress, Spike. Análise focada nos sublinhados

Foco no Lado Público

Público terá a maior quantidade de utilizadores concorrentes. Foco nos endpoints de torneios, seasons e equipas.

Performance - Resumo Geral

Teste	VUs	Duração	Pedidos	Débito	Taxa de Erro	Resultado
Latencia	50	3min 45s	10k	42.2 req/s	0%	PASS
Débito	até 250	11min	145k	220.59 req/s	0%	PASS
Stress	até 400	10min 30s	55k (rutura)	157.91 req/s	4,18%	FAIL

Performance - Latencia por Endpoint

Endpoint	p95	Threshold p96
GET /ranking/general	10.6 ms	< 500 ms
GET /ranking/modality	10.8 ms	< 500 ms
GET /teams	11.2 ms	< 500 ms
GET /nucleos	11.8 ms	< 300 ms
GET /matches	13.2 ms	< 500 ms
GET /tournaments	13.4 ms	< 500 ms

Performance - Otimização

- **N+1 Queries**
  Views materializadas - todos os dados já estão pre-joinados
- **Queries lentas em leitura**
  Sem joins em runtime - cada endpoint é um unico select
- **Carga repetida na DB**
  Redis cache por endpoint (TTL configurável)

Escalabilidade

Escalabilidade - Estrategia

- **CQRS**
  Leitura e escrita separadas a public-api é completamente stateless
- **Event Driven (RabbitMQ)**
  Microserviços desacoplados, cada um escalável independentemente
- **Redis Cache**
  Ativação da cache e aumento da pool
- **Nginx**
  Pronto para distribuir tráfego por múltiplas instâncias

Escalabilidade - Bottleneck

Bottleneck Gerado no Stress Test

Cache desativada em dev mode (por fins de testes)
Pool de DB saturada (pool_size e max_overflow pequenos)

A partir de 291 VUs

A pool começa a falhar - erros de timeout em cascata

Solução

Ativação da cache e aumento da pool

Escalabilidade - Vertical/Horizontal

Verticalmente	Trocar “uvicorn” (que limita o paralelismo com um worker só) por “gunicorn”, que pode iniciar com multiplos workers no Dockerfile de produção
Horizontalmente	A public-api é stateless - logo várias instâncias podem correr em paralelo sem conflitos. O nginx já distribui por round-robin automaticamente.

Segurança

Segurança - Visão Geral

Análise
Autenticação, autorização e auditoria de dependências

Ferramentas Usadas
pip-audit, npm audit e Keycloak JWT

Segurança - Autenticação e Autorização

Autenticação

Todos os endpoints de administração estão protegidos por middleware para validação de acesso

Token inválido? - HTTP 401 Unauthorized

Endpoints públicos

Endpoints públicos intencionais ignoram autenticação, mas usam uma API diferente

Segurança - Auditoria (Python)

Public API

Serviço totalmente atualizado - não tem vulnerabilidades para evidenciar

Competition API

Encontradas 18 vulnerabilidades relacionadas com a versão utilizada do Django

Solução

Atualização da versão do Django

Segurança - Auditoria (Frontend)

Painel Administrativo

Uma vulnerabilidade detetada, relacionada com o PostCSS (Potencial de XSS)

Website publico

Encontradas 10 vulnerabilidades (4 moderadas e 6 altas), que afetam apenas o servidor de dev

Solução

Utilização do comando ‘npm audit fix’ - corrige automaticamente estes problemas

Validação

Validação - Invariantes

Validação de Invariantes de Negócio

- O sistema rejeita dados inválidos antes de qualquer escrita na base de dados

Invariante	Teste	Resultado
Campos obrigatorios em falta	POST /tournaments/ sem modality_id	HTTP 400
Tipo de competidor fora do dominio	competitor_type: “invalid”	HTTP 400
Torneio já terminado	POST /finished	HTTP 400

Validação - Concorrência

- Não há locks no código, nem proteção adicional implementada.
- Em caso de duas escritas simultâneas no mesmo torneio, a ultima a fazer commit vence - comportamento padrão do PostgreSQL

Validação - Inconsistência Transacional

- **Outbox**
  O padrão Outbox garante consistência entre a escrita na DB e a publicação do evento
- **Em caso de erro**
  Se houver algum erro, é feito o rollback
- **E se o Rabbit estiver em baixo?**
  O evento fica na tabela outbox e é reenviado quando recuperar.

Gestão de projeto

Gestão de Projeto - Metodologia

- Metodologia ágil com Scrum adaptado (Uso do JIRA)
- **6 sprints** totais ao longo do ano
- Repositório no GitHub com PRs como unidade de trabalho:
  119 PRs no momento
- CI/CD com GitHub Actions (lint, tests e build)

Gestão de Projeto - Sprints

Sprint	Tarefas	Concluidas	Taxa
1	18	18	100%
2	55	50	91%
3	27	20	74%
4	58	44	76%
5	29	24	83%
6	em curso	-	-

Gestão de Projeto - CI/CD

CI/CD

- Uso de GitHub Actions para criar uma pipeline automática para todos os PRs

Pre-Commit

- Uso de Pre-Commit Hooks para formatação e linting, corridos antes de cada commit.

Docker Compose

- Uso de docker compose para configuração do ambiente de desenvolvimento e de produção

Testes de Desempenho

- Uso de testes para controlo de performance

Links importantes

Microsite

Organização

Jira

Obrigado pela atenção