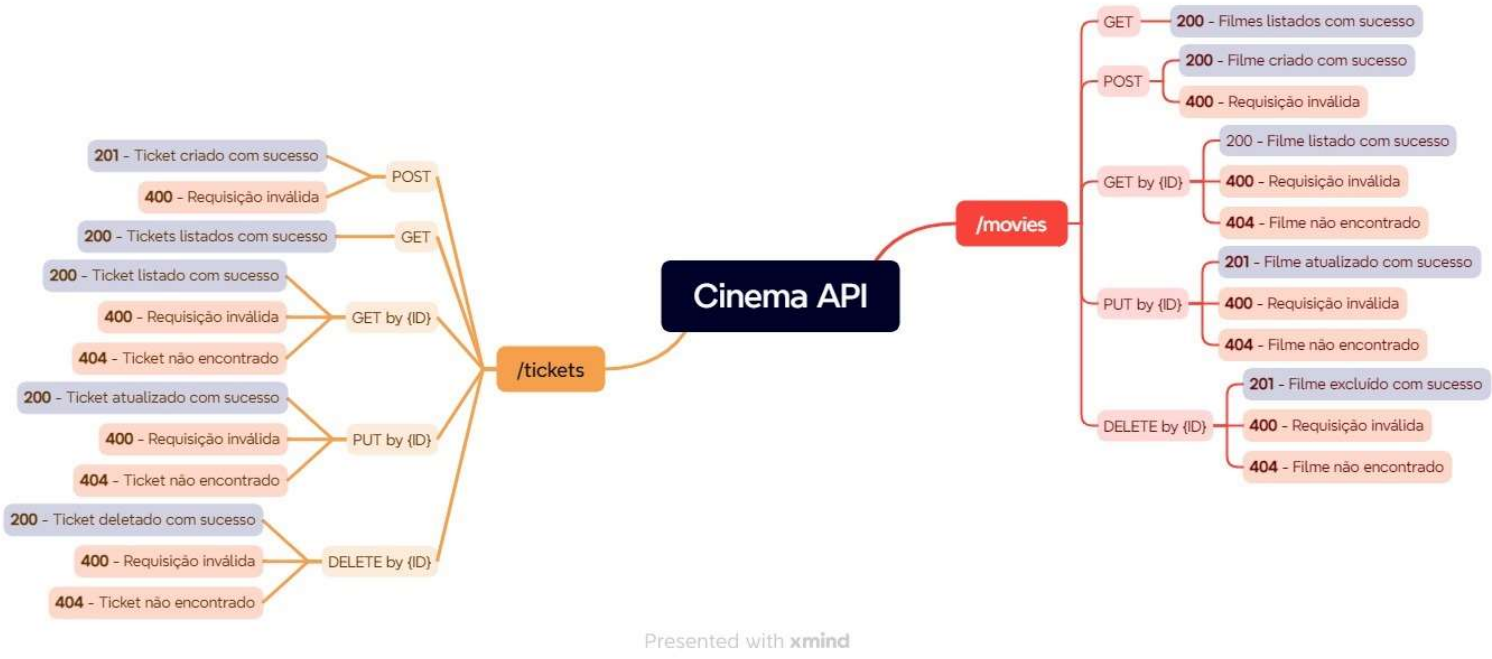
Planejamento de testes para API Cinema by juniorschmitz on GitHub
Projeto do Challenge Final da PBCompassUOL

1. Introdução

Este documento apresenta o planejamento das atividades de testes do sistema e descreve o plano de ação para implementar e testar o CRUD da API Cinema, deseja-se verificar se a usabilidade da interface da API, se cumpre todos os requisitos funcionais e não funcionais, garantindo a qualidade geral do uso da API e identificar possíveis defeitos. Será utilizado o framework K6 em JavaScript para executar os testes e posteriormente feita sua avaliação. O objetivo desse plano é avaliar problemas na documentação nos requisitos funcionais e avaliar o desempenho da API em diferentes cenários de uso sob os requisitos não funcionais. Será testado as principais rotas, garantir que a aplicação atenda aos requisitos de escalabilidade e tempo de resposta.

1.1 Mapa mental da API com seus respectivos 'status code' disponíveis até a versão a ser testada



2. Escopo para testes de performance

Estão englobados no escopo a serem testados da rota '/movies' os verbos GET e POST e da rota '/tickets' o verbo POST, compostos de testes de carga, estresse, pico, resistência e fumaça para cada aplicação.

Funcionalidades e endpoints

- **Movies** – Cria, lista todos os filmes
  - GET /movies - Lista todos os filmes em exibição do cinema
  - POST /movies – Cadastra um novo filme para sessão do cinema

- **Tickets** – Cria, lista todos os tickets
  - POST /tickets – Cadastra ingressos para uma sessão disponível de cinema.

2.1 Fora do escopo

- **Movies -** Com o método {id}, é possível listar, atualizar e deletar um filme da sessão de cinema.
  - GET /movies/{id}
  - PUT /movies/{id}
  - DELETE /movies/{id}

- **Tickets -** com o método {id} é possível listar, atualizar e deletar um ticket para a sessão de cinema.
  - GET /tickets – Lista todos os tickets cadastrados
  - GET /tickets/{id} - Broken
  - PUT /tickets/{id} - Broken
  - DELETE /tickets/{id} - Broken

3. Requisitos do negócio

User Story: Gerenciamento de Filmes na API

Ator: Qualquer usuário da API de Filmes (para consulta) e Usuário administrador da API de Filmes (para criação, atualização e exclusão)

Requisitos Funcionais:

1. Criando um Novo Filme:

- O usuário administrador da API envia uma solicitação POST para o endpoint /movies com os detalhes do filme.
- O sistema valida os campos obrigatórios e a unicidade do título.
- Se as validações passarem, o sistema cria o filme e atribui um ID único.
- O sistema retorna uma resposta de sucesso com o status 201 Created, incluindo o ID do filme.

2. Obtendo a Lista de Filmes:

- O usuário envia uma solicitação GET para o endpoint /movies.
- O sistema retorna uma lista de todos os filmes cadastrados com detalhes.

3. Obtendo Detalhes de um Filme por ID:

- O usuário envia uma solicitação GET para o endpoint /movies/{id}, onde {id} é o ID do filme desejado.
- O sistema verifica a existência do filme e retorna seus detalhes.
- Se o filme não existir, o sistema retorna uma resposta de erro com o status 404 Not Found.

4. Atualizando os Detalhes de um Filme por ID:

- O usuário administrador da API envia uma solicitação PUT para o endpoint /movies/{id}, onde {id} é o ID do filme a ser atualizado.
- O sistema verifica a existência do filme, permite a atualização de campos específicos e valida os dados.
- Se todas as validações passarem, o sistema atualiza os detalhes do filme.
- O sistema retorna uma resposta de sucesso com o status 200 OK e os detalhes atualizados do filme.

5. Excluindo um Filme por ID:

- O usuário administrador da API envia uma solicitação DELETE para o endpoint /movies/{id}, onde {id} é o ID do filme a ser excluído.
- O sistema verifica a existência do filme e o remove permanentemente do banco de dados.
- O sistema retorna uma resposta de sucesso com o status 204 No Content.

Requisitos Não Funcionais de Performance:

- A API deve ser capaz de processar pelo menos 100 solicitações de criação de filmes por segundo.
- O tempo médio de resposta para a criação de um novo filme não deve exceder 200 milissegundos.
- A API deve ser capaz de responder a solicitações GET de listagem de filmes em menos de 100 milissegundos.
- A lista de filmes deve ser paginada, com no máximo 20 filmes por página.
- A API deve ser capaz de responder a solicitações GET de detalhes de um filme em menos de 50 milissegundos.
- A API deve ser capaz de processar pelo menos 50 solicitações de atualização de filmes por segundo.
- O tempo médio de resposta para a atualização dos detalhes de um filme não deve exceder 300 milissegundos.
- A API deve ser capaz de processar pelo menos 30 solicitações de exclusão de filmes por segundo.
- O tempo médio de resposta para a exclusão de um filme não deve exceder 400 milissegundos.

DoR

- Análise de testes cobrindo a rota de movies;
- Matriz de rastreabilidade atualizada;
- Automação de testes baseado na análise realizada;

DoD

- Banco de dados e infraestrutura para desenvolvimento disponibilizados;
- API de cadastro e listagem de filmes implementada;
- Ambiente de testes disponibilizado.

User Story: Reservando Ingressos na API

Ator: Qualquer usuário da API de Ingressos

Cenário: O usuário deseja reservar ingressos para assistir a um filme em um cinema.

Requisitos Funcionais:

- O usuário envia uma solicitação POST para o endpoint /tickets com os seguintes detalhes do ingresso:

- ID do Filme (movieId) - Identifica o filme para o qual o ingresso está sendo reservado.

- ID do Usuário (userId) - Identifica o usuário que está fazendo a reserva.

- Número do Assento (seatNumber) - O número do assento que o usuário deseja reservar.

- Preço do Ingresso (price) - O preço do ingresso para o filme.

- Data de Apresentação (showtime) - A data e hora da apresentação do filme.

- O sistema valida se todos os campos obrigatórios estão preenchidos corretamente.
- O sistema verifica se o número do assento está dentro do intervalo de 0 a 99.
- O sistema verifica se o preço do ingresso está dentro do intervalo de 0 a 60.
- Se todas as validações passarem, o sistema cria uma reserva de ingresso com os detalhes fornecidos.
- O sistema atribui um ID único à reserva de ingresso.
- O sistema retorna uma resposta de sucesso com o status 201 Created, incluindo o ID da reserva de ingresso.

Requisitos Não Funcionais de Performance:

- A API deve ser capaz de processar pelo menos 50 solicitações de reserva de ingressos por segundo.
- O tempo médio de resposta para a reserva de um ingresso não deve exceder 300 milissegundos.

DoR

- Análise de testes cobrindo a rota de tickets;
- Matriz de rastreabilidade atualizada;
- Automação de testes baseado na análise realizada;

DoD

- Banco de dados e infraestrutura para desenvolvimento disponibilizados;
- API de cadastro e listagem de tickets implementada;
- Ambiente de testes disponibilizado.

4. Casos de Testes Funcionais

Casos de testes da rota /movies criada com base na user stories fornecida:

CT01 – Usuário administrador faz POST/movies

CT02 – Validar se campos obrigatórios foram preenchidos corretamente

CT03 - Validar unicidade de título do filme

CT04 - Atribuição de ID único para um filme cadastrado

CT05 - Retornar status code 201 – created incluindo o ID do filme

CT06 - Usuário default faz GET/movies na API

CT07 - Retornar uma lista de todos os filmes cadastrados com detalhes.

CT08 - O sistema valida se todos os campos obrigatórios estão preenchidos corretamente.

CT09 - O sistema verifica se o número do assento está dentro do intervalo de 0 a 99.

CT10 - O sistema verifica se o preço do ingresso está dentro do intervalo de 0 a 60.

CT11 - Se todas as validações passarem, o sistema cria uma reserva de ingresso com os detalhes fornecidos.

CT12 - O sistema atribui um ID único à reserva de ingresso.

CT13 - O sistema retorna uma resposta de sucesso com o status 201 Created, incluindo o ID da reserva de ingresso.

5. Matriz de rastreabilidade de testes funcionais

Numeração dos casos	Cenários	Casos de Testes	Status do Teste
CT01	Criando filme	Usuário admin faz POST/movies	**REPROVADO**

CT02		Validar se campos obrigatórios foram preenchidos corretamente	**REPROVADO**
CT03		Validar unicidade de título do filme	**REPROVADO**
CT04		Atribuição de ID único para um filme cadastrado	**APROVADO**
CT05		Retornar status code 201 – created incluindo o ID do filme	**REPROVADO**
CT06	Listando filmes	Usuário default faz GET/movies na API	**APROVADO**
CT07		Retornar uma lista de todos os filmes cadastrados com detalhes.	**APROVADO**
CT08	Criando ticket	O sistema valida se todos os campos obrigatórios estão preenchidos corretamente.	**REPROVADO**
CT09		O sistema verifica se o número do assento está dentro do intervalo de 0 a 99.	**REPROVADO**
CT10		O sistema verifica se o preço do ingresso está dentro do intervalo de 0 a 60.	**REPROVADO**
CT11		Se todas as validações passarem, o sistema cria uma reserva de ingresso com os detalhes fornecidos.	**APROVADO**
CT12		O sistema atribui um ID único à reserva de ingresso.	**APROVADO**
CT13		O sistema retorna uma resposta de sucesso com o status 201 Created, incluindo o ID da reserva de ingresso.	**APROVADO**

6. Plano para testes não funcionais (Performance)

Objetivo

Simulação de carga gradativa buscando alcançar o momento de degradações, garantindo que as rotas da API /movies, /tickets atendam aos requisitos de performance a partir dos cenários apresentados. O objetivo deste plano é avaliar o desempenho da API Cinema em diferentes cenários de uso. Vamos testar as principais rotas, garantir que a aplicação atenda aos requisitos de escalabilidade e tempo de resposta.

Requisitos Não Funcionais de Performance:

NF01 - A API deve ser capaz de processar pelo menos 100 solicitações de criação de filmes por segundo.

NF02 - O tempo médio de resposta para a criação de um novo filme não deve exceder 200 milissegundos.

NF03 - A API deve ser capaz de responder a solicitações GET de listagem de filmes em menos de 100 milissegundos.

NF04 - A lista de filmes deve ser paginada, com no máximo 20 filmes por página.

NF05 - A API deve ser capaz de processar pelo menos 50 solicitações de reserva de ingressos por segundo.

NF06 - O tempo médio de resposta para a reserva de um ingresso não deve exceder 300 milissegundos.

6.1 Cenários de Testes para Performance

Cenário 1: Teste de Fumaça da rota GET/movies

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições 'http' para a aplicação do verbo 'get' para listar todos os filmes disponíveis abertos para sessões.

Usuários Virtuais: 3 (smoke test)

Duração do Teste: 15 segundos por carga (smoke test)

Principais métricas: Taxa de duração de resposta da requisição e taxa de erro.

Cenário 2: Teste de Carga da rota GET/movies

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’get’ para listar todos os filmes disponíveis abertos para sessões.

Usuários Virtuais: 100

Duração do Teste: 10 minutos por carga.

Principais métricas: Taxa de duração de resposta da requisição (<100ms conforme o requisito não funcional **[NF03]**) e taxa de erro.

Cenário 3: Teste de Estresse da rota GET/movies

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’get’ para listar todos os filmes disponíveis abertos para sessões.

Usuários Virtuais: 100

Duração do Teste: 30 minutos por carga.

Principais métricas: Taxa de duração de resposta da requisição (<100ms conforme o requisito não funcional **[NF03]**) e taxa de erro.

Cenário 4: **Teste de Pico da rota GET/movies**

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’get’ para listar todos os filmes disponíveis abertos para sessões.

Usuários Virtuais: 2000

Duração do Teste: 2 minutos por carga.

Principais métricas: Taxa de duração de resposta da requisição (<100ms conforme o requisito não funcional **[NF03]**) e taxa de erro.

Cenário 5: Teste de Resistência da rota GET/movies

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’get’ para listar todos os filmes disponíveis abertos para sessões.

Usuários Virtuais: 100

Duração do Teste: 45 minutos por carga.

Principais métricas: Taxa de duração de resposta da requisição (<100ms conforme o requisito não funcional **[NF03]**) e taxa de erro.

Cenário 6: Teste de Fumaça da rota POST/movies

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’post’ para para cadastrar filmes a serem abertos para sessões.

Usuários Virtuais: 3 (smoke test)

Duração do Teste: 15 segundos por carga (smoke test)

Principais métricas: Taxa de duração de resposta da requisição e taxa de erro.

Cenário 7: Teste de Carga da rota POST/movies

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’post’ para cadastrar filmes a serem abertos para sessões.

Usuários Virtuais: 100 conforme o Requisito não funcional **[NF01]**

Duração do Teste: 10 minutos por carga.

Principais métricas: Taxa de duração de resposta da requisição (<200ms conforme o requisito não funcional **[NF02]**) e taxa de erro.

Cenário 8: Teste de Estresse da rota POST/movies

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’post’ para cadastrar filmes a serem abertos para sessões.

Usuários Virtuais: 100 conforme o Requisito não funcional **[NF01]**

Duração do Teste: 30 minutos por carga.

Principais métricas: Taxa de duração de resposta da requisição (<200ms conforme o requisito não funcional **[NF02]**) e taxa de erro.

Cenário 9: Teste de Pico da rota POST/movies

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’post’ para cadastrar filmes a serem abertos para sessões.

Usuários Virtuais: 2000

Duração do Teste: 2 minutos por carga.

Principais métricas: Taxa de duração de resposta da requisição (<200ms conforme o requisito não funcional **[NF02]**) e taxa de erro.

Cenário 10: Teste de Resistência da rota POST/movies

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’post’ para cadastrar filmes a serem abertos para sessões.

Usuários Virtuais: 100 conforme o Requisito não funcional **[NF01]**

Duração do Teste: 45 minutos por carga.

Principais métricas: Taxa de duração de resposta da requisição (<200ms conforme o requisito não funcional **[NF02]**) e taxa de erro.

Cenário 11: Teste de Fumaça da rota POST/tickets

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’post’ para cadastrar tickets para as sessões de filmes disponíveis.

Usuários Virtuais: 3 (smoke test)

Duração do Teste: 15 seg por carga (smoke test)

Principais métricas: Taxa de duração de resposta da requisição e taxa de erro.

Cenário 12: Teste de Carga da rota POST/tickets

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’post’ para cadastrar tickets para as sessões de filmes disponíveis.

Usuários Virtuais: 100 conforme o Requisito não funcional **[NF05]**

Duração do Teste: 10 minutos por carga.

Principais métricas: Taxa de duração de resposta da requisição (<300ms conforme o requisito não funcional **[NF02]**) e taxa de erro.

Cenário 13: Teste de Estresse da rota POST/tickets

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’post’ para cadastrar tickets para as sessões de filmes disponíveis.

Usuários Virtuais: 100 conforme o Requisito não funcional **[NF05]**

Duração do Teste: 30 minutos por carga.

Principais métricas: Taxa de duração de resposta da requisição (<300ms conforme o requisito não funcional **[NF06]**) e taxa de erro.

Cenário 14: Teste de Pico da rota POST/tickets

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’post’ para cadastrar tickets para as sessões de filmes disponíveis.

Usuários Virtuais: 2000

Duração do Teste: 2 minutos por carga.

Principais métricas: Taxa de duração de resposta da requisição (<300ms conforme o requisito não funcional **[NF02]**) e taxa de erro.

Cenário 15: Teste de Resistência da rota POST/tickets

Objetivo: Avaliar a resposta da API com um número crescente de usuários simultâneos fazendo requisições ‘http’ para a aplicação do verbo ’post’ para cadastrar tickets para as sessões de filmes disponíveis.

Usuários Virtuais: 100 conforme o Requisito não funcional **[NF05]**

Duração do Teste: 45 minutos por carga.

Principais métricas: Taxa de duração de resposta da requisição (<300ms conforme o requisito não funcional **[NF02]**) e taxa de erro.

7. Objetivo dos Testes para garantir a qualidade

Defina os objetivos gerais do projeto de teste:

- Garantir que a API em teste esteja em conformidade com os requisitos funcionais e não funcionais.
- Identificar e corrigir defeitos antes da implantação.
- Avaliar a qualidade geral do software.

7.1 Funções e Responsabilidades

Analista de QA: Responsáveis por criar casos de teste, executar testes e relatar defeitos.

Mentores: Supervisionam o processo de teste e garantem que os objetivos sejam atingidos.

Desenvolvedores: Colaboram na criação de testes automatizados e corrigem defeitos.

7.2 Níveis de Teste

Tipos de teste a serem realizados:

Testes de Aceitação: Verificação com base nos requisitos do cliente.

Testes de Desempenho para verificação da performance da API.

7.3 Triagem de Bugs

Os critérios para estabelecer os bugs é não responderem ao requisitos mínimos estabelecidos para garantir uma boa funcionalidade da API. Bugs serão relatados no relatório de execução dos testes e identificação de defeitos encontrados e serão disponibilizados do Jira Qality e no repositório do projeto.

7.4 Conclusão do Teste

Os testes serão considerados concluídos após todas as rotas da API com pelo menos um verbo de cada rota comtemplando 5 tipos de testes sejam avaliados e também todos os bugs relatados encontrados. Isso pode incluir critérios específicos de aceitação.

7.5 Resultados de Teste

Registre os resultados dos testes realizados, incluindo casos de teste, ambientes de teste e avaliação de riscos estarão disponibilizados no Jira, e também no repositório do projeto do GitLab em PDF com o bug report no Qality Jira.

8. Necessidades de Recursos e Ambiente

Ferramentas utilizadas: VSCode, Linguagem JavaScript e Framework K6, Pacote Node.js instalado para trazer a API localmente no desktop local do analista para realização da automatização dos testes.

Descreva o ambiente para executar os testes foi o computador pessoal do testador com as seguintes especificações:

- Windows 10 Home
- 16GB RAM DDR3
- Intel Core i7-3770
- SSD SATAII 480GB, HDD 2TB

8.1 Ambiente de Teste

O ambiente de teste foi realizado no computador pessoal do colaborador, utilizando da API localmente disponibilizada com node.js através da porta localhost:3000.

9. Riscos

Os testes deverão ser executados o quanto antes para evitar imprevistos, como ficar sem fornecimento de energia elétrica perto do deadline. Caso a conexão com a internet seja interrompida, pode-se ativar os dados móveis do celular para a execução da tarefa.

10. Execução e Análise dos Resultados

A execução dos testes e os resultados serão analisados e disponibilizados na plataforma Jira Atlassian. Qualquer desvio dos critérios estabelecidos será investigado. Com

base nos resultados, será feita recomendações e reports de bugs para otimizar o desempenho da API.

11. Cronograma

Realização do planejamento de teste: 24/09/2024-02/10/2024

Tabulação de dados: 02/10/2024-03/10/2024

Entrega dos resultados: 04/10/2024