Este projeto visa criar testes unitários para a aplicação de listagem de Pokémons, garantindo que as funcionalidades estejam corretas e os dados da PokéAPI sejam integrados corretamente.

Equipe de Desenvolvimento: Estevão Lucas, Marcos André e Vinicius F. Lima

Plano de Testes

Pokédex

Versão *1.0.0*

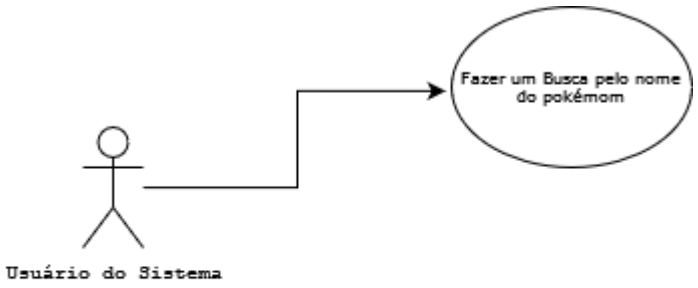
Conteúdo

1 Introdução

O fluxo de testes, assim como os demais fluxos, está presente no processo de desenvolvimento de *software* ao longo de todas as suas fases, concentrando-se, no entanto, no planejamento dos testes na iteração inicial e no início de cada nova iteração e, durante as iterações, tendo seu foco no projeto e na execução dos testes, sobretudo nas iterações da fase de Construção.

Este documento descreve os requisitos a testar, os tipos de testes definidos para cada iteração, os recursos de hardware e software a serem empregados e o cronograma dos testes ao longo do projeto. As seções referentes aos requisitos, recursos e cronograma servem para permitir ao gerente do projeto acompanhar a evolução dos testes.

2 Requisitos a Testar



Caso de Uso 1: Buscar Pokémon pelo Nome (UC1)

Cenário Principal:

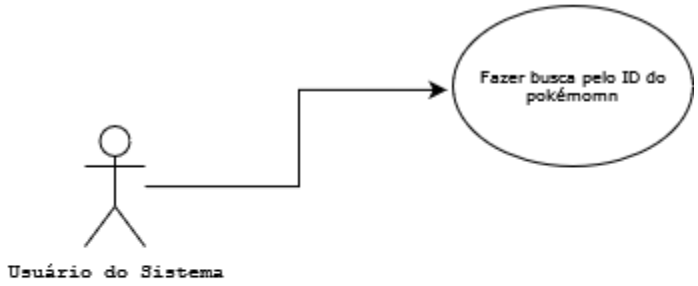
1. O usuário digita o nome do Pokémon.
2. O sistema busca o Pokémon pelo nome.
3. O sistema exibe as informações do Pokémon encontrado.

Cenários Secundários:

- Nome do Pokémon não encontrado:
    1. O sistema informa ao usuário que o Pokémon não foi encontrado.

Cenário de Teste:

- Teste 1: Buscar Pokémon existente pelo nome
    - Entrada: Nome de um Pokémon existente
    - Saída esperada: Informações do Pokémon encontrado são exibidas
- Teste 2: Buscar Pokémon inexistente pelo nome
    - Entrada: Nome de um Pokémon inexistente
    - Saída esperada: Mensagem indicando que o Pokémon não foi encontrado



Caso de Uso 2: Buscar Pokémon pelo ID (UC2)

Cenário Principal:

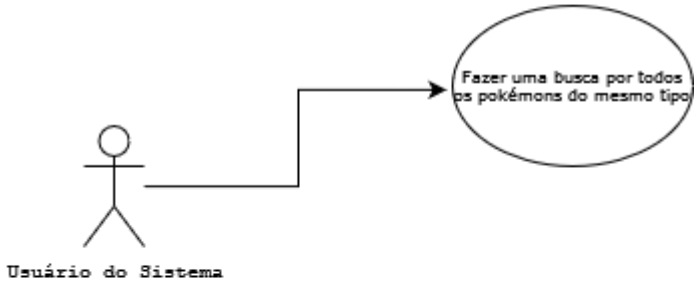
1. O usuário digita o ID do Pokémon.
2. O sistema busca o Pokémon pelo ID.
3. O sistema exibe as informações do Pokémon encontrado.

Cenários Secundários:

- ID do Pokémon não encontrado:
    1. O sistema informa ao usuário que o Pokémon não foi encontrado.

Cenário de Teste:

- Teste 1: Buscar Pokémon existente pelo ID
    - Entrada: ID de um Pokémon existente
    - Saída esperada: Informações do Pokémon encontrado são exibidas
- Teste 2: Buscar Pokémon inexistente pelo ID
    - Entrada: ID de um Pokémon inexistente
    - Saída esperada: Mensagem indicando que o Pokémon não foi encontrado



Caso de Uso 3: Filtrar Pokémon pelo Tipo (UC3)

Cenário Principal:

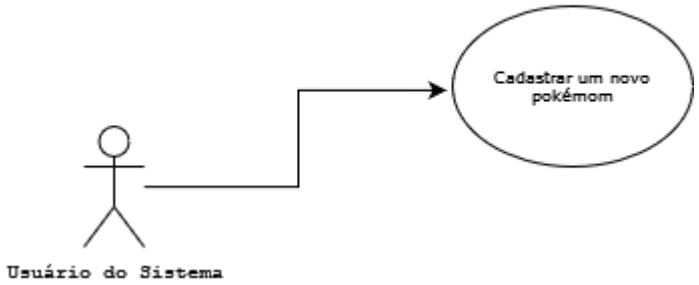
1. O usuário seleciona um tipo de Pokémon.
2. O sistema busca os Pokémon que possuem o tipo selecionado.
3. O sistema exibe uma lista de Pokémon encontrados.

Cenários Secundários:

- Nenhum Pokémon encontrado com o tipo selecionado:
    1. O sistema informa ao usuário que não foram encontrados Pokémon com o tipo selecionado.

Cenários de Testes:

- Teste 1: Filtrar Pokémons por tipo
    - Entrada: Tipo de Pokémon
    - Saída esperada: Lista de Pokémon que possuem o tipo filtrado é exibida
- Teste 2: Recebe Pokémon daquele tipo
    - Entrada: Pokémon daquele Tipo
    - Saída esperada: O pokémon estar contido na lista filtrada pelo tipo



Caso de Uso 4: Cadastrar Novo Pokémon (UC4)

Cenário Principal:

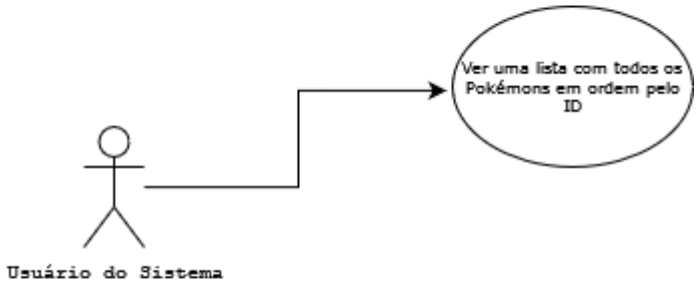
1. O usuário fornece os dados do novo Pokémon, como nome, tipo, habilidades, etc.
2. O sistema valida os dados fornecidos.
3. O sistema cadastra o novo Pokémon no banco de dados.

Cenários Secundários:

- Dados do novo Pokémon inválidos:
    1. O sistema informa ao usuário que os dados fornecidos são inválidos e solicita que sejam corrigidos.

Cenário de Teste:

- Teste 1: Cadastrar novo Pokémon com dados válidos
    - Entrada: Dados válidos de um novo Pokémon
    - Saída esperada: Pokémon é cadastrado com sucesso no sistema
- Teste 2: Cadastrar novo Pokémon com dados inválidos
    - Entrada: Dados inválidos de um novo Pokémon
    - Saída esperada: Mensagem de erro indicando que os dados são inválidos



Caso de Uso 5: Ver Lista de Pokémon em Ordem do ID (UC 5)

Cenário Principal:

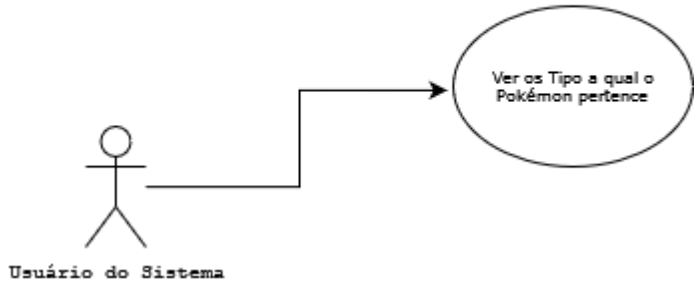
1. O usuário solicita a visualização da lista de Pokémon em ordem do ID.
2. O sistema busca os Pokémon no banco de dados.
3. O sistema ordena a lista de Pokémon pelo ID.
4. O sistema exibe a lista de Pokémon em ordem do ID.

Cenários Secundários:

- Nenhum Pokémon encontrado no banco de dados:
    1. O sistema informa ao usuário que não há Pokémon cadastrados.

Cenário de Teste:

- Teste 1: Verificar lista de Pokémon em ordem crescente do ID
    - Entrada: -
    - Saída esperada: Lista de Pokémon é exibida em ordem crescente do ID
- Teste 2: Verificar lista de Pokémon tem a quantidade especificada
    - Entrada: Quantidade de pokémons desejada
    - Saída esperada: Lista de Pokémon com a quantidade desejada.



Caso de Uso 6: Ver Tipos a que o Pokémon Pertence (UC6)

Cenário Principal:

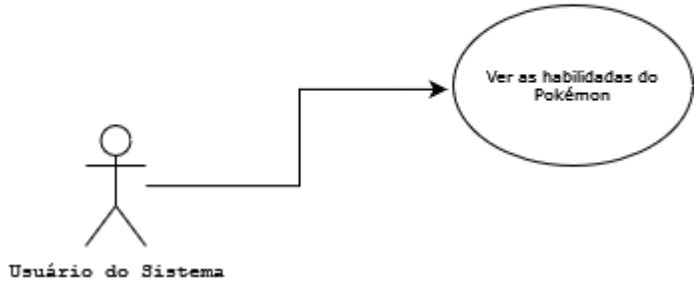
1. O usuário seleciona um Pokémon.
2. O sistema busca os tipos associados ao Pokémon selecionado.
3. O sistema exibe os tipos a que o Pokémon pertence.

Cenários Secundários:

- Nenhum tipo encontrado para o Pokémon selecionado:
    1. O sistema informa ao usuário que não foram encontrados tipos para o Pokémon selecionado.

Cenário de Teste:

- Teste 1: Verificar tipos de um Pokémon existente
    - Entrada: Pokémon existente
    - Saída esperada: Tipos associados ao Pokémon são exibidos
- Teste 2: Verificar tipos de um Pokémon inexistente
    - Entrada: Pokémon inexistente
    - Saída esperada: Mensagem indicando que o Pokémon não foi encontrado



Caso de Uso 7: Ver as Habilidades do Pokémon (UC7)

Cenário Principal:

1. O usuário seleciona um Pokémon.
2. O sistema busca as habilidades associadas ao Pokémon selecionado.
3. O sistema exibe as habilidades do Pokémon.

Cenários Secundários:

- Nenhuma habilidade encontrada para o Pokémon selecionado:
    1. O sistema informa ao usuário que não foram encontradas habilidades para o Pokémon selecionado.

Cenário de Teste:

- Teste 1: Verificar habilidades de um Pokémon existente
    - Entrada: Pokémon existente
    - Saída esperada: Habilidades associadas ao Pokémon são exibidas
- Teste 2: Verificar habilidades de um Pokémon inexistente
    - Entrada: Pokémon inexistente
    - Saída esperada: Mensagem indicando que o Pokémon não foi encontrado

Requisito Não Funcional 1: CRUD no Banco de Dados (Rq1)

Cenário Principal:

1. O sistema recebe os dados do novo Pokémon a ser cadastrado.
2. O sistema realiza a validação dos dados.
3. O sistema salva o Pokémon no banco de dados.

Cenário de Teste:

- Teste 1: Verificar se o Pokémon é salvo corretamente no banco de dados
  - Entrada: Dados válidos de um novo Pokémon
  - Saída esperada: Pokémon é salvo com sucesso no banco de dados
- Teste 2: Verificar se o Pokémon é deletado corretamente no banco de dados
  - Entrada: Dados válidos do a ser Pokémon
  - Saída esperada: Pokémon é deletado com sucesso no banco de dados
- Teste 3: Verificar se o Pokémon é atualizado corretamente no banco de dados
  - Entrada: Dados válidos do Pokémon
  - Saída esperada: Pokémon é atualizado com sucesso no banco de dados
- Teste 4: Verificar se o Pokémon é encontrado no banco de dados
  - Entrada: Dados válidos de busca de um Pokémon
  - Saída esperada: Pokémon é retornado com sucesso no banco de dados

Requisito Não Funcional 2: Buscar Pokémon da API (Rq2)

Cenário Principal:

1. O sistema recebe uma solicitação de busca de Pokémon.
2. O sistema realiza a busca do Pokémon na API externa.
3. O sistema retorna as informações do Pokémon encontrado.

Cenário de Teste:

- Teste 1: Verificar se as informações do Pokémon são buscadas corretamente na API
  - Entrada: ID ou nome de um Pokémon existente
  - Saída esperada: Informações do Pokémon são obtidas corretamente da API

Requisito Não Funcional 3: Se Pokémon no Banco de Dados não Buscar na API (Rq3)

Cenário Principal:

1. O sistema recebe uma solicitação de busca de Pokémon.
2. O sistema verifica se o Pokémon está presente no banco de dados.
3. Se o Pokémon estiver no banco de dados, o sistema retorna as informações do Pokémon encontrado no banco de dados.
4. Se o Pokémon não estiver no banco de dados, o sistema realiza a busca na API externa e retorna as informações do Pokémon encontrado na API.

Cenário de Teste:

- Teste 1: Verificar se o Pokémon é encontrado corretamente no banco de dados
    - Entrada: ID ou nome de um Pokémon existente no banco de dados
    - Saída esperada: Informações do Pokémon são obtidas diretamente do banco de dados

Requisito Não Funcional 4: Cadastrar Novos Pokémon com uma Faixa de ID Maior que da API (Rq4)

Cenário Principal:

1. O usuário fornece os dados do novo Pokémon, incluindo um ID maior que os IDs disponíveis na API.
2. O sistema valida os dados fornecidos.
3. O sistema cadastra o novo Pokémon no banco de dados com o ID fornecido.

Cenário de Teste:

- Teste 1: Verificar se é possível cadastrar um novo Pokémon com um ID maior que os IDs disponíveis na API
    - Entrada: Dados válidos de um novo Pokémon com ID maior que da API
    - Saída esperada: Pokémon é cadastrado com sucesso no banco de dados com o ID fornecido

Esta seção contém os requisitos que são objetos dos testes a serem realizados. Esses requisitos são divididos, por iteração, em casos de uso e requisitos não funcionais conforme descrito abaixo.

2.1 Iteração 1

Casos de Uso

Identificador do Caso de Uso	Nome do Caso de Uso
UC1	Buscar Pokémon pelo Nome
UC2	Buscar Pokémon pelo ID
UC3	Filtrar Pokémon pelo Tipo

2.2 Iteração 2

Casos de Uso

Identificador do Caso de Uso	Nome do Caso de Uso
UC4	Cadastrar Novo Pokémon
UC5	Ver Lista de Pokémon em Ordem do ID

2.3 Iteração 3

Casos de Uso

Identificador do Caso de Uso	Nome do Caso de Uso
UC6	Ver Tipos a que o Pokémon Pertence
UC7	Ver as Habilidades do Pokémon

2.4 Iteração 4

Requisitos Não-Funcionais

Identificador do Requisito	Nome do Requisito
Req 1	Crud no Banco de Dados
Req 2	Buscar Pokémon da API
Req 3	Se Pokémon no Banco de Dados não Buscar na API
Req 4	Cadastrar Novos Pokémon com uma Faixa de ID Maior que da API (Faixa = 100.000)

3 Tipos de Teste

3.1 Iteração 1

Objetivo:	*Nesta iteração serão testados os principais casos de uso do sistema onde.*
Técnica:	☐ Manual ☑ ~~Automática~~
Estágio do teste: ☐ Integração ☐ Sistema ☑ ~~Unidade~~ ☐ Aceitação	Abordagem do teste ☐ Caixa branca ☑ ~~Caixa preta~~
Responsável(is):	*Vinicius F. Lima*

3.2 Iteração 2

Objetivo:	*Nesta iteração serão testados os mais casos de uso. Dessa vez fazendo validação de entradas do usuário e verificação de ordem na lista de pokémons.*
Técnica:	☐ Manual ☑ ~~Automática~~
Estágio do teste: ☑ ~~Integração~~ ☐ Sistema ☑ ~~Unidade~~ ☐ Aceitação	Abordagem do teste ☑ ~~Caixa branca~~ ☑ ~~Caixa preta~~
Responsável(is):	*Marcos André*

3.3 Iteração 3

Objetivo:	*Nesta iteração serão testados as informações sobre os tipos e habilidades de um Pokémon, permitindo que os usuários obtenham detalhes mais específicos sobre as características dos Pokémon*
Técnica:	☐ Manual ☑ ~~Automática~~
Estágio do teste: ☐ Integração ☐ Sistema ☑ ~~Unidade~~ ☐ Aceitação	Abordagem do teste ☑ ~~Caixa branca~~ ☑ ~~Caixa preta~~
Responsável(is):	*Estevão Lucas*

3.4 Iteração 4

Objetivo:	*Nesta iteração serão testados os requisitos não funcionais do sistema onde o principal objetivo é verificar a robustez da integração dos componentes do sistema.*
Técnica:	☐ Manual ☑ ~~Automática~~
Estágio do teste: ☑ ~~Integração~~ ☑ ~~Sistema~~ ☑ ~~Unidade~~ ☐ Aceitação	Abordagem do teste ☑ ~~Caixa branca~~ ☑ ~~Caixa preta~~
Responsável(is):	*Vinicius F. Lima*

4 Recursos

De extrema importância para o bom andamento dos testes, os recursos a serem utilizados durante os testes são descritos nessa seção. Os recursos estão divididos nas subseções que se seguem.

4.1 Ambiente de Teste – Software & Hardware

Visual Studio 2022,
Windows 11,
11th Gen Intel(R) Core(TM) i5-1135G7, 16 gigas de Ram.

4.2 Ferramentas de Teste

Foi utilizado o framework xUnit, que é uma biblioteca de testes unitários para o .NET, para a manipulação dos testes. Além disso, o Excel foi utilizado para criar o gráfico de controle de qualidade.

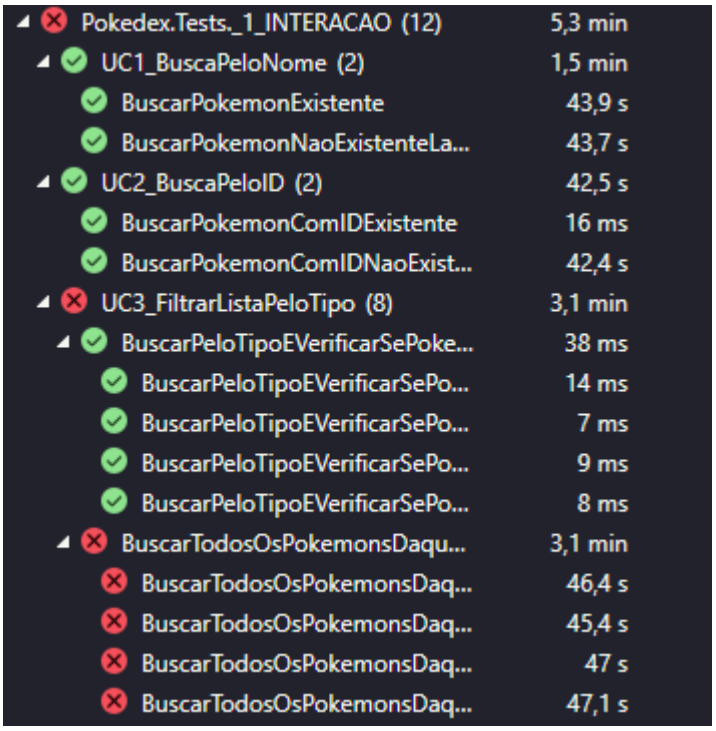
O xUnit é um framework de testes unitários gratuito e de código aberto, que permite escrever e executar testes automatizados para garantir a qualidade do código. Ele é amplamente utilizado na comunidade de desenvolvimento de software para realizar testes em projetos .NET. Com o xUnit, é possível escrever testes independentes, flexíveis e fáceis de manter, proporcionando uma abordagem mais eficiente para testes automatizados.

A documentação oficial do xUnit pode ser encontrada no seguinte link: https://xunit.net/docs/

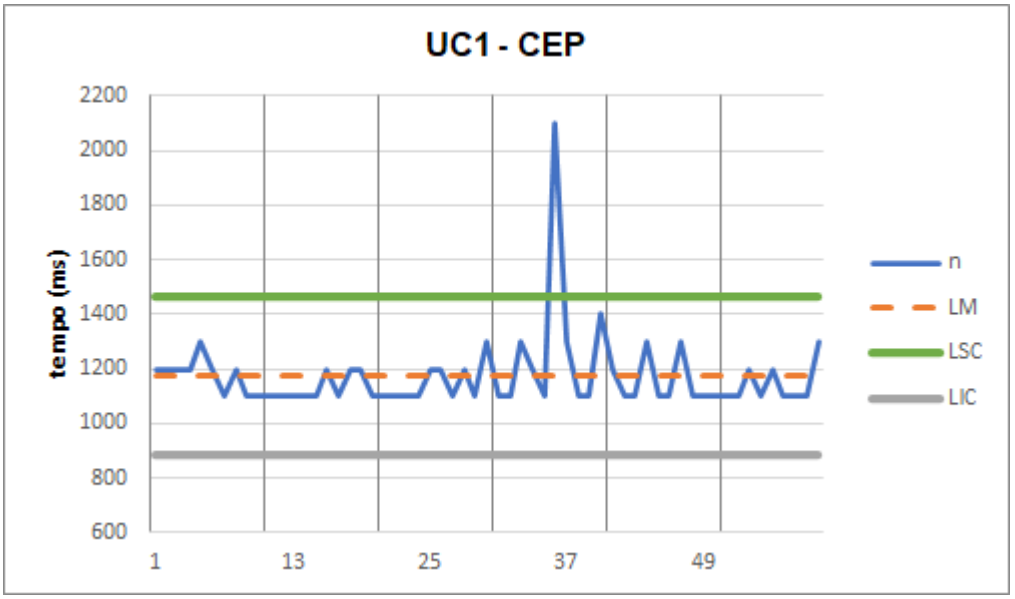
5 Referências

MARTINEZ-ARELLANO, G. et al. A data analytics model for improving process control in flexible manufacturing cells. Decision Analytics Journal, v. 3, p. 100075, 2022. Disponível em: https://www.sciencedirect.com/science/article/pii/S2772662222000285. Acesso em: 06 de junho de 2023.

6 Anexos



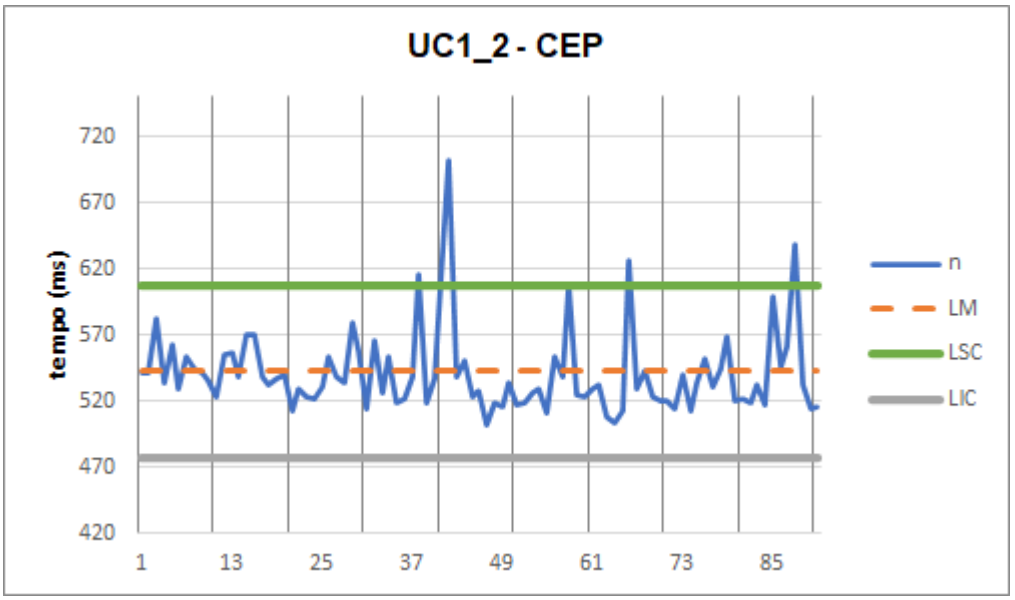
(Imagem: 1 Interação)



Pokedex.Tests._1_INTERACAO.UC1_BuscaPeloNome.BuscarPokemonExistente

(Imagem: 1 Interação, UC1)

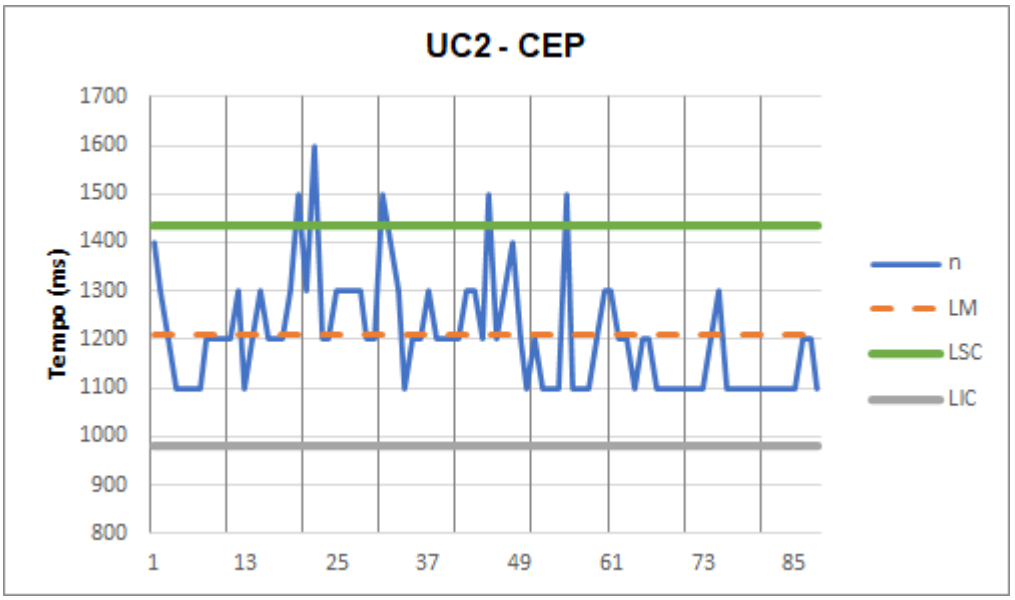
Esse caso de uso é responsável por testar o cenário em que o usuário procura um Pokémon existente pelo nome e obteve uma qualidade de 98,31% de 59 iterações.



Pokedex.Tests._1_INTERACAO.UC1_BuscaPeloNome.BuscarPokemonNaoExistenteLanceNotFoundException

(Imagem: 1 Interação, UC1_2)

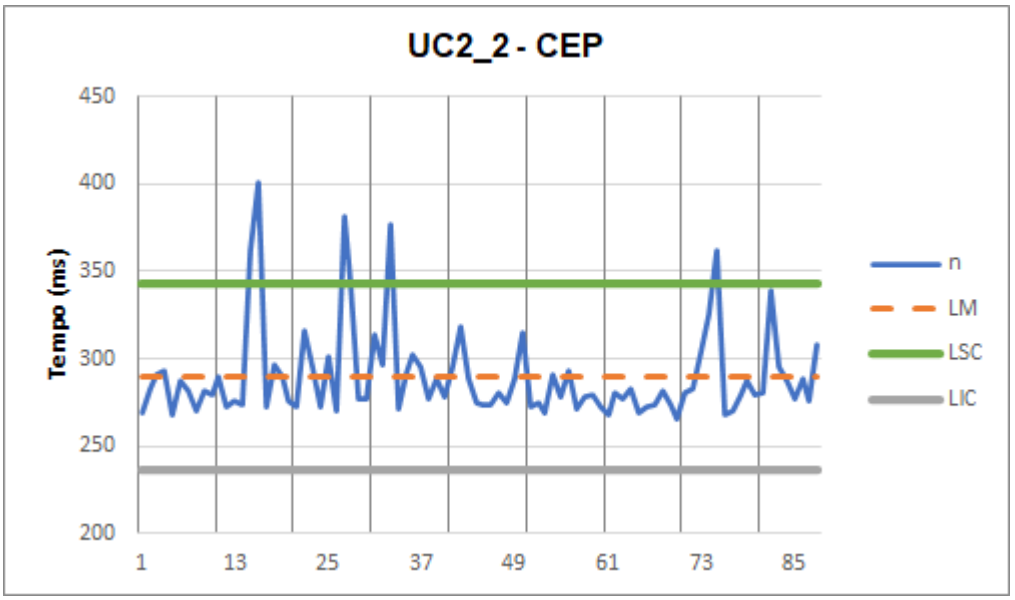
Ao testar a busca por um Pokémon inexistente, os testes obtiveram 94,51% de qualidade em 59 iterações nesse caso de uso.



Pokedex.Tests._1_INTERACAO.UC2_BuscaPeloID.BuscarPokemonComIDExistente

(Imagem: 1 Interação, UC2)

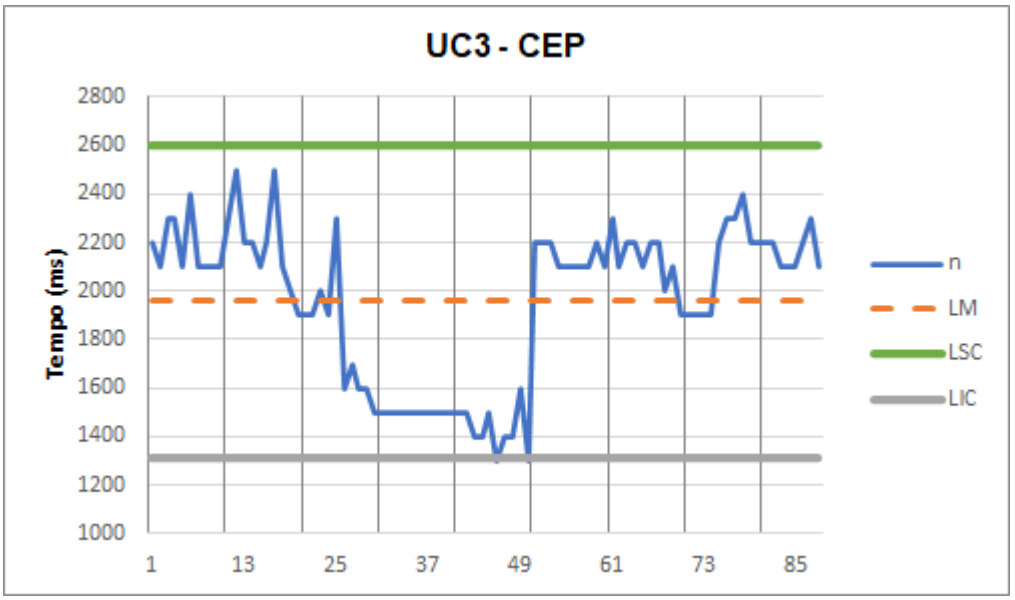
Ao buscar um Pokémon existente pelo ID foi obtido 95.15% de qualidade em 103 iterações.



Pokedex.Tests._1_INTERACAO.UC2_BuscaPeloID.BuscarPokemonComIDNaoExistenteLancePokemonNotFoundException

(Imagem: 1 Interação, UC2_2)

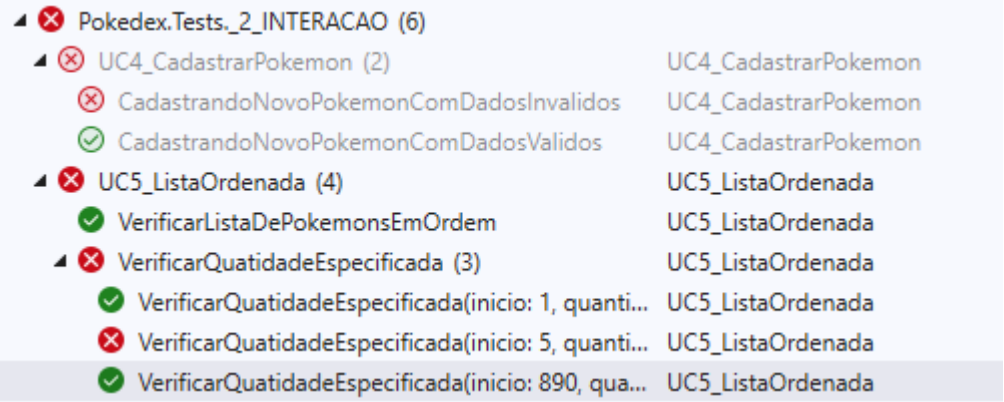
No entanto, ao buscar um Pokémon inexistente pelo ID a qualidade dos testes foi ligeiramente maior, atingindo 97.09% de qualidade em 103 iterações.



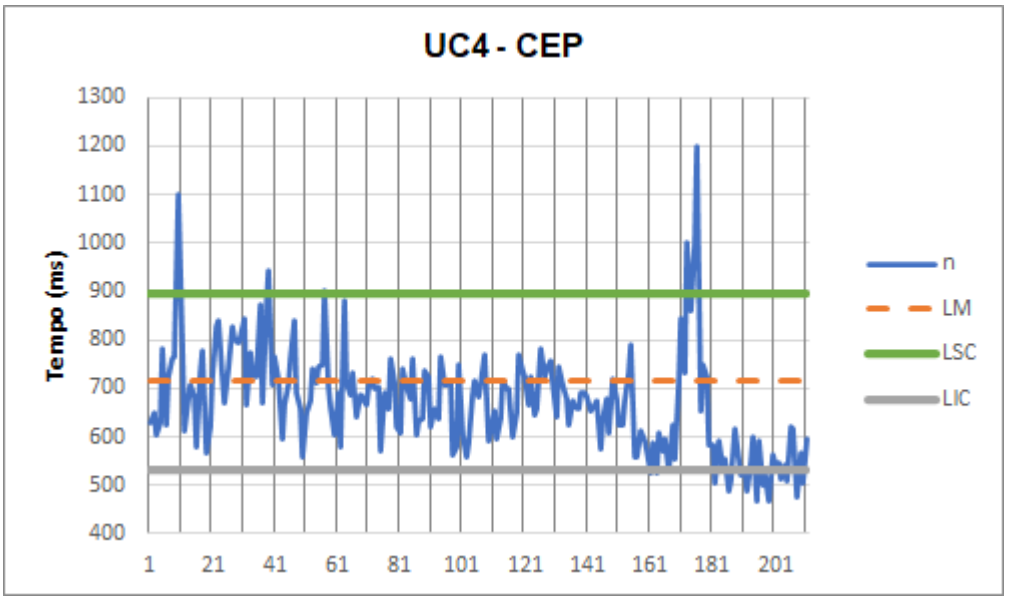
Pokedex.Tests._1_INTERACAO.UC3_FiltrarListaPeloTipo.BuscarPeloTipoEVerificarSePokemonComumAqueleTipoEstarPresente

(Imagem: 1 Interação, UC3)

Ao filtrar o pokémon pelo tipo o teste obteve 100% de qualidade em 99 iterações, o que indica que o sistema atendeu totalmente ao requisito.



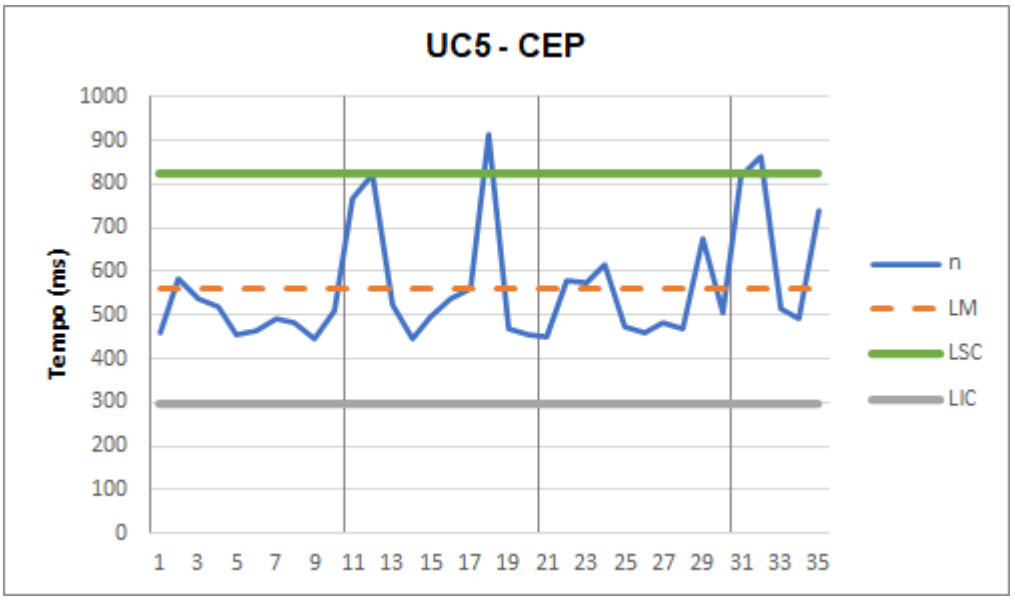
(Imagem: 2 Interação)



Pokedex.Tests._2_INTERACAO.UC4_CadastrarPokemon.CadastrandoNovoPokemonComDadosValidos

(Imagem: 2 Interação, UC4)

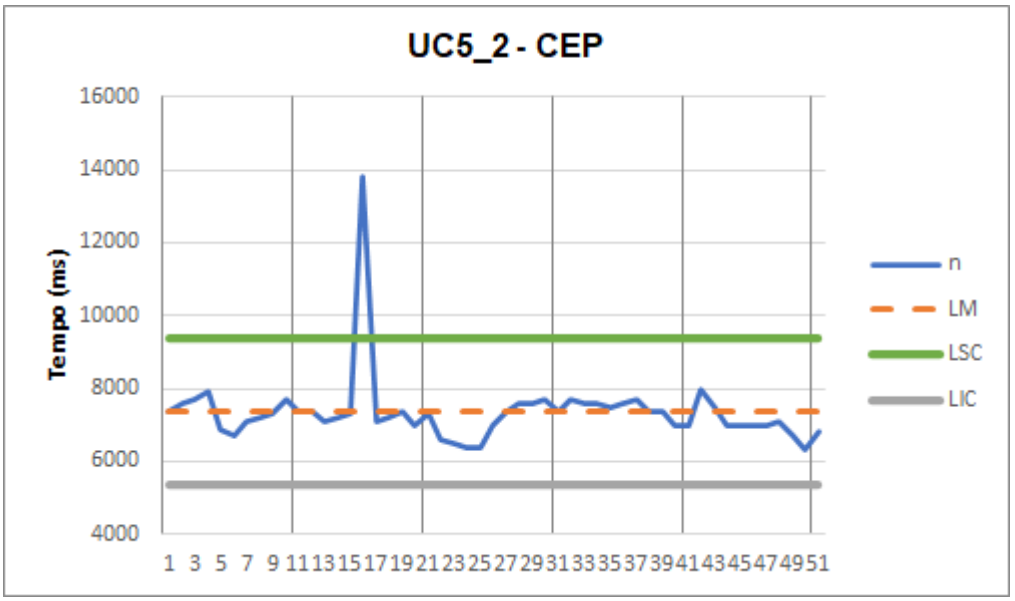
Em 212 iterações o teste de para cadastrar novos pokémons com dados válidos o teste apenas ultrapassou os limites em 4 iterações alcançando uma qualidade de 93.18%



Pokedex.Tests._2_INTERACAO.UC5_ListaOrdenada.VerificarQuatidadeEspecificada

(Imagem: 2 Interação, UC5)

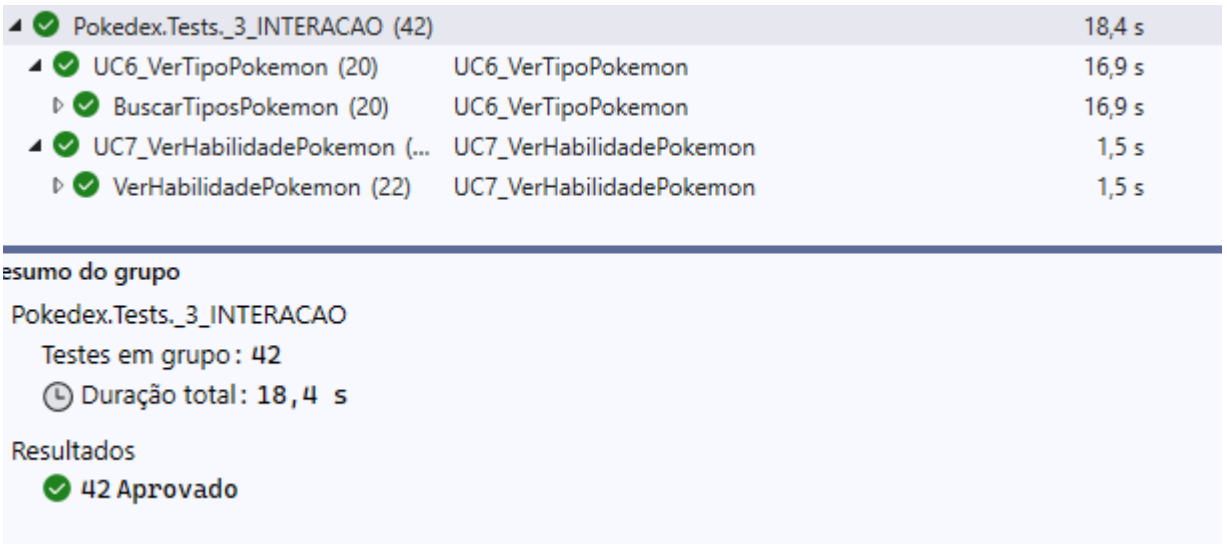
No cadastro de pokémons o teste teve uma qualidade de 94.29% não respeitando os limites em 2 das 35 iterações.



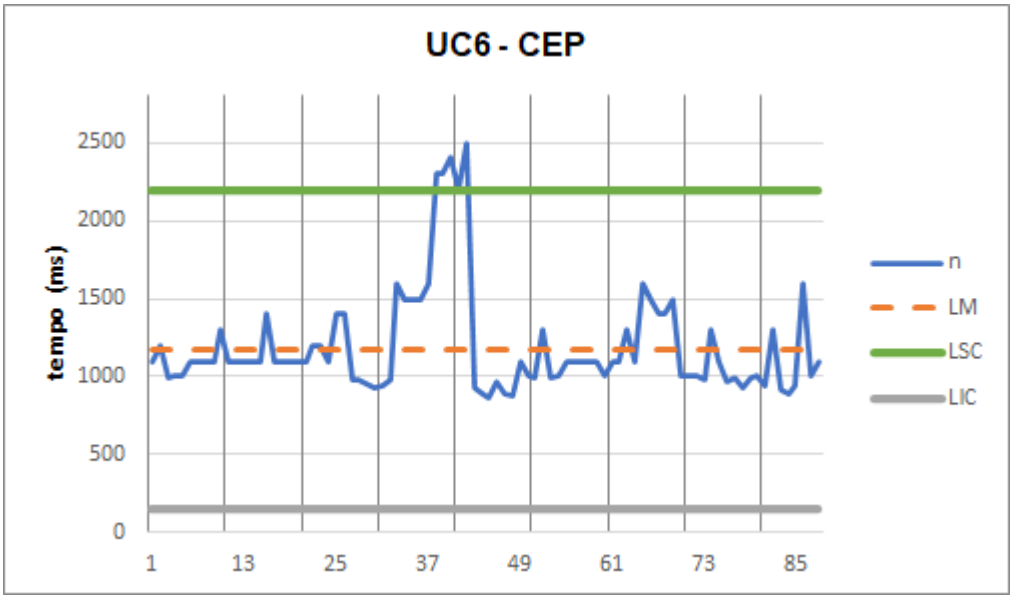
Pokedex.Tests._2_INTERACAO.UC5_ListaOrdenada.VerificarListaDePokemonsEmOrdem

(Imagem: 2 Interação, UC5_2)

Para ver a lista de pokémons ordenada por ID, o teste teve uma qualidade de 94.29% não respeitando os limites em 2 das 35 iterações.



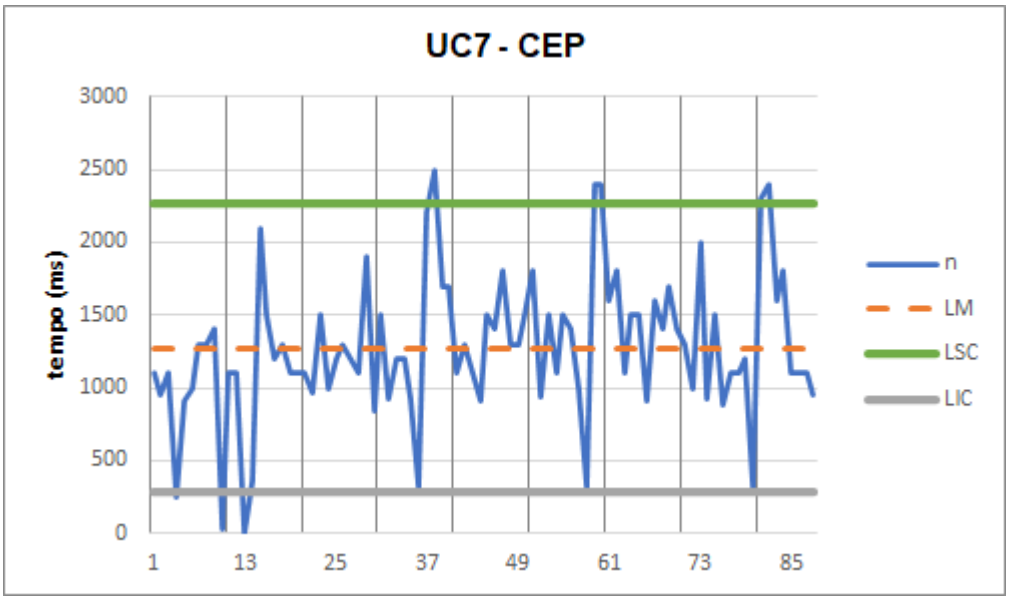
(Imagem: 3 Interação)



Buscar os tipos do pokémon de acordo com o nome dele

(Imagem: 3 Interação, UC6)

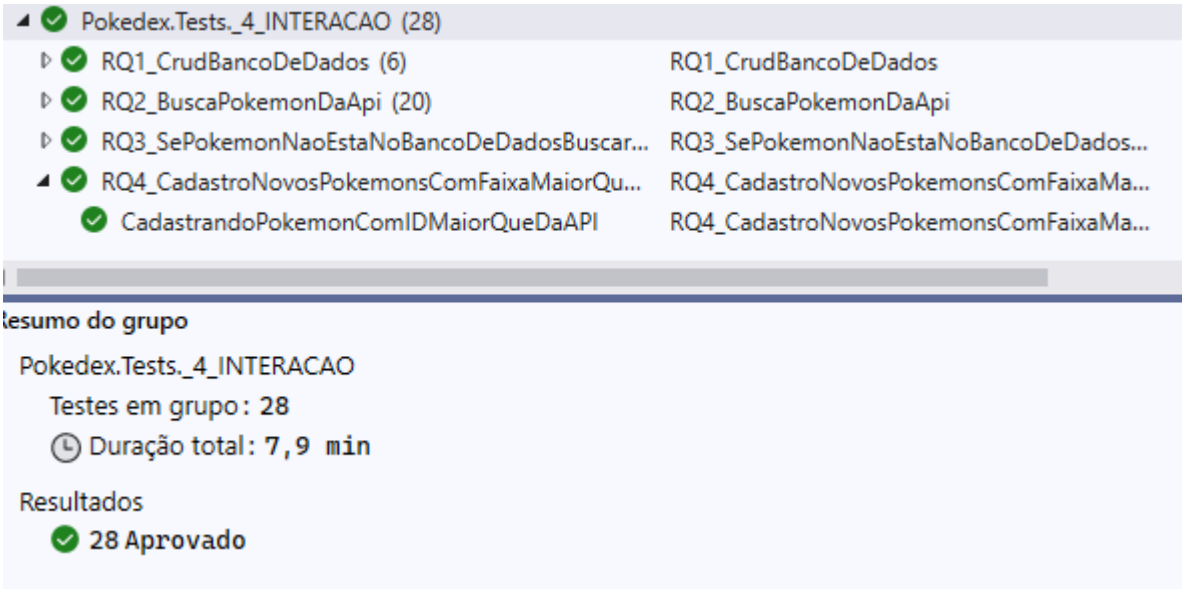
Para verificar os tipos de um Pokémon existente o teste atingiu 96.26% de qualidade em 107 iterações.



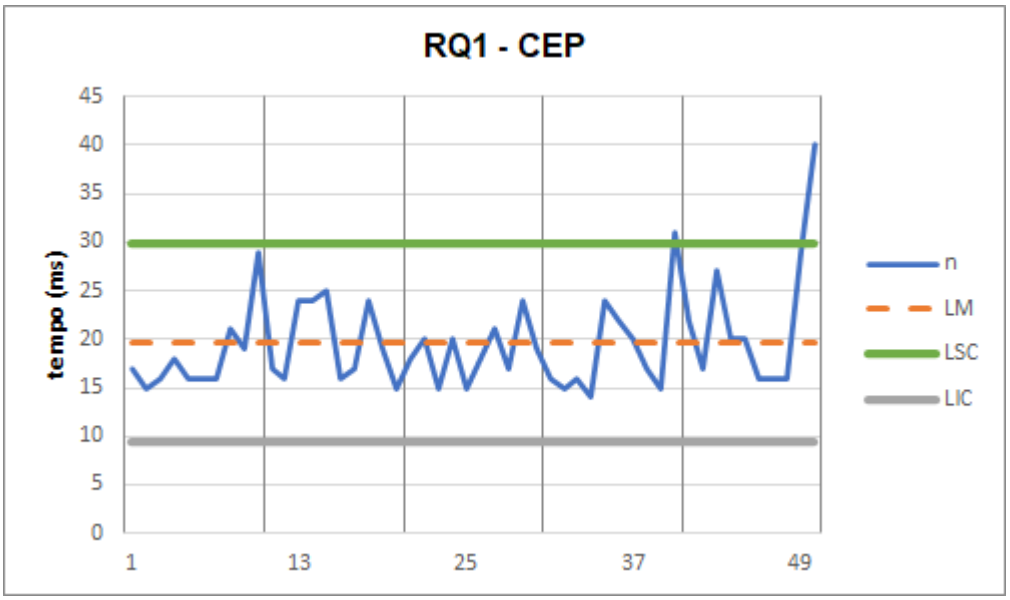
Ver habilidades do pokémon pelo nome fornecido

(Imagem: 3 Interação, UC7)

Nesse caso de uso a qualidade foi de 93.18%, três iterações ficaram abaixo do limite inferior de controle e outras três acima do limite superior de controle.



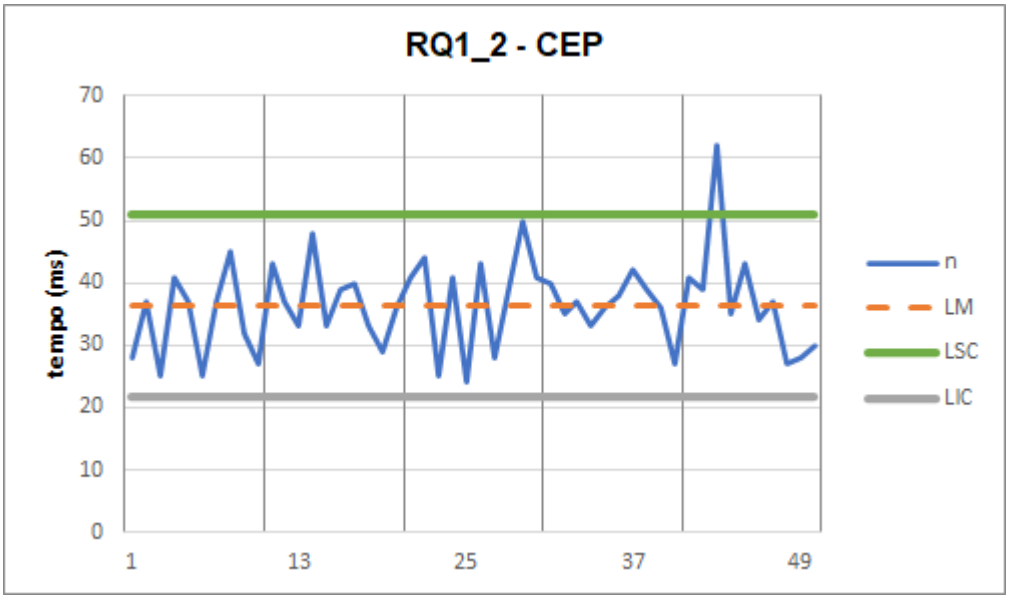
(Imagem: 4 Interação)



Pokedex.Tests._4_INTERACAO.RQ1_CrudBancoDeDados.Create

(Imagem: 4 Interação, RQ1)

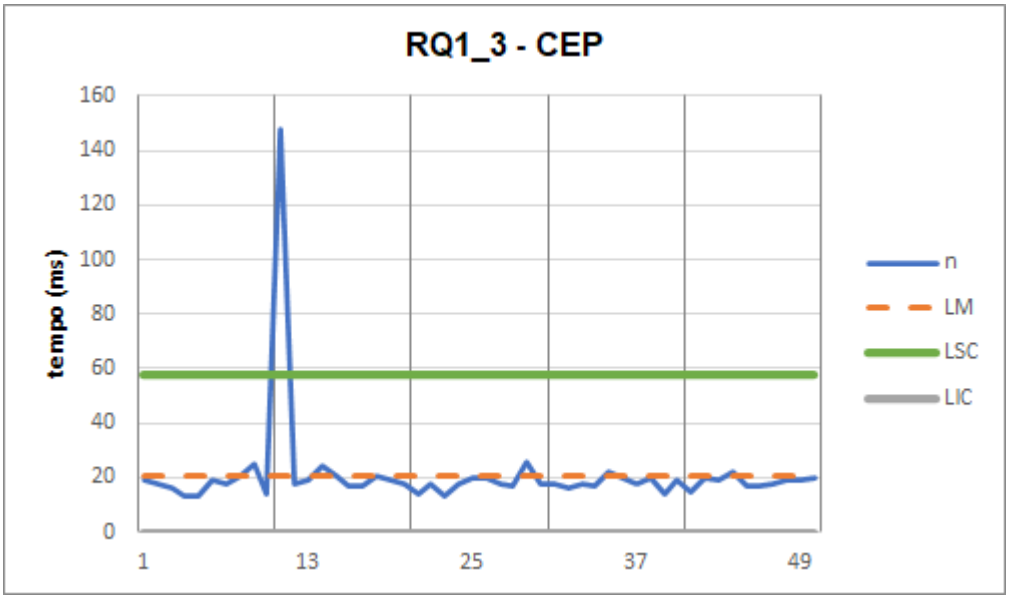
Esse requisito obteve 96% de qualidade em suas 50 iterações, com duas delas superando o limite superior de controle.



(Imagem: 4 Interação, RQ1_2)

Pokedex.Tests._4_INTERACAO.RQ1_CrudBancoDeDados.Delete

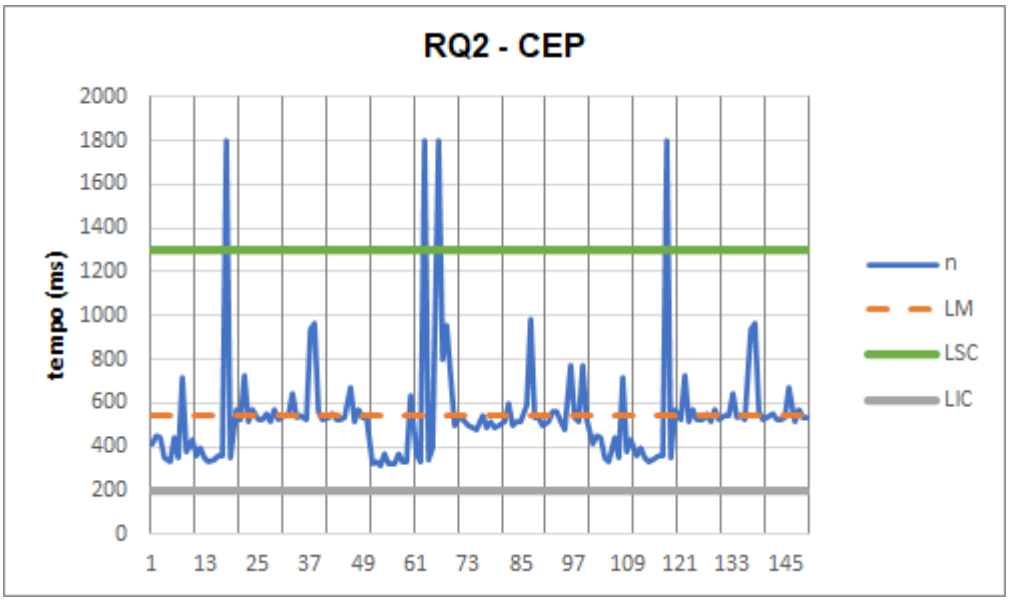
Este requisito obteve 98% de qualidade em 50 iterações. Isso indica um desempenho satisfatório na maioria das iterações, com apenas uma iteração que não atendeu completamente ao requisito.



(Imagem: 4 Interação, RQ1_3)

Pokedex.Tests._4_INTERACAO.RQ1_CrudBancoDeDados.Read

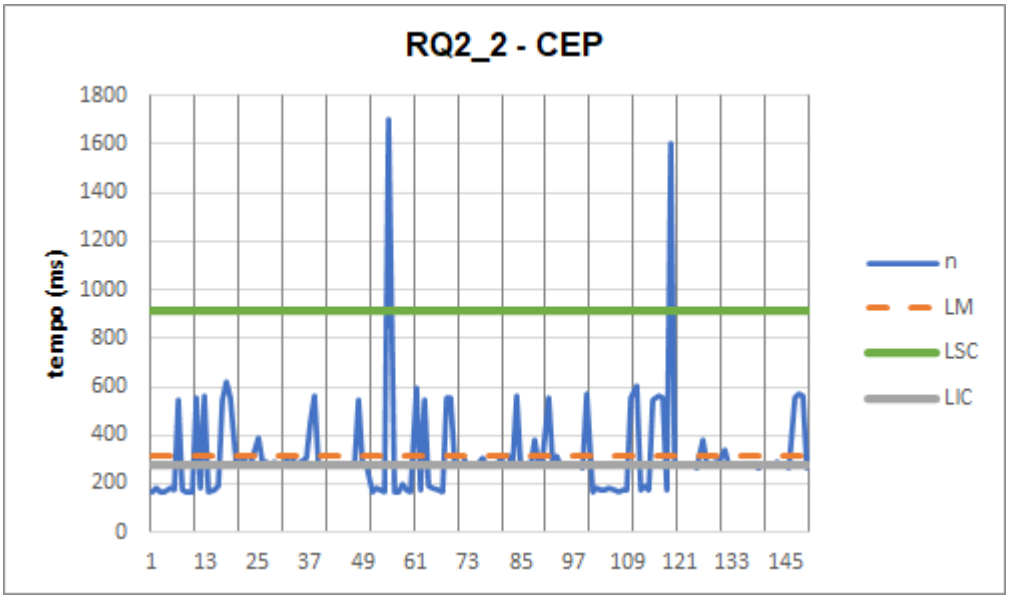
Esse requisito teve 96% de qualidade em 50 iterações. A maioria das iterações atingiu o resultado esperado, mas duas iterações não foram totalmente bem-sucedidas, ultrapassando o limite superior de controle.



Pokedex.Tests._4_INTERACAO.RQ2_BuscaPokemonDaApi.BuscaPorID

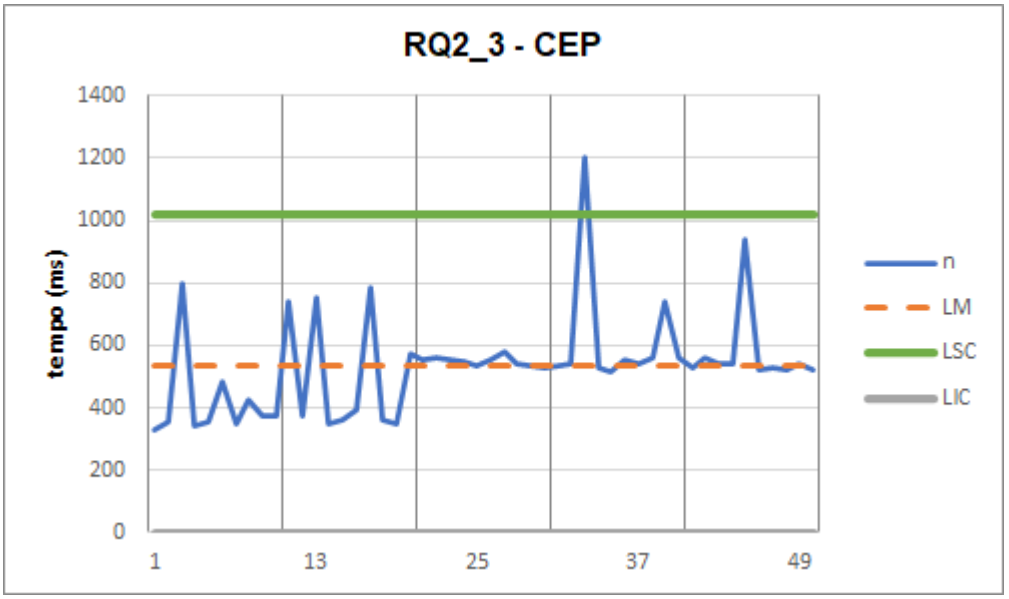
(Imagem: 4 Interação, RQ2)

Esse requisito alcançou 97.99% de qualidade em 149 iterações. A maioria das iterações foi bem-sucedida, mas algumas não conseguiram atender ao requisito ultrapassando o limite superior de controle.



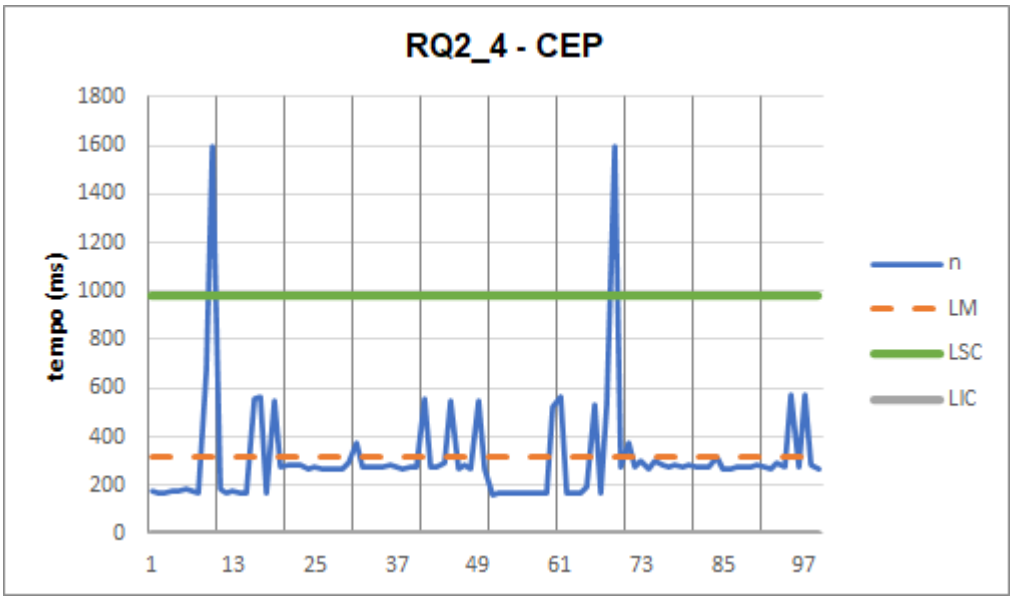
Pokedex.Tests._4_INTERACAO.RQ2_BuscaPokemonDaApi.BuscaPorIDComEntradaInvalida
(Imagem: 4 Interação, RQ2_2)

Nesse requisito, houve um desempenho inferior, com 79.19% de qualidade em 149 iterações. Isso indica que uma parte significativa das iterações não atendeu completamente ao requisito ficando 29 delas abaixo do limite inferior de controle e 2 das iterações acima do limite superior de controle.



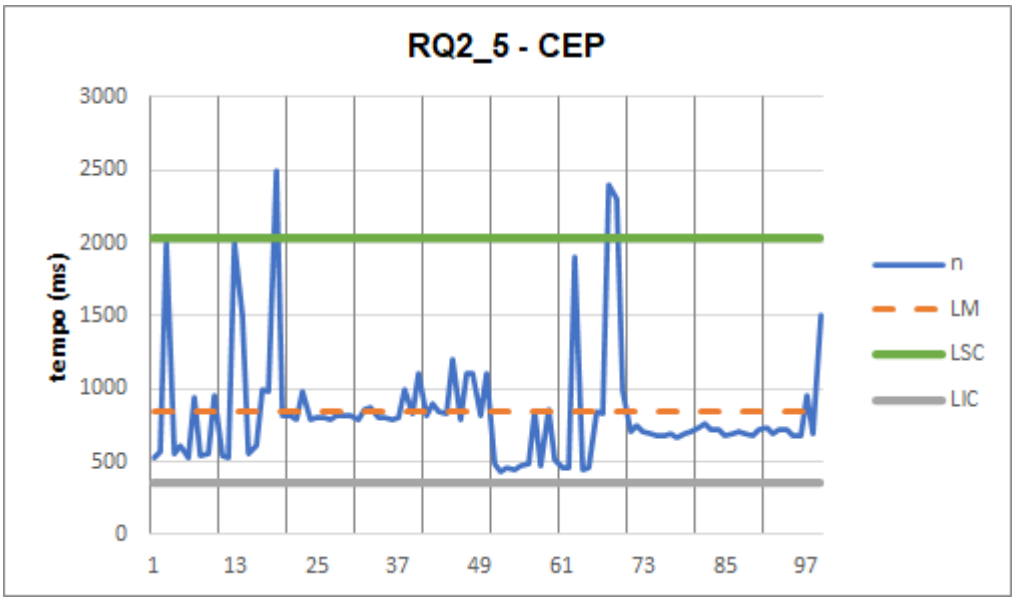
Pokedex.Tests._4_INTERACAO.RQ2_BuscaPokemonDaApi.BuscaPorNome
(Imagem: 4 Interação, RQ2_3)

Esse requisito obteve um bom resultado com 99.33% de qualidade em 149 iterações. A maioria das iterações foi bem-sucedida, com apenas uma iteração que ultrapassou o limite superior de controle.



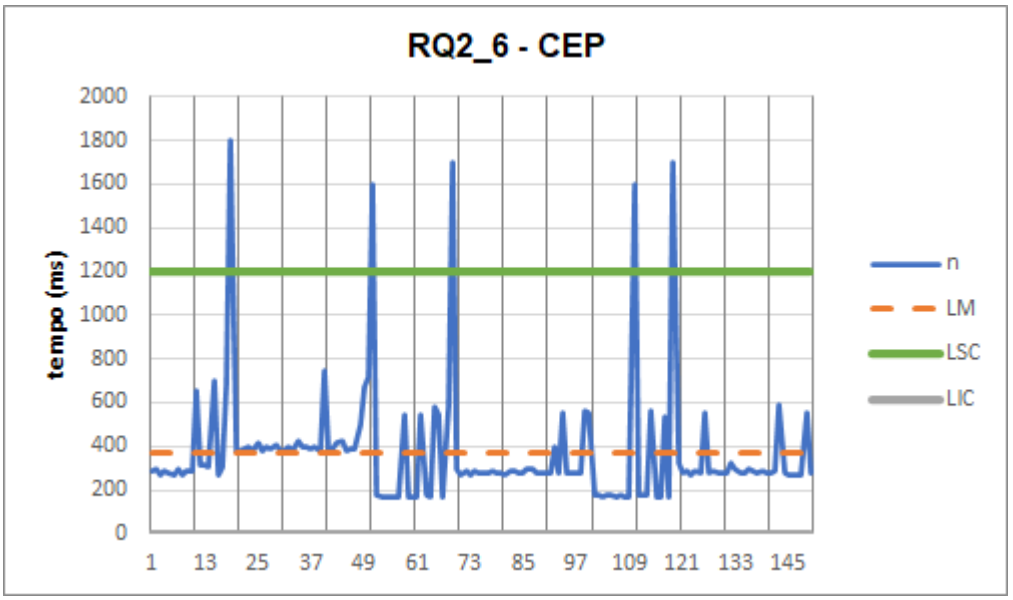
Pokedex.Tests._4_INTERACAO.RQ2_BuscaPokemonDaApi.RetornarListaVaziaQuandoNaoExistirNaApi
(Imagem: 4 Interação, RQ2_4)

Esse requisito atingiu 97.98% de qualidade em 99 iterações. A maioria das iterações foi bem-sucedida, com apenas duas iterações que não atenderam completamente ao requisito, ficando acima do limite superior de controle.



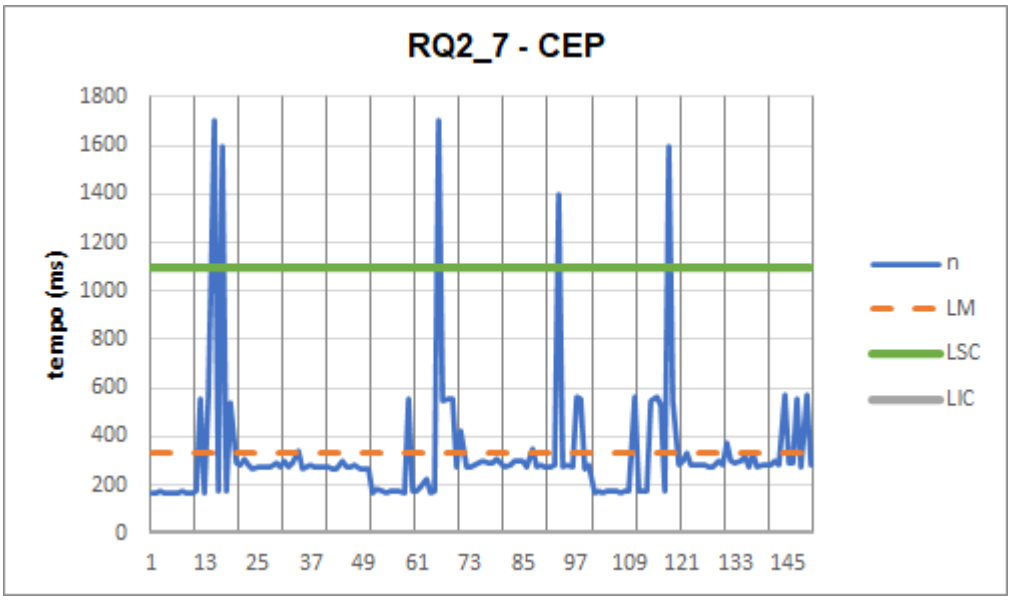
Pokedex.Tests._4_INTERACAO.RQ2_BuscaPokemonDaApi.RetornarQtdApiMenosQtdInformadaQuandoQuantidadeForNegativa
(Imagem: 4 Interação, RQ2_5)

Similar ao requisito anterior, esse requisito também obteve 97.98% de qualidade em 99 iterações. A maioria das iterações foi bem-sucedida, com apenas duas iterações que não atingiram completamente o requisito, porém vale ressaltar que outras duas iterações foram executadas no tempo exato do limite superior de controle.



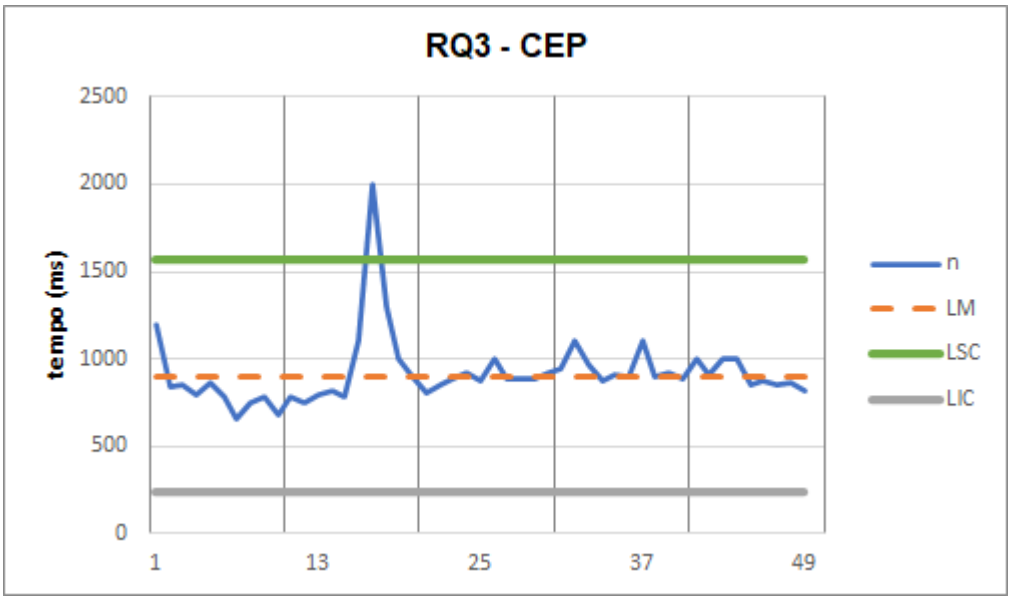
Pokedex.Tests._4_INTERACAO.RQ2_BuscaPokemonDaApi.RetornarQuantidadeInformada
(Imagem: 4 Interação, RQ2_6)

Esse requisito obteve 96.64% de qualidade em 149 iterações. A maioria das iterações foi bem-sucedida, porém cinco delas superaram o limite superior de controle.



Pokedex.Tests._4_INTERACAO.RQ2_BuscaPokemonDaApi.UltimoPokemonDeveTerIdIgualQtdMaisInicio
(Imagem: 4 Interação, RQ2_7)

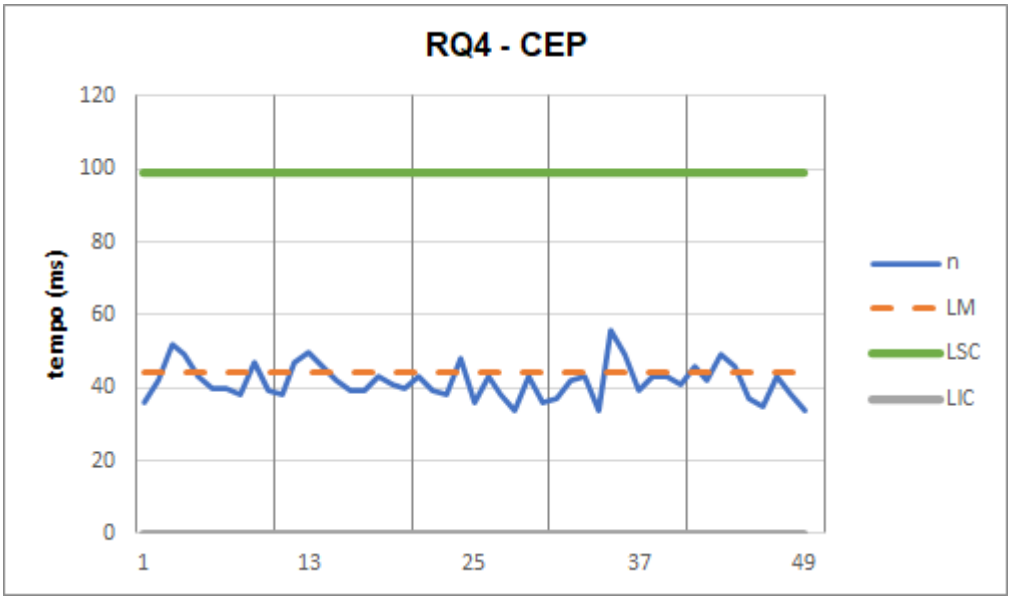
Nesse requisito, houve um desempenho razoável, com 97.49% de qualidade em 199 iterações. A maioria das iterações alcançou o resultado esperado, mas cinco superaram o limite superior de controle.



Pokedex.Tests._4_INTERACAO.RQ3_SePokemonNaoEstaNoBancoDeDadosBuscarNaAPI.BuscaPorPokemonNaoPresenteNoBancoDeDados

(Imagem: 4 Interação, RQ3)

Esse requisito atingiu 98% de qualidade em 50 iterações. A maioria das iterações foi bem-sucedida, com apenas uma iteração que excedeu o limite superior de controle.



Pokedex.Tests._4_INTERACAO.RQ4_CadastroNovosPokemonsComFaixaMaiorQueDaAPI.CadastrandoPokemonComIDMaiorQueDaAPI

(Imagem: 4 Interação, RQ4)

Esse requisito obteve 100% de qualidade, o que indica que o sistema atendeu totalmente ao requisito.