

UNIVERSIDADE FEDERAL DE VIÇOSA · UFV - CAMPUS FLORESTAL

CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADOR 2024

**REFINAMENTO DA SPRINT 4**

CSU02: Gerar e salvar relatório individual

**v.1.0.0**

Guilherme Broedel Zorzal - 5064

Arthur Fernandes Bastos - 4679

Florestal - MG

Novembro de 2024

LISTA DE FIGURAS

[Figura 1: Classe Relatorio 6](#__RefHeading___Toc813_1213076376)

[Figura 2: Classe StatsFrase 6](#__RefHeading___Toc1402_1213076376)

SUMÁRIO

[1. INTRODUÇÃO 3](#__RefHeading___Toc1489_1213076376)

[2. CASO DE USO 4](#__RefHeading___Toc907_1213076376)

[4. BACK-END 5](#__RefHeading___Toc905_1213076376)

[4.1 Classe Relatorio 6](#__RefHeading___Toc925_1213076376)

[4.2 Classe StatsFrase 6](#__RefHeading___Toc923_1213076376)

[5. Banco de Dados 7](#__RefHeading___Toc911_1213076376)

# 1. INTRODUÇÃO

O objetivo do seguinte documento consiste em apresentar especificações relacionadas ao caso de uso abordado na quarta sprint, de acordo com o product backlog proposto com base no calendário.

Uma convenção importante com relação aos diagramas de classe: caso a representação da classe não possua nenhum método, deve-se entender que a classe precisa possuir getters e setters para seus atributos, a menos que tenham sido dadas outras instruções na descrição da classe presente no documento.

# 

# 2. CASO DE USO

O caso de uso abordado na Sprint atual ficou decidido como sendo o CSU02: Gerar e salvar relatório individual. Sua especificação, proposta na documentação dos casos de uso em ESOF I, pode ser conferida abaixo. Note que os casos de usos sofreram poucas alterações em relação ao proposto originalmente, ou seja, o fluxo proposto inicialmente continua sendo o mesmo. Apenas algumas palavras foram atualizadas de forma a melhorar a compreensão.

Esse caso de uso é responsável por receber os dados de cada frase finalizada, a partir dele calcular as métricas e por fim gerar o relatório individual. A documentação do caso de uso pode ser encontrada abaixo.

|  |
| --- |
| CSU02: Gerar e salvar relatório individual  **Nome:** Gerar e salvar relatório individual.  **Ator Primário:** Aluno.  **Sumário:** Finalizar um jogo, salvando as estatísticas relativas a ele e gerando um relatório que descreve o desempenho do aluno no mesmo.  **Pré-condições:**   * O usuário deve ter criado seu personagem; * O texto deve ter sido escolhido pelo professor; * O texto deve ter sido completamente digitado de maneira correta pelo aluno.   **Fluxo Principal:**   1. O sistema gera e exibe o relatório. 2. O usuário pressiona o botão“Jogar novamente”. 3. O sistema dispara a ação “Jogar” com o próximo texto selecionado pelo professor. 4. Fim do caso de uso.   **Fluxo Alternativo (2): o usuário pressiona o botão “Sair”.**   1. O sistema retorna à tela inicial.   **Pós-condições:**   * O aluno pode passar imediatamente para o próximo texto selecionado pelo professor. |

3. ESTRUTURA GERAL

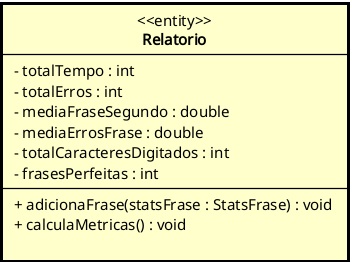
Quando é iniciada cada frase, o código implementado no front-end irá fazer a contagem dos dados necessários para as métricas, passando-as através do End point para o back-end. Após o recebimento dos dados, o sistema deve ser capaz de calcular as métricas e fornecer um relatório em uma nova tela com os dados referentes ao total de tempo gasto e dos erros, a média tempo de frase medido em minutos e a média de erros por frase.

# 4. BACK-END

Na sprint atual, o back-end será responsável por implementar algumas classes e suas relações, conforme o diagrama de classes disponível no arquivo DOCX na mesma pasta. Entre as tarefas, destaca-se o desenvolvimento de uma classe Relatório, que incluirá os métodos getters e setters para cada métrica, e de uma classe Frase, destinada a armazenar as informações de cada frase enviada. Além disso, será criada uma classe para gerenciar o acesso ao banco de dados de relatórios, uma classe controladora e, por fim, uma classe de endpoints responsável por integrar o back-end ao front-end.

## 

## 4.1 Classe Relatorio

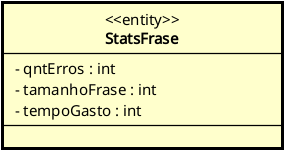


###### Figura 1: Classe Relatorio

A classe Relatorio possui getters e setters referentes a cada tipo de métrica calculada, onde os setters devem ser privados, com exceção do adicionaFrase o qual deve ser capaz de armazenar em uma lista todas as frases enviadas pela classe ControladoraRelatorio, contendo os dados referentes a cada uma. Ou seja, a classe Relatorio deve conter uma lista do tipo StatsFrase.

Dessa forma, o método calculaMetricas ao final do processamento de frases irá acessar essa lista e, a partir dos dados de cada campo da classe Frase, gerar as métricas referentes àquele texto (atualizando os campos da classe Relatorio com os setters privados).

## 4.2 Classe StatsFrase



###### **Figura 2:** Classe StatsFrase

A classe StatsFrase é responsável por armazenar os dados referentes a cada frase finalizada, sendo assim ela deve ser capaz de guardar a quantidade de erros, o tamanho da frase e o tempo gasto. Deve possuir os devidos getters e setters.

# 5. Banco de Dados

Para a sprint atual, a tarefa do banco de dados será o armazenamento dos relatórios individuais dos alunos através da tabela ‘RelatorioIndividual’, contendo as métricas que foram calculadas no back-end. As colunas do banco de dados deverão ser as seguintes:

* id
* totalTempo
* totalErros
* totalErros
* mediaFraseSegundo
* mediaErrosFrase
* totalCaracteresDigitados
* frasesPerfeitas

Onde id é a chave primária.