

UNIVERSIDADE FEDERAL DE VIÇOSA · UFV - CAMPUS FLORESTAL

CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADOR 2024

**REFINAMENTO DA SPRINT 5**

CSU09: CRUD de texto

**v.1.0.0**

Guilherme Broedel Zorzal - 5064

Arthur Fernandes Bastos - 4679

Florestal - MG

Novembro de 2024

LISTA DE FIGURAS

[**Figura 1: Classes Dificuldade e Texto 7**](#_heading=h.26in1rg)

SUMÁRIO

[**1. ATUALIZAÇÕES E CORREÇÕES 3**](#_heading=h.1fob9te)

[**2. INTRODUÇÃO 3**](#_heading=h.3znysh7)

[**3. CASO DE USO 3**](#_heading=h.2et92p0)

[**4. BACK-END 6**](#_heading=h.2s8eyo1)

[4.1 Classe Texto e Classe Dificuldade 6](#_heading=h.3rdcrjn)

[**5. BANCO DE DADOS 8**](#_heading=h.1hmsyys)

# 1. ATUALIZAÇÕES E CORREÇÕES

No front-end, a tela de Área do Professor deve ser atualizada para contemplar as demais funcionalidades pensadas e desenvolvidas. Segue no figma e também na seção relacionada ao front-end, a nova tela.

# 2. INTRODUÇÃO

O objetivo do seguinte documento consiste em apresentar especificações relacionadas ao caso de uso abordado na quinta 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.

# 3. CASO DE USO

O caso de uso abordado na Sprint atual ficou decidido como sendo o CSU09: Crud de texto Sua especificação pode ser conferida abaixo. O Caso de CRUD de texto foi proposto posteriormente devido ao fato de ser uma melhoria para o sistema, por isso não existir sua especificação nos casos de uso original.

| CSU09: CRUD de Texto  **Nome:** CRUD de Texto  **Ator Primário:** Professor  **Sumário:** Permitir que o professor crie, leia, atualize e delete textos que serão utilizados pelos alunos durante o jogo.  **Pré-condições:**   * O professor deve estar autenticado no sistema. * O sistema deve estar conectado a um banco de dados operacional.   **Fluxo Principal:**   1. **Criação (Create):**    1. O professor seleciona a opção "Cadastrar novo texto".    2. O sistema exibe uma interface para inserção do texto.    3. O professor insere o texto e confirma a ação.    4. O sistema salva o texto no banco de dados.    5. O sistema confirma que o texto foi cadastrado com sucesso 2. **Leitura (Read):**    1. O professor seleciona a opção "Visualizar textos".    2. O sistema exibe a lista de textos cadastrados.    3. O professor pode selecionar um texto para visualizar seu conteúdo. 3. **Atualização (Update):**    1. O professor seleciona a opção "Editar texto".    2. O sistema exibe os textos disponíveis para edição.    3. O professor escolhe um texto e edita seu conteúdo.    4. O professor confirma as alterações.    5. O sistema atualiza o texto no banco de dados e exibe uma mensagem de sucesso. 4. **Exclusão (Delete):**    1. O professor seleciona a opção "Excluir texto".    2. O sistema exibe a lista de textos cadastrados.    3. O professor seleciona o texto que deseja excluir.    4. O sistema solicita confirmação.    5. O texto é removido do banco de dados, e uma mensagem de confirmação é exibida.   **Fluxos Alternativos:**   * **Erro de conexão com o banco de dados:**   + O sistema informa ao usuário que não foi possível completar a ação devido a uma falha de conexão e fornece a opção de tentar novamente ou retornar ao menu principal**.** * **Texto não encontrado (Read ou Delete):**   + O sistema informa que o texto selecionado não está disponível e retorna à lista de textos.   **Pós-condições:**   * O texto criado, atualizado ou excluído é refletido no banco de dados. * Alterações estão disponíveis para o uso no sistema pelos alunos. |
| --- |

4. ESTRUTURA GERAL

Como estrutura geral, nessa sprint deverá ser desenvolvido a interface e a lógica por trás da criação, visualização, atualização e remoção de texto dentro do sistema, o back-end deverá tratar da utilização dos textos de forma a atender todas as funcionalidades necessárias e o front-end deverá disponibilizar e possibilitar para o usuário, o uso das funcionalidades implementadas.

# 

# 4. BACK-END

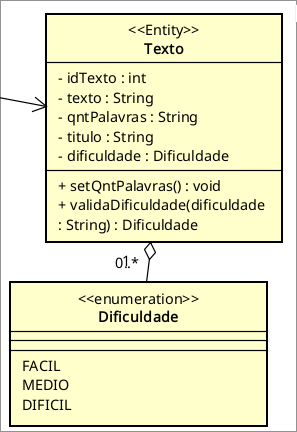
Na sprint atual, o back-end será responsável por implementar algumas classes e suas relações, conforme o diagrama de classes disponível no arquivo DOCX na mesma pasta. O coração da sprint atual é a classe ControladoraCrudTexto, que possuirá boa parte das funcionalidades relacionadas ao uso dos textos. Ela substituirá a classe ControladoraTextoAux assim que for realizado o refinamento do caso de uso SelecionarTexto (próxima sprint).

## 

## 4.1 Classe Texto e Classe Dificuldade

A classe Texto será a classe responsável por guardar o texto escolhido pelo professor, e que será usada posteriormente para possibilitar o uso do jogo. Ela deve possuir uma instância do Enum dificuldade.

O enum dificuldade, por sua vez, será responsável por guardar o nível de dificuldade do jogo, que será escolhido pelo professor durante a criação do jogo. Ele deve possuir as seguintes dificuldades: FACIL, MEDIO e DIFICIL.



###### Figura 1: Classes Dificuldade e Texto

Um ponto importante: o setter de Dificuldade receberá um enum dificuldade (tornando simples sua implementação). Porém a classe Texto terá um novo método chamado validaDificuldade. Esse método receberá uma string contendo o nome da dificuldade (fácil, médio e difícil). Atentem-se ao tratamento de maiúsculas e minúsculas, acentos, dentre outras possibilidades de erro (deve-se aceitar apenas as versões acentuadas sem capitalização, ou seja, tudo minúsculo). No fim validaDificuldade retorna um ENUM Dificuldade.

Note que setQntPalavras está como privado: ele deve ser chamado junto ao setter de texto, calculando o total de palavras presentes no texto e atribuindo o resultado ao atributo qntPalavras.

Além disso, seu construtor deve inicializar o idTexto como -1.

Note que ainda há a necessidade de um getter de dificuldade. “validaDificuldade” não substitui essa função, visto que ele é um método estático. O getter de dificuldade retorna um enum Dificuldade.

Por fim, quando o método getTexto for chamado, caso não haja nenhum texto carregado na classe (idTexto é igual a -1), esse método deve retornar um erro.

# 5. BANCO DE DADOS

O banco de dados deve conter uma tabela chamada Textos, que armazenará as informações dos textos criados. Essa tabela terá os seguintes campos: id (chave primária para identificar o texto), texto (conteúdo do texto), título (título do texto) e a dificuldade (nível de dificuldade). O sistema deve permitir as operações de cadastro, atualização, consulta e exclusão de textos.

Para isso, ao cadastrar um texto, será necessário inserir seu conteúdo, título e dificuldade. As atualizações devem modificar apenas os campos alterados. Além disso, deve ser possível consultar um texto específico pelo seu ID ou recuperar metadados de todos os textos, como título e dificuldade. Por fim, a exclusão deve permitir remover textos com base no ID fornecido.