

UNIVERSIDADE FEDERAL DE VIÇOSA · UFV - CAMPUS FLORESTAL

CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADOR 2024

**REFINAMENTO DA SPRINT 6**

CSU06: Escolher Texto

**v.1.0.0**

Guilherme Broedel Zorzal - 5064

Arthur Fernandes Bastos - 4679

Florestal - MG

Novembro de 2024

LISTA DE FIGURAS

[**Figura 1: Diagrama de classes antigo, referente ao CSU01 3**](#_r66kldu3eoxh)

[**Figura 2: Nova organização do diagrama de classes 4**](#_nmyo8xuwllrk)

[**Figura 3: Classe ConexaoHost 7**](#_q9yokf8ghjzc)

[**Figura 4: Classe ControladoraCrudTexto 8**](#_4swjd16mf01w)

[**Figura 5: Classe EndPointTexto atualizada 9**](#_1tf2dbane5eh)

[**Figura 6: Tela de Seleção de texto 11**](#_ol8nmx3q4ycm)

SUMÁRIO

[**1. ATUALIZAÇÕES E CORREÇÕES 3**](#_tmeox120218u)

[**2. INTRODUÇÃO 5**](#_70quw3oh7ylw)

[**3. CASO DE USO 6**](#_ak5dt6g3marc)

[**4. ESTRUTURA GERAL 7**](#_uczxuu75cxw8)

[**5. BACK-END 7**](#_i3yyqqcljtlj)

[5.1 Classe ConexaoHost 7](#_eh037wi3ezu2)

[5.2 Classe ControladoraCrudTexto 8](#_q9i8i0guxsxz)

[Funções 8](#_hhgvaytwj4ew)

[5.3 EndPointTexto 9](#_xpqg8downqr7)

[5.3.1 EndPoints 10](#_havvsfs20gga)

[**6 FRONT-END 11**](#_zhazk5c76q23)

[6.1 Fluxo de telas 11](#_94mscwsxnwv5)

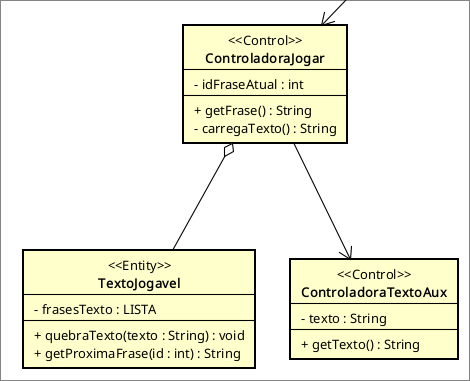
[6.2 End-Points 11](#_yq4787kb5sfw)

[**7 BANCO DE DADOS 11**](#_lw5r5emju9l6)

# 1. ATUALIZAÇÕES E CORREÇÕES

Toda referência a este caso de uso e dessa sprint 06 como “selecionar texto” deve ser entendida como a funcionalidade de “Escolher texto”. Ou seja, escolher texto é a funcionalidade do back-end que escolhe o texto selecionado pelo professor.

Nessa sprint, algumas atualizações deverão ser feitas. A principal delas é a forma de funcionamento da função “carregaTexto” de ControladoraJogar. Na sprint do caso de uso Jogar, a função carrega texto era responsável por carregar o texto presente na “ControladoraTextoAux” através da função “getTexto”. Confira na imagem abaixo:



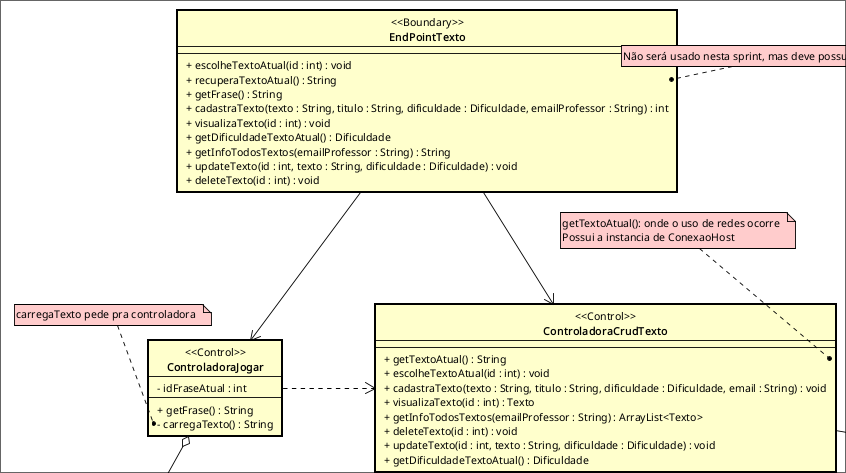
###### Figura 1: Diagrama de classes antigo, referente ao CSU01

A partir dessa sprint, o funcionamento será o seguinte: A classe ControladoraJogar terá uma referência a classe ControladoraCrudTexto presente dentro de EndPointTexto. Ou seja, ambas endPointTexto e ControladoraJogar terão uma instância do mesmo objeto. Além disso, ControladoraJogar passa a recuperar o texto não através de “getTexto”, mas sim de “getTextoAtual”, presente dentro da controladora de crud.

Lembrando que, durante a construção da controladoraJogar na sprint 3, era necessário tratar erros caso não houvesse um texto selecionado (caso o método getTexto retornasse um erro), de forma que chegasse ao front uma mensagem com essa informação. Esse fluxo de funcionamento entrará em prática nesta sprint.

Novamente, a classe endPointTexto e ControladoraJogar terão uma referência para exatamente o mesmo objeto. A instância que for criada em EndPointTexto deve ser passada por referência para ControladoraJogar. Dessa forma, ambas contém exatamente a mesma controladora dentro de si.

As demais alterações são apenas adições de funções a classes já existentes (EndPointTexto e ControladoraCrudTexto) que serão explicadas mais adiante.



###### **Figura 2:** Nova organização do diagrama de classes

Além disso, um erro acabou passando despercebido durante a modelagem e alguns métodos terão que sofrer uma pequena alteração.

Cada professor tem seu conjunto de texto, e os textos não devem ser compartilhados entre si (essa dinâmica de como os textos serão organizados será modelado nessa sprint e pode ser conferida no fim do documento). A ideia é a seguinte: o front guardará o email do professor. Quando operações de recuperação e criação de textos forem realizadas (são as funções cadastraTexto() e getInfoTodosTextos()), o email do professor será enviado ao backend, possibilitando assim saber a quem aquele texto deve ser associado ou quais textos devem ser recuperados ao front.

Como cada texto possui um id único, a remoção e deleção de textos não precisa do uso do email do professor: o front sabe exatamente o id de cada texto e, como cada texto possui um id único, sua exclusão e atualização alterarão exatamente os textos do professor que encontra-se logado (não havendo assim a necessidade de enviar o email para essas requisições).

**IMPORTANTE**: O campo que conterá o email do usuário, que contém o token de autenticação, é o campo “Authorization” dentro do header da requisição (no caso do getInfoTodosTextos que é um GET, fica no header da requisição e no caso do cadastraTexto() que é um POST, tambem fica dentro do header)

# 

# 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 CSU06: Escolher texto. 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 possibilitar o professor selecionar um dos textos disponíveis para uso, o texto selecionado deve ser disponibilizado para os alunos jogarem.

| CSU06: Escolher texto  **Nome:** Escolher texto.  **Ator Primário:** Professor.  **Sumário:** O professor será capaz de escolher um texto a ser utilizado pelos alunos no jogo.  **Pré-condições:**   * O usuário precisa estar autenticado.   **Fluxo Principal:**   1. O usuário seleciona a opção de escolher texto. 2. O sistema abre uma opção para que o usuário escolha qual texto será acessado pelos alunos. 3. O usuário escolhe o texto. 4. O sistema exibe o texto selecionado para os alunos.   **Pós-condições:**   * Os alunos poderão acessar o texto que foi escolhido. |
| --- |

# 4. ESTRUTURA GERAL

Como estrutura geral, nessa sprint deverá ser desenvolvido o funcionamento da requisição a partir do front-end de selecionar o texto a ser utilizado, a qual será recebida pelo back-end que terá de atualizar o texto atual e enviar o texto selecionado para ser exibido para o usuário quando necessário.

# 5. 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 possui 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).

## 5.1 Classe ConexaoHost

A classe ConexaoHost não conterá nada nessa sprint. O seu único método (getTextoHost) deve retornar um erro quando chamado. Ela funcionará dessa forma visto que ela será utilizada apenas para montar o fluxo de execução das requisições de texto e deixar o ambiente de desenvolvimento preparado para receber a funcionalidade de rede.

## 

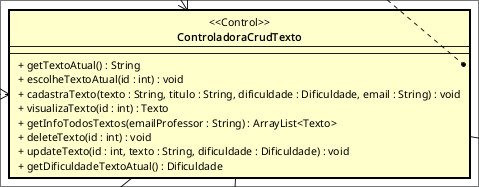
###### Figura 3: Classe ConexaoHost

O erro retornado por seu método (reforçando que nessa sprint, chamar o método deve resultar em erro) simula a conexão ao host, indicando que não foi possível realizar a conexão.

###### 

## 5.2 Classe ControladoraCrudTexto

Nessa sprint, a estrutura geral da classe ControladoraCrudTexto continua a mesma, com a adição de apenas dois métodos: getTextoAtual e escolherTextoAtual.



###### Figura 4: Classe ControladoraCrudTexto

### 

### Funções

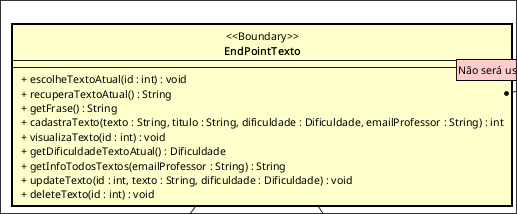
* escolheTextoAtual(id : int) : void = A função deverá receber o id do texto escolhido no front-end da aplicação e enviá-lo para a classe Texto. A classe Texto (a instância presente na controladoraCrudTexto) possui qual texto será efetivamente utilizado na hora de jogar. O texto deve ser recuperado do banco com base no id ( o banco retorna um tipo Texto) e atribuido na instância local de texto.
* getTextoAtual(): String = A função getTextoAtual() possui a funcionalidade de recuperar o texto escolhido pelo professor para que seja utilizado durante o jogo. Ela deve possuir o seguinte fluxo: primeiramente, ela tenta recuperar o texto existente na classe Texto (instanciado dentro da própria classe). Caso a recuperação do texto utilizando o método disponibilizado pela classe resulte em erro, ele deve tentar recuperar o texto por meio do método getTextoHost() presente na classe ConexaoHost(). Caso essa classe também retorne erro, o método getTextoAtual() termina, também retornando um erro. Note que, na sprint atual, o método getTextoHost sempre retornará um erro. Porém o fluxo de funcionamento de getTextoAtual() deve funcionar independente da classe ConexaoHost. A implementação será feita dessa forma pois tornará a implantação do uso de redes algo muito simples e natural, e evitando bastante retrabalho.
  + Além disso, caso o método getTextoHost() retorne um texto (não deve acontecer nessa sprint, mas se tivesse que ocorrer seria uma string), o retorno de getTextoAtual será essa string.
  + REFORÇANDO: Primeiramente deve ser feita a recuperação do texto local. Só então, em caso de erro, é que deve ser recuperado o texto do host (que como não possui implementação, também acabará retornando um erro)

Os demais métodos permanecem da mesma forma.

## 

## 5.3 EndPointTexto

Para a sprint atual, alguns métodos foram adicionados além do existente na sprint 3. O método getFrase permanece inalterado.



###### Figura 5: Classe EndPointTexto atualizada

### 

### 

#### 5.3.1 EndPoints

Para facilitar o entendimento das rotas, cada rota terá o nome do método associado. Isso melhora o entendimento e a interligação entre rota e método na hora de implementar.

Nessa sprint será adicionado os dois métodos na classe EndPointTexto descritos a seguir:

* /texto
  + /escolheAtual/:id
    - POST: A partir da seleção do professor, deve ser recebido o id do texto escolhido e enviado para a controladoraCrudTexto que setará o texto (pelo método escolheTextoAtual).
  + /recuperaAtual
    - GET: Têm a função de recuperar o texto escolhido pelo professor, através do método getTextoAtual da ControladoraCrudTexto (essa rota será usada para o sistema de rede, que não vai ser implementado agora).

# 

A rota texto/recuperaAtual deve ser implementada também nesta sprint. Ela só não terá uso durante a sprint atual. Porém o ideal é que ela já esteja configurada para que, durante a sprint de integração com o sistema de redes, tudo já esteja devidamente funcionando, sendo necessário dedicar o esforço exclusivamente para a questão da rede.

# 6 FRONT-END

Quanto ao que deve ser implementado pelo front-end, será apenas a funcionalidade de selecionar o texto na tela de CRUD de texto, funcionando de forma a habilitar o professor a selecionar o texto que vai ser utilizado.

# 



###### Figura 6: Tela de Seleção de texto

## 6.1 Fluxo de telas

Nessa sprint, o Front-end não tem um fluxo de tela muito relevante, sendo necessário somente a implementação do pop-up de seleção de texto e da requisição após o texto ser selecionado.

## 

## 6.2 End-Points

Os endpoints a serem utilizados pelo front podem ser encontrados dentro da área “endpoints” do capítulo “Backend” deste documento, com exceção de /texto/recuperaAtual. (esse método é de uso esclusivo para o sistema de redes).

# 7 BANCO DE DADOS

Nessa sprint, o banco de dados terá o papel de realizar a união entre a tabela que guardar as informações do professor e a tabela que guarda as informações dos textos.

Cada texto deve estar associado a apenas um professor. O professor é dono de um conjunto de textos exclusivamente dele. Além disso, como já feito anteriormente, cada texto possui um id exclusivo (isso possibilita a manipulação dos textos sem a necessidade de precisar saber quem é o dono do texto: cada id referencia apenas um texto, o que já basta).

A escolha das chaves primárias e estrangeiras fica a critério dos desenvolvedores de BD, sendo que deve ficar modelado de forma que faça sentido.