SERVIÇO PÚBLICO FEDERAL · MINISTÉRIO DA EDUCAÇÃO UNIVERSIDADE FEDERAL DE VIÇOSA · UFV

CAMPUS FLORESTAL



**REFINAMENTO DA SPRINT 03**

**CSU01**

GUILHERME BROEDEL ZORZAL - 5064

ARTHUR FERNANDES BASTOS - 4679

Florestal - MG

Outubro - 2024

**SUMÁRIO**

[**1. INTRODUÇÃO 3**](#_xvx7uekzbnds)

[**2. CASO DE USO 3**](#_xh9ool8zywqb)

[CSU01: Jogar 4](#_6emizi6rx4dk)

[**3. ESTRUTURA GERAL 5**](#_uczxuu75cxw8)

[**Atualizações e correções 5**](#_sk5ylt50r1ca)

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

[4.1 TextoJogavel 6](#_8q4ql9fo76tl)

[Figura 1 : Classe do Texto Jogável 6](#_fpddep1s5810)

[4.2 ControladoraTextoAux 7](#_z4l0zgb5e9rm)

[4.3 EndPointJogar 8](#_16dfec75mrjn)

[4.4 ControladoraJogar 8](#_o55hrkyr7p7x)

[Figura 4: Classe controladora do Jogo 9](#_8bg56tfioofa)

[**5. FRONT-END 10**](#_reoritbgbhi1)

[Figura 5 :Tela do caso de uso Jogar 10](#_c7y9q4vekma4)

[5.1 FLUXO DE TELAS 1](#_jp4nrwv9y752)

[Dica de implementação 1](#_jejcmijhwku6)

[5.2 END-POINTS 1](#_hgwe0x4qf8ds)

[5.3 BANCO DE DADOS 1](#_fbtd85m90bx1)

# 

# 

# **1. INTRODUÇÃO**

O objetivo do seguinte documento consiste em apresentar especificações relacionadas ao caso de uso abordado na terceira 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 CSU01: Jogar. 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.

Esses casos de uso são responsáveis por exibir o texto a ser escrito, computar a digitação do usuário e demonstrar a lógica de erros e acertos. A documentação do caso de uso pode ser encontrada abaixo.

| CSU01: Jogar **Nome:** Jogar.  **Ator Primário:** Aluno.  **Sumário:** Ser capaz de iniciar o jogo e começar a contagem de pontos de acordo com o desempenho do aluno.  **Pré-condições:**   * O usuário deve ter criado seu personagem; * O texto deve ter sido selecionado pelo professor.   **Fluxo Principal:**   1. O usuário seleciona a opção "jogar"; 2. O sistema exibe a primeira frase do texto selecionado pelo professor; 3. O aluno digita; 4. O sistema avança o texto e computa o tempo e as entradas; 5. O aluno termina de digitar o texto; 6. O sistema dispara a ação “gerar e salvar relatório individual”; 7. Fim de caso de uso.   **Fluxo Alternativo (4): O texto não foi terminado.**   1. O sistema exibe a próxima frase do texto selecionado pelo professor; 2. Retorna ao passo 3.   **Pós-condições:**   * O usuário será capaz de visualizar sua pontuação. |
| --- |

# 

# **3. ESTRUTURA GERAL**

Quando a tela de jogo carregar, o nome e o avatar do jogador devem ser exibidos. O acesso à digitação deve ser bloqueado até que o professor escolha um texto, que será carregado em pequenas partes: duas ou três frases por vez. Esse conteúdo estará salvo no banco de dados, e, conforme o jogador digita, o texto já escrito muda de cor, enquanto o que ainda resta para ser digitado permanece na cor branca. Um teclado ilustrativo também acompanha o jogador, acendendo a próxima letra a ser digitada a cada caractere. Ao término de cada frase, devem ser registrados o tempo gasto, a quantidade de erros e a frase seguinte é exibida. Quando o jogador concluir todo o texto, ele será redirecionado para uma tela com os resultados individuais.

# Atualizações e correções

Design de jogo: Reconfigurar tela de seleção de texto e criação de texto para exibir a dificuldade de cada texto (no figma).

IMPORTANTE: O back-end precisa disponibilizar uma rota que não faça nada. Pode ser uma rota /teste, por exemplo. A única coisa que precisa ser retornada é um texto, como exemplo: “Conexão com o Backend de Digitóvsky funcionando”.

Deve haver um script para configurar o arquivo “.env” e configurar o ip do host. Tem que ser bonito e intuitivo para facilitar usuários leigos. O nome da variável que guarda essa configuração fica a critério dos desenvolvedores.

# **4. BACK-END**

O back-end, na sprint atual, seguindo o diagrama de Classes disponível no arquivo docx na mesma pasta, ficará responsável pela implementação de algumas classes e relações envolvendo uma entidade do Texto jogável, uma controladora para o jogo, os endpoints para jogar e uma controladora de texto que será implementada apenas com funções específicas dessa sprint (será aprimorada com o caso de uso de Crud de texto).

### 4.1 TextoJogavel

A classe TextoJogavel vai servir para conter uma lista de frases que serão carregadas aos poucos no front-end, ou seja, uma lista de frases (cada uma contendo entre 40 e 55 caracteres) organizadas em ordem. A estrutura de dados a ser utilizada fica a critério dos desenvolvedores, desde que ela cumpra as funcionalidades especificadas.

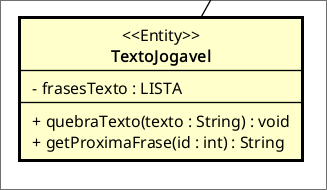
A função quebraTexto() recebe um texto e quebra ele em uma lista de frases (dentro da variável frasesTexto).

A função getProximaFrase() recebe um inteiro (o id da frase sendo utilizada atualmente) e retorna a próxima frase. Por exemplo, se foi recebido o id 2, retorna a frase na posição 3 da lista. Caso a posição que está se tentando acessar não exista, retorna uma string vazia. Além disso, caso o id do parâmetro seja -1, retorna a primeira.

A função dessa classe é prover as frases a serem digitadas no front-end de forma iterativa, para que seja possível realizar a digitação da forma mais fluida possível.

IMPORTANTE: as frases devem possuir um espaço “ “ no fim das frases, com exceção da última. Deve ser feito dessa forma, visto que o front-end irá apenas concatenar as frases de acordo com a ordem em que forem enviadas e, portanto, a concatenação não deve gerar espaços repetidos entre as frases (concatenar uma frase com um espaço no fim e um no início gera dois espaços consecutivos), bem como não deve retirar os espaços (concatenar uniria a última palavra da frase anterior com a primeira, o que é errado).

OBS.: Caso o id passado como parâmetro para a função getProximaFrase não fizer parte da lista, ou for o id da última frase, deve ser retornado uma string vazia “”.

****

###### **Figura 1 :** Classe do Texto Jogável

### 

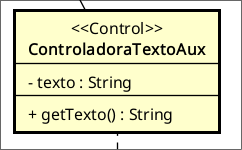
### 4.2 ControladoraTextoAux

A classe ControladoraTextoAux é uma classe temporária que será “substituída” na próxima sprint, mas por enquanto tem a função de receber a requisição vinda da classe controladoraJogar e retorna um texto.

O construtor da classe deve inicializar o campo “texto” com um texto qualquer, que possua por volta de 300 caracteres. O texto é estático dessa forma pois essa é uma classe provisória e será melhorada pelo caso de uso Crud de Texto.

O método getTexto() retorna o texto armazenado.

Novamente, essa classe NÃO FARÁ PARTE DA VERSÃO FINAL. Ela foi pensada apenas para facilitar os testes e funcionalidades das outras três classes. Na próxima sprint, quando for realizado o CRUD de texto, ela será substituída por uma classe que forneça a interface definitiva.

****

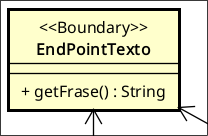
***Figura 2 :*** *Classe auxiliar Controladora de Texto*

###### 

### 

### 4.3 EndPointTexto

A classe EndPointTexto é a classe responsável por receber as requisições enviadas pelo front-end e responder de forma adequada ao que foi solicitado. Essa classe será uma classe de fronteira cuja única função é tratar os endpoints. Toda lógica que será executada por cada rota fica terceirizada na classe “ControladoraJogar”.



***Figura 3:*** *Classe End-Point do Jogo para guardar as rotas*

No back-end, os EndPoints serão os seguintes:

* /jogo
  + /frase:
    - GET: Essa é a função responsável por enviar as frases para o front-end, recuperadas a partir do método getFrase da ControladoraJogarEla retorna um JSON no seguinte formato:

{

frase: contem a frase retornada pela classe TextoJogavel

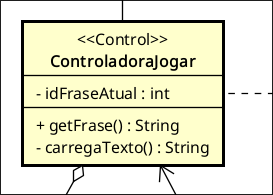
}



Caso não tenha sido possível carregar o texto, retorna um JSON com os campos “field” e “message”, como foi feito nas sprints anteriores.

### 4.4 ControladoraJogar

A ControladoraJogar possui a funcionalidade de gerenciar as requisições de frases que serão realizadas pelo programa, ela vai ter relações com as outras classes abordadas neste documento e também com a classe ControladoraAluno. O construtor da classe deve inicializar idFraseAtual com -1.

****

###### **Figura 4:** Classe controladora do Jogo

**Módulo principal:**

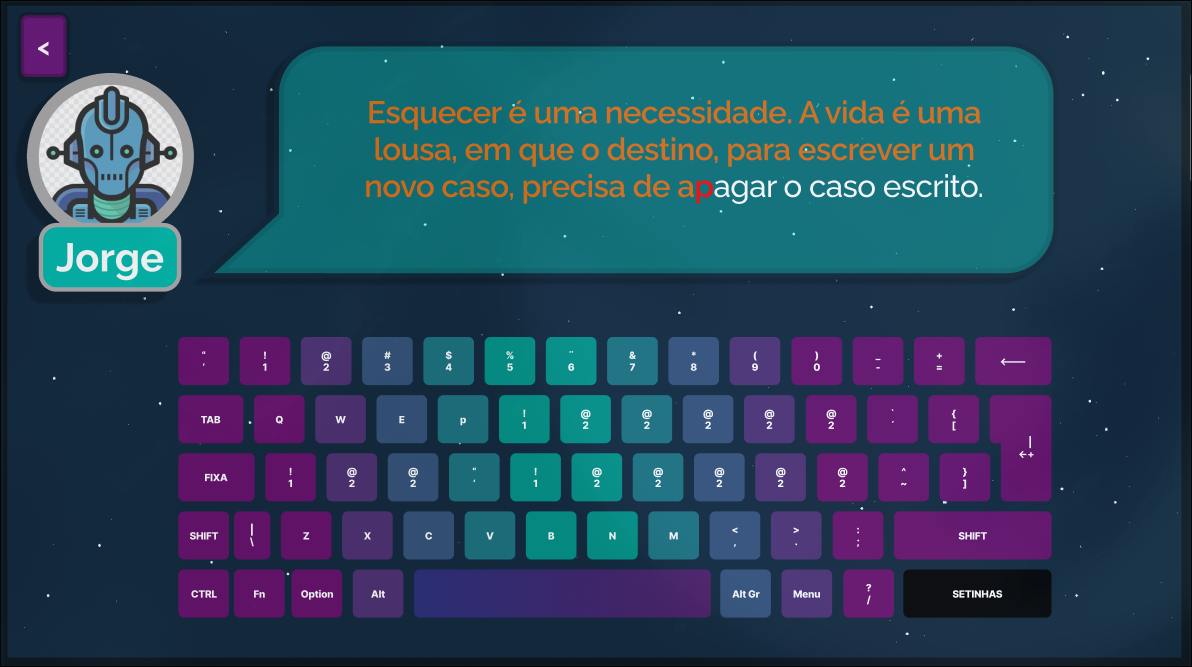
* getFrase() : String
  + A ideia do método getFrase é gerenciar o envio de mensagens para o front contendo as frases que o aluno irá digitar. Ela recupera a próxima frase com base no id armazenado, usando o método getProximaFrase da classe TextoJogavel. Caso o id da frase atual seja -1, deve-se primeiro carregar o texto para a classe TextoJogavel através do método carregaTexto e em seguida retornar a primeira frase do texto (atualizado consequentemente o id da frase atual para 0, ou seja, a frase exibida no front-end é a frase 0).
  + A cada vez que o método é chamado, getFrase atualiza o id e retorna a frase correspondente.
  + Uma vez que o texto tenha terminado(não hajam mais frases), o método retonra uma string vazia (“”).

**Detalhes Gerais:**

* O construtor da classe ControladoraJogar deve inicializar o idFraseAtual como -1. Isso é feito para que, na primeira requisição do front para recuperar as frases, o texto seja carregado de ControladoraTextoAux. Caso a ControladoraTextoAux retorne um erro (não cabe no escopo dessa sprint, mas ocorrerá no futuro, então é importante tratar), deve retornar um erro que avisa ao usuário que não foi possível carregar o texto, e que possivelmente o professor ainda não escolheu o texto a ser usado.
* O iToken para fim de verificações de cadastro ficará localizado no arquivo .env.

# **5. FRONT-END**

Quanto ao que deve ser implementado pelo front, serão criadas as telas desenvolvidas no figma correspondentes a “Jogar”. Segue a figura abaixo.



###### **Figura 5 :**Tela do caso de uso Jogar

## 

## **5.1 FLUXO DE TELAS**

No caso da sprint atual, o único fluxo entre telas que existirá é o redirecionamento da tela Jogar para a tela de Relatório Individual quando o texto terminar de ser digitado. Quanto ao funcionamento da página, temos:

* Na inicialização da tela, devem ser carregados o avatar e o nome do aluno.
* Além disso, na tela de iniciação, devem ser carregadas as duas primeiras frases do texto a serem digitadas.
* A cada tecla digitada:
  + Se o caractere estiver incorreto, o buffer que segura os caracteres errados deve mostrar o caractere digitado.
  + Se o caractere digitado estiver correto, o caractere digitado é removido da frase não digitada e movida para a frase digitada.
* Quanto ao “miniteclado”, ele deve colorir de forma diferente o próximo caractere a ser digitado. Caso seja digitado um caractere errado, ele deve ser colorido de forma a mostrar o erro e qual o caractere correto.

A cada vez que uma frase terminar de ser digitada, o front-end precisa:

* Trocar as frases de lugar de forma que a frase atual seja a frase que estava sendo esperada para ser digitada;
* Trocar a frase que foi digitada de lugar com a frase antiga;
* Carregar uma nova frase a partir do backend;
* Enviar as estatísticas relacionadas a frase que foi digitada para o back;
  + Enquanto cada frase é digitada, o front-end conta quantos caracteres foram digitados errado, qual era o tamanho da frase que foi digitada e quanto tempo o usuário gastou para digitar;
* Quando não houver mais frases a serem enviadas, o back-end enviará uma string vazia (“”).

Quando chegar ao fim do texto, redireciona para a página de visualização de resultados.

### Dica de implementação

Como o teclado virtual ocorre de forma a gerar poucos impactos na digitação e funcionamento do fluxo de texto, pode-se pensar em montá-lo separadamente do resto da página: um componente a parte que se preocupa com o funcionamento do teclado.

## **5.2 END-POINTS**

Os endpoints a serem utilizados pelo front podem ser encontrados dentro da área “endpoints” do capítulo “Backend” deste documento.

## **5.3 BANCO DE DADOS**

Para a sprint atual, a tarefa do banco de dados será uma extensão do caso de uso passado. O objetivo é modelar uma tabela **avatar** que contenha as colunas **id**, **nomeAvatar** e **avatar**. A coluna avatar deve ser capaz de armazenar os avatares dos alunos. Caso seja necessário adicionar outras colunas ao banco de dados para tornar possível a recuperação das imagens, sintam-se livres para criar.

Fica também a cargo de quem estiver modelando o banco de dados encontrar novas imagens e exemplos de avatares para complementar a ambientação do jogo. Tentem encontrar avatares que sejam razoavelmente parecidos com a ambientação do jogo.