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

CAMPUS FLORESTAL



**REFINAMENTO DA SPRINT 02**

**CSU04 e CSU05**

GUILHERME BROEDEL ZORZAL - 5064

ARTHUR FERNANDES BASTOS - 4679

Florestal - MG

Outubro - 2024

**SUMÁRIO**

[**1. INTRODUÇÃO 3**](#_xvx7uekzbnds)

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

[CSU04: Autenticar Professor 4](#_9gw7k2rbveac)

[CSU05: Cadastrar Professor 5](#_try2t8bdhd2y)

[**3. EXPLICAÇÕES GERAIS E ESCLARECIMENTOS 6**](#_uczxuu75cxw8)

[**4. ESTRUTURA GERAL 6**](#_d263pevrteqc)

[**5. BACK-END 6**](#_i3yyqqcljtlj)

[5.1 Professor 7](#_8q4ql9fo76tl)

[5.2 EndPointProfessor 7](#_16dfec75mrjn)

[Figura 1: Classe End-Point do Professor para guardar as rotas 7](#_6cnd3dv3y2hl)

[5.3 ControladoraProfessor 9](#_vzkalwvbioro)

[Figura 2 : Classe controladora para classe Professor 9](#_ope7o99589xv)

[5.4 AcessoBancoGeral 9](#_89ip0ebu5uh3)

[Figura 3 : Classe de Acesso ao Banco de dados 10](#_ux81zc1pm1l7)

[**6. FRONT-END 11**](#_reoritbgbhi1)

[Figura 4 :Tela de Login Professor 11](#_c7y9q4vekma4)

[Figura 5 : Tela de cadastro de Professor 12](#_d5am0t2u4hkm)

[Figura 6: Tela Área do Professor 12](#_xe6ux3i37vag)

[Figura 7 : Pop up do Itoken 12](#_mk1bha8balnx)

[6.1 FLUXO DE TELAS 12](#_jp4nrwv9y752)

[6.2 END-POINTS 13](#_hgwe0x4qf8ds)

# 

# 

# **1. INTRODUÇÃO**

O objetivo do seguinte documento consiste em apresentar especificações relacionadas ao caso de uso abordado na segunda 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 de 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 CSU04: Autenticar Professor e o CSU05: Cadastrar Professor. Sua especificação, proposta na documentação dos casos de uso em ESOF I, pode ser conferida abaixo. Note que os casos de usos quase não sofreram 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 realizar o cadastro e o login do professor. A documentação do caso de uso pode ser encontrada abaixo.

| CSU04: Autenticar Professor **Nome:** Autenticar professor.  **Ator Primário:** Professor  **Sumário:** Ser capaz de autenticar o professor previamente cadastrado.  **Pré-condições:**   * O usuário deve ter uma conta registrada no sistema. * O sistema deve estar conectado ao banco de dados de usuários. * O sistema deve estar operacional e acessível   **Fluxo Principal:**   1. O usuário seleciona a opção "Entrar como professor" 2. O sistema chama o serviço de autenticação pedindo login e senha 3. O usuário digita o respectivo login e senha 4. O sistema acessa o banco de dados e verifica se a autenticação é válida 5. Autenticação é validada e o sistema exibe as opções do menu do professor 6. Fim do caso de uso   **Fluxo Alternativo (3): Usuário selecionou a opção cadastrar professor**  a. O usuário seleciona a opção cadastrar professor.  b. O sistema dispara a ação para cadastrar o professor.  c. Retorna ao passo 2  **Fluxo Alternativo (4): Erro na autenticação**  a. O sistema pede para o professor digitar novamente o usuário e a senha  b. O professor digite novamente o usuário e a senha  **Fluxo de Exceção (4): Falha no acesso ao sistema**  a. O sistema tenta acessar o banco de dados que não se encontra presente, exibe para o usuário a falha e exibe um botão para voltar ao menu principal.  b. O usuário seleciona o botão de voltar.  **Pós-condições:**   * O usuário estará autenticado e capaz de realizar suas funções |
| --- |

# 

| CSU05: Cadastrar Professor **Nome:** Cadastrar Professor.  **Ator primário:** Professor.  **Sumário:** O professor deverá ser capaz de criar uma conta no sistema, que ficará associada a seus textos e relatórios.  **Fluxo principal:**   1. O sistema pede que o usuário preencha as seguintes informações: um email, um nome, uma senha, e o iToken 2. O usuário preenche as informações solicitadas e pressiona um botão de confirmação; 3. O sistema cria a conta de professor e direciona o usuário para a tela de login   **Fluxo de Exceção (2): campo faltando**   1. O sistema notifica o usuário de que um campo está faltando; 2. O caso de uso recomeça do passo 1.   **Fluxo de Exceção (2): confirmação da senha não é igual a senha**   1. O sistema notifica o usuário de que a senha e sua confirmação não são iguais; 2. O caso de uso recomeça do passo 1. |
| --- |

# 

# **3. EXPLICAÇÕES GERAIS E ESCLARECIMENTOS**

Uma explicação importante acerca do funcionamento do sistema é como ocorrerá o uso do sistema. No sistema original, desenvolvido em ESOF I, a ideia seria uma aplicação de rede. Em outras palavras, existiriam vários alunos utilizando o computador. Todos eles fariam requisições para um mesmo computador, o do professor, e a partir dele seriam recuperados os dados dos usuários, textos, etc.

Após a redução de escopo, o fluxo continua parecido, com a exceção de que agora haverá apenas um único computador, que será utilizado tanto pelo aluno quanto pelo professor. Isso facilitará o uso do sistema e o desenvolvimento do mesmo

# **4. ESTRUTURA GERAL**

Uma das mudanças que deverá ser realizada na estrutura geral do sistema é a criação de um arquivo .env com as rotas de cada sistema: Um para o back, outro para o front e outro para o banco de dados. O arquivo deve conter as rotas por onde serão acessadas as informações do front, do back e do banco de dados (por exemplo localhost:3000 para o front). Essa decisão foi tomada para que se possa mudar o sistema com facilidade no futuro para utilização do sistema em diferentes máquinas.

Atualizações e correções

Criação do arquivo .env e atualização das rotas no que foi desenvolvido.

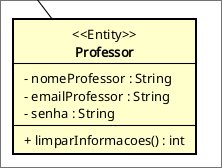
# **5. BACK-END**

O backend, na sprint atual, seguindo o diagrama de Classes disponível no arquivo de docx na mesma pasta, ficará responsável pela implementação de algumas classes e relações envolvendo o Professor, além de criar uma busca de validação com base no email do professor no banco de dados (caso seja encontrado, o email deve ser armazenado esse email na classe professor).

### 5.1 Professor

A classe professor é uma classe relativamente simples. Sua principal função é armazenar os dados do professor uma vez que ele esteja logado no sistema. Os atributos de professor só estarão preenchidos caso o professor esteja logado no sistema.

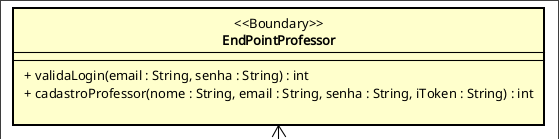
Essa classe deverá possuir getters e setters para os atributos, além de um método limparInformacoes() que remove as informações do professor da classe. Isso é feito para impedir que, uma vez que o professor esteja logado no sistema, o aluno seja capaz de acessar a área do professor



**Figura 1 : Classe Professor**

### 5.2 EndPointProfessor

A classe EndPointProfessor é a classe responsável por receber as requisições enviadas pelo front 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 “ControladoraProfessor”.



###### **Figura 1:** Classe End-Point do Professor para guardar as rotas

Os campos de retorno dos JSONs devem ser os mesmos campos utilizados nas classes. Por exemplo, o nome do aluno deve ficar localizado no campo “nomeProfessor” do JSON.

Os endpoints que devem existir serão:

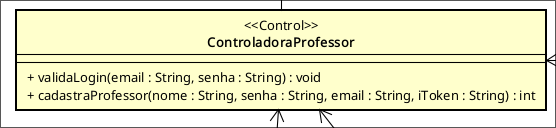
* /professor
  + /login
    - POST (senha: String, emailProfessor: String): verifica a autenticação e guarda as informações no banco de dados e na classe professor. Deve retornar um JSON com os seguintes campos:
      * field: o que está dando errado (login, banco de dados, etc)
      * message: o que aconteceu (credenciais inválidas, etc)
      * IMPORTANTE: não deve retornar informações que prejudiquem a segurança. Por exemplo (field: senha, message: a senha está errada). Isso não deve ser retornado, e sim (field: login, message: as informações de login estão erradas).
      * IMPORTANTE: o campo message será exibido diretamente ao usuário, então deve ser em portugues e bem escrito (não é algo apenas para debug)
      * Em caso de sucesso, basta retornar uma mensagem com status de sucesso (200). Como as informações do professor não serão guardadas no front, não é necessário o envio de nenhuma informação adicional
  + /cadastro
    - POST (nome, senha. email, iToken) : recebe as informações de cadastro do professor e realiza as validações necessárias. Deve realizar as autenticações necessárias e retornar as informações no mesmo padrão que /login GET, com os campos field e message
      * Em caso de sucesso, basta retornar uma mensagem com status de sucesso (200). Como as informações do professor não serão guardadas no front, não é necessário o envio de nenhuma informação adicional
  + /limpar
    - GET: Apaga as informações do professor da classe Professor. Isso é feito para que o aluno não possa logar no sistema como professor

A ideia das verificações de autenticação na parte do backend é a seguinte: Quando forem realizadas funcionalidades relacionadas ao professor (cadastro de texto, deleção de texto, etc) elas só poderão ser realizadas se os dados do professor estiverem na classe professor.

O iToken para fim de verificações de cadastro ficará localizado no arquivo .env

### 5.3 ControladoraProfessor

A classe “ControladoraProfessor” vai comandar as ações de validar o login de um professor e/ou cadastrar um novo. A ideia dessa classe é ter os gets e sets, onde ela precisa conseguir a informação de login de um professor recebendo os dados da requisição do EndPointProfessor, receber os dados da classe “Professor” e através da função “validarLogin” comparar e validar o login de Usuário, além de através da classe “AcessoBancoGeral” e deve conseguir enviar os dados para o cadastro de um novo professor. Fazer respectivos Getters e Setters.



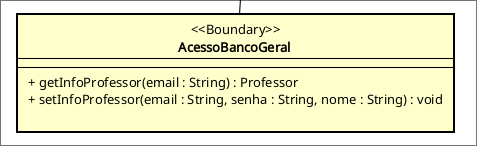
###### **Figura 2 :** Classe controladora para classe Professor

Quanto aos métodos a serem criados:

* **ValidaLogin(email : String, senha : String) : int**  
  Método responsável por receber o email e senha do professor e pesquisar no banco(AcessoBancoGeral) de dados para a validação, retornando um JSON com a estrutura combinada (field e message)
* **cadastraProfessor(nome :String, senha : String, email : String, iToken : String) : int**  
  Método utilizado para receber os dados de um novo cadastro de professor e enviá-los para serem salvos no banco de dados (AcessoBancoGeral)

### 5.4 AcessoBancoGeral

A classe de acesso ao banco de dados tem a função de armazenar os dados de professores que já têm cadastro e armazenar novos cadastros que vão ser requisitados, como método apenas o get e set para preenchimento dos dados de cadastro e login é suficiente.



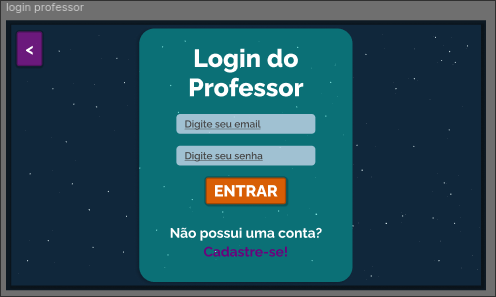
###### **Figura 3 :** Classe de Acesso ao Banco de dados

# 

# **6. FRONT-END**

Quanto ao que deve ser implementado pelo front, serão criadas as telas desenvolvidas no figma correspondentes a: “Login Professor” e “Cadastro de Professor”. Alterações feitas: Trocar nome por email na tela de Login;

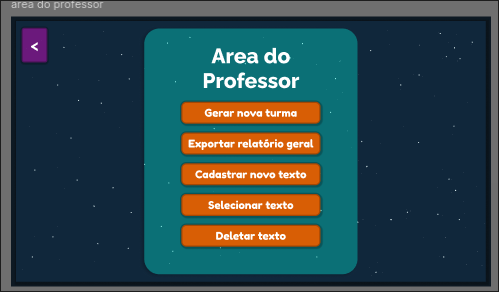
Não terá cadastro na área do professor e utilizará o Itoken para cadastro agora. O link do design pode ser encontrado na área inicial do clickup. Tais telas também podem ser visualizadas abaixo.



###### **Figura 4 :**Tela de Login Professor



###### **Figura 5 :** Tela de cadastro de Professor

****

###### **Figura 6:** Tela Área do Professor

****

###### **Figura 7** : Pop up do Itoken

A tela área do professor deverá ser implementada, mas ainda não terá nenhuma funcionalidade associada

## **6.1 FLUXO DE TELAS**

O fluxo das telas deve ser o seguinte:

* Login Professor:
  + O botão “Entrar” deve ser capaz de realizar a verificação da autenticação do professor
  + O botão de voltar deve redirecionar para a tela inicial
  + O botão Cadastre-se deve redirecionar para a página Cadastro do Professor
* Cadastro Professor
  + O botão de voltar deve redirecionar para a tela de login
  + O botão entrar deve abrir o Pop-up do iToken
  + O botão de Validar deve tentar realizar o cadastro
* Área do professor
  + O botão área do professor deve redirecionar para a tela de login
  + Os outros campos não precisam fazer nada na sprint atual

Espera-se que o sistema seja robusto, capaz de tratar erros que possam ocorrer no sistema e lidar com situações como falta de preenchimento dos campos obrigatórios.

IMPORTANTE: Na parte dos campos obrigatórios, espera-se que sejam utilizado um html forms para verificação automática de campos obrigatórios

## **6.2 END-POINTS**

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

6.3 BANCO DE DADOS

Para essa sprint, a única coisa que será exigida do banco de dados é uma tabela Professor, que contenha as colunas nome, email e senha. Todos os três campos devem ser strings (a escolha do tipo específico, como varchar, text e outras alternativas ficam a critério do desenvolvedor). A chave primária deverá ser o email.

Após o desenvolvimento do diagrama relacional, é necessário gerar o script SQL correspondente no MySQL Workbench. Os produtos finais devem ser: o diagrama relacional (o arquivo de edição do workbench), a imagem do diagrama produzido e o script SQL (arquivo .sql).

OBS.: Para gerar o SQL, existe a opção de *forward engineering* dentro do MySQL Workbench. Em caso de dúvidas, consultem os Dev Seniors.