FullStack [Education]

Módulo 3 - Projeto Avaliativo

SUMÁRIO

1 INTRODUÇÃO

O **LAB365 Developer**, empresa líder no segmento tecnológico de gestão estudantil, está com um novo projeto intitulado **labPCP**, um software audacioso para gestão de processos educacionais. O seu perfil chamou a atenção dos gestores, para criar a aplicação completa do software, que deverá ser construída utilizando o *framework* **Angular**, **Spring e PostgreSQL.**

Está animado para iniciar o desenvolvimento?

Leia **atentamente** os itens abaixo para ficar por dentro das regras de negócio e rotas/telas que devem ser criadas na aplicação. Lembre-se também de **ler atentamente as regras** de entrega do projeto.

2 DEFINIÇÃO DAS SQUADS

Definição das squads para realização do projeto. Cada squad possui 5 membros, sendo um deles o Product Owner.

Squad 1:

- Nome: PREENCHER
- Product Owner: **David Alves Dutra**
- Integrantes:

Nome	GitHub	E-mail do Trello
David Alves Dutra	dutra357	david_a_dutra@estudante.sesisenai.org.br
Barbara Idaerla Santos Calderon	barbaracalderon	barbara_calderon@estudante.sesisenai.org.br
Luís Pedro Trindade	LuisPedroCTrindade	luis_trindade@estudante.sesisenai.org.br

Squad 2:

- Nome: ***Breakpoint***
- Product Owner: **Gabriela Silva**
- Integrantes:

Nome	GitHub	E-mail do Trello
Gabriela Silva	gabrielagabriela/	gabriela_silva19@estudante.sesisenai.org.br
Arthur Ferreira Borba	ArthurFerreiraBorba	arthur_f_borba@estudante.sesisenai.org.br
Andrew Matheus Nobrega Cabral	andrewmatheus	andrew_cabral@estudante.sesisenai.org.br
João Guilherme Ito Cypriano	ajjaum/	joao_cypriano@estudante.sesisenai.org.br

Squad 3:

- Nome: PREENCHER
- Product Owner: **Leandro da Silveira Dias**
- Integrantes:

Nome	GitHub	E-mail do Trello
Leandro da Silveira Dias	leandroCodeDev	leandro_sd@estudante.sc.senai.br
Suene Souza	SueneVS	suene_souza@estudante.sesisenai.org.br
Ederson Sparenberger	esparen	ederson_sparenber@estudante.sesisenai.org.br
João Victor de Souza Olivo	joaoxn	joao_v_olivo@estudante.sesisenai.org.br

Squad 4:

- Nome: PREENCHER
- Product Owner: **Scheila Stihler**
- Integrantes:

Nome	GitHub	E-mail do Trello

Scheila Stihler	scheiladev	scheila_stihler@estudante.sesisenai.org.br
Antonio Santiago Ziliotto Martinez	antonioszm	antonio_martniez@estudante.sesisenai.org.br
Leonardo Vieira	LeoVieira93	leonardo_vieira13@estudante.sesisenai.org.br

Squad 5:

- Nome: PREENCHER
- Product Owner: **Ana Helena Fernandes Peres**
- Integrantes:

Nome	GitHub	E-mail do Trello
Ana Helena Fernandes Peres	**anahperes**	**ana_peres@estudante.sesisenai.org.br**
Vitor Schultz Sertã Machado	vXultz	suzi_harima@estudante.sesisenai.org.br
Leonardo Madeira Barbosa da Silva	devleo-m	leonardo_mb_silva@estudante.sesisenai.org.br
Heverton Luan Roieski	hevertonlr	heverton_roieski@estudante.sesisenai.org.br
Suzi Fortunata Harima	SuziHarima	suzi_harima@estudante.sesisenai.org.br
Cheryl Henkels de Souza	CherylHenkels/	cheryl_souza@estudante.sesisenai.org.br

Squad 6: ASSÍNCRONA

- Nome: PREENCHER
- Product Owner:
- Integrantes:

Nome	GitHub	E-mail do Trello
Bruno Miguel Corrêa	CorreaBrunoMiguel	bruno_miguel-correa@estudante.sesisenai.org.br
Gabriela Almeida Silva		gabriela_silva33@estudante.sesisenai.org.br
Breno Dal Bello Rippel		
James Garcia Lucas	JamesLucas777	james_lucas@estudante.sesisenai.org.br
Lucas Antonio Canuccio		
Gustavo Rafael Dos Santos	gustavo93santos	gustavo_santos55@estudante.sesisenai.org.br
Gabriel Ferreira	Gabrielkamfpp	gabriel_ferreira35@estudante.sesisenai.org.br
Pamela Vieira da Silva	panhavsilva	pamela_vs@estudante.sc.senai.br
Marcos Vinicios Dias dos Santos	marcosvnDias	marcos_vd_santos@estudante.sesisenai.org.br
Michel da Rocha Vieira	micheldrv	michel_r_vieira@estudante.sesisenai.org.br
Felipe Longarai Trisotto	Trisotto	felipe_l_trisotto@estudante.sesisenai.org.br
Daniel Alan Steinheisen	danielSteinheisen	daniel_steinheisen@estudante.sesisenai.org.br
Ketlen Possoli	kpossoli	ketlen_possoli@estudante.sesisenai.org.br

3 ENTREGA E APRESENTAÇÃO

Após o projeto ser finalizado e revisado, **cada membro de cada squad deverá submeter os links** do **GitHub** e **Trello** na tarefa **Módulo 3 - Projeto Avaliativo**, presente na semana 8 do AVA até o dia **04/11/2024** às **22h**.

As squads irão utilizar repositórios GitHub e quadros Trello diferentes, sendo eles:

Squad	Back-End	Front-End
1	**Trello**: Acesse aqui **GitHub**: Acesse aqui	**Trello**: Acesse aqui **GitHub**: Acesse aqui
2	**Trello**: Acesse aqui **GitHub**: Acesse aqui	**Trello**: Acesse aqui **GitHub**: Acesse aqui
3	**Trello**: Acesse aqui **GitHub**: Acesse aqui	**Trello**: Acesse aqui **GitHub**: Acesse aqui
4	**Trello**: Acesse aqui **GitHub**: Acesse aqui	**Trello**: Acesse aqui **GitHub**: Acesse aqui
5	**Trello**: Acesse aqui **GitHub**: Acesse aqui	**Trello**: Acesse aqui **GitHub**: Acesse aqui
6	**Trello**: Acesse aqui **GitHub**: Acesse aqui	**Trello**: Acesse aqui **GitHub**: Acesse aqui

Importante:

1. Não serão aceitos projetos submetidos **após a data limite da atividade**, e, ou **alterados** depois de entregues. Lembre-se de **não modificar** o código no GitHub até receber sua nota e feedback.
2. Não esqueça de **submeter os links no AVA**. Não serão aceitos projetos em que os links não tenham sido submetidos.

3.1 APRESENTAÇÃO

Após o projeto ser finalizado, sua *squad* deverá criar um modelo de apresentação sobre o projeto para apresentar a solução criada na data **06/11/2024** no período das **19h** às **20h30.**

Para a apresentação você precisa organizar um breve *pitch* de até **15 minutos** onde **todos** os membros da *squad* devem apresentar. O *Product Owner* poderá articular a apresentação, introduzindo o problema, soluções e desafios tecnológicos. Na sequência os demais membros podem apresentar cada funcionalidade criada trazendo aspectos técnicos.

Dicas para o *pitch* de apresentação:

1. Crie uma apresentação chamativa que envolve o problema e sua solução
2. Mostre como sua solução é utilizada e quais seus benefícios
3. Defenda o uso da sua solução e o porque ela se diferencia das demais
4. Traga dados técnicos mostrando a codificação e o motivo de sua utilização

Durante a apresentação, os mentores irão estar avaliando o seu comportamento e o seu desenvolvimento técnico do projeto. Utilize este momento para colocar em prática o conhecimento adquirido, bem como sua habilidade de comunicação e argumentação.

4 REQUISITOS DA APLICAÇÃO

A aplicação que deverá ser realizada **em squads** deve contemplar os seguintes requisitos:

1. O sistema deverá ser desenvolvido utilizando os *framework*s Angular, Spring e como banco de dados a tecnologia PostgreSQL.
2. O sistema deve seguir o Roteiro da Aplicação, tanto de BackEnd quanto de FrontEnd
3. Você deverá modelar o layout da aplicação com os formatos, tipografias, cores e organização de layout que achar melhor.
   a. Caso julgue necessário, poderá usar Bootstrap, MUI (Material UI) e/ou outra ferramenta para estilização do *layout*.
4. Você deverá se preocupar com a **usabilidade** da aplicação.
   a. Pense no contraste das cores para não atrapalhar na visualização.
5. Você deverá utilizar o GitHub como versionador de código com repositórios separados para front-end e back-end.
6. Você deverá criar uma documentação readme.md em cada repositório do projeto.
7. Você deverá organizar as tarefas a serem realizadas no projeto, em um quadro no Trello, com a sua squad.

5 ROTEIRO DA APLICAÇÃO FRONT-END

A aplicação deverá conter os requisitos apresentados anteriormente, sendo codificada em **TypeScript** através do *framework* **Angular**, também deverá ser utilizado **markdown** para descrever o projeto no readme.md.

5.1 FORMATO DO SISTEMA

A aplicação deverá seguir o seguinte formato:

1. Ser construída utilizando o *framework* **Angular**:
    a. Utilizar roteamento para gerenciamento de páginas.
    b. Gerenciar o estado global quando necessário.
    c. Utilizar animações de loading e ou transições entre páginas.
        i. Exemplo: Spinner de loading
    d. Utilizar consumo da API ViaCEP para cadastro de endereço.
2. Utilizar favicon, título de página e demais assets.
3. Utilizar o GitHub como versionador de código:
    a. Utilização do padrão baseado em GitFlow com *main*, *develop*, *features* e *releases (release é opcional)*.
    b. Utilização de commits curtos e concisos.
4. Utilizar Trello para organização das tarefas a serem realizadas:
    a. Criar um quadro com estrutura *kanban* (*backlog*, *todo*, *doing*, *blocked*, *review* e *done*) para organização das tarefas.
    b. Criar cartões com tarefas a serem realizadas e arrastar cada um conforme a necessidade.

5. Possuir as seguintes páginas e funcionalidades:
    a. **Página de Login**: Contendo um formulário com campos de email e senha, botões para criar uma conta, entrar na aplicação, e um link para esqueci minha senha.
        i. Crie um mock inicial com alguns usuários para fazer o login. Este mock deve conter usuários com os perfis: Administrador, Docente e Aluno.
        ii. A funcionalidade "esqueceu sua senha" não precisa ser implementada. Caso não seja implementada, utilize um *pop-up*, *toast* ou *alert* para avisar que esta funcionalidade está em construção.
        iii. A funcionalidade "criar uma conta" não precisa ser implementada. Caso não seja implementada, utilize um *pop-up*, *toast* ou *alert* para avisar que esta funcionalidade está em construção.
        iv. O botão de entrar deve direcionar o usuário para a página de Início, caso o e-mail e senha sejam válidos e estejam cadastrados, caso contrário, deve ser sinalizado “Usuário e/ou senha incorretos” através de um *pop-up*, *toast* ou *alert.*

b. **Menu Lateral:** Deverá estar presente em todas as outras páginas, com exceção da página de login, apresentando os seguintes botões de navegação:
    i. O botão de sair deve 'deslogar' e redirecionar o usuário para a página de login.
    ii. Deverá conter botões de opções de navegação para as outras telas do portal:
        1. Início: redireciona para a "*Página de Início*", *item d* ou *item e*, a depender do perfil de acesso do usuário.
        2. Cadastro Docente: redireciona para a "*Página de Cadastro de Docente*", *item f*. Visível apenas para usuários Administradores.
        3. Cadastro de Aluno: redireciona para a "*Página de Cadastro de Aluno*", *item g*. Visível apenas para usuários Administradores.
        4. Cadastro de Turma: redireciona para a "*Página de Cadastro de Turma*", item h. Visível apenas para usuários Administradores e Docentes.
        5. Cadastro de Avaliação: redireciona para a "*Página de Cadastro de Avaliação/Nota*", item i.
        Visível apenas para usuários Administradores e Docentes.
        6. Listagem de Docentes: redireciona para a "*Página de Listagem de Docentes*", item j.
        Visível apenas para usuários Administradores.
        7. Notas: redireciona para a "*Página de Notas do Aluno*", item k.
        Visível apenas para usuários Alunos.
    iii. Um botão para recolher/esconder e expandir/mostrar o menu lateral.
c. **Toolbar:** Deverá estar presente em todas as outras páginas, com exceção da página de login, apresentando o título da página, nome do usuário logado e ícone para representar a foto do usuário.
    i. Opcionalmente poderá ser adicionado outras funcionalidades como *dropdown* e botão de sair, por exemplo.
d. **Página de Início (Administradores e Docentes)**: Deve conter as seguintes funcionalidades:
    i. Uma seção com as estatísticas do sistema. Elas devem representar ao menos a quantidade de alunos cadastrados, quantidade de docentes cadastrados e a quantidade de turmas.
    Você está livre para adicionar novas estatísticas caso ache necessário.
        1. Esta seção somente será apresentada para usuários Administradores
    ii. Deverá existir um campo para buscar alunos pelo nome, telefone ou e-mail.
    iii. A listagem de alunos deve mostrar um *card* para cada um com ao menos um ícone, nome completo, idade, contato (telefone ou e-mail) e um botão:
        1. Caso o perfil do usuário seja Administrador: o botão possuirá o texto "*ver mais*" e, ao ser clicado, o usuário deve ser redirecionado para a "*Página de Cadastro de Aluno*", funcionalidade especificada no *item g*, que deve ser aberta, com os botões de edição, deleção e salvamento ativados. Nesta página, o usuário poderá editar as informações ou deletar um aluno caso o

mesmo não possua turmas e avaliações vinculadas. Para isso, a tela deve preencher os campos com as informações daquele aluno.

2. Caso o perfil do usuário seja Docente: o botão possuirá o texto "*lançar nota*" e, ao ser clicado, o usuário deve ser redirecionado para a "*Página de Cadastro de Avaliação/Nota*", funcionalidade especificada no *item i*.

e. **Página de Início (Alunos)**: A tela inicial para o aluno deverá conter três sessões:
   i. Sessão Minhas Avaliações: Nesta sessão serão exibidas as últimas 3 avaliações realizadas pelo aluno, sendo exibido o nome da avaliação, a matéria e a data de realização. O clique em uma avaliação redireciona o usuário para a "*Página de Notas do Aluno*", funcionalidade especificada no *item k*.
   ii. Sessão Meu Curso: Nesta sessão será exibido o curso em que o aluno está matriculado.
   iii. Sessão Cursos extras: Sessão onde mostrará todos os cursos que o aluno poderá fazer para complementar sua grade curricular.

f. **Página de Cadastro de Docente:** Deve conter um formulário para cadastro de Docente com botões para editar, deletar e salvar.
   i. Essa tela só deverá ser acessada pelo usuário do tipo administrador.
   ii. Durante o cadastro, os botões de editar e deletar devem ficar desativados (desabilitados).
   iii. Campos do formulário:
      1. Nome Completo: Obrigatório, com máximo e mínimo de 64 e 8 caracteres, respectivamente.
      2. Gênero: Obrigatório com *dropdown* de opções pré-definidas.
      3. Data de Nascimento: Obrigatório, data válida.
      4. CPF: Obrigatório com o formato 000.000.000-00
      5. RG com órgão expedidor: Obrigatório, com máximo de 20 caracteres.
      6. Estado Civil: Obrigatório com *dropdown* de opções pré-definidas.
      7. Telefone: Obrigatório com o formato (99) 9 9999-9999
      8. E-mail: Obrigatório e com validação do formato de email.
      9. Senha: Obrigatório sendo no mínimo 8 caracteres.
      10. Naturalidade: Obrigatório, com máximo e mínimo de 64 e 8 caracteres, respectivamente.
      11. Endereço: Cep, Cidade, Estado, Logradouro, Número, Complemento, Bairro e Ponto de Referência.
      12. Matérias que o docente ministra: Obrigatório; campo multi-select (pode receber mais de uma seleção) com opções de matérias pré-setadas.
   iv. Deverá utilizar a API do ViaCEP para buscar os dados de endereço.
   v. Deverá verificar os dados informados antes de cadastrar.
   vi. Deverá criar um identificador único para cada docente cadastrado.
   vii. Deverá apresentar uma confirmação ao salvar. Exemplo: *toast* ou *alert*.

g. **Página de Cadastro de Aluno:** Deve conter um formulário para cadastro de Aluno com botões para editar, deletar e salvar.
    i. Essa tela só deverá ser acessada pelo usuário do tipo administrador.
    ii. Durante o cadastro, os botões de editar e deletar devem ficar desativados (desabilitados).
    iii. Campos do formulário:
        1. Nome Completo: Obrigatório, com máximo e mínimo de 64 e 8 caracteres, respectivamente.
        2. Gênero: Obrigatório com *dropdown* de opções pré-definidas.
        3. Data de Nascimento: Obrigatório, data válida.
        4. CPF: Obrigatório com o formato 000.000.000-00
        5. RG com órgão expedidor: Obrigatório, com máximo de 20 caracteres.
        6. Telefone: Obrigatório com o formato (99) 9 9999-99999
        7. E-mail: Não obrigatório e com validação.
        8. Senha: Obrigatório sendo no mínimo 8 caracteres.
        9. Naturalidade: Obrigatório, com máximo e mínimo de 64 e 8 caracteres, respectivamente.
        10. Endereço: Cep, Cidade, Estado, Logradouro, Número, Complemento, Bairro e Ponto de Referência.
        11. Turma: Obrigatório, dropdown/select com a listagem de turmas cadastradas no portal.
    iv. Deverá utilizar a API do ViaCEP para buscar os dados de endereço.
    v. Deverá verificar os dados informados antes de cadastrar.
    vi. Deverá criar um identificador único para cada aluno cadastrado.
    vii. Deverá apresentar uma confirmação ao salvar. Exemplo: *toast* ou *alert*.
h. **Página de Cadastro de Turma:** Deve conter um formulário para cadastro de Turmas com botões editar, deletar e salvar.
    i. Essa tela só deverá ser acessada pelos usuários dos tipos Administradores e Docentes.
    ii. Durante o cadastro, os botões de editar e deletar devem ficar desativados (desabilitados).
    iii. Campos do formulário:
        1. Nome da Turma: Obrigatório, com máximo e mínimo de 64 e 8 caracteres, respectivamente.
        2. Data de Início/Término: Obrigatório, buscando data atual do sistema e liberando para edição.
        3. Horário da Turma: Obrigatório, buscando horário atual do sistema e liberando para edição.
        4. Curso: Obrigatório, dropdown com a listagem de cursos disponíveis para a aplicação;

5. Professor: Obrigatório, dropdown com a listagem de professores cadastrados no portal.
    a. Caso o usuário possua o perfil Administrador, deverão ser listados todos os professores/docentes cadastrados no sistema.
    b. Caso o usuário possua o perfil Docente, o dropdown deverá vir preenchido com o nome do próprio usuário selecionado e não permitir alteração.

iv. Deverá verificar os dados informados antes de cadastrar.
v. Deverá criar um identificador único para cada turma cadastrada.
vi. Deverá apresentar uma confirmação ao salvar. Exemplo: *toast* ou *alert*.

i. **Página de Cadastro de Avaliação/Nota:** Deve conter um dropdown para a seleção de uma turma para a inclusão de uma avaliação e um formulário para cadastro das notas com botões editar, deletar e salvar.
    i. Essa tela só deverá ser acessada pelos usuários dos tipos Administradores e Docentes.
    ii. Durante o cadastro, os botões de editar e deletar devem ficar desativados (desabilitados).
    iii. Campos do formulário:
        1. Professor: Obrigatório, dropdown com a listagem de professores cadastrados no portal.
            a. Caso o usuário possua o perfil Administrador, deverão ser listados todos os professores/docentes cadastrados no sistema.
            b. Caso o usuário possua o perfil Docente, o dropdown deverá vir preenchido com o nome do próprio usuário selecionado e não permitir alteração.
        2. Nome da Matéria: Obrigatório, dropdown contendo as opções de matérias pré-setadas.
        3. Nome da Avaliação: Obrigatório, campo de texto aberto para a inserção do nome da avaliação.
        4. Data da Avaliação: Obrigatório, buscando horário atual do sistema e liberando para edição.
        5. Aluno: Obrigatório, deve ser selecionado em um dropdown, pesquisável pelo nome do aluno, o aluno que receberá essa nota.
        6. Nota Avaliação: Obrigatório, numérico com o valor mínimo e máximo sendo 0 e 10.
    iv. Deverá verificar os dados informados antes de cadastrar.
    v. Deverá criar um identificador único para cada avaliação/nota cadastrada.
    vi. Deverá apresentar uma confirmação ao salvar. Exemplo: *toast* ou *alert*.

j. **Página de Listagem de Docentes:** Deverá possuir uma barra de pesquisa, listagem de docentes cadastrados e botão para abrir os detalhes.
    i. Essa tela só deverá ser acessada pelo usuário do tipo administrador.

ii. A barra de busca deverá buscar o docente pelo nome ou identificador cadastrado.
iii. A listagem de docentes deve mostrar um *card* para cada um com ao menos o número do registro (id), nome completo, contato (telefone ou e-mail) e um botão "*ver mais*".
iv. Ao clicar no botão "*ver mais*", o usuário deve ser redirecionado para a "*Página de Cadastro de Docente*", funcionalidade especificada no *item f*, que deve ser aberta, com os botões de edição, deleção e salvamento ativados. Nesta página, o usuário poderá editar as informações ou deletar um docente caso o mesmo não possua turmas e avaliações vinculadas. Para isso, a tela deve preencher os campos com as informações daquele docente.

k. **Página de Notas do Aluno**: Deverá possuir histórico de avaliações e notas do aluno.
i. Essa tela só deverá ser acessada pelo usuário do tipo aluno.
ii. As informações a serem exibidas são: turma, docente da turma, matéria, nome da avaliação e nota.
iii. O histórico de notas deve mostrar **todas** as avaliações realizadas pelo aluno, sendo mostrados em ordem cronológica.
iv. Dica: Você pode utilizar uma tabela com todas as informações, categorizada por matéria ou turma.

5.2 GITHUB NO FRONT-END

A partir do repositório recebido pela sua squad, indicado pelo Item 3 deste documento, iremos trabalhar utilizando a estrutura de GitFlow. Para isso, crie uma branch *main*, que será inicializada com o projeto de Frontend, produzido no final do Módulo 2, que a squad selecionou como base.
Uma vez inicializada a *main* com o projeto base, crie a branch *develop*, que será utilizada para todos os desenvolvimentos a serem realizados no projeto.
Uma vez criadas as branches *main* e *develop*, a squad poderá iniciar seu desenvolvimento **sempre** criando uma nova branch a partir da develop, realizar seu desenvolvimento nesta branch, e uma vez finalizado, abrir um PR (Pull Request) para a branch *develop* e, quando aprovado, finalizar o merge.
A aprovação do PR deverá ser realizada por, **no mínimo**, um outro desenvolvedor além do dev que abriu o PR; 1 dev + 1 revisor.
Uma vez finalizados todos os desenvolvimentos, deverá ser aberto um PR da branch *develop* para a branch *main*. A aprovação desse PR deverá ser realizada por, **no mínimo**, outros dois desenvolvedores além do dev que abriu o PR; 1 dev + 2 revisor.
São consideradas também nos critérios de utilização do Github a utilização das boas práticas:

- Nomes significativos de branches onde se estão realizando os desenvolvimentos
  ex.: *feature-X*, *integracao-Y*, *testes-unitarios-Z*;

- Mensagens de commits claras e adequadas para as alterações realizadas;

5.3 DOCUMENTAÇÃO README.MD NO FRONT-END

Crie um arquivo README.md no repositório do seu projeto no GitHub, para documentar a sua solução, bem como demonstrar as técnicas e linguagens utilizadas, além do escopo do projeto e como o usuário pode executar o seu sistema.

Algumas dicas interessantes para utilizar na criação do seu portfólio são:

- Criar um nome para o seu software;
- Descrever qual o problema ele resolve;
- Descrever quais técnicas e tecnologias utilizadas. Aqui você também pode inserir alguma imagem ou diagrama para melhor entendimento;
- Descrever como executar;
- Descrever quais melhorias podem ser aplicadas;
- Entre outras coisas.

5.4 TRELLO NO FRONT-END

A partir do Trello recebido pela sua squad, indicado pelo Item 3 deste documento, utilizando a metodologia *kanban* (*backlog*, *todo*, *doing*, *blocked*, *review* e *done*) para realizar a gestão do seu projeto. A partir das informações extraídas deste documento, organize as tarefas a serem realizadas em cartões (*cards*) e os arraste a cada etapa. Note que você pode fazer um paralelo com a metodologia do *GitFlow*, criando cartões com o mesmo nome, por exemplo *feature/create-login-page*, desta forma, o projeto estará organizado e documentado sendo possível rastrear cada etapa e sanar as dúvidas quando necessário. Não esqueça de habilitar o quadro para modo público e colocar o link no readme.md e na submissão no AVA.

Importante: Não utilize o quadro de exercícios para realizar tarefas de projeto. Todos os cards diferentes aos de exercícios serão deletados de forma automática.

5.5 EXEMPLO DE APLICAÇÃO

A aplicação deverá conter os requisitos apresentados anteriormente, sendo codificada em **html**, **css** e **javascript** através do *framework* **Angular**, também deverá ser utilizado **markdown** para o readme.md.
As imagens a seguir demonstram **exemplos da aplicação** que deverá ser desenvolvida. Você poderá **desenvolver igual ao modelo, ou criar um layout diferente** com estilo próprio.

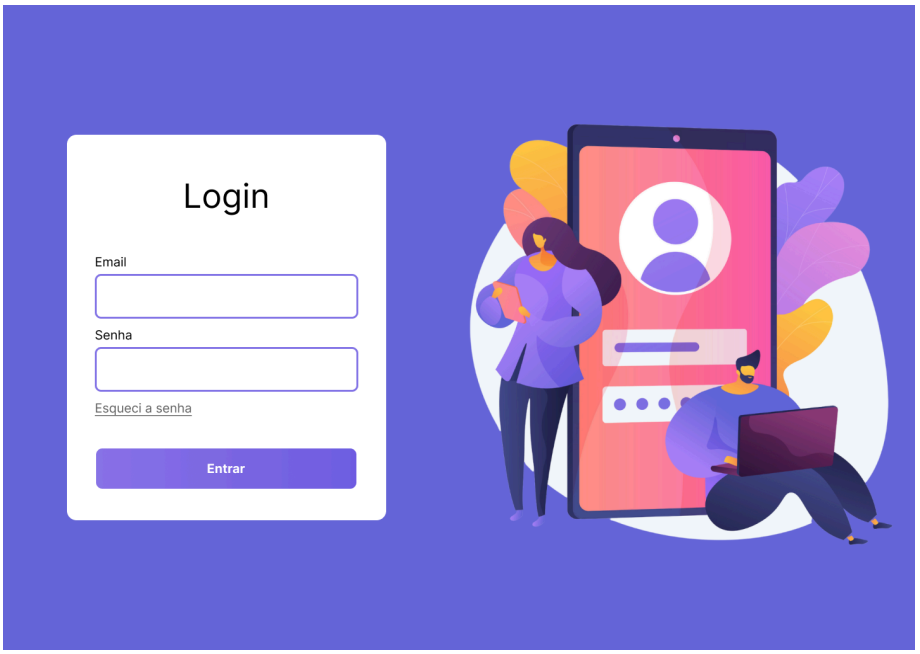

Imagem 1: Página de Login

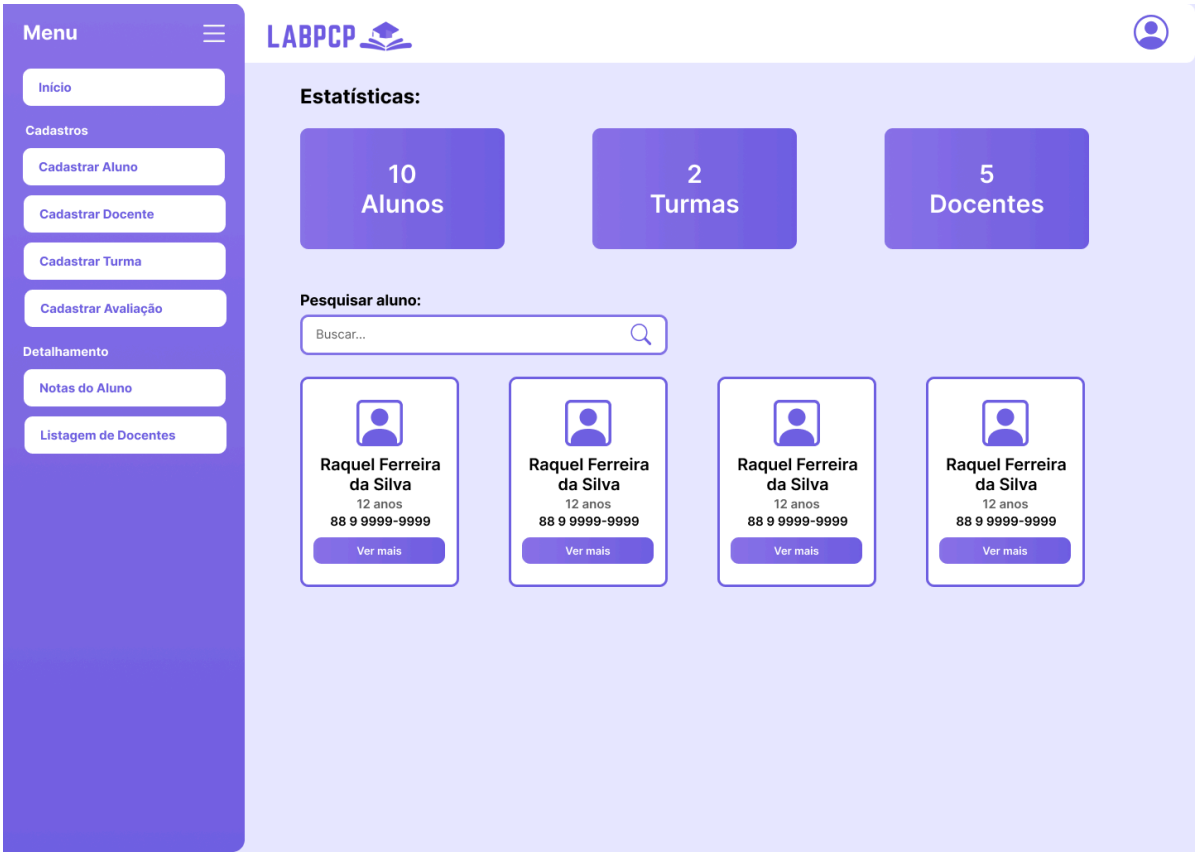

Imagem 2: Página de Início (Estatística e Informações)

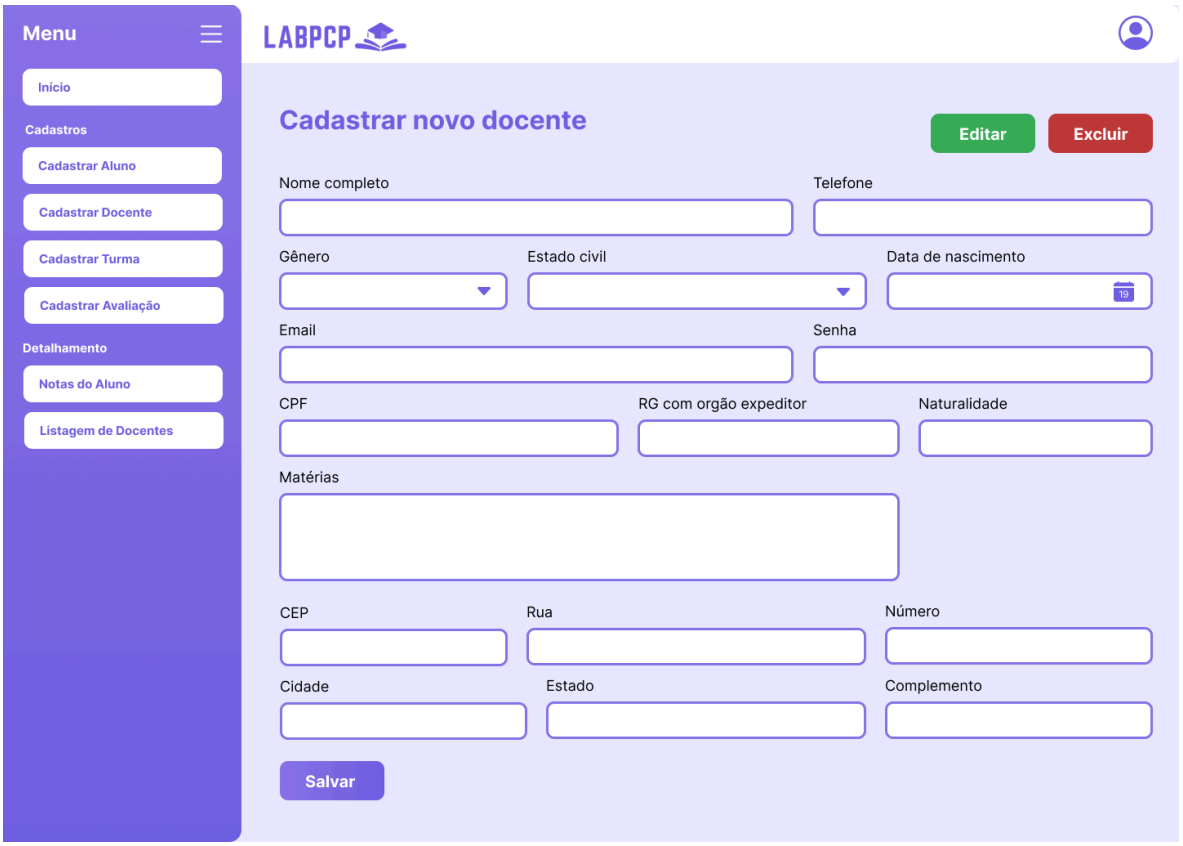

Imagem 3: Página de Cadastro/Edição de Docente

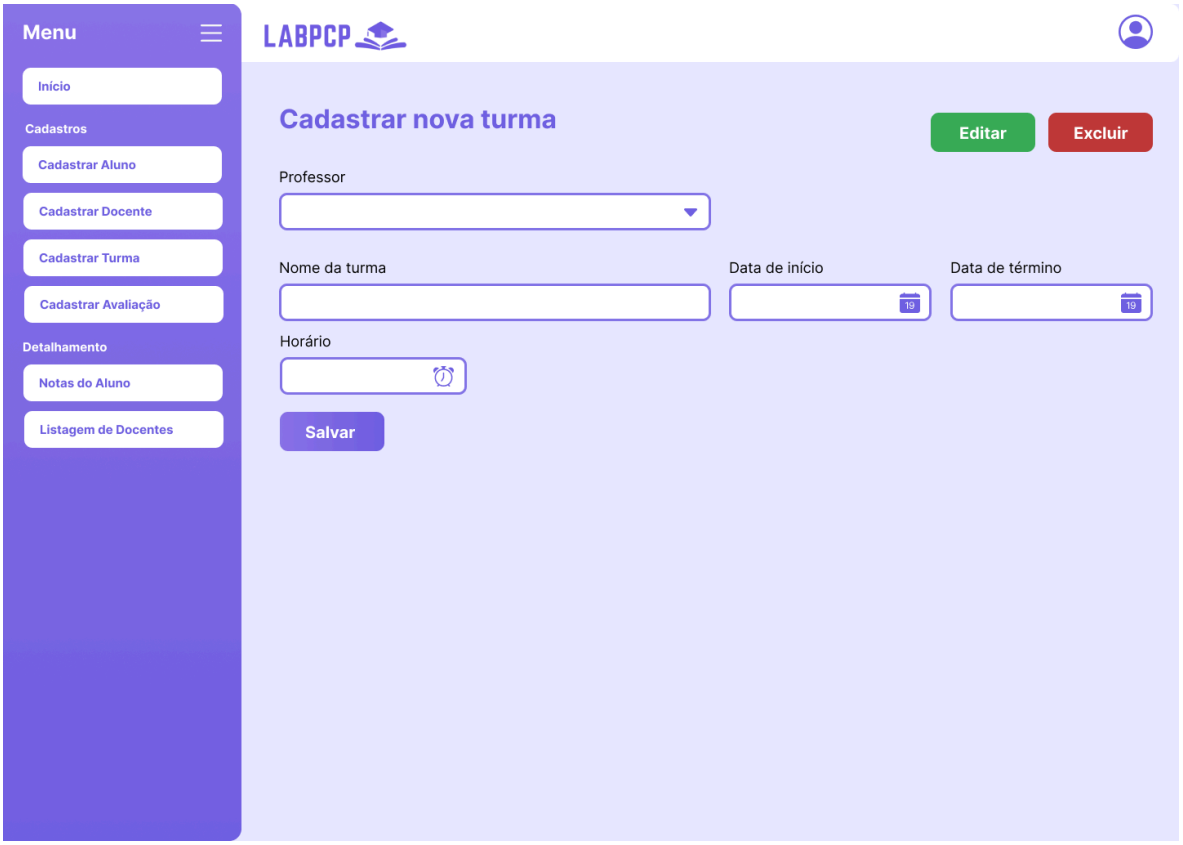

Imagem 4: Página de Cadastro/Edição de Turma

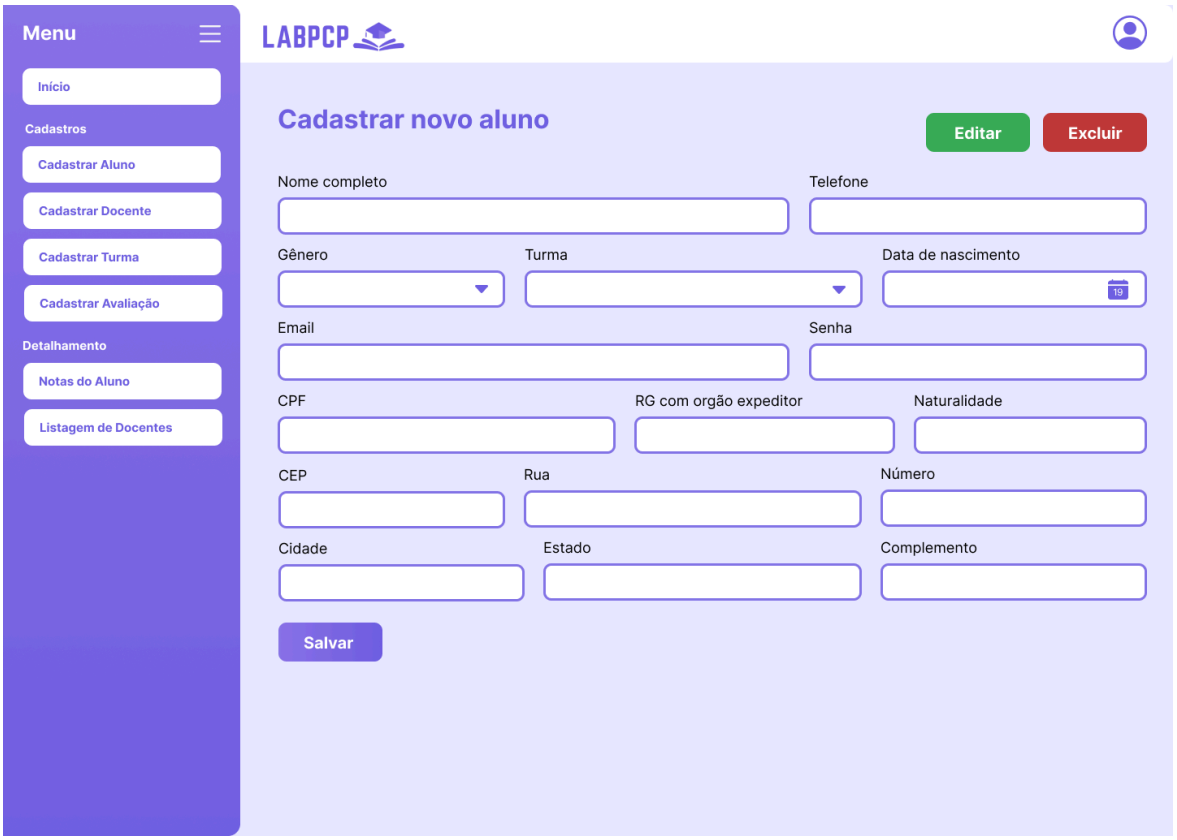

Imagem 5: Página de Cadastro/Edição de Aluno

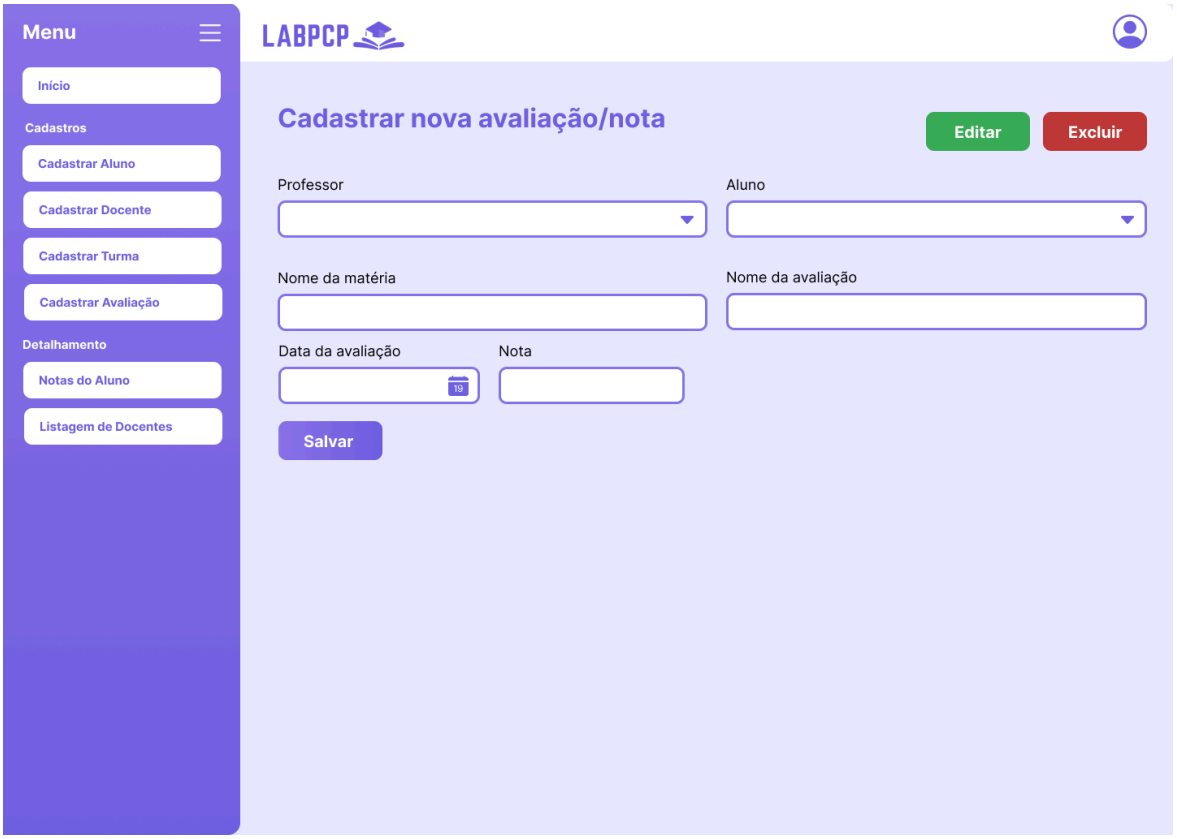

Imagem 6: Página de Cadastro/Edição de Nota/Avaliação

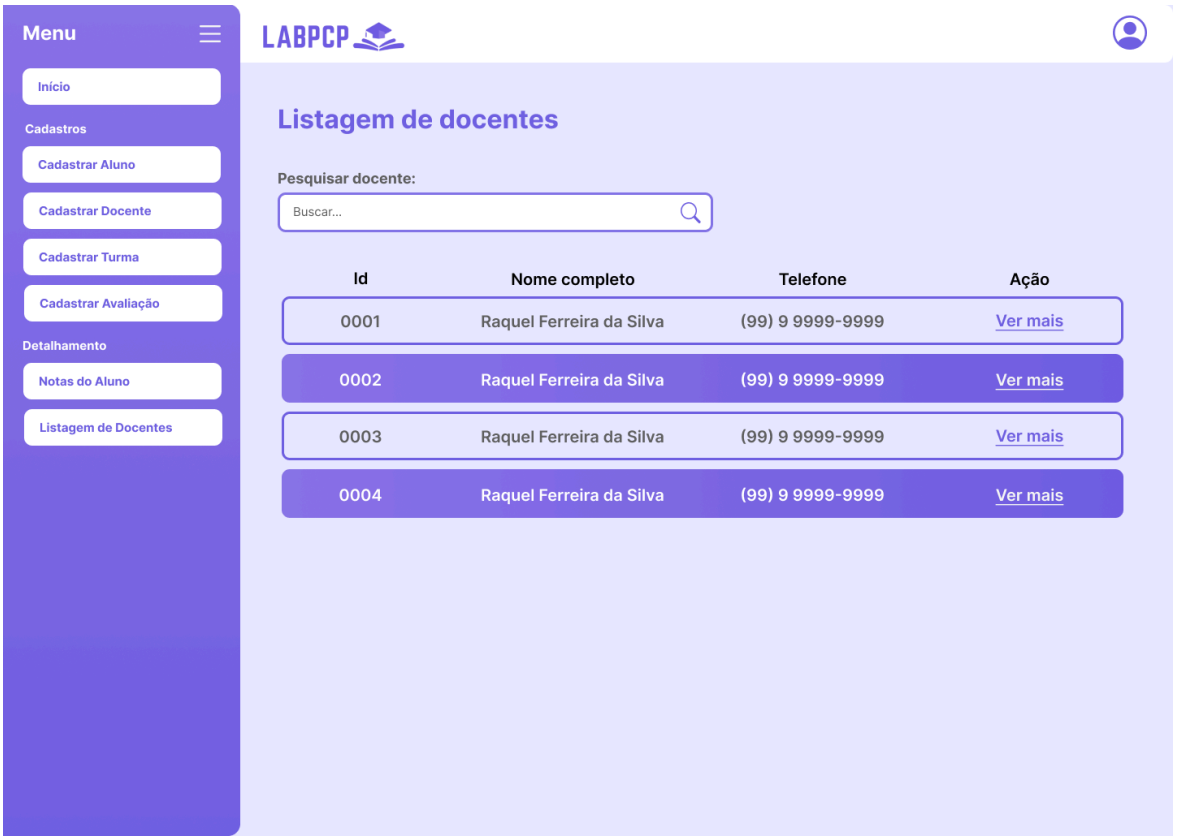

Imagem 7: Página de Listagem de Docente

Imagem 8: Página de Notas do aluno

6 ROTEIRO DA APLICAÇÃO BACK-END

A aplicação deverá conter os requisitos apresentados anteriormente, sendo codificada em **Java** através do *framework* **Spring Boot**, também deverá ser utilizado **markdown** para o readme.md.

6.1 FORMATO DO SISTEMA

A seguir estão especificados os *endpoints* que devem ser criados na aplicação. Vale ressaltar que para cada *endpoint* deve ser fornecido um Token JWT que servirá para acesso aos dados do mesmo, para isso utilize o Spring Security (exceto o endpoint de *login*).
Cobertura de testes unitários: A aplicação deve possuir, no mínimo, 50% de cobertura de testes unitários nas linhas de código da aplicação.
Documentação de API: Documentar os endpoints utilizando o padrão OpenAPI / Swagger.
CI/CD: A aplicação deve possuir um arquivo Dockerfile próprio e um docker-compose contendo instruções para sua execução, bem como de aplicações das quais ela possa depender para que funcione corretamente em qualquer ambiente (por exemplo, banco de dados).

Endpoint de Login

- Login de Usuário (POST /login)
    - Descrição: Permite que um usuário faça login no sistema.
    - Body: JSON contendo as credenciais do usuário (usuario e senha).
    - Respostas Possíveis:
        - Código 200 (OK) - Login bem-sucedido. Retorna um token JWT (JSON Web Token) no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.

Endpoint de Cadastro

- Cadastro de Usuário (POST /cadastro)
    - Descrição: Permite que um novo usuário seja cadastrado no sistema.
    - Body: JSON representando os dados do novo usuário a ser cadastrado.
    - Respostas Possíveis:
        - Código 201 (Created) - Usuário cadastrado com sucesso. Retorna o JSON do usuário criado no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.

        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.

Endpoint de Dashboard

- Página de Início (GET /dashboard)
    - Descrição: Retorna a quantidade de alunos cadastrados, a quantidade de docentes cadastrados e a quantidade de turmas cadastradas.
    - Body: JSON representando os dados a serem exibidos no dashboard.
    - Respostas Possíveis:
        - Código 200 (Ok) - Retorna o JSON com os dados no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.

Endpoints para entidade Docente

- Criar Docente (POST /docentes)
    - Cada docente cadastrado deve inserir também um registro na tabela Usuario, para servir de credencial de acesso do Docente.
    - Body: JSON representando os dados do docente a ser criado.
    - Respostas Possíveis:
        - Código 201 (Created) - Docente criado com sucesso. Retorna o JSON do docente criado no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.
- Obter Docente por ID (GET /docentes/{id})
    - Parâmetros de URL: ID do docente.
    - Respostas Possíveis:
        - Código 200 (OK) - Docente encontrado. Retorna o JSON do docente no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Docente não encontrado.
- Atualizar Docente (PUT /docentes/{id})
    - Parâmetros de URL: ID do docente a ser atualizado.
    - Body: JSON representando os novos dados do docente.
    - Respostas Possíveis:
        - Código 200 (OK) - Docente atualizado com sucesso. Retorna o JSON do docente atualizado no corpo da resposta.

        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Docente não encontrado.
- Excluir Docente (DELETE /docentes/{id})
    - Parâmetros de URL: ID do docente a ser excluído.
    - Respostas Possíveis:
        - Código 204 (No Content) - Docente excluído com sucesso.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Docente não encontrado.
- Endpoint para listar todos os docentes:
    - Listar Docentes (GET /docentes)
        - Respostas Possíveis:
            - Código 200 (OK) - Retorna uma lista de todos os docentes no corpo da resposta.
            - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
            - Código 404 (Not Found) - Não há docentes cadastrados.

Endpoints para entidade Turma

- Criar Turma (POST /turmas)
    - Body: JSON representando os dados da turma a ser criada.
    - Respostas Possíveis:
        - Código 201 (Created) - Turma criada com sucesso. Retorna o JSON da turma criada no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.
- Obter Turma por ID (GET /turmas/{id})
    - Parâmetros de URL: ID da turma.
    - Respostas Possíveis:
        - Código 200 (OK) - Turma encontrada. Retorna o JSON da turma no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Turma não encontrada.
- Atualizar Turma (PUT /turmas/{id})
    - Parâmetros de URL: ID da turma a ser atualizada.

    - Body: JSON representando os novos dados da turma.
    - Respostas Possíveis:
        - Código 200 (OK) - Turma atualizada com sucesso. Retorna o JSON da turma atualizada no corpo da resposta.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Turma não encontrada.
- Excluir Turma (DELETE /turmas/{id})
    - Parâmetros de URL: ID da turma a ser excluída.
    - Respostas Possíveis:
        - Código 204 (No Content) - Turma excluída com sucesso.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Turma não encontrada.
- Endpoint para listar todas as turmas:
    - Listar Turmas (GET /turmas)
    - Respostas Possíveis:
        - Código 200 (OK) - Retorna uma lista de todas as turmas no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Não há turmas cadastradas.

Endpoints para entidade Aluno

- Criar Aluno (POST /alunos)
    - Cada aluno cadastrado deve inserir também um registro na tabela Usuario, para servir de credencial de acesso do Aluno.
    - Body: JSON representando os dados do aluno a ser criado.
    - Respostas Possíveis:
        - Código 201 (Created) - Aluno criado com sucesso. Retorna o JSON do aluno criado no corpo da resposta.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
- Obter Aluno por ID (GET /alunos/{id})
    - Parâmetros de URL: ID do aluno.
    - Respostas Possíveis:

        - Código 200 (OK) - Aluno encontrado. Retorna o JSON do aluno no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Aluno não encontrado.
- Atualizar Aluno (PUT /alunos/{id})
    - Parâmetros de URL: ID do aluno a ser atualizado.
    - Body: JSON representando os novos dados do aluno.
    - Respostas Possíveis:
        - Código 200 (OK) - Aluno atualizado com sucesso. Retorna o JSON do aluno atualizado no corpo da resposta.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Aluno não encontrado.
- Excluir Aluno (DELETE /alunos/{id})
    - Parâmetros de URL: ID do aluno a ser excluído.
    - Respostas Possíveis:
        - Código 204 (No Content) - Aluno excluído com sucesso.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Aluno não encontrado.
- Listar Alunos (GET /alunos)
    - Respostas Possíveis:
        - Código 200 (OK) - Retorna uma lista de todos os alunos no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Não há alunos cadastrados.

Endpoints para entidade Curso

- Criar Curso (POST /cursos)
    - Body: JSON representando os dados do curso a ser criado.
    - Respostas Possíveis:
        - Código 201 (Created) - Curso criado com sucesso. Retorna o JSON do curso criado no corpo da resposta.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.

- Obter Curso por ID (GET /cursos/{id})
    - Parâmetros de URL: ID do curso.
    - Respostas Possíveis:
        - Código 200 (OK) - Curso encontrado. Retorna o JSON do curso no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Curso não encontrado.
- Atualizar Curso (PUT /cursos/{id})
    - Parâmetros de URL: ID do curso a ser atualizado.
    - Body: JSON representando os novos dados do curso.
    - Respostas Possíveis:
        - Código 200 (OK) - Curso atualizado com sucesso. Retorna o JSON do curso atualizado no corpo da resposta.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Curso não encontrado.
- Excluir Curso (DELETE /cursos/{id})
    - Parâmetros de URL: ID do curso a ser excluído.
    - Respostas Possíveis:
        - Código 204 (No Content) - Curso excluído com sucesso.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Curso não encontrado.
- Listar Cursos (GET /cursos)
    - Respostas Possíveis:
        - Código 200 (OK) - Retorna uma lista de todos os cursos no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Não há cursos cadastrados.
- Listar Cursos de um Aluno (GET /cursos?idAluno={idAluno})
    - Parâmetros de Query: ID do aluno a ser filtrado.
    - Respostas Possíveis:
        - Código 200 (OK) - Retorna no corpo da resposta uma lista de todos os cursos em que o aluno em questão está matriculado. Caso o aluno não esteja matriculado em nenhum curso, retorna a lista vazia.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.

        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - O aluno fornecido não existe.

Endpoints para entidade Matérias

- Listar Matérias por Curso (GET /cursos/{id_curso}/materias)
    - Parâmetros de URL: ID do curso.
    - Respostas Possíveis:
        - Código 200 (OK) - Retorna uma lista de todas as matérias do curso no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Não há matérias cadastradas para o curso especificado.
- Criar Matéria (POST /materias)
    - Body: JSON representando os dados da matéria a ser criada.
    - Respostas Possíveis:
        - Código 201 (Created) - Matéria criada com sucesso. Retorna o JSON da matéria criada no corpo da resposta.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
- Obter Matéria por ID (GET /materias/{id})
    - Parâmetros de URL: ID da matéria.
    - Respostas Possíveis:
        - Código 200 (OK) - Matéria encontrada. Retorna o JSON da matéria no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Matéria não encontrada.
- Atualizar Matéria (PUT /materias/{id})
    - Parâmetros de URL: ID da matéria a ser atualizada.
    - Body: JSON representando os novos dados da matéria.
    - Respostas Possíveis:
        - Código 200 (OK) - Matéria atualizada com sucesso. Retorna o JSON da matéria atualizada no corpo da resposta.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.

        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Matéria não encontrada.
- Excluir Matéria (DELETE /materias/{id})
    - Parâmetros de URL: ID da matéria a ser excluída.
    - Respostas Possíveis:
        - Código 204 (No Content) - Matéria excluída com sucesso.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Matéria não encontrada.

Endpoints para entidade Notas

- Listar Notas por Aluno (GET /alunos/{id_aluno}/notas)
    - Parâmetros de URL: ID do aluno.
    - Respostas Possíveis:
        - Código 200 (OK) - Retorna uma lista de todas as notas do aluno no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Não há notas cadastradas para o aluno especificado.
- Criar Nota (POST /notas)
    - Body: JSON representando os dados da nota a ser criada.
    - Respostas Possíveis:
        - Código 201 (Created) - Nota criada com sucesso. Retorna o JSON da nota criada no corpo da resposta.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
- Obter Nota por ID (GET /notas/{id})
    - Parâmetros de URL: ID da nota.
    - Respostas Possíveis:
        - Código 200 (OK) - Nota encontrada. Retorna o JSON da nota no corpo da resposta.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Nota não encontrada.
- Atualizar Nota (PUT /notas/{id})
    - Parâmetros de URL: ID da nota a ser atualizada.
    - Body: JSON representando os novos dados da nota.

    - Respostas Possíveis:
        - Código 200 (OK) - Nota atualizada com sucesso. Retorna o JSON da nota atualizada no corpo da resposta.
        - Código 400 (Bad Request) - Requisição inválida, por exemplo, dados ausentes ou incorretos.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Nota não encontrada.
- Excluir Nota (DELETE /notas/{id})
    - Parâmetros de URL: ID da nota a ser excluída.
    - Respostas Possíveis:
        - Código 204 (No Content) - Nota excluída com sucesso.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Nota não encontrada.

Endpoint para obter Pontuação Total do Aluno

- Criar Turma (GET /alunos/{id}/pontuacao)
    - Respostas Possíveis:
        - Código 200 (OK) - Pontuação calculada com Sucesso. Retorna o JSON com o campo “pontuação” preenchido.
        - Código 401 (Unauthorized) - Credenciais inválidas. O usuário não está autorizado a acessar o sistema.
        - Código 404 (Not Found) - Aluno não encontrado.

6.2 GITHUB NO BACK-END

Utilizar o repositório Git informado no item 3 para controle de versionamento do projeto. Siga as diretrizes de boas práticas de uso do Git abaixo:

- Os integrantes das squads deverão seguir commits com descrição sucinta e direta e, de preferência, com poucos arquivos alterados por commit.
    - Prefira mensagens como “implementa X” ou “corrige Y”, em vez de “X foi implementado” ou “Y foi corrigido”.
- Crie uma branch “develop” para concentrar os merges enquanto a equipe estiver desenvolvendo as funcionalidades.
- Utilizar *feature branches* a partir da *develop* para cada tarefa.
    - Não excluir a branch após o pull request concluído.
- O código final do projeto deve estar todo “mergeado” no branch “main”.

6.3 DOCUMENTAÇÃO README.MD NO BACK-END

Crie um arquivo README.md no repositório do seu projeto no GitHub, para documentar a sua solução, bem como demonstrar as técnicas e linguagens utilizadas, além do escopo do projeto e como o usuário pode executar o seu sistema.

Algumas dicas interessantes para utilizar na criação do seu portfólio são:

- Criar um nome para o seu software;
- Descrever qual o problema ele resolve;
- Descrever quais técnicas e tecnologias utilizadas. Aqui você também pode inserir alguma imagem ou diagrama para melhor entendimento;
- Descrever como executar;
- Descrever quais melhorias podem ser aplicadas;
- Entre outras coisas.

6.4 TRELLO NO BACK-END

Na ferramenta **Trello**, utilize o quadro (*board*) fornecido no item 3 conforme a metodologia *kanban* (*backlog*, *todo*, *doing*, *blocked*, *review* e *done*) para realizar a gestão do seu projeto. A partir das informações extraídas deste documento, organize as tarefas a serem realizadas em cartões (*cards*) e os arraste a cada etapa. Note que você pode fazer um paralelo com a metodologia do *GitFlow*, criando cartões com o mesmo nome das *branches*, por exemplo *feature/create-login-page*, desta forma, o projeto estará organizado e documentado sendo possível rastrear cada etapa e sanar as dúvidas quando necessário. Não esqueça de habilitar o quadro para modo público e colocar o link no readme.md e na submissão no AVA.

Importante: Não utilize o quadro de exercícios para realizar tarefas de projeto. Todos os cards diferentes aos de exercícios serão deletados de forma automática.

7 CRITÉRIOS DE AVALIAÇÃO

Cada aluno será avaliado de forma **individual** dentro das aplicações de front-end e back-end. A nota final, com variação de 0 (zero) a 10 (dez), se dará pela média aritmética entre as duas aplicações. A validação e avaliação da participação no projeto se dará pelos commits realizados e, em alguns critérios, pelo projeto final.

Serão desconsiderados e atribuída a nota 0 (zero) às codificações (individuais) que apresentarem **plágio** de soluções encontradas na internet ou de outros colegas. Lembre-se: Você está livre para utilizar outras soluções como base, mas **não é permitida** a cópia.

Para cada aplicação, os integrantes serão avaliados em 3 pilares distintos:

- Uso adequado do Trello (T)
- Uso adequado do Github (G)
- Desenvolvimento da funcionalidade especificada (D)[1]

Obs [1].: Para a nota de desenvolvimento da funcionalidade especificada será considerada a média de todas as tarefas desenvolvidas pelo aluno, caso exista mais de uma tarefa desenvolvida pelo mesmo.

A nota (N) do aluno em cada aplicação será calculada de acordo com a seguinte fórmula:

$$N = (0{,}15 * T) + (0{,}15 * G) + (0{,}70 * D)$$

A nota do projeto (NP) de cada aluno será calculada de acordo com a seguinte fórmula:

$$NP = (0{,}5 * \text{Aplicação BackEnd}) + (0{,}5 * \text{Aplicação FrontEnd})$$

Os critérios de avaliação das ferramentas de gestão utilizadas pelas squads estão descritos na tabela abaixo.

Uso adequado do Trello				
Nº	**Critério de Avaliação**	**0**	**0,50 a 1,75**	**3,50**
1	Aluno atribuiu adequadamente seu nome no card, bem como especificações da tarefa?	O aluno não criou/atribuiu o card.	Aluno criou o card, mas não colocou seu nome ou comentários especificados nos requisitos.	Aluno criou o card, atribuiu seu nome nele e colocou as especificações de implementação no mesmo.
2	O aluno movimentou adequadamente o Card no board?	Aluno não movimentou o card no board.	Aluno movimentou o card de maneira inadequada no board.	O aluno movimentou adequadamente o

				card, respeitando as regras no Kanbanize.
Nº	**Critério de Avaliação**	**0**	**0,50 a 1,75**	**3,00**
3	O aluno movimentou apenas 1 card com seu nome por vez durante o projeto?	O aluno movimentou mais do que 3 tarefas simultaneamente no board.	O aluno movimentou entre 2 e 3 tarefas simultaneamente no board.	O aluno movimentou apenas 1 tarefa por vez no board.

Uso adequado do GitHub				
Nº	**Critério de Avaliação**	**0**	**0,50 a 1,75**	**3,50**
4	As mensagens do commit estão claras e adequadas?	Os commits do aluno não tinham nenhuma mensagem.	Os alunos fizeram commits com comentários, mas esses estavam inadequados ou incompletos com relação ao conteúdo das alterações.	As mensagens do commit estão claras e adequadas em relação ao conteúdo das alterações.
5	Foi aberto um branch separado para que os alunos realizassem sua tarefa?	Os alunos commitaram direto na master.	Os alunos criaram branch, mas atuaram em mais de uma issue simultaneamente ou não usaram de maneira adequada.	Os alunos criaram uma branch para cada tarefa em que atuaram e só mergearam na develop quando a branch estava finalizada.
Nº	**Critério de Avaliação**	**0**	**0,50 a 1,75**	**3,00**
6	Os commits estão concisos com relação ao número de classes que sofreram alterações?	O aluno frequentemente commitou alterações com mais de 20 arquivos alterados	O aluno frequentemente alterou entre 10 e 19 classes no mesmo commit	O aluno frequentemente alterou menos de 10 classes em cada commit

Os critérios de avaliação das funcionalidades especificadas da aplicação de **Front-End** estão descritos na tabela abaixo.

Nº	**Critério de Avaliação**	**0**	**0,15 a 0,65**	**0,7**
7	Inovou, construindo um layout agradável, responsivo e com acessibilidade?	Não inovou e construiu um layout sem responsividade e acessibilidade.	Inovou, construindo um layout agradável, personalizado e responsivo.	Inovou, construindo um layout agradável, personalizado, responsivo e com acessibilidade.

8	Estruturou o projeto, separando os recursos de acordo com o padrão de projetos?	Não foi efetuado a separação das páginas e componentes e a criação do router e store estão sendo feitas direto no main.	Foi efetuado a separação parcial de páginas e componentes ou inicialização do router e store em arquivos separados, com poucos erros na padronização.	Foi efetuado a separação de componentes, criação da store e router em seus devidos diretórios, separando regras de negócios e criação, sempre que necessário, em arquivos específicos.
9	Desenvolveu a página de login com a funcionalidade de autenticação funcional e com devidos tratamentos no formulário.	Não desenvolveu a página ou não está funcional.	Desenvolveu a página com a autenticação funcional porém com falhas na validação do formulário, alertas ou falta de integração com o backend.	Desenvolveu a página de login com autenticação funcional integrada ao backend, com alertas de avisos e falhas, além criar o guard com redirecionamento para página inicial se usuário já estiver autenticado.
10	Criou o template da aplicação, contendo o Menu Lateral e Toolbar funcional de acordo com os requisitos?	Não desenvolveu os componentes ou não está funcional.	Desenvolveu os componentes, porém sem alguns dos elementos visuais ou sem a validação de exibição após login	Desenvolveu o template utilizando Menu Lateral e Toolbar, com todas as funcionalidades de acordo com os requisitos e com a validação de exibição apenas após o login.
11	Desenvolveu a página de início de administradores e docentes, contendo as estatísticas, pesquisa, listagem de alunos, e redirecionamento para a página solicitada	Não desenvolveu a página ou não está funcional.	Desenvolveu a página de início, com todas as estatísticas e listagem de alunos com dados estáticos e/ou sem redirecionamento	Desenvolveu a página de início, com todas as estatísticas, pesquisa, listagem alunos integradas ao backend e acesso a página solicitada no item.
12	Desenvolveu a página de inicio do aluno, contendo todas as sessões e realizando o redirecionamento para a página de notas	Não desenvolveu a página ou não está funcional.	Desenvolveu a página de início, com dados estáticos e/ou sem redirecionamento	Desenvolveu a página de início, com todas as sessões e realizando o redirecionamento para a página solicitada.

13	Desenvolveu a página de cadastro de docentes de acordo com os requisitos.	Não desenvolveu a página ou não está funcional.	Desenvolveu parcialmente a página ou com erros de validação, não completamente integrado ao backend ou não efetuando a busca de CEP com a API ViaCep.	Desenvolveu a página de cadastro de docentes, com todas as validações, efetuando consulta na API ViaCep e integrado ao backend.
14	Desenvolveu a página de cadastro de alunos de acordo com os requisitos.	Não desenvolveu a página ou não está funcional.	Desenvolveu parcialmente a página ou com erros de validação, não completamente integrado ao backend ou não efetuando a busca de CEP com a API ViaCep.	Desenvolveu a página de cadastro de alunos, com todas as validações, efetuando consulta na API ViaCep e integrado ao backend.
15	Desenvolveu a página de cadastro de turma de acordo com os requisitos.	Não desenvolveu a página ou não está funcional.	Desenvolveu parcialmente a página ou com erros de validação ou não completamente integrado ao backend.	Desenvolveu a página de cadastro com todas as validações funcionais, integrado ao backend e retornando para o usuário a mensagem de conclusão.
16	Desenvolveu a página de cadastro de avaliações/notas, de acordo com os requisitos	Não desenvolveu a página ou não está funcional.	Desenvolveu parcialmente a página ou com erros de validação ou não completamente integrado ao backend.	Desenvolveu a página de cadastro com todas as validações funcionais, integrado ao backend e retornando para o usuário a mensagem de conclusão.
17	Desenvolveu a página de listagem de docentes, com todos os elementos solicitados e realizando o redirecionamento para a página solicitada	Não desenvolveu a página ou não está funcional.	Desenvolveu a página, porém sem todos os elementos visuais, não realizando o redirecionamento solicitado ou com dados fixos em tela.	Desenvolveu a página totalmente funcional, integrada ao backend e realizando o redirecionamento solicitado
18	Desenvolveu a página de notas dos alunos, com todos os elementos solicitados	Não desenvolveu a página ou não está funcional.	Desenvolveu a página, porém sem todos os elementos visuais ou não completamente integrado ao backend	Desenvolveu a página totalmente funcional, seguindo todas as especificações solicitadas e integrado ao backend.
19	Desenvolveu testes unitários para a página de Login	Não desenvolveu os testes unitários	Desenvolveu parcialmente os testes; não executando corretamente ou sem realizar a cobertura para todas as funções.	Desenvolveu os testes unitários para o componente e todas suas funções
20	Desenvolveu testes unitários para a	Não desenvolveu os testes unitários	Desenvolveu parcialmente os testes; não executando corretamente ou sem	Desenvolveu os testes unitários para o

	página de Cadastro de Docentes		realizar a cobertura para todas as funções.	componente e todas suas funções
21	Desenvolveu testes unitários para a página de Listagem de Docentes	Não desenvolveu os testes unitários	Desenvolveu parcialmente os testes; não executando corretamente ou sem realizar a cobertura para todas as funções.	Desenvolveu os testes unitários para o componente e todas suas funções

Os critérios de avaliação das funcionalidades especificadas da aplicação de **Back-End** estão descritos na tabela abaixo.

Nº	Critério de Avaliação	0	0,25 a 1,00	1,00 a 2,00
7	O aluno implementou a funcionalidade de Login, Cadastro de Usuário e Autorização de acesso?	O aluno não implementou essas funcionalidades.	O aluno implementou parcialmente essas funcionalidades.	O aluno implementou corretamente essas funcionalidades.
8	O aluno implementou as funcionalidades de Docente, Curso, Matéria, Aluno, Turma e Nota?	O aluno não implementou essas funcionalidades.	O aluno implementou parcialmente essas funcionalidades.	O aluno implementou corretamente essas funcionalidades.
9	O aluno implementou testes unitários?	O aluno não implementou testes unitários.	O aluno implementou menos de 50% de cobertura de testes unitários.	O aluno implementou mais de 50% de cobertura de testes unitários.
Nº	**Critério de Avaliação**	**0**	**0,25 a 0,50**	**0,75 a 1,00**
10	O aluno escreveu uma documentação para a API?	O aluno não documentou a API.	O aluno documentou parcialmente a API.	O aluno documentou corretamente a API.
11	O aluno realizou a inclusão do Dockerfile e do docker-compose ao projeto?	O aluno não realizou a inclusão do Dockerfile e do docker-compose ao projeto.	O aluno realizou parcialmente a inclusão do Dockerfile e/ou do docker-compose ao projeto.	O aluno realizou a inclusão do Dockerfile e do docker-compose ao projeto.
12	O aluno implementou o endpoint de Dashboard?	O aluno não implementou essas funcionalidades.	O aluno implementou parcialmente essas funcionalidades.	O aluno implementou corretamente essas funcionalidades.

13	O aluno implementou os novos campos/ajustes necessários nos endpoints de Docente, Curso, Matéria, Aluno, Turma e Nota?	O aluno não implementou essas funcionalidades.	O aluno implementou parcialmente essas funcionalidades.	O aluno implementou corretamente essas funcionalidades.

8 PLANO DE PROJETO

Ao construir a aplicação proposta, o aluno estará colocando em prática os aprendizados em:

- Trabalho em **grupo**, simulando um ambiente real de squads, em que cada uma é responsável por determinadas funcionalidades do sistema;
- Trabalhar com **Git**, entendendo que suas alterações podem impactar no desenvolvimento de outras equipes;
- Atuar com processo de desenvolvimento de Software, utilizando o **Trello** como board de Kanban;
- Atuar com projeto de back-end com **Java** usando **Spring** para criação de APIs com conexão ao banco de dados relacional **PostgreSQL**.
- Atuar em projeto de front-end com **Angular**, utilizando os conceitos de navegação entre páginas, storage, consumo de api, gerenciamento de estado, componentização, testes automatizados e etc.