FullStack [Education]

Módulo 1 - Projeto Avaliativo

SUMÁRIO

1 INTRODUÇÃO

O **LAB365 Developer**, empresa líder no segmento tecnológico de gestão estudantil, está com um novo projeto intitulado **labPCP**, uma API Rest completa para gestão de cursos, turmas, conteúdos e docentes, que será futuramente integrada à soluções web de gestão em escolas e creches da rede pública. O seu perfil chamou a atenção dos gestores, para criar a primeira versão da API, que deverá ser construída utilizando o *framework* **Spring**.

Está animado para iniciar o desenvolvimento?

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

2 ENTREGA

O código deverá ser inserido e versionado no **GitHub** em modo privado, e o planejamento deverá ser realizado em um quadro *Kanban* no **Trello**. Ambos os links deverão ser disponibilizados na tarefa **Módulo 1 - Projeto Avaliativo**, presente na semana 13 do AVA até o dia **22/04/2024** às **22h**.

O repositório deverá ser privado, com as seguintes pessoas adicionadas:

- André Santana Nunes - **andresantnunes**
- Gabriel Augustin - **AugustinGabriel**
- Operação LAB 365 - **lab365-operacao**

Não serão aceitos projetos submetidos **após a data limite da atividade**, e, ou alterados depois de entregues.

Importante:

1. Não modifique o seu projeto antes de receber a nota e feedback. Caso contrário, o mesmo não será corrigido.
2. Não esqueça de **submeter os links no AVA**. Não serão aceitos projetos em que os links não tenham sido submetidos.
3. Crie um quadro Trello **diferente do quadro de exercícios**. No quadro de exercícios, os cartões diferentes serão deletados.

3 REQUISITOS DA APLICAÇÃO

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

1. Ser uma API Rest back-end desenvolvida em **Java**.
2. Ser construída utilizando o *framework* **Spring Boot**:
    - Utilizar a estrutura Controller, Service e Repository.
    - Adicionar controle de segurança com **Spring Security.**
    - As requisições e respostas devem ter classes DTO.
3. Utilizar o **PostgreSQL** para guardar as informações cadastradas, podendo ser via *docker* ou conexão direta na máquina.
4. Utilizar o GitHub como versionador de código:
    - Utilização do padrão baseado em GitFlow com *main*, *features* e *releases (release é opcional)*.
    - Utilização de commits curtos e concisos.
5. Utilizar Trello para organização das tarefas a serem realizadas:

- Criar um quadro (*board*) em sua conta pessoal ou educacional, colocando o mesmo em modo público.
- Criar um quadro com estrutura *kanban* (*backlog*, *todo*, *doing*, *blocked*, *review* e *done*) para organização das tarefas.
- Criar cartões com tarefas a serem realizadas e arrastar cada um conforme a necessidade.

6. Possuir as seguintes **entidades**:
   - Entidade Papel:
     - Atributos:
       id (int): O identificador único do papel.
       nome (String): O nome do papel.
   - Entidade Usuário:
     - Atributos:
       id (int): O identificador único do usuário.
       nome_usuario/login (String): O nome de usuário para login.
       senha (String): A senha do usuário para autenticação.
       id_papel (Papel): O papel ou função do usuário no sistema.
   - Entidade Docente:
     - Atributos:
       id (int): O identificador único do docente.
       nome (String): O nome completo do docente.
       data_entrada(Date): Data de entrada do docente na empresa.
       id_usuario(int): O identificador do usuário, deve ser único.
   - Entidade Turma:
     - Atributos:
       id (int): O identificador único da turma.
       nome (String): O nome ou identificador da turma.
       alunos (List<Aluno>): A lista de alunos matriculados na turma. O relacionamento pode ser no modelo One-To-Many ou Many-to-One.
       professor (Docente): O docente responsável pela turma.
       id_curso (int): O relacionamento pode ser no modelo One-To-Many ou Many-to-One.
   - Entidade Aluno:
     - Atributos:
       id (int): O identificador único do aluno.
       nome (String): O nome completo do aluno.
       data_nascimento (Data): Data de nascimento do aluno.
       id_usuario (int): O identificador do usuário, deve ser único.
       id_turma (int): O relacionamento pode ser no modelo One-To-Many ou Many-to-One.

- Entidade Curso:
  - Atributos:
    id (int): O identificador único do curso.
    nome (String): O nome do curso.
    turmas (List<Turma>): A lista de turmas associadas ao curso. O relacionamento pode ser no modelo One-To-Many ou Many-to-One.
    materias (List<Materia>): A lista de matérias oferecidas no curso. O relacionamento pode ser no modelo One-To-Many ou Many-to-One.
- Entidade Matéria:
  - Atributos:
    id (identificador único da matéria)
    nome (nome da matéria)
    id_curso (int): O relacionamento pode ser no modelo One-To-Many ou Many-to-One.
- Entidade Notas:
  - Atributos:
    id (identificador único da nota)
    id_aluno (identificador do aluno ao qual a nota está associada)
    id_professor (identificador do professor que atribuiu a nota)
    id_materia (identificador da matéria à qual a nota pertence)
    valor (valor da nota)
    data (data em que a nota foi atribuída)

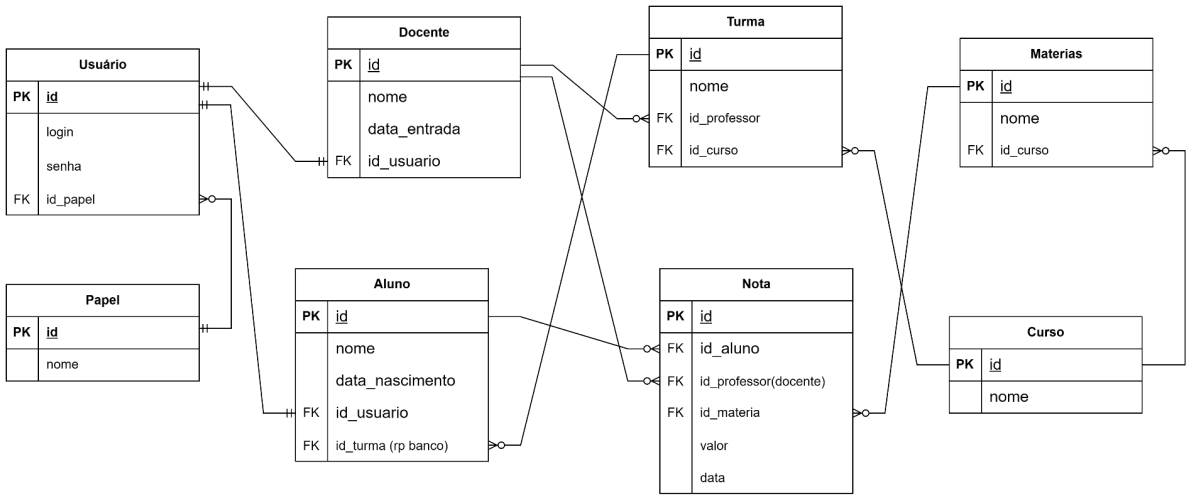

Imagem 1: Modelo das entidades no banco de dados

7. Utilizar os seguintes **papéis do usuário**:
   - **ADM (Entidade Docente)**: que pode acessar tudo.

- **PEDAGOGICO (Entidade Docente)**: que pode acessar tudo sobre turma, curso e professor, aluno, menos deletar dados.
- **RECRUITER (Entidade Docente)**: que pode acessar tudo sobre professor, menos deletar dados.
- **PROFESSOR (Entidade Docente)**: que pode acessar tudo sobre notas, menos deletar dados.
- **ALUNO (Entidade Aluno):** que apenas acessa as suas próprias notas e a pontuação total pessoal, menos deletar dados.

8. Seguir as seguintes **regras de negócio**:
   - Só o **ADM** pode deletar entidades.
   - Só o **ADM** pode criar usuários (realizar cadastros)
   - Os **Logs** são pré requisitos, sendo um log para cada erro e pelo menos um log para cada método executado.
   - A API deverá gerar a pontuação total de um aluno, a pontuação é dada pela seguinte lógica: *soma das notas, dividido pelo número de matérias, multiplicado por 10.*

4 ROTEIRO DA APLICAÇÃO

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.

4.1 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.

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

Endpoints para entidade Docente

- Criar Docente (POST /docentes)
    - 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 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 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 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 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 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 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 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 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 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 404 (Not Found) - Não há turmas cadastradas.

Endpoints para entidade Aluno

- Criar Aluno (POST /alunos)

    - 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.
- 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 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 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 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 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.
- 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 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 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 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 404 (Not Found) - Não há cursos cadastrados.

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 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.
- 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 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 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 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 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.
- 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 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 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 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 404 (Not Found) - Aluno não encontrado.

4.2 DOCUMENTANÇÃO NO README.MD

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.

4.3 USO DO TRELLO

Na ferramenta **Trello**, crie uma área de trabalho (*workspace*) e um quadro (*board*) 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 CRITÉRIOS DE AVALIAÇÃO

A tabela abaixo apresenta os critérios que serão avaliados durante a correção do projeto. O mesmo possui variação de nota de 0 (zero) a 10 (dez) como nota mínima e máxima, e possui peso de **60% sobre a avaliação do módulo**.
Serão **desconsiderados e atribuída a nota 0 (zero)** os projetos 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.

Uso adequado do Trello				
Nº	**Critério de Avaliação**	**0**	**0,25**	**0,50**
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, movimentando 1 por vez, e respeitando as regras no Kanbanize.

Uso adequado do GitHub				
Nº	**Critério de Avaliação**	**0**	**0,25**	**0,50**
3	As mensagens do commit estão claras e adequadas?	Os commits do aluno não tiveram nenhuma mensagem	Os alunos fizeram commits com comentários, mas esses estavam inadequados	As mensagens do commit estão claras e adequadas em relação

			ou incompletos com relação ao conteúdo das alterações	ao conteúdo das alterações
4	Foi aberto um branch separado para cada etapa, funcionalidade?	Os commits foram realizados diretamente na master/main.	Foram abertas branchs para cada funcionalidade, porém foram realizados commits de mesma funcionalidade em branchs diferentes.	Foram abertas branchs para cada funcionalidades, e os commits foram realizados de forma concisa nas mesmas.

Desenvolvimento adequado da Aplicação				
Nº	**Critério de Avaliação**	**0**	**0,25 a 0,50**	**1,00**
5	Implementou todas as regras de negócio dos Requisitos da Aplicação. E adicionou a conexão com o banco de dados	Não implementou todas as regras de negócio e não conectou ao banco de dados.	Implementou parcialmente as regras de negócio, e conexão com um banco H2	Implementou completamente as regras de negócio, e conexão com o banco de dados PostgreSQL.
6	Desenvolveu os endpoints de Cadastro e Login. Adicionou o Spring Security?	Não desenvolveu os endpoints.	Implementou parcialmente os endpoints, ou com falhas.	Implementou completamente os endpoints com todos funcionando.
7	Desenvolveu os endpoints de Docente.	Não desenvolveu os endpoints.	Implementou parcialmente os endpoints, ou com falhas.	Implementou completamente os endpoints com todos funcionando.
8	Desenvolveu os endpoints de Turma.	Não desenvolveu os endpoints.	Implementou parcialmente os endpoints, ou com falhas.	Implementou completamente os endpoints com todos funcionando.
9	Desenvolveu os endpoints de Aluno.	Não desenvolveu os endpoints.	Implementou parcialmente os endpoints, ou com falhas.	Implementou completamente os endpoints com todos funcionando.
10	Desenvolveu os endpoints de Curso.	Não desenvolveu os endpoints.	Implementou parcialmente os endpoints, ou com falhas.	Implementou completamente os endpoints com todos funcionando.
11	Desenvolveu os endpoints de Matérias.	Não desenvolveu os endpoints.	Implementou parcialmente os endpoints, ou com falhas.	Implementou completamente os endpoints com todos funcionando.
12	Desenvolveu os endpoints de Notas.	Não desenvolveu os endpoints.	Implementou parcialmente os endpoints, ou com falhas.	Implementou completamente os

				endpoints com todos funcionando.

6 PLANO DE PROJETO

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

- **Kanban Board**: O que é Kanban e utilização no Trello.
- **Versionamento**: O que é versionamento de código, aplicações e utilização no Github
- **Programação Orientada a Objetos**: Conceitos de POO, Classes, Objetos, Métodos, Atributos, Construtores, Modificadores, Encapsulamento, Sobrecarga, Herança e Polimorfismo.
- **Java**: Java básico, Estruturas de Controle de Fluxo, Arrays e ArrayList, Documentação de Código e Tratamento de Exceções.
- **Spring**: Spring Boot, Maven, Spring Data, Spring Security, CRUD Rest API.
- **Banco de Dados**: Modelo Relacional e SQL com PostgreSQL.