Basis - Prova Avaliação

Prova de Avaliação de conhecimentos de candidatos a desenvolvedores Java.

Sobre a prova

Este teste tem como objetivo, verificar o conhecimento técnico e a qualidade de desenvolvimento de código. Diante disso, caso o candidato não consiga concluir todas as tarefas definidas neste teste, faça o que conseguir com a melhor qualidade possível. Seu teste será avaliado mesmo que não tenha concluído todas as tarefas solicitadas.

O teste deverá ser iniciado a partir de um projeto pré-configurado disponibilizado pela Basis (https://github.com/BasisTI/avaliacao-basis). Após realizar o clone do repositório, siga as instruções abaixo para executar o projeto.

Executando a aplicação

Para a execução da aplicação, é necessário que você tenha instalado em seu PC:

- Maven
- JDK 8
- Docker

Com as ferramentas instaladas corretamente, basta abrir o arquivo `pom.xml` em sua IDE (IntelliJ, Eclipse ou STS) e importar todas as dependências do projeto. Caso deseje executar o projeto pelo terminal, utilize o comando: `mvn spring-boot:run`

A aplicação poderá ser acessada na URL: `http://localhost:8080/api`

Banco de Dados

O banco de dados usado pela aplicação é o MySQL. Para subir o banco, é necessário ter instalado o Docker Compose em seu PC, feito isso, base abrir o terminal na pasta raiz do projeto e executar o seguinte comando: `docker-compose up -d`.

- Para acessar o banco usar o endereço: `http://localhost:3306`
- Usar o nome do banco de dados: `provabasis`
- Usar o Usuário e Senha: `root`

Observação: Caso tenha problemas com a instalação do Docker Compose, e queira utilizar outro banco, basta alterar as configurações de conexão com o banco no arquivo `application.properties` localizado dentro do projeto de acordo com o banco de dados que irá utilizar.

1. Aplicação

1.1. Introdução

A aplicação a ser desenvolvida representa um sistema acadêmico básico. Onde é necessário o cadastro de **Alunos**, **Disciplinas** e **Professores**. Deve ser desenvolvida uma API RestFul que será responsável por receber requisições de Clientes (Angular, React, Vue, REST-Client, etc).

1.2. Tecnologias

Abaixo estão listadas todas as tecnologias utilizadas na implementação do projeto:

- JPA
- Hibernate
- Spring Framework
- Spring Data
- Spring Boot
- Maven
- Liquibase

1.3. Documentação

1.3.1. Alunos

O cadastro de alunos deve armazenar as informações:

- **Id:** Campo **único** e **obrigatório** numérico**.**
- **PF:** Campo **único** e **obrigatório**, contendo 11 caracteres numéricos.
- **Nome:** Campo **obrigatório**, contendo de 1 a 100 caracteres.
- **Data de nascimento:** Campo **obrigatório**, contendo apenas dd/mm/aaaa.
- **Matrícula:** Campo **único** e **obrigatório**, contendo 6 caracteres.
- **Disciplinas:** Campo que armazena as disciplinas vinculadas ao aluno.

Consulta: A API deve fornecer uma consulta de alunos retornando apenas seu nome, matrícula e idade.

Consulta detalhada: A API deve fornecer uma consulta de alunos retornando apenas seu nome, matrícula, idade e os nomes das disciplinas em que estão matriculados.

Exclusão: A API deve fornecer uma forma de exclusão de um aluno pela matrícula.

REGRAS: Um aluno só pode se matricular em disciplinas ativas. O Aluno só pode ser excluído se não estiver vinculado a uma disciplina.

1.3.2. Disciplinas

O cadastro de disciplinas deve armazenar as informações:

- **Id:** Campo **único** e **obrigatório** numérico**.**
- **Nome:** Campo **único** e **obrigatório**, contendo de 1 a 50 caracteres.
- **Descrição:** Campo **obrigatório**, contendo de 1 a 200 caracteres.
- **Carga horária (hrs):** Campo **obrigatório** numérico.
- **Ativa:** Campo que armazena o status da disciplina.
- **Professor:** Campo que armazena o professor responsável pela disciplina.

Consulta: A API deve fornecer uma consulta de disciplinas retornando apenas seu nome, descrição e carga horária.

Consulta detalhada: A API deve fornecer uma consulta de disciplinas retornando apenas seu nome, carga horária e o nome do professor responsável.

Exclusão: A API deve fornecer uma forma de exclusão de uma disciplina pelo identificador.

REGRAS: Uma disciplina só pode ser cadastrada informando qual professor será responsável por ela. A disciplina só poderá ser excluída caso não haja nenhum aluno matriculado.

1.3.3. Professores

O cadastro de professores deve armazenar as informações:

- **Id:** Campo **único** e **obrigatório** numérico**.**
- **Matrícula:** Campo **único** e **obrigatório**, contendo 6 caracteres.
- **Nome:** Campo **único** e **obrigatório**, contendo de 1 a 50 caracteres.
- **Área de atuação:** Campo contendo de 1 a 200 caracteres.
- **Data de nascimento:** Campo **obrigatório**, contendo apenas dd/mm/aaaa.
- **Disciplinas:** Campo que armazena por quais disciplinas o professor é responsável.

Consulta: A API deve fornecer uma consulta de professores retornando apenas seu nome, matrícula, área de atuação e idade.

Consulta detalhada: A API deve fornecer uma consulta de professores retornando apenas seu nome, matrícula e os nomes das disciplinas ativas pelas quais o professor é responsável.

Exclusão: A API deve fornecer uma forma de exclusão de um professor pela matrícula.

REGRAS: Um professor só pode ser excluído se não for responsável por nenhuma disciplina.

2. Entrega

O desafio deve ser submetido a um repositório git de sua preferência (Github, Bitbucket, GitLab) em modo `público`. O link do repositório deve ser enviado para **o email que lhe enviou este documento** com o assunto “Avaliação Basis” e com uma breve descrição sobre quais funcionalidades foram implementadas pelo candidato durante o desenvolvimento do projeto.