

## Especificação do Trabalho

O primeiro trabalho da disciplina de Banco de Dados I consiste no desenvolvimento de parte de um microserviço de *back-end* que permite a aplicações clientes (por ex., *front-end* ou *Application Programming Interface client*) gerenciar turmas e objetos de aprendizagem armazenados em um SGBD através de uma API REST<sup>1</sup>. Este trabalho oportuniza a prática dos conhecimentos obtidos na disciplina com tecnologias atuais e em um sistema real, o ambiente BOCA (*BOCA Online Contest Administrator*), o qual é usado para gerenciar competições da Maratona de Programação da SBC<sup>2</sup> e do Topcom<sup>3</sup> e, mais recentemente, como ferramenta de apoio em disciplinas de programação oferecidas pelo Departamento de Informática (DI) da Ufes.

## Requisitos

O trabalho poder ser feito em grupos de até 2 alunos(as) e a implementação deve, preferencialmente, utilizar JavaScript e Node.js. Qualquer *framework*/biblioteca auxiliar (por ex., TypeScript) pode ser utilizado(a) desde que as consultas ao banco de dados sejam escritas explicitamente através de *statements* SQL. Deve ser utilizada como ponto de partida a versão containerizada e baseada em microserviços do BOCA disponível em <https://github.com/joaofazolo/boca-docker>. A descrição sobre o seu funcionamento e a estrutura do seu banco de dados está presente no documento em anexo, enquanto as instruções para sua implantação estão disponíveis no referido repositório de código. Para favorecer a manutenibilidade e o desenvolvimento de novas funcionalidades, é imprescindível que o *back-end* seja encapsulado em um contêiner independente (vide Figura 1) e que alterações na estrutura do banco de dados (por ex., criação de novas tabelas) sejam dispostas em arquivos separados (ou *migrations*) e executadas na implantação do serviço (vide exemplos no repositório de código). Inclua um arquivo README.MD<sup>4</sup> no repositório que, além das informações para execução do projeto, também possua: decisões, desafios/problemas com os quais vocês se depararam durante a execução do projeto; maneiras através das quais vocês podem melhorar a aplicação, seja em performance, estrutura ou padrões.

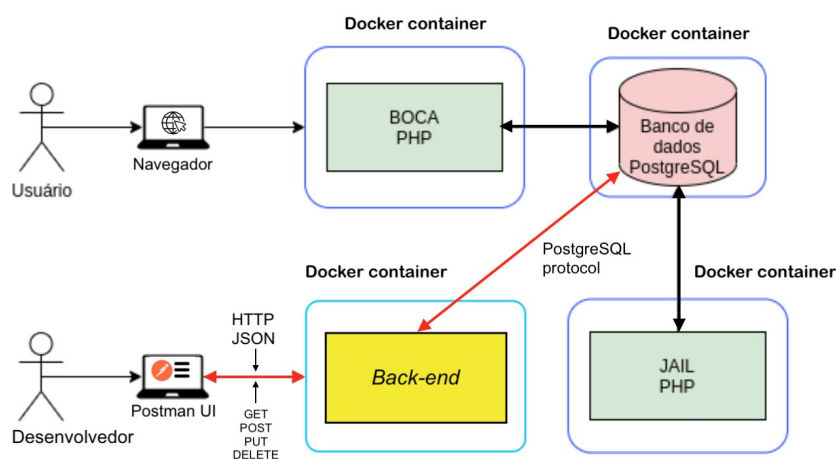


Figura 1 – Arquitetura containerizada do BOCA. Fonte: Modificado de Fazolo, 2021.

## Dicas

- Tenham sempre um *mindset* de usabilidade, escalabilidade e colaboração;
- Documente o projeto e o código explicando a estrutura, processo de setup e requisitos;
- A organização das *branches* e os *commits* no repositório falam muito sobre a participação dos membros durante o desenvolvimento do trabalho;

<sup>1</sup> Também conhecida como REST ou Representational State Transfer; um padrão de comunicação utilizado para interoperabilidade entre sistemas de computação.

<sup>2</sup> <http://maratona.ime.usp.br/manualBOCA.html>

<sup>3</sup> <https://topcom.pet.inf.ufes.br>

<sup>4</sup> <https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>

- Usem boas práticas de programação;
- Considerem que não sei nada sobre os seus conhecimentos, então quanto mais vocês mostrarem e o quão mais descritiva for a documentação e testes, melhor.

### Podemos utilizar *frameworks*/bibliotecas?

R: Vocês podem usar bibliotecas, mas é importante que o impacto/relevância dessa decisão sobre a solução proposta seja devidamente justificada.

### Posso usar um ORM (*Object Relational Mapper*)?

R: O uso de ORM é permitido apenas para estabelecer a conexão com o banco de dados. No entanto, para que seja possível avaliar os conhecimentos em SQL não usem ORM nas consultas ao banco. As bibliotecas normalmente possuem métodos que permitem consultas diretamente em SQL (*raw queries*) mesmo através do ORM.

### Links úteis:

- <https://nodejs.org/en/docs/guides/nodejs-docker-webapp/>
- <https://blog.logrocket.com/nodejs-expressjs-postgresql-crud-rest-api-example/>
- <https://dev.to/nedsoft/testing-nodejs-express-api-with-jest-and-supertest-1km6>

## API

O objetivo do trabalho é poder criar e gerenciar *tags* para entidades do BOCA. Uma *tag* é uma espécie de rótulo que pode ser atribuído a qualquer entidade. É um par de uma chave e um valor opcional que permite categorizar os recursos, o que facilita a pesquisa. Por exemplo, aplicando uma *tag* chamada *domain* com o valor *basic select* em problemas de interesse, um professor e/ou aluno podem obter uma lista de todos os exercícios de consultas básicas usando a API REST. A especificação completa das rotas (com exemplos) é apresentada abaixo.

## Submissão

Cada grupo deve criar um **repositório privado** no GitHub (sob a organização [UFES20231BDCOMP](#)) para hospedar a implementação do trabalho. Para isso, será preciso antes informar os *usernames* dos membros do grupo através deste [link](#) (somente um dos membros precisa fazê-lo até 24/05/2023, mas todos devem fazer parte do projeto posteriormente). Além do código fonte, o projeto deve conter toda a documentação necessária para que um usuário de um curso de graduação em computação na Ufes possa executá-lo (ou seja, desde informações sobre os pré-requisitos de hardware e software até mesmo comandos básicos para configurar o ambiente e implantar o serviço). Para finalizar o trabalho, deve ser criada uma *release* do projeto até a data limite de entrega.

## Prazos

Os prazos de entrega do trabalho podem ser consultados no calendário da página do curso no Google Classroom.

## Avaliação

O trabalho será avaliado com base nos seguinte critérios:

- Funciona? Cumpre os requisitos?
- Qualidade da solução proposta (modelagem, desempenho, etc.)
- Boas práticas (estrutura e organização do código, documentação, etc.)
- Testes unitários (de preferência, automatizados)

## Observações finais

Caso haja algum erro neste documento, serão publicadas novas versões e divulgadas erratas nas aulas. É responsabilidade do(a) aluno(a) manter-se informado(a), frequentando as aulas ou acompanhando as novidades através das ferramentas utilizadas na disciplina.

## Referências

- João Vitor Alves Fazolo, 2021. AlgoChallenger - Reengenharia do BOCA Online Contest Administrator utilizando uma arquitetura de microserviços para o ensino de programação. Trabalho de conclusão de curso de graduação (Ciência da Computação) - Universidade Federal do Espírito Santo.

## 1. Recuperar tags de entidade

Esta operação recupera uma lista de *tags* que são definidas para qualquer entidade do BOCA.

### Requisição

*Sintaxe (método e URL)*

```
GET <webservice>/api/contest/{contestId}/tags/{entityType}/{entityId}?  
tagId={tagId}&tagName={tagName}&tagValue={tagValue}
```

onde <webservice> é o caminho raiz usado para rotear as requisições de API para o servidor Web.

*Parâmetros*

Nome	Descrição	Obrigatório
contestId	O ID de uma competição/concurso do BOCA.	Sim
entityType	O tipo da entidade.  Os valores válidos são: <ul style="list-style-type: none"><li>• <i>problem</i></li><li>• <i>language</i></li><li>• <i>site</i></li><li>• <i>site/user</i></li></ul>	Sim
entityId	O ID da entidade de um tipo de entidade específica do BOCA.	Sim
tagId	O ID de uma <i>tag</i> .  Se o ID da <i>tag</i> não for conhecido, use esta API para recuperar uma lista de IDs de <i>tag</i> para um tipo de entidade ou ID de entidade.	Não
tagName	O nome de uma <i>tag</i> .  Se o nome da <i>tag</i> não for conhecido, use esta API para recuperar o nome da <i>tag</i> para um determinado ID de <i>tag</i> .	Não
tagValue	O valor de uma <i>tag</i> .  Se o valor da <i>tag</i> não for conhecido, use esta API para recuperar o valor da <i>tag</i> para um determinado ID de <i>tag</i> .	Não

### Resposta

Código	Descrição
200	Sucess: <i>tag(s)</i> encontradas(s).
400	Bad Request: Pelo menos um dos parâmetros fornecidos na requisição é inválido.
404	Not Found: O ID da competição ou da entidade especificado na requisição não existe.

*Corpo (body)*

A tabela a seguir exibe os parâmetros para o corpo da resposta (JSON).

Nome	Descrição	Elementos
entityType	O tipo da entidade.  Os valores válidos são: <ul style="list-style-type: none"><li>• <i>problem</i></li><li>• <i>language</i></li></ul>	entityTag

	<ul style="list-style-type: none"> <li>• <i>site</i></li> <li>• <i>site/user</i></li> </ul>	
entityId	O ID da entidade de um tipo de entidade específico.	entityTag
name	O nome da <i>tag</i> da entidade.	tag
id	O ID da <i>tag</i> da entidade.	tag
value	O valor da <i>tag</i> da entidade.	value

### Exemplo

GET <webservice>/api/contest/1/tags/problem/2006

```
{
  "entityTag": [
    {
      "entityType": "problem",
      "entityId": 2006,
      "tag": [
        {
          "id": 1,
          "name": "group",
          "value": "silberschatz"
        },
        {
          "id": 2,
          "name": "level",
          "value": "easy"
        },
        {
          "id": 3,
          "name": "domain",
          "value": "basic select"
        },
        {
          "id": 4,
          "name": "lang",
          "value": "relax"
        }
      ]
    }
  ]
}
```

## 2. Criar *tags* de entidades

Esta operação cria novas *tags* para uma ou mais entidades do BOCA.

### Requisição

*Sintaxe (método e URL)*

POST <webservice>/api/contest/{contestId}/tags

onde <webservice> é o caminho raiz usado para rotear as requisições de API para o servidor Web.

*Parâmetros*

Nome	Descrição	Obrigatório
contestId	O ID de uma competição/concurso do BOCA.	Sim

*Corpo (body)*

A tabela a seguir exibe os parâmetros para o corpo da requisição (JSON).

Nome	Descrição	Elementos
entityType	O tipo da entidade no BOCA.  Os valores válidos são: <ul style="list-style-type: none"><li>• <i>problem</i></li><li>• <i>language</i></li><li>• <i>site</i></li><li>• <i>site/user</i></li></ul>	entityTag
entityId	O ID da entidade de um tipo de entidade específica do BOCA.	entityTag
id	O ID de uma <i>tag</i> .	tag
name	O nome de uma <i>tag</i> .	tag
value	O valor de uma <i>tag</i> .	tag

### Resposta

Código	Descrição
204	Sucess: <i>tag(s)</i> atualizad(s).
400	Bad Request: O ID da competição ou o JSON fornecido no corpo da requisição é inválido.
404	Not Found: O ID da competição especificado na requisição não existe.

## Exemplo

POST <webservice>/api/contest/1/tags

```
{
  "entityTag": [
    {
      "entityType": "problem",
      "entityId": 2006,
      "tag": [
        {
          "id": 5,
          "name": "lang",
          "value": "mysql"
        },
        {
          "id": 6,
          "name": "working",
          "value": "atividade 02"
        },
        {
          "id": 7,
          "name": "working",
          "value": "p1"
        }
      ]
    },
    {
      "entityType": "site/user",
      "entityId": "1/1001",
      "tag": [
        {
          "id": 8,
          "name": "working",
          "value": "atividade 01"
        }
      ]
    }
  ]
}
```

### 3. Atualizar *tags* de entidades

Esta operação atualiza *tags* existentes para uma ou mais entidades do BOCA.

#### Requisição

*Sintaxe (método e URL)*

PUT <webservice>/api/contest/{contestId}/tags

onde <webservice> é o caminho raiz usado para rotear as requisições de API para o servidor Web.

*Parâmetros*

Nome	Descrição	Obrigatório
contestId	O ID de uma competição/concurso do BOCA.	Sim

*Corpo (body)*

A tabela a seguir exibe os parâmetros para o corpo da requisição (JSON).

Nome	Descrição	Elementos
entityType	O tipo da entidade no BOCA.  Os valores válidos são: <ul style="list-style-type: none"><li>• <i>problem</i></li><li>• <i>language</i></li><li>• <i>site</i></li><li>• <i>site/user</i></li></ul>	entityTag
entityId	O ID da entidade de um tipo de entidade específica do BOCA.	entityTag
id	O ID de uma <i>tag</i> .	tag
name	O nome de uma <i>tag</i> .	tag
value	O valor de uma <i>tag</i> .	tag

#### Resposta

Código	Descrição
204	Sucess: <i>tag(s)</i> atualizad(s).
400	Bad Request: O ID da competição ou o JSON fornecido no corpo da requisição é inválido.
404	Not Found: O ID da competição especificado na requisição não existe.

## Exemplo

PUT <webservice>/api/contest/1/tags

```
{
  "entityTag": [
    {
      "entityType": "problem",
      "entityId": 2006,
      "tag": [
        {
          "id": 5,
          "name": "lang",
          "value": "postgres"
        }
      ]
    },
    {
      "entityType": "site/user",
      "entityId": "1/1001",
      "tag": [
        {
          "id": 8,
          "name": "working",
          "value": "atividade 02"
        }
      ]
    }
  ]
}
```



## 4. Excluir *tags* de entidades

Esta operação exclui *tags* existentes para uma ou mais entidades do BOCA.

### Requisição

*Sintaxe (método e URL)*

DELETE <webservice>/api/contest/{contestId}/tags

onde <webservice> é o caminho raiz usado para rotear as requisições de API para o servidor Web.

*Parâmetros*

Nome	Descrição	Obrigatório
contestId	O ID de uma competição/concurso do BOCA.	Sim

*Corpo (body)*

A tabela a seguir exibe os parâmetros para o corpo da requisição (JSON).

Nome	Descrição	Elementos
entityType	O tipo da entidade no BOCA.  Os valores válidos são: <ul style="list-style-type: none"><li>• <i>problem</i></li><li>• <i>language</i></li><li>• <i>site</i></li><li>• <i>site/user</i></li></ul>	entityTag
entityId	O ID da entidade de um tipo de entidade específica do BOCA.	entityTag
id	O ID de uma <i>tag</i> .	tag
name	O nome de uma <i>tag</i> .	tag
value	O valor de uma <i>tag</i> .	tag

### Resposta

Código	Descrição
204	Sucess: <i>tag(s)</i> excluída(s).
400	Bad Request: O ID da competição ou o JSON fornecido no corpo da requisição é inválido.
404	Not Found: O ID da competição especificado na requisição não existe.

## Exemplo

DELETE <webservice>/api/contest/1/tags

```
{
  "entityTag": [
    {
      "entityType": "problem",
      "entityId": 2006,
      "tag": [
        {
          "id": 4,
          "name": "lang",
          "value": "relax"
        },
        {
          "id": 7,
          "name": "working",
          "value": "p1"
        }
      ]
    },
    {
      "entityType": "site/user",
      "entityId": "1/1001",
      "tag": [
        {
          "id": 8,
          "name": "working",
          "value": "lista 2"
        }
      ]
    }
  ]
}
```