**13/10/2020 – Atividade #6 – Projeto Integrador:**

**- DOCUMENTAÇÃO BACK-END:**

**TABELA POSTAGEM - USUÁRIO**

|  |  |
| --- | --- |
| **ATRIBUTO** | **TIPO** |
| ID | Long (BigInt - MySql) |
| Título | String (Varchar – 50 - MySql) |
| Descrição | String (Text - MySql) |
| Região | String (Varchar - 50) |
| Data | Date (Date - MySql) |
| Imagens | String (Varchar-255 - MySql) |
| Material | String (Varchar – 100 - MySql) |
| Tipo\_presenca | Boolean |
| Tema\_id [Foreign Key] | Long (Bigint - MySql) |
| Usuário\_id [Foreign Key] | Long (Bigint - MySql) |

A tabela POSTAGEM possuirá o atributo (i) título, com o resumo do objetivo da postagem que o usuário fará; (ii) descrição, com os detalhes da postagem; (iii) região, com o objetivo de informar em qual região da cidade os materiais estarão disponíveis para retirada e/ou a região em que ocorrerá a oficina, p.ex.; (iv) data, com a data da postagem; (v) imagens, para que o usuário detalhe seu post utilizando quaisquer imagens que entender relevantes; (vi) material, na qual o usuário poderá escolher se será sobre oficinas, cursos, doações, qual tipo de material doado etc.

Haverá, ainda, dois atributos que farão o papel de chaves estrangeiras da tabela, ligando-a com a tabela TEMA e a tabela USUÁRIO, quais sejam: tema\_id e usuário\_id. A conexão será feia por meio da anotação @ManyToOne.

Para evitar o problema da recursividade, usaremos, também, a anotação @JsonIgnoreProperties.

**CRUD**

|  |  |  |
| --- | --- | --- |
| **MÉTODOS** | **END-POINTS** | **DESCRIÇÃO** |
| **GET** | /postagem | Lista todas as postagens existentes. |
| **GET** | /postagem/{id} | Lista as postagens especificadas pela id. |
| **GET** | /postagem/titulo/{titulo} | Lista as postagens de acordo com o título. |
| **GET** | /postagem/regiao/{regiao} | Lista as postagens de acordo com a região do usuário. |
| **GET** | /postagem/data/{data} | Lista as postagens de acordo com a data de publicação. |
| **POST** | /postagem | Insere dados na tabela. |
| **PUT** | /postagem | Altera dados na tabela. |
| **DELETE** | /postagem/{id} | Deleta dados pelo ID. |

Nossa tabela terá todos os end-points básicos, o que possibilitará as operações básicas, isto é, **get, post, put** e **delete.** Ainda, teremos um método geral para buscar todas as postagens, bem como alguns métodos específicos filtrando a pesquisa de acordo com a id, o título, região ou data da postagem.

**Json:**

1. **Postando dados (inserindo dados no banco de dados – método POST):**

{

“titulo”: “doação”,

“descricao”: “Olá pessoal, tenho materiais em vidro para doação. Não é uma quantidade muito grande, mas o suficiente para quem estiver trabalhando sozinho e não precise fazer nada em larga escala.”,

“regiao”: “Jabaquara”,

“data”: “2020-09-22”,

“imagens”: “”,

“material”: “vidro”,

“tipo\_presenca”: true,

“tema\_id”: {

“id”: 1,

“conteudo”: “doação”,

“dataehora”: “2020-09-01T13:47:44.663+00:00”

}

}

1. **Recebendo/consultando dados (método GET):**

* Método para pesquisa: **/postagem**
* Retorno esperado:

{

“titulo”: “doação”,

“descricao”: “Olá pessoal, tenho materiais em vidro para doação. Não é uma quantidade muito grande, mas o suficiente para quem estiver trabalhando sozinho e não precise fazer nada em larga escala.”,

“regiao”: “Jabaquara”,

“data”: “2020-09-22”,

“imagens”: “”,

“material”: “vidro”,

“tipo\_presenca”: true,

“tema\_id”: {

“id”: 1,

“conteudo”: “doação”,

}

}

1. **Deletando dados (método DELETE):**

* Método para pesquisa: **/postagem/{id}**
* Retorno esperado: [] //retorna status 200 OK. Isso quer dizer que a requisição foi atendida sem erros.

**TABELA USUÁRIO – MODEL:**

|  |  |
| --- | --- |
| **ATRIBUTO** | **TIPO** |
| ID | Long (bigint - MySql) |
| Nome | String (varchar – 100 - MySql) |
| E-mail | String (varchar – 30 - MySql) |
| Senha | String (varchar – 12 - MySql) |
| Telefone | int |
| Postagem [Foreign Key] | List<Postagem> |

A tabela USUÁRIO terá o atributo ID, referente ao código de cada usuário, além dos atributos nome, e-mail, senha e telefone.

**CRUD:**

|  |  |  |
| --- | --- | --- |
| **MÉTODOS** | **END-POINTS** | **DESCRIÇÃO** |
| **POST** | /usuario/cadastrar | Cadastra o usuário na rede social. |
| **POST** | /usuario/logar | Faz o login de um usuário existente. |
| **PUT** | /usuario/atualizarUsuario | Faz atualizações em um cadastro de usuário já existente. |

Nossa tabela usuário terá os end-points básicos, que possibilitarão as operações **post** e **put**, para que o usuário consiga se cadastrar, logar e alterar seu cadastro, se assim desejar.

**Json:**

1. **Postando dados (inserindo dados no banco de dados – método POST):**

{

“nome”: “Lisbeth Salander”,

“e-mail”: “lisbeth.salander@gmail.com”,

“senha”: “123deoliveiraquatro”,

“telefone”: 11999999999

}

1. **Atualizando dados do usuário:**

* Método para pesquisa: **/usuario/atualizarUsuario**
* Resultado esperado:

{

“id”: 1,

“nome”: “Lisbeth Salander”,

“e-mail”: “lisbeth.salander@gmail.com”,

“senha”: “123deoliveiraquatro”,

“telefone”: 1199999994444

}