02

Pesquisando aluno por nota

Transcrição

Concluída a busca por nome, podemos criar a pesquisa por nota, com a qual faremos a classificação de aprovação ou reprovação dos alunos. O processo inicial é praticamente o mesmo, criaremos um novo cartão no `index.html`.

```
<div class="col s12 m4 l4 waves-effect waves-light">
    <a href="/nota/iniciarpesquisa">
        <div class="card-panel hoverable z-depth-1 center grey lighten-4">
            <i class="large material-icons">search</i>
            <div class="truncate">Pesquisar por Nota</div>
        </div>
    </a>
</div>
```

Note que o domínio da pesquisa mudou, antes era o aluno, agora estamos apontando diretamente para as notas, em que teremos `/nota/iniciarpesquisa`. Criaremos o método que responde a esta rota em `NotaController`.

```
@GetMapping("/nota/iniciarpesquisa")
public String iniciarPesquisa() {
  return "nota/pesquisar";
}
```

Por ora retornaremos o *template* da página de pesquisa, que se chama `pesquisar.html` e está em `resources/templates/nota`, contendo o seguinte código:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8" />
<link type="text/css" rel="stylesheet" href="../../materialize/css/materialize.min.css" media="
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet" />
<title>EscolAlura</title>
<script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
<script type="text/javascript" src="../../materialize/js/inicializar.js"></script>
</head>
<body class="grey lighten-3">
  <div id="pesquisaAluno" class="container">
    <h3 class="main-title center">Pesquisar Alunos por Classificação</h3>
    <form class="col s12" action="/nota/pesquisar" method="get">
      <div class="row">
        <div class="row">
          <div class="input-field col s12">
            <select name="classificacao" id="classificacao">
                <option value="" disabled="disabled" selected="selected">Escolha aqui</option>
                <option value="aprovados">Aprovados</option>
                <option value="reprovados">Reprovados</option>
              </select>
```

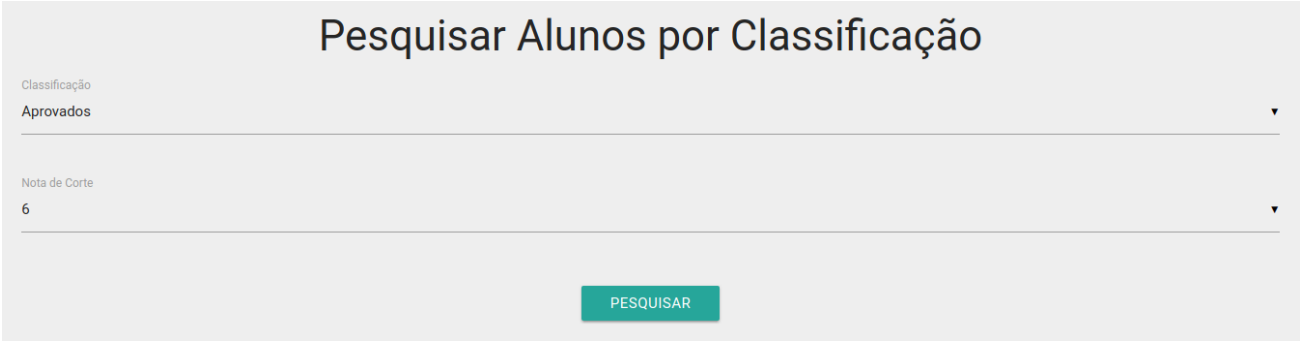
```
            <label>Classificação</label>
          </div>
        </div>
        <div class="row">
          <div class="input-field col s12">
            <select name="notacorte" id="notacorte">
                <option value="" disabled="disabled" selected="selected">Escolha aqui</option>
                <option value="7">7</option>
                <option value="6">6</option>
                <option value="5">5</option>
              </select>
              <label>Nota de Corte</label>
          </div>
        </div>

        <div class="row">
          <div class="input-field col s12 center">
            <button class="btn waves-effect waves-light" type="submit">Pesquisar</button>
          </div>
        </div>
      </div>
    </form>
    <div th:if="${alunos} != null">
      <div class="row">
        <div class="input-field col s12">
          <table class="striped responsive-table">
            <thead>
              <tr>
                <th style="text-align: center;">Nome</th>
                <th style="text-align: center;">Dt. Nascimento</th>
                <th style="text-align: center;">Curso</th>
              </tr>
            </thead>
            <tr th:each="aluno : ${alunos}">
              <td style="text-align: center;" th:text="${aluno.nome}"></td>
              <td style="text-align: center;"
                th:text="${#dates.format(aluno.dataNascimento, 'dd/MM/yyyy')}"></td>
              <td style="text-align: center;" th:text="${aluno.curso.nome}"></td>
            </tr>
          </table>
        </div>
      </div>
    </div>
  </div>

  <script type="text/javascript" src="../../materialize/js/materialize.min.js"></script>

</body>
</html>
```

Quando abrirmos a página de pesquisa, veremos que os critérios são dois (a nota de corte e o status), ou seja, poderemos indicar que a nota de corte é `6` e queremos listar os alunos reprovados ou aprovados. Os critérios são fixos no próprio HTML, é possível customizar as opções. O formulário é semelhante ao da imagem abaixo:



Agora precisaremos capturar o valor destes dois campos na `NotasController` e fazer a pesquisa em nossa coleção de alunos. Primeiro fazendo a captura dos dados:

```
@GetMapping("/nota/pesquisar")
public String pesquisarPor(@RequestParam("classificacao") String classificacao, @RequestParam("
  List<Aluno> alunos = repository.pesquisarPor(classificacao, Double.parseDouble(notaCorte));
  model.addAttribute("alunos", alunos);
  return "nota/pesquisar";
}
```

Note que já temos um método `pesquisarPor` em nosso repositório, porém, ele só pesquisa por nome. Portanto, criaremos este novo, que considera a classificação de aprovado ou não, além da nota de corte.

```
public List<Aluno> pesquisaPor(String classificacao, double nota) {
  criarConexao();
  MongoCollection<Aluno> alunoCollection = this.bancoDeDados.getCollection("alunos", Aluno.clas
  MongoCursor<Aluno> resultados = null;

  if(classificacao.equals("reprovados")) {
    resultados = alunoCollection.find(Filters.lt("notas", nota)).iterator();
  }else if(classificacao.equals("aprovados")) {
    resultados = alunoCollection.find(Filters.gte("notas", nota)).iterator();
  }

  List<Aluno> alunos = popularAlunos(resultados);

  fecharConexao();

  return alunos;

}
```

Perceba que a refatoração feita para criar o método `popularAlunos` está sendo reusada. A classificação do status do aluno aqui serve apenas para indicar qual o tipo de comparação utilizada no filtro da coleção. Caso o aluno seja reprovado utilizaremos o método `lt` e, caso seja aprovado, usaremos o método `gte`, siglas para *lower than* e *greater than or equals* ("menor que" e "maior ou igual a").

Vamos testar! Verifique se os resultados são apresentados corretamente, experimente adicionar várias notas e alunos, e utilize a busca para filtrar diferentes resultados.