



# ***DESAFIO ORANGE TALENTS***

GABRIELA VIANA DE OLIVEIRA

# O DESAFIO

Criar uma API REST que precisará gerar números aleatórios para loteria utilizando linguagem Java e Spring + Hibernate. Para facilitar na identificação da pessoa, o programa deverá associar cada número a um e-mail.

## ENTENDENDO A API REST

API REST, também chamada de API RESTful, é uma interface de programação de aplicações que segue conformidade com as restrições da arquitetura REST. A sigla REST significa Representational State Transfer (Transferência Representacional de Estado, em português). Uma interface de programação de aplicações (API) é um conjunto de definições e protocolos para criar e integrar softwares de aplicações.

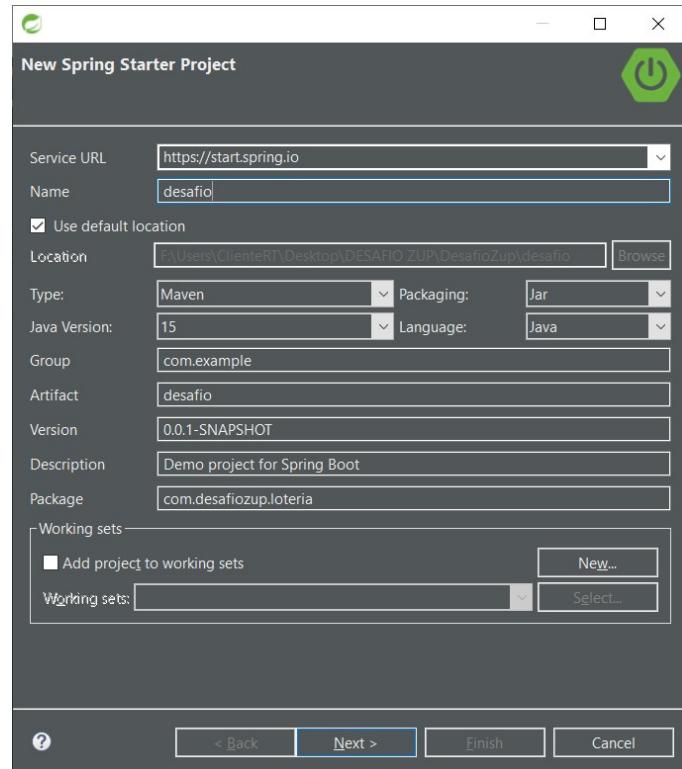
## LINGUAGEM UTILIZADA



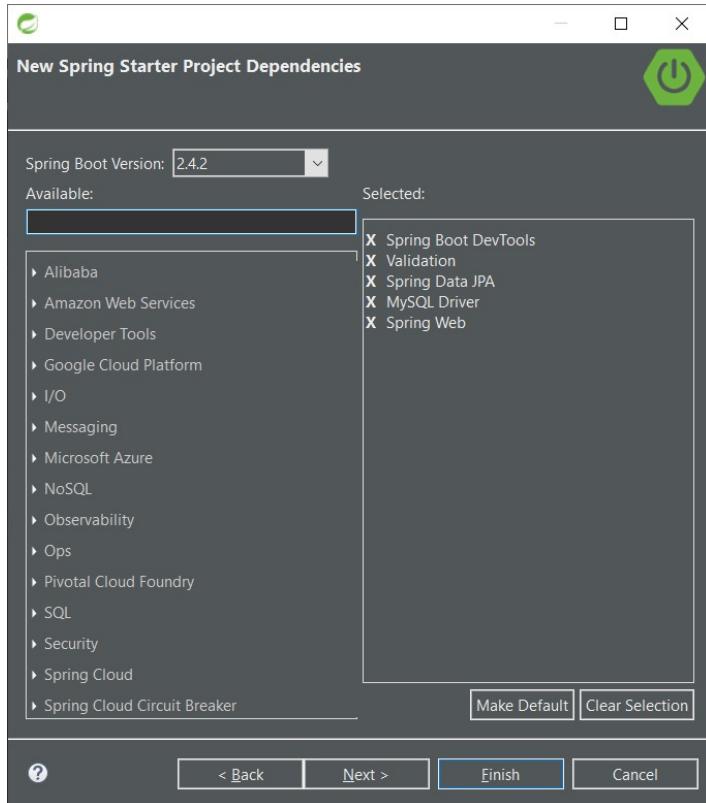
- Java e Spring + Hibernate;
- Postman para testes;
- Xampp como servidor;
- MySQL Workbench 8.0 para banco de dados.

# INICIANDO O PROJETO

Existem formas diferentes de criar um projeto Spring. A escolhida para realização desse desafio, foi pelo programa Spring Tool Suite 3. Com ele não existe a necessidade de criar o projeto no site Spring Initializr, baixar para o computador e importa-lo no Eclipse, o que economiza tempo e trabalho.



# TECNOLOGIAS UTILIZADAS NO SPRING

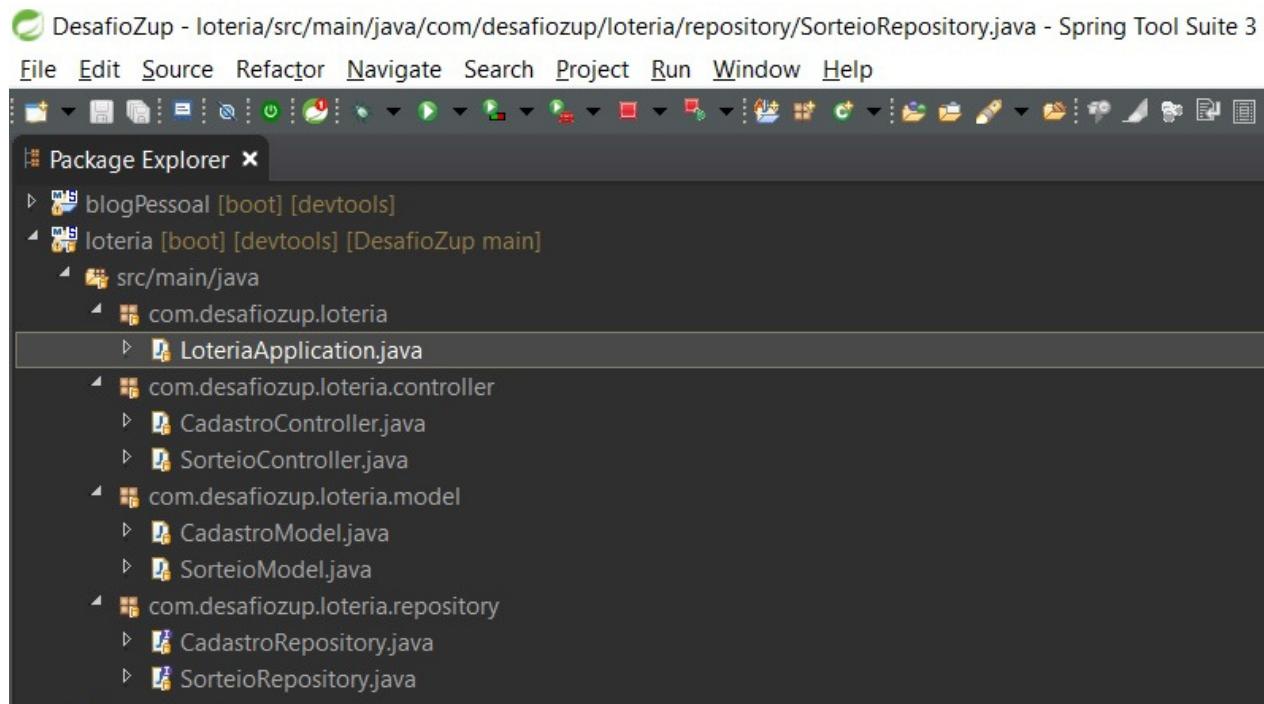


- Spring Web: cria um web service;
- Spring Boot DevTools: atualiza automaticamente as mudanças no projeto;
- Spring Data JPA: utiliza o Data e outros recursos do JPA e Hibernate;
- Validation: permite a validação de atributos, propriedades, parâmetros e retornos de métodos, parâmetros de construtores, a criação de validadores customizados, o agrupamento de validações, entre outras funcionalidades. Tudo isto serve para evitar o armazenamento de dados inválidos no repositório de dados usado pelo sistema.
- MySQL Driver: responsável pela conexão entre a API e o Banco de Dados MySQL.

---

## 06

Com o projeto pronto, foram criadas duas classes para Model (CadastroModel e SorteioModel), duas Interfaces para Repository (CadastroRepository e SorteioRepository) e duas classes Controller (CadastroController e SorteioController).



Cada classe tem suas Annotations, complementos que uma classe, atributo ou método pode ter. Na Model, responsável pelo instanciamento da API, utilizamos a biblioteca javax.persistence e as principais anotações: @Entity, @Table, @Id, @GeneratedValue, @NotNull, @Size etc, e criamos os getters e setters para que possamos modelar os métodos.

Na classe CadastroModel, foram criadas três colunas: Id, Nome e Email, sendo a coluna Id a chave-primária da tabela. Cada usuário cadastrado recebe um Id único, e seu número é gerado automaticamente no ato do cadastro. A Annotation @OneToMany é a responsável pela relação entre as tabelas Cadastro e Sorteio, onde um usuário cadastrado pode ter vários números sorteados.

# CADASTRO MODEL

```

DesafioZup - loteria/src/main/java/com/desafiozup/loteria/model/CadastroModel.java - Spring Tool Suite 3
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer CadastrModel.java
I blogPessoal [root] [devtools] DesafioZup main
└ loteria
    └ com.desafiozup.loteria
        └ loteriaApplication.java
        └ loteriaController.java
        └ sorteioController.java
        └ com.desafiozup.loteria.model
            └ CadastroModel.java
        └ com.desafiozup.loteria.repository
            └ CadastroRepository.java
            └ SorteioRepository.java
        └ config
            └ application.properties
    └ IRE System Library (JavaSE-13)
    └ Maven Dependencies
    └ pom.xml
    └ Servers [DesafioZup main]
Servers Boot Dashboard
Pivotal tc Server Developer Edition v4.0 [Stopped]
loteria - LoteriaApplication (Spring Boot App) C:\Program Files\Java\jdk-15\bin\javaw.exe (22 de jan de 2021 22:29:14)
Console Progress Problems
CadastrModel.java
1 package com.desafiozup.loteria.model;
2
3 import java.util.List;
4
5 @Entity
6 @Table(name = "cadastro")
7 public class CadastroModel {
8
9     // criando a tabela cadastro, que salvará os dados do usuário.
10
11     @Id
12     @GeneratedValue(strategy = GenerationType.IDENTITY)
13     private Integer id;
14
15     @Column(name="Nome")
16     @NotNull
17     private String nome;
18
19     @Column(name="Email")
20     @NotNull
21     private String email;
22
23     @OneToOne(mappedBy = "sorteio", cascade = CascadeType.ALL)
24     @JsonIgnoreProperties("sorteio")
25     private List<SorteioModel> sorteio;
26
27     // Getters and Setters
28     public Integer getId() {
29         return id;
30     }
31
32     public String getNome() {
33         return nome;
34     }
35
36     public void setNome(String nome) {
37         this.nome = nome;
38     }
39
40     public String getEmail() {
41         return email;
42     }
43
44     public void setEmail(String email) {
45         this.email = email;
46     }
47
48     public List<SorteioModel> getSorteio() {
49         return sorteio;
50     }
51
52     public void setSorteio(List<SorteioModel> sorteio) {
53         this.sorteio = sorteio;
54     }
55
56     @Override
57     public String toString() {
58         return "CadastroModel{" +
59             "id=" + id +
60             ", nome='" + nome + '\'' +
61             ", email='" + email + '\'' +
62             ", sorteio=" + sorteio +
63             '}';
64
65     }
66
67     @Override
68     public int hashCode() {
69         return Objects.hash(id);
70     }
71
72     @Override
73     public boolean equals(Object o) {
74         if (this == o) return true;
75         if (o == null || getClass() != o.getClass()) return false;
76         CadastroModel cadastroModel = (CadastroModel) o;
77         return id.equals(cadastroModel.id);
78     }
79
80     @Override
81     public String toString() {
82         return "CadastroModel{" +
83             "id=" + id +
84             ", nome='" + nome + '\'' +
85             ", email='" + email + '\'' +
86             ", sorteio=" + sorteio +
87             '}';
88     }
89
90 }

```

```

DesafioZup - loteria/src/main/java/com/desafiozup/loteria/model/CadastroModel.java - Spring Tool Suite 3
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer CadastrModel.java
I blogPessoal [root] [devtools] DesafioZup main
└ loteria
    └ com.desafiozup.loteria
        └ loteriaApplication.java
        └ loteriaController.java
        └ sorteioController.java
        └ com.desafiozup.loteria.model
            └ CadastroModel.java
        └ com.desafiozup.loteria.repository
            └ CadastroRepository.java
            └ SorteioRepository.java
        └ config
            └ application.properties
    └ IRE System Library (JavaSE-13)
    └ Maven Dependencies
    └ pom.xml
    └ Servers [DesafioZup main]
Servers Boot Dashboard
Pivotal tc Server Developer Edition v4.0 [Stopped]
loteria - LoteriaApplication (Spring Boot App) C:\Program Files\Java\jdk-15\bin\javaw.exe (22 de jan de 2021 22:29:14)
Console Progress Problems
CadastrModel.java
44*     public void setId(Integer id) {
45         this.id = id;
46     }
47
48     public String getNome() {
49         return nome;
50     }
51
52     public void setNome(String nome) {
53         this.nome = nome;
54     }
55
56     public String getEmail() {
57         return email;
58     }
59
60     public void setEmail(String email) {
61         this.email = email;
62     }
63
64     public List<SorteioModel> getSorteio() {
65         return sorteio;
66     }
67
68     public void setSorteio(List<SorteioModel> sorteio) {
69         this.sorteio = sorteio;
70     }
71
72     @Override
73     public String toString() {
74         return "CadastroModel{" +
75             "id=" + id +
76             ", nome='" + nome + '\'' +
77             ", email='" + email + '\'' +
78             ", sorteio=" + sorteio +
79             '}';
80
81     }
82
83     @Override
84     public int hashCode() {
85         return Objects.hash(id);
86     }
87
88     @Override
89     public boolean equals(Object o) {
90         if (this == o) return true;
91         if (o == null || getClass() != o.getClass()) return false;
92         CadastroModel cadastroModel = (CadastroModel) o;
93         return id.equals(cadastroModel.id);
94     }
95
96     @Override
97     public String toString() {
98         return "CadastroModel{" +
99             "id=" + id +
100             ", nome='" + nome + '\'' +
101             ", email='" + email + '\'' +
102             ", sorteio=" + sorteio +
103             '}';
104
105 }

```

## 09

Na classe SorteioModel, foram criados: a chave estrangeira e o método dos números que serão sorteados. Também foi adicionada uma Annotation @Temporal, para que seja mostrado quando o cadastro foi feito. Já a Annotation @ManyToOne serve para criar a relação da tabela SorteioModel com a CadastroModel, onde vários números sorteados podem ser gerados para um único usuário.

The screenshot shows the Spring Tool Suite 3 interface with the following details:

- Package Explorer:** Shows the project structure under the package `com.desafiozup.loteria.model`. It includes `SorteioModel.java`, `CadastroModel.java`, and `LoteriaApplication.java`.
- SorteioModel.java:** The code editor displays the Java code for the `SorteioModel` class. The code includes annotations like `@Entity`, `@Table(name = "sorteio")`, `@Id`, `@GeneratedValue(strategy = GenerationType.IDENTITY)`, `@Temporal(TemporalType.TIMESTAMP)`, and `@ManyToOne`. The class contains private fields for `id`, `date`, and `sorte`, and methods for generating random numbers from 1 to 50 and setting them to the `CadastroModel`.
- Outline:** A sidebar on the right shows the class structure with fields `numero1` through `numero6` and the `cadastro` association.
- Console:** The bottom console shows the output of a command-line query: `sorteio0.numero6 as numero9_1_1_`.

10

The screenshot shows the Spring Tool Suite interface with the following details:

- Project Explorer:** Shows the project structure for "DesafioZip".
- Code Editor:** Displays the content of `SorteoModel.java`. The code defines a class `SorteoModel` with fields `id`, `date`, and `sorte`, and methods for setting and getting these values.
- Outline View:** Shows the class hierarchy and member details for `SorteoModel`.
- Spring Explorer:** Shows the configuration for the project.
- Console:** Displays the output of the application.

The screenshot shows the Spring Tool Suite interface with the following details:

- Project Explorer:** Shows the project structure for "DesafioZip".
- Code Editor:** Displays the content of `SorteoModel.java`. The code includes methods for setting and getting multiple numbers (`numero1` through `numero6`) and a `CadastroModel`.
- Outline View:** Shows the class hierarchy and member details for `SorteoModel`.
- Spring Explorer:** Shows the configuration for the project.
- Console:** Displays the output of the application.

Para construir a Interface Repository, utilizamos a biblioteca java.util, importamos o JPA da aplicação e usamos a anotação @Repository. No Repository contém a lógica da persistência de dados. É onde acessamos a lógica de dados do nosso programa (se comunica diretamente com o banco de dados). Precisa injetar o JPA.

Como o parâmetro de busca solicitado foi achar o usuário por email, no CadastroRepository indicamos o método que será utilizado.

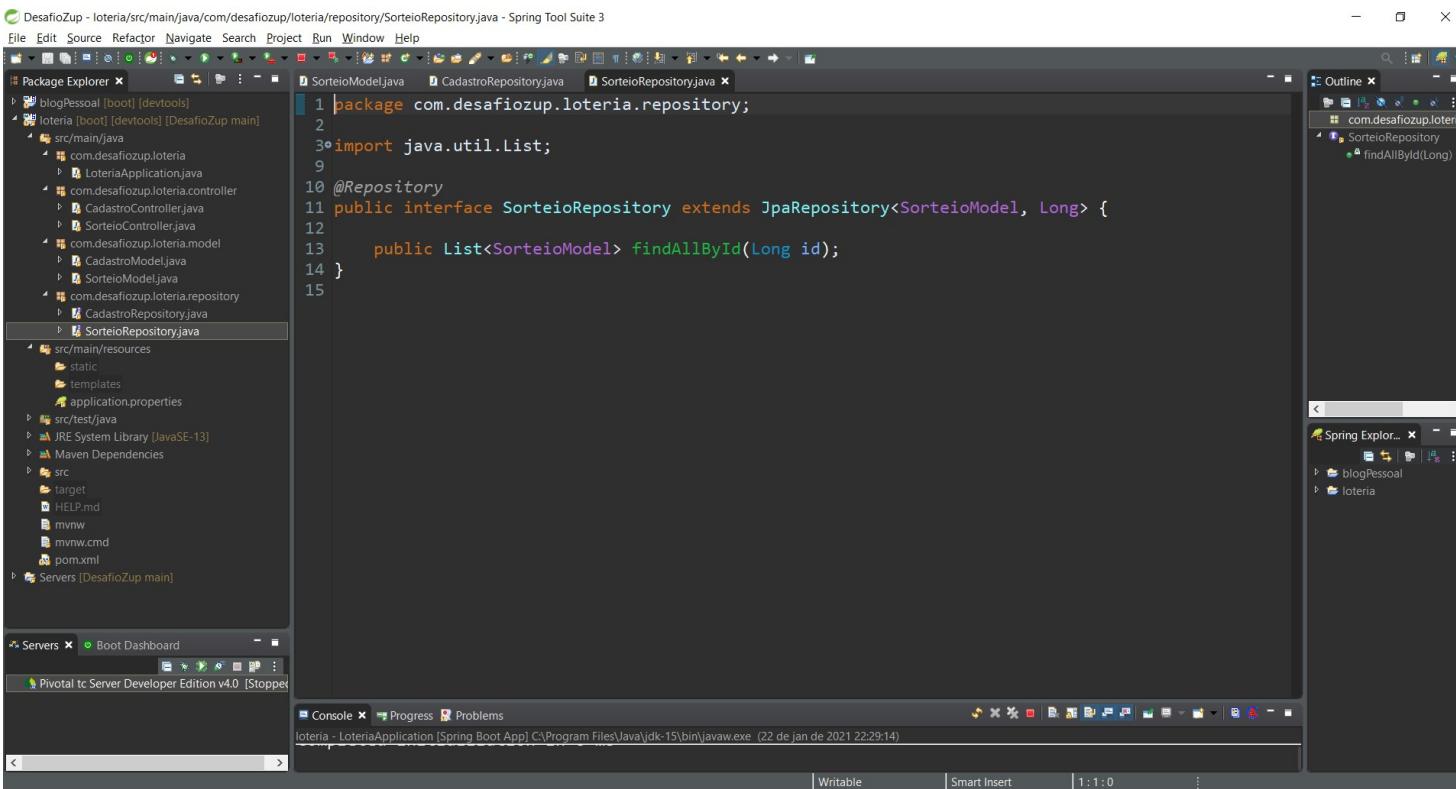
# CADASTRO REPOSITORY

The screenshot shows the Spring Tool Suite 3 interface with the following details:

- Project Explorer:** Shows the project structure for "DesafioZup - loteria". It includes packages like blogPessoal, loteria, SorteoModel, CadastroModel, CadastroController, SorteoController, and CadastroRepository.
- Editor:** Displays the code for `CadastroRepository.java` in the `CadastroModel` package. The code defines a repository interface extending `JpaRepository<CadastroModel, Long>` with a method `findAllByEmailContainingIgnoreCase(String email)`.
- Outline View:** Shows the class structure with `CadastroRepository` and its method `findAllByEmailConta...`.
- Spring Explorer:** Shows the application context with beans `blogPessoal` and `loteria`.
- Console:** Shows the command line output for running the application.

## 13

O mesmo é feito no SorteioRepository, porém o método escolhido foi por meio de sua chave estrangeira, Id, a qual relacionará os números sorteados ao e-mail solicitado.



The screenshot shows the Spring Tool Suite 3 interface with the following details:

- Title Bar:** DesafioZup - loteria/src/main/java/com/desafiozup/loteria/repository/SorteioRepository.java - Spring Tool Suite 3
- File Menu:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbars:** Standard, Java, Database, XML, CSS, JavaScript, CSS3, and others.
- Package Explorer:** Shows the project structure under the 'DesafioZup main' tab, including packages like blogPessoal, loteria, and com.desafiozup.loteria, and files like SorteioModel.java, CadastroRepository.java, and SorteioRepository.java.
- Editor:** Displays the content of SorteioRepository.java:

```
1 package com.desafiozup.loteria.repository;
2
3 import java.util.List;
4
5 @Repository
6 public interface SorteioRepository extends JpaRepository<SorteioModel, Long> {
7
8     public List<SorteioModel> findAllById(Long id);
9
10}
```
- Outline View:** Shows the class hierarchy and methods, including SorteioRepository and its method findAllById(Long id).
- Servers View:** Shows a Pivotal tc Server Developer Edition v4.0 [Stopped] entry.
- Console View:** Shows the output of 'loteria - LoteriaApplication [Spring Boot App] C:\Program Files\Java\jdk-15\bin\javaw.exe (22 de jan de 2021 22:29:14)'.
- Spring Explorer View:** Shows the 'loteria' application context.

Na classe Controller (onde serão criados os métodos), administraremos e manipulamos todos os id points. Utiliza-se a biblioteca java.util e as principais anotações são: @RestController, @RequestMapping (é o caminho), @CrossOrigin (define as origens aceitas), @Autowired (injeta o repository na classe controller). E as anotações do CRUD são: @GetMapping, @PostMapping, @PutMapping e @DeleteMapping.

Na classe CadastroController, criamos o método para que os dados possam ser manipulados.

# CADASTROCONTROLLER

The screenshot shows the Spring Tool Suite (STS) interface with the following details:

- Project Explorer:** Shows the project structure for "DesafioZUp - loteria". It includes packages like blogPessoal, loteria, com.desafiozup.loteria, and com.desafiozup.loteria.controller. Under controller, there is a file named CadastroController.java.
- Code Editor:** The main window displays the code for CadastroController.java. The code defines a REST controller for managing CadastroModel entities using a CadastroRepository.
- Outline View:** On the right side, the Outline view shows the class structure with methods: GetAll(), getByEmail(@PathVariable String email), post(@RequestBody CadastroModel cadastro), and delete(@PathVariable long id).
- Servers:** A section at the bottom left shows a Pivotal tc Server Developer Edition v4.0 [Stopped] server entry.
- Console:** The bottom right shows the console output for "loteria - LoteriaApplication [Spring Boot App] C:\Program Files\Java\jdk-15\bin\javaw.exe (22 de jan de 2021 22:29:14)".

## O mesmo é feito com o SorteioController.

The screenshot shows the Spring Tool Suite interface with the following details:

- Package Explorer:** Shows the project structure for "DesafioZup - loteria/src/main/java/com/desafiozup/loteria/controller". The file "SorteioController.java" is selected and open in the editor.
- Editor:** Displays the Java code for the SorteioController:1 package com.desafiozup.loteria.controller;
2
3 import java.util.List;
4
5 @RestController
6 @CrossOrigin(origins="\*", allowedHeaders= "\*")
7 @RequestMapping("/sorteio")
8 public class SorteioController {
9
10 @Autowired
11 private SorteioRepository repository;
12
13 @GetMapping
14 public ResponseEntity<List<SorteioModel>> getAll() {
15 return ResponseEntity.ok(repository.findAll());
16 }
17
18 @PostMapping
19 public ResponseEntity<SorteioModel> post (@RequestBody SorteioModel sorteio) {
20 return ResponseEntity.status(HttpStatus.CREATED).body(repository.save(sorteio));
21 }
22
23 @DeleteMapping("/{id}")
24 public void delete(@PathVariable Long id) {
25 repository.deleteById(id);
26 }
27 }
- Outline:** Shows the class structure with methods: getAll(), post(), and delete().
- Servers:** Shows "Pivotal tc Server Developer Edition v4.0 [Stopped]".
- Console:** Shows the output of the application running: "Loteria - LoteriaApplication [Spring Boot App] C:\Program Files\Java\jdk-15\bin\javaw.exe (22 de jan de 2021 22:29:14)".

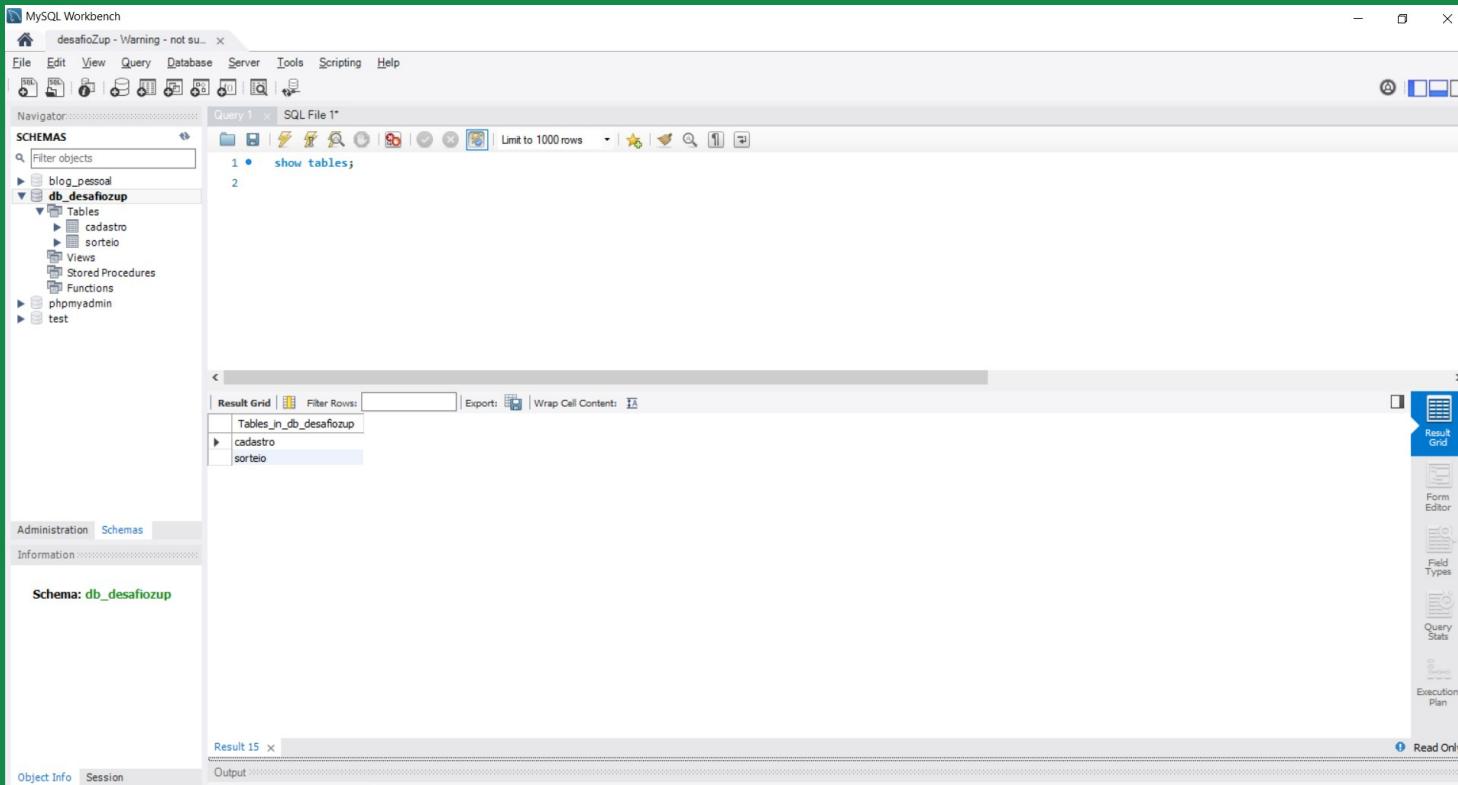
17

Para que possamos criar as tabelas diretamente do STS no MySQL, escrevemos a configuração no diretório application.properties.

```
## ESTRATEGIA PARA CRIAÇÃO DE TABELAS
spring.jpa.hibernate.ddl-auto=update
##
## MySQL
spring.datasource.url=jdbc:mysql://localhost/db_desafiozup?createDatabaseIfNotExist=true&serverTimezone=UTC&useSSL=false
spring.datasource.username=root
spring.datasource.password=
spring.jpa.properties.hibernate.show_sql=true
server.port=8080
##
## O QUE É?
spring.jpa.properties.hibernate.use_sql_comments=true
spring.jpa.properties.hibernate.format_sql=true
logging.level.com.desafiozup.loteria=DEBUG
```

18

Ao executar a classe LoteriaApplication, as tabelas são criadas no MySQL Workbench.



19

Para saber se a aplicação está funcionando e salvando os dados com sucesso, testamos no Postman.

The screenshot shows two instances of the Postman application interface. Both instances have the URL `http://localhost:8080/cadastro` and the method set to GET. The left instance displays a JSON response with multiple objects, each containing an 'id' field (ranging from 1 to 29) and a 'numeros' array. The right instance also displays a similar JSON response, but it includes an additional field 'sorteo1' which contains a single object with an 'id' of 2 and a 'numeros' array. A large green arrow points from the text 'Aqui podemos identificar a relação de @OneToMany.' towards the 'sorteo1' field in the right instance's response.

Aqui podemos identificar a relação de @OneToMany.

```
POST /cadastro
{
    "id": 2,
    "numeros": [
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
    ],
    "sorteo1": [
        {
            "id": 2,
            "date": "2021-01-23T22:51:30.000+00:00",
            "numeros": [
                1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
            ]
        }
    ]
}
```

```
POST /cadastro
{
    "id": 2,
    "numeros": [
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
    ],
    "sorteo1": [
        {
            "id": 2,
            "date": "2021-01-23T22:55:29.000+00:00",
            "numeros": [
                1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
            ]
        }
    ]
}
```

Outra opção, é checar no navegador da internet, através do endereço  
<http://localhost:8080/cadastro>



```
// 20210123210912
// http://localhost:8080/cadastro
[
  {
    "id": 1,
    "name": "Adriana",
    "email": "adri@adri.com",
    "sorteio": [
      {
        "id": 2,
        "date": "2021-01-23T22:51:00.000+00:00",
        "numero1": 1,
        "numero2": 14,
        "numero3": 23,
        "numero4": 39,
        "numero5": 40,
        "numero6": 54
      },
      {
        "id": 3,
        "date": "2021-01-23T22:55:29.000+00:00",
        "numero1": 4,
        "numero2": 11,
        "numero3": 22,
        "numero4": 36,
        "numero5": 42,
        "numero6": 55
      }
    ]
  },
  {
    "id": 2,
    "name": "Gabi",
    "email": "gabi@gabi.com",
    "sorteio": [
      {
        "id": 4,
        "date": "2021-01-23T22:56:52.000+00:00",
        "numero1": 1,
        "numero2": 19,
        "numero3": 27,
        "numero4": 34,
        "numero5": 47,
        "numero6": 59
      }
    ]
  }
]
```

Também podemos  
identificar a relação  
de @OneToMany.



**OBRIGADA!**

GABRIELA VIANA DE OLIVEIRA