Tecnologias Para Back-End

Prof. JUNIO FIGUEIRÊDO

JUNIOINF@GMAIL.COM

AULA 03 – SPRING DATA JPA

MySQL

Preparando o Ambiente: MySQL

- Com a aplicação criada, vamos desenvolver as funcionalidades do

No curso será utilizado o MySQL como sistema gerenciador de banco de dados, sendo recomendado que você também o utilize.

- Caso você não tenha o MySQL instalado em seu computador, pode seguir o tutorial de instalação disponível nesse artigo:
- MySQL: da instalação até a configuração

Adicionando Novas Dependências

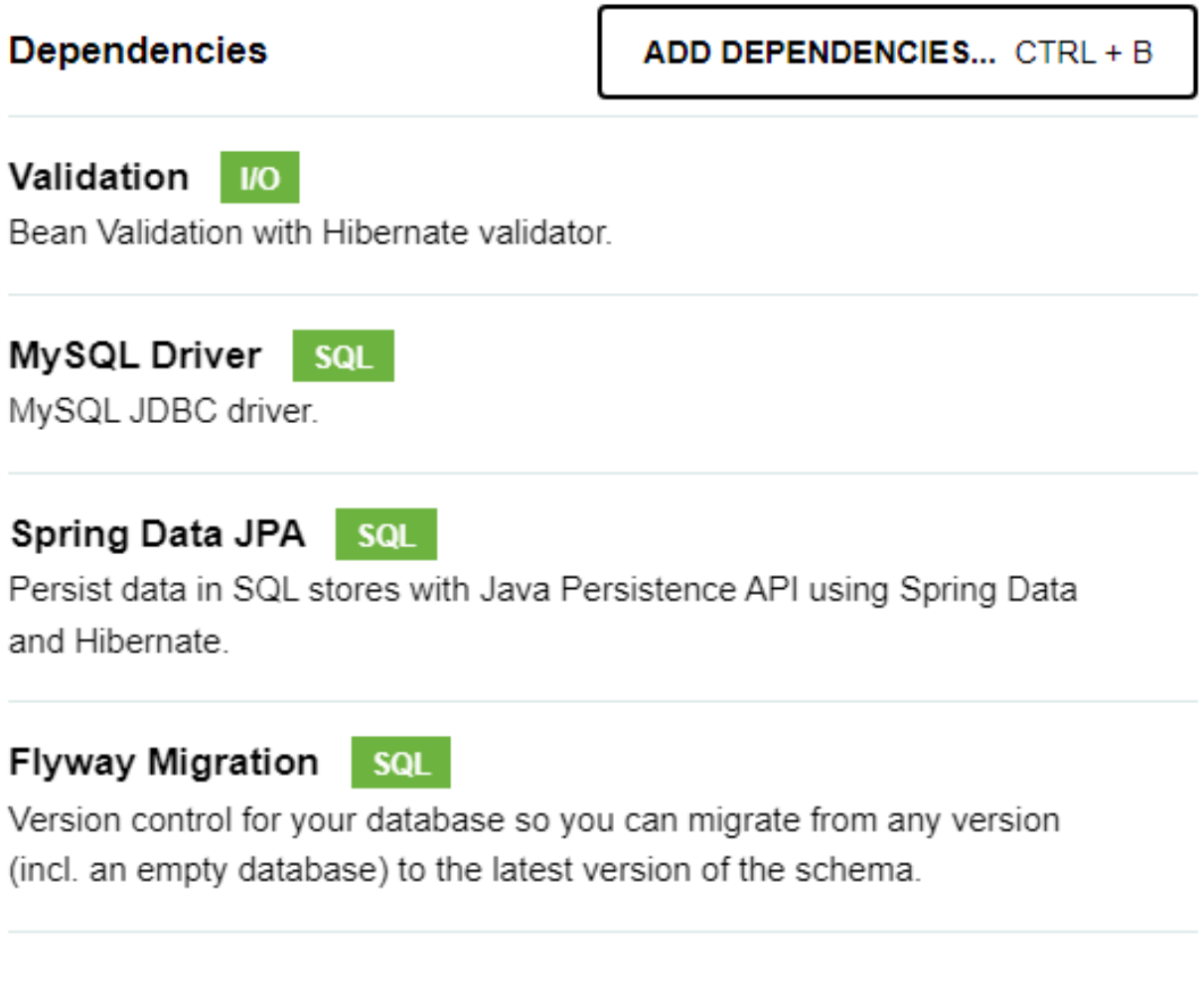
Spring

- Abra o spring initializr
- Vamos adicionar novas dependências

Spring

Selecione o Projeto: Maven

Linguagem Java e sua versão 21



Depois clica no botão

Agora vc vai copiar tudo que está entre.

```
<dependencies>
```

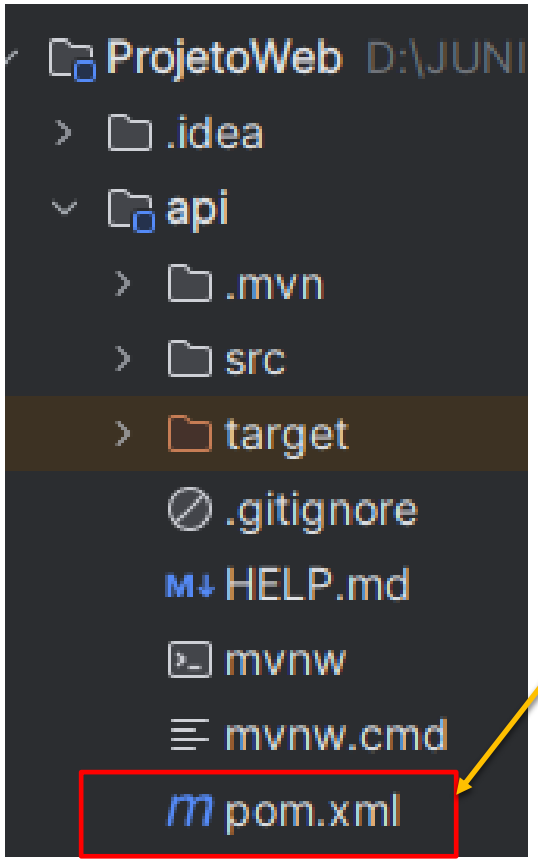
e

```
</dependencies>
```

Obs. Não copie a dependência de teste

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
```

Spring



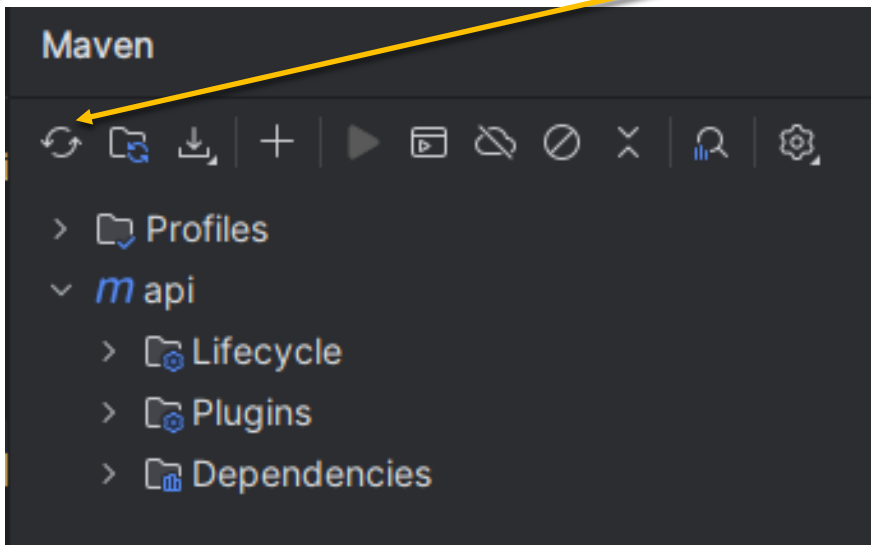
- Dê uma parada no servidor!!!!
- Uma vez copiado basta levá-lo para nosso arquivo "pom.xml". Coloque entre

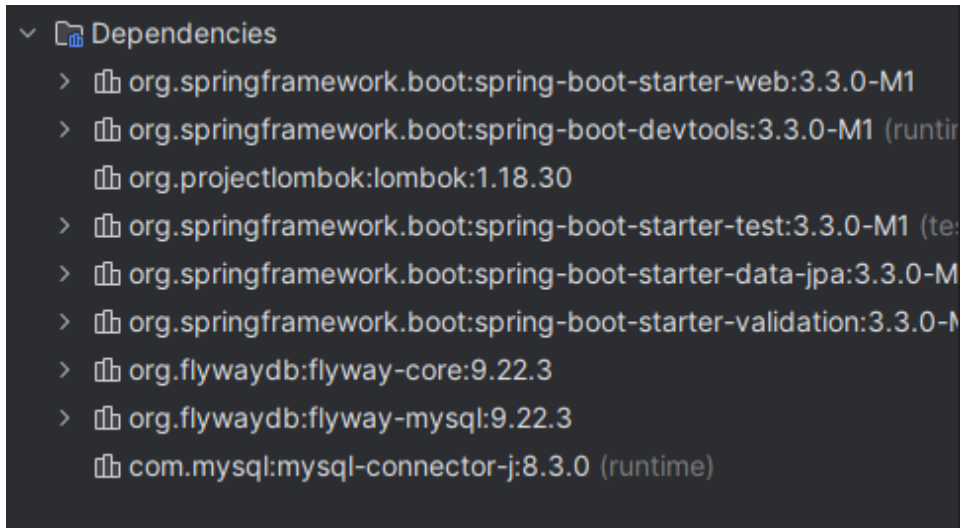
```
<dependencies>
```

e

```
</dependencies>
```

- Acessar a aba do Maven, à direita da IDE, e clicar em "Reload All Maven Projects", para recarregar as dependências do projeto.

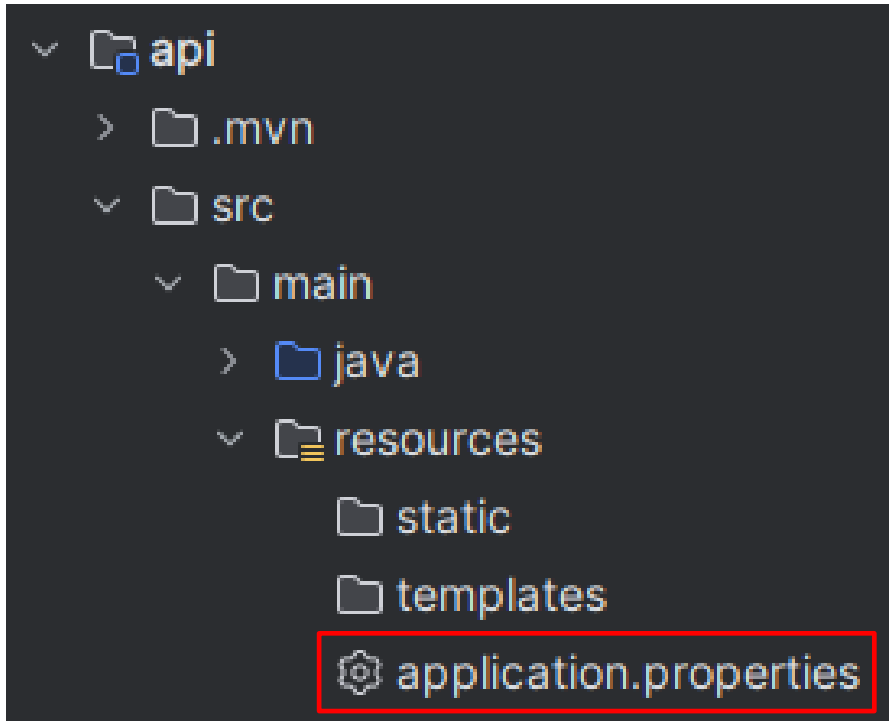




Spring

- Caso você reinicie o servidor, irá dar um erro de configuração de DataSource, BANCO DE DADOS.
- Vamos fazer isso no arquivo "src > main > resources > application.properties".

Por padrão, o Spring Boot acessa as configurações definidas no arquivo application.properties, que usa um formato de chave=valor:



Spring

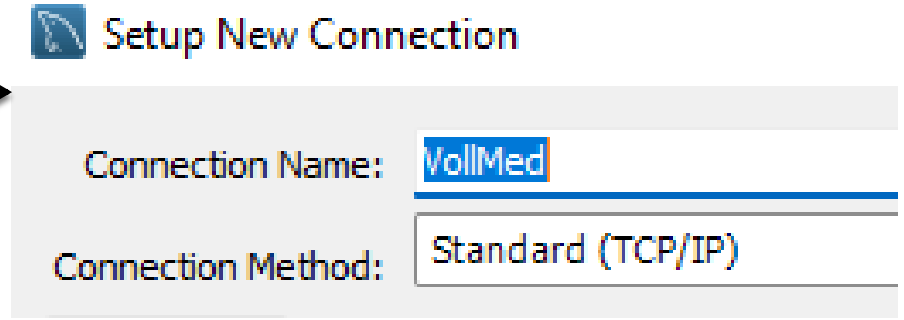
- Precisamos adicionar três propriedades a ele,
  - a URL de conexão com o banco de dados,
  - o login e a senha do banco de dados

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/vollmed_api
spring.datasource.username=root
spring.datasource.password=root
```

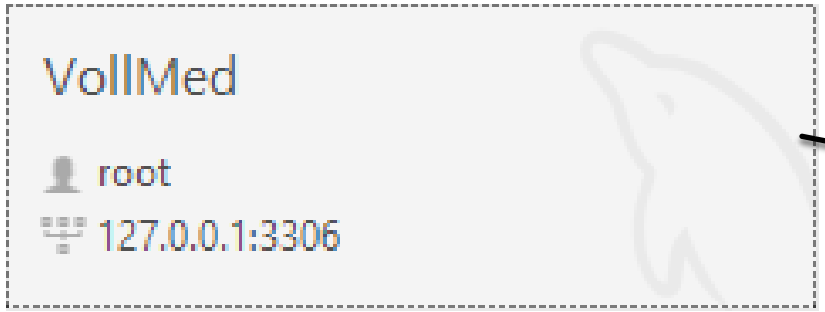
  - Vollmed_api é o Nome do BD que será criado
  - Obs: Usuario e Senha são definidos na instalação do MySql

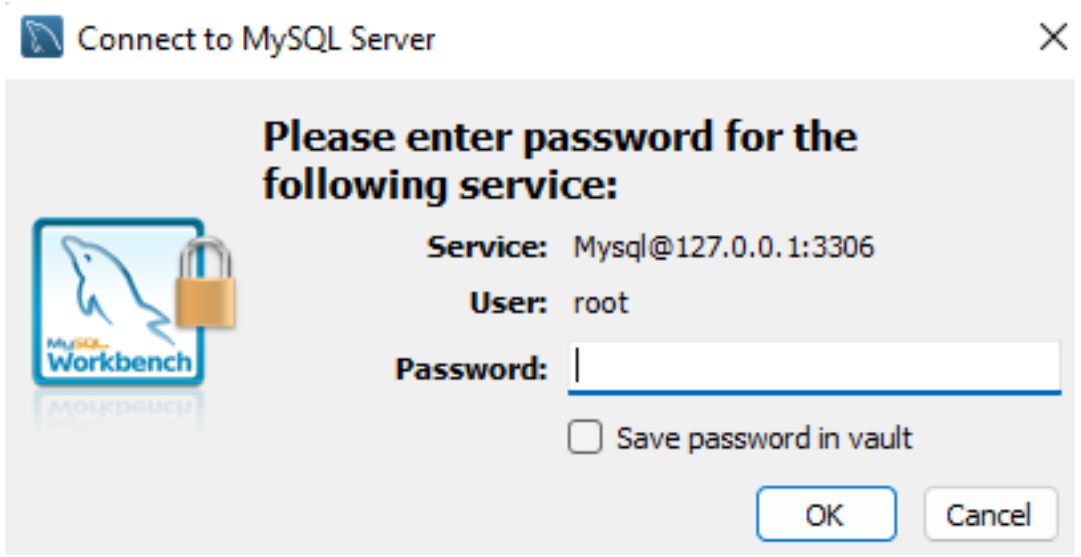
- Agora o Spring Data conseguirá se conectar ao banco de dados da aplicação corretamente.
- O que precisamos fazer é acessar o MySQL e criar o database, porque esse não é um processo automático.

- Vamos criar o BD, abra o MySQL Workbench



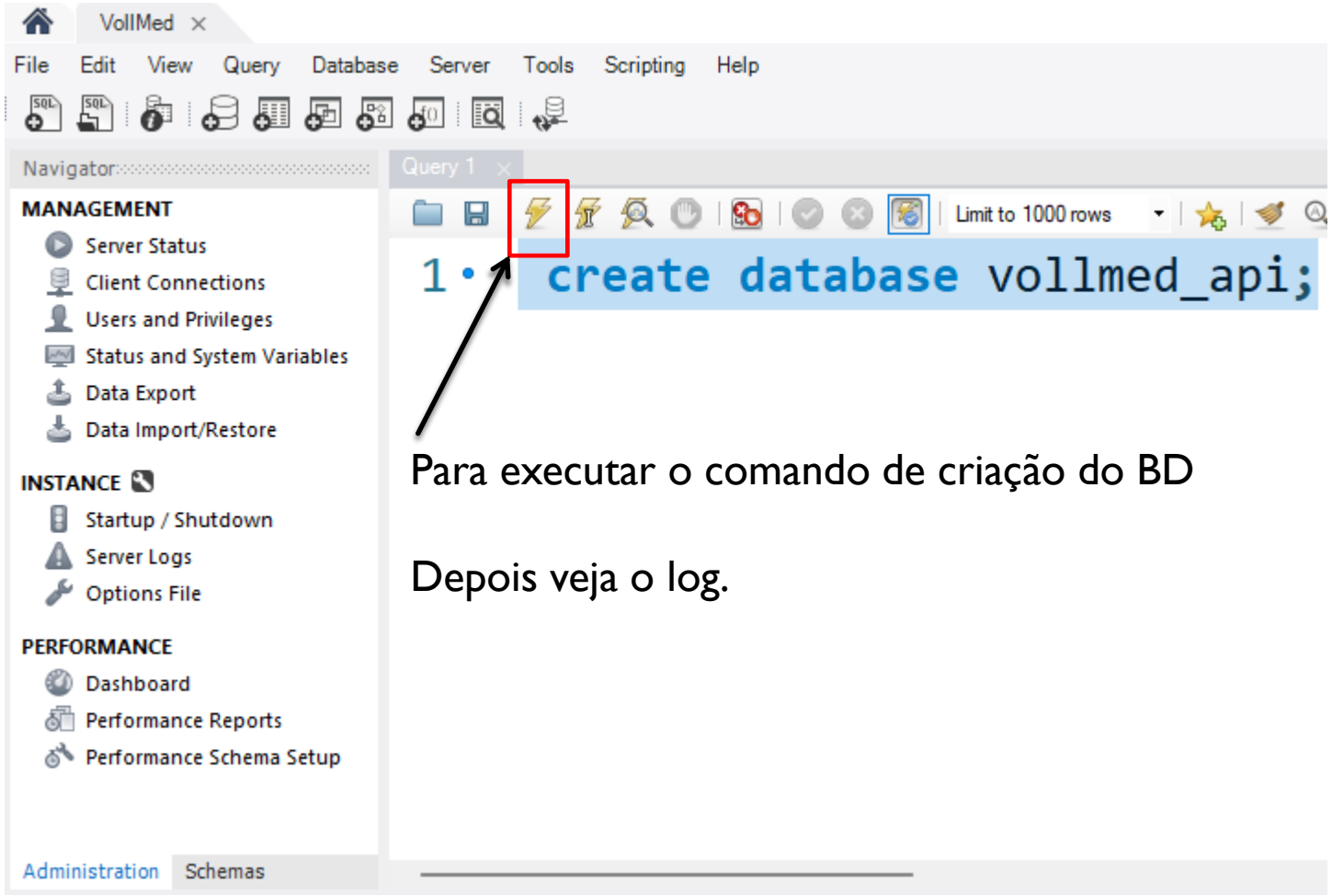
- Clique no botão Ok





Spring

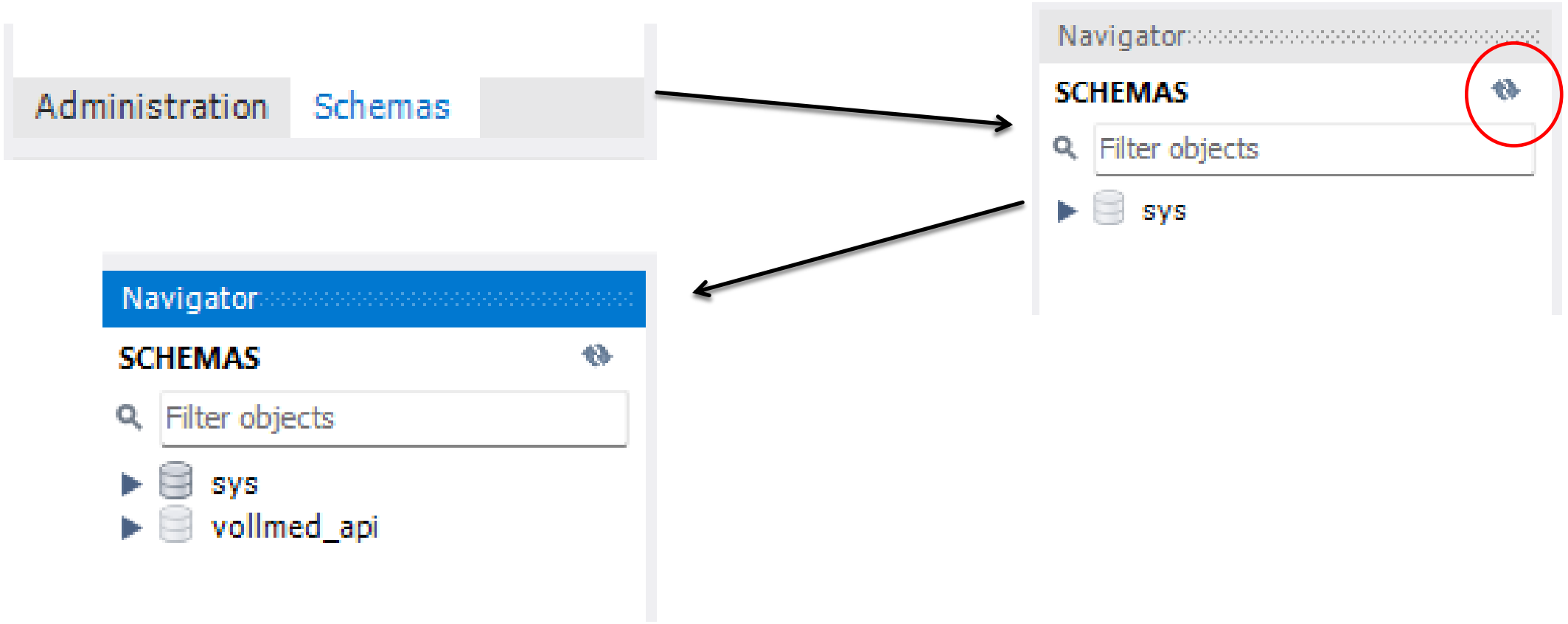
- Agora criaremos o database, com o comando
- create database vollmed_api;



Para executar o comando de criação do BD

Depois veja o log.

Spring



- gora criaremos o database, com o comando
- create database vollmed_api;

Entidades JPA

- Vamos voltar para "MedicoController.java".

```java
@RestController
@RequestMapping("medicos")
public class MedicoController {

    no usages
    @PostMapping
    public void cadastrar(@RequestBody DadosCadastroMedico dados) {
        System.out.println(dados);

    }
}
```

- Nele, estamos recebendo um DTO, representado pelo record DadosCadastroMedico e dando um System.out.

Spring

- O que precisamos fazer, porém, é pegar o objeto e salvá-lo no banco de dados.
- O Spring Data JPA utiliza o JPA como ferramenta de mapeamento de objeto relacional.
- Por isso, criaremos uma entidade JPA para representar uma tabela no banco de dados.

Spring

Criando a Entidade Médicos

- Dentro do Pacote Médico crie a “Class Medico”.

São os mesmos atributos
Declarados no Java Record
Que é um DTO

A Classe Medico é uma
Entidade JPA

Temos que criar uma
Entidade Endereço dentro
Do Pacote Endereço

```java
package med.voll.api.medico;


public class Medico {


        private Long id;
        private String nome;
        private String email;
        private String crm;
        private Especialidade especialidade;
        private Endereco endereco;


}
```

Criando a Entidade Médicos – CÓDIGO PARA COPIAR

```
package med.voll.api.medico;
 public class Medico {
        private Long id;
        private String nome;
        private String email;
        private String crm;
        private Especialidade especialidade;
        private Endereco endereco;
 }
```

Spring

- **DTO** é uma classe referênte aos campos que devem aparecer na página web (ou ferramente de teste, como Insomnia).
- **JPA** é uma classe referêntes aos campos que serão mapeados como tabelas no banco de dados.

Criando a Entidade Endereco

- Dentro do Pacote Endereco crie a “Class Endereco”.

```
package med.voll.api.endereco;

public class Endereco {

        private String logradouro;
        private String bairro;
        private String cep;
        private String numero;
        private String complemento;
        private String cidade;
        private String uf;

}
```

Criando a Entidade Endereco – CÓDIGO PARA COPIAR

```
package med.voll.api.endereco;
public class Endereco {
        private String logradouro;
        private String bairro;
        private String cep;
        private String numero;
        private String complemento;
        private String cidade;
        private String uf;
}
```

- Agora que tudo está mapeado corretamente, vamos voltar à **entidade Medico**, que até o momento é uma classe Java, com nada da JPA.
- Vamos adicionar as anotações da JPA para transformar isso em uma entidade:

Spring

Criando a Entidade Médicos

```java
package med.voll.api.medico;

import jakarta.persistence.*;
import med.voll.api.endereco.Endereco;

no usages
@Table(name = "medicos")
@Entity(name = "Medico")
public class Medico {

    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    no usages
    private String nome;
    no usages
    private String email;
    no usages
    private String crm;


    no usages
    @Enumerated(EnumType.STRING)
    private Especialidade especialidade;


    no usages
    @Embedded
    private Endereco endereco;
```

Mapeando a Entidade Medico – CÓDIGO PARA COPIAR

```
@Table(name = "medicos")
@Entity(name = "Medico")
public class Medico {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;
  private String nome;
  private String email;
  private String crm;
```

```
@Enumerated(EnumType.STRING
)
   private Especialidade
especialidade;


   @Embedded
   private Endereco endereco;
```

Mapeamento - Principais **Anotações**

```
@Entity
class Pessoa {
  @Id
  private Long id;
}
```

@Entity

- É a principal anotação do JPA.
- Ela deve aparecer antes do nome de uma classe
- Diz que a classe é uma entidade, que representa uma tabela do banco de dados.

Mapeamento - Anotações

@Id

- É utilizada para indicar qual atributo de uma classe anotada com @Entity será mapeado para a chave primária da tabela correspondente à classe.
- Geralmente o atributo anotado com @Id é do tipo Long.

```
@Entity
class Pessoa {
  @Id
  private Long id;
}
```

Mapeamento - Anotações

@GeneratedValue

- Geralmente vem acompanhado da anotação @Id.
- Serve para indicar que o valor (auto-incremento) de um atributo que compõe uma chave primária deve ser gerado pelo banco no momento em que um novo registro é inserido.

```
@Entity
class Pessoa {
  @Id
  @GeneratedValue
  private Long id;
}
```

Mapeamento - Anotações

Estratégia IDENTITY

- O tipo de geração *IDENTITY* nada mais e do que o famoso autoincremento do banco de dados.
- Para as pessoas que trabalham com *MySQL* ou *SQLServer*, esse e o padrão.

```
@Id
@GeneratedValue(strategy=GenerationType.IDENTITY)
private Long codigo;
```

Mapeamento - Principais **Anotações**

```
@Embedded
private Endereco endereco;
```

@Embedded

- Vamos usar Embeddable Attribute da JPA para que Endereco fique em uma classe separada, mas faça parte da mesma tabela de Medicos junto ao banco de dados.
- Para que isso funcione, vamos acessar a classe Endereco e adicionar, no topo do código, a anotação @Embeddable logo acima da classe.
- Vamos importar a biblioteca Lombok, para gerar os códigos Java que faltam automaticamente.
- Adicionaremos @Getter, @NoArgsConstructor, @AllArgsConstructor, @EqualsAndHashCode(of = "id")

Anotações da Entidade Médicos

```
@Table(name = "medicos")
@Entity(name = "Medico")
@Getter
@NoArgsConstructor
@AllArgsConstructor
@EqualsAndHashCode(of = "id")
public class Medico {
```

Anotações da Entidade Endereco

```
@Embeddable
@Getter
@NoArgsConstructor
@AllArgsConstructor
public class Endereco {
```

Agora finalizamos o mapeamento da JPA...

Spring

DTO - Data Transfer Object