**Sumário**

[Introdução 2](#_Toc121942749)

[Pré requisitos 3](#_Toc121942750)

[Java JDK 17 3](#_Toc121942751)

[IntelliJ IDE 3](#_Toc121942752)

[1 - Criação do projeto 4](#_Toc121942753)

[Spring initializr 4](#_Toc121942754)

[Estrutura do projeto 4](#_Toc121942755)

[Executando o projeto 5](#_Toc121942756)

[Build automático 6](#_Toc121942757)

[Controller 6](#_Toc121942758)

[2 - Requisições POST 7](#_Toc121942759)

[Criação do cadastro de médicos 7](#_Toc121942760)

[Fazendo a requisição 8](#_Toc121942761)

[3 - Spring Data JPA 9](#_Toc121942762)

[Instalação do MySQL 9](#_Toc121942763)

[Adicionando dependências 9](#_Toc121942764)

[Executando a aplicação sem configurações 10](#_Toc121942765)

[Definindo configurações com o banco de dados 11](#_Toc121942766)

[Criando banco de dados 11](#_Toc121942767)

[Executando a aplicação com configurações 12](#_Toc121942768)

[Criação de entidades JPA 12](#_Toc121942769)

[Interfaces Repository 13](#_Toc121942770)

[Migrations com Flyway 16](#_Toc121942771)

[Executando a aplicação e fazendo a requisição: 20](#_Toc121942772)

[Bean Validation 20](#_Toc121942773)

[Anotando o DTO com o Validation: 21](#_Toc121942774)

[Anotando controller 21](#_Toc121942775)

[Testando as validações 22](#_Toc121942776)

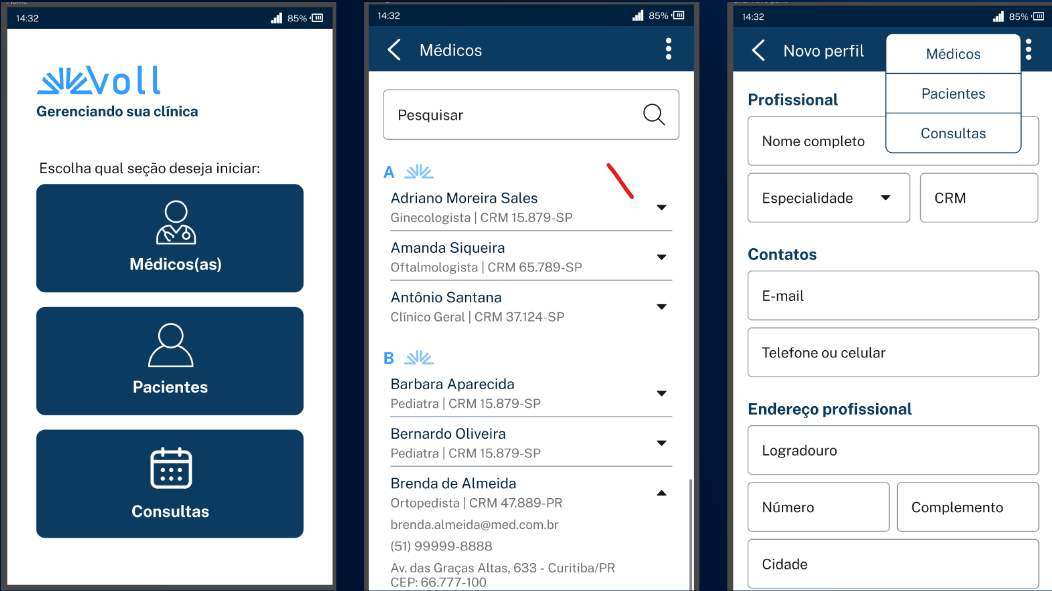
[Nova migration: Alteração propriedades do projeto 23](#_Toc121942777)

# Introdução

**Objetivos**

* Desenvolvimento de uma API Rest
* CRUD (Create, Read, Update e Delete)
* Validações
* Paginação e ordenação

Trabalharemos em um projeto de uma clínica médica fictícia. Temos uma empresa chamada Voll Med, que possui uma clínica que precisa de um aplicativo para monitorar o cadastro de médicos, pacientes e agendamento de consultas.

Será um aplicativo com algumas opções, em que a pessoa que for usar pode fazer o CRUD, tanto de médicos quanto de pacientes e o agendamento e cancelamento das consultas.  
  


# Pré requisitos

# Java JDK 17

<https://www.oracle.com/java/technologies/downloads/>

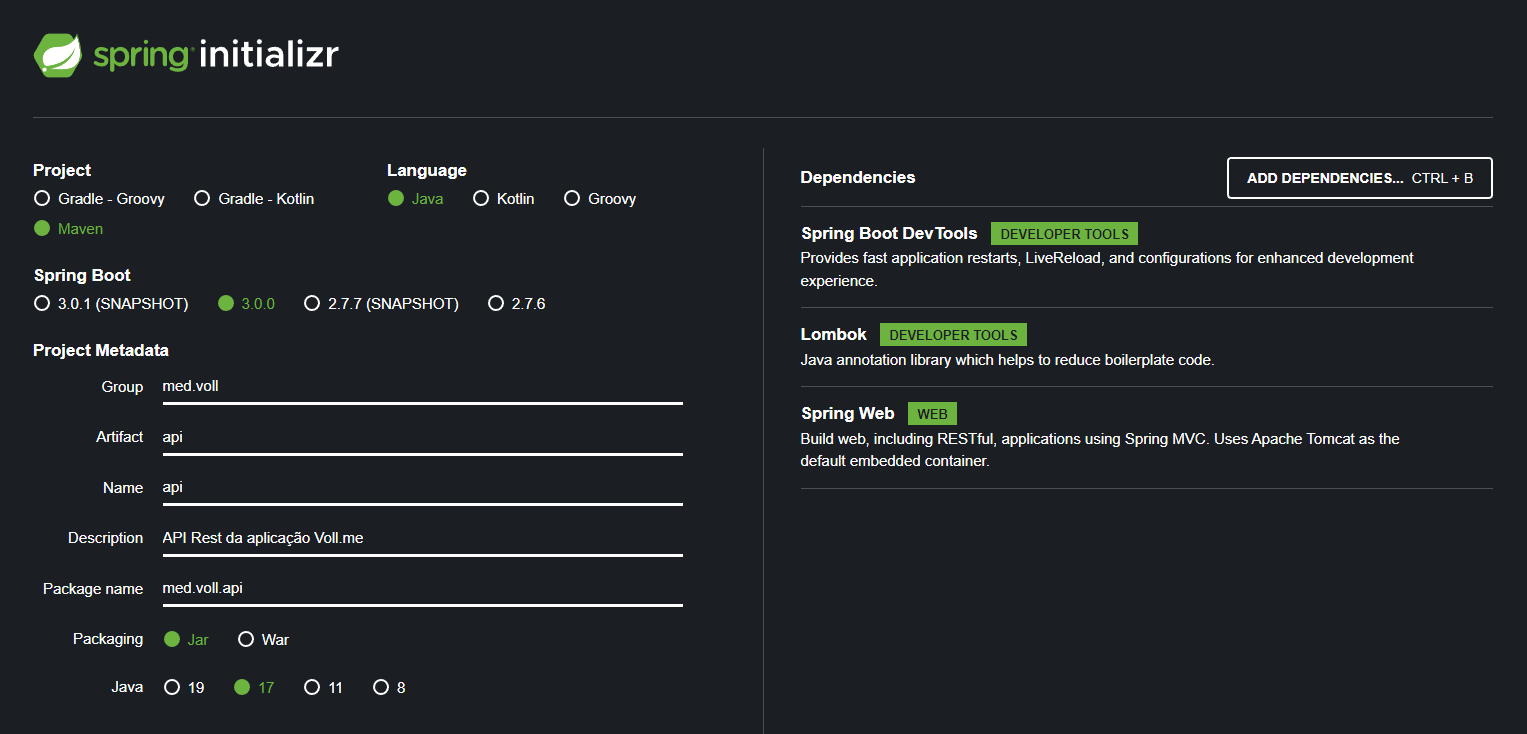
# IntelliJ IDE

<https://www.jetbrains.com/pt-br/idea/>

# 1 - Criação do projeto

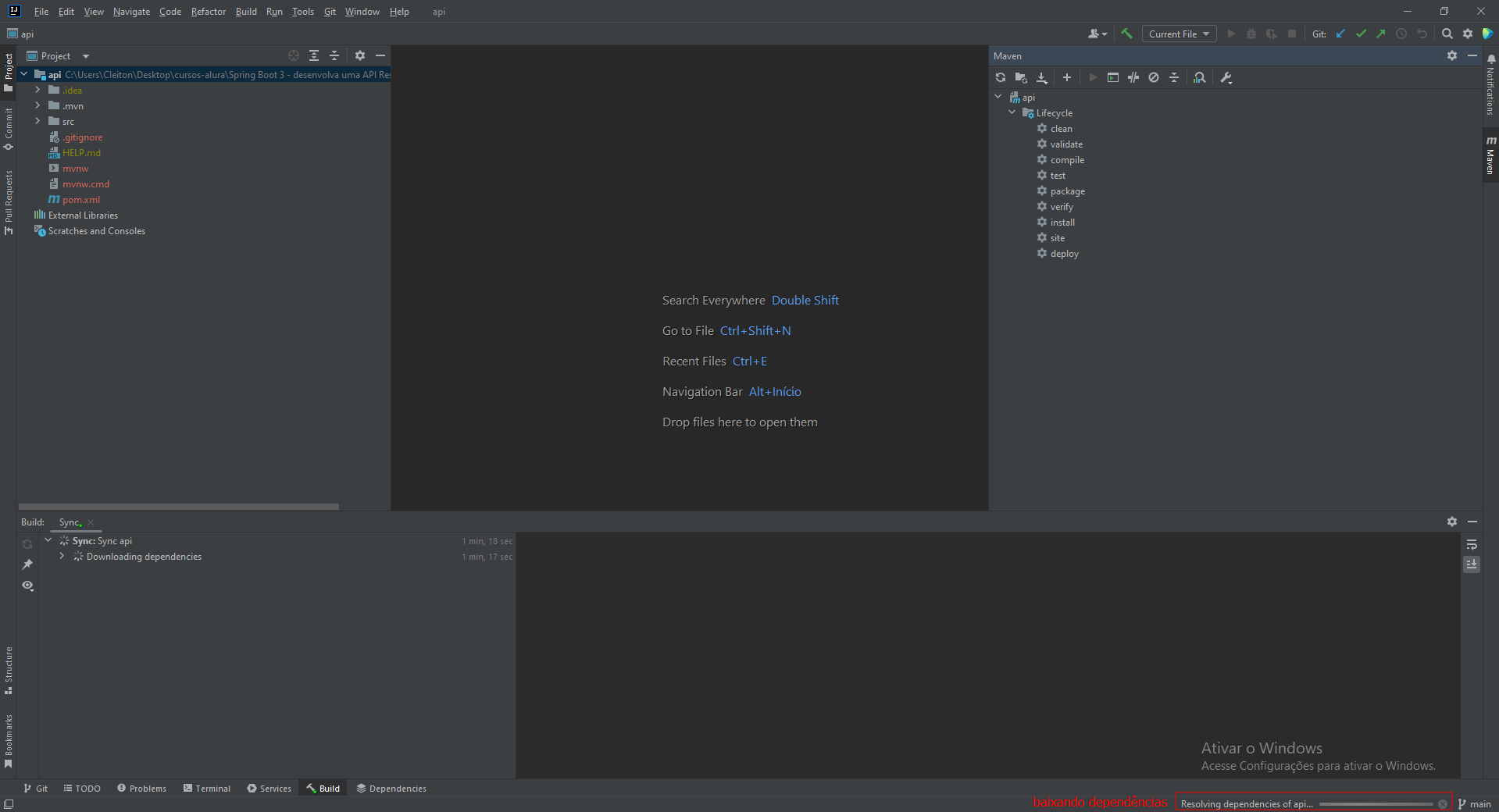
# Spring initializr

https://start.spring.io/



# Estrutura do projeto

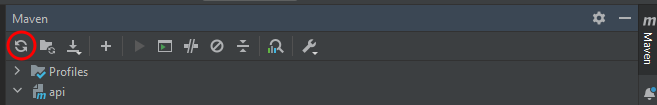
Validar se dependências já foram baixadas



Todas as dependências que foram adicionadas devem estar listadas

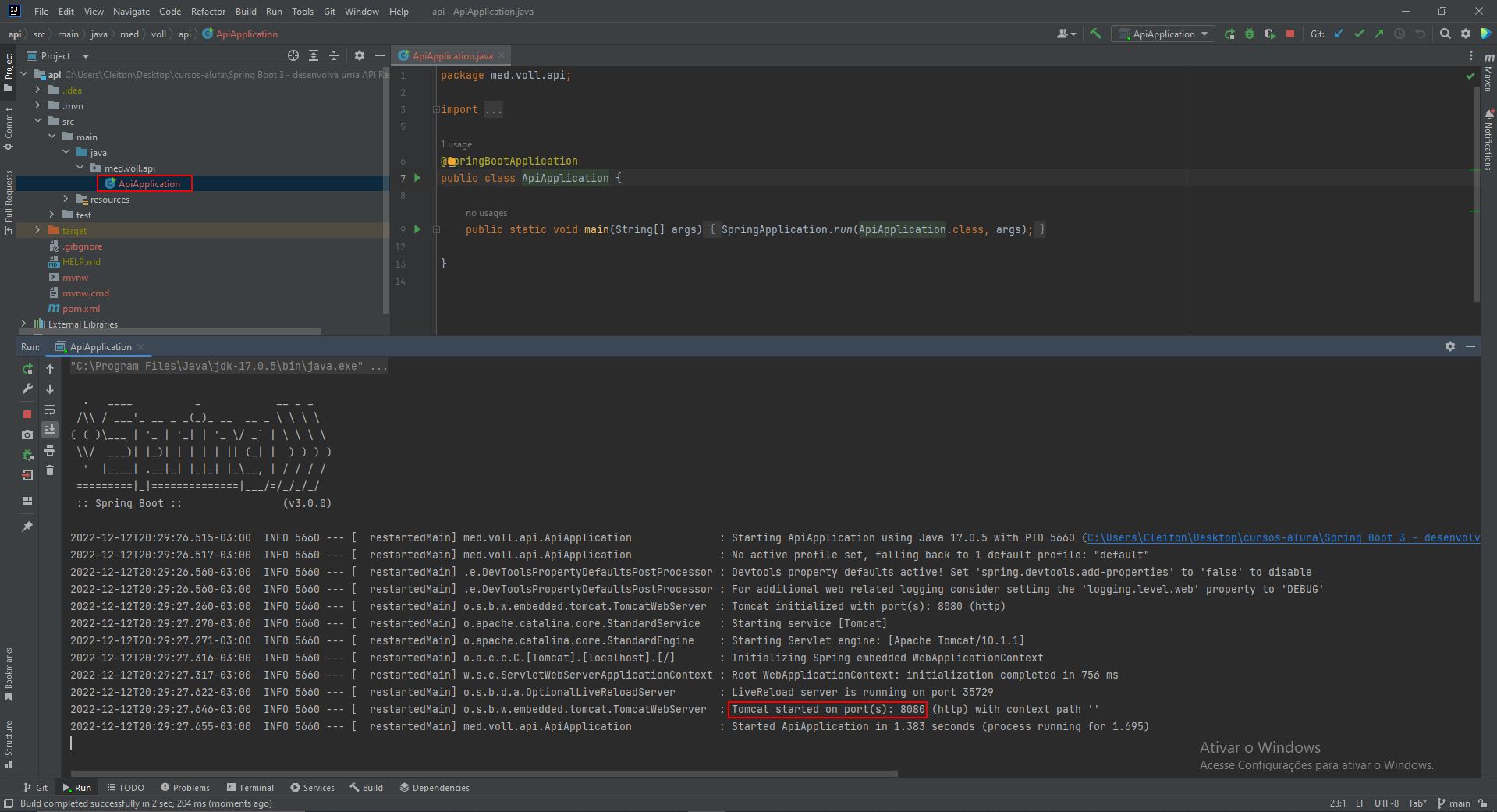


Caso der algum problema, basta clicar em reload que o projeto será recarregado e tentará baixas as dependências que falta.

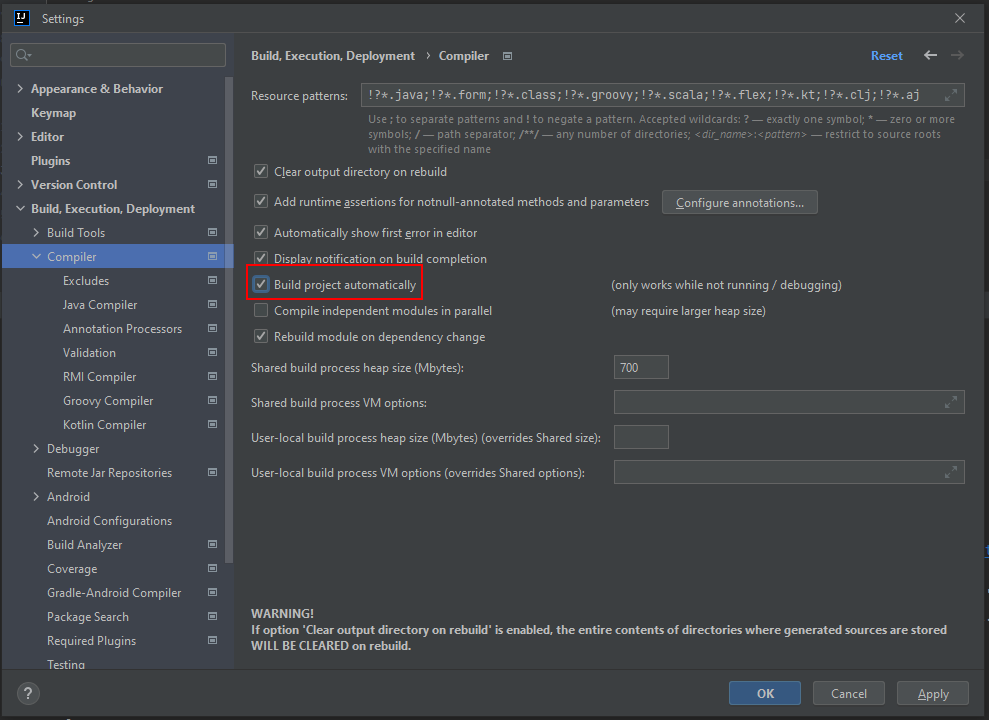


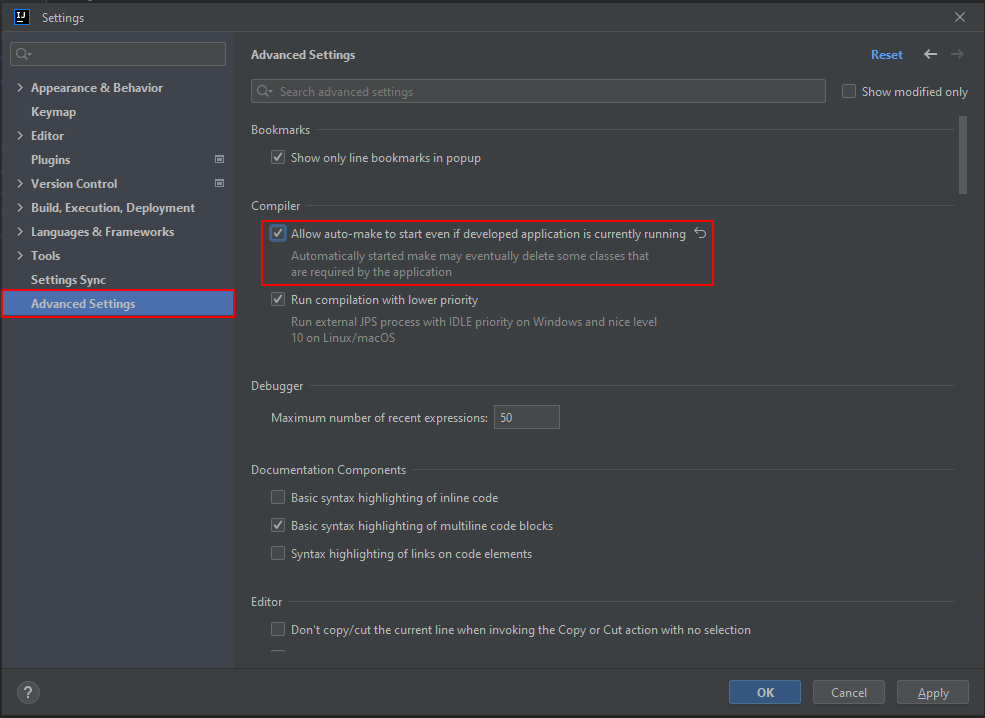
# Executando o projeto

Antigamente para executar um projeto Spring, era necessário criar um servidor e colocar o projeto dentro. Agora com o SpringBoot, o servidor está dentro do projeto, incluso.

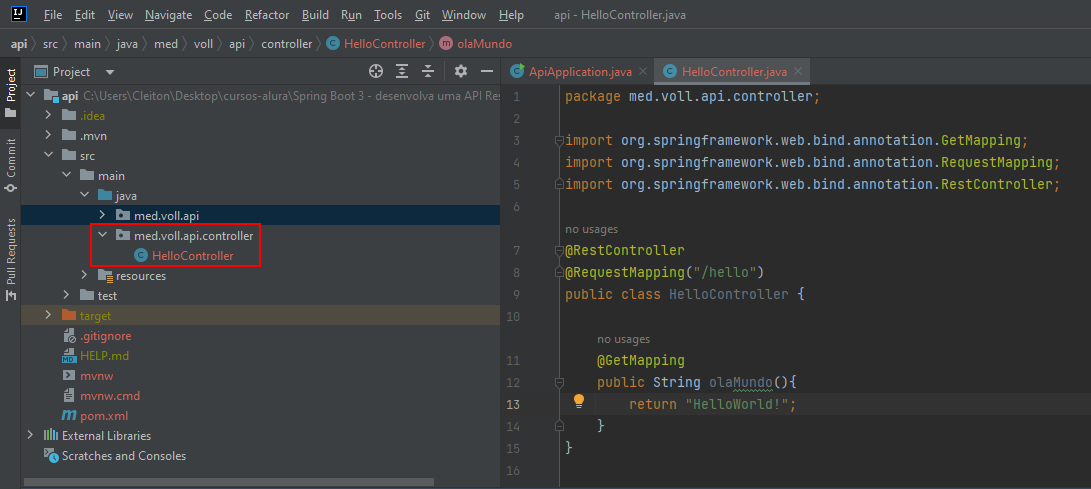


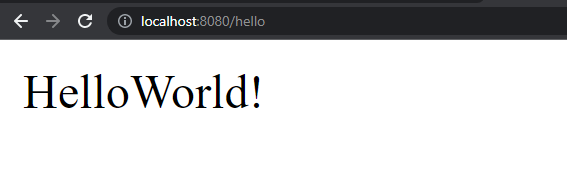
# Build automático





# Controller



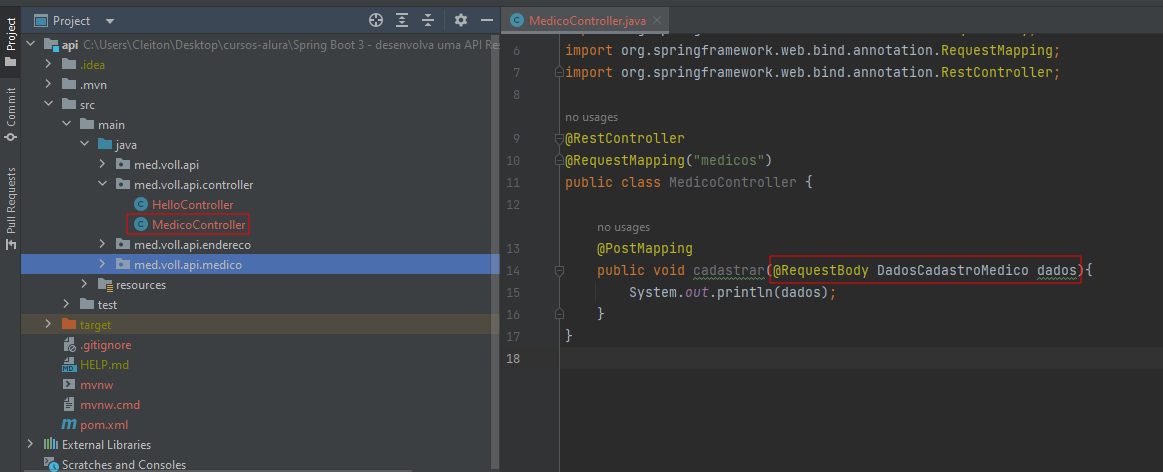


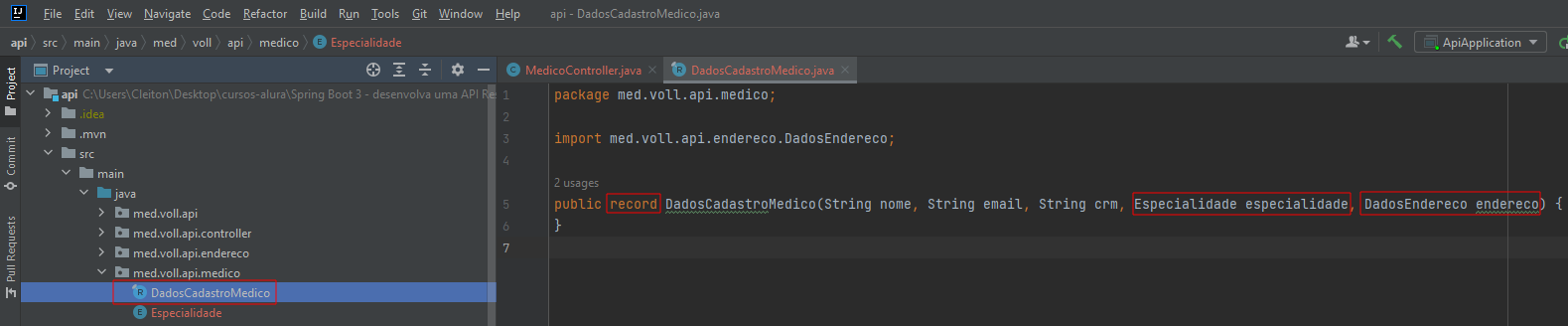
# 2 - Requisições POST

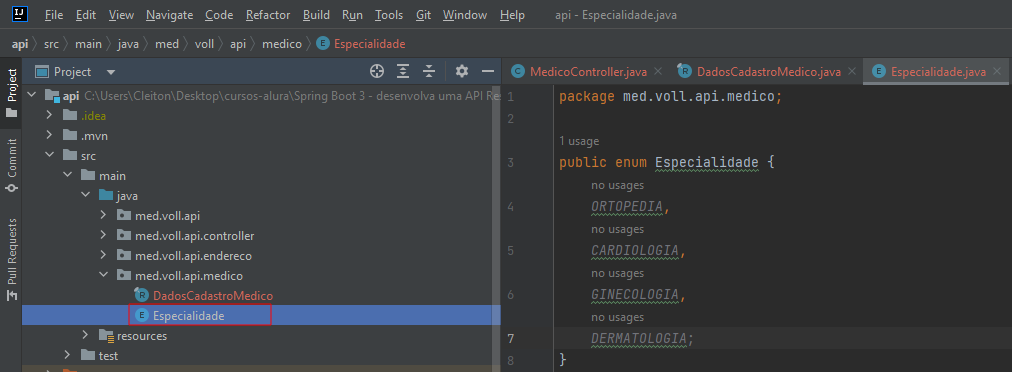
# Criação do cadastro de médicos

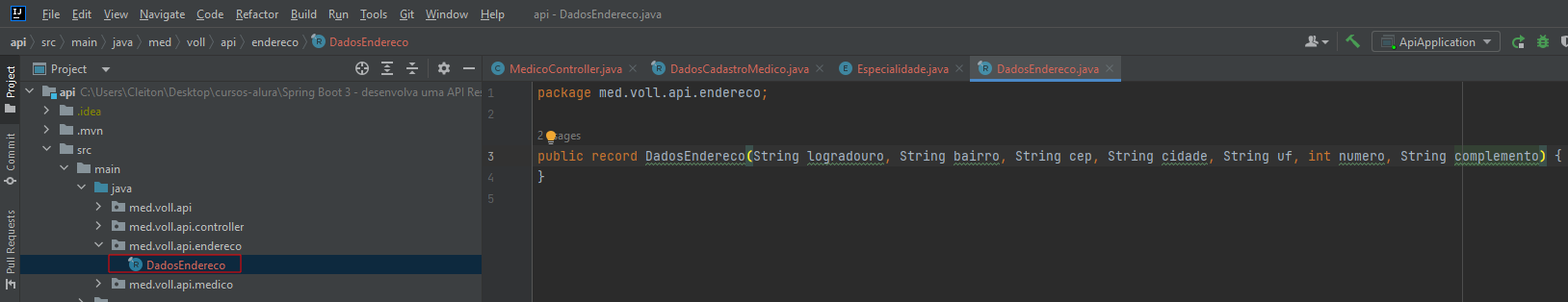
<https://trello.com/b/O0lGCsKb/api-voll-med>  
<https://www.figma.com/file/N4CgpJqsg7gjbKuDmra3EV/Voll.med>







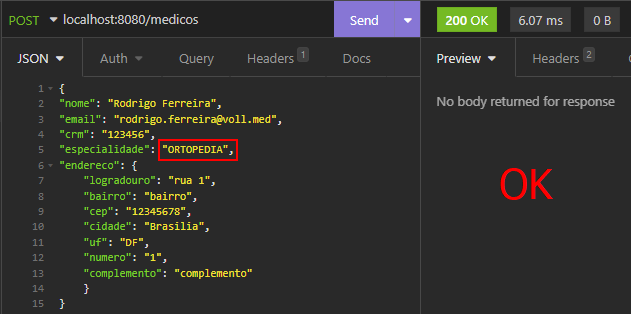


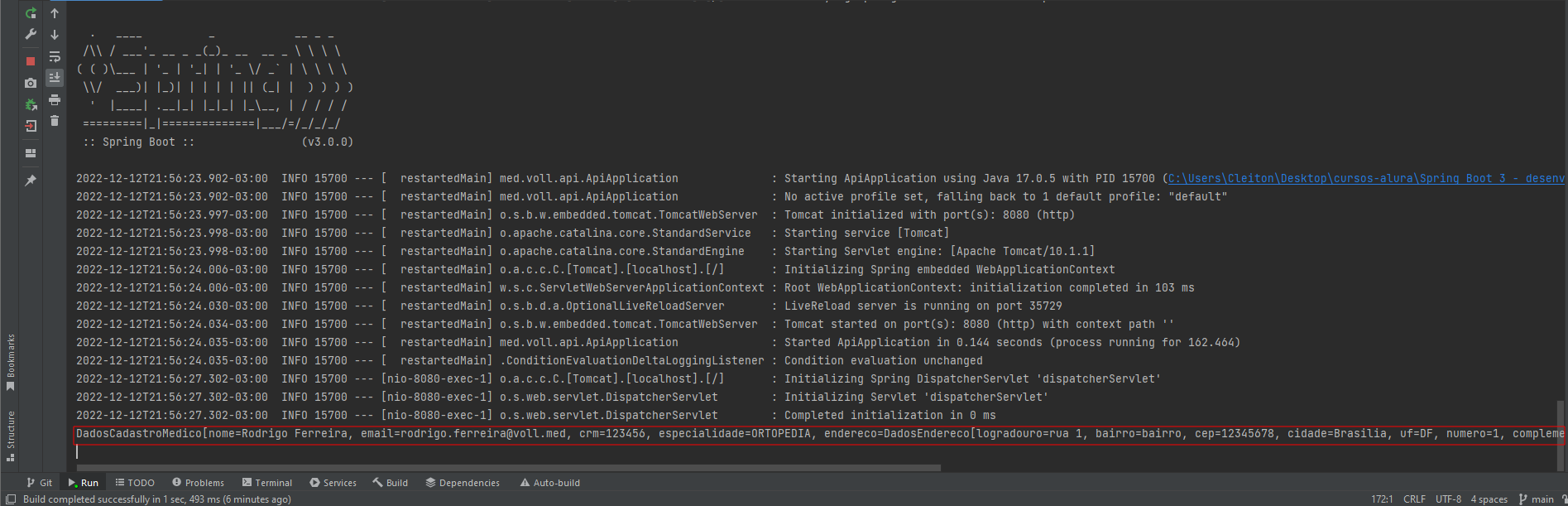


# Fazendo a requisição

Observe que deu um erro por conta do enum que está minúsculo. Temos que colocar exatamente igual ao codificado.







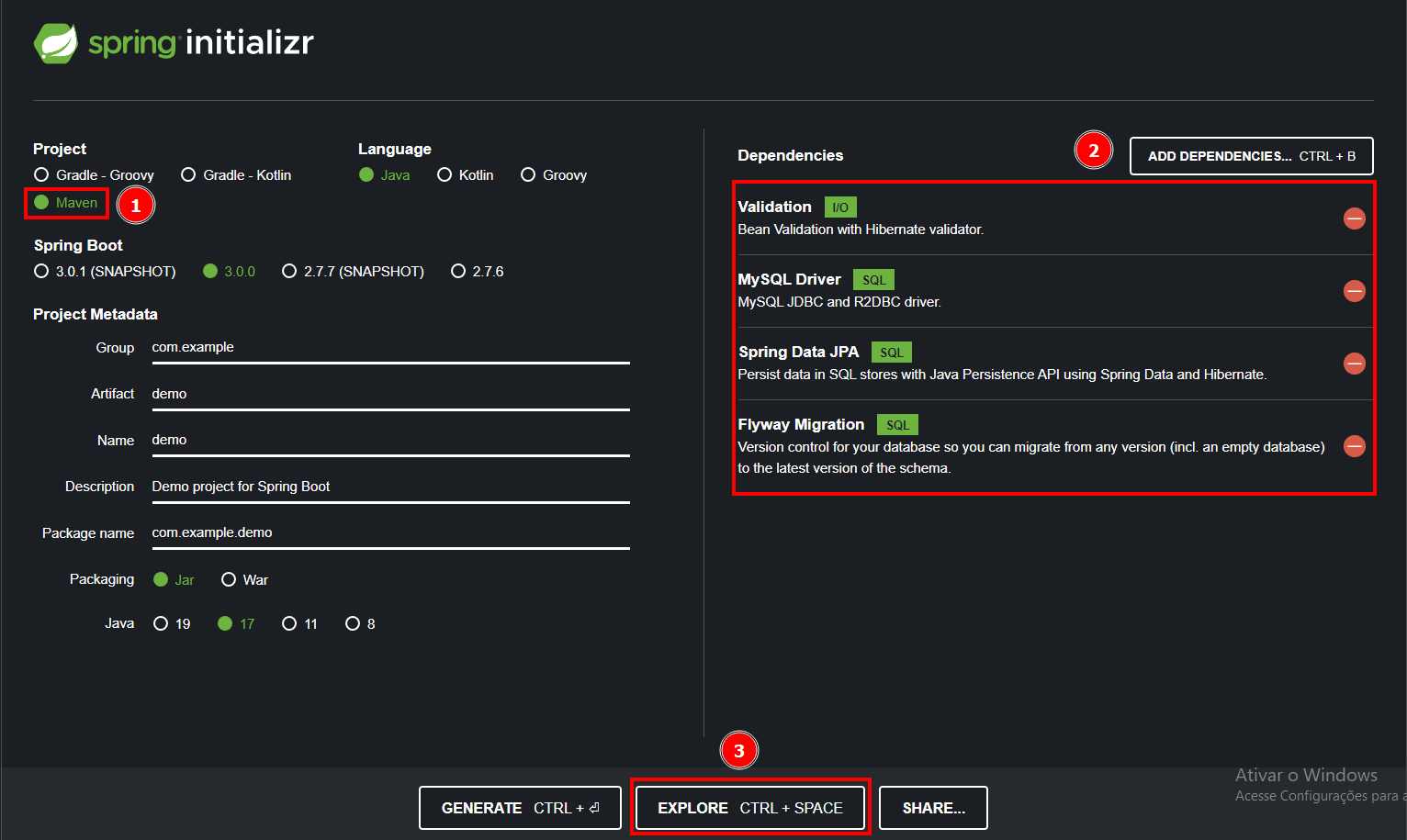
Obs.: Certifique que TODOS os atributos estão com os mesmos nomes do Json

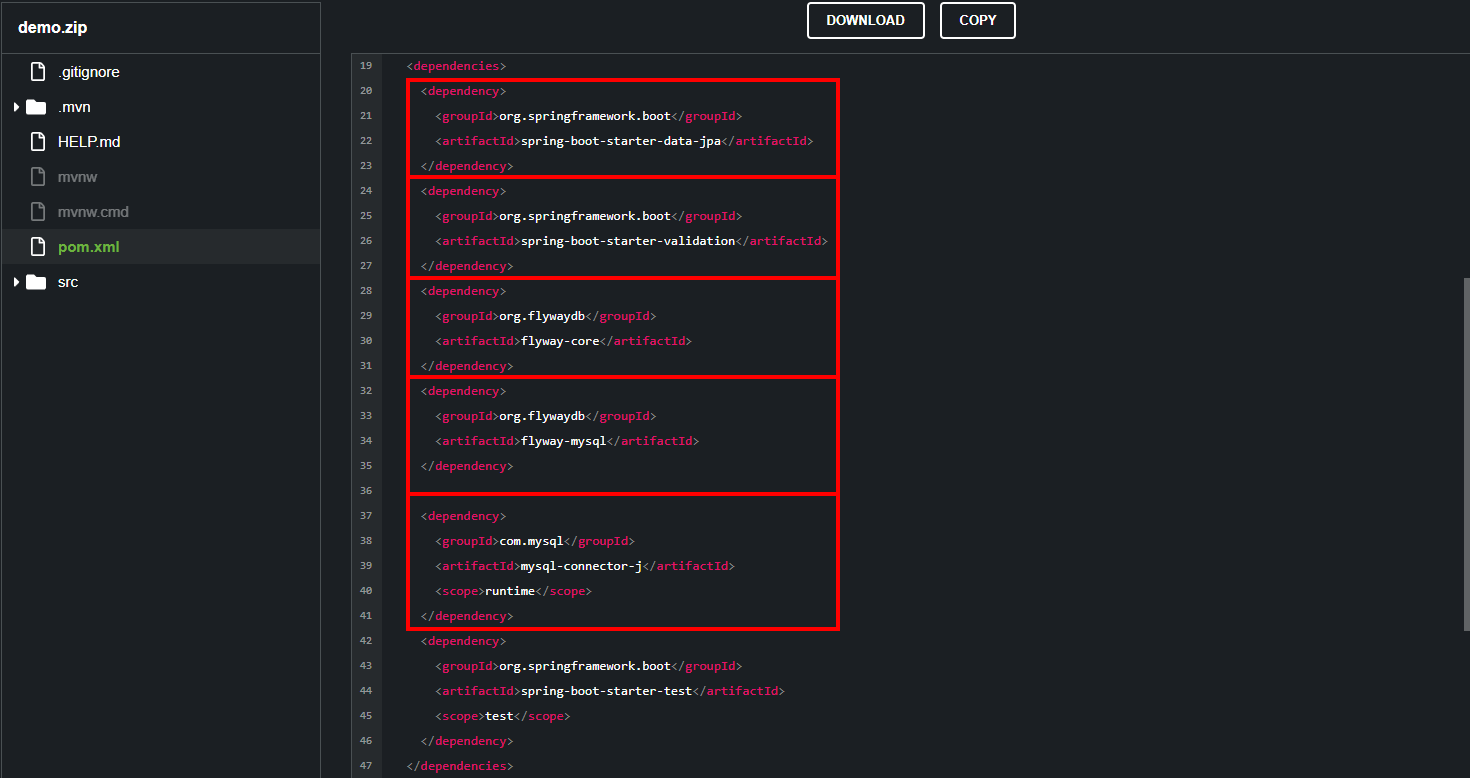
# 3 - Spring Data JPA

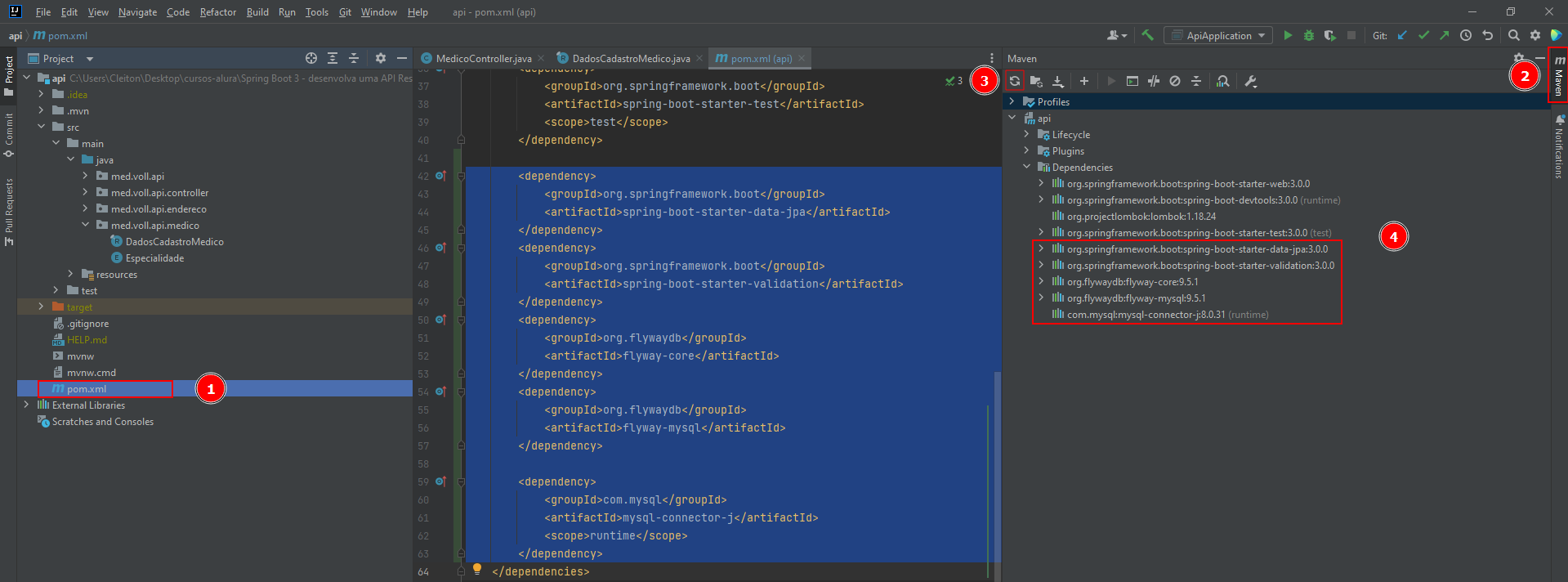
# Instalação do MySQL

<https://www.alura.com.br/artigos/mysql-instalacao-configuracao>

# Adicionando dependências

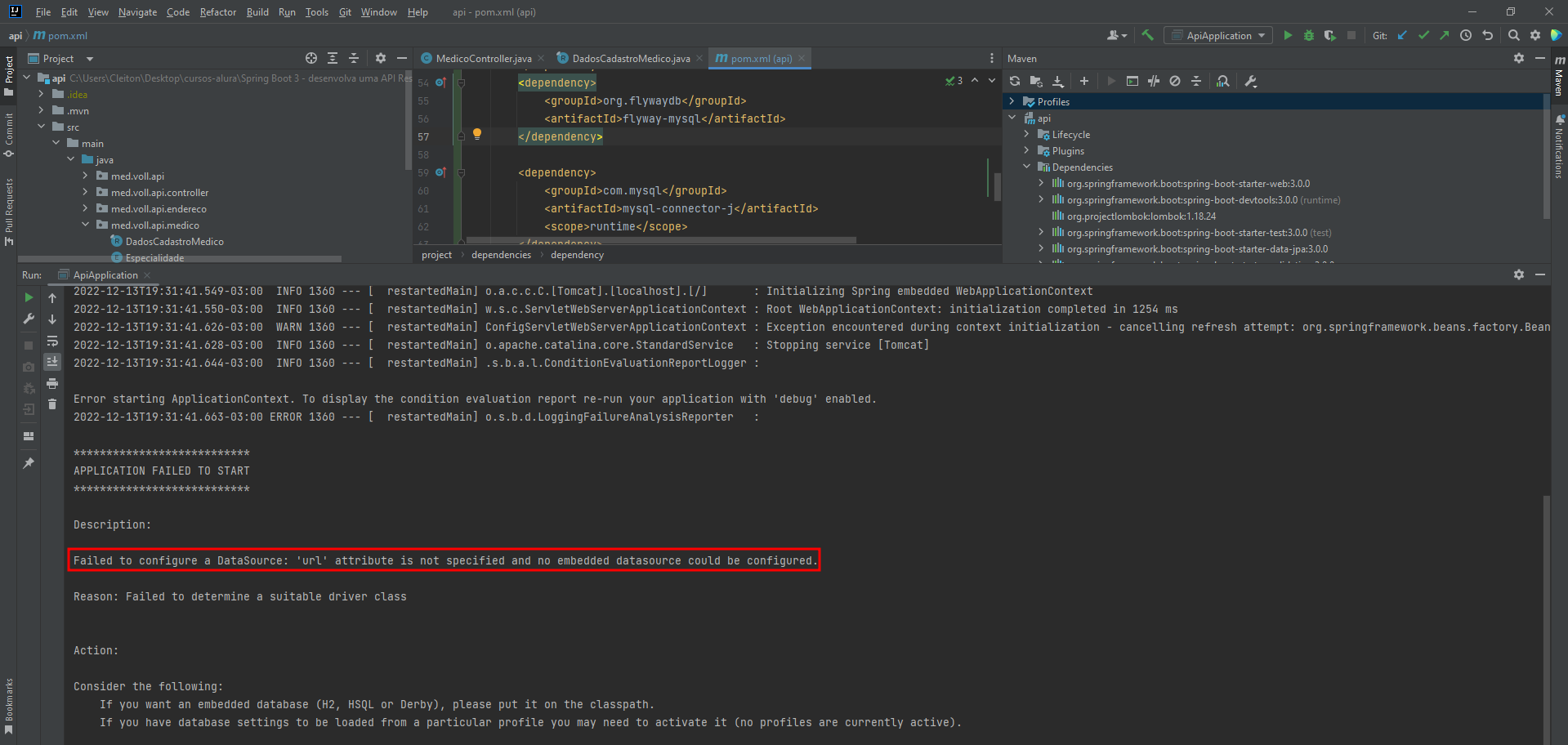






# Executando a aplicação sem configurações

Quando adicionamos as dependências do JPA, o Spring tenta pegar as configurações para fazer a conexão com o banco de dados, mas como não adicionamos, o seguinte erro é exibido:

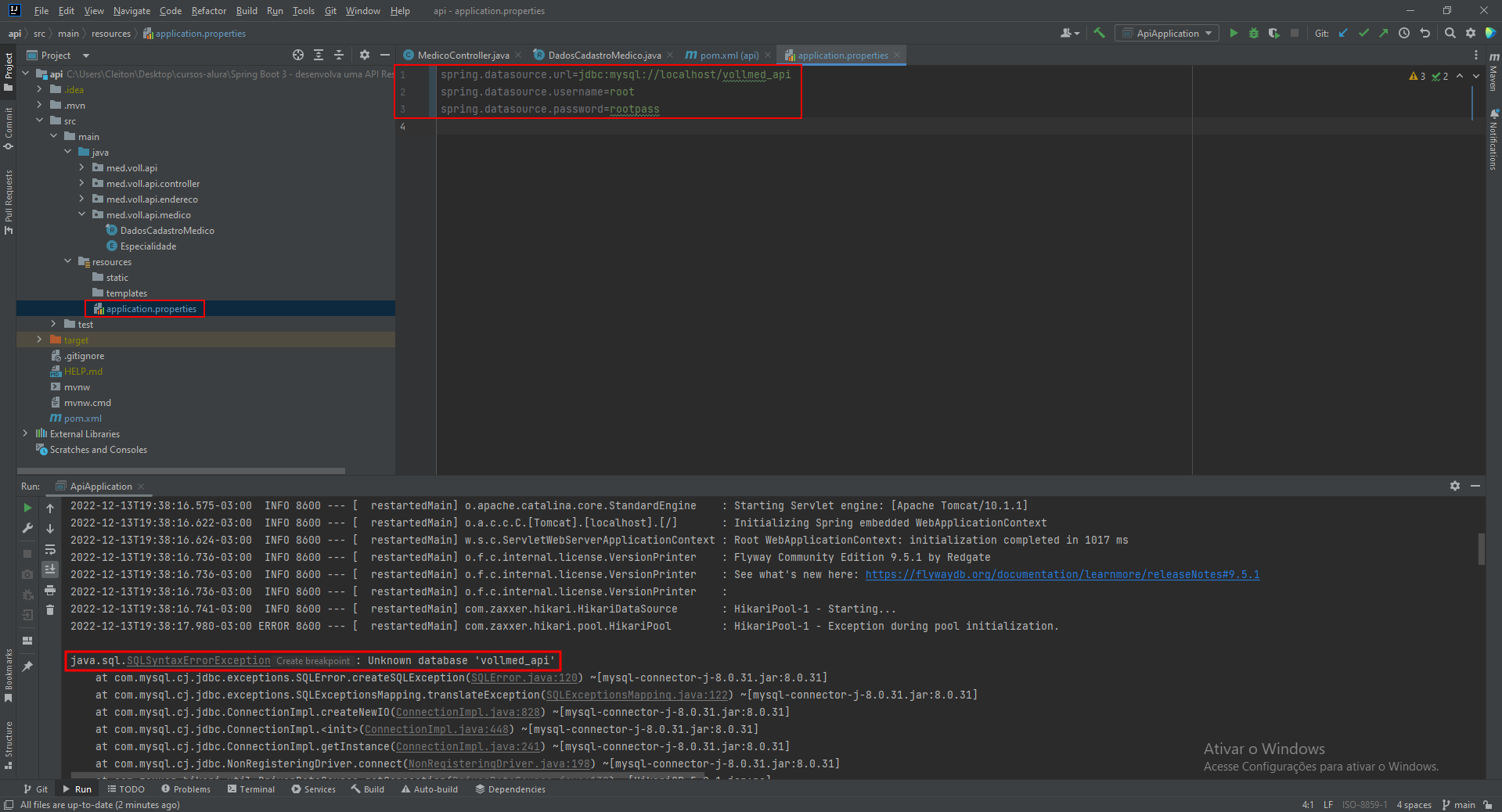


# Definindo configurações com o banco de dados

Para passar a URL de conexão com o banco de dados, vamos inserir o código spring.datasource.url=jdbc:mysql://localhost/vollmed\_api. Passaremos também spring.datasource.username=root, para definir o username, e spring.datasource.password=root, para definir a senha.

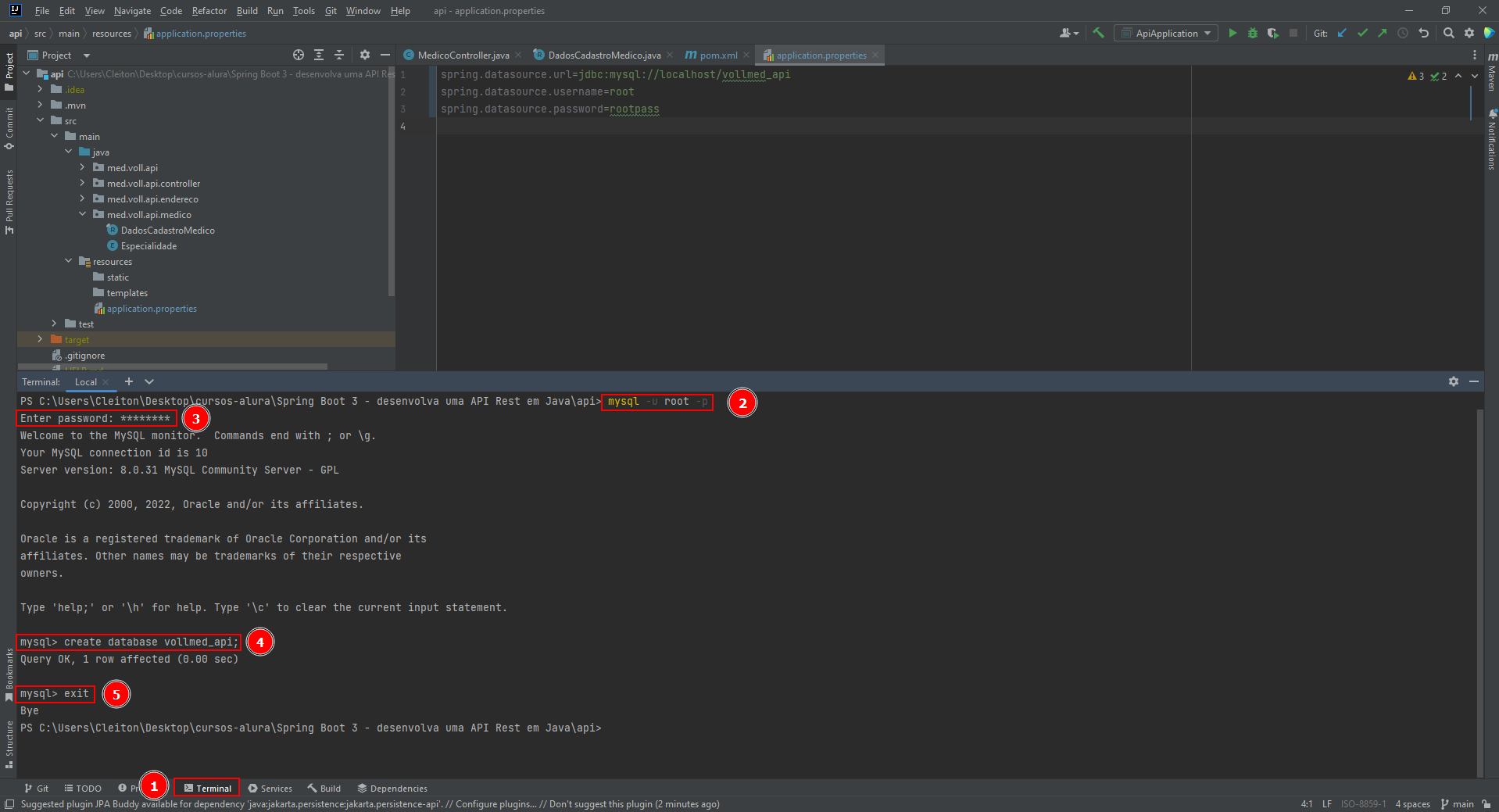
Ao executar aparece o seguinte erro: nknown database 'vollmed\_api'

Faz sentido, já que não criamos o nosso banco de dados.

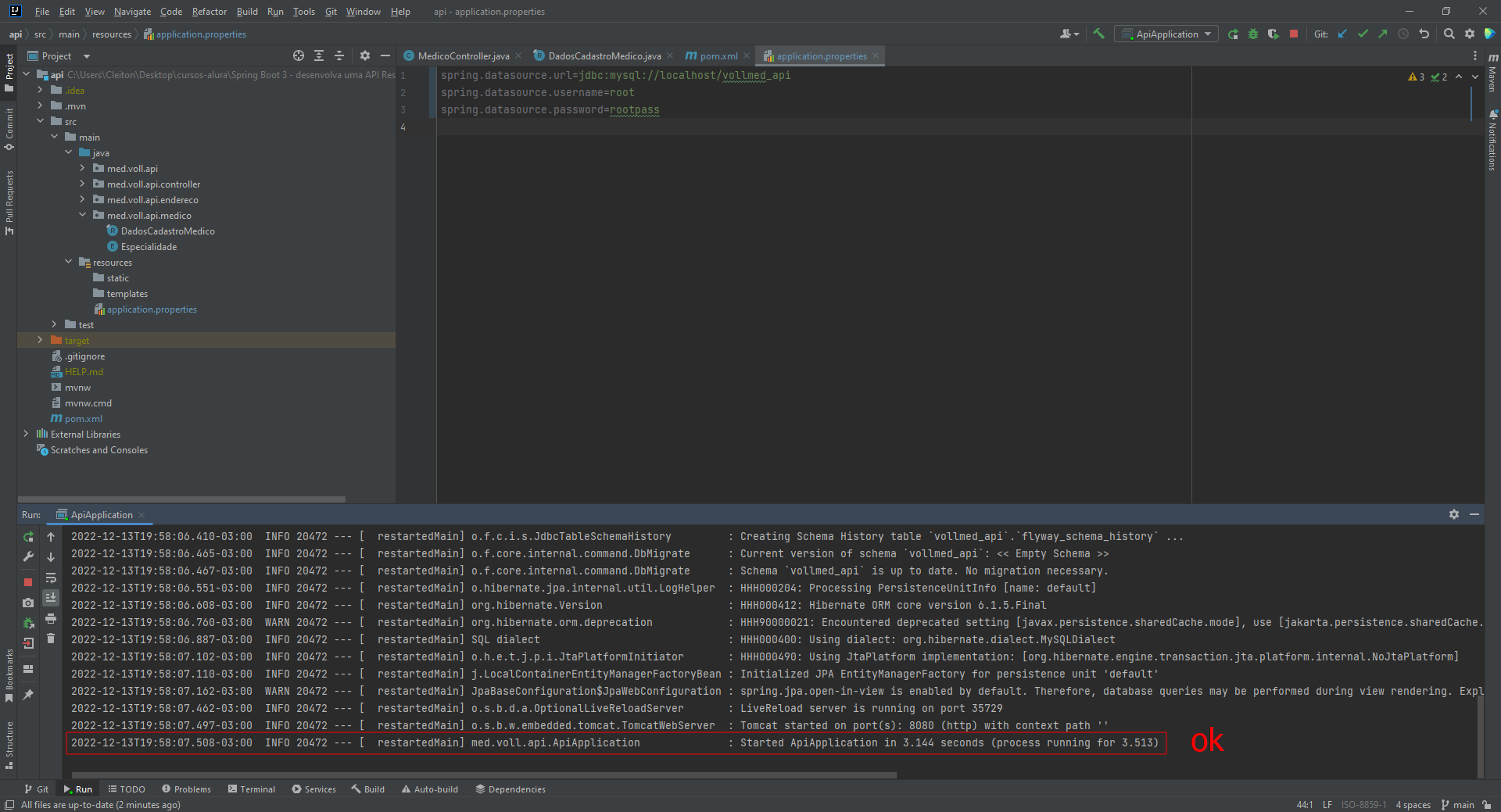


# Criando banco de dados

Caso o comando não seja reconhecido, adicionar nas variáveis do sistema!



# Executando a aplicação com configurações



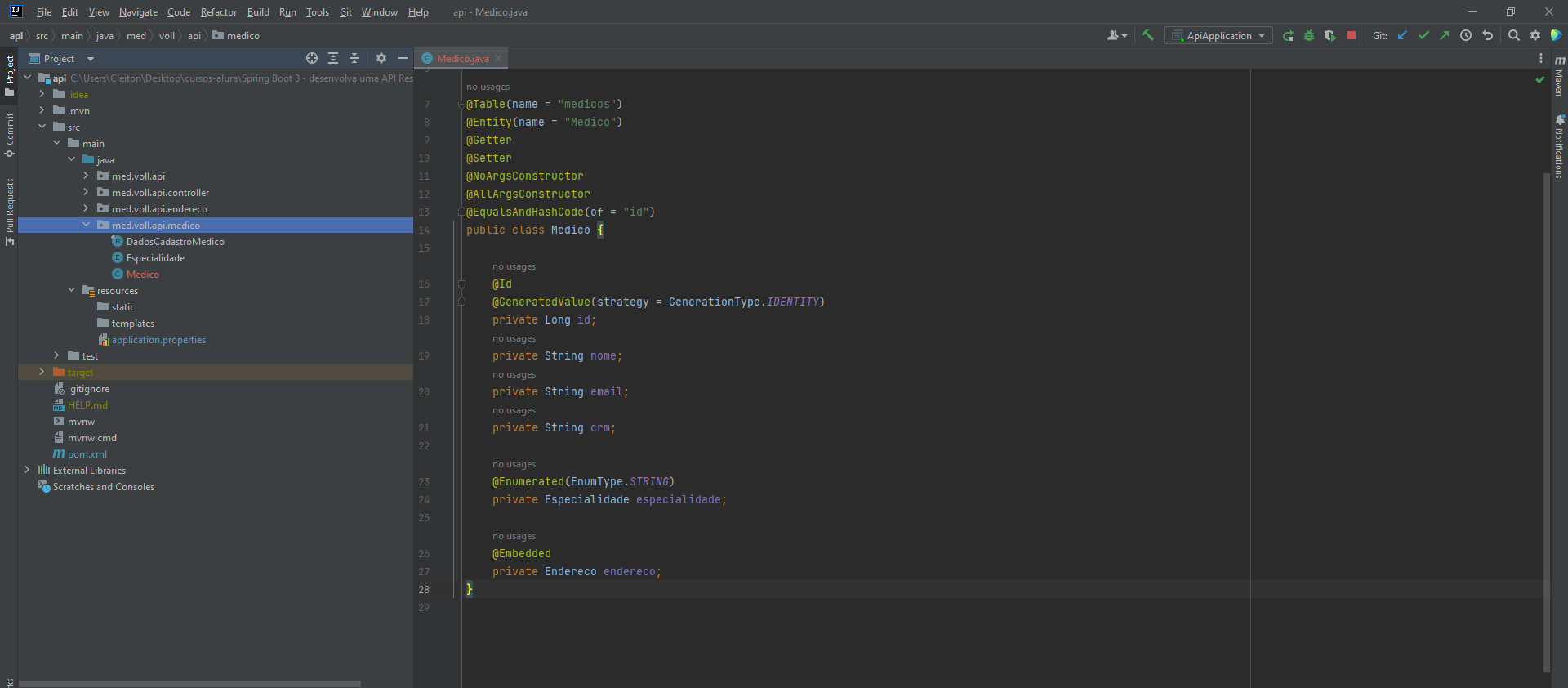
# Criação de entidades JPA

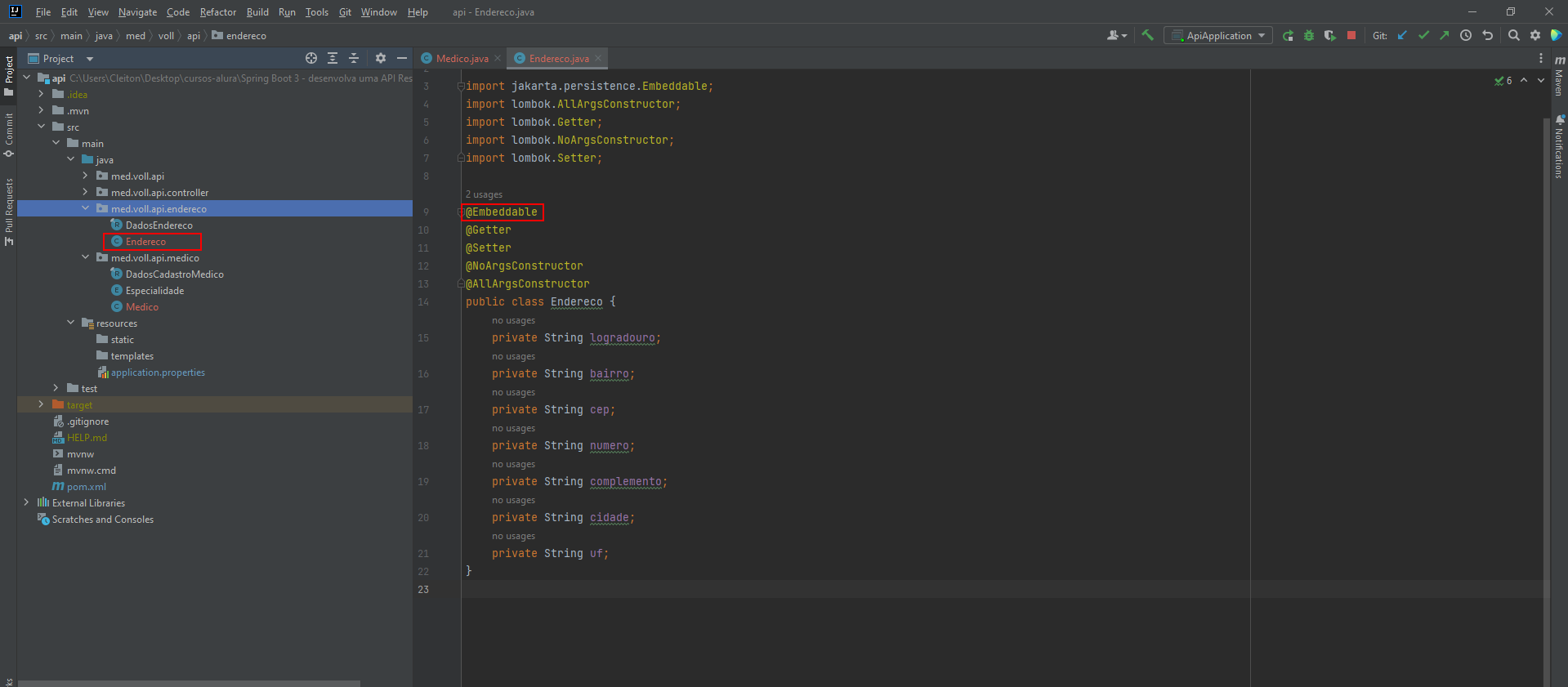
Criamos as nossas Records para o DTO (Data Transfer Object), agora precisamos de nossas entidades JPA (Java Persistence API), ou seja, as classes que irão mapear uma entidade do banco de dados.

Para facilitar nossa vida, vamos usar o lombok par acriação de getters, setters, construtores, equals e hash code.

Nota: @Embedded em Medico

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.

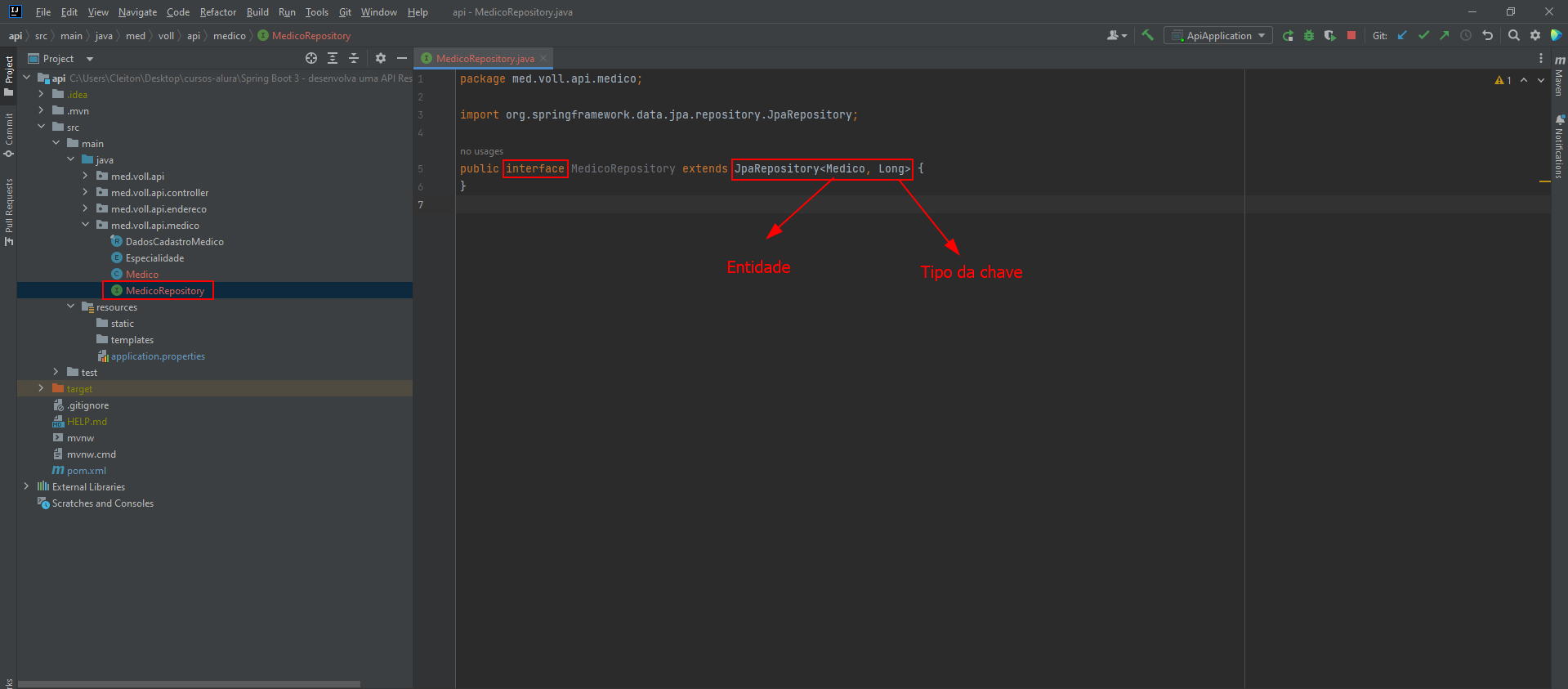




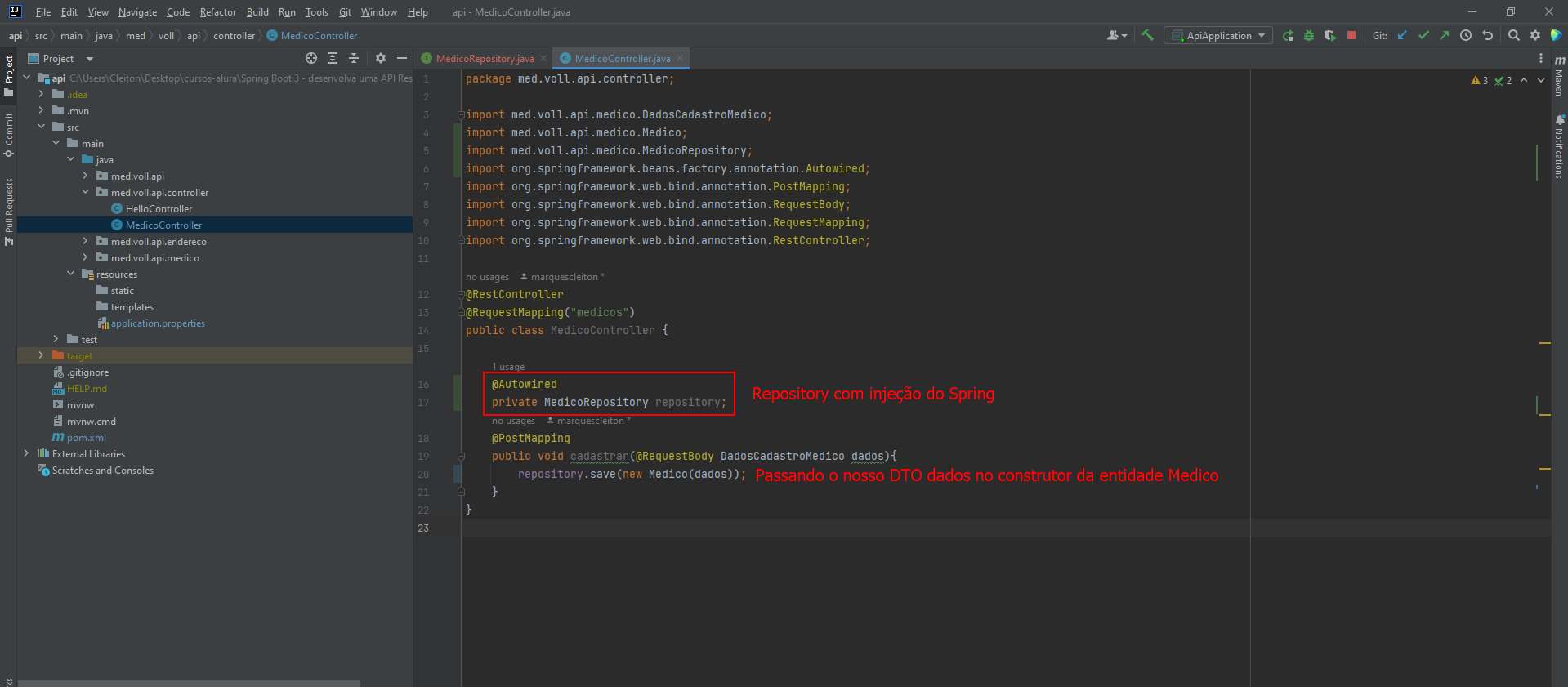
# Interfaces Repository

Para fazer a persistência, pegar o objeto Medico e salvar no banco de dados, o Spring Data tem o Repository, que são interfaces. O Spring já nos fornece a implementação.

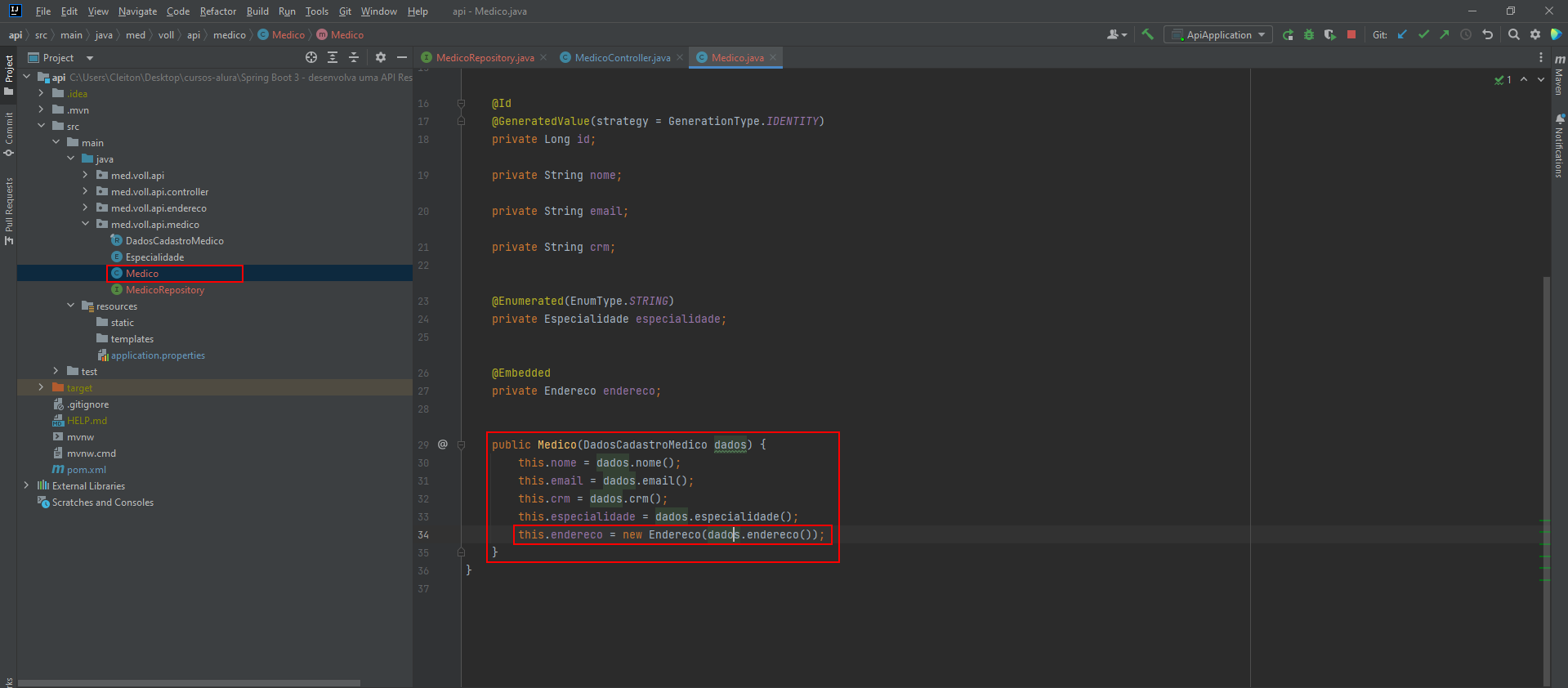
Repository:



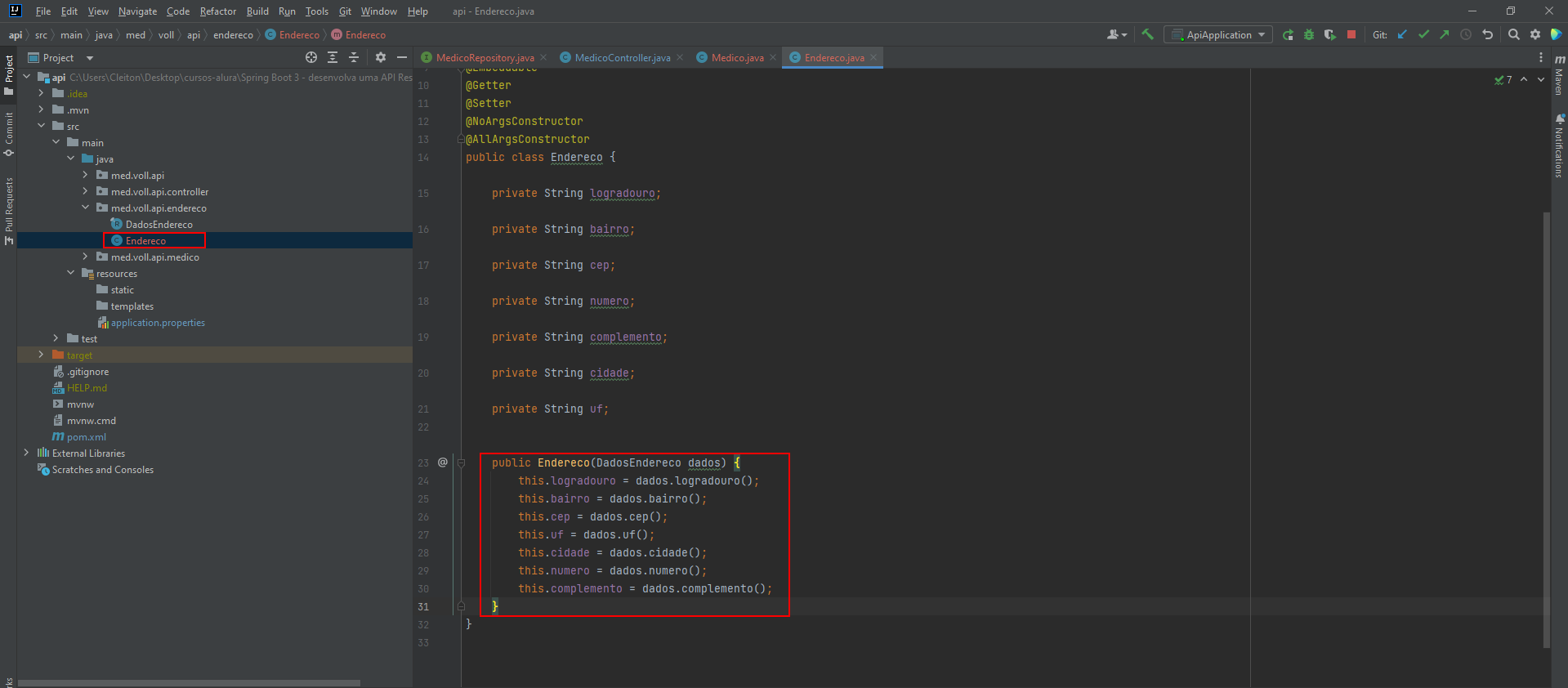
Controller



Médico:

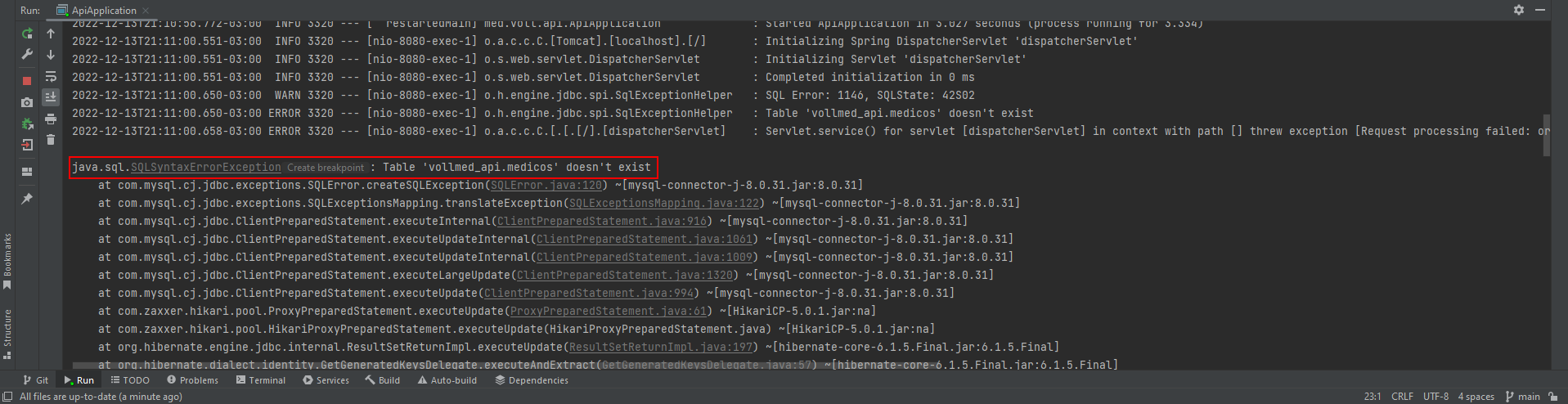


Endereço:

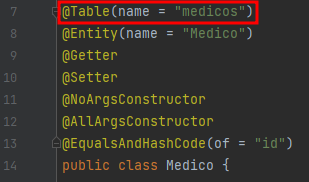


Executando o projeto a aplicação:





Um erro esperado, já que apenas criamos o banco de dados mas não a tabela médicos, como mapeado na entidade Medico



# Migrations com Flyway

Usaremos migrations, ou ferramentas de migrações, para **registrar as atualizações no banco de dados**. Já registramos o Flyway nas dependências, uma dessas ferramentas suportadas pelo Spring Boot. Dessa forma iremos ter todo um histórico de banco de dados, assim como o histórico de nossos códigos no Git.

Para cada mudança que quisermos executar no banco de dados, precisamos criar um arquivo .sql no projeto e, nele, escrever o trecho do comando SQL que será executado no banco de dados.

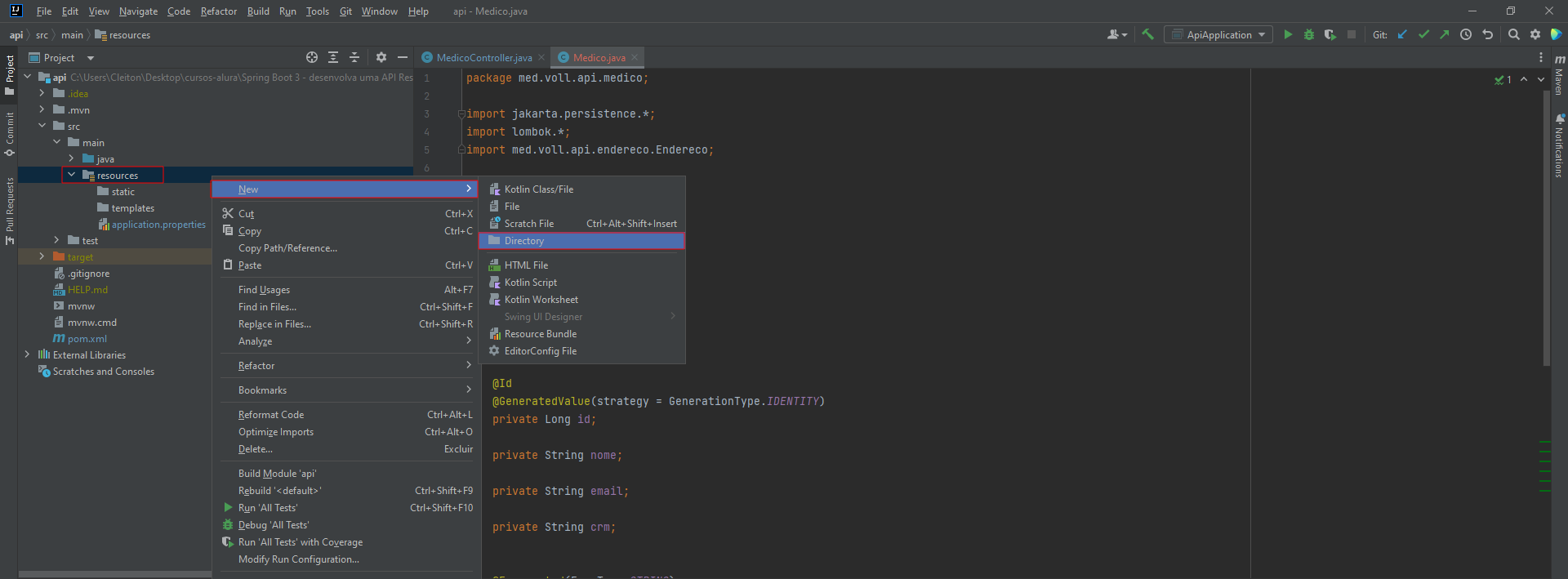
Precisamos salvá-los em um diretório específico. Criaremos essa nova pasta em "main > resources". Com "Alt + Insert", vamos escolher a opção "Directory" e digita o nome da pasta: "db/migration".

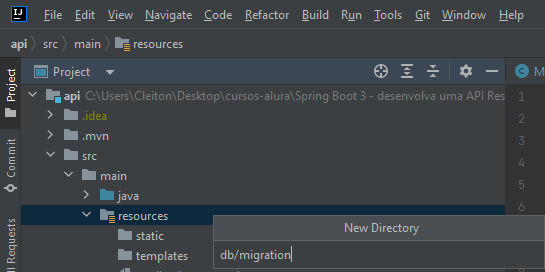
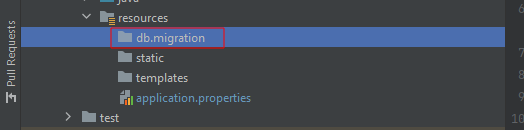
Obs: A barra significa que será criada uma subpasta, "migration", dentro da pasta "db".

Dentro da pasta, criaremos um arquivo SQL que servirá como nossa primeira migration, responsável por criar a tabela de médicos. Antes disso, é preciso interromper o projeto antes de usar migrations.

Obs: Sempre interrompa o projeto ao usar migrations.

**Criação do diretório:**



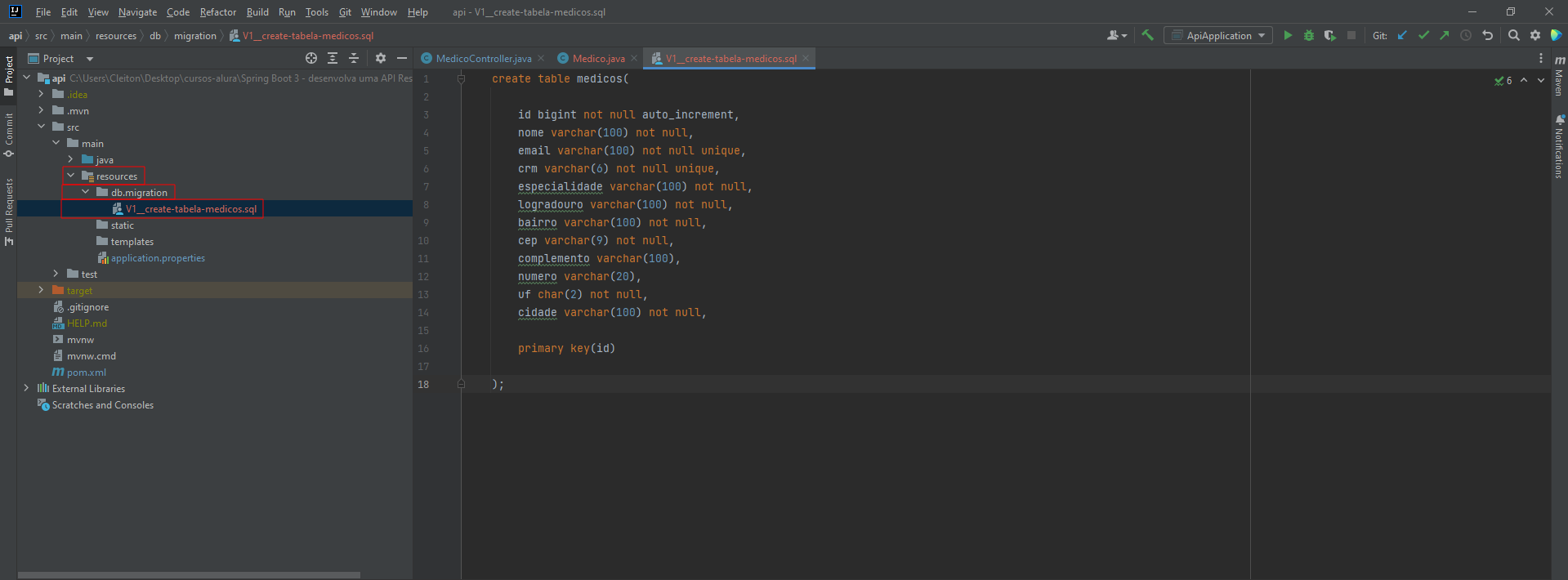
\*\*\* OBS: SEMPRE INTERROMPA O PROJETO AO USAR MIGRATIONS. SE AINDA NÃO PAROU, PARE AGORA, QUE PODE DAR RUIM ISSO AÍ!!!

**Criação do arquivo:**

A Migration tem um padrão: V{numero da versão}\_ \_nome-sugestivo.sql

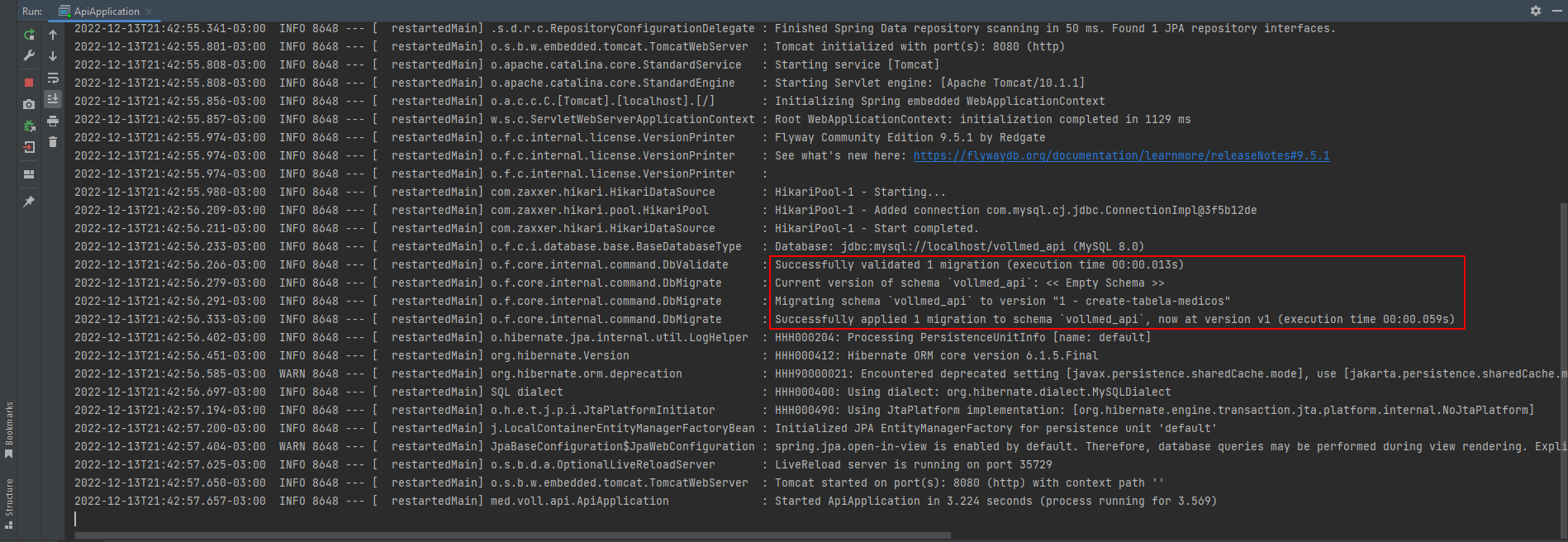
Obs.: a letra V da versão sempre maiúscula.

Ex.: V1\_\_create-tabela-medicos.sql

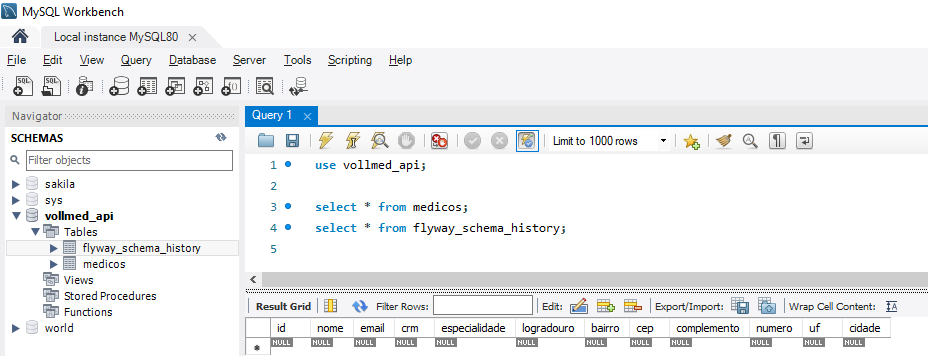


**Executando o projeto com a Migration configurada:**

Encontrou uma Migration, validou a versão e estava vazia, executou o script e logou que executou com sucesso

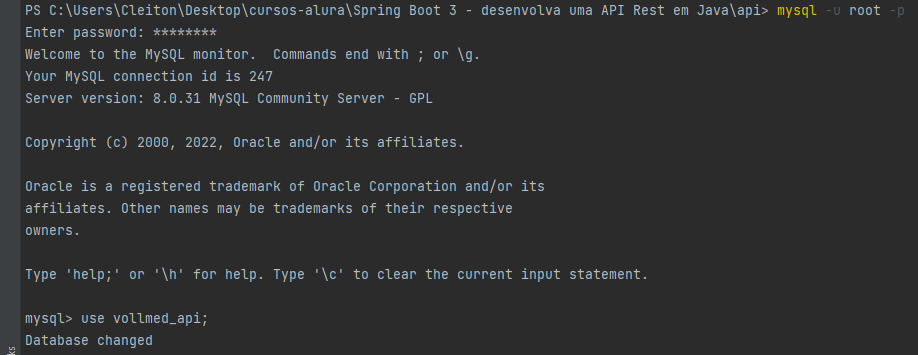


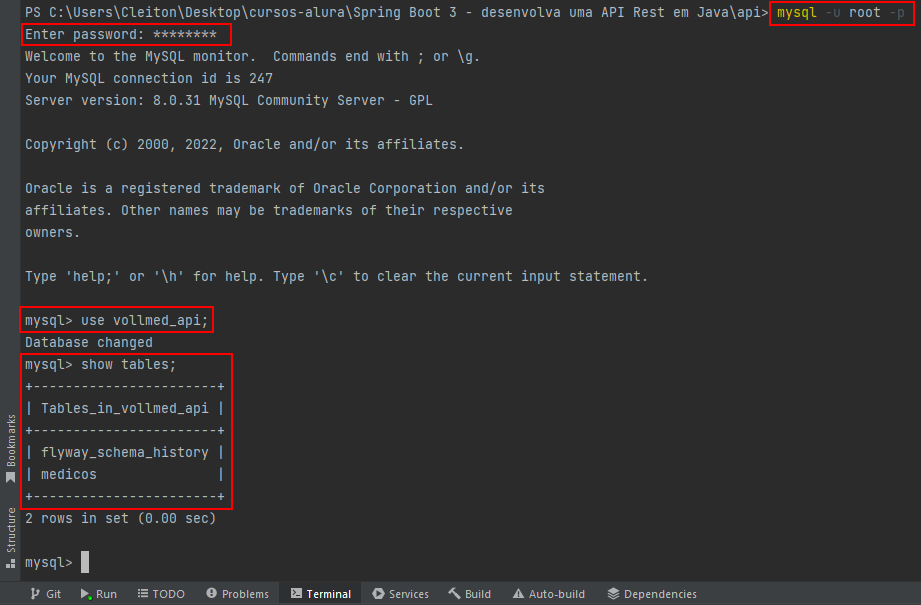
**Validando no Banco de dados:**



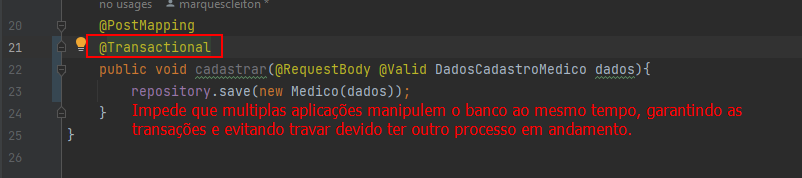


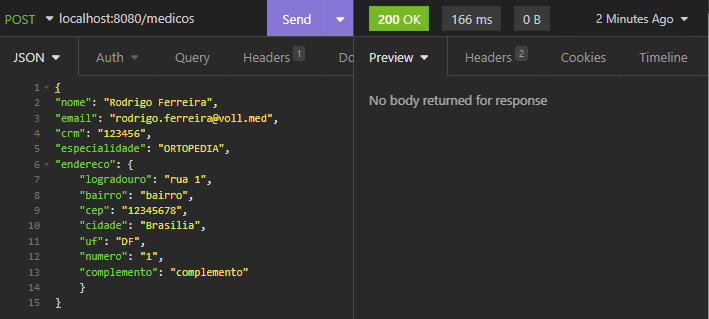
**Pelo terminal:**

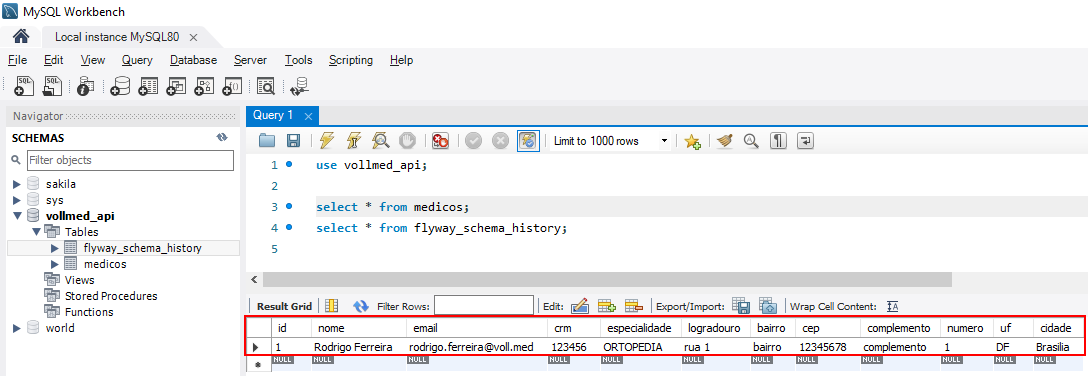




# Executando a aplicação e fazendo a requisição:





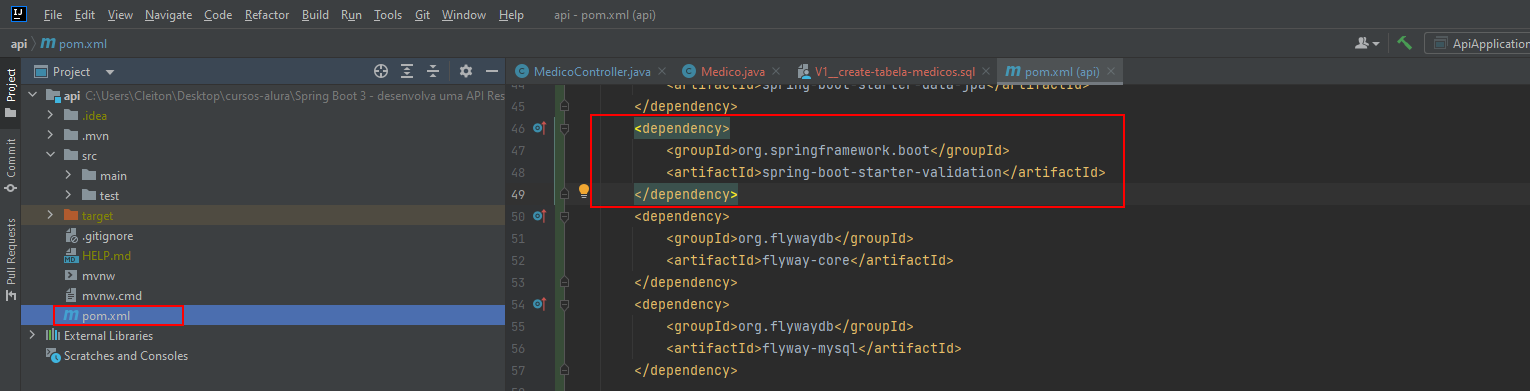


# Bean Validation

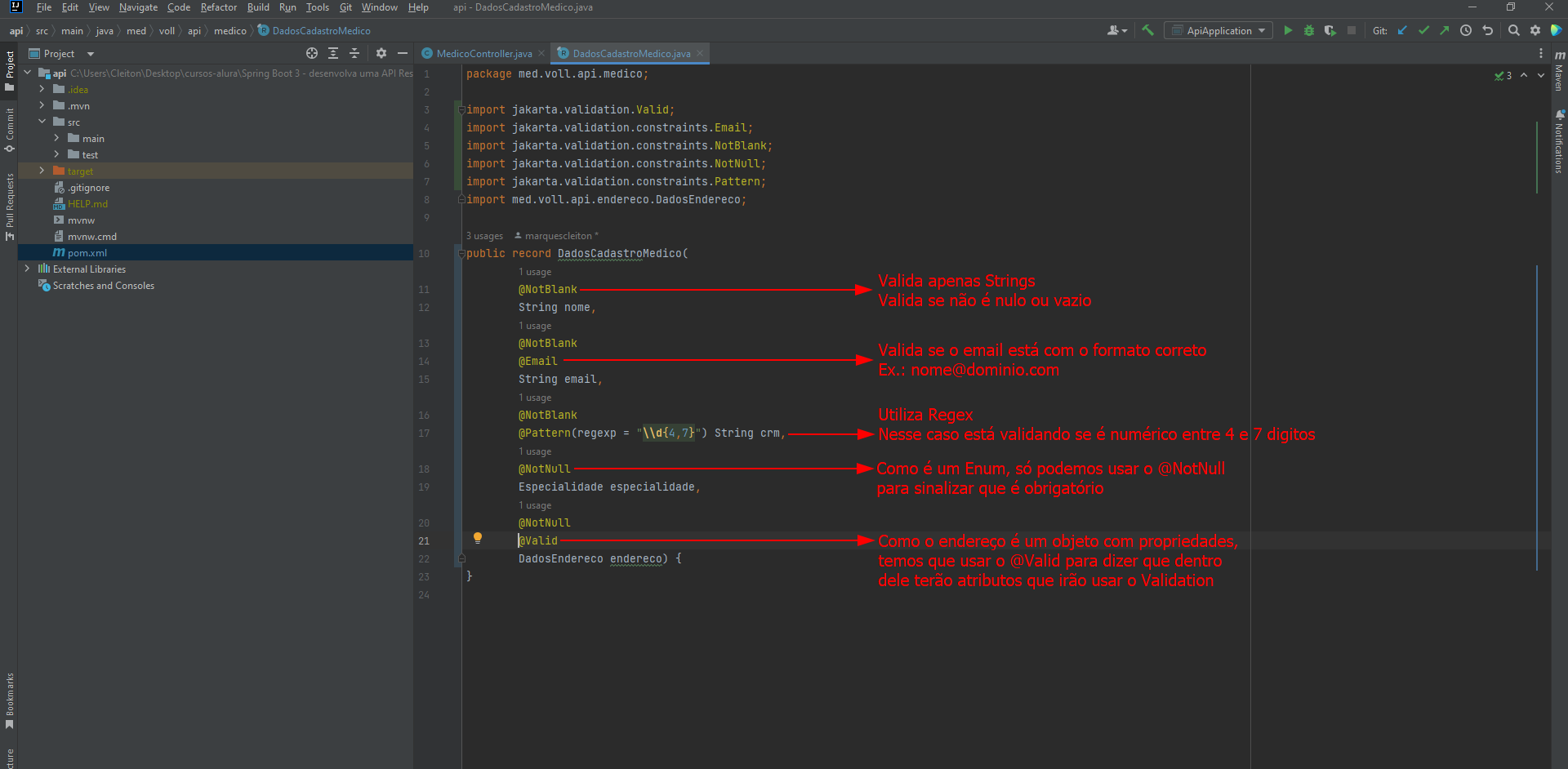
As validações nos ajuda a colocar restrições em nossos DTOs. Dessa forma podemos validar se a informações como:

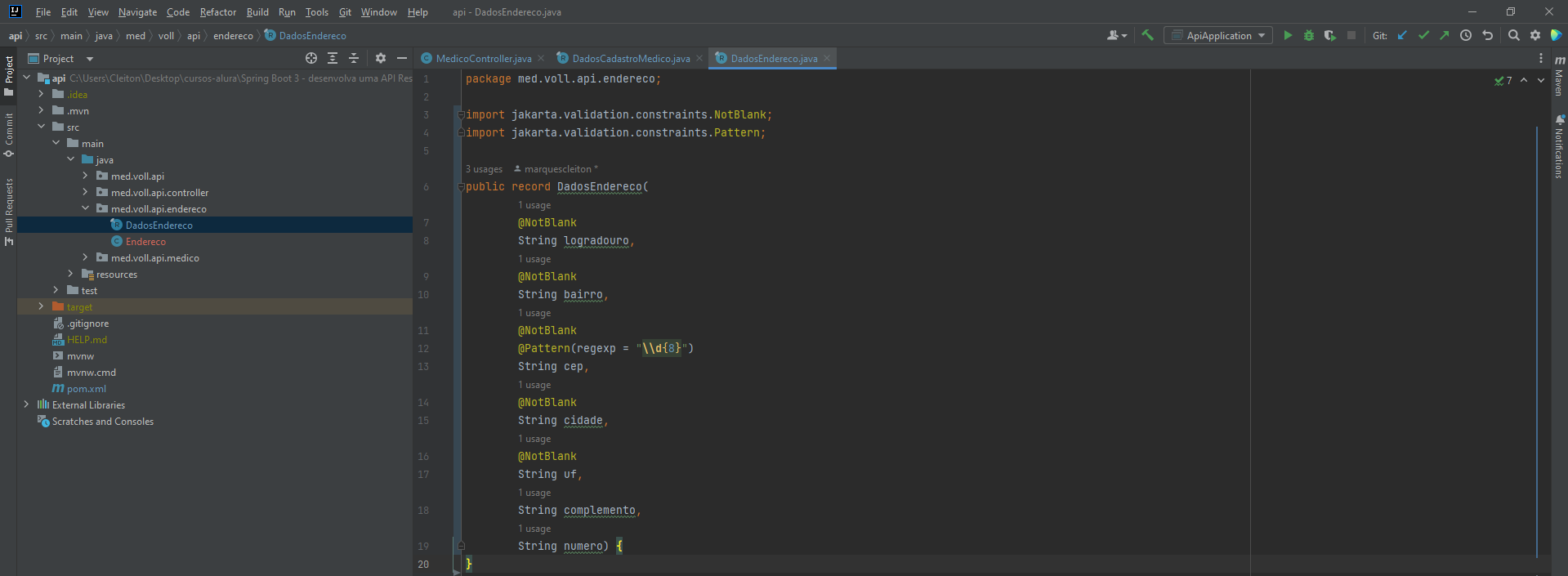
* Campo não pode ser nulo;
* Apenas valor numérico;
* Apenas letras;
* Valor mínimo;
* Valor máximo;
* Patters para restringir formatos específicos: CPF, CNPJ, Email...

Dependência foi adicionado na construção do projeto



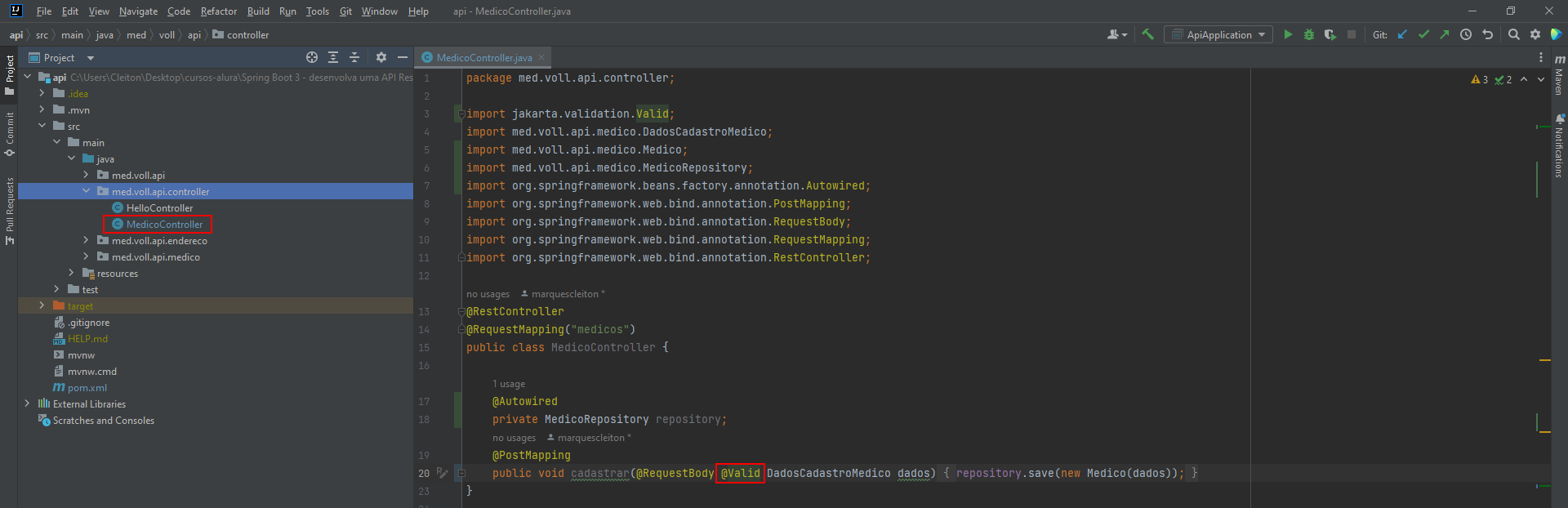
# Anotando o DTO com o Validation:



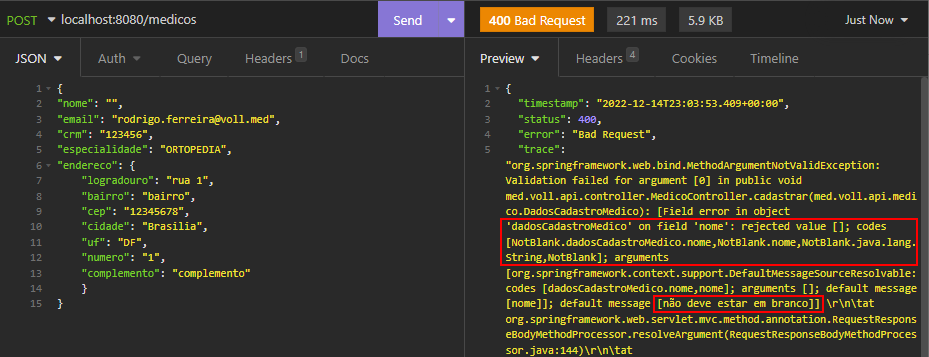


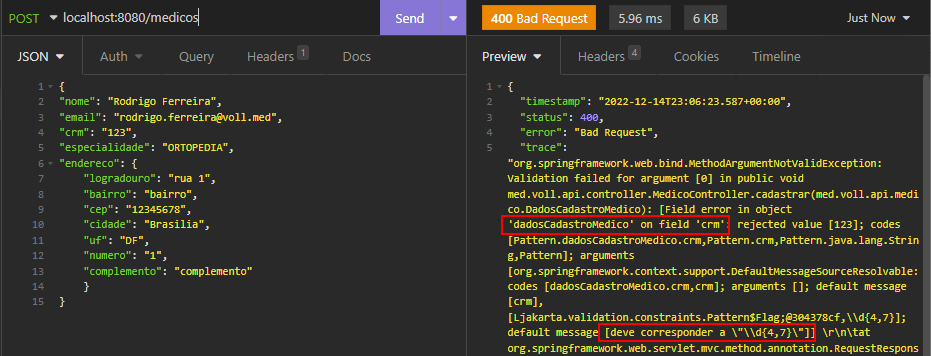
# Anotando controller

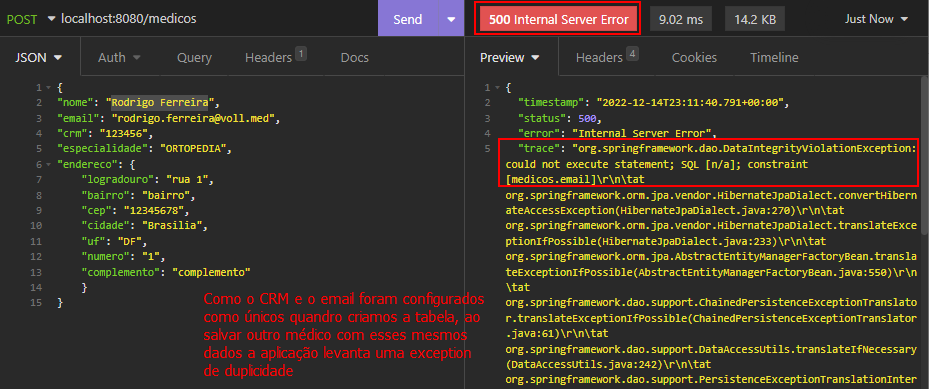
É importante adicionaremos @Valid ao nosso Controller, para solicitar que o Spring se integre ao Bean Validation e execute as validações.

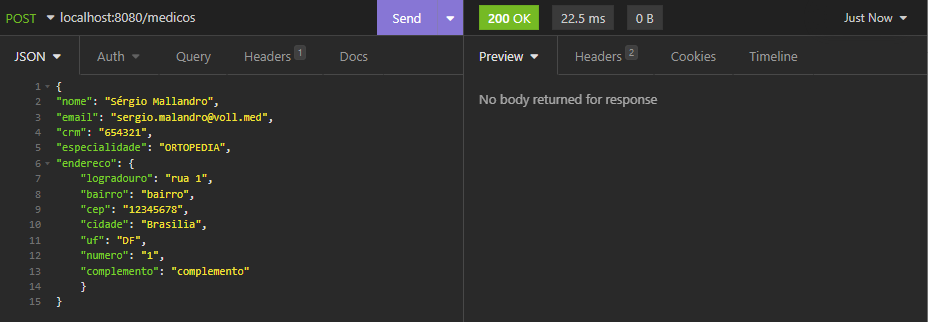


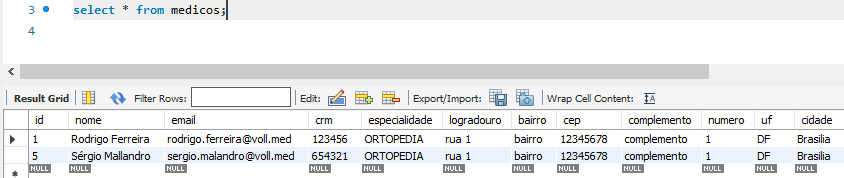
# Testando as validações









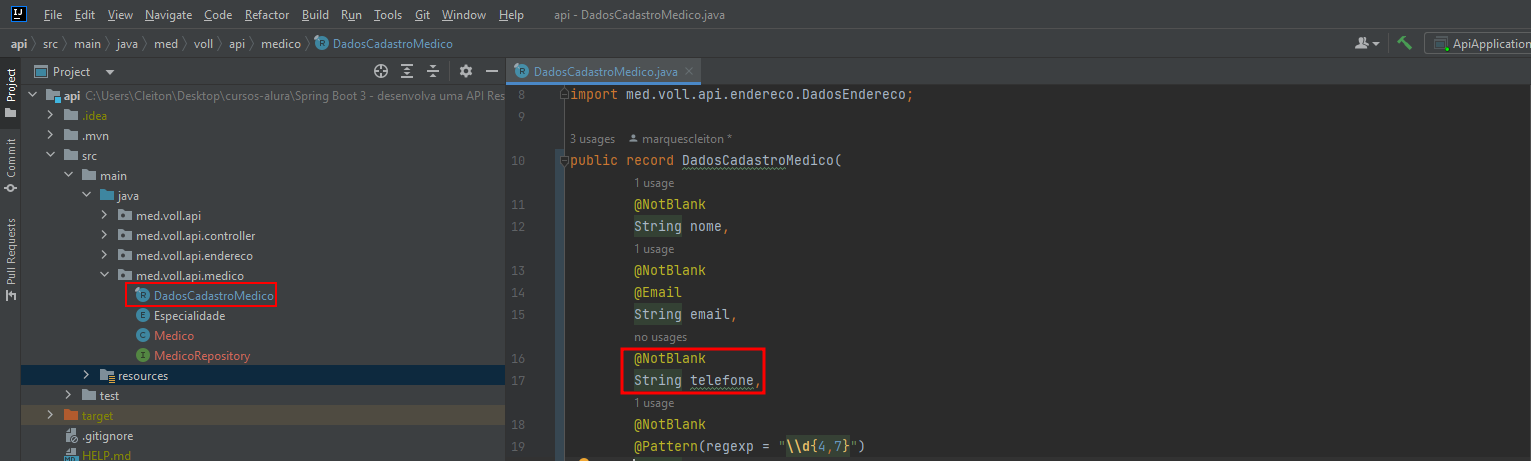


# Nova migration: Alteração propriedades do projeto

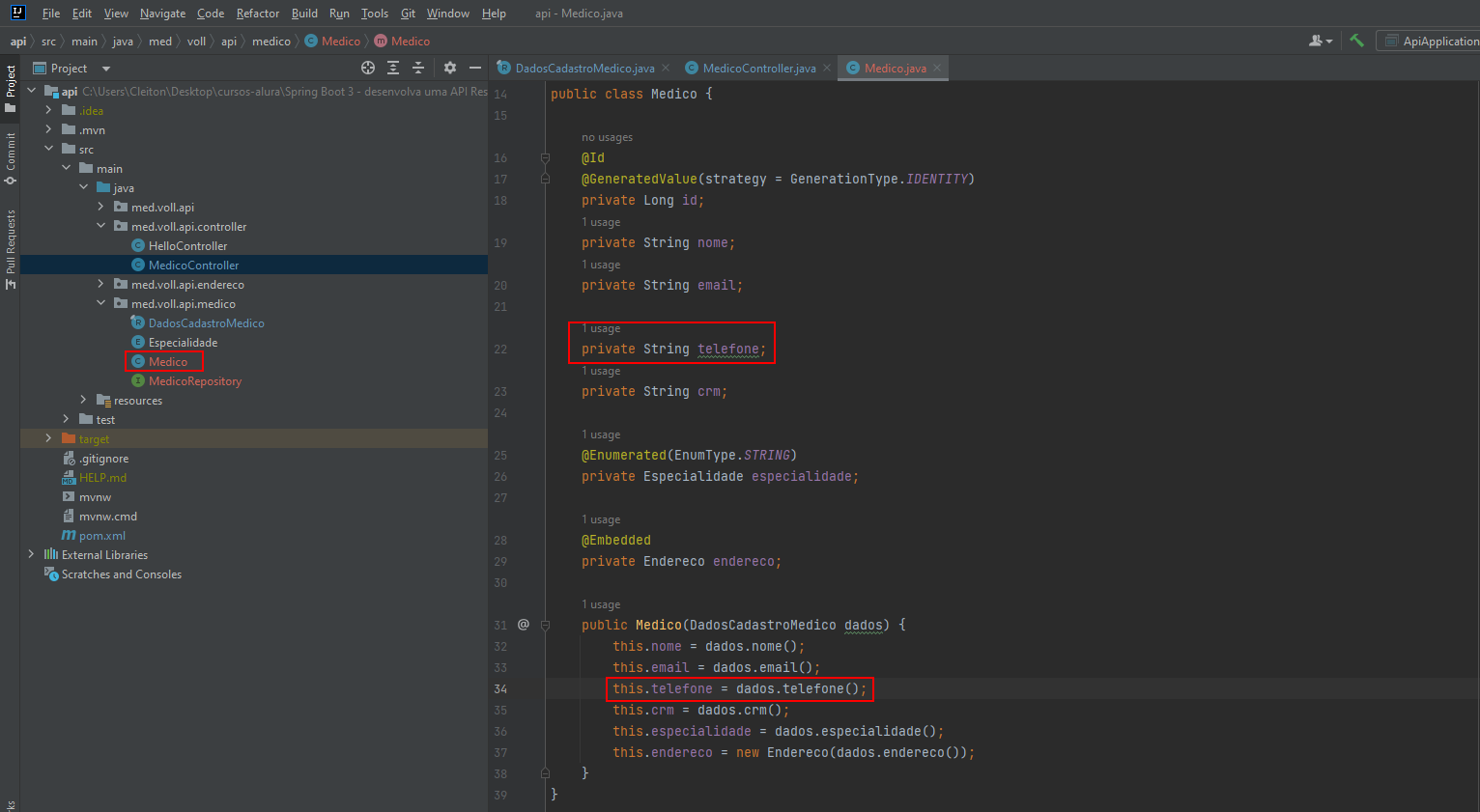
Ao criar a entidade e a tabela no banco, acabamos não colocando o telefone



**Alterando DTO**

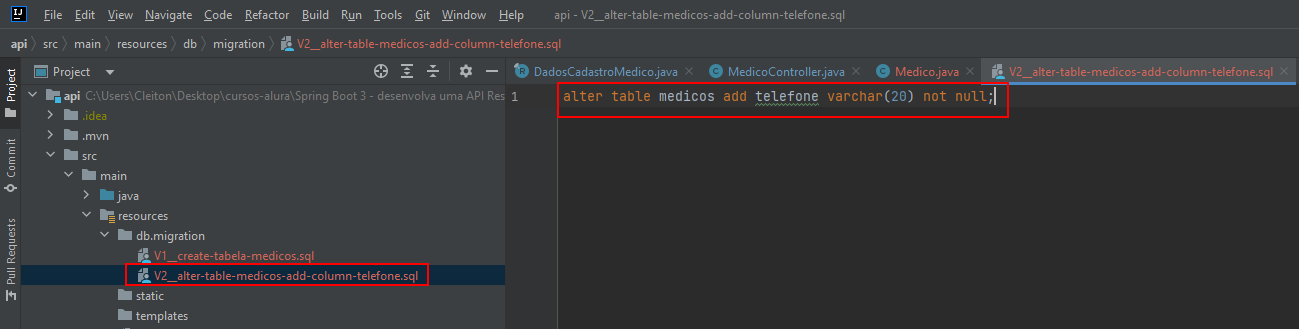


**Alterando JPA**



**Criar nova Migration**

Obs: Sempre interrompa o projeto ao usar migrations.



Executando o projeto...

