Olá pessoal, tudo bem com vocês?  
O desafio de hoje foi proposto pela ZUP, através do programa Orange Talents, e consiste em criar uma API REST para controle de aplicação de vacinas na população brasileira (nada mais atual, não é mesmo?).  
E aí, topa vir nessa comigo?  
Então, vamos começar!

Se você nunca construiu uma API REST, você deve estar se perguntando: “o que é isso??”

API é um acrônimo em inglês para “Interface de Programação de Aplicações”. Então, de forma geral, uma API é uma estrutura para fornecer dados a uma aplicação.  
“Mas e o REST?” REST é uma arquitetura de envio de dados, muito utilizada pelo protocolo HTTP, que é o protocolo que utilizamos nas nossas consultas à internet.  
Construir uma API REST será fundamental para fazermos os cadastros propostos em nosso **desafio**!  
  
Agora que já sabemos um pouco mais sobre API REST, vamos às linguagens e tecnologias que iremos utilizar:

* Linguagem de programação Java;
* Banco de dados MySQL, que guardará nossos cadastros;
* Framework Spring

Java e o framework (ou “ferramenta”) Spring. Além disso, para persistirmos (“salvar”) nossos cadastros, utilizaremos MySQL, que é um sistema de gerenciamento de banco de dados bem fácil de trabalharmos.

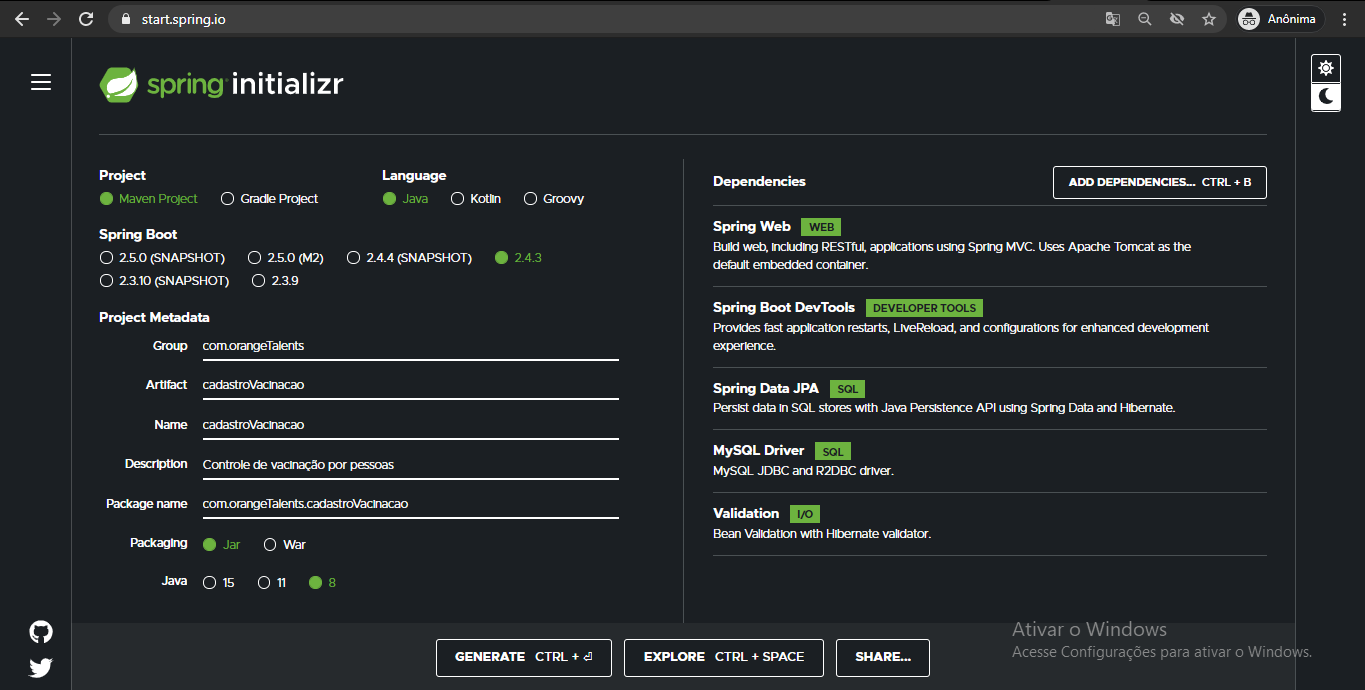
O Spring possui uma série de ~~Além disso, também utilizaremos algumas~~ tecnologias específicas ~~do mundo Spring~~ que iremos usar em nosso projeto:

- Spring web: permite-nos trabalhar com web services, que são os métodos usados para trafegar os dados da nossa API através de requisições HTTP;

- Spring Boot DevTools: auxilia-nos em produtividade, reiniciando a aplicação após alterações no código;

- Spring Data JPA: como a API e o banco de dados trabalham com paradigmas diferentes, o JPA facilita nosso trabalho “traduzindo” a comunicação entre os dois, realizando para isso um mapeamento objeto-relacional da nossa aplicação para persistência dos dados.

- MySQL Driver: será a conexão com nosso banco de dados;

- Validation: validação dos dados da nossa aplicação.  
  
O próximo passo será a criação do nosso projeto (uhuuu!). Podemos criá-lo de diversas formas, mas o Spring nos dá uma maneira muito rápida e prática através do site [start.spring.io](http://www.start.spring.io), onde podemos configurar os dados iniciais, importar dependências e gerar nosso projeto com alguns poucos cliques. Dá uma olhada:

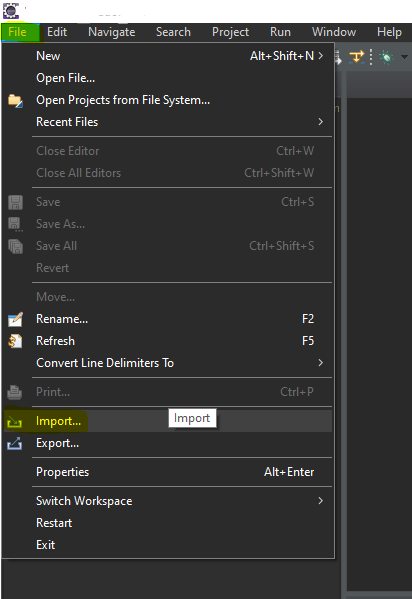
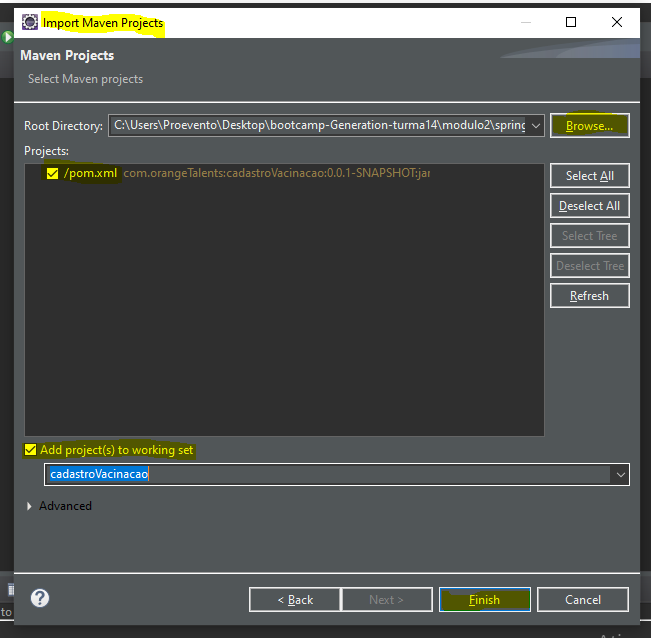
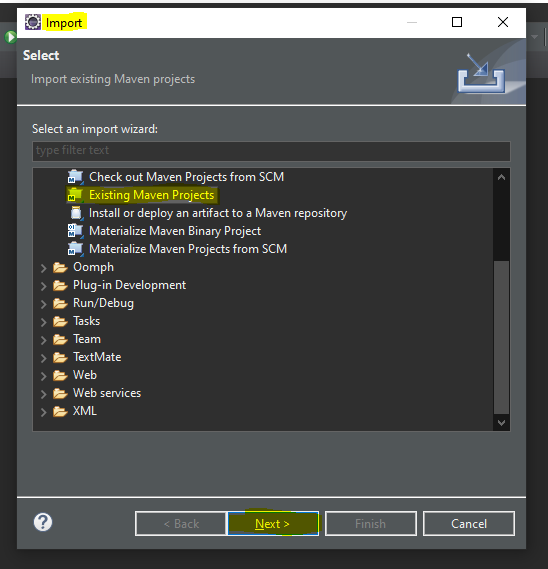
De acordo com a imagem acima, definimos algumas coisas importantes para nosso projeto:

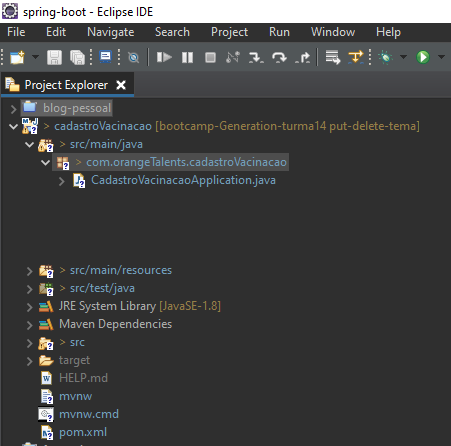
- nosso gerenciador de pacotes será o Maven;

- Nos campos de “Project Metadata” definimos os nomes dos nossos pacotes e demos uma breve descrição do projeto (fique à vontade para definir os seus 😉);

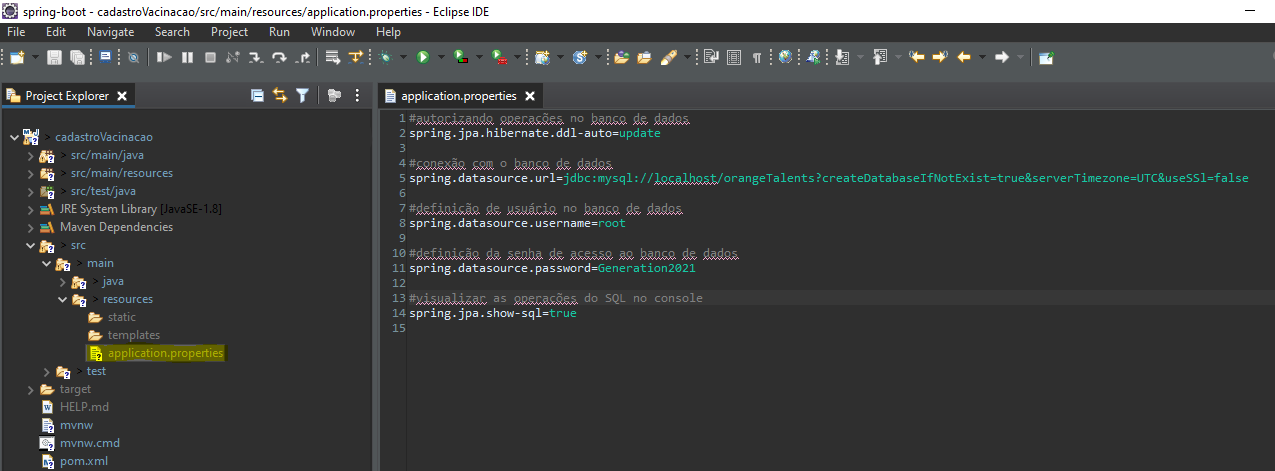
- Definimos como dependências as tecnologias já abordadas acima.

Após isso, ao clicar no botão **Generate**, o Spring Initialzr gera um arquivo **.ZIP** com nosso projeto, que pode ser importado para a IDE de sua preferência.  
Aqui, nossa IDE será o Eclipse, e podemos importar nosso projeto da seguinte forma: **File -> Import -> Existing Maven Projects** e selecionamos a pasta em que o arquivo foi extraído

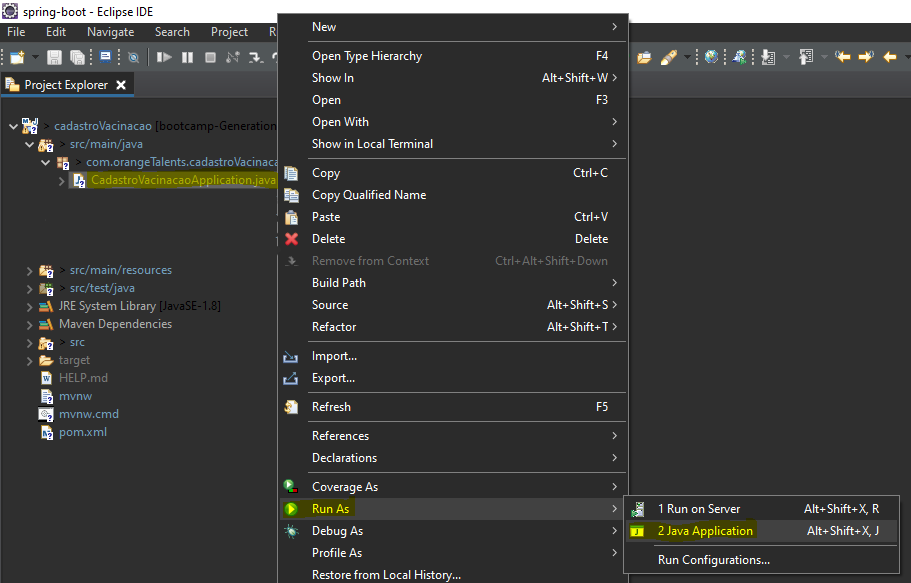
 

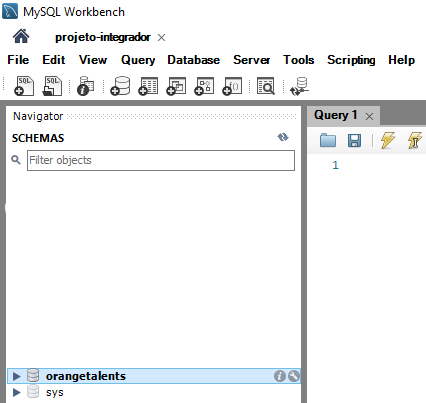
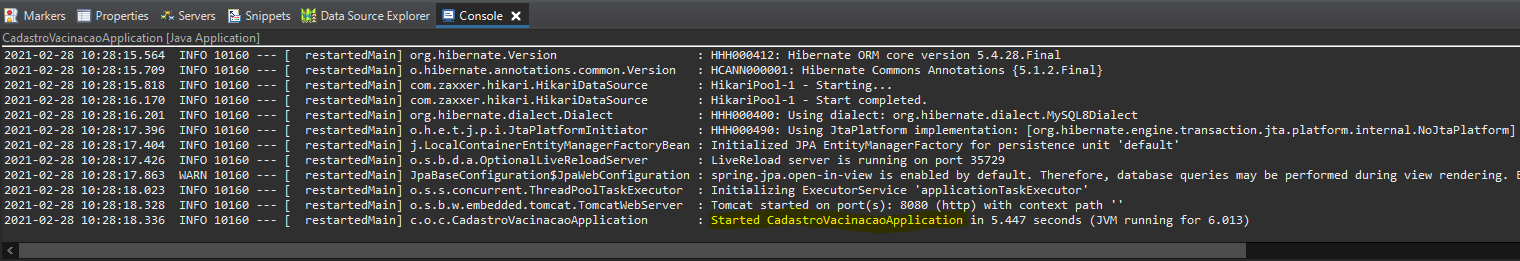
\*Não esqueça de marcar a opção de **pom.xml**, que é o arquivo que contém todas as nossas dependências, e a opção “Add Project(s) to working set” para definirmos a visualização do nosso projeto na área de trabalho da nossa IDE.  
Após isso, nosso projeto está importado e já temos uma Classe criada automaticamente! 

Mas antes de prosseguirmos com nossa API, precisamos instalar e configurar nosso gerenciador de banco de banco de dados, o MySQL Workbench 8.0. Você pode baixa-lo por meio do link: <https://dev.mysql.com/downloads/workbench/>.

Ok, agora que já criamos e importamos nosso projeto e instalamos o MySQL, vamos finalmente começar a construir nossa API!  
A primeira coisa que precisamos fazer é configurar o acesso da nossa aplicação ao banco de dados para que possamos enviar, salvar e manipular nossos dados, vamos fazer isso pelo arquivo **application.properties** que está no caminho src -> main -> resources.

Configurada nossa conexão com o banco de dados, vamos rodar nossa aplicação CadastroVacinacaoApplication, que está em **src/main/Java/com.orangeTalents.cadastroVacinacao**, clicando no botão direito Run As ->Java Application



Se tudo deu certo, a aplicação será “startada” no console e nosso banco de dados orangeTalents estará criado no MySQL:

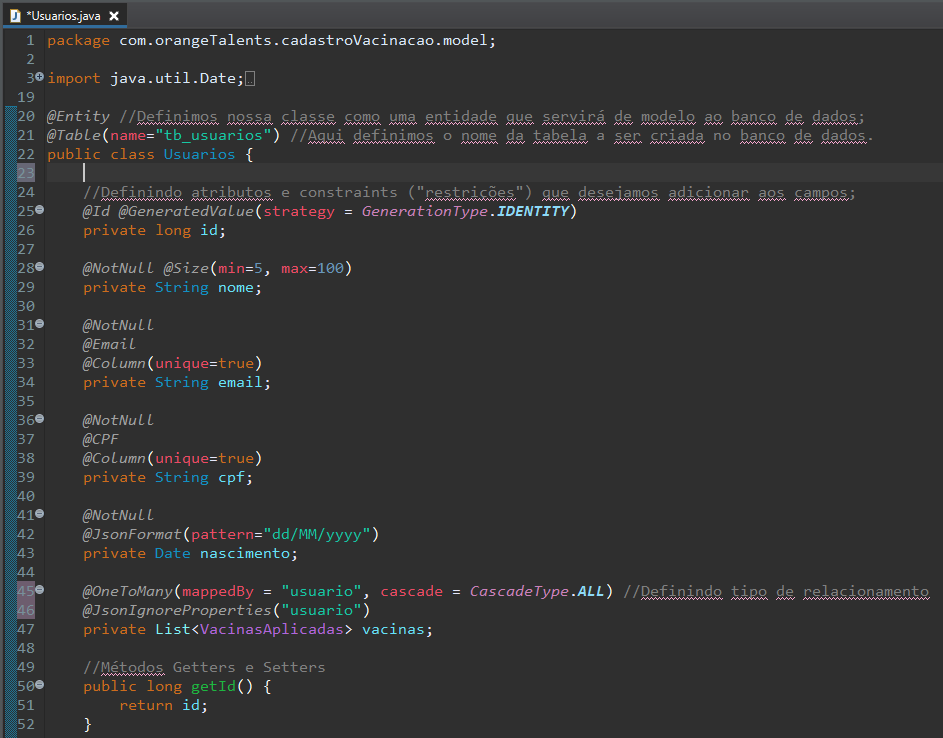
Criado nosso banco de dados, vamos falar um pouco sobre a arquitetura da nossa aplicação, quais classes precisaremos e como vamos configurá-las para nossa API rodar lindamente!  
Utilizaremos o padrão MVC (Model, View, Controller), onde cada camada tem uma atribuição específica.

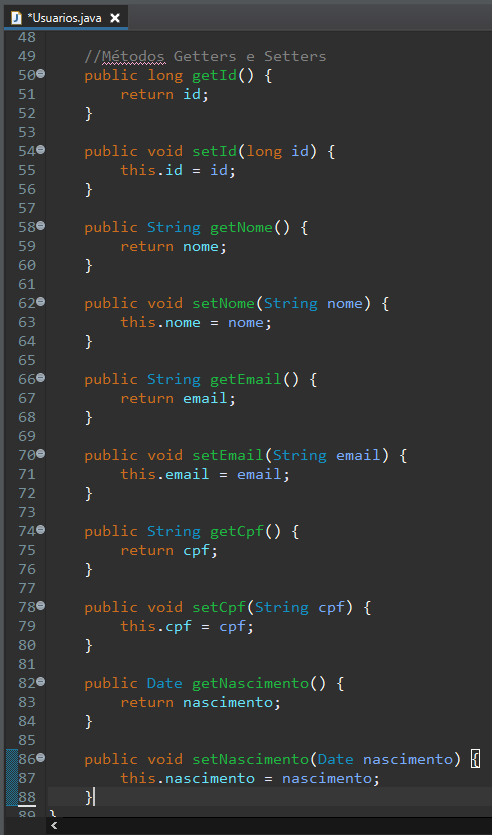
- A camada Model é a responsável pelo modelo da aplicação. Aqui criaremos os pacotes model (onde estarão as classes modelo para nossas tabelas no banco de dados), repository (responsável pelas transações diretas com o banco de dados) e service (responsável por nossas regras de negócios e que faz a ponte entre o controller e o repositor);

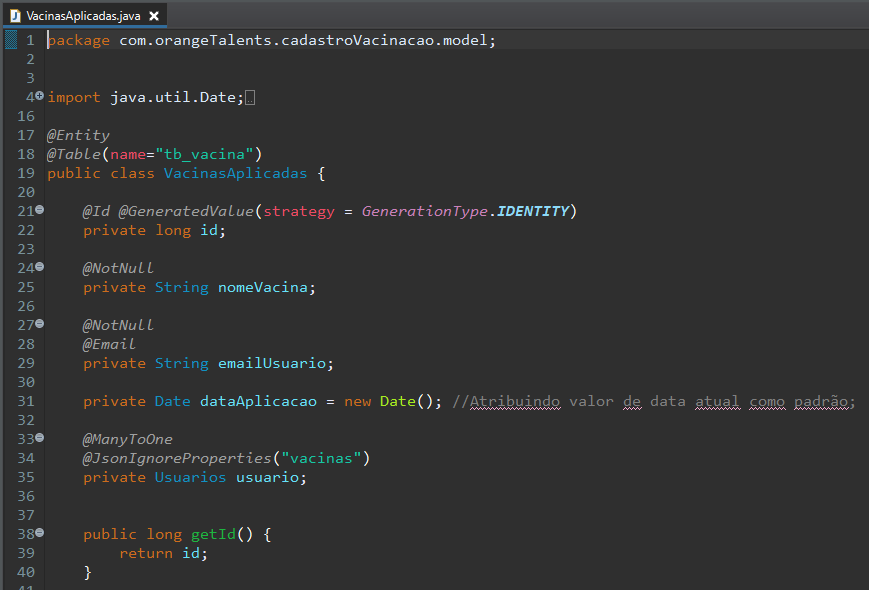
- Por falar em controller, ele é nossa próxima camada! É o responsável pela comunicação da nossa aplicação com a View (a camada visível ao usuário), e define os métodos que chamados nas requisições que faremos.

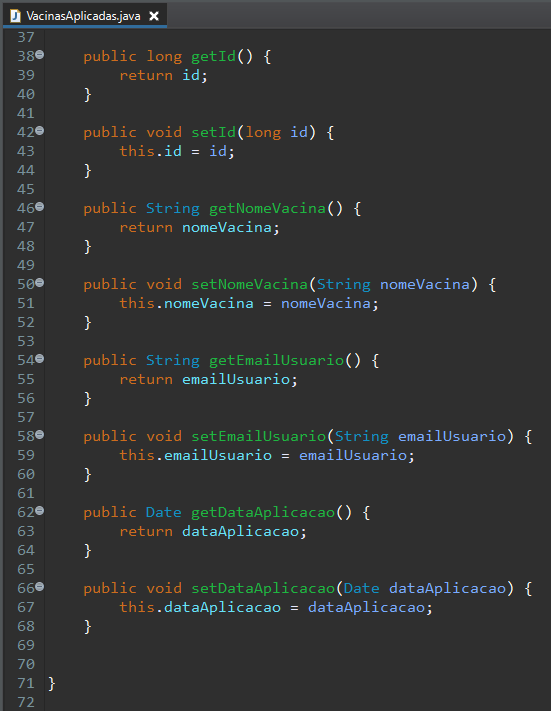
- View é a camada em que os dados são exibidos ao usuário e que criaremos mais para frente.

Criando nosso pacote Model:

Aqui criaremos duas classes: Usuários e VacinasAplicadas, que serão, respectivamente, os modelos para cadastrarmos usuários e aplicação de vacinas. Veja como ficarão:



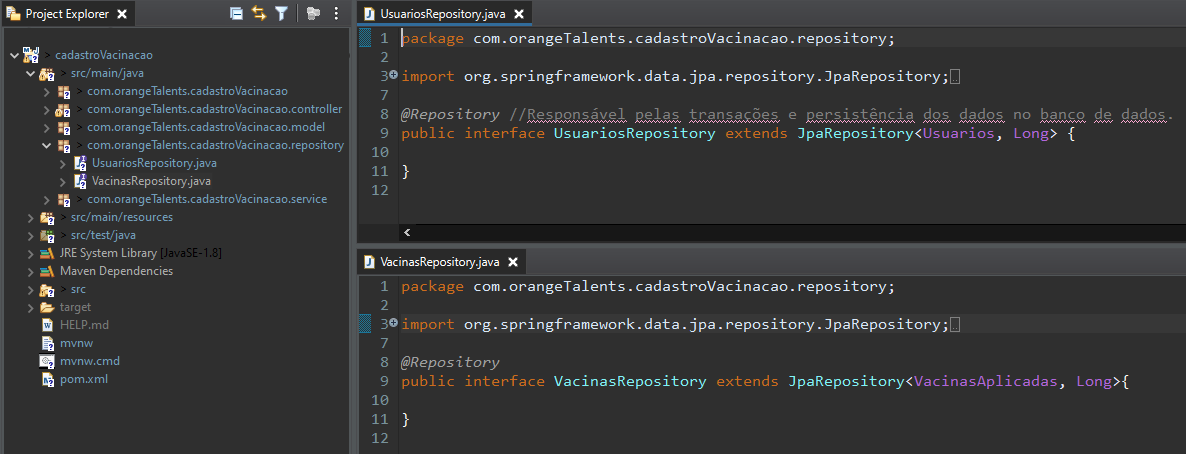




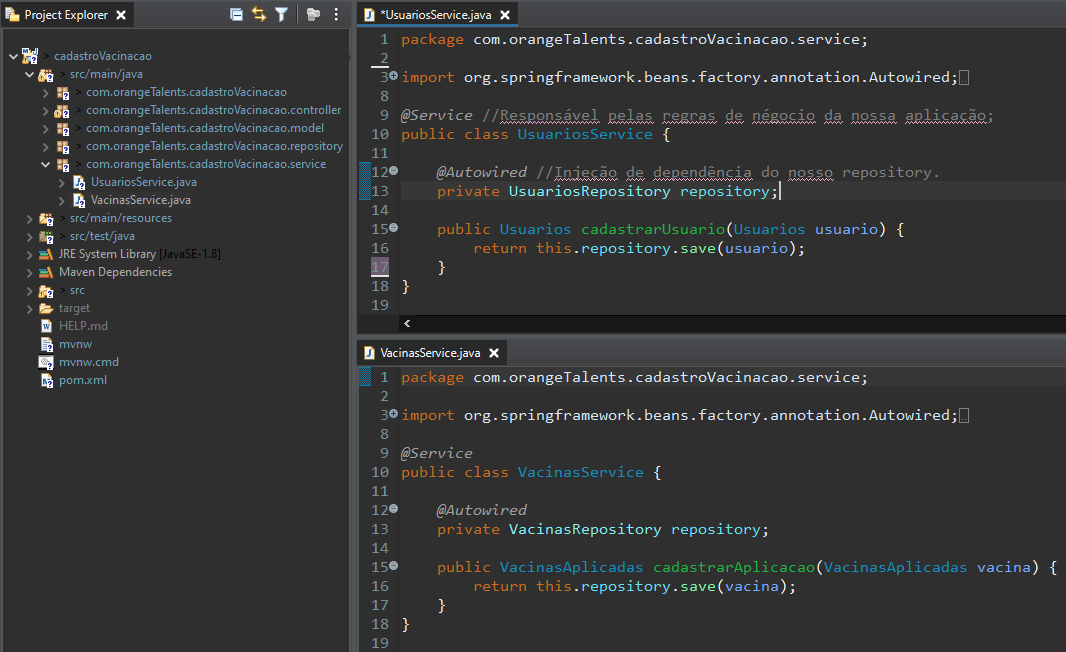
Você pode perceber alguns “@’s” no nosso código, eles são chamados de “anotações” (anotations) e você pode conhecer mais sobre eles neste link: <https://domineospring.wordpress.com/2016/07/13/guia-das-annotations-do-spring/>

.

Vamos construir agora nossas interfaces Repository para nos comunicarmos efetivamente com o banco de dados que criamos. Elas ficarão assim:



Agora é a vez dos Services, onde vamos definir um método de cadastro para nossos dados (notem que injetamos aqui nosso Repository e o utilizamos no método):



Após isso, é hora de criarmos nossos Controllers, nossa camada de comunicação entre as requisições realizadas pelo usuário e nossa aplicação. Como vamos apenas realizar cadastros por enquanto, criamos apenas um método Post, veja como ficou:

Se 