Criando a primeira API Rest em Spring Boot

Spring

O spring são pacotes de classes pré definidas em linguagem java que por sua vez facilita o dia de trabalho do desenvolvedor.

Estou criando um blog de um api rest com a ferramenta framework spring boot, sobre a criação de controle de veículos de usuários.

Ferramentas utilizada para essa api rest

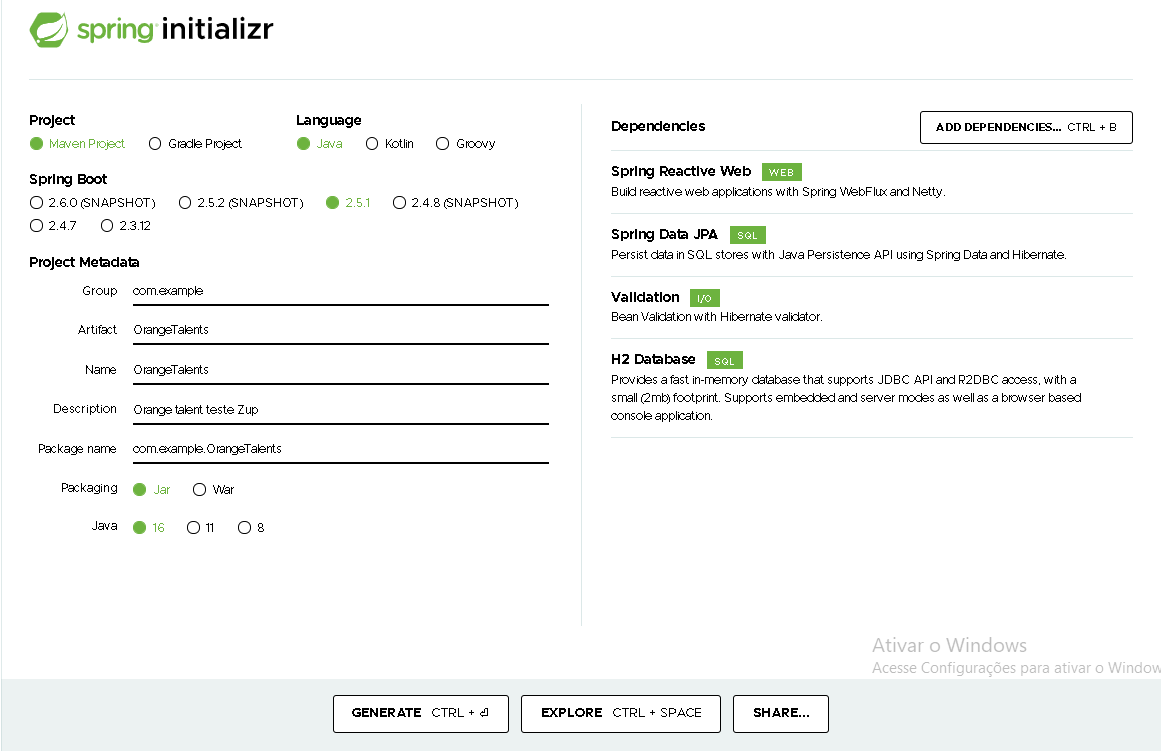
intellij community :, ferramenta de desenvolvimento do código da api na linguagem java que por sua vez é uma ferramenta de trabalho que traz conforto e clareza na amostra de suas aplicações e interações tanto com o usuário quanto com os mecanismos que ela interage

Banco de dados H2

o h2 é um banco de dados versátil de fácil abordagem tanto de aprendizado e apresentação e por ser um banco de dados na memória ajuda na velocidade de trabalho.

Postman

para visualização e ver meu projeto em andamento eu utilizei o postman, que é uma plataforma que ajuda no desenvolvimento de api que através dela não preciso criar interface de visualização e nem criar caminhos complicados para que ela me mostre algum resultado, assim facilitando meu dia de trabalho e economizando meu tempo.

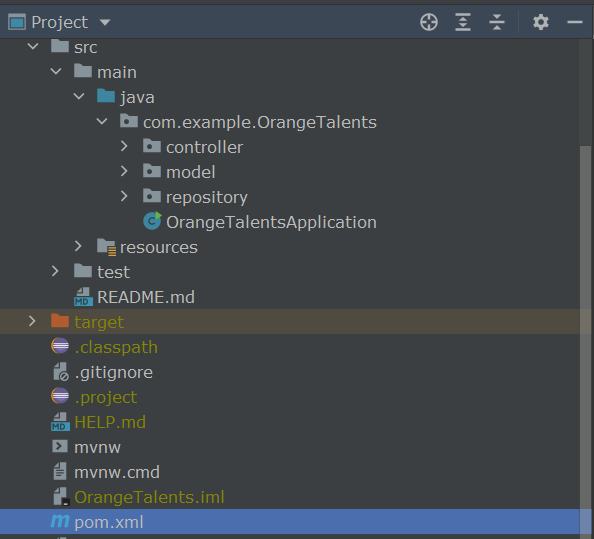
O primeiro passo para inicializar um projeto Spring é ir neste site <https://start.spring.io/> , este site comtém as inserções de depenedencias para a criação de um projeto em spring boot, essas dependências são: 

cada dependência tem sua função para que o projeto compile, a primeira dependência: spring web é para criar uma web service, spring data JPA é a camada da persistência de dados, validation é para validar as informações inseridas no banco de dados, H2 é a dependência relacionada ao banco h2, onde será salva as informações gerado do projeto.

Após as inserções das dependências, inserir o nome do projeto no campo Name, e inserir uma descrição do projeto no campo Description, verificar a versão selecionado do spring, a linguagem que será utilizada será java, o framework utilizado e o Maven, verificar como deseja exportar seu projeto em ponto jar ou em war( utilizei o jar), verificar também a versão da linguagem java que deseja utilizar(utilizei a versão 16), após tudo isso vamos clicar no botão Generate para gerar seu projeto.

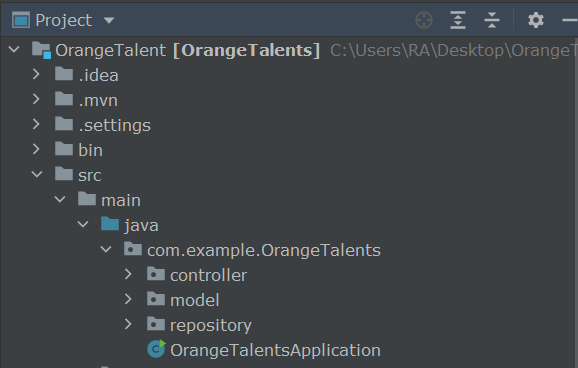
As dependências do projeto ficam no arquivo chamado pom xml:



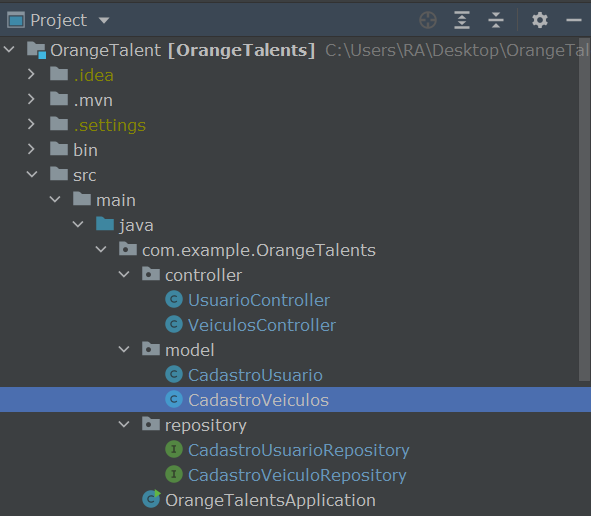


O projeto ao inicializar ele já vem com uma pasta inicial, ao clicar no arquivo, dentro da pasta será executado o projeto.

Criar os pacotes model, controller, repository



as classes criadas dentro da Model/Entity: CadastroUsuario e CadastroVeiculos:



no pacote model será criado a seguinte classe cadastroUsuario, com os seguintes atributos id nome, data de nascimento, cpf e email, contendo as anotações para validação, @Id indica que é um id,

@GeneradValue usado para que o banco de dados gerencie a entrada do Id,

@NotBlank o campo não pode ser vazio,

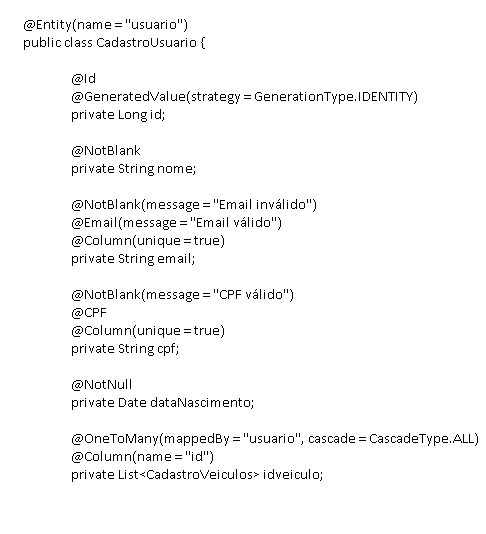
@NotNull o campo não pode ser nulo,

@CPF o campo segue o padrão brasileiro de 11 dígitos,

@Email segue o padrão email Ex: [email@email.com](mailto:email@email.com)

@Collum usado para referenciar uma coluna específica no banco de dados, onde não deve ser inserida uma informação repetida.

@OneToMany e a ligação de uma tabela a outra onde um usuário contém vários itens,



na questão da data do usuário foi usado o Date com a importação

import java.util.Date;

no atributo de ligação @OneToMany foi usado uma List onde são vários veículos para um usuário.O List com a importação import java.util.List;

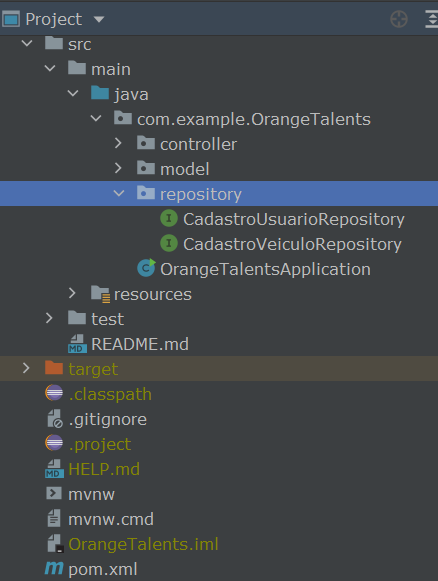
Na classe veículos foi inserido os seguintes atributos

:

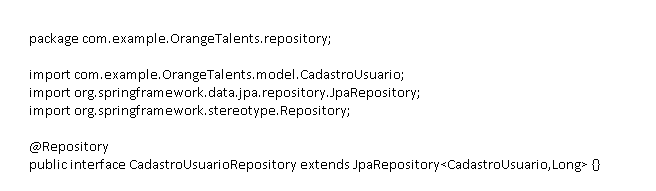
O REPOSITORY é uma interface que faz o controle e a consulta de entrada e saída dos dados entre um endpoint e outro.

CadastroUsuario Repository

CadastroVeiculo Repository



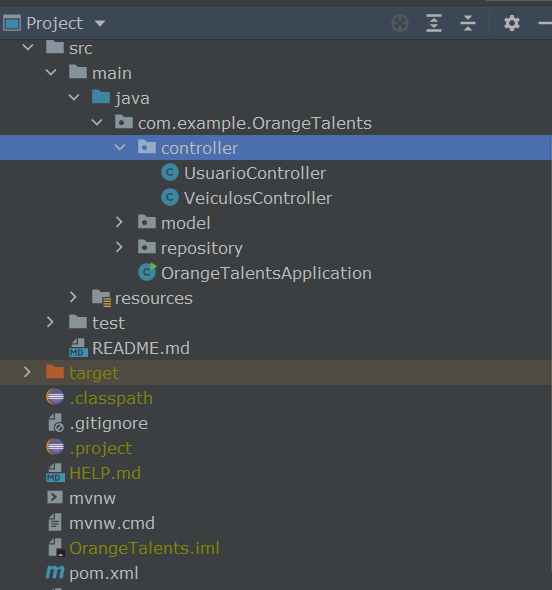
no repository fiz uma extensão para a interface JpaRepository onde ela consulta, e faz atualização de dados.



no controller crie as seguintes classe:

UsuarioController

Veiculos Controller



O controller é responsável pelo endpoints/métodos de requisição HTTP GET, POST entre outros(DELETE, PUT, GET, POST).



A anotação @RestController informa ao spring que esta é uma classe de controller, controla as requisições de entrada e saída.

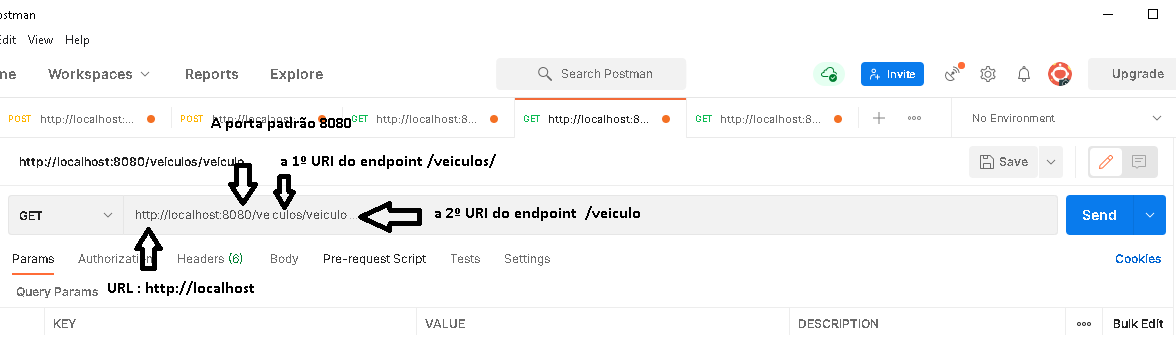
A anotação @CrossOrigin irá manter a segurança das informações passadas pela URI em seu corpo.

@RequestMapping é usada para dar nome às URI para ser acessada, sendo que pode ser colocado também o nome da URI no endpoint

ex: @GetMapping(value = "/usuario")

Para a consulta dos endpoits pelo postmanl ou na web passar as URL para o acesso

[http://localhost:8080/veiculos/veiculo](http://localhost:8080/usuarios/usuario)



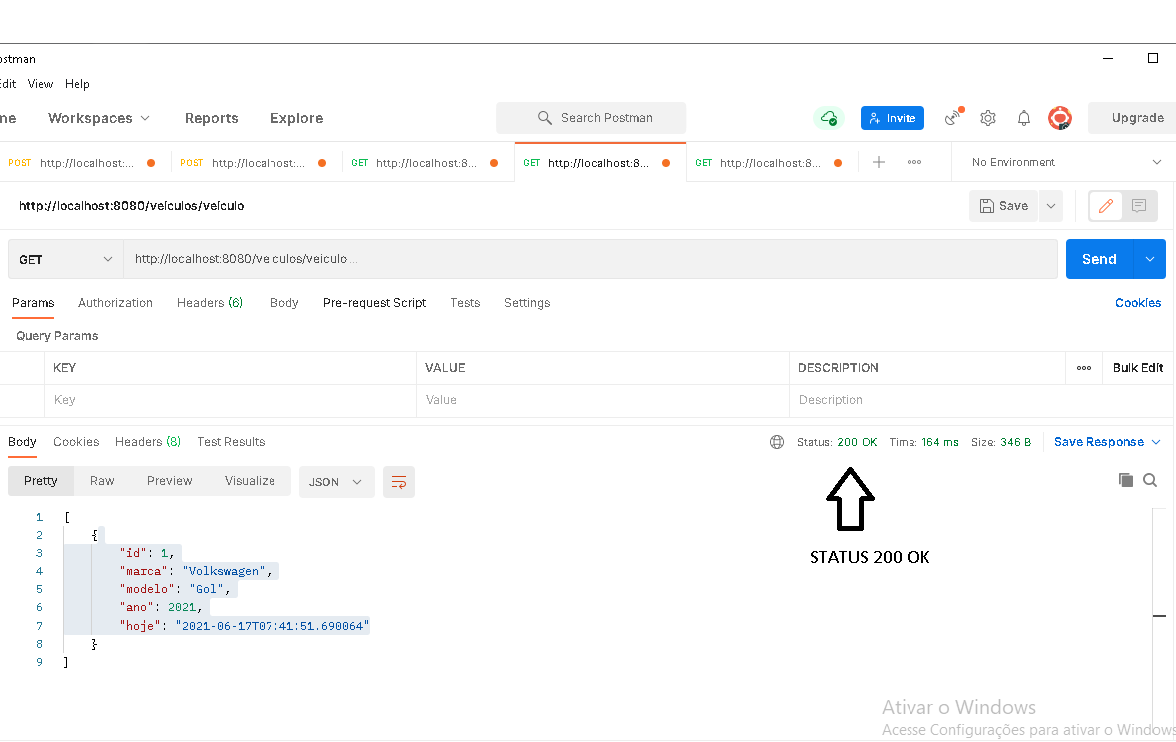
A anotação @Autowired ele faz a injeção de dependência, passando a responsabilidade ao spring.

A anotação @GetMapping está realizada na busca das informações inseridas no banco.

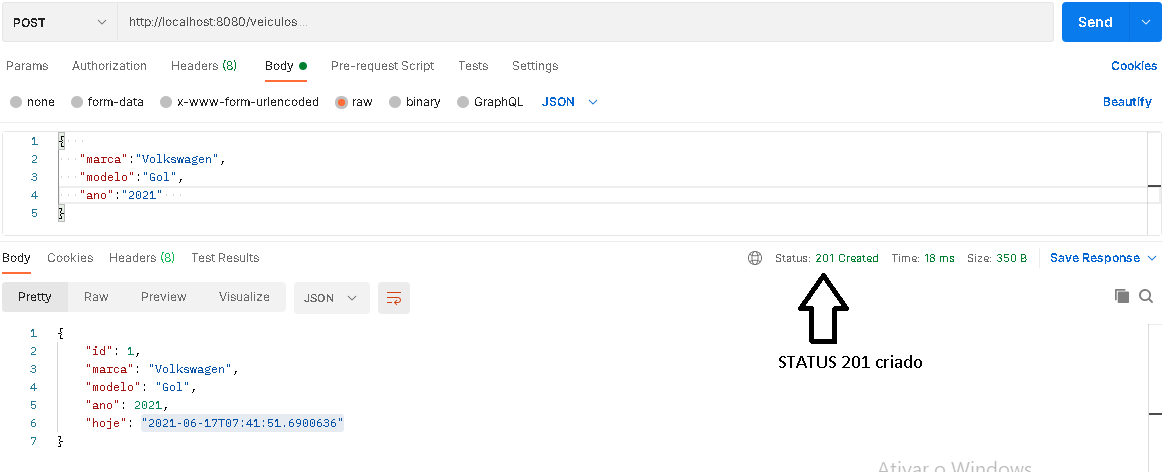
A anotação @PostMappinmg está inserindo os dados no banco do usuário.

O postman foi utilizado para a consulta dos endpoints Post e Get;

O GET: Esta trazendo os dados do banco, retornado o STATUS 200 onde houve sucesso ao trazer as informações.



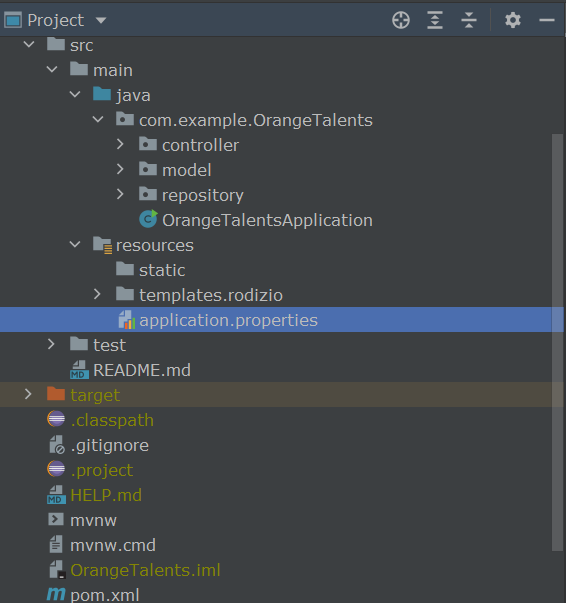
O POST salva as informações do usuário no banco retornado status 201:

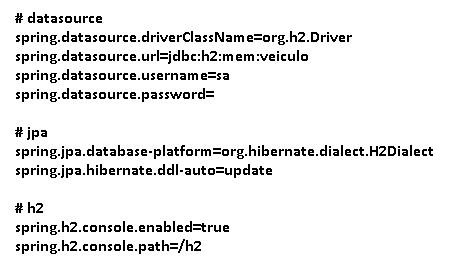


O POST erro no cadastro das informações status 400:



Para salvar as informações no banco de dados precisa ser feito as configurações no application.properties





para acessar o h2 no navegador usar este link : <http://localhost:8080/h2>

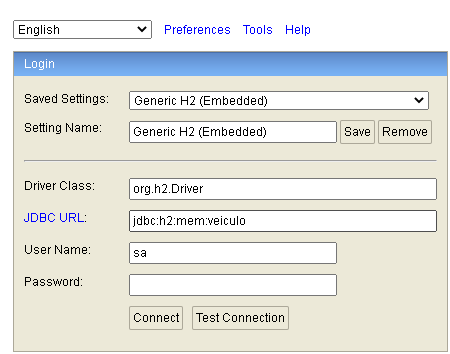
para acessar o painel do h2 inserir a seguinte informaçãospring.datasource.url=**JDBC:H2:MEM:VEICULO**, esta informação colocar no campo **JDBC URL** do painel h2.

o password e user tem que ser igual aos da linhas spring.datasource.username=sa

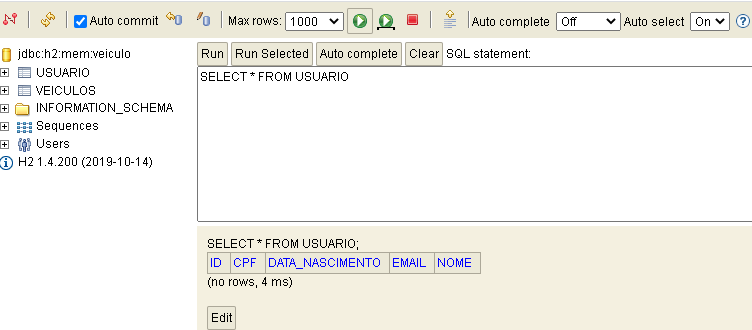
spring.datasource.password=

clicar em CONNECT para acessar.

As informações foram salvas no H2, sua interface inicial.



Dentro do banco de dados h2:



Informações inserida no h2:

