Orange TALENTS

FALA GALERA BELEZA? VAMOS CODAR HOJE, O DESAFIO SERÁ,,,, CRIAR UMA:

API REST CONTROLE PARA VACINAÇÃO

ZUPER RESPONSÁVEL: JONATHAN C. DE PAULA

ZUP

Orange TALENTS

Antes de iniciarmos o projeto, vamos ao escopo do que iremos elaborar:

- 1º Vamos construir um cadastro de usuários, sendo obrigatório os seguintes dados como: nome, e-mail, CPF e data de nascimento, onde e-mail e CPF serão únicos;

- 2º Iremos criar um cadastro de aplicação de vacinas, sendo obrigatórios os seguintes dados como: nome da vacina, e-mail do usuário e a data que foi realizada a vacina;

Orange TALENTS

Antes de iniciarmos o projeto, vamos ao escopo do que iremos elaborar:

- 1º Vamos construir um cadastro de usuários, sendo obrigatório os seguintes dados como: nome, e-mail, CPF e data de nascimento, onde e-mail e CPF serão únicos;

- 2º Iremos criar um cadastro de aplicação de vacinas, sendo obrigatórios os seguintes dados como: nome da vacina, e-mail do usuário e a data que foi realizada a vacina;

Orange TALENTS

Ferramentas que iremos utilizar, serão:

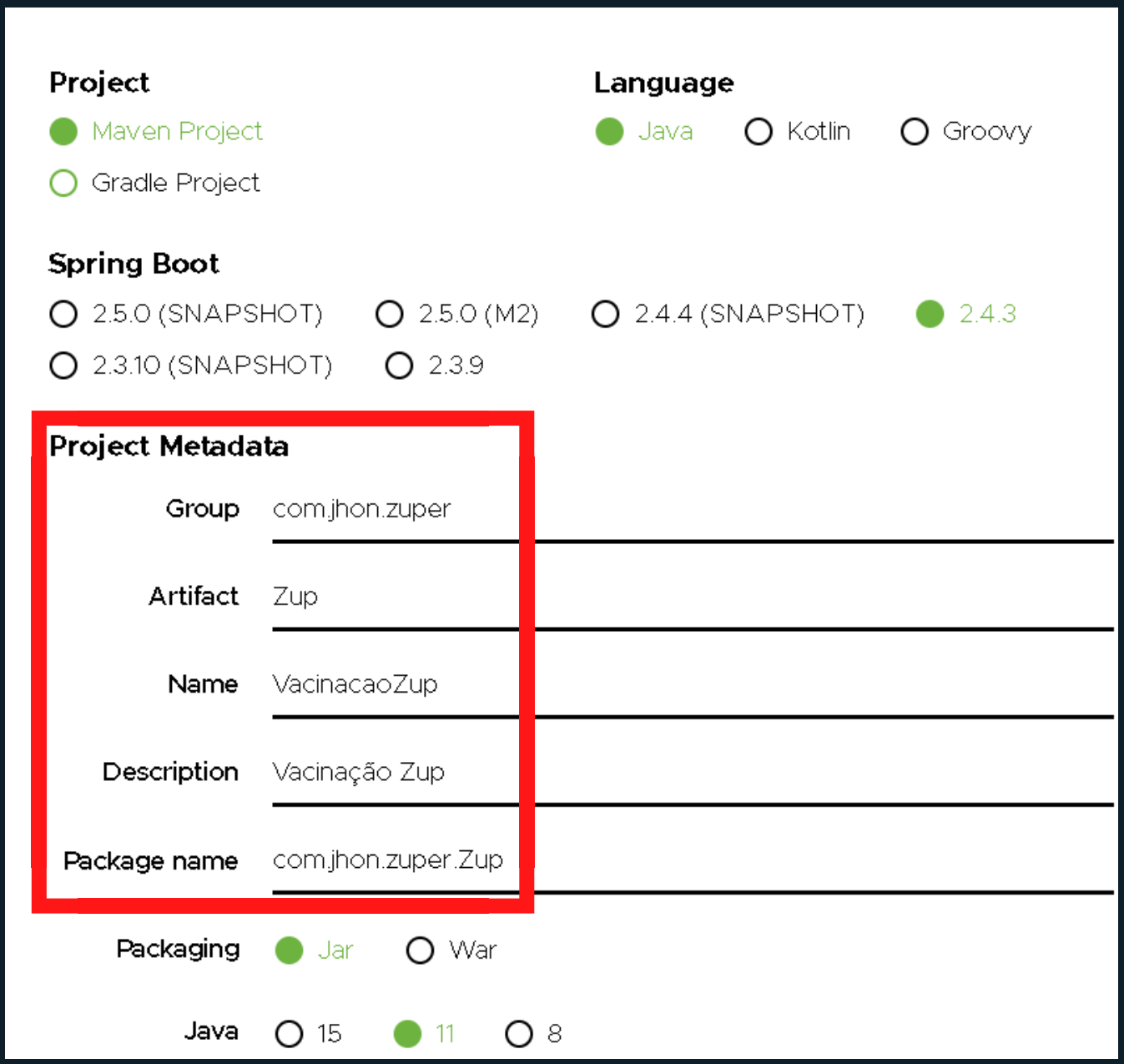
Antes de iniciarmos o projeto, vamos falar um pouco sobre as tecnologias usadas:

- Java: Será a Linguagem de Programação que optamos para criar nosso projeto, trata-se de linguagem das mais importantes e usadas atualmente;
- Spring Tool Suite: É a IDE que vamos utilizar para codar, ela trás muitas facilidades e robustez ao código. Iremos utilizar algumas de suas facilidades aqui;
- MySQL: É o banco de dados, do estilo relacional (ou seja, estruturado por tabelas);
- Postman: Serve para testarmos nossa API, por meio do envio de requisições HTTP e da análise do seu retorno;
- XAMPP: É um pacote com os principais servidores de código aberto, ele vai nos ajudar abrir as portas para conexão com o banco de dados;

Orange TALENTS

Spring

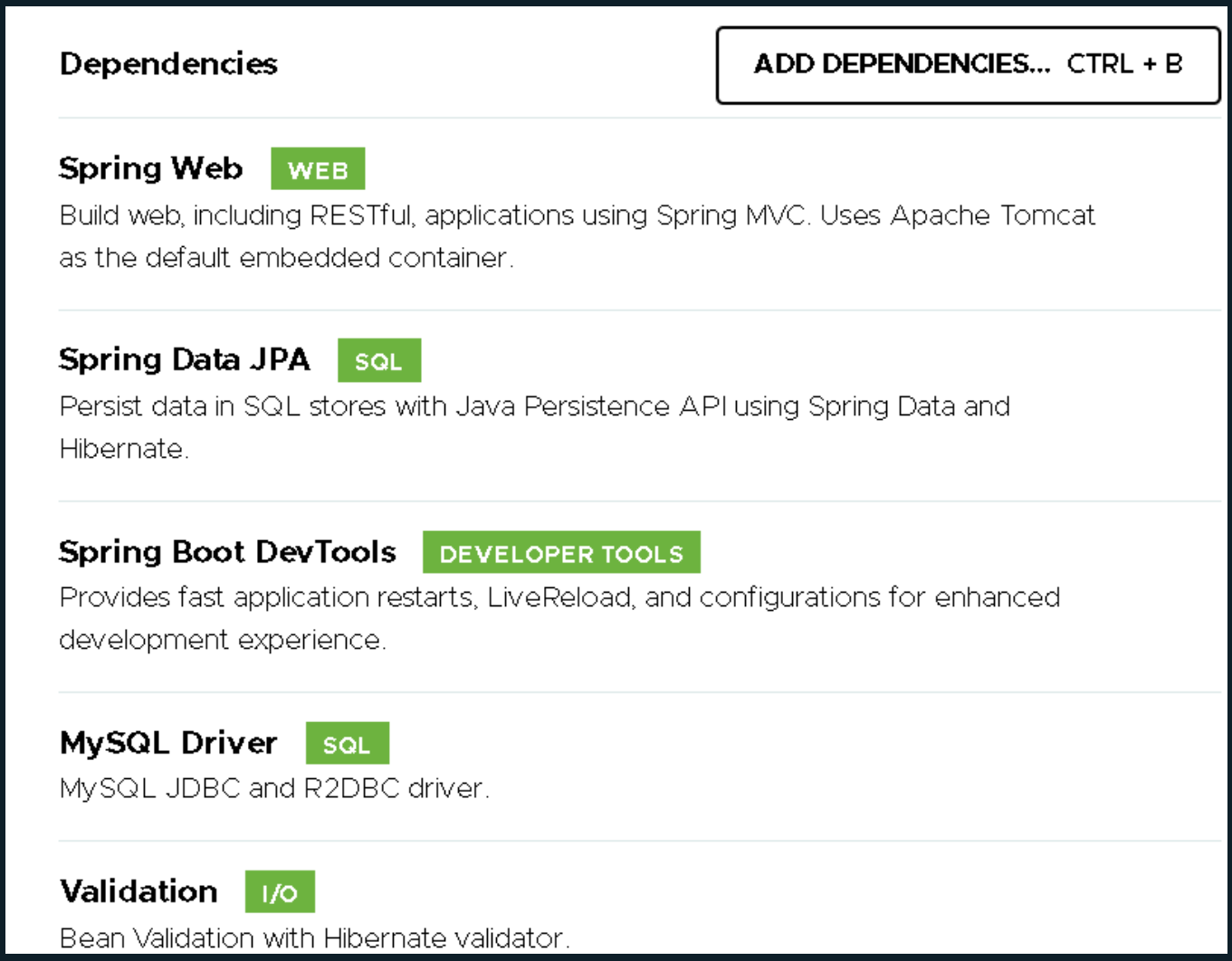
Vamos falar um pouco mais sobre o Spring (STS) e como iremos usá-lo, primeiramente devemos configurar o projeto, para que qualquer pessoa possa fazer - basta acessar o link https://start.spring.io/ e alterar a parte destacada em vermelho:



ZUP

Orange TALENTS

Agora devemos escolher as dependências, lembra das facilidades que o Spring nos dá - seguem as que devemos selecionar:



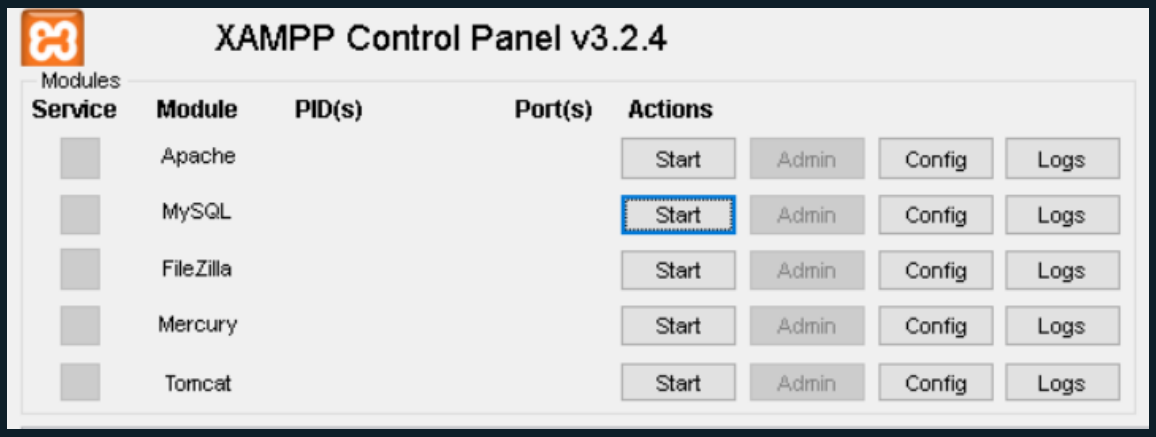
Tá, mas você deve estar se perguntando o que são essas dependências que escolhemos. As facilidades Spring: Web, Data JPA e DevTools, servem respectivamente para criar um ambiente Web, persistência de dados e criação de repositórios, e facilitação na utilização do sistema e reinicialização automática. Quanto ao MySQL é isso mesmo, para podermos usar o banco de dados MwSQL, e o Validation - serve para validarmos alguns objetos.

ZUP

Orange TALENTS

Tudo certo aí? Agora vamos avançar mais um pouco. mas antes de tudo vamos usar o XAMPP, lembra dele? Não podemos esquecer, senão nosso Banco de Dados não vai funcionar e nosso projeto não vai rodar, tá bom?

Como está:

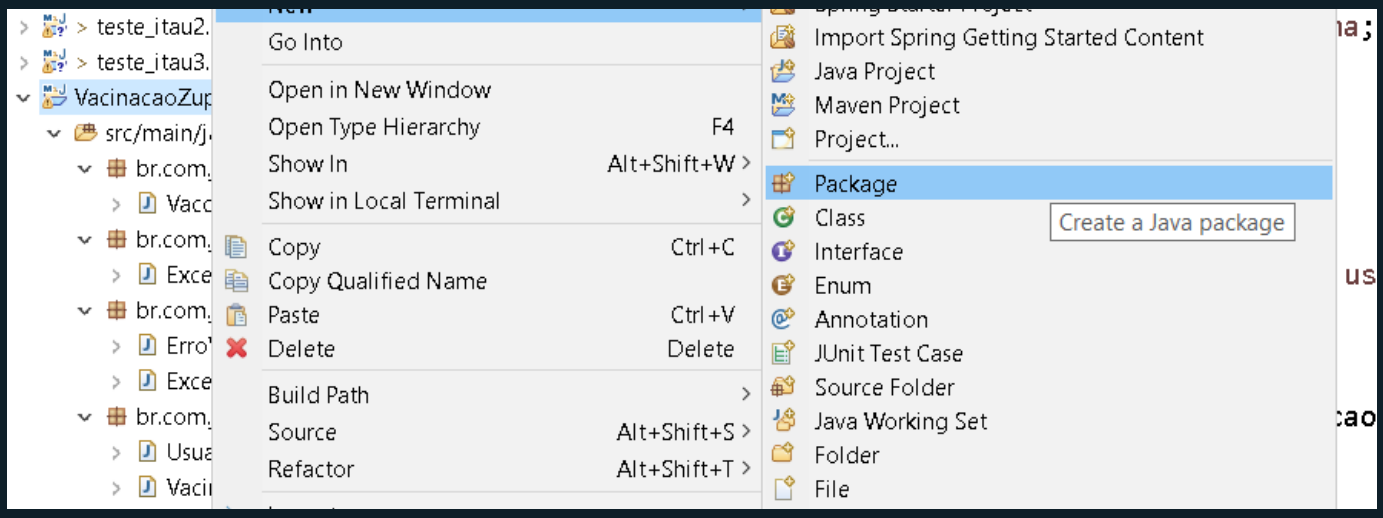


Como deve ficar:

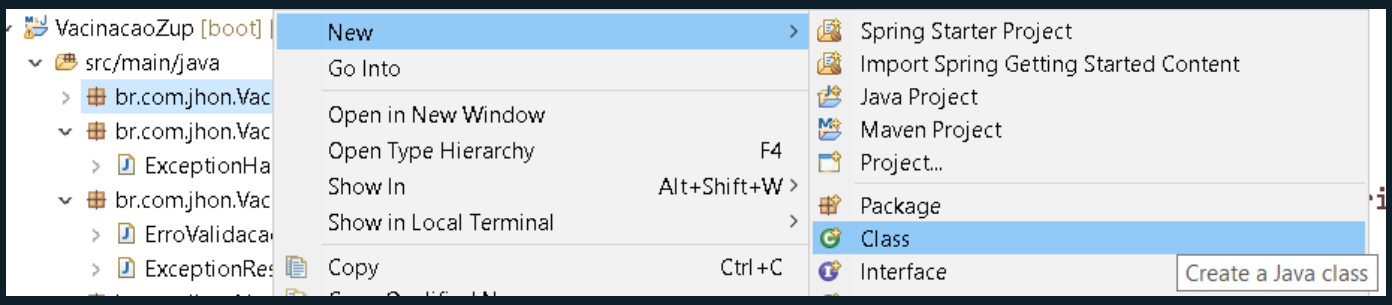
ZUP

Orange TALENTS

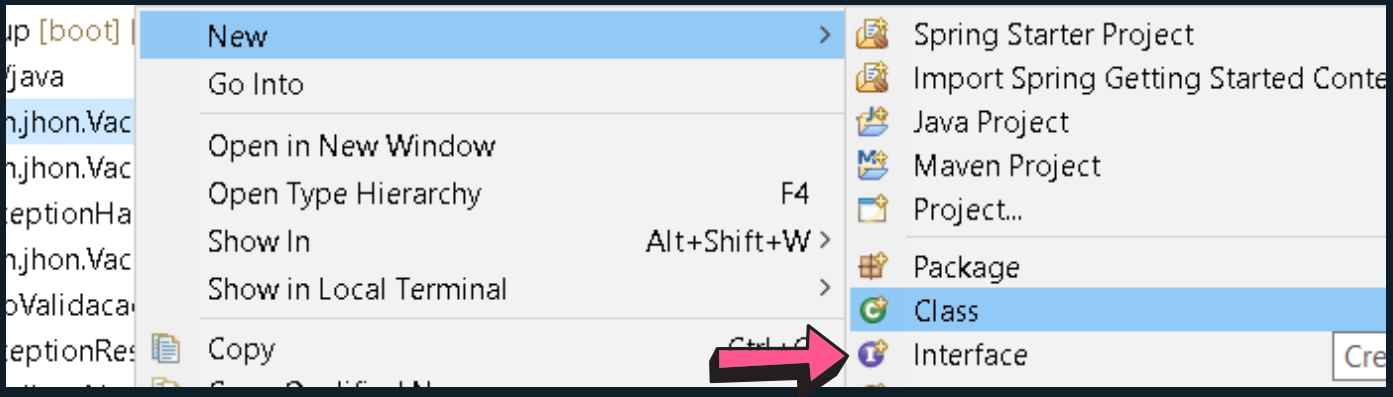
Agora, bora criar nossos pacotes click com o botão direito do mouse na pasta do projeto depois -> new -> Package:



Agora, bora criar nossas classes click com o botão direito do mouse nos pacotes do projeto depois -> new -> class:



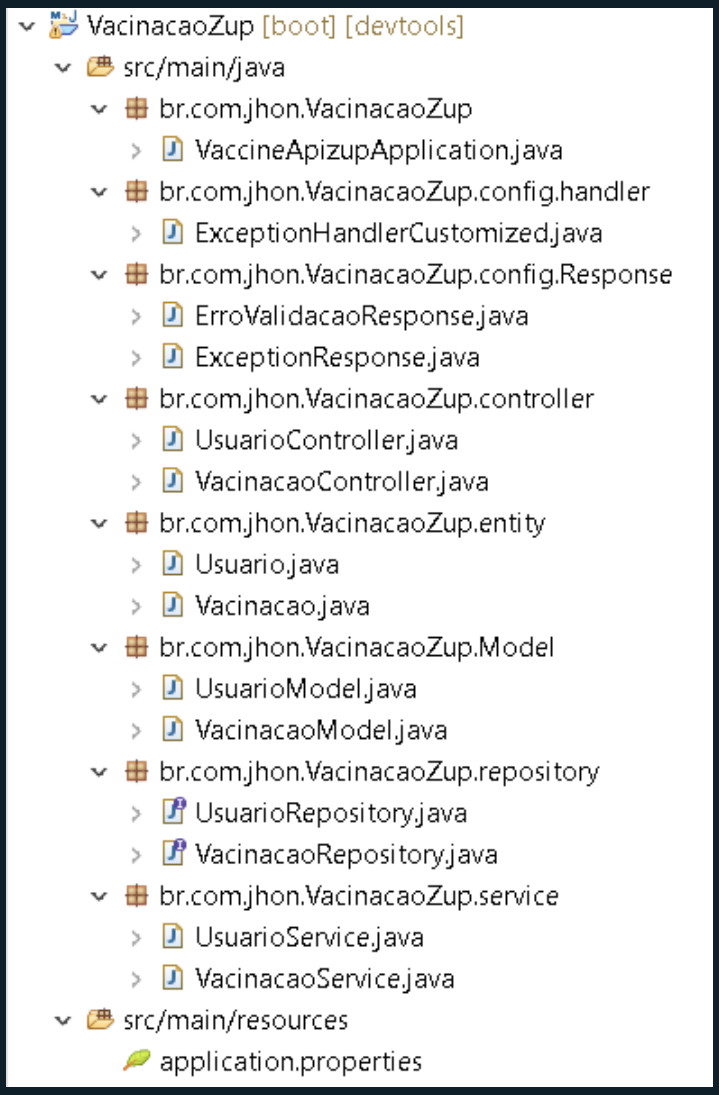
Um ponto de tenção agora, no pacote repository ao invés de criarmos uma classe, vamos usar - Interface, destacado abaixo:



ZUP

Orange TALENTS

Como deve ficar:

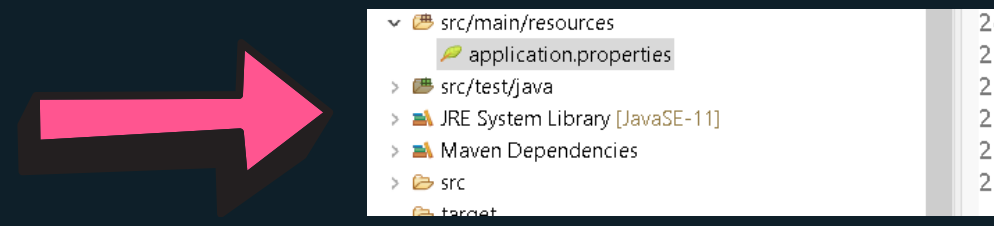


- Model: Aqui serão as classes de abstração de dados (objetos);
- Controller: Aqui vamos ter os controladores que ficarão responsáveis por receber as requisições, executar ações e retornar os resultados;
- Entity: Aqui ficaram nossas classes de entidades, responsável por persistidas no banco;
- Repository: Aqui estão as interfaces responsáveis por fazer transições com o banco de dados;
- Service: Onde vão ficar as classes de lógica de negócio e de operações envolvendo as entidades.
- Config: Aqui vai ficar um Handler para capturar as exceções, retornar Status Codes e retornar uma Response personalizada e mais sucinta que facilitará na identificação do erro e na aparência da resposta.

ZUP

Orange TALENTS

Agora vamos configurar o application.properties e inserir algumas informações:




```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost/db_vacinacaoZUP?createDatabaseIfNo
spring.datasource.username=root
spring.datasource.password=
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

Como deve ficar:

spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost/db_vacinacaoZUP?createDatabaseIfNotExist=true&serverTimezone=UTC&useSSl=false

spring.datasource.username=root

spring.datasource.password=

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format_sql=true

ZUP

Orange TALENTS

O que acabamos de fazer?

Bom, acabamos de configurar nosso banco de dados (MySQL), bem como o usuário (ROOT) e a senha (NULA) do banco de dados no arquivo. Vale lembrar que a propriedade “ddl-auto ” , serve para recriar o banco de dados todas as vezes que o projeto se iniciar.

AVISO DE SPOILEIR, VEJA COMO ISSO APARECE NO MYSQL:

ZUP

Orange TALENTS

BORA

OBS:

Agora vamos por a mão na massa, então muita atenção nos próximos passos, qualquer informação mal colocada pode impedir nosso projeto de funcionar perfeitamente, então atenção aos detalhas, primeiramente, vou explicar as informações que devemos colocar em cada pacote e classe, depois irei mostrar como devem ficar, ok?

CODAR

ZUP

Orange TALENTS

PRIMEIRAMENTE VAMOS ATUAR NOS PACOTES: ENTITY E MODEL

Essa vai ser a classe responsável por guardar as informações dos usuários, iremos criar os atributos *(id,nome,email,cpf e dataNascimento)* e seus *getters* e *setters*. Também adicionamos as *annotations* da *jpa* e de validações do *bean validation.*

@Entity: determina que a classe vai ser gerenciada pela jpa/hibernate, isso vai fazer com que a tabela seja criada automaticamente(caso configurado no properties)
@Table: especificar detalhes na tabela que será criada no banco
@UniqueConstraint: essa parte é responsável por criar uma constraint dentro do banco, vamos utilizar ela para criar a constraint que vai setar os campos email e cpf como únicos no banco, impedindo que os dados desses campos se repitam.
@Id: determina a chave primária da entidade
@GeneratedValue: faz com que o valor do id seja gerado e controlado automaticamente de acordo com o banco de dados.

Validações feitas pelo bean validation e aquela dependência que falamos anteriormente :
@NotNull: Impede o campo de ser nulo
@NotBlank: Impede o campo de ser nulo, e ter um valor vazio, exemplo: " ";
@Email: faz a verificação para ver se é um email válido
@CPF: faz a verificação para ver se é um CPF válido

Orange TALENTS

Como devem ficar:

```java
public class UsuarioModel {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotBlank
    private String nome;
    @Email @NotBlank @Column(unique = true)
    private String email;
    @CPF @NotBlank @Column(unique = true)
    private String cpf;
    @NotNull
    private LocalDate dataNascimento;

    public UsuarioModel() {}

    public UsuarioModel(Usuario usuario) {
        this.id = usuario.getId();
        this.nome = usuario.getNome();
        this.email = usuario.getEmail();
        this.cpf = usuario.getCpf();
        this.dataNascimento = usuario.getDataNascimento();
    }
```

```java
@Entity
@Table(uniqueConstraints ={
    @UniqueConstraint(columnNames = "email", name = "email_uk"),
    @UniqueConstraint(columnNames = "cpf", name = "cpf_uk")}
)
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotBlank
    private String nome;
    @Email @NotBlank
    private String email;
    @CPF @NotBlank
    private String cpf;
    @NotNull
    private LocalDate dataNascimento;
```

ZUP

Orange TALENTS

Como devem ficar:

```java
public class VacinacaoModel{
    private Long id;
    @NotBlank
    private String nomeVacina;
    @NotBlank @Email
    private String usuarioEmail;
    @NotNull
    private LocalDate dataAplicacao;

    public VacinacaoModel() {
    }

    public VacinacaoModel(Vacinacao vacinacao) {
        this.id = vacinacao.getId();
        this.nomeVacina = vacinacao.getNomeVacina();
        this.usuarioEmail = vacinacao.getUsuario().getEmail();
        this.dataAplicacao = vacinacao.getDataAplicacao();
    }

    public static Vacinacao converter(VacinacaoModel vacinacaoModel, Usuario usuario){
        Vacinacao vacinacao = new Vacinacao();
        vacinacao.setNomeVacina(vacinacaoModel.getNomeVacina());
        vacinacao.setUsuario(usuario);
        vacinacao.setDataAplicacao(vacinacaoModel.getDataAplicacao());
        return vacinacao;
    }
```

```java
@Entity
public class Vacinacao {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @NotBlank
    private String nomeVacina;
    @NotNull @ManyToOne @JoinColumn(name = "user_id")
    private Usuario usuario;
    @NotNull
    private LocalDate dataAplicacao;
```

ZUP

Orange TALENTS

AGORA O PACOTE: REPOSITORY

Como devem ficar:

UsuarioRepository.java | VacinacaoRepository.java

```java
package br.com.jhon.VacinacaoZup.repository;

import org.springframework.data.jpa.repository.JpaRepository;

@Repository
public interface VacinacaoRepository extends JpaRepository<Vacinacao, Long> {
}

```

```java
package br.com.jhon.VacinacaoZup.repository;

import org.springframework.data.jpa.repository.JpaRepository;

@Repository
public interface UsuarioRepository extends JpaRepository<Usuario, Long> {
    Optional<Usuario> findUsuarioByEmail(String email);
}

```

O repository são as interfaces responsáveis por fazer as transações com o banco de dados. Para definir uma interface como repository basta colocar a annotation @Repository em cima dela, abaixo estão as repositories do usuário e da vacinação.

@Repository: informa para a JPA que esse é um repository.

Nossas interfaces estão estendendo a interface JpaRepository que já vem por padrão com diversas funcionalidades básicas para o banco, como por exemplo as funcionalidades de CRUD, na hora de estender a classe também devemos informar para ela qual é a classe(entidade) que ela vai representar e qual é o tipo do ID, no caso usamos o Long.

Orange TALENTS

AGORA O PACOTE: CONTROLLER

Agora vamos implementar os nossos controladores, eles são os responsáveis por receber as requisições, executar uma ação respectiva e devolver a response para o cliente.

Abaixo temos os controllers responsáveis pelos usuários e pelas vacinações, dentro de cada um deles temos apenas um endpoint, o create que vai receber um JSON no body da request e vai passar ele para nossa entity e salvá-la no banco de dados utilizando o repository, e então retornar uma response para o solicitante.

@RestController: diz ao spring que esse é o nosso controlador e que ele irá receber requisições, ele já coloca por padrão a anotação @ResponseBody(indica que o retorno do método deve ser automaticamente inserido na response para o cliente)
@RequestMapping: indica o caminho de requisição para chamar esse controller
@Autowired: faz o spring injetar esse recurso automaticamente sem que precisemos passar pra ele.(conhecido como injeção de dependências)
@PostMapping: Significa que sempre que for realizado uma requisição do tipo post para o caminho especificado no @RequestMapping, ele vai executar o método abaixo.
@Valid: Essa anotação vai cuidar para que os campos que marcamos com as annotations de validação sejam devidamente preenchidos, caso alguma validação falhe, ela lança uma exceção e não permite a criação do objeto.
@RequestBody: Indica que o parâmetro do método deve ser associado ao corpo da solicitação HTTP

Orange TALENTS

Como devem ficar:

suarioController.java | VacinacaoController.java

```java
package br.com.jhon.VacinacaoZup.controller;

import org.springframework.beans.factory.annotation.Autowired;

@RestController
@RequestMapping("/usuario")
public class UsuarioController {

    @Autowired
    UsuarioService usuarioService;

    @PostMapping
    public ResponseEntity<UsuarioModel> create(@Valid @RequestBody UsuarioModel usuario, UriComponentsBuilder uriBuilde
        UsuarioModel usuarioModel =  new UsuarioModel(usuarioService.create(UsuarioModel.converter(usuario)));
        URI uri = uriBuilder.path("/usuario/{id}").buildAndExpand(usuarioModel.getId()).toUri();
        return ResponseEntity.created(uri).body(usuarioModel);
    }
}
```

UsuarioController.java | VacinacaoController.java

```java
package br.com.jhon.VacinacaoZup.controller;
import org.springframework.beans.factory.annotation.Autowired;

@RestController
@RequestMapping("/vacinacao")
public class VacinacaoController {

    @Autowired
    VacinacaoService vacinacaoService;

    @Autowired
    UsuarioService usuarioService;

    @PostMapping()
    public ResponseEntity<VacinacaoModel> create(@Valid @RequestBody VacinacaoModel vacinacaoModel,
        UriComponentsBuilder uriBuilder){
        Usuario usuario = usuarioService.findByEmail(vacinacaoModel.getUsuarioEmail());
        VacinacaoModel vacinacao = new  VacinacaoModel(vacinacaoService.create(VacinacaoModel.converter
        (vacinacaoModel, usuario)));
        URI uri = uriBuilder.path("/vacinacao/{id}").buildAndExpand(vacinacao.getId()).toUri();
        return ResponseEntity.created(uri).body(vacinacao);
    }
}
```

ZUP

Orange TALENTS

AGORA O PACOTE: SERVICE

UsuarioService.java | VacinacaoService.java

```java
package br.com.jhon.VacinacaoZup.service;

import org.springframework.beans.factory.annotation.Autowired;

@Service
public class VacinacaoService {
    @Autowired
    VacinacaoRepository vacinacaoRepository;

    public Vacinacao create(Vacinacao vacinacao){
        return vacinacaoRepository.save(vacinacao);
    }


}
```

UsuarioService.java | VacinacaoService.java

```java
package br.com.jhon.VacinacaoZup.service;

import org.springframework.beans.factory.annotation.Autowired;

@Service
public class UsuarioService {

    @Autowired
    UsuarioRepository usuarioRepository;

    public Usuario create(Usuario usuario) {
        return usuarioRepository.save(usuario);
    }

    public Usuario findByEmail(String email) {
        Optional<Usuario> entity = Optional.ofNullable(usuarioRepository.findUsuarioByEmail(email)
                .orElseThrow(() -> new EntityNotFoundException("Usuario não encontrado")));
        return entity.get();
    }

}
```

Agora vamos criar as nossas camadas de serviços, nelas ficarão os métodos para CRUD, pesquisas e outras ações que estejam relacionadas às nossas entidades, aqui fazemos com que as services se comuniquem com a camada de repositories para realizar as transações com o banco de dados.

@Service: Indicamos ao spring que essa classe pertence a nossa camada de serviço

ZUP

Orange TALENTS

AGORA OS PACOTES: HANDLER E RESPONSE

Aqui o Handler irá pegar todas as exceptions que forem lançadas para o cliente e fazer uma tratativa com ela, seja formatar ou especificar de forma melhor o código que ela deve lançar. Também iremos criar uma classe de response personalizada para retornar para o cliente em caso de erros.

Iremos criar nossas classes dentro de seus próprios packages dentro do package config.

Vamos criar duas Responses personalizadas, uma contendo o campo e a mensagem do erro para quando ocorrem erros nas validações dos dados, e uma geral contendo o campo, a data e hora e a mensagem do erro para todas as outras exceptions que o servidor lançar

E o handler será o responsável por capturar as exceptions:

@RestControllerAdvice: Indica ao Spring que teremos um interceptador que fica escutando as saídas de dados até encontrar algo pelo qual ele seja responsável(no caso Exceptions) ele também adiciona por padrão a ***@ResponseBody*** nos métodos.
@ExceptionHandler: Caso seja detectado uma exception da classe especificado dentro da *annotation* ele executa o método abaixo.
@ResponseStatus: Serve para indicar o *status code* que deve ser retornado ao final do método, utilizamos ela aqui pois sem ela depois de finalizar o método o handler lança o Status Code 200 de que foi feito com sucesso e muitas vezes não é isso que queremos que aconteça.

Orange TALENTS

Como devem ficar:

```java
public class ExceptionResponse {

    private Date timestamp;
    private String message;
    private String details;

    public ExceptionResponse(Date timestamp, String message, String details) {
        this.timestamp = timestamp;
        this.message = message;
        this.details = details;
    }

    public Date getTimestamp() {
        return timestamp;
    }

    public String getMessage() {
        return message;
    }

    public String getDetails() {
        return details;
    }
}
```

```java
public class ErroValidacaoResponse{

    private  String campo;
    private  String erro;

    public ErroValidacaoResponse(String campo, String erro) {
        this.campo = campo;
        this.erro = erro;
    }

    public String getCampo() {
        return campo;
    }

    public String getErro() {
        return erro;
    }
}
```

ZUP

Orange TALENTS

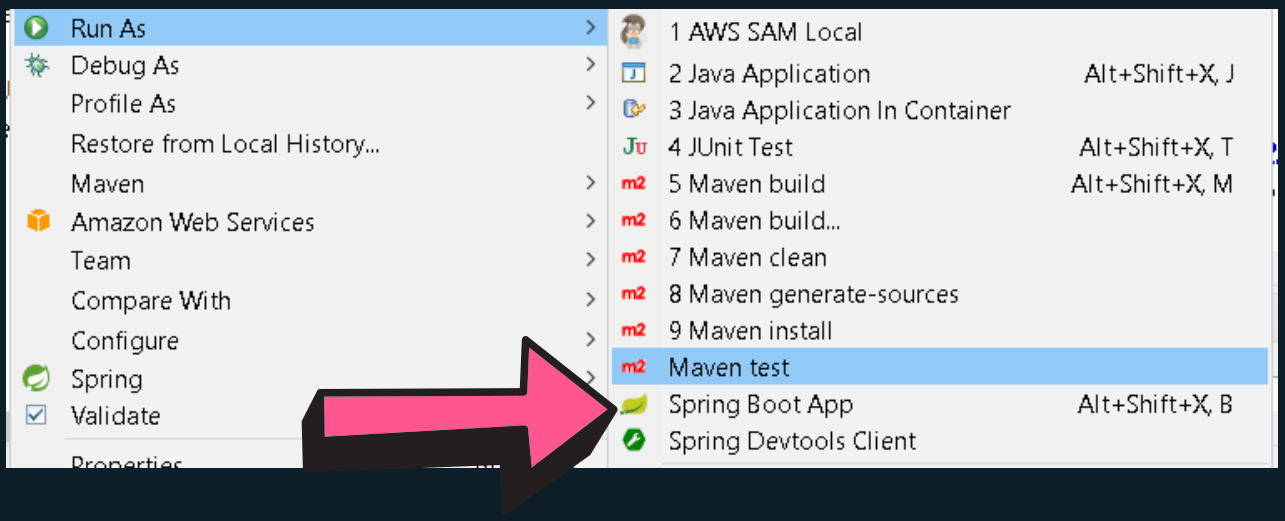
Como devem ficar:

```java
@RestControllerAdvice
public class ExceptionHandlerCustomized {

    @Autowired
    private MessageSource messageSource;

    @ExceptionHandler({DataIntegrityViolationException.class})
    @ResponseStatus(code = HttpStatus.BAD_REQUEST)
    public ErroValidacaoResponse handleAll(DataIntegrityViolationException ex, WebRequest webRequest) {
        String message = ex.getMessage();
        String campo = webRequest.getDescription(false);
        if (ex.getMessage().contains("email_uk")) {
            message = "E-mail já cadastrado!";
            campo = "email";
        }
        if (ex.getMessage().contains("cpf_uk")) {
            message = "CPF já cadastrado!";
            campo = "cpf";
        }
        return new ErroValidacaoResponse(campo,message);
    }



    @ExceptionHandler(MethodArgumentNotValidException.class)
    @ResponseStatus(code = HttpStatus.BAD_REQUEST)
    public List<ErroValidacaoResponse> handle(MethodArgumentNotValidException exception){
        List<ErroValidacaoResponse> dto = new ArrayList<>();
        List<FieldError> fieldErrors = exception.getBindingResult().getFieldErrors();
        fieldErrors.forEach(e -> {
            String mensagem = messageSource.getMessage(e, LocaleContextHolder.getLocale());
            ErroValidacaoResponse erro = new ErroValidacaoResponse(e.getField(), mensagem);
            dto.add(erro);
        });
        return dto;
    }

    @ExceptionHandler({Exception.class})
    @ResponseStatus(code = HttpStatus.INTERNAL_SERVER_ERROR)
    public ExceptionResponse handleAll(Exception ex, WebRequest webRequest){
        return new ExceptionResponse(
                new Date(),
                ex.getMessage(),
                webRequest.getDescription(false));
    }
}
```

ZUP

Orange TALENTS

Agora vamos dar Start no projeto, tá mas como vamos fazer isso? Click com o botão direito do mouse no nome do projeto -> depois Run As -> depois Spring Boot App



PLAY

ZUP

Orange TALENTS

SE TUDO DER CERTO, VAI ACONTECER ALGO PARECIDO COM ISSO:

Console

VaccineAPIZUP - VaccineApizupApplication [Spring Boot App] C:\Users\Jow\Desktop\sts-4.9.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.1.v20201027-0507\jre\bin\javaw.exe (11 de

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.3)

2021-03-11 12:33:02.595  INFO 1368 --- [  restartedMain] b.c.j.V.VaccineApizupApplication         : Starting VaccineApizupApplic
2021-03-11 12:33:02.601  INFO 1368 --- [  restartedMain] b.c.j.V.VaccineApizupApplication         : No active profile set, falli
2021-03-11 12:33:02.700  INFO 1368 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults a
2021-03-11 12:33:02.701  INFO 1368 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related l
2021-03-11 12:33:04.553  INFO 1368 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JP
2021-03-11 12:33:04.726  INFO 1368 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data reposit
2021-03-11 12:33:06.279  INFO 1368 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port
2021-03-11 12:33:06.304  INFO 1368 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-03-11 12:33:06.305  INFO 1368 --- [  restartedMain] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Ap
2021-03-11 12:33:06.546  INFO 1368 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
2021-03-11 12:33:06.547  INFO 1368 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
2021-03-11 12:33:07.058  INFO 1368 --- [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing Persis
2021-03-11 12:33:07.226  INFO 1368 --- [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM cor
2021-03-11 12:33:07.656  INFO 1368 --- [  restartedMain] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commo
2021-03-11 12:33:08.020  INFO 1368 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2021-03-11 12:33:08.429  INFO 1368 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start complet
2021-03-11 12:33:08.485  INFO 1368 --- [  restartedMain] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: or
Hibernate:

    create table usuario (
       id bigint not null auto_increment,
        cpf varchar(255),
        data_nascimento date not null,
        email varchar(255),
        nome varchar(255),
        primary key (id)
    ) engine=InnoDB
Hibernate:

    create table vacinacao (
       id bigint not null auto_increment,
        data_aplicacao date not null,
        nome_vacina varchar(255),
        user_id bigint not null,
        primary key (id)
    ) engine=InnoDB
Hibernate:

    alter table usuario
       drop index email_uk
Hibernate:

    alter table usuario
       add constraint email_uk unique (email)
Hibernate:

    alter table usuario
       drop index cpf_uk
Hibernate:

    alter table usuario
       add constraint cpf_uk unique (cpf)
Hibernate:

    alter table vacinacao
       add constraint FK1vl330gfk2jb98ve6gmexoape
       foreign key (user_id)
       references usuario (id)
2021-03-11 12:33:10.747  INFO 1368 --- [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform
2021-03-11 12:33:10.763  INFO 1368 --- [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManage
2021-03-11 12:33:10.841  INFO 1368 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running
2021-03-11 12:33:12.030  WARN 1368 --- [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is e
2021-03-11 12:33:12.385  INFO 1368 --- [  restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService
2021-03-11 12:33:12.962  INFO 1368 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8
2021-03-11 12:33:12.999  INFO 1368 --- [  restartedMain] b.c.j.V.VaccineApizupApplication         : Started VaccineApizupApplica
```


ZUP

Orange TALENTS

Calma... tá acabando...

Agora vamos testar nosso projeto? lembra lá do começo que falamos do Postman? Então agora vamos lá... Aqui devemos usar para inserir as informações na formatação Json, primeiro iremos cadastrar um usuário e ver se foi persistido (gravado) no banco de dados, depois vamos cadastrar a data de vacinação, vinculada ao e-mail. Já vamos aproveitar para testar nossas exceções... vejamos abaixo:

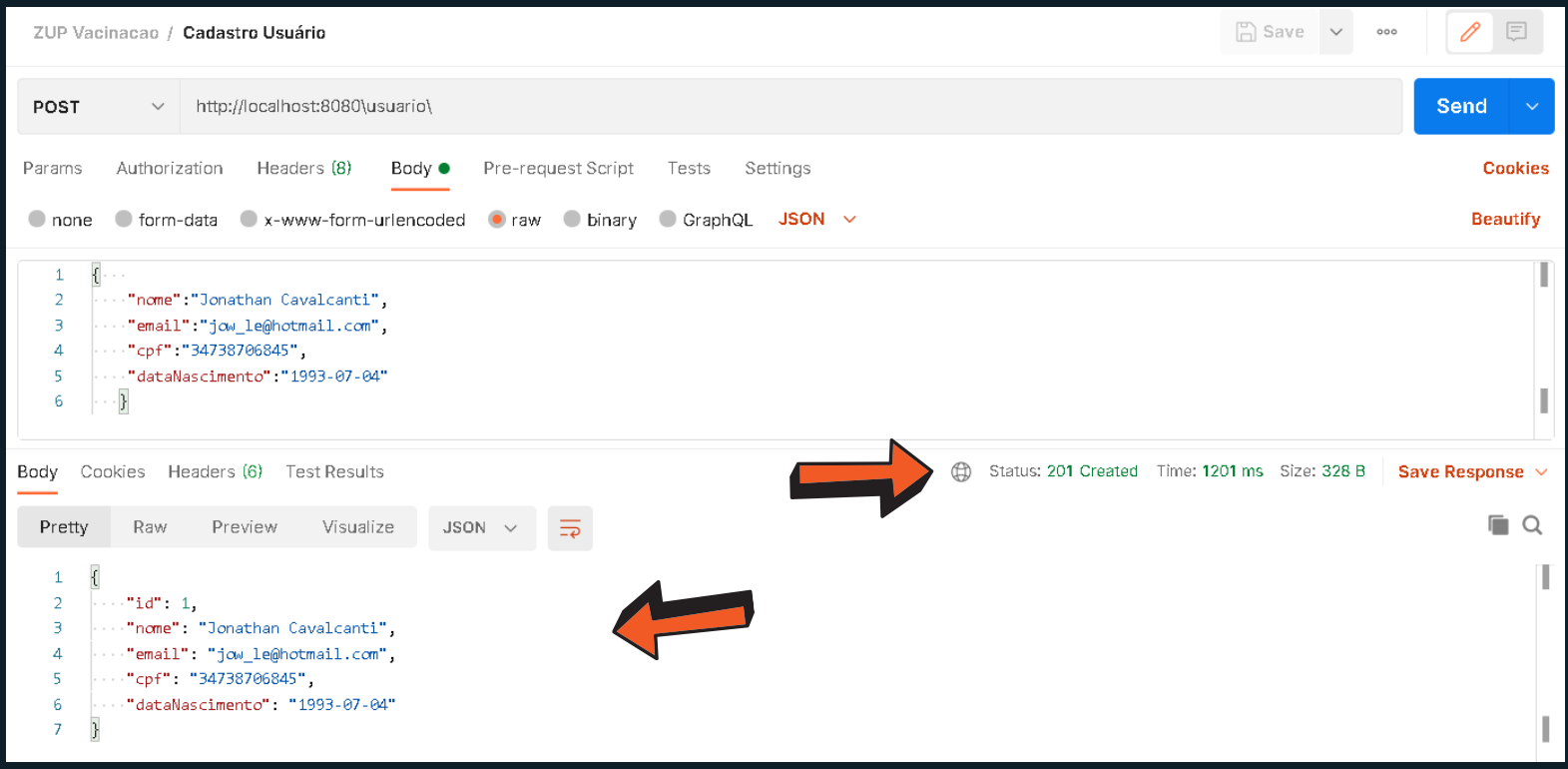
O caminho que vamos usar são:

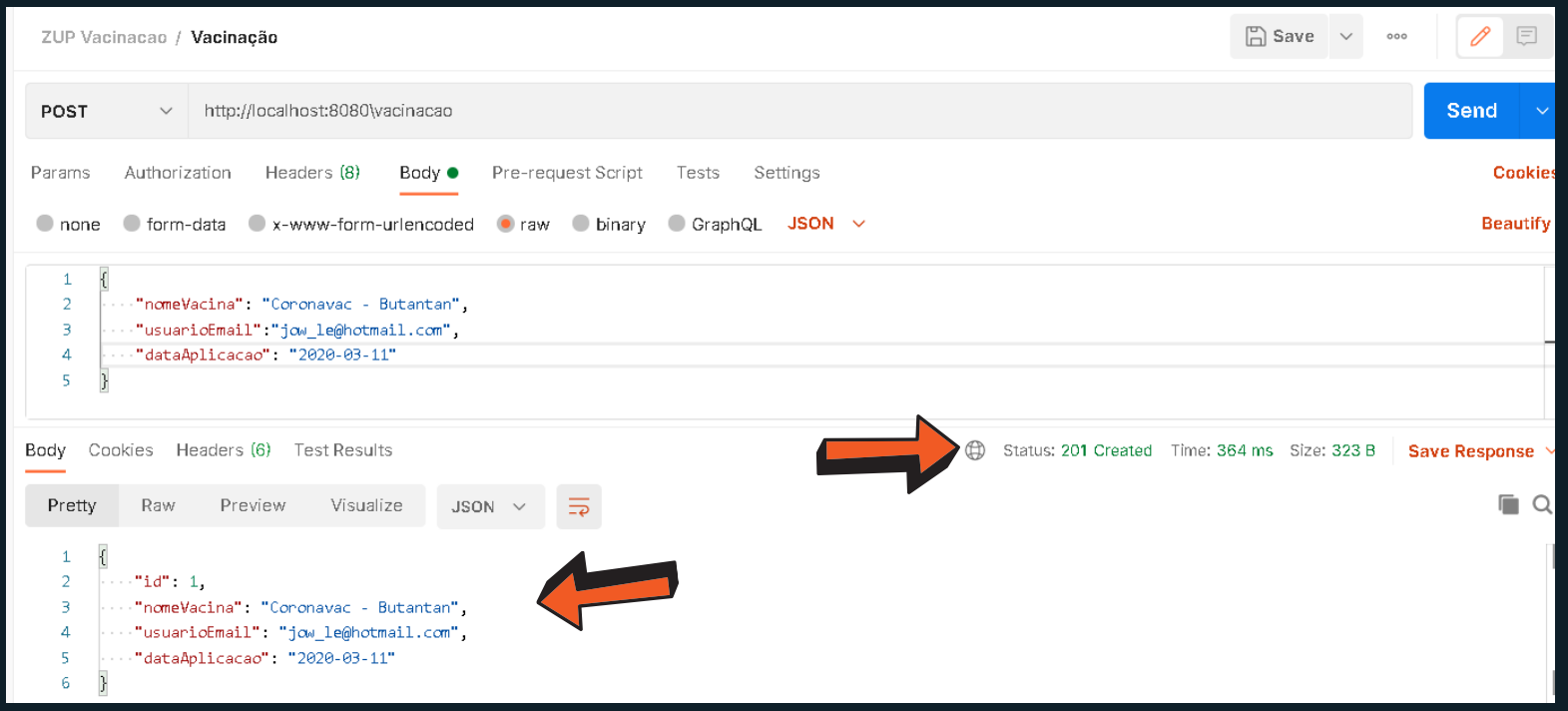
- http://localhost:8080\usuario\
- http://localhost:8080\vacinacao

ZUP

Orange TALENTS

Como devem ficar:

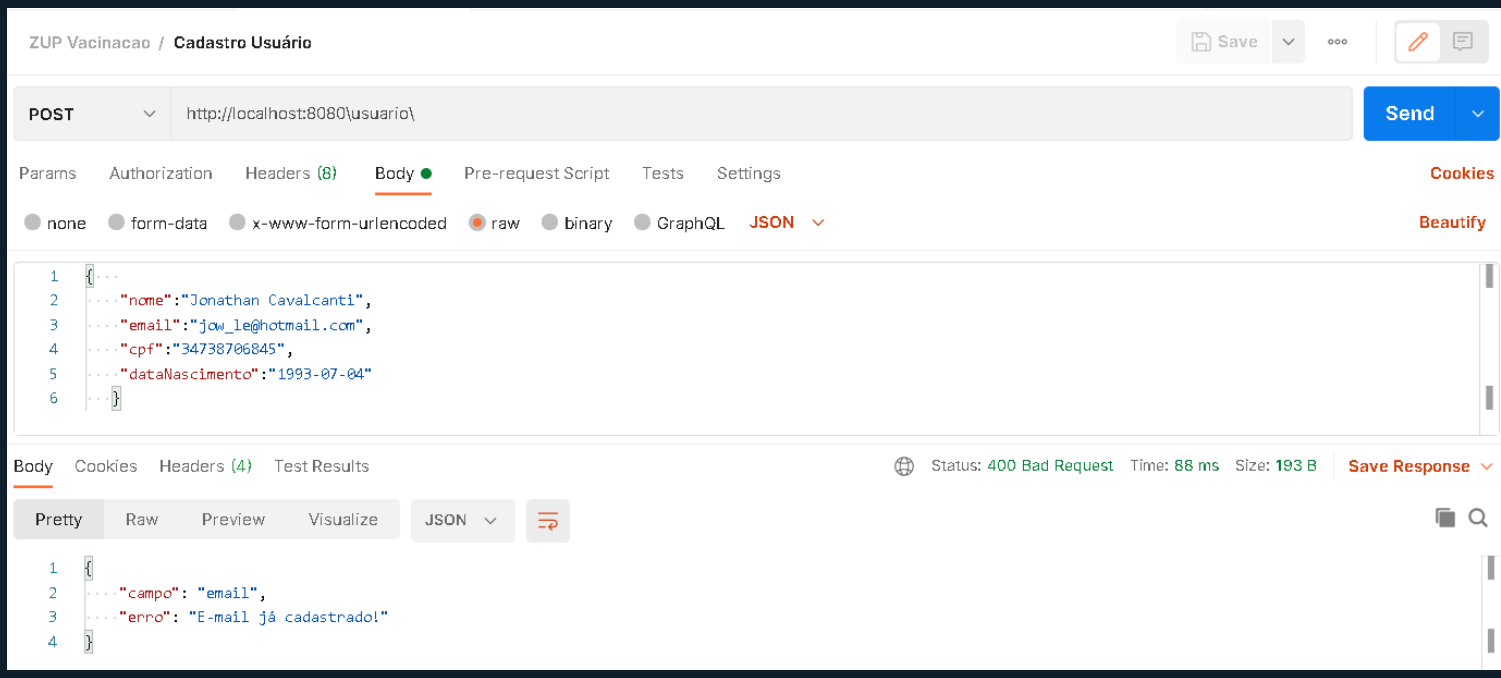


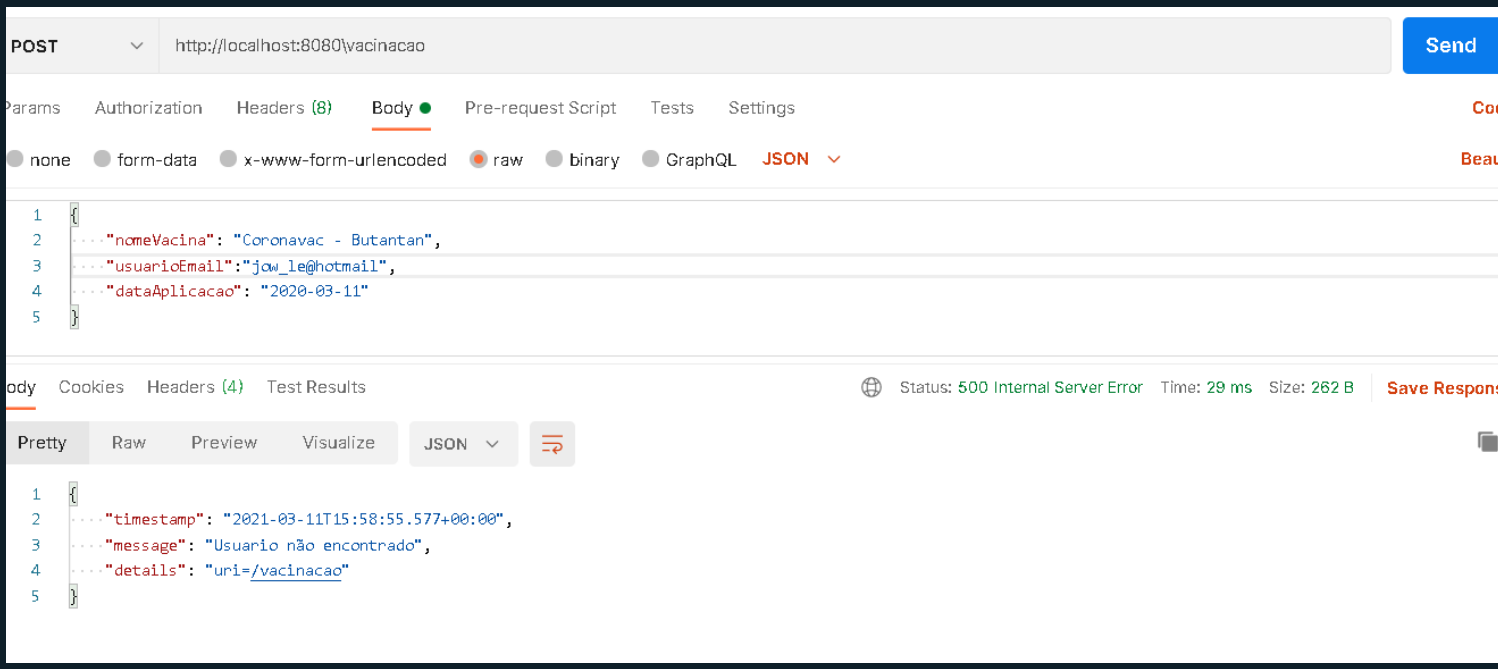


ZUP

Orange TALENTS

Como devem ficar:





ZUP

Orange TALENTS

API REST CONTROLE PARA VACINAÇÃO

THE END

ZUPER RESPONSÁVEL: JONATHAN C. DE PAULA

ZUP