**Formador:** ROGER STEVENS HERNANDEZ ARIZA

**Grupo**: CLOUDY RAINBOW **Sprint No** 3

**Nombre del Proyecto: Doctor Patitas**

**Integrantes**:

* Juan Felipe Ortiz Franco
* Lina Marcela Triana Bolaños
* Emerson Alexis Vargas Buenaventura
* Giovanni Ernesto Ibañez Acevedo
* Iván Arturo Ávila Gómez

**Tipo de Actividad:** Producto

1. Visualice a través de imágenes tomadas a su pantalla la construcción de su BD en mongoDB (Colecciones y Documentos).

* Crear base de datos

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente con confianza media

* Crear colección

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* Crear Colección en BD existente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

* Crear o adjuntar Json

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

*NOTA: Importante presentar mínimo 6 documentos por colección de datos.*

2. Visualice la construcción de sus controladores a través de imágenes tomadas de su computadora a utilizando SpringFox-Swagger. Y tome captura de cada proceso realizado en ello.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

crear base de datos en netbeans

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

para este proceso es necesario hacer los siguientes pasos:

* abrir NetBeans:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*En el netbeans\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Crear un nuevo proyecto

java with maven -> sprintboot inizialzr project:

{

(Group: com.patitas,

artifact: crud)

}

-> Seleccionar las dependencias:

{

(Spring web,

Spring Data MongoDB)

}

-> Dar un nombre a la carpeta en donde

se va a guardar el proyecto.

{

(nombre = patitas\_crud)

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*patitas\_crud (proyecto en netbeans):\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Se crea el modelo del proyecto

carpetas:

Source Packages:

-> com.patitas.crud:

-> New - Java Package

--Package Name: com.patitas.crud.modelo

-> New - Java Class -

--Class Name: (usuariosModelo) veterinariaModelo:

package com.patitas.crud.Modelo;

import org.springframework.data.mongodb.core.mapping

.Document;

import org.springframework.data.annotation.Id;

author;

@Document(collection="usuarios")

public class usuariosModelo{

/\*

El comando @Document me va permitir crear una colección

en mongodb a través de la conexión con la base de

datos y del proyecto del crud

A continuación, se crean los campos de la colección

\*/

----@Id

(con esto se define a la llave primaria, en

este caso el id, después se añade el import

y si aparece algún error se soluciona.)

----private String id;

private String nombre;

private String apellidos;

private String telefono;

private String direccion;

/\*

se necesita transformar los campos en objetos para

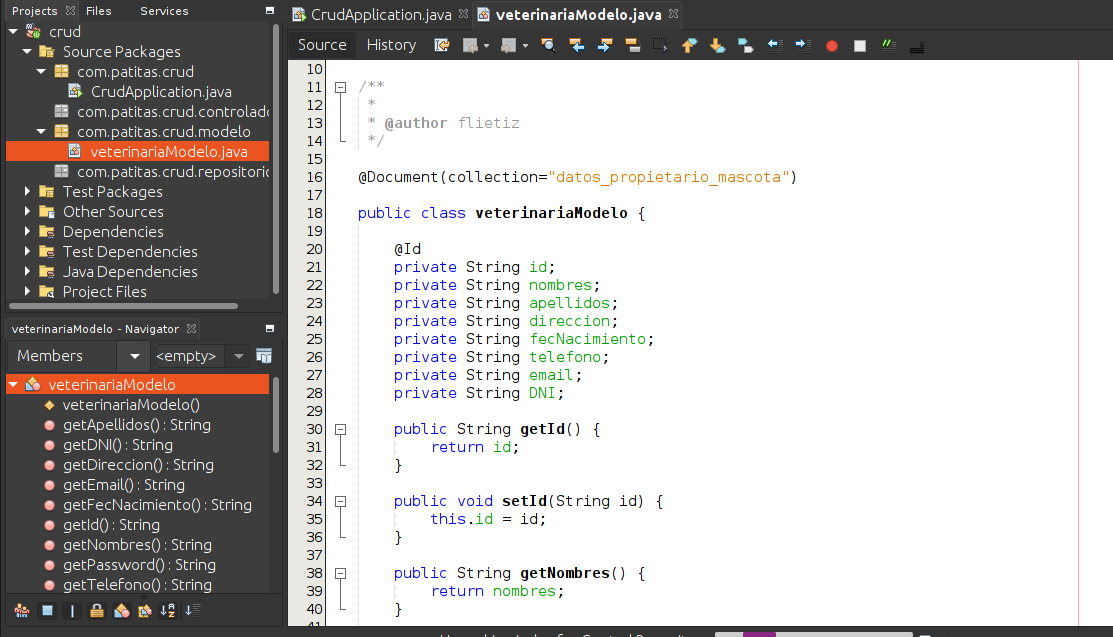
ello se agregan sus procedimientos getters and setters

-> insert code - getters and setters - all data

structure - generate.

\*/

----}



=======================================================

-> New - Java Package -

=======================================================

* Se crea el paquete Vista

--Package Name: com.patitas.crud.repositorio:

-> New - Java Interface -

-- Class Name: (usuariosRepositorio):

package com.patitas.crud.repositorio;

import com.patitas.crud.modelo.usuariosModelo;

import org.springframework.data.mongodb,repository.

MongoRepository;

import org.springframework.stereotype.Repository;

/\*

se importan usuariosModelo, MongoRepository

cuando se ha definido el repositorio que es el que

se está conectando (usuariosModelo) pide el tipo de

dato que corresponde a la llave primaria, en este

caso la ID

\*/

@Repository

public interface usuariosRepositorio extends

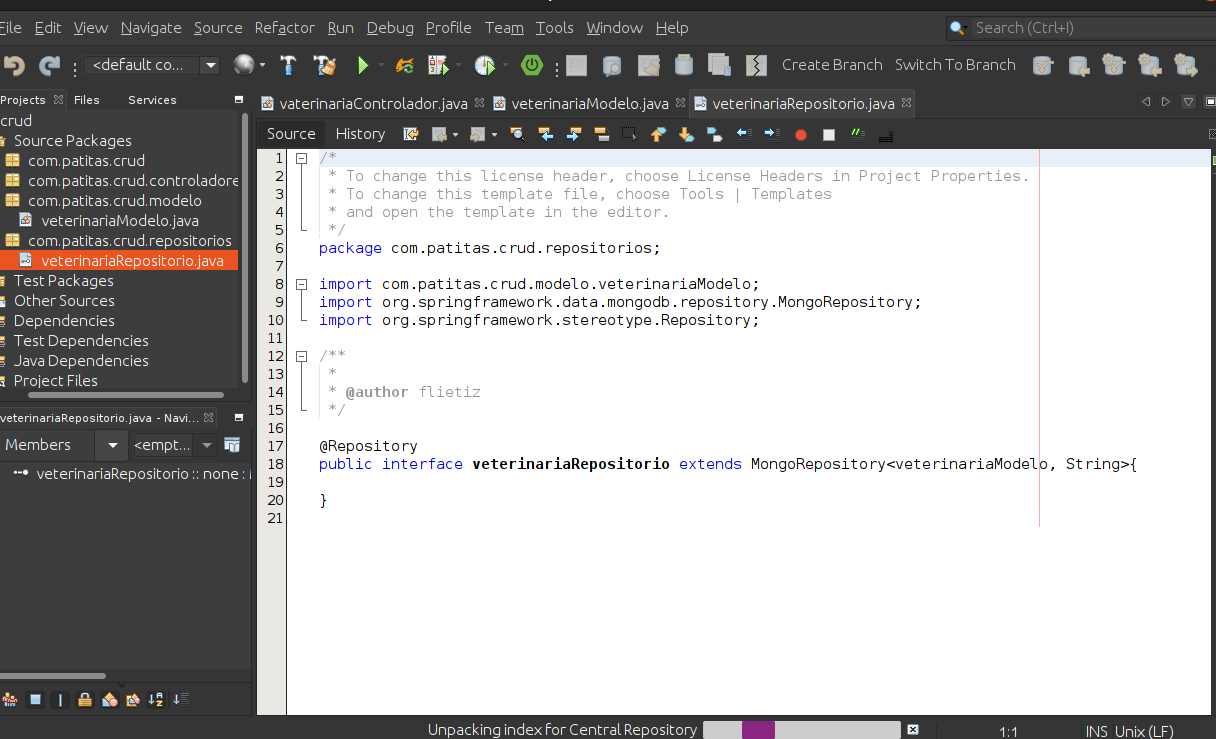
MongoRepository<usuariosModelo, String>{

----}

/\*

después de crear la conexión se crean los controladores

\*/



=======================================================

-> New - Java Package -

=======================================================

* Se crea el paquete controlador

--Package Name: com.patitas.crud.controlador

-> New - Java Class -

--Class Name: (usuariosControlador):

package com.patitas.crud.controlador;

import com.patitas.crud.modelo.usuariosModelo;

import com.patitas.crud.repositorio

.usuariosRepositorio;

import java.util.List;

import org.springframework.beans.factory.annotation

.Autowired;

import org.springframework.validation.annotation

.Validated;

import org.springframework.web.bind.annotation

.CrossOrigin;

import org.springframework.web.bind.annotation

.DeleteMapping;

import org.springframework.web.bind.annotation

.GetMapping;

import org.springframework.web.bind.annotation

.PathVariable;

import org.springframework.web.bind.annotation

.PostMapping,

import org.springframework.web.bind.annotation

.PutMapping;

import org.springframework.web.bind.annotation

.RequestBody;

import org.springframework.web.bind.annotation

.RequestMapping;

import org.springframework.web.bind.annotation

.RequestMethod;

import org.springframework.web.bind.annotation

.RestController;

/\*Author\*/

/\*

Agregamos las directivas

\*/

@RestController

@CrossOrigin(origins = "\*", methods = {RequestMethod

.POST,RequestMethod.GET,RequestMethod.DELETE,

RequestMethod.PUT})

/\*

Por último se define cual será la dirección relativa

del acceso al API (uno la define).(requestMapping(

"/api/usuarios"))

y luego definir cuál va a ser su acceso al servicio

principal del controlador

el (@Autowired) ayuda a forzar los procedimientos de

guardado y actualización dentro de la arquitectura

springframework

\*/

@RequestMapping("/api/usuarios")

public class usuariosControlador{

----//////variable de insertar con el modelo

----@Autowired

----private ususariosRepositorio usu;

----//////procedimiento guardar

----@PostMapping("/guardar")

----public usuariosModelo guardarusuarios(@Validated

@RequestBody usuariosModelo varU){

----

----return usu.insert(varU);

----

--------}

/\*

se necesita llamar al modelo para que este haga los

procedimientos almecenados a través de la variable

que se creó de tipo interfaz

\*/

----////////Procedimiento consulta general

----@GetMapping("/consultar")

----public List<usuariosModelo> consultarUsuarios(){

----return usu.findAll();

----}

----///////procedimiento actualizar información

----@PutMapping("/actualizar/{id}")

----public usuariosModelo actualizarUsuarios (

----@Pathvariable String id,@Validated @RequestBody

----usuariosModelo varU){

----

----return usu.save(varU);

----}

----//////////Procedimiento eliminar usuario

----@DeleteMappint("/eliminar{id}")

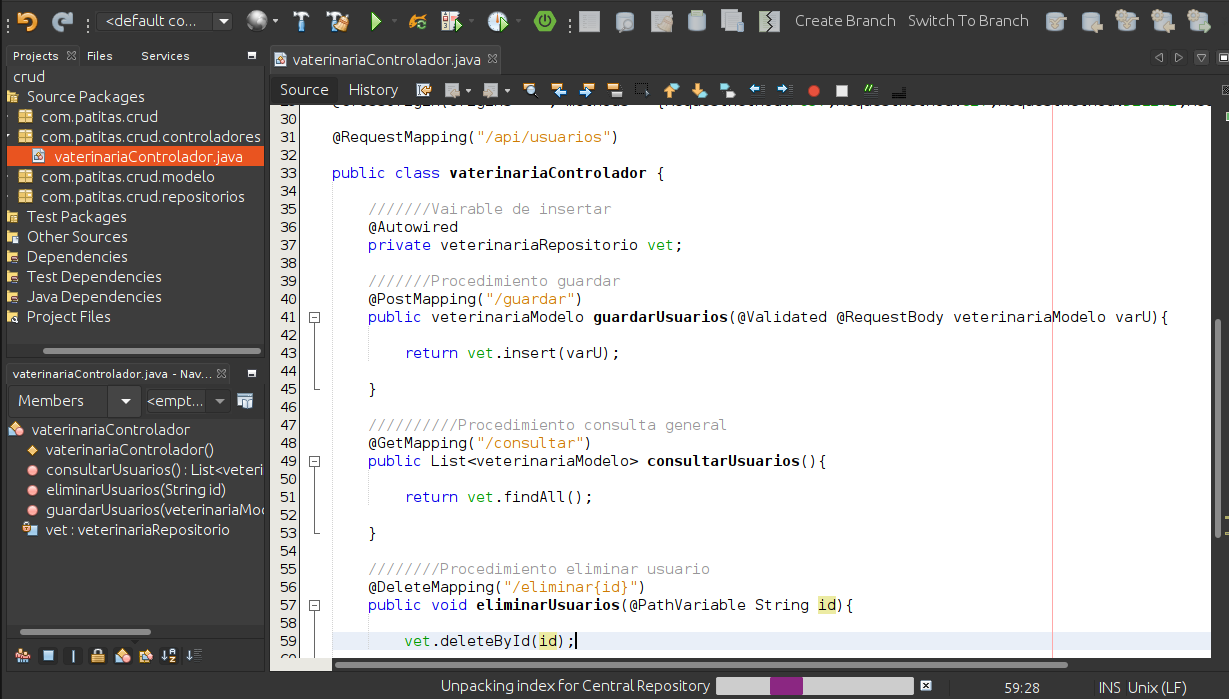
----public void eliminarUsuarios(@PathVariable String

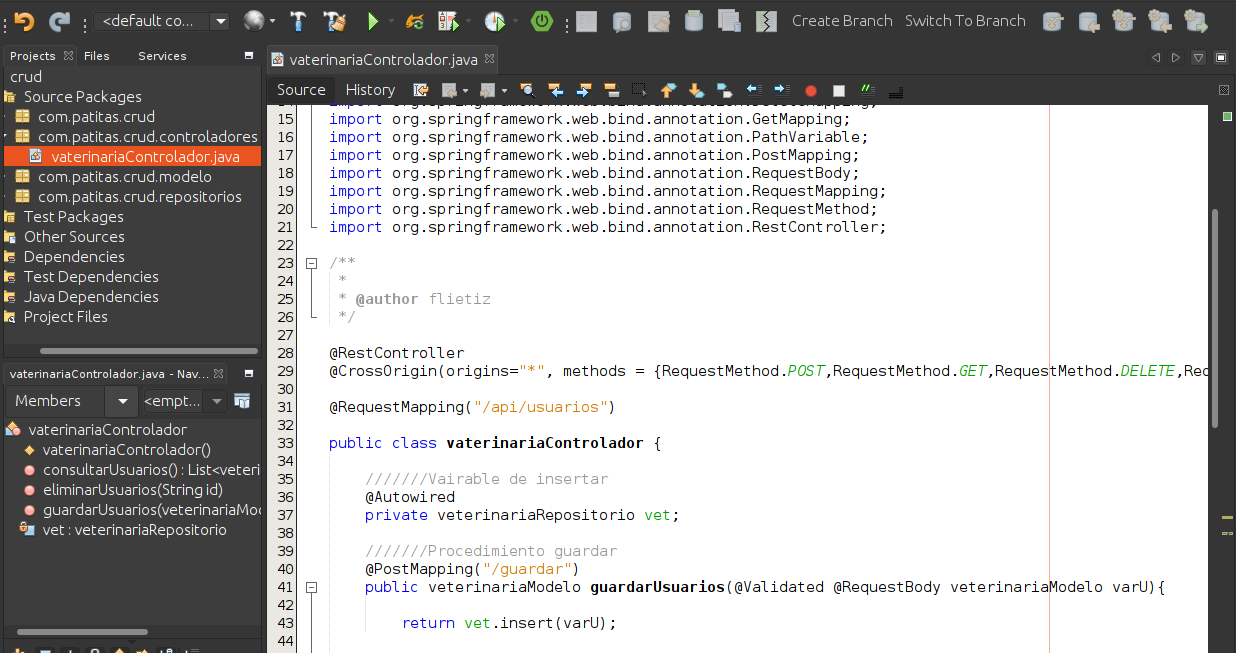
----id){

--------usu.deleteById(id);

--------

----}





* Se agrega la interfaz de usuario Springfox Swagger

-> CrudApplication.java:

(desde aquí se ejecuta el proyecto)

||||||||||||||||||||||||||||||||||||

(mirar en que puerto se está ejecuta

el proyecto---en el caso del profe

en el puerto 8080).

||||||||||||||||||||||||||||||||||||

(paso 1 de MVC:)

--import springfox.documentation.swagger2.annotations

.EnableSwagger2;

--@SpringBootApplication

--@EnableSwagger2 - (Ctrl + shift + I)

(paso 2 de MVC:)

--public class CrudApplication {

--@Bean

--public Docket petApi(){

------return new Docket(DocumentationType.SWAGGER\_2)

----------.select()

----------.apis(RequestHandlerSelectors.any())

----------.paths(pathSelectors.any())

----------.build();

/\*

(paso3 del MVC:)

(como las dependencias dependen del docket, hay que

finalizarlas con ;) - (explica paso 5)

\*/

------}

/\*

(la llave hay que agregarla para cerrar la función)

seguido hay que agregar las depencencias que se indican

importar en bean, en docket, DocumentationType,

RequestHandlesSelectors, PathSelectors.

Después se le cambia el nombre a la api.

--public Docket usuariosApi(){

Ahora se verifica la interfaz de Swagger

\*/

(paso 4 de MVC:)

------public static void main(String [] args){

----------SpringApplication.run(Swagger2SpringBoot

.class,args);

------}

--}

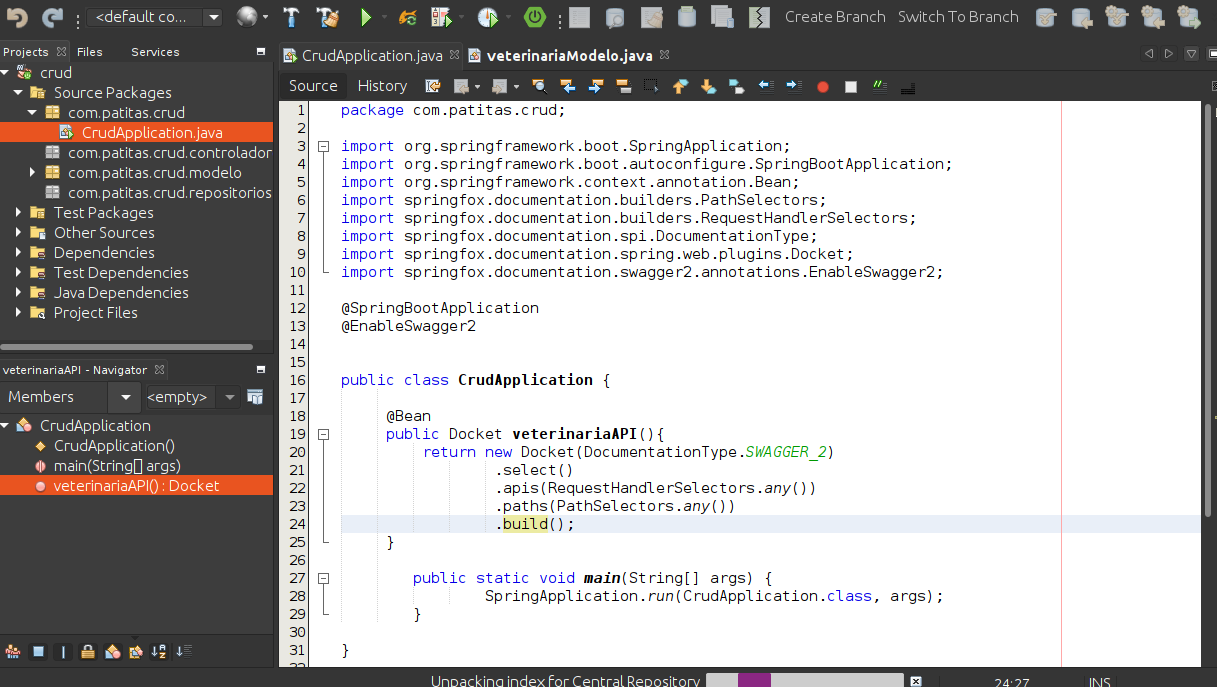
Test Packages;

Other Sources:

-> scr/main/resources:

-> <default package>

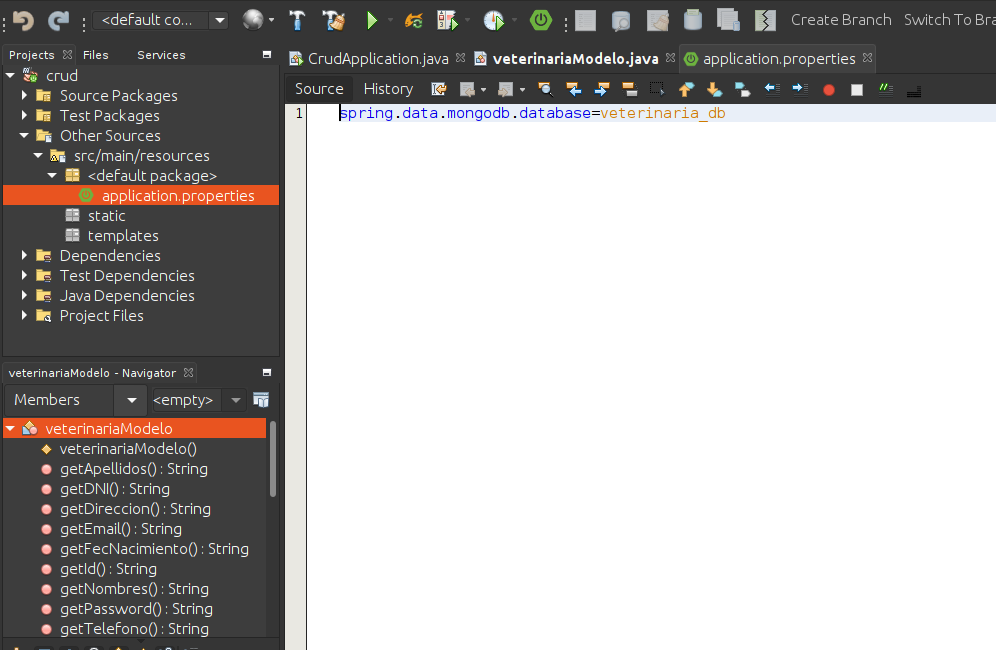
-> application.properties:



* Se hace la conexión con el MongoDB

====================================================

spring.data.mongodb.database=veterinaria\_db



* Se hace la instalación de las dependencias

Dependencies:

-> +springfox-starter-data-mongodb-2.5.6.jar

+spring-boot-starter-web-2.5.6.jar

+springfox-swagger-ui-2.9.2.jar

+springfox-swagger2-2.9.2.jar

Test Dependencies;

Java Dependencies;

Project Files:

-> pom.xml:

-> <dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

<dependency>

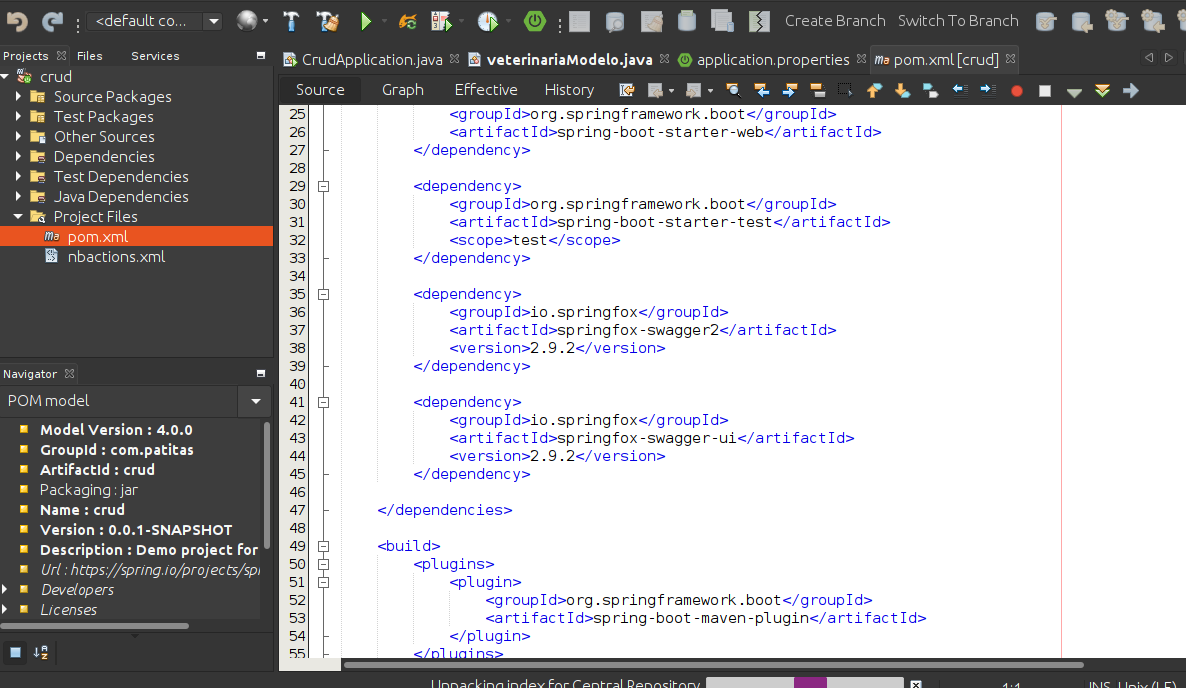
<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.9.2</version>

</dependency>

* Esta versión es la más estable



* Comprobamos que se ejecute correctamente

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*En el navegador\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1>>>>>:

-> localhost:(número de servidor en el que

se está ejecutando el servidor)

Si en el navegador aparece (Whitelabel Error page)

significa que se está ejecutando pero está vacía

no tiene contenido.

2>>>>>:

-> busca: springfox swagger

(https://springfox.github.io): Springfox Reference

Documentation -

GitHub pages

3>>>>>:

https://springfox.github.io:

-> 2.1.2: Maven ( Vamos a utilizar Springfox Swagger

para crear unproyecto maven )

-> se copia la dependencia en (pom.xml):

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

-> se pega en el archivo pom.xml:

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.9.2</version>

</dependency>

4>>>>>:

se revisa en la carpeta que estén bien instalados

las dependencias:

(si salen errores, se le da a resolver

problemas y en todo caso se lee a cerca de la

solución).

[5]>>>>>:

(luego de haber agregado las dependencias

lo que sigue es poder transformar la

estructura del pyecto a una arquitectura

de software titulada MVC -modelo -vista

-controlador).

-> 3.1. Springfox Spring MVC and Spring Book:

-> @EnableSwagger2:

(el primer paso es

habilitar el springfox swagger en el proyecto)

-> (CrudApplication.java)

(paso 2 de MVC:

copiar el código @Bean)

-> (CrudApplication.java)

(Explica paso [5]: basta con utilizarlo hasta

aquí donde yo puedo seleccionar la api, puedo

como tal direccionarla y puedo compilarla)

(paso 3 del MVC: Corregir errores)

(paso 4 del MVC: verificar la interfaz de

Swagger):

(localhost:8080/swagger-ui.html): Api Documentation

ya la interfaz está creada - conexión con el servidor

springboot y springfox swagger - conexión con "mongo"

(ahora se crean los servicios los cuales permiten

consumir la API) se necesita crear los servicios que

permiten interactuar con la base de datos que ya está

definida

--Application.properties=

---- ->(spring.data.mongodb.database=veterinaria\_db)

--------> veterinaria\_db.

se crea el modelo MVC a través de mi proyecto

crud.

-->(Source Packages):

---->(com.patitas.crud):

------> (crear los modelos)

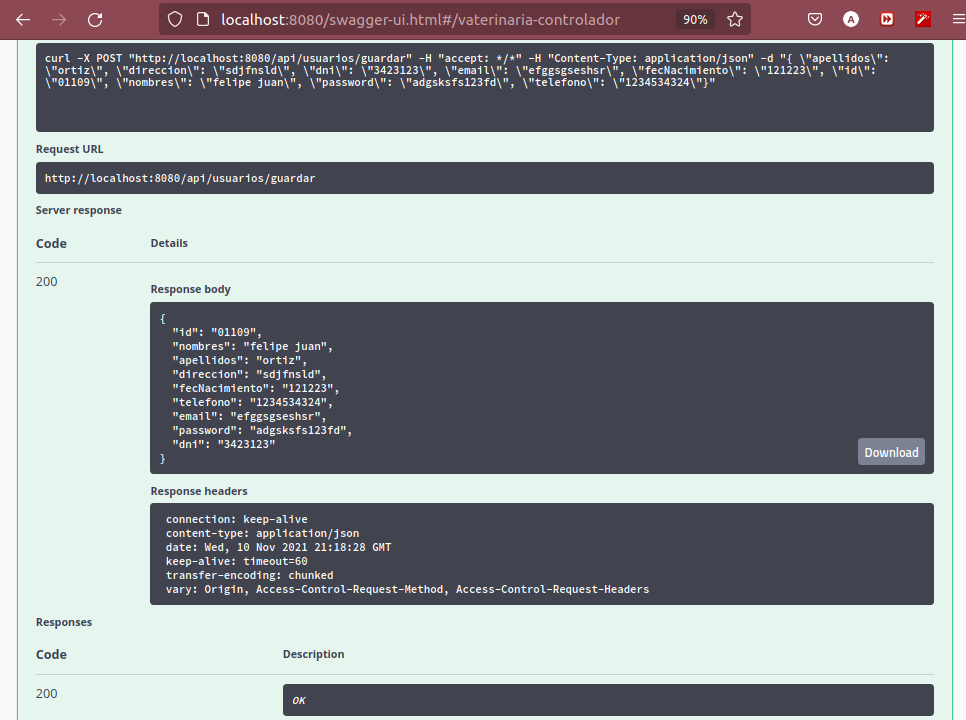
------> (crear los repositorios)

------> (crear controladores)

/\*

la respuesta del navegadro al momento de dar execute

tiene que ser code:200.

\*/