

**Nombre Completo: Univ. Ricardo David La Madrid Alarcón**

**Asignatura: PROGRAMACIÓN III**

**Carrera: INGENIERÍA DE SISTEMAS**

**Paralelo:**  **PROG (1)**

**Docente: Lic. William R. Barra Paredes**

**fecha: 11/05/2020**

**Tarea de fin de Hito 3 Manejo de Spring Framework (Rest API)**

**Parte Teórica.**

1. Preguntas.

Responda de manera breve, clara y concisa posible.

* **Defina y  muestre ejemplo de un servicio REST.**

Es un conjunto de principios de arquitectura para describir cualquier interfaz

entre sistemas que utilice HTTP para obtener datos o indicar la ejecución de

operaciones sobre los datos, en cualquier formato.

* **Que es JPA y como configurar en un entorno Spring.**

JPA es una abstracción de JDBC que permite comunicar ambos modelos de manera sencilla, realizando la conversión entre objetos y tablas.

Esta conversión se llama ORM (Mapeo Relacional de Objetos) y puede configurarse mediante un xml o anotaciones. Toda esta relación es transparente el desarrollador y para ello se deben crear los objetos de una manera singular, a éstos se les llama entidades.

* **Que es MAVEN - POM.**

**MAVEN**

Es una herramienta para la gestión y construcción de proyectos java, que se basa en

el concepto Proyect Object Model. Con Maven se pueden generar arquetipos,

gestionar librerías, compilar, empaquetar, generar documentación.

**POM**

Es un fichero de configuración

etiquetas básicas, entre las que se puede destacar:

&lt;groupId &gt;: Suele ser el nombre o la web de la organización. Aunque no es

necesario que tenga puntos de separación, se recomienda para actúe como paquete

de Java.

&lt;artifactId&gt;: El nombre del artefacto.

&lt;name&gt;: El nombre del proyecto.

&lt;versión&gt;: La versión de del proyecto. Por defecto se suele usar “1.0.0”. Aunque es

posible usar la metodología de versiones que más nos guste.

&lt;packaging&gt;: Con ella se indica cómo se desea que sea empaquetado el proyecto

cuando Maven lo construya.

* **Qué son los Spring estereotipos y anotaciones muestre ejemplos.**

**Estereotipos**

**@Component:** es el estereotipo principal, indica que la clase anotada es un componente o

bean de Spring.

**@Repository:** Es el estereotipo que tiene como función dar de alta un bean para que

implemente el patrón repositorio, que es el encargado de almacenar datos en una base de

datos o repositorio de información que se necesite.

**@Service:** Este estereotipo se encarga de gestionar las operaciones de negocio más

importantes a nivel de la aplicación, Su tarea fundamental es la de agregador.

**@Controller:** Es el que realiza las tareas de controlador y gestión de la comunicación entre el

usuario y el aplicativo.

**Anotaciones**

Una de las anotaciones más importantes de Spring es **@Autowired,** con ella es posible inyectarun componente como puede ser un servicio o un bean.

* **Describa las características principales de REST.**

**Escalabilidad**. Gracias al protocolo HTTP, pueden interactuar con cualquier servidor

HTTP sin ninguna configuración especial.

**Independencia**. HTTP permite la extensibilidad mediante el uso de las cabeceras, a

través de las URIs. 7

**Compatibilidad**. En ocasiones existen componentes intermedios que dificultan la

comunicación entre sistemas, como pueden ser los firewalls. Es importante señalar

que los servicios web se pueden utilizar sobre cualquier protocolo, sin embargo, TCP

es el más común.

**Identificación de recursos.** REST utiliza una sintaxis universal como es el uso de

URIs. HTTP es un protocolo centrado en URIs, donde los recursos son los objetos

lógicos a los que se le envían mensajes.

**Protocolo cliente/servidor sin estado**. Cada mensaje HTTP contiene toda la

información necesaria para comprender la petición. Como resultado, ni el cliente ni el

servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin

embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y

otros mecanismos para mantener el estado de la sesión.

**Operaciones bien definidas**. HTTP en sí define un conjunto pequeño de

operaciones, las más importantes son POST, GET, PUT y DELETE.

# Parte Práctica.

Debe de trabajarlos en un nuevo proyecto mismo que debe estar en la carpeta de github del hito actual.

* 1. Crear los PACKAGEs necesarios(debe reflejarse el modelo MVC).
  2. Debe de crear los servicios REST para el siguiente escenario.

1-Actualmente toda la humanidad está pasando por una pandemia conocida como Corona Virus COVID19. En bolivia se pretende crear una plataforma en tiempo real para mostrar estos datos a cada habitante.

Es decir mostrar casos contagiados, casos sospechosos, casos recuperados, etc.

2-Para este propósito la primera fase de de desarrollo es la creación de un servicio rest que pueda crear, modificar y retornar estos datos.

3-Se tiene como base principal la siguiente tabla que nos servirá para poder generar toda esta información.



## Bases de Datos

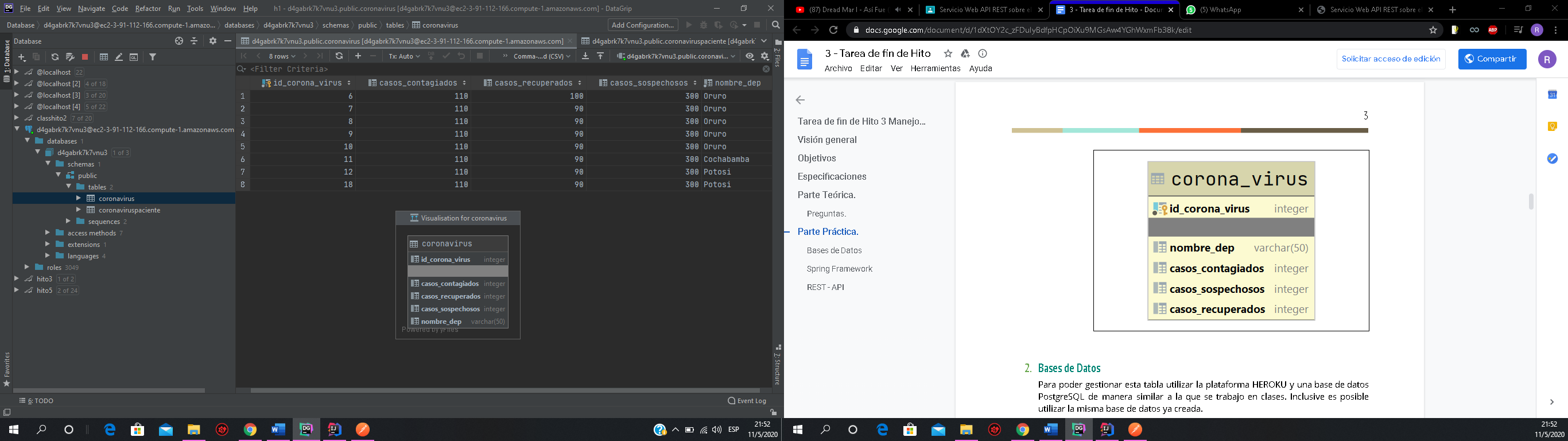
Para poder gestionar esta tabla utilizar la plataforma HEROKU y una base de datos PostgreSQL de manera similar a la que se trabajo en clases. Inclusive es posible utilizar la misma base de datos ya creada.

## Spring Framework

Es necesario crear un modelo MVC para poder solución a este problema, deberá de crear MODELS, SERVICES, REPOS y CONTROLLERS.

## REST - API

Generar los siguientes servicios.



**POST**

**CODIGO**

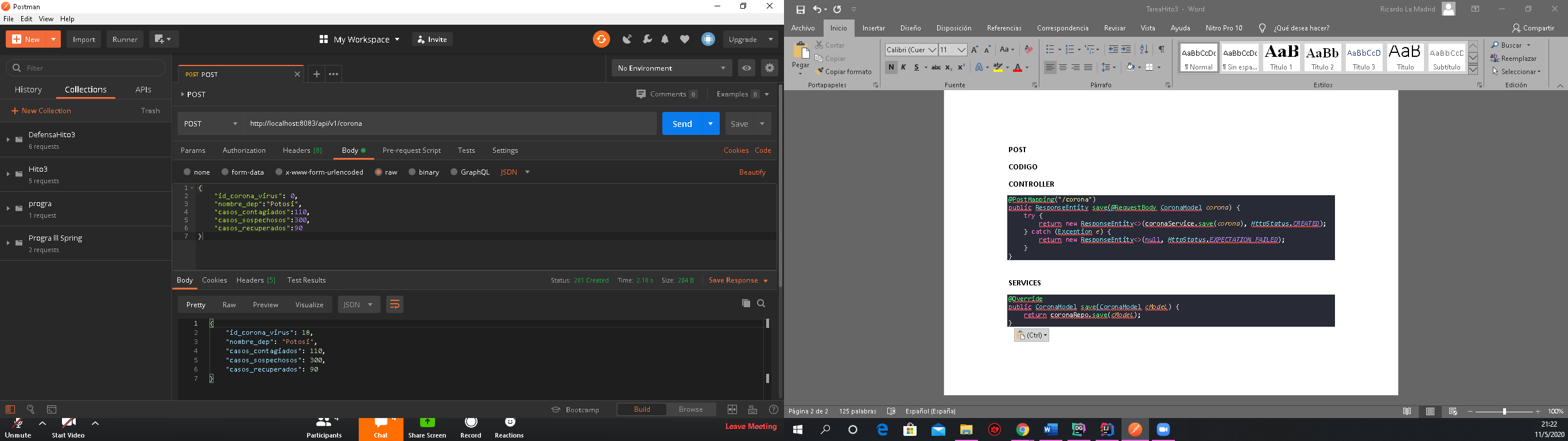
**CONTROLLER**

@PostMapping("/corona")  
public ResponseEntity save(@RequestBody CoronaModel *corona*) {  
 try {  
 return new ResponseEntity<>(coronaService.save(*corona*), *HttpStatus*.*CREATED*);  
 } catch (Exception *e*) {  
 return new ResponseEntity<>(null, *HttpStatus*.*EXPECTATION\_FAILED*);  
 }  
}

**SERVICES**

@Override  
public CoronaModel save(CoronaModel *cModel*) {  
 return coronaRepo.save(*cModel*);  
}

**CAPTURA**



**PUT**

**CODIGO**

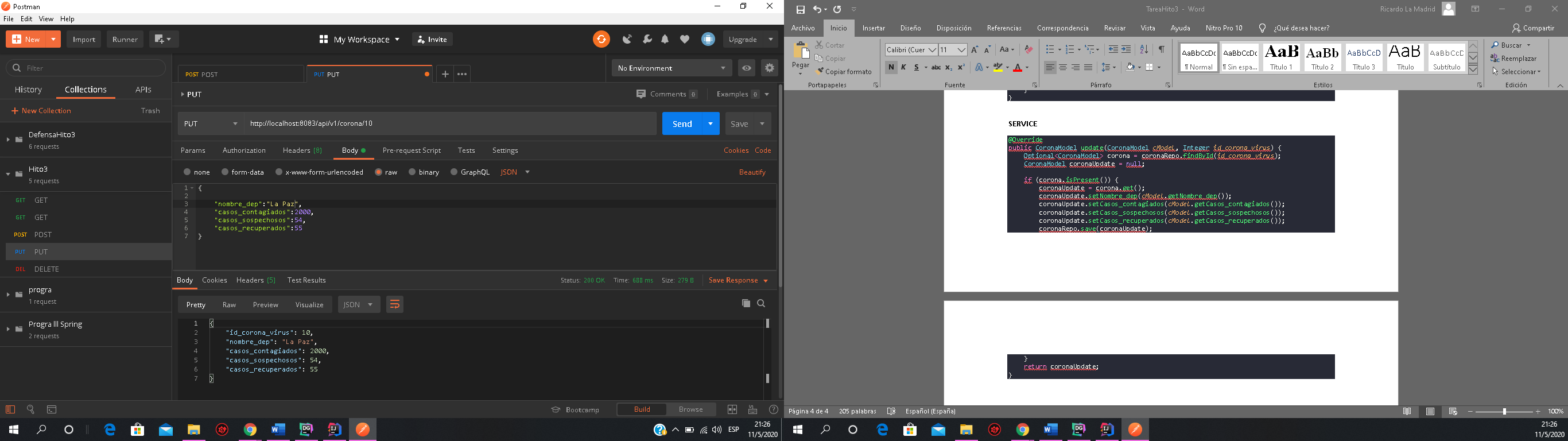
**CONTROLLER**

@PutMapping("/corona/{id\_corona\_virus}")  
public ResponseEntity<CoronaModel> updateMaterias(@PathVariable("id\_corona\_virus") Integer *id\_corona\_virus*, @RequestBody CoronaModel *cModel*) {  
 try {  
 CoronaModel pUpdate = coronaService.update(*cModel*, *id\_corona\_virus*);  
 if (pUpdate != null) {  
 return new ResponseEntity<>(pUpdate, *HttpStatus*.*OK*);  
 } else {  
 return new ResponseEntity<>(*HttpStatus*.*NOT\_FOUND*);  
 }  
 } catch (Exception *e*) {  
 return new ResponseEntity<>(null, *HttpStatus*.*INTERNAL\_SERVER\_ERROR*);  
 }  
}

**SERVICE**

@Override  
public CoronaModel update(CoronaModel *cModel*, Integer *id\_corona\_virus*) {  
 Optional<CoronaModel> corona = coronaRepo.findById(*id\_corona\_virus*);  
 CoronaModel coronaUpdate = null;  
  
 if (corona.isPresent()) {  
 coronaUpdate = corona.get();  
 coronaUpdate.setNombre\_dep(*cModel*.getNombre\_dep());  
 coronaUpdate.setCasos\_contagiados(*cModel*.getCasos\_contagiados());  
 coronaUpdate.setCasos\_sospechosos(*cModel*.getCasos\_sospechosos());  
 coronaUpdate.setCasos\_recuperados(*cModel*.getCasos\_recuperados());  
 coronaRepo.save(coronaUpdate);  
 }  
 return coronaUpdate;  
}

**CAPTURA**



**DELETE**

**CODIGO**

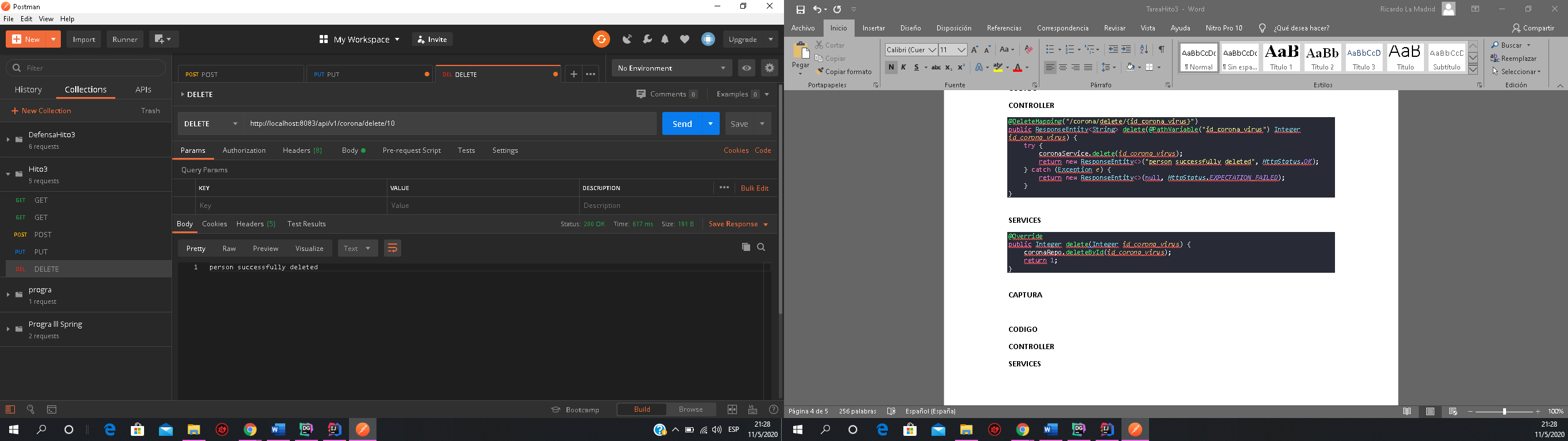
**CONTROLLER**

@DeleteMapping("/corona/delete/{id\_corona\_virus}")  
public ResponseEntity<String> delete(@PathVariable("id\_corona\_virus") Integer *id\_corona\_virus*) {  
 try {  
 coronaService.delete(*id\_corona\_virus*);  
 return new ResponseEntity<>("person successfully deleted", *HttpStatus*.*OK*);  
 } catch (Exception *e*) {  
 return new ResponseEntity<>(null, *HttpStatus*.*EXPECTATION\_FAILED*);  
 }  
}

**SERVICES**

@Override  
public Integer delete(Integer *id\_corona\_virus*) {  
 coronaRepo.deleteById(*id\_corona\_virus*);  
 return 1;  
}

**CAPTURA**



**GET ID**

**CODIGO**

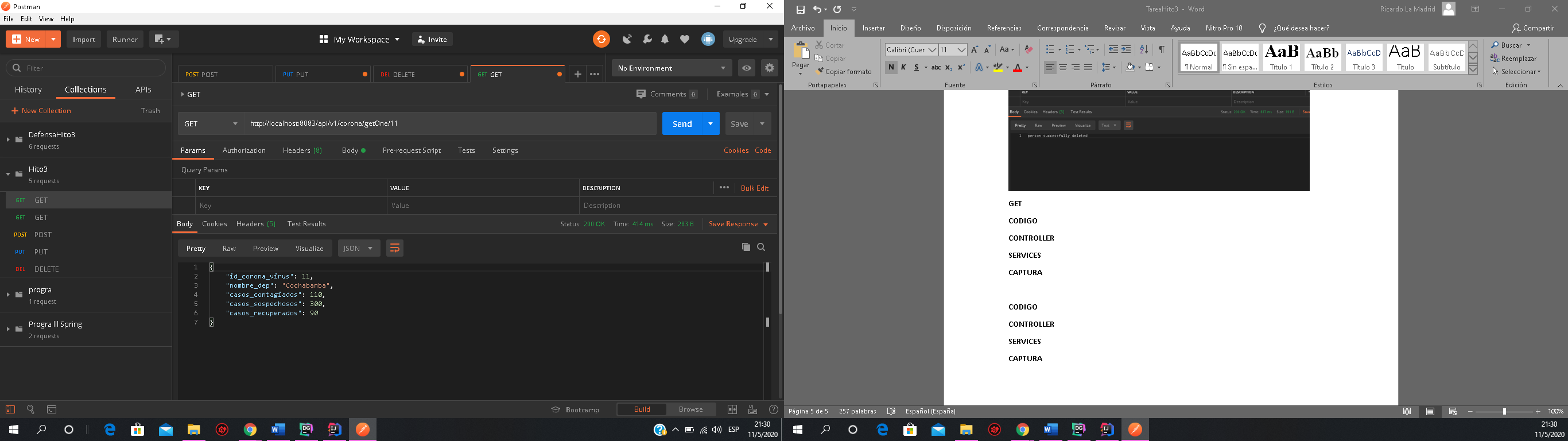
**CONTROLLER**

@GetMapping("/corona/getOne/{id\_corona\_virus}")  
public ResponseEntity<CoronaModel> getPersonByIdPer(@PathVariable("id\_corona\_virus") Integer *id\_corona\_virus*) {  
 try {  
 CoronaModel pModel = coronaService.getPersonByIdPer(*id\_corona\_virus*);  
  
 if (pModel != null) {  
 return new ResponseEntity<>(pModel, *HttpStatus*.*OK*);  
 } else {  
 return new ResponseEntity<>(*HttpStatus*.*NOT\_FOUND*);  
 }  
 } catch (Exception *e*) {  
 return new ResponseEntity<>(null, *HttpStatus*.*INTERNAL\_SERVER\_ERROR*);  
 }  
}

**SERVICES**

@Override  
public CoronaModel getPersonByIdPer(Integer *id\_corona\_virus*) {  
 Optional<CoronaModel> corona = coronaRepo.findById(*id\_corona\_virus*);  
 CoronaModel cModel = null;  
  
 if (corona.isPresent()) {  
 cModel = corona.get();  
 }  
 return cModel;  
}

**CAPTURA**



**GET ALL**

**CODIGO**

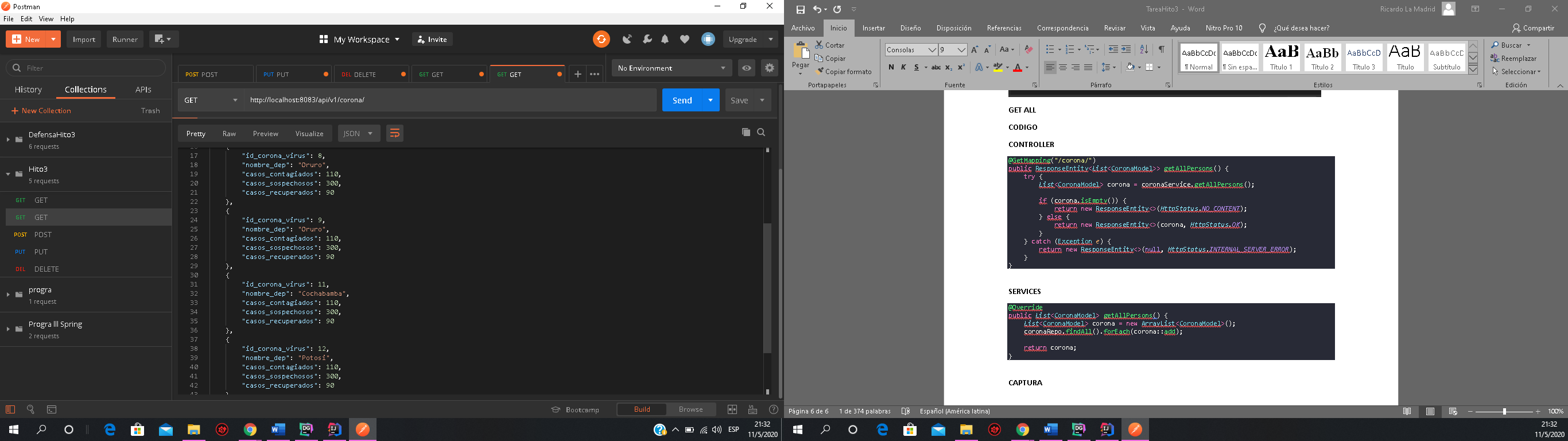
**CONTROLLER**

@GetMapping("/corona/")  
public ResponseEntity<*List*<CoronaModel>> getAllPersons() {  
 try {  
 *List*<CoronaModel> corona = coronaService.getAllPersons();  
  
 if (corona.isEmpty()) {  
 return new ResponseEntity<>(*HttpStatus*.*NO\_CONTENT*);  
 } else {  
 return new ResponseEntity<>(corona, *HttpStatus*.*OK*);  
 }  
 } catch (Exception *e*) {  
 return new ResponseEntity<>(null, *HttpStatus*.*INTERNAL\_SERVER\_ERROR*);  
 }  
}

**SERVICES**

@Override  
public *List*<CoronaModel> getAllPersons() {  
 *List*<CoronaModel> corona = new ArrayList<CoronaModel>();  
 coronaRepo.findAll().forEach(corona::add);  
  
 return corona;  
}

**CAPTURA**



**CLASE MODEL**

import javax.persistence.\*;  
import java.util.Date;  
  
@Entity  
@Table(name = "coronavirus")  
public class CoronaModel {  
  
 @Id  
 @GeneratedValue(strategy = *GenerationType*.*AUTO*)  
 private int id\_corona\_virus;  
 @Column(name = "nombre\_dep", length = 50, nullable = false)  
 private String nombre\_dep;  
 @Column(name = "casos\_contagiados")  
 private int casos\_contagiados;  
 @Column(name = "casos\_sospechosos")  
 private int casos\_sospechosos;  
 @Column(name = "casos\_recuperados")  
 private int casos\_recuperados;  
  
 public int getId\_corona\_virus() {  
 return id\_corona\_virus;  
 }  
 public void setId\_corona\_virus(Integer *id\_corona\_virus*) {  
 *this*.id\_corona\_virus = *id\_corona\_virus*;  
 }  
 public String getNombre\_dep() {  
 return nombre\_dep;  
 }  
 public void setNombre\_dep(String *nombre\_dep*) {  
 *this*.nombre\_dep = *nombre\_dep*;  
 }  
 public int getCasos\_contagiados() {  
 return casos\_contagiados;  
 }  
 public void setCasos\_contagiados(Integer *casos\_contagiados*) {  
 *this*.casos\_contagiados = *casos\_contagiados*;  
 }  
 public int getCasos\_sospechosos() {  
 return casos\_sospechosos;  
 }  
 public void setCasos\_sospechosos(Integer *casos\_sospechosos*) {  
 *this*.casos\_sospechosos = *casos\_sospechosos*;  
 }  
 public int getCasos\_recuperados() {  
 return casos\_recuperados;  
 }  
 public void setCasos\_recuperados(Integer *casos\_recuperados*) {  
 *this*.casos\_recuperados = *casos\_recuperados*;  
 }  
  
}

**INTERFACE CORONASERVICEINTERFACE**

import coronavaitus.coronavairus.Model.CoronaModel;  
  
import java.util.*List*;  
  
public interface *CoronaServiceInterface* {  
  
 public CoronaModel save(CoronaModel *cModel*);  
 public CoronaModel update(CoronaModel *cModel*, Integer *id\_corona\_virus*);  
 public Integer delete(Integer *id\_corona\_virus*);  
 public *List*<CoronaModel> getAllPersons();  
 public CoronaModel getPersonByIdPer(Integer *id\_corona\_virus*);  
  
}

**INTERFACE CORONAREPO**

import coronavaitus.coronavairus.Model.CoronaModel;  
import org.springframework.data.jpa.repository.*JpaRepository*;  
  
  
public interface *CoronaRepo* extends *JpaRepository*<CoronaModel, Integer> {  
}

**CONEXIÓN**

spring.jpa.database=*POSTGRESQL*spring.jpa.show-sql=true  
spring.jpa.hibernate.ddl-auto=update  
spring.datasource.driver-class-name=org.postgresql.Driver  
spring.datasource.url=jdbc:postgresql://ec2-3-91-112-166.compute-1.amazonaws.com:5432/d4gabrk7k7vnu3  
spring.datasource.username=ytutllpgunpvgx  
spring.datasource.password=f8c7db36c571d71d03efaec82abdd381624737f5a51b5f648c002503fab23b47  
spring.jpa.properties.hibernate.jdbc.lob.non\_contextual\_creation=true  
  
server.port=8083