



DEFENSA HITO 3

JUAN ELIAN ALVAREZ VALLEJOS - SIS
PROGRAMACION III

CONSIGNA

Se tiene como objetivo generar un servicio REST utilizando SPRING FRAMEWORK para tratar los casos de CORONA VIRUS PACIENTE en Bolivia.

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

Deberá de generar los siguientes servicios REST.

| | | | |
|--------|---|-----------------------------------|-----|
| POST | /coronaVirusPaciente | Adds the new case CV in the store | 🔒 ↕ |
| PUT | /coronaVirusPaciente/{idCoronaVirus} | Updates a specific CV row | 🔒 ↕ |
| GET | /coronaVirusPaciente/getOne/{idCoronaVirus} | Find CV row by ID | 🔒 ↕ |
| GET | /coronaVirusPaciente/ | Gets all cv records | 🔒 ↕ |
| DELETE | /coronaVirusPaciente/deleteCV/{idCoronaVirus} | Deletes a CV | 🔒 ↕ |

Base de datos

- Se basara en el siguiente modelo.



The image shows a screenshot of a database schema diagram for a table named 'corona_virus_pacien'. The table has the following columns:


| id_corona_virus |
|--------------------|
| nombre_dep |
| nombre_paciente |
| apellidos_paciente |
| edad_paciente |
| categoria |
| fullname |
| casos_contagiados |
| casos_sospechosos |
| casos_recuperados |

The columns 'categoria' and 'fullname' are highlighted with a yellow border. The column 'id_corona_virus' is marked as the primary key with a key icon.

Paso 1

- Debes crear tu Spring Application desde la pagina Spring.io.
- Colocar el nombre del Proyecto y sus dependencias - “Spring Web” “Thymeleaf” y después presionar GENERATE.
- Como se muestra en la siguiente diapositiva.

Paso 1

 **spring** initializr

Project
☒ Maven Project
☐ Gradle Project

Language
☒ Java ☐ Kotlin ☐ Groovy

Spring Boot
☐ 2.3.1 (SNAPSHOT) ☒ 2.3.0 ☐ 2.2.8 (SNAPSHOT) ☐ 2.2.7
☐ 2.1.15 (SNAPSHOT) ☐ 2.1.14

Project Metadata

Group

com.COVID19

Artifact

Hito3

Name

Hito3

Description

Demo project for Spring Boot

Package name

com.COVID19.Hito3



Dependencies [ADD DEPENDENCIES... CTRL + B](#)

Spring Web **WEB**

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Thymeleaf **TEMPLATE ENGINES**

A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.



GENERATE CTRL + G

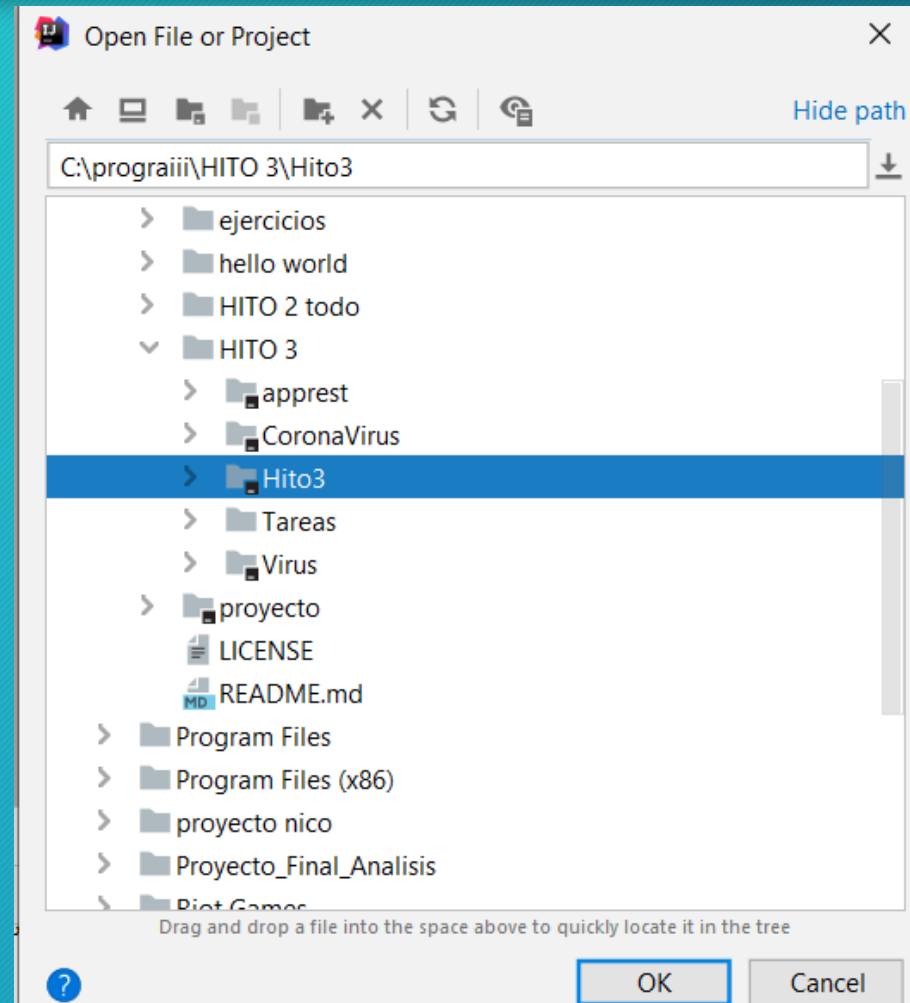
EXPLORE CTRL + SPACE

SHARE...

Paso 2

- Desde en IntelliJ IDEA se abre la carpeta que se descargo de la pagina Spring.io.
- Como en la siguiente diapositiva

Paso 2



Paso 3

- Se debe agregar las dependencias que harán falta que son:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <scope>runtime</scope>
</dependency>
```

Después se agrega a las dependencias entrando a pom.xml

Ir a la siguiente diapositiva

Paso 3

The screenshot shows an IDE with a project named 'Hito3'. The left sidebar displays the project structure, including folders like 'main', 'resources', and 'test'. The main editor shows the 'pom.xml' file with the following XML content:

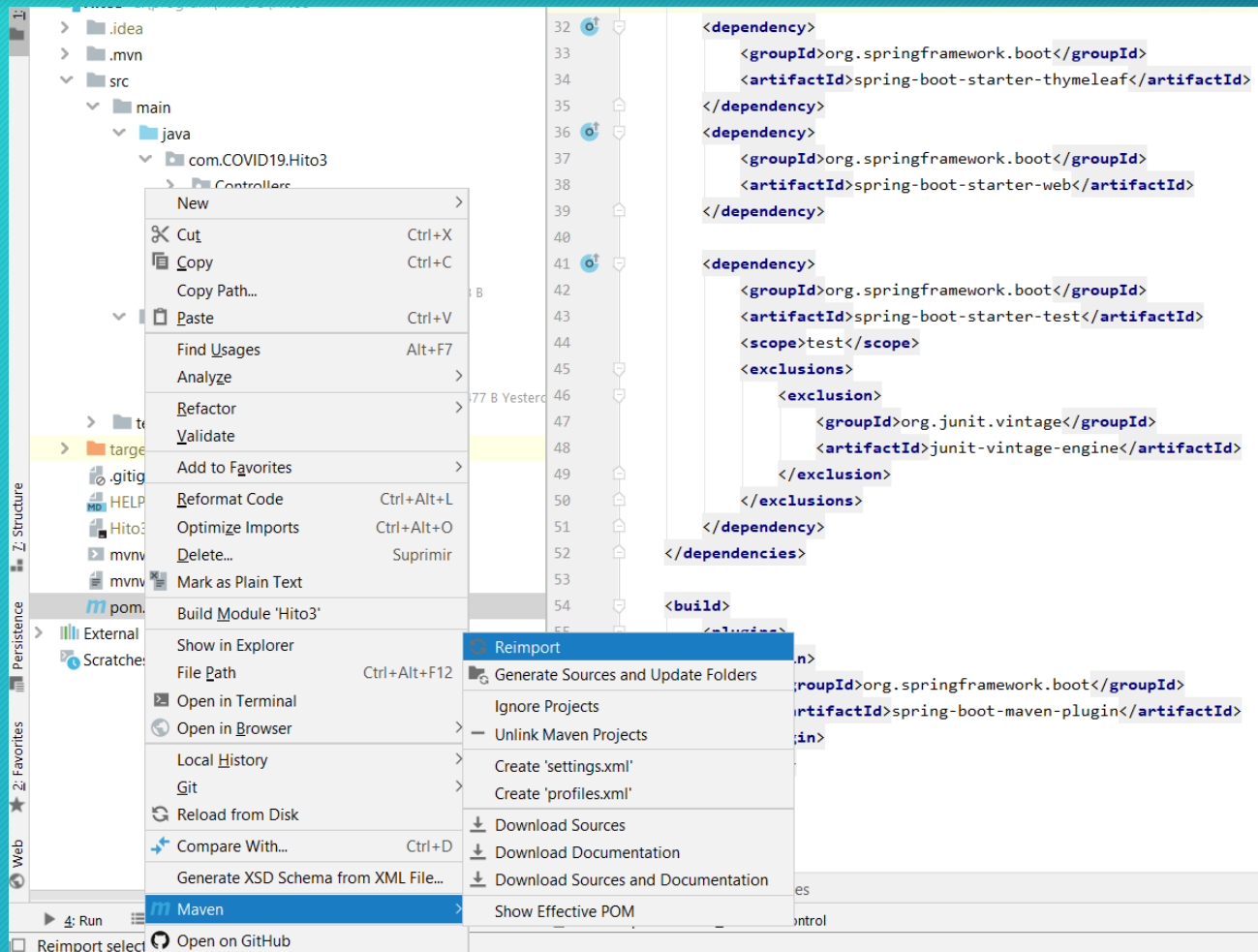
```
18 <java.version>1.8</java.version>
19 </properties>
20
21 <dependencies>
22   <dependency>
23     <groupId>org.springframework.boot</groupId>
24     <artifactId>spring-boot-starter-data-jpa</artifactId>
25   </dependency>
26   <dependency>
27     <groupId>org.postgresql</groupId>
28     <artifactId>postgresql</artifactId>
29     <scope>runtime</scope>
30   </dependency>
31
32   <dependency>
33     <groupId>org.springframework.boot</groupId>
34     <artifactId>spring-boot-starter-thymeleaf</artifactId>
35   </dependency>
36   <dependency>
37     <groupId>org.springframework.boot</groupId>
38     <artifactId>spring-boot-starter-web</artifactId>
39   </dependency>
40
41   <dependency>
42     <groupId>org.springframework.boot</groupId>
43     <artifactId>spring-boot-starter-test</artifactId>
44     <scope>test</scope>
45     <exclusions>
46       <exclusion>
47         <groupId>org.junit.vintage</groupId>
48         <artifactId>junit-vintage-engine</artifactId>
49       </exclusion>
50     </exclusions>
51   </dependency>
</dependencies>
```

Annotations on the image:

- An orange arrow points to the opening `<dependencies>` tag with the text: "Se agregan debajo de <dependencies>".
- A green box highlights the first two dependency blocks, with a green arrow pointing to them from the text: "Dependencias que se agregaron".

The status bar at the bottom indicates the current view is 'project > dependencies'.

Paso 3

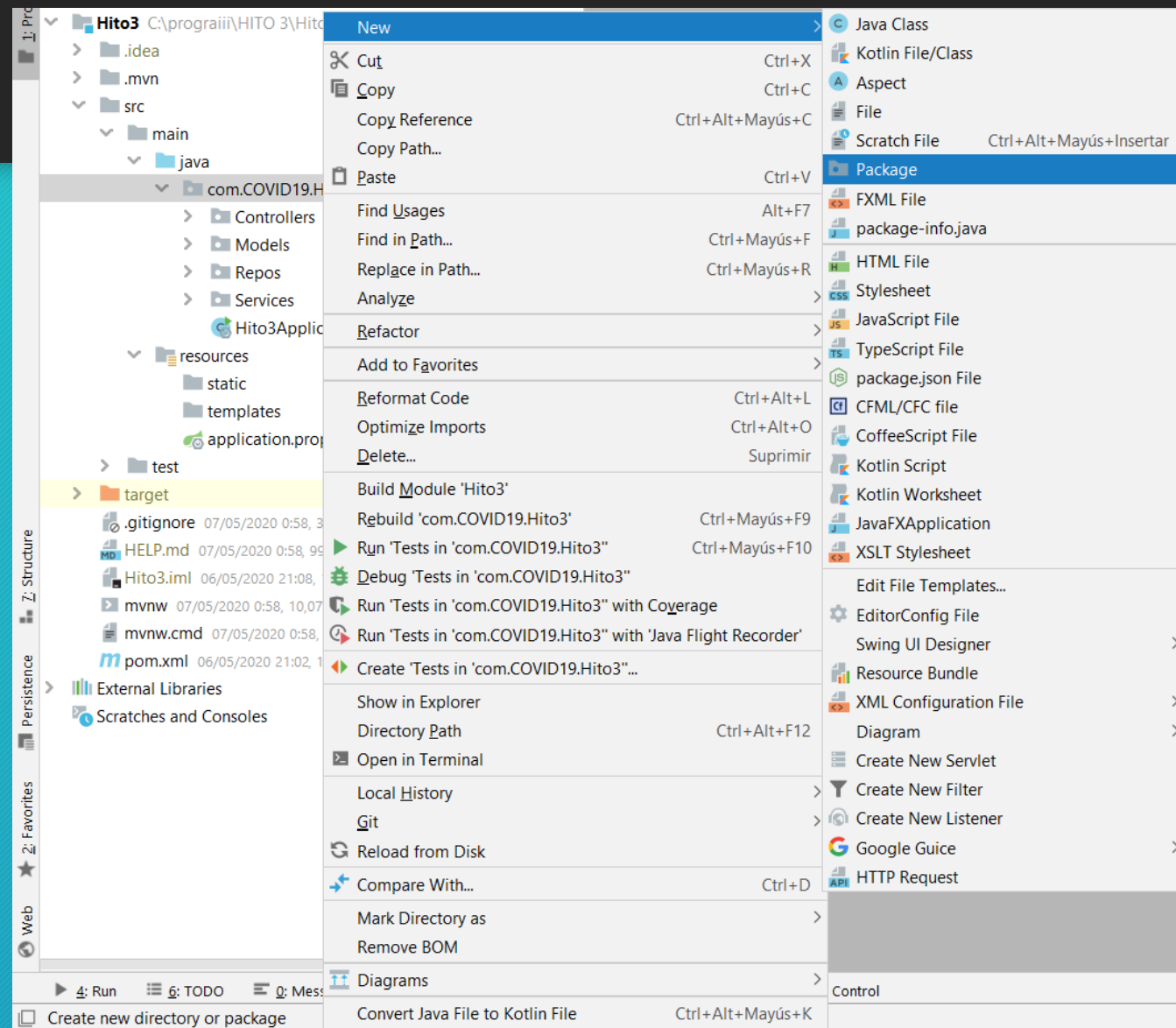


Después de agregar las dependencias se da clic derecho donde dice pom.xml
Después a maven y por ultimo Reimport
“para que actualice las dependencias”

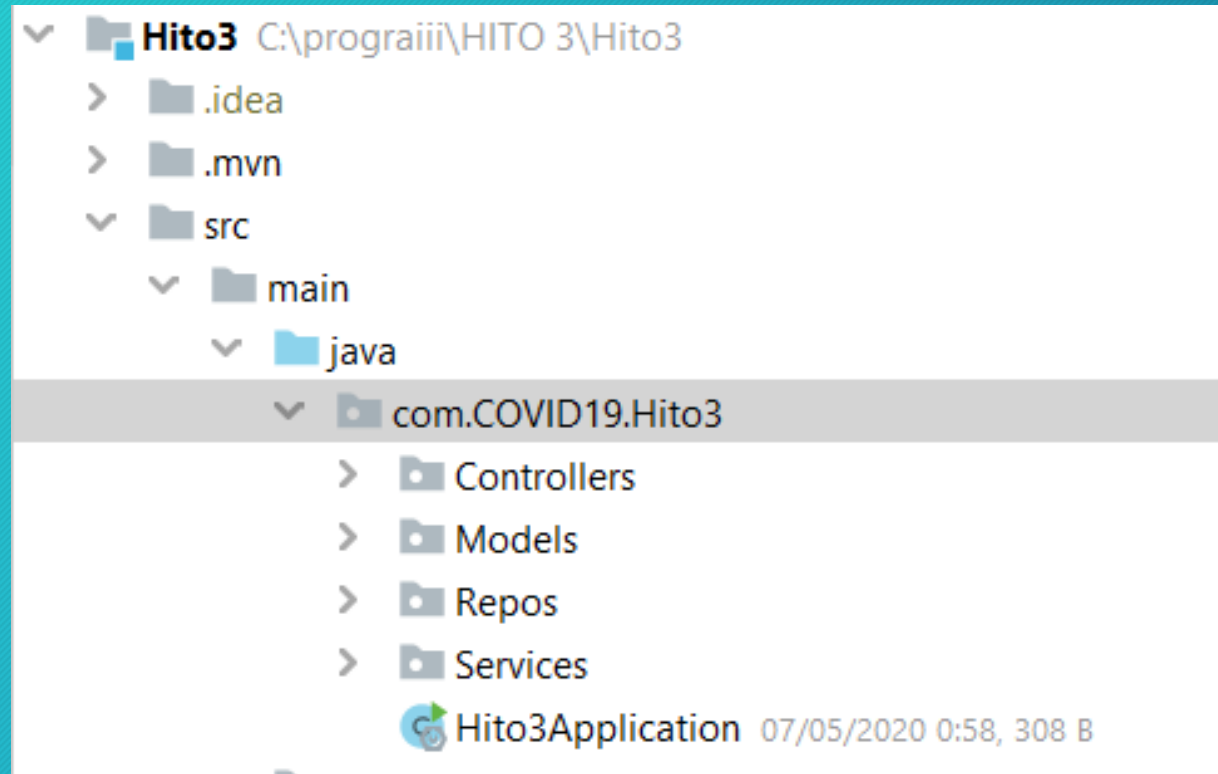
Paso 4

- Se debe crear los PACKAGES “Controller”, “Service”, “Repo” y “Model”. Dando clic derecho en la carpeta que empiece su nombre por “com.”, después new y después en package.
- Como en las siguientes imagenes

Paso 4



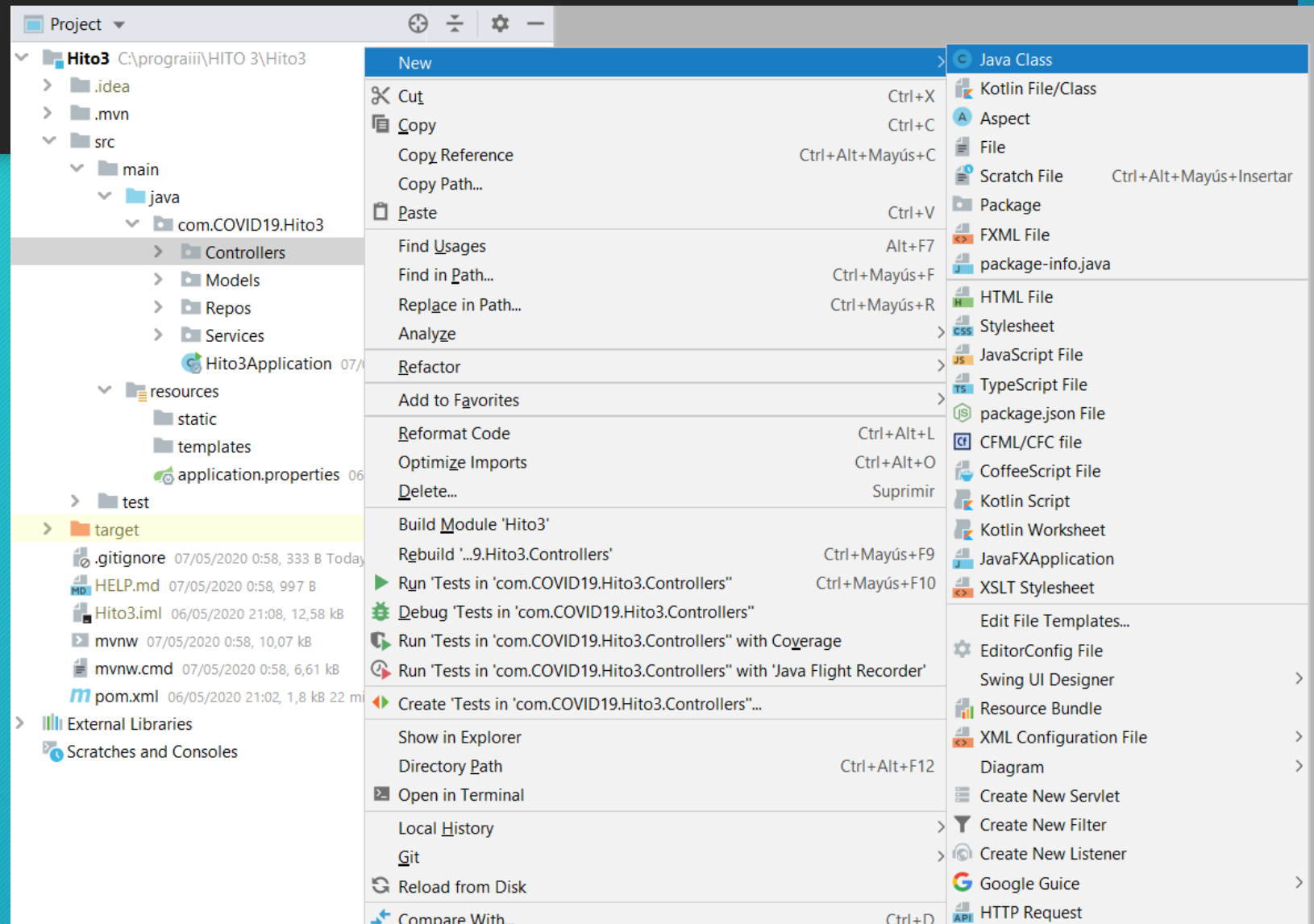
Paso 4 - packages creados



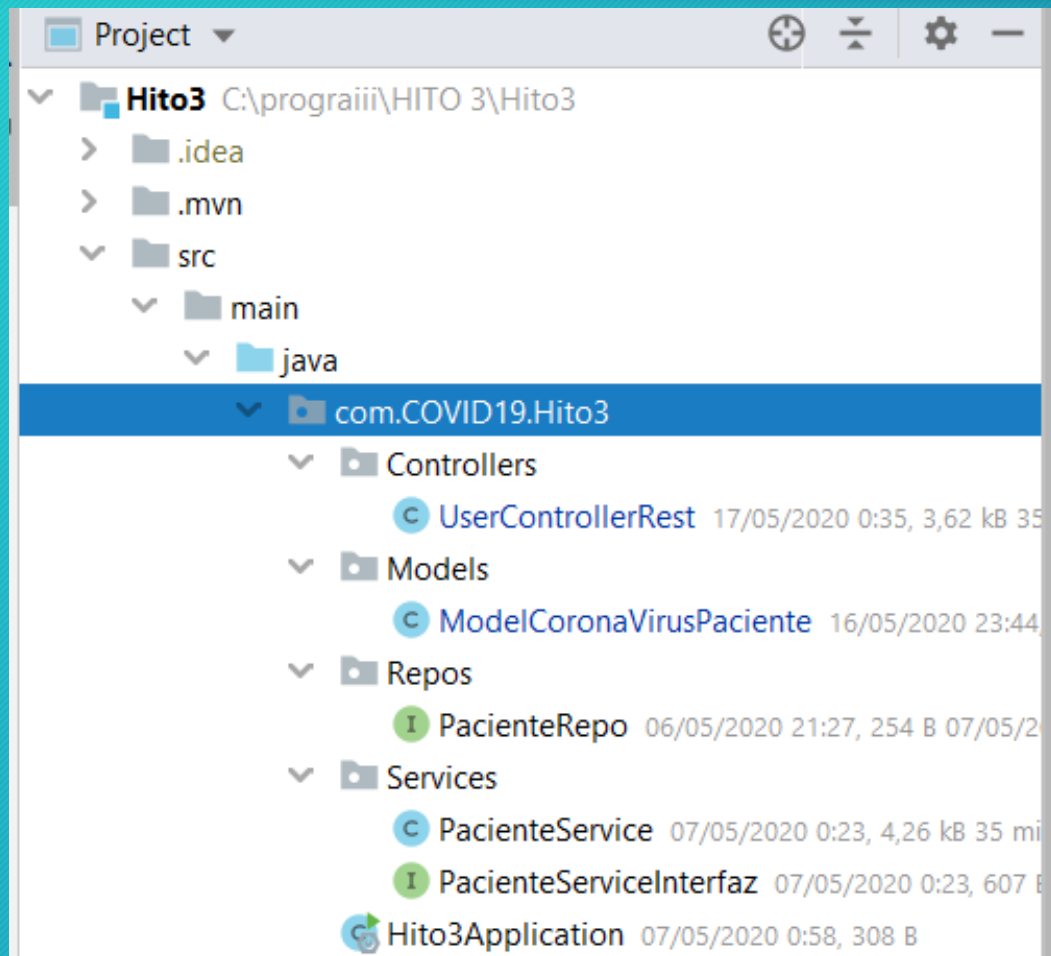
Paso 5

- Se debe crear las clases e interfaces necesarias para dar solución a la defensa.
- Dando clic derecho en el package, después new, después en Java Class y por ultimo darle el nombre.
- Como en la siguiente imagen.

Paso 5



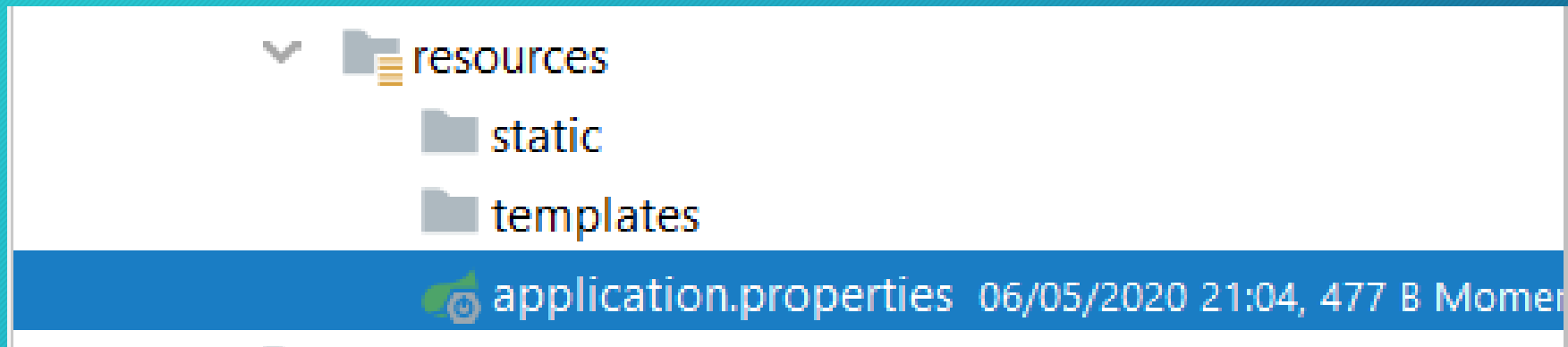
Paso 5



Después de crear las clases e interfaces
Se procederá a resolver las siguientes
preguntas.

Paso 6

- Se debe conectar el IntelliJ IDEA con Datagrip.
- Para eso entramos en la carpeta “resources” y después “application.properties” y generar el siguiente código.




Paso 6

```
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://ec2-52-201-55-4.compute-1.amazonaws.com:5432/d3bhe5r04rr91p
spring.datasource.username=nnqpuwzgiglk1w
spring.datasource.password=4de05cdd255b7e75e308fa5de53d61cfe9f351fa682e7a475bfb999b10368db6
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
```

```
server.port = 8083
```

La url, el username y el password
Se debe colocar según las credenciales que te brinda Heruko como en la siguiente imagen.





Paso 6 - HERUKO


- Primero debes crearte una cuenta en heruko ingresando a heruko.com después creas un proyecto con el nombre que quieras, después entras a la opción Resources y a Add-ons para luego añadir un elemento llamado “Heruko Postgres” y le das clic.

Add-ons


Find more add-ons

Q Quickly add add-ons from Elements

 Heroku Postgres 

Attached as DATABASE 

Hobby Dev

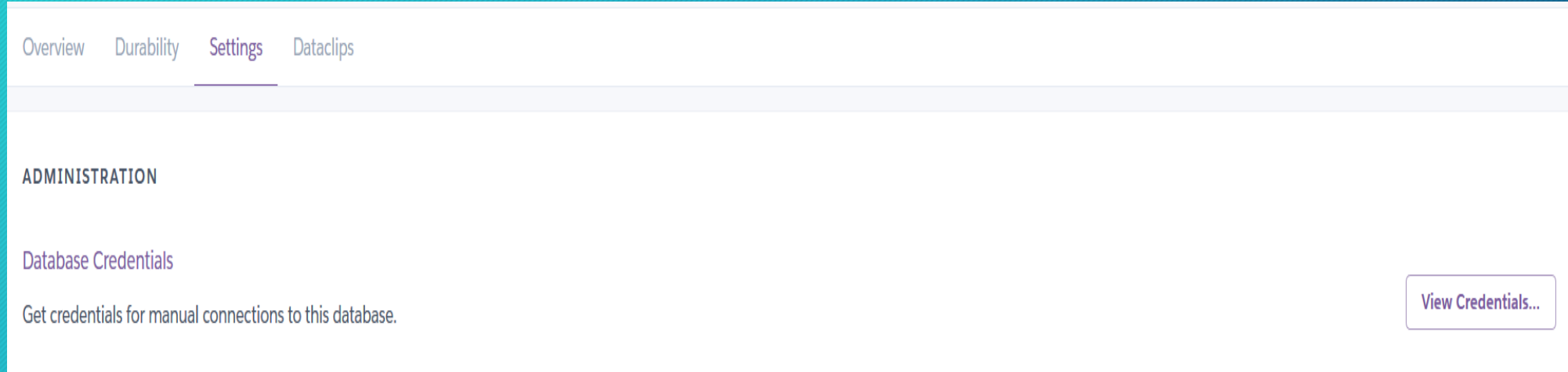
Free 

Estimated Monthly Cost

\$0.00

Paso 6 - HERUKO

- Después de darle clic te enviara a otra ventana. Y ahí vas a Settings y das clic a tus view credentials, donde veras tus credenciales.



Paso 6 - HERUKO - credenciales

Esos datos que nos brinda HERUKO serán de gran utilidad ya que esos credenciales se colocara a Datagrip Y en IntelliJ IDEA.

ADMINISTRATION

Database Credentials

Get credentials for manual connections to this database.

Cancel

Please note that **these credentials are not permanent**.

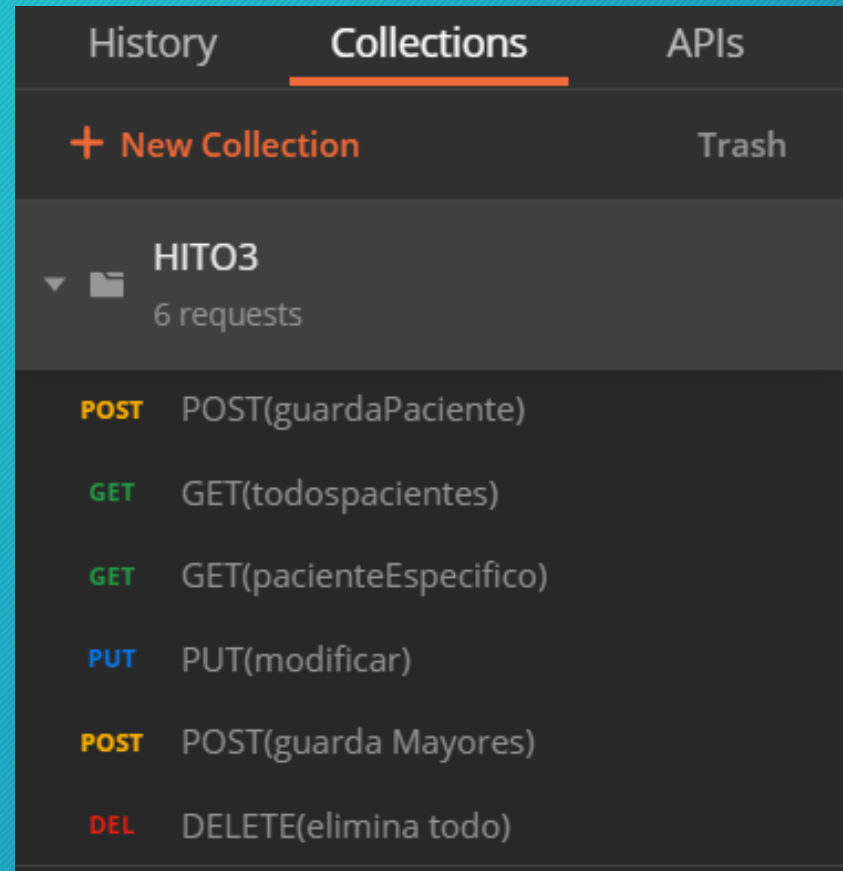
Heroku rotates credentials periodically and updates applications where this database is attached.

| | |
|------------|---|
| Host | ec2-52-201-55-4.compute-1.amazonaws.com |
| Database | d3bhe5r04rr9lp |
| User | nnqpuwzgigklw |
| Port | 5432 |
| Password | 4de05cdd255b7e75e308fa5de53d61cfe9f351fa682e7a475bfb999b10368db6 |
| URI | postgres://nnqpuwzgigklw:4de05cdd255b7e75e308fa5de53d61cfe9f351fa682e7a475bfb999b10368db6@ec2-52-201-55-4.compute-1.amazonaws.com:5432/d3bhe5r04rr9lp |
| Heroku CLI | heroku pg:psql postgresql-contoured-94333 --app prograiii2020elian |

Paso 7

- Se debe usar POSTMAN que “es una extensión del navegador Google Chrome, que permite el envío de peticiones HTTP REST sin necesidad de desarrollar un cliente.”
- En POSTMAN se debe crear una nueva collection con el nombre que quieras y después le das clic derecho y “add request” y crear las que pide la consigna.
- Creas las siguientes:

Paso 7 - POSTMAN



Ejercicio 1

1 Creare el Entity para el modelo **CoronaVirusPaciente**.

Base de datos

| corona_virus_pacie | |
|--------------------|------|
| id_corona_virus | |
| nombre_dep | varc |
| nombre_paciente | varc |
| apellidos_paciente | varc |
| edad_paciente | varc |
| categoria | varc |
| fullname | varc |
| casos_contagiados | |
| casos_sospechosos | |
| casos_recuperados | |

Adjuntar como respuesta

DOL

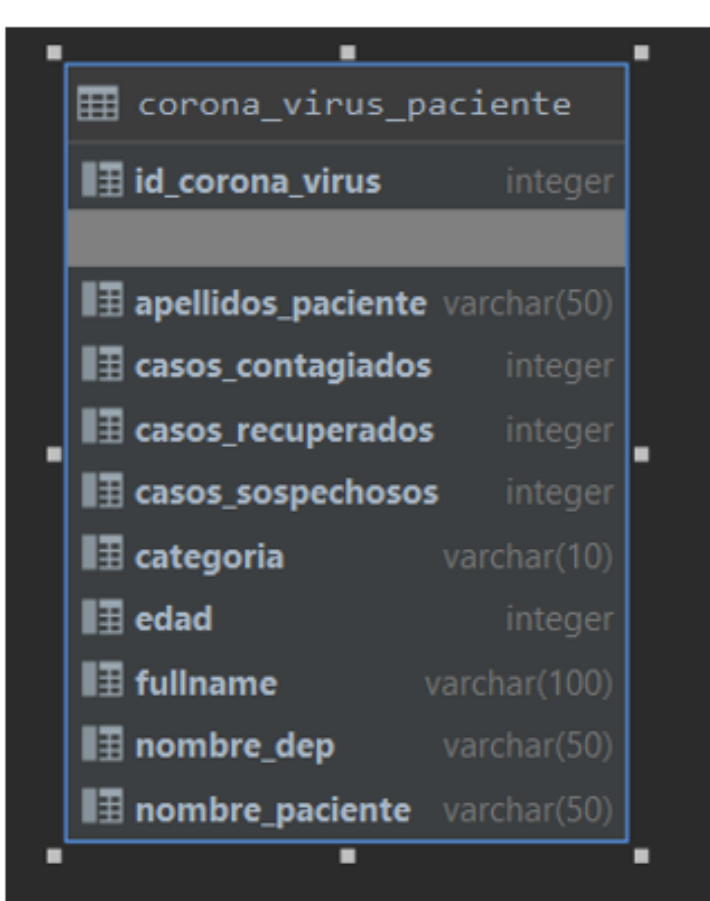
```
SELECT
  COLUMN_NAME
FROM
  Information_schema.COLUMNS
WHERE
  TABLE_NAME = 'city';
```

El Modelo creado: ModelCoronaVirusPaciente.java

Ejercicio 1

- Primero se crea la base de datos desde el IntelliJ Idea mediante código en java y con una conexión con heruko que después se conectara a Datagrip con las credenciales anteriormente mencionadas.
- HERUKO “Es una plataforma como servicio de computación en la Nube que soporta distintos lenguajes de programación y donde te crea una base de datos vía WEB y hace la conexión con Datagrip”.
- Datagrip - “Es una IDE multiplataforma para trabajar con SQL y base de datos.”

Ejercicio 1 - La base de datos desde Datagrip



| corona_virus_paciente | |
|---------------------------|--------------|
| id_corona_virus | integer |
| | |
| apellidos_paciente | varchar(50) |
| casos_contagiados | integer |
| casos_recuperados | integer |
| casos_sospechosos | integer |
| categoria | varchar(10) |
| edad | integer |
| fullname | varchar(100) |
| nombre_dep | varchar(50) |
| nombre_paciente | varchar(50) |

BASE DE DATOS

Ejercicio 1 - el DDL de la base de datos

DDL

```
-- auto-generated definition
create table corona_virus_paciente
(
  id_corona_virus integer not null
    constraint corona_virus_paciente_pkey
    primary key,
  apellidos_paciente varchar(50) not null,
  casos_contagiados integer,
  casos_recuperados integer,
  casos_sospechosos integer,
  categoria varchar(10),
  edad integer,
  fullname varchar(100),
  nombre_dep varchar(50) not null,
  nombre_paciente varchar(50) not null
);

alter table corona_virus_paciente
owner to nnqpwwzgigklw;
```

(Data Definition Language)

Ejercicio 1

- Como se menciono antes la base de datos se crea desde el código que generamos en IntelliJ IDEA - “en la clase Modelo”. Después de hacer eso simplemente debe hacer correr el programa.

Ejercicio 1 - CODIGO

- El código esta dividido en 3 partes para que entre en la presentación pero al final todas las partes son uno solo.

Parte 1

```
package com.COVID19.Hito3.Models;

import javax.persistence.*;

@Entity
@Table(name = "corona_virus_paciente")
public class ModelCoronaVirusPaciente {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id_corona_virus;

    @Column(name = "nombre_dep", length = 50, nullable =
false)
    private String nombre_dep;

    @Column(name = "nombre_paciente", length = 50, nullable =
false)
    private String nombre_paciente;

    @Column(name = "apellidos_paciente", length = 50, nullable
= false)
    private String apellidos_paciente;

    @Column(name = "edad")
    private int edad;

    @Column(name = "categoria", length = 10)
    private String categoria;

    @Column(name = "fullname", length = 100)
    private String fullname;

    @Column(name = "casos_contagiados")
    private int casos_contagiados;

    @Column(name = "casos_sospechosos")
    private int casos_sospechosos;

    @Column(name = "casos_recuperados")
    private int casos_recuperados;
```

Parte 2

```
public Integer 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 String getNombre_paciente() {
    return nombre_paciente;
}

public void setNombre_paciente(String nombre_paciente) {
    this.nombre_paciente = nombre_paciente;
}

public String getApellidos_paciente() {
    return apellidos_paciente;
}

public void setApellidos_paciente(String apellidos_paciente) {
    this.apellidos_paciente = apellidos_paciente;
}

public int getEdad() {
    return edad;
}

public void setEdad(int edad) {
    this.edad = edad;
}
```

Parte 3

```
public String getCategoria() {
    return categoria;
}

public void setCategoria(String categoria) {
    this.categoria = categoria;
}

public String getFullname() {
    return fullname;
}

public void setFullname(String fullname) {
    this.fullname = fullname;
}

public int getCasos_contagiados() {
    return casos_contagiados;
}

public void setCasos_contagiados(int casos_contagiados) {
    this.casos_contagiados = casos_contagiados;
}

public int getCasos_sospechosos() {
    return casos_sospechosos;
}

public void setCasos_sospechosos(int casos_sospechosos) {
    this.casos_sospechosos = casos_sospechosos;
}

public int getCasos_recuperados() {
    return casos_recuperados;
}

public void setCasos_recuperados(int casos_recuperados) {
    this.casos_recuperados = casos_recuperados;
}
}
```


Código del PacienteRepo

Permite que se conecte con el modelo y sus parámetros.

```
PacienteRepo.java X
1 package com.COVID19.Hito3.Repos;
2
3 import com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface PacienteRepo extends JpaRepository<ModelCoronaVirusPaciente, Integer> {
7     |
8 }
```

Ejercicio 2

Generar el servicio REST - POST para poder crear un nuevo caso.

2

Generar el servicio REST - POST para poder crear un nuevo caso.

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** /coronaVirusPaciente
- Description:** Adds the new case CV in the store
- Parameters:** A table with columns 'Name' and 'Description'. It is currently empty.
- body:** A red asterisk indicates it is required. The type is 'object' with a sub-label '(body)'. The description is 'Parametros necesarios para la creacion.' and the example value is 'Model'.
- Example Value:** A JSON object:

```
{  "nombreDepartamento": "Cochabamba",  "nombrePaciente": "Martejk Tyy",  "apellidosPaciente": "Martejk Tyy",  "edad": 22,  "casosContagiados": 56,  "casosSospechosos": 120,  "casosRecuperados": 7}
```
- Parameter content type:** A dropdown menu set to 'application/json'.
- Responses:** A table with columns 'Code' and 'Description'. It contains one entry:

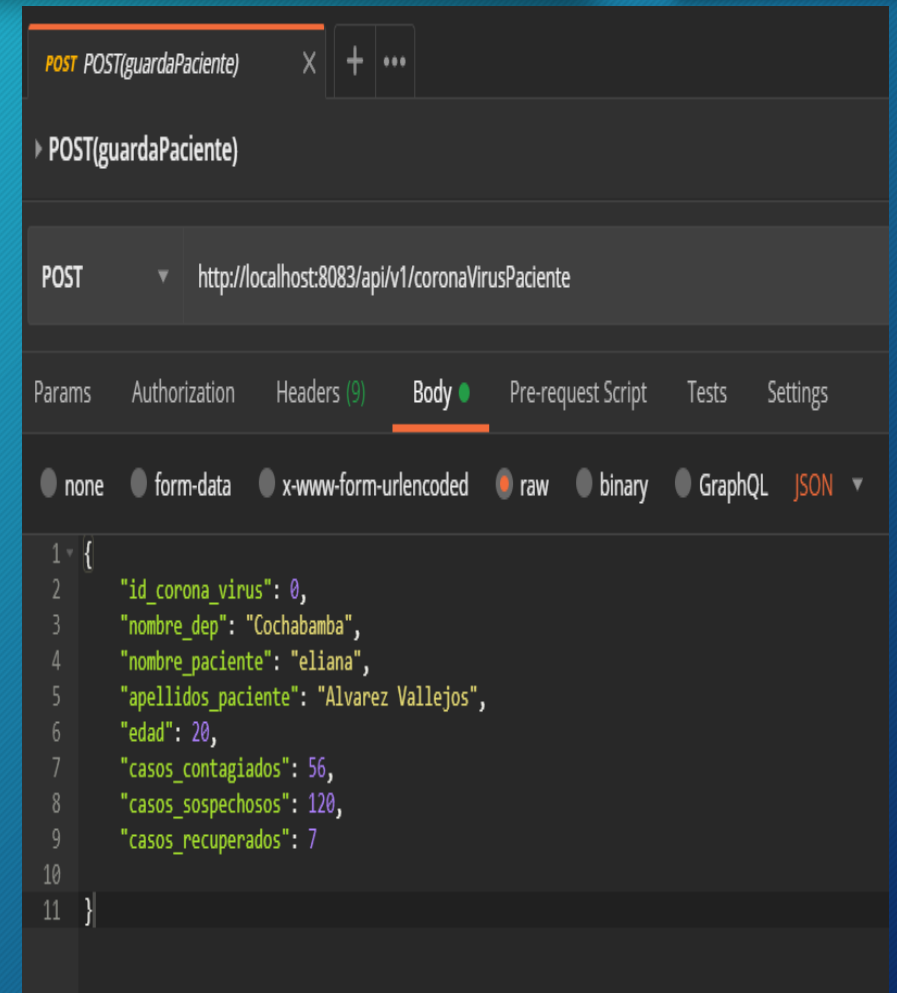
| Code | Description |
|------|-------------|
| 201 | Created |
- Response content type:** A dropdown menu set to 'application/json'.

Ejercicio 2

- Además se tiene una condición que si EDAD es
- Mayor a 20 en su categoría se pondrá ADULTO.
- Menor a 20 en su categoría se pondrá ADOLESCENTE
- Menor a 10 en su categoría se pondrá NINO
- También en el fullname se mandará el nombre y el apellido concatenados.

Ejercicio 2

- Primero pongo los datos en formato Json en el POSTMAN
- y su URL que es la misma que el código se programa.



Ejercicio 2 - CODIGO

Este código se coloca
En la clase "UserControllerRest"

```
package com.COVID19.Hito3.Controllers;

import
com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
import com.COVID19.Hito3.Services.PacienteService;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping(value = "/api/v1")
public class UserControllerRest {

    @Autowired
    private PacienteService pacienteService;

    @PostMapping("/coronaVirusPaciente")
    public ResponseEntity save(@RequestBody
ModelCoronaVirusPaciente paciente){
        try{
            return new
ResponseEntity<>(pacienteService.save(paciente),
HttpStatus.CREATED);
        } catch (Exception e){
            return new ResponseEntity<>(null,
HttpStatus.EXPECTATION_FAILED);
        }
    }
}
```

CODIGO DEL CONTROLLER

"este es el primer
parámetro que se manda"

"este lo que se mandara en
la URL después de mandar
el primer parámetro"

Ejercicio 2 - CODIGO

Este código se coloca en clase
“PacienteService”

```
package com.COVID19.Hito3.Services;

import
com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
import com.COVID19.Hito3.Repos.PacienteRepo;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

@Service
public class PacienteService implements
PacienteServiceInterfaz{
    @Autowired
    private PacienteRepo pacienteRepo;
```

CODIGO DEL SERVICE

```
    @Override
    public ModelCoronaVirusPaciente
    save(ModelCoronaVirusPaciente cModel) {
        if(cModel.getEdad() > 20){
            cModel.setCategoria("ADULTO");

            cModel.setFullname((cModel.getNombre_paciente() +
            cModel.getApellidos_paciente()));

        }
        else if(cModel.getEdad() < 20 &&
        cModel.getEdad() > 9){
            cModel.setCategoria("ADOLESCENTE");

            cModel.setFullname((cModel.getNombre_paciente() +
            cModel.getApellidos_paciente()));

        }
        else{
            cModel.setCategoria("NINO");

            cModel.setFullname((cModel.getNombre_paciente() +
            cModel.getApellidos_paciente()));

        }
        return pacienteRepo.save(cModel);
    }
}
```

“Aquí es donde se
programa el método de
guardar”

La condicional de la EDAD

se coloca aquí y la

Concatenación del nombre
y el apellido.

Ejercicio 2 - CODIGO

```
package com.COVID19.Hito3.Services;  
  
import  
com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;  
  
import java.util.List;  
  
public interface PacienteServiceInterfaz {  
    public ModelCoronaVirusPaciente  
    save(ModelCoronaVirusPaciente cModel);  
}
```

**CODIGO DE LA INTERFAZ DEL
SERVICE**

“Permite la conexión del
service con el modelo”

Este código se coloca en la interface “PacienteServiceInterfaz”

Ejercicio 3

Crear los servicios para poder listar todos los pacientes y en su caso uno solo. REST - GET

3

Crear los servicios para poder listar todos los pacientes y en su caso uno solo. REST - GET

GET /coronaVirusPaciente/getOne/{idCoronaVirus} Find CV row by ID

Returns a single CV

Parameters Try it out

| Name | Description |
|---|--|
| idCoronaVirus * required integer(\$int64) (path) | ID of Corona Virus idCoronaVirus - ID of Corona Virus |

Responses Response content type: application/json

| Code | Description |
|------|---|
| 200 | successful operation Example Value Model <pre>{ "nombredepartamento": "Cachabamba", "nombrePaciente": "Nartejk Iyy", "apellidosPaciente": "Nartejk Iyy", "edad": 22, "casosContagiados": 56, "casosSospechosos": 118, "casosRecuperados": 7}</pre> |
| 404 | Pet not found |
| 500 | Internal server error |

GET de todos los pacientes

- Se tiene:
- Código del Controller.
- Código del Service.

Codigo del Controller

```
package com.COVID19.Hito3.Controllers;

import
com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
import com.COVID19.Hito3.Services.PacienteService;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping(value = "/api/v1")
public class UserControllerRest {

    @Autowired
    private PacienteService pacienteService;

    @GetMapping("/coronaVirusPaciente")
    public ResponseEntity<List<ModelCoronaVirusPaciente>>
    getAllpaciente() {
        try {
            List<ModelCoronaVirusPaciente> paciente =
            pacienteService.getAllpaciente();

            if (paciente.isEmpty()) {
                return new
                ResponseEntity<>(HttpStatus.NO_CONTENT);
            } else {
                return new ResponseEntity<>(paciente,
                HttpStatus.OK);
            }
        } catch (Exception e) {
            return new ResponseEntity<>(null,
            HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
}
```

CODIGO DEL CONTROLLER

“este es el
primer
parámetro que
se manda al
postman”

“este lo que se
mandara en la
URL después de
mandar el
primer
parámetro”

Codigo del Service

```
package com.COVID19.Hito3.Services;

import
com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
import com.COVID19.Hito3.Repos.PacienteRepo;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;


import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

@Service
public class PacienteService implements
PacienteServiceInterfaz{
    @Autowired
    private PacienteRepo pacienteRepo;
    @Override
    public List<ModelCoronaVirusPaciente> getAllpaciente()
    {
        List<ModelCoronaVirusPaciente> paciente = new
        ArrayList<ModelCoronaVirusPaciente>();
        pacienteRepo.findAll().forEach(paciente::add);

        return paciente;
    }
}
```

CODIGO DEL SERVICE

"Aqui es donde se
hace el codigo
para mostrar la
lista de todos los
pacientes"



GET de los pacientes por ID de su departamento donde vive

- Se tiene:
- Código del Controller.
- Código del Service.

Codigo del Controller

```
package com.COVID19.Hito3.Controllers;

import com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
import com.COVID19.Hito3.Services.PacienteService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping(value = "/api/v1")
public class UserControllerRest {

    @Autowired
    private PacienteService pacienteService;

    @GetMapping("/coronaVirusPaciente/getOne/{id_corona_virus}")
    public ResponseEntity<ModelCoronaVirusPaciente>
    getcasosById corona_virus(@PathVariable("id_corona_virus") Integer
    id_corona_virus) {
        try {
            ModelCoronaVirusPaciente cModel =
            pacienteService.getcasosById corona_virus(id corona_virus);

            if (cModel != null) {
                return new ResponseEntity<>(cModel, HttpStatus.OK);
            } else {
                return new ResponseEntity<>(HttpStatus.NOT_FOUND);
            }
        } catch (Exception e) {
            return new ResponseEntity<>(null,
            HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
}
```

CODIGO DEL CONTROLLER

“este es el primer
parámetro que se
manda al postman”

“este lo que se
mandara en la URL
después de mandar el
primer parámetro”

Codigo del Service

```
package com.COVID19.Hito3.Services;

import com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
import com.COVID19.Hito3.Repos.PacienteRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

@Service
public class PacienteService implements PacienteServiceInterfaz{
    @Autowired
    private PacienteRepo pacienteRepo;
    @Override
    public ModelCoronaVirusPaciente
getcasosById corona virus(Integer id corona virus) {
        Optional<ModelCoronaVirusPaciente> paciente =
pacienteRepo.findById(id corona virus);
        ModelCoronaVirusPaciente cModel = null;

        if (paciente.isPresent()) {
            cModel = paciente.get();
        }
        return cModel;
    }
}
```

CODIGO DEL SERVICE

"Aqui es donde se hace el código para mostrar al paciente según su ID"



Código de la Interfaz del Service de ambos GETs

```
package com.COVID19.Hito3.Services;

import com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;

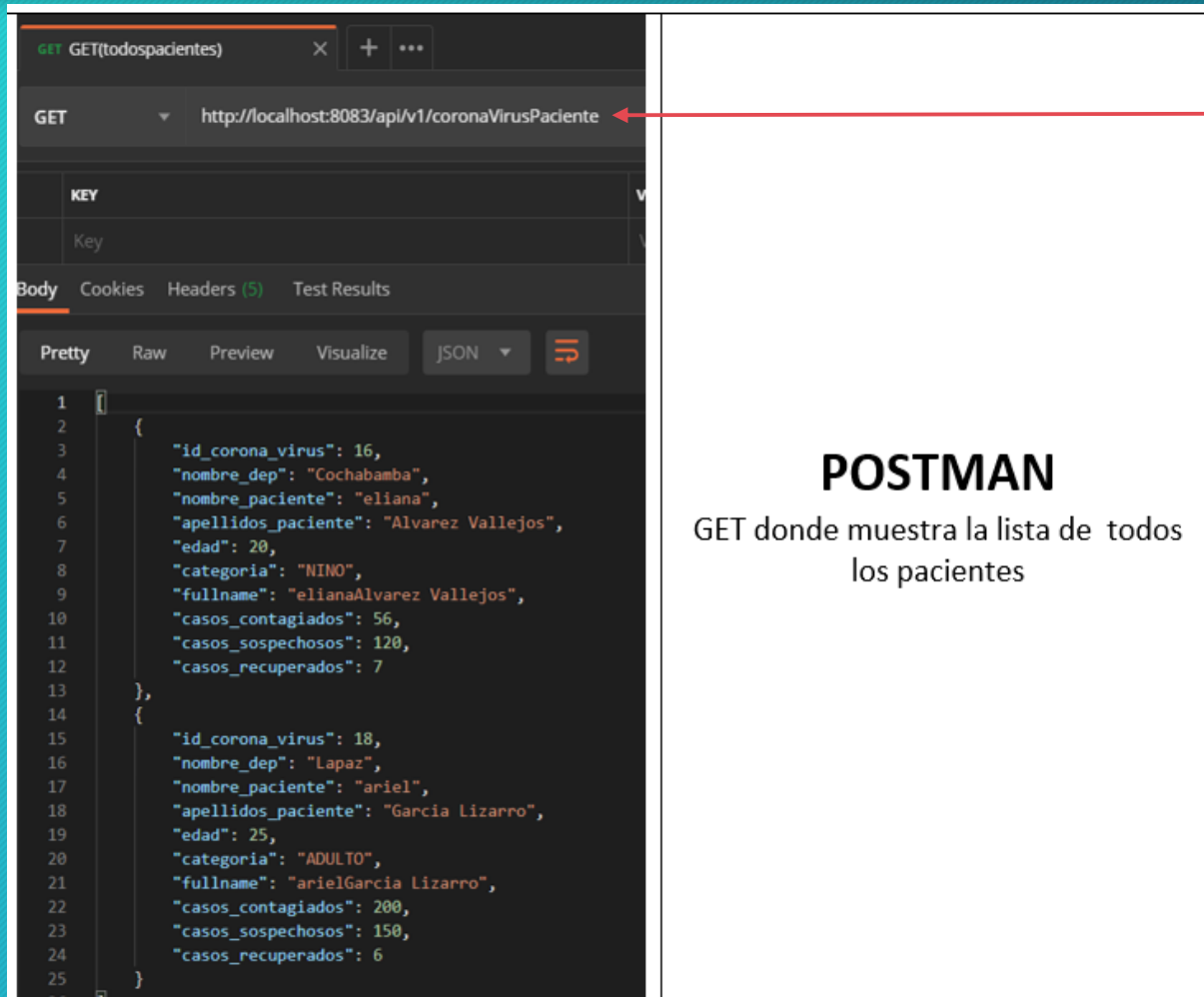
import java.util.List;

public interface PacienteServiceInterfaz {

    public List<ModelCoronaVirusPaciente> getAllpaciente();
    public ModelCoronaVirusPaciente
    getcasosById corona virus(Integer id corona virus);
}
```

**CODIGO DE LA
INTERFAZ DEL SERVICE
DE AMBOS GETs**

POSTMAN DEL PRIMER GET

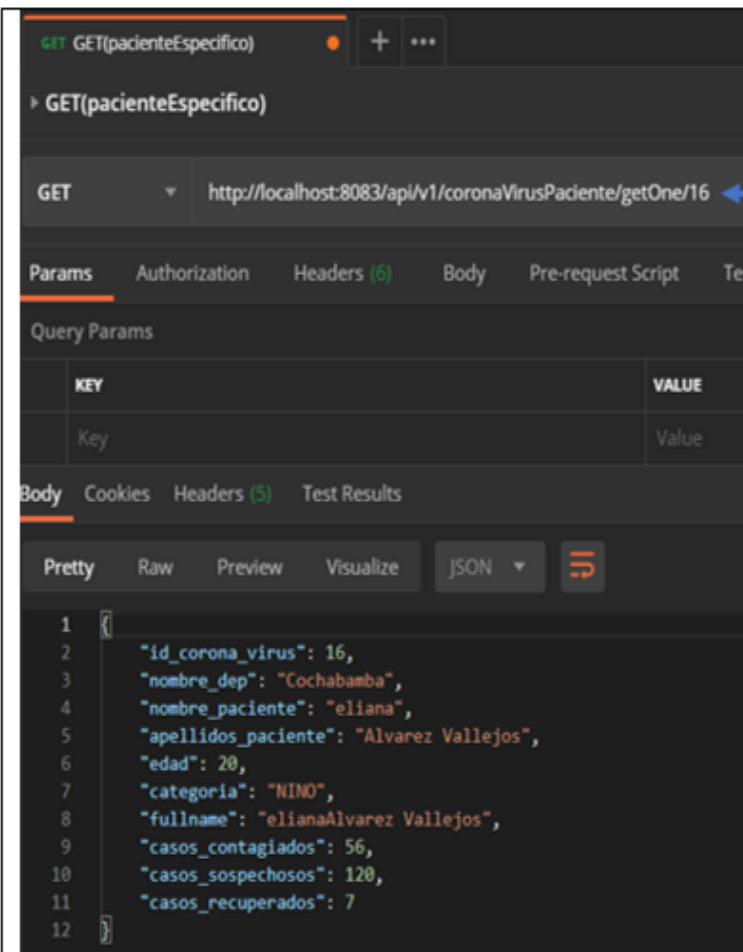


En esta URL es donde se manda los parámetros

POSTMAN

GET donde muestra la lista de todos los pacientes

POSTMAN DEL SEGUNDO GET



The screenshot shows a Postman interface with a GET request to `http://localhost:8083/api/v1/coronaVirusPaciente/getOne/16`. The response is a JSON object with the following data:

| KEY | VALUE |
|--------------------|------------------------|
| id_corona_virus | 16 |
| nombre_dep | Cochabamba |
| nombre_paciente | eliana |
| apellidos_paciente | Alvarez Vallejos |
| edad | 20 |
| categoria | NINO |
| fullname | elianaAlvarez Vallejos |
| casos_contagiados | 56 |
| casos_sospechosos | 120 |
| casos_recuperados | 7 |

POSTMAN

GET donde muestra a un paciente en específico mediante su ID

EJERCICIO 4

Crear un servicio REST - PUT que permita modificar un registro CVP.

Considere que también debe modificar los campos: FULLNAME y CATEGORIA

4

Crear un servicio REST - PUT que permita modificar un registro CVP.

PUT /coronaVirusPaciente /{idCoronaVirus} Updates a specific CV row

Parameters

Try it out

Name

Description

body * required

object

(body)

Parámetros necesarios para la modificación

Example Value Model

```
{
  "nombreDepartamento": "Cochabamba",
  "nombrePaciente": "Martejk Iyy",
  "apellidosPaciente": "Martejk Iyy",
  "edad": 22,
  "casosContagados": 56,
  "casosSospechosos": 120,
  "casosRecuperados": 7
}
```

Parameter content type

application/json

idCoronaVirus * required

integer(\$int32)

(path)

ID of Corona Virus

idCoronaVirus - ID of Corona Virus

Responses

Response content type

application/json

Code

Description

EJERCICIO 4

- Se tiene:
- Código del Controller
- Código del Service
- Código de la Interfaz
- POSTMAN
- Base de datos

Código del Controller

```
package com.COVID19.Hito3.Controllers;

import com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
import com.COVID19.Hito3.Services.PacienteService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping(value = "/api/v1")
public class UserControllerRest {

    @Autowired
    private PacienteService pacienteService;

    @PutMapping("/coronaVirusPaciente/{id_corona_virus}")
    public ResponseEntity<ModelCoronaVirusPaciente>
updatePaciente(@PathVariable("id_corona_virus") Integer
id_corona_virus, @RequestBody ModelCoronaVirusPaciente
cModel) {
        try {
            ModelCoronaVirusPaciente cUpdate =
pacienteService.update(cModel, id_corona_virus);
            if (cUpdate != null) {
                return new ResponseEntity<>(cUpdate,
HttpStatus.OK);
            } else {
                return new
ResponseEntity<>(HttpStatus.NOT_FOUND);
            }
        } catch (Exception e) {
            return new ResponseEntity<>(null,
HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
}
```

CODIGO CONTROLLER

“Este es el comando
que se mandara en el
postman donde se
especifica la ID del
paciente”

Código del Service

```
package com.COVID19.Hito3.Services;

import com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
import com.COVID19.Hito3.Repos.PacienteRepo;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
```

```
@Service
public class PacienteService implements
PacienteServiceInterfaz{
    @Autowired
    private PacienteRepo pacienteRepo;
```

```
@Override
public ModelCoronaVirusPaciente
update(ModelCoronaVirusPaciente cModel, Integer
id_corona_virus) {
    Optional<ModelCoronaVirusPaciente> paciente =
pacienteRepo.findById(id_corona_virus);
    ModelCoronaVirusPaciente pacienteUpdate = null;

    if (paciente.isPresent()) {
        pacienteUpdate = paciente.get();

        pacienteUpdate.setNombre_dep(cModel.getNombre_dep());

        pacienteUpdate.setNombre_paciente(cModel.getNombre_pacien
te());

        pacienteUpdate.setApellidos_paciente(cModel.getApellidos_
paciente());

        pacienteUpdate.setEdad(cModel.getEdad());
        if(cModel.getEdad() < 10){
            pacienteUpdate.setCategoria("NINO");
        } else if(cModel.getEdad() < 20){
            pacienteUpdate.setCategoria("ADOLESCENTE");
        }
        else{
            pacienteUpdate.setCategoria("ADULTO");
        }

        pacienteUpdate.setFullname(cModel.getNombre_paciente()
+"-"+ cModel.getApellidos_paciente());

        pacienteUpdate.setCasos_contagiados(cModel.getCasos_conta
giados());

        pacienteUpdate.setCasos_sospechosos(cModel.getCasos_sospe
```

```
chosos());

        pacienteUpdate.setCasos_recuperados(cModel.getCasos_recup
erados());

        pacienteRepo.save(pacienteUpdate);
    }
    return pacienteUpdate;
}
}
```

CODIGO SERVICE

“Aquí es donde se hace el código para modificar o actualizar los datos del paciente según su ID”

Código de la Interfaz

```
package com.COVID19.Hito3.Services;

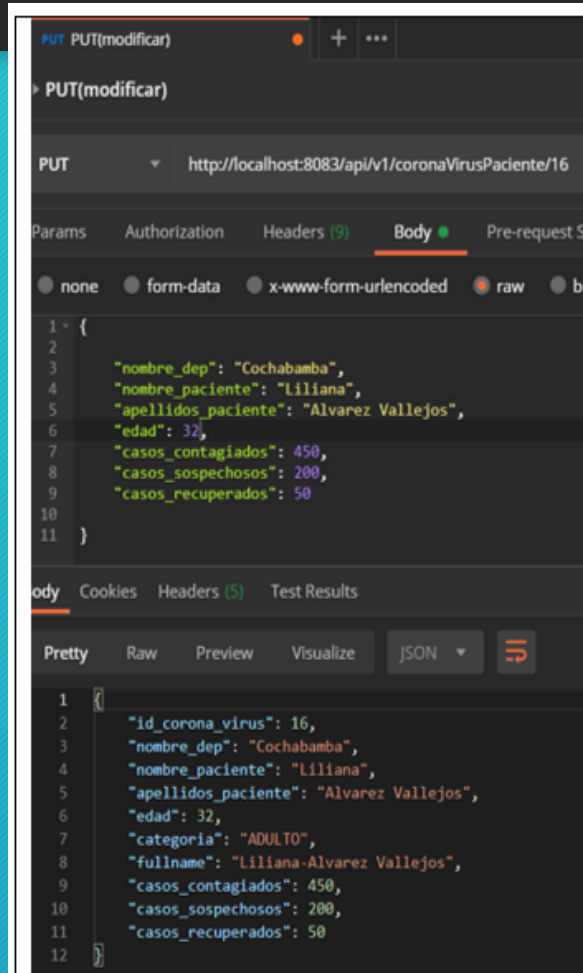
import com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;

import java.util.List;

public interface PacienteServiceInterfaz {
    public ModelCoronaVirusPaciente
    update(ModelCoronaVirusPaciente cModel, Integer
    id_corona_virus);
}
```

CODIGO DE LA
INTERFAZ SERVICE

POSTMAN



POSTMAN

“se especifica el paciente del cual se modificaran algún dato mediante su ID y después de corregir y mandar en la parte abajo se muestra como está el paciente con sus nuevos datos”

Base de datos

Q* <Filter criteria> X

| corona_virus | apellidos_paciente | casos_contagiados | casos_recuperados | casos_sospechosos | categoria | edad | fullname |
|--------------|---------------------|-------------------|-------------------|-------------------|-----------|------|-----------------|
| 1 | 16 Alvarez Vallejos | 56 | 7 | 120 | NINO | 20 | elianaAlvarez \ |
| 2 | 18 Garcia Lizarro | 200 | 6 | 150 | ADULTO | 25 | arielGarcia Li: |

Los datos del paciente antes de modificarlos

Q* <Filter criteria> X

| corona_virus | apellidos_paciente | casos_contagiados | casos_recuperados | casos_sospechosos | categoria | edad | fullname |
|--------------|---------------------|-------------------|-------------------|-------------------|-----------|------|-----------------|
| 1 | 18 Garcia Lizarro | 200 | 6 | 150 | ADULTO | 25 | arielGarcia Li: |
| 2 | 16 Alvarez Vallejos | 450 | 50 | 200 | ADULTO | 32 | Liliana-Alvare: |

Los datos del paciente despues de modificarlos

Ejercicio 5

Evitar insertar en Base de Datos nuevos casos CVP si la edad De el paciente es mayor 70.

5

Evitar insertar en la Base de Datos nuevos casos CVP si la edad de el paciente es mayor 70.

Debe de crear otro servicio rest POST

Preguntas abiertas o de respuestas cortas - 20 puntos - Subjetiva

Ejercicio 5

- Se tiene:
- Código del Controller.
- Código del Service.
- Código de la Interfaz Service.
- POSTMAN.
- Base de datos.

Código del Controller.

Este es parámetro que se mandara en el POSTMAN



```
package com.COVID19.Hito3.Controllers;

import
com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
import com.COVID19.Hito3.Services.PacienteService;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping(value = "/api/v1")
public class UserControllerRest {

    @Autowired
    private PacienteService pacienteService;

    @PostMapping("/coronaVirusPaciente2")
    public ResponseEntity save2(@RequestBody
ModelCoronaVirusPaciente paciente2){
        try{
            return new
ResponseEntity<>((pacienteService.save2(paciente2),
HttpStatus.EXPECTATION_FAILED);
        } catch (Exception e){
            return new ResponseEntity<>(null,
HttpStatus.EXPECTATION_FAILED);
        }
    }
}
```

CODIGO DEL
CONTROLLER

Código del Service

```
package com.COVID19.Hito3.Services;

import
com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
import com.COVID19.Hito3.Repos.PacienteRepo;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

@Service
public class PacienteService implements
PacienteServiceInterfaz{
    @Autowired
    private PacienteRepo pacienteRepo;

    @Override
    public ModelCoronaVirusPaciente
    save2(ModelCoronaVirusPaciente cModel) {
        if(cModel.getEdad() > 70){
            return null;
        }
        else{
            if(cModel.getEdad() > 20){
                cModel.setCategoria("ADULTO");

                cModel.setFullname((cModel.getNombre_paciente() +
                cModel.getApellidos_paciente()));
            }
            else if(cModel.getEdad() < 20 &&
            cModel.getEdad() > 9){
                cModel.setCategoria("ADOLESCENTE");

                cModel.setFullname((cModel.getNombre_paciente() +
                cModel.getApellidos_paciente()));
            }
            else{
                cModel.setCategoria("NINO");

                cModel.setFullname((cModel.getNombre_paciente() +
                cModel.getApellidos_paciente()));
            }
            return pacienteRepo.save(cModel);
        }
    }
}
```

CODIGO DEL SERVICE

“Aquí está el código
que resolverá el
problema”

“Aquí está la condición
de la edad”

Código de la Interfaz Service

```
package com.COVID19.Hito3.Services;

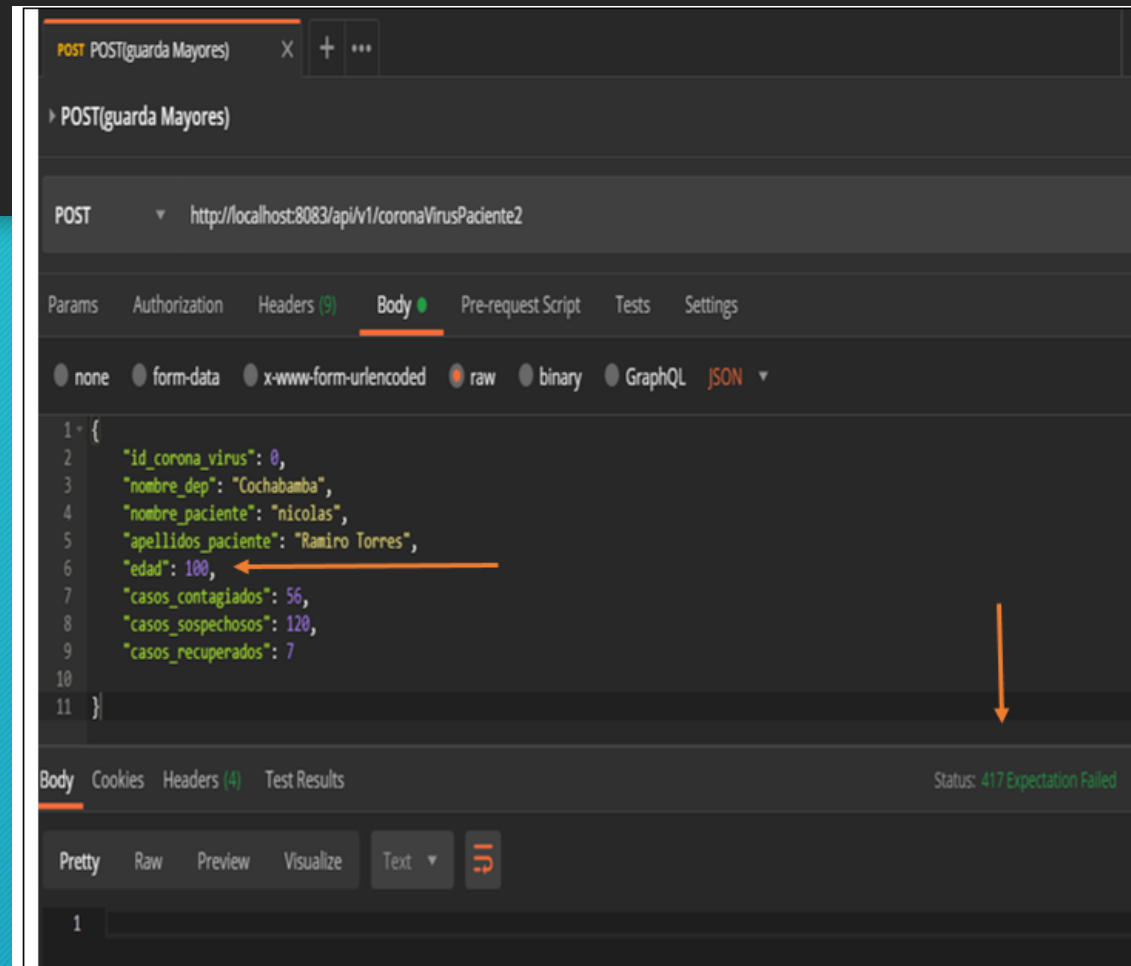
import
com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;

import java.util.List;

public interface PacienteServiceInterfaz {
    public ModelCoronaVirusPaciente
    save2(ModelCoronaVirusPaciente cModel);
}
```

CODIGO DE LA INTERFAZ
SERVICE

POSTMAN



POSTMAN

“cuando se ingresa un paciente que tiene más de 70 años como el ejemplo no se guarda en la base de datos y salta un error que se manda”

Base de datos

| id | corona_virus | apellidos_paciente | casos_contagiados | casos_recuperados | casos_sospechosos | categoria | edad | fullname |
|----|--------------|--------------------|-------------------|-------------------|-------------------|-----------|------|-----------------|
| 1 | 18 | Garcia Lizarro | 200 | 6 | 150 | ADULTO | 25 | arielGarcia Li: |
| 2 | 16 | Alvarez Vallejos | 450 | 50 | 200 | ADULTO | 32 | Liliana-Alvare: |

“Como se ve la base de datos no cambio, ya que no se puede registrar ese paciente porque es mayor de 70 años”

Ejercicio 6

Crear un servicio REST - DELETE que elimina a todos los registros de la base de datos.

6

Crear un servicio **REST - DELETE** que elimina todos los registros de la base de datos.

Preguntas abiertas o de respuestas cortas - 10 puntos - Subjetiva

Ejercicio 6

- Se tiene:
- Código del Controller.
- Código del Service.
- Código de la Interfaz.
- POSTMAN.
- Base de datos.

Código del Controller

```
package com.COVID19.Hito3.Controllers;

import
com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
import com.COVID19.Hito3.Services.PacienteService;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping(value = "/api/v1")
public class UserControllerRest {

    @Autowired
    private PacienteService pacienteService;
    @DeleteMapping("/coronaVirusPaciente/deleteCV")
    public ResponseEntity<String> delete() {
        try {
            pacienteService.delete();
            return new ResponseEntity<>("Departamento
            successfully deleted", HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<>(null,
            HttpStatus.EXPECTATION_FAILED);
        }
    }
}
```

CODIGO DEL
CONTROLLER

Código del Service

```
package com.COVID19.Hito3.Services;

import
com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;
import com.COVID19.Hito3.Repos.PacienteRepo;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

@Service
public class PacienteService implements
PacienteServiceInterfaz{
    @Autowired
    private PacienteRepo pacienteRepo;

    @Override
    public Integer delete() {
        pacienteRepo.deleteAll();
        return 1;
    }
}
```

CODIGO DEL
SERVICE

Código de la Interfaz

```
package com.COVID19.Hito3.Services;

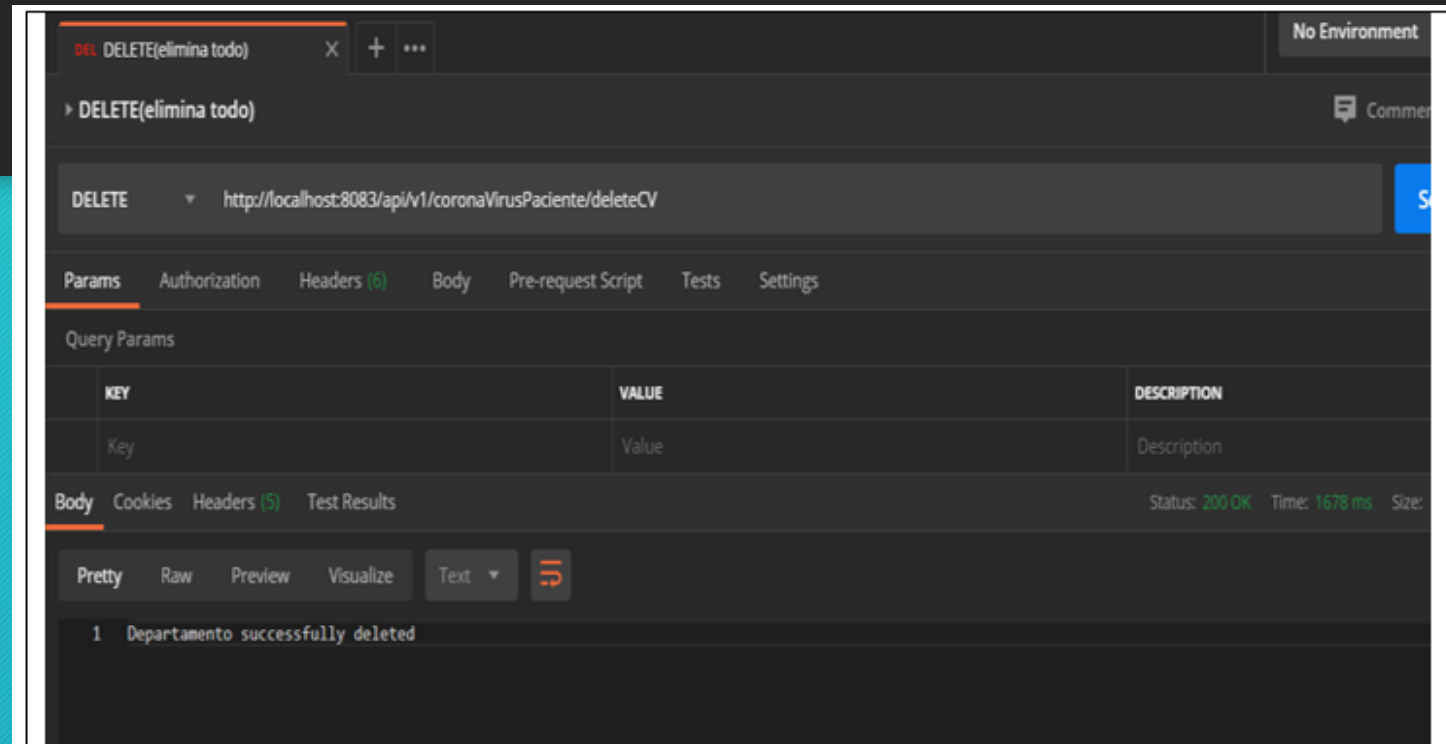
import
com.COVID19.Hito3.Models.ModelCoronaVirusPaciente;

import java.util.List;

public interface PacienteServiceInterfaz {
    public Integer delete();
}
```

CODIGO DE LA
INTERFAZ SERVICE

POSTMAN



POSTMAN

“Se manda el comando en la URL y después de ejecutarlo todos los registros de los pacientes se borrarán”

Base de datos

| corona_virus | apellidos_paciente | casos_contagiados | casos_recuperados | casos_sospechosos | categoria | edad | fullname |
|--------------|---------------------|-------------------|-------------------|-------------------|-----------|------|-----------------|
| 1 | 18 Garcia Lizarro | 200 | 6 | 150 | ADULTO | 25 | arielGarcia Li: |
| 2 | 16 Alvarez Vallejos | 450 | 50 | 200 | ADULTO | 32 | Liliana-Alvarez |

“Los datos de los pacientes antes de ser borrados”

| corona_virus | apellidos_paciente | casos_contagiados | casos_recuperados | casos_sospechosos | categoria | edad | fullname |
|--------------|--------------------|-------------------|-------------------|-------------------|-----------|------|----------|
| | | | | | | | |

“Como se aprecia ya no hay ningún registro guardado”

GRACIAS !!