

Diego Emiliano Rivera Tapia

# Resolución de la Defensa Hito 3

# Pregunta uno

Crear 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

Data definition language generado en la base de datos

```
SELECT
  COLUMN_NAME
FROM
  INFORMATION_SCHEMA.COLUMNS
WHERE
  TABLE_NAME = 'corona_virus_pacie'
```

# Creación del proyecto en el Framework Spring

The screenshot shows the Spring Initializr web application interface. It is divided into several sections: 'Project' (Maven Project, Gradle Project), 'Language' (Java, Kotlin, Groovy), 'Spring Boot' (2.3.0 RC1, 2.3.0 (SNAPSHOT), 2.2.7 (SNAPSHOT), 2.2.6, 2.1.14 (SNAPSHOT), 2.1.13), 'Project Metadata' (Group, Artifact, Name, Description, Package name), 'Packaging' (Jar, War), and 'Dependencies' (No dependency selected). The 'Generate' button is highlighted with a yellow box and the text 'Se genera el entorno.'.

Se seleccionan las dependencias.

Se seleccionan las características.

Se seleccionan los datos.

Se genera el entorno.

# Proyecto Spring

The image shows the Spring Initializr web application interface. It is annotated with yellow boxes and lines to highlight specific sections:

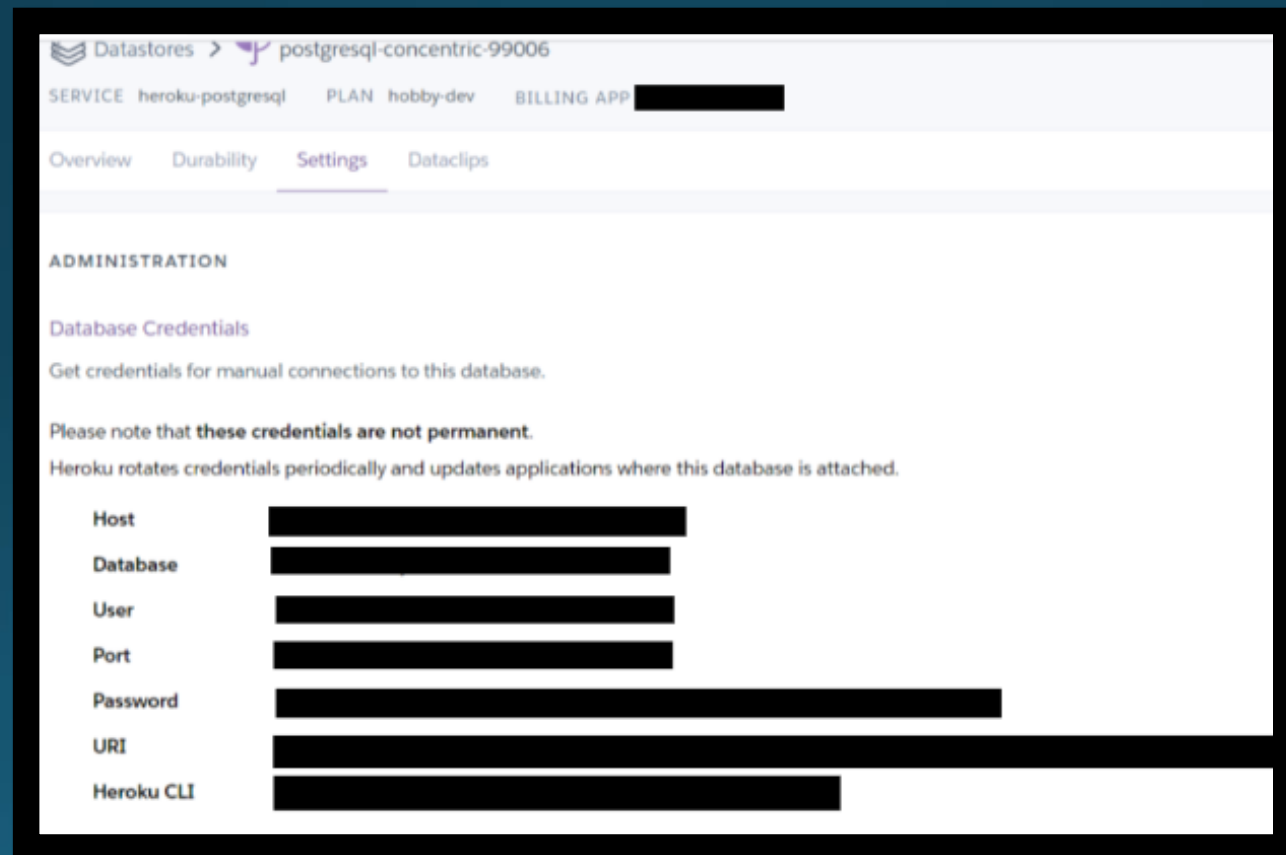
- Dependencias**: A yellow box at the top right pointing to the **Dependencies** section on the right side of the form.
- Datos del proyecto Y BOOT**: A yellow box on the left pointing to the **Project** and **Spring Boot** sections.
- Metadata**: A yellow box on the left pointing to the **Project Metadata** section.
- Genera proyecto**: A yellow box at the bottom pointing to the **GENERATE** button.

The interface itself contains the following elements:

- spring initializr** logo at the top left.
- Project** section: Radio buttons for **Maven Project** (selected) and **Gradle Project**.
- Language** section: Radio buttons for **Java** (selected), **Kotlin**, and **Groovy**.
- Spring Boot** section: Radio buttons for versions **2.3.0 RC1**, **2.3.0 (SNAPSHOT)**, **2.2.8 (SNAPSHOT)**, **2.2.7** (selected), **2.1.15 (SNAPSHOT)**, and **2.1.14**.
- Project Metadata** section: Text input fields for **Group** (com.Hito3), **Artifact** (Procesual), **Name** (Procesual), **Description** (Demo project for Spring Boot), and **Package name** (com.Hito3.Procesual). A **Packaging** section with radio buttons for **Jar** (selected) and **War**.
- Dependencies** section: A list of dependencies including **Spring Web** (WEB) and **Thymeleaf** (TEMPLATE ENGINES). An **ADD DEPENDENCIES... CTRL + B** button is at the top right of this section.
- Buttons** at the bottom: **GENERATE CTRL + G**, **EXPLORE CTRL + SPACE**, and **SHARE...**.

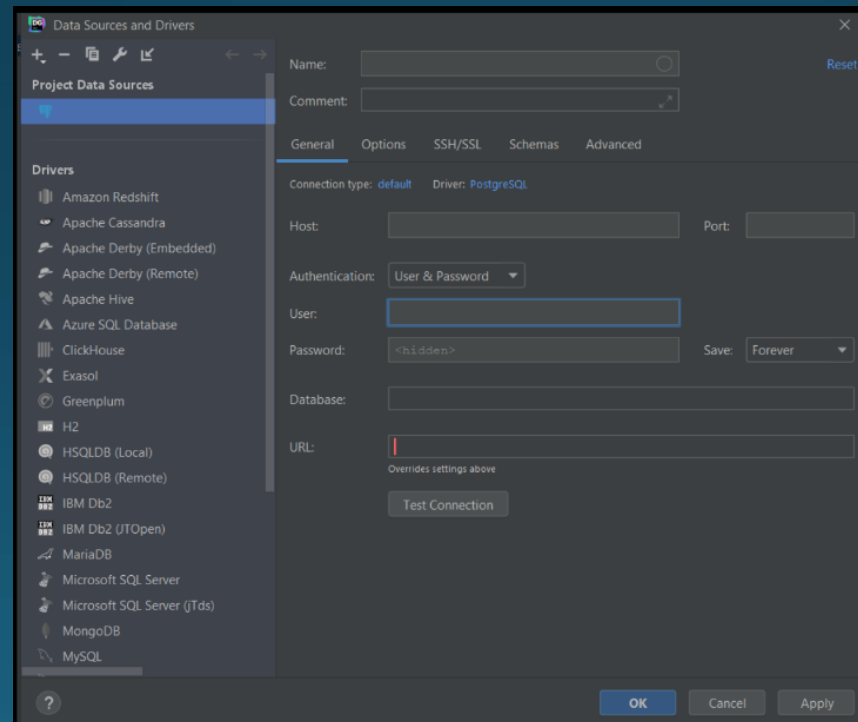
# Heruko

- Destinamos la plataforma Heruko como servicio de computación en la nube para la creación de nuestra base de datos.

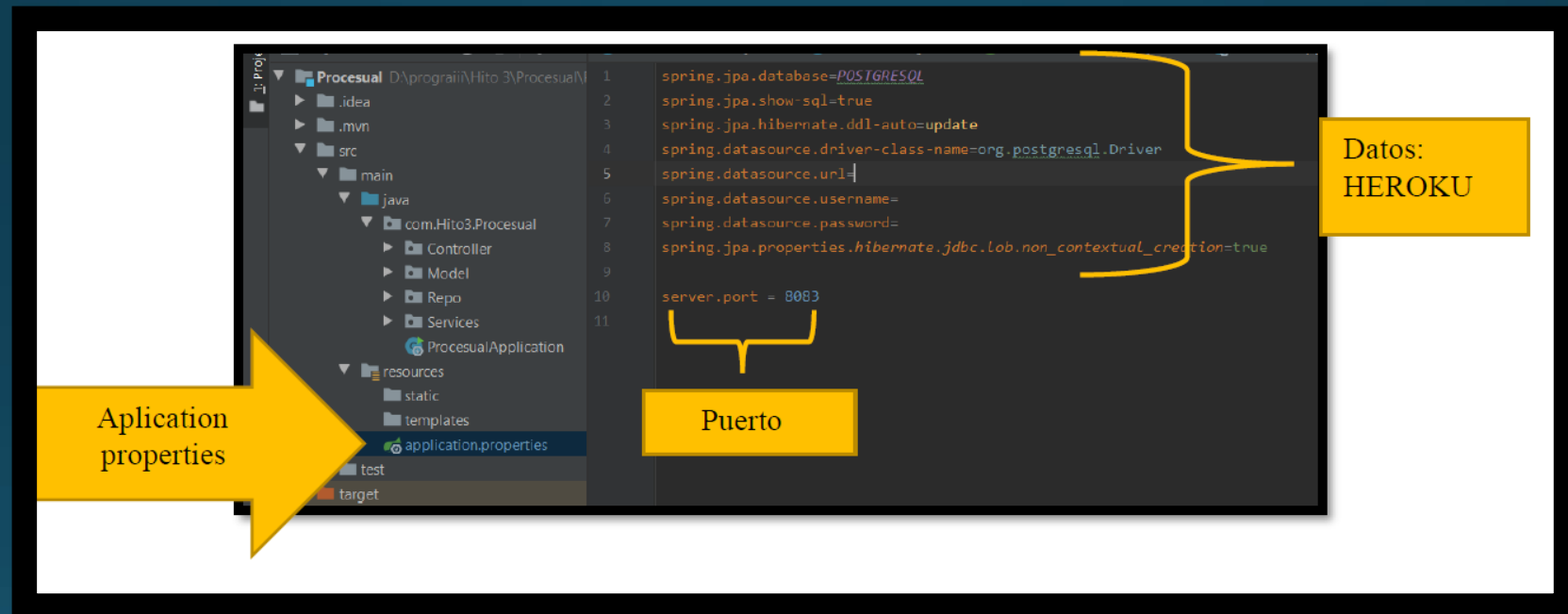


# Base de datos en DataGrip

- Creamos la base de datos y la sincronizamos con el Heruko mediante los datos asignados por el Heruko.



# Conexión del jetbrains intellij idea de igual forma que la base de datos.



# Añadimos las dependencias

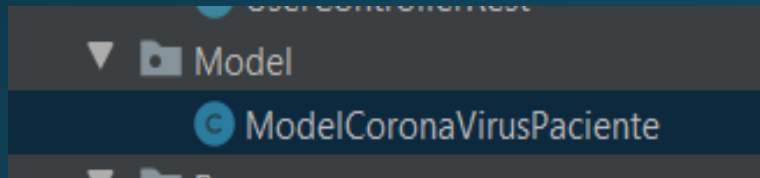
```
16
17 <properties>
18   <java.version>1.8</java.version>
19 </properties>
20
21 <dependencies>
22   <dependency>
23     <groupId>org.springframework.boot</groupId>
24     <artifactId>spring-boot-starter-data-jpa</artifactId>
25   </dependency>
26   <dependency>
27     <groupId>org.postgresql</groupId>
28     <artifactId>postgresql</artifactId>
29     <scope>runtime</scope>
30   </dependency>
31   <dependency>
32     <groupId>org.springframework.boot</groupId>
33     <artifactId>spring-boot-starter-thymeleaf</artifactId>
34   </dependency>
35   <dependency>
36     <groupId>org.springframework.boot</groupId>
37     <artifactId>spring-boot-starter-web</artifactId>
38   </dependency>
39
40   <dependency>
41     <groupId>org.springframework.boot</groupId>
42     <artifactId>spring-boot-starter-test</artifactId>
43     <scope>test</scope>
44     <exclusions>
```

Dependencias  
necesarias



# Solución: Pregunta 1

- Creamos el Package Model y la clase ModelCoronaVirusPaciente (Sera la tabla para la base de datos)



**Id, auto incrementable**

**Columnas de la tabla y variables**

- Creamos la tabla y las columnas.

**Nombre de la tabla**

```
1 package com.Hito3.Defensa.Model;
2 import javax.persistence.*;
3 @Entity
4 @Table(name = "corona_virus_paciente")
5 public class ModelCoronaVirusPaciente {
6     @Id
7     @GeneratedValue(strategy = GenerationType.AUTO)
8     private int id_corona_virus;
9     @Column(name = "nombre_dep", length = 50, nullable = false)
10    private String nombre_dep;
11    @Column(name = "nombre_paciente", length = 50, nullable = false)
12    private String nombre_paciente;
13    @Column(name = "apellido_paciente", length = 50, nullable = false)
14    private String apellido_paciente;
15    @Column(name = "edad_paciente", length = 50, nullable = false)
16    private Integer edad_paciente;
17    @Column(name = "categoria", length = 50, nullable = false)
18    private String categoria;
19    @Column(name = "fullname", length = 50, nullable = false)
20    private String fullname;
21    @Column(name = "casos_contagiados")
22    private Integer casoscontagiados;
23    @Column(name = "casos_sospechosos")
24    private Integer CasosSospechosos;
25    @Column(name = "casos_recuperados")
26    private Integer CasosRecuperados;
```

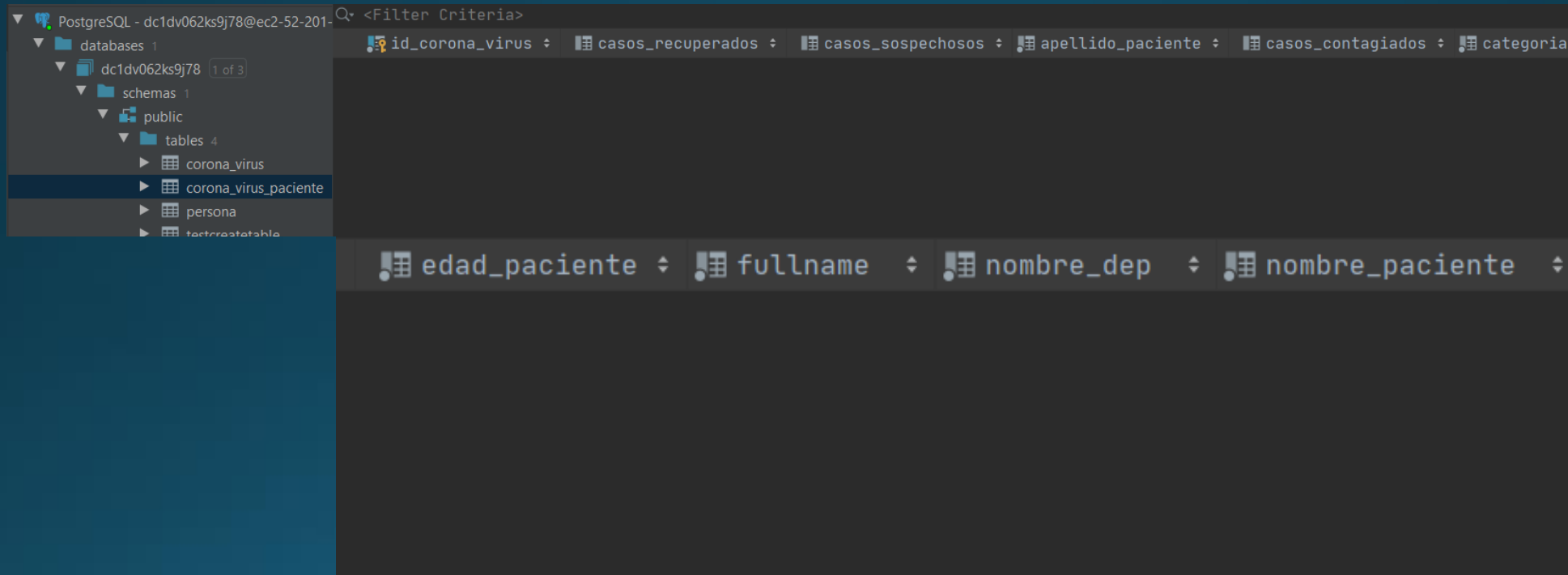
# Solución: Pregunta 1

- Encapsulamiento las variables de la tabla

```
31
32 public void setId_corona_virus(int id_corona_virus) { this.id_corona_virus = id_corona_virus; }
35 public String getNombre_dep() { return nombre_dep; }
38 public void setNombre_dep(String nombre_dep) { this.nombre_dep = nombre_dep; }
41 public String getNombre_paciente() { return nombre_paciente; }
44 public void setNombre_paciente(String nombre_paciente) { this.nombre_paciente = nombre_paciente; }
47 public String getApellido_paciente() { return apellido_paciente; }
50 public void setApellido_paciente(String apellido_paciente) { this.apellido_paciente = apellido_paciente; }
53 public Integer getEdad_paciente() { return edad_paciente; }
56 public void setEdad_paciente(Integer edad_paciente) { this.edad_paciente = edad_paciente; }
59 public String getCategoria() { return categoria; }
62 public void setCategoria(String categoria) { this.categoria = categoria; }
65 public String getFullname() { return fullname; }
68 public void setFullname(String fullname) { this.fullname = fullname; }
71 public Integer getCasoscontagiados() { return casoscontagiados; }
74 public void setCasoscontagiados(Integer casoscontagiados) { this.casoscontagiados = casoscontagiados; }
77 public Integer getCasosSospechosos() { return CasosSospechosos; }
80 public void setCasosSospechosos(Integer casosSospechosos) { CasosSospechosos = casosSospechosos; }
83 public Integer getCasosRecuperados() { return CasosRecuperados; }
86 public void setCasosRecuperados(Integer casosRecuperados) { CasosRecuperados = casosRecuperados; }
89 }
90
```

# Tabla creada

- En DataGrip



# Pregunta dos

2

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

**POST** /coronaVirusPaciente Adds the new case CV in the store

Parameters Try it out

Name	Description
<b>body</b> <small>required</small>	Parametros necesarios para la creacion.
object (body)	Example Value   Model

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

Parameter content type  
application/json

Responses Response content type application/json

Code	Description
201	Created
471	Expectation Failed

Code	Description
------	-------------

object  
(body)

Example Value | Model

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

Parameter content type  
application/json

Responses Response content type application/json

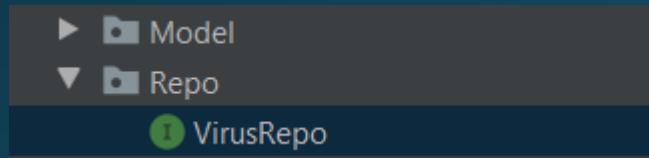
Code	Description
201	Created
471	Expectation Failed

IMPORTANTE: El SERVICE debe de generar las siguientes columnas

categoria	edad > 20	ADULTO
	edad < 20	ADOLESCENTE
	edad < 10	NINO

# Solución: pregunta 2

- Creación del Repo
- Creamos un package Repo
- Y una interfaz.

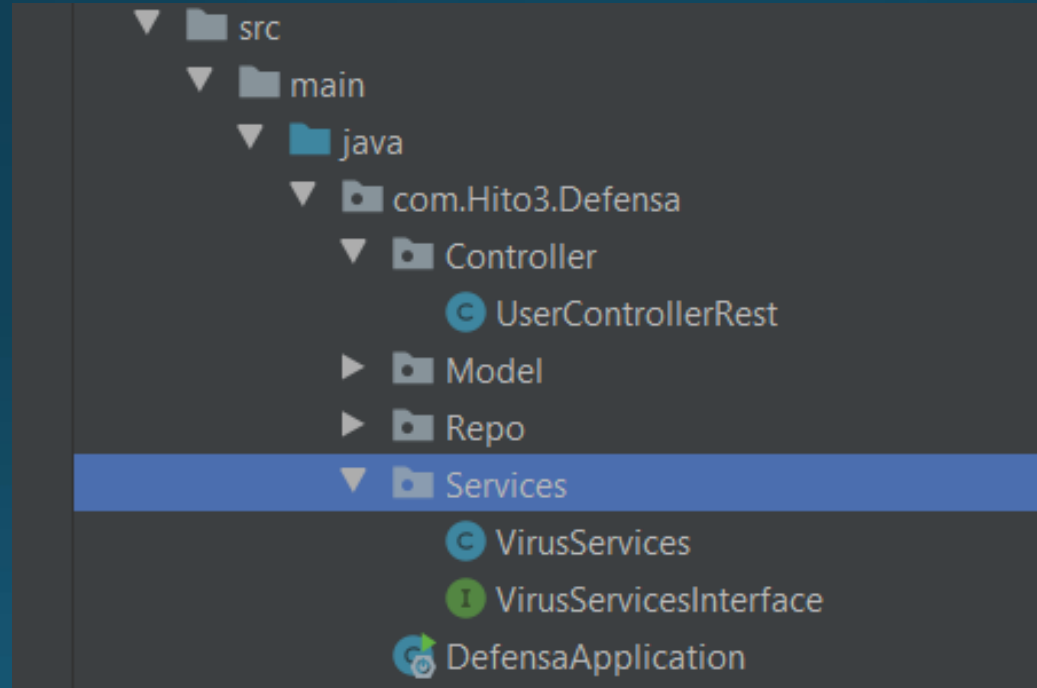


- Código de la interfaz

```
1 package com.Hito3.Defensa.Repo;  
2  
3 import com.Hito3.Defensa.Model.ModelCoronaVirusPaciente;  
4 import org.springframework.data.jpa.repository.JpaRepository;  
5  
6 public interface VirusRepo extends JpaRepository<ModelCoronaVirusPaciente,Integer> {  
7  
8 }
```

# Creamos el Service y Controller

- Creamos los packages, clases y interfaz de Service y Controller

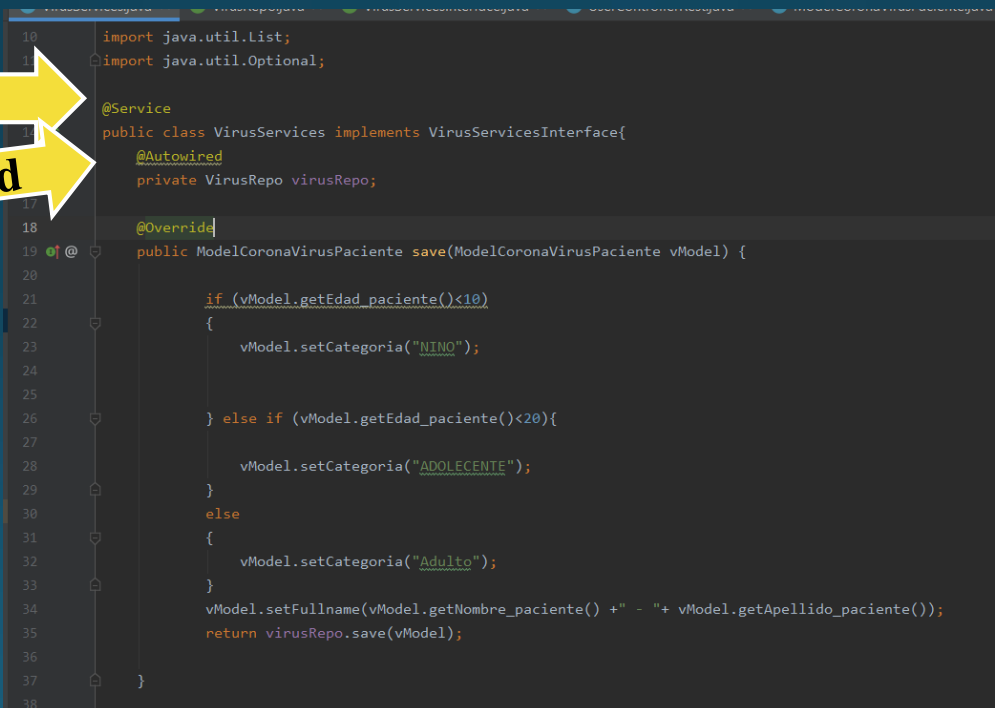


# Interfaz del Service

```
1 package com.Hito3.Defensa.Services;
2
3 import com.Hito3.Defensa.Model.ModelCoronaVirusPaciente;
4
5 import java.util.List;
6
7 public interface VirusServicesInterface {
8     public ModelCoronaVirusPaciente save(ModelCoronaVirusPaciente vModel);
9     public ModelCoronaVirusPaciente saveMayores(ModelCoronaVirusPaciente vModel);
10    public ModelCoronaVirusPaciente update(ModelCoronaVirusPaciente vModel, Integer idCoronaVirus);
11    public Integer delete();
12    public List<ModelCoronaVirusPaciente> getAllDep();
13    public ModelCoronaVirusPaciente getVirusByIdPer(Integer idCoronaVirus);
14 }
15
```

# Clase Service

- Declaramos la clase como un `@service` e implementamos la interface. Usamos el `@Autowired` para llamar a la interfaz `VirusRepo`



The screenshot shows a Java code editor with the following code:

```
10 import java.util.List;
11 import java.util.Optional;
12
13 @Service
14 public class VirusServices implements VirusServicesInterface{
15     @Autowired
16     private VirusRepo virusRepo;
17
18     @Override
19     public ModelCoronaVirusPaciente save(ModelCoronaVirusPaciente vModel) {
20
21         if (vModel.getEdad_paciente() < 10)
22         {
23             vModel.setCategoria("NINO");
24
25
26         } else if (vModel.getEdad_paciente() < 20){
27
28             vModel.setCategoria("ADOLECENTE");
29         }
30         else
31         {
32             vModel.setCategoria("Adulto");
33         }
34         vModel.setFullname(vModel.getNombre_paciente() + " - " + vModel.getApellido_paciente());
35         return virusRepo.save(vModel);
36     }
37 }
38
```

Two yellow arrows point to the annotations in the code:

- A yellow arrow points to the `@Service` annotation on line 13.
- A yellow arrow points to the `@Autowired` annotation on line 15.



- Especificamos las características según la edad y concatenamos el nombre y apellido.

**Condición para la  
tabla categoría:**

	edad > 20	ADULTO
categoría	edad < 20	ADOLESCENTE
	edad < 10	NINO

**Concatenación**

```
17
18 @Override
19 public ModelCoronaVirusPaciente save(ModelCoronaVirusPaciente vModel) {
20
21     if (vModel.getEdad_paciente() < 10)
22     {
23         vModel.setCategoria("NINO");
24     } else if (vModel.getEdad_paciente() < 20) {
25
26         vModel.setCategoria("ADOLESCENTE");
27     }
28     else
29     {
30         vModel.setCategoria("Adulto");
31     }
32
33     vModel.setFullname(vModel.getNombre_paciente() + " - " + vModel.getApellido_paciente());
34     return virusRepo.save(vModel);
35 }
```

# Clase Controller

- Usa el verbo Post

Señalamos el RestController

Controlamos con un try catch y llamamos a la clase mostrando un mensaje de aprobación HttpStatus o en el caso de error un mensaje de error

Dirección URL

Llamamos al Save mediante el URL

```
@RestController
@RequestMapping(value = "/api/v2/")
public class UserControllerRest {
    @Autowired
    private VirusServices virusServices;

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

# Ejecución en Postman

URL

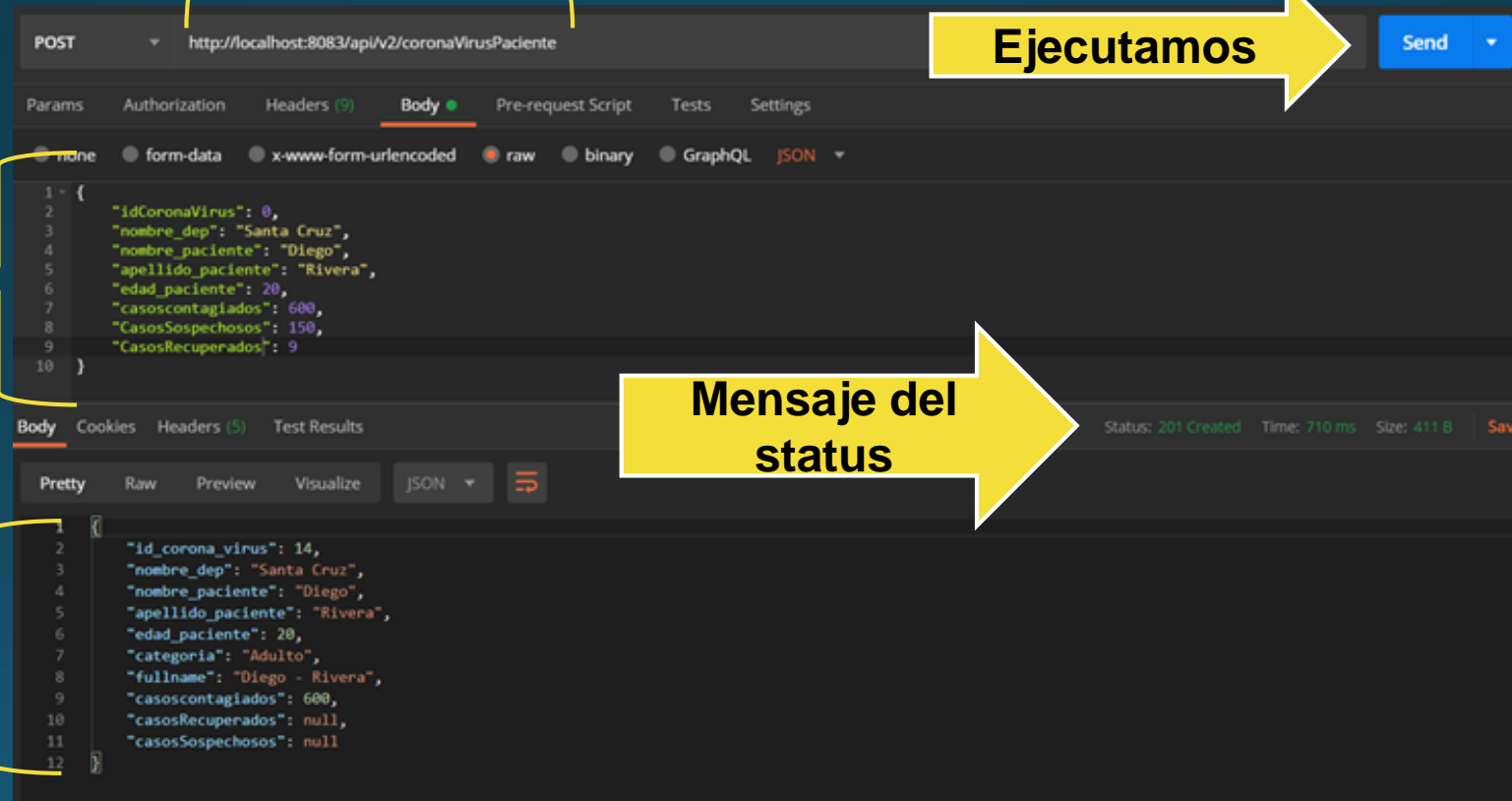
Verbo

Ejecutamos

Datos  
ingresados

Datos  
guardados

Mensaje del  
status



# Tabla actualizada

casos	apellido_paciente	casos_contagiados	categoria	edad_paciente	fullname	nombre_dep	nombre_paciente
	Rivera	600	Adulto	20	Diego - Rivera	Santa Cruz	Diego

# Pregunta tres

3

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

The image shows a Swagger UI interface for a REST API endpoint. The endpoint is `GET /coronaVirusPaciente/findOne/{idCoronaVirus}` with a description "Find CV row by ID". The response is described as "Returns a single CV". The parameters section shows a required path parameter `idCoronaVirus` of type `Integer($int32)` with a description "ID of Corona Virus". The responses section shows a 200 status code with a description "successful operation" and an example JSON response.

**GET** `/coronaVirusPaciente/findOne/{idCoronaVirus}` Find CV row by ID

Returns a single CV

Parameters Try it out

Name	Description
<b>idCoronaVirus</b> <small>* required</small>	ID of Corona Virus
<code>Integer(\$int32)</code> <i>(path)</i>	<code>idCoronaVirus</code> - ID of Corona Virus

Responses Response content type application/json

Code	Description
200	successful operation

Example Value Model

```
{
  "nombreDepartamento": "Cochabamba",
  "nombrePaciente": "Nortejk Tyy",
  "apellidosPaciente": "Nortejk Tyy",
  "edad": 22,
  "casosContagiados": 56,
  "casosSospitosos": 120,
  ...
}
```

# Solución Pregunta 3: Service

- Creamos los servicios getAllDep y getVirusByIdPer. Usando el verbo Get.

**Muestra todos  
los datos  
ingresados**

**Muestra los  
datos de una  
persona por Id**

```
98
99
100 @Override
101 public List<ModelCoronaVirusPaciente> getAllDep() {
102     List<ModelCoronaVirusPaciente> virus = new ArrayList<>();
103     virusRepo.findAll().forEach(virus::add);
104     return virus;
105 }
106
107 @Override
108 public ModelCoronaVirusPaciente getVirusByIdPer(Integer idPer) {
109     Optional<ModelCoronaVirusPaciente> virus = virusRepo.findById(idPer);
110     ModelCoronaVirusPaciente vModel = null;
111     if (virus.isPresent()) {
112         vModel = virus.get();
113     }
114     return vModel;
115 }
116
```

# Solución Pregunta 3: Controller

- Usando el verbo Get. Búsqueda de todos los datos.

Llamamos al Buscar mediante el URL

Controlamos con un try catch y llamamos a la clase mostrando un mensaje de aprobación HttpStatus o en el caso de error un mensaje de error

```
53      @GetMapping("/coronaVirusPaciente")
54      public ResponseEntity<List<ModelCoronaVirusPaciente>> getAllDep() {
55          try {
56              List<ModelCoronaVirusPaciente> persons = virusServices.getAllDep();
57
58              if (persons.isEmpty()) {
59                  return new ResponseEntity<>(HttpStatus.NO_CONTENT);
60              } else {
61                  return new ResponseEntity<>(persons, HttpStatus.OK);
62              }
63          } catch (Exception e) {
64              return new ResponseEntity<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
65          }
66      }
67
```

# Solución Pregunta 3: Controller

- Usando el verbo Get. Búsqueda específica por Id.

Llamamos al Buscar mediante el URL y buscamos por Id

Controlamos con un try catch y llamamos a la clase mostrando un mensaje de aprobación HttpStatus o en el caso de error un mensaje de error

```
77
78
79 @GetMapping("/coronaVirusPaciente/{idPer}")
80 public ResponseEntity<ModelCoronaVirusPaciente> getVirusByIdPer(@PathVariable("idPer") Integer idPer) {
81     try {
82         ModelCoronaVirusPaciente vModel = virusServices.getVirusByIdPer(idPer);
83
84         if (vModel != null) {
85             return new ResponseEntity<>(vModel, HttpStatus.OK);
86         } else {
87             return new ResponseEntity<>(HttpStatus.NOT_FOUND);
88         }
89     } catch (Exception e) {
90         return new ResponseEntity<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
91     }
92 }
93 }
```



# Ejecución en Postman: Todos los casos

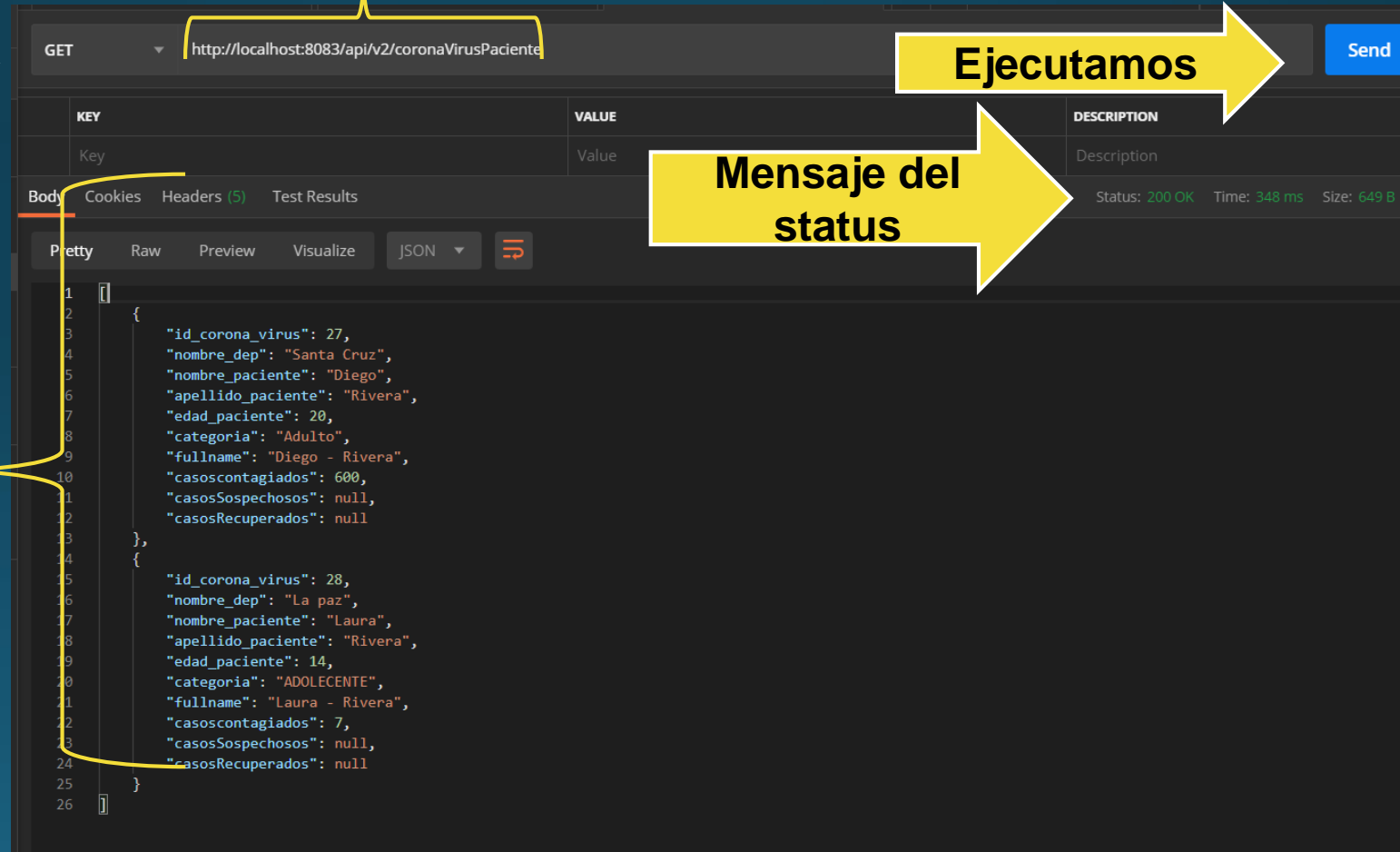
URL

Verbo

Ejecutamos

Mensaje del  
status

DatosMostrados



# Ejecución en Postman: Mostrar caso específico

The image shows a Postman interface with a GET request to `http://localhost:8083/api/v2/coronaVirusPaciente/27`. The request is annotated with yellow boxes and arrows:

- Verbo**: Points to the `GET` method.
- URL**: Points to the request URL.
- Id a buscar**: Points to the `27` in the URL, indicating the specific patient ID to search for.
- Ejecutamos**: Points to the `Send` button.
- Mensaje del status**: Points to the status bar showing `Status: 200 OK`.
- Datos encontrados por Id**: Points to the JSON response body, which contains patient details.

The JSON response body is as follows:

```
1 {
2   "id_corona_virus": 27,
3   "nombre_dep": "Santa Cruz",
4   "nombre_paciente": "Diego",
5   "apellido_paciente": "Rivera",
6   "edad_paciente": 20,
7   "categoria": "Adulto",
8   "fullname": "Diego - Rivera",
9   "casoscontagiados": 600,
10  "casosSospechosos": null,
11  "casosRecuperados": null
12 }
```

# Pregunta cuatro

4

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

**PUT** /coronaVirusPaciente Updates a specific CV row

Parameters [Try it out](#)

Name	Description
<b>body</b> <small>* required</small>	Parametros necesarios para la modificacion
object (body)	Example Value   Model
	<pre>{   "nombredepartamento": "Cochabamba",   "nombrePaciente": "Martejk Tyy",   "apellidosPaciente": "Martejk Tyy",   "edad": 22,   "casoscontagados": 16,   "casosseguichos": 128,   "casosdecurados": 7 }</pre>
	Parameter content type application/json
<b>idCoronaVirus</b> <small>* required</small>	ID of Corona Virus
Integer(\$int32) (path)	idCoronaVirus - ID of Corona Virus

Responses [Response content type](#) application/json

Code	Description
200	Corona Virus updated
404	Corona Virus not found
417	Expectation Failed

Code Description

Integer(\$int32)  
(path)

idCoronaVirus - ID of Corona Virus

Responses

Response content type

application/json

Code

Description

200

Corona Virus updated

404

Corona Virus not found

417

Expectation Failed

Considere que tambien debe de modificar los campos: **FULLNAME y CATEGORIA**

Preguntas abiertas o de respuestas cortas - 15 puntos - Subjetiva

# Solución Pregunta 4: Srevice

- Usando el verbo Put para actualizar los datos.

**Controlamos  
mediante una  
condición la  
actualización de los  
datos categoría**

**Actualizamos la  
concatenación**

```
59  
60  
61 @Override  
62 public ModelCoronaVirusPaciente update(ModelCoronaVirusPaciente vModel, Integer idPer) {  
63     Optional<ModelCoronaVirusPaciente> person = virusRepo.findById(idPer);  
64     ModelCoronaVirusPaciente vUpdate = null;  
65     if (person.isPresent()) {  
66         vUpdate = person.get();  
67         vUpdate.setNombre_dep(vModel.getNombre_dep());  
68         vUpdate.setNombre_paciente(vModel.getNombre_paciente());  
69         vUpdate.setApellido_paciente(vModel.getApellido_paciente());  
70         vUpdate.setEdad_paciente(vModel.getEdad_paciente());  
71         if (vModel.getEdad_paciente() < 10)  
72         {  
73             vUpdate.setCategoria("NINO");  
74  
75         } else if (vModel.getEdad_paciente() < 20){  
76  
77             vUpdate.setCategoria("ADOLECENTE");  
78         }  
79         else  
80         {  
81             vUpdate.setCategoria("Adulto");  
82         }  
83         vUpdate.setFullname(vModel.getNombre_paciente() + " - " + vModel.getApellido_paciente());  
84         vUpdate.setCasoscontagiados(vModel.getCasoscontagiados());  
85         vUpdate.setCasosSospechosos(vModel.getCasosSospechosos());  
86         vUpdate.setCasosRecuperados(vModel.getCasosRecuperados());  
87         virusRepo.save(vUpdate);  
88     }  
89  
90     return vUpdate;  
91 }
```

# Solución Pregunta 4: Controller

- Usando el verbo Put, actualizaremos los datos ingresados por Id.

Controlamos con un try catch y llamamos a la clase mostrando un mensaje de aprobación HttpStatus o en el caso de error un mensaje de error

Llamamos al Modificar mediante el URL y buscamos por Id

```
40 @PutMapping("/coronaVirusPaciente/{idCoronaVirus}")
41 public ResponseEntity<ModelCoronaVirusPaciente> updateVirus(@PathVariable("idCoronaVirus") Integer idCoronaVirus,
42                                                              @RequestBody ModelCoronaVirusPaciente vModel) {
43
44     try {
45         ModelCoronaVirusPaciente vUpdate = virusServices.update(vModel, idCoronaVirus);
46         if (vUpdate != null) {
47             return new ResponseEntity<>(vUpdate, HttpStatus.OK);
48         } else {
49             return new ResponseEntity<>(HttpStatus.NOT_FOUND);
50         }
51     } catch (Exception e) {
52         return new ResponseEntity<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
53     }
54 }
```

# Ejecución en Postman:

Verbo

URL

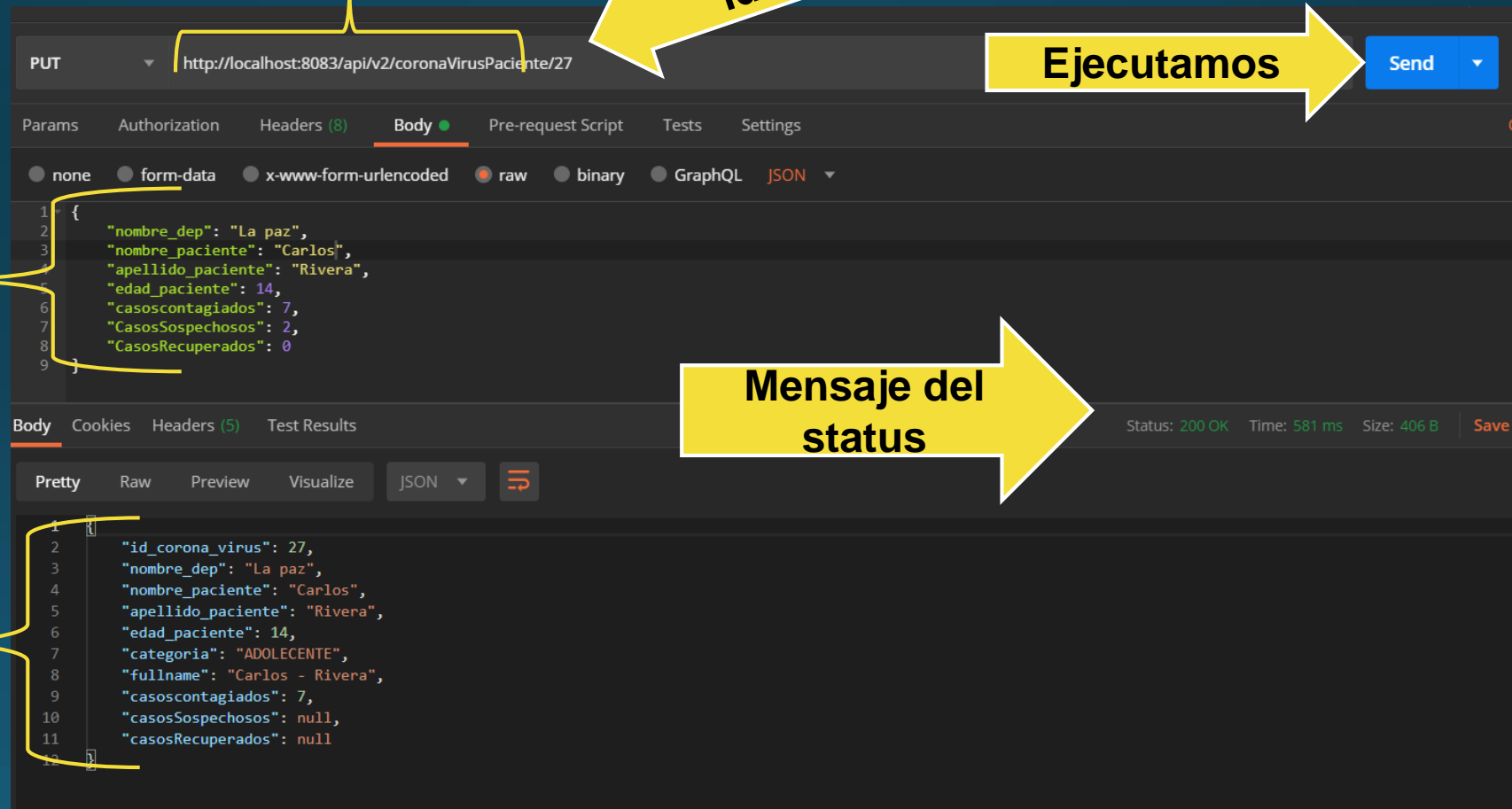
Id a actualizar

Ejecutamos

Datos modificados

Mensaje del status

Cambios de los datos guardados



# Pregunta cinco

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*

# Solución Pregunta 5: Srevice

- Creamos otro servicio Rest saveMayores

**Controlamos  
mediante una  
condición el ingreso  
de personas  
mayores a 70 años**

```
38
39     @Override
40     public ModelCoronaVirusPaciente saveMayores(ModelCoronaVirusPaciente vModel) {
41         if(vModel.getEdad_paciente() < 70) {
42             if (vModel.getEdad_paciente() < 10) {
43                 vModel.setCategoria("NINO");
44             } else if (vModel.getEdad_paciente() < 20) {
45             } else {
46                 vModel.setCategoria("ADOLECENTE");
47             } else {
48                 vModel.setCategoria("Adulto");
49             }
50             vModel.setFullname(vModel.getNombre_paciente() + " - " + vModel.getApellido_paciente());
51             return virusRepo.save(vModel);
52         }
53     }
54     else{
55         return null;
56     }
57 }
58 }
```



# Solución Pregunta 5: Controller

- Usamos el verbo Post para guardar los datos.

Señalamos el RestController

Dirección URL

Controlamos con un try catch y llamamos a la clase mostrando un mensaje de aprobación HttpStatus o en el caso de error un mensaje de error

```
29 @PostMapping("/coronaVirusPacienteMayor")
30 public ResponseEntity saveMayores(@RequestBody ModelCoronaVirusPaciente persona2){
31     try{
32         return new ResponseEntity<>(virusServices.saveMayores(persona2), HttpStatus.EXPECTATION_FAILED);
33     } catch (Exception e)
34     {
35         return new ResponseEntity<>( headers: null ,HttpStatus.EXPECTATION_FAILED);
36     }
37 }
```

# Ejecución en Postman:

URL

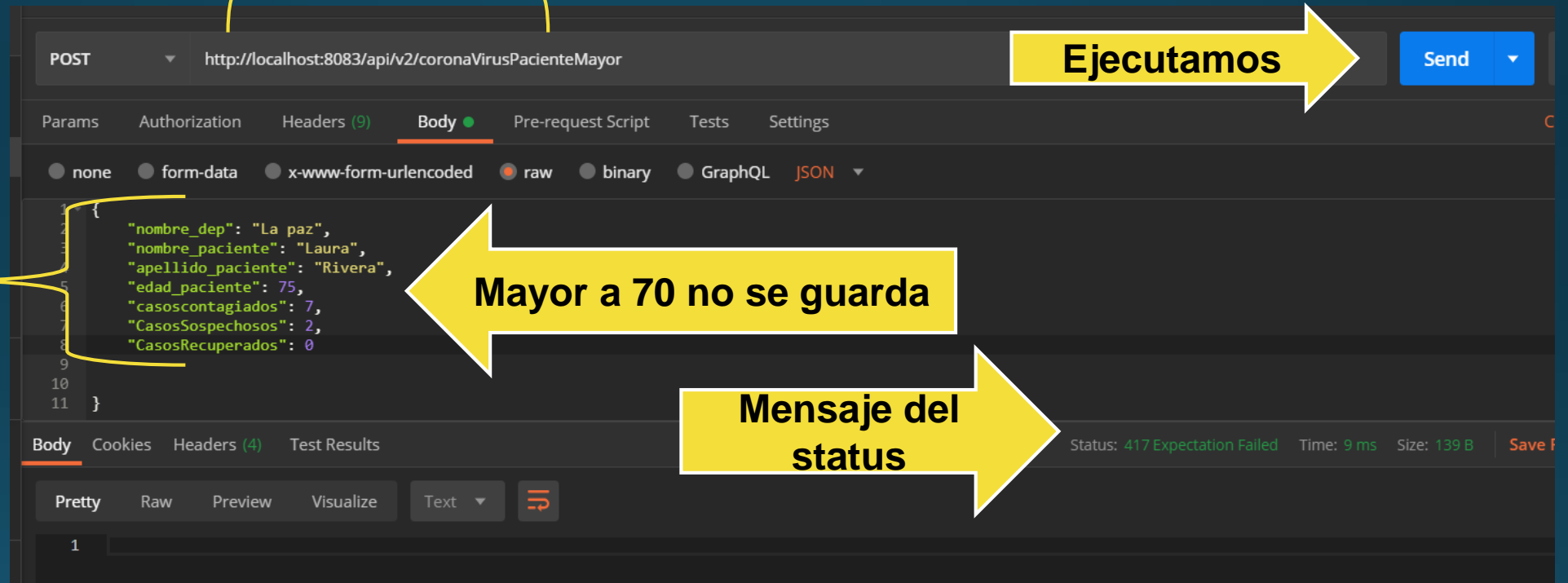
Verbo

Ejecutamos

Datos modificados

Mayor a 70 no se guarda

Mensaje del status



# Pregunta seis

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

# Solución Pregunta 6: Srevice

- Usamos el verbo DELETE para eliminar todos los datos.

**Elimina todos los datos**

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

# Solución Pregunta 6: Controller

- Borra todos los datos guardados.

Controlamos con un try catch y llamamos a la clase mostrando un mensaje de aprobación HttpStatus o en el caso de error un mensaje de error

```
68
69 @DeleteMapping("/coronaVirusPaciente")
70 public ResponseEntity<String> delete() {
71     try {
72         virusServices.delete();
73         return new ResponseEntity<>(body: "ALL Virus successfully deleted", HttpStatus.OK);
74     } catch (Exception e) {
75         return new ResponseEntity<>(null, HttpStatus.EXPECTATION_FAILED);
76     }
77 }
78
```

# Ejecución en Postman:

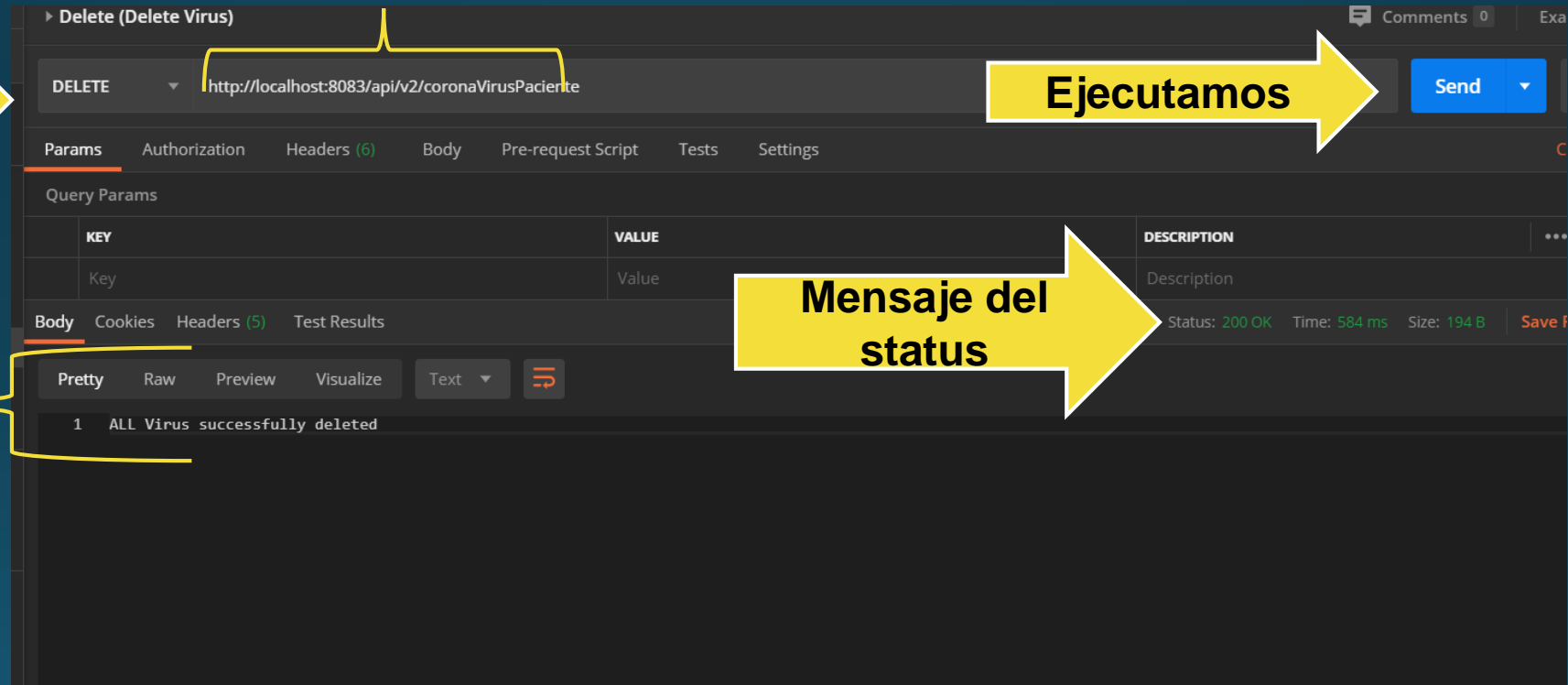
URL

Verbo

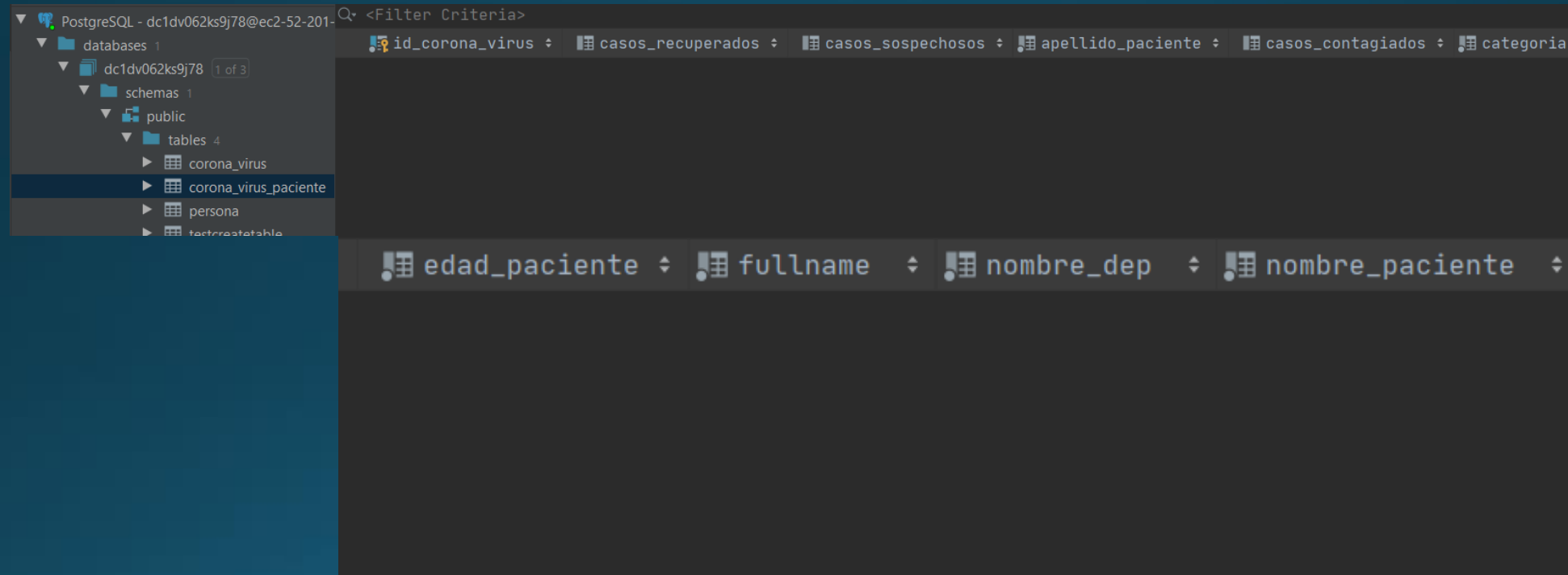
Ejecutamos

Mensaje del  
status

Eliminaodos



# Tabla actualizada



The screenshot displays a PostgreSQL database interface. On the left, a tree view shows the database structure: 'databases' > 'dc1dv062ks9j78' > 'schemas' > 'public' > 'tables' > 'corona\_virus\_paciente'. The main panel shows the table's structure with columns: 'id\_corona\_virus', 'casos\_recuperados', 'casos\_sospechosos', 'apellido\_paciente', 'casos\_contagiados', 'categoria', 'edad\_paciente', 'fullname', 'nombre\_dep', and 'nombre\_paciente'.

id_corona_virus	casos_recuperados	casos_sospechosos	apellido_paciente	casos_contagiados	categoria	edad_paciente	fullname	nombre_dep	nombre_paciente
-----------------	-------------------	-------------------	-------------------	-------------------	-----------	---------------	----------	------------	-----------------

FIN