

DEFENSA HITO 3

PROGRAMACION III - SPRING FRAMEWORK

ÍNDICE

01

SPRING FRAMEWORK

02

DEFENSA HITO3

03

CREAR EL MODELO
"CORONA VIRUS PACIENTE"

04

GENERAR UN SERVICIO REST-POST

05

LISTAR TODOS LOS PACIENTES Y UNO
SOLO REST-GET

06

CREAR SERVICIO REST PUT
PERMITA MODIFICAR

07

EVITAR LA INSERCIÓN DE PACIENTES
MAYORES A 70 AÑOS

08

CREAR UN REST-DELETE QUE
ELIMINE TODOS LOS REGISTROS

09

GRACIAS

01

SPRING FRAMEWORK

HERRAMIENTA PRINCIPAL PARA LA RESOLUCION



SPRING FRAMEWORK

Programacion III

Spring es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java. La primera versión fue escrita por Rod Johnson, quien lo lanzó junto a la publicación de su libro Expert One-on-One J2EE Design and Development

[+ INFO](#)

PROBLEMAS

DEFENSA HIT03

CREAR EL MODELO "CORONA VIRUS PACIENTE"

creara la tabal de nombre"corona virus paciente" con los atributos que especifica el enunciado

corona_virus_pacien	
id corona virus	
nombre_dep	varchar
nombre paciente	varchar
apellidos paciente	varchar
edad paciente	varchar
categoria	varchar
fullname	varchar
casos contagiados	
casos sospechosos	
casos recuperados	

GENERAR UN SERVICIO REST-POST

crear un servicio rest-post para agregar un nuevo registro, las condiciones son: si es mayor a 20 se seteara ADULTO en el campo categoria, si es menor sera ADOLESCENTE y si la edad es menor a 10 sera NIÑO

y para el campo FULLNAME se debera concatenar el nombre y el apellido del usuario

+ INFO

LISTAR TODOS LOS PACIENTES Y UNO SOLO REST-GET

usando el verbo GET debera leer todos los registros ingresados a la base de datos y asi tambien de ser capaz de leer un registro especifico

+ INFO

PROBLEMAS

DEFENSA HITO3

CREAR SERVICIO REST PUT PERMITA MODIFICAR

crear un servicio utilizando el verbo PUT para modificar un registro específico

EVITAR LA INSERCIÓN DE PACIENTES MAYORES A 70 AÑOS

al momento de agregar un nuevo registro de tomar en cuenta que no puede ingresar registros de personas mayores a 70 años

+ INFO

CREAR UN REST- DELETE QUE ELIMINE TODOS LOS REGISTROS

debe eliminar todos los registros que contenga la base de datos sin excepciones

+ INFO

1 RESOLUCION

*creara la tabl a de nombre "corona virus
paciente" con los atributos que especifica
el enunciado*



CODIGO DDL GENERADO EN LA BASE DE DATOS



```
-- auto-generated definition
create table corona_virus
(
  id_corona_virus integer not null
  constraint corona_virus_pkey
  primary key,
  casos_contagiados integer,
  casos_recuperados integer,
  casos_sospechosos integer,
  nombre_dep varchar(50) not null
);
alter table corona_virus
  owner to aecovanrdqbkfr;
```

2 RESOLUCION

crear un servicio rest-post para agregar un nuevo registro, las condiciones son: si es mayor a 20 se seteara ADULTO en el campo categoria, si es menor sera ADOLESCENTE y si la edad es menor a 10 sera NIÑO


y para el campo FULLNAME se debera concatenar el nombre y el apellido del usuario

CODIGO PARA AGREGAR UN NUEVO REGISTRO DENTRO EL "CORONA SERVICE"


```
@Override
public CoronaVirusModel save(CoronaVirusModel pModel) {
    if(pModel.getEdad() > 20){
        pModel.setCategoria("ADULTO");
        String nombrecompleto = (pModel.getNombrepaciente() + " "+pModel.getApellidos());
        pModel.setFullname(nombrecompleto);
    }
    else if(pModel.getEdad() < 20 && pModel.getEdad() > 10 ){
        pModel.setCategoria("Adolecente");
        String nombrecompleto = (pModel.getNombrepaciente() + " "+pModel.getApellidos());
        pModel.setFullname(nombrecompleto);
    }
    if(pModel.getEdad() < 10){
        pModel.setCategoria("ninhio");
        String nombrecompleto = (pModel.getNombrepaciente() + " "+pModel.getApellidos());
        pModel.setFullname(nombrecompleto);
    }
    else {
        pModel.setCategoria("ERROR");
    }
    return coronaRepo.save(pModel);
}
```


2 RESOLUCION

CODIGO PARA AGREGAR UN NUEVO REGISTRO DENTRO DEL "USER CONTROLLER REST"



```
@PostMapping("/coronaVirusPaciente")
public ResponseEntity save(@RequestBody CoronaVirusModel persona) {
    try{
        if(persona.getEdad() > 70)
            persona= null;
        return new ResponseEntity<>(coronaservice.save(persona), HttpStatus.EXPECTATION_FAILED);
    } catch (Exception e) {
        return new ResponseEntity<>(null, HttpStatus.EXPECTATION_FAILED);
    }
}
```



POST

/coronaVirusPaciente Adds the new case CV in the store



4 RESOLUCION

usando el verbo GET debera leer todos los registros ingresados a la base de datos y asi tambien de ser capaz de leer un registro especifico

CODIGO PARA LEER TODOS LOS REGISTROS DENTRO EL "USER CONTROLLER REST"

```
@GetMapping("/coronaVirusPaciente")
public ResponseEntity<List<CoronaVirusModel>>
getAllPersons() {
    try {
        List<CoronaVirusModel> registroCoronaVirus =
        coronaservice.getAllPersons();
        if (registroCoronaVirus.isEmpty()) {
            return new ResponseEntity<>(HttpStatus.NO_CONTENT);
        } else {
            return new ResponseEntity<>(registroCoronaVirus,
            HttpStatus.OK);
        }
    } catch (Exception e) {
        return new ResponseEntity<>(null,
        HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

3 RESOLUCION

CODIGO PARA LEER UN REGISTRO ESPECIFICO DENTRO EL "CORONA SERVICE"


```
@Override
public CoronaVirusModel getPersonByIdPer(Integer idPer) {
    Optional<CoronaVirusModel> person = coronaRepo.findById(idPer);
    CoronaVirusModel pModel = null;
    if (person.isPresent()) {
        pModel = person.get();
    }
    return pModel;
}
```

CODIGO PARA LEER UN REGISTRO ESPECIFICO DENTRO EL "USER C ONTROLLER REST"

```
@GetMapping("/coronaVirusPaciente/getOne/{idCorona}")
public ResponseEntity<CoronaVirusModel>
getPersonByIdPer(@PathVariable("idCorona") Integer idCorona) {
    try {
        CoronaVirusModel pModel =
        coronaservice.getPersonByIdPer(idCorona);
        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);
    }
}
```

3 RESOLUCION

CODIGO PARA LEER TODOS
REGISTROS DENTRO EL "CORONA
SERVICE"



```
@Override
public List<CoronaVirusModel> getAllPersons() {
    List<CoronaVirusModel> persons = new
    ArrayList<CoronaVirusModel>();
    coronaRepo.findAll().forEach(persons::add);
    return persons;
}
```

GET

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



GET


/coronaVirusPaciente/ Gets all cv records



4 RESOLUCION

*crear un servicio utilizando el verbo
PUT para modificar un registro
específico*


CODIGO PARA EDITAR LOS REGISTROS DENTRO EL "USER CONTROLLER REST"



```
@PutMapping("/coronaVirusPaciente/{idCorona}")
public ResponseEntity<CoronaVirusModel>
updateMaterias(@PathVariable("idCorona") Integer idPer,
@RequestBody CoronaVirusModel pModel) {
    try {
        CoronaVirusModel pUpdate = coronaservice.update(pModel,
idPer);///
        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);
    }
}
```


4 RESOLUCION

CODIGO PARA EDITAR LOS REGISTROS DENTRO EL "CORONA SERVICE"



```
@Override
public CoronaVirusModel update(CoronaVirusModel pModel, Integer idPer) {
    Optional<CoronaVirusModel> registroCoronaVirus = coronaRepo.findById(idPer);
    CoronaVirusModel personaUpdate = null;
    if (registroCoronaVirus.isPresent()) {
        personaUpdate = registroCoronaVirus.get();
        personaUpdate.setNombreDep(pModel.getNombreDep());
        personaUpdate.setCasos_contagiados(pModel.getCasos_contagiados());
        personaUpdate.setCasos_sospechosos(pModel.getCasos_sospechosos());
        personaUpdate.setCasos_recuperados(pModel.getCasos_recuperados());
        personaUpdate.setNombrepaciente(pModel.getNombrepaciente());
        personaUpdate.setEdad(pModel.getEdad());
        if (pModel.getEdad() >= 20) {
            personaUpdate.setCategoria("Adulto");
        } else if (pModel.getEdad() <= 20) {
            personaUpdate.setCategoria("Adolecente");
        } else if (pModel.getEdad() <= 10) {
            personaUpdate.setCategoria("ninhio");
        }
        personaUpdate.setFullname(pModel.getFullname());
        personaUpdate.setCategoria(pModel.getCategoria());
        personaUpdate.setApellidos(pModel.getApellidos());
        coronaRepo.save(personaUpdate);
    }
    return personaUpdate;
}
```

PUT

/coronaVirusPaciente/{idCoronaVirus} Updates a specific CV row



5 RESOLUCION

al momento de agregar un nuevo registro de tomar en cuenta que no puede ingresar registros de personas mayores a 70 años

CODIGO QUE EVITA EL REGISTRO DE PERSONAS MAYORES A 70 AÑOS



```
@PostMapping("/coronaVirusPaciente")
public ResponseEntity
nosavemayores(@RequestBody
CoronaVirusModel persona) {
    try{
        return new ResponseEntity<>
(coronaservice.save(persona),
HttpStatus.EXPECTATION_FAILED);
    } catch (Exception e) {
        return new ResponseEntity<>(null,
HttpStatus.EXPECTATION_FAILED);
    }
}
```

//NO CREAR REGISTRO SI ES MAYOR A 70

```
@PostMapping("/coronaVirusPaciente")
public ResponseEntity save(@RequestBody CoronaVirusModel persona){
    try{
        if(persona.getEdad() > 70)
            persona= null;
        return new ResponseEntity<>(coronaservice.save(persona),
HttpStatus.EXPECTATION_FAILED);
    } catch (Exception e) {
        return new ResponseEntity<>(null, HttpStatus.EXPECTATION_FAILED);
    }
}
```

6 RESOLUCION

*debe elimiar todos los registro que
contenga la base de datos sin excepciones*



METODO QUE NO RECIBE NINGUN PARAMETRO DE
ESTA MANERA ELIMINA TODO

```
@Override  
public Integer deleteTodo() {  
    return null;  
}
```

CODIGO QUE ELIMINA TODOS LOS
REGISTROS



```
@DeleteMapping("/CoronaVirusPaciente/deleteTodo")  
public ResponseEntity<String> deleteTodo() {  
    try {  
        coronaservice.deleteTodo();  
        return new ResponseEntity<>("info successfully deleted",  
            HttpStatus.OK);  
    } catch (Exception e) {  
        return new ResponseEntity<>(null,  
            HttpStatus.EXPECTATION_FAILED);  
    }  
}
```

iGracias!

