



UNIVERSIDAD PRIVADA FRANZ TAMAYO

DEFENSA HITO 3 - TAREA FINAL

Nombre Completo: **ROGER CAMACHO VALLEJOS**

Asignatura: **PROGRAMACIÓN III**

Carrera: **INGENIERÍA DE SISTEMAS**

Paralelo: **PROGIII (1)**

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

fecha: **10/05/2020**

github: <https://github.com/RogerCVXD/prograiii/tree/master/HITO3>

Parte Teórica.

1. Preguntas.

Responda de manera breve, clara y concisa posible.

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

Es una interfaz para conectar varios sistemas basados en el protocolo HTTP

```
@GetMapping("/nameApp")
public String nameApp(){
    return NAME_APP;
}

@GetMapping("/saludo")
public String saludo(@RequestParam(value = "name", defaultValue = "World") String name) {
    return userService.GetName(name);
}
```

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

JPA Es la propuesta estándar para persistencia orientada a objetos en Java, mediante **JPA** es posible persistir objetos sin necesidad de crear consultas.

Configurar:

The screenshot shows the Spring Initializr web application interface. It includes a sidebar with a hamburger menu and social media icons (GitHub, Twitter). The main content area is divided into sections for Project, Language, Spring Boot, Project Metadata, and Dependencies. The Project section has radio buttons for Maven Project (selected) and Gradle Project. The Language section has radio buttons for Java (selected), Kotlin, and Groovy. The Spring Boot section has radio buttons for various versions, with 2.2.7 selected. The Project Metadata section has input fields for Group (com.RcvS), Artifact (CoronaFinHito3), Name (CoronaFinHito3), Description (Demo project for Spring Boot), and Package name (com.RcvS.CoronaFinHito3). The Packaging section has radio buttons for Jar (selected) and War. The Java section has radio buttons for versions 14, 11, and 8, with 8 selected. The Dependencies section has a button to add dependencies. At the bottom, there are buttons for GENERATE, EXPLORE, and SHARE.

spring initializr

Project

☒ Maven Project ☐ Gradle Project

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 2.3.0 RC1 ☐ 2.3.0 (SNAPSHOT) ☐ 2.2.8 (SNAPSHOT) ☒ 2.2.7 ☐ 2.1.15 (SNAPSHOT) ☐ 2.1.14

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 14 ☐ 11 ☒ 8

Dependencies

Spring Web ☒ WEB


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



- **Que es MAVEN - POM.**
MAVEN. - Es una herramienta para la gestión y construcción de proyectos java, que se basa en el concepto POM (Project Object Model). Con Maven se pueden generar arquetipos, gestionar librerías, compilar, empaquetar, generar documentación.
POM. - Es un modelo de objeto para un proyecto, o como describe la documentación de Maven
- **Qué son los Spring estereotipos y anotaciones muestre ejemplos.**
Estereotipos. - `@Component`, `@Repository`, `@Service`, `@Controller`.
Anotaciones. - `@Autowired`
- **Describe las características principales de REST.**
Es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la Web.
Es un estilo de arquitectura basado en estándares como son HTTP, URL, la representación de los recursos: XML/HTML/GIF/JPEG/etc. y los tipos MIME: text/xml, text/html.


Parte Práctica.

1. 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.

 DATA

 Datastores >  postgresql-tetrahedral-52646

SERVICE heroku-postgresql PLAN hobby-dev BILLING APP  pro-rcv676

Overview Durability Settings Dataclips

ADMINISTRATION

Database Credentials

Get credentials for manual connections to this database.

Please note that **these credentials are not permanent**.
Heroku rotates credentials periodically and updates applications where this database is attached.

Host	ec2-3-234-169-147.compute-1.amazonaws.com
Database	ddpsu5u3jd63rc
User	mrqmtxrdsdrfmm
Port	5432
Password	387d1259cda0e0679ba21e7926b05765cf928df56a74eee8d887ab0323bba10b
URI	postgres://mrqmtxrdsdrfmm:387d1259cda0e0679ba21e7926b05765cf928df56a74eee8d887ab0323bba10b@ec2-3-234-169-147.compute-1.amazonaws.com:5432/ddpsu5u3jd63rc
Heroku CLI	heroku pg:psql postgresql-tetrahedral-52646 --app pro-rcv676

Reset Database

Reset the database to its originally-provisioned state, deleting all data inside it.

Destroy Database

Destroys the database and all of the data inside it.

```
Build Run Tools VCS Window Help CoronaFinHito3 [C:\Github\programa\HITO3\Tareas\CoronaFinHito3] - ...src\main\resources\application.properties - IntelliJ IDEA
application.properties
CVServiceInterface.java x application.properties x
1 spring.jpa.database=POSTGRESQL
2 spring.jpa.show-sql=true
3 spring.jpa.hibernate.ddl-auto=update
4 spring.datasource.driver-class-name=org.postgresql.Driver
5 #spring.datasource.url=jdbc:postgres://mqmtxrdsdrfmm:387d1259cda0e0679ba21e7926b05765cf928df56a74eee8d887ab0323bba10b@ec2-3-234-169-147.compute-1.amazonaws.com:5432/ddpsu5u3jd63rc
6 spring.datasource.url=jdbc:postgres://ec2-3-234-169-147.compute-1.amazonaws.com:5432/ddpsu5u3jd63rc
7 spring.datasource.username=mqmtxrdsdrfmm
8 spring.datasource.password=387d1259cda0e0679ba21e7926b05765cf928df56a74eee8d887ab0323bba10b
9 spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
10
11 server.port=8083
12
```

2.Spring Framework

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

- **CONTROLLER:**

```
package com.RcvS.CoronaFinHito3.CONTROLLER;

import com.RcvS.CoronaFinHito3.MODEL.CVModel;
import com.RcvS.CoronaFinHito3.SERVICE.CVService;
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 UserControllerCV {
    @Autowired
    private CVService cvService;

    @PostMapping("/coronaVirus")
    public ResponseEntity save(@RequestBody CVModel cvModel){
        try {
            return new ResponseEntity(cvService.save(cvModel),
            HttpStatus.CREATED);
        }catch (Exception e){
            return new ResponseEntity(null, HttpStatus.EXPECTATION_FAILED);
        }
    }

    @PutMapping("/coronaVirus/{IdCV}")
    public ResponseEntity<CVModel> update(@PathVariable("IdCV") Integer IdCV,
```

```

@RequestBody CVModel cvModel){
    try {
        CVModel cUpdat = cvService.update(cvModel, IdCV);
        if (cUpdat != null){
            return new ResponseEntity<>(cUpdat, HttpStatus.OK);
        }else{
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }catch (Exception e){
        return new ResponseEntity<>(null,
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}

@GetMapping("/coronaVirus/getOne/{IdCV}")
public ResponseEntity<CVModel> GetCVByIdCV(@PathVariable("IdCV") Integer
IdCV){
    try {
        CVModel pModel = cvService.GetCVByIdCV(IdCV);
        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);
    }
}

@GetMapping("/coronaVirus/")
public ResponseEntity<List<CVModel>> GetAllCV(){
    try {
        List<CVModel> Coronv = cvService.GetAllCV();
        if (Coronv.isEmpty()){
            return new ResponseEntity<>(HttpStatus.NO_CONTENT);
        }else {
            return new ResponseEntity<>(Coronv, HttpStatus.OK);
        }
    }catch (Exception e){
        return new ResponseEntity<>(null,
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}

@DeleteMapping("/coronaVirus/deleteCV/{IdCV}")
public ResponseEntity<String> delete(@PathVariable("IdCV") Integer IdCV){
    try {
        cvService.delete(IdCV);
        return new ResponseEntity<>("DataBase sccesfully deleted",
HttpStatus.OK);
    }catch (Exception e){
        return new ResponseEntity<>(null, HttpStatus.EXPECTATION_FAILED);
    }
}
}

```

- **MODEL:**

- ```
package com.RcvS.CoronaFinHito3.MODEL;

import javax.persistence.*;

@Entity
@Table(name = "CoronaVirus")
public class CVModel {
 @Id
 @GeneratedValue(strategy = GenerationType.AUTO)
 private Integer IdCV;
 @Column(name = "NombreD", length = 50, nullable = false)
 private String NombreD;
 @Column(name = "CasosContagiados")
 private Integer CasosContagiados;
 @Column(name = "CasosSospechosos")
 private Integer CasosSospechosos;
 @Column(name = "CasosRecuperados")
 private Integer CasosRecuperados;

 public Integer getIdCV() {
 return IdCV;
 }

 public void setIdCV(Integer idCV) {
 this.IdCV = idCV;
 }

 public String getNombreD() {
 return NombreD;
 }

 public void setNombreD(String nombreD) {
 this.NombreD = nombreD;
 }

 public Integer getCasosContagiados() {
 return CasosContagiados;
 }

 public void setCasosContagiados(Integer casosContagiados) {
 this.CasosContagiados = casosContagiados;
 }

 public Integer getCasosSospechosos() {
 return CasosSospechosos;
 }

 public void setCasosSospechosos(Integer casosSospechosos) {
 this.CasosSospechosos = casosSospechosos;
 }

 public Integer getCasosRecuperados() {
 return CasosRecuperados;
 }
}
```

```

 public void setCasosRecuperados(Integer casosRecuperados) {
 this.CasosRecuperados = casosRecuperados;
 }
 }
}

```

- **REPO:**

```

• package com.RcvS.CoronaFinHito3.REPO;

import com.RcvS.CoronaFinHito3.MODEL.CVModel;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CVRepoInterface extends JpaRepository<CVModel, Integer>{
}

```

- **SERVICE:**

```

• package com.RcvS.CoronaFinHito3.SERVICE;

import com.RcvS.CoronaFinHito3.MODEL.CVModel;
import com.RcvS.CoronaFinHito3.REPO.CVRepoInterface;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

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

@Service
public class CVService implements CVServiceInterface{
 @Autowired
 private CVRepoInterface repoInterface;

 @Override
 public CVModel save(CVModel pModel) {
 return repoInterface.save(pModel);
 }

 @Override
 public CVModel update(CVModel cModel, Integer IdCV) {
 Optional<CVModel> corona = repoInterface.findById(IdCV);
 CVModel coronaUpdate = null;
 if (corona.isPresent()){
 coronaUpdate = corona.get();
 coronaUpdate.setNombreD(cModel.getNombreD());
 coronaUpdate.setCasosContagiados(cModel.getCasosContagiados());
 coronaUpdate.setCasosSospechosos(cModel.getCasosSospechosos());
 coronaUpdate.setCasosRecuperados(cModel.getCasosRecuperados());
 repoInterface.save(coronaUpdate);
 }
 return coronaUpdate;
 }

 @Override
 public Integer delete(Integer IdCV) {
 repoInterface.deleteById(IdCV);
 }
}

```

```

 return 1;
 }

 @Override
 public List<CVModel> GetAllCV() {
 List<CVModel> coron = new ArrayList<CVModel>();
 repoInterface.findAll().forEach(coron::add);
 return coron;
 }

 @Override
 public CVModel GetCVByIdCV(Integer IdCV) {
 Optional<CVModel> Corona = repoInterface.findById(IdCV);
 CVModel CModel = null;
 if (Corona.isPresent()){
 CModel = Corona.get();
 }
 return CModel;
 }
}

```

```

package com.RcvS.CoronaFinHito3.SERVICE;

import com.RcvS.CoronaFinHito3.MODEL.CVModel;

import java.util.List;

public interface CVServiceInterface {
 public CVModel save(CVModel pModel);
 public CVModel update(CVModel pModel, Integer IdCV);
 public Integer delete(Integer IdCV);
 public List<CVModel> GetAllCV();
 public CVModel GetCVByIdCV(Integer IdCV);
}

```



### 3.REST – API

```
@PostMapping("/coronaVirus")
public ResponseEntity save(@RequestBody CVModel cvModel){
 try {
 return new ResponseEntity(cvService.save(cvModel), HttpStatus.CREATED);
 }catch (Exception e){
 return new ResponseEntity(null, HttpStatus.EXPECTATION_FAILED);
 }
}
```

```
@PutMapping("/coronaVirus/{IdCV}")
public ResponseEntity<CVModel> update(@PathVariable("IdCV") Integer IdCV,
@RequestBody CVModel cvModel){
 try {
 CVModel cUpdat = cvService.update(cvModel, IdCV);
 if (cUpdat != null){
 return new ResponseEntity<>(cUpdat, HttpStatus.OK);
 }else{
 return new ResponseEntity<>(HttpStatus.NOT_FOUND);
 }
 }catch (Exception e){
 return new ResponseEntity<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
 }
}
```

```
@GetMapping("/coronaVirus/getOne/{IdCV}")
public ResponseEntity<CVModel> GetCVByIdCV(@PathVariable("IdCV") Integer IdCV){
 try {
 CVModel pModel = cvService.GetCVByIdCV(IdCV);
 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);
 }
}
```

```

@GetMapping("/coronaVirus/")
public ResponseEntity<List<CVModel>> GetAllCV(){
 try {
 List<CVModel> Coronv = cvService.GetAllCV();
 if (Coronv.isEmpty()){
 return new ResponseEntity<>(HttpStatus.NO_CONTENT);
 }else {
 return new ResponseEntity<>(Coronv, HttpStatus.OK);
 }
 }catch (Exception e){
 return new ResponseEntity<>(null, HttpStatus.INTERNAL_SERVER_ERROR);
 }
}

```

```

@DeleteMapping("/coronaVirus/deleteCV/{IdCV}")
public ResponseEntity<String> delete(@PathVariable("IdCV") Integer IdCV){
 try {
 cvService.delete(IdCV);
 return new ResponseEntity<>("DataBase sccesfully deleted", HttpStatus.OK);
 }catch (Exception e){
 return new ResponseEntity<>(null, HttpStatus.EXPECTATION_FAILED);
 }
}

```