

Universidad Francisco Gavidia



Facultad de Ingeniería

Carrera: Ingeniería en desarrollo de software

Desarrollo de aplicaciones Web - Grupo 01

Docente: Ing. Carlos Boris Martínez Calzadia

Tema: Parcial 3

Presentan:

| Apellidos | Nombres | Carnet | Participación |
|---------------|-----------------|----------|---------------|
| Ramos Argueta | Gerson Vladimir | RA102011 | 100% |
| Flores Guzmán | Miguel Ángel | FG100220 | 100% |

Sábado 28 de octubre de 2023

Contenido

| | |
|---|----|
| Indicaciones | 3 |
| Paso 1: Diagrama de solución, con su documentación correspondiente. | 3 |
| Paso 2: API Departamentos de El Salvador..... | 5 |
| API Municipios de El Salvador ingresa el Departamento y retorna los municipios | 11 |
| Crear una API que inserte nuevos empleados, apellidos, nombres, géneros, dirección, teléfono. | 15 |
| Crear una API que inserte el código de empleado y el salario en la base de datos debe guardar el salario liquido | 21 |
| Crear un API que inactive el empleado y el motivo | 23 |

Indicaciones

Desarrollar Apis para el sistema de RRHH, la base de datos SQL SERVER, backend Spring Boot 2.7, JDK 11

Parte I 100% Crear una solución que incluya:

10% Diagrama de solución, con su documentación correspondiente.

10% API Departamentos de El Salvador

20% API Municipios de El Salvador ingresa el Departamento y retorna los municipios

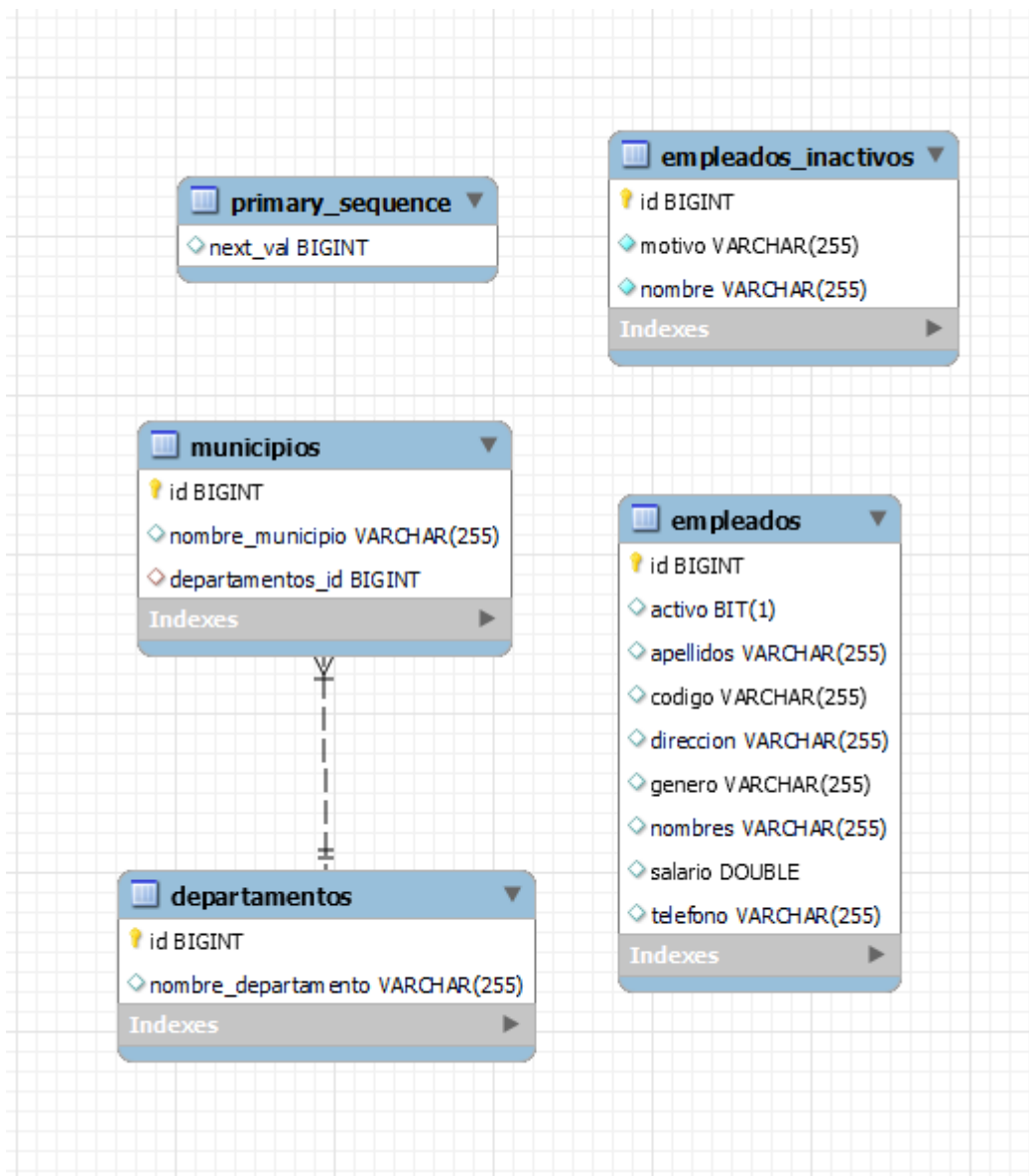
20% Crear una API que inserte nuevos empleados, apellidos, nombres, géneros, dirección, teléfono.

20% Crear una API que inserte el código de empleado y el salario en la base de datos debe guardar el salario liquido

20% Crear un API que inactive el empleado y el motivo

Paso 1: Diagrama de solución, con su documentación correspondiente.

Para la solución se creó un diagrama que contiene tablas de empleado, así como tablas de Municipio con una relación de muchos a uno con Departamento.



Tecnologías Utilizadas:

Java 11 y Spring 2.7.14

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.14</version>
    <relativePath /><!-- lookup parent from repository -->
  </parent>
  <groupId>com.ra</groupId>
  <artifactId>parcial3</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>parcial3</name>

  <properties>
    <java.version>11</java.version>
  </properties>
</project>
```

Paso 2: API Departamentos de El Salvador

Paso 1: Crear modelos

```
1 package com.ra.parcial3.domain;
2
3 > import
16
17
18 @Entity
19 @Getter
20 @Setter
21 public class Departamentos {
22
23     @Id
24     @Column(nullable = false, updatable = false)
25     @SequenceGenerator(
26         name = "primary_sequence",
27         sequenceName = "primary_sequence",
28         allocationSize = 1,
29         initialValue = 10000
30     )
31     @GeneratedValue(
32         strategy = GenerationType.SEQUENCE,
33         generator = "primary_sequence"
34     )
35     private Long id;
36
37     @Column
38     private String nombreDepartamento;
39
40     @OneToMany(mappedBy = "departamentos")
41     private Set<Municipios> municipios;
42
43 }
```

```
1 package com.ra.parcial3.model;
2
3 > import ...
4
5
6
7
8 15 usages Gerson Vladimir Ramos Argueta
9 @Getter
10 @Setter
11 public class DepartamentosDTO {
12     private Long id;
13
14     @Size(max = 255)
15     private String nombreDepartamento;
16
17 }
18
```

Paso 2: Crear repositorio, servicio y controlador.

```
6 usages Gerson Vladimir Ramos Argueta
public interface DepartamentosRepository extends IGenericRepo<Departamentos, Long> {
    1 usage Gerson Vladimir Ramos Argueta
    boolean existsByNombreDepartamentoIgnoreCase(String nombreDepartamento);
}

```

```

@Service
public class DepartamentosService {

    9 usages
    private final DepartamentosRepository departamentosRepository;

    Gerson Vladimir Ramos Argueta
    public DepartamentosService(final DepartamentosRepository departamentosRepository) {
        this.departamentosRepository = departamentosRepository;
    }

    1 usage Gerson Vladimir Ramos Argueta
    public List<DepartamentosDTO> findAll() {
        final List<Departamentos> departamentos = departamentosRepository.findAll(Sort.by(...property));
        return departamentos.stream().map(departamentos -> mapToDTO(departamentos, new DepartamentosDTO()))
            .collect(Collectors.toList());
    }

    1 usage Gerson Vladimir Ramos Argueta
    public DepartamentosDTO get(final Long id) {
        return departamentosRepository.findById(id).orElseThrow(NotFoundException::new);
    }

```

```

1 usage Gerson Vladimir Ramos Argueta
    public Long create(final DepartamentosDTO departamentosDTO) {
        final Departamentos departamentos = new Departamentos();
        mapToEntity(departamentosDTO, departamentos);
        return departamentosRepository.save(departamentos).getId();
    }

    1 usage Gerson Vladimir Ramos Argueta
    public void update(final Long id, final DepartamentosDTO departamentosDTO) {
        final Departamentos departamentos = departamentosRepository.findById(id)
            .orElseThrow(NotFoundException::new);
        mapToEntity(departamentosDTO, departamentos);
        departamentosRepository.save(departamentos);
    }

    1 usage Gerson Vladimir Ramos Argueta
    public void delete(final Long id) { departamentosRepository.deleteById(id); }

    2 usages Gerson Vladimir Ramos Argueta
    private DepartamentosDTO mapToDTO(final Departamentos departamentos,
        final DepartamentosDTO departamentosDTO) {
        departamentosDTO.setId(departamentos.getId());
        departamentosDTO.setNombreDepartamento(departamentos.getNombreDepartamento());
        return departamentosDTO;
    }

```

2 usages  Gerson Vladimir Ramos Argueta

```
private Departamentos mapToEntity(final DepartamentosDTO departamentosDTO,
                                     final Departamentos departamentos) {
    departamentos.setNombreDepartamento(departamentosDTO.getNombreDepartamento());
    return departamentos;
}
```

no usages  Gerson Vladimir Ramos Argueta

```
public boolean nombreDepartamentoExists(final String nombreDepartamento) {
    return departamentosRepository.existsByNombreDepartamentoIgnoreCase(nombreDepartamento);
}
```

1 usage new *

```
public List<MunicipiosDTO> getMunicipiosByDepartamento(Long departamentoId) {
    Departamentos departamento = departamentosRepository.findById(departamentoId)
        .orElseThrow(NotFoundException::new);

    Set<Municipios> municipios = departamento.getMunicipios();
    return municipios.stream() Stream<Municipios>
        .map(this::mapMunicipiosToDTO) Stream<MunicipiosDTO>
        .collect(Collectors.toList());
}
```

1 usage new *

```
private MunicipiosDTO mapMunicipiosToDTO(Municipios municipio) {
    MunicipiosDTO municipioDTO = new MunicipiosDTO();
    municipioDTO.setId(municipio.getId());
    municipioDTO.setNombreMunicipio(municipio.getNombreMunicipio());
    municipioDTO.setDepartamentos(municipio.getDepartamentos().getId());
    return municipioDTO;
}

}
```



```

@RestController
@RequestMapping(value = "/api/departamentos", produces = MediaType.APPLICATION_JSON_VALUE)
public class DepartamentosResource {

    7 usages
    private final DepartamentosService departamentosService;

    Gerson Vladimir Ramos Argueta
    public DepartamentosResource(final DepartamentosService departamentosService) {
        this.departamentosService = departamentosService;
    }

    Gerson Vladimir Ramos Argueta
    @GetMapping
    public ResponseEntity<List<DepartamentosDTO>> getAllDepartamentoss() {
        return ResponseEntity.ok(departamentosService.findAll());
    }

    Gerson Vladimir Ramos Argueta
    @GetMapping("/{id}")
    public ResponseEntity<DepartamentosDTO> getDepartamentos(
        @PathVariable(name = "id") final Long id) {
        return ResponseEntity.ok(departamentosService.get(id));
    }
}

```

```

Gerson Vladimir Ramos Argueta
@PostMapping

public ResponseEntity<Long> createDepartamentos(
    @RequestBody @Valid final DepartamentosDTO departamentosDTO) {
    final Long createdId = departamentosService.create(departamentosDTO);
    return new ResponseEntity<>(createdId, HttpStatus.CREATED);
}

Gerson Vladimir Ramos Argueta
@PutMapping("/{id}")
public ResponseEntity<Long> updateDepartamentos(@PathVariable(name = "id") final Long id,
    @RequestBody @Valid final DepartamentosDTO depart
    departamentosService.update(id, departamentosDTO);
    return ResponseEntity.ok(id);
}

Gerson Vladimir Ramos Argueta
@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteDepartamentos(@PathVariable(name = "id") final Long id) {
    departamentosService.delete(id);
    return ResponseEntity.noContent().build();
}

```

Paso 3: Probar los Endpoints usando Postman.

Parcial 3 APIS / api departamentos get

GET http://localhost:8080/api/departamentos

Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

Body Cookies Headers (5) Test Results 200 OK 261 ms 447 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 10000,
4     "nombreDepartamento": "San Salvador"
5   },
6   {
7     "id": 10001,
8     "nombreDepartamento": "Cabañas"
9   },
10  {
11    "id": 10004,
12    "nombreDepartamento": "La Libertad"
13  },
14  {
15    "id": 10005,
16    "nombreDepartamento": "La Paz"
17  },
18  {
19    "id": 10006,
20    "nombreDepartamento": "San Vicente"
21  },
22 }
```

Parcial 3 APIS / api departamentos post

POST http://localhost:8080/api/departamentos

Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "nombreDepartamento": "Chalatenango"
3 }
```

Body Cookies Headers (5) Test Results 201 Created 12 ms 174 B Save as example

Pretty Raw Preview Visualize JSON

```
1 10008
```

HTTP Parcial 3 APIS / api departamentos put

PUT http://localhost:8080/api/departamentos/10007

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "nombreDepartamento": "La Union"
3 }
```

Body Cookies Headers (5) Test Results 200 OK 56 ms 169 B Save as example

Pretty Raw Preview Visualize JSON

```
1 10007
```

HTTP Parcial 3 APIS / api departamentos delete

DELETE http://localhost:8080/api/departamentos/10007

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| | Key | Value | Description | ... | Bulk Edit |
|--|-----|-------|-------------|-----|-----------|
| | Key | Value | Description | | |

Body Cookies Headers (3) Test Results 204 No Content 62 ms 112 B Save as example

Pretty Raw Preview Visualize Text

```
1
```

API Municipios de El Salvador ingresa el Departamento y retorna los municipios

Paso1: Creamos el servicio

```
1 usage new *
public List<MunicipiosDTO> getMunicipiosByDepartamento(Long departamentoId) {
    Departamentos departamento = departamentosRepository.findById(departamentoId)
        .orElseThrow(NotFoundException::new);

    Set<Municipios> municipios = departamento.getMunicipios();
    return municipios.stream()
        .map(this::mapMunicipiosToDTO)
        .collect(Collectors.toList());
}
```

Paso 2: En este controlador

```
@RestController
@RequestMapping(value = "/api/departamentos", produces = MediaType.APPLICATION_JSON_VALUE)
public class DepartamentosResource {
```

Paso 3: Creamos el siguiente método utilizando el servicio creado

```
@GetMapping("/{id}/municipios")
public ResponseEntity<List<MunicipiosDTO>> getMunicipiosByDepartamento(
    @PathVariable(name = "id") final Long id) {
    List<MunicipiosDTO> municipios = departamentosService.getMunicipiosByDepartamento(id);
    return ResponseEntity.ok(municipios);
}
```

Parcial 3 APIS / api municipios por departamento

GET http://localhost:8080/api/departamentos/10000/municipios

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| Key | Value | Description | Bulk Edit |
|-----|-------|-------------|-----------|
| Key | Value | Description | |

Body Cookies Headers (5) Test Results 200 OK 10 ms 435 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 10002,
4     "nombreMunicipio": "Santa tecla",
5     "departamentos": 10000
6   },
7   {
8     "id": 10010,
9     "nombreMunicipio": "Apopa",
10    "departamentos": 10000
11  },
12  {
13    "id": 10009,
14    "nombreMunicipio": "Antiguo Cuscatlan",
15    "departamentos": 10000
16  },
17  {
18    "id": 10011,
19    "nombreMunicipio": "Ayutuxtepeque",
```

GET http://localhost:8080/api/departamentos/10001/municipios Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| | Key | Value | Description | ... | Bulk Edit |
|--|-----|-------|-------------|-----|-----------|
| | Key | Value | Description | | |

ody Cookies Headers (5) Test Results 200 OK 94 ms 357 B Save as example

```

1  {
2    {
3      "id": 10013,
4      "nombreMunicipio": "Ilobasco",
5      "departamentos": 10001
6    },
7    {
8      "id": 10012,
9      "nombreMunicipio": "Cinquera",
10     "departamentos": 10001
11   },
12   {
13     "id": 10014,
14     "nombreMunicipio": "Victoria",
15     "departamentos": 10001
16   }
17 }

```

GET http://localhost:8080/api/departamentos/10004/municipios Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| | Key | Value | Description ⓘ | ... | Bulk Edit |
|--|-----|-------|---------------|-----|-----------|
| | Key | Value | Description | | |

Body Cookies Headers (5) Test Results 200 OK 39 ms 367 B Save as example

```

1  {
2    {
3      "id": 10017,
4      "nombreMunicipio": "Ciudad Arce",
5      "departamentos": 10004
6    },
7    {
8      "id": 10015,
9      "nombreMunicipio": "San Matias",
10     "departamentos": 10004
11   },
12   {
13     "id": 10016,
14     "nombreMunicipio": "Quezaltepeque",
15     "departamentos": 10004
16   }
17 }

```

Parcial 3 APIS / api municipios por departamento

Save

GET

http://localhost:8080/api/departamentos/10005/municipios

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

| | Key | Value | Description | ... | Bulk Edit |
|--|-----|-------|-------------|-----|-----------|
| | Key | Value | Description | | |

Body

Cookies

Headers (5)

Test Results

200 OK 44 ms 369 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   {
3     "id": 10020,
4     "nombreMunicipio": "Olocuilta",
5     "departamentos": 10005
6   },
7   {
8     "id": 10019,
9     "nombreMunicipio": "El Rosario",
10    "departamentos": 10005
11  },
12  {
13    "id": 10018,
14    "nombreMunicipio": "Santiago Nonualco",
15    "departamentos": 10005
16  }
17 }
```

Parcial 3 APIS / api municipios por departamento

Save

GET

http://localhost:8080/api/departamentos/10006/municipios

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

| | Key | Value | Description | ... | Bulk Edit |
|--|-----|-------|-------------|-----|-----------|
| | Key | Value | Description | | |

Body

Cookies

Headers (5)

Test Results

200 OK 24 ms 297 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   {
3     "id": 10022,
4     "nombreMunicipio": "Apastepeque",
5     "departamentos": 10006
6   },
7   {
8     "id": 10021,
9     "nombreMunicipio": "Guadalupe",
10    "departamentos": 10006
11  }
12 }
```

Parcial 3 APIS / api municipios por departamento

GET http://localhost:8080/api/departamentos/10008/municipios

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| Key | Value | Description | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description | | |

Body Cookies Headers (5) Test Results 200 OK 23 ms 293 B Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": 10024,
3   "nombreMunicipio": "Cancasque",
4   "departamentos": 10008
5 },
6 {
7   "id": 10023,
8   "nombreMunicipio": "Arcatao",
9   "departamentos": 10008
10 },
11 }
12

```

departamentos X empleados municipios

Properties Data ER Diagram

departamentos Enter a SQL expression to filter results

| | id | nombre_departamento |
|---|--------|---------------------|
| 1 | 10,000 | San Salvador |
| 2 | 10,001 | Cabañas |
| 3 | 10,004 | La Libertad |
| 4 | 10,005 | La Paz |
| 5 | 10,006 | San Vicente |
| 6 | 10,008 | Chalatenango |

Crear una API que inserte nuevos empleados, apellidos, nombres, géneros, dirección, teléfono.

Paso 1: crear los modelos

```
12 usages  Gerson Vladimir Ramos Argueta +1
@Entity
@Getter
@Setter
public class Empleados {

    @Id
    @Column(nullable = false, updatable = false)
    @SequenceGenerator(
        name = "primary_sequence",
        sequenceName = "primary_sequence",
        allocationSize = 1,
        initialValue = 10000
    )
    @GeneratedValue(
        strategy = GenerationType.SEQUENCE,
        generator = "primary_sequence"
    )
    private Long id;

    @Column
    private String apellidos;

    @Column
    private String nombres;

    @Column
    @Enumerated(EnumType.STRING)
    private Genero genero;

    @Column
    private String direccion;

    @Column
    private String telefono;

    @Column
    private Double salario;
```



```
@Getter
@Setter
public class EmpleadosDTO {

    private Long id;

    @Size(max = 255)
    private String apellidos;

    @Size(max = 255)
    private String nombres;

    private Genero genero;

    @Size(max = 255)
    private String direccion;

    @Size(max = 255)
    private String telefono;

    private Double salario;

    private Boolean activo;

    @Size(max = 255)
    private String codigo;

}
```

Paso 2: crear el repositorio y los servicios.

```

@Service
public class EmpleadosService {

    7 usages
    private final EmpleadosRepository empleadosRepository;

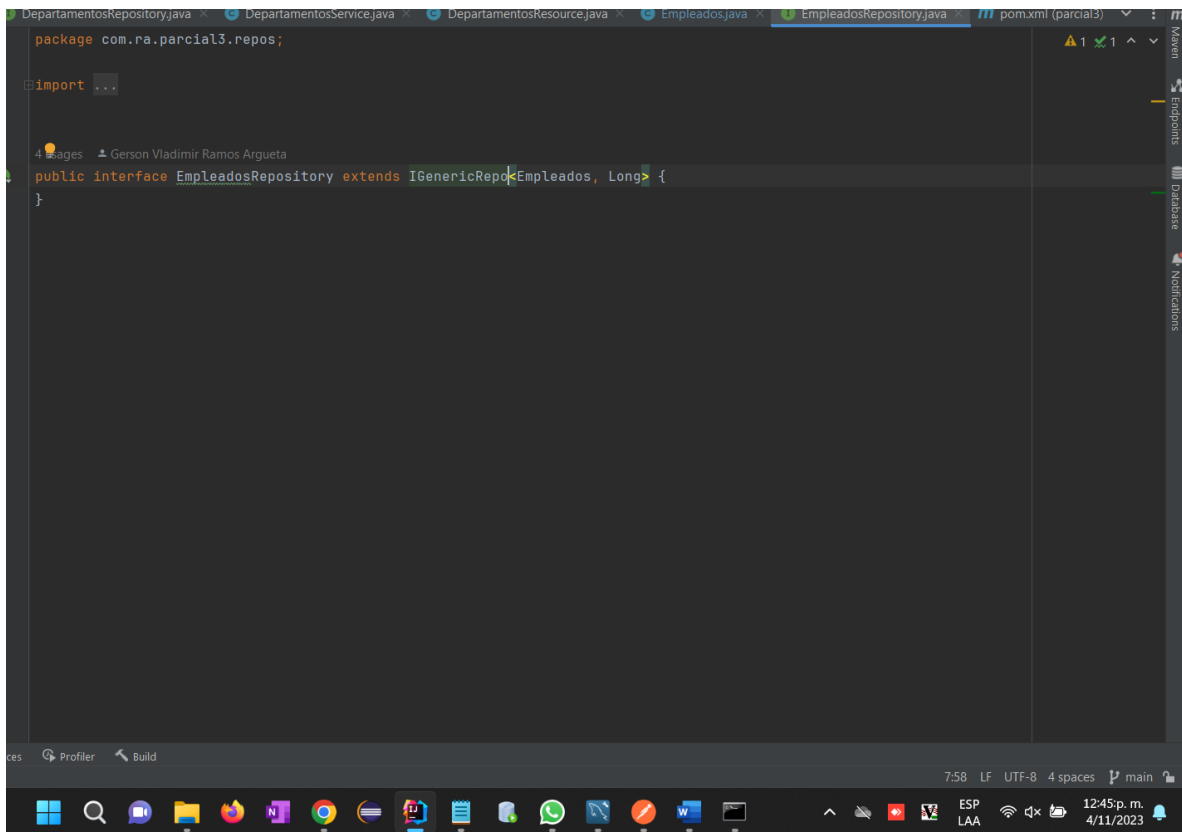
    Gerson Vladimir Ramos Argueta
    public EmpleadosService(final EmpleadosRepository empleadosRepository) {
        this.empleadosRepository = empleadosRepository;
    }

    1 usage Gerson Vladimir Ramos Argueta
    public List<EmpleadosDTO> findAll() {
        final List<Empleados> empleadoses = empleadosRepository.findAll(Sort.by( ...properties: "id"));
        return empleadoses.stream() Stream<Empleados>
            .map(empleados -> mapToDTO(empleados, new EmpleadosDTO())) Stream<EmpleadosDTO>
            .collect(Collectors.toList());
    }

    1 usage Gerson Vladimir Ramos Argueta
    public EmpleadosDTO get(final Long id) {
        return empleadosRepository.findById(id) Optional<Empleados>
            .map(empleados -> mapToDTO(empleados, new EmpleadosDTO())) Optional<EmpleadosDTO>
            .orElseThrow(NotFoundException::new);
    }

    1 usage Gerson Vladimir Ramos Argueta
    public Long create(final EmpleadosDTO empleadosDTO) {
        final Empleados empleados = new Empleados();
        mapToEntity(empleadosDTO, empleados);
        return empleadosRepository.save(empleados).getId();
    }
}

```



```
package com.ra.parcial3.repos;

import ...

public interface EmpleadosRepository extends IGenericRepository<Empleados, Long> {
}
```

Paso 3: Creamos el controlador del empleado con los métodos básicos del mismo y en el método de crear empleado configuramos el código y salario para que estos sean nulos y que los empleados se creen solamente con apellidos, nombres, géneros, dirección, teléfono, etc.

```

@RestController
@RequestMapping(value = "/api/empleados", produces = MediaType.APPLICATION_JSON_VALUE)
public class EmpleadosResource {

    6 usages
    private final EmpleadosService empleadosService;

    Gerson Vladimir Ramos Argueta
    public EmpleadosResource(final EmpleadosService empleadosService) { this.empleadosService = empleadosService; }

    Gerson Vladimir Ramos Argueta
    @GetMapping
    public ResponseEntity<List<EmpleadosDTO>> getAllEmpleados() {
        return ResponseEntity.ok(empleadosService.findAll());
    }

    Gerson Vladimir Ramos Argueta
    @GetMapping("/{id}")
    public ResponseEntity<EmpleadosDTO> getEmpleado(@PathVariable(name = "id") final Long id) {
        return ResponseEntity.ok(empleadosService.get(id));
    }

    Gerson Vladimir Ramos Argueta
    @PutMapping("/{id}")
    public ResponseEntity<Long> updateEmpleado(@PathVariable(name = "id") final Long id,
                                                @RequestBody @Valid final EmpleadosDTO empleadosDTO) {
        empleadosService.update(id, empleadosDTO);
        return ResponseEntity.ok(id);
    }
}

```

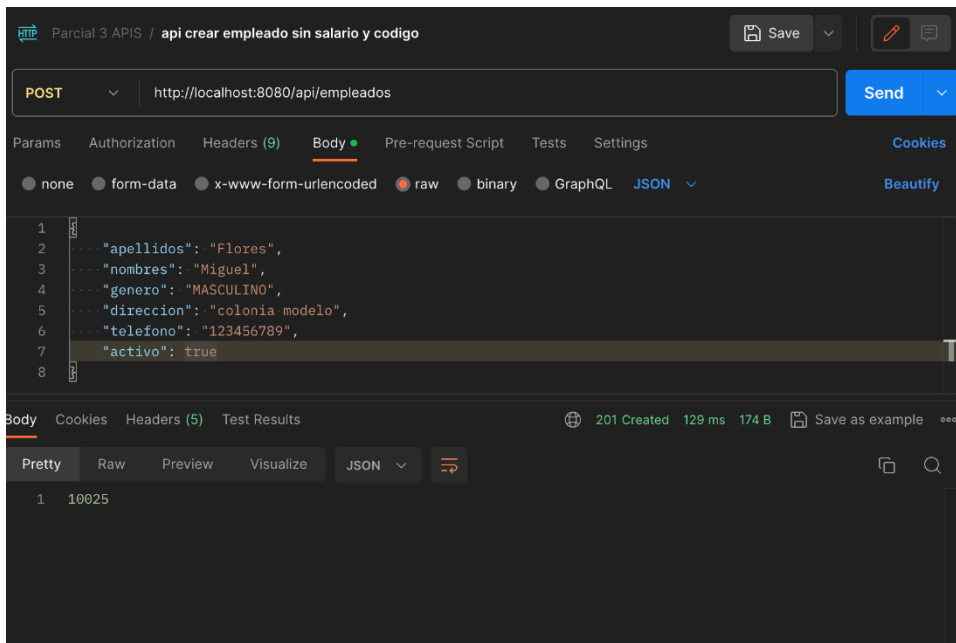
```

new *
@PostMapping
public ResponseEntity<Long> createEmpleado(
    @RequestBody @Valid final EmpleadosDTO empleadosDTO) {
    empleadosDTO.setSalario(null);
    empleadosDTO.setCodigo(null);

    final Long createdId = empleadosService.create(empleadosDTO);
    return new ResponseEntity<>(createdId, HttpStatus.CREATED);
}

```

Paso 4: Probar el api de crear empleado



| empleados | | | | | | | | | |
|-----------|--------|--------|-----------|--------|----------------|-----------|---------|---------|-----------|
| | id | activo | apellidos | codigo | direccion | genero | nombres | salario | telefono |
| 1 | 10,003 | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] |
| 2 | 10,025 | [v] | Flores | [NULL] | colonia modelo | MASCULINO | Miguel | [NULL] | 123456789 |

Crear una API que inserte el código de empleado y el salario en la base de datos debe guardar el salario liquido

Creamos un método dentro del controlador de empleado para asignarle un salario y un código al empleado.

```

@PutMapping("/{id}/asignar-salario-codigo")
public ResponseEntity<EmpleadosDTO> assignSalaryAndCode(
    @PathVariable(name = "id") final Long id,
    @RequestParam Double salario,
    @RequestParam String codigo) {

    EmpleadosDTO empleadosDTO = empleadosService.get(id);

    if (empleadosDTO != null) {
        empleadosDTO.setSalario(calcularSalarioNeto(salario));
        empleadosDTO.setCodigo(codigo);
        empleadosService.update(id, empleadosDTO);
        return ResponseEntity.ok(empleadosDTO);
    } else {
        return ResponseEntity.notFound().build();
    }
}

```

Creamos funciones para calcular el salario neto del empleado antes de guardarlo en la base de datos.

```

//Metodos para calcular el salario neto
1 usage new *
public static double calcularSalarioNeto(double salarioBruto) {
    double renta = calcularRenta(salarioBruto);
    double AFP = calcularAFP(salarioBruto);
    double ISSS = calcularISSS(salarioBruto);

    double salarioNeto = salarioBruto - renta - AFP - ISSS;

    return salarioNeto;
}
1 usage new *
public static double calcularRenta(double salarioBruto) {
    double renta = 0;

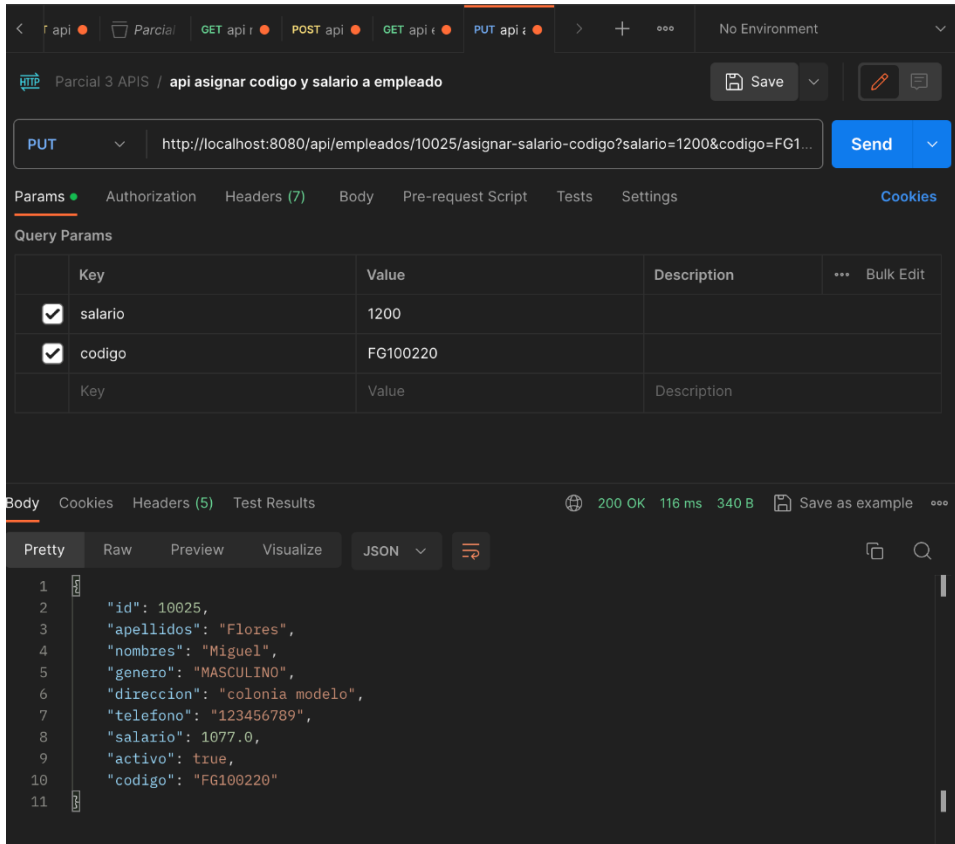
    return renta;
}
1 usage new *
public static double calcularAFP(double salarioBruto) {
    double AFP = salarioBruto * 0.0725;

    return AFP;
}
1 usage new *
public static double calcularISSS(double salarioBruto) {
    double ISSS = salarioBruto * 0.03;

    return ISSS;
}

```

Probamos el api en postman



Comprobamos en la base de datos

The screenshot shows a database client interface with the 'empleados' table selected. The table has the following columns: id, activo, apellidos, codigo, direccion, genero, nombres, salario, and telefono. The data is as follows:

| | id | activo | apellidos | codigo | direccion | genero | nombres | salario | telefono |
|---|--------|--------|-----------|----------|----------------|-----------|---------|---------|-----------|
| 1 | 10,003 | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] |
| 2 | 10,025 | [v] | Flores | FG100220 | colonia modelo | MASCULINO | Miguel | 1,077 | 123456789 |

Crear un API que inactive el empleado y el motivo

Primero creamos una nueva tabla llamada Empleados Inactivos con dos campos nombre y motivo.

```

@Entity
@Getter
@Setter
public class EmpleadosInactivos {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String nombre;

    @Column(nullable = false)
    private String motivo;
}

```

Le creamos un repositorio a dicha clase

```

public interface EmpleadosInactivosRepository extends JpaRepository<EmpleadosInactivos, Long> {
}

```

En el controlador de empleados inyectamos el repositorio EmpleadosInactivosRepository para hacer uso de sus funcionalidades.

```

@RestController
@RequestMapping(value = "/api/empleados", produces = MediaType.APPLICATION_JSON_VALUE)
public class EmpleadosResource {

    10 usages
    private final EmpleadosService empleadosService;
    2 usages
    private final EmpleadosInactivosRepository empleadosInactivosRepository;

    Gerson Vladimir Ramos Argueta *
    public EmpleadosResource(final EmpleadosService empleadosService,
                             final EmpleadosInactivosRepository empleadosInactivosRepository) {
        this.empleadosService = empleadosService;
        this.empleadosInactivosRepository = empleadosInactivosRepository;
    }
}

```

Creamos un nuevo api llamado inactivar Empleado que cambiará el estado del campo 'activo' a false en la tabla Empleados y posteriormente hará una inserción

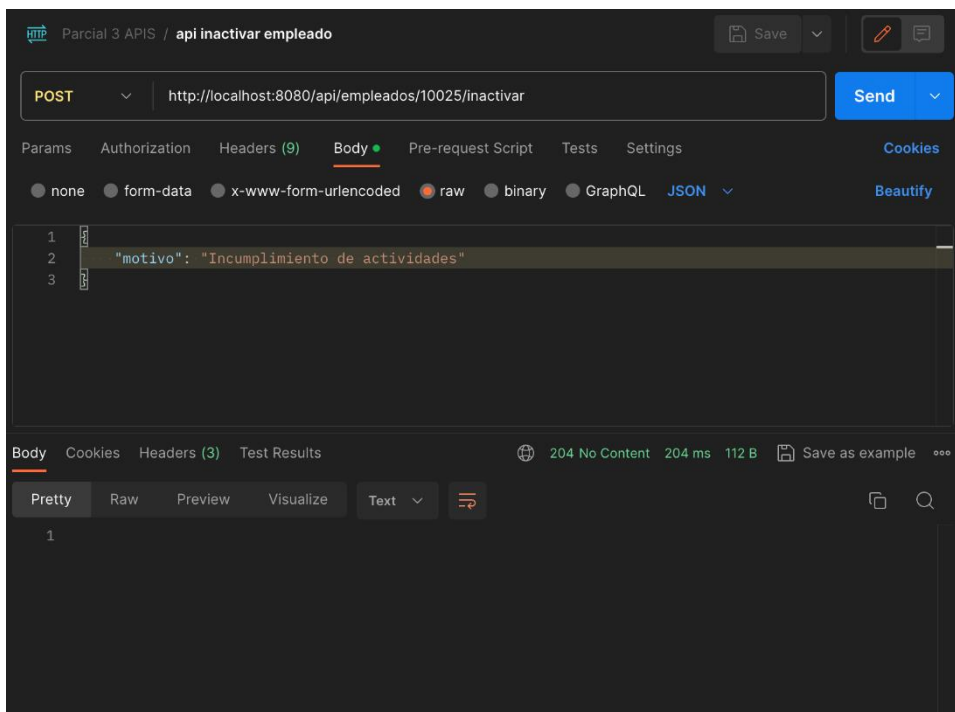
en la tabla empleados Inactivos con el nombre del empleado y el motivo detallado de porque ya no está activo.

```
@PostMapping("/{id}/inactivar")
public ResponseEntity<Void> inactivarEmpleado(
    @PathVariable(name = "id") final Long id,
    @RequestBody Map<String, String> motivoJson) {
    EmpleadosDTO empleadoDTO = empleadosService.get(id);
    if (empleadoDTO != null) {
        empleadoDTO.setActivo(false);
        empleadosService.update(id, empleadoDTO);
        String motivo = motivoJson.get("motivo");

        EmpleadosInactivos empleadoInactivo = new EmpleadosInactivos();
        empleadoInactivo.setNombre(empleadoDTO.getNombres() + " " + empleadoDTO.getApellidos());
        empleadoInactivo.setMotivo(motivo);
        empleadosInactivosRepository.save(empleadoInactivo);

        return ResponseEntity.noContent().build();
    } else {
        return ResponseEntity.notFound().build();
    }
}
```

Validamos la API con postman



Validamos en la base de datos

departamentos empleados X municipios empleados_inactivos

Properties Data ER Diagram

postgres Databases parcial3 Schemas public Tables empleados

empleados Enter a SQL expression to filter results (use Ctrl+Space)

| | id | activo | apellidos | codigo | direccion | genero | nombres | salario | telefono |
|---|--------|-------------------------------------|-----------|----------|----------------|-----------|---------|---------|-----------|
| 1 | 10,003 | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] | [NULL] |
| 2 | 10,025 | <input checked="" type="checkbox"/> | Flores | FG100220 | colonia modelo | MASCULINO | Miguel | 1,077 | 123456789 |

departamentos empleados municipios empleados_inactivos X

Properties Data ER Diagram

postgres Databases parcial3 Schemas public Tables empleados_inactivos

empleados_inactivos Enter a SQL expression to filter results (use Ctrl+Space)

| | id | motivo | nombre |
|---|----|-------------------------------|---------------|
| 1 | 1 | Incumplimiento de actividades | Miguel Flores |

Link al repositorio:

<https://github.com/migueDevUFG/parcial3>