1.Crear una base de datos mysql que conecta mínimo cuatro tablas con mínimo cinco atributos y que haya diferentes tipos de datos. Añadir el modelado de la base de datos

Propietario:

- Atributos:
  - id_propietario (int, clave primaria)
  - nombre (varchar)
  - telefono (varchar)
  - direccion (varchar)
  - correo (varchar)

Mascota:

- Atributos:
  - id_mascota (int, clave primaria)
  - nombre (varchar)
  - edad (int)
  - id_raza (int, clave foránea hacia la tabla Raza)
  - id_propietario (int, clave foránea hacia la tabla Propietario)

Raza:

- Atributos:
  - id_raza (int, clave primaria)
  - nombre (varchar)
  - descripcion (text)
  - origen (varchar)
  - expectativa_vida (int)
  -

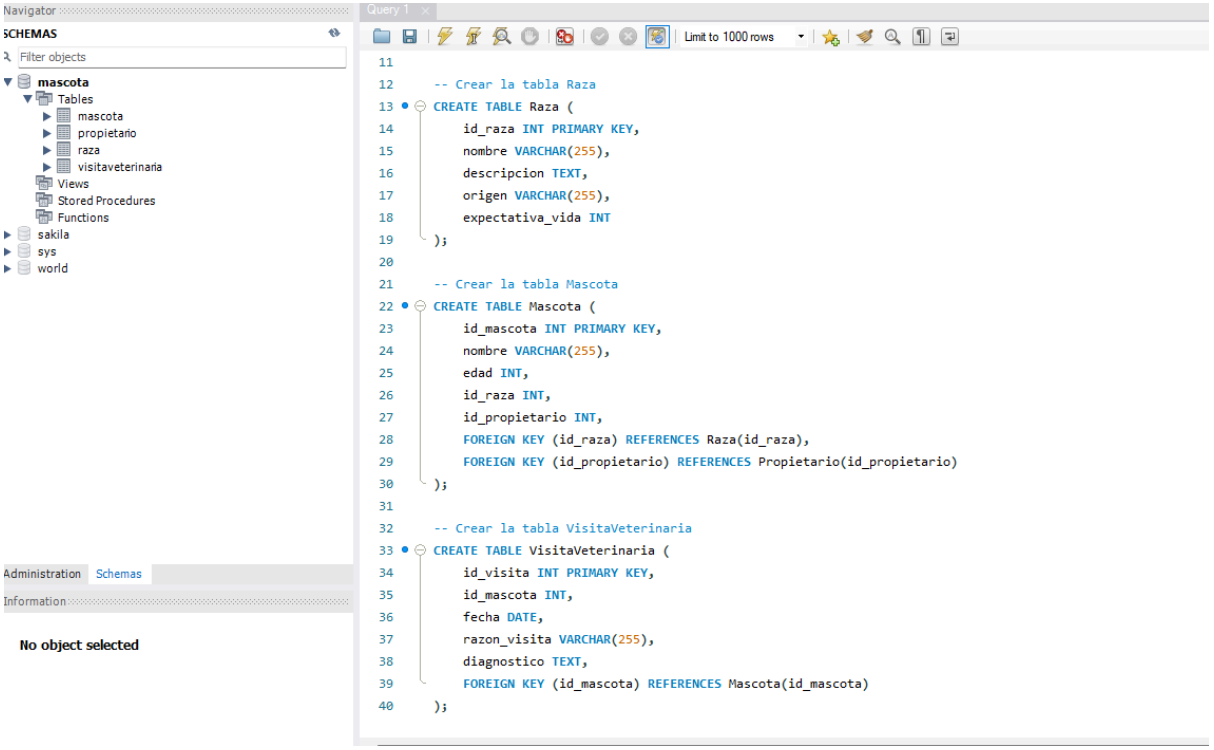
VisitaVeterinaria:

- Atributos:
  - id_visita (int, clave primaria)
  - id_mascota (int, clave foránea hacia la tabla Mascota)
  - fecha (date)
  - razon_visita (varchar)
  - diagnostico (text)

```
CREATE DATABASE IF NOT EXISTS mascota;
USE mascota;
```

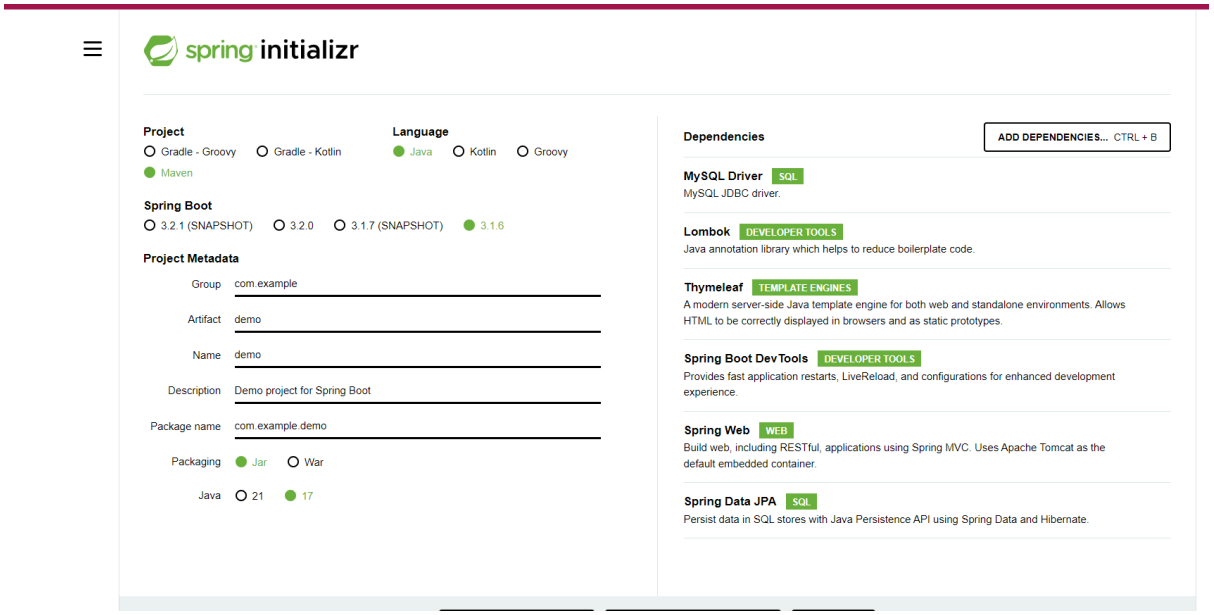
```
CREATE TABLE Propietario (
    id_propietario INT PRIMARY KEY,
    nombre VARCHAR(255),
    telefono VARCHAR(20),
    direccion VARCHAR(255),
```

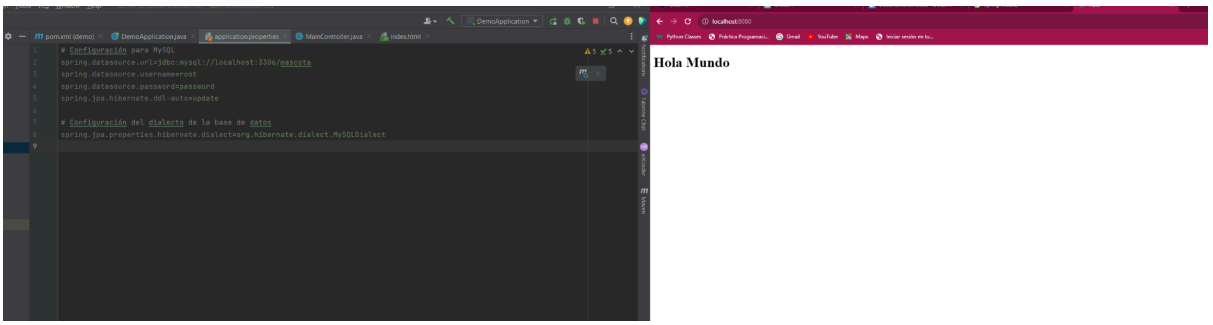
```
    correo VARCHAR(255)
);

-- Crear la tabla Raza
CREATE TABLE Raza (
    id_raza INT PRIMARY KEY,
    nombre VARCHAR(255),
    descripcion TEXT,
    origen VARCHAR(255),
    expectativa_vida INT
);

-- Crear la tabla Mascota
CREATE TABLE Mascota (
    id_mascota INT PRIMARY KEY,
    nombre VARCHAR(255),
    edad INT,
    id_raza INT,
    id_propietario INT,
    FOREIGN KEY (id_raza) REFERENCES Raza(id_raza),
    FOREIGN KEY (id_propietario) REFERENCES Propietario(id_propietario)
);

-- Crear la tabla VisitaVeterinaria
CREATE TABLE VisitaVeterinaria (
    id_visita INT PRIMARY KEY,
    id_mascota INT,
    fecha DATE,
    razon_visita VARCHAR(255),
    diagnostico TEXT,
    FOREIGN KEY (id_mascota) REFERENCES Mascota(id_mascota)
);
```



2. La aplicación deberá conectar con una Base de Datos de forma transparente para el usuario, de forma que los datos de conexión puedan configurarse en un fichero aparte .

Se crea en application.properties y luego se compara en el localhost. Hay que descargarse las dependencias de spring.





3.Se creará en Spring Boot: el modelado de cada tabla de base de datos , el repositorio correspondiente, interfaz y clase de servicio.

El modelado de datos en una aplicación Spring Boot generalmente se realiza mediante la creación de clases de entidad que representan las tablas de la base de datos. Estas clases suelen mapearse directamente a las tablas en la base de datos y contienen campos que representan las columnas de esas tablas. A continuación, proporcionaremos cómo modelar las clases de entidad para los modelos Propietario, Raza, Mascota y VisitaVeterinaria.

PROPIETARIO:

```
package com.example.demo.Models;

import jakarta.persistence.*;

// Propietario.java
@Entity
@Table(name = "propietario")
public class Propietario {

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

    private String nombre;
    private String telefono;
    private String direccion;
    private String correo;

    // Getter y Setter para idPropietario
    public Long getIdPropietario() {
        return idPropietario;
    }

    public void setIdPropietario(Long idPropietario) {
        this.idPropietario = idPropietario;
```

```
    }

    // Getters y Setters para los demás atributos
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getTelefono() {
        return telefono;
    }

    public void setTelefono(String telefono) {
        this.telefono = telefono;
    }

    public String getDireccion() {
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    public String getCorreo() {
        return correo;
    }

    public void setCorreo(String correo) {
        this.correo = correo;
    }
}
```

```
package com.example.demo.Repositories;


import com.example.demo.Models.Propietario;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PropietarioRepository extends
JpaRepository<Propietario, Long> {
        // Puedes agregar consultas personalizadas si es necesario
```

```
    }
```

```
package com.example.demo.Services;

import com.example.demo.Models.Propietario;

import java.util.List;

public interface PropietarioService {
    List<Propietario> getAllPropietarios();

    Propietario getPropietarioById(Long id);

    Propietario createPropietario(Propietario propietario);

    Propietario updatePropietario(Long id, Propietario propietario);

    void deletePropietario(Long id);
}
```

```
package com.example.demo.Services;

import com.example.demo.Models.Propietario;
import com.example.demo.Repositories.PropietarioRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

// PropietarioServiceImpl.java
@Service
public class PropietarioServiceImpl implements PropietarioService
{

    @Autowired
    private PropietarioRepository propietarioRepository;

    @Override
    public List<Propietario> getAllPropietarios() {
        return propietarioRepository.findAll();
    }

    @Override
```

```
    public Propietario getPropietarioById(Long id) {
        return propietarioRepository.findById(id).orElse(null);
    }

    @Override
    public Propietario createPropietario(Propietario propietario) {
        return propietarioRepository.save(propietario);
    }

    @Override
    public Propietario updatePropietario(Long id, Propietario
propietario) {
        if (propietarioRepository.existsById(id)) {
            propietario.setIdPropietario(id);
            return propietarioRepository.save(propietario);
        }
        return null;
    }

    @Override
    public void deletePropietario(Long id) {
        propietarioRepository.deleteById(id);
    }
}
```

MASCOTA

```
package com.example.demo.Models;


import jakarta.persistence.*;

@Entity
@Table(name = "mascota")
public class Mascota {

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

    private String nombre;
    private int edad;

    @ManyToOne
    @JoinColumn(name = "id_raza")
    private Raza raza;
```

```java
    @ManyToOne
    @JoinColumn(name = "id_propietario")
    private Propietario propietario;

    // Getter y Setter para idMascota
    public Long getIdMascota() {
        return idMascota;
    }

    public void setIdMascota(Long idMascota) {
        this.idMascota = idMascota;
    }

    // Getters y Setters para los demás atributos
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
    }

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

    public Raza getRaza() {
        return raza;
    }

    public void setRaza(Raza raza) {
        this.raza = raza;
    }

    public Propietario getPropietario() {
        return propietario;
    }

    public void setPropietario(Propietario propietario) {
        this.propietario = propietario;
    }
}
```

```java
package com.example.demo.Repositories;

import com.example.demo.Models.Mascota;
import org.springframework.data.jpa.repository.JpaRepository;


public interface MascotaRepository extends JpaRepository<Mascota,
Long> {
    // Puedes agregar consultas personalizadas si es necesario
}
```

```java
package com.example.demo.Services;


import com.example.demo.Models.Mascota;

import java.util.List;

public interface MascotaService {
    List<Mascota> getAllMascotas();

    Mascota getMascotaById(Long id);

    Mascota createMascota(Mascota mascota);

    Mascota updateMascota(Long id, Mascota mascota);

    void deleteMascota(Long id);
}
```

```java
package com.example.demo.Services;


import com.example.demo.Models.Mascota;
import com.example.demo.Repositories.MascotaRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class MascotaServiceImpl implements MascotaService {

    @Autowired
    private MascotaRepository mascotaRepository;
```

```
    @Override
    public List<Mascota> getAllMascotas() {
        return mascotaRepository.findAll();
    }

    @Override
    public Mascota getMascotaById(Long id) {
        return mascotaRepository.findById(id).orElse(null);
    }

    @Override
    public Mascota createMascota(Mascota mascota) {
        return mascotaRepository.save(mascota);
    }

    @Override
    public Mascota updateMascota(Long id, Mascota mascota) {
        if (mascotaRepository.existsById(id)) {
            mascota.setIdMascota(id);
            return mascotaRepository.save(mascota);
        }
        return null;
    }

    @Override
    public void deleteMascota(Long id) {
        mascotaRepository.deleteById(id);
    }
}
```

RAZA

```
package com.example.demo.Models;

import jakarta.persistence.*;

// Raza.java
@Entity
@Table(name = "raza")
public class Raza {

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

```
    private String nombre;
    private String descripcion;
    private String origen;
    private int expectativaVida;

    // Getter y Setter para idRaza
    public Long getIdRaza() {
        return idRaza;
    }

    public void setIdRaza(Long idRaza) {
        this.idRaza = idRaza;
    }

    // Getters y Setters para los demás atributos
    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getDescripcion() {
        return descripcion;
    }

    public void setDescripcion(String descripcion) {
        this.descripcion = descripcion;
    }

    public String getOrigen() {
        return origen;
    }

    public void setOrigen(String origen) {
        this.origen = origen;
    }

    public int getExpectativaVida() {
        return expectativaVida;
    }

    public void setExpectativaVida(int expectativaVida) {
        this.expectativaVida = expectativaVida;
    }
}
```

```java
package com.example.demo.Repositories;

import com.example.demo.Models.Raza;
import org.springframework.data.jpa.repository.JpaRepository;

// RazaRepository.java
public interface RazaRepository extends JpaRepository<Raza, Long>
{
    // Puedes agregar consultas personalizadas si es necesario
}
```

```java
package com.example.demo.Services;

import com.example.demo.Models.Raza;

import java.util.List;

// RazaService.java
public interface RazaService {
    List<Raza> getAllRazas();
    Raza getRazaById(Long id);
    Raza createRaza(Raza raza);
    Raza updateRaza(Long id, Raza raza);
    void deleteRaza(Long id);
}
```

```java
package com.example.demo.Services;

import com.example.demo.Models.Raza;
import com.example.demo.Repositories.RazaRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

// RazaServiceImpl.java
@Service
public class RazaServiceImpl implements RazaService {

    @Autowired
    private RazaRepository razaRepository;

    @Override
    public List<Raza> getAllRazas() {
```

```
        return razaRepository.findAll();
    }

    @Override
    public Raza getRazaById(Long id) {
        return razaRepository.findById(id).orElse(null);
    }

    @Override
    public Raza createRaza(Raza raza) {
        return razaRepository.save(raza);
    }

    @Override
    public Raza updateRaza(Long id, Raza raza) {
        if (razaRepository.existsById(id)) {
            raza.setIdRaza(id);
            return razaRepository.save(raza);
        }
        return null;
    }

    @Override
    public void deleteRaza(Long id) {
        razaRepository.deleteById(id);
    }
}
```

VETERINARIA

```
package com.example.demo.Models;

import jakarta.persistence.*;

import java.util.Date;

@Entity
@Table(name = "visita_veterinaria")
public class VisitaVeterinaria {

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

    @ManyToOne
    @JoinColumn(name = "id_mascota")
```

```java
    private Mascota mascota;

    private Date fecha;
    private String razonVisita;
    private String diagnostico;

    // Getter y Setter para idVisita
    public Long getIdVisita() {
        return idVisita;
    }

    public void setIdVisita(Long idVisita) {
        this.idVisita = idVisita;
    }

    // Getter y Setter para mascota
    public Mascota getMascota() {
        return mascota;
    }

    public void setMascota(Mascota mascota) {
        this.mascota = mascota;
    }

    // Getters y Setters para los demás atributos
    public Date getFecha() {
        return fecha;
    }

    public void setFecha(Date fecha) {
        this.fecha = fecha;
    }

    public String getRazonVisita() {
        return razonVisita;
    }

    public void setRazonVisita(String razonVisita) {
        this.razonVisita = razonVisita;
    }

    public String getDiagnostico() {
        return diagnostico;
    }

    public void setDiagnostico(String diagnostico) {
        this.diagnostico = diagnostico;
    }
```

```
}
```

```
package com.example.demo.Repositories;

import com.example.demo.Models.VisitaVeterinaria;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface VisitaVeterinariaRepository extends
JpaRepository<VisitaVeterinaria, Long> {
   List<VisitaVeterinaria> findAll();
   // Puedes agregar consultas personalizadas si es necesario
}
```

```
package com.example.demo.Services;

import com.example.demo.Models.VisitaVeterinaria;

import java.util.List;

public interface VisitaVeterinariaService {
   List<VisitaVeterinaria> getAllVisitasVeterinarias();

   VisitaVeterinaria getVisitaVeterinariaById(Long id);

   VisitaVeterinaria createVisitaVeterinaria(VisitaVeterinaria
visitaVeterinaria);

   VisitaVeterinaria updateVisitaVeterinaria(Long id,
VisitaVeterinaria visitaVeterinaria);

   void deleteVisitaVeterinaria(Long id);
}
```

```
package com.example.demo.Services;
import com.example.demo.Models.VisitaVeterinaria;
import com.example.demo.Repositories.VisitaVeterinariaRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
```

```java
public class VisitaVeterinariaServiceImpl implements
VisitaVeterinariaService {

    @Autowired
    private VisitaVeterinariaRepository visitaVeterinariaRepository;

    @Override
    public List<VisitaVeterinaria> getAllVisitasVeterinarias() {
        return visitaVeterinariaRepository.findAll();
    }

    @Override
    public VisitaVeterinaria getVisitaVeterinariaById(Long id) {
        return visitaVeterinariaRepository.findById(id).orElse(null);
    }

    @Override
    public VisitaVeterinaria
createVisitaVeterinaria(VisitaVeterinaria visitaVeterinaria) {
        return visitaVeterinariaRepository.save(visitaVeterinaria);
    }

    @Override
    public VisitaVeterinaria updateVisitaVeterinaria(Long id,
VisitaVeterinaria visitaVeterinaria) {
        if (visitaVeterinariaRepository.existsById(id)) {
            visitaVeterinaria.setIdVisita(id);
            return
visitaVeterinariaRepository.save(visitaVeterinaria);
        }
        return null;
    }

    @Override
    public void deleteVisitaVeterinaria(Long id) {
        visitaVeterinariaRepository.deleteById(id);
    }
}
```

Explicaciones clave:

- @Entity: Anotación que marca la clase como una entidad JPA, lo que significa que se mapeará a una tabla en la base de datos.
- @Id: Anotación que indica la clave primaria de la entidad.
- @GeneratedValue: Anotación que especifica cómo se generará automáticamente el valor de la clave primaria.
- @ManyToOne: Anotación que indica una relación de muchos a uno con otra entidad.

4. Crear el controlador de la API, al que haremos las llamadas del servicio (RestController) y crear dos métodos de cada tipo, que tengancoherencia.

PROPIETARIO

```
package com.example.demo.controllers;

import com.example.demo.Models.Propietario;
import com.example.demo.Services.PropietarioService;
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("/propietarios")
public class PropietarioController {

    @Autowired
    private PropietarioService propietarioService;

    private List<Propietario> otrosPropietarios;  // Lista para otro
conjunto de datos

    @GetMapping
    public ResponseEntity<List<Propietario>> getAllPropietarios() {
        List<Propietario> propietarios =
propietarioService.getAllPropietarios();
        return new ResponseEntity<>(propietarios, HttpStatus.OK);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Propietario>
getPropietarioById(@PathVariable Long id) {
        Propietario propietario =
propietarioService.getPropietarioById(id);
        return propietario != null ?
                new ResponseEntity<>(propietario, HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @PostMapping
```

```
    public ResponseEntity<Propietario> createPropietario(@RequestBody
Propietario propietario) {
        Propietario nuevoPropietario =
propietarioService.createPropietario(propietario);
        return new ResponseEntity<>(nuevoPropietario,
HttpStatus.CREATED);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Propietario>
updatePropietario(@PathVariable Long id, @RequestBody Propietario
propietario) {
        Propietario actualizadoPropietario =
propietarioService.updatePropietario(id, propietario);
        return actualizadoPropietario != null ?
                new ResponseEntity<>(actualizadoPropietario,
HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deletePropietario(@PathVariable Long
id) {
        propietarioService.deletePropietario(id);
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    }

    @GetMapping("/otro")
    public ResponseEntity<List<Propietario>> getAllPropietariosOtro()
{
        return new ResponseEntity<>(otrosPropietarios,
HttpStatus.OK);
    }

    @GetMapping("/otro/{id}")
    public ResponseEntity<Propietario>
getPropietarioOtroById(@PathVariable Long id) {
        Propietario otroPropietario =
obtenerPropietarioDesdeOtro(id);
        return otroPropietario != null ?
                new ResponseEntity<>(otroPropietario, HttpStatus.OK)
:
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @PostMapping("/otro")
    public ResponseEntity<Propietario>
createPropietarioOtro(@RequestBody Propietario propietario) {
```

```
        Propietario nuevoPropietarioOtro =
crearPropietarioEnOtro(propietario);
        return new ResponseEntity<>(nuevoPropietarioOtro,
HttpStatus.CREATED);
    }

    @PutMapping("/otro/{id}")
    public ResponseEntity<Propietario>
updatePropietarioOtro(@PathVariable Long id, @RequestBody
Propietario propietario) {
        Propietario actualizadoPropietarioOtro =
actualizarPropietarioOtro(id, propietario);
        return actualizadoPropietarioOtro != null ?
                new ResponseEntity<>(actualizadoPropietarioOtro,
HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @DeleteMapping("/otro/{id}")
    public ResponseEntity<Void> deletePropietarioOtro(@PathVariable
Long id) {
        boolean eliminado = eliminarPropietarioOtro(id);
        return eliminado ?
                new ResponseEntity<>(HttpStatus.NO_CONTENT) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    // Métodos ficticios para el conjunto "otro"
    private Propietario obtenerPropietarioDesdeOtro(Long id) {
        return otrosPropietarios.stream()
                .filter(propietario ->
propietario.getIdPropietario().equals(id))
                .findFirst()
                .orElse(null);
    }

    private Propietario crearPropietarioEnOtro(Propietario
propietario) {
        propietario.setIdPropietario(null); // Asignar null para que
la base de datos genere el ID automáticamente
        otrosPropietarios.add(propietario);
        return propietario;
    }

    private Propietario actualizarPropietarioOtro(Long id,
Propietario propietario) {
        for (int i = 0; i < otrosPropietarios.size(); i++) {
            Propietario existente = otrosPropietarios.get(i);
```

```
            if (existente.getIdPropietario().equals(id)) {
                propietario.setIdPropietario(id);
                otrosPropietarios.set(i, propietario);
                return propietario;
            }
        }
        return null;
    }

    private boolean eliminarPropietarioOtro(Long id) {
        return otrosPropietarios.removeIf(propietario ->
propietario.getIdPropietario().equals(id));
    }
}
```

a) getAllPropietarios:
   i) Descripción: Maneja las solicitudes GET a la ruta principal "/propietarios".
   ii) Acción: Llama al servicio para obtener todos los propietarios y devuelve una lista de ellos con el código de estado HTTP 200 (OK).

b) getPropietarioById:
   i) Descripción: Maneja las solicitudes GET a "/propietarios/{id}" para obtener un propietario por su ID.
   ii) Acción: Llama al servicio para obtener el propietario con el ID proporcionado y devuelve el propietario si se encuentra, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.

c) createPropietario:
   i) Descripción: Maneja las solicitudes POST a "/propietarios" para crear un nuevo propietario.
   ii) Acción: Llama al servicio para crear un nuevo propietario con los datos proporcionados en el cuerpo de la solicitud y devuelve el nuevo propietario con un código de estado HTTP 201 (CREATED).

d) updatePropietario:
   i) Descripción: Maneja las solicitudes PUT a "/propietarios/{id}" para actualizar un propietario existente.
   ii) Acción: Llama al servicio para actualizar el propietario con el ID proporcionado y los datos proporcionados en el cuerpo de la solicitud. Devuelve el propietario actualizado si se encuentra, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.

e) deletePropietario:
   i) Descripción: Maneja las solicitudes DELETE a "/propietarios/{id}" para eliminar un propietario por su ID.
   ii) Acción: Llama al servicio para eliminar el propietario con el ID proporcionado y devuelve un código de estado HTTP 204 (NO CONTENT) indicando que la operación se realizó con éxito.

f) getAllPropietariosOtro:

i) Descripción: Maneja las solicitudes GET a "/propietarios/otro" para obtener otro conjunto de propietarios ficticios.
ii) Acción: Devuelve otro conjunto de propietarios ficticios almacenados en la lista otrosPropietarios con el código de estado HTTP 200 (OK).

g) getPropietarioOtroById:
i) Descripción: Maneja las solicitudes GET a "/propietarios/otro/{id}" para obtener un propietario del otro conjunto por su ID.
ii) Acción: Devuelve el propietario con el ID proporcionado del otro conjunto ficticio si se encuentra, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.

h) createPropietarioOtro:
i) Descripción: Maneja las solicitudes POST a "/propietarios/otro" para crear un nuevo propietario en otro conjunto.
ii) Acción: Crea un nuevo propietario en el otro conjunto ficticio con los datos proporcionados en el cuerpo de la solicitud y devuelve el nuevo propietario con un código de estado HTTP 201 (CREATED).

i) updatePropietarioOtro:
i) Descripción: Maneja las solicitudes PUT a "/propietarios/otro/{id}" para actualizar un propietario en otro conjunto.
ii) Acción: Actualiza el propietario en el otro conjunto ficticio con el ID proporcionado y los datos proporcionados en el cuerpo de la solicitud. Devuelve el propietario actualizado si se encuentra, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.

j) deletePropietarioOtro:
i) Descripción: Maneja las solicitudes DELETE a "/propietarios/otro/{id}" para eliminar un propietario en otro conjunto por su ID.
ii) Acción: Elimina el propietario en el otro conjunto ficticio con el ID proporcionado y devuelve un código de estado HTTP 204 (NO CONTENT) si la operación se realizó con éxito, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.

k)

MASCOTA

```java
package com.example.demo.controllers;

import com.example.demo.Models.Mascota;
import com.example.demo.Services.MascotaService;
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("/mascotas")
public class MascotaController {
```

```java
    @Autowired
    private MascotaService mascotaService;

    private List<Mascota> otrasMascotas;  // Lista para otro conjunto
de datos

    @GetMapping
    public ResponseEntity<List<Mascota>> getAllMascotas() {
        List<Mascota> mascotas = mascotaService.getAllMascotas();
        return new ResponseEntity<>(mascotas, HttpStatus.OK);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Mascota> getMascotaById(@PathVariable Long
id) {
        Mascota mascota = mascotaService.getMascotaById(id);
        return mascota != null ?
                new ResponseEntity<>(mascota, HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @PostMapping
    public ResponseEntity<Mascota> createMascota(@RequestBody Mascota
mascota) {
        Mascota nuevaMascota = mascotaService.createMascota(mascota);
        return new ResponseEntity<>(nuevaMascota,
HttpStatus.CREATED);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Mascota> updateMascota(@PathVariable Long
id, @RequestBody Mascota mascota) {
        Mascota actualizadaMascota = mascotaService.updateMascota(id,
mascota);
        return actualizadaMascota != null ?
                new ResponseEntity<>(actualizadaMascota,
HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteMascota(@PathVariable Long id)
{
        mascotaService.deleteMascota(id);
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    }
```

```java
    @GetMapping("/otro")
    public ResponseEntity<List<Mascota>> getAllMascotasOtro() {
        return new ResponseEntity<>(otrasMascotas, HttpStatus.OK);
    }

    @GetMapping("/otro/{id}")
    public ResponseEntity<Mascota> getMascotaOtroById(@PathVariable
Long id) {
        Mascota otraMascota = obtenerMascotaDesdeOtro(id);
        return otraMascota != null ?
                new ResponseEntity<>(otraMascota, HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @PostMapping("/otro")
    public ResponseEntity<Mascota> createMascotaOtro(@RequestBody
Mascota mascota) {
        Mascota nuevaMascotaOtro = crearMascotaEnOtro(mascota);
        return new ResponseEntity<>(nuevaMascotaOtro,
HttpStatus.CREATED);
    }

    @PutMapping("/otro/{id}")
    public ResponseEntity<Mascota> updateMascotaOtro(@PathVariable
Long id, @RequestBody Mascota mascota) {
        Mascota actualizadaMascotaOtro = actualizarMascotaOtro(id,
mascota);
        return actualizadaMascotaOtro != null ?
                new ResponseEntity<>(actualizadaMascotaOtro,
HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @DeleteMapping("/otro/{id}")
    public ResponseEntity<Void> deleteMascotaOtro(@PathVariable Long
id) {
        boolean eliminada = eliminarMascotaOtro(id);
        return eliminada ?
                new ResponseEntity<>(HttpStatus.NO_CONTENT) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    // Métodos ficticios para el conjunto "otro"
    private Mascota obtenerMascotaDesdeOtro(Long id) {
        return otrasMascotas.stream()
                .filter(mascota -> mascota.getIdMascota().equals(id))
                .findFirst()
                .orElse(null);
```

```
    }

    private Mascota crearMascotaEnOtro(Mascota mascota) {
        mascota.setIdMascota(null); // Asignar null para que la base
de datos genere el ID automáticamente
        otrasMascotas.add(mascota);
        return mascota;
    }

    private Mascota actualizarMascotaOtro(Long id, Mascota mascota) {
        for (int i = 0; i < otrasMascotas.size(); i++) {
            Mascota existente = otrasMascotas.get(i);
            if (existente.getIdMascota().equals(id)) {
                mascota.setIdMascota(id);
                otrasMascotas.set(i, mascota);
                return mascota;
            }
        }
        return null;
    }

    private boolean eliminarMascotaOtro(Long id) {
        return otrasMascotas.removeIf(mascota ->
mascota.getIdMascota().equals(id));
    }
}
```

1. `getAllMascotas`: **Obtiene todas las mascotas.**
2. `getMascotaById`: **Obtiene una mascota por su ID.**
3. `createMascota`: **Crea una nueva mascota.**
4. `updateMascota`: **Actualiza una mascota existente.**
5. `deleteMascota`: **Elimina una mascota por su ID.**
6. `getAllMascotasOtro`: **Obtiene otro conjunto de mascotas ficticias.**
7. `getMascotaOtroById`: **Obtiene una mascota del otro conjunto por su ID.**
8. `createMascotaOtro`: **Crea una nueva mascota en el otro conjunto ficticio.**
9. `updateMascotaOtro`: **Actualiza una mascota en el otro conjunto ficticio.**
10. `deleteMascotaOtro`: **Elimina una mascota del otro conjunto ficticio por su ID.**
11.

RAZA

```
package com.example.demo.controllers;
```

```java
import com.example.demo.Models.Raza;
import com.example.demo.Services.RazaService;
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("/razas")
public class RazaController {

    @Autowired
    private RazaService razaService;

    private List<Raza> otrasRazas;  // Lista para otro conjunto de datos

    @GetMapping
    public ResponseEntity<List<Raza>> getAllRazas() {
        List<Raza> razas = razaService.getAllRazas();
        return new ResponseEntity<>(razas, HttpStatus.OK);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Raza> getRazaById(@PathVariable Long id) {
        Raza raza = razaService.getRazaById(id);
        return raza != null ?
                new ResponseEntity<>(raza, HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @PostMapping
    public ResponseEntity<Raza> createRaza(@RequestBody Raza raza) {
        Raza nuevaRaza = razaService.createRaza(raza);
        return new ResponseEntity<>(nuevaRaza, HttpStatus.CREATED);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Raza> updateRaza(@PathVariable Long id,
@RequestBody Raza raza) {
        Raza actualizadaRaza = razaService.updateRaza(id, raza);
        return actualizadaRaza != null ?
                new ResponseEntity<>(actualizadaRaza, HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
```

```
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteRaza(@PathVariable Long id) {
        razaService.deleteRaza(id);
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    }

    @GetMapping("/otro")
    public ResponseEntity<List<Raza>> getAllRazasOtro() {
        return new ResponseEntity<>(otrasRazas, HttpStatus.OK);
    }

    @GetMapping("/otro/{id}")
    public ResponseEntity<Raza> getRazaOtroById(@PathVariable Long id) {
        Raza otraRaza = obtenerRazaDesdeOtro(id);
        return otraRaza != null ?
                new ResponseEntity<>(otraRaza, HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @PostMapping("/otro")
    public ResponseEntity<Raza> createRazaOtro(@RequestBody Raza raza) {
        Raza nuevaRazaOtro = crearRazaEnOtro(raza);
        return new ResponseEntity<>(nuevaRazaOtro, HttpStatus.CREATED);
    }

    @PutMapping("/otro/{id}")
    public ResponseEntity<Raza> updateRazaOtro(@PathVariable Long id,
@RequestBody Raza raza) {
        Raza actualizadaRazaOtro = actualizarRazaOtro(id, raza);
        return actualizadaRazaOtro != null ?
                new ResponseEntity<>(actualizadaRazaOtro, HttpStatus.OK)
:
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @DeleteMapping("/otro/{id}")
    public ResponseEntity<Void> deleteRazaOtro(@PathVariable Long id) {
        boolean eliminada = eliminarRazaOtro(id);
        return eliminada ?
                new ResponseEntity<>(HttpStatus.NO_CONTENT) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    // Métodos ficticios para el conjunto "otro"
```

```
    private Raza obtenerRazaDesdeOtro(Long id) {
        return otrasRazas.stream()
                .filter(raza -> raza.getIdRaza().equals(id))
                .findFirst()
                .orElse(null);
    }

    private Raza crearRazaEnOtro(Raza raza) {
        raza.setIdRaza(null); // Asignar null para que la base de datos
genere el ID automáticamente
        otrasRazas.add(raza);
        return raza;
    }

    private Raza actualizarRazaOtro(Long id, Raza raza) {
        for (int i = 0; i < otrasRazas.size(); i++) {
            Raza existente = otrasRazas.get(i);
            if (existente.getIdRaza().equals(id)) {
                raza.setIdRaza(id);
                otrasRazas.set(i, raza);
                return raza;
            }
        }
        return null;
    }

    private boolean eliminarRazaOtro(Long id) {
        return otrasRazas.removeIf(raza -> raza.getIdRaza().equals(id));
    }
}
```

`getAllRazas`:

- **Descripción: Maneja las solicitudes GET a la ruta principal "/razas".**
- **Acción: Llama al servicio para obtener todas las razas y devuelve una lista de ellas con el código de estado HTTP 200 (OK).**

`getRazaById`:

- **Descripción: Maneja las solicitudes GET a "/razas/{id}" para obtener una raza por su ID.**
- **Acción: Llama al servicio para obtener la raza con el ID proporcionado y devuelve la raza si se encuentra, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.**

`createRaza`:

- **Descripción: Maneja las solicitudes POST a "/razas" para crear una**

nueva raza.

- **Acción: Llama al servicio para crear una nueva raza con los datos proporcionados en el cuerpo de la solicitud y devuelve la nueva raza con un código de estado HTTP 201 (CREATED).**

`updateRaza`:

- **Descripción: Maneja las solicitudes PUT a "/razas/{id}" para actualizar una raza existente.**
- **Acción: Llama al servicio para actualizar la raza con el ID proporcionado y los datos proporcionados en el cuerpo de la solicitud. Devuelve la raza actualizada si se encuentra, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.**

`deleteRaza`:

- **Descripción: Maneja las solicitudes DELETE a "/razas/{id}" para eliminar una raza por su ID.**
- **Acción: Llama al servicio para eliminar la raza con el ID proporcionado y devuelve un código de estado HTTP 204 (NO CONTENT) indicando que la operación se realizó con éxito.**

`getAllRazasOtro`:

- **Descripción: Maneja las solicitudes GET a "/razas/otro" para obtener otro conjunto de razas ficticias.**
- **Acción: Devuelve otro conjunto de razas ficticias almacenadas en la lista `otrasRazas` con el código de estado HTTP 200 (OK).**

`getRazaOtroById`:

- **Descripción: Maneja las solicitudes GET a "/razas/otro/{id}" para obtener una raza del otro conjunto por su ID.**
- **Acción: Devuelve la raza con el ID proporcionado del otro conjunto ficticio si se encuentra, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.**

`createRazaOtro`:

- **Descripción: Maneja las solicitudes POST a "/razas/otro" para crear una nueva raza en otro conjunto.**
- **Acción: Crea una nueva raza en el otro conjunto ficticio con los datos proporcionados en el cuerpo de la solicitud y devuelve la nueva raza con un código de estado HTTP 201 (CREATED).**

`updateRazaOtro`:

- **Descripción: Maneja las solicitudes PUT a "/razas/otro/{id}" para actualizar una raza en otro conjunto.**
- **Acción: Actualiza la raza en el otro conjunto ficticio con el ID proporcionado y los datos proporcionados en el cuerpo de la solicitud. Devuelve la raza actualizada si se encuentra, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.**

`deleteRazaOtro`:

- **Descripción: Maneja las solicitudes DELETE a "/razas/otro/{id}" para eliminar una raza en otro conjunto por su ID.**
- **Acción: Elimina la raza en el otro conjunto ficticio con el ID proporcionado y devuelve un código de estado HTTP 204 (NO CONTENT) si la operación se realizó con éxito, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.**

VISITA VETERINARIA

```
package com.example.demo.controllers;

import com.example.demo.Models.VisitaVeterinaria;
import com.example.demo.Services.VisitaVeterinariaService;
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("/visitasveterinarias")
public class VisitaVeterinariaController {

    @Autowired
    private VisitaVeterinariaService visitaVeterinariaService;

    private List<VisitaVeterinaria> otrasVisitasVeterinarias;  //
Lista para otro conjunto de datos

    @GetMapping
    public ResponseEntity<List<VisitaVeterinaria>>
getAllVisitasVeterinarias() {
        List<VisitaVeterinaria> visitasVeterinarias =
visitaVeterinariaService.getAllVisitasVeterinarias();
        return new ResponseEntity<>(visitasVeterinarias,
HttpStatus.OK);
    }

    @GetMapping("/{id}")
    public ResponseEntity<VisitaVeterinaria>
getVisitaVeterinariaById(@PathVariable Long id) {
        VisitaVeterinaria visitaVeterinaria =
visitaVeterinariaService.getVisitaVeterinariaById(id);
        return visitaVeterinaria != null ?
                new ResponseEntity<>(visitaVeterinaria,
```

```
HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @PostMapping
    public ResponseEntity<VisitaVeterinaria>
createVisitaVeterinaria(@RequestBody VisitaVeterinaria
visitaVeterinaria) {
        VisitaVeterinaria nuevaVisitaVeterinaria =
visitaVeterinariaService.createVisitaVeterinaria(visitaVeterinaria);
        return new ResponseEntity<>(nuevaVisitaVeterinaria,
HttpStatus.CREATED);
    }

    @PutMapping("/{id}")
    public ResponseEntity<VisitaVeterinaria>
updateVisitaVeterinaria(@PathVariable Long id, @RequestBody
VisitaVeterinaria visitaVeterinaria) {
        VisitaVeterinaria actualizadaVisitaVeterinaria =
visitaVeterinariaService.updateVisitaVeterinaria(id,
visitaVeterinaria);
        return actualizadaVisitaVeterinaria != null ?
                new ResponseEntity<>(actualizadaVisitaVeterinaria,
HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteVisitaVeterinaria(@PathVariable
Long id) {
        visitaVeterinariaService.deleteVisitaVeterinaria(id);
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    }

    @GetMapping("/otro")
    public ResponseEntity<List<VisitaVeterinaria>>
getAllVisitasVeterinariasOtro() {
        return new ResponseEntity<>(otrasVisitasVeterinarias,
HttpStatus.OK);
    }

    @GetMapping("/otro/{id}")
    public ResponseEntity<VisitaVeterinaria>
getVisitaVeterinariaOtroById(@PathVariable Long id) {
        VisitaVeterinaria otraVisitaVeterinaria =
obtenerVisitaVeterinariaDesdeOtro(id);
        return otraVisitaVeterinaria != null ?
                new ResponseEntity<>(otraVisitaVeterinaria,
```

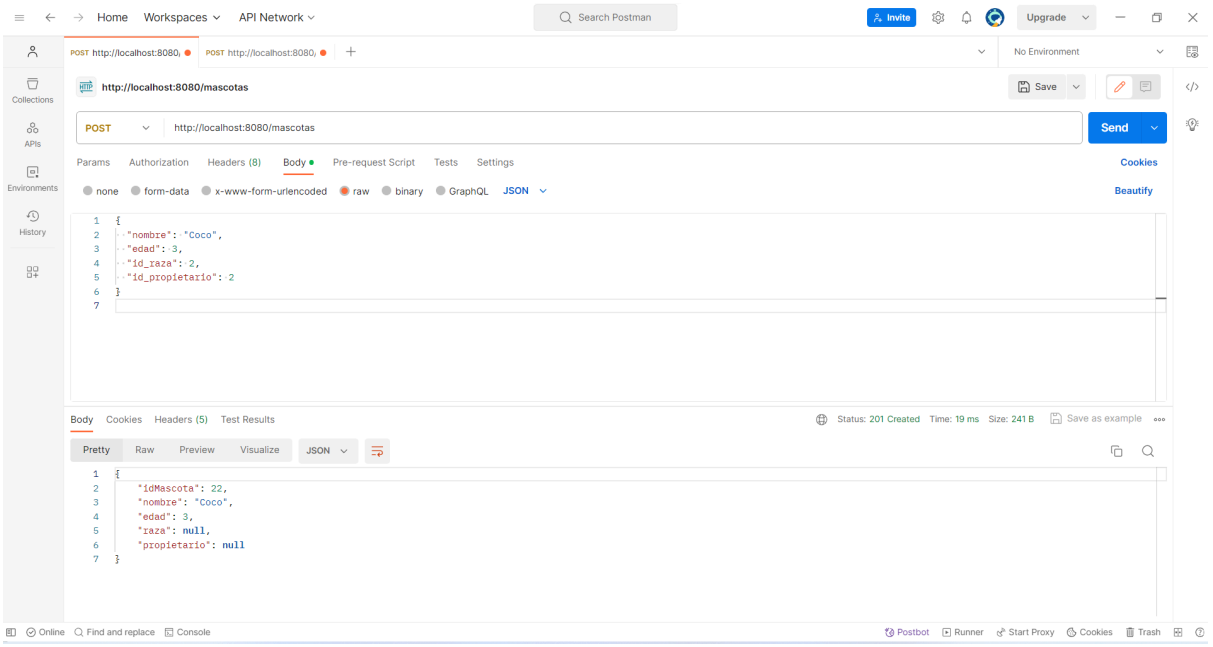
```
HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @PostMapping("/otro")
    public ResponseEntity<VisitaVeterinaria>
createVisitaVeterinariaOtro(@RequestBody VisitaVeterinaria
visitaVeterinaria) {
        VisitaVeterinaria nuevaVisitaVeterinariaOtro =
crearVisitaVeterinariaEnOtro(visitaVeterinaria);
        return new ResponseEntity<>(nuevaVisitaVeterinariaOtro,
HttpStatus.CREATED);
    }

    @PutMapping("/otro/{id}")
    public ResponseEntity<VisitaVeterinaria>
updateVisitaVeterinariaOtro(@PathVariable Long id, @RequestBody
VisitaVeterinaria visitaVeterinaria) {
        VisitaVeterinaria actualizadaVisitaVeterinariaOtro =
actualizarVisitaVeterinariaOtro(id, visitaVeterinaria);
        return actualizadaVisitaVeterinariaOtro != null ?
                new
ResponseEntity<>(actualizadaVisitaVeterinariaOtro, HttpStatus.OK) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    @DeleteMapping("/otro/{id}")
    public ResponseEntity<Void>
deleteVisitaVeterinariaOtro(@PathVariable Long id) {
        boolean eliminada = eliminarVisitaVeterinariaOtro(id);
        return eliminada ?
                new ResponseEntity<>(HttpStatus.NO_CONTENT) :
                new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }

    // Métodos ficticios para el conjunto "otro"
    private VisitaVeterinaria obtenerVisitaVeterinariaDesdeOtro(Long
id) {
        return otrasVisitasVeterinarias.stream()
                .filter(visita -> visita.getIdVisita().equals(id))
                .findFirst()
                .orElse(null);
    }


    private VisitaVeterinaria
crearVisitaVeterinariaEnOtro(VisitaVeterinaria visitaVeterinaria) {
        visitaVeterinaria.setIdVisita(null); // Asignar null para que
```

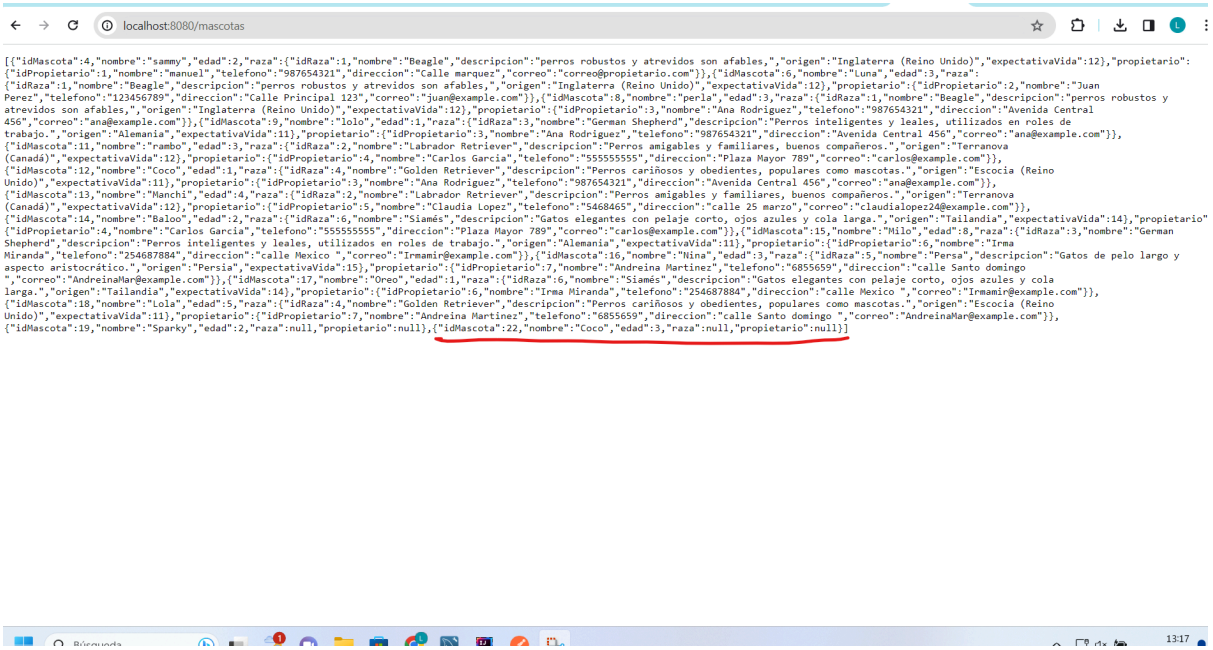
```
la base de datos genere el ID automáticamente
        otrasVisitasVeterinarias.add(visitaVeterinaria);
        return visitaVeterinaria;
    }



    private VisitaVeterinaria actualizarVisitaVeterinariaOtro(Long
id, VisitaVeterinaria visitaVeterinaria) {
        for (int i = 0; i < otrasVisitasVeterinarias.size(); i++) {
            VisitaVeterinaria existente =
otrasVisitasVeterinarias.get(i);
            if (existente.getIdVisita().equals(id)) {
                visitaVeterinaria.setIdVisita(id);
                otrasVisitasVeterinarias.set(i, visitaVeterinaria);
                return visitaVeterinaria;
            }
        }
        return null;
    }

    private boolean eliminarVisitaVeterinariaOtro(Long id) {
        return otrasVisitasVeterinarias.removeIf(visita ->
visita.getIdVisita().equals(id));
    }
}
```

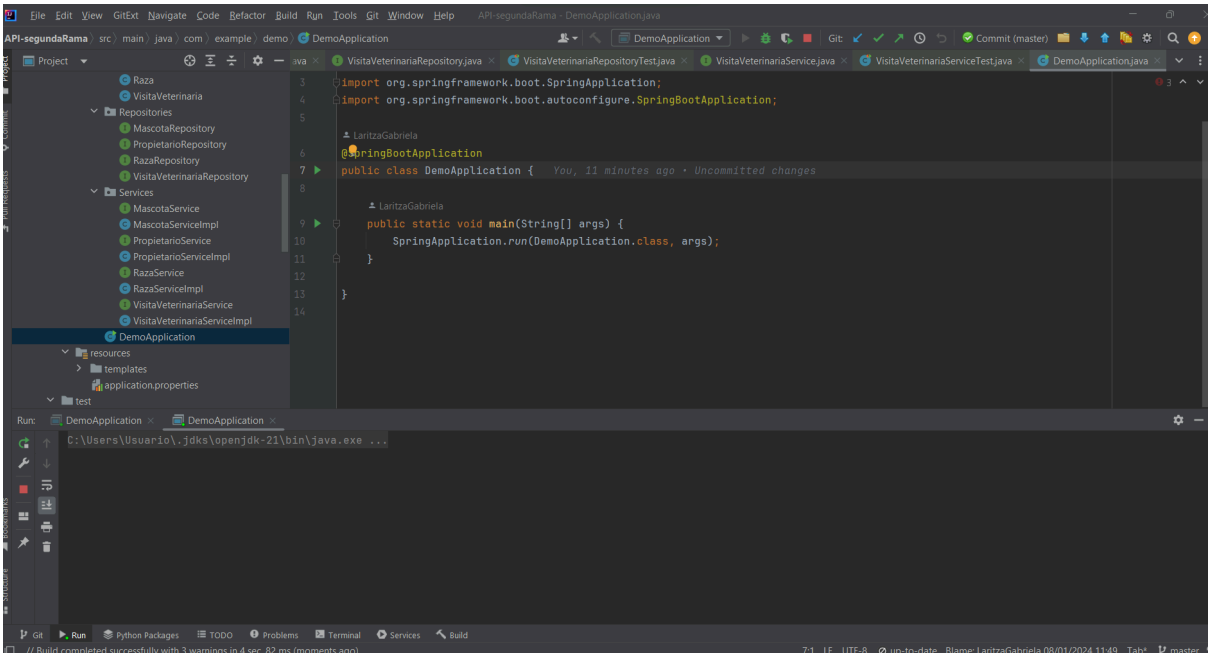
1. getAllVisitasVeterinarias:
    a. Descripción: Este método maneja las solicitudes GET a la ruta principal "/visitasveterinarias".
    b. Acción: Llama al servicio para obtener todas las visitas veterinarias y devuelve una lista de ellas con el código de estado HTTP 200 (OK).
2. getVisitaVeterinariaById:
    a. Descripción: Maneja las solicitudes GET a "/visitasveterinarias/{id}" para obtener una visita veterinaria por su ID.
    b. Acción: Llama al servicio para obtener la visita veterinaria con el ID proporcionado y devuelve la visita veterinaria si se encuentra, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.
3. createVisitaVeterinaria:
    a. Descripción: Maneja las solicitudes POST a "/visitasveterinarias" para crear una nueva visita veterinaria.
    b. Acción: Llama al servicio para crear una nueva visita veterinaria con los datos proporcionados en el cuerpo de la solicitud y devuelve la nueva visita veterinaria con un código de estado HTTP 201 (CREATED).
4. updateVisitaVeterinaria:
    a. Descripción: Maneja las solicitudes PUT a "/visitasveterinarias/{id}" para actualizar una visita veterinaria existente.

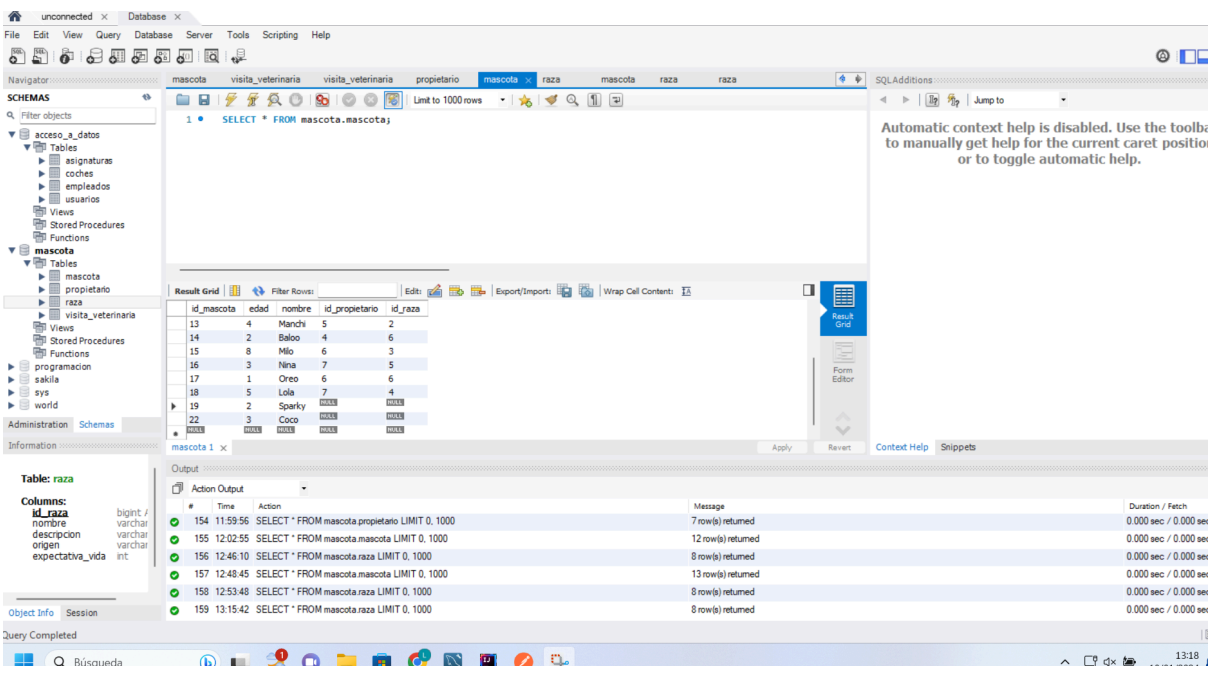
    b. Acción: Llama al servicio para actualizar la visita veterinaria con el ID proporcionado y los datos proporcionados en el cuerpo de la solicitud. Devuelve la visita veterinaria actualizada si se encuentra, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.
5. deleteVisitaVeterinaria:
    a. Descripción: Maneja las solicitudes DELETE a "/visitasveterinarias/{id}" para eliminar una visita veterinaria por su ID.
    b. Acción: Llama al servicio para eliminar la visita veterinaria con el ID proporcionado y devuelve un código de estado HTTP 204 (NO CONTENT) indicando que la operación se realizó con éxito.
6. getAllVisitasVeterinariasOtro:
    a. Descripción: Maneja las solicitudes GET a "/visitasveterinarias/otro" para obtener otro conjunto de visitas veterinarias.
    b. Acción: Devuelve otro conjunto de visitas veterinarias ficticias almacenadas en la lista otrasVisitasVeterinarias con el código de estado HTTP 200 (OK).
7. getVisitaVeterinariaOtroById:
    a. Descripción: Maneja las solicitudes GET a "/visitasveterinarias/otro/{id}" para obtener una visita veterinaria de otro conjunto por su ID.
    b. Acción: Devuelve la visita veterinaria con el ID proporcionado del otro conjunto ficticio si se encuentra, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.
8. createVisitaVeterinariaOtro:
    a. Descripción: Maneja las solicitudes POST a "/visitasveterinarias/otro" para crear una nueva visita veterinaria en otro conjunto.
    b. Acción: Crea una nueva visita veterinaria en el otro conjunto ficticio con los datos proporcionados en el cuerpo de la solicitud y devuelve la nueva visita veterinaria con un código de estado HTTP 201 (CREATED).
9. updateVisitaVeterinariaOtro:
    a. Descripción: Maneja las solicitudes PUT a "/visitasveterinarias/otro/{id}" para actualizar una visita veterinaria en otro conjunto.
    b. Acción: Actualiza la visita veterinaria en el otro conjunto ficticio con el ID proporcionado y los datos proporcionados en el cuerpo de la solicitud. Devuelve la visita veterinaria actualizada si se encuentra, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.
10. deleteVisitaVeterinariaOtro:
    a. Descripción: Maneja las solicitudes DELETE a "/visitasveterinarias/otro/{id}" para eliminar una visita veterinaria en otro conjunto por su ID.
    b. Acción: Elimina la visita veterinaria en el otro conjunto ficticio con el ID proporcionado y devuelve un código de estado HTTP 204 (NO CONTENT) si la operación se realizó con éxito, o un código de estado HTTP 404 (NOT FOUND) si no se encuentra.

5 Capturas del postman, mínimo una de cada método creado.
Explicar brevemente el código o lo que vais a añadir.
Agregar también un apartado de cómo os ha resultado la práctica, si os ha gustado y si habéis tenido algún problema.


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
No Environment
http://localhost:8080/mascotas
Save
POST
http://localhost:8080/mascotas
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"nombre": "Coco",
"edad": 3,
"id_raza": 2,
"id_propietario": 2
}
Body
Cookies
Headers (5)
Test Results
Status: 201 Created
Time: 19 ms
Size: 241 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"idMascota": 22,
"nombre": "Coco",
"edad": 3,
"raza": null,
"propietario": null
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash


localhost:8080/mascotas
[{"idMascota":4,"nombre":"sammy","edad":2,"raza":{"idRaza":1,"nombre":"Beagle","descripcion":"perros robustos y atrevidos son afables,","origen":"Inglaterra (Reino Unido)","expectativaVida":12},"propietario":
{"idPropietario":1,"nombre":"manuel","telefono":"9876543211","direccion":"Calle marquez","correo":"correo@propietario.com"}},{"idMascota":6,"nombre":"Luna","edad":3,"raza":
{"idRaza":1,"nombre":"Beagle","descripcion":"perros robustos y atrevidos son afables,","origen":"Inglaterra (Reino Unido)","expectativaVida":12},"propietario":{"idPropietario":2,"nombre":"Juan
Perez","telefono":"123456789","direccion":"Calle Principal 123","correo":"juan@example.com"}},{"idMascota":8,"nombre":"perla","edad":3,"raza":{"idRaza":1,"nombre":"Beagle","descripcion":"perros robustos y
atrevidos son afables,","origen":"Inglaterra (Reino Unido)","expectativaVida":12},"propietario":{"idPropietario":3,"nombre":"Ana Rodriguez","telefono":"987654321","direccion":"Avenida Central
456","correo":"ana@example.com"}},{"idMascota":9,"nombre":"lolo","edad":1,"raza":{"idRaza":3,"nombre":"German Shepherd","descripcion":"Perros inteligentes y leales, utilizados en roles de
trabajo.","origen":"Alemania","expectativaVida":11},"propietario":{"idPropietario":3,"nombre":"Ana Rodriguez","telefono":"987654321","direccion":"Avenida Central 456","correo":"ana@example.com"}},
{"idMascota":11,"nombre":"rambo","edad":3,"raza":{"idRaza":2,"nombre":"Labrador Retriever","descripcion":"Perros amigables y familiares, buenos compañeros.","origen":"Terranova
(Canadá)","expectativaVida":12},"propietario":{"idPropietario":4,"nombre":"Carlos Garcia","telefono":"555555555","direccion":"Plaza Mayor 789","correo":"carlos@example.com"}},
{"idMascota":12,"nombre":"Coco","edad":1,"raza":{"idRaza":4,"nombre":"Golden Retriever","descripcion":"Perros cariñosos y obedientes, populares como mascotas.","origen":"Escocia (Reino
Unido)","expectativaVida":11},"propietario":{"idPropietario":3,"nombre":"Ana Rodriguez","telefono":"987654321","direccion":"Avenida Central 456","correo":"ana@example.com"}},
{"idMascota":13,"nombre":"Manchi","edad":4,"raza":{"idRaza":2,"nombre":"Labrador Retriever","descripcion":"Perros amigables y familiares, buenos compañeros.","origen":"Terranova
(Canadá)","expectativaVida":12},"propietario":{"idPropietario":5,"nombre":"Claudia Lopez","telefono":"5468465","direccion":"calle 25 marzo","correo":"claudialopez24@example.com"}},
{"idMascota":14,"nombre":"Baloo","edad":2,"raza":{"idRaza":6,"nombre":"Siamés","descripcion":"Gatos elegantes con pelaje corto, ojos azules y cola larga.","origen":"Tailandia","expectativaVida":14},"propietario"
{"idPropietario":4,"nombre":"Carlos Garcia","telefono":"555555555","direccion":"Plaza Mayor 789","correo":"carlos@example.com"}},{"idMascota":15,"nombre":"Milo","edad":8,"raza":{"idRaza":3,"nombre":"German
Shepherd","descripcion":"Perros inteligentes y leales, utilizados en roles de trabajo.","origen":"Alemania","expectativaVida":11},"propietario":{"idPropietario":6,"nombre":"Irma
Miranda","telefono":"254687884","direccion":"calle Mexico ","correo":"Irmamir@example.com"}},{"idMascota":16,"nombre":"Nina","edad":3,"raza":{"idRaza":5,"nombre":"Persa","descripcion":"Gatos de pelo largo y
aspecto aristocrático.","origen":"Persia","expectativaVida":15},"propietario":{"idPropietario":7,"nombre":"Andreina Martinez","telefono":"6855659","direccion":"calle Santo domingo
","correo":"AndreinaMar@example.com"}},{"idMascota":17,"nombre":"Oreo","edad":1,"raza":{"idRaza":6,"nombre":"Siamés","descripcion":"Gatos elegantes con pelaje corto, ojos azules y cola
larga.","origen":"Tailandia","expectativaVida":14},"propietario":{"idPropietario":6,"nombre":"Irma Miranda","telefono":"254687884","direccion":"calle Mexico ","correo":"Irmamir@example.com"}},
{"idMascota":18,"nombre":"Lola","edad":5,"raza":{"idRaza":4,"nombre":"Golden Retriever","descripcion":"Perros cariñosos y obedientes, populares como mascotas.","origen":"Escocia (Reino
Unido)","expectativaVida":11},"propietario":{"idPropietario":7,"nombre":"Andreina Martinez","telefono":"6855659","direccion":"calle Santo domingo ","correo":"AndreinaMar@example.com"}},
{"idMascota":19,"nombre":"Sparky","edad":2,"raza":null,"propietario":null},{"idMascota":22,"nombre":"Coco","edad":3,"raza":null,"propietario":null}]


File Edit View GitExt Navigate Code Refactor Build Run Tools Git Window Help
API-segundaRama - DemoApplication.java
Project
Raza
VisitaVeterinaria
Repositories
MascotaRepository
PropietarioRepository
RazaRepository
VisitaVeterinariaRepository
Services
MascotaService
MascotaServiceImpl
PropietarioService
PropietarioServiceImpl
RazaService
RazaServiceImpl
VisitaVeterinariaService
VisitaVeterinariaServiceImpl
DemoApplication
resources
templates
application.properties
test
VisitaVeterinariaRepository.java
VisitaVeterinariaRepositoryTest.java
VisitaVeterinariaService.java
VisitaVeterinariaServiceTest.java
DemoApplication.java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
LaritzaGabriela
@SpringBootApplication
public class DemoApplication {
You, 11 minutes ago · Uncommitted changes
LaritzaGabriela
public static void main(String[] args) {
SpringApplication.run(DemoApplication.class, args);
}
}
Run: DemoApplication
DemoApplication
C:\Users\Usuario\.jdks\openjdk-21\bin\java.exe ...
Git
Run
Python Packages
TODO
Problems
Terminal
Services
Build


unconnected
Database
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
acceso_a_datos
Tables
asignaturas
coches
empleados
usuarios
Views
Stored Procedures
Functions
mascota
Tables
mascota
propietario
raza
visita_veterinaria
Views
Stored Procedures
Functions
programacion
sakila
sys
world
Administration Schemas
Information
Table: raza
Columns:
id_raza bigint
nombre varchar
descripcion varchar
origen varchar
expectativa_vida int
Object Info Session
mascota visita_veterinaria visita_veterinaria propietario mascota raza mascota raza raza
Limit to 1000 rows
1 SELECT * FROM mascota.mascota;
Result Grid
id_mascota edad nombre id_propietario id_raza
13 4 Manchi 5 2
14 2 Baloo 4 6
15 8 Milo 6 3
16 3 Nina 7 5
17 1 Oreo 6 6
18 5 Lola 7 4
19 2 Sparky NULL NULL
22 3 Coco NULL NULL
mascota 1
Apply
Revert
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolba
to manually get help for the current caret positio
or to toggle automatic help.
Context Help
Snippets
Output
Action Output
Time Action Message Duration / Fetch
154 11:59:56 SELECT * FROM mascota.propietario LIMIT 0, 1000 7 row(s) returned 0.000 sec / 0.000 se
155 12:02:55 SELECT * FROM mascota.mascota LIMIT 0, 1000 12 row(s) returned 0.000 sec / 0.000 se
156 12:46:10 SELECT * FROM mascota.raza LIMIT 0, 1000 8 row(s) returned 0.000 sec / 0.000 se
157 12:48:45 SELECT * FROM mascota.mascota LIMIT 0, 1000 13 row(s) returned 0.000 sec / 0.000 se
158 12:53:48 SELECT * FROM mascota.raza LIMIT 0, 1000 8 row(s) returned 0.000 sec / 0.000 se
159 13:15:42 SELECT * FROM mascota.raza LIMIT 0, 1000 8 row(s) returned 0.000 sec / 0.000 se
Query Completed
Búsqueda
13:18