**Proyecto gestortienda**

**REUNIÓN SEMANAL DE SCRUM**

**ACTA No. 01**

**09/09/2022, 09:00 pm**

**SPRINT No. \_\_1\_ SEMANA No. \_\_1\_**

**ORDEN DEL DÍA**

1. Acuerdos sobre dinámica de trabajo

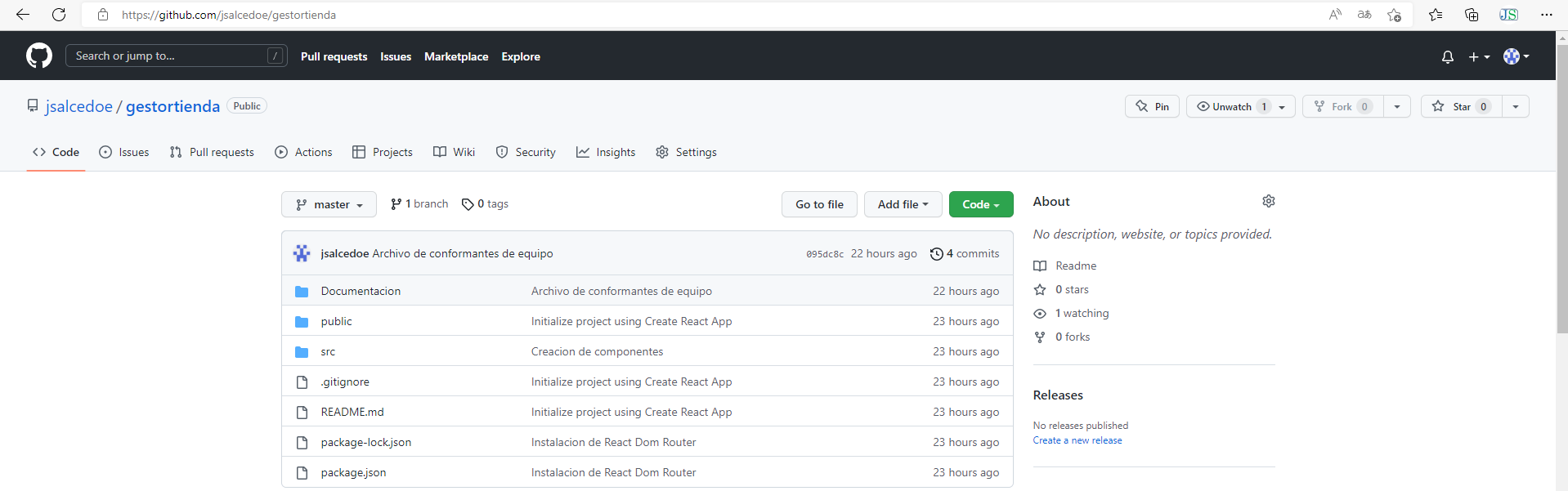
1. Seguimiento a tareas

1. Dificultades presentadas

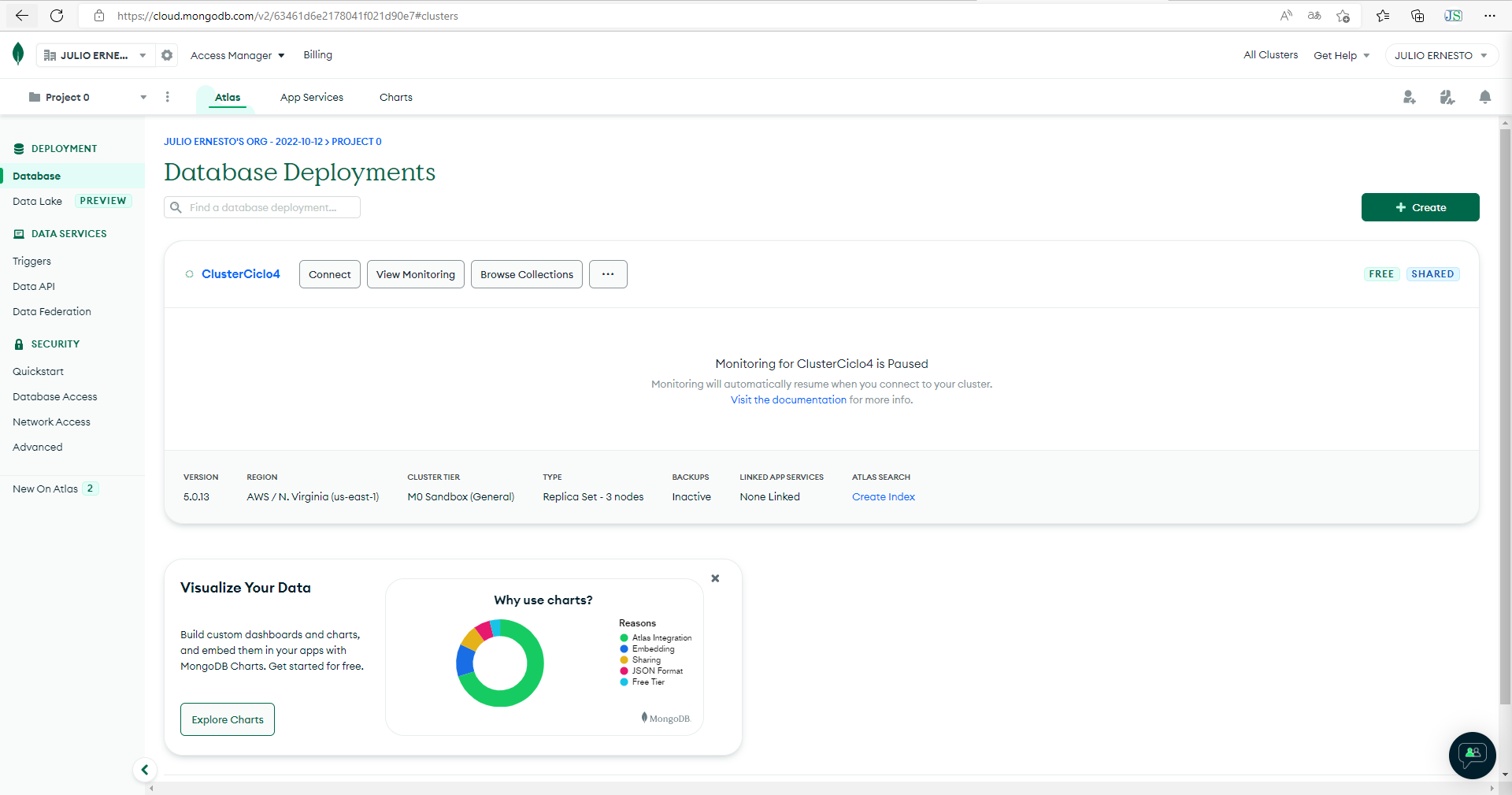
1. Varios

**DESARROLLO DEL ORDEN DEL DÍA**

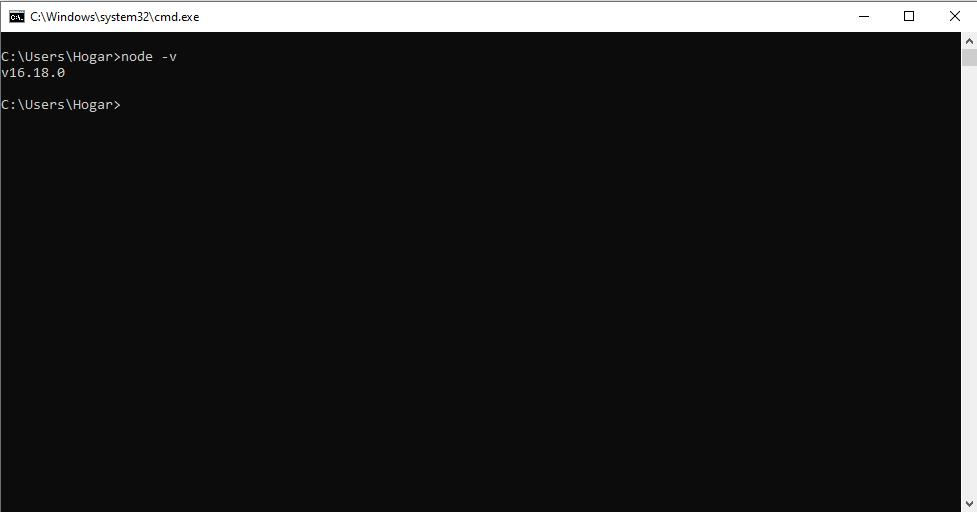
**1.** Creación del repositorio en Github para la publicación del proyecto gestortienda <https://github.com/jsalcedoe/gestortienda.git>



**2.** Se realiza la instalación del motor de base de datos MongoDB, adicional se configura el MongoDB Cloud



**3.** Se realizó la instalación del entorno de desarrollo de Node.js



1. Se realiza la creación del entorno para el desarrollo del proyecto

**COMPROMISOS**

|  |  |  |
| --- | --- | --- |
| **COMPROMISO** | **RESPONSABLE** | **FECHA** |
| Diseñar el entorno grafico el registro de datos en la tabla Dueno de la base  de datos | Cesar Augusto Collazos Perdomo, Cristian Agudelo Zapata, Cindy Natali Torres Castillo | 2022/09/19 |
| Diseño de la base de datos | Julio Ernesto Salcedo Pelaez | 2022/09/19 |
| Desarrollo de los paquetes entity, controller y view para el CRUD a la tabla Dueno | Juan Camilo Cruz Sotelo | 19/10/2020 |

**ASISTENTES**

Estudiante 1 (Julio Ernesto Salcedo Pelaez)

Estudiante 2 (Product owner, Juan Camilo Cruz Sotelo)

Estudiante 3 (Development Team, Cindy Natali Torres Castillo)

Estudiante 4 (Development Team, Cristian Agudelo Zapata)

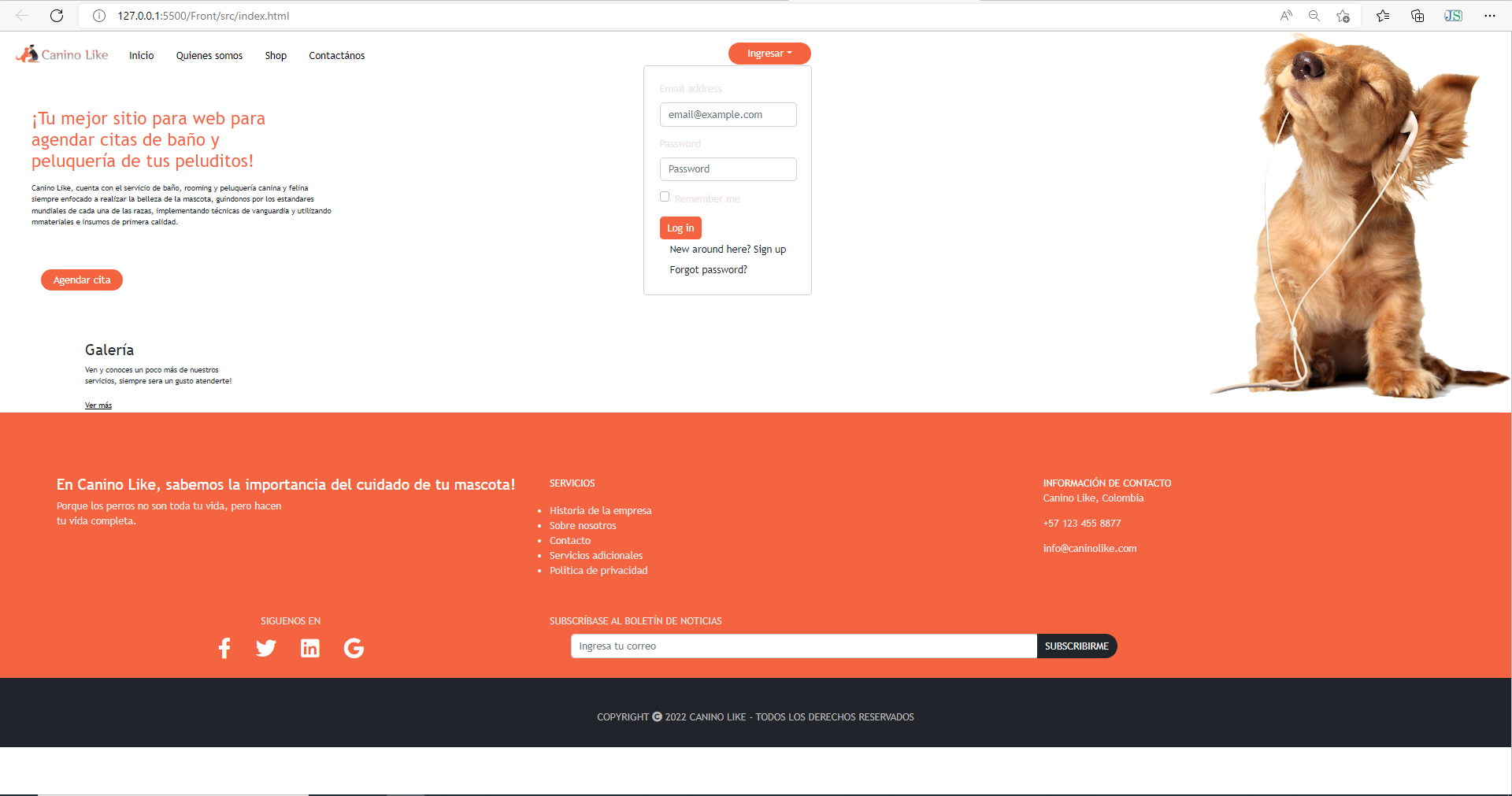
Estudiante 5 (Development Team, Cesar Augusto Collazos Perdomo)

**EVIDENCIAS CORRESPONDIENTES AL DESARROLLO DE LAS TAREAS**

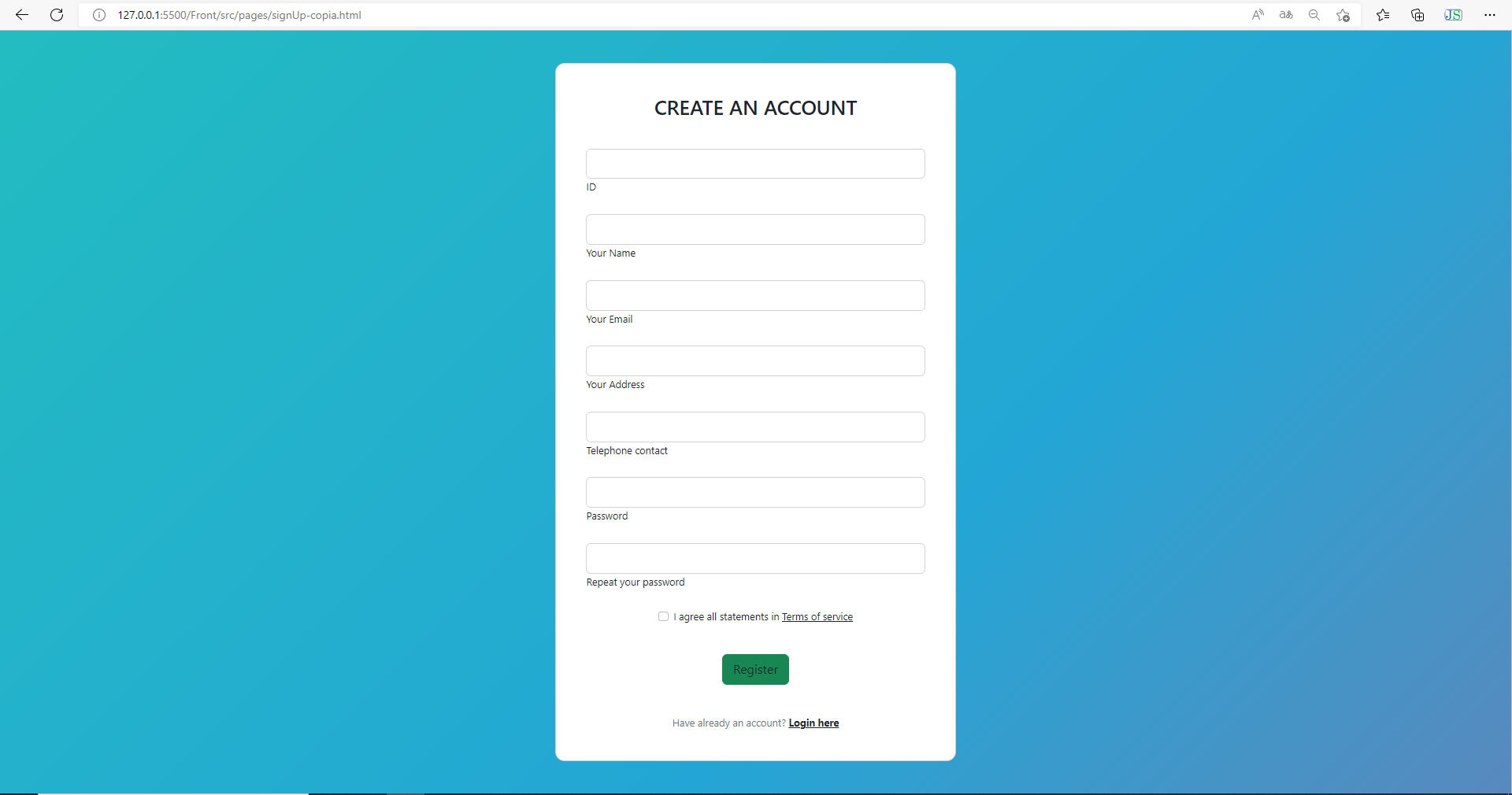
1. Diseño del entorno grafico del proyecto

Para el diseño de la página se utilizó como base el lenguaje de etiquetado HTML y CSS, los cuales se manejaron con bootstrap@5.2.1

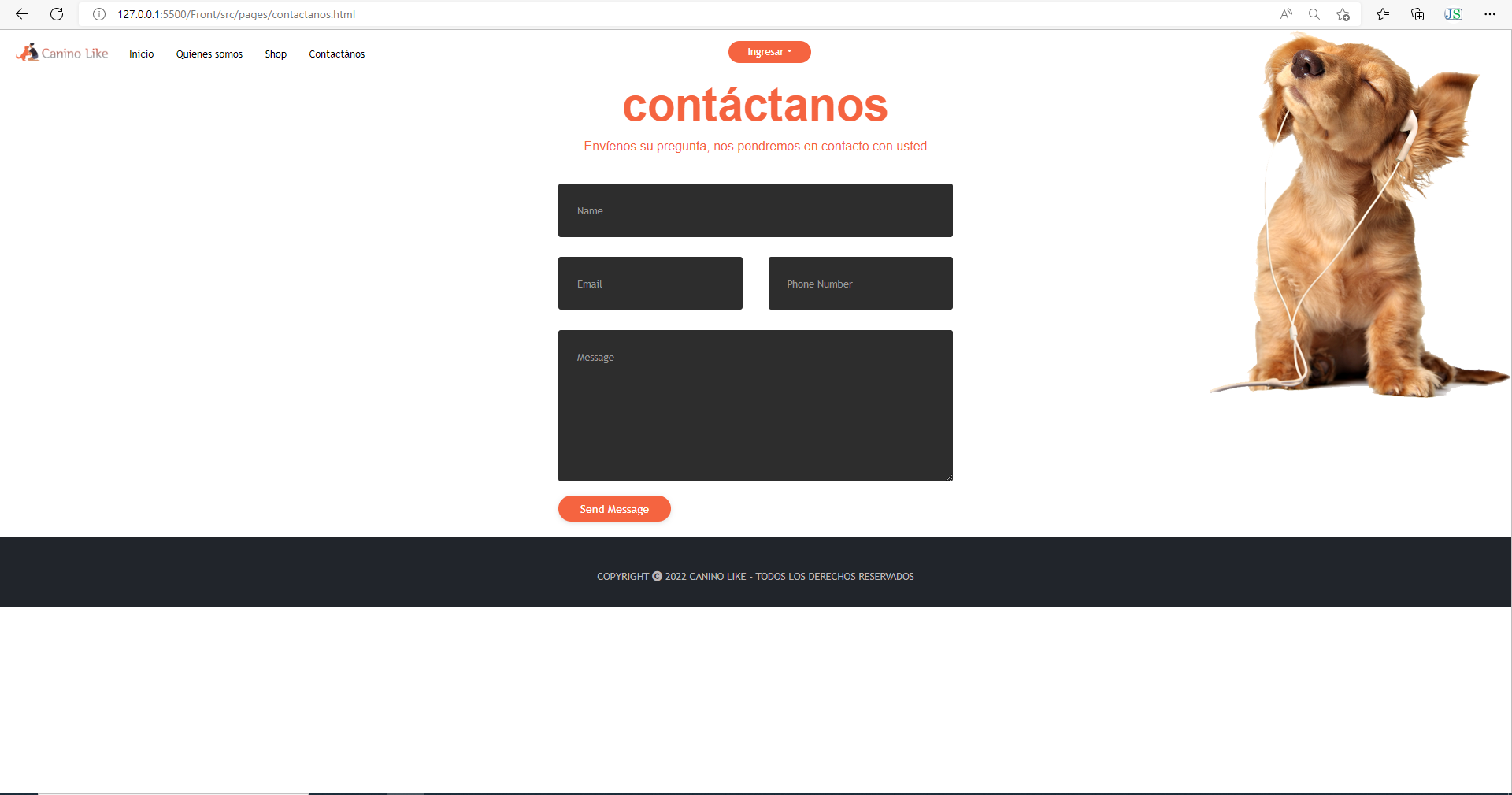
Página de Índex



Página para el registro de clientes

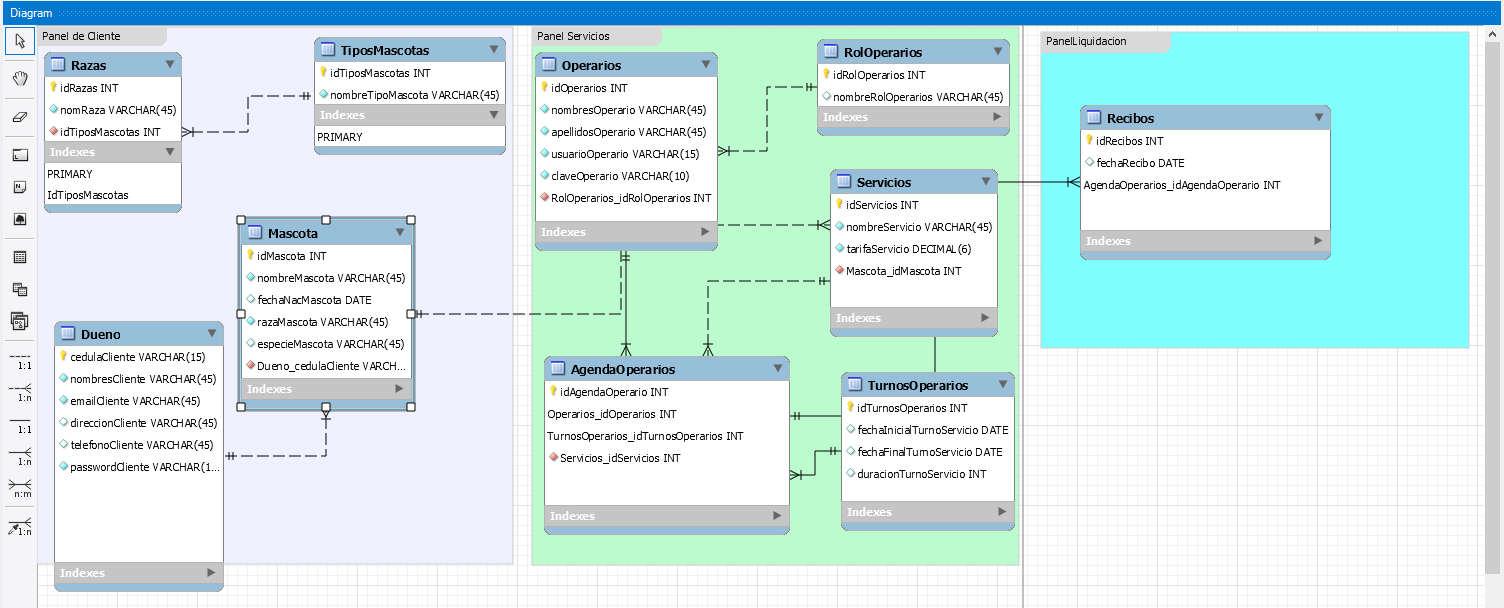


Página de contacto

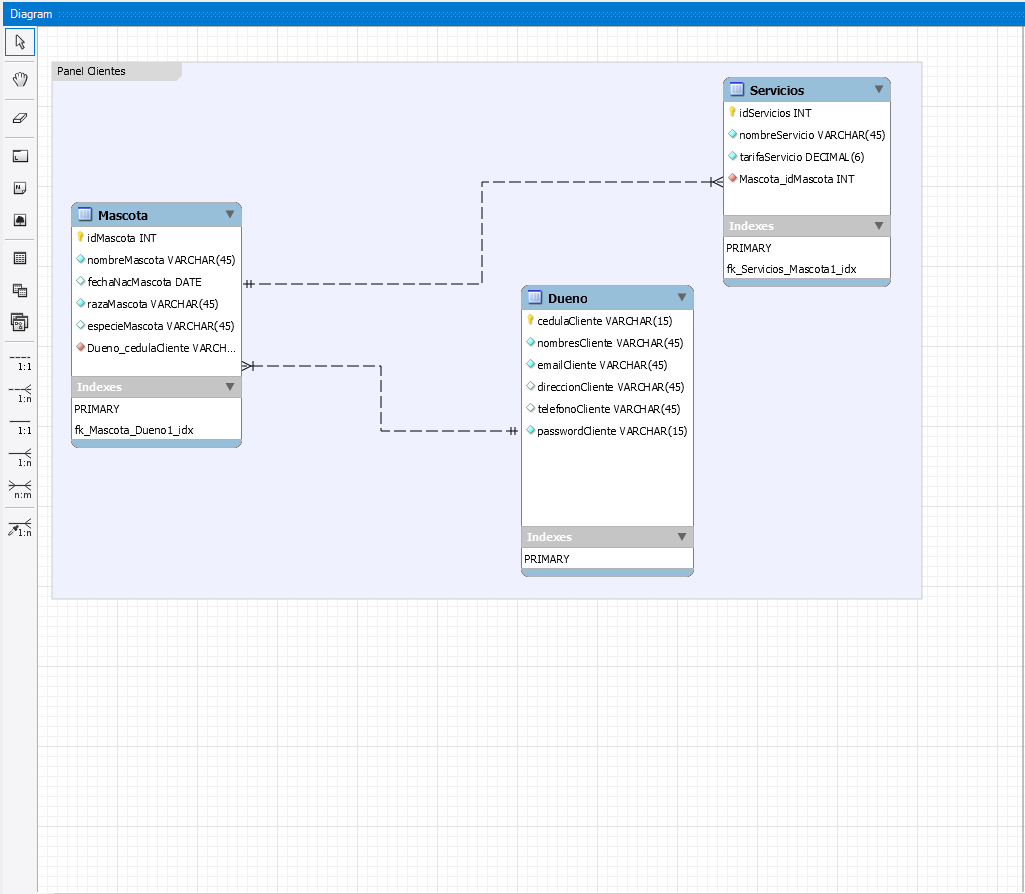


2. Diseño de la base de datos

Para el diseño de la base de datos se realizó el desarrollo de los diferentes modelos lógico, conceptual y físico, los cuales nos brindaron las pautas para el desarrollo del modelo entidad relación en MySQL workbench.



En vista que el modelo ER se tornó extenso y el tiempo que se dispone no alcanza para llevarlo a cabo 100% se decide en reunión hacer un nuevo alcance del proyecto y se concluyó trabajar inicialmente sobre el panel de clientes, para lo cual se realizó un ajuste al diseño, quedando de la siguiente forma.



El script para la creación de la base de datos es el siguiente

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema BDSpaMascota

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema BDSpaMascota

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `BDSpaMascota` DEFAULT CHARACTER SET utf8 COLLATE utf8\_spanish\_ci ;

USE `BDSpaMascota` ;

-- -----------------------------------------------------

-- Table `BDSpaMascota`.`TiposMascotas`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `BDSpaMascota`.`TiposMascotas` (

`idTiposMascotas` INT NOT NULL AUTO\_INCREMENT,

`nombreTipoMascota` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NOT NULL COMMENT 'En esta tabla tenemos todos los tipos de mascotas, por ejemplo, perros, gatos entre otros\n',

PRIMARY KEY (`idTiposMascotas`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `BDSpaMascota`.`Razas`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `BDSpaMascota`.`Razas` (

`idRazas` INT NOT NULL AUTO\_INCREMENT,

`nomRaza` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NOT NULL COMMENT 'Tabla donde se almacenaran las razas de los diferentes tipos de mascotas\n',

`idTiposMascotas` INT NOT NULL,

PRIMARY KEY (`idRazas`),

INDEX `IdTiposMascotas` (`idTiposMascotas` ASC) VISIBLE,

CONSTRAINT `IdTiposMascotas`

FOREIGN KEY (`idTiposMascotas`)

REFERENCES `BDSpaMascota`.`TiposMascotas` (`idTiposMascotas`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `BDSpaMascota`.`Dueno`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `BDSpaMascota`.`Dueno` (

`cedulaCliente` VARCHAR(15) NOT NULL,

`nombresCliente` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NOT NULL,

`emailCliente` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NOT NULL,

`direccionCliente` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NULL,

`telefonoCliente` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NULL,

`passwordCliente` VARCHAR(15) NOT NULL,

PRIMARY KEY (`cedulaCliente`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `BDSpaMascota`.`Mascota`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `BDSpaMascota`.`Mascota` (

`idMascota` INT NOT NULL AUTO\_INCREMENT,

`nombreMascota` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NOT NULL,

`fechaNacMascota` DATE NULL,

`razaMascota` VARCHAR(45) NOT NULL,

`especieMascota` VARCHAR(45) NULL,

`Dueno\_cedulaCliente` VARCHAR(15) NOT NULL,

PRIMARY KEY (`idMascota`),

INDEX `fk\_Mascota\_Dueno1\_idx` (`Dueno\_cedulaCliente` ASC) VISIBLE,

CONSTRAINT `fk\_Mascota\_Dueno1`

FOREIGN KEY (`Dueno\_cedulaCliente`)

REFERENCES `BDSpaMascota`.`Dueno` (`cedulaCliente`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `BDSpaMascota`.`Servicios`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `BDSpaMascota`.`Servicios` (

`idServicios` INT NOT NULL AUTO\_INCREMENT,

`nombreServicio` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NOT NULL,

`tarifaServicio` DECIMAL(6) NOT NULL,

`Mascota\_idMascota` INT NOT NULL,

PRIMARY KEY (`idServicios`),

INDEX `fk\_Servicios\_Mascota1\_idx` (`Mascota\_idMascota` ASC) VISIBLE,

CONSTRAINT `fk\_Servicios\_Mascota1`

FOREIGN KEY (`Mascota\_idMascota`)

REFERENCES `BDSpaMascota`.`Mascota` (`idMascota`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `BDSpaMascota`.`RolOperarios`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `BDSpaMascota`.`RolOperarios` (

`idRolOperarios` INT NOT NULL AUTO\_INCREMENT,

`nombreRolOperarios` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NULL COMMENT 'Tabla que almacena los roles de los operarios como por ejemplo conductor, estilista, veterinario, facturador, administrador',

PRIMARY KEY (`idRolOperarios`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `BDSpaMascota`.`Operarios`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `BDSpaMascota`.`Operarios` (

`idOperarios` INT NOT NULL AUTO\_INCREMENT,

`nombresOperario` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NOT NULL,

`apellidosOperario` VARCHAR(45) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NOT NULL,

`usuarioOperario` VARCHAR(15) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NOT NULL,

`claveOperario` VARCHAR(10) CHARACTER SET 'utf8' COLLATE 'utf8\_spanish\_ci' NOT NULL,

`RolOperarios\_idRolOperarios` INT NOT NULL,

PRIMARY KEY (`idOperarios`),

INDEX `fk\_Operarios\_RolOperarios1\_idx` (`RolOperarios\_idRolOperarios` ASC) VISIBLE,

CONSTRAINT `fk\_Operarios\_RolOperarios1`

FOREIGN KEY (`RolOperarios\_idRolOperarios`)

REFERENCES `BDSpaMascota`.`RolOperarios` (`idRolOperarios`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `BDSpaMascota`.`TurnosOperarios`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `BDSpaMascota`.`TurnosOperarios` (

`idTurnosOperarios` INT NOT NULL AUTO\_INCREMENT,

`fechaInicialTurnoServicio` DATE NULL,

`fechaFinalTurnoServicio` DATE NULL,

`duracionTurnoServicio` INT NULL,

PRIMARY KEY (`idTurnosOperarios`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `BDSpaMascota`.`AgendaOperarios`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `BDSpaMascota`.`AgendaOperarios` (

`idAgendaOperario` INT NOT NULL,

`Operarios\_idOperarios` INT NOT NULL,

`TurnosOperarios\_idTurnosOperarios` INT NOT NULL,

`Servicios\_idServicios` INT NOT NULL,

PRIMARY KEY (`idAgendaOperario`, `Operarios\_idOperarios`, `TurnosOperarios\_idTurnosOperarios`),

INDEX `fk\_Operarios\_has\_TurnosOperarios\_TurnosOperarios1\_idx` (`TurnosOperarios\_idTurnosOperarios` ASC) VISIBLE,

INDEX `fk\_Operarios\_has\_TurnosOperarios\_Operarios1\_idx` (`Operarios\_idOperarios` ASC) VISIBLE,

INDEX `fk\_AgendaOperarios\_Servicios1\_idx` (`Servicios\_idServicios` ASC) VISIBLE,

CONSTRAINT `fk\_Operarios\_has\_TurnosOperarios\_Operarios1`

FOREIGN KEY (`Operarios\_idOperarios`)

REFERENCES `BDSpaMascota`.`Operarios` (`idOperarios`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_Operarios\_has\_TurnosOperarios\_TurnosOperarios1`

FOREIGN KEY (`TurnosOperarios\_idTurnosOperarios`)

REFERENCES `BDSpaMascota`.`TurnosOperarios` (`idTurnosOperarios`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_AgendaOperarios\_Servicios1`

FOREIGN KEY (`Servicios\_idServicios`)

REFERENCES `BDSpaMascota`.`Servicios` (`idServicios`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `BDSpaMascota`.`Recibos`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `BDSpaMascota`.`Recibos` (

`idRecibos` INT NOT NULL,

`fechaRecibo` DATE NULL,

`AgendaOperarios\_idAgendaOperario` INT NOT NULL,

PRIMARY KEY (`idRecibos`, `AgendaOperarios\_idAgendaOperario`),

INDEX `fk\_Recibos\_AgendaOperarios1\_idx` (`AgendaOperarios\_idAgendaOperario` ASC) VISIBLE,

CONSTRAINT `fk\_Recibos\_AgendaOperarios1`

FOREIGN KEY (`AgendaOperarios\_idAgendaOperario`)

REFERENCES `BDSpaMascota`.`AgendaOperarios` (`idAgendaOperario`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

3. Desarrollo de la estructura del Backend con SpringBoot

3.1 Nuevo archivo POM con las dependencias necesarias

<?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>2.7.4</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <groupId>com.co</groupId>

    <artifactId>api</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>api</name>

    <description>Demo project for Spring Boot</description>

    <properties>

        <java.version>17</java.version>

    </properties>

    <dependencies>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-data-jpa</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-data-rest</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-thymeleaf</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-devtools</artifactId>

            <scope>runtime</scope>

            <optional>true</optional>

        </dependency>

        <dependency>

            <groupId>mysql</groupId>

            <artifactId>mysql-connector-java</artifactId>

            <scope>runtime</scope>

        </dependency>

        <dependency>

            <groupId>org.projectlombok</groupId>

            <artifactId>lombok</artifactId>

            <optional>true</optional>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-test</artifactId>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

                <configuration>

                    <excludes>

                        <exclude>

                            <groupId>org.projectlombok</groupId>

                            <artifactId>lombok</artifactId>

                        </exclude>

                    </excludes>

                </configuration>

            </plugin>

        </plugins>

    </build>

</project>

3.2 Configuración del archivo aplication.properties para el funcionamiento del servidor

spring.datasource.url = jdbc:Mysql://localhost:3306/bdspamascota

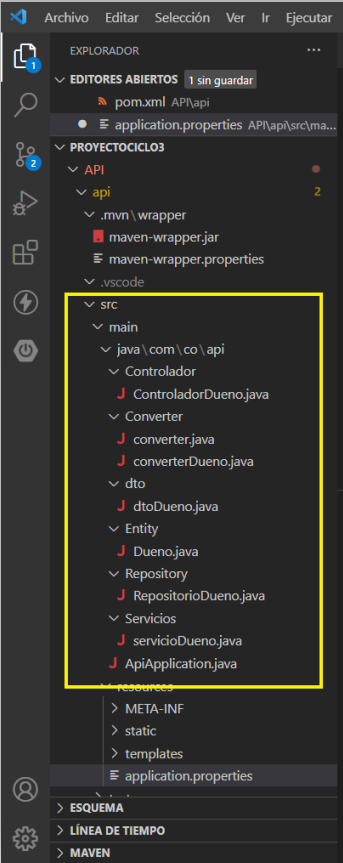
spring.datasource.username = root

spring.datasource.password =

spring.jpa.hibernate.ddl-auto = create

server.port = 8088

3.3 Estructura de paquetes



3.4 Codificación

3.4.1 Creacion de entidades

package com.co.api.Entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

import lombok.AccessLevel;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Entity

@Table(name = "Dueno")

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

@Builder

public class Dueno {

    @Id

    @Column(name = "cedulaCliente")

    private String cedulaCliente;

    @Column(name = "nombresCliente")

    private String nombresCliente;

    @Column(name = "emailCliente")

    private String emailCliente;

    @Column(name ="direccionCliente")

    private String direccionCliente;

    @Column(name = "telefonoCliente")

    private String telefonoCliente;

    @Getter(value = AccessLevel.NONE)

    @Column(name = "passwordCliente")

    private String passwordCliente;

}

3.4.2 Creación de clase de datos con la estructura de la tabla Dueno

package com.co.api.dto;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

@Builder

public class dtoDueno {

    private String cedulaCliente;

    private String nombresCliente;

    private String emailCliente;

    private String direccionCliente;

    private String telefonoCliente;

    private String passwordCliente;

}

3.4.3 Creación de clase absetacta de converter para proporcionar una capa de seguridad a la aplicación.

package com.co.api.Converter;

import java.util.List;

import java.util.stream.Collectors;

public abstract class converter<E,D> {

    public abstract E toEntidad(D object);

    public abstract D todto(E object);

    public List<E> toEntidad(List<D> list){

        return list.stream()

                .map(object -> toEntidad(object))

                .collect(Collectors.toList());

    }

    public List<D> todto(List<E> list){

        return list.stream()

                .map(object -> todto(object))

                .collect(Collectors.toList());

    }

}

3.4.4 Creación de la clase converterDueno

package com.co.api.Converter;

import com.co.api.Entity.Dueno;

import com.co.api.dto.dtoDueno;

public class converterDueno extends converter<Dueno,dtoDueno> {

    @Override

    public Dueno toEntidad(dtoDueno object) {

        return Dueno.builder()

            .cedulaCliente(object.getCedulaCliente())

            .nombresCliente(object.getNombresCliente())

            .emailCliente(object.getEmailCliente())

            .direccionCliente(object.getDireccionCliente())

            .telefonoCliente(object.getTelefonoCliente())

            .passwordCliente(object.getPasswordCliente())

            .build();

    }

    @Override

    public dtoDueno todto(Dueno object) {

            return dtoDueno.builder()

            .cedulaCliente(object.getCedulaCliente())

            .nombresCliente(object.getNombresCliente())

            .emailCliente(object.getEmailCliente())

            .direccionCliente(object.getDireccionCliente())

            .telefonoCliente(object.getTelefonoCliente())

            .build();

    }

}

3.4.5 Creación de la interfaz RepositorioDueno que hereda de JpaRepository

package com.co.api.Repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.co.api.Entity.Dueno;

@Repository

public interface RepositorioDueno extends JpaRepository<Dueno, String> {

}

3.4.6 Creacion de la clase servicioDueno

package com.co.api.Servicios;

import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.stereotype.Service;

import org.springframework.web.server.ResponseStatusException;

import com.co.api.Converter.converterDueno;

import com.co.api.Entity.Dueno;

import com.co.api.Repository.RepositorioDueno;

import com.co.api.dto.dtoDueno;

@Service

public class servicioDueno {

    @Autowired

    private RepositorioDueno repositorioDueno;

    private converterDueno converterDueno = new converterDueno();

    // listar dueños

    public List<dtoDueno> findAll(){

        return converterDueno.todto(repositorioDueno.findAll());

    }

    // agregar dueño

    public dtoDueno insert(dtoDueno dueno){

        if(repositorioDueno.existsById(dueno.getCedulaCliente()))

            throw new ResponseStatusException(HttpStatus.CONFLICT,"El Dueño ya existe...");

            return converterDueno.todto(repositorioDueno.save(converterDueno.toEntidad(dueno)));

    }

    // consultar dueño (cedula)

    public dtoDueno FindById(String cedulaDueno){

        Optional<Dueno> dueno = repositorioDueno.findById(cedulaDueno);

        if(dueno.isEmpty())

            throw new ResponseStatusException(HttpStatus.CONFLICT,"El Dueño no existe");

        return converterDueno.todto(dueno.get());

    }

    // editar un dueño

    public dtoDueno update(dtoDueno dueno){

        if(!repositorioDueno.existsById(dueno.getCedulaCliente()))

            throw new ResponseStatusException(HttpStatus.CONFLICT,"El Dueño no existe");

            return converterDueno.todto(repositorioDueno.save(converterDueno.toEntidad(dueno)));

    }

    // eliminar un dueño

    public dtoDueno deleteById(String cedulaDueno){

        Optional<Dueno> dueno = repositorioDueno.findById(cedulaDueno);

        if(dueno.isEmpty())

        throw new ResponseStatusException(HttpStatus.CONFLICT,"El Dueño no existe");

        dtoDueno dtodueno = converterDueno.todto(dueno.get());

        repositorioDueno.deleteById(cedulaDueno);

        return dtodueno;

    }

}

3.4.7 Creacion de la clase ControladorDueno

package com.co.api.Controlador;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

import com.co.api.Servicios.servicioDueno;

import com.co.api.dto.dtoDueno;

@RestController

@RequestMapping(path = "/api/dueno")

@CrossOrigin(origins = "\*", methods = {RequestMethod.GET,RequestMethod.POST,RequestMethod.PUT, RequestMethod.DELETE})

public class ControladorDueno {

    @Autowired

    private servicioDueno servicioDueno;

    // listar dueños

    @GetMapping

    public ResponseEntity<?> findAll(){

        return new ResponseEntity<>(servicioDueno.findAll(),HttpStatus.OK);

    }

    // agregar dueño

    @PostMapping

    public ResponseEntity<?> insert(@RequestBody dtoDueno dueno){

        return new ResponseEntity<>(servicioDueno.insert(dueno),HttpStatus.CREATED);

    }

    // consultar dueño (cedula)

    @GetMapping("{cedulaCliente}")

    public ResponseEntity<?> finById(@PathVariable String cedulaCliente){

        return new ResponseEntity<>(servicioDueno.FindById(cedulaCliente),HttpStatus.OK);

    }

    // editar un dueño

    @PutMapping

    public ResponseEntity<?> update(@RequestBody dtoDueno dueno){

        return new ResponseEntity<>(servicioDueno.update(dueno),HttpStatus.CREATED);

    }

    // eliminar un dueño

    @DeleteMapping("{cedulaCliente}")

    public ResponseEntity<?> deleteById(@PathVariable String cedulaCliente){

        return new ResponseEntity<>(servicioDueno.deleteById(cedulaCliente),HttpStatus.OK);

    }

}