INSTITUTO DE EDUCACIÓN SUPERIOR CIBERTEC
DIRECCIÓN ACADÉMICA
CARRERA PROFESIONALES

CURSO	:	**Desarrollo de Aplicaciones Web I**
PROFESOR	:	**César Enrique Santos Torres**
CICLO	:	**Quinto**
SECCIÓN	:	
GRUPO	:	
FECHA	:	
DURACIÓN	:	**50 minutos**

NOTA

ALUMNO (A) : Jesus Medina Lazaro

CASO DE LABORATORIO 3 (EF)

Consideraciones generales:

- El laboratorio consta de 1 Crud implementado con Spring RESTFul + Spring Data JPA, cada operación del CRUD deberá ir acompañada (De forma obligatoria) de capturas de pantalla de lo implementado.

- Sólo debe subir este documento, con sus evidencias y respuestas en él. El código fuente del proyecto debe ser subido a Github (Adjuntar link del repositorio). No se aceptará código zipeado.

- El nombre del presente archivo deberá tener la siguiente estructura: "DAWI-APELLIDOPATERNO-APELLIDOMATERNO-NOMBRES.pdf".

LOGRO DE LA EVALUACION:
Al término de la evaluación, el alumno deberá implementar un CRUD con Spring RESTful, dicho CRUD deberá incluir las siguientes operaciones:

- **/all** (Consulta de todos los items)
- **/detail** (Consulta de un item)
- **/update** (Actualización de un item)
- **/delete/{id}** (Eliminación de un item)
- **/create** (Creación de un item)

CONSOLIDADO

Pregunta	Puntaje		Llenar solo en caso de Recalificación justificada	
	Máximo	Obtenido	Sustento	Puntaje
1	5			
2	5			
3	5			
4	5			
Total	20			

Nota Recalificada	

Alcance de la prueba

Implementar un CRUD de la siguiente tabla (Deberá crear un base de datos "**fabric**" y en ella la tabla especificada):

```
CREATE TABLE car (
    car_id INT AUTO_INCREMENT PRIMARY KEY,
    make VARCHAR(50),
    model VARCHAR(50),
    year INT,
    vin VARCHAR(50),
    license_plate VARCHAR(20),
    owner_name VARCHAR(100),
    owner_contact VARCHAR(50),
    purchase_date DATE,
    mileage INT,
    engine_type VARCHAR(50),
    color VARCHAR(30),
    insurance_company VARCHAR(100),
    insurance_policy_number VARCHAR(50),
    registration_expiration_date DATE,
    service_due_date DATE
);
```

Ejecutar los siguientes registros de la tabla previa:

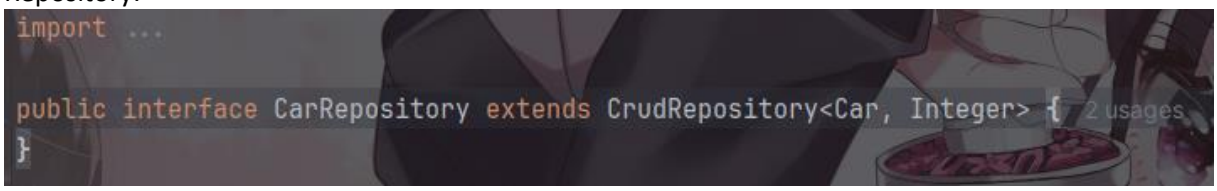
```
INSERT INTO car (make, model, year, vin, license_plate, owner_name,
owner_contact, purchase_date, mileage, engine_type, color, insurance_company,
insurance_policy_number, registration_expiration_date, service_due_date) VALUES
('Toyota', 'Corolla', 2018, '1NXBR12E3YZ123456', 'ABC123', 'Juan Perez', '555-
1234', '2018-01-15', 25000, 'Gasoline', 'Red', 'Seguros del Sol', 'INS123456',
'2025-01-15', '2024-07-15'),
('Honda', 'Civic', 2020, '2HGES26785H654321', 'XYZ789', 'Maria Lopez', '555-
5678', '2020-05-10', 15000, 'Gasoline', 'Blue', 'ProtectAuto', 'INS789012',
'2026-05-10', '2025-11-10'),
('Ford', 'Focus', 2019, '1FAFP34N06W765432', 'DEF456', 'Carlos Jimenez', '555-
8765', '2019-03-20', 30000, 'Diesel', 'Black', 'AutoSeguro', 'INS345678', '2024-
03-20', '2023-09-20'),
('Chevrolet', 'Malibu', 2017, '1G1ZD5ST1HF123456', 'GHI123', 'Ana Martinez',
'555-4321', '2017-08-05', 45000, 'Gasoline', 'White', 'AutoProtegido',
'INS901234', '2023-08-05', '2023-02-05'),
('Nissan', 'Altima', 2021, '1N4AL11D75C123456', 'JKL456', 'Luis Rodriguez',
'555-3456', '2021-06-15', 10000, 'Gasoline', 'Silver', 'SegurosTotal',
'INS567890', '2027-06-15', '2026-12-15'),
('Mazda', '3', 2022, 'JM1BPACL1M1234567', 'MNO789', 'Sofia Gonzalez', '555-
6789', '2022-09-20', 5000, 'Gasoline', 'Gray', 'ProtecCar', 'INS234567', '2028-
09-20', '2028-03-20'),
('Hyundai', 'Elantra', 2016, 'KMHDH4AE6DU123456', 'PQR123', 'Pedro Alvarez',
'555-9876', '2016-11-05', 60000, 'Gasoline', 'Green', 'AseguraTodo',
'INS876543', '2022-11-05', '2022-05-05'),
('Kia', 'Optima', 2015, '5XXGT4L34FG123456', 'STU456', 'Isabel Ramirez', '555-
2345', '2015-02-10', 75000, 'Gasoline', 'Yellow', 'CarSeguros', 'INS345678',
'2021-02-10', '2020-08-10'),
('Volkswagen', 'Jetta', 2014, '3VW2K7AJ6EM123456', 'VWX789', 'Manuel Diaz',
'555-8765', '2014-05-15', 90000, 'Gasoline', 'Blue', 'SeguroMovil', 'INS901234',
'2020-05-15', '2019-11-15'),
```

```
('Subaru', 'Impreza', 2013, 'JF1GJAA67DG123456', 'YZA123', 'Patricia Sanchez',
'555-6543', '2013-12-01', 100000, 'Gasoline', 'Red', 'AsegurAuto', 'INS567890',
'2019-12-01', '2019-06-01');
```

Entity:

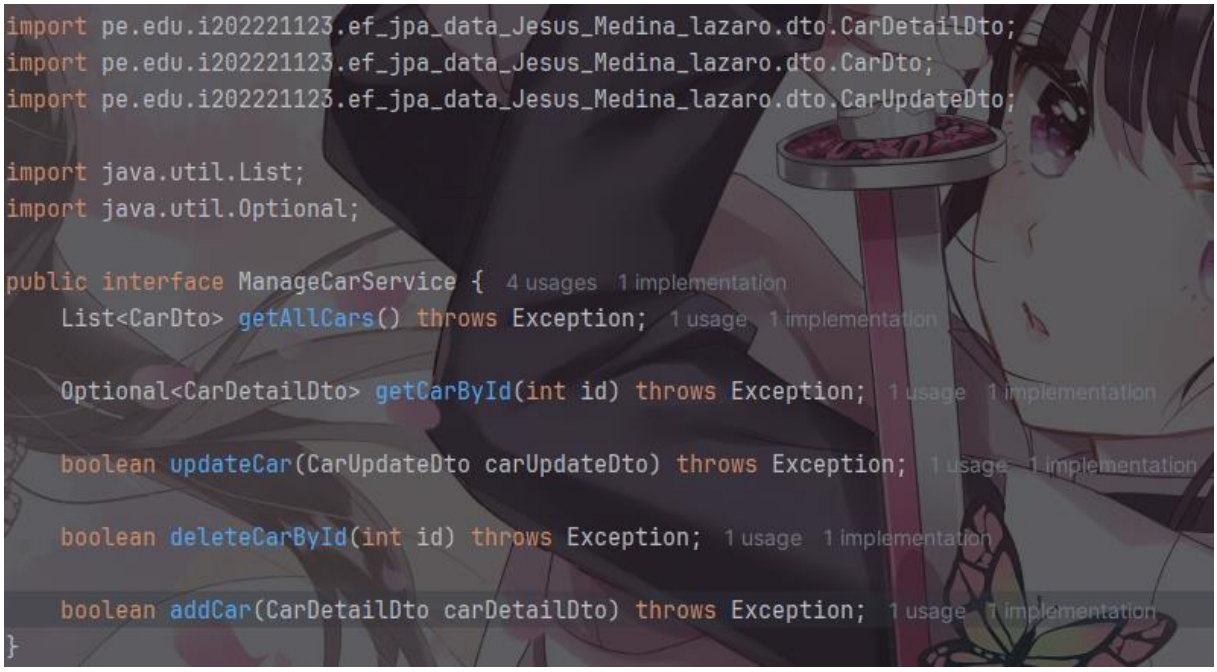
```java
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.*;

import java.util.Date;

@Entity  10 usages
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Car {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer car_id;
    private String make;
    private String model;
    private Integer year;
    private String vin;
    private String license_plate;
    private String owner_name;
    private String owner_contact;
    private Date purchase_date;
    private Integer mileage;
    private String engine_type;
    private String color;
    private String insurance_company;
    private String insurance_policy_number;
    private Date registration_expiration_date;
    private Date service_due_date;
}
```

Repository:



```
import ...

public interface CarRepository extends CrudRepository<Car, Integer> {  2 usages
}
```

Service – Interface:



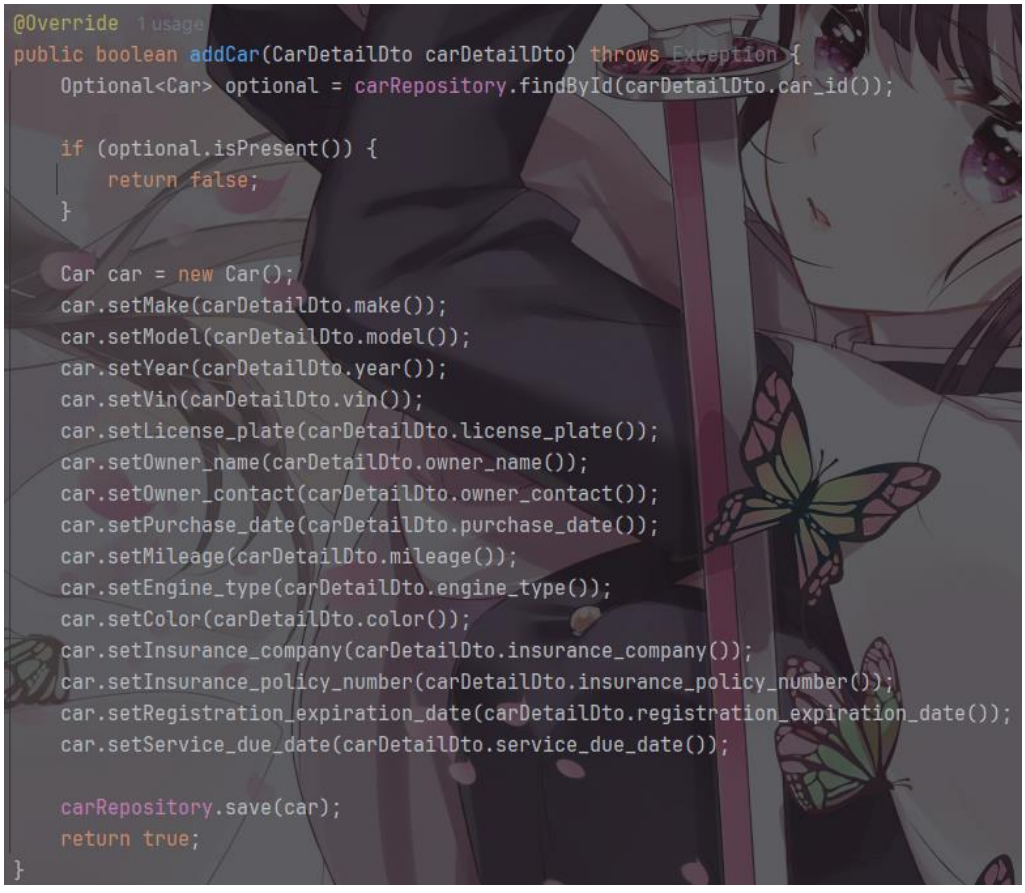
```
import pe.edu.i202221123.ef_jpa_data_Jesus_Medina_lazaro.dto.CarDetailDto;
import pe.edu.i202221123.ef_jpa_data_Jesus_Medina_lazaro.dto.CarDto;
import pe.edu.i202221123.ef_jpa_data_Jesus_Medina_lazaro.dto.CarUpdateDto;

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

public interface ManageCarService {  4 usages  1 implementation
    List<CarDto> getAllCars() throws Exception;  1 usage  1 implementation

    Optional<CarDetailDto> getCarById(int id) throws Exception;  1 usage  1 implementation

    boolean updateCar(CarUpdateDto carUpdateDto) throws Exception;  1 usage  1 implementation

    boolean deleteCarById(int id) throws Exception;  1 usage  1 implementation

    boolean addCar(CarDetailDto carDetailDto) throws Exception;  1 usage  1 implementation
}
```

Service – Impl:

```
@Override  1 usage
public List<CarDto> getAllCars() throws Exception {
    List<CarDto> cars = new ArrayList<CarDto>();
    Iterable<Car> iterable = carRepository.findAll();

    iterable.forEach(car -> {
        cars.add(new CarDto(
                car.getCar_id(),
                car.getModel(),
                car.getYear(),
                car.getMileage(),
                car.getColor()
        ));
    });

    return cars;
}
```

```
@Override  1 usage
public Optional<CarDetailDto> getCarById(int id) throws Exception {
    Optional<Car> optional = carRepository.findById(id);

    return optional.map(car -> new CarDetailDto(
            car.getCar_id(),
            car.getMake(),
            car.getModel(),
            car.getYear(),
            car.getVin(),
            car.getLicense_plate(),
            car.getOwner_name(),
            car.getOwner_contact(),
            car.getPurchase_date(),
            car.getMileage(),
            car.getEngine_type(),
            car.getColor(),
            car.getInsurance_company(),
            car.getInsurance_policy_number(),
            car.getRegistration_expiration_date(),
            car.getService_due_date()
    ));
}
```

```java
@Override 1 usage
public boolean updateCar(CarUpdateDto carUpdateDto) throws Exception {
    Optional<Car> optional = carRepository.findById(carUpdateDto.car_id());

    return optional.map(car -> {
        car.setMake(carUpdateDto.make());
        car.setModel(carUpdateDto.model());
        car.setYear(carUpdateDto.year());
        car.setOwner_name(carUpdateDto.owner_name());
        car.setColor(carUpdateDto.color());

        carRepository.save(car);
        return true;
    }).orElse( other: false);
}

@Override 1 usage
public boolean deleteCarById(int id) throws Exception {
    Optional<Car> optional = carRepository.findById(id);

    return optional.map(car -> {
        carRepository.delete(car);
        return true;
    }).orElse( other: false);
}
```



```java
@Override 1 usage
public boolean addCar(CarDetailDto carDetailDto) throws Exception {
    Optional<Car> optional = carRepository.findById(carDetailDto.car_id());

    if (optional.isPresent()) {
        return false;
    }

    Car car = new Car();
    car.setMake(carDetailDto.make());
    car.setModel(carDetailDto.model());
    car.setYear(carDetailDto.year());
    car.setVin(carDetailDto.vin());
    car.setLicense_plate(carDetailDto.license_plate());
    car.setOwner_name(carDetailDto.owner_name());
    car.setOwner_contact(carDetailDto.owner_contact());
    car.setPurchase_date(carDetailDto.purchase_date());
    car.setMileage(carDetailDto.mileage());
    car.setEngine_type(carDetailDto.engine_type());
    car.setColor(carDetailDto.color());
    car.setInsurance_company(carDetailDto.insurance_company());
    car.setInsurance_policy_number(carDetailDto.insurance_policy_number());
    car.setRegistration_expiration_date(carDetailDto.registration_expiration_date());
    car.setService_due_date(carDetailDto.service_due_date());

    carRepository.save(car);
    return true;
}
```

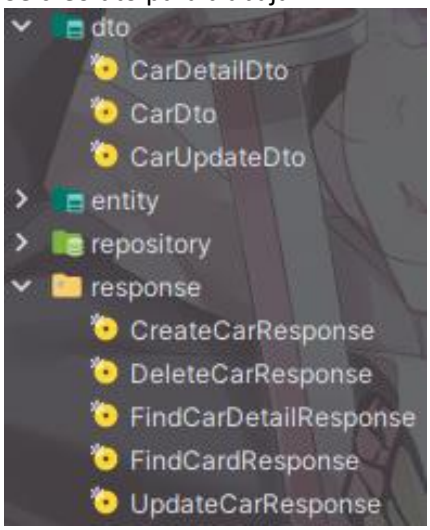
Controller:

```
@GetMapping("/all")  no usages
public FindCardResponse findUser() {
    try {
        List<CarDto> cars = manageCarService.getAllCars();

        if (cars.isEmpty()) {
            return new FindCardResponse( code: "02",  error: "Cars not found",  cars: null);
        }

        return new FindCardResponse( code: "01",  error: null, cars);
    } catch (Exception e) {
        e.printStackTrace();
        return new FindCardResponse( code: "99",  error: "Error ocurred: " + e.getMessage(),  cars: null);
    }
}
```

```
@GetMapping("/detail")  no usages
public FindCarDetailResponse findCar(@RequestParam String id) {
    try {
        Optional<CarDetailDto> optional = manageCarService.getCarById(Integer.parseInt(id));

        return optional.map(car →
                new FindCarDetailResponse( code: "01",  error: null, car)
        ).orElse(
                new FindCarDetailResponse( code: "02",  error: "Car not found",  car: null)
        );
    } catch (Exception e) {
        e.printStackTrace();
        return new FindCarDetailResponse( code: "99",  error: "Car not found" + e.getMessage(),  car: null);
    }
}

@PutMapping("/update")  no usages
public UpdateCarResponse updateCar(@RequestBody CarUpdateDto carUpdateDto) {
    try {
        if (manageCarService.updateCar(carUpdateDto)) {
            return new UpdateCarResponse( code: "01",  error: null);
        } else {
            return new UpdateCarResponse( code: "02",  error: "Car not found");
        }
    } catch (Exception e) {
        return new UpdateCarResponse( code: "99",  error: "Error ocurred: " + e.getMessage());
    }
}
```

```java
@DeleteMapping("/delete/{id}")  no usages
public DeleteCarResponse deleteCar(@PathVariable String id) {
    try {
        if (manageCarService.deleteCarById(Integer.parseInt(id))) {
            return new DeleteCarResponse( code: "01",  error: null);
        } else {
            return new DeleteCarResponse( code: "02",  error: "Car not found");
        }
    } catch (Exception e) {
        return new DeleteCarResponse( code: "99",  error: "Error ocurred: " + e.getMessage());
    }
}

@PostMapping("/create")  no usages
public CreateCarResponse createCar(@RequestBody CarDetailDto carDetailDto) {
    try {
        if (manageCarService.addCar(carDetailDto)) {
            return new CreateCarResponse( code: "01",  error: null);
        } else {
            return new CreateCarResponse( code: "02",  error: "Car already exist");
        }
    } catch (Exception e) {
        e.printStackTrace();
        return new CreateCarResponse( code: "99",  error: "Error ocurred: " + e.getMessage());
    }
}
```

Se creo dto para trabajar:



Link del Repositorio en GitHub: https://github.com/LGsus113/Examen-Final-DAWI.git