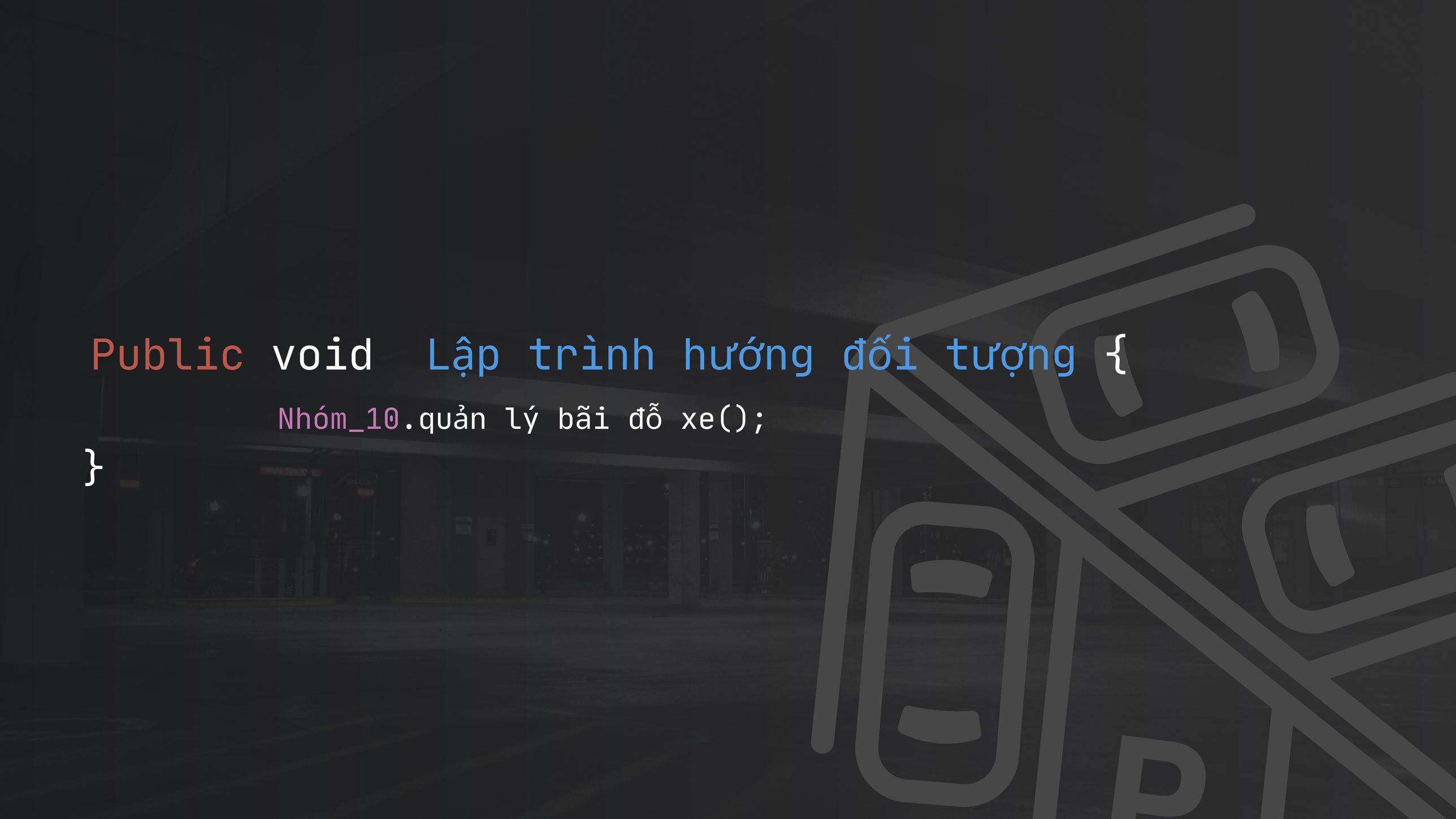


```
Public void  Lập trình hướng đối tượng {
        Nhóm_10.quản lý bãi đỗ xe();
}
```

Contents

- Giới thiệu thành viên
- Giới thiệu & Phân tích bài toán
- Công nghệ sử dụng
- Kiến trúc hệ thống
- Thiết kế hệ thống
- Các tính chất của OOP

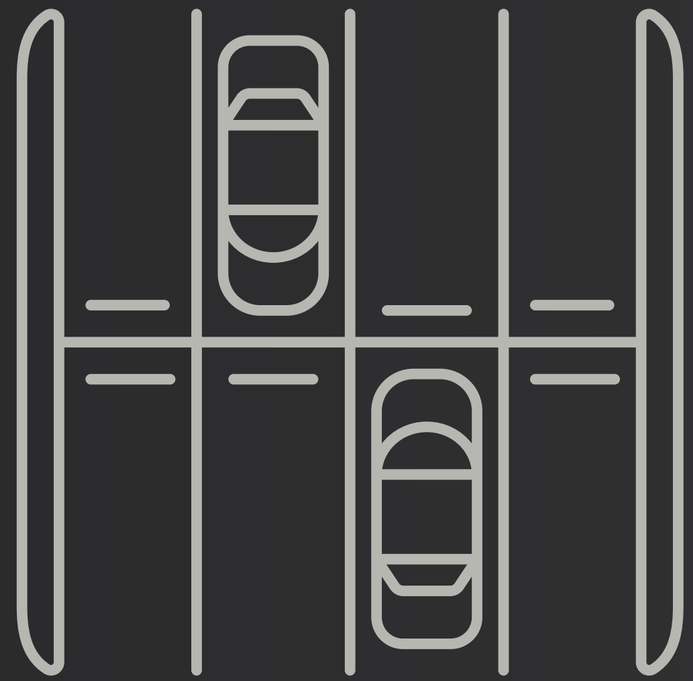
```
Public class Giới thiệu thành viên {

        new Student("Phạm Văn Đồng","Fullstack");

        new Student("Phạm Đức Long","Thiết kế hệ thống, service, thuyết trình");

        new Student("Phan Đức Duy","Báo cáo, slide, entity, DTOs");

        new Student("Nguyễn Bá Hoàng","Usecase, Database, service");

        new Student("Phạm Quốc Minh","Frontend, ERD, Controller");

                                                                    }
```

Giới thiệu bài toán

Xây dựng một hệ thống quản lý thông tin bãi gửi xe cơ bản, sử dụng app cho phép người dùng có thể đăng ký gửi xe online

Gồm các chức năng cơ bản như:

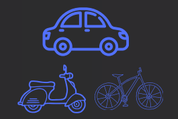
- đăng ký thông tin người dùng
- đăng ký gửi xe
- quản lý xe
- quản lý vé xe
- ...



Các thực thể chính

Khách hàng

Xe



Loại xe

Bãi đỗ

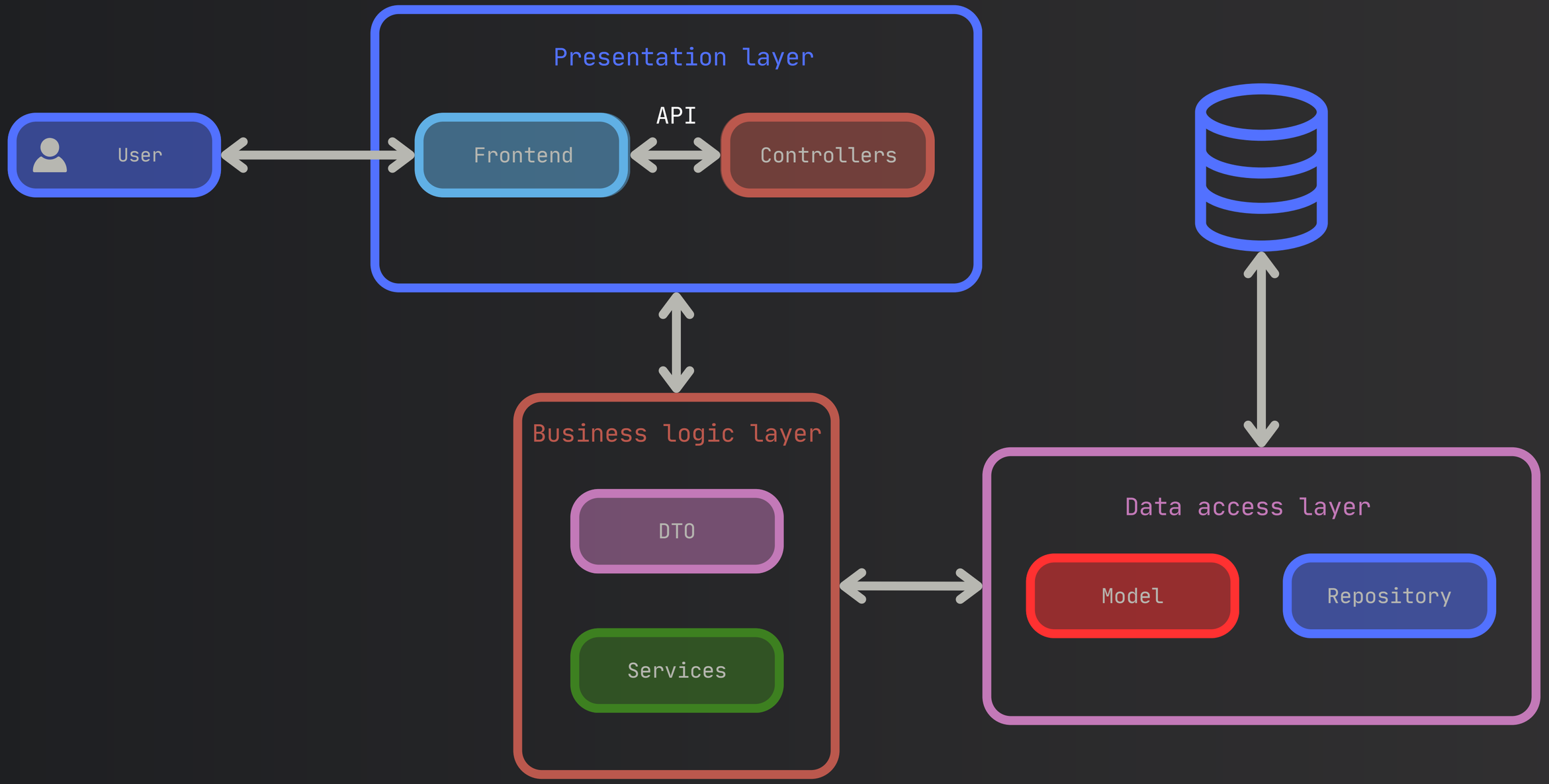
Vé

Công nghệ sử dụng

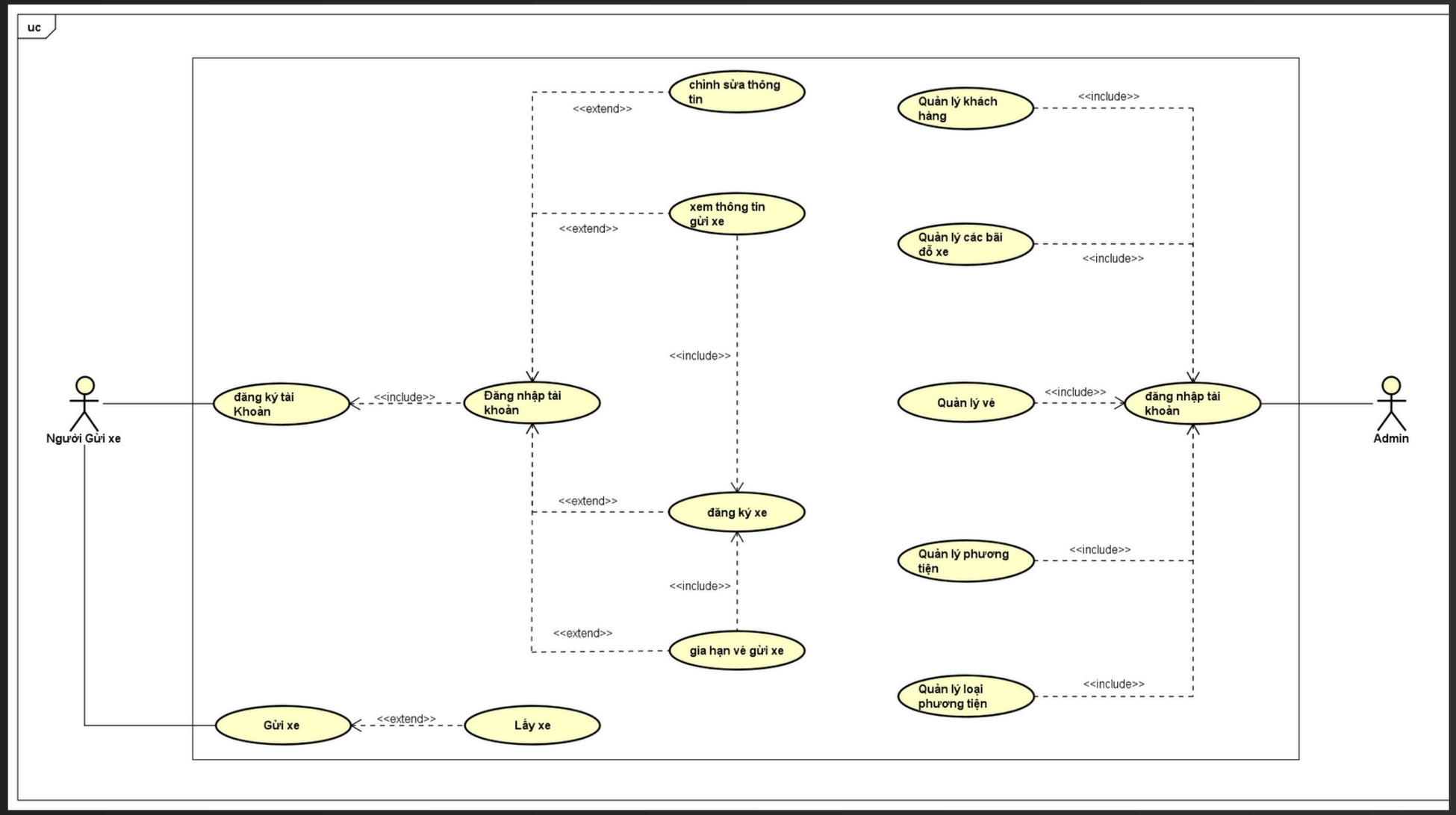
Backend: java spring boot

Frontend: ReactJS

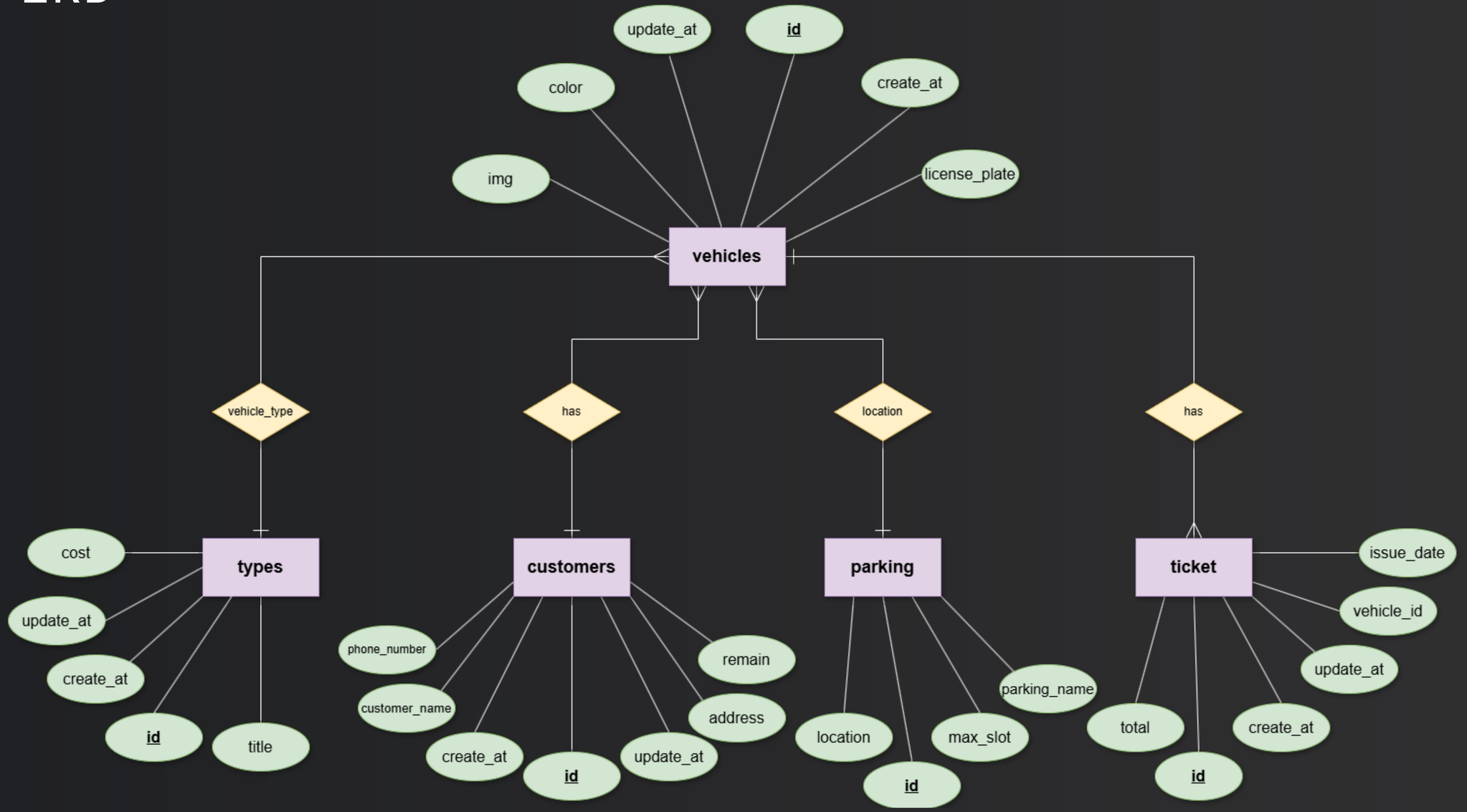
Database: SQL server


Presentation layer
User
Frontend
API
Controllers
Business logic layer
DTO
Services
Data access layer
Model
Repository

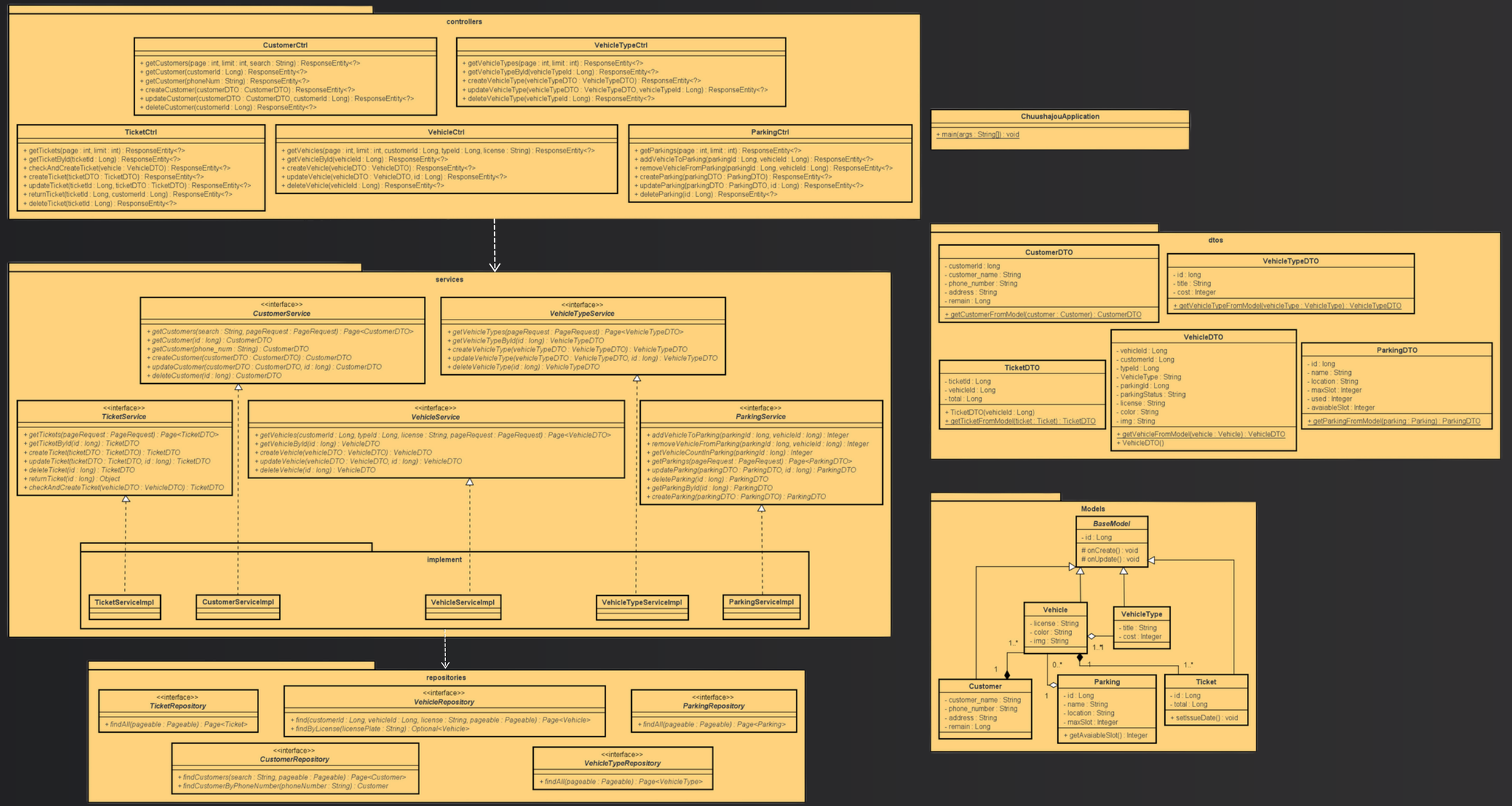
Usecase


uc
chỉnh sửa thông tin
<<extend>>
xem thông tin gửi xe
<<extend>>
<<include>>
đăng ký tài Khoản
<<include>>
Đăng nhập tài khoản
Người Gửi xe
<<extend>>
đăng ký xe
<<include>>
<<extend>>
gia hạn vé gửi xe
Gửi xe
<<extend>>
Lấy xe
Quản lý khách hàng
<<include>>
Quản lý các bãi đỗ xe
<<include>>
Quản lý vé
<<include>>
đăng nhập tài khoản
Admin
Quản lý phương tiện
<<include>>
Quản lý loại phương tiện
<<include>>

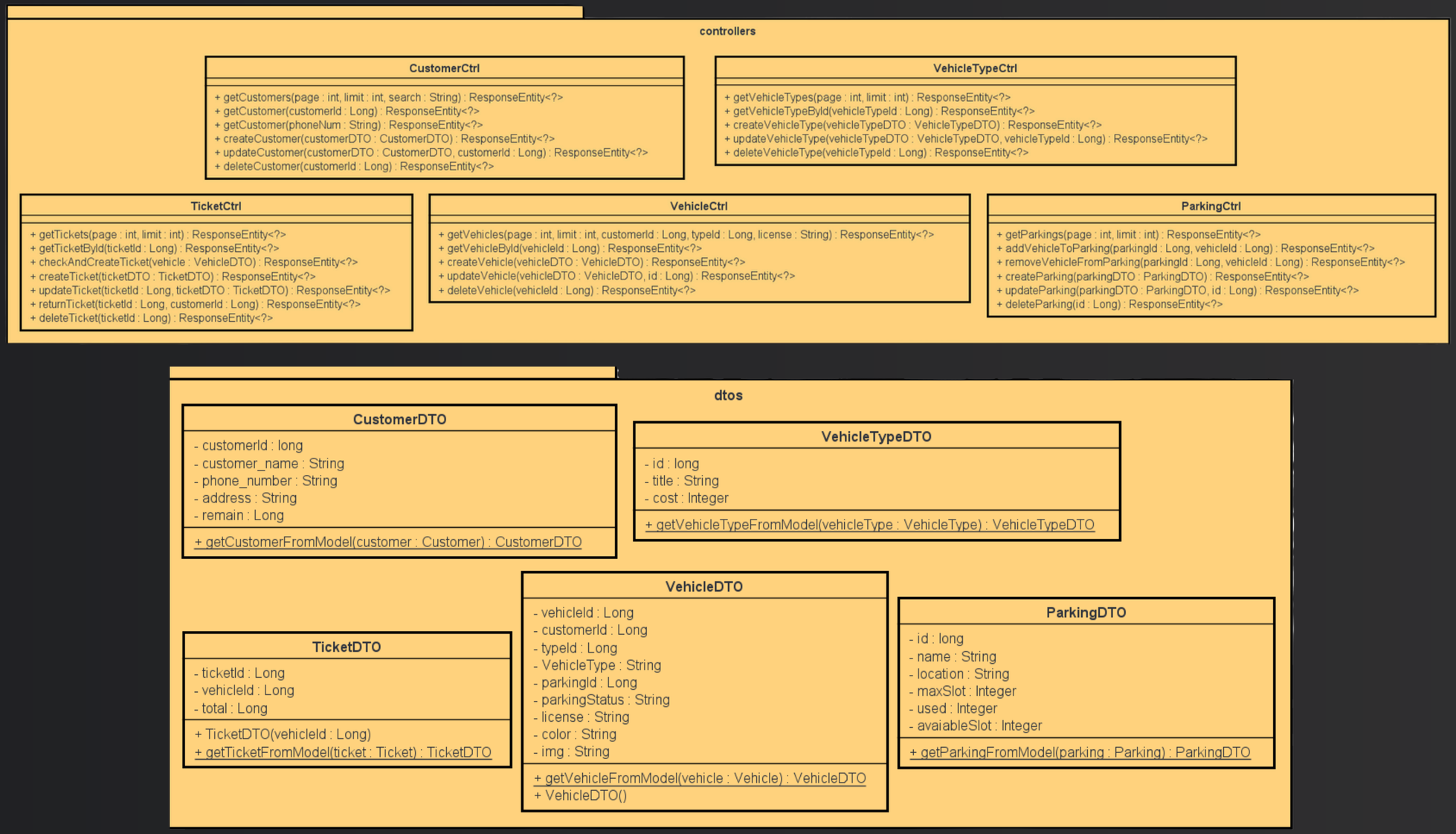
ERD


update_at
id
color
create_at
img
license_plate
vehicles
vehicle_type
has
location
has
cost
types
update_at
create_at
id
title
customers
phone_number
customer_name
create_at
id
update_at
address
remain
parking
location
id
max_slot
parking_name
ticket
issue_date
vehicle_id
update_at
total
id
create_at

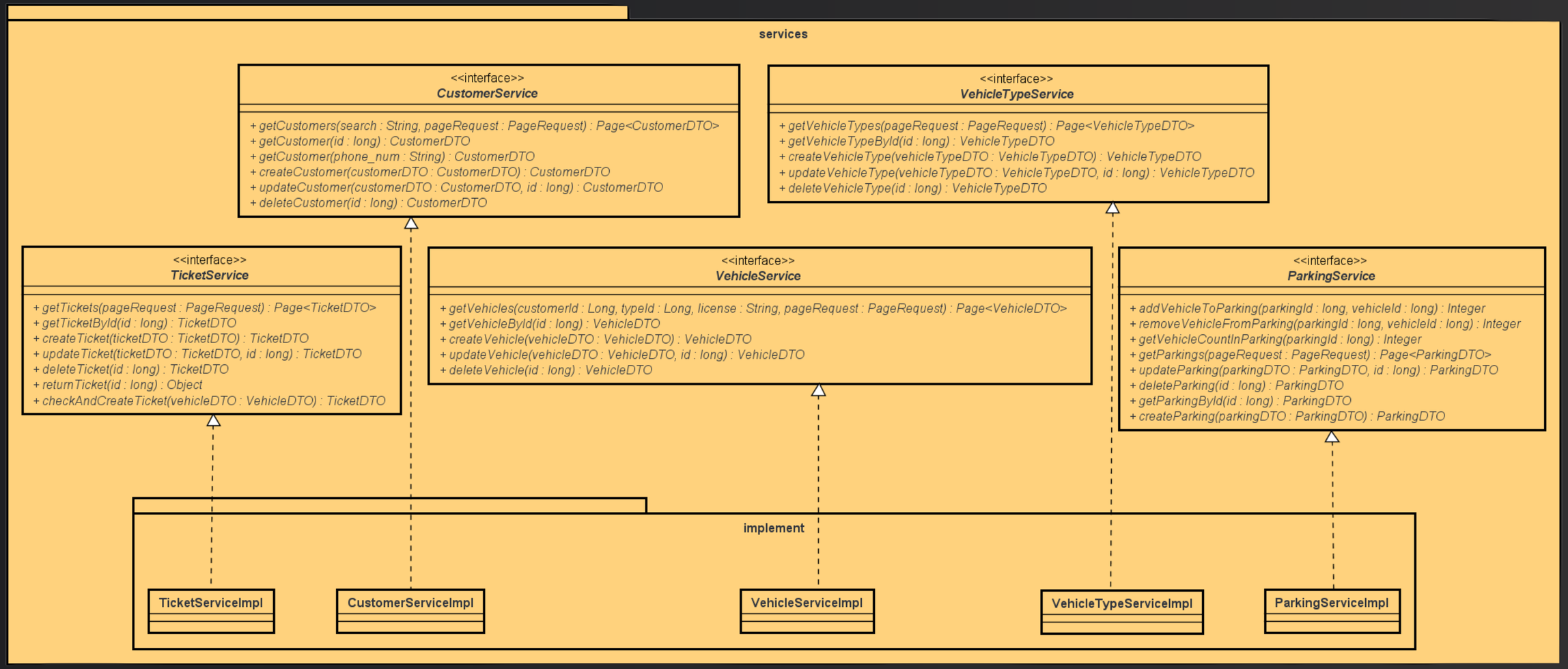
Class Diagram



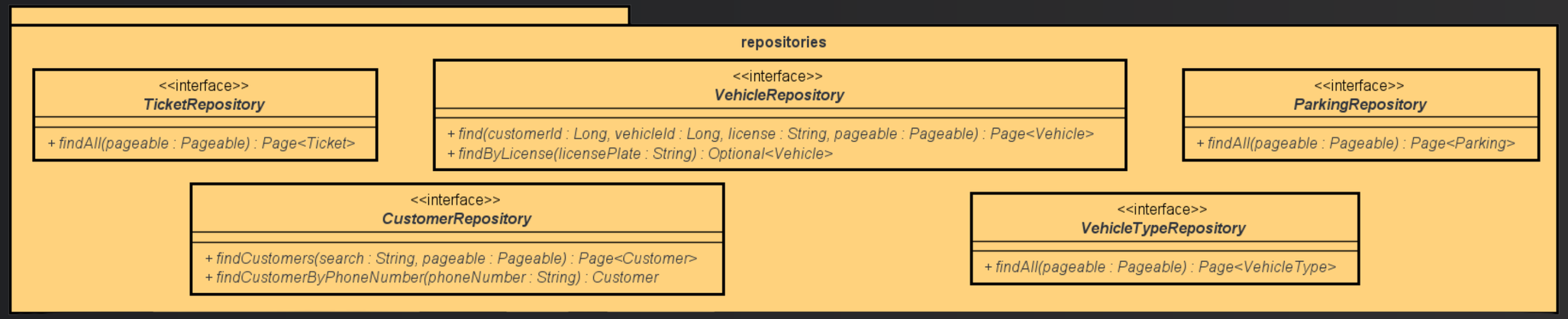
Controllers & DTOs

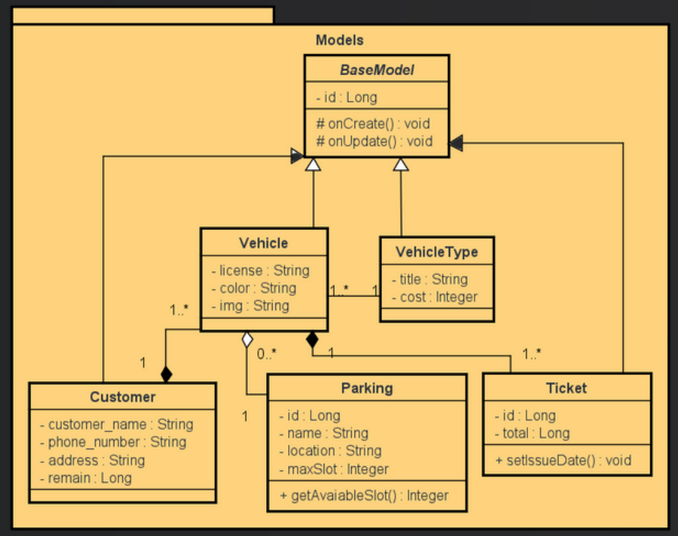


Services


services
<<interface>>
CustomerService
+ getCustomers(search : String, pageRequest : PageRequest) : Page<CustomerDTO>
+ getCustomer(id : long) : CustomerDTO
+ getCustomer(phone_num : String) : CustomerDTO
+ createCustomer(customerDTO : CustomerDTO) : CustomerDTO
+ updateCustomer(customerDTO : CustomerDTO, id : long) : CustomerDTO
+ deleteCustomer(id : long) : CustomerDTO
<<interface>>
VehicleTypeService
+ getVehicleTypes(pageRequest : PageRequest) : Page<VehicleTypeDTO>
+ getVehicleTypeById(id : long) : VehicleTypeDTO
+ createVehicleType(vehicleTypeDTO : VehicleTypeDTO) : VehicleTypeDTO
+ updateVehicleType(vehicleTypeDTO : VehicleTypeDTO, id : long) : VehicleTypeDTO
+ deleteVehicleType(id : long) : VehicleTypeDTO
<<interface>>
TicketService
+ getTickets(pageRequest : PageRequest) : Page<TicketDTO>
+ getTicketById(id : long) : TicketDTO
+ createTicket(ticketDTO : TicketDTO) : TicketDTO
+ updateTicket(ticketDTO : TicketDTO, id : long) : TicketDTO
+ deleteTicket(id : long) : TicketDTO
+ returnTicket(id : long) : Object
+ checkAndCreateTicket(vehicleDTO : VehicleDTO) : TicketDTO
<<interface>>
VehicleService
+ getVehicles(customerId : Long, typeId : Long, license : String, pageRequest : PageRequest) : Page<VehicleDTO>
+ getVehicleById(id : long) : VehicleDTO
+ createVehicle(vehicleDTO : VehicleDTO) : VehicleDTO
+ updateVehicle(vehicleDTO : VehicleDTO, id : long) : VehicleDTO
+ deleteVehicle(id : long) : VehicleDTO
<<interface>>
ParkingService
+ addVehicleToParking(parkingId : long, vehicleId : long) : Integer
+ removeVehicleFromParking(parkingId : long, vehicleId : long) : Integer
+ getVehicleCountInParking(parkingId : long) : Integer
+ getParkings(pageRequest : PageRequest) : Page<ParkingDTO>
+ updateParking(parkingDTO : ParkingDTO, id : long) : ParkingDTO
+ deleteParking(id : long) : ParkingDTO
+ getParkingById(id : long) : ParkingDTO
+ createParking(parkingDTO : ParkingDTO) : ParkingDTO
implement
TicketServiceImpl
CustomerServiceImpl
VehicleServiceImpl
VehicleTypeServiceImpl
ParkingServiceImpl

Repositories & Models





Các tính chất của OOP

Tính đóng gói

Tính kế thừa

Tính đa hình

Tính trừu tượng

Tính đóng gói

```java
@Entity  11 usages  Dng +2 *
@Table(name = "vehicle_types")
public class VehicleType extends BaseModel {
    @Column(name = "title")  2 usages
    private String title;
    @Column(name = "cost")  2 usages
    private Integer cost;
    @OneToMany(mappedBy = "type", cascade = CascadeType.ALL)
    private List<Vehicle> vehicles;

    public List<Vehicle> getVehicles() {  no usages  new *
        return vehicles;
    }

    public void setVehicles(List<Vehicle> vehicles) {  no usag
        this.vehicles = vehicles;
    }

    public String getTitle() {  2 usages  new *
        return title;
    }

    public void setTitle(String title) {  2 usages  new *
        this.title = title;
    }
```

Tính kế thừa

```java
@Getter  4 usages  4 inheritors  Dng
@Setter
@MappedSuperclass
public abstract class BaseModel {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "created_at")
    private LocalDateTime createdAt;

    @Column(name = "updated_at")
    private LocalDateTime updatedAt;

    @PrePersist  Dng
    protected void onCreate() {
        this.createdAt = LocalDateTime.now();
        this.updatedAt = LocalDateTime.now();
    }

    @PreUpdate  Dng
    protected void onUpdate() {
        this.updatedAt = LocalDateTime.now();
    }
}
```

```java
@Entity  17 usages  Dng +1
@Getter
@Setter
@Table(name = "customers")
public class Customer extends BaseModel {
    @Column(name="customer_name")
    private String customer_name;

    @Column(name="phone_number", unique = true)
    private String phone_number;

    @Column(name = "address")
    private String address;

    @OneToMany(mappedBy = "customer", cascade = CascadeType.ALL)
    private List<Vehicle> vehicles;

    @Column(name = "remain")
    private Long remain;
}
```

Tính đa hình

```
@Service  4 usages  1 implementation  Dng
public interface CustomerService {
    Page<CustomerDTO> getCustomers(String search, PageRequest pageRequest);

    CustomerDTO getCustomer(long id);  1 usage  1 implementation  Dng

    CustomerDTO getCustomer(String phone_num);  1 usage  1 implementation  Dng

    CustomerDTO createCustomer(CustomerDTO customerDTO);  1 usage  1 implementati

    CustomerDTO updateCustomer(CustomerDTO customerDTO, long id);  1 usage  1 i

    CustomerDTO deleteCustomer(long id);  1 usage  1 implementation  Dng
}
```

Tính trừu tượng

```java
@Service  4 usages  1 implementation  Dng
public interface CustomerService {
    Page<CustomerDTO> getCustomers(String search, PageRequest pageRequest);

    CustomerDTO getCustomer(long id);  1 usage  1 implementation  Dng

    CustomerDTO getCustomer(String phone_num);  1 usage  1 implementation  Dng

    CustomerDTO createCustomer(CustomerDTO customerDTO);  1 usage  1 implementati

    CustomerDTO updateCustomer(CustomerDTO customerDTO, long id);  1 usage  1 i

    CustomerDTO deleteCustomer(long id);  1 usage  1 implementation  Dng
}
```

```java
@RequiredArgsConstructor
public class CustomerServiceImpl implements CustomerService {
    private final CustomerRepository customerRepository;

    @Override  1 usage  Dng
    public Page<CustomerDTO> getCustomers(String search, PageRequest pageRequest){
        return customerRepository.findCustomers(search, pageRequest).map(CustomerDTO::getCustomerFromModel);
    }

    @Override  1 usage  Dng
    public CustomerDTO getCustomer(long id) {
        Customer customer = customerRepository.findById(id).orElseThrow();
        return CustomerDTO.getCustomerFromModel(customer);
    }

    @Override  1 usage  Dng
    public CustomerDTO getCustomer(String phone_num){
        Customer customer = customerRepository.findCustomerByPhoneNumber(phone_num);
        return CustomerDTO.getCustomerFromModel(customer);
    }

    @Override  1 usage  Dng
    public CustomerDTO createCustomer(CustomerDTO customerDTO) {
        Customer customer = new Customer();

        customer.setCustomer_name(customerDTO.getCustomer_name());
        customer.setPhone_number(customerDTO.getPhone_number());
        customer.setAddress(customerDTO.getAddress());
        customer.setRemain(customerDTO.getRemain());
        customer.setCreatedAt(customerDTO.getCreatedAt());
        customer.setUpdatedAt(customerDTO.getUpdatedAt());

        return CustomerDTO.getCustomerFromModel(customerRepository.save(customer));
    }
```

THANKS FOR LISTENING