TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

Tài liệu thiết kế phần mềm

(Software Design Document – SDD)

PHẦN MỀM THUÊ XE ĐẠP THEO GIỜ

Môn: Thiết kế và xây dựng phần mềm

Nhóm 6

Phạm Trung Kiên 20170088

Phạm Minh Khiêm 20170084

Lê Vũ Lợi 20173240

*Hà Nội, ngày 8 tháng 10 năm 2020*

Table of Contents

[1 Giới thiệu 4](#__RefHeading___Toc6061_815482102)

[1.1 Mục tiêu 4](#__RefHeading___Toc6063_815482102)

[1.2 Phạm vi 4](#__RefHeading___Toc6065_815482102)

[1.3 Từ điển thuật ngữ 4](#__RefHeading___Toc6067_815482102)

[1.4 Tài liệu tham khảo 4](#__RefHeading___Toc6069_815482102)

[2 Kiến trúc hệ thống và thiết kế kiến trúc 5](#__RefHeading___Toc6071_815482102)

[2.1 Mẫu thiết kế kiến trúc 5](#__RefHeading___Toc6073_815482102)

[2.2 Biểu đồ trình tự 5](#__RefHeading___Toc6075_815482102)

[2.3 Biểu đồ lớp phân tích 7](#__RefHeading___Toc6077_815482102)

[2.4 Biểu đồ lớp phân tích gộp 8](#__RefHeading___Toc6079_815482102)

[3 Thiết kế chi tiết 9](#__RefHeading___Toc6081_815482102)

[3.1 Thiết kế giao diện 9](#__RefHeading___Toc6083_815482102)

[3.1.1 Thiết kế giao diện người dùng 9](#__RefHeading___Toc6085_815482102)

[3.1.2 Thiết kế giao diện hệ thống 20](#__RefHeading___Toc6089_815482102)

[3.2 Mô hình hóa dữ liệu 24](#__RefHeading___Toc6091_815482102)

[3.2.1 Mô hình hóa dữ liệu mức khái niệm 24](#__RefHeading___Toc6093_815482102)

[3.2.2 Thiết kế cơ sở dữ liệu 24](#__RefHeading___Toc6095_815482102)

[3.3 Thiết kế chi tiết lớp 30](#__RefHeading___Toc6097_815482102)

[3.3.1 Biểu đồ lớp tổng quan 30](#__RefHeading___Toc6099_815482102)

[3.3.2 Biểu đồ lớp cho Interbank subsystem 31](#__RefHeading___Toc6101_815482102)

[3.3.3 Thiết kế chi tiết lớp 31](#__RefHeading___Toc6103_815482102)

[4 Các vấn đề thiết kế 50](#__RefHeading___Toc5893_1350119617)

[4.1 Coupling and cohesion 50](#__RefHeading___Toc5895_1350119617)

[4.1.1 High cohesion 50](#__RefHeading___Toc5897_1350119617)

[4.1.2 Loose coupling 50](#__RefHeading___Toc5899_1350119617)

[4.2 Các nguyên tắc thiết kế 51](#__RefHeading___Toc6113_815482102)

[4.2.1 Single Responsibility Principle 51](#__RefHeading___Toc6115_815482102)

[4.2.2 Open/Closed Principle 51](#__RefHeading___Toc6117_815482102)

[4.2.3 Liskov substitution principle 52](#__RefHeading___Toc5913_3824570499)

[4.2.4 Interface segregation principle 52](#__RefHeading___Toc5915_3824570499)

[4.2.5 Dependency Inversion principle 53](#__RefHeading___Toc5917_3824570499)

[4.3 Design Pattern 53](#__RefHeading___Toc5901_1350119617)

# Giới thiệu

## Mục tiêu

* Mục tiêu của tài liệu này là đưa ra thiết kế phần mềm cho ứng dụng EcoBikeRental cho thuê xe đạp theo giờ với nhiều bãi để xe để thuê/trả xe tự động trong khu đô thị Ecopark.
* Tài liệu này phục vụ cho designers, developers và testers của hệ thống có thể dễ dàng đi tới các bước tiếp theo của quy trình phát triển ứng dụng này.

## Phạm vi

Thuê xe đạp dạo quanh là một trong những dịch vụ được thu hút nhất tại khu đô thị sinh thái Ecopark. Hiện nay có 2 điểm cho thuê và đỗ xe đạp tại khu đô thị. Để dịch vụ này tiếp tục phát triển mở rộng ra, cần giải quyết khâu hạ tầng, bao gồm hệ thống làn đường cho xe đạp, điểm dừng, trông giữ, bảo quản, và đặc biệt là hệ thống thông tin thuê xe và trả xe tự động có thể hoạt động 24/7

## Từ điển thuật ngữ

|  |  |  |
| --- | --- | --- |
| STT | Thuật ngữ, từ viêt tắt | Giải thích |
| 1 | API | viết tắt của Application Programming Interface, là phần mềm trung gian cho phép kết nối 2 ứng dụng với nhau |
|  |  |  |
|  |  |  |

## Tài liệu tham khảo

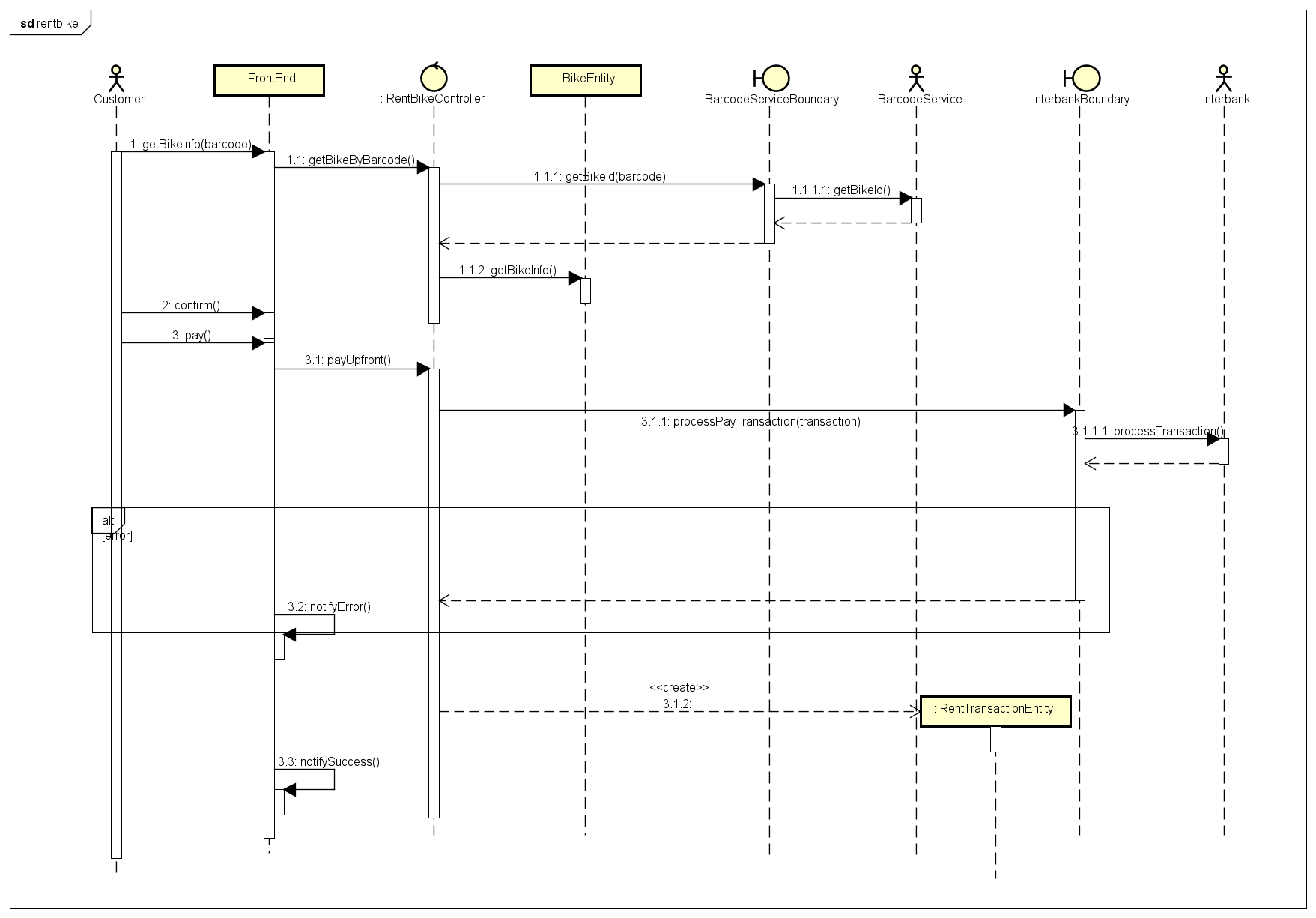
|  |  |
| --- | --- |
| STT | Tên tài liệu |
| 1 | D. Budgen. Software Design, 2nd Edition. Addison-Wesley. 2004 |
|  |  |
|  |  |

# Kiến trúc hệ thống và thiết kế kiến trúc

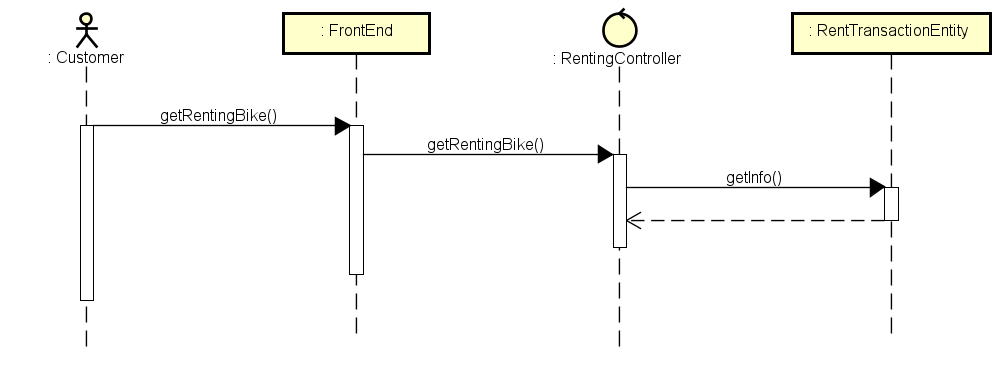
## Mẫu thiết kế kiến trúc

*Nhóm chọn thiết kế theo kiến trúc Frontend – Backend*

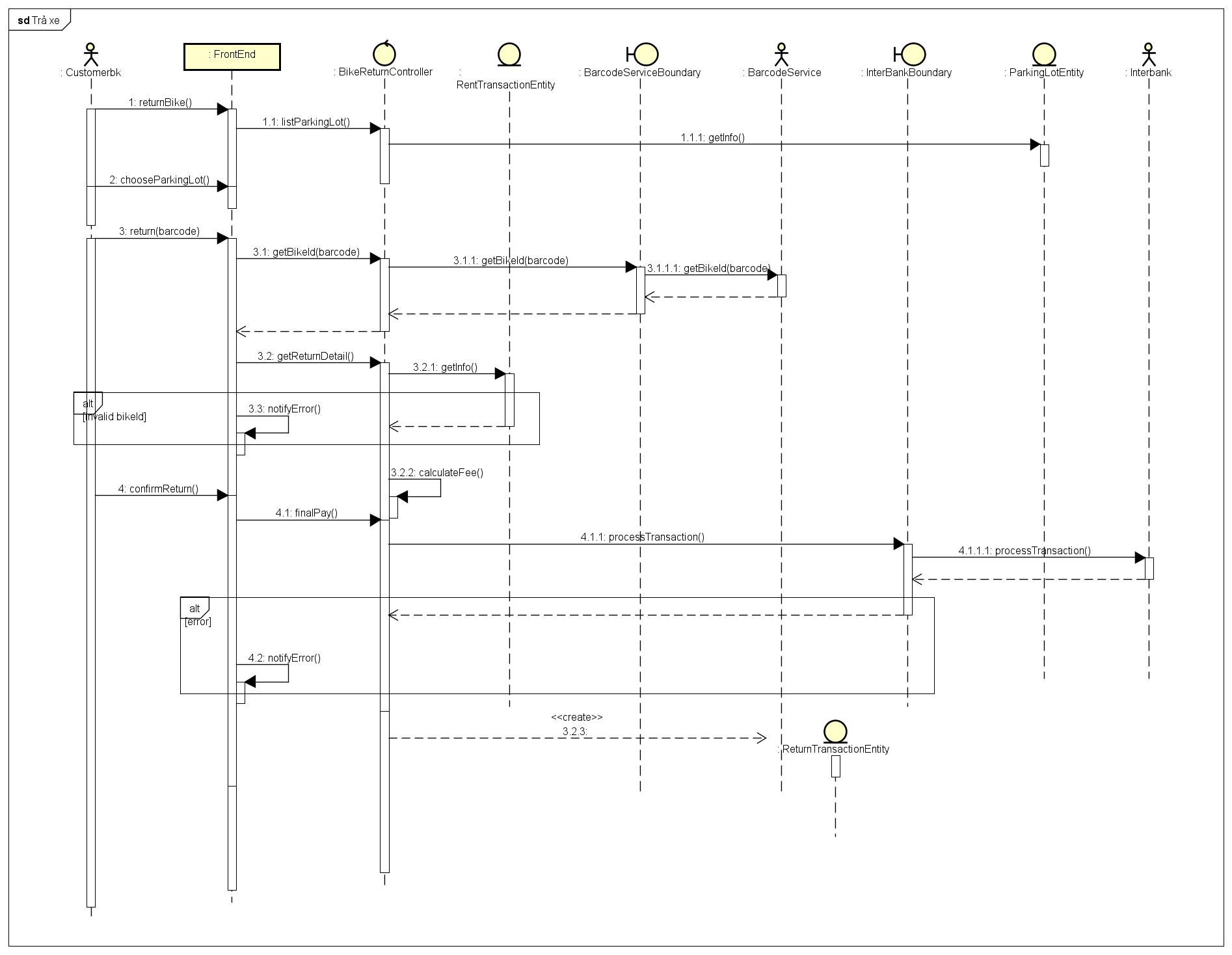
## Biểu đồ trình tự



Hình : Biểu đồ trình tự cho thuê xe

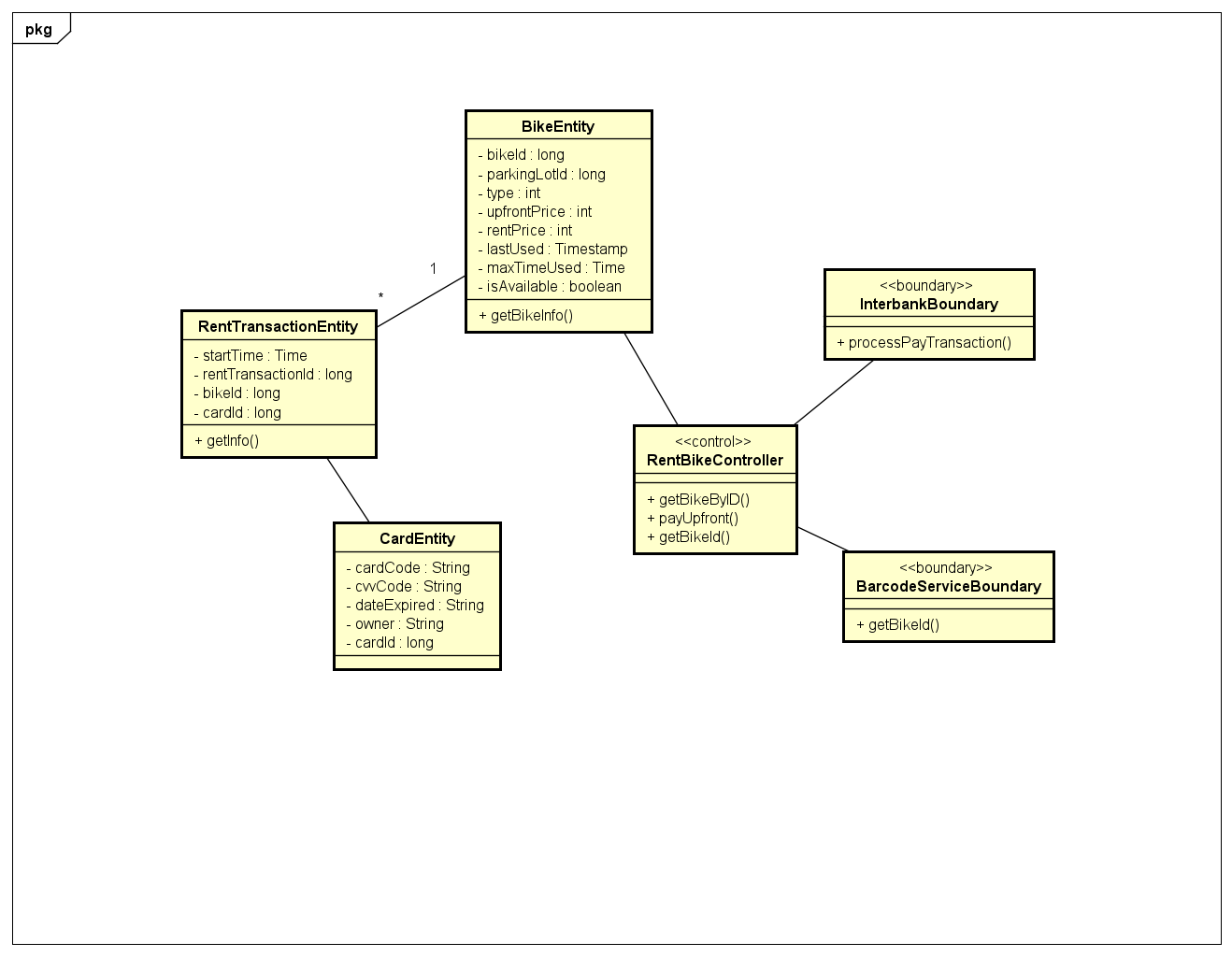


Hình : Biểu đồ trình tự cho xem xe đang thuê

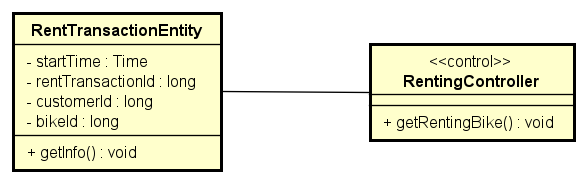


Hình : Biểu đồ trình tự cho trả xe

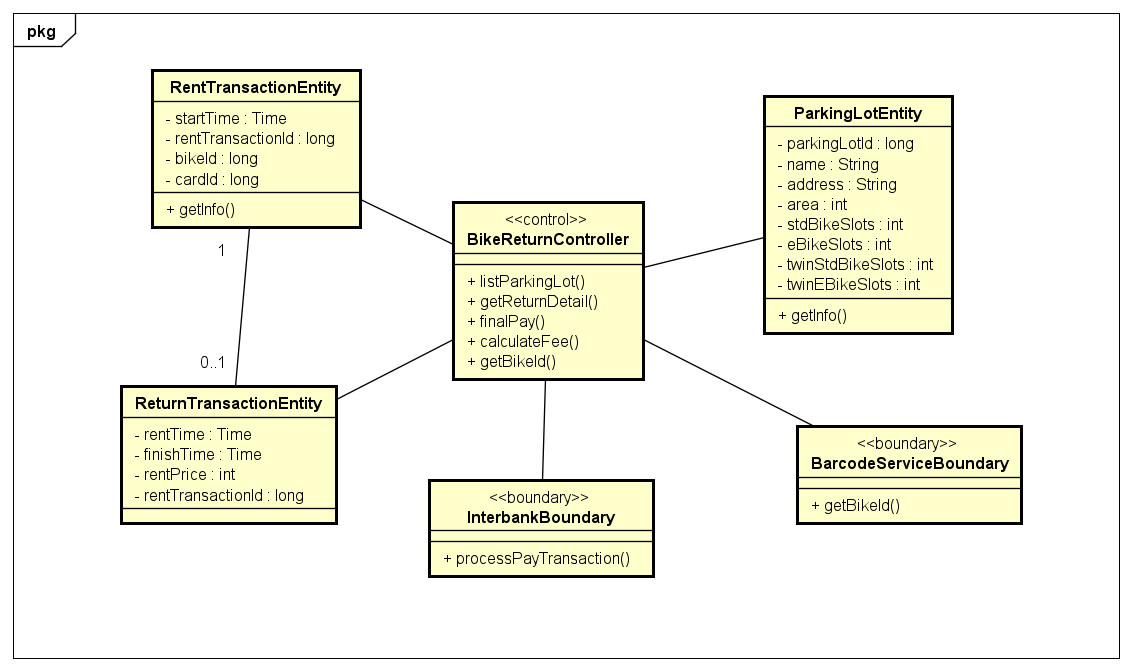
## Biểu đồ lớp phân tích



Hình : Biểu đồ cho thuê xe

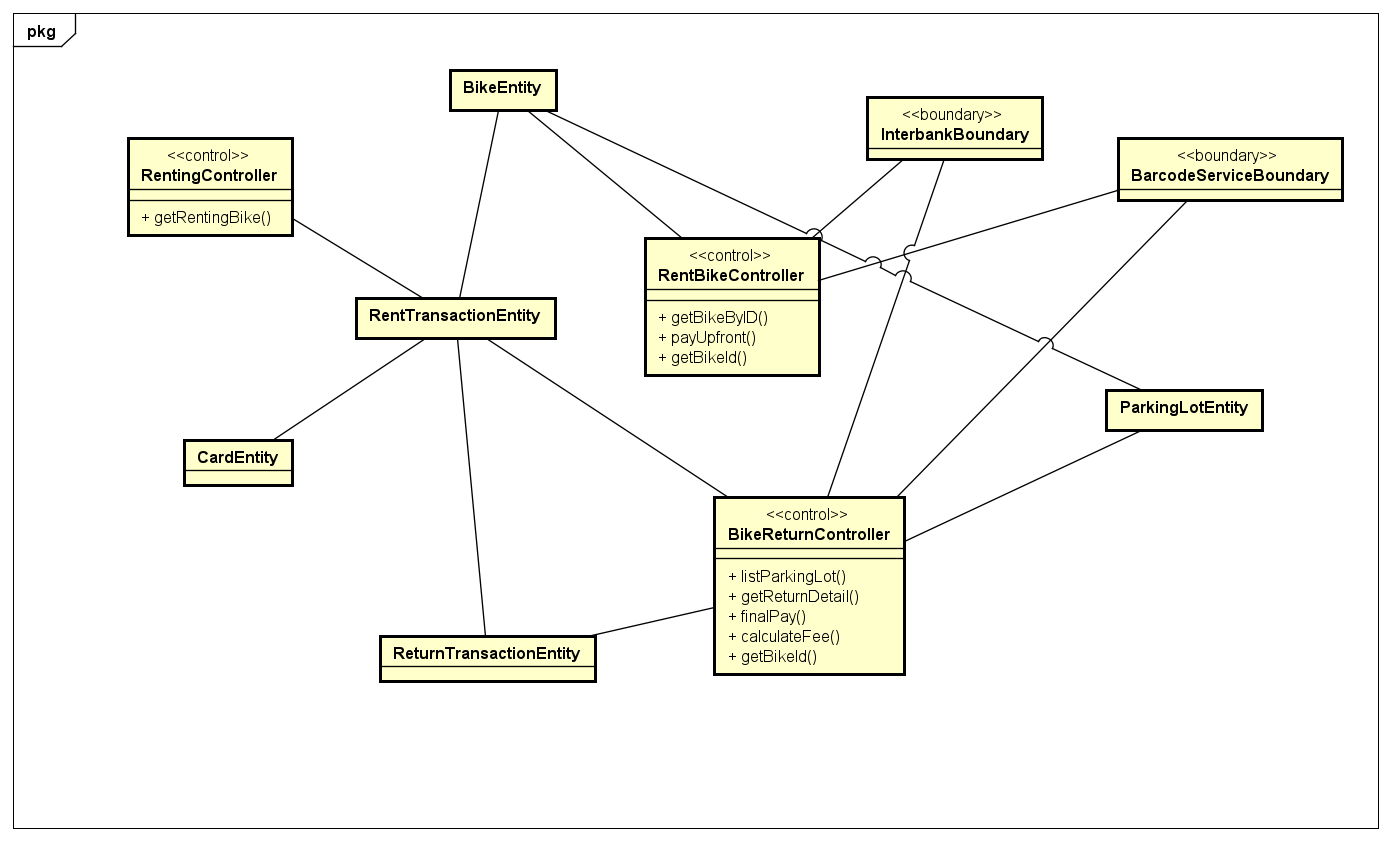


Hình : Biểu đồ cho xem xe đang thuê



Hình : Biểu đồ cho trả xe

## Biểu đồ lớp phân tích gộp



# Thiết kế chi tiết

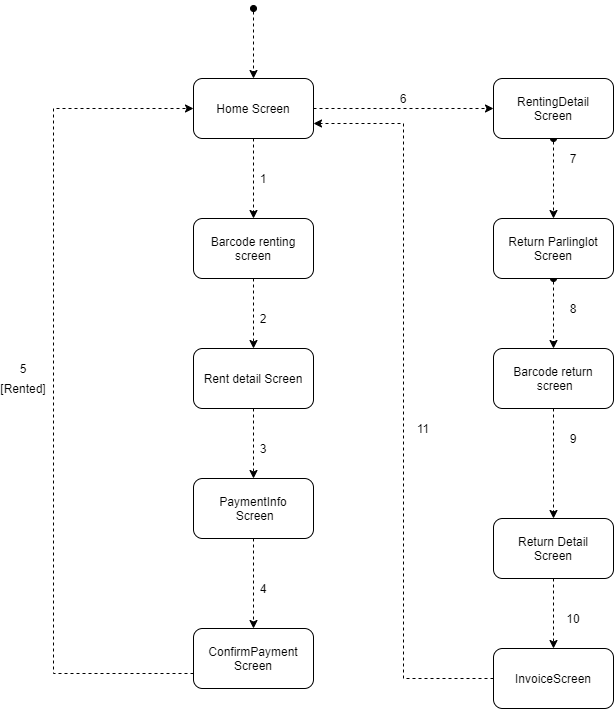
## Thiết kế giao diện

### Thiết kế giao diện người dùng

#### Danh sách các màn hình

|  |  |  |  |
| --- | --- | --- | --- |
| Home screen | Barcode renting screen | Rent detail screen | Payment info screen |
|  |  |  |  |
| Confirm payment screen | Home rented screen | Renting detail screen | Return parkinglot screen |
|  |  |  |  |
| Barcode return screen | Return detailed screen | Invoice screen | Bike not found alert |
|  |  |  |  |
| Success renting alert | Rent payment failed alert | Barcode return error | Payment error alert |
|  |  |  |  |

#### Sơ đồ dịch chuyển màn hình



#### Đặc tả màn hình

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Home screen* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Khu vực hiển thị các bãi đỗ xe* | *Initial* | *Hiển thị tên và địa chỉ của các bãi đỗ xe gần nhất* | |
| *Nút Xem* | *Click* | *Xem thông tin chi tiết bãi đỗ xe* | |
| *Nút Thuê xe* | *Click* | *Hiển thị màn hình nhập barcode* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Barcode renting screen* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Nút Thuê xe* | *Click* | *Thuê xe có barcode tương ứng* | |
| *Khu vực nhập mã vạch* | *Type* | *Nhập mã vạch* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Rent detail screen* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Khu vực hiển thị thông tin xe* | *Initial* | *Hiển thị thông tin chi tiết của xe tương ứng* | |
| *Khu vực hiển thị giá cọc* | *Initial* | *Hiển thị giá cọc* | |
| *Nút Tiếp tục* | *Click* | *Hiển thị màn hình thanh toán* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Payment info screen* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Khu vực nhập thông tin thanh toán* | *Initial* | *Nhập thông tin thanh toán* | |
| *Nút Thanh toán* | *Click* | *Thực hiện đặt cọc* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Confirm payment screen* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Khu vực hiển thị thông tin thanh toán* | *Initial* | *Hiển thị tên và địa chỉ của các bãi đỗ xe gần nhất* | |
| *Nút Thanh toán* | *Click* | *Xác nhận giao dịch* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Home rented screen* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Nút Bạn đang có 1 xe đang thuê* | *Click* | *Xem thông tin xe đang thuê* | |
| *Nút Xem* | *Click* | *Xem thông tin chi tiết bãi đỗ xe* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Renting detail screen* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Khu vực hiển thị thông tin xe đang thuê* | *Initial* | *Hiển thị thông tin chi tiết của xe đang thuê* | |
| *Nút Tạm dừng* | *Click* | *Tạm dừng thuê xe* | |
| *Nút Trả xe* | *Click* | *Hiển thị màn hình trả xe* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Return parkinglot screen* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Khu vực hiển thị các bãi đỗ xe* | *Initial* | *Hiển thị tên và địa chỉ của các bãi đỗ xe gần nhất* | |
| *Nút Chọn* | *Click* | *Chọn bãi đỗ xe để trả* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Barcode return screen* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Khu vực nhập mã vạch* | *Initial* | *Nhập mã vạch xe muốn trả* | |
| *Nút Trả xe* | *Click* | *Hiển thị màn hình trả xe* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Home screen* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Khu vực hiển thị thông tin trả xe* | *Initial* | *Hiển thị thông tin chi tiết trả xe* | |
| *Nút Thanh toán* | *Click* | *Xác nhận trả xe* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Invoice screen* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Khu vực hiển thị thông tin giao dịch trả xe* | *Initial* | *Hiển thị thông tin giao dịch trả xe* | |
| *Nút Quay về trang chủ* | *Click* | *Về trang chủ* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Bike not found alert* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Thông báo không thấy xe* | *Initial* | *Thông báo không thấy xe phù hợp mã vạch* | |
| *Nút OK* | *Click* | *Đóng thông báo* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Success renting alert* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Thông báo thuê xe thành công* | *Initial* | *Thông báo đã thuê xe thành công* | |
| *Nút OK* | *Click* | *Đóng thông báo* | |
|  |  |  | |

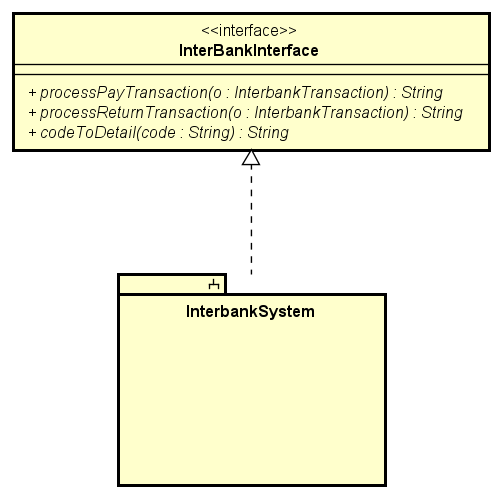
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Rent payment failed alert* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Thông báo thanh toán lỗi* | *Initial* | *Thông báo thanh toán gặp lỗi khi thuê xe* | |
| *Nút OK* | *Click* | *Đóng thông báo* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Barcode return error* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Thông báo mã vạch không khớp* | *Initial* | *Thông báo mã vạch nhập vào không khớp với xe nào* | |
| *Nút OK* | *Click* | *Đóng thông báo* | |
|  |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *EcoBike* | | *Date of creation* | *Approved by* | *Reviewed by* | *Person in charge* |
| *Screen specification* | *Payment error alert* |  |  |  | *Phạm Minh Khiêm* |
|  | | *Control* | *Operation* | *Function* | |
| *Thông báo thanh toán lỗi* | *Initial* | *Thông báo thanh toán gặp lỗi khi trả xe* | |
| *Nút OK* | *Click* | *Đóng thông báo* | |
|  |  |  | |

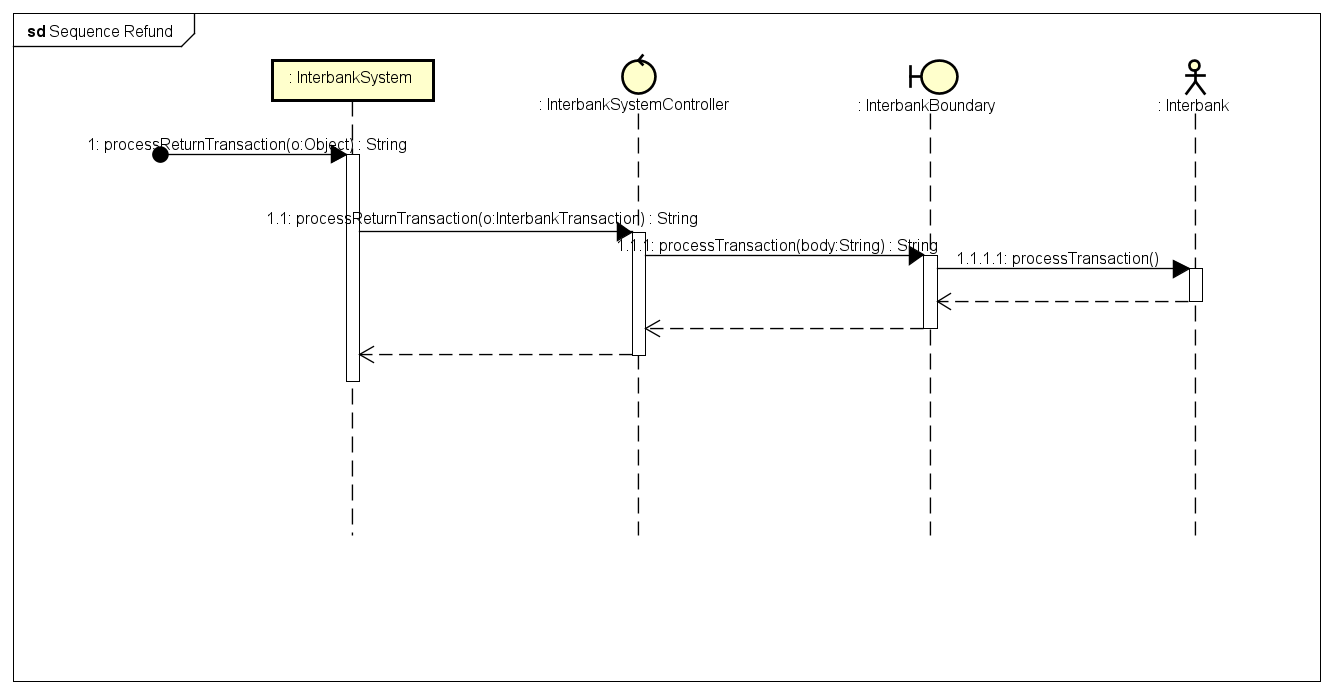
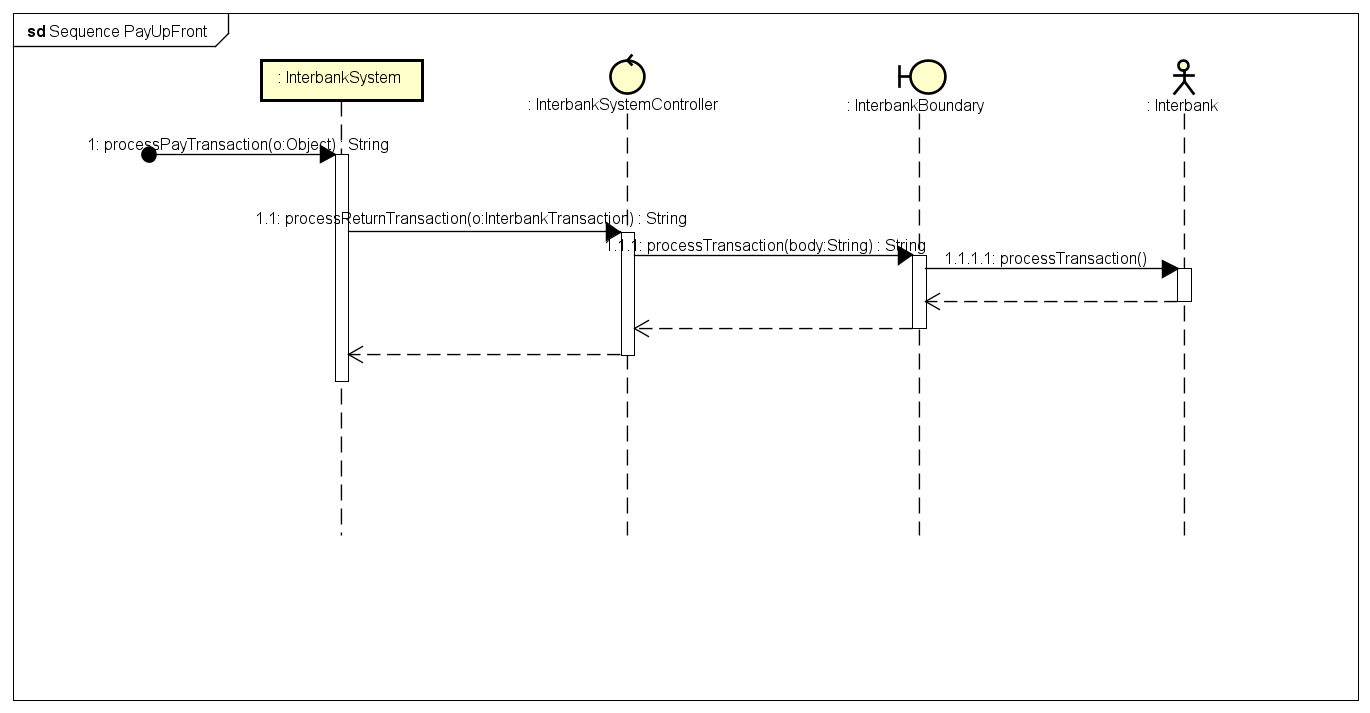
### Thiết kế giao diện hệ thống

#### Thiết kế interface cho Interbank subsystem:

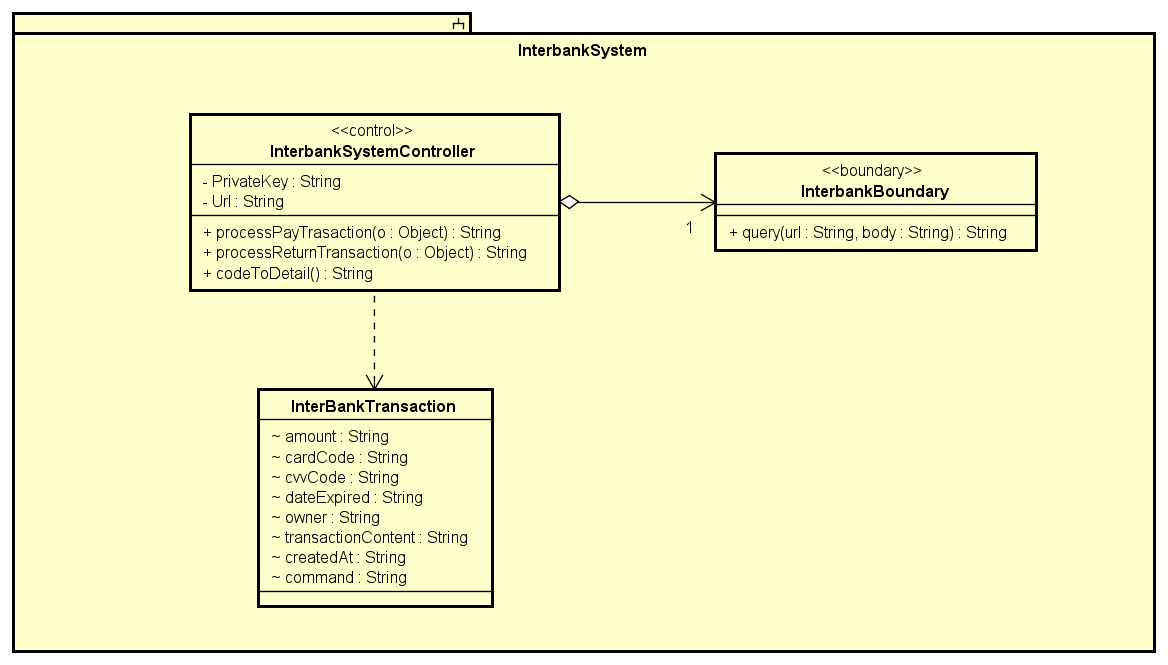


#### Thiết kế Interbank subsystem:

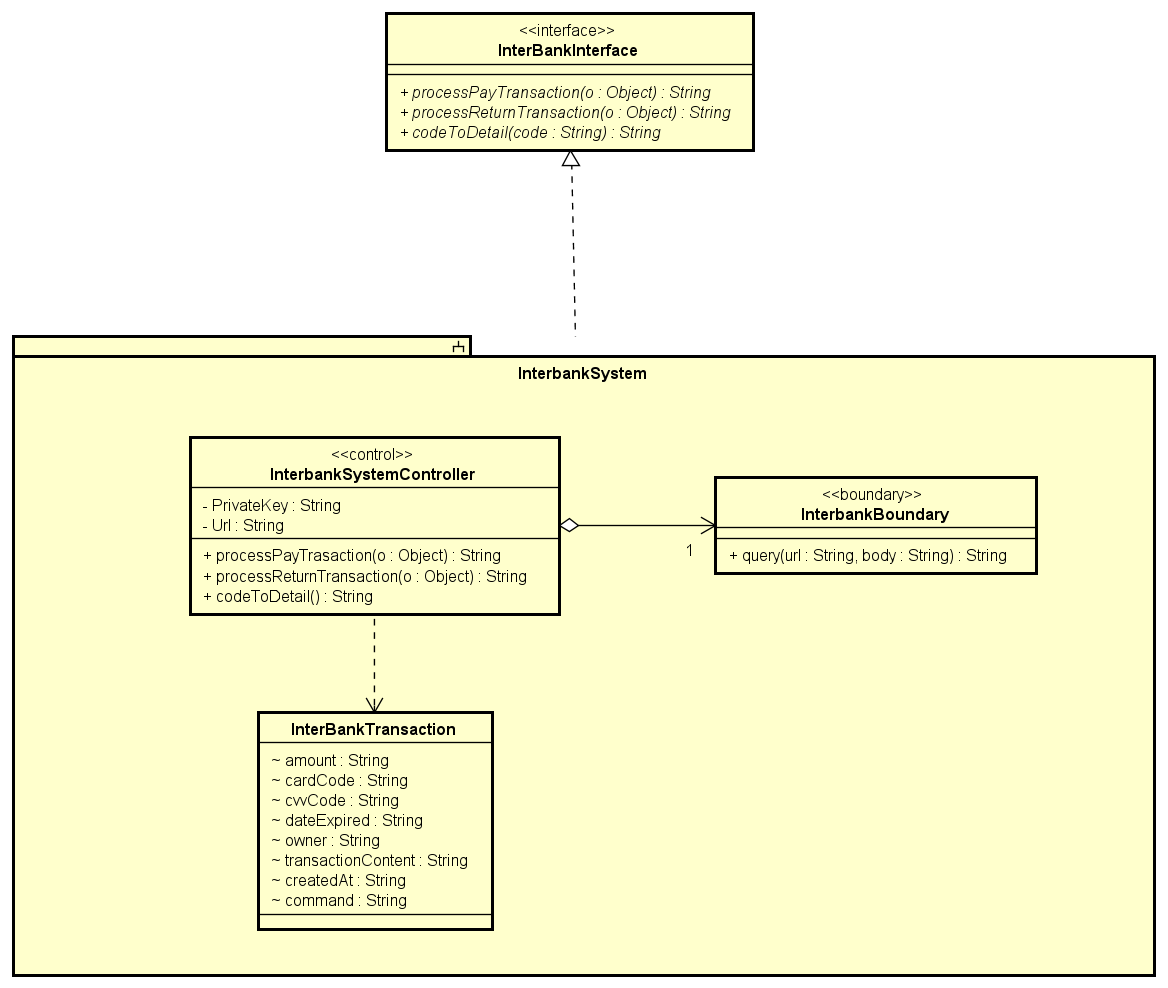
##### Phân bổ hành vi của subsystem cho các thành phần bên trong subsystem



##### Mô tả các thành phần của subsystem

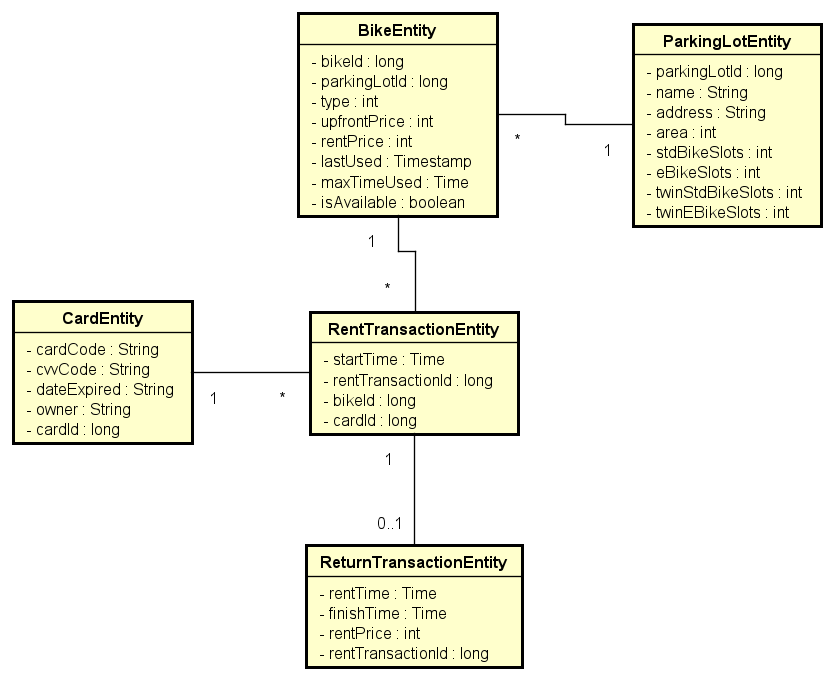


##### Tổng hợp thiết kế của subsystem



## Mô hình hóa dữ liệu

### Mô hình hóa dữ liệu mức khái niệm

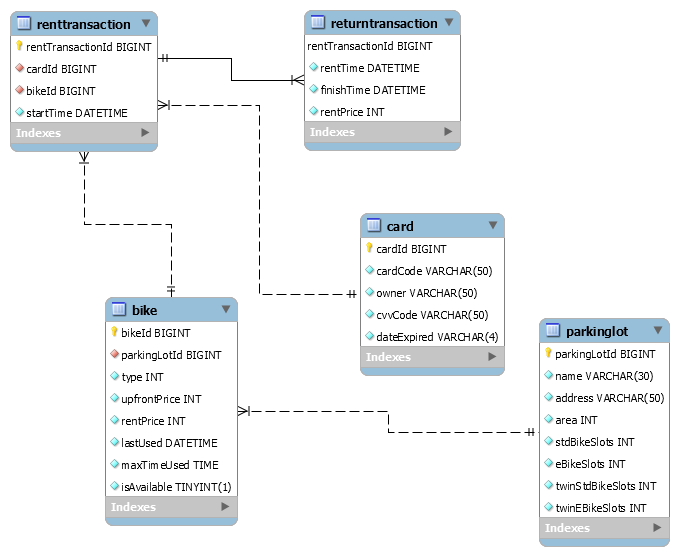


### Thiết kế cơ sở dữ liệu

#### Hệ quản trị cơ sở dữ liệu

*Hệ quản trị dữ liệu mySQL*

#### Mô hình hóa dữ liệu mức logic



#### Mô hình hóa dữ liệu mức vật lý

##### Schemas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| parkingLotId | bigint | NO | PRI | NULL | auto\_increment |
| name | varchar(30) | NO |  | NULL |  |
| address | varchar(50) | NO |  | NULL |  |
| area | int | NO |  | NULL |  |
| stdBikeSlots | int | NO |  | NULL |  |
| eBikeSlots | int | NO |  | NULL |  |
| twinStdBikeSlots | int | NO |  | NULL |  |
| twinEBikeSlots | int | NO |  | NULL |  |

Bang:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| cardId | bigint | NO | PRI | NULL | auto\_increment |
| cardCode | varchar(50) | NO |  | NULL |  |
| owner | varchar(50) | NO |  | NULL |  |
| cvvCode | varchar(50) | NO |  | NULL |  |
| dateExpired | varchar(4) | NO |  | NULL |  |

Bang:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| bikeId | bigint | NO | PRI | NULL | auto\_increment |
| parkingLotId | bigint | NO | MUL | NULL |  |
| type | int | NO |  | NULL |  |
| upfrontPrice | int | NO |  | NULL |  |
| rentPrice | int | NO |  | NULL |  |
| lastUsed | datetime | NO |  | NULL |  |
| maxTimeUsed | time | NO |  | NULL |  |
| isAvailable | tinyint(1) | NO |  | NULL |  |

Bang:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| rentTransactionId | bigint | NO | PRI | NULL | auto\_increment |
| cardId | bigint | NO | MUL | NULL |  |
| bikeId | bigint | NO | MUL | NULL |  |
| startTime | datetime | NO |  | NULL |  |

Bang:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| rentTransactionId | bigint | NO | PRI | NULL |  |
| rentTime | datetime | NO |  | NULL |  |
| finishTime | datetime | NO |  | NULL |  |
| rentPrice | int | NO |  | NULL |  |

##### SQL Scripts

**Create schemas**

create table parkinglot (

parkingLotId bigint primary key auto\_increment,

name varchar(30) not null,

address varchar(50) not null,

area int not null,

stdBikeSlots int not null,

eBikeSlots int not null,

twinStdBikeSlots int not null,

twinEBikeSlots int not null

);

create table card (

cardId bigint primary key auto\_increment,

cardCode varchar(50) not null,

owner varchar(50) not null,

cvvCode varchar(50) not null,

dateExpired varchar(4) not null

);

create table bike (

bikeId bigint primary key auto\_increment,

parkingLotId bigint not null,

type int not null,

upfrontPrice int not null,

rentPrice int not null,

lastUsed datetime not null,

maxTimeUsed time not null,

isAvailable bool not null,

foreign key (parkingLotId) references parkinglot(parkingLotId)

);

create table renttransaction (

rentTransactionId bigint primary key auto\_increment,

cardId bigint not null,

bikeId bigint not null,

startTime datetime not null,

foreign key (cardId) references card(cardId),

foreign key (bikeId) references bike(bikeId)

);

create table returntransaction (

rentTransactionId bigint primary key,

rentTime datetime not null,

finishTime datetime not null,

rentPrice int not null,

foreign key (rentTransactionId) references renttransaction(rentTransactionId)

);

**Insert data**

insert into parkinglot(name, address, area, stdBikeSlots, eBikeSlots, twinStdBikeSlots, twinEBikeSlots)

values ('Vườn hoa Hướng Dương', 'Ha Noi', 1000, 30, 30, 30, 30);

insert into parkinglot(name, address, area, stdBikeSlots, eBikeSlots, twinStdBikeSlots, twinEBikeSlots)

values ('Hồ con cá', 'Ha Noi', 1200, 40, 30, 30, 30);

insert into parkinglot(name, address, area, stdBikeSlots, eBikeSlots, twinStdBikeSlots, twinEBikeSlots)

values ('Công viên Thống Nhất', 'Ha Noi', 1200, 30, 30, 40, 30);

insert into parkinglot(name, address, area, stdBikeSlots, eBikeSlots, twinStdBikeSlots, twinEBikeSlots)

values ('Viện bảo tàng', 'Ha Noi', 1200, 30, 40, 30, 30);

insert into parkinglot(name, address, area, stdBikeSlots, eBikeSlots, twinStdBikeSlots, twinEBikeSlots)

values ('Parking Lot 5', 'Ha Noi', 1200, 30, 30, 30, 40);

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

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

values (1, 1, 100, 20, '2018-01-01', '6:00', true);

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

values (2, 1, 100, 20, '2018-01-01', '6:00', true);

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

values (2, 2, 200, 25, '2018-01-01', '6:00', true);

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

values (2, 2, 200, 25, '2018-01-01', '6:00', true);

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

values (1, 3, 300, 30, '2018-01-01', '6:00', true);

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

values (3, 3, 300, 30, '2018-01-01', '6:00', true);

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

values (3, 4, 400, 35, '2018-01-01', '6:00',true);

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

values (2, 1, 100, 20, '2018-01-01', '6:00',true);

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

values (1, 4, 400, 35, '2018-01-01', '6:00',true);

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

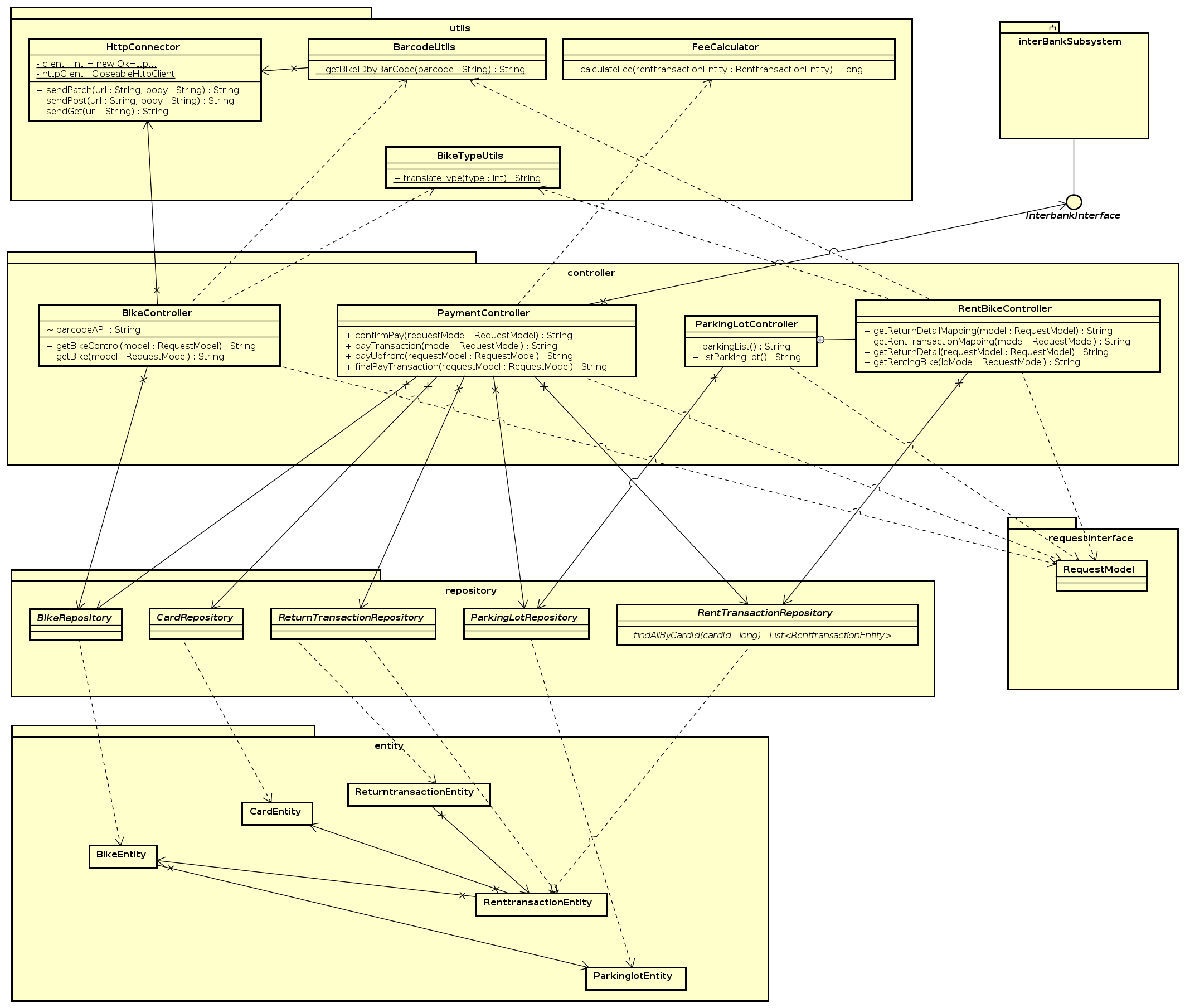
values (3, 3, 300, 30, '2018-01-01', '6:00',true);

insert into bike(parkingLotId, type, upfrontPrice, rentPrice, lastUsed, maxTimeUsed,isAvailable)

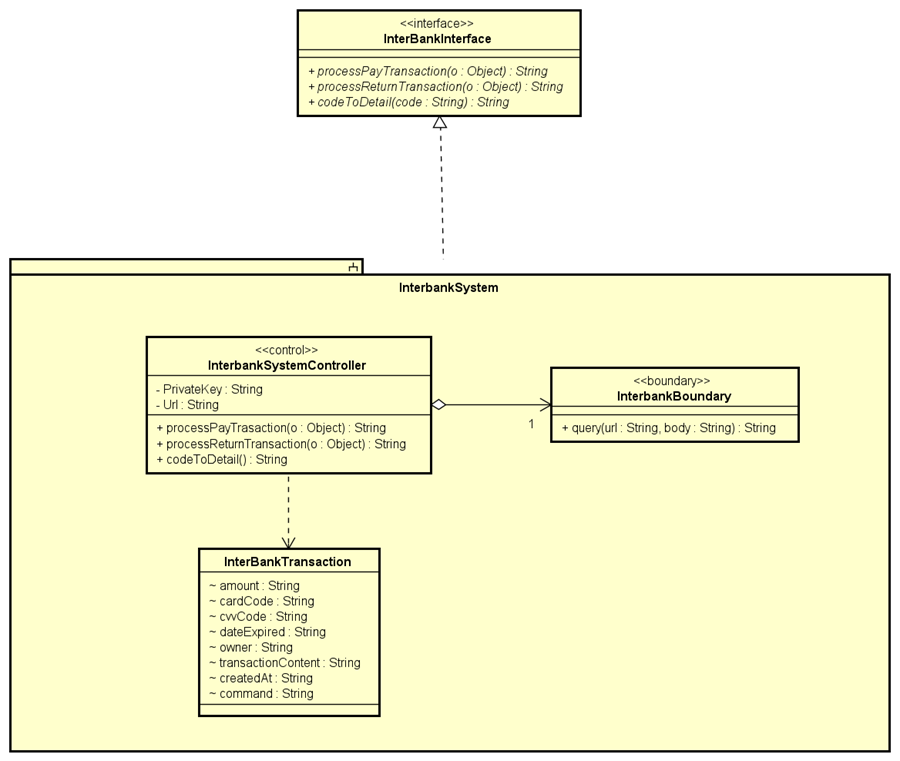
values (3, 3, 150000, 30000, '2018-01-01', '6:00',true);

## Thiết kế chi tiết lớp

### Biểu đồ lớp tổng quan



### Biểu đồ lớp cho Interbank subsystem



### Thiết kế chi tiết lớp

#### Gói controller

##### Lớp BikeController

**Mục đích sử dụng**

Điều khiển các tác vụ liên quan đến đối tượng xe.

Định nghĩa REST API tương ứng để front-end có thể tương tác đến.

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | bikeRepository | BikeRepository |  | Đối tượng sử dụng để truy vấn bảng Bike trong cơ sở dữ liệu |
| 2 | barcodeAPI | String | https://barcodeservicebykv2.herokuapp.com/barcode | Đường dẫn đến API chuyển mã vạch |
| 3 | httpConnector | HttpConnector |  | gửi request đến API barcode yêu cầu chuyển mã vạch thành mã xe |

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | getBikeControl | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Tạo API để front-end truy vấn thông tin xe:   * Method: POST * Path: /getBikeByBarcode | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| model | RequestModel | Đóng gói các thông tin cần thiết để yêu cầu thông tin về một xe muốn thuê, bao gồm:   * barcode |
|  | | | | |
| **2** | ***Tên*** | getBikeByBarcode | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Lấy thông tin xe tương ứng với barcode truyền vào | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| model | RequestModel | Tương tự getBikeControl |

**Method**

Không

**State**

Không

##### Lớp PaymentController

**Mục đích sử dụng**

Lớp điều khiển các tác vụ liên quan đến thanh toán hóa đơn thuê và trả xe.

Định nghĩa REST API tương ứng để front-end có thể tương tác đến.

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | parkinglotRepository | ParkingLotRepository |  | Đối tượng sử dụng để truy vấn bảng ParkingLot trong cơ sở dữ liệu |
| 2 | bikeRepository | BikeRepository |  | Đối tượng sử dụng để truy vấn bảng Bike trong cơ sở dữ liệu |
| 3 | rentTransactionRepository | RentTransactionRepository |  | Đối tượng sử dụng để truy vấn bảng RentTransaction trong cơ sở dữ liệu |
| 4 | returnTransactionRepository | ReturnTransactionRepository |  | Đối tượng sử dụng để truy vấn bảng ReturnTransaction trong cơ sở dữ liệu |
| 5 | cardRepository | CardRepository |  | Đối tượng sử dụng để truy vấn bảng Card trong cơ sở dữ liệu |
| 6 | interbankSubsystem | InterbankInterface |  | Đối tượng sử dụng để giao tiếp với Interbank API |

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | payUpFrontControl | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Định nghĩa API để yêu cầu front-end thanh toán cọc:   * Method: POST * Path: /payUpFront | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| model | RequestModel | Đóng gói các trường dữ liệu liên quan đến thông tin thanh toán, bao gồm:   * barcode * cardCode * cardOwner * cvv * expireDate |
|  | | | | |
| **2** | ***Tên*** | finalPayControl | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Định nghĩa API yêu cầu thanh toán trả xe | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| model | RequestModel | Đóng gói các trường dữ liệu liên quan đến việc xác nhận người thanh toán (người thuê xe phải đúng là người trả xe), bao gồm:   * barcode * cardID * parkinglotID |
|  | | | | |
| **3** | ***Tên*** | payUpFront | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Phương thức dùng để thanh toán cọc | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| model | RequestModel | Tương tự payUpFrontControl |
|  | | | | |
| **4** | ***Tên*** | finalPay | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Phương thức thanh toán khi người dùng trả xe | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| model | RequestModel | Tương tự finalPayControl |

**Method**

Không

**State**

Không

##### Lớp ParkingLotController

**Mục đích sử dụng**

Lớp điều khiển các tác vụ liên quan đến truy vấn các bãi xe.

Định nghĩa REST API tương ứng để front-end có thể tương tác đến.

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | parkinglotRepository | ParkingLotRepository |  | Đối tượng sử dụng để truy vấn bảng ParkingLot trong cơ sở dữ liệu |

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | parkingList | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Định nghĩa API để front-end truy vấn thông tin xe:   * Method: GET * Path: /listParkingLot | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
|  |  |  |
|  | | | | |
| **2** | ***Tên*** | listParkingLot | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Lấy thông tin danh sách các bãi xe trong hệ thống | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
|  |  |  |

**Method**

Không

**State**

Không

##### Lớp RentBikeController

**Mục đích sử dụng**

Lớp điều khiển các tác vụ liên quan đến thông tin thuê và trả xe.

Định nghĩa REST API tương ứng để front-end có thể tương tác đến.

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | rentTransactionRepository | RentTransactionRepository |  | Đối tượng sử dụng để truy vấn bảng RentTransaction trong cơ sở dữ liệu |

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | getReturnDetailControl | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Định nghĩa API truy vấn thông tin trả xe:   * Method: POST * Path: /returnDetail | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| model | RequestModel | Đóng gói các trường dữ liệu liên quan đến trả xe, bao gồm:   * barcode * cardID |
|  | | | | |
| **2** | ***Tên*** | getRentTransactionControl | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Định nghĩa API truy vấn thông tin xe đang thuê:   * Method: POST * Path: /getRentTransaction | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| model | RequestModel | Đóng gói các trường dữ liệu liên quan đến yêu cầu thông tin giao dịch thuê xe:   * cardID |
|  | | | | |
| **3** | ***Tên*** | getReturnDetail | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Lấy thông tin dùng để trả xe | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| model | RequestModel | Tương tự getReturnDetailControl |
|  | | | | |
| **4** | ***Tên*** | getRentTransaction | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Lấy thông tin xe đang được thuê | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| model | RequestModel | Tương tự getRentTransactionControl |

**Method**

Không

**State**

Không

#### Gói repository

Sử dụng Java Persistence API để thực hiện các thao tác CRUD đến database bằng cách tạo ra các repository interface:

• BikeRepository

• CardRepository

• ParkingLotRepository

• RentTransactionRepository

• ReturnTransactionRepository

#### Gói entity

##### Lớp BikeEntity

**Mục đích sử dụng**

Định nghĩa một đối tượng Bike, mapping với các trường thông tin tương ứng trong CSDL.

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | bikeId | long |  | Mã định danh xe |
| 2 | parkingLotId | long |  | Mã bãi xe |
| 3 | type | int |  | Loại xe |
| 4 | upfrontPrice | int |  | Giá cọc của xe |
| 5 | rentPrice | int |  | Giá thuê |
| 6 | lastUsed | Timestamp |  | Lần cuối sử dụng |
| 7 | maxTimeUsed | Time |  | Thời gian sử dụng tối đa (chỉ có với xe điện) |
| 8 | isAvailable | byte |  | Có đang được thuê hay không? |
| 8 | parkinglotByParkingLotId | ParkingLotEntity |  | Bãi xe tương ứng với xe |
| 9 | renttransactionsByBikeId | Collection<RentTransactionEntity> |  | Danh sách các giao dịch thuê xe tương ứng với xe |

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | getParkingLotByParkingLotId | ***Kiểu dữ liệu trả về*** | ParkingLot |
| ***Mô tả*** | Lấy về bãi xe tương ứng với xe | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
|  |  |  |
|  | | | | |
| **2** | ***Tên*** | getRentTransactionsByBikeId | ***Kiểu dữ liệu trả về*** | Collection<RentTransactionEntity> |
| ***Mô tả*** | Lấy về danh sách các giao dịch thuê xe tương ứng với xe | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
|  |  |  |

**Method**

Không

**State**

Không

##### Lớp CardEntity

**Mục đích sử dụng**

Định nghĩa một đối tượng Card, mapping với các trường thông tin tương ứng trong CSDL.

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | cardId | long |  | ID định danh thẻ mỗi lần thanh toán |
| 2 | cardCode | String |  | Mã định danh thẻ |
| 3 | owner | String |  | Tên chủ thẻ |
| 4 | cvvCode | String |  | Card Verification Value |
| 5 | dateExpired | String |  | Ngày hết hạn |
| 6 | renttransactionsByCardId | Collection<RentTransactionEntity> |  | Danh sách các giao dịch thuê xe sử dụng thẻ cardID |

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | getRentTransactionByCardId | ***Kiểu dữ liệu trả về*** | Collection<RentTransactionEntity> |
| ***Mô tả*** | Lấy về danh sách giao dịch thuê xe sử dụng thẻ có cardID tương ứng | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
|  |  |  |

**Method**

Không

**State**

Không

##### Lớp ReturnTransactionEntity

**Mục đích sử dụng**

Định nghĩa một đối tượng ReturnTransaction, mapping với các trường thông tin tương ứng trong CSDL.

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | rentTransactionId | long |  | Mã giao dịch thuê xe tương ứng |
| 2 | rentTime | Timestamp |  | Thời điểm bắt đầu thuê xe |
| 3 | finishTime | Timestamp |  | Thời điểm trả xe |
| 4 | rentPrice | int |  | Tổng tiền thuê xe |
| 5 | renttransactionsByRentTransactionId | RenttransactionEntity |  | Đối tượng RentTransactionEntity của giao dịch thuê xe tương ứng. |

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | getRenttransactionsByRentTransactionId | ***Kiểu dữ liệu trả về*** | RentTransactionEntity |
| ***Mô tả*** | Lấy về đối tượng RentTransactionEntity của giao dịch thuê xe tương ứng | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
|  |  |  |

**Method**

Không

**State**

Không

##### Lớp ParkinglotEntity

**Mục đích sử dụng**

Định nghĩa một đối tượng ParkingLot, mapping với các trường thông tin tương ứng trong CSDL.

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | parkingLotId | long |  | Mã bãi xe |
| 2 | name | String |  | Tên bãi xe |
| 3 | address | String |  | Địa chỉ bãi xe |
| 4 | area | int |  | Diện tích bãi xe |
| 5 | stdBikeSlots | int |  | Số lượng chỗ trống xe đạp đơn |
| 6 | eBikeSlots | int |  | Số lượng chỗ trống xe đạp điện đơn |
| 7 | twinStdBikeSlots | int |  | Số lượng chỗ trống xe đạp đôi |
| 8 | twinEBikeSlots | int |  | Số lượng chỗ trống của xe đạp điện đôi |
| 9 | bikesByParkingLotId | Collection<BikeEntity> |  | Danh sách các xe có trong bãi |

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | getBikeByParkinglotId | ***Kiểu dữ liệu trả về*** | Collection<BikeEntity> |
| ***Mô tả*** | Lấy về danh sách các xe có trong bãi | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
|  |  |  |

**Method**

Không

**State**

Không

##### Lớp RentTransactionEntity

**Mục đích sử dụng**

Định nghĩa một đối tượng RentTransaction, mapping với các trường thông tin tương ứng trong CSDL.

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | rentTransactionId | long |  | Mã giao dịch thuê xe |
| 2 | cardId | long |  | ID định danh thẻ mỗi lần thanh toán |
| 3 | bikeId | long |  | Mã định danh xe |
| 4 | startTime | Timestamp |  | Thời điểm bắt đầu thuê xe |
| 5 | cardByCardId | CardEntity |  | Thông tin thẻ tương ứng với cardId |
| 6 | bikeByBikeId | BikeEntity |  | Thông tin xe đang thuê tương ứng |

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | getCardByCardId | ***Kiểu dữ liệu trả về*** | CardEntity |
| ***Mô tả*** | Lấy về thông tin thẻ tương ứng với cardId | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
|  |  |  |
|  | | | | |
| **2** | ***Tên*** | getBikeByBikeId | ***Kiểu dữ liệu trả về*** | BikeEntity |
| ***Mô tả*** | Lấy về thông tin xe tương ứng với xe được thuê | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
|  |  |  |

**Method**

Không

**State**

Không

#### Gói utils

##### Lớp HttpConnector

**Mục đích sử dụng**

Thực hiện HTTP connect, các method POST, GET, PATCH, ...

**Attribute**

Không

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | sendPatch | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Gửi một HTTP request dạng PATCH và lấy kết quả trả về | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| url | String | Đường dẫn tới máy chủ cần request |
| body | String | Nội dung gói tin gửi trong request |
|  | | | | |
| **2** | ***Tên*** | sendPost | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Gửi một HTTP request dạng POST và lấy kết quả trả về | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| url | String | Đường dẫn tới máy chủ cần request |
| body | String | Nội dung gói tin gửi trong request |

**Method**

Không

**State**

Không

##### Lớp BikeTypeUtils

**Mục đích sử dụng**

Mapping từ mã loại xe (dạng int) sang tên loại xe (dạng String) tương ứng.

**Attribute**

Không

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | translateType | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Mapping từ mã loại xe (dạng int) sang tên loại xe (dạng String) tương ứng. | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| type | int | Mã loại xe:   * 1: Xe đạp đơn * 2: Xe đạp đôi * 3: Xe đạp điện * 4: Xe đạp điện đôi |

**Method**

Không

**State**

Không

##### Lớp BarcodeUtils

**Mục đích sử dụng**

Gọi Barcode API, chuyển mã vạch thành mã xe tương ứng.

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | httpConnector | HttpConnector |  | Đối tượng sử dụng để thực hiện gửi HTTP request đến Barcode API |

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | getBikeByBarcode | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Lấy về thông tin xe có mã vạch tương ứng | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| barcode | String | Mã vạch của xe |

**Method**

Không

**State**

Không

##### Lớp FeeCalculator

**Mục đích sử dụng**

Tính toán chi phí thuê xe

**Attribute**

Không

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | calculateFee | ***Kiểu dữ liệu trả về*** | Long |
| ***Mô tả*** | Tính toán chi phí thuê xe | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| renttransactionEntity | RentTransactionEntity | Giao dịch thuê xe tương ứng |

**Method**

Không

**State**

Không

#### Gói requestInterface

##### Lớp RequestModel

**Mục đích sử dụng**

Chứa thông tin nhận được từ front-end

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | cardID | String | NULL | ID định danh thẻ mỗi lần thanh toán |
| 2 | barcode | String | NULL | Mã vạch của xe |
| 3 | parkinglotID | String | NULL | Mã bãi xe |
| 4 | cardCode | String | NULL | Mã thẻ thanh toán |
| 5 | cardOwner | String | NULL | Tên chủ thẻ |
| 6 | cvv | String | NULL | Card Verification Value |
| 7 | expireDate | String | NULL | Ngày hết hạn |

**Operation**

Không

**Method**

Không

**State**

Không

#### Interbank subsystem

##### InterbankInterface

**Mục đích sử dụng**

Định nghĩa các phương thức để kết nối thanh toán

**Attribute**

Không

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | processPayTransaction | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Xử lý giao dịch loại thanh toán | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| transaction | Object | Thông tin thanh toán |
|  | | | | |
| **2** | ***Tên*** | processReturnTransaction | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Xử lý giao dịch loại hoàn tiền | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| transaction | Object | Thông tin thanh toán |
|  | | | | |
| **3** | ***Tên*** | codeToDetail | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Hàm giải mã lỗi tương ứng | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| code | String | Mã lỗi trả về |

**Method**

Không

**State**

Không

##### InterbankSystemController

**Mục đích sử dụng**

Điều khiển các phương thức thanh toán với Interbank

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | transactionPath | String | api/card/processTransaction | Đường dẫn tới API xử lý giao dịch |
| 2 | baseUrl | String | https://ecopark-system-api.herokuapp.com/ | Tên miền của API |
| 3 | secretKey | String |  | Mã bí mật của app với API |
| 4 | interbankBoundary | InterbankBoundary |  | Đối tượng sử dụng để giao tiếp với API |

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | processTransaction | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Phương thức thực hiện giao dịch | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| transaction | InterbankTransaction | Thông tin giao dịch thanh toán |
|  | | | | |
| **2** | ***Tên*** | processReturnTransaction | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Phương thức thực hiện giao dịch hoàn tiền | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| transaction | Object | Thông tin giao dịch hoàn tiền |
|  | | | | |
| **3** | ***Tên*** | processPayTransaction | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Phương thức thực hiện giao dịch thanh toán | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| transaction | Object | Thông tin giao dịch hoàn tiền |
|  | | | | |
| **4** | ***Tên*** | codeToDetail | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Hàm chuyển mã lỗi thành message tương ứng | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| code | String | Mã lỗi |

**Method**

Không

**State**

Không

##### Lớp InterbankBoundary

**Mục đích sử dụng**

Giao tiếp với API Interbank, trả về mã lỗi.

**Attribute**

Không

**Operation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | ***Tên*** | processTransaction | ***Kiểu dữ liệu trả về*** | String |
| ***Mô tả*** | Phương thức truy vấn API | | |
| ***Danh sách tham số*** | *Tên* | *Kiểu dữ liệu* | *Mô tả* |
| url | String | Địa chỉ của API thanh toán |
| body | String | Nội dung của giao dịch thanh toán |

**Method**

Không

**State**

Không

##### Lớp InterbankTransaction

**Mục đích sử dụng**

Chức các thông tin dùng để thực hiện thanh toán với Interbank

**Attribute**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *#* | *Tên* | *Kiểu dữ liệu* | *Giá trị mặc định* | *Mô tả* |
| 1 | amount | String | NULL | Số tiền giao dịch |
| 2 | cardCode | String | NULL | Mã định danh thẻ |
| 3 | cvvCode | String | NULL | Card Verification Value |
| 4 | dateExpired | String | NULL | Ngày hết hạn |
| 5 | owner | String | NULL | Tên chủ thẻ |
| 6 | transactionContent | String | NULL | Nội dung giao dịch |
| 7 | createdAt | String | NULL | Thời điểm tạo |
| 8 | command | String | NULL | Loại giao dịch: thanh toán hoặc hoàn tiền |

**Operation**

Không

**Method**

Không

**State**

Không

# Các vấn đề thiết kế

## Coupling and cohesion

Thiết kế của hệ thống tuân theo tính chất high cohesion and loose coupling.

### High cohesion

Các phương thức và thuộc tính của một lớp nếu chỉ được gọi đến ngay bên trong lớp thì sẽ được để là private. Điều này đảm bảo các phương thức này chỉ phục vụ cho các thành phần ở bên trong lớp và không phục vụ các lớp khác ở bên ngoài. Ví dụ:

* Lớp InterbankController có phương thức processTransaction được để là private. Phương thức này được gọi bởi 2 phương thức public khác là processPayTransaction và processReturnTransaction.
* Lớp PaymentController có các phương thức payUpFront và finalPay được để là private, lần lượt được sử dụng bởi 2 phương thức public khác là payUpFrontControl và finalPayControl.
* Trong lớp HttpConnector, thuộc tính client được để private và chỉ phục vụ mục đích sử dụng ngay bên trong lớp này.

### Loose coupling

Vì trong mỗi lớp có độ cohesion cao nên độ coupling giữa các lớp sẽ thấp xuống. Mỗi lớp chỉ cung cấp ra bên ngoài một public API là các phương thức public có thể gọi được từ lớp đó. Khi có thay đổi về logic xử lý trong một lớp, các lớp bên ngoài phụ thuộc vào public API sẽ không cần phải thay đổi gì, các thay đổi sẽ chỉ cần thực hiện ở bên trong từng lớp, đảm bảo rằng public API vẫn được giữ nguyên. Ví dụ:

* Trong lớp PaymentController, nếu cần thay đổi trong phương thức payUpFront, chỉ cần có một thay đổi nhỏ kéo theo trong phương thức payUpFrontControl, mà không làm ảnh hưởng đến bất kỳ thành phần nào khác trong hệ thống.

Độ coupling thấp còn thể hiện ở sự độc lập giữa lớp PaymentController và Interbank subsystem. Cụ thể:

* PaymentController không phụ thuộc trực tiếp vào Interbank subsystem mà phụ thuộc vào một interface định nghĩa các dịch vụ cung cấp bởi subsystem. Như vậy, khi có thay đổi bên phía Interbank, có thể là thay đổi toàn bộ các API liên quan đến thanh toán, ta có thể viết một subsystem mới implement InterbankInterface mà không khiến cho các thành phần phụ thuộc khác như lớp PaymentController ngừng hoạt động.

## Các nguyên tắc thiết kế

*Thiết kế của hệ thống tuân theo nguyên tắc SOLID. Các phần phía dưới mô tả các thiết kế của hệ thống theo 5 nguyên tắc của SOLID.*

### Single Responsibility Principle

Nguyên tắc này nói rằng mỗi một lớp chỉ nên đảm nhận một trách nhiệm duy nhất. Nếu nó đảm nhận nhiều hơn một trách nhiệm, lớp đó nên được thành ra thành một vài lớp khác nhau.

Trách nhiệm của hệ thống được phân bổ tới các package, các subsystem và trong mỗi package, subsystem, trách nhiệm được chia nhỏ cho từng lớp, mỗi lớp đảm nhận một trách nhiệm duy nhất. Bảng dưới đây mô tả trách nhiệm của một số lớp đại diện cho các package sử dụng trong hệ thống.

|  |  |  |
| --- | --- | --- |
| *Class* | *Package* | *Responsibility* |
| PaymentController | controller | Lớp điều khiển các tác vụ liên quan đến thanh toán hóa đơn thuê và trả xe. Lớp này sẽ không phụ trách các nghiệp vụ không liên quan đến thanh toán. |
| BikeRepository | repository | Lớp phụ trách việc thực hiện các truy vấn đến bảng Bike trong cơ sở dữ liệu. Mỗi bảng trong cơ sở dữ liệu được truy vấn bởi một lớp Repository khác nhau |
| BikeEntity | entity | Lớp này mapping với bảng entity trong cơ sở dữ liệu. Mỗi bảng trong cơ sỡ dữ liệu được mapping tương ứng đến một lớp Entity. |
| HttpConnector | utils | Lớp này đảm nhận việc thực hiện các HTTP request đến một server. Tất cả các hàm của lớp đều phục vụ mục đích này, ví dụ: sendPatch, sendPost; đồng thời lớp cũng không đảm nhận chức năng nào khác. |

### Open/Closed Principle

Open/Closed principle được hiểu là open for extension và closed for modification. Các thay đổi của một lớp, hay một component phải là trong suốt với các lớp, các component phụ thuộc (close for modification). Đồng thời, việc mở rộng mã nguồn không nên làm ảnh hưởng đến các thành phần đã có (open for extension). Nguyên tắc này được thực hiện bằng việc sử dụng các interface. Để thay đổi hay mở rộng các chức năng cung cấp bởi một interface, ta chỉ cần viết một lớp mới implement lại interface này.

Interbank subsystem implement các phương thức được định nghĩa trong Interbank interface. Các lớp của hệ thống chỉ phụ thuộc vào Interbank Interface chứ không phụ thuộc trực tiếp vào Interbank subsystem. Do đó, có thể dễ dàng thay thế subsystem sẵn có bằng một subsystem khác hoặc thêm một số phương thức khác cho Interbank interface và implement các phương thức này trong subsystem. Các thay đổi bên phía subsystem hoàn toàn trong suốt với các bên liên quan sử dụng giao diện của Interbank interface.

### Liskov substitution principle

Nguyên tắc này yêu cầu một lớp con kế thừa từ một lớp cha sẽ có thể sử dụng để thay thế lớp cha trong bất kì tình huống nào. Để đảm bảo nguyên tắc này thì các phương thức ghi đè hoặc các phương thức implement lại một phương thức abstract phải đảm bảo các yêu cầu sau:

* Các tham số của phương thức ghi đè phải có miền giá trị rộng hơn tham số của phương thức bị ghi đè. Điều này đảm bảo một tham số truyền cho lớp cha cũng có thể truyền cho lớp con mà không gây nên lỗi.
* Giá trị trả về của phương thức ghi đè phải giống với hoặc là lớp con của giá trị trả về của phương thức bị ghi đè.

Trong thiết kế của hệ thống hiện tại, lớp PaymentController có một thuộc tính tên interbankSubsystem với kiểu dữ liệu là InterbankInterface. Lớp InterbankSystemController là một lớp implement InterbankInterface. Trong mọi tình huống, kiểu dữ liệu của thuộc tính interbankSubsystem có thể chuyển thành InterbankSystemController hoặc bất kỳ lớp nào khác implement InterbankInterface.

### Interface segregation principle

Nguyên tắc chỉ ra rằng các interface không nên chứa quá nhiều phương thức, vì nếu chúng ta cần 2 implementations của 1 interface với sự khác nhau rất ít (có thể chỉ khác nhau một phương thức duy nhất), ta sẽ phải copy code của toàn bộ những phương thức giống nhau giữa 2 implementations này 2 lần. Điều tương tự xảy ra nếu chúng ta có nhiều implementations của cùng một interface nhưng các implementations này lại không có quá nhiều khác biệt.

Trong thiết kế của hệ thống hiện tại, toàn bộ gói repository đều là các interface. Các interface này là các Java Persistence API. Chúng định nghĩa các phương thức liên quan đến object-relational-mapping. Một implementation của API này có thể tương ứng với loại cơ sở dữ liệu quan hệ. Do đó, các implementations khác nhau sẽ không có các phương thức bị lặp lại ở nhiều hơn một implementation. Điều này có nghĩa là các phương thức định nghĩa bởi API đã được phân tách một cách hợp lý và đảm bảo đúng nguyên tắc Interface segregation.

Đối với interface InterbankInterface, có 3 phương thức được định nghĩa là: processPayTransaction, processReturnTransaction và codeToDetail. Với mỗi hệ thống ngân hàng, cách thức xử lý giao dịch và các thông báo trả về là khác nhau. Sẽ không có sự trùng lặp mã nguồn giữa các implementations khác nhau của InterbankInterface. Do đó, thiết kế của InterbankInterface cũng tuân theo nguyên tắc Interface segregation.

### Dependency Inversion principle

Nguyên tắc này nói rằng các lớp ở tầng trên không nên sử dụng trực tiếp dịch vụ cung cấp bởi các lớp ở tầng dưới mà nên sử dụng thông qua một interface. Việc này giúp phân tách hoạt động giữa các tầng (layer) trong hệ thống, làm giảm coupling giữa các thành phần trong hệ thống. Nếu không tuân theo nguyên tắc này, hệ thống sẽ trở nên khó sửa chữa và mở rộng.

Trong thiết kế của hệ thống hiện tại, các lớp sử dụng dịch vụ của gói repository sẽ không cần phải thay đổi nếu hệ thống sử dụng SQLServer thay vì MySQL. Như vậy, các lớp phụ trách nghiệp vụ như các lớp trong gói controller không phụ thuộc vào các module cấp thấp như module kết nối cơ sở dữ liệu. Thay vào đó, các lớp của controller sử dụng các interface của gói repository và các module kết nối cơ sở dữ liệu sẽ implement các phương thức truy vấn cơ sở dữ liệu cụ thể. Điều này tuân theo nguyên tắc Dependency Inversion.

## Design Pattern

Do hệ thống hiện tại không quá phức tạp nên các design pattern chưa được sử dụng. Tuy nhiên, hệ thống sử dụng các thư viện và framework và các thành phần này có sử dụng một số design pattern như sau:

* Simple Logging Facade for Java (SLF4J): Thư viện này sử dụng các design pattern như Singleton, Facade, Factory method, cụ thể:
* Hàm static LoggerFactory.getLogger(Class<>) trả về một đối tượng log tương ứng với class truyền vào. Hàm này sử dụng singleton design pattern.
* Toàn bộ các chức năng cung cấp bởi thư viện được đóng gói và cung cấp cho người dùng một API để có thể dễ dàng sử dụng mà không cần quan tâm đến logic phức tạp bên trong. Đây chính là facade design pattern.
* Thư viện này sử dụng một LoggerFactory để tạo ra các logger. Người dùng chỉ cần truyền vào một tham số là tên lớp và mọi công việc khởi tạo phức tạp đằng sau sẽ được factory xử lý. Đây là chính là Factory method design pattern.
* Java Persistence API: Thư viện này sử dụng Singleton design pattern, cụ thể:
* Các lớp PaymentController và RentBikeController đều có một thuộc tính với kiểu dữ liệu là RentTransactionRepository. Hai đối tượng tương ứng với 2 thuộc tính này thực chất chỉ là một. Java Persistence API sử dụng annotation @Autowired cho các thuộc tính này và các đối tượng loại repository này chỉ được tạo ra đúng một lần.