

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  
Viện Công nghệ thông tin và Truyền thông

Tài liệu mô tả thiết kế phần mềm  
(Software Design Description)  
Phiên bản 1

THIẾT KẾ VÀ XÂY DỰNG PHẦN MỀM  
CHO HỆ THỐNG EcobikeRental  
Môn: Thiết kế và xây dựng phần mềm

Nhóm 18

Họ tên sinh viên	MSSV
Trần Văn Trí	20173410
Đỗ Viết Trí	20173412
Nguyễn Ngọc Trinh	20173413
Nguyễn Mạnh Trường	20177022

Giảng viên hướng dẫn: TS.Nguyễn Thị Thu Trang

*Hà Nội, ngày 2 tháng 1 năm 2021*



# Mục lục

1	Giới thiệu.....	3
1.1	Mục đích .....	3
1.2	Phạm vi .....	3
1.3	Từ điển thuật ngữ.....	3
1.4	Tham khảo.....	3
2	Thiết kế kiến trúc .....	3
2.1	Lựa chọn kiến trúc phần mềm .....	3
2.2	Thiết kế tổng quan .....	5
2.3	Thiết kế chi tiết gói.....	6
2.3.1	Gói application.....	6
2.3.2	Gói Model.....	7
2.3.3	Gói controller.....	8
2.3.4	Gói views.....	9
2.3.5	Gói utilities .....	10
2.3.6	Gói Subsystem .....	10
2.4	Biểu đồ tương tác.....	11
2.4.1	Biểu đồ tương tác cho UC001 – Xem thông tin bãi xe .....	11
2.4.2	Biểu đồ tương tác cho UC002 – Xem thông tin chi tiết xe .....	11
2.4.3	Biểu đồ tương tác cho UC003 - Thuê xe .....	12
2.4.4	Biểu đồ tương tác cho UC004 – Trả xe .....	13
3	Thiết kế giao diện.....	14
3.1	Giao diện với thiết bị phần cứng.....	14
3.2	Giao diện với phần mềm khác .....	14
3.3	Giao diện người dùng .....	17
3.3.1	Biểu đồ dịch chuyển màn hình.....	17
3.3.2	Thiết kế giao diện.....	17
4	Thiết kế lớp.....	24
4.1	Biểu đồ lớp thiết kế.....	24
4.2	Thiết kế lớp chi tiết.....	25
4.2.1	Thiết kế chi tiết lớp Contants.....	25
4.2.2	Thiết kế lớp InterbankService .....	27
4.2.3	Thiết kế lớp Bike.....	28
4.2.4	Thiết kế chi tiết lớp Station.....	29
4.2.5	Thiết kế chi tiết lớp Card .....	31

4.2.6	Thiết kế chi tiết lớp Customer.....	32
4.2.7	Thiết kế chi tiết lớp Rent .....	33
4.2.8	Thiết kế chi tiết lớp HomeController.....	35
4.2.9	Thiết kế chi tiết lớp ListStationController .....	36
4.2.10	Thiết kế chi tiết lớp PaymentFormController .....	37
4.2.11	Thiết kế chi tiết lớp ReturnBikeController .....	38
4.2.12	Thiết kế chi tiết lớp ViewBikeController.....	39
4.2.13	Thiết kế chi tiết lớp ViewStationController .....	40
5	Thiết kế mô hình dữ liệu.....	41
5.1	Mô hình dữ liệu mức khái niệm.....	41
5.2	Mô hình dữ liệu mức logic .....	41
5.3	Thiết kế chi tiết.....	42
5.3.1	Bảng Station(Bãi xe) .....	42
5.3.2	Bảng Bike (Xe) .....	43
5.3.3	Bảng Customer (Khách hàng).....	43
5.3.4	Bảng Card (Tài khoản thẻ) .....	44
5.3.5	Bảng Transaction (Thông tin giao dịch) .....	44
5.3.6	Bảng Rent (Thuê xe) .....	45
6	Xem xét thiết kế.....	46
6.1	Mục tiêu và Nguyên tắc (Goals and Guidelines) .....	46
6.2	Chiến lược kiến trúc ( <i>Architectural Strategies</i> ).....	46
6.3	Coupling và Cohesion.....	46
6.4	Nguyên tắc thiết kế (Design Principles).....	48
6.5	Mẫu thiết kế (Design patterns) .....	50

# 1 Giới thiệu

## 1.1 Mục đích

Tài liệu này đưa ra mô tả chi tiết cho người dùng về thiết kế kiến trúc, thiết kế giao diện và thiết kế lớp cho từng chức năng của hệ thống, cũng như việc thiết kế cơ sở dữ liệu của cả hệ thống thuê xe EcobikeRental. Từ đó các bên liên quan sẽ có cái nhìn rõ ràng hơn về phần mềm cần xây dựng.

Tài liệu dành cho các bên liên quan (stakeholder) và các nhà phát triển phần mềm.

## 1.2 Phạm vi

Hệ thống sử dụng mô hình MVC để cài đặt chương trình. Hệ thống có tương tác với ngân hàng để xử lý giao dịch thanh toán hoặc hoàn tiền.

## 1.3 Từ điển thuật ngữ

STT	Thuật ngữ	Giải thích
1	MVC	Mô hình MVC viết tắt của Model-View-Controller

## 1.4 Tham khảo

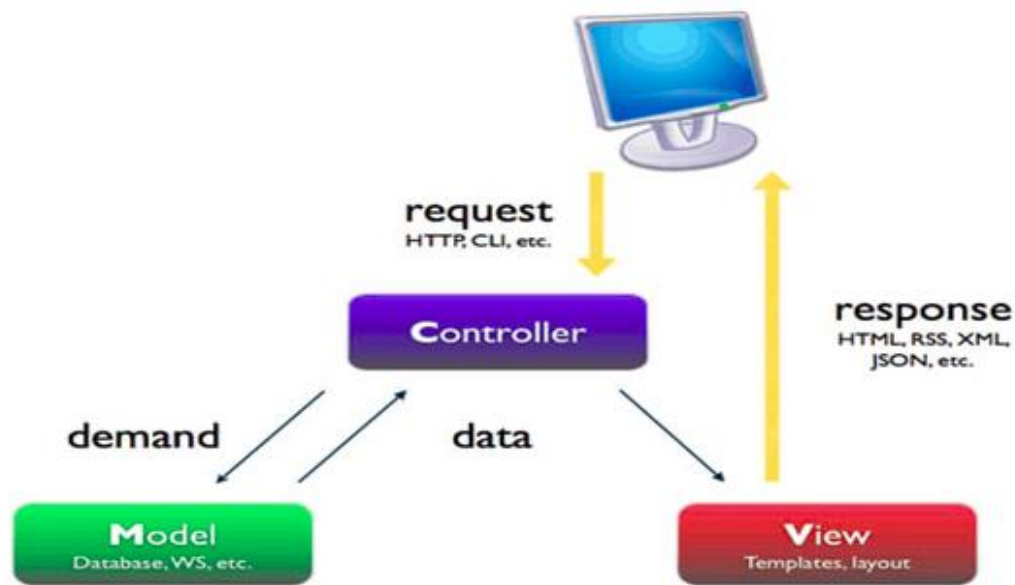
Tài liệu tham khảo được sử dụng trong tài liệu này

+ Tài liệu đặt tả yêu cầu phần mềm SRS

# 2 Thiết kế kiến trúc

## 2.1 Lựa chọn kiến trúc phần mềm

Hệ thống xây dựng theo kiến trúc mô hình MVC



### - Các thành phần trong kiến trúc:

- + Model: Đây là thành phần chứa tất cả các nghiệp vụ logic, phương thức xử lý, truy xuất database, đối tượng mô tả dữ liệu như các class, hàm xử lý, ....
- + View: Đảm nhận việc hiển thị thông tin, tương tác với người dùng, nơi chứa tất cả các đối tượng GUI như textbox, images,... Hiểu một cách đơn giản, nó là tập hợp các form hoặc các file HTML
- + Controller: Giữ nhiệm vụ nhận điều hướng các yêu cầu từ người dùng và gọi đúng những phương thức xử lý chúng,... Chẳng hạn thành phần này sẽ nhận request từ url và form nào để thao tác trực tiếp với model.

### - Luồng hoạt động trong MVC

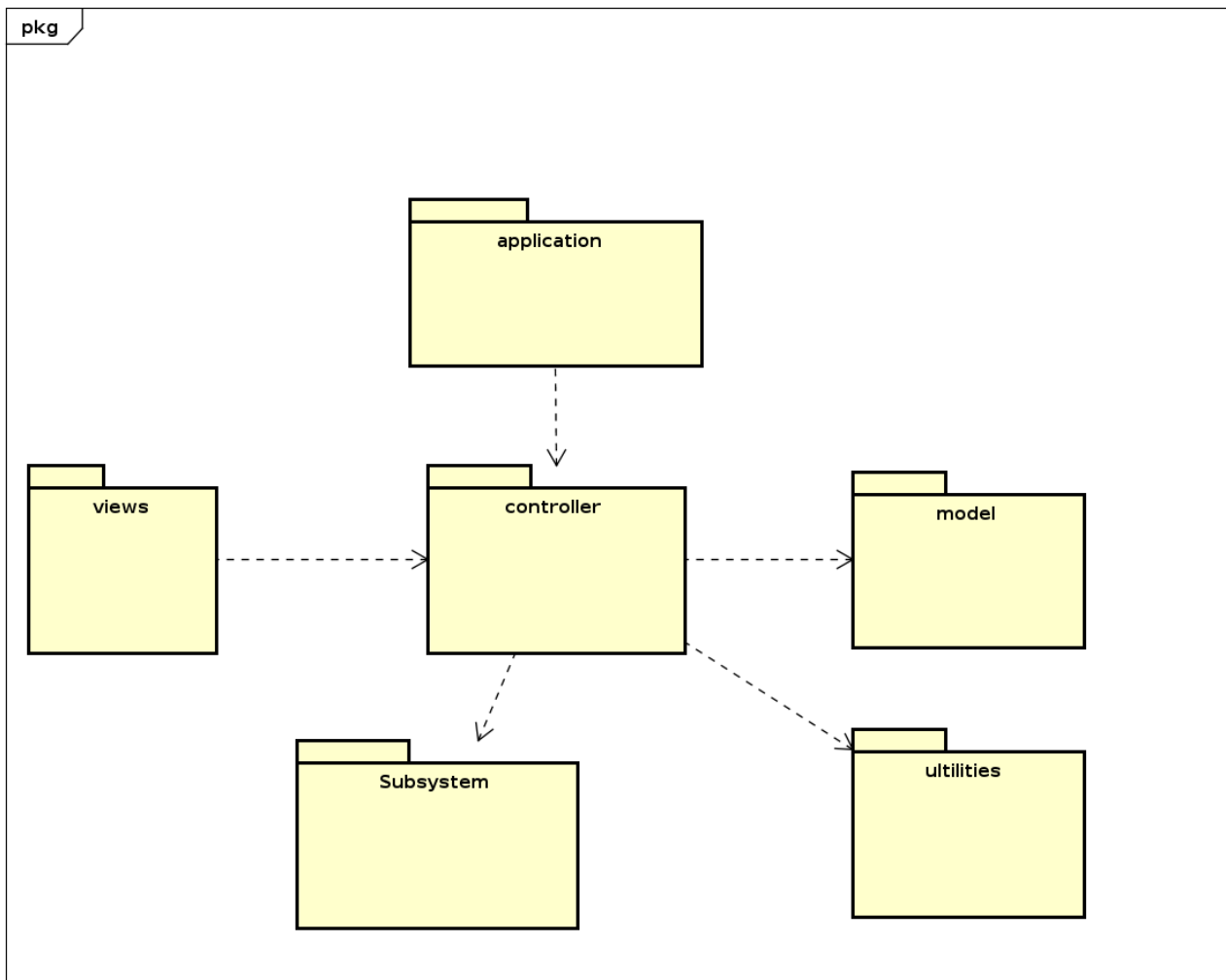
Khi có một yêu cầu từ phía client gửi đến server, Bộ phận controller có nhiệm vụ nhận yêu cầu, xử lý yêu cầu đó. Và nếu cần, nó sẽ gọi đến phần model, vốn là bộ phận làm việc với Database..

Sau khi xử lý xong, toàn bộ kết quả được đẩy về phần View. Tại View, sẽ gen ra mã Html tạo nên giao diện, và trả toàn bộ html về trình duyệt để hiển thị.

### - Ưu điểm và nhược điểm của MVC

- + Ưu điểm: Thể hiện tính chuyên nghiệp trong lập trình, phân tích thiết kế. Do được chia thành các thành phần độc lập nên giúp phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp, bảo trì,...
- + Nhược điểm: Đối với dự án nhỏ việc áp dụng mô hình MVC gây cồng kềnh, tốn thời gian trong quá trình phát triển. Tốn thời gian trung chuyển dữ liệu của các thành phần.

## 2.2 Thiết kế tổng quan



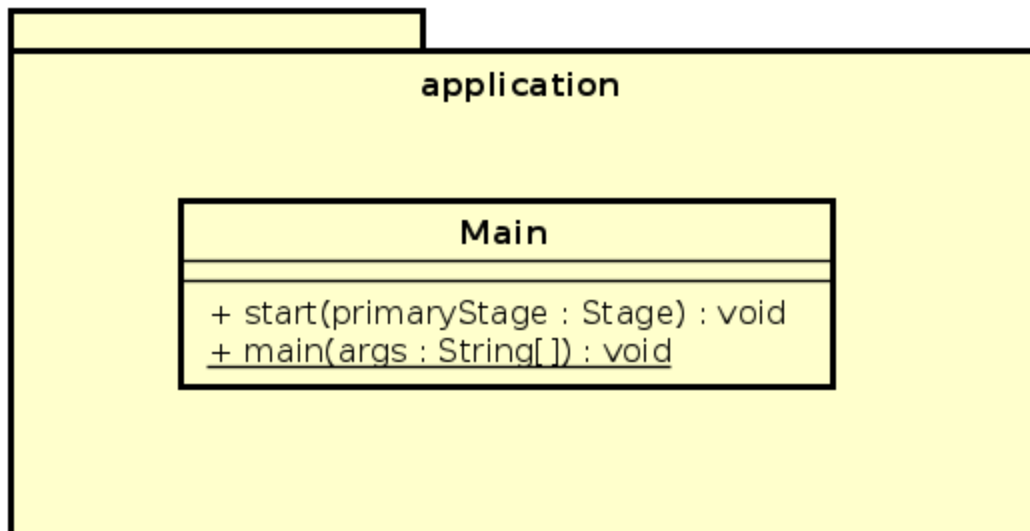
**Hình 1 biểu đồ phụ thuộc gói.**

- Gói application:
  - Chứa hàm main
- Gói views
  - Chứa toàn bộ lớp giao diện của chương trình
- Gói controllers
  - Chức toàn bộ lớp điều khiển của chương trình
- Gói model
  - Chứa toàn bộ lớp thực thể của chương trình, tương tác với cơ sở dữ liệu
- Gói utilites:
  - Chức các lớp tiện ích cho ứng dụng
- Gói Subsystem

- Chứa lớp giao tiếp với ngân hàng

## 2.3 *Thiết kế chi tiết gói*

### 2.3.1 Gói application



Hình 1 Ví dụ thiết kế gói.

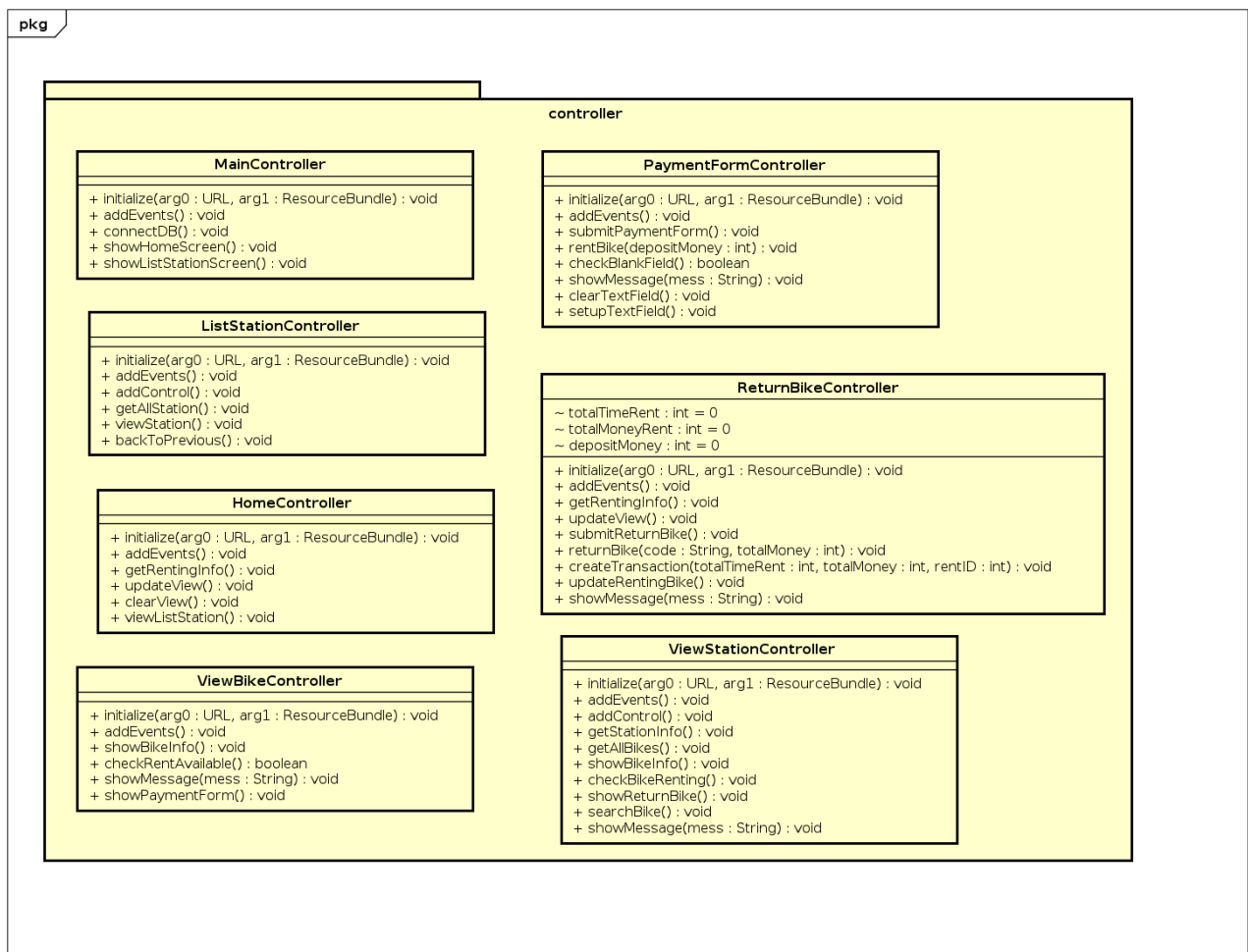


## 2.3.2 Gói Model



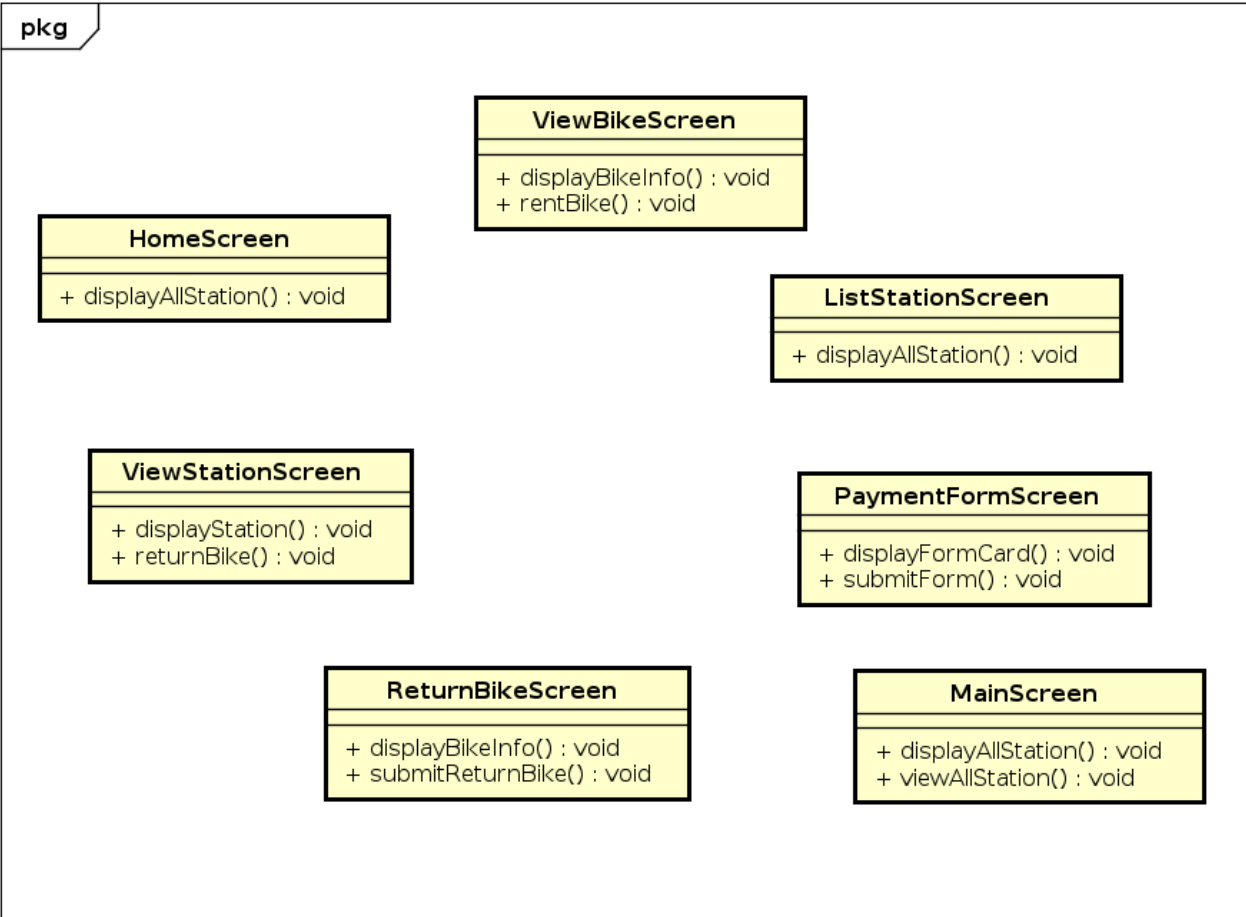
Hình 2: Gói model

### 2.3.3 Gói controller



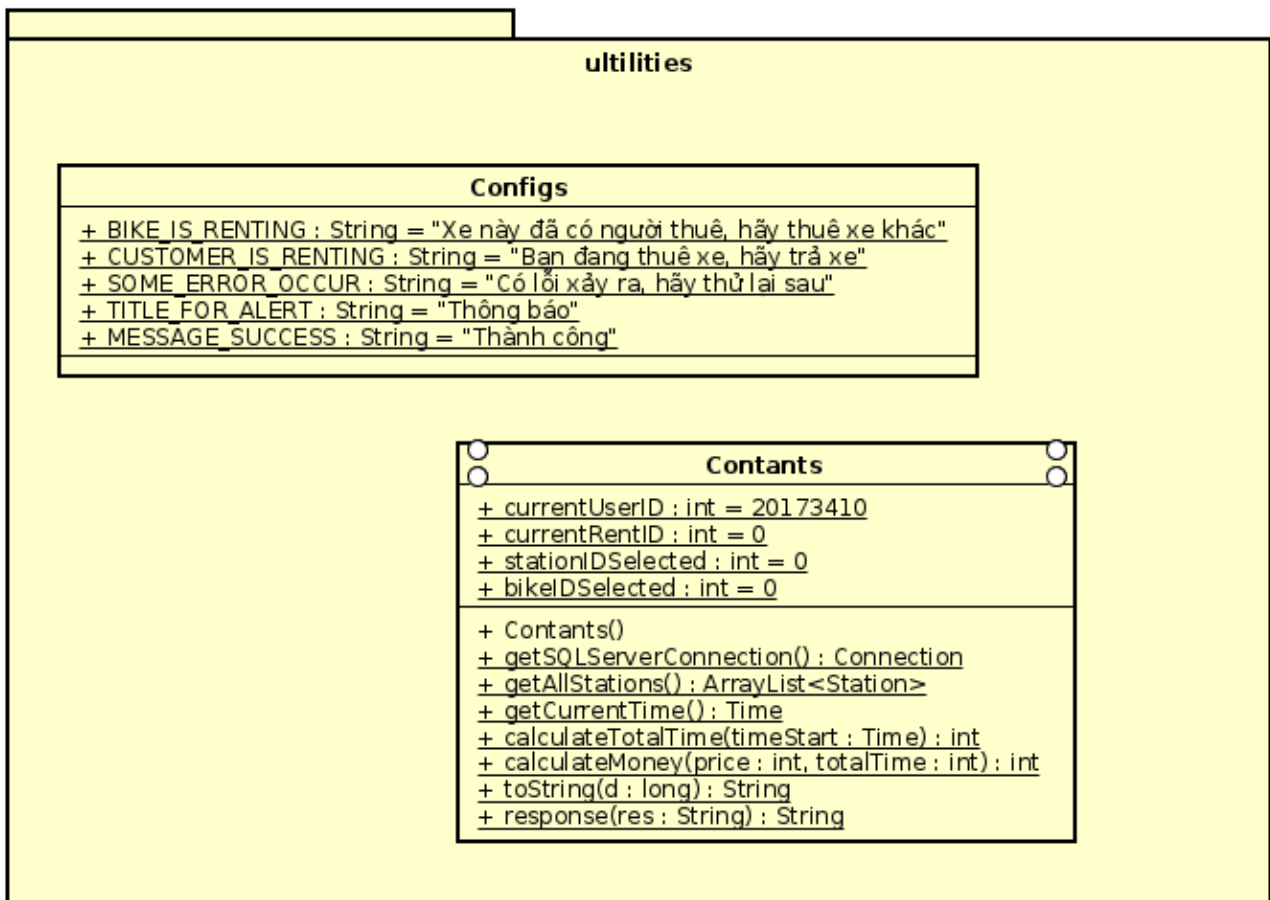
Hình 3: Gói controller

2.3.4 Gói views



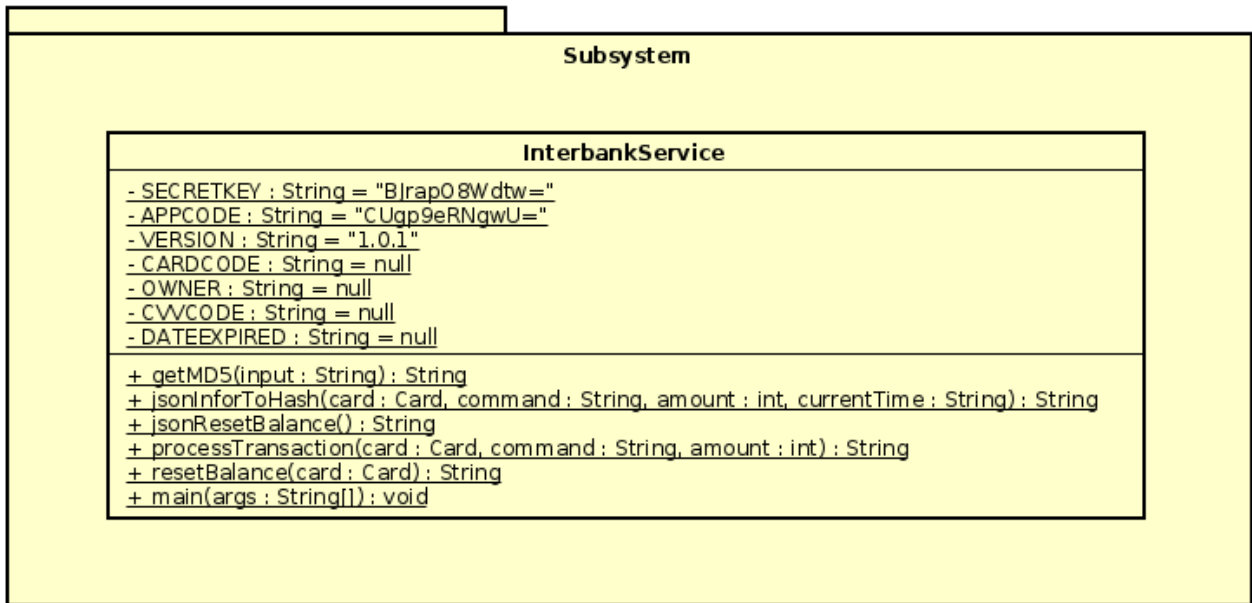
Hình 4: Gói views

### 2.3.5 Gói utilities



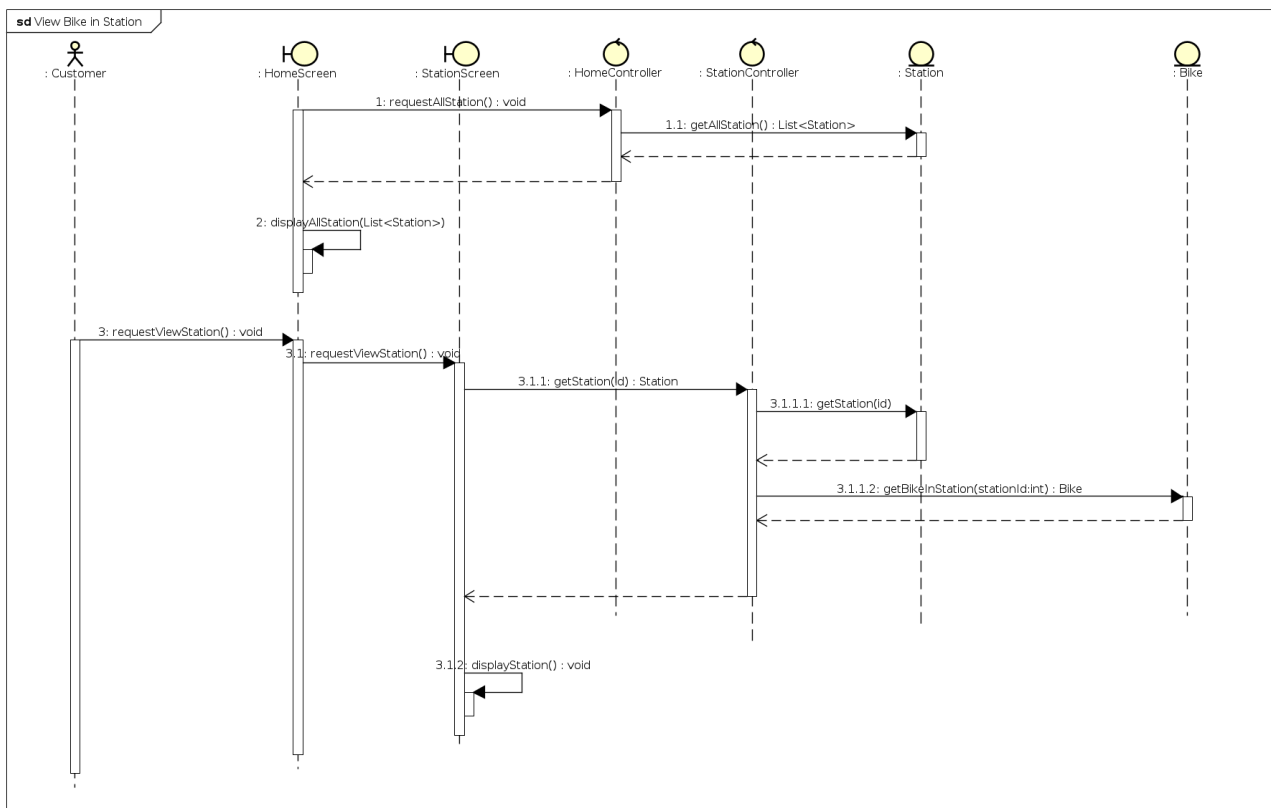
Hình 5: Gói utilities

### 2.3.6 Gói Subsystem

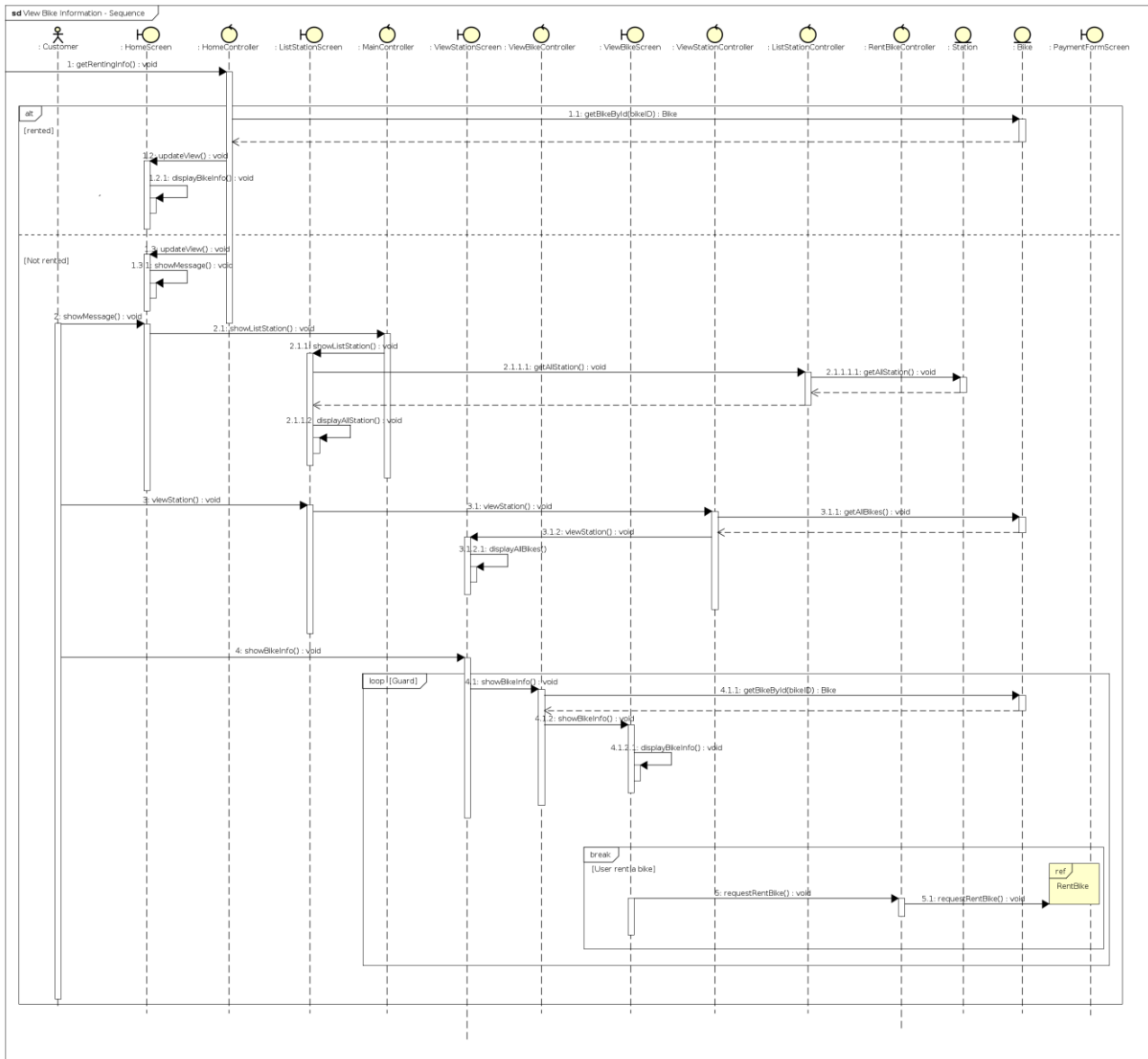


## 2.4 Biểu đồ tương tác

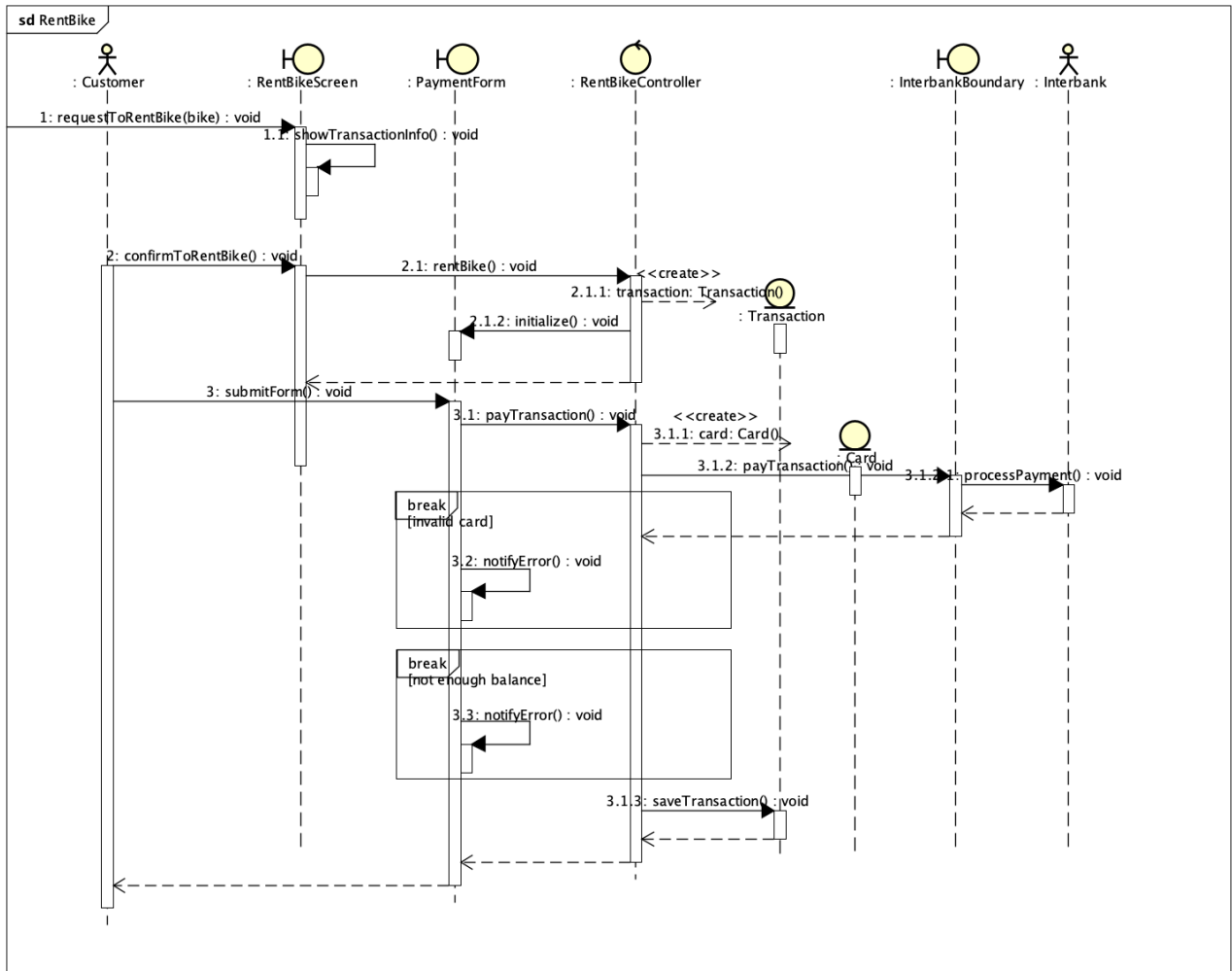
### 2.4.1 Biểu đồ tương tác cho UC001 – Xem thông tin bãi xe



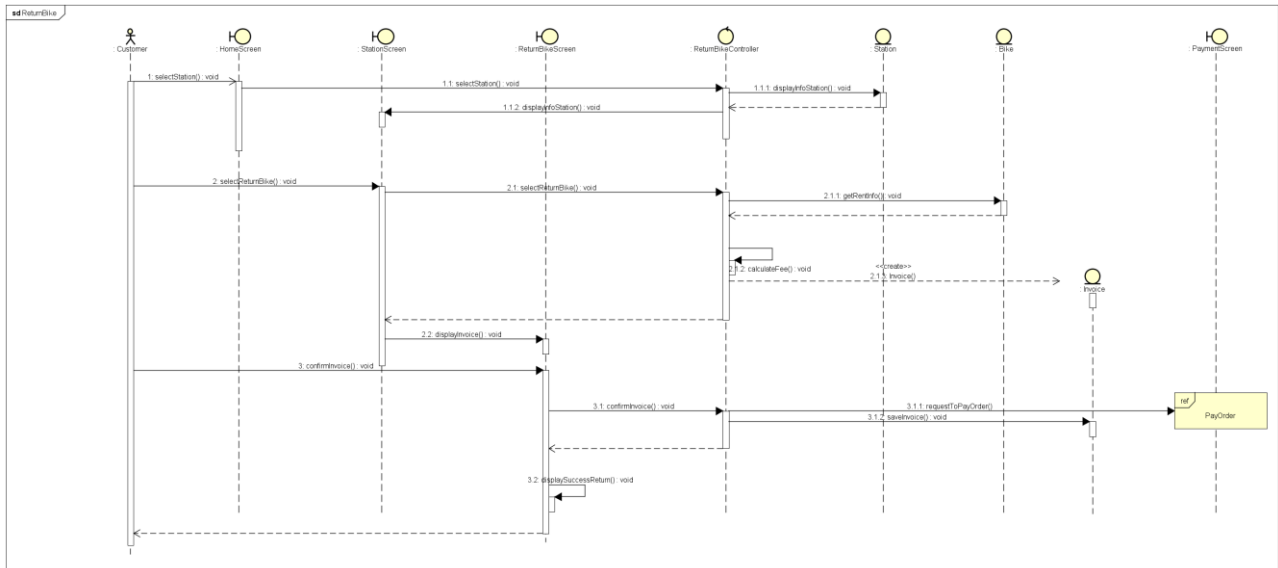
## 2.4.2 Biểu đồ tương tác cho UC002 – Xem thông tin chi tiết xe



### 2.4.3 Biểu đồ tương tác cho UC003 - Thuê xe



#### 2.4.4 Biểu đồ tương tác cho UC004 – Trả xe



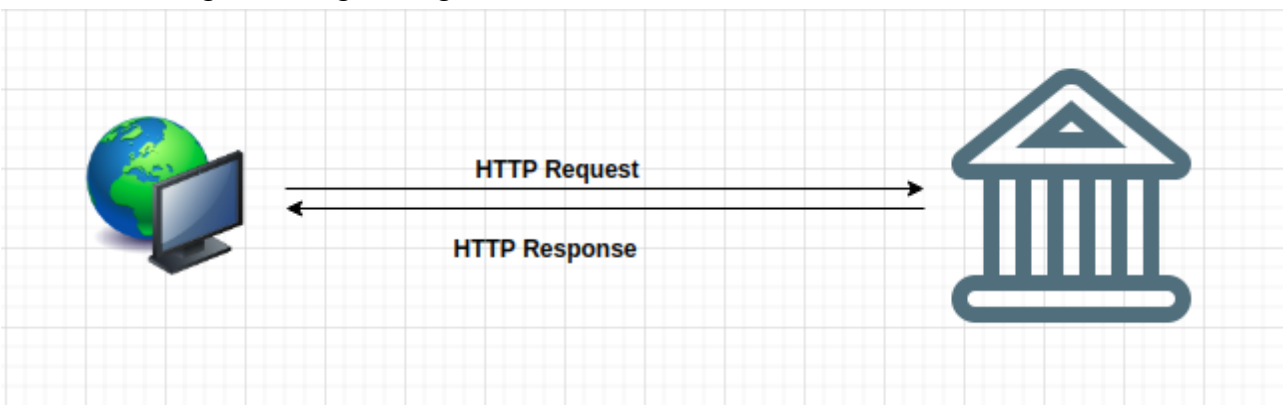
### 3 Thiết kế giao diện

#### 3.1 Giao diện với thiết bị phần cứng

Không tương tác với phần cứng

#### 3.2 Giao diện với phần mềm khác

Phần mềm tương tác với ngân hàng :





Phần mềm tương tác với ngân hàng thông qua giao thức Http qua method PATCH để thanh toán hoặc hoàn tiền:

- Ngân hàng cung cấp API

- Method: PATCH
- Path: <https://ecopark-system-api.herokuapp.com/api/card/processTransaction>

- Mỗi người dùng sẽ được cung cấp 1 cặp key là appCode và secretKey và có một tài khoản ngân hàng riêng

- Dữ liệu cần truyền đi là một chuỗi json

- Định dạng dữ liệu truyền đi

Field	Type	Required	Description
version	String	Yes	Phiên bản API: 1.0.1
transaction	Object	Yes	Giao dịch
cardCode	String	Yes	Mã thẻ
owner	String	Yes	Chủ tài khoản
cvvCode	String	Yes	Mã CVV
dateExpired	String	Yes	Ngày hết hạn
command	String	Yes	Mã API sử dụng, - Mã cho giao dịch thanh toán là pay - Hoàn tiền là refund
transactionContent	String	Yes	Nội dung giao dịch
amount	Number	Yes	Số tiền cần thanh toán
createdAt	String	Yes	Thời điểm tạo giao dịch (cần tuân thủ đúng format “năm-tháng-ngày giờ:phút:giây”)
appCode	String	Yes	Mã app sử dụng hệ thống thanh toán
hashCode	String	Yes	Mã kiểm tra, để đảm bảo không bị thanh đổi khi chuyển từ app lên

			server thanh toán
--	--	--	-------------------

- Trước khi gửi yêu cầu lên ngân hàng thì cần phải băm chuỗi thông tin giao dịch bằng mã băm MD5 kết hợp với secretKey đã được cho trước. Dưới đây là ví dụ một chuỗi JSON băm:

```
{
  "secretKey": "BJrapO8Wdtw=",
  "transaction": {
    "command": "pay",
    "cardCode": "118609_group18_2020",
    "owner": "Group 18",
    "cvvCode": "390",
    "dateExpired": "Thanh toán",
    "amount": 100
  }
}
```

- Sau khi băm xong sẽ lưu thông tin mã băm vào trường hashCode

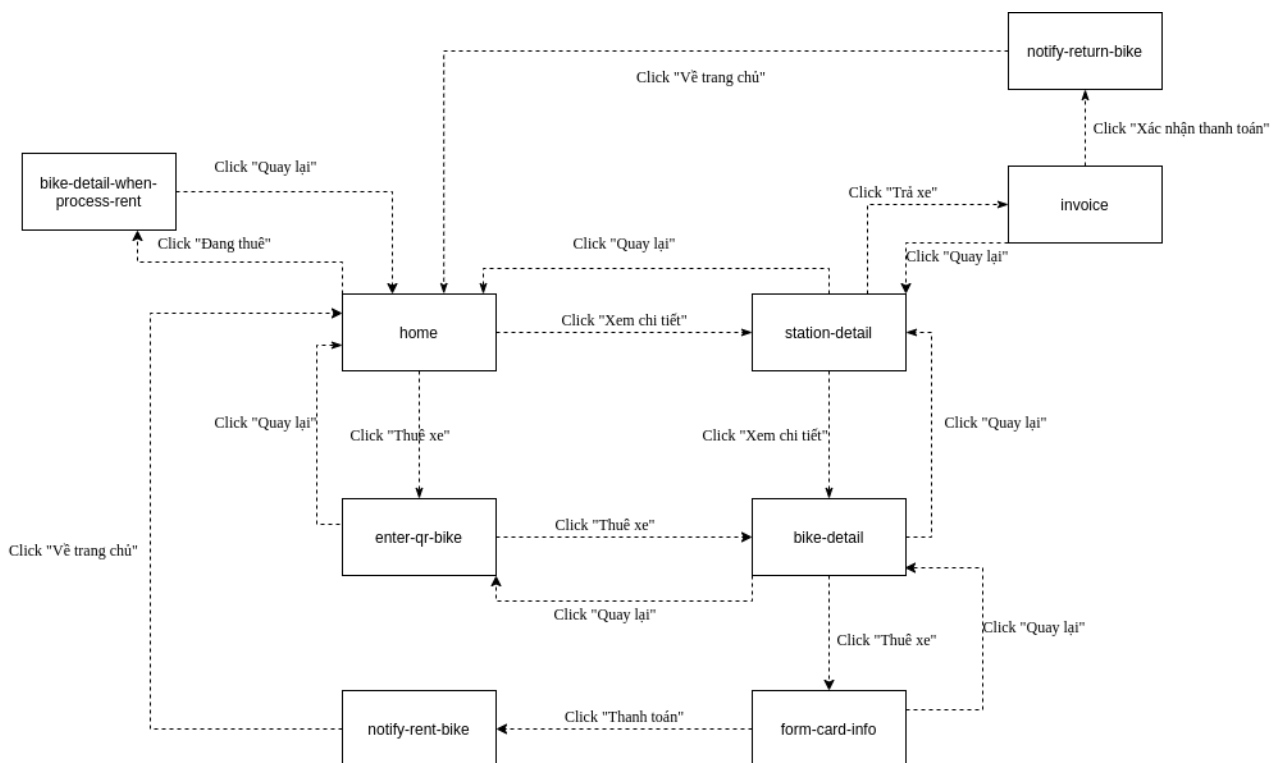
- Dữ liệu trả về:

Field	Type	Description
errorCode	Number	Mã lỗi
transaction	Object	Giao dịch
cardCode	String	Mã thẻ
owner	String	Chủ tài khoản
cvvCode	String	Mã CVV
dateExpired	String	Ngày hết hạn
command	String	Mã API sử dụng, - Mã cho giao dịch thanh toán là pay - Hoàn tiền là refund

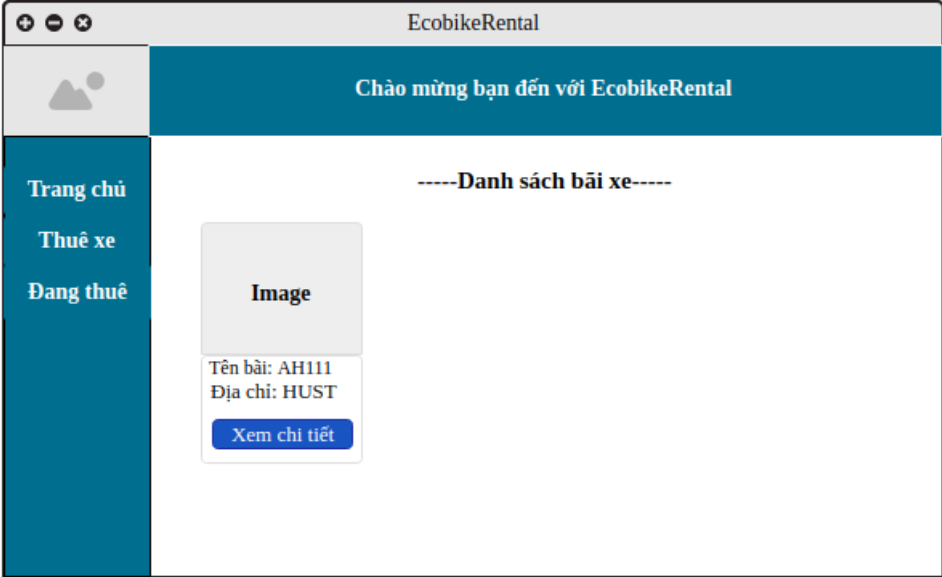
transactionContent	String	Nội dung giao dịch
amount	Number	Số tiền cần thanh toán
createdAt	Date	Thời điểm tạo giao dịch (cần tuân thủ đúng format “năm-tháng-ngày giờ:phút:giây”)

### 3.3 Giao diện người dùng

#### 3.3.1 Biểu đồ dịch chuyển màn hình



### 3.3.2 Thiết kế giao diện

Đặc tả màn hình Trang chủ		
		
Control	Operation	Function
Khu vực hiển thị	Tự sinh	Khi hệ thống khởi chạy, một danh sách các bãi xe hiển thị lên màn hình
Nút "Trang chủ"	Nhấn (Click)	Người dùng ở bất cứ vị trí nào khi nhấn vào nút này thì sẽ đưa người dùng về trang chủ
Nút "Thuê xe"	Nhấn (Click)	Hệ thống sẽ đưa người dùng đến trang nhập mã QR của xe
Nút "Đang thuê"	Nhấn (Click)	Khi người dùng đang thuê xe thì hệ thống sẽ đưa người dùng đến trang thông tin của xe đang thuê còn nếu chưa thuê thì hệ thống sẽ hiển thị thông báo là khách hàng chưa thuê xe
Nút "Xem chi tiết"	Nhấn (Click)	Hệ thống sẽ đưa người dùng đến trang xem chi tiết bãi xe

## Đặc tả màn hình Chi tiết bãi xe

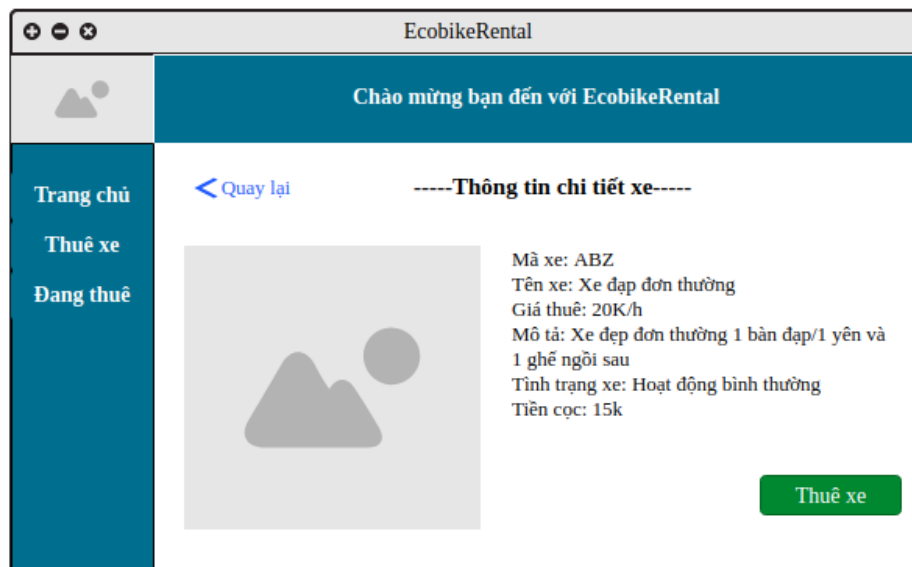
The screenshot shows a web browser window titled 'EcobikeRental'. The page has a dark blue sidebar on the left with links: 'Trang chủ', 'Thuê xe', and 'Đang thuê'. The main content area has a header with a welcome message and a 'Quay lại' button. Below this is a section titled 'Thông tin chi tiết bãi gửi xe'. This section is split into two columns. The left column, 'Thông tin bãi xe', contains a table with the following data:

Tên bãi	AH111
Địa chỉ	HUST
Tổng xe	15
Số xe hiện tại	10

Below the table is an orange 'Trả xe' button. The right column, 'Danh sách xe trong bãi', contains a card for a 'Xe đạp đơn' (Single Bike) with a rental price of '20K/h' and a green 'Xem chi tiết' button.

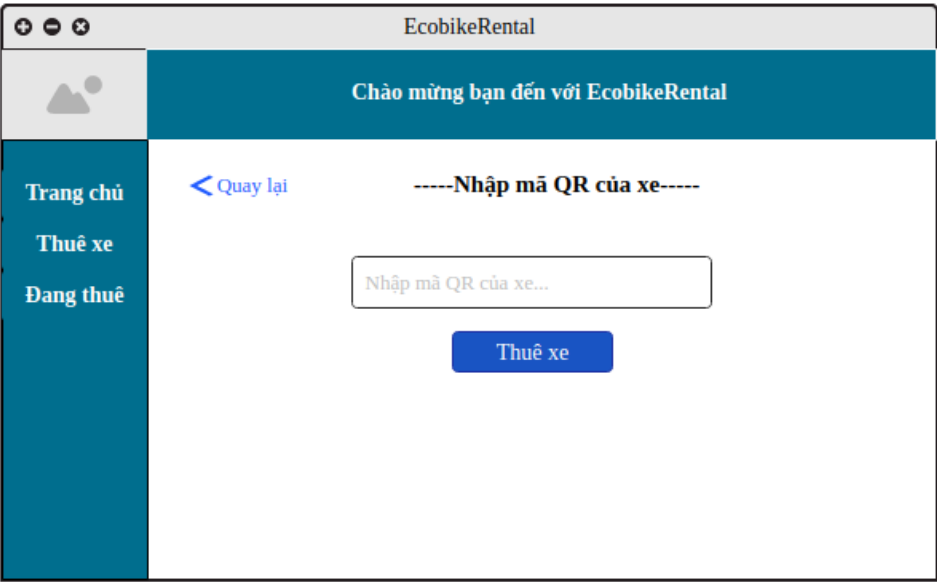
Control	Operation	Function
Khu vực hiển thị	Khi nhấn vào nút "Xem chi tiết" ở màn hình Trang chủ	Hệ thống sẽ hiển thị thông tin chi tiết của bãi xe gồm có thông tin và danh sách các xe có trong bãi
Nút "Quay lại"	Nhấn (Click)	Hệ thống sẽ đưa người dùng về Trang chủ
Nút "Trả xe"	Nhấn (Click)	Hệ thống sẽ đưa người dùng đến trang hóa đơn
Nút "Xem chi tiết"	Nhấn (Click)	Hệ thống sẽ đưa người dùng đến trang xem thông tin chi tiết của xe

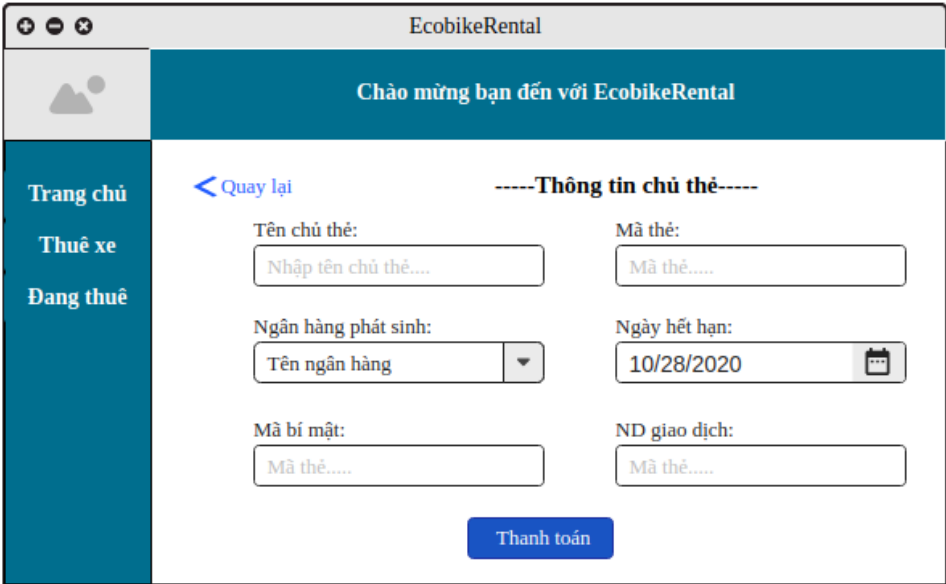
## Đặc tả màn hình Chi tiết xe



Control	Operation	Function
Khu vực hiển thị	Khi nhấn vào nút "Xem chi tiết" ở trang chi tiết bãi xe	Hệ thống sẽ hiển thị thông tin chi tiết của xe trong bãi
Nút "Quay lại"	Nhấn (Click)	Hệ thống sẽ đưa người dùng về Trang chi tiết bãi xe
Nút "Thuê xe"	Nhấn (Click)	Hệ thống sẽ đưa người dùng đến trang nhập thông tin của tài khoản thẻ

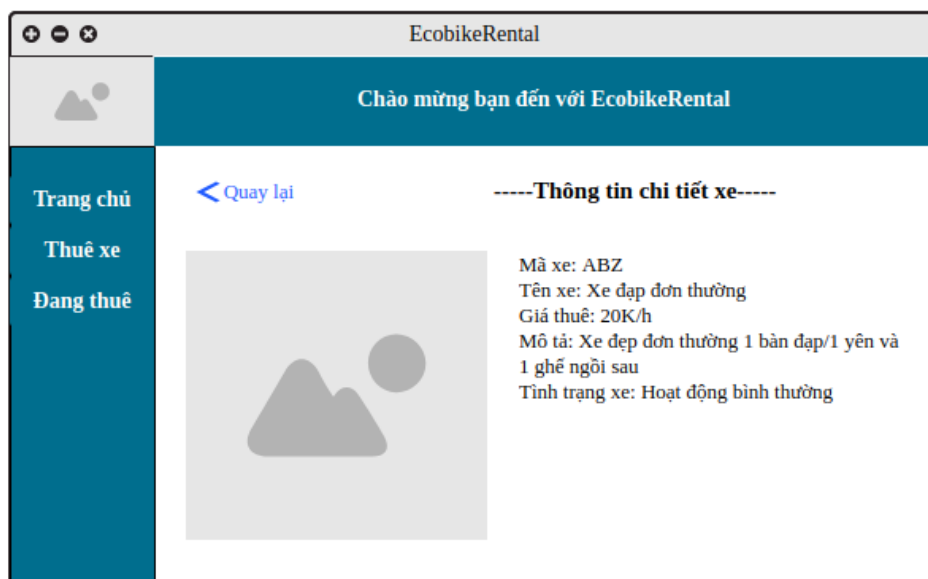
## Đặc tả màn hình Nhập mã QR của xe

		
Control	Operation	Function
Khu vực hiển thị	Khi người dùng nhấn vào nút "Thuê xe" bên sidebar	Hệ thống sẽ hiển thị form để người dùng nhập mã QR của xe
Nút "Quay lại"	Nhấn (Click)	Hệ thống sẽ đưa người dùng về Trang chủ
Nút "Thuê xe"	Nhấn (Click)	Hệ thống sẽ đưa người dùng đến trang chi tiết xe

Đặc tả màn hình Nhập thông tin tài khoản thẻ		
		
Control	Operation	Function
Khu vực hiển thị	Khi người dùng nhấn vào nút	Hệ thống hiển thị form để

	"Thuê xe" ở màn hình Nhập mã QR của xe hoặc nhấn nút "Thuê xe" ở màn hình Chi tiết xe	người dùng nhập thông tin tài khoản của người dùng vào
Nút "Quay lại"	Nhấn (Click)	Hệ thống sẽ đưa người dùng về trang Chi tiết xe
Nút "Thanh toán"	Nhấn (Click)	Hệ thống sẽ giao tiếp với ngân hàng và trừ tiền trong tài khoản của khách hàng

### Đặc tả màn hình Chi tiết xe khi đang thuê



Control	Operation	Function
Khu vực hiển thị	Khi người dùng nhấn vào nút "Thuê xe" ở màn hình Nhập mã QR của xe hoặc nhấn nút "Thuê xe" ở màn hình Chi tiết xe	Hệ thống hiển thị form để người dùng nhập thông tin tài khoản của người dùng vào
Nút "Quay lại"	Nhấn (Click)	Hệ thống sẽ đưa người dùng về trang Chi tiết xe



### Đặc tả màn hình Hóa đơn

EcobikeRental

Chào mừng bạn đến với EcobikeRental

[Trang chủ](#)  
[Thuê xe](#)  
[Đang thuê](#)

[< Quay lại](#) -----Thông tin hóa đơn-----

Thông tin hóa đơn	
Tài khoản	Trinh123
Mã số xe	ABZ
Tiền cọc	20.000
Thời gian thuê	11:11
Tiền thuê xe	350.000
Tiền hoàn lại	10.000

[Xác nhận thanh toán](#)

Control	Operation	Function
Khu vực hiển thị	Khi người dùng nhấn nút "Trả xe" trong trang Chi tiết bãi xe	Hệ thống hiển thị hóa đơn của khách hàng
Nút "Quay lại"	Nhấn (Click)	Hệ thống sẽ đưa người dùng về trang Chi tiết bãi xe
Nút "Xác nhận thanh toán"	Nhấn (Click)	Hệ thống sẽ trừ tiền trong tài khoản của khách hàng

### Đặc tả màn hình Thuê xe thành công

EcobikeRental

Chào mừng bạn đến với EcobikeRental

[Trang chủ](#)  
[Thuê xe](#)  
[Đang thuê](#)

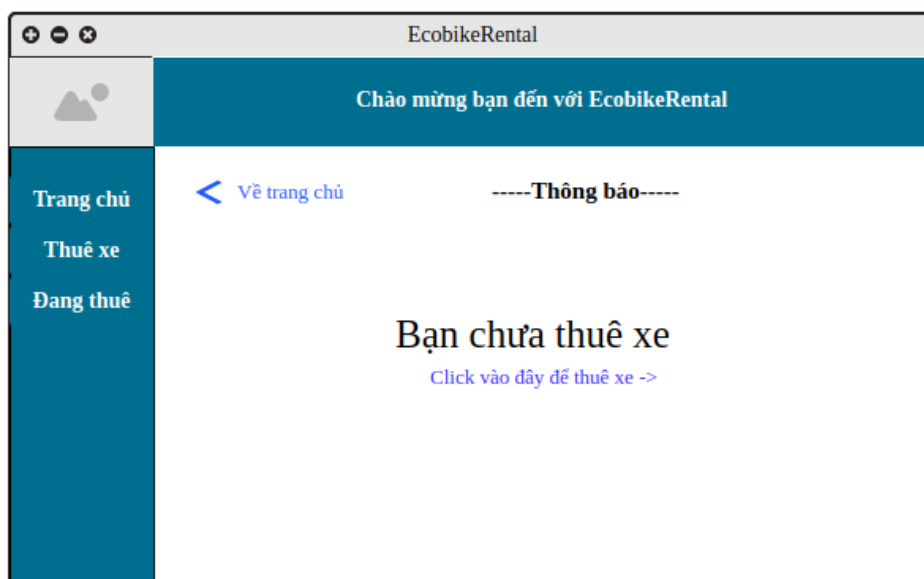
[< Về trang chủ](#) -----Thông báo-----

Thuê xe thành công

Control	Operation	Function
---------	-----------	----------

Khu vực hiển thị	Tự sinh	Hệ thống hiển thị thông báo thuê xe thành công
Nút "Về trang chủ"	Nhấn (Click)	Hệ thống sẽ đưa người dùng về Trang chủ

### Đặc tả màn hình Chưa thuê xe



Control	Operation	Function
Khu vực hiển thị	Khi click vào nút "Đang thuê" bên sidebar	Khi người dùng nhấn vào nút "Đang thuê" nếu người dùng chưa thuê xe thì sẽ hiển thị trang này
Nút "Về trang chủ"	Nhấn (Click)	Hệ thống sẽ đưa người dùng về Trang chủ
Nút "Click vào đây để thuê xe"	Nhấn (Click)	Hệ thống sẽ chuyển đến trang nhập mã QR của xe

Contents
<ul style="list-style-type: none"><li>+ <u>currentUserID : int = 20173410</u></li><li>+ <u>currentRentID : int = 0</u></li><li>+ <u>stationIDSelected : int = 0</u></li><li>+ <u>bikeIDSelected : int = 0</u></li></ul>
<ul style="list-style-type: none"><li>+ Contents()</li><li>+ <u>getSQLServerConnection() : Connection</u></li><li>+ <u>getAllStations() : ArrayList&lt;Station&gt;</u></li><li>+ <u>getCurrentTime() : Time</u></li><li>+ <u>calculateTotalTime(timeStart : Time) : int</u></li><li>+ <u>calculateMoney(price : int, totalTime : int) : int</u></li><li>+ <u>toString(d : long) : String</u></li><li>+ <u>response(res : String) : String</u></li></ul>

**Attribute**

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	currentUserID	int	20173410	Lưu thông tin fix cứng của người dùng
2	currentRentID	int	0	mã thuê xe nếu người đó đã thuê, nếu không thì mặc định là 0 (người đó chưa thuê xe)
3	stationIDSelected	int	0	Lưu mã bãi xe khi người đó chọn vào
4	bikeIDSelected	int	0	Lưu mã xe khi người đó chọn vào

**Operation**

#	Tên	Kiểu dữ liệu trả về	Mô tả
1	getSQLServerConnection	Connection	Kết nối đến cơ sở dữ liệu
2	getAllStations	ArrayList<Station>	Lấy tất cả bãi xe trong cơ sở dữ liệu
3	getCurrentTime	Time	Lấy thời gian hiện tại
4	calculateTotalTime	int	tính tổng thời gian khi bắt đầu thuê xe đến thời điểm hiện tại
5	calculateMoney	int	Tính tổng số tiền cần phải thanh toán
6	toString	String	Chuyển một số thành một chuỗi
7	response	String	trả về các lỗi của interbank

#### 4.2.2 Thiết kế lớp InterbankService

InterbankService	
- SECRETKEY : String = "Bjrap08Wdtw="           - APPCODE : String = "CUgp9eRNgwU="           - VERSION : String = "1.0.1"           - CARDCODE : String = null           - OWNER : String = null           - CVVCODE : String = null           - DATEEXPIRED : String = null	
+ getMD5(input : String) : String           + jsonInforToHash(card : Card, command : String, amount : int, currentTime : String) : String           + jsonResetBalance() : String           + processTransaction(card : Card, command : String, amount : int) : String           + resetBalance(card : Card) : String           + main(args : String[]) : void	

#### Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	SECRETKEY	String	Bjrap08Wdtw=	Mã bảo mật của ứng dụng
2	APPCODE	String	CUgp9eRNgwU=	Mã ứng dụng
3	VERSION	String	1.0.1	Phiên bản của ứng dụng

#### Operation

#	Tên	Kiểu dữ liệu trả về	Mô tả (Mục đích)
1	getMD5	String	Tạo ra chuỗi băm với mã băm MD5
2	processTransaction	String	Thanh toán giao dịch với ngân hàng
3	resetBalance	String	Reset tài khoản về số dư ban đầu

## Parameter

- input - Chuỗi bản rõ cần hash
- card - Tài khoản thẻ của người dùng
- command - lệnh thực hiện giao dịch (pay: thanh toán, refund: hoàn tiền)
- amount - Tổng số tiền cần giao dịch

### 4.2.3 Thiết kế lớp Bike

Bike	
<pre>- id : int - price : int - battery : int - stationID : int - name : int - status : int - type : int - description : int</pre>	
<pre>+ Bike(id : int) + Bike() + Bike(id : int, battery : int, stationID : int, price : int, name : int, status : int, type : int, description : int) + setBikeFromID(bikeID : int) : void + updateBike(status : int, stationID : int) : boolean + createRent(startTime : int, customerID : int) : boolean + getDepositMoney() : int + getBattery() : int + setBattery(battery : int) : void + getStationID() : int + setStationID(stationID : int) : void + getDescription() : int + setDescription(description : int) : void + setBike(b : Bike) : void + getPrice() : int + setPrice(price : int) : void + getName() : int + setName(name : int) : void + getStatus() : int + setStatus(status : int) : void + getType() : int + setType(type : int) : void + getid() : int + setid(id : int) : void</pre>	

## Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	id	int	null	Mã xe
2	price	int	null	Giá cọc của xe
3	battery	int	null	Lượng pin còn lại của xe
4	stationID	int	null	Mã bãi xe
5	name	String	null	Tên xe
6	status	String	null	Trạng thái của xe

7	type	String	null	Kiểu xe
8	description	String	null	Mô tả xe

### Operation

#	Tên	Kiểu dữ liệu trả về	Mô tả (Mục đích)
1	setBikeFromID	void	Lưu bike hiện tại từ id truyền vào lấy được từ trong cơ sở dữ liệu
2	updateBike	boolean	Cập nhật lại xe
3	createRent	boolean	Tạo một rent mới
4	getDepositMoney	int	Lấy ra số tiền cần đặt cọc của xe dựa vào kiểu xe

### Parameter

- bikeID - mã xe truyền vào
- status - trạng thái của xe (renting hay available)
- stationID - mã của bãi xe
- timeStart - Thời gian bắt đầu thuê xe
- customerID - Mã khách hàng

#### 4.2.4 Thiết kế chi tiết lớp Station

Station	
- stationID : int - totalBike : int - available : int - name : int - address : int	
+ setStation(s : Station) : void + Station(stationID : int, name : int, position : int) + setStationFromID(stationID : int) : void + getAllBikes() : int + getBikeByID(bikeID : int) : Bike + Station(stationID : int, totalBike : int, available : int, name : int, address : int) + Station() + getStationID() : int + setStationID(stationID : int) : void + getTotalBike() : int + setTotalBike(totalBike : int) : void + getAvailable() : int + setAvailable(available : int) : void + getAddress() : int + getPosition() : int + setAddress(address : int) : void + getName() : int + setName(name : int) : void	

## Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	stationID	int	null	Mã bãi xe
2	totalBike	int	null	Tổng số xe có trong bãi
3	available	int	null	Số xe còn lại trong bãi
4	name	String	null	Tên bãi xe
5	address	String	null	Địa chỉ của bãi xe

## Operation

#	Tên	Kiểu dữ liệu trả về	Mô tả (mục đích)
1	setStationFromID	void	Lưu station hiện tại từ id truyền vào lấy được từ trong cơ sở dữ liệu
2	getAllBikes	ArrayList<Bike>	Lấy tất cả xe có trong bãi
3	getBikeByID	Bike	Lấy thông tin của xe có id có trong bãi xe



## Parameter

- stationID - mã của bãi xe
- bikeID - mã của xe

### 4.2.5 Thiết kế chi tiết lớp Card

Card
<ul style="list-style-type: none"><li>- cardID : int</li><li>- cardHolderName : int</li><li>- cardNumber : int</li><li>- transactionDescription : int</li><li>- expirationDate : int</li><li>- securityCode : int</li><li>- issuingBank : int</li></ul>
<ul style="list-style-type: none"><li>+ Card()</li><li>+ Card(cardID : int)</li><li>+ Card(cardID : int, cardHolderName : int, cardNumber : int, transactionDescription : int, expirationDate : int, securityCode : int, issuingBank : int)</li><li>+ setCard(c : Card) : void</li><li>+ setCardFromCardNumber(cardNumber : int) : void</li><li>+ getCardID() : int</li><li>+ setCardID(cardID : int) : void</li><li>+ getCardHolderName() : int</li><li>+ setCardHolderName(cardHolderName : int) : void</li><li>+ getCardNumber() : int</li><li>+ setCardNumber(cardNumber : int) : void</li><li>+ getTransactionDescription() : int</li><li>+ setTransactionDescription(transactionDescription : int) : void</li><li>+ getExpirationDate() : int</li><li>+ setExpirationDate(expirationDate : int) : void</li><li>+ getSecurityCode() : int</li><li>+ setSecurityCode(securityCode : int) : void</li><li>+ getIssuingBank() : int</li><li>+ setIssuingBank(issuingBank : int) : void</li></ul>

## Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	cardID	int	null	ID của thẻ
2	cardHolderName	String	null	Tên chủ thẻ

3	cardNumber	String	null	Mã thẻ
4	transactionDescription	String	null	Nội dung giao dịch
5	securityCode	String	null	Mã bảo mật của thẻ
6	issuingBank	String	null	Ngân hàng phát hành thẻ

### Operation

#	Tên	Kiểu dữ liệu trả về	Mô tả (Mục đích)
1	setCardFromCardNumber	void	Lưu thông tin thẻ bởi mã thẻ truyền vào lấy từ cơ sở dữ liệu

### Parameter

- cardNumber - mã thẻ

## 4.2.6 Thiết kế chi tiết lớp Customer

Customer	
- cardNumber : int - customerID : int - rentID : int - customerName : int	
+ Customer() + getRentingBike() : Rent + getCustomerCard() : Card + Customer(customerID : int, cardNumber : int, rentID : int, customerName : int) + Customer(customerID : int, rentID : int, customerName : int) + Customer(id : int, name : int) + getCardNumber() : int + setCardNumber(cardNumber : int) : void + getCustomerID() : int + setCustomerID(customerID : int) : void + getRentID() : int + setRentID(rentID : int) : void + getCustomerName() : int + setCustomerName(customerName : int) : void	

### Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả (Mục đích)
---	-----	--------------	------------------	------------------

1	customerID	int	null	Mã khách hàng
2	customerName	String	null	Tên khách hàng
3	cardNumber	String	null	Mã thẻ của khách hàng
4	rentID	int	null	Mã thuê xe

### Operation

#	Tên	Kiểu dữ liệu trả về	Mô tả (Mục đích)
1	getRentingInfo	Rent	Lấy thông tin thuê xe của khách hàng
2	getCustomerCard	Card	Lấy thông tin tài khoản thẻ của khách hàng

Parameter

Không

#### 4.2.7 Thiết kế chi tiết lớp Rent

Rent
- rentID : int - customerID : int - bikeID : int - totalTimeRent : int - status : int
+ Rent() + Rent(id : int) + setRent(r : Rent) : void + updateRent(timeEnd : int, totalTimeRent : int) : boolean + setRentFromID(rentID : int) : void + createTransaction(code : int, transactionName : int, totalMoney : int) : boolean + Rent(rentID : int, status : int, customerID : int, bikeID : int, timeStart : int, timeEnd : int, totalTimeRent : int) + Rent(rentID : int, status : int, customerID : int, bikeID : int, timeStart : int) + getStatus() : int + setStatus(status : int) : void + getRentID() : int + setRentID(rentID : int) : void + getCustomerID() : int + setCustomerID(customerID : int) : void + getBikeID() : int + setBikeID(bikeID : int) : void + getTimeStart() : int + setTimeStart(timeStart : int) : void + getTimeEnd() : int + setTimeEnd(timeEnd : int) : void + getTotalTimeRent() : int + setTotalTimeRent(totalTimeRent : int) : void

### Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả (Mục đích)
1	rentID	int	null	Mã thuê xe
2	timeStart	Time	null	Thời gian bắt đầu thuê xe
3	timeEnd	Time	null	Thời gian kết thúc thuê xe
4	totalTimeRent	int	null	Tổng thời gian thuê
5	status	String	null	Trạng thái thuê xe (đang thuê hay đã hoàn thành thuê xe)
6	customerID	int	null	Mã khách hàng
7	bikeID	int	null	Mã xe đang thuê

### Operation

#	Tên	Kiểu dữ liệu trả về	Mô tả (Mục đích)
1	updateRent	boolean	Cập nhật lại rent

2	setRentFromID	void	Lưu dữ liệu của rent từ id truyền vào lấy được từ cơ sở dữ liệu
3	createTransaction	boolean	Tạo một giao dịch mới với ngân hàng(là giao dịch đặt cọc tiền hay giao dịch trả xe)

#### Parameter

- timeEnd - Thời gian kết thúc thuê xe
- totalTimeRent - tổng thời gian thuê xe
- rentID - mã thuê xe
- code - mã giao dịch (deposit hay return)
- transactionName - tên giao dịch
- totalMoney - tổng số tiền giao dịch

#### 4.2.8 Thiết kế chi tiết lớp HomeController

HomeController
+ initialize(arg0 : URL, arg1 : ResourceBundle) : void + addEvents() : void + getRentingInfo() : void + updateView() : void + clearView() : void + viewListStation() : void

### Attribute

Không

### Operation

#	Tên	Kiểu dữ liệu trả về	Mô tả (Mục đích)
1	initialize	void	Khởi tạo màn hình home
2	addEvents	void	Thêm các sự kiện cho các button, label,...
3	getRentingInfo	void	Kiểm tra và lấy thông tin thuê xe của khách hàng nếu khách hàng đã thuê xe trước đó
4	updateView	void	Nếu khách hàng có thuê xe trước đó thì cập nhật thông tin thuê đó lên màn hình home
5	clearView	void	Nếu khách hàng chưa thuê xe thì màn hình home sẽ không hiển thị thông tin gì
6	viewListStation	void	Gọi đến và hiển thị đến màn hình danh sách bãi xe

#### 4.2.9 Thiết kế chi tiết lớp ListStationController

ListStationController
+ initialize(arg0 : URL, arg1 : ResourceBundle) : void + addEvents() : void + addControl() : void + getAllStation() : void + viewStation() : void + backToPrevious() : void

#### Attribute

Không

#### Operation

#	Tên	Kiểu dữ liệu trả về	Mô tả (Mục đích)
1	initialize	void	Khởi tạo màn hình danh sách bãi xe
2	addEvents	void	Thêm các sự kiện cho các button, label,...
3	addControl	void	
4	getAllStation	void	Lấy tất cả bãi xe có trong cơ sở dữ liệu và hiển thị lên màn hình
5	viewStation	void	Gọi và hiển thị màn hình bãi xe chi tiết
6	backToPrevious	void	Hàm xử lý việc nhấn nút quay lại trên màn hình

#### 4.2.10 Thiết kế chi tiết lớp PaymentFormController

PaymentForm Controller
+ initialize(arg0 : URL, arg1 : ResourceBundle) : void + addEvents() : void + submitPaymentForm() : void + rentBike(depositMoney : int) : void + checkBlankField() : boolean + showMessage(mess : String) : void + clearTextField() : void + setupTextField() : void

### Attribute

Không

### Operation

#	Tên	Kiểu dữ liệu trả về	Mô tả (Mục đích)
1	initialize	void	Hàm khởi tạo màn hình form thanh toán
2	addEvents	void	Hàm thêm sự kiện cho các button, label,...
3	submitPaymentForm	void	Xử lý sự kiện khi click vào nút thanh toán
4	rentBike	void	xử lý các logic liên quan đến thuê xe
5	checkBlankField	boolean	Kiểm tra xem các trường bên màn hình form đã được nhập hết theo yêu cầu chưa
6	showMessage	void	Hiển thị thông báo
7	clearTextField	void	Làm rỗng các trường trong form thanh toán
8	setupTextField	void	Cập nhật thông tin thẻ của khách hàng nếu thẻ đó đã tồn tại

#### 4.2.11 Thiết kế chi tiết lớp ReturnBikeController



ReturnBikeController
~ totalTimeRent : int = 0 ~ totalMoneyRent : int = 0 ~ depositMoney : int = 0
+ initialize(arg0 : URL, arg1 : ResourceBundle) : void + addEvents() : void + getRentingInfo() : void + updateView() : void + submitReturnBike() : void + returnBike(code : String, totalMoney : int) : void + createTransaction(totalTimeRent : int, totalMoney : int, rentID : int) : void + updateRentingBike() : void + showMessage(mess : String) : void

### Attribute

#	Tên	Kiểu dữ liệu	Giá trị mặc định	Mô tả
1	totalTimeRent	int	0	Lưu tổng thời gian thuê xe
2	totalMoneyRent	int	0	Lưu tổng tiền thuê xe
3	depositMoney	int	0	Lưu tiền đặt cọc

### Operation

#	Tên	Kiểu dữ liệu trả về	Mô tả (Mục đích)
1	initialize	void	Hàm khởi tạo màn hình trả xe
2	addEvents	void	Thêm sự kiện cho các button, label,...
3	getRentingInfo	void	Lấy thông tin của người thuê xe, thông tin thuê xe, thông tin xe thuê
4	updateView	void	Nếu người đó đang thuê xe thì cập nhật lại màn hình trả xe với thông tin được lấy trước đó
5	submitReturnBike	void	Xử lý sự kiện khi click vào nút trả xe
6	returnBike	void	Xử lý logic của việc trả xe
7	createTransaction	void	Tạo giao dịch
8	updateRentingBike	void	Cập nhật lại trạng thái thuê xe của người dùng

9	showMessage	void	Hiển thị thông báo cho người dùng
---	-------------	------	-----------------------------------

#### 4.2.12 Thiết kế chi tiết lớp ViewBikeController

<b>ViewBikeController</b>
+ initialize(arg0 : URL, arg1 : ResourceBundle) : void + addEvents() : void + showBikeInfo() : void + checkRentAvailable() : boolean + showMessage(mess : String) : void + showPaymentForm() : void

#### Attribute

Không

#### Operation

#	Tên	Kiểu dữ liệu trả về	Mô tả (Mục đích)
1	initialize	void	Hàm khởi tạo màn hình xem tiết xe
2	addEvents	void	Hàm thêm sự kiện cho các button, label,...
3	showBikeInfo	void	Ghi thông tin chi tiết của xe đó và hiển thị lên màn hình
4	checkRentAvailable	boolean	Kiểm tra xem xe đó đã có người thuê hay chưa và kiểm tra xem bạn đã thuê xe nào hay chưa
5	showMessage	void	Hiển thị thông báo cho người dùng
6	showPaymentForm	void	Gọi và hiển thị màn hình form thanh toán

#### 4.2.13 Thiết kế chi tiết lớp ViewStationController

ViewStationController
+ initialize(arg0 : URL, arg1 : ResourceBundle) : void + addEvents() : void + addControl() : void + getStationInfo() : void + getAllBikes() : void + showBikeInfo() : void + checkBikeRenting() : void + showReturnBike() : void + searchBike() : void + showMessage(mess : String) : void

## Attribute

Không

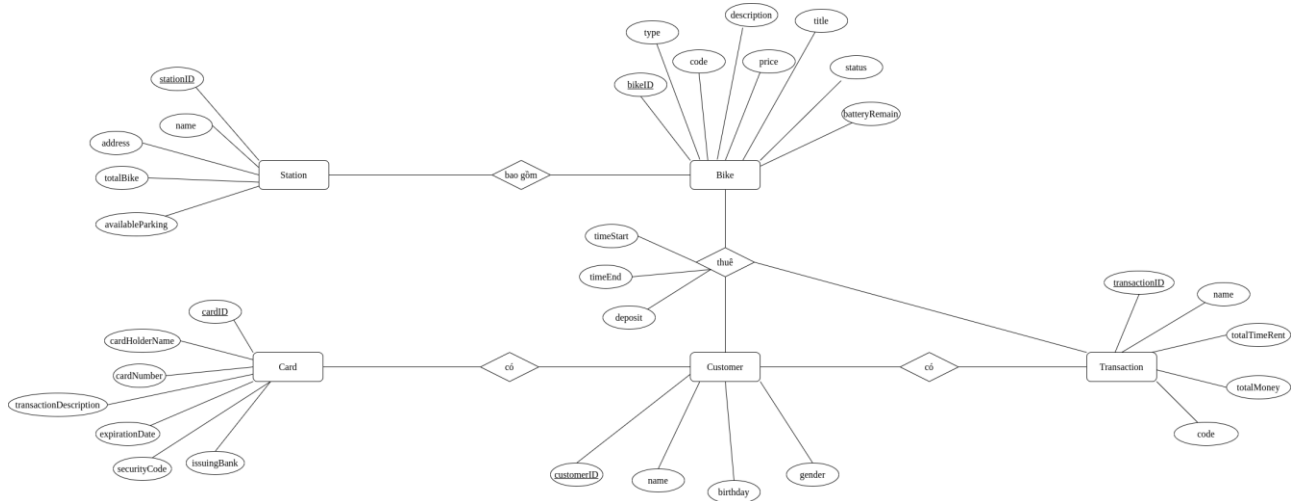
## Operation

#	Tên	Kiểu dữ liệu trả về	Mô tả (Mục đích)
1	initialize	void	Hàm khởi tạo màn hình bãi xe chi tiết
2	addEvents	void	Hàm thêm các sự kiện cho các button, label,...
3	addControl	void	
4	getStationInfo	void	Hiển thị thông tin chi tiết của bãi xe
5	getAllBikes	void	Hiển thị danh sách của tất cả các xe có trong bãi xe
6	showBikeInfo	void	Gọi và hiển thị màn hình thông tin chi tiết của xe
7	checkBikeRenting	void	Kiểm tra xem bạn đã thuê xe chưa, nếu đã thuê xe rồi thì gọi đến màn hình trả xe còn nếu chưa thuê xe thì sẽ hiển thị thông báo chưa thuê xe
8	showReturnBike	void	Gọi và hiển thị màn hình trả xe
9	searchBike	void	Tìm kiếm xe theo mã xe, nếu tồn tại xe thì sẽ hiển thị thông tin chi tiết của xe đó, nếu không tồn tại thì sẽ thông báo cho người dùng
10	showMessage	void	Hiển thị thông báo cho người dùng

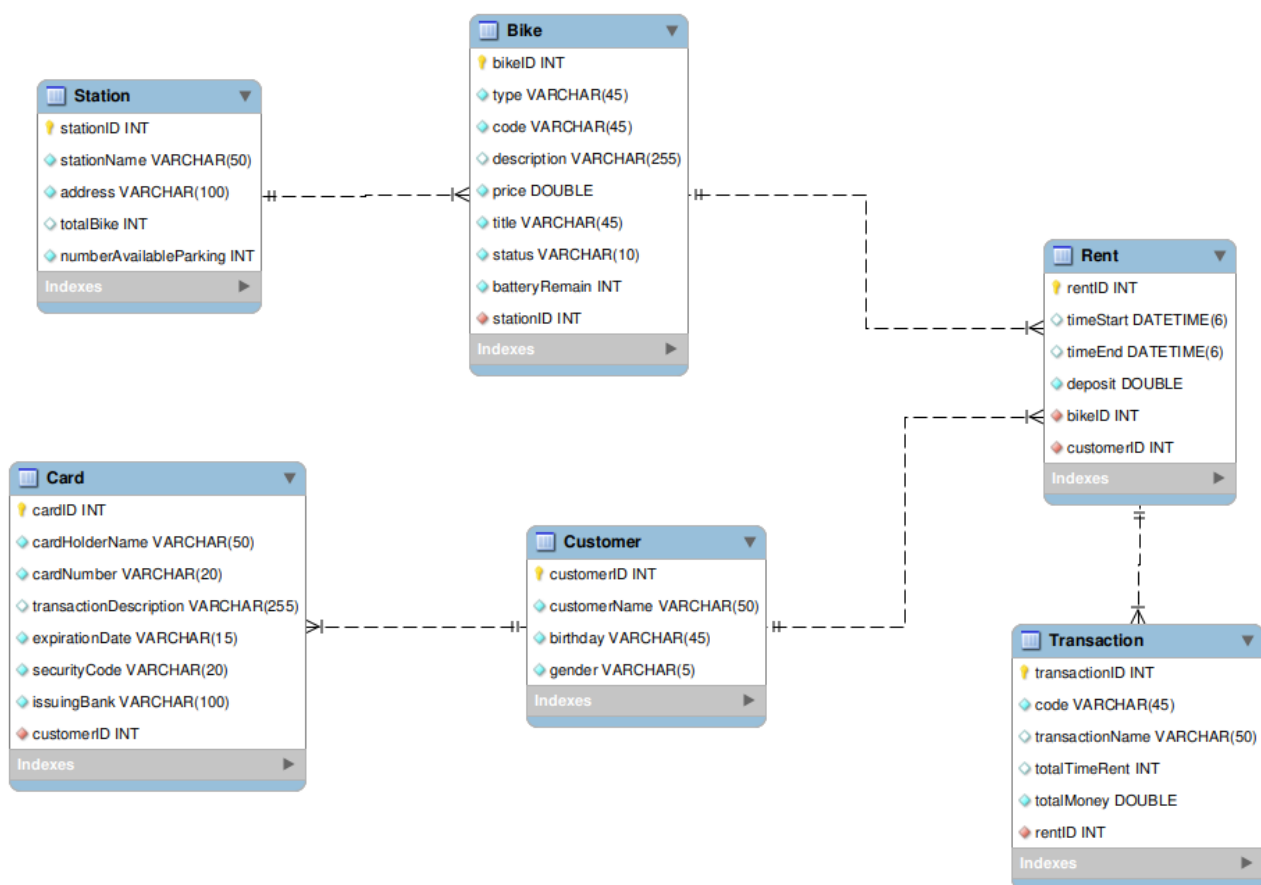
## 5 Thiết kế mô hình dữ liệu

### 5.1 Mô hình dữ liệu mức khái niệm

- Biểu đồ thực thể liên kết:



### 5.2 Mô hình dữ liệu mức logic



## 5.3 Thiết kế chi tiết

### 5.3.1 Bảng Station(Bãi xe)

Lưu thông của bãi gửi xe

Tên cột	Kiểu dữ liệu	Khóa chính	Khóa ngoại	Duy nhất	Ràng buộc	Mô tả
stationID	integer	Yes		Yes	NOT NULL	Khóa chính của bảng tự tăng
stationName	varchar(50)			Yes	NOT NULL	Tên của bến xe
address	varchar(100)				NOT NULL	Địa chỉ của bến xe
numberTotalBike	integer				NOT NULL	Tổng số xe trong bãi có thể chứa
availableParking	integer				NOT NULL	Số xe còn lại có thể nhận

### 5.3.2 Bảng Bike (Xe)

Lưu thông tin của xe

Tên cột	Kiểu dữ liệu	Khóa chính	Khóa ngoại	Duy nhất	Ràng buộc	Mô tả
bikeID	integer	Yes		Yes	NOT NULL	Khóa chính của bảng tự tăng
type	varchar(50)					Kiểu xe (xe đạp thường hay xe đạp điện,...)
code	varchar(45)			Yes	NOT NULL	Mã của xe
description	varchar(255)					Mô tả của xe
price	double					Giá tiền của xe
title	varchar(45)					Tên của xe
status	varchar(10)					Thẻ hiện xe đang được thuê hay chưa
batteryRemain	integer					Lượng pin còn lại của xe đối với xe điện
stationID	integer		Yes	Yes		Khóa ngoại liên kết với bảng Station

### 5.3.3 Bảng Customer (Khách hàng)

Lưu thông tin của khách hàng

Tên cột	Kiểu dữ liệu	Khóa chính	Khóa ngoại	Duy nhất	Ràng buộc	Mô tả
customerID	integer	Yes		Yes	NOT NULL	Khóa chính của bảng tự tăng
customerName	varchar(50)				NOT NULL	Tên khách hàng

birthday	varchar(100)					Ngày tháng năm sinh của khách hàng
gender	integer					Giới tính của khách hàng

#### 5.3.4 Bảng Card (Tài khoản thẻ)

Lưu thông tin tài khoản thẻ của khách hàng

Tên cột	Kiểu dữ liệu	Khóa chính	Khóa ngoại	Duy nhất	Ràng buộc	Mô tả
cardID	integer	Yes		Yes	NOT NULL	Khóa chính của bảng tự tăng
cardHolderName	varchar(50)				NOT NULL	Tên chủ thẻ
cardNumber	varchar(100)			Yes	NOT NULL	Số thẻ
transactionDescription	varchar(255)					Nội dung giao dịch
expirationDate	varchar(15)				NOT NULL	Ngày hết hạn
securityCode	varchar(20)				NOT NULL	Mã bí mật
issuingBank	varchar(100)				NOT NULL	Ngân hàng phát hành
customerID	integer		Yes	Yes	NOT NULL	Khóa ngoại liên kết với bảng Customer

#### 5.3.5 Bảng Transaction (Thông tin giao dịch)

Lưu thông tin giao dịch của khách hàng với hệ thống

Tên cột	Kiểu dữ liệu	Khóa chính	Khóa ngoại	Duy nhất	Ràng buộc	Mô tả
transactionID	integer	Yes		Yes	NOT NULL	Khóa chính của bảng tự tăng
code	varchar(50)				NOT NULL	Mã loại giao dịch
transactionName	varchar(100)				NOT NULL	Tên giao dịch

totalTimeRent	float					Tổng thời gian thuê xe tính theo giờ
totalMoney	double				NOT NULL	Tổng số tiền phải trả
rentID	varchar(20)		Yes	Yes	NOT NULL	Khóa ngoại liên kết với bảng Rent

### 5.3.6 Bảng Rent (Thuê xe)

Lưu thông tin thuê xe

Tên cột	Kiểu dữ liệu	Khóa chính	Khóa ngoại	Duy nhất	Ràng buộc	Mô tả
rentID	integer	Yes		Yes	NOT NULL	Khóa chính của bảng tự tăng
timeStart	datetime(6)				NOT NULL	Thời gian bắt đầu thuê
timeEnd	datetime(6)				NOT NULL	Thời gian kết thúc thuê
deposit	float				NOT NULL	Số tiền đặt cọc
bikeID	double		Yes	Yes	NOT NULL	Khóa ngoại liên kết với bảng Bike
customerID	varchar(20)		Yes	Yes	NOT NULL	Khóa ngoại liên kết với bảng Customer



## 6 Xem xét thiết kế

### 6.1 Mục tiêu và Nguyên tắc (Goals and Guidelines)

- Mục tiêu:
  - o Phần mềm hoạt động tốt trên mọi nền tảng
  - o Các chức năng hoạt động bình thường
  - o Người dùng thuê được xe và trả được xe sau thời gian thuê xe
  - o Người dùng thanh toán thành công
  - o Các trường hợp ngoại lệ được thông báo tới người dùng kịp thời
  - o Hạn chế tối đa các trường hợp rủi ro
- Mục tiêu:
  - o Trong thời gian tới sẽ phát triển thêm chức năng định vị bãi gửi xe trên bản đồ
  - o Dữ liệu sát thực tế hơn

### 6.2 Chiến lược kiến trúc (Architectural Strategies)

a. Ngôn ngữ lập trình: java

Các thư viện đã sử dụng trong chương trình

- javafx : thư viện dùng để làm giao diện
- jdbc: dùng để kết nối đến cơ sở dữ liệu
- junit test: dùng để xây dựng test case

b. Cơ sở dữ liệu

- Hệ quản trị cơ sở dữ liệu SQL Server

### 6.3 Coupling và Cohesion

#### Coupling

- Content coupling: Không có
- Common Coupling: Không có
- Bản thiết kế đang ở mức control coupling:

```

/**
 * Nhiệm vụ: lấy tiền cọc của xe dựa vào type
 * @return tiền cọc
 */
public int getDepositMoney() {
    switch (this.type) {
        case "1": {
            return 400000;
        }
        case "2": {
            return 700000;
        }
        case "3": {
            return 550000;
        }
        default:
            return 400000;
    }
}

```

Nhìn vào hình trên ta thấy: Lớp bike đang bị phụ thuộc vào trường type để lấy ra số tiền đặt cọc của xe

Giải pháp: Xây dựng một lớp abstract bike và các lớp bike có kiểu khác nhau (xe đạp thường, xe đạp điện,...) kế thừa lớp abstract bike đó

- Stamp Coupling: Không có

## Cohesion

- Logical cohesion

Contants
<u>+ currentUserID : int = 20173410</u> <u>+ currentRentID : int = 0</u> <u>+ stationIDSelected : int = 0</u> <u>+ bikeIDSelected : int = 0</u>
<u>+ Contants()</u> <u>+ getSQLServerConnection() : Connection</u> <u>+ getAllStations() : ArrayList&lt;Station&gt;</u> <u>+ getCurrentTime() : Time</u> <u>+ calculateTotalTime(timeStart : Time) : int</u> <u>+ calculateMoney(price : int, totalTime : int) : int</u> <u>+ toString(d : long) : String</u> <u>+ response(res : String) : String</u>

- Ở lớp constants: phương thức getSQLServerConnection không liên quan gì về mặt logic đối với các phương thức trong lớp này

Giải pháp: Nên tách phương thức này và đặt vào trong model

- Sequential cohesion: Đầu ra của phương thức này là đầu vào của phương thức kia

```
/**
 * Nhiệm vụ: Kiểm tra xem tất cả các trường trong form đã được nhập chưa
 * @return true hoặc false
 */
public boolean checkBlankField() {
    if (txtCardHolderName.getText().isEmpty()) return false;
    if (txtCardNumber.getText().isEmpty()) return false;
    if (txtCardBank.getText().isEmpty()) return false;
    if (txtCardExpirationDate.getText().isEmpty()) return false;
    if (txtCardCVV.getText().isEmpty()) return false;
    return true;
}

/**
 * Nhiệm vụ: xử lý khi submit form
 */
public void submitPaymentForm() {
    if (checkBlankField()) {
        card.setCardHolderName(txtCardHolderName.getText());
        card.setCardNumber(txtCardNumber.getText());
        card.setExpirationDate(txtCardExpirationDate.getText());
        card.setSecurityCode(txtCardCVV.getText());
        card.setIssuingBank(txtCardBank.getText());
        current = Constants.getCurrentTime();
        rentBike(bike.getDepositMoney());
    } else {
        showMessage("Hãy nhập đầy đủ thông tin !");
    }
}
```

## 6.4 Nguyên tắc thiết kế (Design Principles)

- Nguyên lý S (Single Responsibility)

- Cũng như bao controller khác, các controller đang làm 2 nhiệm vụ đó là render ra màn hình và thu thập dữ liệu để điều hướng đến model

ViewStationController
<pre> + initialize(arg0 : URL, arg1 : ResourceBundle) : void + addEvents() : void + addControl() : void + getStationInfo() : void + getAllBikes() : void + showBikeInfo() : void + checkBikeRenting() : void + showReturnBike() : void + searchBike() : void + showMessage(mess : String) : void </pre>

- Giải pháp: Nên tách việc xử lý render màn hình ra thành 1 lớp riêng, controller chỉ nên làm nhiệm vụ điều hướng

- Nguyên lý O (Open/Close)

Bike
<pre> - id : int - price : int - battery : int - stationID : int - name : int - status : int - type : int - description : int  + Bike(id : int) + Bike() + Bike(id : int, battery : int, stationID : int, price : int, name : int, status : int, type : int, description : int) + setBikeFromID(bikeID : int) : void + updateBike(status : int, stationID : int) : boolean + createRentTimeStart : int, customerID : int) : boolean + getDepositMoney() : int + getBattery() : int + setBattery(battery : int) : void + getStationID() : int + setStationID(stationID : int) : void + getDescription() : int + setDescription(description : int) : void + setBike(b : Bike) : void + getPrice() : int + setPrice(price : int) : void + getName() : int + setName(name : int) : void + getStatus() : int + setStatus(status : int) : void + getType() : int + setType(type : int) : void + getId() : int + setId(id : int) : void </pre>

- Nhìn vào lớp bike ta thấy điều này là vi phạm nguyên lý O : việc mở rộng sẽ rất khó khăn khi chúng ta mở rộng có nhiều loại xe hơn.

Giải pháp: nên tách lớp bike thành 1 lớp abstract có những thuộc tính và phương thức đặc trưng xong sau đó có các lớp bike có kiểu khác nhau (xe đạp thường, xe đạp điện,...) kế thừa lớp abstract đó

- Nguyên lý L (Liskov Substitution)

- Trong project của chúng em không có mối liên hệ cha con, mỗi lớp đều độc lập với nhau

- **Nguyên lý I (Interface Segregation)**

- Trong project của chúng em không sử dụng Interface

- **Nguyên lý D (Dependency Inversion)**

## **6.5 Mẫu thiết kế (*Design patterns*)**

- Trong project không sử dụng mẫu thiết kế nào