

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)

HỆ THỐNG THUÊ XE ĐIỆN TỬ

ECOBIKE

Môn: Thiết kế và xây dựng phần mềm

Nhóm 8

Giáo viên hướng dẫn: TS. Nguyễn Thị Thu Trang

Danh sách sinh viên: 1. Lê Thế Nam - 20173265

2. Vũ Trung Nghĩa - 20173284

3. Nguyễn Thanh Nhã - 20170103

Hà Nội, ngày 18 tháng 12 năm 2020

MỤC LỤC

MỤC LỤC	1
1 Giới thiệu.....	4
1.1 Mục đích	4
1.2 Phạm vi.....	4
1.3 Từ điển thuật ngữ:.....	4
1.4 Tham khảo.....	5
2 Thiết kế kiến trúc	5
2.1 Lựa chọn kiến trúc phần mềm.....	5
2.2 Thiết kế tổng quan.....	7
2.3 Biểu đồ tương tác	8
2.3.1 Biểu đồ tương tác cho UC001 – Xem thông tin xe trong bãi	8
2.3.2 Biểu đồ tương tác cho UC002 – Thuê xe:	9
2.3.3 Biểu đồ tương tác cho UC03 – Trả xe:.....	12
2.4 Sơ đồ lớp phân tích.....	14
2.4.1 Use case 001 – Xem thông tin xe trong bãi	14
2.4.2 Use case 002 – Thuê Xe.....	15
2.4.3 Use case 003 - Trả xe	15
2.4.4 Sơ đồ lớp phân tích gộp	16
3 Thiết kế giao diện.....	16
3.1 Chuẩn hóa cấu hình màn hình	16
3.2 Giao diện người dùng	18
3.2.1 Biểu đồ dịch chuyển màn hình.....	18
3.2.2 Thiết kế giao diện.....	19
4 Thiết kế lớp	30
4.1 Biểu đồ lớp thiết kế	30
4.1.1 Biểu đồ lớp thiết kế.....	30
4.1.2 Thiết kế chi tiết gói “entity”	31
4.1.3 Thiết kế chi tiết gói “presentationlayer”	31
4.1.4 Thiết kế chi tiết gói controller.....	32
4.1.5 Thiết kế chi tiết subsystem “interbanksubsystem”	32
4.1.6 Thiết kế chi tiết subsystem “barcodeconvertersubsystem”	33
4.1.7 Thiết kế chi tiết gói “dataaccesslayer” và gói “connection”	33
4.2 Thiết kế lớp chi tiết.....	34
4.2.1 Thiết kế lớp “ReturnBikeController”	34
4.2.2 Thiết kế lớp “InitializeController”	34

4.2.3	Thiết kế lớp “MainScreen”	35
4.2.4	Thiết kế lớp “ReturnBikeScreen”	36
4.2.5	Thiết kế lớp “RentBikeTransactionScreen”	37
4.2.6	Thiết kế lớp “DockScreen”	37
4.2.7	Thiết kế lớp “BikeDetailScreen”.....	38
4.2.8	Thiết kế lớp “BikeDAO”.....	38
4.2.9	Thiết kế lớp “DockDAO”.....	39
4.2.10	Thiết kế lớp “PaymentTransactionDAO”	39
4.2.11	Thiết kế lớp “RentBikeTransactionDAO”.....	39
4.2.12	Thiết kế lớp “DBConnection”	40
4.2.13	Thiết kế lớp “RentBikeScreen”.....	41
4.2.14	Thiết kế lớp “RentBikeInfoScreen”	41
4.2.15	Thiết kế lớp “RentBikeController”	42
4.2.16	Thiết kế lớp “UserInfo”	43
4.2.17	Thiết kế lớp “RentBikeTransaction”	43
4.2.18	Thiết kế lớp “Dock”	44
4.2.19	Thiết kế lớp “InterbankTransaction”	45
4.2.20	Thiết kế lớp “Card”.....	45
4.2.21	Thiết kế lớp “PaymentTransaction”.....	46
4.2.22	Thiết kế lớp “Bike”	47
4.2.23	Thiết kế lớp “IInterbank”	48
4.2.24	Thiết kế lớp “InterbankSubsysController”	48
4.2.25	Thiết kế lớp “InterbankBoundary”	49
4.2.26	Thiết kế lớp “ConvertToTransaction”	49
4.2.27	Thiết kế lớp “HttpConnector” (của Interbank Subsystem).....	50
4.2.28	Thiết kế lớp “IbarcodeConverter”	50
4.2.29	Thiết kế lớp “BarcodeConverterController”	51
4.2.30	Thiết kế lớp “BarcodeConverterBoundary”	51
4.2.31	Thiết kế lớp “HttpConnector” (của Barcode Converter Subsystem)	51
5	Thiết kế mô hình dữ liệu	52
5.1	Mô hình dữ liệu mức khái niệm.....	52
5.2	Mô hình dữ liệu mức logic	53
5.3	Thiết kế chi tiết.....	53
5.3.1	Thành phần ‘Dock’:	53
5.3.2	Thành phần ‘bike’:.....	54
5.3.3	Thành phần ‘user’:	54

5.3.4	Thành phần ‘paymenttransaction’:.....	54
5.3.5	Thành phần ‘rentbiketransaction’:	55
5.3.6	Thành phần ‘card’:	55
6	Đánh giá bản thiết kế	55
6.1	Mục tiêu và định hướng.....	55
6.2	Chiến lược thiết kế	56
6.3	Coupling and Cohesion.....	56
6.3.1	Phân tích tính Coupling và Cohesion cho gói “controller”	56
6.3.2	Phân tích Coupling giữa các gói trong toàn hệ thống	59

DANH SÁCH HÌNH ẢNH

Hình 1	Kiến trúc thiết kế 3 layers	5
Hình 2	Thiết kế tổng quan của hệ thống	7
Hình 3	Biểu đồ tương tác UC001	8
Hình 4	Biểu đồ tương tác UC002 – Luồng sự kiện chính.....	9
Hình 5	Biểu đồ tương tác UC002 – Luồng sự kiện thay thế 1	10
Hình 6	Biểu đồ tương tác UC002 – Luồng sự kiện thay thế 2	11
Hình 7	Biểu đồ tương tác UC002 – Luồng sự kiện thay thế 3	12
Hình 8	Biểu đồ tương tác UC003 – Luồng sự kiện chính.....	13
Hình 9	Biểu đồ tương tác UC003 – Luồng sự kiện thay thế 1	13
Hình 10	Biểu đồ tương tác UC003 – Luồng sự kiện thay thế 2	14
Hình 11	Sơ đồ lớp phân tích UC001.....	14
Hình 12	Sơ đồ lớp phân tích UC002	15
Hình 13	Sơ đồ lớp phân tích UC003.....	15
Hình 14	Sơ đồ lớp phân tích gộp	16
Hình 15	Sơ đồ dịch chuyển màn hình.....	18
Hình 16	Biểu đồ lớp thiết kế.....	30
Hình 17	Thiết kế chi tiết gói “entity”.....	31
Hình 18	Thiết kế chi tiết gói “presentationlayer”	31
Hình 19	Thiết kế chi tiết gói “controller”	32
Hình 20	Thiết kế chi tiết gói “interbanksubsystem”	32
Hình 21	Thiết kế chi tiết gói “barcodeconvertersubsystem”	33
Hình 22	Thiết kế chi tiết gói “dataaccesslayer” và gói “connection”	33

1 Giới thiệu

1.1 Mục đích

Tài liệu này đưa ra mô tả chi tiết về thiết kế kiến trúc, thiết kế giao diện, thiết kế cơ sở dữ liệu cũng như thiết kế chi tiết lớp cho từng chức năng và các thành phần trong hệ thống. Tài liệu giúp cho người lập trình viên, testers, maintainers, ... có cái nhìn cụ thể chi tiết về phần mềm để dễ dàng trong quá trình xây dựng.

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

1.2 Phạm vi

Phương pháp thiết kế được sử dụng cho phần mềm đó là thiết kế hướng đối tượng. Hệ thống được nhìn nhận như một bộ các đối tượng. Hệ thống được phân tán, mỗi đối tượng có thông tin và trạng thái của riêng nó. Đối tượng được mô tả thông qua một bộ các thuộc tính xác định trạng thái của nó và các phép toán thực hiện trên đó. Mỗi đối tượng là một khách thể của một lớp mà lớp được xác định bởi thuộc tính và các phép toán của nó. Nó được thừa kế từ một vài lớp đối tượng cấp cao hơn, sao cho định nghĩa nó chỉ cần nêu đủ các khác nhau giữa nó và các lớp cao hơn nó. Các đối tượng liên lạc với nhau chỉ bằng trao đổi các thông báo.

Hệ thống ngoài được liên kết tương tác đối với hệ thống này là Interbank và API Barcode Converter. Interbank phụ trách các thanh toán “pay” và “refund” trong quá trình giao dịch thuê xe của khách hàng được diễn ra. Còn API Barcode Converter phụ trách chuyển đổi mã barcode thành mã xe tương ứng để hệ thống tiền hành kiểm tra và thực hiện nghiệp vụ cần thiết.

1.3 Từ điển thuật ngữ:

STT	Thuật ngữ	Giải thích	Ví dụ	Ghi chú
1	API	API là các phương thức, giao thức kết nối với các thư viện và ứng dụng khác. Nó là viết tắt của Application Programming Interface – giao diện lập trình ứng dụng. API cung cấp khả năng cung cấp khả năng truy xuất đến một tập các hàm hay dùng. Và từ đó có thể trao đổi dữ liệu giữa các ứng dụng	google translate api	API được thiết kế đơn giản và dễ dùng

1.4 Tham khảo

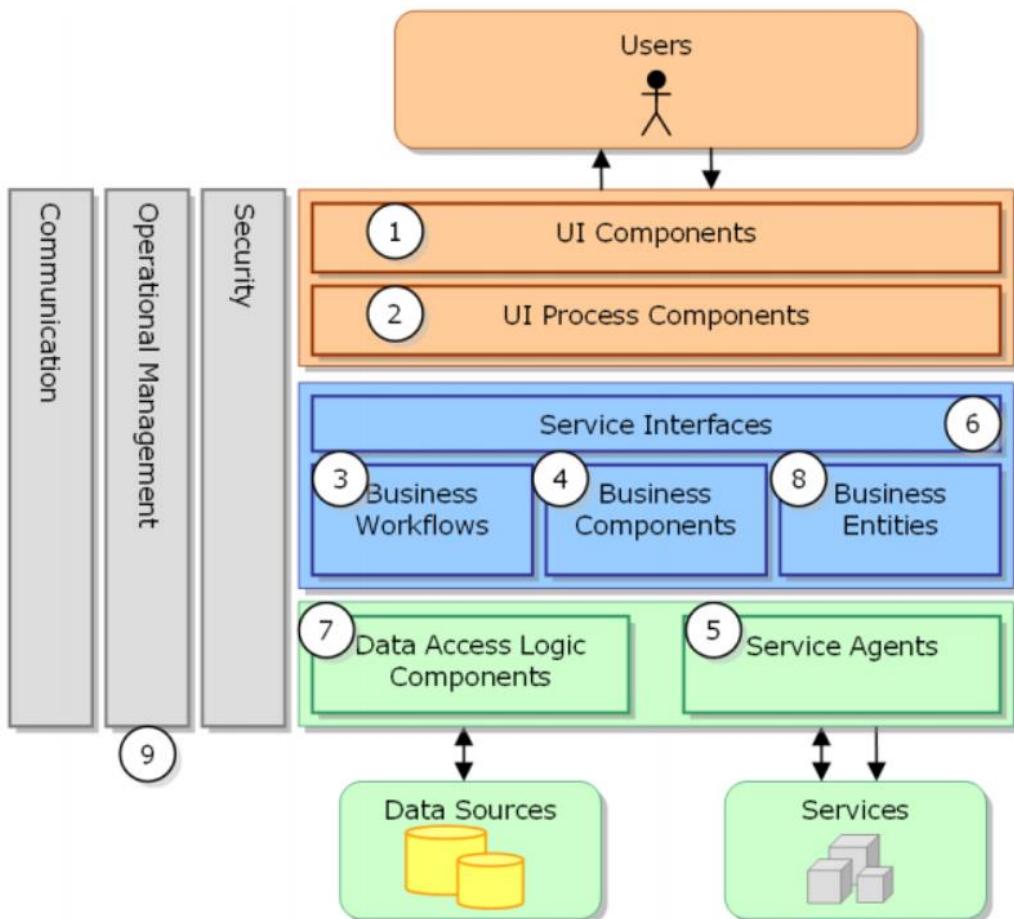
1. Slide môn học Thiết kế và xây dựng phần mềm – giảng viên TS. Nguyễn Thị Thu

Trang

2 Thiết kế kiến trúc

2.1 Lựa chọn kiến trúc phần mềm

Với ứng dụng EcoBike chúng em lựa chọn kiến trúc 3-layers để tiến hành thiết kế và xây dựng phần mềm. Kiến trúc mô hình 3-layers gồm 3 lớp: Presentation Layer, Business Logic Layer, Data Access Layer.



Hình 1 Kiến trúc thiết kế 3 layers

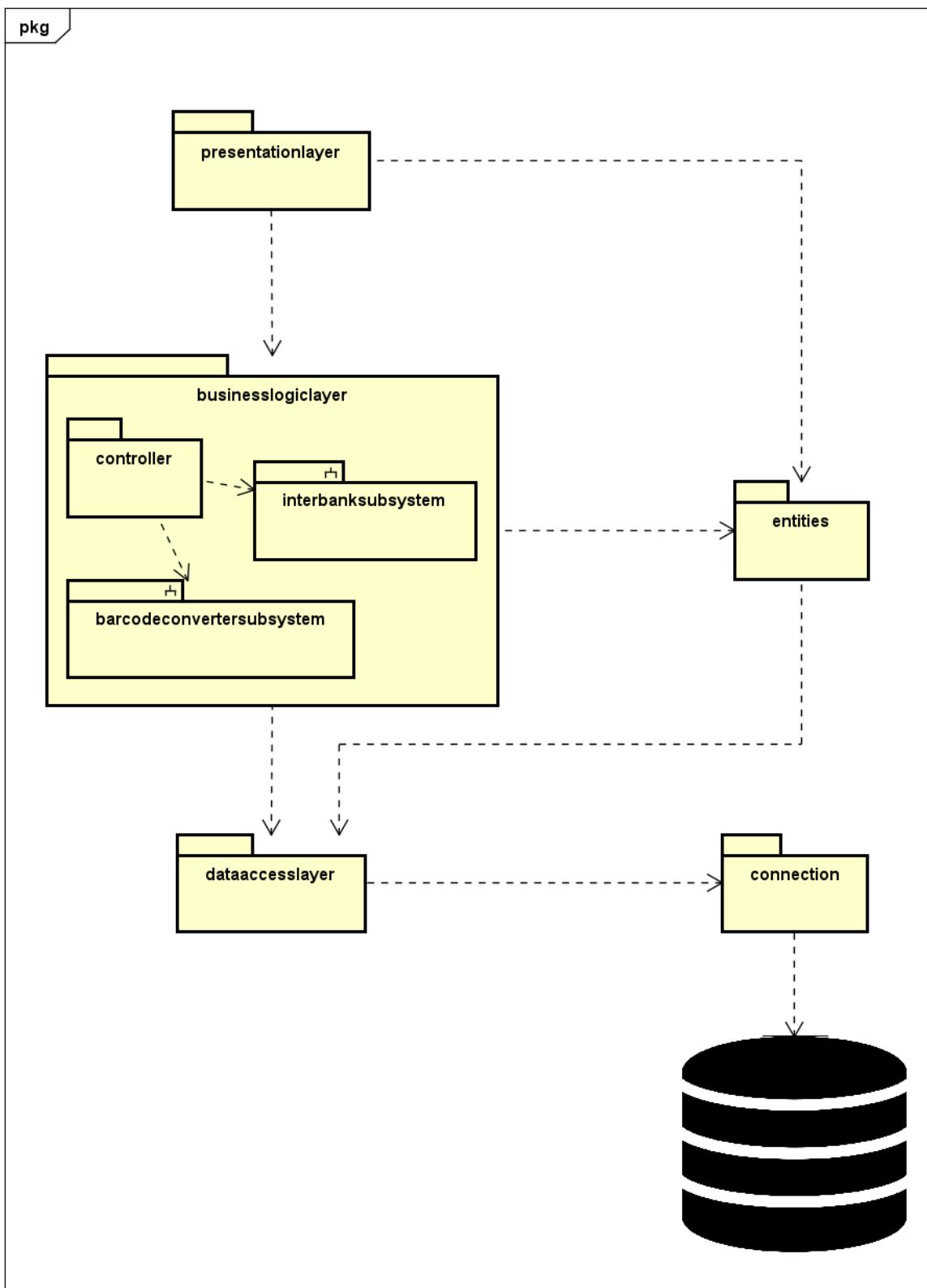
* Chức năng của từng lớp:

- Presentation Layers: Lớp này làm nhiệm vụ giao tiếp với người dùng cuối để thu thập dữ liệu và hiển thị kết quả/dữ liệu thông qua các thành phần trong giao diện người sử dụng.
- Busines Logic Layers: Đây là layer xử lý chính các dữ liệu trước khi được đưa lên hiển thị trên màn hình hoặc xử lý các dữ liệu trước khi chuyển xuống Data Access Layer để lưu dữ liệu xuống cơ sở dữ liệu. Đây cũng là nơi để kiểm tra ràng buộc, các yêu cầu nghiệp vụ, tính toán, xử lý các yêu cầu và lựa chọn kết quả trả về cho Presentation Layers.
- Data Access Layers: Lớp này thực hiện các nghiệp vụ liên quan đến lưu trữ và truy xuất dữ liệu của ứng dụng như đọc, lưu, cập nhật cơ sở dữ liệu.

* Quy trình vận hành của mô hình:

- Đầu tiên, người dùng giao tiếp với lớp presentation layer, sau khi tiếp xúc với giao diện, người dùng gửi đi các yêu cầu cho hệ thống và kèm theo là các thông tin. Các thông tin này sẽ được ở xử lý, kiểm tra, sau đó được gửi kèm theo yêu cầu tới lớp thứ 2 là Business Logic Layer.
- Tại lớp Business Logic Layer, các thông tin sẽ được xử lý theo yêu cầu của người dùng. Kết quả sẽ được trả về và hiển thị cho người dùng. Nếu dữ liệu cần thiết phải truy cập database thì dữ liệu sẽ được chuyển xuống lớp thứ 3 là Data Access Layer.
- Tại lớp Data Access Layer, hệ thống sẽ thực hiện các thao tác với database có thể là lưu trữ, lấy thông tin,...và chuyển lại lên tầng Business Logic Layer. Business Logic Layer kiểm tra và gửi nó lên Presentation Layer để hiển thị cho người dùng.

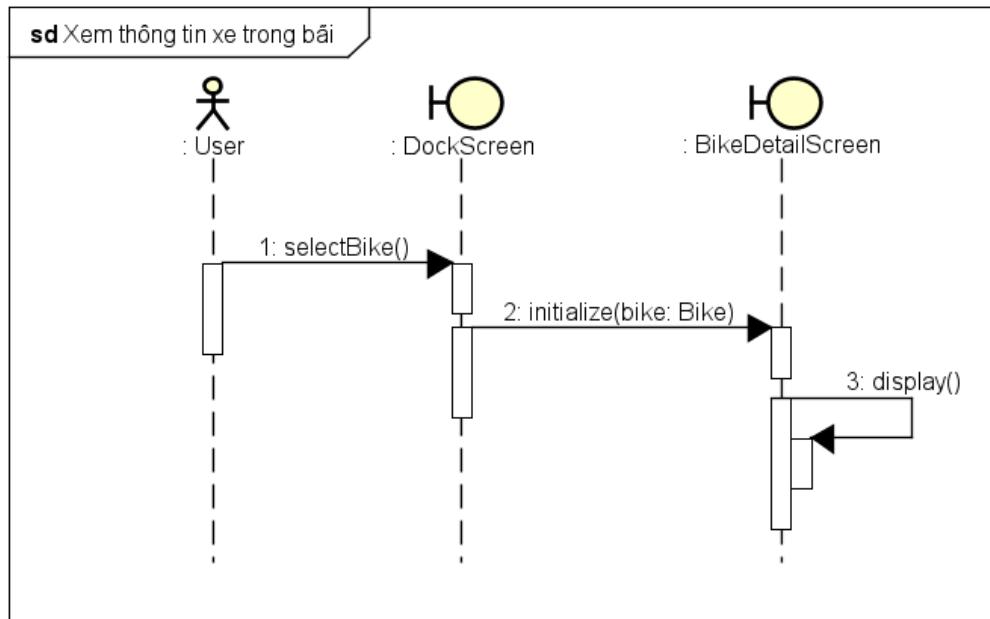
2.2 Thiết kế tổng quan



Hình 2 Thiết kế tổng quan của hệ thống

2.3 Biểu đồ tương tác

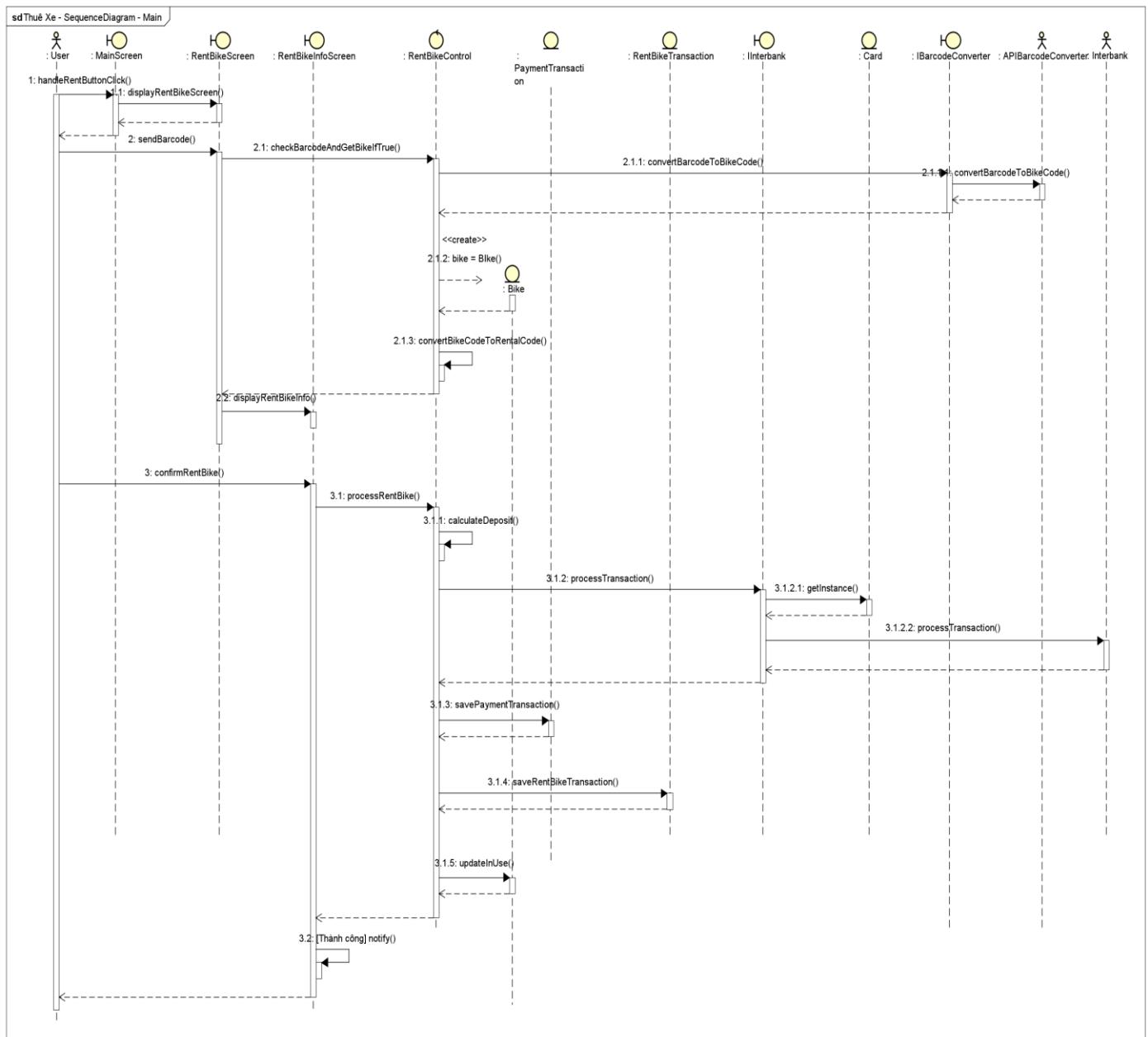
2.3.1 Biểu đồ tương tác cho UC001 – Xem thông tin xe trong bãi



Hình 3 Biểu đồ tương tác UC001

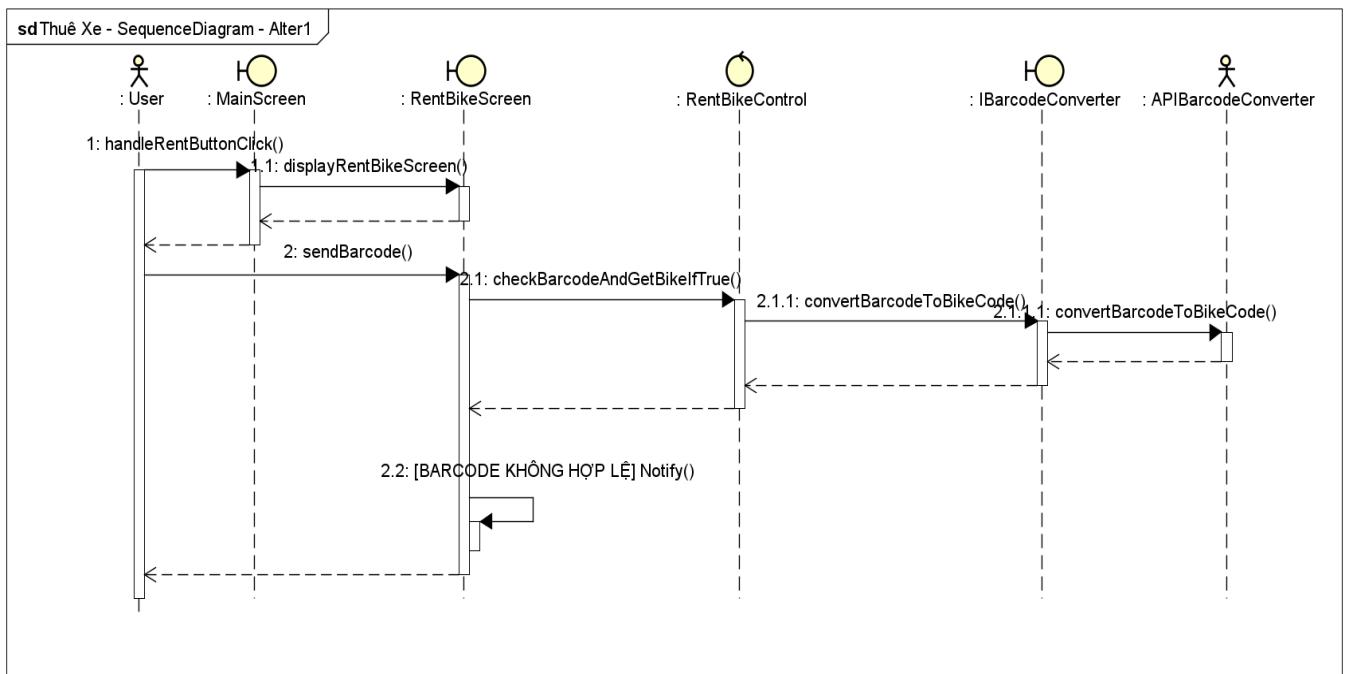
2.3.2 Biểu đồ tương tác cho UC002 – Thuê xe:

* Luồng sự kiện chính:



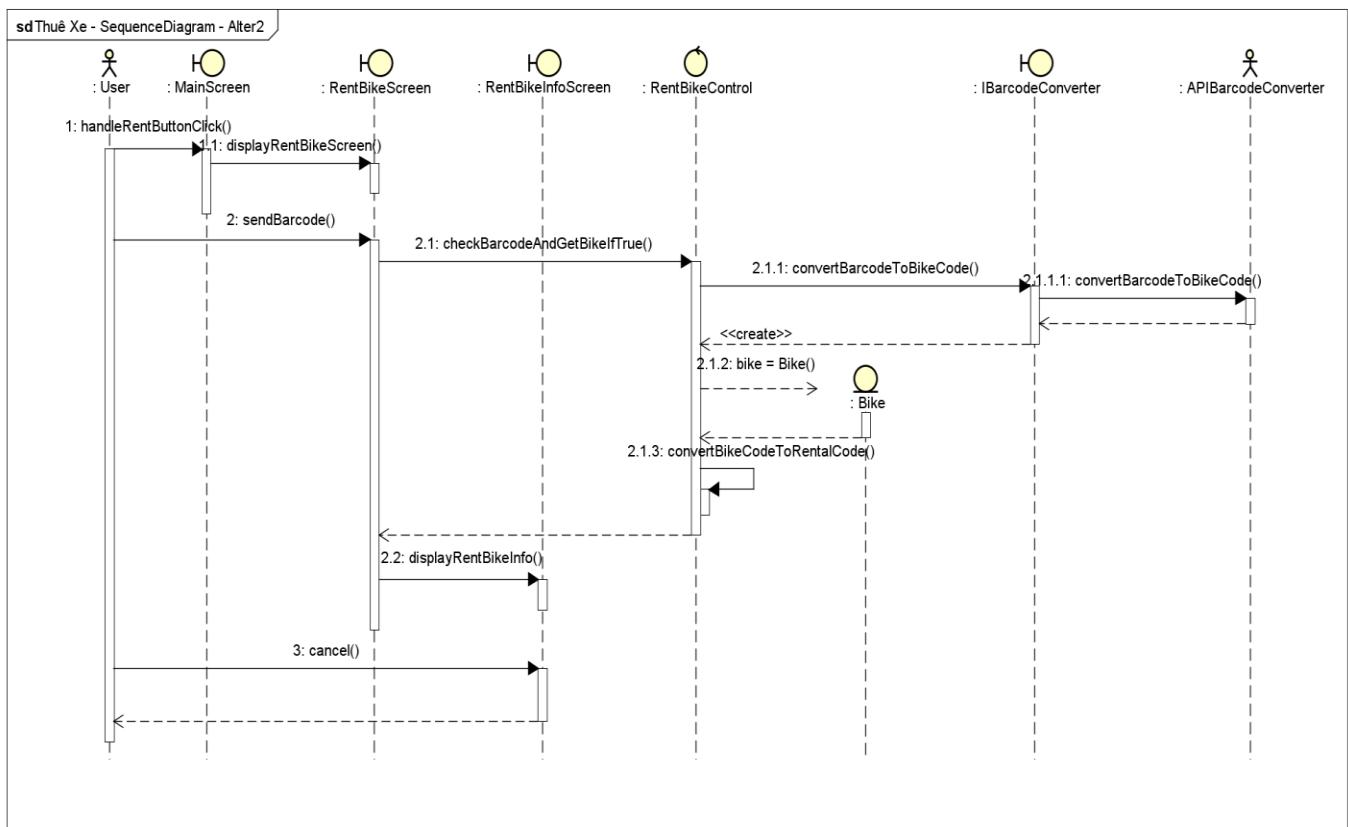
Hình 4 Biểu đồ tương tác UC002 – Luồng sự kiện chính

* Luồng sự kiện thay thế 1:



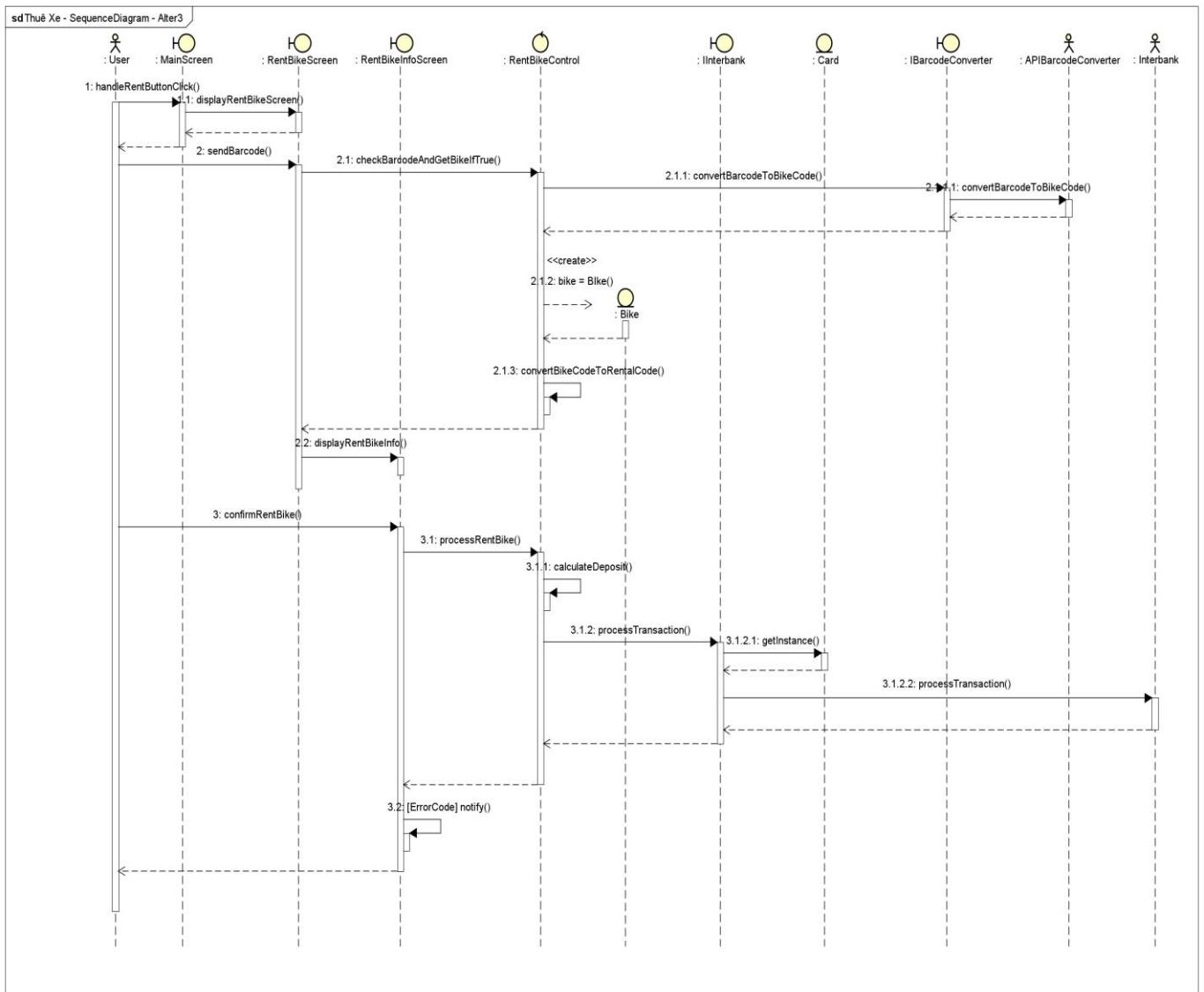
Hình 5 Biểu đồ tương tác UC002 – Luồng sự kiện thay thế 1

* Luồng sự kiện thay thế 2:



Hình 6 Biểu đồ tương tác UC002 – Luồng sự kiện thay thế 2

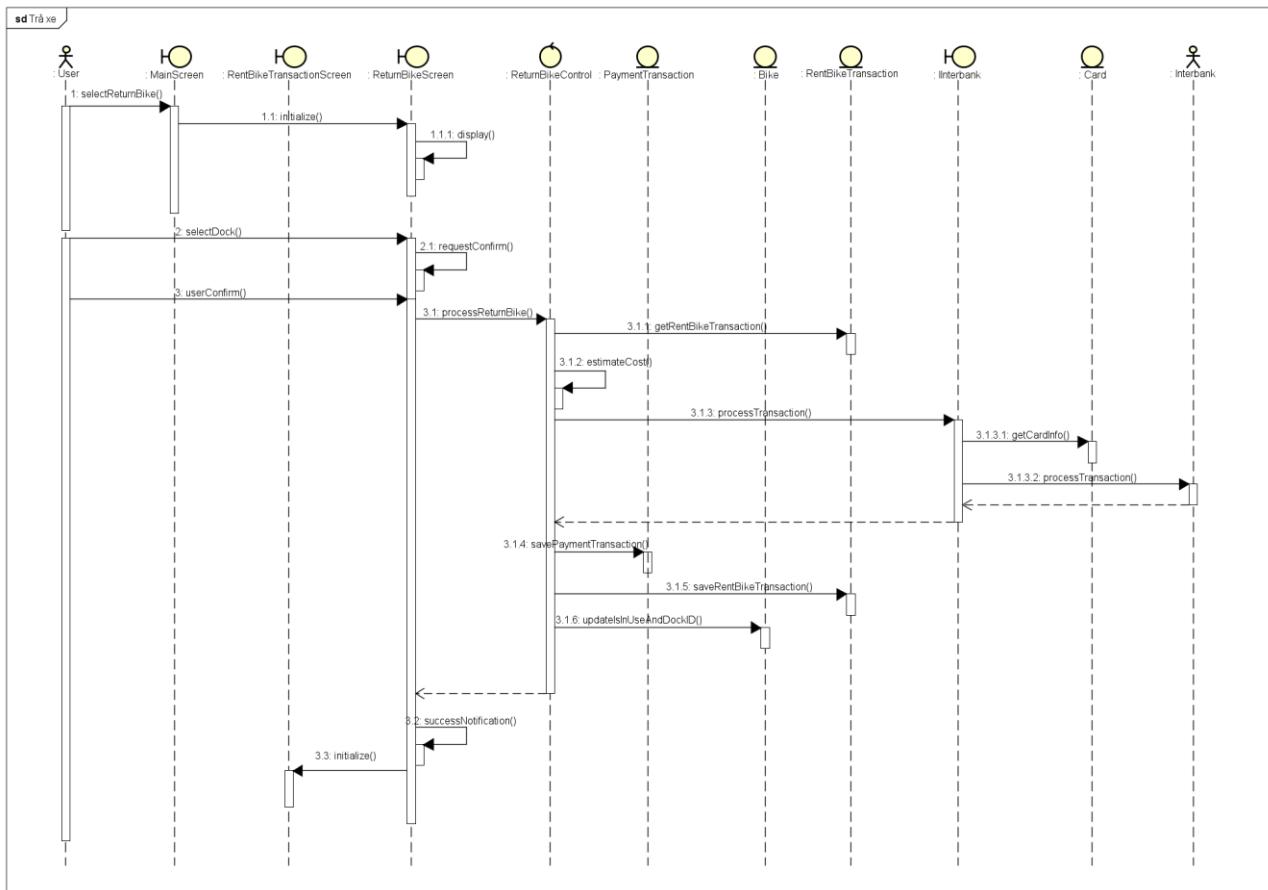
* Luồng sự kiện thay thế 3:



Hình 7 Biểu đồ tương tác UC002 – Luồng sự kiện thay thế 3

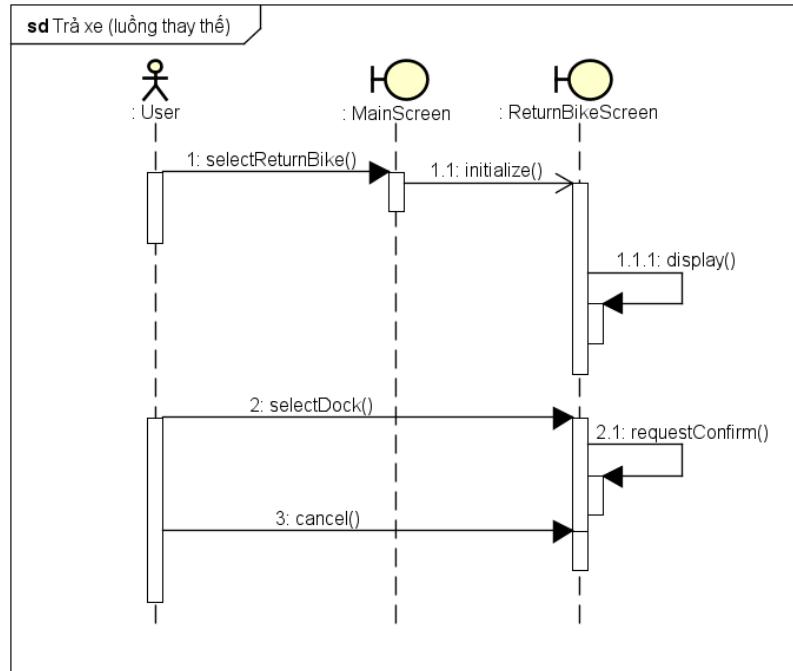
2.3.3 Biểu đồ tương tác cho UC03 – Trả xe:

* Luồng sự kiện chính



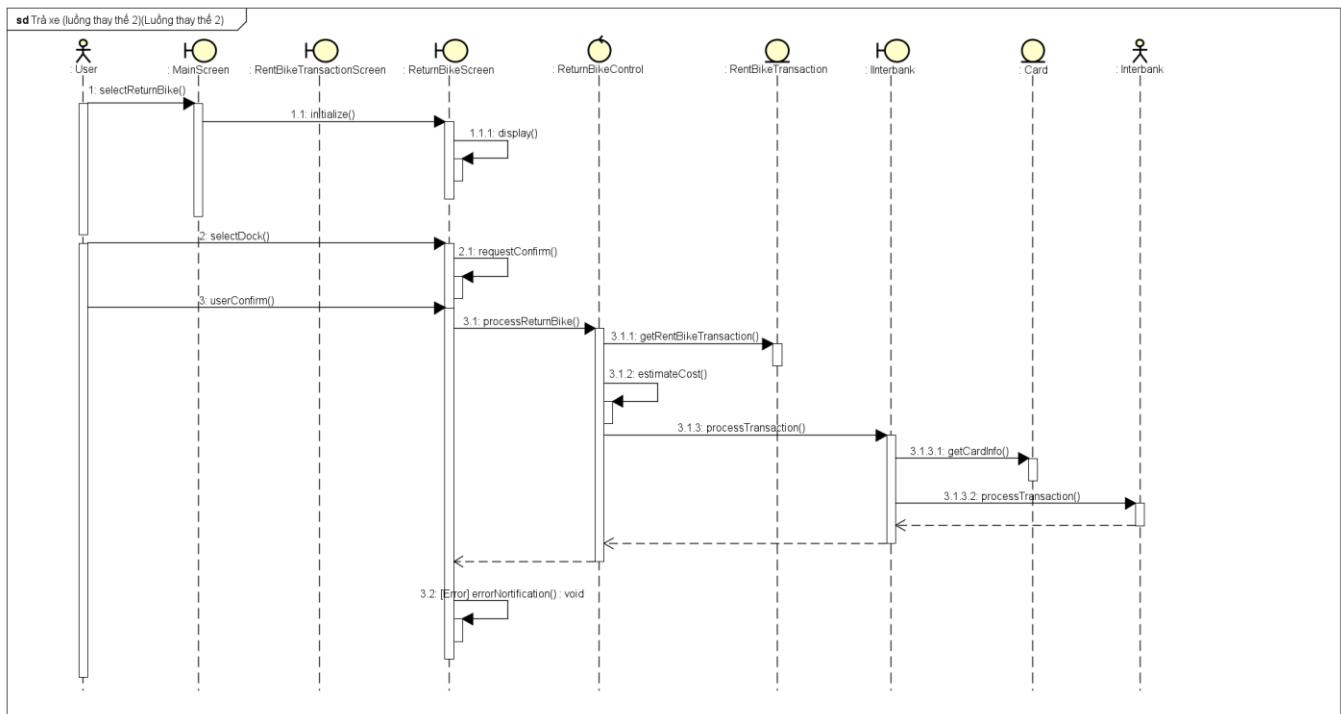
Hình 8 Biểu đồ tương tác UC003 – Luồng sự kiện chính

* Luồng sự kiện thay thế 1:



Hình 9 Biểu đồ tương tác UC003 – Luồng sự kiện thay thế 1

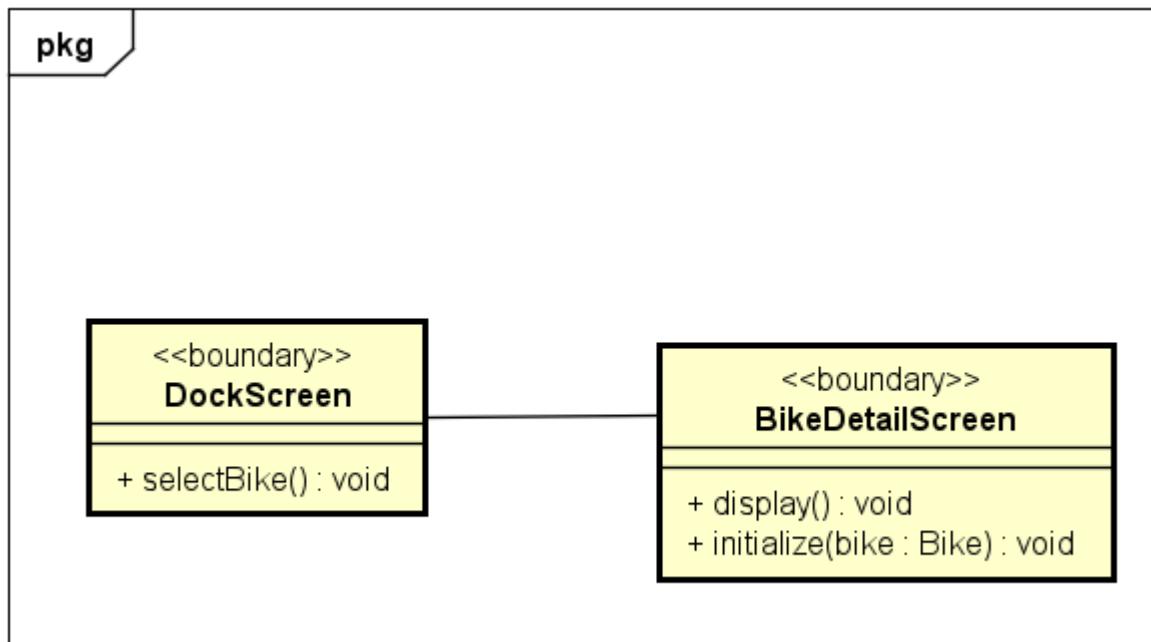
* Luồng sự kiện thay thế 2:



Hình 10 Biểu đồ tương tác UC003 – Luồng sự kiện thay thế 2

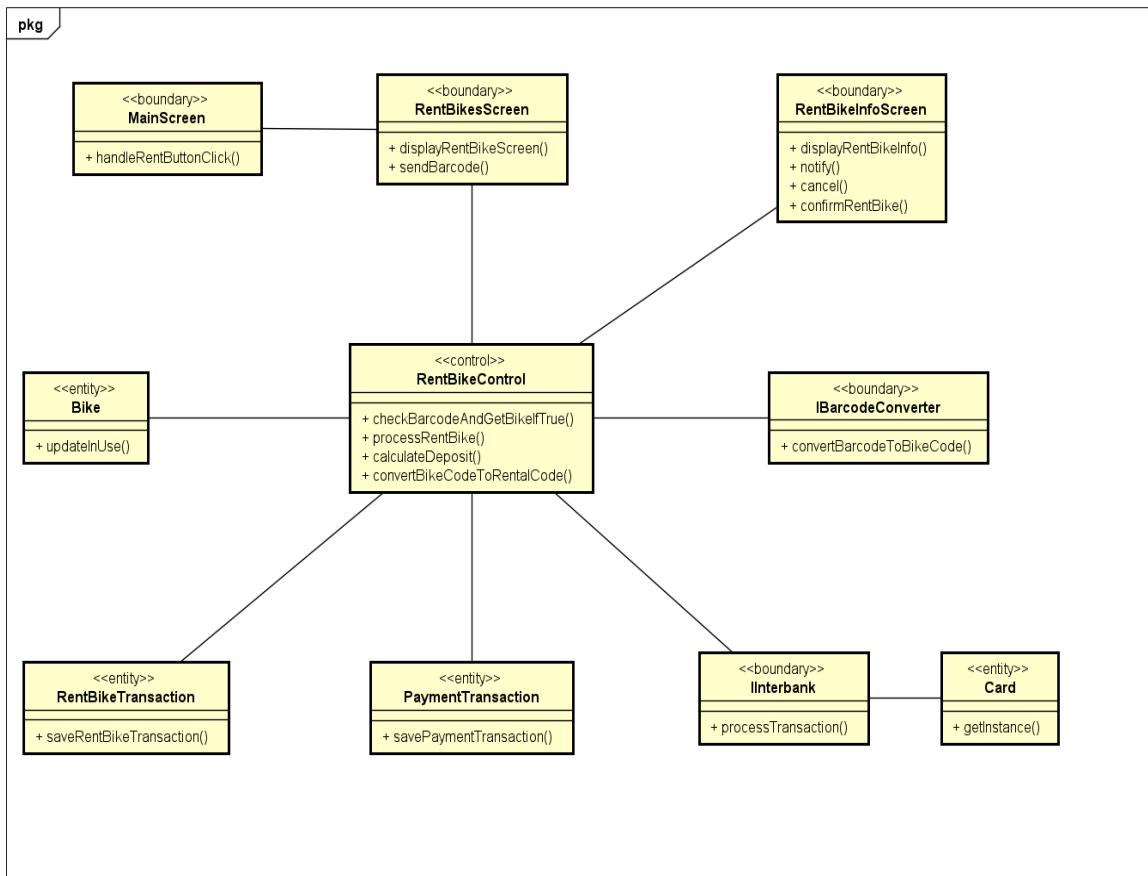
2.4 Sơ đồ lớp phân tích

2.4.1 Use case 001 – Xem thông tin xe trong bãi



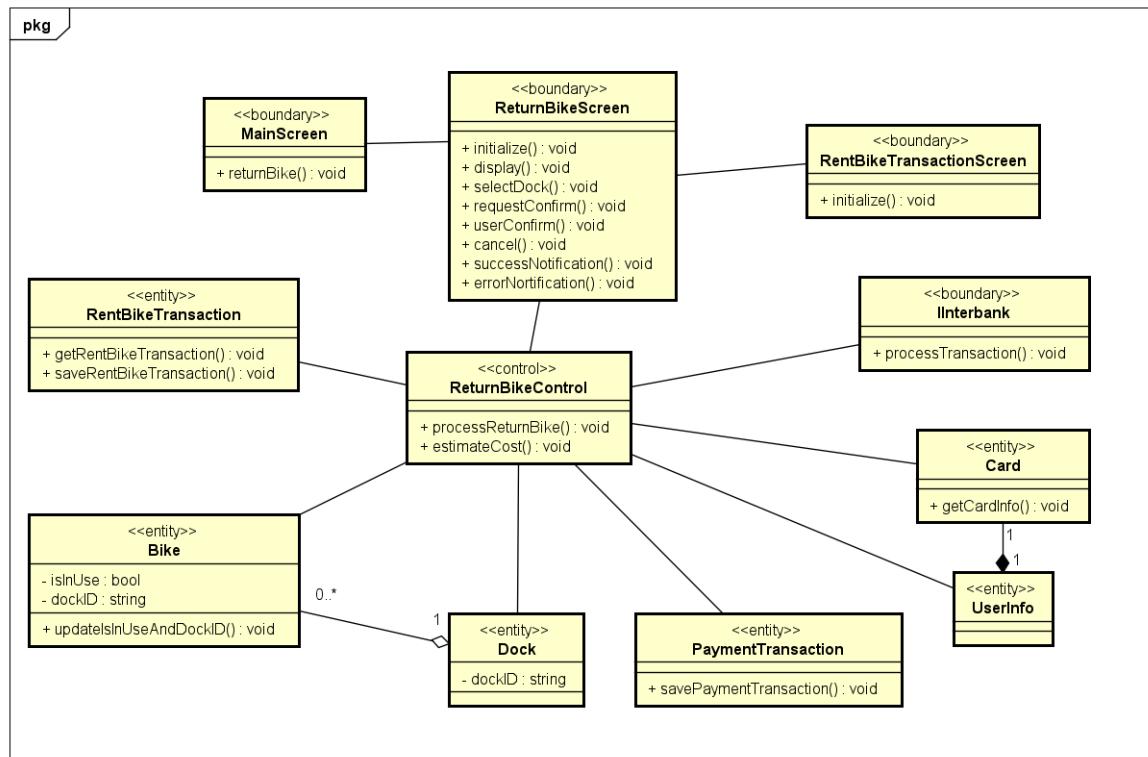
Hình 11 Sơ đồ lớp phân tích UC001

2.4.2 Use case 002 – Thuê Xe



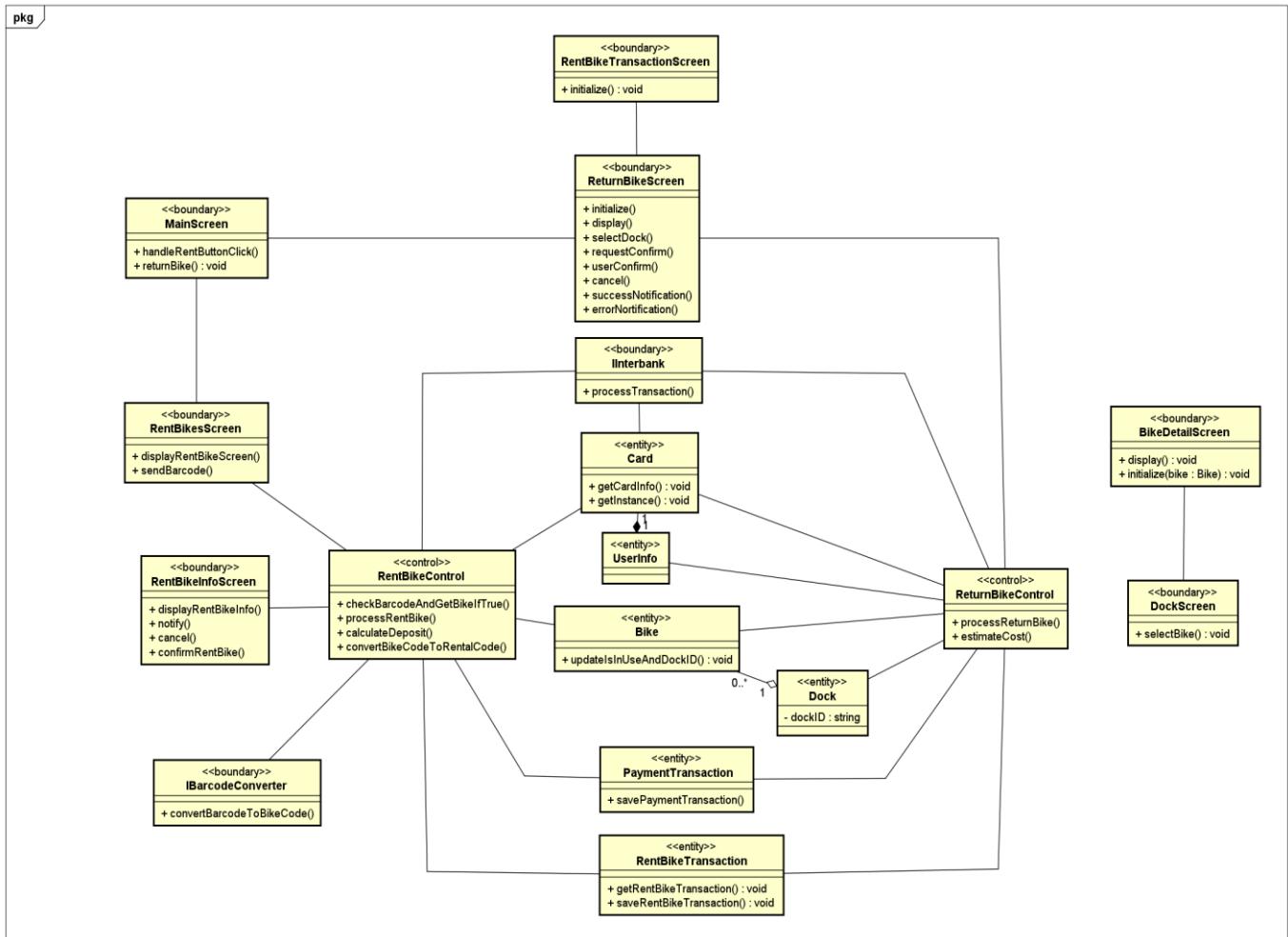
Hình 12 Sơ đồ lớp phân tích UC002

2.4.3 Use case 003 - Trả xe



Hình 13 Sơ đồ lớp phân tích UC003

2.4.4 Sơ đồ lớp phân tích gộp



Hình 14 Sơ đồ lớp phân tích gộp

3 Thiết kế giao diện

3.1 Chuẩn hóa cấu hình màn hình

* Display:

- Số lượng màu hỗ trợ: 16,777,216 màu
- Độ phân giải: Main Screen: 680x800 pixels, các màn hình liên quan nghiệp vụ còn lại có độ phân giải: 636x556 pixels, các notification box có cấu hình độ phân giải 200x300 pixels.

* Screen:

- Vị trí đặt các nút: "Xác nhận", "Hủy", "OK", "YES", "NO" sẽ đặt ở vị trí hàng cuối của màn hình

- Chủ đề của màn hình sẽ đặt ở chính giữa hàng trên của màn hình
- Khu vực giới thiệu thông tin về app được đặt ở khu vực dưới cùng của màn hình

* Control: Trình dự di chuyển chính của màn hình:

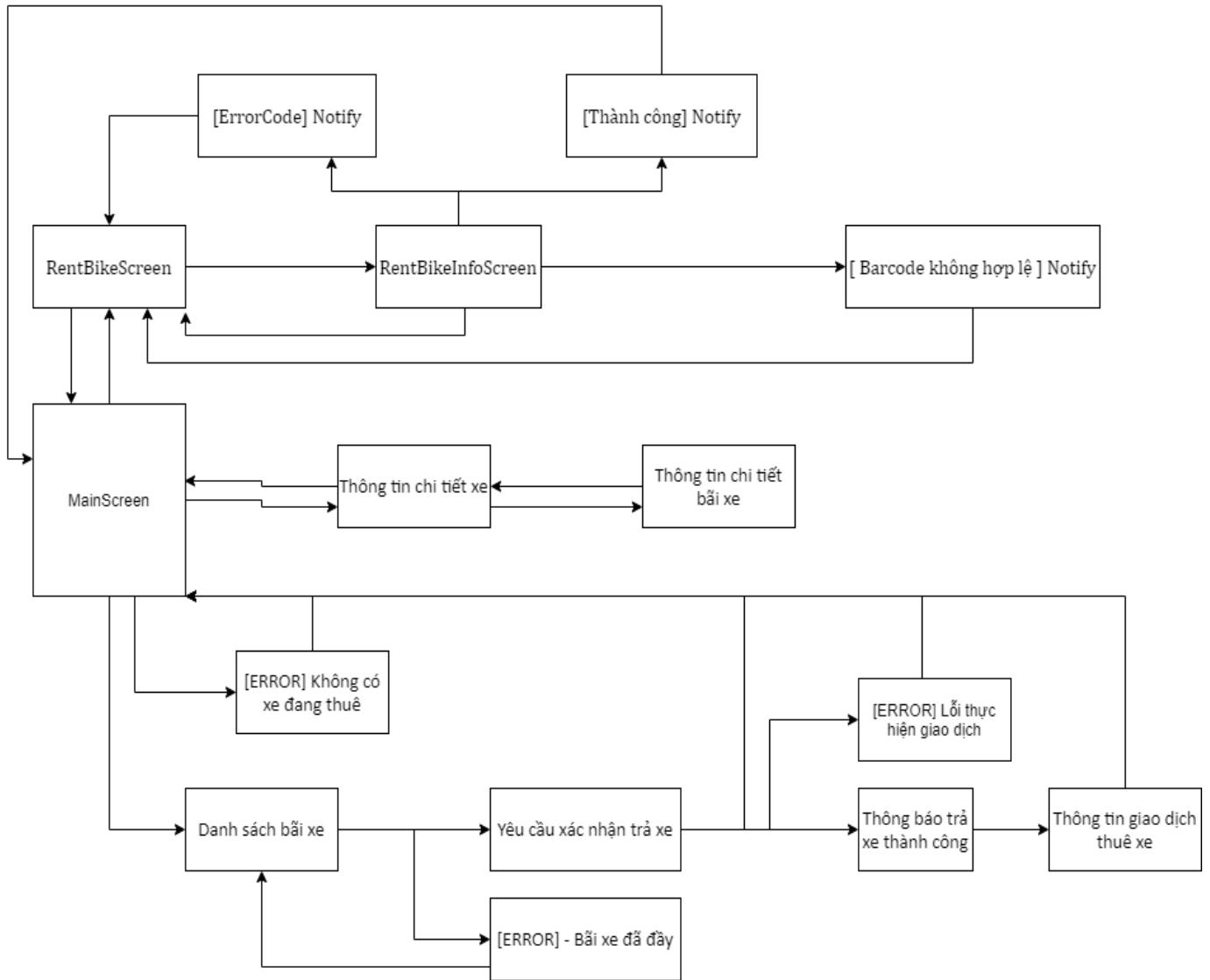
- Main Screen
- Rent Bike Screen
- Rent Bike Info Screen
- Notification Screen
- Return Bike Screen
- Rent Bike Transaction Screen

* Điều hướng màn hình:

Chúng em xây dựng và thiết kế không sử dụng nút “Back” để quay về màn hình trước mà thông qua nút “X” ở góc bên phải của mỗi màn hình.

3.2 Giao diện người dùng

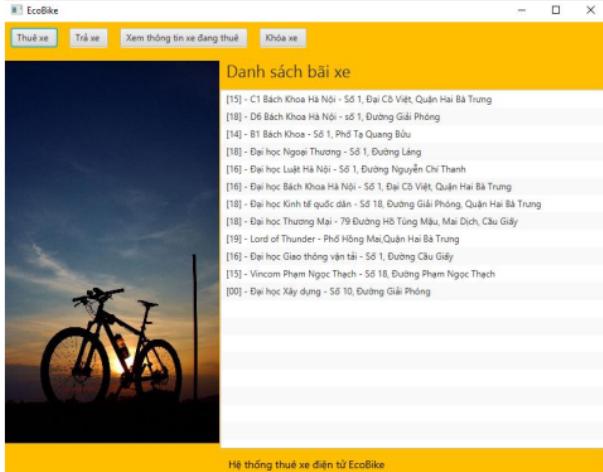
3.2.1 Biểu đồ dịch chuyển màn hình



Hình 15 Sơ đồ dịch chuyển màn hình

3.2.2 Thiết kế giao diện

3.2.2.1 Thiết kế giao diện “MainScreen”

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Danh sách bãi xe	16/12/2020			Vũ Trung Nghĩa
	Control	Operation	Function		
	Khu vực hiện thị danh sách bãi xe	Khởi tạo	Hiển thị danh sách các bãi xe để người dùng trả xe		
	Khu vực hiện thị danh sách bãi xe	Double clicks	Người dùng muốn xem chi tiết thông tin một bãi xe		
	Nút “Thuê xe”	Click	Người dùng muốn thuê xe		
	Nút “Trả xe”	Click	Người dùng muốn trả xe		
	Nút “Khóa xe”	Click	Người dùng muốn tạm dừng xe và khóa xe		
	Nút “Xem thông tin xe đang thuê”	Click	Người dùng muốn xem thông tin xe đang thuê		

Các trường thuộc tính:

Screen Name	Danh sách bãi xe		
Attribute	Type	Field Attribute	Remarks
Số vị trí đỗ xe còn trống	String	Đen	Căn trái
Tên bãi xe	String	Đen	Căn trái
Địa chỉ bãi xe	String	Đen	Căn trái

3.2.2.2 Thiết kế giao diện Use case “Thuê xe”

1. Màn hình Rent Bike Screen:

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	RentBikeScreen	14/12/2020			Lê Thế Nam
		Control	Operation	Function	
RentBikeScreen	Thuê Xe	Khu vực nhập barcode	Nhập từ bàn phím	Người dùng nhập từ bàn phím barcode của xe mà người dùng muốn thuê	
	Qúy khách vui lòng nhập barcode Nhập barcode Thuê xe	Nút “Thuê xe”	Click	Người dùng xác nhận mong muốn thuê xe với barcode như trên để hệ thống kiểm tra và trả lại kết quả	
	Hệ thống thuê xe điện tử EcoBike				

Các trường thuộc tính

Screen Name	RentBikeScreen		
Attribute	Type	Field Attribute	Remarks
Barcode	Numeral	Trắng	Căn trái

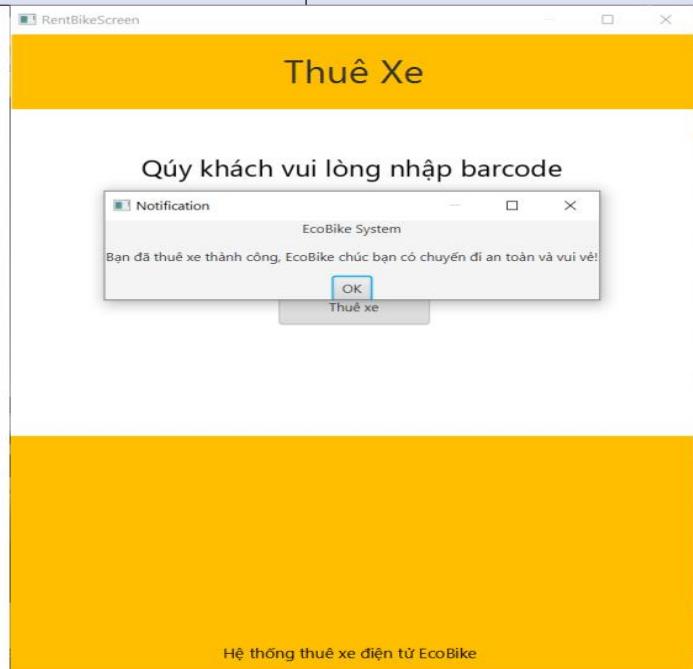
2. Màn hình RentBikeInfoScreen:

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	RentBikeInfoScreen	14/12/2020			Lê Thế Nam
 <p>Thông tin chi tiết xe</p> <p>Barcode: 20200001</p> <p>Loại xe: Xe đạp điện</p> <p>Giá trị: 1200000</p> <p>Giá thuê trong 30 phút đầu tiên: 15000</p> <p>Giá thuê mỗi 15 phút sau: 5000</p> <p>Lượng pin còn lại (%): 60</p> <p>Thời gian sử dụng tối đa: 4.0</p> <p>Biển số xe: XDD-200001</p> <p>Số tiền đặt cọc: 480000</p> <p>Bạn có muốn thuê xe không? <input type="button" value="Xác nhận"/> <input type="button" value="Hủy"/></p> <p>Hệ thống thuê xe điện tử EcoBike</p>		Control	Operation	Function	
Khu vực hiển thị thông tin chi tiết về xe và tiền cọc		Khởi tạo	Hiển thị thông tin chi tiết về xe mà người dùng đã nhập barcode hợp lệ vào hệ thống và tiền cọc cần trả tương ứng.		
<p>Nút “Xác nhận”</p> <p>Nút “Hủy”</p>		Click	Người dùng xác nhận thuê xe với thông tin hiển thị ở trên		
		Click	Người dùng hủy hành động thuê xe sau khi đã xem thông tin		

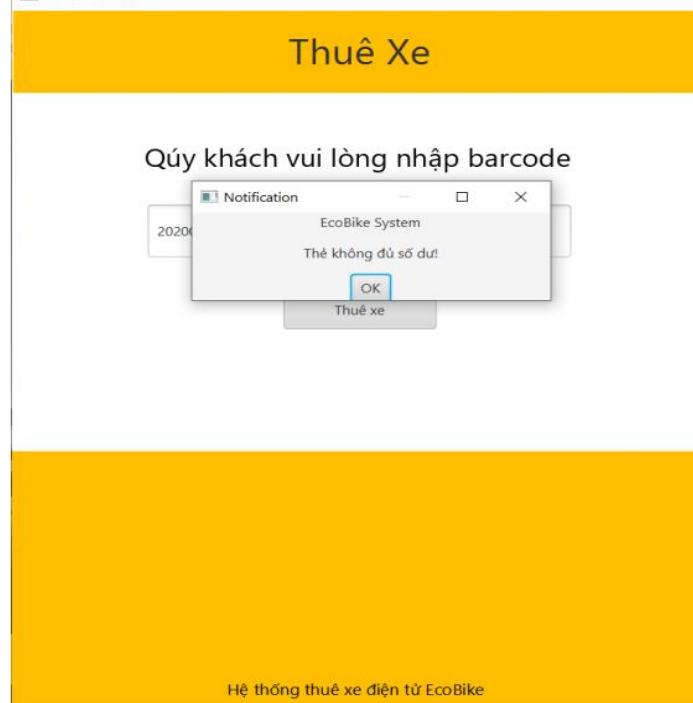
Các trường thuộc tính

Screen Name	RentBikeScreen		
Attribute	Type	Field Attribute	Remarks
Barcode	Numeral	Đỏ	Căn trái
Loại xe	String	Đỏ	Căn trái
Giá trị	Numeral	Đỏ	Căn trái
Giá thuê trong 30 phút đầu	Numeral	Đỏ	Căn trái
Giá thuê mỗi 15 phút sau	Numeral	Đỏ	Căn trái
Lượng pin còn lại	Numeral	Đỏ	Căn trái
Thời gian sử dụng tối đa	Numeral	Đỏ	Căn trái
Biển số xe	String	Đỏ	Căn trái
Số tiền cọc	Numeral	Đỏ	Căn trái

3. Màn hình [Thành công] Notify:

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	[Thành công] Notify	14/12/2020			Lê Thế Nam
		Control	Operation	Function	
		Khu vực hiển thị thông báo thành công của hệ thống	Khởi tạo	Thông báo cho khách hàng giao dịch thuê xe đã thành công	
	Nút "OK"	Click	Người dùng xác nhận thông báo		

4. Màn hình [ErrorCode] Notify:

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	[ErrorCode] Notify	14/12/2020			Lê Thế Nam
		Control	Operation	Function	
		Khu vực hiển thị thông báo lỗi của hệ thống	Khởi tạo	Sau khi nhận lỗi từ api interbank, hệ thống thông báo cho khách hàng giao dịch thuê xe đã gặp lỗi và hiển thị lỗi.	
	Nút "OK"	Click	Người dùng xác nhận thông báo		

5. Màn hình [Barcode không hợp lệ] Notify:

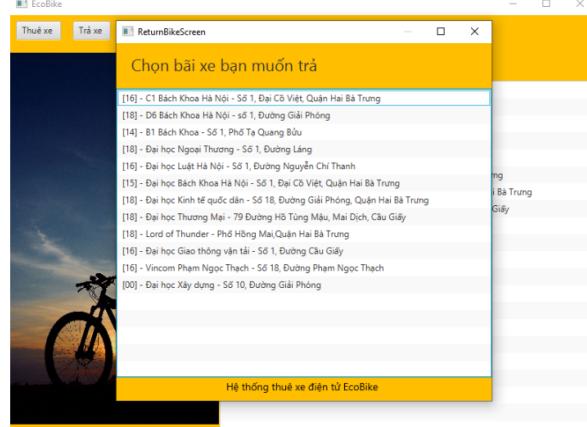
EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	[Barcode không hợp lệ] Notify	14/12/2020			Lê Thế Nam
		Control	Operation	Function	
		Khu vực hiển thị thông báo lỗi barcode của hệ thống	Khởi tạo	Sau khi nhận barcode từ người dùng, hệ thống kiểm tra và thấy không hợp lệ, sau đó thông báo cho khách hàng lỗi barcode để khách hàng có thể kiểm tra lại.	
		Nút "OK"	Click	Người dùng xác nhận thông báo	

3.2.2.3 Thiết kế giao diện Use case “Trả xe”

1. Màn hình “[ERROR] Không có xe đang thuê”:

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	[ERROR] Không có xe đang thuê	16/12/2020			Vũ Trung Nghĩa
		Control	Operation	Function	
		Nút "OK"	Click	Người dùng xác nhận đã xem và tắt cửa sổ báo lỗi	
		Khu vực hiển thị thông báo không có xe đang thuê của hệ thống	Khởi tạo	Thông báo cho người dùng không có xe đang thuê nên không thể trả xe	

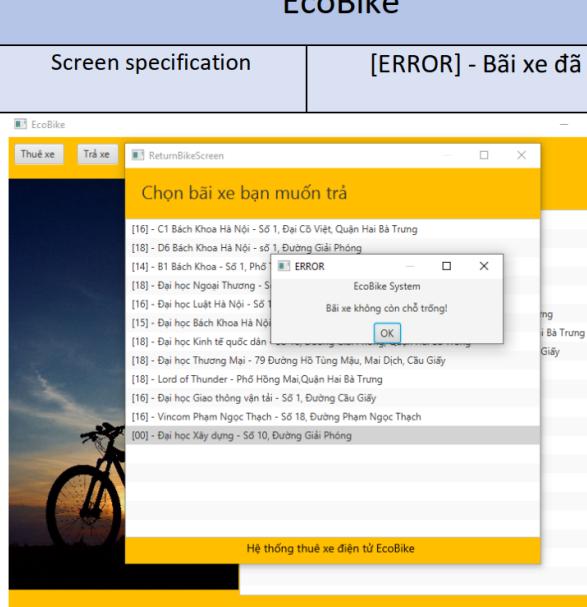
2. Màn hình “Danh sách bãi xe”

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Danh sách bãi xe	16/12/2020			Vũ Trung Nghĩa
		Control	Operation	Function	
		Khu vực hiện thị danh sách bãi xe	Khởi tạo	Hiển thị danh sách các bãi xe để người dùng trả xe	
		Khu vực hiện thị danh sách bãi xe	Double clicks	Người dùng muốn xem chi tiết thông tin một bãi xe	

Các trường thuộc tính:

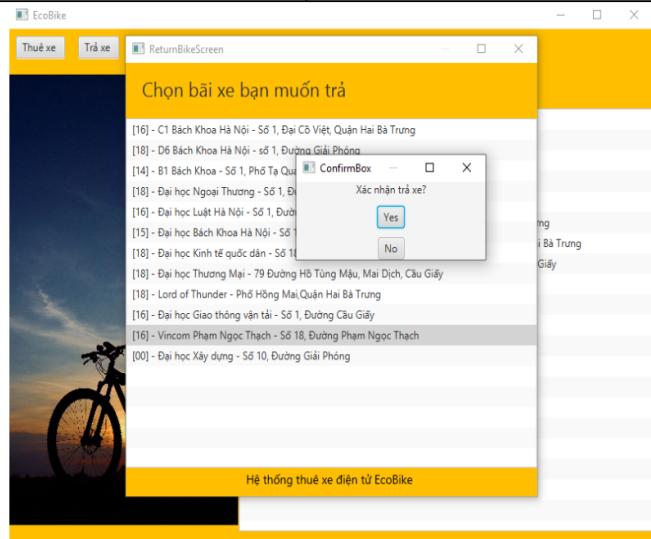
Screen Name	Danh sách bãi xe		
Attribute	Type	Field Attribute	Remarks
Số vị trí chỗ xe còn trống	String	Đen	Căn trái
Tên bãi xe	String	Đen	Căn trái
Địa chỉ bãi xe	String	Đen	Căn trái

3. Màn hình “[ERROR] - Bãi xe đã đầy”

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	[ERROR] - Bãi xe đã đầy	16/12/2020			Vũ Trung Nghĩa
		Control	Operation	Function	
		Nút “OK”	Click	Người dùng xác nhận thông báo	
		Khu vực hiển thị thông báo	Khởi tạo	Thông báo cho người dùng bãi xe không còn chỗ trống, cần trả vào một bãi xe khác	

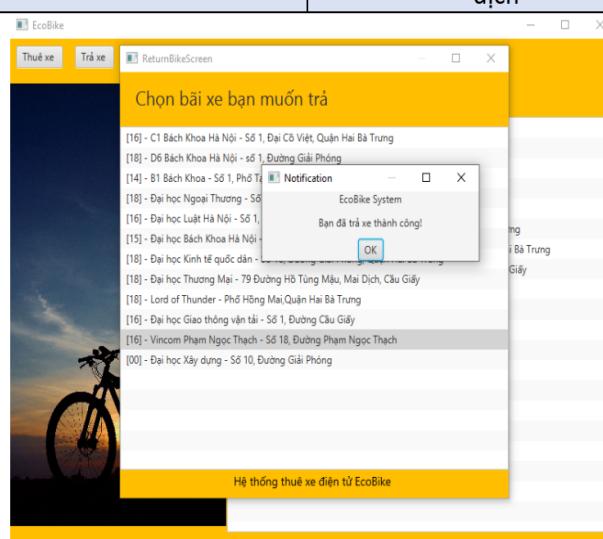
4. Màn hình “Yêu cầu xác nhận trả xe”

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Yêu cầu xác nhận trả xe	16/12/2020			Vũ Trung Nghĩa
		Control	Operation	Function	
	Nút “Yes”	Click		Người dùng xác nhận trả xe	
	Nút “Hủy”	Click		Người dùng hủy trả xe	
	Khu vực hiện thị thông báo yêu cầu xác nhận trả xe	Khởi tạo		Yêu cầu người dùng xác nhận trả xe	

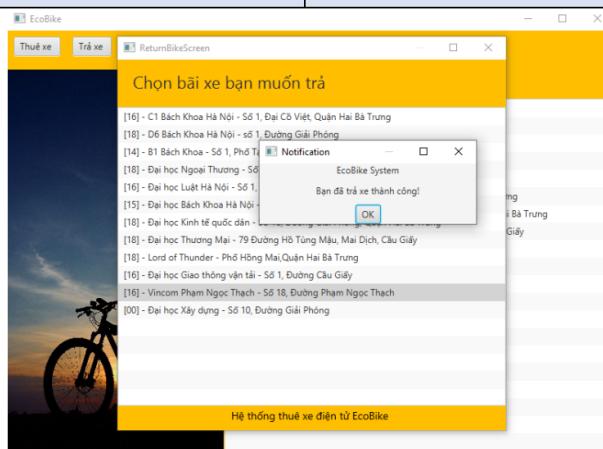


5. Màn hình “[ERROR] Lỗi thực hiện giao dịch”

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	[ERROR] Lỗi thực hiện giao dịch	16/12/2020			Vũ Trung Nghĩa
		Control	Operation	Function	
	Nút “OK”	Click		Người dùng xác nhận thông báo	
	Khu vực hiển thị thông báo có lỗi xảy ra khi thực hiện giao dịch	Khởi tạo		Thông báo có lỗi xảy ra khi thực hiện giao dịch	



6. Màn hình “Thông báo trả xe thành công”

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge		
Screen specification	Thông báo trả xe thành công	16/12/2020			Vũ Trung Nghĩa		
		Control	Operation	Function			
	Nút “OK”	Click	Người dùng xác nhận thông báo				
Khu vực hiển thị thông báo trả xe thành công		Khởi tạo	Thông báo người dùng trả xe thành công				

7. Màn hình “Thông tin giao dịch thuê xe”

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge		
Screen specification	Thông tin giao dịch thuê xe	16/12/2020			Vũ Trung Nghĩa		
		Control	Operation	Function			
	Nút “OK”	Click	Người dùng xác nhận giao dịch				
Khu vực hiển thị thông tin chi tiết giao dịch thuê xe		Khởi tạo	Hiện thị thông tin chi tiết giao dịch thuê xe của người dùng				

* Định nghĩa trường thuộc tính:

Screen Name	Thông tin giao dịch thuê xe		
Attribute	Type	Field Attribute	Remarks
Mã thuê xe	String	Đỏ	Căn trái
Mã xe	String	Đỏ	Căn trái
Loại xe	String	Đỏ	Căn trái
Chi phí thuê xe	String	Đỏ	Căn trái
Người thuê	String	Đỏ	Căn trái
Giá thuê trong 30 phút đầu	String	ĐỎ	Căn trái
Giá thuê mỗi 15 phút sau	String	ĐỎ	Căn trái
Thời điểm thuê xe	String	ĐỎ	Căn trái
Thời điểm trả xe	String	ĐỎ	Căn trái
Tiền đặt cọc	String	ĐỎ	Căn trái

3.2.2.4 Thiết kế giao diện Use case “Xem thông tin xe trong bãi”

1. Màn hình “Thông tin chi tiết bãi xe”

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Thông tin chi tiết bãi xe	16/12/2020		Vũ Trung Nghĩa	
		Control	Operation	Function	
Mã bãi xe:	NEU	Khu vực hiện thị thông tin chi tiết bãi xe	Khởi tạo	Hiển thị thông tin chi tiết bãi xe	
Tên bãi:	Dai hoc Kinh te quoc dan	Khu vực hiện thị danh sách các xe hiện có trong bãi.	Khởi tạo	Hiển thị danh sách các xe hiện có trong bãi. Thông tin hiển thị gồm mã xe – loại xe	
Địa chỉ:	Số 18, Đường Giải Phóng, Quận Hai Bà Trưng	Khu vực hiện thị danh sách xe trong bãi	Double clicks	Người dùng xem chi tiết thông tin một xe	
Khu vực:	Hà Nội				
Số điểm đỗ xe:	20				
Số xe trong bãi:	2				
					
Hệ thống thuê xe điện tử EcoBike					

* Định nghĩa trường thuộc tính:

Screen Name	Thông tin chi tiết bãi xe		
Attribute	Type	Field Attribute	Remarks
Mã bãi xe	String	Đỏ	Căn trái
Tên bãi	String	Đỏ	Căn trái
Địa chỉ	String	Đỏ	Căn trái
Khu vực	String	Đỏ	Căn trái
Số điểm đỗ xe	String	Đỏ	Căn trái
Số xe trong bãi	String	Đỏ	Căn trái
Mã xe	String	Đen	Căn trái
Loại xe	String	Đen	Căn trái

2. Màn hình “Thông tin chi tiết xe”

EcoBike		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Thông tin chi tiết xe	16/12/2020			Vũ Trung Nghĩa
		Control Nút “OK”	Operation Click	Function Người dùng xác nhận và đóng cửa sổ	
Khu vực hiển thị thông tin chi tiết xe				Khởi tạo	Hiển thị thông tin chi tiết xe

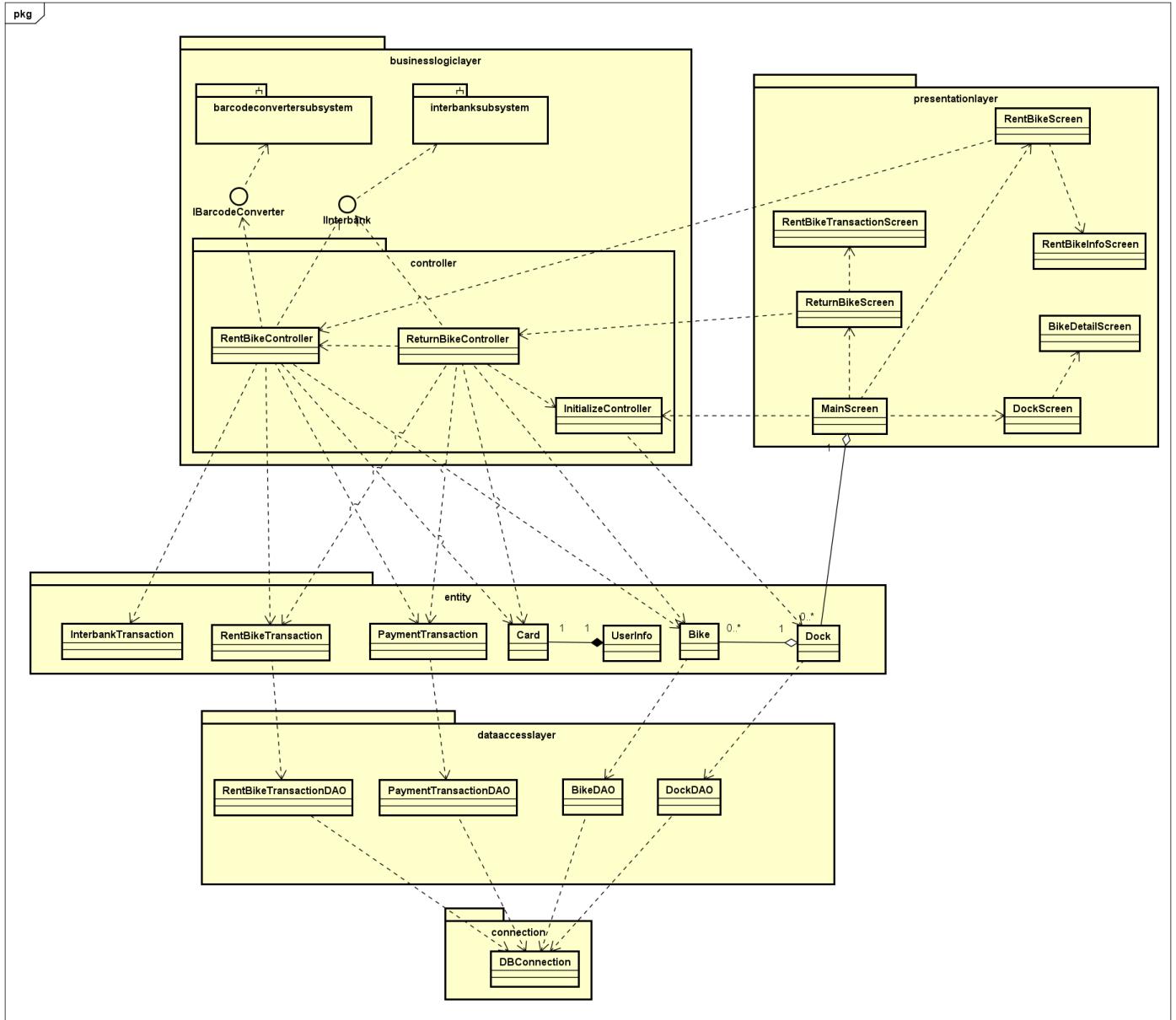
* Định nghĩa trường thuộc tính:

Screen Name	Thông tin chi tiết bãi xe		
Attribute	Type	Field Attribute	Remarks
Mã xe	String	Đỏ	Căn trái
Loại xe	String	Đỏ	Căn trái
Giá trị	String	Đỏ	Căn trái
Giá thuê 30 phút đầu	String	Đỏ	Căn trái
Giá thuê mỗi 15 phút sau 3 phút đầu	String	Đỏ	Căn trái
Lượng pin còn lại	String	Đỏ	Căn trái
Thời gian sử dụng tối đa	String	Đỏ	Căn trái
Biển số xe	String	Đỏ	Căn trái

4 Thiết kế lớp

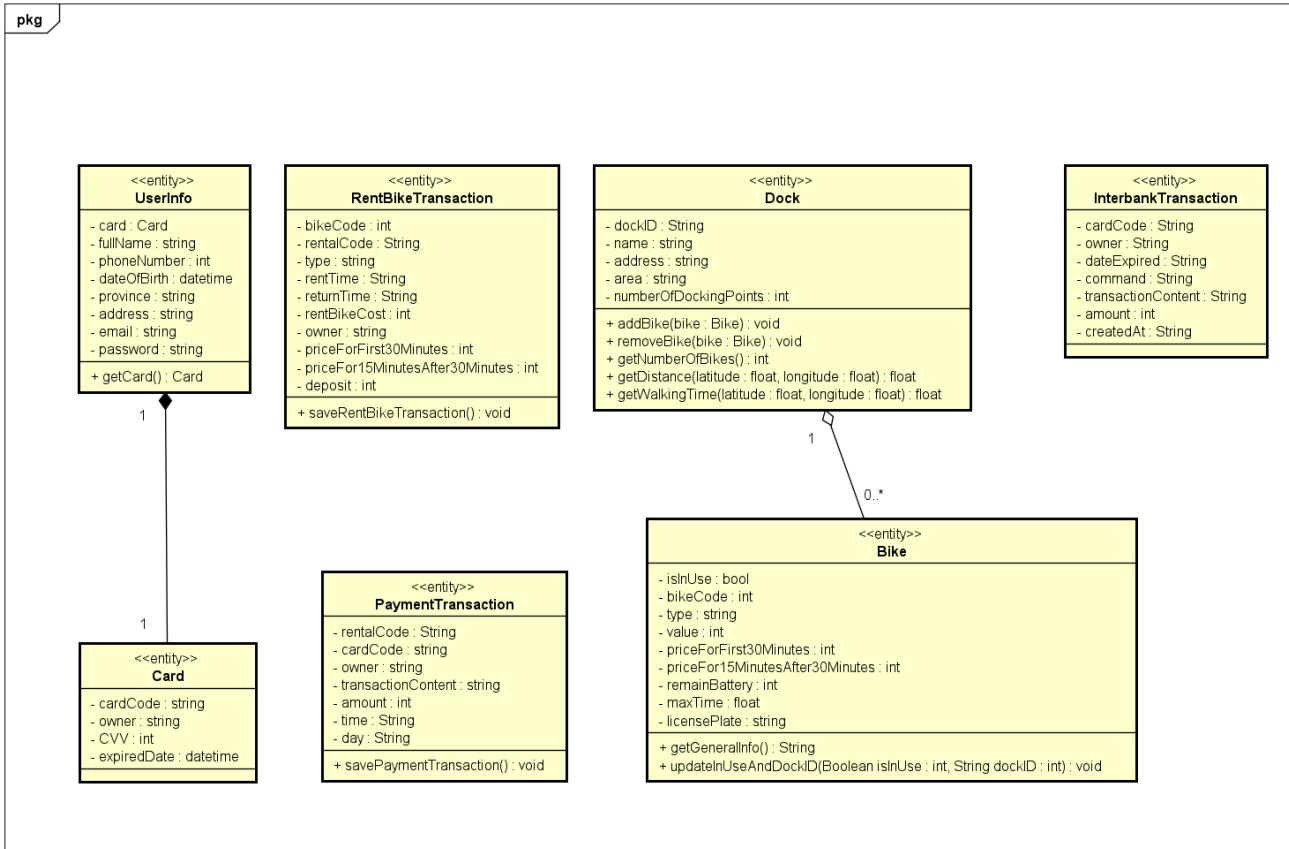
4.1 Biểu đồ lớp thiết kế

4.1.1 Biểu đồ lớp thiết kế



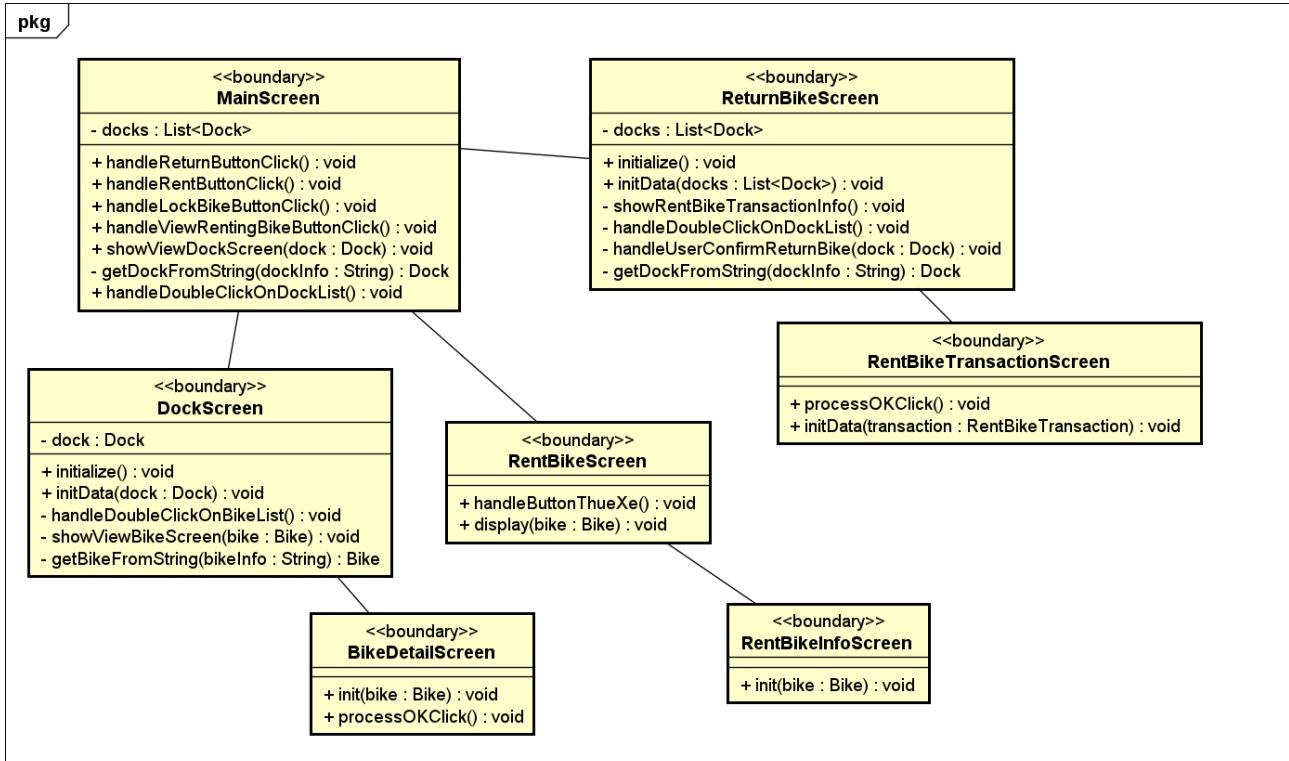
Hình 16 Biểu đồ lớp thiết kế

4.1.2 Thiết kế chi tiết gói “entity”



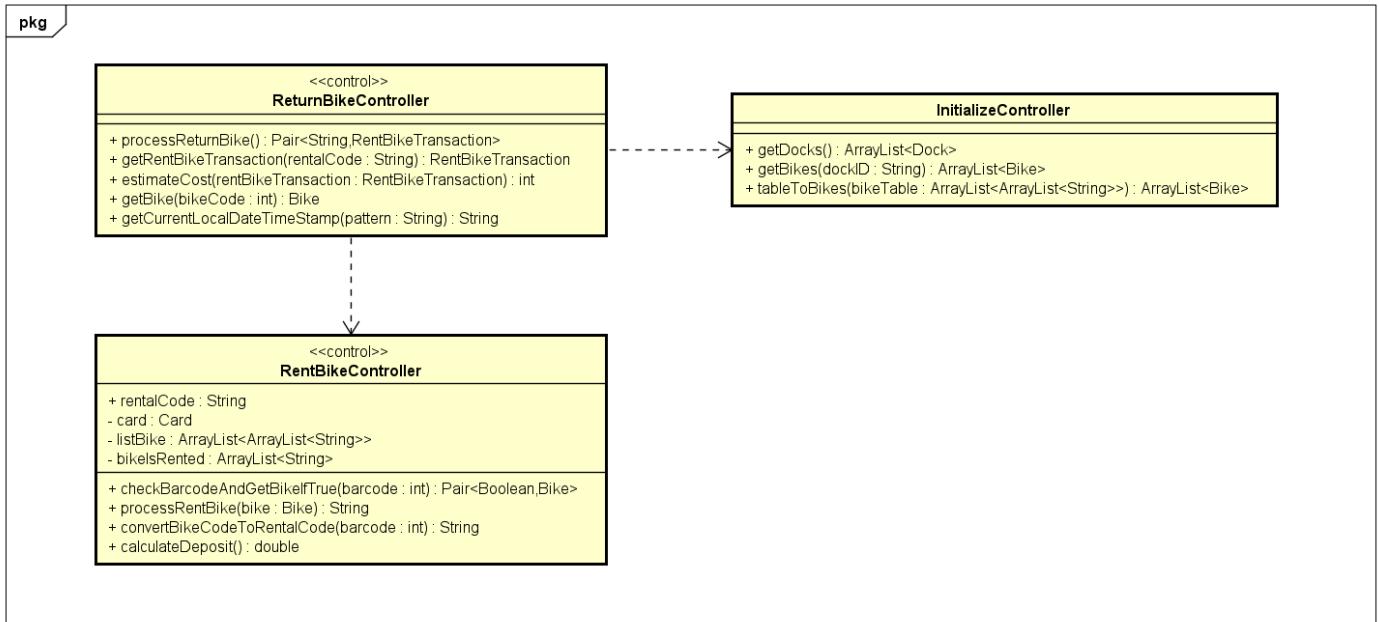
Hình 17 Thiết kế chi tiết gói “entity”

4.1.3 Thiết kế chi tiết gói “presentationlayer”



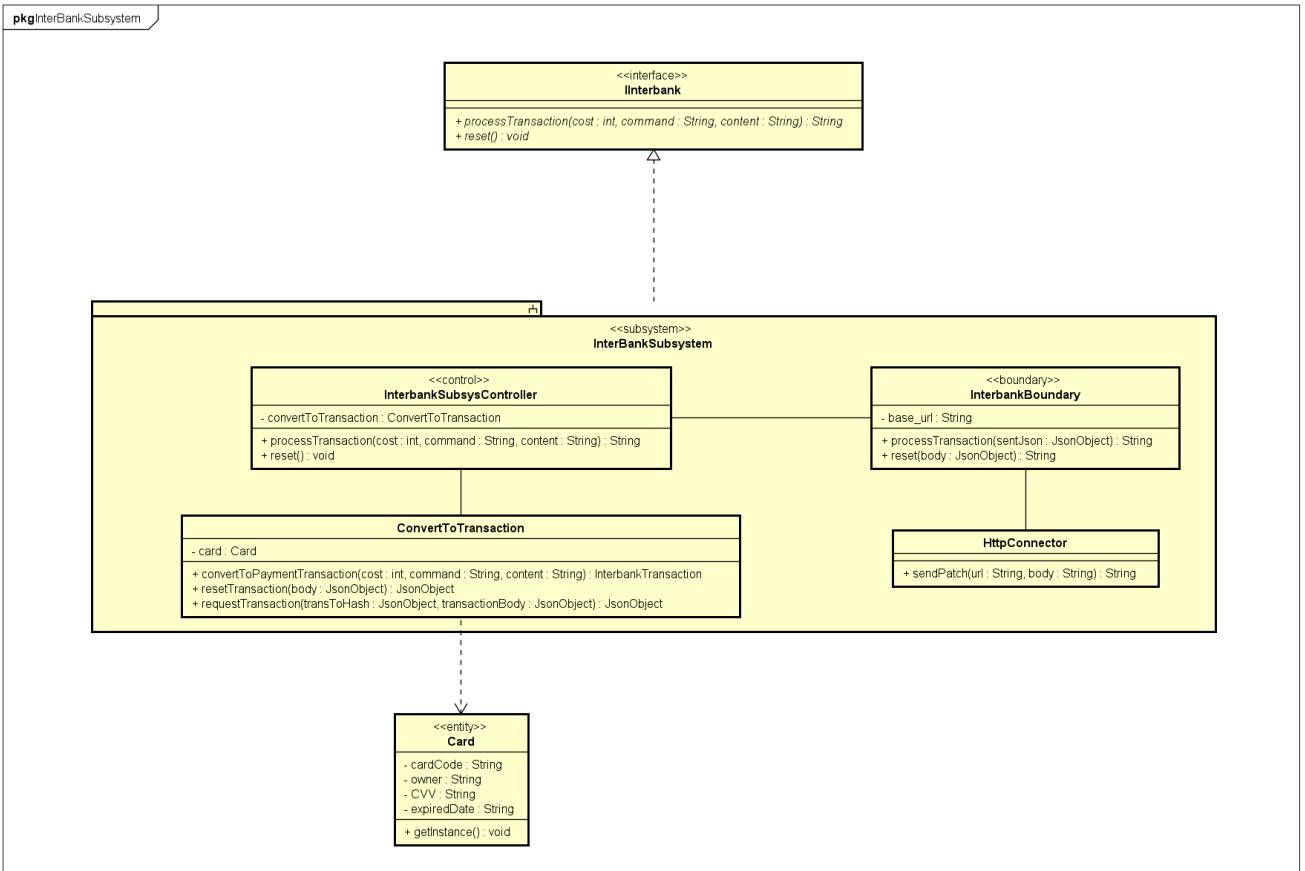
Hình 18 Thiết kế chi tiết gói “presentationlayer”

4.1.4 Thiết kế chi tiết gói controller



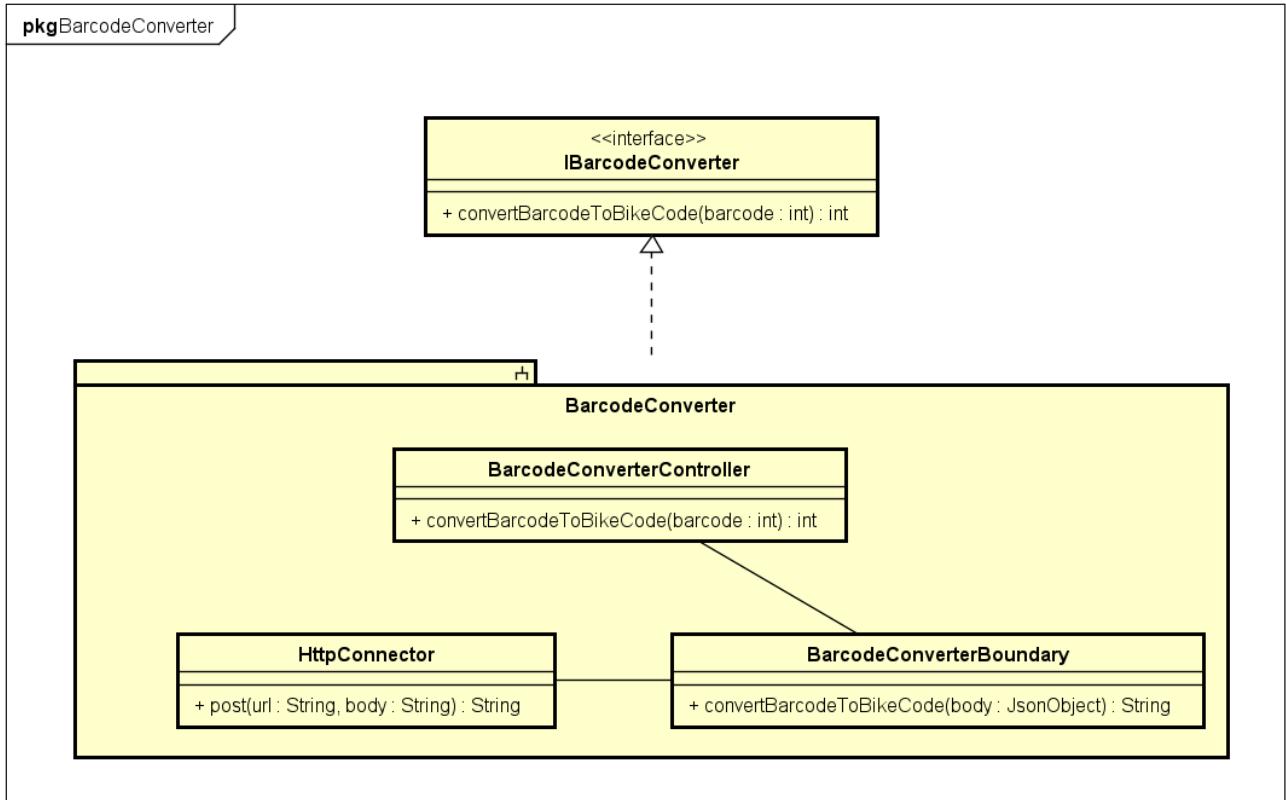
Hình 19 Thiết kế chi tiết gói “controller”

4.1.5 Thiết kế chi tiết subsystem “interbanksubsystem”



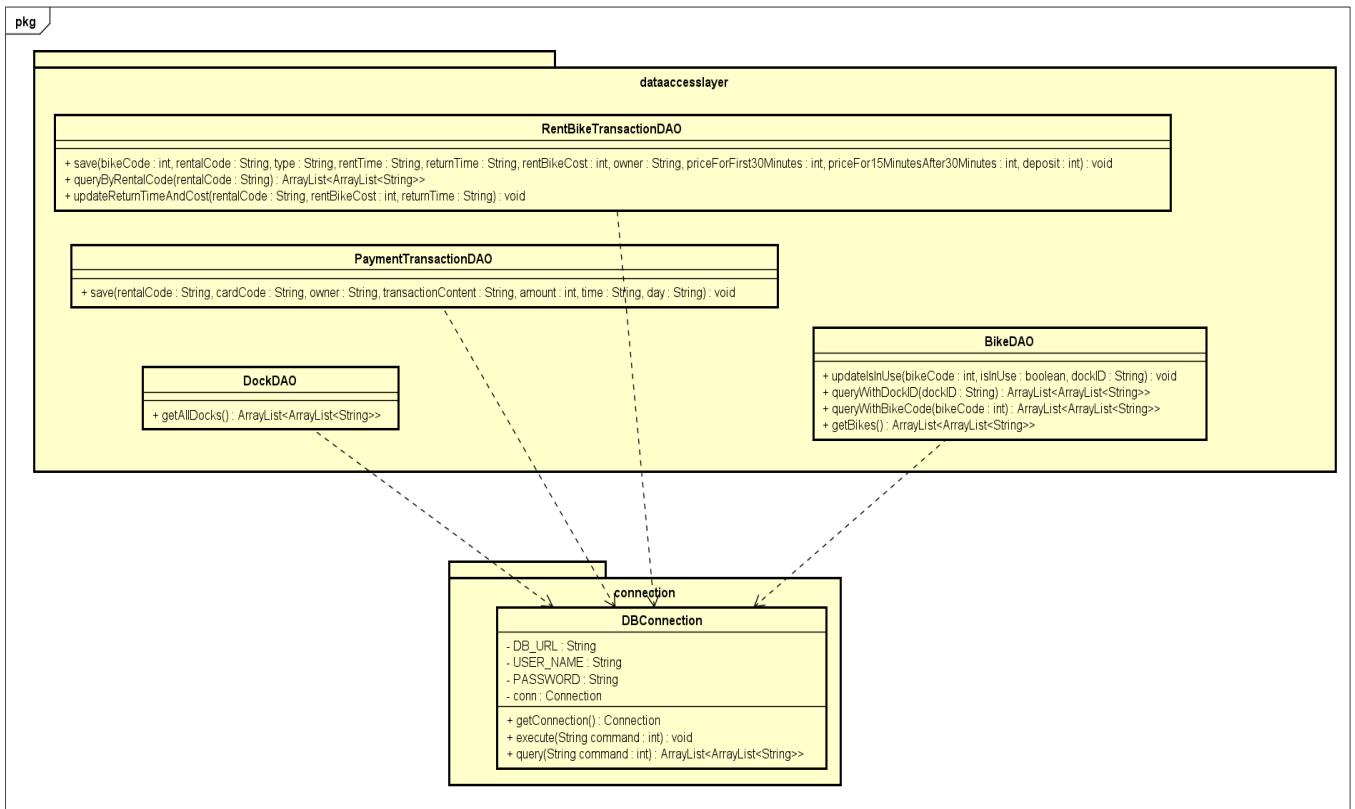
Hình 20 Thiết kế chi tiết gói “interbanksubsystem”

4.1.6 Thiết kế chi tiết subsystem “barcodeconvertersubsystem”



Hình 21 Thiết kế chi tiết gói “barcodeconvertersubsystem”

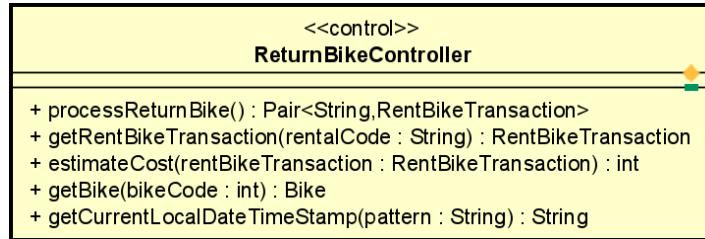
4.1.7 Thiết kế chi tiết gói “dataaccesslayer” và gói “connection”



Hình 22 Thiết kế chi tiết gói “dataaccesslayer” và gói “connection”

4.2 Thiết kế lớp chi tiết

4.2.1 Thiết kế lớp “ReturnBikeController”



Operation			
#	Name	Return Type	Description
1	processReturnBike	Pair<String, RentBineTransaction>	Xử lý giao dịch từ giao diện và trả về mã lỗi kèm theo giao dịch nếu thành công và null nếu không thành công
2	getRentBikeTransaction	RentBikeTransaction	Tìm kiếm giao dịch thuê xe trong cơ sở dữ liệu dựa theo mã thuê xe khi người dùng chọn trả xe
3	estimateCost	int	Tính toán chi phí thuê xe dựa theo thông tin trong giao dịch thuê xe
4	getBike	Bike	Lấy thông tin của xe từ cơ sở dữ liệu dựa theo bikeCode
5	getCurrentLocalDateTimeStamp	String	Lấy thời gian hiện theo theo mẫu cho trước

- Tham số:
 - rentalCode: mã thuê xe
 - rentBikeTransaction: giao dịch thuê xe
 - bikeCode: mã xe
 - pattern: định dạng thời gian
- Exception:

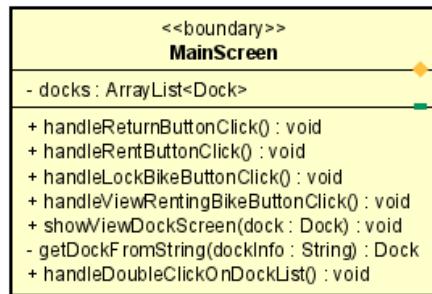
4.2.2 Thiết kế lớp “InitializeController”



Operation			
#	Name	Return Type	Description
1	getDocks	ArrayList<Dock>	Lấy danh sách tất cả các bến xe trong cơ sở dữ liệu
2	getBikes	ArrayList<Bike>	Lấy danh sách tất cả các xe trong bến xe theo ID
3	tableToBikes	ArrayList<Bike>	Chuyển kết quả truy vấn trả về dạng bảng thành danh sách các xe

- Tham số:
 - o dockID: mã bến xe
 - o bikeTable: mảng hai chiều danh sách các xe trong đó thông tin các thuộc tính được lưu dưới dạng String

4.2.3 Thiết kế lớp “MainScreen”



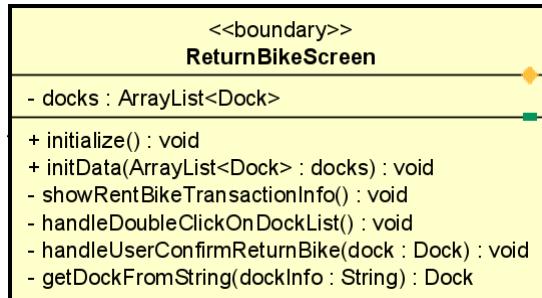
Attribute				
#	Name	Data type	Default value	Description
1	docks	ArrayList<Dock>	null	danh sách tất cả các bến xe trong cơ sở dữ liệu

Operation			
#	Name	Return Type	Description
1	handleReturnButtonClick	void	Xử lý yêu cầu trả xe của người dùng
2	handleRentButtonClick	void	Xử lý yêu cầu thuê xe của người dùng
3	handleLockBikeButtonClick	void	Xử lý yêu cầu khóa xe của người dùng
4	handleViewBikeRentingButtonClick	void	Xử lý yêu cầu xem thông tin chi tiết xe của người dùng
5	showViewDockScreen	void	Hiển thị màn hình chi tiết bến xe
6	getDockFromString	Dock	Tìm ra đối tượng bến xe tương ứng với click của người dùng
7	handleDoubleClickOnDockList	void	Xử lý yêu cầu xem thông tin chi tiết bến xe

- Tham số:

- dock: Đối tượng bãi giao diện dùng để hiển thị thông tin
- dockInfo: chuỗi trả về khi người dùng double clicks vào một bãi xe trong danh sách

4.2.4 Thiết kế lớp “ReturnBikeScreen”



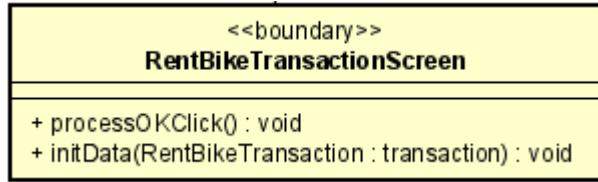
Attribute				
#	Name	Data type	Default value	Description
1	docks	ArrayList<Dock>	null	danh sách tất cả các bãi xe trong cơ sở dữ liệu

Operation			
#	Name	Return Type	Description
1	initialize	void	Hàm khởi tạo mặc định
2	initData	void	Khởi tạo danh sách các bãi xe
3	showRentBikeTransactionInfo	void	Hiển thị giao diện thông tin chi tiết giao dịch thuê xe
4	handleDoubleClickOnDockList	void	Xử lý yêu cầu trả xe vào vị trí bãi xe của người dùng
5	handleUserConfirmReturnBike	void	Gọi đến lớp controller để thực hiện yêu cầu trả xe của người dùng
6	getDockFromString	Dock	Tìm ra đối tượng bãi xe tương ứng với click của người dùng

- Tham số:

- docks: danh sách các xe dùng để hiển thị
- dock: Đối tượng bãi xe ứng với bãi xe người dùng muốn trả xe vào
- dockInfo: chuỗi trả về khi người dùng double clicks vào một bãi xe trong danh sách

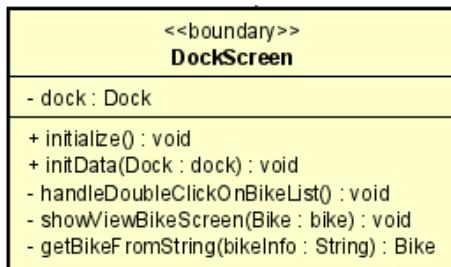
4.2.5 Thiết kế lớp “RentBikeTransactionScreen”



Operation			
#	Name	Return type	Description
1	processOKClick	void	đóng cửa sổ khi người dùng xác nhận OK
2	initData	void	Khởi tạo theo thông tin của giao dịch

- Tham số:
 - o transaction: đối tượng giao dịch thuê xe dùng để hiển thị

4.2.6 Thiết kế lớp “DockScreen”



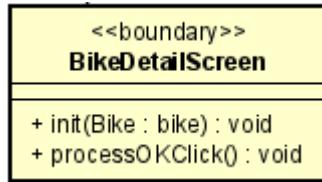
Attribute				
#	Name	Data type	Default value	Description
1	dock	Dock	null	đối tượng bãi xe cần hiển thị thông tin chi tiết

Operation				
#	Name	Return type	Description	
1	initialize	void	Hàm khởi tạo mặc định	
2	initData	void	Khởi tạo đối tượng bãi xe cho giao diện	
3	handleDoubleClickOnBikeList	void	Xử lý yêu cầu xem thông tin chi tiết 1 xe của người dùng	
4	showBikeViewScreen	void	Hiển thị giao diện thông tin chi tiết xe	
5	getBikeFromString	Bike	Tìm ra đối tượng xe tương ứng với click của người dùng	

- Tham số:
 - o dock: đối tượng bãi xe dùng để hiển thị thông tin chi tiết bãi xe
 - o bike: đối tượng xe dùng để hiển thị thông tin chi tiết xe

- bikeInfo: chuỗi trả về khi người dùng double clicks vào một xe trong danh sách

4.2.7 Thiết kế lớp “BikeDetailScreen”

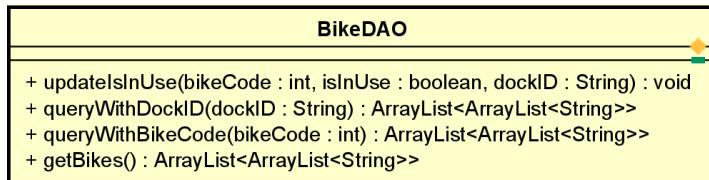


Operation			
#	Name	Return type	Description
1	init	void	Khởi tạo đối tượng xe cho giao diện
2	processOKClick	void	đóng cửa sổ khi người dùng xác nhận OK

- Tham số:

- bike: đối tượng xe dùng để hiển thị thông tin chi tiết xe

4.2.8 Thiết kế lớp “BikeDAO”



Operation			
#	Name	Return Type	Description
1	updateIsInUse	void	cập nhật trạng thái có đang sử dụng của xe và vị trí bến xe mới của xe
2	queryWithDockID	ArrayList<ArrayList<String>>	trả về danh sách xe ứng với dockID cho trước
3	queryWithBikeCode	ArrayList<ArrayList<String>>	trả về xe ứng với bikeCode cho trước
4	getBikes	ArrayList<ArrayList<String>>	lấy danh sách tất cả xe trong cơ sở dữ liệu

- Tham số:

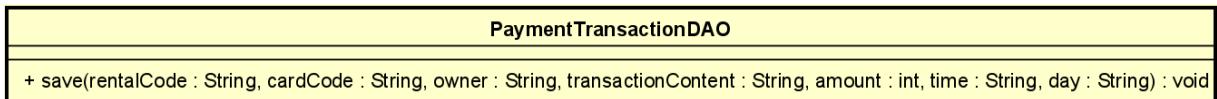
- bikeCode: mã xe
- isInUse: trạng thái có đang được sử dụng của xe
- dockID (updateIsInUse): ID bến xe mới mà xe được đỗ vào
- dockID (queryWithDockID): ID bến xe cần lấy danh sách các xe trong bến
- bikeCode (queryWithBikeCode): mã bến xe dùng để lấy xe tương ứng trong cơ sở dữ liệu

4.2.9 Thiết kế lớp “DockDAO”



Operation			
#	Name	Return Type	Description
1	getAllDocks	ArrayList<ArrayList<String>>	lấy danh sách tất cả bãi xe trong cơ sở dữ liệu

4.2.10 Thiết kế lớp “PaymentTransactionDAO”



Operation			
#	Name	Return Type	Description
1	save	void	lưu giao dịch vào cơ sở dữ liệu

- Tham số:

- o rentalCode: mã thuê xe
- o cardCode: mã thẻ
- o owner: chủ thẻ
- o transactionContent: nội dung giao dịch
- o amount: lượng tiền giao dịch
- o time: thời gian diễn ra giao dịch dưới dạng “hh-mm-ss”
- o day: thời gian diễn ra giao dịch dưới dạng “yyyy-MM-dd”

4.2.11 Thiết kế lớp “RentBikeTransactionDAO”



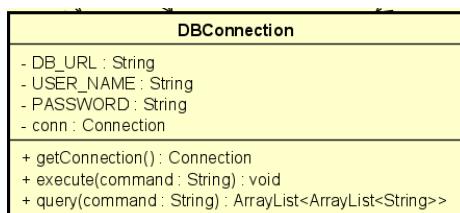
Operation			
#	Name	Return Type	Description
1	save	void	lưu giao dịch vào cơ sở dữ liệu
2	queryByRentalCode	ArrayList<ArrayList<String>>	lấy ra giao dịch thuê xe dựa theo rentalCode

3	updateReturnTimeAndCost	void	cập nhật thời gian trả xe và chi phí thuê xe cho giao dịch thuê xe vào cơ sở dữ liệu sau khi người dùng đã trả xe
---	-------------------------	------	---

- Tham số:

- o bikeCode: mã xe
- o rentalCode: mã giao dịch
- o type: loại xe
- o rentTime: thời điểm thuê xe
- o returnTime: thời điểm trả xe
- o rentBikeCost: chi phí thuê xe
- o owner: người thuê xe
- o priceForFirst30Minutes: giá thuê 30 phút đầu tiên
- o priceFor15MinutesAfter30Minutes: giá thuê mỗi 15 phút sau 30 phút đầu
- o deposit: tiền đặt cọc

4.2.12 Thiết kế lớp “DBConnection”



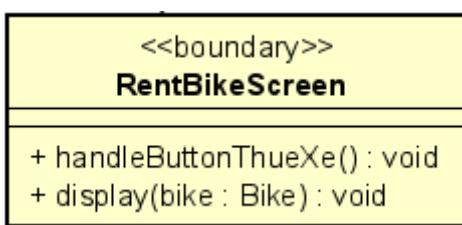
Attribute				
#	Name	Data type	Default value	Description
1	DB_URL	String	jdbc:mysql://localhost:3306/ecodatabase	đường dẫn đến cơ sở dữ liệu
2	USER_NAME	String	vutrunghnia	tên tài khoản
3	PASSWORF	String	*****	mật khẩu
4	conn	Connection	Được khởi tạo ngay từ đầu	kết nối đến cơ sở dữ liệu

Operation			
#	Name	Return Type	Description
1	getConnection	Connection	tạo kết nối với cơ sở dữ liệu
2	executeCommand	void	thực hiện một lệnh không lấy kết quả trả về

3	query	ArrayList<ArrayList<String>>	thực hiện một lệnh và lấy kết quả trả về
---	-------	------------------------------	--

- Tham số:
 - o command: lệnh cần thực thi

4.2.13 Thiết kế lớp “RentBikeScreen”



Operation			
#	Name	Return type	Description
1	handleButtonThueXe	void	xử lý yêu cầu khi khách hàng ấn nút thuê xe
2	display	void	hiển thị màn hình

* Parameter:

- Bike: đối tượng chứa thông tin xe được thuê

4.2.14 Thiết kế lớp “RentBikeInfoScreen”

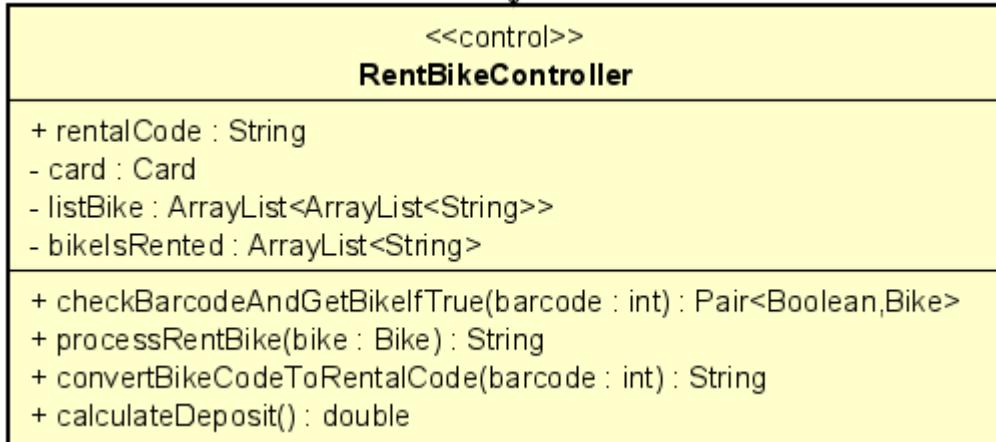


Operation			
#	Name	Return type	Description
1	init	void	khởi tạo các control màn hình

* Parameter:

- bike: Chứa thông tin xe tương ứng với barcode mà người dùng nhập

4.2.15 Thiết kế lớp “RentBikeController”



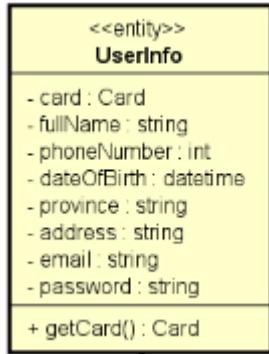
Attribute				
#	Name	Data type	Default value	Description
1	rentalCode	String	NULL	Mã thuê xe phục vụ cho lưu trữ cơ sở dữ liệu
2	card	Card	NULL	Chứa thông tin thẻ người dùng
3	listBike	ArrayList<ArrayList<String>>	NULL	Danh sách các xe của hệ thống
4	bikeIsRented	ArrayList<String>	NULL	Thông tin xe mà được thuê bởi người dùng

Operation			
#	Name	Return type	Description
1	checkBarcodeAndGetBikeIfTrue	Pair<Boolean, Bike>	Chuyển barcode thành bikeCode và kiểm tra trong cơ sở dữ liệu
2	processRentBike	String	Xử lý nghiệp vụ thuê xe
3	convertBikeCodeToRentalCode	String	Chuyển bikeCode thành rentalCode
4	calculateDeposit	double	Tính toán tiền cọc

* Parameter:

- barcode: mã vạch mà người dùng nhập vào màn hình
- bike: Lưu thông tin của xe

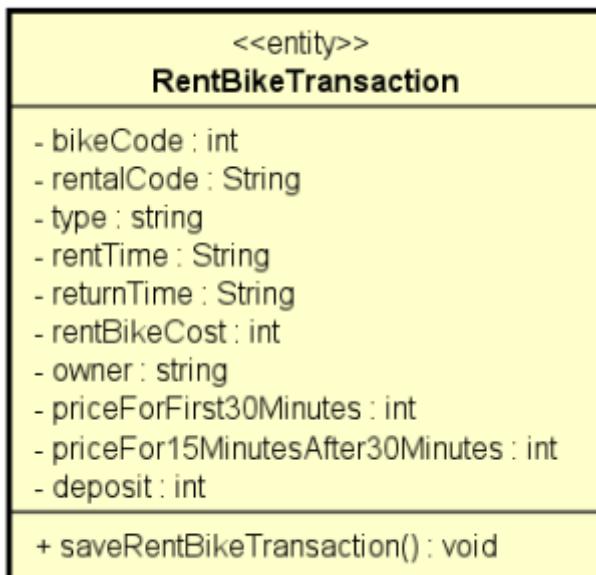
4.2.16 Thiết kế lớp “UserInfo”



Attribute				
#	Name	Data type	Default value	Description
1	card	Card	NULL	Chứa thông tin thẻ người dùng
2	fullName	String	NULL	Tên người dùng
3	phoneNumber	int	NULL	Số điện thoại
4	dateOfBirth	String	NULL	Ngày sinh
5	province	String	NULL	Tỉnh
6	address	String	NULL	Địa chỉ
7	email	String	NULL	Email
8	password	String	NULL	Mật khẩu

Operation			
#	Name	Return type	Description
1	getCard	Card	Lấy thông tin card của người dùng

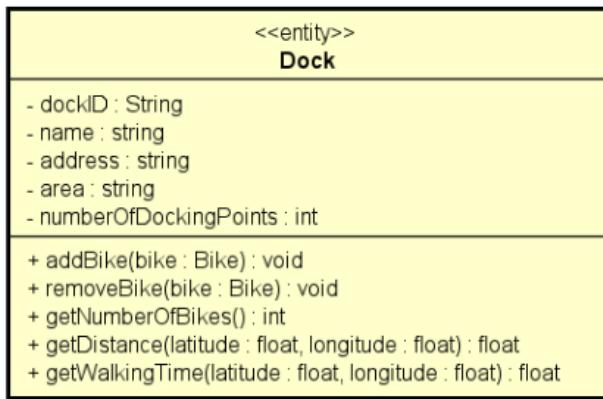
4.2.17 Thiết kế lớp “RentBikeTransaction”



Attribute				
#	Name	Data type	Default value	Description
1	bikeCode	int	NULL	Mã xe cho thuê
2	rentalCode	String	NULL	Mã thuê xe phục vụ lưu trữ cơ sở dữ liệu
3	type	String	NULL	Loại xe
4	rentTime	String	NULL	Thời gian bắt đầu thuê
5	returnTime	String	NULL	Thời gian trả xe
6	rentBikeCost	int	NULL	Gía tiền thuê xe
7	owner	String	NULL	Người sử dụng
8	priceFor30FirstMinutes	int	NULL	Gía thuê 30 phút đầu tiên
9	priceFor15MinutesAfter30Minutes	int	NULL	Gía thuê mỗi 15 phút sau đó
10	deposit	int	NULL	Tiền cọc

Operation			
#	Name	Return type	Description
1	saveRentBikeTransaciton	void	Lưu giao dịch thuê xe

4.2.18 Thiết kế lớp “Dock”



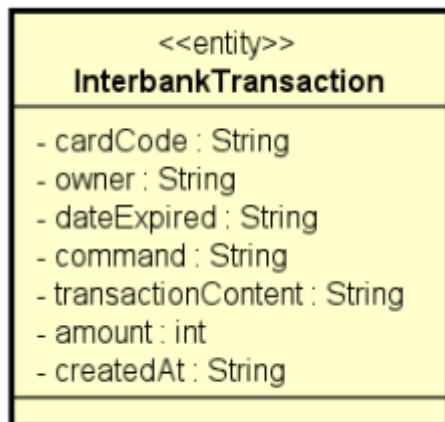
Attribute				
#	Name	Data type	Default value	Description
1	dockID	String	NULL	Mã dock
2	name	String	NULL	Tên bãi xe
3	address	String	NULL	Địa chỉ
4	numberOfDockingPoints	int	NULL	số lượng vị trí xe trong bãi
5	area	String	NULL	Khu vực bãi xe

Operation			
#	Name	Return type	Description
1	addBike	void	Thêm xe vào dock
2	removeBike	void	Xóa xe khỏi dock
3	getNumberOfBikes	int	Lấy số lượng Bike
4	getDistance	float	Thông tin khoảng cách từ người dùng tới dock
5	getWalkingTime	float	Thời gian di chuyển tới dock

* Parameter:

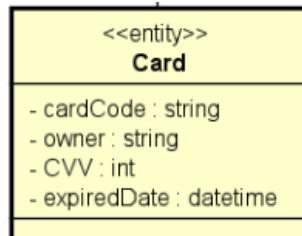
- bike: Lưu thông tin xe
- lattitude: vĩ độ của người dùng
- longitude: kinh độ của người dùng

4.2.19 Thiết kế lớp “InterbankTransaction”



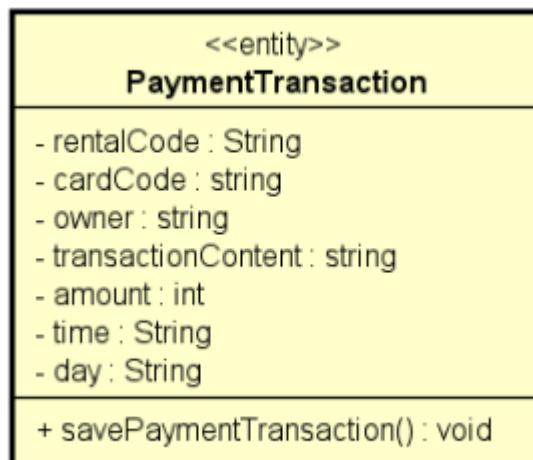
Attribute				
#	Name	Data type	Default value	Description
1	cardCode	String	NULL	Mã thẻ
2	owner	String	NULL	Người dùng
3	dateExpired	String	NULL	Ngày hết hạn thẻ
4	command	String	NULL	Yêu cầu về giao dịch
5	transactionContent	String	NULL	Nội dung giao dịch
6	amount	int	NULL	Số tiền cần giao dịch
7	createdAt	String	NULL	Ngày tạo

4.2.20 Thiết kế lớp “Card”



Attribute				
#	Name	Data type	Default value	Description
1	cardCode	String	118131_group8_2020	Mã thẻ
2	owner	String	Group 8	Người dùng
3	CVV	String	427	Mã xác minh thẻ
4	expiredDate	String	1125	Ngày hết hạn

4.2.21 Thiết kế lớp “PaymentTransaction”



Attribute				
#	Name	Data type	Default value	Description
1	rentalCode	String	NULL	mã thuê xe phục vụ lưu trữ cơ sở dữ liệu
2	cardCode	String	NULL	mã thẻ
3	owner	String	NULL	người sở hữu thẻ
4	transactionContent	String	NULL	nội dung giao dịch
5	amount	int	NULL	số tiền giao dịch
6	time	String	NULL	thời gian giao dịch
7	day	String	NULL	ngày giao dịch

Operation			
#	Name	Return type	Description
1	savePaymentTransaction	void	Lưu giao dịch thanh toán vào cơ sở dữ liệu

4.2.22 Thiết kế lớp “Bike”

<<entity>> Bike	
- isInUse : Boolean - bikeCode : int - type : String - value : int - priceForFirst30Minutes : int - priceFor15MinutesAfter30Minutes : int - remainBattery : int - maxTime : float - licensePlate : String	
+ updateInUseAndDockID(isInUse : Boolean, dockID : String) : void + getGeneralInfo() : String	

Attribute				
#	Name	Data type	Default value	Description
1	isInUse	String	NULL	Mã thuê xe phục vụ cho lưu trữ cơ sở dữ liệu
2	bikeCode	int	NULL	Chứa thông tin thẻ người dùng
3	type	String	NULL	Danh sách các xe của hệ thống
4	value	int	NULL	Thông tin xe mà được thuê bởi người dùng
5	priceFor30FirstMinutes	int	NULL	Gía thuê xe 30 phút đầu tiên
6	priceFor15MinutesAfter30Minutes	int	NULL	Gía thuê xe mỗi 15 phút sau 30 phút đầu tiên
7	remainBattery	int	NULL	Lượng pin còn lại
8	maxTime	String	NULL	Thời gian sử dụng tối đa
9	licensePlate	String	NULL	Biển số xe

Operation			
#	Name	Return type	Description
1	getGeneralInfo	String	Lấy thông tin chung của xe
2	updateInUseAndDockID	void	update xe đang sử dụng và bãi xe khi trả

* Parameter:

- isInUse: Biến trạng thái xe có đang được thuê hay không
- dockID: mã bãi xe mà xe đang được đặt

4.2.23 Thiết kế lớp “IInterbank”

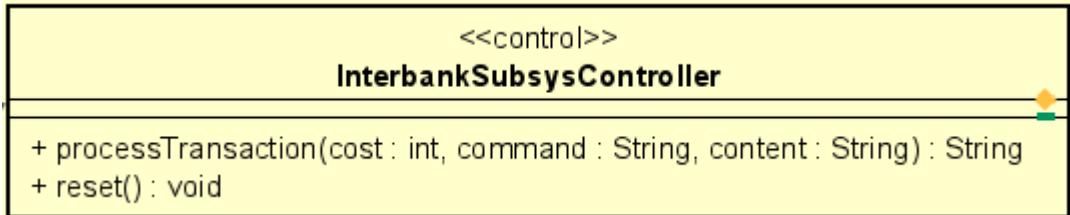


Operation			
#	Name	Return type	Description
1	processTransaciton	String	Xử lý giao dịch theo yêu cầu của hệ thống
2	reset	void	reset tiền trong thẻ

* Parameter:

- cost: giá tiền cần giao dịch
- command: yêu cầu của giao dịch
- content: nội dung giao dịch

4.2.24 Thiết kế lớp “InterbankSubsysController”

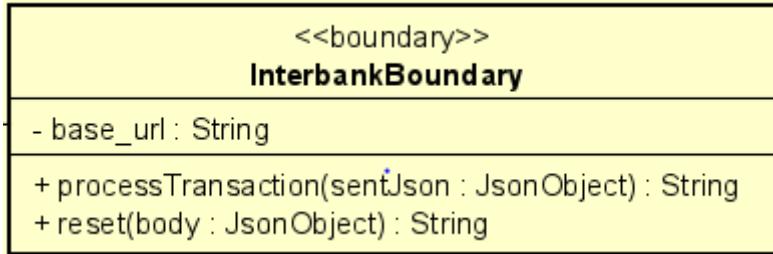


Operation			
#	Name	Return type	Description
1	processTransaciton	String	Xử lý giao dịch theo yêu cầu của hệ thống
2	reset	void	reset tiền trong thẻ

* Parameter:

- cost: giá tiền cần giao dịch
- command: yêu cầu của giao dịch
- content: nội dung giao dịch

4.2.25 Thiết kế lớp “InterbankBoundary”



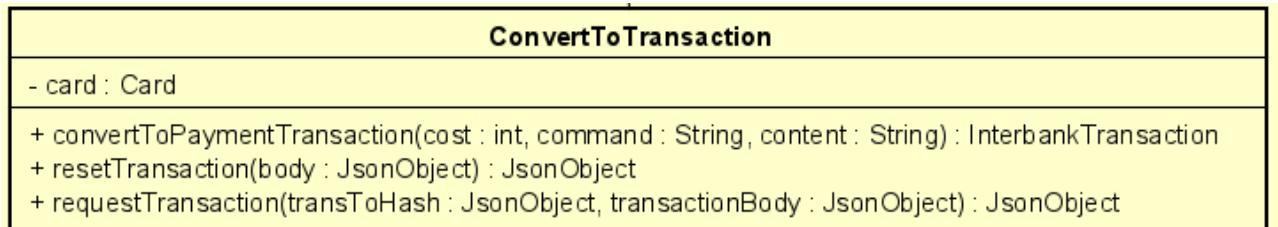
Attribute				
#	Name	Data type	Default value	Description
1	base_url	String	NULL	https://ecopark-system-api.herokuapp.com/

Operation			
#	Name	Return type	Description
1	processTransaciton	String	Xử lý giao dịch theo yêu cầu của hệ thống
2	reset	String	reset tiền trong thẻ

* Parameter:

- `sentJson`: đối tượng `JsonObject` chưa thông tin request
- `body`: đối tượng `JsonObject` đối tượng chưa thông tin request

4.2.26 Thiết kế lớp “ConvertToTransaction”



Attribute				
#	Name	Data type	Default value	Description
1	card	Card	NULL	Chứa thông tin thẻ người dùng

Operation			
#	Name	Return type	Description
1	convertToPaymentTransaction	InterbankTransaction	Chuyển thành format của giao dịch thanh toán
2	resetTransaction	JsonObject	Chuyển thành format của giao dịch reset
3	requestTransaction	JsonObject	Tạo request theo format ứng với api

* Parameter:

- transToHash: đối tượng Json chứa thông tin cần được hash
- transactionBody: đối tượng JsonObject chứa thông tin của transaction
- body: đối tượng JsonObject đối tượng chứa thông tin request
- cost: giá tiền cần giao dịch
- command: yêu cầu về giao dịch
- content: nội dung giao dịch

4.2.27 Thiết kế lớp “HttpConnector” (của Interbank Subsystem)



Operation			
#	Name	Return type	Description
1	sendPatch	String	Gửi kết nối lên api

* Parameter:

- url: đường dẫn tới API
- body: đối tượng JsonObject đối tượng chứa thông tin request gửi lên API

4.2.28 Thiết kế lớp “IBarcodeConverter”



Operation			
#	Name	Return type	Description
1	convertBarcodeToBikeCode	int	Chuyển mã vạch thành mã xe

* Parameter:

- barcode: mã vạch của xe mà người dùng nhập vào

4.2.29 Thiết kế lớp “BarcodeConverterController”

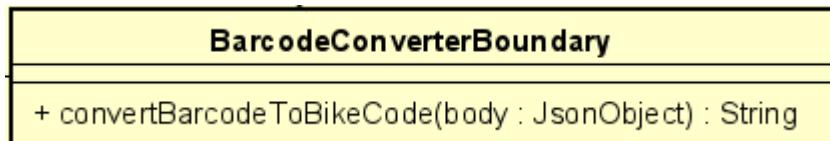


Operation			
#	Name	Return type	Description
1	convertBarcodeToBikeCode	int	Chuyển mã vạch thành mã xe

* Parameter:

- barcode: mã vạch của xe mà người dùng nhập vào

4.2.30 Thiết kế lớp “BarcodeConverterBoundary”



Operation			
#	Name	Return type	Description
1	convertBarcodeToBikeCode	String	Chuyển mã vạch thành mã xe

* Parameter:

- body: chứa thông tin request gửi lên API

4.2.31 Thiết kế lớp “HttpConnector” (của Barcode Converter Subsystem)



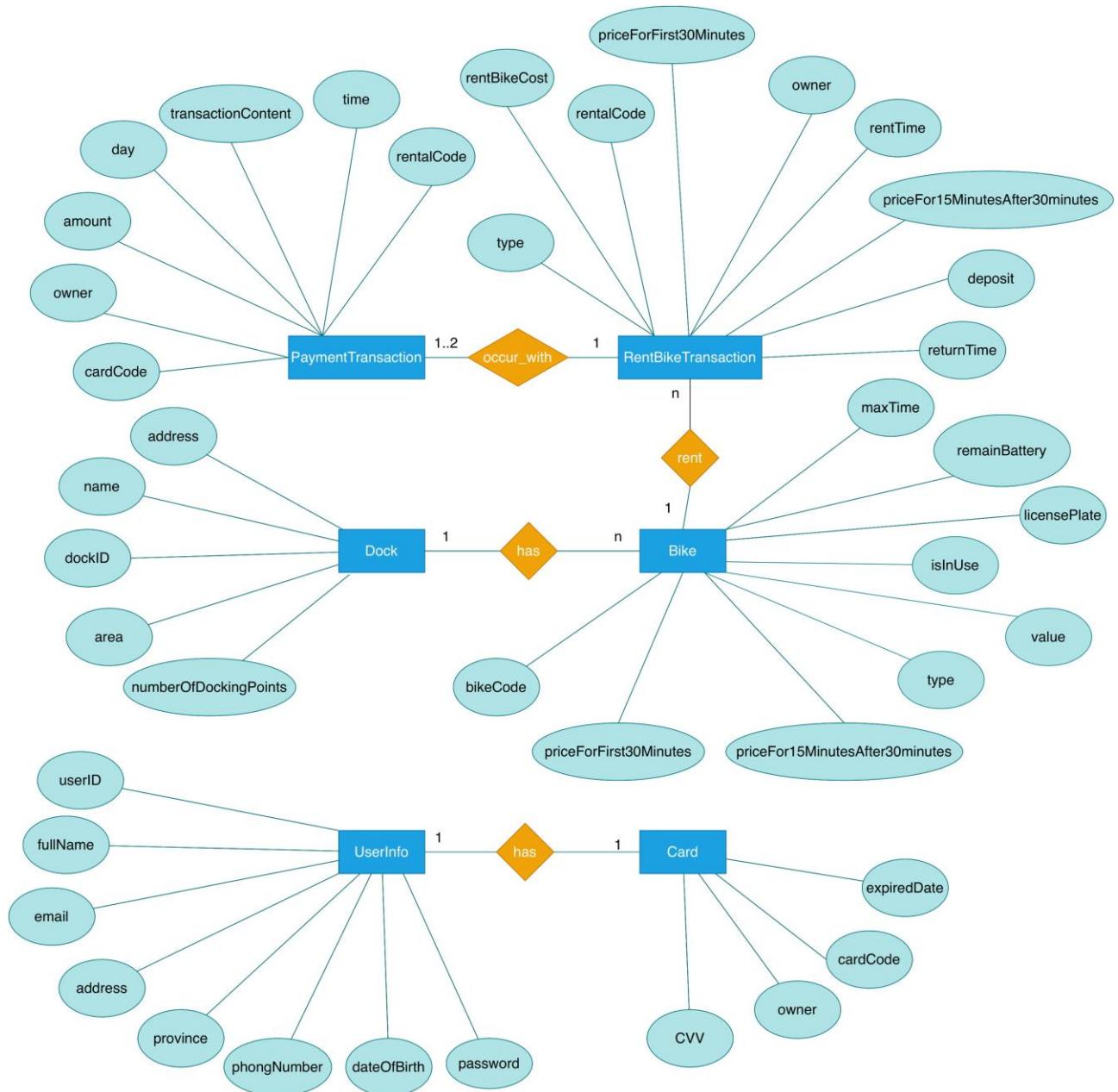
Operation			
#	Name	Return type	Description
1	post	String	Thực hiện gửi kết nối lên api

* Parameter:

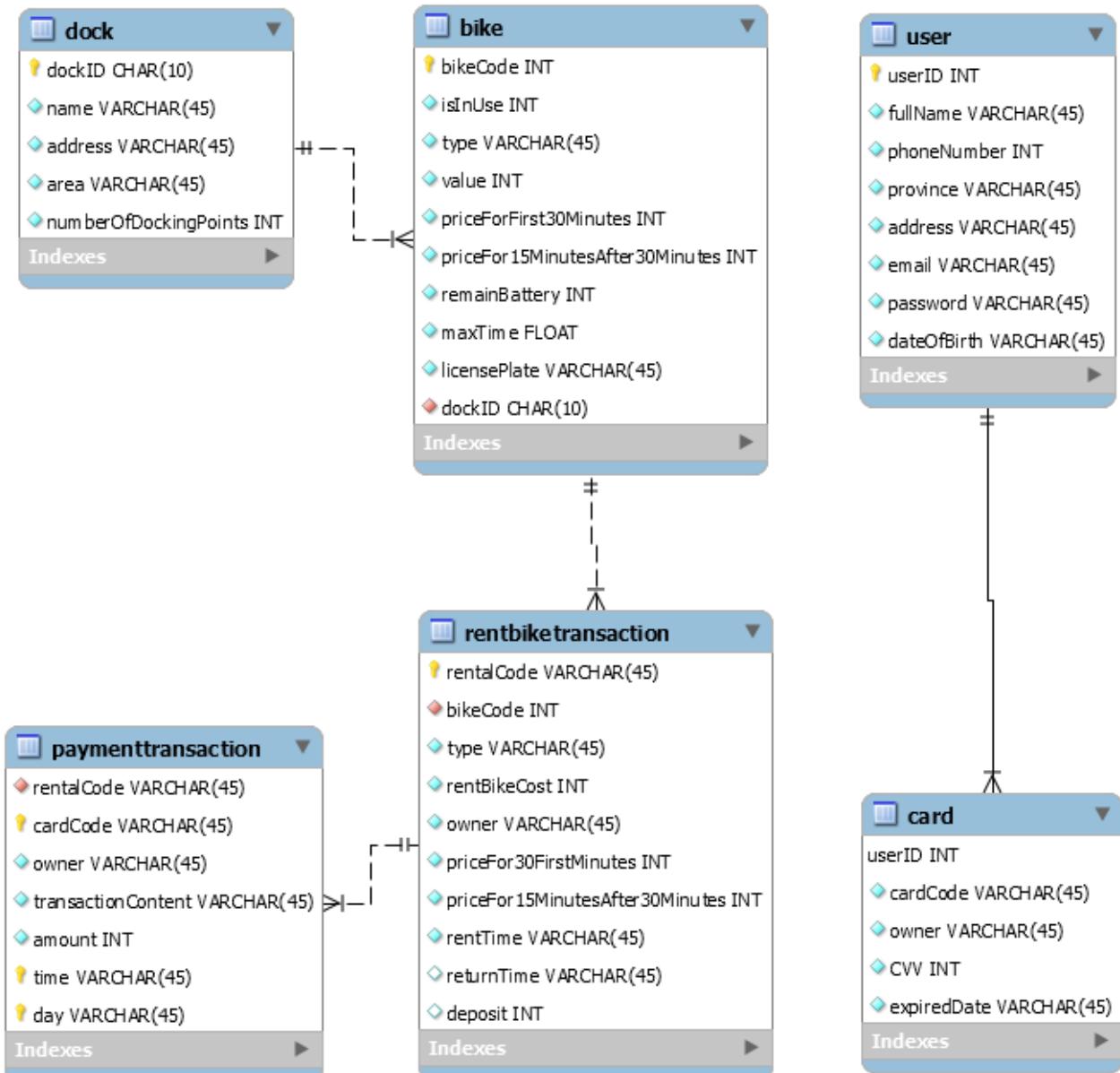
- url: chứa thông tin request gửi lên API
- body: chứa thông tin nội dung request gửi lên API

5 Thiết kế mô hình dữ liệu

5.1 Mô hình dữ liệu mức khái niệm



5.2 Mô hình dữ liệu mức logic



5.3 Thiết kế chi tiết

5.3.1 Thành phần 'Dock':

dock						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1	X		dockID	CHAR(10)	Có	ID của bãi xe
2			name	VARCHAR(45)	Có	tên bãi xe
3			address	VARCHAR(45)	Có	địa chỉ bãi xe
4			area	VARCHAR(45)	Có	khu vực của bãi xe
5			numberOfDockingPoints	INT	Có	số vị trí đỗ xe tối đa của bãi xe

5.3.2 Thành phần 'bike':

bike						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1	X		bikeCode	INT	Có	mã bãi xe
2			isInUse	INT	Có	bằng 1 nếu đang sử dụng và bằng 0 nếu không sử dụng
3			type	VARCHAR(45)	Có	loại xe
4			value	INT	Có	giá xe
5			priceForFirst30Minutes	INT	Có	giá thuê 30 phút đầu
6			priceFor15MinutesAfter30Minutes	INT	Có	giá thuê 15 phút sau 30 phút đầu
7			remainBattery	INT	Có	lượng pin còn lại (với xe điện)
8			maxTime	FLOAT	Có	thời gian sử dụng tối đa (với xe điện)
9			licensePlate	VARCHAR(45)	Có	biển số xe
10		X	dockID	CHAR(10)	Có	vị trí bãi xe của xe hiện tại (chỉ có ý nghĩa với xe đang không được sử dụng)

5.3.3 Thành phần 'user':

user						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1	X		userID	INT	Có	ID của người dùng
2			fullName	VARCHAR(45)	Có	tên đầy đủ
3			phoneNumber	INT	Có	số điện thoại
4			province	VARCHAR(45)	Có	tỉnh/thành phố
5			address	VARCHAR(45)	Có	địa chỉ
6			email	VARCHAR(45)	Có	email
7			password	VARCHAR(45)	Có	mật khẩu
8			dataOfBirth	VARCHAR(45)	Có	ngày sinh

5.3.4 Thành phần 'paymenttransaction':

paymenttransaction						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1		X	rentalCode	VARCHAR(45)	Có	mã thuê xe
2	X		cardCode	VARCHAR(45)	Có	mã thẻ
3			owner	VARCHAR(45)	Có	chủ thẻ
4			transactionContent	VARCHAR(45)	Có	nội dung giao dịch
5			amount	INT	Có	lượng tiền giao dịch
6	X		time	VARCHAR(45)	Có	thời gian giao dịch (hh-mm-ss)
7	X		day	VARCHAR(45)	Có	thời gian giao dịch (yyyy-MM-dd)

5.3.5 Thành phần 'rentbiketransaction':

rentbiketransaction						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1	X		rentalCode	CHAR(10)	Có	mã thuê xe
2		X	bikeCode	INT	Có	mã xe được thuê
3			type	VARCHAR(45)	Có	loại xe
4			rentBikeCost	INT	Không	chi phí thuê xe (khui chưa trả xe thì đặt là -1)
5			owner	VARCHAR(45)	Có	người thuê
6			priceForFirst30Minutes	INT	Có	giá thuê 30 phút đầu
7			priceFor15MinutesAfter30Minutes	INT	Có	giá thuê 15 phút sau 30 phút đầu
8			rentTime	VARCHAR(45)	Có	thời gian thuê
9			returnTime	VARCHAR(45)	Không	thời gian trả (khi chưa trả thì đặt là "")
10			deposit	INT	Có	tiền đặt cọc

5.3.6 Thành phần 'card':

card						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1	X	X	userID	INT	Có	ID của chủ thẻ
2			cardCode	VARCHAR(45)	Có	mã thẻ
3			owner	VARCHAR(45)	Có	tên chủ thẻ
4			CVV	INT	Có	mã CVV
5			expiredDate	VARCHAR(45)	Có	ngày hết hạn

6 Đánh giá bản thiết kế

6.1 Mục tiêu và định hướng

Mục tiêu:

- Mang đến trải nghiệm tốt với người dùng

Định hướng:

- Sử dụng chuẩn java trong khi lập trình, các lớp và hàm được thiết kế theo nguyên lý hướng đối tượng
- Tránh hash code và việc lưu dữ liệu trong mã nguồn
- Giải thích code đầy đủ và chi tiết, viết java doc cho các hàm và các lớp phục vụ cho việc bảo trì sau này

6.2 Chiến lược thiết kế

Hệ thống được thiết kế với ba mục tiêu chính: dễ mở rộng, dễ bảo trì và tái sử dụng mã nguồn. Các công nghệ được sử dụng gồm có:

- Ngôn ngữ lập trình: Java 11
- Hệ quản trị cơ sở dữ liệu: MySQL
- IDE: IntelliJ

Trong khi thiết kế và cài đặt, việc tối ưu hóa bộ nhớ và hiệu năng cũng được xem xét chi tiết và đầy đủ.

6.3 Coupling and Cohesion

6.3.1 Phân tích tính Coupling và Cohesion cho gói “controller”

6.3.1.1 Phân tích tính Cohesion cho từng lớp

Lớp *InitializeController* thỏa mãn Functional Cohesion vì: Nhiệm vụ của class này là đọc dữ liệu từ database và trả về danh sách các bến xe để *MainScreen* hiển thị khi chương trình bắt đầu. Để thực hiện nhiệm vụ này, lớp có một hàm chính là *getDocks*, hàm này có nhiệm vụ đọc danh sách các bến xe, để lấy danh sách xe của một bến xe, *getDocks* gọi đến hàm *getBikes*, trong hàm *getBikes*, để chuyển dữ liệu dạng mảng hai chiều về danh sách các bến xe, *getBikes* gọi đến hàm *tableToBikes*. Như vậy ba hàm *getDocks*, *getBikes* và *tableToBikes* có các chức năng riêng biệt nhưng cũng đồng thời đóng góp vào nhiệm vụ chung của lớp.

```
public class InitializeController {  
    /**  
     * Lấy tất cả các bến xe từ cơ sở dữ liệu và chuyển thành dãy tương ứng  
     * @return danh sách các bến xe  
     */  
    public static ArrayList<Dock> getDocks(){  
        ArrayList<Dock> docks = new ArrayList<>();  
        ArrayList<String>> dockTable = Dock.getDockTable();  
        for(ArrayList<String> row: dockTable){  
            String dockID = row.get(0);  
            String name = row.get(1);  
            String address = row.get(2);  
            String area = row.get(3);  
            int numberOfDockingPoints = Integer.parseInt(row.get(4));  
            ArrayList<Bike> bikes = getBikes(dockID);  
            Dock dock = new Dock(dockID, name, address, area, numberOfDockingPoints, bikes);  
            docks.add(dock);  
        }  
        return docks;  
    }  
  
    /**  
     * Lấy danh sách các xe trong một bến xe cho trước từ cơ sở dữ liệu  
     * @param dockID: ID của bến xe  
     * @return danh sách xe  
     */  
    public static ArrayList<Bike> getBikes(String dockID){  
        ArrayList<ArrayList<String>> bikeTable = BikeDAO.queryWithDockID(dockID);  
        return InitializeController.tableToBikes(bikeTable);  
    }  
  
    /**  
     * Chuyển kết quả dạng bảng sau khi query về danh sách các xe  
     * @param bikeTable: mảng hai chiều lưu dưới dạng ArrayList<ArrayList<String>>  
     * @return danh sách các xe  
     */  
    public static ArrayList<Bike> tableToBikes(ArrayList<ArrayList<String>> bikeTable){...}
```

Lớp *ReturnBikeController* thỏa mãn tính Functional Cohesion vì: Nhiệm vụ của class này là xử lý yêu cầu trả xe của người dùng, việc thực thi được chia làm các bước nhỏ:

- Lấy giao dịch thuê xe từ cơ sở dữ liệu
- Tính toán chi phí thuê xe
- Gửi yêu cầu interbanksubsystem để thực hiện giao dịch
- Nếu thành công:
 - o Cập nhật chi phí thuê xe và thời gian trả xe vào giao dịch thuê xe
 - o Cập nhật trạng thái của xe
 - o Lưu lại giao dịch với ngân hàng vào cơ sở dữ liệu
 - o Trả về giao dịch thuê xe và mã code tương ứng để hiển thị cho người dùng
- Nếu không thành công:
 - o Trả về null và mã code tương ứng để hiển thị cho người dùng

Các bước trên đều nằm trong hàm chính *processReturnBike*. Để lấy giao dịch từ cơ sở dữ liệu, *processReturnBike* gọi đến hàm *getRentBikeTransaction*, để tính toán chi phí thuê xe gọi đến hàm *estimateCost*, để lấy thời gian trả xe theo định dạng gọi đến hàm *getCurrentLocalDateTimeStamp*, để cập nhật trạng thái xe thì cần tìm được xe tương ứng trong cơ sở dữ liệu, gọi đến hàm *getBike* để thực hiện điều này.

```
public class ReturnBikeController {
    private static final String PATTERN = "yyyy-MM-dd HH:mm:ss";

    /**
     * Xử lý yêu cầu trả xe của người dùng
     * @return RentBikeTransaction nếu thành công và null nếu thất bại
     */
    public static Pair<String, RentBikeTransaction> processReturnBike(){...}

    /**
     * Lấy thông tin giao dịch từ cơ sở dữ liệu dựa trên mã thuê xe
     *
     * @param rentalCode: mã thuê xe
     * @return Đối tượng RentBikeTransaction
     */
    public static RentBikeTransaction getRentBikeTransaction(String rentalCode){...}

    /**
     * Tính toán chi phí thuê xe từ thông tin trong giao dịch thuê xe
     *
     * @param rentBikeTransaction: Giao dịch thuê xe
     * @return Chi phí tính toán
     */
    public static int estimateCost(RentBikeTransaction rentBikeTransaction, boolean isTest){...}

    /**
     * Lấy thông tin của xe từ cơ sở dữ liệu dựa theo bikeCode
     *
     * @param bikeCode: bikeCode của xe
     * @return ArrayList<String> là một mảng các thuộc tính của xe
     */
    public static Bike getBike(int bikeCode){...}

    /**
     * Lấy thời gian hiện tại và chuyển đổi thành format
     *
     * @param pattern: format thời gian
     * @return Thời gian hiện tại theo format thời gian phía trên
     */
    public static String getCurrentLocalDateTimeStamp(String pattern) {...}
}
```

Lớp *RentBikeController* thỏa mãn tính Informational Cohesion vì: Nhiệm vụ của class là thực hiện xử lý yêu cầu thuê xe của người dùng. Việc thực thi được chia thành hai phase riêng biệt là kiểm tra barcode người dùng nhập vào có hợp lệ hay không lấy thông tin xe tương ứng nếu hợp lệ, nếu hợp lệ thì tiếp tục thực hiện phase thứ 2 để xử lý yêu cầu, cả hai phase đều tương tác với dữ liệu là các thuộc tính trong lớp *RentBikeController*. Cụ thể như sau:

- Hàm *checkBarcodeAndGetBikeIfTrue* kiểm tra barcode có hợp lệ hay không, nếu hợp lệ thì tìm xe tương ứng trong danh sách *listBike*.
- Hàm *processRentBike* thực hiện xử lý giao dịch sau khi đã kiểm tra barcode hợp lệ và có đối tượng xe tương ứng. Cụ thể gồm các bước:
 - o Tính toán tiền đặt cọc
 - o Chuyển *bikeCode* thành *rentalCode* và đặt lại giá trị thuộc tính *rentalCode*
 - o Gửi yêu cầu thực hiện giao dịch trừ tiền đến *interbanksussystem*
 - o Nếu thành công:
 - Tạo và lưu giao dịch thuê xe
 - Tạo và lưu giao dịch trừ tiền với ngân hàng
 - Cập nhật trạng thái xe
 - o Nếu không thành công:
 - Đặt lại *rentalCode* về rỗng

Để tính toán tiền cọc, gọi đến hàm *calculateDeposit*, để chuyển *bikeCode* thành *rentalCode* gọi đến hàm *converterBikeCodeToRentalCode*, để chuyển barcode thành *bikeCode* gọi đến *barcodeconvertersubsystem*.

```
public class RentBikeController {
    /**
     * rentalCode : Mã thuê xe được dùng cho toàn bộ những giao dịch và phiên thuê xe hiện tại.
     *             Phiên thuê xe khác nhau sẽ có rentalCode khác nhau.
     */
    public static String rentalCode = "";
    private static final Card card = Card.getInstance();
    private static ArrayList<ArrayList<String>> listBike = BikeDAO.getBikes();
    private static ArrayList<String> bikeIsRented;
    private static int bikeCode;

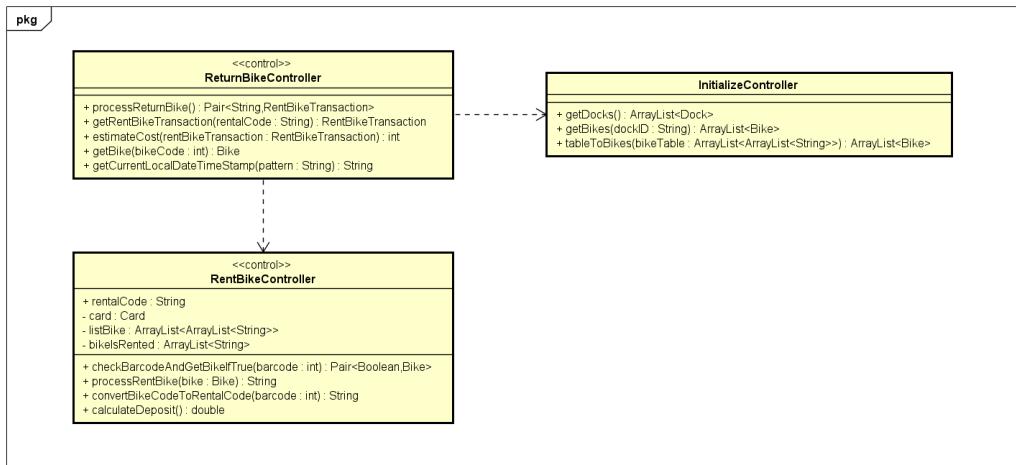
    /**
     *
     * @param barcode : mã xe
     * @return <mã check bikeCode, thông tin xe (nếu bikeCode đúng)>
     */
    public static Pair<Boolean, Bike> checkBarcodeAndGetBikeIfTrue(int barcode){...}

    /**
     * Xử lý giao dịch thuê xe
     * Nếu giao dịch thành công thì sẽ tiến hành lưu lại giao dịch thanh toán,
     * thông tin phiên thuê xe và cập nhật xe thành đang sử dụng.
     *
     * Nếu giao dịch thất bại thì sẽ đưa ra thông báo lỗi và không lưu lại thông tin.
     */
    public static String processRentBike(Bike bike){...}

    /**
     *
     * @param bikeCode
     * Sinh ra rental code cho phiên thuê xe
     * @return rental code
     */
    public static String convertBikeCodeToRentalCode(int bikeCode){...}

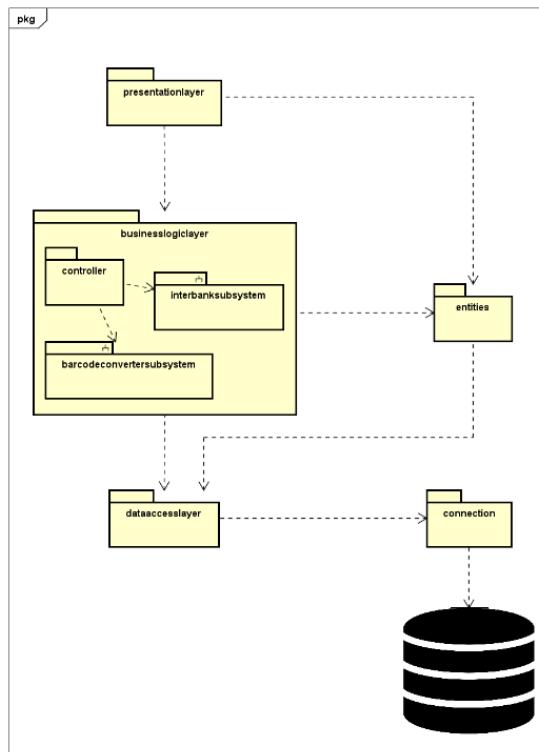
    /**
     * @return giá trị tiền đặt cọc = 40% giá trị xe
     */
    public static double calculateDeposit() { return Integer.parseInt(bikeIsRented.get(3)) * 0.4; }
}
```

6.3.1.2 Phân tích tính Coupling giữa các lớp controller



- Để tái sử dụng code, *ReturnBikeController* gọi hàm *tableToBikes* để chuyển danh sách các bãi xe dạng mảng hai chiều thành dãy danh sách đối tượng các xe. Đây là phụ thuộc data coupling do hai lớp chỉ giao tiếp với nhau thông qua dữ liệu
- Sự phụ thuộc giữa *ReturnBikeController* và *RentBikeController* là data coupling vì *ReturnBikeController* chỉ sử dụng biến toàn cục *rentalCode* của class *RentBikeController*

6.3.2 Phân tích Coupling giữa các gói trong toàn hệ thống



Mỗi gói trong hệ thống có nhiệm vụ riêng biệt và việc phụ thuộc từ trên xuống dưới, không có gói ở tầng dưới phụ thuộc vào tầng trên, không có phụ thuộc vòng lặp.