

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Tài liệu thiết kế phần mềm

ECOBIKE

Môn: Thiết kế xây dựng phần mềm

Giảng viên hướng dẫn: TS. Nguyễn Thị Thu Trang

Nhóm 9

Nguyễn Quốc Hào – 20194043

Nguyễn Ngọc Bảo – 20193989

Cao Như Đạt - 20194013

Hanoi, 2/2023

Mục lục

Mục lục	1
1 Giới thiệu.....	9
1.1 Mục tiêu	9
1.2 Phạm vi.....	9
1.3 Từ điển thuật ngữ.....	9
1.4 Tài liệu tham khảo	10
2 Tổng quan về tài liệu.....	11
2.1 Mô tả chung	11
2.1.1 Các tác nhân	11
2.1.2 Biểu đồ use case tổng quan.....	11
2.2 Các vấn đề	12
2.2.1 Một số giả thiết	12
2.2.2 Một số ràng buộc	12
2.2.3 Một số rủi ro.....	13
3 Thiết kế kiến trúc hệ thống.....	14
3.1 Mẫu thiết kế kiến trúc	14
3.2 Biểu đồ tuần tự	14
3.2.1 UC001 – Xem thông tin chi tiết xe	14
3.2.2 UC002 – Thuê xe	16
3.2.3 UC003 – Trả xe.....	17
3.3 Biểu đồ lớp phân tích.....	18
3.3.1 Biểu đồ lớp phân tích cho UC001	18
3.3.2 Biểu đồ lớp phân tích cho UC002.....	19
3.3.3 Biểu đồ lớp phân tích cho UC003.....	20
3.4 Biểu đồ lớp phân tích gộp.....	21
3.5 Kiến trúc an toàn thông tin cho hệ thống.....	21

4	Thiết kế chi tiết.....	23
4.1	Thiết kế giao diện người dùng.....	23
4.1.1	Chuẩn hóa cấu hình màn hình.....	23
4.1.2	Biểu đồ dịch chuyển màn hình.....	24
4.1.3	Đặc tả màn hình	25
4.2	Mô hình hóa dữ liệu.....	39
4.2.1	Mô hình hóa dữ liệu dạng khái niệm (ERD).....	39
4.2.2	Thiết kế cơ sở dữ liệu.....	39
4.3	Hệ quản trị các tệp phi cơ sở dữ liệu	42
4.4	Thiết kế lớp.....	42
4.4.1	Biểu đồ lớp tổng quan.....	42
4.4.2	Biểu đồ lớp chi tiết.....	45
4.4.3	Class Design.....	51
5	Đánh giá bản thiết kế.....	91
5.1	Mục tiêu và định hướng.....	91
5.2	Chiến lược thiết kế.....	91
5.3	Đánh giá bản thiết kế theo các cấp độ Coupling & Cohesion	91
5.3.1	Đánh giá theo các cấp độ Coupling	91
5.4	Đánh giá theo các cấp độ Cohesion.....	96
5.5	Các nguyên lý trong thiết kế.....	102
5.5.1	Single Responsibility Principle.....	102
5.5.2	Open/Closed Principle	103
5.5.3	Liskov Substitution Principle.....	103
5.5.4	Interface Segregation	104
5.5.5	Dependency Inversion	104
5.6	Các mẫu thiết kế (Design Pattern).....	105

Danh mục hình ảnh

Hình 2-1: Biểu đồ use case tổng quan.....	11
Hình 3-1: Kiến trúc MVC tổng quan	14
Hình 3-2: Biểu đồ tuần tự cho UC001	15
Hình 3-3: Biểu đồ tuần tự cho UC002	16
Hình 3-4: Biểu đồ phân tích lớp UC001	18
Hình 3-5: Biểu đồ lớp phân tích UC002	19
Hình 4-1: Biểu đồ dịch chuyển màn hình	24
Hình 4-2: Biểu đồ ER cho cơ sở dữ liệu đề xuất	39
Hình 4-3: Biểu đồ cơ sở dữ liệu sử dụng MySQL80	40
Hình 4-4: Biểu đồ thể hiện sự phụ thuộc của các gói	43
Hình 4-5: Biểu đồ thể hiện sự liên quan của các lớp	44
Hình 4-6: Biểu đồ lớp chi tiết cho gói entity	45
Hình 4-7: Biểu đồ gói cho lớp util	46
Hình 4-8: Biểu đồ lớp chi tiết cho gói exception	47
Hình 4-9: Biểu đồ lớp chi tiết cho gói controller	48
Hình 4-10: Biểu đồ lớp chi tiết cho gói view.screen.....	49
Hình 4-11: Thiết kế lớp cho subsystem Barcode Converter	50
Hình 4-12: Thiết kế lớp chi tiết cho subsystem Interbank	50
Hình 4-13: Biểu đồ lớp RentbikeController.....	51
Hình 4-14: Biểu đồ lớp BarcodeController.....	52
Hình 4-15: Biểu đồ lớp ReturnBikeController.....	52
Hình 4-16: Biểu đồ lớp CardController	53
Hình 4-17: Biểu đồ lớp DockController	54
Hình 4-18: Biểu đồ lớp BarcodeConverter	55
Hình 4-19: Biểu đồ lớp IBarcodeConverter.....	55
Hình 4-20: Biểu đồ lớp BarcodeConversion	56
Hình 4-21: Biểu đồ lớp Message (của gói subsystem.barcodeconverter).....	57

Hình 4-22: Biểu đồ lớp IInterbank.....	57
Hình 4-23: Biểu đồ lớp Interbank.....	58
Hình 4-24: Biểu đồ lớp Message (của gói subsystem.interbank)	59
Hình 4-25: Biểu đồ lớp ResetTransaction.....	60
Hình 4-26: Biểu đồ lớp ProcessTransaction	61
Hình 4-27: Biểu đồ lớp HashFunction.....	62
Hình 4-28: Chuyển đổi InterbankTransaction thành đối tượng Json.....	63
Hình 4-29: Biểu đồ lớp HTTPBinder	63
Hình 4-30: Biểu đồ lớp Bike.....	64
Hình 4-31: Biểu đồ lớp Card.....	65
Hình 4-32: Biểu đồ lớp Dock.....	66
Hình 4-33: Biểu đồ lớp DoubleNormalBike.....	68
Hình 4-34: Biểu đồ lớp InterbankTransaction	68
Hình 4-35: Biểu đồ lớp PaymentTransaction	69
Hình 4-36: Biểu đồ lớp RentBikeInvoice	71
Hình 4-37: Biểu đồ lớp SingleElectricBike	72
Hình 4-38: Biểu đồ lớp SingleNormalBike	73
Hình 4-39: Biểu đồ lớp BikeDAO	73
Hình 4-40: Biểu đồ lớp DBBinder.....	74
Hình 4-41: Biểu đồ lớp DockDAO	75
Hình 4-42: Biểu đồ lớp PaymentTransactionDAO.....	76
Hình 4-43: Biểu đồ lớp RentBikeInvoiceDAO.....	77
Hình 4-44: Biểu đồ lớp BikeDetailScreen	77
Hình 4-45: Biểu đồ lớp CardInfoScreen	78
Hình 4-46: Biểu đồ trạng thái cho các Text Field.....	79
Hình 4-47: Biểu đồ lớp CardInfoScreenForReturnBike	80
Hình 4-48: Biểu đồ lớp DockListScreen.....	81
Hình 4-49: Biểu đồ lớp DockScreen.....	82
Hình 4-50: Biểu đồ lớp Dock.....	83

Hình 4-51: Biểu đồ lớp mainScreen	83
Hình 4-52: Biểu đồ lớp RentalBikeScreen.....	85
Hình 4-53: Biểu đồ lớp RentBikeInfoScreen.....	86
Hình 4-54: Biểu đồ lớp RentBikeInvoiceScreen	87
Hình 4-55: Biểu đồ lớp RentBikeScreen	88
Hình 4-56: Biểu đồ lớp ReturnBikeScreen	89
Hình 5-1: Common coupling trong các lớp view	92
Hình 5-2: Thay đổi biến global docks bởi phương thức do DockController quản lý	92
Hình 5-3: Control coupling trong HandleException.....	93
Hình 5-4: Control coupling trong phương thức processReturnBike trong ReturnBikeController	93
Hình 5-5: Stamp coupling trong ReturnBikeController.....	94
Hình 5-6: Stamp coupling trong RentBikeController	94
Hình 5-7: Stamp coupling trong InterbankTransaction	95
Hình 5-8: Ví dụ về data coupling trong validateCardInfo trong CardController.....	95
Hình 5-9: 2 phương thức patch và post trong HTTPBinder	97
Hình 5-10: Redesign HTTPBinder	97
Hình 5-11: Temporal cohesion trong DockListScreen	98
Hình 5-12: Các giai đoạn triển khai của CardController	99
Hình 5-13: Hàm xử lý băm giao dịch	100
Hình 5-14: Prototype Interface của Interbank.....	101
Hình 5-15: Triển khai của lớp thực thi Interface IInterbank.....	101
Hình 5-16: Mỗi controller trong gói controller phụ trách duy nhất một chức năng	102
Hình 5-17: Phương thức calculateRentCost của các lớp kế thừa Bike	102
Hình 5-18: Thay thế Card bởi lớp trừu tượng.....	103
Hình 5-19: Các phương thức trong IInterbank.....	104
Hình 5-20: InterbankTransaction phụ thuộc vào lớp Card	105
Hình 5-21: Thay sự kết nối Card bằng GenericCard	105
Hình 5-22: Strategy để thực hiện tính giá thuê xe	106

Hình 5-23: Factory với lớp Bike.....	106
Hình 5-24: Singleton với lớp Card.....	106

Danh mục bảng

Bảng 4-1: Đặc tả chi tiết bảng Dock	40
Bảng 4-2: Đặc tả chi tiết bảng Bike	41
Bảng 4-3: Đặc tả chi tiết bảng PaymentTransaction.....	41
Bảng 4-4: Đặc tả chi tiết bảng RentBikeInvoice	42
Bảng 4-5: Đặc tả các thuộc tính của RentbikeController	51
Bảng 4-6: Đặc tả các phương thức của RentbikeController	51
Bảng 4-7: Đặc tả các thuộc tính của BarcodeController.....	52
Bảng 4-8: Đặc tả các phương thức của BarcodeController	52
Bảng 4-9: Đặc tả các phương thức của ReturnBikeController	53
Bảng 4-10: Đặc tả các phương thức của CardController	54
Bảng 4-11: Đặc tả các phương thức của DockController	54
Bảng 4-12: Đặc tả các phương thức của BarcodeConverter	55
Bảng 4-13: Đặc tả các phương thức của IBarcodeConverter	56
Bảng 4-14: Đặc tả các thuộc tính của BarcodeConverter	56
Bảng 4-15: Đặc tả các phương thức của BarcodeConversion	56
Bảng 4-16: Đặc tả các phương thức của Message (của gói subsystem.barcodeconverter)	57
Bảng 4-17: Đặc tả các phương thức của IInterbank	58
Bảng 4-18: Đặc tả phương thức của Interbank	58
Bảng 4-19: Đặc tả các phương thức của Message (của gói subsystem.interbank)	59
Bảng 4-20: Đặc tả các thuộc tính của ResetTransaction.....	60
Bảng 4-21: Đặc tả các phương thức của ResetTransaction	60
Bảng 4-22: Đặc tả các thuộc tính của ProcessTransaction	61
Bảng 4-23: Đặc tả các phương thức của ProcessTransaction	61
Bảng 4-24: Đặc tả các phương thức của HashFunction.....	62

Bảng 4-25: Đặc tả các phương thức của HTTPBinder	63
Bảng 4-26: Đặc tả các thuộc tính của Bike.....	64
Bảng 4-27: Đặc tả các phương thức của Bike.....	65
Bảng 4-28: Đặc tả các thuộc tính của Card.....	65
Bảng 4-29: Đặc tả các phương thức của Card	66
Bảng 4-30: Đặc tả các thuộc tính của Dock.....	67
Bảng 4-31: Đặc tả các phương thức của Dock	67
Bảng 4-32: Đặc tả các phương thức của DoubleNormalBike.....	68
Bảng 4-33: Đặc tả các thuộc tính của InterbankTransaction	69
Bảng 4-34: Đặc tả các phương thức của InterbankTransaction	69
Bảng 4-35: Đặc tả các thuộc tính của PaymentTransaction	70
Bảng 4-36: Đặc tả các phương thức của PaymentTransaction	70
Bảng 4-37: Đặc tả các thuộc tính của RentBikeInvoice	71
Bảng 4-38: Đặc tả các phương thức của RentBikeInvoice	71
Bảng 4-39: Đặc tả các thuộc tính của SingleElectricBike	72
Bảng 4-40: Đặc tả các phương thức của SingleElectricBike	72
Bảng 4-41: Đặc tả các phương thức của SingleNormalBike	73
Bảng 4-42: Đặc tả các phương thức của BikeDAO	74
Bảng 4-43: Đặc tả các thuộc tính của DBBinder.....	75
Bảng 4-44: Đặc tả các phương thức của DBBinder.....	75
Bảng 4-45: Đặc tả các phương thức của DockDAO.....	75
Bảng 4-46: Đặc tả các phương thức của PaymentTransactionDAO.....	76
Bảng 4-47: Đặc tả các phương thức của RentBikeInvoiceDAO	77
Bảng 4-48: Đặc tả các thuộc tính của BikeDetailScreen	78
Bảng 4-49: Đặc tả các phương thức của BikeDetailScreen	78
Bảng 4-50: Đặc tả các thuộc tính của CardInfoScreen	79
Bảng 4-51: Đặc tả các phương thức của CardInfoScreen.....	79
Bảng 4-52: Đặc tả các thuộc tính của CardInfoScreenForReturnBike	80
Bảng 4-53: Đặc tả các phương thức của CardInfoScreenForReturnBike	80

Bảng 4-54: Đặc tả các thuộc tính của DockListScreen.....	81
Bảng 4-55: Đặc tả các phương thức của DockListScreen	81
Bảng 4-56: Đặc tả các thuộc tính của DockScreen.....	82
Bảng 4-57: Đặc tả các phương thức của DockScreen.....	82
Bảng 4-58: Đặc tả các thuộc tính của Lobby	83
Bảng 4-59: Đặc tả các phương thức của Lobby.....	83
Bảng 4-60: Đặc tả các thuộc tính của MainScreen	84
Bảng 4-61: Đặc tả các phương thức của MainScreen.....	84
Bảng 4-62: Đặc tả các thuộc tính của RentalBikeScreen.....	85
Bảng 4-63: Đặc tả các phương thức của RentalBikeScreen	85
Bảng 4-64: Đặc tả các thuộc tính của RentBikeInfoScreen.....	86
Bảng 4-65: Đặc tả các phương thức của RentBikeInfoScreen.....	86
Bảng 4-66: Đặc tả các thuộc tính của RentBikeInvoiceScreen	87
Bảng 4-67: Đặc tả các phương thức của RentBikeInvoiceScreen	87
Bảng 4-68: Đặc tả các thuộc tính của RentBikeScreen	88
Bảng 4-69: Đặc tả các phương thức của RentBikeScreen	88
Bảng 4-70: Đặc tả các thuộc tính của ReturnBikeScreen	89
Bảng 4-71: Đặc tả các phương thức của ReturnBikeScreen.....	89

1 Giới thiệu

1.1 Mục tiêu

Tài liệu này thể hiện 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 cho. 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

Hệ thống Ecobike là hệ thống cho thuê xe được sử dụng nội bộ trong khu đô thị Ecopark. Hệ thống này phục vụ 24/7 đối với người dùng cũng như có thể hỗ trợ lên đến 100 người cùng với đó là khả năng chịu lỗi lên đến 200 giờ. Nhờ có hệ thống mà việc thuê xe ở khu đô thị sẽ diễn ra nhanh chóng, thuận tiện và tự động hơn.

Hệ thống Ecobike thực sự có thể đem lại những lợi ích gì cho người dùng? Thứ nhất, khi sử dụng hệ thống, người dùng có thể biết được vị trí, cũng như thông tin các bãi xe, không chỉ vậy, hệ thống còn cung cấp cả dữ liệu các loại xe trong bãi. Sau khi biết được vị trí, thông tin bãi xe, người dùng trực tiếp tới bãi xe để chọn xe ưng ý. Thứ hai, người dùng có thể thực hiện thuê xe thông qua ứng dụng được cài trên điện thoại bằng cách cung cấp mã vạch của xe và thực hiện thanh toán điện tử ngay trên ứng dụng. Các giao dịch được diễn ra trực tuyến, nhanh chóng và bảo mật giữa người dùng với thẻ ngân hàng đã được liên kết khi sử dụng, qua đó tiết kiệm thời gian, thủ tục thuê xe, cũng như diễn ra một cách tự động. Thứ ba, khi sử dụng xe, người dùng có thể xem được thông tin xe đang thuê (pin) và quan trọng hơn là biết được số tiền mình đang phải trả tại thời điểm hiện tại và thậm chí có thể tạm dừng việc thuê xe. Cuối cùng, người dùng có thể sử dụng phần mềm để chọn cho mình bãi trả xe mong muốn.

Đối với hệ thống giả lập mà tài liệu này mô tả sẽ không yêu cầu định danh người dùng, khi tham gia vào các giao dịch, người sử dụng sẽ nhập thông tin thẻ tín dụng.

1.3 Từ điển thuật ngữ

STT	Thuật ngữ	Giải thích	Ví dụ	Ghi chú
1	Token	Một phần dữ liệu được tạo ở phía server ra chứa thông tin về người dùng và mã token. Token được sử dụng để xác	JSON Web Token (JWT)	Token được thiết kế nhỏ gọn, an toàn

		thực người dùng khi muốn đăng nhập.		
2	User	Người sử dụng hệ thống, có thể hiểu là người thuê xe hoặc cư dân muốn thuê xe trong khu đô thị Ecopark.	Nguyễn Quốc Hào	
3	Interbank	Ngân hàng liên kết, là đối tác của phần mềm, đã được ký kết các hợp đồng kinh tế đòi thực và có thể tham gia trong quá trình giao dịch.	Vietinbank, Techcombank, MSBank, MBBank,..	
4	Thẻ tín dụng (Card)	Các thẻ ngân hàng hợp pháp đòi thực của các ngân hàng liên kết mà người dùng sở hữu. Trong quá trình giao dịch, thẻ tín dụng sẽ đóng vai trò trung gian. Các API trong quá trình xây dựng phần mềm sẽ thao tác trực tiếp trên các thẻ tín dụng này, thay vì các ngân hàng liên kết để tăng khả năng tự động và tiết kiệm thời gian, chi phí.	9653 523 071	
5	Dock	Bãi xe, địa điểm tương ứng với điểm chứa những xe đạp trong thực tế (lấy xe và trả xe)	Bãi xe Hoàng Mai	
6	Bike	Xe đạp		

1.4 Tài liệu tham khảo

[1] Tài liệu đặc tả: EcoBikeRental-ProblemStatement-VI, Nguyễn Thị Thu Trang

2 Tổng quan về tài liệu

2.1 Mô tả chung

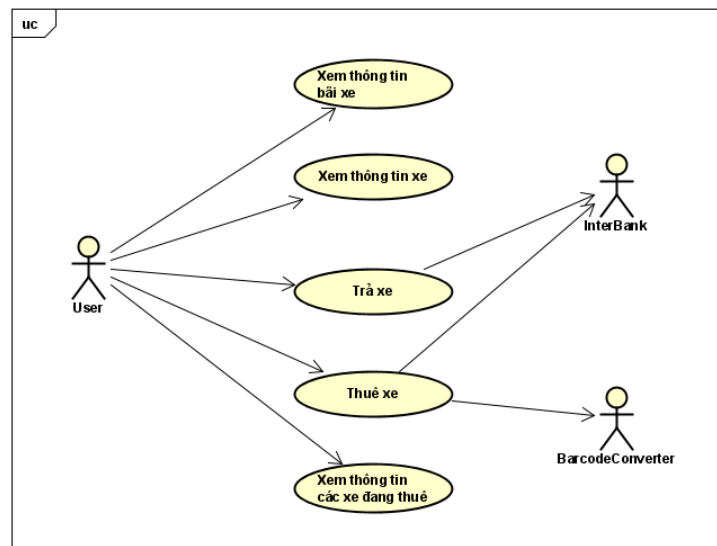
2.1.1 Các tác nhân

Phần mềm có các tác nhân chính là người dùng, Interbank và BarcodeConverter. Hệ thống còn một tác nhân khác là khách, tuy nhiên phạm vi môn học không tập trung đến nên không đưa vào trong báo cáo này. Người dùng là khách sau khi đã đăng nhập thành công vào hệ thống.

2.1.2 Biểu đồ use case tổng quan

Khi chưa đăng nhập, khách có thể đăng ký tài khoản mới, đăng nhập, yêu cầu thiết lập lại mật khẩu khi quên mật khẩu, và thiết lập lại mật khẩu khi nhận được chỉ dẫn thiết lập qua email. Khi khách đăng nhập thành công, hệ thống tạo ra menu chứa các chức năng tương ứng với nhóm người dùng mà người dùng đó thuộc về. Khi khách yêu cầu thiết lập lại mật khẩu, hệ thống thực hiện tạo token và gửi chỉ dẫn thiết lập lại mật khẩu qua email. Tuy nhiên trong phạm vi của môn học, tài liệu này không phân tích và thiết kế cho use case trên.

Sau khi đăng nhập, người dùng có thể tiến hành thuê xe, xem thông tin chi tiết về bãi xe, xem thông tin chi tiết về xe, xem thông tin về xe đang thuê và trả xe



Hình 2-1: Biểu đồ use case tổng quan

2.2 Các vấn đề

2.2.1 Một số giả thiết

Trong khuôn khổ tài liệu này, phần mềm được triển khai chạy trên máy địa phương (local) và có kết nối Internet tới các API được cung cấp. Để có thể xây dựng được một hệ thống chạy địa phương cần một số những nền tảng sau:

- Nền tảng về phần cứng: máy tính core i5 – 1135G7, hệ điều hành Windows 11, hỗ trợ truy nhập SSD vào bộ nhớ ngoài và hỗ trợ kết nối Internet.
- Nền tảng về phần mềm: Ngôn ngữ lập trình Java với nền tảng IDE là Eclipse. Phần mềm hỗ trợ JavaFX là SceneBuilder. Cơ sở dữ liệu được sử dụng là MySQL 8.0. Một số thư viện được sử dụng thêm trong quá trình phát triển phần mềm bao gồm: okhttp3 (để kết nối HTTP), json (để đóng gói các dữ liệu JSON), log4j (nền tảng phổ biến để log), ...

Đối với đặc tả đầy đủ của phần mềm thì yêu cầu phải được triển khai thành một server và các client có thể truy cập vào được bằng các phiên đăng nhập có xác thực, vì vậy, phần cứng trong tương lai có thể được nâng cấp để có thể phục vụ được. Quá trình phát triển phần mềm không phụ thuộc quá nhiều vào hệ điều hành nên hệ điều hành thay đổi trong tương lai cũng không phải vấn đề. Các giả thiết về truy nhập bộ nhớ ngoài và kết nối Internet cũng không phải vấn đề to lớn. Vấn đề cốt lõi nhất mà cần phải được chú ý đó là nền tảng về phần mềm: MySQL không phải là cơ sở dữ liệu có tính an toàn thông tin cao nên trong tương lai có thể bị thay thế, các công nghệ DAO được sử dụng trong quá trình phát triển phần mềm cần được thay thế để đáp ứng, vì vậy, quá trình phát triển phần mềm cần lưu ý điều này. Các thư viện để phát triển các tính năng có tính thủ tục như okhttp3, json, ... cũng có thể được thay thế trong tương lai nên quá trình phát triển phần mềm cần phải thiết kế các giao diện (Interface) và đảm bảo nguyên lý Open/Close trong SOLID nhằm việc thay đổi trong tương lai sẽ diễn ra suôn sẻ.

2.2.2 Một số ràng buộc

Về cơ bản, hệ thống được triển khai trên máy địa phương nên tốc độ xử lý và tính toán là ở mức chấp nhận được, quy trình sử dụng phần mềm cũng đơn giản, có GUI thân thiện với người sử dụng. Các giới hạn về phần cứng hầu như không có ảnh hưởng nhiều tới quá trình phần mềm do lượng tài nguyên cần sử dụng cũng không đòi hỏi quá nhiều.

Tuy nhiên, có một ràng buộc khá lớn đối với phần mềm đó là nghiệp vụ của phần mềm bị ràng buộc bởi các API bên ngoài. Trước hết, tính ổn định trong quá trình sử dụng phần mềm được quy định bởi tính ổn định của API: nếu server bên ngoài không đáp ứng được nghiệp vụ thì phần mềm có thể chạy sai hoặc không đáp ứng được nghiệp vụ. Tương tự, tính đáp ứng của hệ thống cũng được quy định bởi thời gian xử lý của server và tốc độ kết nối đường truyền. Việc chậm trễ trong quá trình truyền tin có thể dẫn đến trải nghiệm

của người dùng không được tốt. Cuối cùng, tính an toàn của hệ thống đặc biệt phụ thuộc vào các API bên ngoài. Các kết nối HTTPS luôn tiềm ẩn những rủi ro lớn về an toàn thông tin. Chính vì vậy, quá trình phát triển phần mềm cũng cần lưu ý việc để đảm bảo hệ thống phần mềm không bị tấn công.

2.2.3 Một số rủi ro

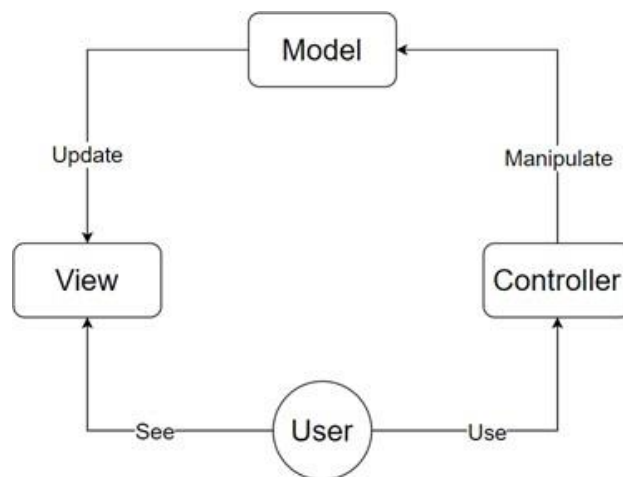
Như đã phân tích ở phần trước, kết nối HTTPS với server bên ngoài không đảm bảo hệ thống chúng ta luôn ở trong trạng thái an toàn. Thứ nhất, việc trao đổi các gói tin qua mạng Internet rất dễ dàng bị nghe lén nên chúng ta cần phải có cơ chế để che giấu dữ liệu với bên ngoài. Có nhiều cách để che giấu dữ liệu, ví dụ như là sử dụng hàm băm, khóa công khai, khóa đối xứng, ... Các phương án này cần được xem xét trong quá trình phát triển phần mềm. Thứ hai, có thể có các tấn công giả mạo (phishing) để làm kẻ trung gian thao túng các nghiệp vụ. Do vậy, cần có các cơ chế xác thực, các giải pháp có thể tính đến bao gồm: Xác thực bằng Kerberos, Needham-Schroeder, ... Bên cạnh đó, các tấn công điển hình khác cũng có thể xảy ra đó là DDoS với server bên ngoài làm cho hệ thống không thể kết nối tới các API để xử lý nghiệp vụ và một trường hợp phải được đặc biệt lưu ý, đó là SQL Injection. Vì hệ thống sử dụng MySQL và chỉ có một tầng bảo mật đó là xác thực danh tính người dùng thông qua mật khẩu nên rất dễ bị SQL Injection. Một cuộc tấn công SQL Injection có thể làm sai số toàn bộ dữ liệu hệ thống và rất khó để ngăn chặn. Chính vì vậy, cần tính tới các phương án để ngăn chặn việc này xảy ra và thậm chí có thể phải tính đến thay thế cơ sở dữ liệu bảo mật hơn để đảm bảo tính toàn vẹn cho hệ thống.

3 Thiết kế kiến trúc hệ thống

3.1 Mẫu thiết kế kiến trúc

Hệ thống của nhóm sử dụng kiến trúc MVC (Model-View-Controller).

- Model chứa dữ liệu và các tính toán xử lý logic để giải quyết vấn đề mà phần mềm hướng tới (business logic). Thành phần model thường được trình bày ở dạng Domain Model. Model thông thường sẽ cung cấp thêm một số những phương thức để có thể thực hiện truy vấn vào cơ sở dữ liệu (DAO).
- View là thành phần đảm nhận trình bày từ những dữ liệu của Model. View bao gồm những gì thể hiện trên màn hình như các GUI, form, widget, thông báo, ...
- Controller là thành phần đảm nhận việc xử lý đáp trả lại các dữ liệu được đưa vào từ người dùng như các sự kiện chuột, bàn phím, các tương tác lên các control... Controller là cầu nối giữa người sử dụng và hệ thống bên trong.

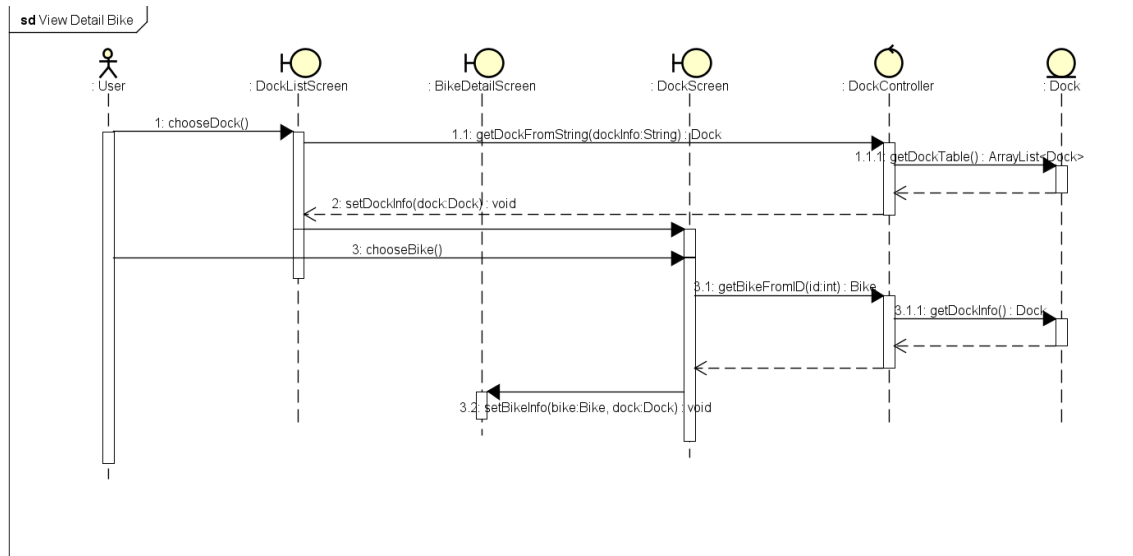


Hình 3-1: Kiến trúc MVC tổng quan

3.2 Biểu đồ tuần tự

3.2.1 UC001 – Xem thông tin chi tiết xe

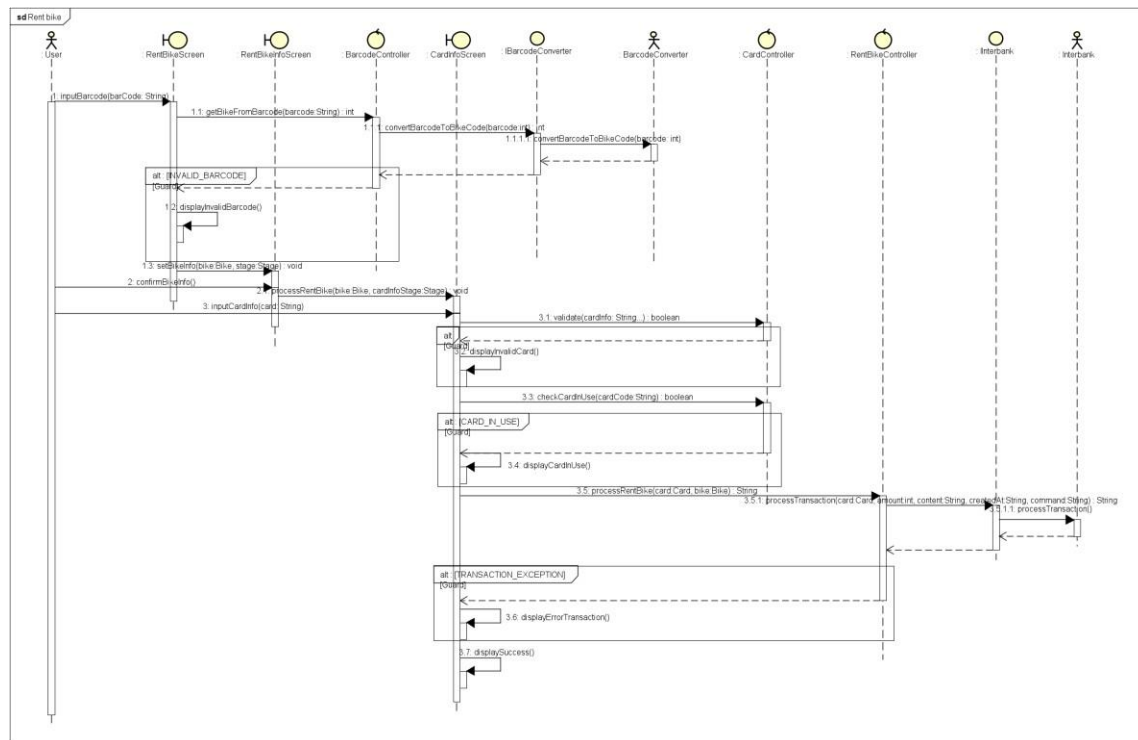
Chú ý, luồng sự kiện của UC này chứa cả luồng sự kiện cho việc xem thông tin chi tiết bãi xe.



Hình 3-2: Biểu đồ tuần tự cho UC001

Về cơ bản, để thực hiện use case, cần có các view để hiển thị GUI tương tác với người dùng, các GUI này bao gồm: DockListScreen, BikeDetailScreen, DockScreen. Controller để điều khiển là DockController. Để tìm kiếm đối tượng Dock và Bike cần thiết, cần phải duyệt qua toàn bộ Dock, Bike trong cơ sở dữ liệu để tìm ra đối tượng cần tìm, sau đó cập nhật view.

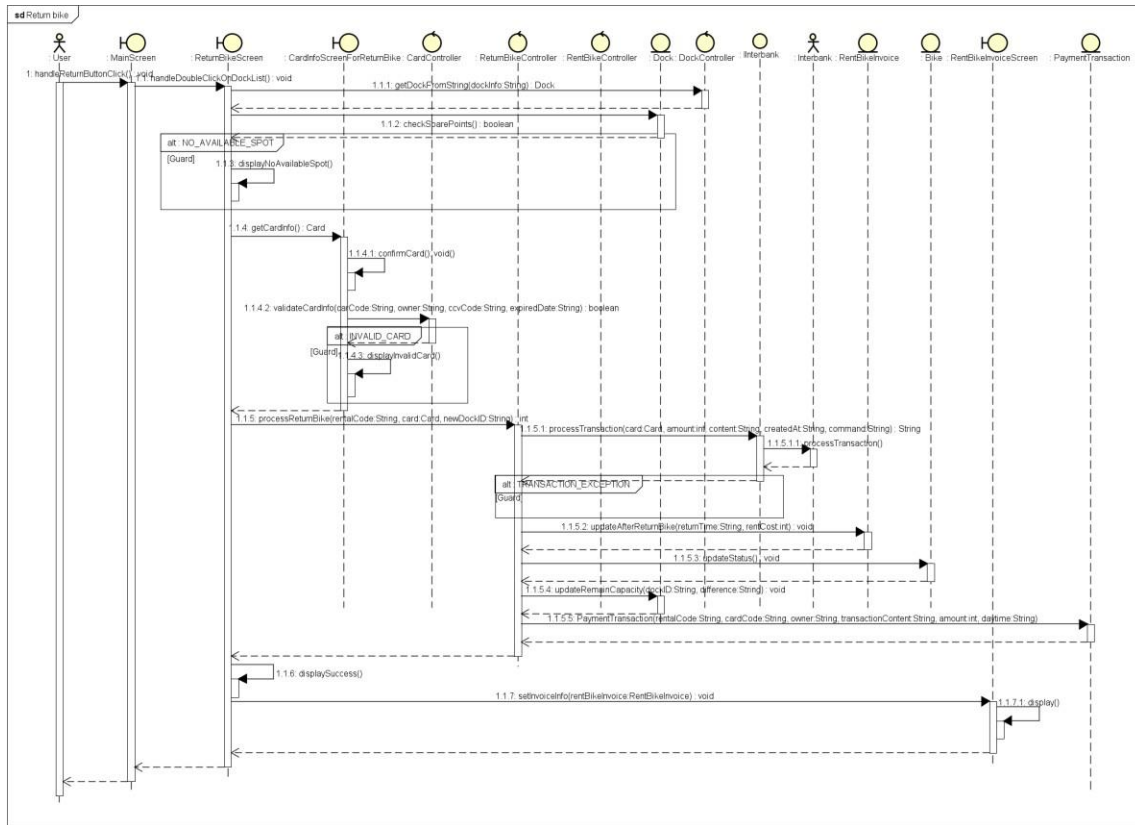
3.2.2 UC002 – Thuê xe



Hình 3-3: Biểu đồ tuần tự cho UC002

Các view phụ trách cho use case bao gồm: RentBikeScreen, RentBikeInfoScreen, CardInfoScreen. Các controller bao gồm: BarcodeController, CardController, RentBikeController. Use case này sẽ giao tiếp với 2 subsystem là Interbank và BarcodeConverter. Trong quá trình giao tiếp, nếu nhận được phản hồi về các ngoại lệ, sẽ tung ra các luồng sự kiện thay thế như được mô tả trong hình vẽ.

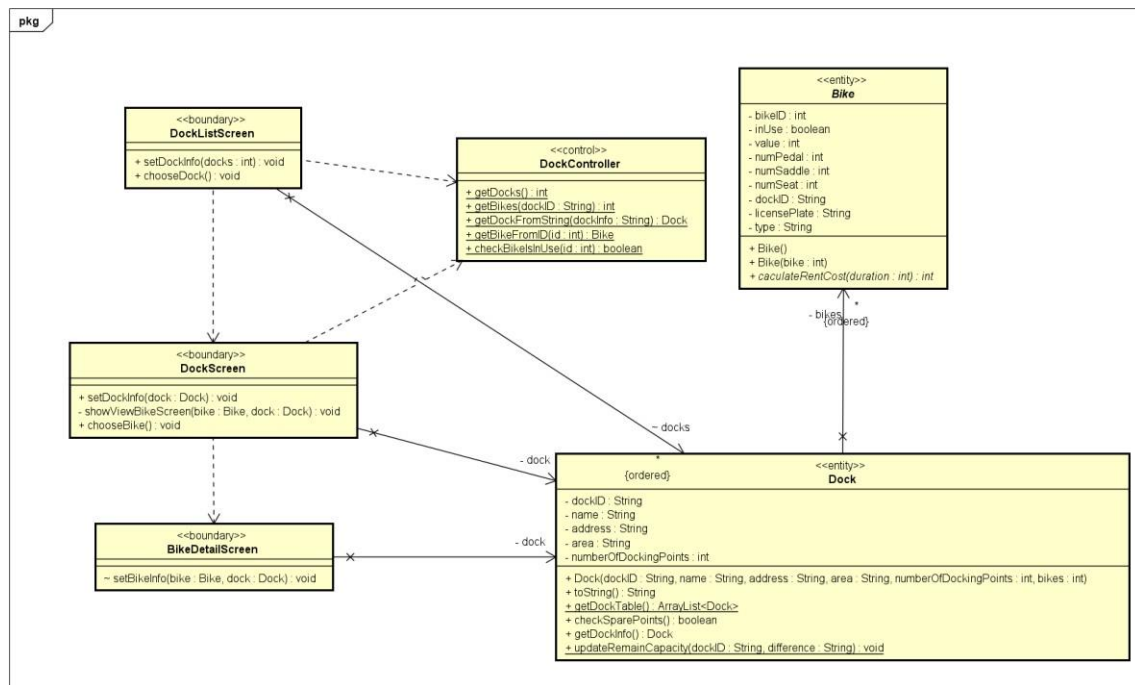
3.2.3 UC003 – Trả xe



Các view phụ trách cho use case bao gồm: MainScreen, ReturnBikeScreen, CardInfoScreen, RentBikeInvoiceScreen. Các controller bao gồm: CardController, ReturnBikeController, DockController. Các entity bao gồm: RentBikeInvoice, Dock và Bike, PaymentTransaction. Use case này sẽ giao tiếp với subsystem là Interbank. Trong quá trình giao tiếp, nếu nhận được phản hồi về các ngoại lệ, sẽ tung ra các luồng sự kiện thay thế như được mô tả trong hình vẽ.

3.3 Biểu đồ lớp phân tích

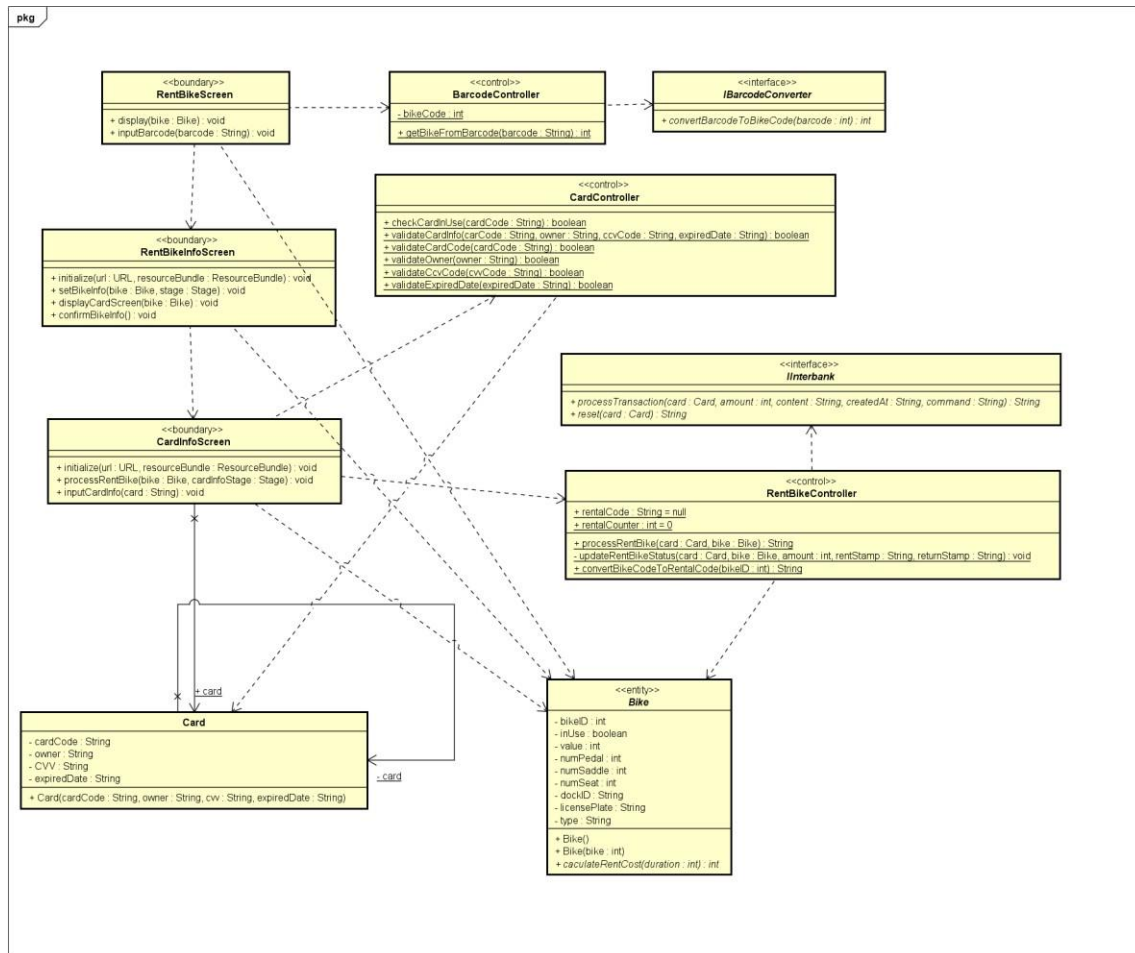
3.3.1 Biểu đồ lớp phân tích cho UC001



Hình 3-4: Biểu đồ phân tích lớp UC001

Trong biểu đồ lớp phân tích này, ta sẽ có thêm lớp Bike mà Dock có quan hệ association, chính vì có quan hệ association nên ở biểu đồ tuần tự UC001 ta thấy không có sự xuất hiện của Bike vì ta có thể truy vấn tìm đối tượng Bike bằng cách xem thuộc tính nằm bên trong Dock.

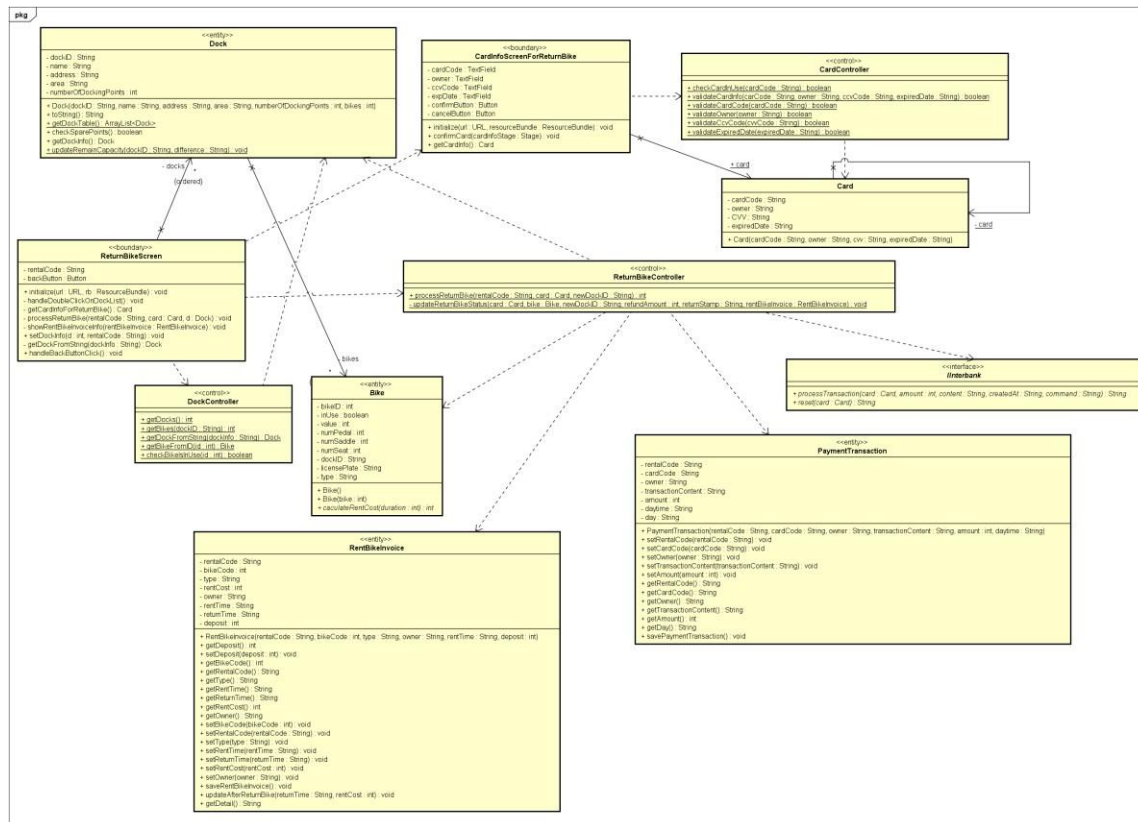
3.3.2 Biểu đồ lớp phân tích cho UC002



Hình 3-5: Biểu đồ lớp phân tích UC002

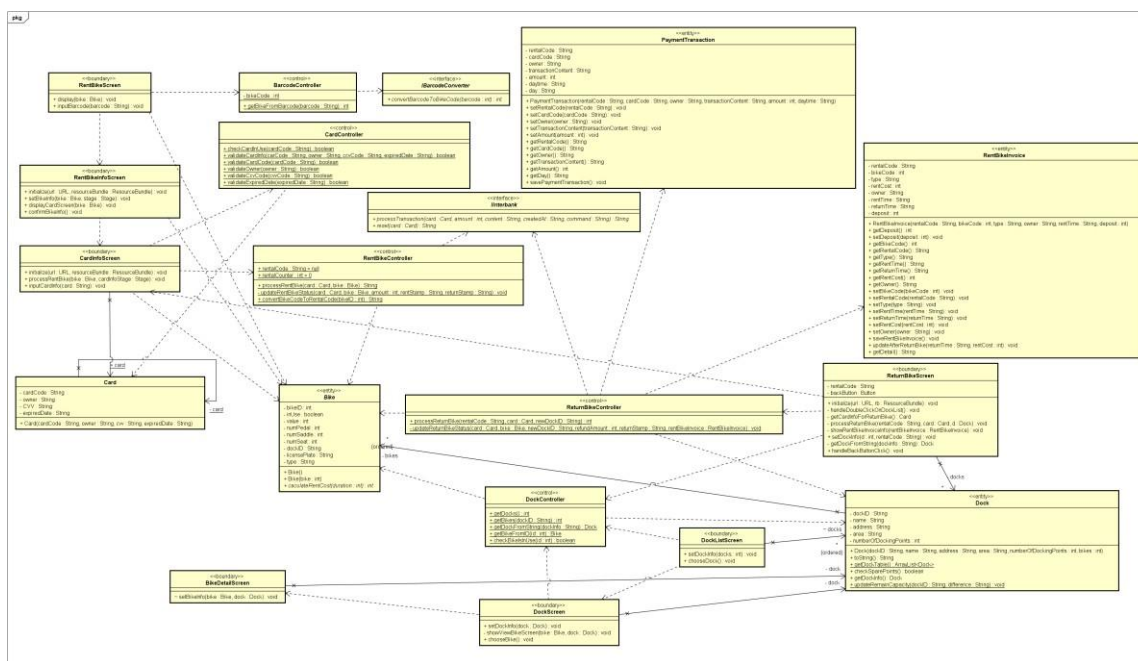
Trong quá trình gọi các phương thức, ta đóng gói dữ liệu dưới dạng các đối tượng Bike và Dock, chính vì vậy, có sự xuất hiện thêm của entity Bike và Dock trong use case với vai trò bị phụ thuộc (dependency). Ngoài ra đối với CardInfoScreen có liên kết (association) với Card để cho truy vấn tiện hơn (do có tới 3 nhiệm vụ nhỏ cần phải thực hiện: kiểm tra thông tin thẻ nhập có hợp lệ không, kiểm tra thẻ có đang được sử dụng không và thực hiện giao dịch với Interbank). Bên cạnh các entity, biểu đồ lớp phân tích còn có sự xuất hiện của các interface của các subsystem.

3.3.3 Biểu đồ lớp phân tích cho UC003



Trong quá trình gọi các phương thức, ta đóng gói dữ liệu dưới dạng các đối tượng Bike, Dock, RentBikeInvoice và PaymentTransaction chính vì vậy, có sự xuất hiện thêm của entity Bike, Dock, RentBikeInvoice và PaymentTransaction trong use case với vai trò bị phụ thuộc (dependency). Ngoài ra đối với CardInfoScreen có liên kết (association) với Card để cho truy vấn tiện hơn (do có tới 3 nhiệm vụ nhỏ cần phải thực hiện: kiểm tra thông tin thẻ nhập có hợp lệ không, kiểm tra thẻ có đang được sử dụng không và thực hiện giao dịch với Interbank). Bên cạnh các entity, biểu đồ lớp phân tích còn có sự xuất hiện của các interface của các subsystem.

3.4 Biểu đồ lớp phân tích gộp



3.5 Kiến trúc an toàn thông tin cho hệ thống

Về cơ bản, nhóm chưa tập trung vào phát triển kiến trúc an toàn thông tin cho hệ thống bởi vì ứng dụng xây dựng được chỉ ở chạy trên máy địa phương (local) và không có xác thực danh tính (ID) cho các phiên (session) đăng nhập. Chính vì không có các phiên đăng nhập và việc hệ thống quản lý chỉ là máy địa phương (local) nên nhóm không áp dụng các phương thức xác thực.

Tuy nhiên, dù chỉ là một ứng dụng chạy địa phương nhưng bên trong quá trình nghiệp vụ có sử dụng tới các API bên ngoài, bao gồm:

- API Interbank:
<https://ecopark-system-api.herokuapp.com/api/card/processTransaction>
- API Barcode converter:
<https://barcodeservicebykv2.herokuapp.com/barcode>

Các kết nối tới các API này đều qua giao thức HTTPS và có quá trình mã hóa các secret key được dùng bằng hàm băm MD5 nên quá trình trao đổi được đảm bảo. Các gói tin được gửi về đều được kiểm tra để xem có đúng so với API được đặc tả trước hay không. Tuy nhiên, do không tập trung phát triển các giao thức xác thực API nên rất có thể trong tương lai, việc đánh cắp, giả mạo các API có thể xảy ra. Bên cạnh đó, nhóm cũng có hiểu biết hạn hẹp trong lĩnh vực này nên cũng không thể lường trước được liệu rằng có cách tấn công nào với ứng dụng thông qua việc trao đổi các gói tin hay không.

Bên cạnh nguy hiểm tiềm tàng trong lĩnh vực kết nối qua API, các SQL Injection là cũng cần được xem xét, lý do bởi cơ sở dữ liệu nhóm sử dụng là MySQL, dù có một tầng bảo vệ bởi xác thực danh tính người dùng nhưng những rủi ro tiềm tàng vẫn còn khá cao.

Tóm lại, đối với tài liệu phát triển phần mềm phiên bản này, nhóm xin phép chỉ trình bày được những thách thức mà cần phải giải quyết.

4 Thiết kế chi tiết

4.1 Thiết kế *giao diện người dùng*

4.1.1 Chuẩn hóa cấu hình màn hình

Display:

- Phong: Màu xanh lá cây
- Kiểu font: calibri, cỡ chữ 12
- Số lượng màu hỗ trợ: 16,777,216 màu
- Độ phân giải: height: 640, width: 720 cho tất cả các màn hình

Screen:

- Ở phía trên sẽ có Logo (đối với các screen thẻ thì sẽ không có)
- Ở trên cùng góc trái có nút “back”
- Tất cả các nút đều được bôi đen, nền xanh
- Các nút “confirm”, “cancel”, ... đặt ở dưới màn hình

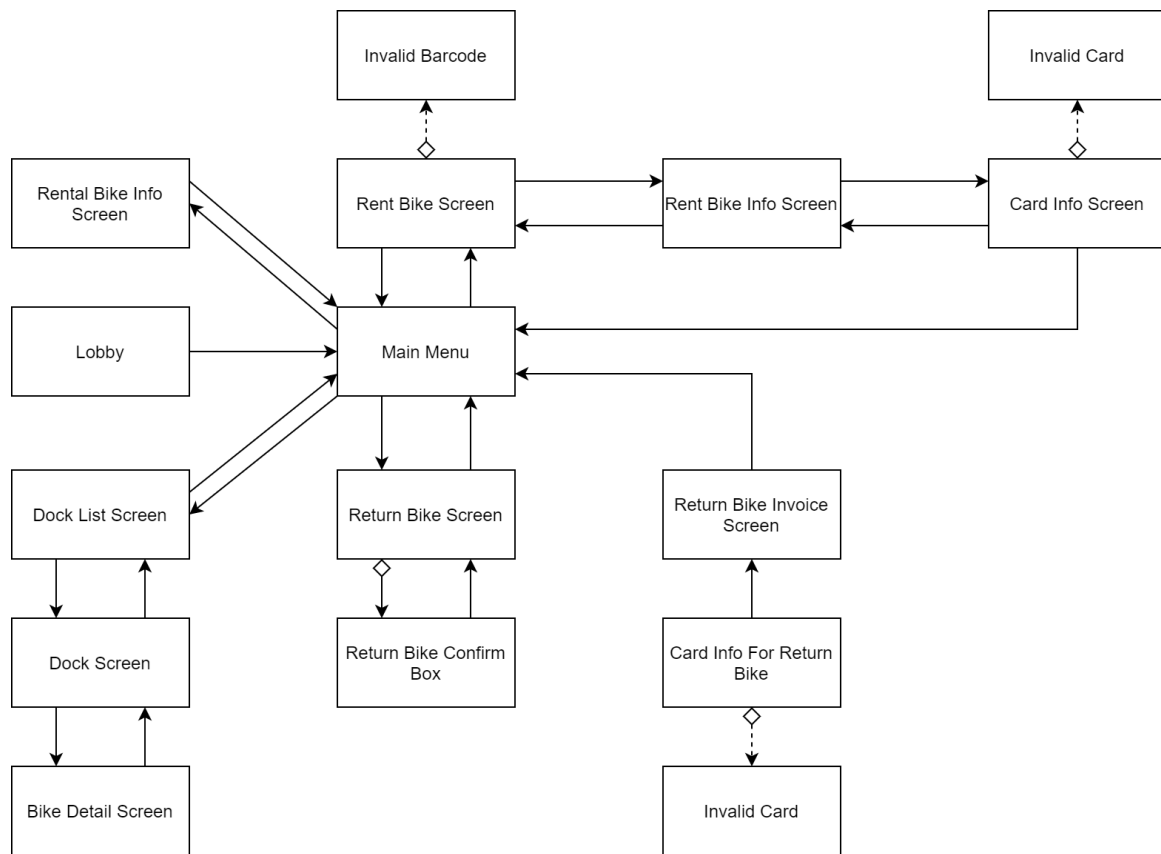
Control: Trình dự di chuyển chính của màn hình:

- Lobby
- Main Screen
- Rent Bike Screen – Bike Information Screen – Card Information (for rent) Screen (đối với UC002)
- Return Bike Screen – Card Information (for return) Screen – Rent Bike Invoice (đối với UC003)
- Dock List Screen – Dock Screen – Bike Detail Screen (đối với UC001)
- Rental Bike Screen (xem thông tin xe đang thuê)

Điều hướng màn hình:

- Các màn hình trước có thể được quay lại bởi nút “back”, các màn hình nghiệp vụ tiếp theo sẽ được gọi tới khi chọn nút “confirm”, “rent”, ...

4.1.2 Biểu đồ dịch chuyển màn hình

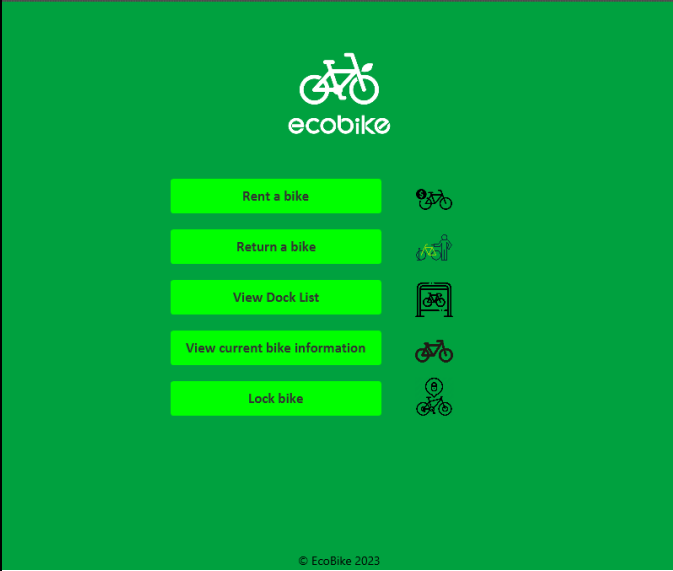


Hình 4-1: Biểu đồ dịch chuyển màn hình

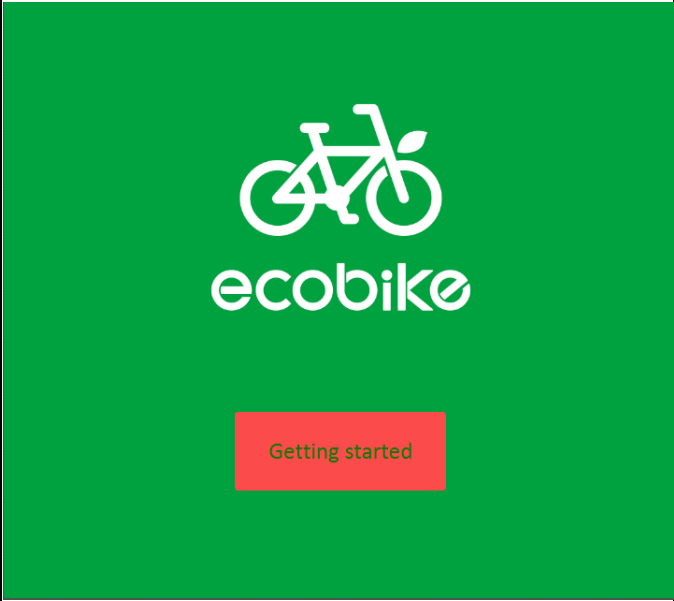
4.1.3 Đặc tả màn hình

4.1.3.1 Đặc tả màn hình “Main Screen”

1. Màn hình “Main Screen”

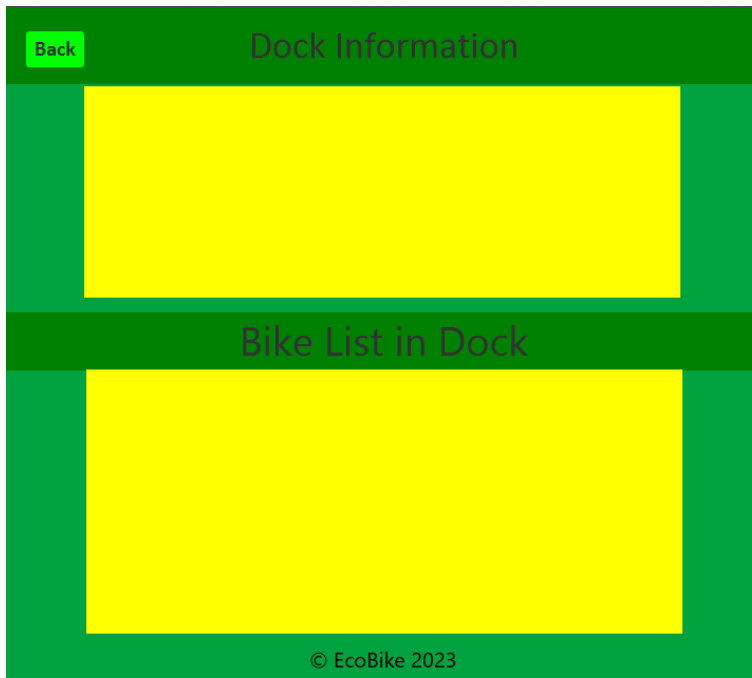
ECOBIKE		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Danh sách bãi xe	31/12/2022			Cao Như Đạt
		Control	Operation	Function	
		Nút “ View Dock List”	Double clicks	Người dùng muốn xem chi tiết thông tin một bãi xe	
		Nút “Rent a bike”	Click	Người dùng muốn thuê xe	
		Nút “Return a bike”	Click	Người dùng muốn trả xe	
		Nút “Lock Bike”	Click	Người dùng muốn tạm dừng xe và khóa xe	
		Nút “View current bike information”	Click	Người dùng muốn xem thông tin xe đang thuê	

2. Đặc tả màn hình “Lobby”

ECOBIKE		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Danh sách bãi xe	31/12/2022			Cao Như Đạt
		Control	Operation	Function	
		Nút “Getting Started”	Click	Chuyển đến giao diện màn hình chính	

4.1.3.2 Đặc tả màn hình “Xem thông tin”

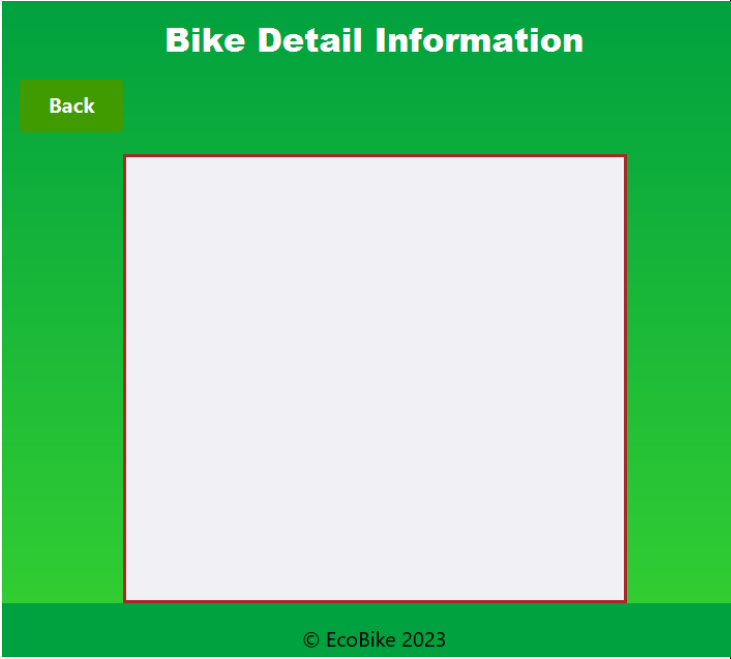
1. Xem thông tin chi tiết bãi xe

ECOBIKE		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Xem thông tin chi tiết bãi xe	31/12/2022			Cao Như Đạt
		Control	Operation	Function	
		Khu vực hiển thị thông tin chi tiết bãi xe	Khởi tạo	Hiển thị thông tin chi tiết bãi xe	
		Khu vực hiển thị danh sách xe trong bãi	Khởi tạo	Hiển thị danh sách các xe hiện có trong bãi. Thông tin hiển thị gồm mã xe – loại xe	
		Khu vực hiển thị danh sách xe trong bãi	Double clicks	Người dùng xem chi tiết thông tin một xe	

* Định nghĩa trường thuộc tính:

Screen Name	Thông tin chi tiết bãi xe		
Attribute	Type	Field Attribute	Remarks
Thông tin toàn bộ của bãi xe hiển thị dưới dạng một chuỗi ký tự duy nhất (bao gồm bãi xe, v.v)	String	Đen	Căn trái

2. Xem thông tin chi tiết xe

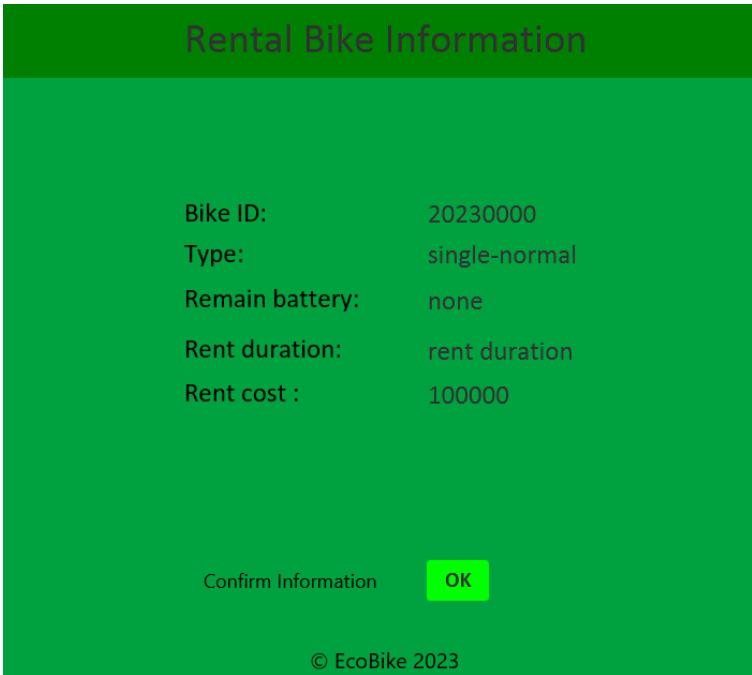
ECOBIKE		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Xem thông tin chi tiết xe	31/12/2022			Cao Như Đạt
		Control	Operation	Function	
		Nút “Back”	Click	Người dùng xác nhận và đóng cửa sổ	
		Khu vực hiển thị thông tin chi tiết xe	Khởi tạo	Hiển thị thông tin chi tiết xe	

* Định nghĩa trường thuộc tính:

Screen Name	Thông tin chi tiết bãi xe		
Attribute	Type	Field Attribute	Remarks
Mã xe	String	Đen	Căn trái
Loại xe	String	Đen	Căn trái
Giá trị	String	Đen	Căn trái
Số lượng bàn đạp	String	Đen	Căn trái
Số lượng yên xe	String	Đen	Căn trái
Số lượng ghế sau	String	Đen	Căn trái

Thông tin bổ sung	String	Đen	Căn trái
Bãi xe hiện tại	String	Đen	Căn trái
Biển số xe	String	Đen	Căn trái
Số tiền đặt cọc	String	Đen	Căn trái

3. Xem thông tin xe đang thuê

ECOBIKE		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Xem thông tin xe đang thuê	31/12/2022			Cao Như Đạt
		Control	Operation	Function	
		Khu vực hiện thị thông tin chi tiết xe đang thuê	Initial	Hiển thị thông tin chi tiết xe đang thuê	
		Nút “OK”	Click	Quay trở lại màn hình chính	
		Đầu đề của màn hình	Initial	Hiển thị thông tin màn hình	

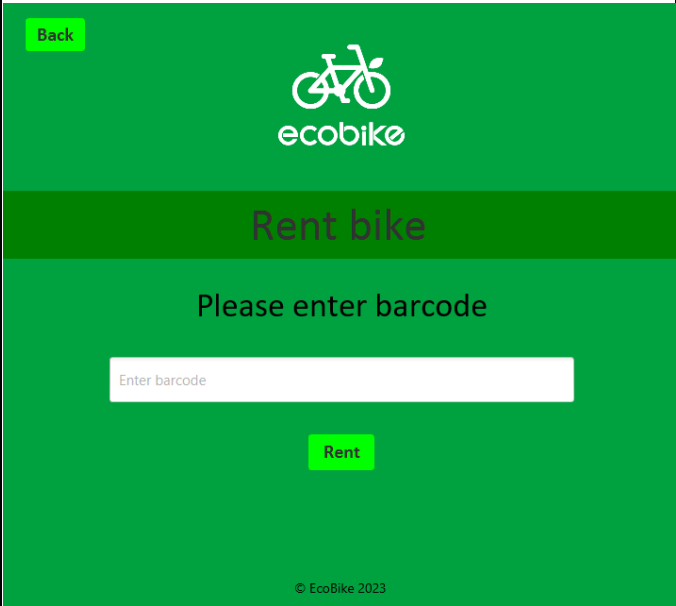
* Định nghĩa trường thuộc tính:

Screen Name	Thông tin chi tiết bãi xe		
Attribute	Type	Field Attribute	Remarks
Mã xe	String	Đen	Căn trái

Loại xe	String	Đen	Căn trái
Giá trị	String	Đen	Căn trái
Số pin còn lại	String	Đen	Căn trái
Thời gian đã thuê	String	Đen	Căn trái

4.1.3.3 Đặc tả màn hình “Thuê xe”

1. Nhập barcode để thuê xe

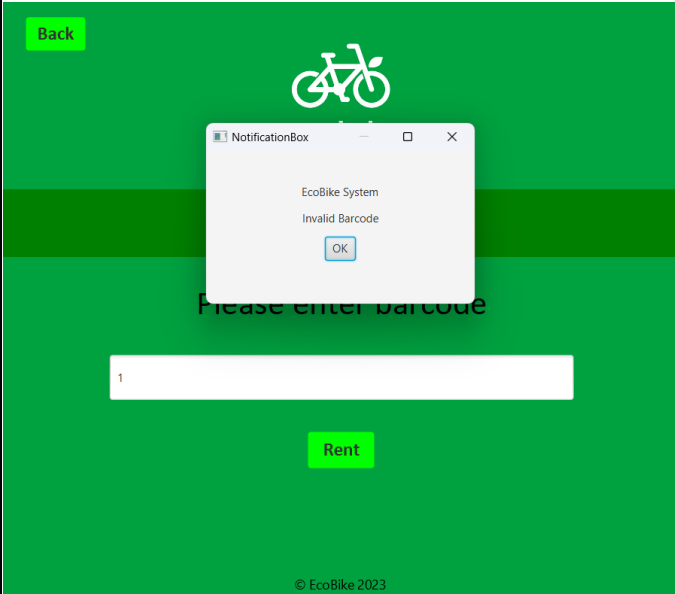
ECOBIKE		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Nhập barcode để thuê xe	31/12/2022			Cao Như Đạt
		Control	Operation	Function	
		Khu vực nhập barcode	Initial	Người dùng nhập từ bàn phím barcode xe mà người dùng muốn thuê	
		Nút “Rent”	Initial	Người dùng xác nhận muốn thuê xe với mã đã nhập, hệ thống kiểm tra, nếu không hợp lệ, chuyển đến màn hình “Nhập barcode – Không hợp lệ”, ngược lại chuyển đến màn hình “Thuê xe”	
		Nút “Back”	Initial	Chuyển đến màn hình “Main screen”	

--	--	--	--

Các trường thuộc tính:

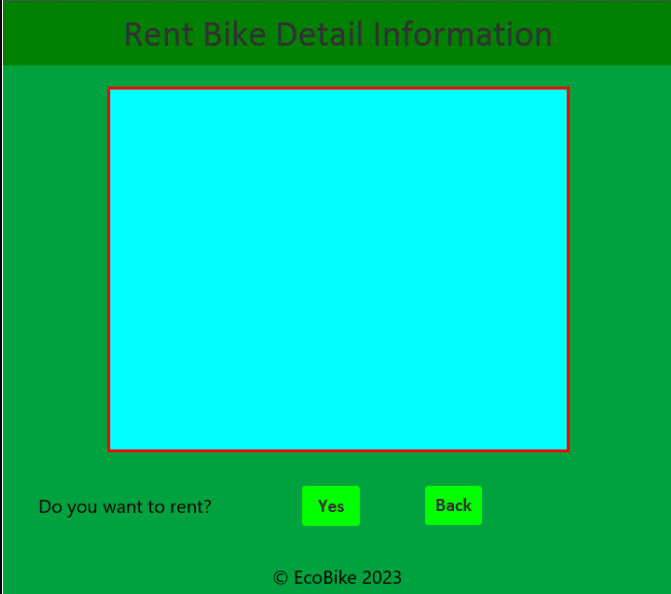
Screen Name	Nhập barcode để thuê xe			
Attribute	Number of digits(bytes)	Type	Field Attribute	Remarks
Barcode	10	Numerical	Đen	Căn giữa

2. Nhập barcode – Không hợp lệ

ECOBIKE		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Nhập barcode – Không hợp lệ	31/12/2022			Cao Như Đạt
		Control	Operation	Function	
		Khu vực hiển thị thông báo lỗi barcode của hệ thống	Initial	Hệ thống kiểm tra barcode người dùng nhập là không hợp lệ, nên thông báo cho khách hàng lỗi để khách hàng kiểm tra lại	
		Nút “OK”	Initial	Người dùng xác nhận thông báo	

3. Màn hình thuê xe

ECOBIKE	Date of			
---------	---------	--	--	--

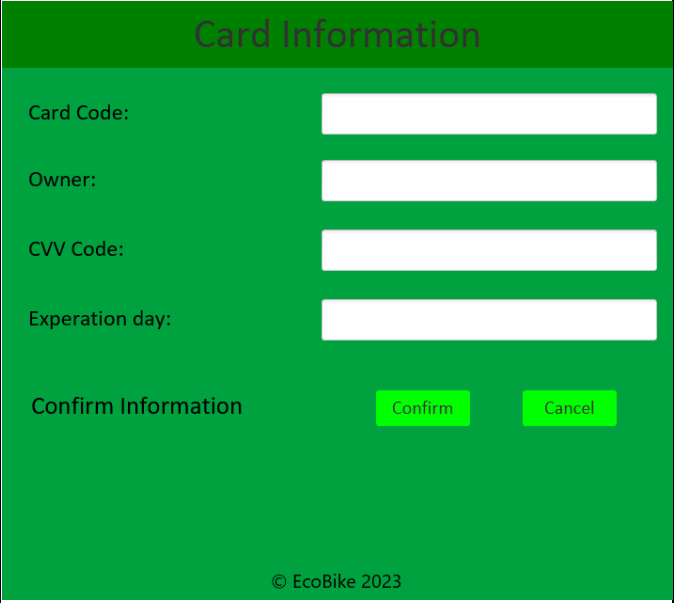
		creation	Approved by	Reviewed by	Person in charge
Screen specification	Thuê xe	31/12/2022			Cao Như Đạt
		Control	Operation	Function	
		Khu vực hiển thị thông tin chi tiết về xe	Initial	Hiển thị thông tin chi tiết về xe mà người dùng đã nhập barcode hợp lệ vào hệ thống	
		Khu vực hiển thị tiền cọc của xe	Initial	Hiển thị tiền cọc cần trả tương ứng	
		Nút “Xác nhận”	Click	Chuyển đến màn hình “Nhập thông tin thẻ”	
		Nút “Hủy”	Click	Chuyển đến màn hình “Nhập barcode để thuê xe”	

Các trường thuộc tính:

Screen Name	Thuê xe			
Attribute	Number of digits(bytes)	Type	Field Attribute	Remarks
Mã xe	10	Numeral	Đen	Căn trái
Loại xe	5	String	Đen	Căn trái

Giá trị	20	Numeral	Đen	Căn trái
Số lượng	5	Numeral	Đen	Căn trái
Bãi xe	10	String	Đen	Căn trái
Số tiền (cọc)	29	Numeral	Đen	Căn trái

4. Nhập thông tin thẻ

ECOBIKE		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Nhập thông tin thẻ	31/12/2022			Cao Như Đạt
		Control	Operation	Function	
		Khu vực hiển thị nhập thông tin thẻ	Initial	Khu vực yêu cầu người dùng nhập thông tin thẻ sử dụng để thanh toán	
		Nút “X nhận”	Initial	Người dùng xác nhận muốn thanh toán để thuê xe, hệ thống kiểm tra các điều kiện để thanh toán thành công và chuyển đến màn hình “Thuê xe – Thành công”, ngược lại nếu có lỗi thì sẽ chuyển đến màn hình thông báo lỗi tương ứng	
		Nút “Hủy”	Initial	Chuyển đến màn hình “Nhập barcode để thuê xe”	

--	--	--	--

Các trường thuộc tính:

Screen Name	Nhập barcode để thuê xe			
Attribute	Number of digits(bytes)	Type	Field Attribute	Remarks
Mã thẻ	10	String	Đen	Căn trái
Chủ sở hữu	10	String	Đen	Căn trái
Mã CCV	10	Numeral	Đen	Căn trái
Ngày hết hạn	10	Numeral	Đen	Căn trái

5. Thuê xe thành công

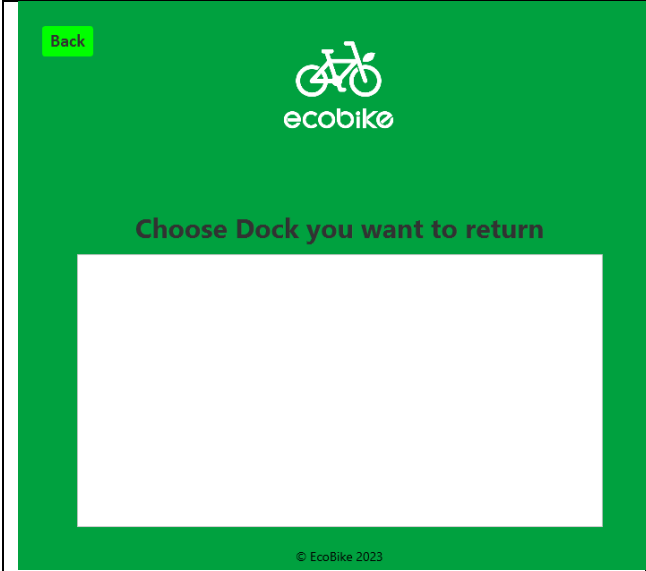
ECOBIKE		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Thuê xe – Thành công	31/12/2022			Cao Như Đạt
		Control	Operation	Function	
<div> <div>Card Information</div> <div> <div>Card Code: kstn_group9_20230</div> <div>Owner: </div> <div>CVV Code: </div> <div>Experation day: 1231</div> <div> <div>Confirm Information</div> <div>Confirm</div> <div>Cancel</div> </div> <div>© EcoBike 2023</div> </div> <div> <div>RentalCode</div> <div>EcoBike System</div> <div>Successful transaction, your rental code is: 202300060</div> <div>OK</div> </div> </div>		Khu vực hiển thị thông báo thuê xe thành công của hệ thống	Initial	Thông báo khách hàng đã giao dịch thuê xe thành công	
		Nút “Đóng”	Initial	Chuyển đến màn hình “Nhập barcode để thuê xe”	

	Nút “Quay lại trang chủ”	Initial	Chuyển đến màn hình “Main screen”
--	--------------------------	---------	-----------------------------------

4.1.3.4 Đặc tả màn hình “Trả xe”

1. Chọn bãi xe để trả

ECOBIKE		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Danh sách bãi xe	5/1/2023			Cao Như Đạt

	Control	Operation	Function
	Khu vực hiện thị danh sách bãi xe	Khởi tạo	Hiện 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 đỗ xe còn trống	String	Đen	Căn trái
Tên bãi xe	String	Đen	Căn trái

2. Nhập thông tin thẻ để trả xe

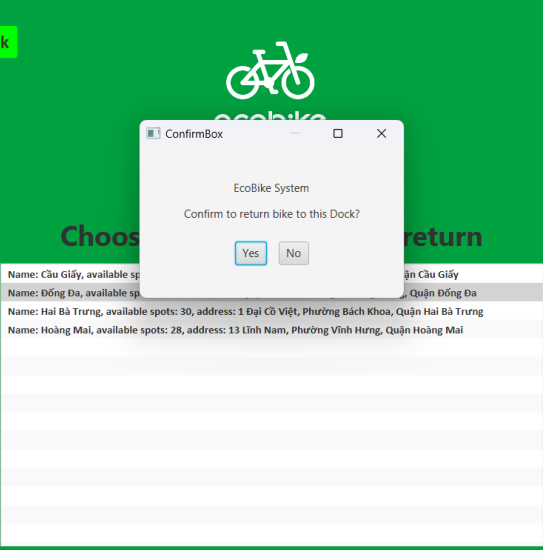
ECOBIKE		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Màn hình “Nhập thông tin thẻ để trả xe”	5/1/2023			Cao Như Đạt
		Control	Operation	Function	
		Nút “Confirm”	Click	Người dùng gửi thông tin thẻ	

<div> <div>Card Information For Return Bike</div> <div> <div>Card Code:</div> <div></div> </div> <div> <div>Owner:</div> <div></div> </div> <div> <div>CVV Code:</div> <div></div> </div> <div> <div>Experation day:</div> <div></div> </div> <div> <div>Confirm Information</div> <div>Confirm</div> <div>Cancel</div> </div> <div>© EcoBike 2023</div> </div>		Khu vực nhập thông tin thẻ	Người dùng nhập	Người dùng nhập thông tin thẻ
---	--	----------------------------	-----------------	-------------------------------

Các trường thuộc tính:

Screen Name	Danh sách bãi xe		
Attribute	Type	Field Attribute	Remarks
Card code	String	Đen	Căn trái
Người sở hữu	String	Đen	Căn trái
Mã CVV	String	Đen	Căn trái
Ngày hết hạn	String	Đen	Căn trái

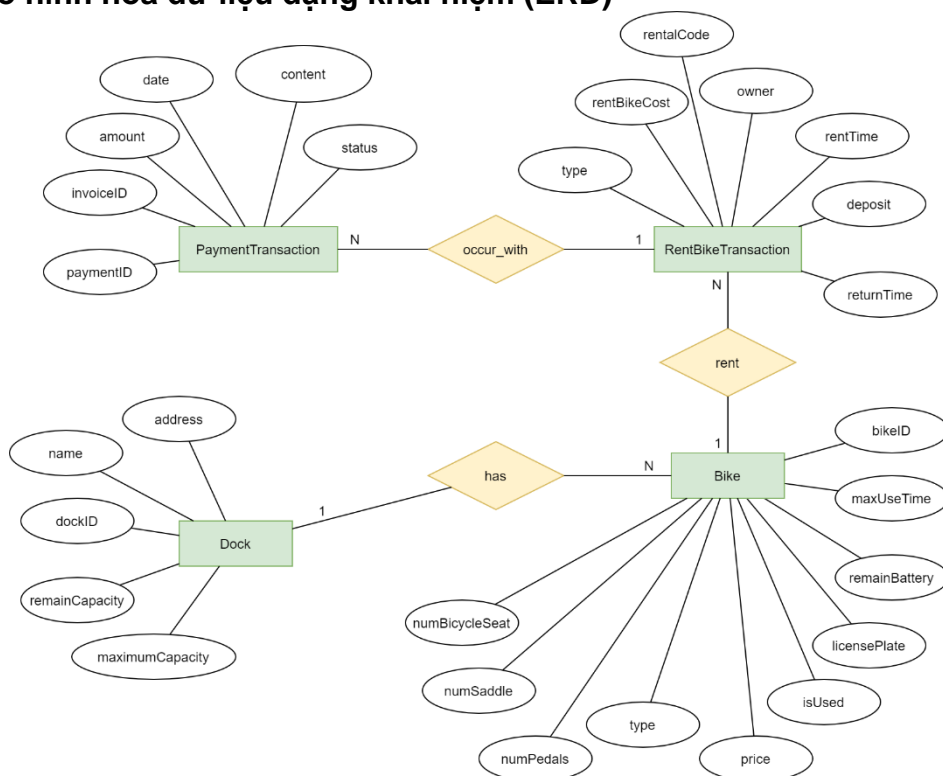
3. Xác nhận trả xe vào bãi

ECOBIKE		Date of creation	Approved by	Reviewed by	Person in charge
Screen specification	Thuê xe	5/1/2023			Cao Như Đạt
		Control	Operation	Function	
		Khu vực hiển thị thông tin bãi xe	Initial	Hiển thị thông tin chi tiết về xe mà người dùng đã nhập barcode hợp lệ vào hệ thống	
		Confirm Box	Initial		
		Nút “Yes”	Click	Chuyển đến màn hình “Nhập	

			thông tin thẻ cho trả xe”
	Nút “No”	Click	Chuyển đến màn hình “Trả xe”

4.2 Mô hình hóa dữ liệu

4.2.1 Mô hình hóa dữ liệu dạng khái niệm (ERD)



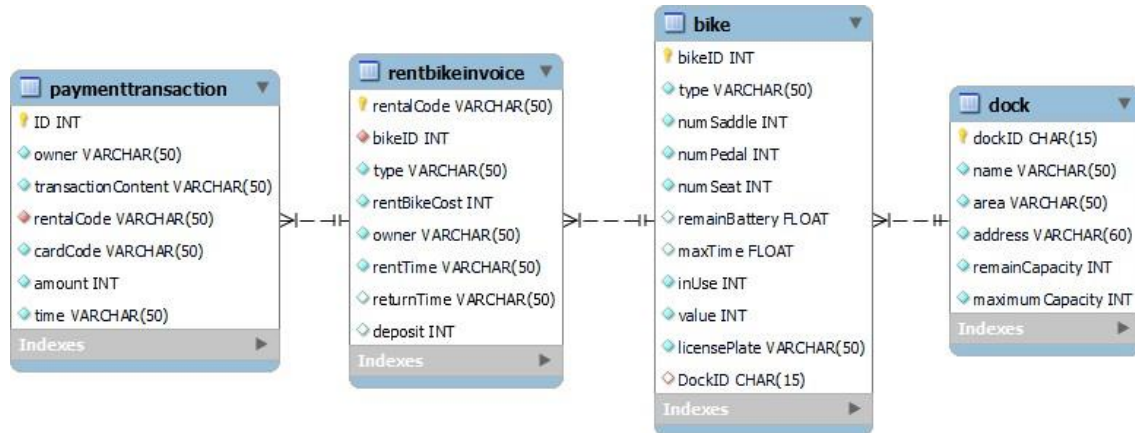
Hình 4-2: Biểu đồ ER cho cơ sở dữ liệu đề xuất

4.2.2 Thiết kế cơ sở dữ liệu

4.2.2.1 Hệ quản trị cơ sở dữ liệu (MySQL)

Nhóm sử dụng hệ quản trị cơ sở dữ liệu là MySQL 80. Để kết nối với hệ quản trị cơ sở dữ liệu, nhóm sử dụng thư viện JDBC. Toàn bộ mã nguồn của hệ quản trị cơ sở dữ liệu trong tệp bike.sql.

4.2.2.2 Biểu đồ cơ sở dữ liệu



Hình 4-3: Biểu đồ cơ sở dữ liệu sử dụng MySQL80

4.2.2.3 Thiết kế chi tiết cơ sở dữ liệu

4.2.2.3.1 Dock

Dock						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1	X		dockID	CHAR(15)	Có	ID của bãi xe
2			name	VARCHAR(50)	Có	tên bãi xe
3			address	VARCHAR(60)	Có	địa chỉ bãi xe
4			remainCapacity	INT	Có	số vị trí đỗ xe còn lại của bãi xe
5			maximumCapacity	INT	Có	số vị trí đỗ xe tối đa của bãi xe
6			area	VARCHAR(50)	Có	Khu vực của bãi xe

Bảng 4-1: Đặc tả chi tiết bảng Dock

4.2.2.3.2 Bike

Bike						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1	X		bikeID	INT	Có	mã xe
2			inUse	INT	Có	bằng 1 nếu đang sử dụng và bằng 0 nếu không sử dụng

3			type	VARCHAR(50)	Có	loại xe
4			value	INT	Có	giá xe
5			numSeat	INT	Có	Số lượng chỗ ngồi
6			numSaddle	INT	Có	Số lượng yên
7			numPedal	INT	Có	Số lượng bàn đạp
8			remainBattery	FLOAT	Có	lượng pin còn lại (với xe điện)
9			maxTime	FLOAT	Có	thời gian sử dụng tối đa (với xe điện)
10			licensePlate	VARCHAR(50)	Có	biển số xe
11		X	dockID	CHAR(15)	Có	vị trí bãi xe của xe hiện tại (chỉ có ý nghĩa với xe đang không được sử dụng)

Bảng 4-2: Đặc tả chi tiết bảng Bike

4.2.2.3.3 PaymentTransaction

PaymentTransaction						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1	X		ID	INT	Có	Mã của giao dịch
2			owner	VARCHAR(50)	Có	người thuê
3			transactionContent	VARCHAR(50)	Có	nội dung giao dịch
4			amount	INT	Có	lượng tiền giao dịch
5			time	VARCHAR(50)	Có	thời gian giao dịch (yyyy-MM-dd)
6			cardCode	VARCHAR(50)	Có	mã thẻ
7		X	rentalCode	VARCHAR(50)	Có	Mã thuê xe
8			type	VARCHAR(50)	Có	Loại của xe đang thuê

Bảng 4-3: Đặc tả chi tiết bảng PaymentTransaction

4.2.2.3.4 RentBikeInvoice

RentBikeInvoice						
#	PK	FK	Tên cột	Kiểu dữ liệu	Bắt buộc	Mô tả
1	X		rentalCode	INT	Có	Mã thuê xe
2		X	bikeID	INT	Có	mã xe được thuê
3			type	VARCHAR(50)	Có	loại xe
4			rentBikeCost	INT	Không	chi phí thuê xe (khuì chưa trả xe thì đặt là -1)
5			owner	VARCHAR(50)	Có	người thuê
6			rentTime	VARCHAR(50)	Có	thời gian thuê
7			returnTime	VARCHAR(50)	Không	thời gian trả (khi chưa trả thì đặt là "")
8			deposit	INT	Có	tiền đặt cọc

Bảng 4-4: Đặc tả chi tiết bảng RentBikeInvoice

4.3 Hệ quản trị các tệp phi cơ sở dữ liệu

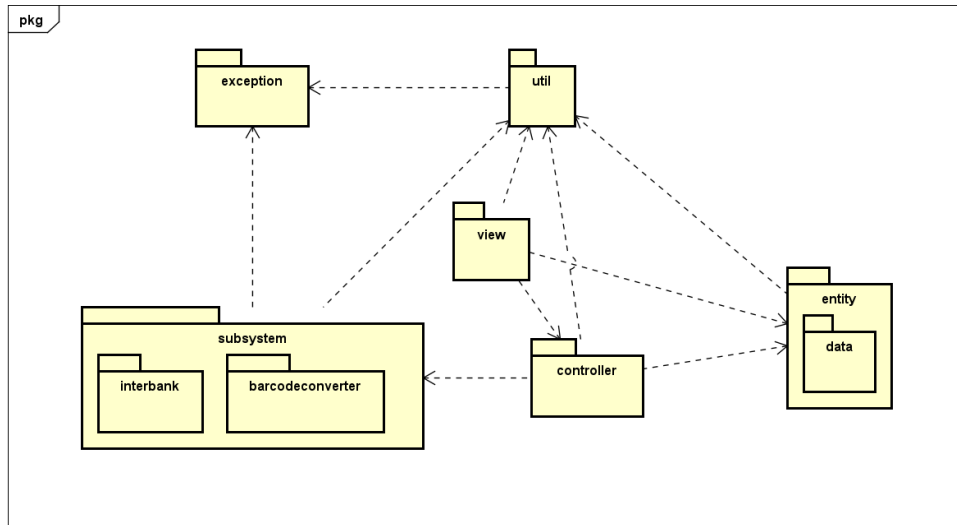
Về cơ bản, trong quá trình người dùng thao tác với hệ thống sẽ sinh ra những dữ liệu lưu lại log, tuy nhiên, vì với phiên bản phần mềm chỉ chạy trên máy địa phương và làm việc theo từng phiên riêng biệt (có nghĩa là phiên làm việc sau không bị ảnh hưởng bởi phiên làm việc trước) nên nhóm chỉ lưu những log ra màn hình console chứ chưa lưu lại các tệp log để sử dụng sau này.

Bàn về vấn đề này, nhóm cũng thấy rằng, việc lưu thành các log để sử dụng trong tương lai cũng rất quan trọng và cần được xem xét. Bởi vì, trong quá trình sử dụng, rất có thể có những lỗi tiềm ẩn nên lưu lại log sẽ giúp cho việc tiến hành backup, revert lại trạng thái ban đầu diễn ra suôn sẻ. Để làm được điều này, trong tương lai, cần xây dựng một hệ thống con để quản lý các log nghiệp vụ và lưu trữ các log này theo dạng tệp phi cơ sở dữ liệu.

4.4 Thiết kế lớp

4.4.1 Biểu đồ lớp tổng quan

Để có cái nhìn tổng quan về thiết kế lớp, tài liệu sẽ trình bày một số biểu đồ đơn giản sau:

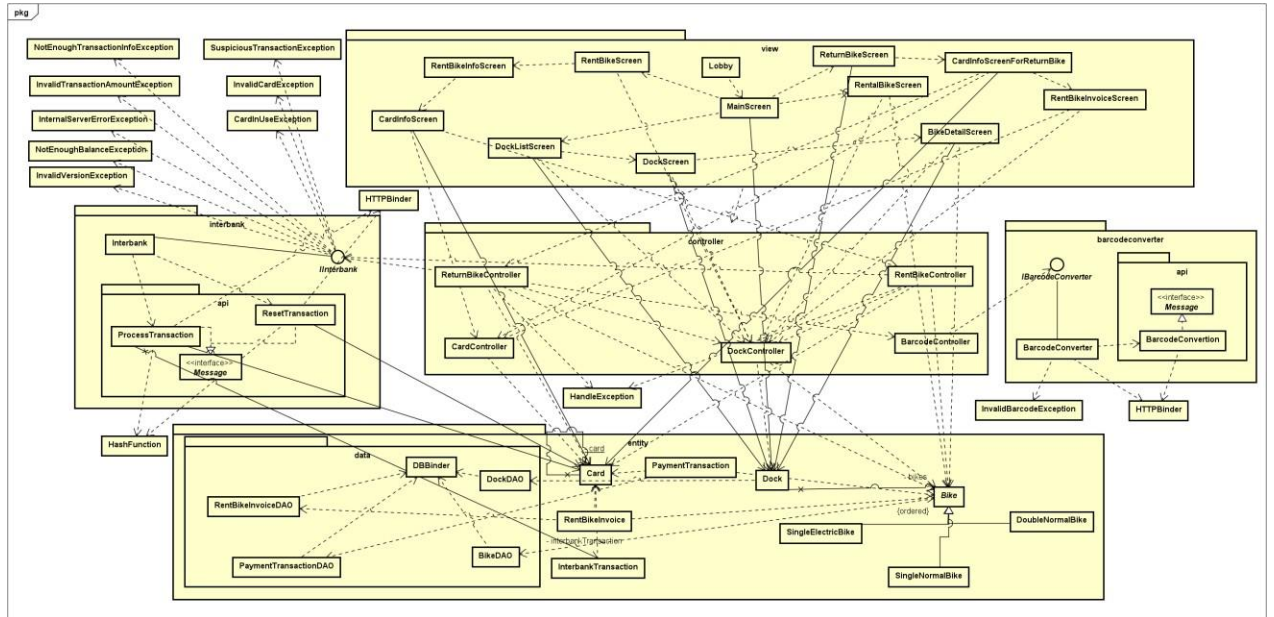


Hình 4-4: Biểu đồ thể hiện sự phụ thuộc của các gói

Hệ thống bao gồm các gói như sau:

- Gói util là lớp chứa các chức năng hỗ trợ cho hệ thống như kết nối HTTP, lưu các hằng số, các hàm băm, ... Gói này hầu như sẽ được sử dụng bởi toàn bộ hệ thống.
- Gói entity sẽ chứa các lớp mô hình hóa dữ liệu dạng khái niệm và chứa một gói con data để thực hiện cao yêu cầu liên quan đến truy vấn vào cơ sở dữ liệu.
- Gói subsystem chứa hai gói con interbank và barcodeconverter để thực hiện vai trò kết nối với API bên ngoài xử lý nghiệp vụ.
- Gói controller chứa các lớp có vai trò điều khiển nghiệp vụ của hệ thống, nhóm này sẽ sử dụng entity và subsystem.
- Gói view là gói thực hiện chức năng hiển thị và giao tiếp với người dùng. Để thực hiện yêu cầu của người dùng, gói view sẽ gọi tới các lớp xây dựng bên trong controller để giải quyết nghiệp vụ.

Tiếp đến, nhóm trình bày tổng quan về sự liên quan của tất cả các lớp:

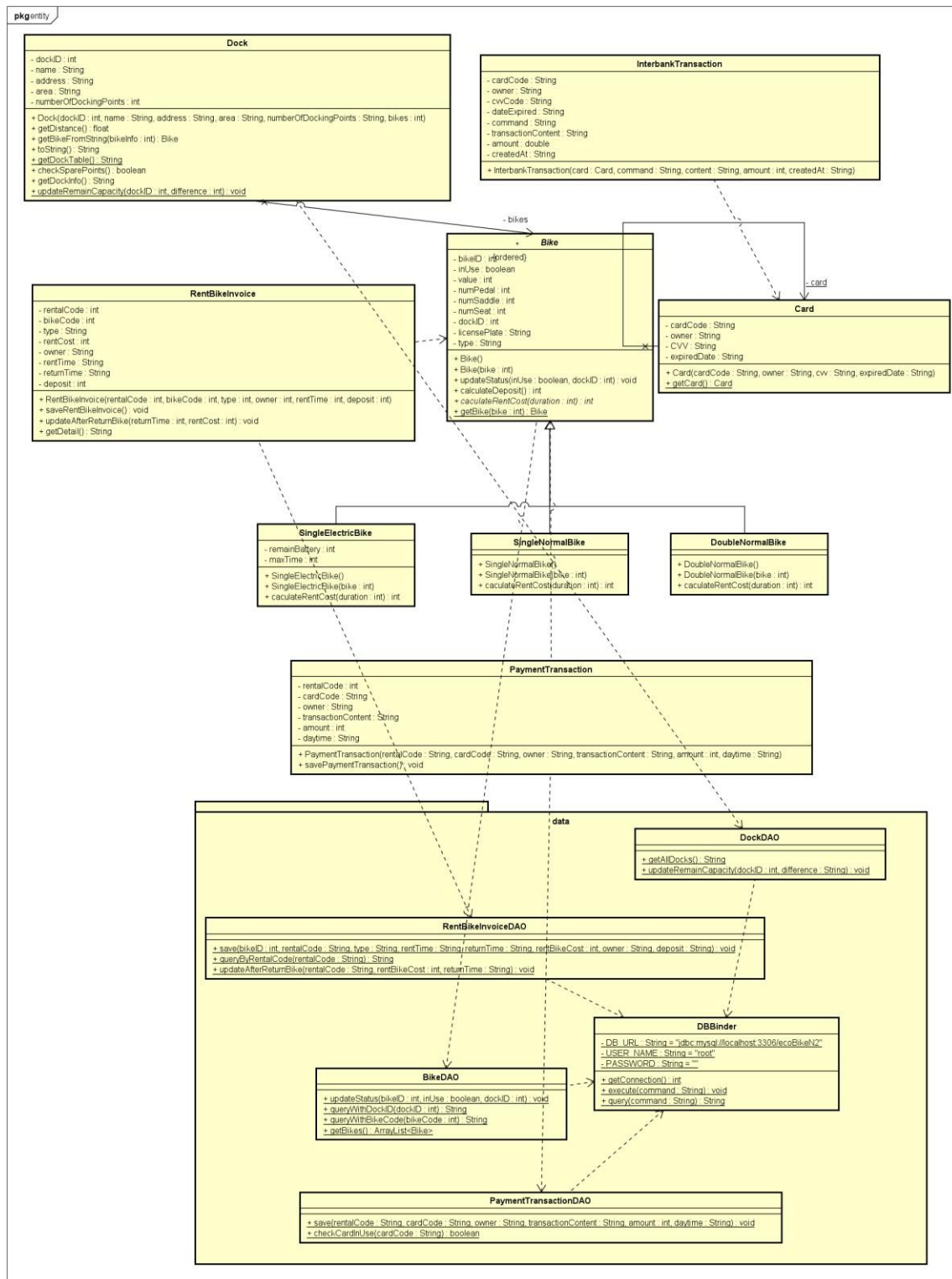


Hình 4-5: Biểu đồ thể hiện sự liên quan của các lớp

Đối với các lớp util, vì được sử dụng bởi hầu hết các lớp trong hệ thống nên được tách riêng ra để dễ nhìn hơn. Tương tự, hai lớp con interbank và barcodeconverter cũng được tách ra khỏi subsystem và exception cũng tách ra với mục đích tương tự. Về cơ bản, cohesion của các lớp trong cùng một package là khá cao, sự phụ thuộc chặt sẽ đảm bảo cho các lớp cùng hướng tới một chức năng chung. Tuy nhiên tính coupling cũng khá chặt dẫn đến việc sửa chữa trong tương lai khó khăn hơn. Tuy nhiên, sự phụ thuộc của các lớp cũng rất tự nhiên, dựa vào quá trình phân chia chức năng của kiến trúc MVC, đó là view sử dụng trực tiếp controller, controller sử dụng trực tiếp model, từ đó, dựa vào các kết quả của các lời gọi phương thức sẽ tiến hay thay đổi lớp gọi nó.

4.4.2 Biểu đồ lớp chi tiết

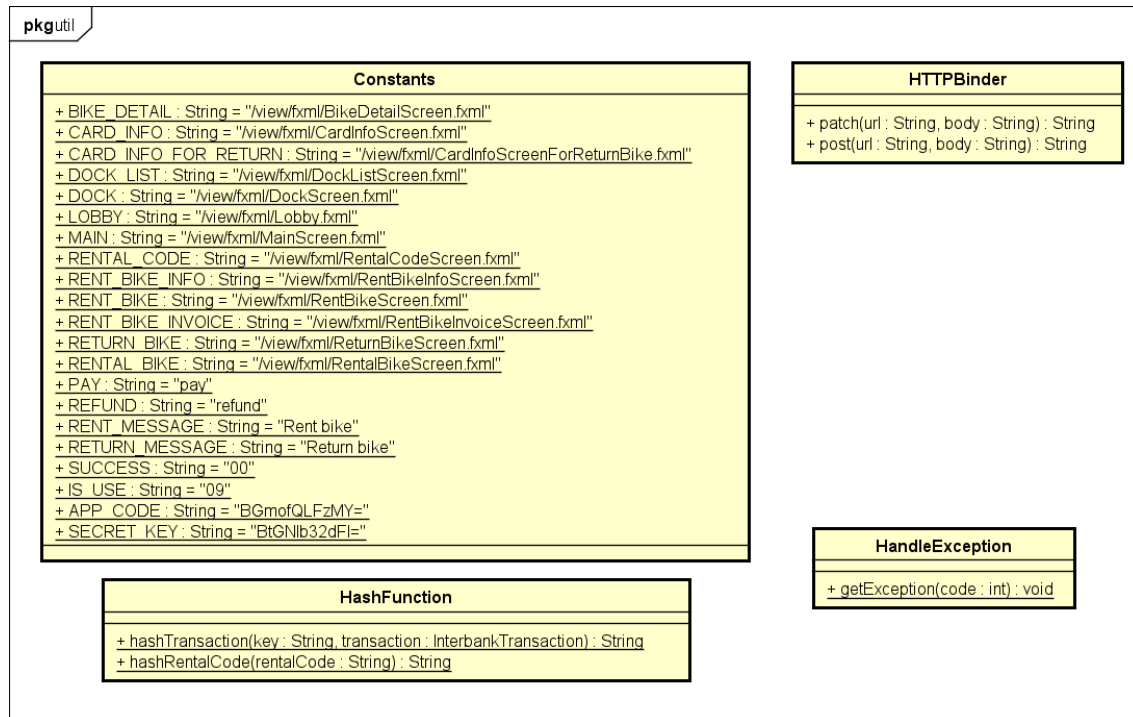
4.4.2.1 Biểu đồ lớp cho gói entity



Hình 4-6: Biểu đồ lớp chi tiết cho gói entity

Đối với gói entity sẽ bao gồm các lớp mô hình hóa mức độ khái niệm và có một gói con là data để thực hiện việc truy vấn vào cơ sở dữ liệu. Cơ chế truy vấn vào cơ sở dữ liệu được đảm bảo bởi lớp DBBinder có vai trò thực hiện các câu lệnh SQL.

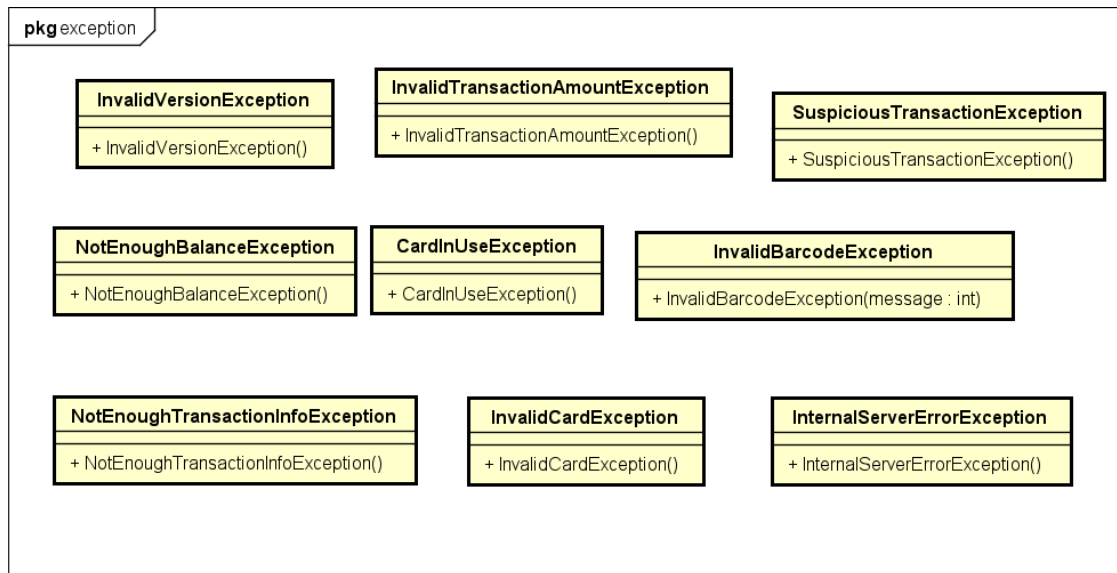
4.4.2.2 Biểu đồ lớp chi tiết cho gói util



Hình 4-7: Biểu đồ gói cho lớp util

Gói này chỉ đơn giản chứa các lớp mang tính chất thủ tục, lớp Constants sẽ dùng để ghi lại những hằng số cho các lớp, lớp HashFunction cung cấp cơ chế băm, lớp HandleException để dựa vào exception sinh ra các thông điệp tới người dùng, lớp HTTPBinder cung cấp khả năng kết nối HTTPS.

4.4.2.3 Biểu đồ lớp chi tiết cho gói exception

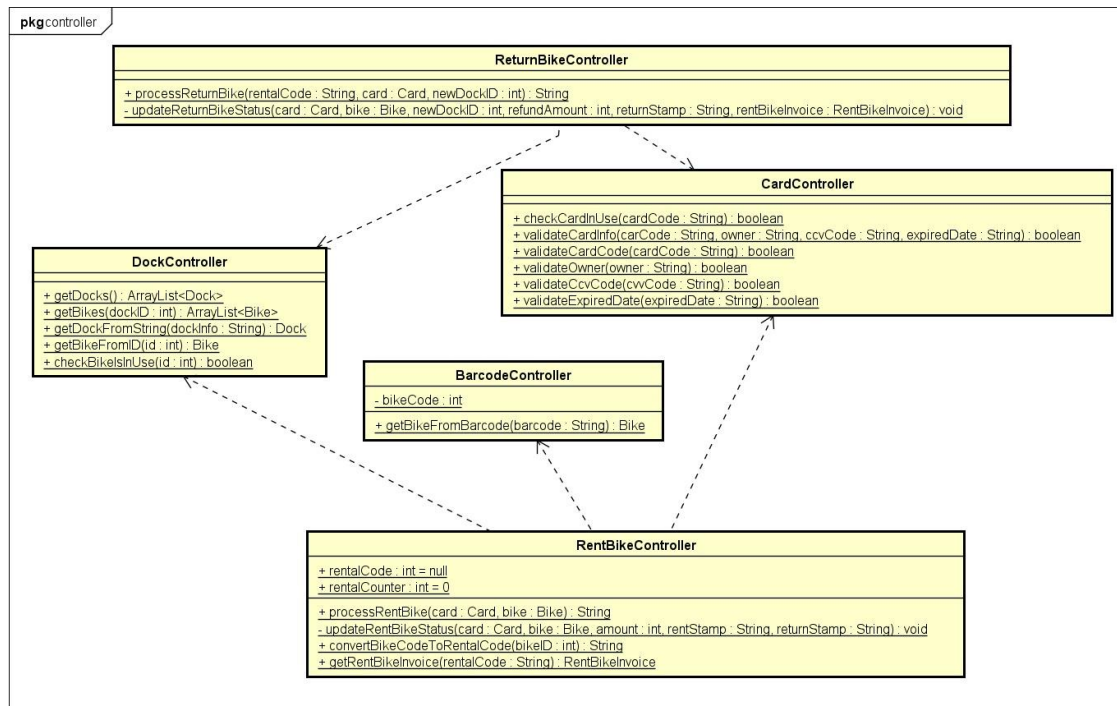


Hình 4-8: Biểu đồ lớp chi tiết cho gói exception

Gói exception về cơ bản sẽ chứa những lớp khác nhau để thực hiện tung các ngoại lệ khác nhau.

Về cơ bản, subsystem sẽ gồm 2 gói khác là interbank và barcodeconverter, mỗi gói này sẽ triển khai thành các hệ thống con để xử lý các nghiệp vụ cần kết nối tới API bên ngoài. Cơ chế triển khai của các gói là như nhau, bao gồm 1 giao diện, 1 lớp để triển khai giao diện và 1 gói api nhằm triển khai các gói tin gửi tới server bên ngoài. Các phương thức nghiệp vụ đều có thể tung các ngoại lệ dựa vào mã lỗi cũng như gói tin gửi về từ server, các ngoại lệ này là các lớp được triển khai trong gói exception.

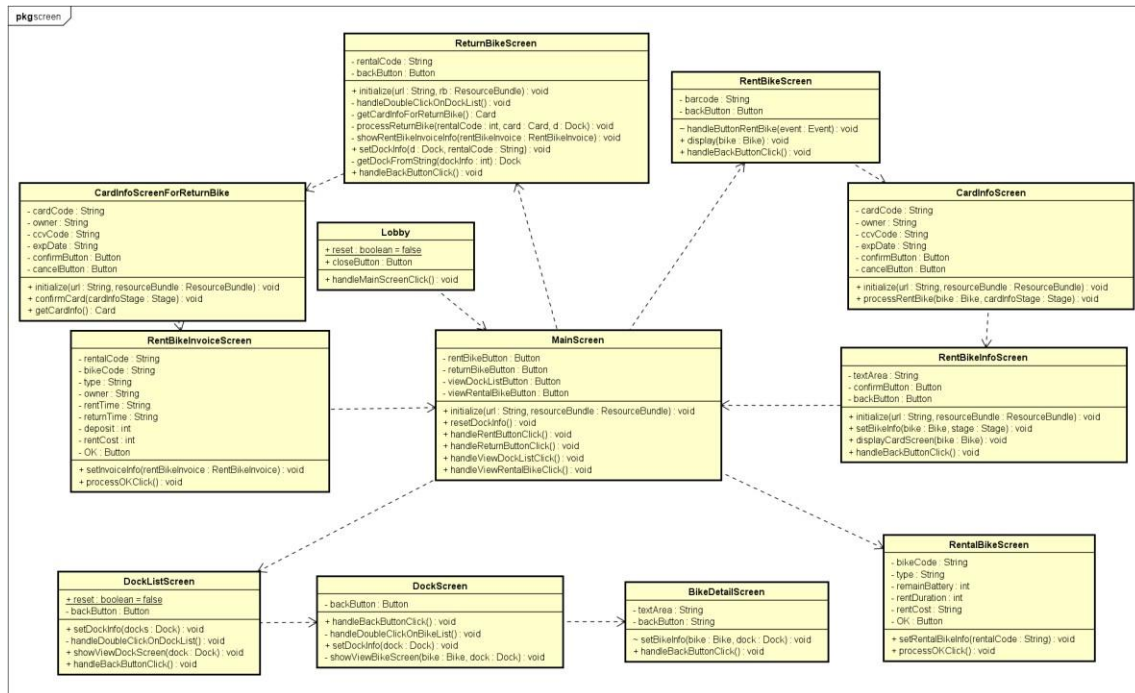
4.4.2.4 Biểu đồ lớp chi tiết gói controller



Hình 4-9: Biểu đồ lớp chi tiết cho gói controller

Gói controller sẽ bao gồm các lớp điều khiển nghiệp vụ cho hệ thống. Ta có thể thấy rằng, tính cohesion của gói này chưa được cao.

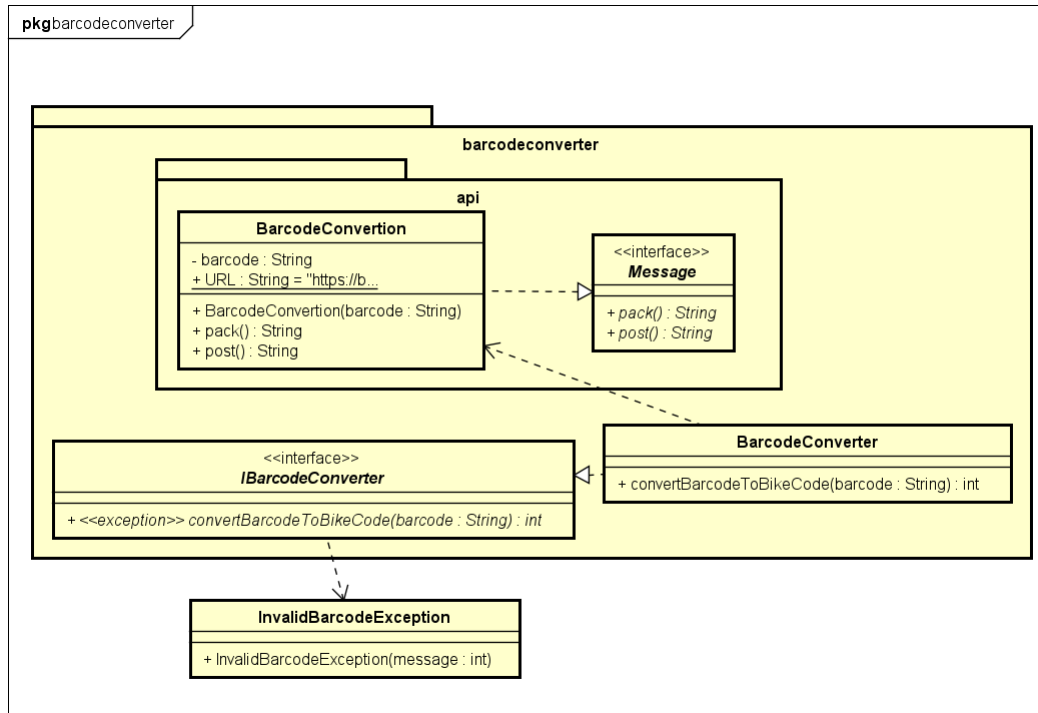
4.4.2.5 Biểu đồ lớp chi tiết gói view.screen



Hình 4-10: Biểu đồ lớp chi tiết cho gói view.screen

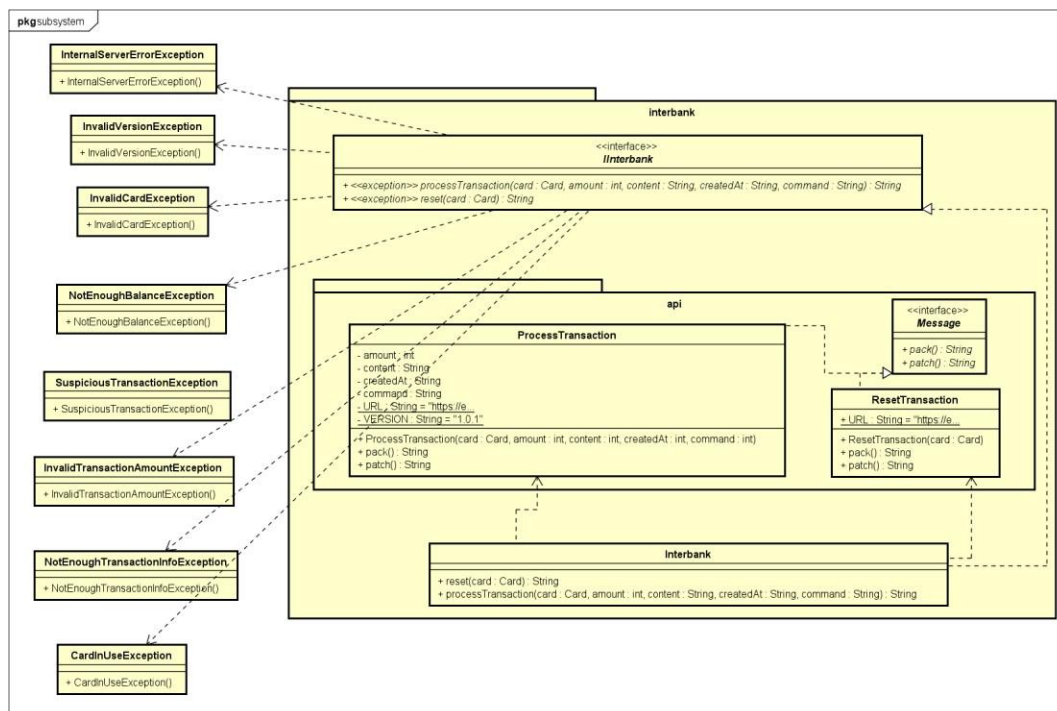
Về cơ bản, trong gói view còn một thư mục khác là fxml, tuy nhiên đây chỉ là thư mục chứa các tệp JavaFX và các GUI này đã được đặc tả ở phần trước. Trong gói view.screen sẽ có các lớp làm chức năng điều khiển các GUI. Ta thấy, lớp này có phụ thuộc chu trình (có nghĩa là tồn tại A – B – C – A phụ thuộc vào nhau). Tuy nhiên, điều này là bình thường, bởi vì một hệ thống phần mềm khi sử dụng thì các biểu đồ dịch chuyển các màn hình cũng sẽ là chu trình, lý do bởi xuất phát là màn hình chính thì sau khi sử dụng một số chức năng sẽ quay trở lại màn hình chính. Có thể thấy, coupling của gói này là rất cao nên việc sửa chữa trong tương lai sẽ tốn kém thời gian hơn.

4.4.2.6 Thiết kế lớp cho subsystem BarcodeConverter



Hình 4-11: Thiết kế lớp cho subsystem Barcode Converter

4.4.2.7 Thiết kế lớp cho subsystem Interbank



Hình 4-12: Thiết kế lớp chi tiết cho subsystem Interbank

4.4.3 Class Design

4.4.3.1 Class “RentbikeController”

RentBikeController	
+ rentalCode : int = null	
+ rentalCounter : int = 0	
+ processRentBike(card : Card, bike : Bike) : String	
- updateRentBikeStatus(card : Card, bike : Bike, amount : int, rentStamp : String, returnStamp : String) : void	
+ convertBikeCodeToRentalCode(bikeID : int) : String	
+ getRentBikeInvoice(rentalCode : String) : RentBikeInvoice	

Hình 4-13: Biểu đồ lớp RentbikeController

Đặc tả các thuộc tính

#	Name	Data type	Default value	Description
1	rentalCode	int	null	Mã thuê xe
2	rentalCounter	int	0	Bộ đếm mã thuê xe phục vụ cho rentalCode

Bảng 4-5: Đặc tả các thuộc tính của RentbikeController

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	processRentBike	String	Xử lý thuê xe
2	updateRentBikeStatus	void	Cập nhật trạng thái thuê xe
3	convertBikeCodeToRentalCode	String	Chuyển đổi mã xe sang mã thuê xe
4	getRentBikeInvoice	RentBikeInvoice	Hóa đơn thuê xe

Bảng 4-6: Đặc tả các phương thức của RentbikeController

Parameter:

- card – thẻ được sử dụng
- bike – xe muốn thuê
- amount – tổng số tiền thuê xe
- rentStamp – timestamp lúc thuê
- returnStamp – timestamp lúc trả
- bikeID – mã xe
- rentalCode – mã thuê xe

Exception:

- Các exception sẽ xảy ra trong quá trình thanh toán: InvalidCardException, ...

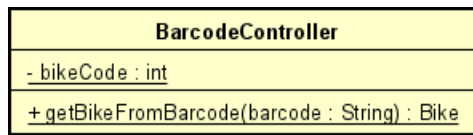
Method

Không

State

Không

4.4.3.2 Class “BarcodeController”



Hình 4-14: Biểu đồ lớp BarcodeController

Đặc tả các thuộc tính

#	Name	Data type	Default value	Description
1	bikeCode	int	null	Mã code xe

Bảng 4-7: Đặc tả các thuộc tính của BarcodeController

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	getBikeFromBarcode	Bike	Chuyển đổi từ barcode sang xe

Bảng 4-8: Đặc tả các phương thức của BarcodeController

Parameter:

- Barcode: Mã barcode được nhập từ người dùng

Exception:

- InvalidBarcodeException

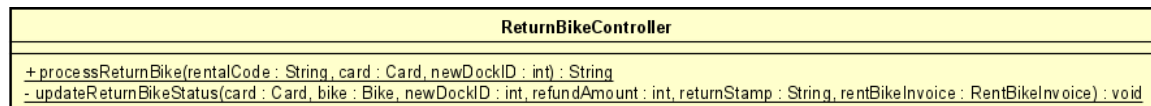
Method

Không

State

Không

4.4.3.3 Class “ReturnbikeController”



Hình 4-15: Biểu đồ lớp ReturnBikeController

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	processReturnBike	String	Xử lý trả xe

2	updateReturnBikeStatus	void	Cập nhật trạng thái trả xe
---	------------------------	------	----------------------------

Bảng 4-9: Đặc tả các phương thức của ReturnBikeController

Parameter:

- rentalCode – mã thuê xe
- card – thẻ được sử dụng
- newDockID – ID bãi xe mới
- bike – xe đang được thuê và sẽ tiến hành trả xe
- refundAmount – tổng tiền trả lại
- returnStamp – timestamp lúc trả xe
- rentBikeInvoice – hóa đơn thuê xe

Exception:

- Các exception sẽ xảy ra trong quá trình thanh toán: InvalidCardException, ...

Method

Không

State

Không

4.4.3.4 Class “CardController”

CardController
+ checkCardInUse(cardCode : String) : boolean + validateCardInfo(carCode : String, owner : String, ccvCode : String, expiredDate : String) : boolean + validateCardCode(cardCode : String) : boolean + validateOwner(owner : String) : boolean + validateCcvCode(ccvCode : String) : boolean + validateExpiredDate(expiredDate : String) : boolean

Hình 4-16: Biểu đồ lớp CardController

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	checkCardInUse	boolean	Kiểm tra thẻ có đang hoạt động
2	validateCardInfo	boolean	Kiểm tra thông tin thẻ
3	validateCardCode	boolean	Kiểm tra mã thẻ
4	validateOwner	boolean	Kiểm tra người sở hữu
5	validateCcvCode	boolean	Kiểm tra mã ccv
6	validateExpiredDate	boolean	Kiểm tra ngày hết hạn

Bảng 4-10: Đặc tả các phương thức của CardController

Parameter:

- cardCode – mã thẻ
- owner – người sở hữu
- cvvCode – mã cvv
- ExpiredDate – ngày hết hạn

Exception:

- Không

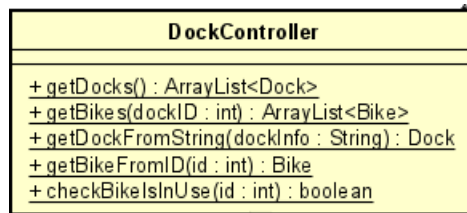
Method

Không

State

Không

4.4.3.5 Class “DockController”



Hình 4-17: Biểu đồ lớp DockController

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	getDocks	ArrayList<Dock>	Lấy thông tin bãi xe
2	getBikes	ArrayList<Bike>	Lấy thông tin xe
3	getDockFromString	Dock	Lấy thông tin bãi xe
4	getBikeFromID	Bike	Lấy thông tin xe từ ID
5	checkBikeIsInUse	boolean	Kiểm tra xe đang dùng

Bảng 4-11: Đặc tả các phương thức của DockController

Parameter:

- dockID – mã bãi xe
- dockInfo – thông tin bãi xe
- id – mã xe/ mã bãi xe tùy theo phương thức

Exception:

- Không

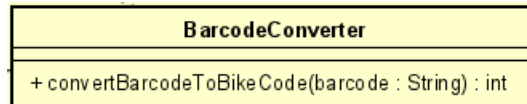
Method

Không

State

Không

4.4.3.6 Class “BarcodeConverter”



Hình 4-18: Biểu đồ lớp BarcodeConverter

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	convertBarcodeToBikeCode	int	Chuyển barcode sang mã xe

Bảng 4-12: Đặc tả các phương thức của BarcodeConverter

Parameter:

- barcode – barcode nhập vào từ người dùng

Exception:

- InvalidBarcodeException

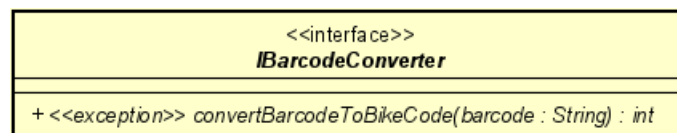
Method

Không

State

Không

4.4.3.7 Class “IBarcodeConverter”



Hình 4-19: Biểu đồ lớp IBarcodeConverter

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	convertBarcodeToBikeCode	int	Chuyển barcode sang mã xe

Bảng 4-13: Đặc tả các phương thức của IBarcodeConverter

Parameter:

- Barcode: Mã barcode nhập từ người dùng

Exception:

- InvalidBarcode

Method

Không

State

Không

4.4.3.8 Class “BarcodeConversion”

BarcodeConversion
- barcode : String + URL : String = "https://b...
+ BarcodeConversion(barcode : String) + pack() : String + post() : String

Hình 4-20: Biểu đồ lớp BarcodeConversion

Đặc tả các thuộc tính

#	Name	Data type	Default value	Description
1	barcode	String		Barcode nhập vào từ người dùng
2	URL	String	https://...	URL kết nối đến API chuyển barcode

Bảng 4-14: Đặc tả các thuộc tính của BarcodeConverter

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	BarcodeConversion	BarcodeConversion	Chuyển đổi barcode
2	pack	String	Phương thức đóng gói một gói tin
3	post	String	Phương thức POST lên 1 URL

Bảng 4-15: Đặc tả các phương thức của BarcodeConversion

Parameter:

- barcode – barcode nhập vào từ người dùng

Exception:

- Không

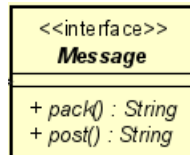
Method

Không

State

Không

4.4.3.9 Class “Message” (của gói subsystem.barcodeconverter)



Hình 4-21: Biểu đồ lớp Message (của gói subsystem.barcodeconverter)

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	pack	String	Phương thức đóng gói một gói tin
2	post	String	Phương thức POST lên 1 URL

Bảng 4-16: Đặc tả các phương thức của Message (của gói subsystem.barcodeconverter)

Parameter:

- Không

Exception:

- Không

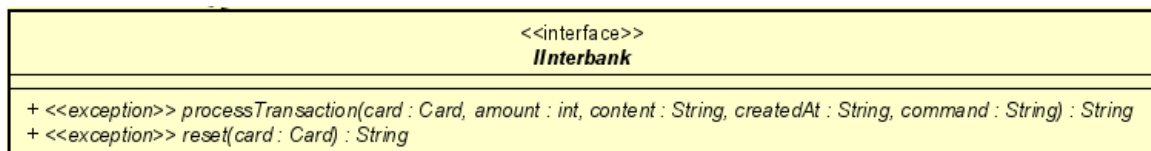
Method

Không

State

Không

4.4.3.10 Class “IInterBank”



Hình 4-22: Biểu đồ lớp IInterbank

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	processTransaction	String	Xử lý giao dịch, trả về mã lỗi giao dịch
2	reset	String	Reset thẻ, trả về mã lỗi giao dịch

Bảng 4-17: Đặc tả các phương thức của IInterbank

Parameter:

- card – thẻ của người dùng
- amount – số tiền cần phải trả
- content – nội dung giao dịch
- createAt – thời gian giao dịch
- command – lệnh của giao dịch theo quy ước của API: pay/refund.

Exception:

- Các exception xảy ra trong quá trình giao dịch như: InvalidCardException, ...

Method

Không

State

Không

4.4.3.11 Class “Interbank”

Interbank
+ reset(card : Card) : String + processTransaction(card : Card, amount : int, content : String, createdAt : String, command : String) : String

Hình 4-23: Biểu đồ lớp Interbank

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	processTransaction	String	Xử lý giao dịch
2	reset	String	Reset thẻ

Bảng 4-18: Đặc tả phương thức của Interbank

Parameter:

- card – thẻ được sử dụng
- amount – số tiền được giao dịch
- content – nội dung của giao dịch
- createAt – thời gian giao dịch
- command – lệnh của giao dịch: pay/ refund

Exception:

- Các exception xảy ra trong quá trình giao dịch như: InvalidCardException, ...

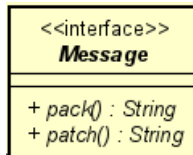
Method

Không

State

Không

4.4.3.12 Class “Message” (của gói subsystem.interbank)



Hình 4-24: Biểu đồ lớp Message (của gói subsystem.interbank)

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	pack	String	Phương thức đóng gói một gói tin
2	patch	String	Phương thức PATCH lên 1 URL

Bảng 4-19: Đặc tả các phương thức của Message (của gói subsystem.interbank)

Parameter:

- Không

Exception:

- Không

Method

Không

State

Không

4.4.3.13 Class “ResetTransaction”

ResetTransaction
+ URL : String = "https://e...
+ ResetTransaction(card : Card)
+ pack() : String
+ patch() : String

Hình 4-25: Biểu đồ lớp ResetTransaction

Đặc tả các thuộc tính

#	Name	Data type	Default value	Description
1	URL	int	"https://ecopark-system-api.herokuapp.com/api/card/reset-balance"	URL đến API để thực hiện reset thẻ

Bảng 4-20: Đặc tả các thuộc tính của ResetTransaction

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	ResetTransaction	ResetTransaction	Phương thức khởi tạo
2	pack	String	Đóng gói gói tin
3	patch	String	Phương thức PATCH lên URL

Bảng 4-21: Đặc tả các phương thức của ResetTransaction

Parameter:

- card – đối tượng thẻ cần thao tác

Exception:

- Không

Method

Không

State

Không

4.4.3.14 Class “ProcessTransaction”

Process Transaction
- amount : int - content : String - createdAt : String - command : String - URL : String = "http://e..." - VERSION : String = "1.0.1"
+ ProcessTransaction(card : Card, amount : int, content : int, createdAt : int, command : int) + pack() : String + patch() : String

Hình 4-26: Biểu đồ lớp ProcessTransaction

Đặc tả các thuộc tính

#	Name	Data type	Default value	Description
1	amount	int		Số tiền giao dịch
2	content	String		Nội dung giao dịch
3	createAt	String		Thời gian giao dịch
4	command	String		Lệnh của API: pay/refund
5	URL	String	https://ecopark-system-api.herokuapp.com/api/card/processTransaction	Link tới server của API
6	VERSION	String	1.0.1	Phiên bản, cần thiết để gọi API

Bảng 4-22: Đặc tả các thuộc tính của ProcessTransaction

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	ProcessTransaction	ProcessTransaction	Phương thức khởi tạo
2	pack	String	Phương thức đóng gói gói tin
3	patch	String	Phương thức PATCH lên 1 URL

Bảng 4-23: Đặc tả các phương thức của ProcessTransaction

Parameter:

- card – thẻ được sử dụng
- amount – số tiền cần giao dịch
- content – nội dung cần giao dịch
- createAt – thời gian giao dịch
- command – lệnh của giao dịch: pay/ refund

Exception:

- Không

Method

Không

State

Không

4.4.3.15 Class “HashFunction”

HashFunction
+ hashTransaction(key : String, transaction : InterbankTransaction) : String + hashRentalCode(rentalCode : String) : String

Hình 4-27: Biểu đồ lớp HashFunction

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	hashTransaction	String	Xử lý thuê xe
2	hashRentalCode	String	Xử lý để hash mã thuê xe

Bảng 4-24: Đặc tả các phương thức của HashFunction

Parameter:

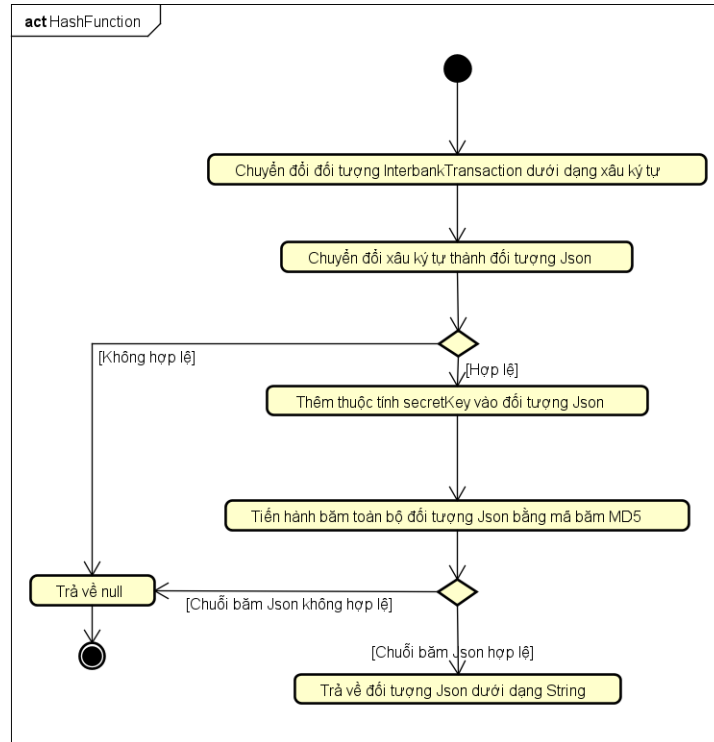
- key – được sử dụng làm nguyên liệu cho hàm băm
- transaction – Thông tin giao dịch, được đóng gói theo đặc tả API
- rentalCode – mã thuê xe

Exception:

- Không

Method

hashTransaction: chuyển đổi từ một đối tượng được trừu tượng hóa InterbankTransaction thành đối tượng Json phù hợp với API:



Hình 4-28: Chuyển đổi InterbankTransaction thành đối tượng Json

State

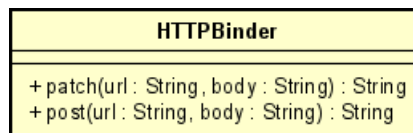
Không

4.4.3.16 Class “HTTPBinder”

Đặc tả các thuộc tính

Không

Đặc tả các phương thức



Hình 4-29: Biểu đồ lớp HTTPBinder

#	Name	Return type	Description (purpose)
1	patch	String	Phương thức PATCH lên 1 URL
2	post	String	Phương thức POST lên 1 URL

Bảng 4-25: Đặc tả các phương thức của HTTPBinder

Parameter:

- url – liên kết tới server cần thực hiện
- body – Nội dung của gói tin

Exception:

- Không

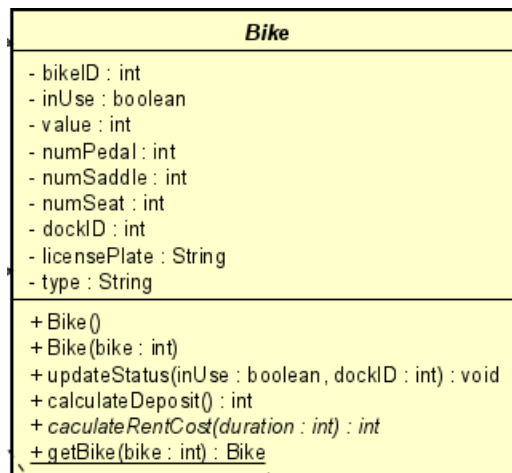
Method

Không

State

Không

4.4.3.17 Class “Bike”



Hình 4-30: Biểu đồ lớp Bike

Đặc tả các thuộc tính

#	Name	Data type	Description
1	bikeID	int	Mã xe
2	inUse	boolean	Có đang dùng không
3	value	int	Giá trị xe
4	numPedal	int	Số lượng bàn đạp
5	numSaddle	int	Số lượng yên xe
6	numSeat	int	Số chỗ ngồi
7	dockID	int	Mã bãi xe
8	licensePlate	String	Biển số xe
9	type	String	Loại xe

Bảng 4-26: Đặc tả các thuộc tính của Bike

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	Bike	Bike	Phương thức khởi tạo
2	updateStatus	void	Cập nhật trạng thái xe
3	calculateDeposit	int	Tính tiền cọc
4	caculateRentCost	int	Tính tiền thuê xe
5	getBike	Bike	Lấy thông tin xe

Bảng 4-27: Đặc tả các phương thức của Bike

Parameter:

- bike – xe muốn thuê
- inUse – xe có đang được sử dụng
- dockID – mã bãi xe
- duration – thời gian thuê xe

Exception:

- Không

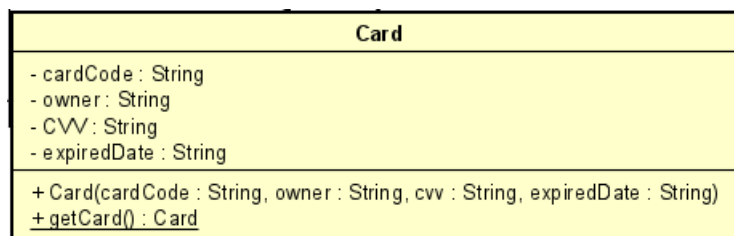
Method

Không

State

Không

4.4.3.18 Class “Card”



Hình 4-31: Biểu đồ lớp Card

Đặc tả các thuộc tính

#	Name	Data type	Description
1	cardCode	String	Mã thẻ
2	owner	String	Người sở hữu
3	CVV	String	Cvv
4	expiredDate	String	Ngày hết hạn

Bảng 4-28: Đặc tả các thuộc tính của Card

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	Card	Card	Phương thức khởi tạo
2	getCard	Card	Lấy thông tin thẻ

Bảng 4-29: Đặc tả các phương thức của Card

Parameter:

- cardCode – mã thẻ
- owner – người sở hữu
- CVV – CVV của thẻ
- expiredDate – ngày hết hạn

Exception:

- Không

Method

Không

State

Không

4.4.3.19 Class “Dock”

Dock
- dockID : int - name : String - address : String - area : String - numberOfDockingPoints : int
+ Dock(dockID : int, name : String, address : String, area : String, numberOfDockingPoints : String, bikes : int) + getDistance() : float + getBikeFromString(bikeInfo : int) : Bike + toString() : String + getDockTable() : String + checkSparePoints() : boolean + getDockInfo() : String + updateRemainCapacity(dockID : int, difference : int) : void

Hình 4-32: Biểu đồ lớp Dock

Đặc tả các thuộc tính

#	Name	Data type	Description
1	dockID	int	Mã bãi xe
2	name	String	Tên bãi xe
3	address	String	Địa chỉ
4	area	String	Khu vực

5	numberOfDockingPoints	int	Số lượng bãi đỗ xe
---	-----------------------	-----	--------------------

Bảng 4-30: Đặc tả các thuộc tính của Dock

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	Dock	Dock	Phương thức khởi tạo
2	getDistance	float	Lấy khoảng cách
3	getBikeFromString	Bike	Lấy thông tin xe
4	toString	String	Phương thức chuyển thành dạng String (ghi đề phương thức của Object)
5	getDockTable	String	Lấy danh sách bãi xe
6	checkSparePoints	boolean	Kiểm tra chỗ trống còn lại trong bãi
7	getDockInfo	String	Lấy thông tin bãi xe
8	updateRemainCapacity	void	Cập nhật chỗ trống bãi xe

Bảng 4-31: Đặc tả các phương thức của Dock

Parameter:

- dockID – mã bãi xe
- name – tên bãi xe
- address – địa chỉ của bãi xe
- area – khu vực của bãi xe
- numberOfDockingPoints – số lượng chỗ trống gửi xe trong bãi (ban đầu)
- bikes – danh sách xe
- bikeInfo – thông tin xe

Exception:

- Không

Method

Không

State

Không

4.4.3.20 Class “DoubleNormalBike”

DoubleNormalBike
+ DoubleNormalBike() + DoubleNormalBike(bike : String) + caculateRentCost(duration : int) : int

Hình 4-33: Biểu đồ lớp DoubleNormalBike

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	DoubleNormalBike	DoubleNormalBike	Phương thức khởi tạo
2	caculateRentCost	Bike	Tính tiền thuê xe

Bảng 4-32: Đặc tả các phương thức của DoubleNormalBike

Parameter:

- bike – Thông tin xe muốn truyền vào, dưới dạng String
- duration – thời gian thuê

Exception:

- Không

Method

Không

State

Không

4.4.3.21 Class “InterbankTransaction”

InterbankTransaction
- cardCode : String - owner : String - cvv Code : String - dateExpired : String - command : String - transactionContent : String - amount : double - createdAt : String
+ InterbankTransaction(card : Card, command : String, content : String, amount : int, createdAt : String)

Hình 4-34: Biểu đồ lớp InterbankTransaction

Đặc tả các thuộc tính

#	Name	Data type	Description
---	------	-----------	-------------

1	cardCode	String	Mã thẻ
2	owner	String	Người sở hữu
3	cvvCode	String	Mã cvv của thẻ
4	dateExpired	String	Ngày hết hạn
5	command	String	Lệnh của giao dịch: pay/ refund
6	transactionContent	String	Nội dung giao dịch
7	amount	double	Số tiền giao dịch
8	createAt	String	Thời điểm tạo

Bảng 4-33: Đặc tả các thuộc tính của InterbankTransaction

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	InterbankTransaction	InterbankTransaction	Phương thức khởi tạo

Bảng 4-34: Đặc tả các phương thức của InterbankTransaction

Parameter:

- card – thẻ được sử dụng
- command – lệnh của giao dịch: pay/ refund
- content – nội dung của giao dịch
- amount – số tiền được giao dịch
- createAt – thời gian giao dịch

Exception:

- Không

Method

Không

State

Không

4.4.3.22 Class “PaymentTransaction”

PaymentTransaction
<ul style="list-style-type: none"> - rentalCode : int - cardCode : String - owner : String - transactionContent : String - amount : int - daytime : String
<ul style="list-style-type: none"> + PaymentTransaction(rentalCode : String, cardCode : String, owner : String, transactionContent : String, amount : int, daytime : String) + savePaymentTransaction() : void

Hình 4-35: Biểu đồ lớp PaymentTransaction

Đặc tả các thuộc tính

#	Name	Data type	Description
1	rentalCode	int	Mã thuê xe
2	cardCode	String	Mã thẻ
3	owner	String	Người sở hữu
4	transactionContent	String	Nội dung giao dịch
5	amount	int	Tổng số tiền
6	daytime	String	Thời gian giao dịch

Bảng 4-35: Đặc tả các thuộc tính của PaymentTransaction**Đặc tả các phương thức**

#	Name	Return type	Description (purpose)
1	PaymentTransaction	PaymentTransaction	Phương thức khởi tạo
2	savePaymentTransaction	void	Lưu thông tin thanh toán

Bảng 4-36: Đặc tả các phương thức của PaymentTransaction*Parameter:*

- rentalCode – mã thuê xe
- cardCode – mã thẻ
- owner – người sở hữu
- transactionContent – nội dung giao dịch
- amount – tổng số tiền
- daytime – thời gian giao dịch

Exception:

- Không

Method

Không

State

Không

4.4.3.23 Class “RentBikeInvoice”

RentBikeInvoice
- rentalCode : int - bikeCode : int - type : String - rentCost : int - owner : String - rentTime : String - returnTime : String - deposit : int
+ RentBikeInvoice(rentalCode : String, bikeCode : int, type : String, owner : String, rentTime : String, deposit : String) + saveRentBikeInvoice() : void + updateAfterReturnBike(returnTime : int, rentCost : int) : void + getDetail() : String

Hình 4-36: Biểu đồ lớp RentBikeInvoice

Đặc tả các thuộc tính

#	Name	Data type	Description
1	rentalCode	int	Mã thuê xe
2	bikeCode	int	Mã xe
3	type	String	Loại xe
4	rentCost	int	Giá thuê xe
5	owner	String	Người sở hữu
6	rentTime	String	Thời gian thuê
7	returnTime	String	Thời gian trả
8	deposit	int	Tiền cọc

Bảng 4-37: Đặc tả các thuộc tính của RentBikeInvoice

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	RentBikeInvoice	RentBikeInvoice	Phương thức khởi tạo
2	saveRentBikeInvoice	void	Lưu hóa đơn thuê xe
3	updateAfterReturnBike	void	Cập nhật thông tin xe sau khi trả
4	getDetail	String	Lấy thông tin chi tiết

Bảng 4-38: Đặc tả các phương thức của RentBikeInvoice

Parameter:

- rentalCode – mã thuê xe
- bikeCode – mã xe
- type – loại xe
- owner – người sở hữu

- rentTime – thời gian bắt đầu thuê xe
- deposit – tiền cọc
- returnTime – thời gian trả xe
- rentCost – giá thuê xe

Exception:

- Không

Method

Không

State

Không

4.4.3.24 Class “SingleElectricBike”

SingleElectricBike
- remainBattery : int - maxTime : int
+ SingleElectricBike() + SingleElectricBike(bike : String) + caculateRentCost(duration : int) : int

Hình 4-37: Biểu đồ lớp SingleElectricBike

Đặc tả các thuộc tính

#	Name	Data type	Description
1	remainBattery	int	Phần trăm pin còn lại
2	maxTime	int	Thời gian sử dụng tối đa

Bảng 4-39: Đặc tả các thuộc tính của SingleElectricBike

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	SingleElectricBike	SingleElectricBike	Phương thức khởi tạo
2	SingleElectricBike	SingleElectricBike	Phương thức khởi tạo
3	caculateRentCost	int	Tính giá thuê xe

Bảng 4-40: Đặc tả các phương thức của SingleElectricBike

Parameter:

- bike – Thông tin xe dưới dạng String

Exception:

- Không

Method

Không

State

Không

4.4.3.25 Class “SingleNormalBike”

SingleNormalBike
+ SingleNormalBike() + SingleNormalBike(bike : String) + caculateRentCost(duration : int) : int

Hình 4-38: Biểu đồ lớp SingleNormalBike

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	SingleNormalBike	SingleNormalBike	Phương thức khởi tạo
2	SingleNormalBike	SingleNormalBike	Phương thức khởi tạo
3	caculateRentCost	int	Tính tiền thuê xe

Bảng 4-41: Đặc tả các phương thức của SingleNormalBike

Parameter:

- bike – Thông tin xe dưới dạng String

Exception:

- Không

Method

Không

State

Không

4.4.3.26 Class “BikeDAO”

BikeDAO
+ updateStatus(bikeID : int, inUse : boolean, dockID : int) : void + queryWithDockID(dockID : int) : String + queryWithBikeCode(bikeCode : int) : String + getBikes() : ArrayList<Bike>

Hình 4-39: Biểu đồ lớp BikeDAO

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	updateStatus	void	Cập nhật tình trạng xe (thuê hoặc chưa thuê)
2	queryWithDockID	String	Truy vấn Dock dựa vào DockID
3	queryWithBikeCode	String	Truy vấn Bike dựa vào BikeID
4	getBikes	ArrayList<Bike>	Lấy toàn bộ thông tin Bike

Bảng 4-42: Đặc tả các phương thức của BikeDAO

Parameter:

- bikeID – mã xe
- inUse – xe có đang sử dụng không
- dockID – mã bãi xe

Exception:

- Không

Method

Không

State

Không

4.4.3.27 Class “DBBinder”

DBBinder
- DB_URL : String = "jdbc:mysql://localhost:3306/ecoBikeN2"
- USER_NAME : String = "root"
- PASSWORD : String = ""
+ getConnection() : int
+ execute(command : String) : void
+ query(command : String) : String

Hình 4-40: Biểu đồ lớp DBBinder

Đặc tả các thuộc tính

#	Name	Data type	Default	Description
1	DB_URL	String	jdbc:mysql://localhost:3306/ecoBikeN2	Đường dẫn của Database
2	USER_NAME	String	“root”	Tên đăng nhập

3	PASSWORD	String		Mật khẩu
---	----------	--------	--	----------

Bảng 4-43: Đặc tả các thuộc tính của DBBinder

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	getConnection	int	Lấy kết nối
2	execute	void	Thực thi câu lệnh vào MySQL
3	query	String	Truy vấn dữ liệu vào MySQL

Bảng 4-44: Đặc tả các phương thức của DBBinder

Parameter:

- command – các câu lệnh MySQL mong muốn thực hiện.

Exception:

- Các exception khi sử dụng JDBC.

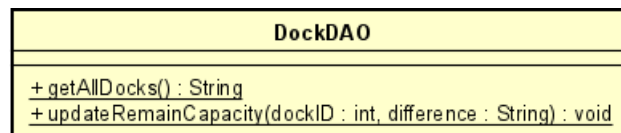
Method

Không

State

Không

4.4.3.28 Class “DockDAO”



Hình 4-41: Biểu đồ lớp DockDAO

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	getAllDocks	String	Lấy thông tin tất cả bãi xe
2	updateRemainCapacity	void	Cập nhật chỗ trống

Bảng 4-45: Đặc tả các phương thức của DockDAO

Parameter:

- dockID – mã bãi xe
- difference – sự thay đổi vào cơ sở dữ liệu mong muốn: “+1”/ “-1”, ...

Exception:

- Các exception khi sử dụng JDBC.

Method

Không

State

Không

4.4.3.29 Class “PaymentTransactionDAO”

PaymentTransactionDAO
+ save(rentalCode : String, cardCode : String, owner : String, transactionContent : String, amount : int, daytime : String) : void + checkCardInUse(cardCode : String) : boolean

Hình 4-42: Biểu đồ lớp PaymentTransactionDAO

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	save	void	Lưu giao dịch
2	checkCardInUse	boolean	Kiểm tra thẻ đang sử dụng

Bảng 4-46: Đặc tả các phương thức của PaymentTransactionDAO

Parameter:

- rentalCode – mã thuê xe
- cardCode – mã thẻ
- owner – người sở hữu
- transactionContent – nội dung giao dịch
- amount – tổng số tiền
- daytime – thời gian giao dịch

Exception:

- Các exception khi sử dụng JDBC.

Method

Không

State

Không

4.4.3.30 Class “RentBikeInvoiceDAO”

RentBikeInvoiceDAO
+ save(bikeID : int, rentalCode : String, type : String, rentTime : String, returnTime : String, rentBikeCost : int, owner : String, deposit : String) : void + queryByRentalCode(rentalCode : String) : String + updateAfterReturnBike(rentalCode : String, rentBikeCost : int, returnTime : String) : void

Hình 4-43: Biểu đồ lớp RentBikeInvoiceDAO

Đặc tả các thuộc tính

Không

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	save	void	Lưu hóa đơn
2	queryByRentalCode	String	Truy vấn bằng mã trả xe
3	updateAfterReturnBike	void	Cập nhật sau khi trả xe

Bảng 4-47: Đặc tả các phương thức của RentBikeInvoiceDAO

Parameter:

- bikeID – mã xe
- rentalCode – mã thuê xe
- type – loại xe
- owner – người sở hữu
- rentTime – thời gian bắt đầu thuê xe
- deposit – tiền cọc
- returnTime – thời gian trả xe
- rentBikeCost – giá thuê xe

Exception:

- Các exception khi sử dụng JDBC.

Method

Không

State

Không

4.4.3.31 Class “BikeDetailScreen”

BikeDetailScreen
- textArea : TextField - backButton : Button
~ setBikeInfo(bike : Bike, dock : Dock) : void + handleBackButtonClick() : void

Hình 4-44: Biểu đồ lớp BikeDetailScreen

Đặc tả các thuộc tính

#	Name	Data type	Description
1	textArea	TextField	Xâu ký tự hiển thị thông tin về xe
2	backButton	Button	Nút Back

Bảng 4-48: Đặc tả các thuộc tính của BikeDetailScreen**Đặc tả các phương thức**

#	Name	Return type	Description (purpose)
1	setBikeInfo	void	Thiết lập thông tin xe
2	handleBackButtonClick	void	Xử lý sự kiện nhấn nút Back

Bảng 4-49: Đặc tả các phương thức của BikeDetailScreen*Parameter:*

- bike – xe
- dock – bãi xe

Exception:

- Không

Method

Không

State

Không

4.4.3.32 Class “CardInfoScreen”

CardInfoScreen
- cardCode : TextField - owner : TextField - ccvCode : TextField - expDate : TextField - confirmButton : Button - cancelButton : Button
+ initialize(url : String, resourceBundle : ResourceBundle) : void + processRentBike(bike : Bike, cardInfoStage : Stage) : void

Hình 4-45: Biểu đồ lớp CardInfoScreen**Đặc tả các thuộc tính**

#	Name	Data type	Description
1	cardCode	TextField	Mã thẻ
2	owner	TextField	Người sở hữu

3	CVV	TextField	Cvv
4	expDate	TextField	Ngày hết hạn
5	confirmButton	Button	Phím xác nhận
6	cancelButton	Button	Phím hủy

Bảng 4-50: Đặc tả các thuộc tính của CardInfoScreen

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	initialize	void	Khởi tạo
2	processRentBike	void	Xử lý thuê xe

Bảng 4-51: Đặc tả các phương thức của CardInfoScreen

Parameter:

- url – đường dẫn tới fxml scene
- resourceBundle – tài nguyên đóng gói để khởi tạo fxml scene (sử dụng thư viện)
- bike – xe
- cardInfoStage – Stage để gọi tới CardInfoScreen

Exception:

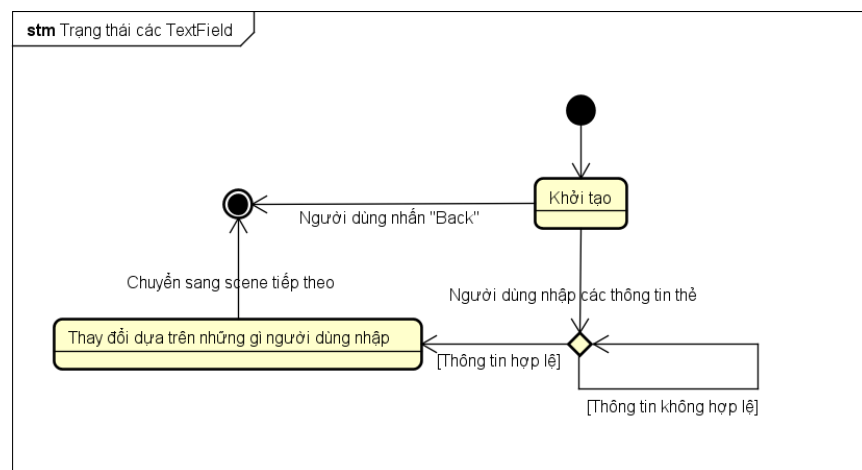
- Không

Method

Không

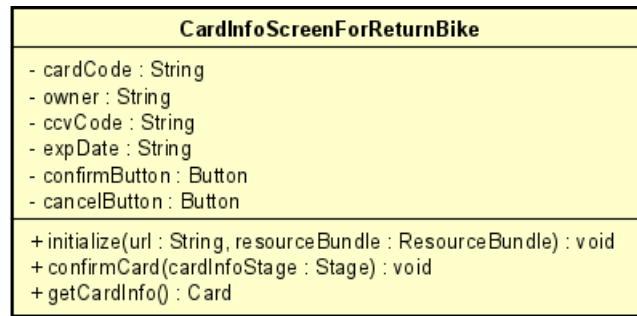
State

Trạng thái thay đổi các TextField khi người dùng nhập vào các thông tin về thẻ:



Hình 4-46: Biểu đồ trạng thái cho các Text Field

4.4.3.33 Class “CardInfoScreenForReturnBike”



Hình 4-47: Biểu đồ lớp CardInfoScreenForReturnBike

Đặc tả các thuộc tính

#	Name	Data type	Description
1	cardCode	String	Mã thẻ
2	owner	String	Người sở hữu
3	CVV	String	Cvv
4	expDate	String	Ngày hết hạn
5	confirmButton	Button	Phím xác nhận
6	cancelButton	Button	Phím hủy

Bảng 4-52: Đặc tả các thuộc tính của CardInfoScreenForReturnBike

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	initialize	void	Khởi tạo cho Screen
2	confirmCard	void	Xác nhận thẻ
3	getCardInfo	Card	Lấy thông tin thẻ

Bảng 4-53: Đặc tả các phương thức của CardInfoScreenForReturnBike

Parameter:

- url – đường dẫn tới fxml scene
- resourceBundle – tài nguyên đóng gói để khởi tạo fxml scene (sử dụng thư viện)
- bike – xe
- cardInfoStage – Stage để gọi tới CardInfoScreen

Exception:

- Không

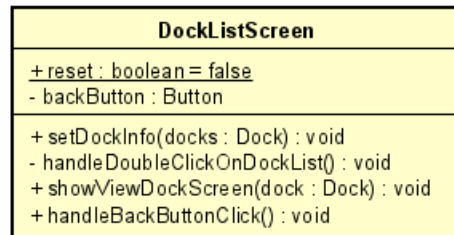
Method

Không

State

Không

4.4.3.34 Class “DockListScreen”



Hình 4-48: Biểu đồ lớp DockListScreen

Đặc tả các thuộc tính

#	Name	Data type	Default	Description
1	reset	boolean	false	Reset
2	backButton	Button		Nút Back

Bảng 4-54: Đặc tả các thuộc tính của DockListScreen

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	setDockInfo	void	Đặt thông tin bãi xe
2	handleDoubleClickOnDockList	void	Xử lý sự kiện nhấp 2 lần vào DockList
3	showViewDockScreen	void	Đưa ra màn hình ViewDockScreen
4	handleBackButtonClick	void	Xử lý sự kiện click nút Back

Bảng 4-55: Đặc tả các phương thức của DockListScreen

Parameter:

- dock – bãi xe

Exception:

- Không

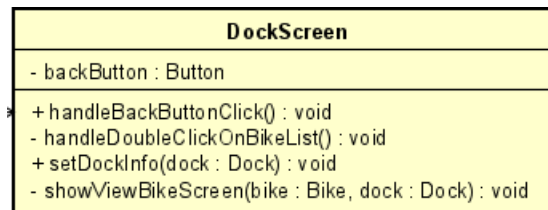
Method

Không

State

Không

4.4.3.35 Class “DockScreen”



Hình 4-49: Biểu đồ lớp DockScreen

Đặc tả các thuộc tính

#	Name	Data type	Description
1	backButton	Button	Nút Back

Bảng 4-56: Đặc tả các thuộc tính của DockScreen

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	handleBackButtonClick	void	Xử lý sự kiện click nút Back
2	handleDoubleClickOnBikeList	void	Xử lý sự kiện nhấp 2 lần vào BikeList
3	setDockInfo	void	Đặt thông tin bãi xe
4	showViewBikeScreen	void	Đưa ra màn hình ViewBikeScreen

Bảng 4-57: Đặc tả các phương thức của DockScreen

Parameter:

- bike – xe
- dock – bãi xe

Exception:

- Không

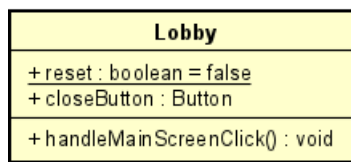
Method

Không

State

Không

4.4.3.36 Class “Lobby”



Hình 4-50: Biểu đồ lớp Dock

Đặc tả các thuộc tính

#	Name	Data type	Default	Description
1	reset	boolean	false	Reset
2	closeButton	Button		Nút Close

Bảng 4-58: Đặc tả các thuộc tính của Lobby

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	handleMainScreenClick	void	Xử lý sự kiện nhấn mainScreen

Bảng 4-59: Đặc tả các phương thức của Lobby

Parameter:

- Không

Exception:

- Không

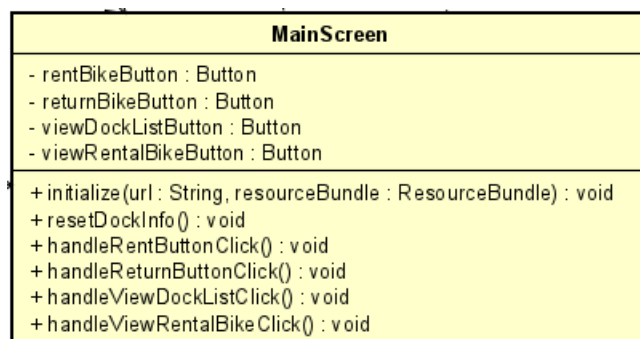
Method

Không

State

Không

4.4.3.37 Class “MainScreen”



Hình 4-51: Biểu đồ lớp MainScreen

Đặc tả các thuộc tính

#	Name	Data type	Description
1	rentBikeButton	Button	Nút thuê xe
2	returnBikeButton	Button	Nút trả xe
3	viewDockListButton	Button	Nút xem danh sách bãi xe
4	viewRentalBikeButton	Button	Nút xem xe đã thuê

Bảng 4-60: Đặc tả các thuộc tính của mainScreen

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	initialize	Dock	Khởi tạo mainScreen
2	resetDockInfo	float	Reset thông tin bãi xe
3	handleRentButtonClick	Bike	Xử lý sự kiện nhấn nút thuê xe
4	handleReturnButtonClick	String	Xử lý sự kiện nhấn nút trả xe
5	handleViewDockListClick	String	Xử lý sự kiện nhấn nút xem danh sách bãi xe
6	handleViewRentalBikeClick	boolean	Xử lý sự kiện nhấn nút ViewRentalBike

Bảng 4-61: Đặc tả các phương thức của mainScreen

Parameter:

- url – đường dẫn tới fxml scene
- resourceBundle – tài nguyên đóng gói để khởi tạo fxml scene (sử dụng thư viện)

Exception:

- Không

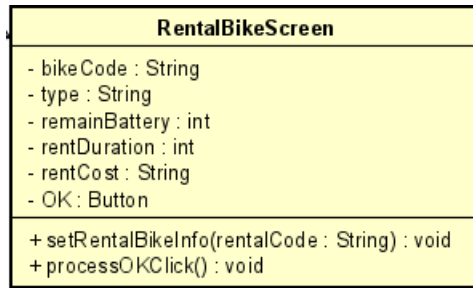
Method

Không

State

Không

4.4.3.38 Class “RentalBikeScreen”



Hình 4-52: Biểu đồ lớp RentalBikeScreen

Đặc tả các thuộc tính

#	Name	Data type	Description
1	bikeCode	String	Mã xe
2	type	String	Loại xe
3	remainBattery	int	Phần trăm pin còn lại
4	rentDuration	int	Thời gian thuê xe
5	rentCost	String	Giá thuê xe
6	OK	Button	Nút OK

Bảng 4-62: Đặc tả các thuộc tính của RentalBikeScreen

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	setRentalBikeInfo	void	Đặt thông tin trả xe
2	processOKClick	void	Xử lý sự kiện click nút OK

Bảng 4-63: Đặc tả các phương thức của RentalBikeScreen

Parameter:

- rentalCode – mã thuê xe

Exception:

- Không

Method

Không

State

Không

4.4.3.39 Class “RentBikeInfoScreen”

RentBikeInfoScreen
- textArea : TextField - confirmButton : Button - backButton : Button
+ initialize(url : String, resourceBundle : ResourceBundle) : void + setBikeInfo(bike : Bike, stage : Stage) : void + displayCardScreen(bike : Bike) : void + handleBackButtonClick() : void

Hình 4-53: Biểu đồ lớp RentBikeInfoScreen

Đặc tả các thuộc tính

#	Name	Data type	Description
1	textArea	TextField	Khu vực nhập barcode
2	confirmButton	Button	Nút xác nhận
3	backButton	Button	Nút Back

Bảng 4-64: Đặc tả các thuộc tính của RentBikeInfoScreen

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	initialize	void	Khởi tạo
2	setBikeInfo	void	Đặt thông tin xe
3	displayCardScreen	void	Hiển thị CardScreen
4	handleBackButtonClick	void	Xử lý sự kiện click nút Back

Bảng 4-65: Đặc tả các phương thức của RentBikeInfoScreen

Parameter:

- url – đường dẫn tới fxml scene
- resourceBundle – tài nguyên đóng gói để khởi tạo fxml scene (sử dụng thư viện)
- bike – xe
- stage – Stage để gọi tới RentBikeInfoScreen

Exception:

- Không

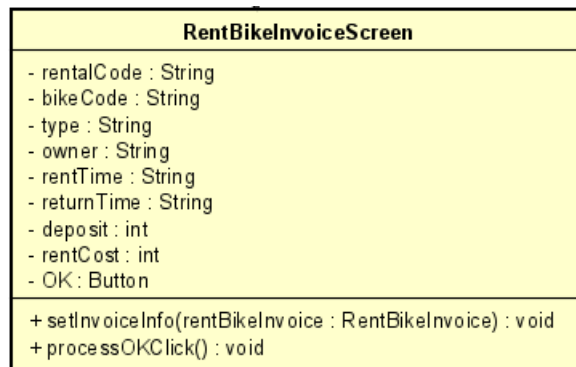
Method

Không

State

Không

4.4.3.40 Class “RentBikeInvoiceScreen”



Hình 4-54: Biểu đồ lớp RentBikeInvoiceScreen

Đặc tả các thuộc tính

#	Name	Data type	Description
1	rentalCode	String	Mã thuê xe
2	bikeCode	String	Mã xe
3	type	String	Loại xe
4	owner	String	Người sở hữu
5	rentTime	String	Thời gian thuê
6	returnTime	String	Thời gian trả
7	deposit	int	Tiền cọc
8	rentCost	int	Giá thuê xe
9	OK	Button	Nút OK

Bảng 4-66: Đặc tả các thuộc tính của RentBikeInvoiceScreen

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	setInvoiceInfo	void	Đặt thông tin hóa đơn
2	processOKClick	void	Xử lý sự kiện click nút OK

Bảng 4-67: Đặc tả các phương thức của RentBikeInvoiceScreen

Parameter:

- rentBikeInvoice – hóa đơn thuê xe

Exception:

- Không

Method

Không

State

Không

4.4.3.41 Class “RentBikeScreen”

RentBikeScreen
- barcode : String - backButton : Button
~ handleButtonRentBike(event : Event) : void + display(bike : Bike) : void + handleBackButton Click() : void

Hình 4-55: Biểu đồ lớp RentBikeScreen

Đặc tả các thuộc tính

#	Name	Data type	Description
1	barcode	String	Mã barcode
2	backButton	Button	Nút Back

Bảng 4-68: Đặc tả các thuộc tính của RentBikeScreen

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	handleButtonRentBike	void	Xử lý sự kiện nhấn nút thuê xe
2	display	void	Hiển thị màn hình nhập thẻ
3	handleBackButtonClick	void	Xử lý sự kiện nhấn nút Back

Bảng 4-69: Đặc tả các phương thức của RentBikeScreen

Parameter:

- event – sự kiện bắt được khi nhấn nút, sử dụng thư viện của JavaFX
- bike – xe muốn thuê

Exception:

- Không

Method

Không

State

Không

4.4.3.42 Class “ReturnBikeScreen”

ReturnBikeScreen
- rentalCode : String - backButton : Button
+ initialize(url : String, rb : ResourceBundle) : void - handleDoubleClickOnDockList() : void - getCardInfoForReturnBike() : Card - processReturnBike(rentalCode : int, card : Card, d : Dock) : void - showRentBikeInvoiceInfo(rentBikeInvoice : RentBikeInvoice) : void + setDockInfo(d : Dock, rentalCode : String) : void - getDockFromString(dockInfo : int) : Dock + handleBackButtonClick() : void

Hình 4-56: Biểu đồ lớp ReturnBikeScreen

Đặc tả các thuộc tính

#	Name	Data type	Description
1	rentalCode	String	Mã thuê xe
2	backButton	Button	Nút Back

Bảng 4-70: Đặc tả các thuộc tính của ReturnBikeScreen

Đặc tả các phương thức

#	Name	Return type	Description (purpose)
1	initialize	void	Khởi tạo
2	handleDoubleClickOnDockList	void	Xử lý nhấn 2 lần vào danh sách bãi xe
3	getCardInfoForReturnBike	Card	Lấy thông tin thẻ
4	processReturnBike	void	Xử lý trả xe
5	showRentBikeInvoiceInfo	void	Hiển thị thông tin hóa đơn thuê xe
6	setDockInfo	void	Đặt thông tin cho bãi xe
7	getDockFromString	Dock	Lấy thông tin bãi xe từ String
8	handleBackButtonClick	void	Xử lý nhấn nút Back

Bảng 4-71: Đặc tả các phương thức của ReturnBikeScreen

Parameter:

- url – đường dẫn tới fxml scene
- rb – tài nguyên đóng gói để khởi tạo fxml scene (sử dụng thư viện)
- rentalCode – mã thuê xe
- card – thẻ
- d – bãi xe
- rentBikeInvoice – hóa đơn thuê xe

Exception:

- Không

Method

Không

State

Không

5 Đánh giá bản thiết kế

5.1 Mục tiêu và định hướng

Mục tiêu: Sản phẩm được đánh giá tốt từ phía người dùng, ít phát sinh lỗi trong quá trình vận hành

Định hướng:

- Lập trình java tuân thủ theo convention, theo nguyên lý hướng đối tượng
- Viết java doc để tạo điều kiện thuận lợi cho việc bảo trì và cải tiến nếu có yêu cầu phát sinh trong tương lai
- Giải thích các thành phần code quan trọng đối với việc đảm bảo chức năng của một usecase trong lúc thực hiện

5.2 Chiến lược thiết kế

Công nghệ được sử dụng:

- Ngôn ngữ lập trình: Java (JDK-17)
- IDE: Eclipse (2022-12)
- Hệ quản trị cơ sở dữ liệu: MySQL 8.0

Phần mềm được thiết kế để đảm bảo các nguyên tắc:

- Tái sử dụng được mã nguồn
- Dễ mở rộng
- Dễ bảo trì

Xem xét các yêu cầu có thể phát sinh trong tương lai, ví dụ như:

- Thêm một loại xe mới
- Thay đổi cách tính giá thuê xe
- Thêm tính năng mới, ví dụ như khóa xe tạm thời không sử dụng
- Thêm các loại thẻ để thanh toán ví dụ như thẻ nội địa, ...

5.3 Đánh giá bản thiết kế theo các cấp độ Coupling & Cohesion

5.3.1 Đánh giá theo các cấp độ Coupling

5.3.1.1 Content coupling

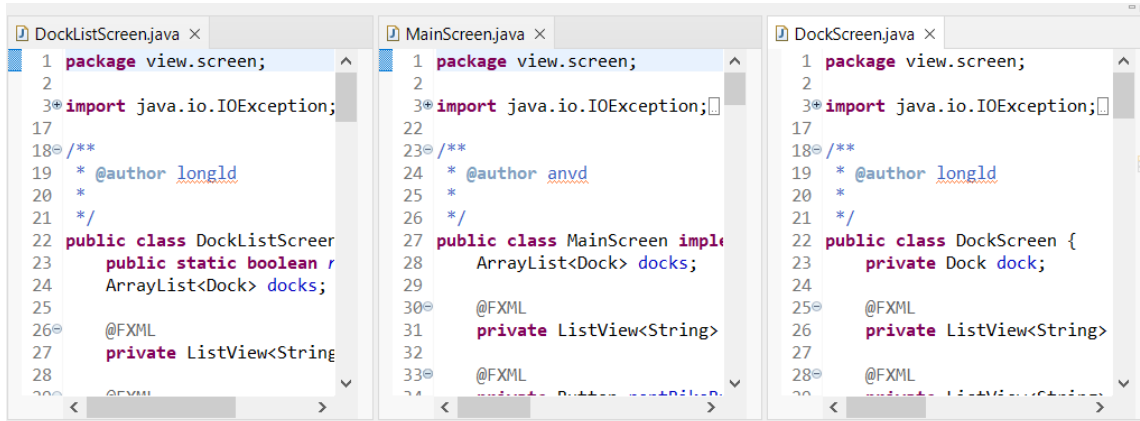
Content coupling là cấp độ coupling cao nhất và cần phải đặc biệt tránh trong quá trình phát triển phần mềm và bản thiết kế của nhóm cũng đã đảm bảo trong quá trình gọi lẫn nhau, các lớp không trực tiếp thay đổi giá trị của nhau.

Related Modules	Description	Improvement
-----------------	-------------	-------------

Không có		
----------	--	--

5.3.1.2 Common coupling

Thấp hơn content coupling, ta có common coupling, đó là khi có dữ liệu chung được sử dụng bởi nhiều thành phần. Cấp độ coupling này có xuất hiện trong bản thiết kế, đó là:

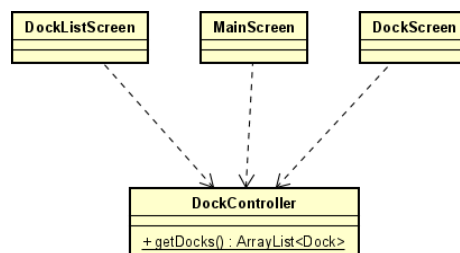


Hình 5-1: Common coupling trong các lớp view

Có thể thấy rằng, các 3 class mainScreen, DockScreen và DockListScreen đều chia sẻ chung biến dùng chung là docks và dock (lưu ý: dock là 1 thành viên của ArrayList <Dock> docks). Và trong quá trình gọi lẫn nhau, các biến này cũng được truyền qua các phương thức. Điều này sẽ có điểm yếu chung của common coupling đó là lỗi có thể bị lan truyền, nghĩa là nếu như docks đang có vấn đề từ mainScreen thì sẽ bị lan truyền tới các view DockListScreen và DockScreen.

Related Modules	Description	Improvement
view.screen	Các class mainScreen, DockScreen, DockListScreen đều chia sẻ biến dùng chung docks	Xóa các biến dùng chung này đi và sẽ sử dụng phương thức truy vấn các docks từ class DockController

Kết quả đạt được:



Hình 5-2: Thay đổi biến global docks bởi phương thức do DockController quản lý

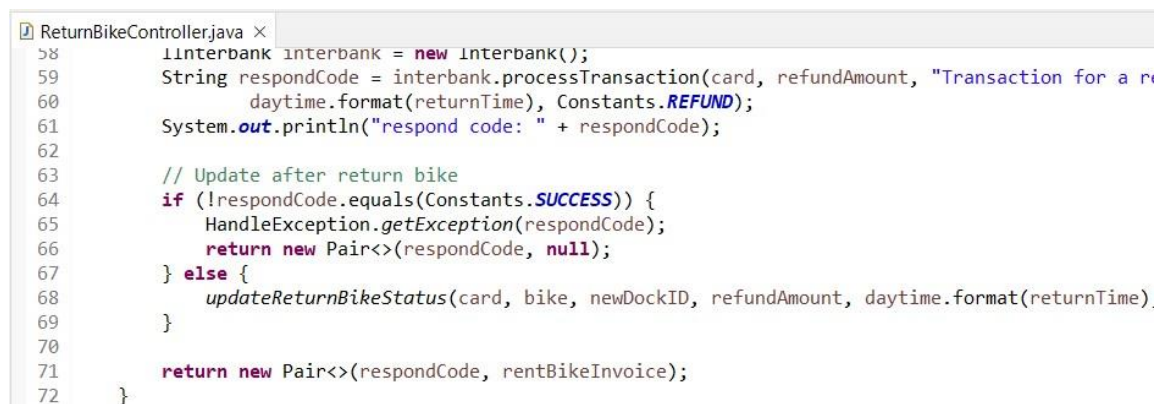
5.3.1.3 Control coupling

Trong lớp `util.HandleException`, bản thiết kế sử dụng phân tích dựa trên mã lỗi để đưa ra các thông điệp tương ứng với mã lỗi đó:

```
public static void getException(String code) throws RuntimeException {
    switch (code) {
        case "01":
            throw new InvalidCardException();
        case "02":
            throw new NotEnoughBalanceException();
        case "03":
            throw new InternalServerErrorException();
        case "04":
            throw new SuspiciousTransactionException();
        case "05":
            throw new NotEnoughTransactionInfoException();
        case "06":
            throw new InvalidVersionException();
        case "07":
            throw new InvalidTransactionAmountException();
        case Constants.IS_USE:
            throw new CardInUseException();
    }
}
```

Hình 5-3: Control coupling trong `HandleException`

Tuy nhiên, có thể thấy rằng, thiết kế này đã đủ đơn giản để có thể tránh việc `HandleException` phụ thuộc quá nhiều vào các `Exception`, lý do bởi xử lý của các câu lệnh `switch case` chỉ đơn giản dựa trên khởi tạo một đối tượng tương ứng. Bên cạnh đó, vẫn còn `control coupling` ở phương thức xử lý trong `ReturnbikeController`:



```
ReturnBikeController.java
58     Interbank interbank = new Interbank();
59     String respondCode = interbank.processTransaction(card, refundAmount, "Transaction for a r
60         daytime.format(returnTime), Constants.REFUND);
61     System.out.println("respond code: " + respondCode);
62
63     // Update after return bike
64     if (!respondCode.equals(Constants.SUCCESS)) {
65         HandleException.getException(respondCode);
66         return new Pair<>(respondCode, null);
67     } else {
68         updateReturnBikeStatus(card, bike, newDockID, refundAmount, daytime.format(returnTime)
69     }
70
71     return new Pair<>(respondCode, rentBikeInvoice);
72 }
```

Hình 5-4: Control coupling trong phương thức `processReturnBike` trong `ReturnBikeController`

Trong phương thức này có thể thấy, việc xử lý dựa trên mã trả về làm code rẽ nhánh 2 hướng là xử lý báo lỗi và xử lý thành công. Trong tương lai, có thể các thay đổi từ phần mềm khiến cho việc xử lý báo lỗi thay đổi, nó không còn đơn giản như chỉ là thông báo cho khách hàng mà có thể có nghiệp vụ khác nên chúng ta cần tách các khối xử lý `if-else` thành các chương trình riêng.

Related Modules	Description	Improvement
util.HandleException	Dựa trên mã lỗi để truy vấn ra thông điệp tương ứng.	Chấp nhận coupling này vì các phương thức xử lý cũng đủ đơn giản.
controller.ReturnBikeController	Phương thức xử lý phân tích mã code để xử lý	Có thể thêm một số prototype của các hàm xử lý riêng

5.3.1.4 Stamp coupling

Stamp coupling là cấp độ coupling chấp nhận được và hầu như đây là cấp độ mà nhóm sử dụng xuyên suốt trong bản thiết kế này.

```

ReturnBikeController.java ×
19 public class ReturnBikeController {
20
21     /**
22      * Process return bike by passing rental code, card and newDockID
23      *
24      * @param rentalCode: rental code of renting invoice
25      * @param card: credit card user uses to process transaction
26      * @param newDockID: id of the dock which user chooses to return bike
27      * @return respond code and rent bike invoice
28      */
29     public static Pair<String, RentBikeInvoice> processReturnBike(String rentalCode, Card card, String newDockID)
30         throws RuntimeException {

```

Hình 5-5: Stamp coupling trong ReturnBikeController

```

RentBikeController.java ×
34
35     /**
36      * Process a rent by passing card and bike arguments
37      *
38      * @param bike : bike wish to rent
39      * @return error code
40      */
41     public static String processRentBike(Card card, Bike bike) throws RuntimeException {

```

Hình 5-6: Stamp coupling trong RentBikeController

Ở trên là một số ví dụ stamp coupling trong module controller, hầu như các phương thức xử lý nghiệp vụ của các lớp controller đều làm theo stamp coupling, đó thực hiện bằng cách truyền các tham số là các đối tượng entity cần thao tác, ví dụ như để trả xe ta cần phải truyền: mã thuê xe, thẻ, ID của bãi xe mới muốn trả; ví dụ như để thuê xe ta cần phải truyền thông tin về thẻ, về xe, Về cơ bản, stamp coupling là cấp độ coupling chấp nhận được và việc mở rộng trong tương lai sẽ diễn ra nhẹ nhàng hơn.

Ngoài module controller, stamp coupling còn xuất hiện ở các lớp trừu tượng các giao dịch Interbank, giao dịch thực tế, ví dụ như:

```

InterbankTransaction.java x
46      * @param amount
47      * @param createdAt
48      */
49  public InterbankTransaction(Card card, String command, String content, int amount, String createdAt) {
50      this.setCardCode(card.getCardCode());
51      this.setOwner(card.getOwner());
52      this.setCvvCode(card.getCV());
53      this.setDateExpired(card.getExpiredDate());
54      this.setCommand(command);
55      this.setTransactionContent(content);
56      this.setAmount(amount);
57      this.setCreatedAt(createdAt);
58  }

```

Hình 5-7: Stamp coupling trong InterbankTransaction

Lớp InterbankTransaction có vai trò chỉ để trừu tượng hóa giao dịch đang thực hiện để đưa vào hàm bấm (do đặc tả API yêu cầu) nên nó chỉ yêu cầu truyền các đối tượng cần thiết rồi đóng gói thành một đối tượng. Trong tương lai, nếu đối tượng Card bị thay đổi nhưng một số thuộc tính cơ bản vẫn giữ nguyên thì hầu như InterbankTransaction sẽ không phải thay đổi.

Related Modules	Description	Improvement
controller	Các lớp trong controller đều được xây dựng dựa theo stamp coupling	Cấp độ này chấp nhận được, nhưng ta có thể improve bằng cách xem xét có thể truyền ID của xe thay vì truyền cả xe
entity.InterbankTransaction, entity.PaymentTransaction	Các lớp này cũng được xây dựng dựa theo stamp coupling.	Có thể chấp nhận cấp độ này

5.3.1.5 Data coupling

Data coupling là cấp độ coupling trên stamp coupling, đó là chúng ta chỉ sử dụng những thuộc tính cần thiết khi truyền tham số vào một phương thức. Có thể thấy một số phần trong bản thiết kế có được cấp độ coupling này, điển hình nhất ở các phương thức validate tại CardController:

```

CardController.java x
28      * @return right format (true) and otherwise (false)
29      */
30  public static boolean validateCardInfo(String carCode, String owner, String ccvCode, String expiredDate) {
31      if (validateCardCode(carCode) && validateOwner(owner) && validateCcvCode(ccvCode)
32          && validateExpiredDate(expiredDate)) {
33          return true;
34      } else {
35          return false;
36      }
37  }

```

Hình 5-8: Ví dụ về data coupling trong validateCardInfo trong CardController

Đối với phương thức này, ta chỉ cần truyền vào các giá trị mà người dùng đã nhập, chính vì vậy, trong tương lai, dù ta có thay đổi các GUI, các form khác nhưng chỉ cần truyền các tham số là thông tin người dùng nhập thì các phương thức validate này sẽ không phải thay đổi

Related Modules	Description	Improvement
controller	Các phương thức validate trong controller.CardController	Cấp độ coupling chấp nhận được

5.4 Đánh giá theo các cấp độ Cohesion

Ta đánh giá việc kết nối các lớp trong module dựa vào các mối quan hệ: dependency, association, ... và ý nghĩa của các mối quan hệ đó.

5.4.1.1 Coincidental cohesion

Đây là cấp độ cohesion lỏng lẻo nhất, các lớp trong modules hầu như không hề liên quan gì tới nhau. Trong bản thiết kế của nhóm, có thể dễ dàng nhận ra hai gói mà không có sự liên quan gì giữa các lớp đó là util và exception.

Tuy nhiên, có thể thấy rằng, đôi khi chúng ta vẫn sẽ chấp nhận trường hợp coincidental này, nhưng chỉ trong một số trường hợp. Ví dụ, ta có thể thấy rằng trong gói thư viện java.util, ta vừa có Random (để sinh số ngẫu nhiên) và ArrayList (cấu trúc dữ liệu) không liên quan tới nhau nhưng chúng ta vẫn chấp nhận chúng ở trong cùng gói vì chúng cung cấp những phương thức có tính “thủ tục” và sử dụng rộng rãi ở mọi nơi. Một cách tương tự, ta có thể thấy rằng, các lớp trong util cung cấp các chức năng mang tính được sử dụng nhiều bởi hệ thống, như là Constants, HTTPBinder, Tương tự, các exception cũng cung cấp cho chúng ta những đối tượng để có thể đóng gói các thông báo lỗi tiện lợi hơn.

Related Modules	Description	Improvement
util	Các lớp không có liên kết với nhau	Chấp nhận vì các lớp này được sử dụng rộng rãi trên toàn hệ thống
exception	Các lớp không có liên kết với nhau	Chấp nhận vì các lớp này được sử dụng nhiều trong HandleException

5.4.1.2 Logical cohesion

Cấp độ này nói lên rằng các thành phần chỉ liên quan tới nhau về mặt logic chứ không liên quan tới nhau về bản chất hoạt động, có thể nhận thấy cấp độ này xuất hiện ở HTTPBinder như sau:

```

HTTPBinder.java x
23  * @param url
24  * @param body
25  * @return
26  */
27  public String patch(String url, String body) {
28      try {
29          RequestBody requestBody = RequestBody
30              .create(MediaType.parse(org.springframework.http.MediaType.APPLICATION_JSON_VALUE), body)
31          Request request = new Request.Builder().url(url).patch(requestBody).build();
32          Response response = client.newCall(request).execute();
33          assert response.body() != null;
34          JSONObject returnJson = new JSONObject(response.body().string());
35          return returnJson.getString("errorCode");
36      } catch (Exception e) {
37          System.out.println("Exception at utils.HTTPConnector.sendPatch");
38      }
39      return null;
40  }

HTTPBinder.java x
44  * @param url
45  * @param body
46  * @return
47  */
48  public String post(String url, String body) {
49      try {
50          RequestBody requestBody = RequestBody
51              .create(MediaType.parse(org.springframework.http.MediaType.APPLICATION_JSON_VALUE), body)
52          Request request = new Request.Builder().url(url).post(requestBody).build();
53          Response response = client.newCall(request).execute();
54          assert response.body() != null;
55          JSONObject returnJson = new JSONObject(response.body().string());
56          return returnJson.getString("id");
57      } catch (Exception e) {
58          System.out.println("Exception at HTTPConnector.post");
59      }
60      return null;

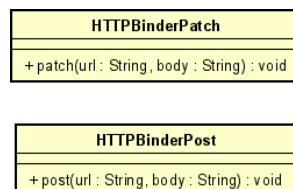
```

Hình 5-9: 2 phương thức patch và post trong HTTPBinder

Có thể thấy rằng, hai phương thức patch và post về bản chất khác nhau và cách thức triển khai mã nguồn cũng khác nhau, song vẫn chung trên một mã nguồn. Chúng chỉ liên quan tới nhau về mặt logic chứ độc lập với nhau, vì vậy, ta có thể tách ra làm các phương thức HTTPBinderPost và HTTPBinderPatch

Related Modules	Description	Improvement
util.HTTPBinder	Các phương thức post và patch độc lập với nhau về mặt thực hiện, chỉ liên quan tới nhau về mặt logic	Tách ra làm hai lớp HTTPBinderPost và HTTPBinderPatch

Kết quả đạt được:



Hình 5-10: Redesign HTTPBinder

5.4.1.3 Temporal cohesion

Các phương thức khởi tạo một số thuộc tính trong các view DockListScreen, DockScreen. Ta có thể thấy rằng, ở các view này, sẽ có một số phương thức nhằm để khởi tạo dữ liệu về bãi xe, xe, Việc khởi tạo chỉ diễn ra 1 lần và sau đó, hầu như chỉ tập trung để xử lý các yêu cầu từ người dùng. Vì vậy, đặt hàm này ở trong các lớp này sẽ tạo ra temporal cohesion.

```
DockListScreen.java ×
35  */
36  public void setDockInfo(ArrayList<Dock> docks) {
37      this.docks = docks;
38      // show list of docks
39      for (Dock dock : docks) {
40          docksView.getItems().add(dock.toString());
41      }
42
43      // listen when user double click on the dock in listview => show ViewDockScreen
44      docksView.setOnMouseClicked(click -> {
45          if (click.getClickCount() == 2) {
46              handleDoubleClickOnDockList();
47          }
48      });
49  }
```

Hình 5-11: Temporal cohesion trong DockListScreen

Ta có thể cải thiện bằng cách sử dụng Interface Initialize của thư viện JavaFX, cho phép chúng ta khởi tạo các lớp điều khiển view từ đầu.

Related Modules	Description	Improvement
view	Các view thường gặp phải tình trạng có một hàm khởi tạo 1 lần sau đó không dùng nữa	Kết hợp hàm khởi tạo này vào Interface Initialize của JavaFX

5.4.1.4 Procedural cohesion

Hầu hết các lớp trong gói controller đều được thiết kế theo mức cohesion này. Lấy ví dụ về lớp CardController, ta có thể phân chia chức năng của CardController thành các giai đoạn như sau: giai đoạn đầu: kiểm tra xem thẻ nhập đã hợp lệ hay chưa, giai đoạn hai: kiểm tra thẻ đã đang được sử dụng hay chưa, giai đoạn cuối: nhờ API Interbank để thực hiện giao dịch. Dựa trên quy trình này, mã nguồn của nhóm thiết kế trong các gói controller cũng dựa vào cách thức này.

```

CardController.java x
28      * @return right format (true) and otherwise (false)
29      */
30      public static boolean validateCardInfo(String carCode, String owner, String ccvCode, String expiredDate)
31      {
32          if (validateCardCode(carCode) && validateOwner(owner) && validateCcvCode(ccvCode)
33              && validateExpiredDate(expiredDate)) {
34              return true;
35          } else {
36              return false;
37          }
38      }
39
40      CardController.java x
41      16      */
42      17      public static boolean checkCardInUse(String cardCode) {
43      18          return PaymentTransactionDAO.checkCardInUse(cardCode);
44      19      }
45      20

```

Hình 5-12: Các giai đoạn triển khai của CardController

Related Modules	Description	Improvement
controller	Các lớp trong controller được thiết kế thực hiện nghiệp vụ thành từng giai đoạn	Nên có một thống nhất chung về dữ liệu được sử dụng để tăng tính cohesion

5.4.1.5 Communication cohesion

Ta có thể thấy rằng Communication cohesion xuất hiện ở HTTPBinder với hai phương thức là post và patch, hai phương thức này đều thực hiện trên dữ liệu như nhau và chỉ khác về triển khai bên trong mã nguồn

Related Modules	Description	Improvement
util.HTTPBinder	Phương thức post và patch thực hiện trên cùng dữ liệu	Mức cohesion này chấp nhận được

5.4.1.6 Sequential cohesion

Có thể thấy mức cohesion này xuất hiện khi HashFunction nhận đầu vào là một giao dịch cần phải băm và xuất đầu ra là xâu ký tự cần đáp ứng cho API:

```

HashFunction.java x
21  * @param transaction
22  * @return
23  */
24  public static String hashTransaction(String key, InterbankTransaction transaction) {
25      try {
26          String transact = new ObjectMapper().writeValueAsString(transaction);
27          @SuppressWarnings("deprecation")
28          JSONObject transactionBody = new JsonParser().parse(transact).getAsJsonObject();
29
30          // convert to request transaction
31          JSONObject transToHash = new JsonObject();
32          transToHash.addProperty("secretKey", key);
33          transToHash.add("transaction", transactionBody);
34          MessageDigest md = null;
35          try {
36              md = MessageDigest.getInstance("MD5");
37          } catch (Exception e) {
38              e.printStackTrace();

```

Hình 5-13: Hàm xử lý băm giao dịch

Đầu ra của hàm này chính là thông tin cần phải gửi qua API. Mặc dù HashFunction nằm ở gói util và lớp sử dụng nó là ProcessTransaction ở subsystem.interbank.api nhưng ta vẫn xét cohesion cho 2 lớp này, bởi vì, như đã phân tích ở phần trước, các lớp triển khai trong util mang tính sử dụng rộng rãi bởi hệ thống và hầu như nó sẽ liên quan tới tất cả các công việc.

Related Modules	Description	Improvement
util.HashFunction	Phương thức hashTransaction nhận đầu vào là thông tin giao dịch đóng gói và đầu ra sẽ là dữ liệu gửi lên API	Mức cohesion này chấp nhận được

5.4.1.7 Functional cohesion

Mức cohesion này chỉ xuất hiện ở các subsystem, về cơ bản, các subsystem có tính cohesion rất tốt, có thể thấy qua các giao diện triển khai:

```

10 public interface IInterbank {
11     /**
12      * process transaction
13      * @param card
14      * @param amount
15      * @param content
16      * @param createdAt
17      * @param command
18      * @return
19      */
20     String processTransaction(Card card, int amount, String content, String createdAt, String command);
21
22     /**
23      * reset card
24      * @param card
25      * @return
26      */
27     String reset(Card card);
28 }

```

Hình 5-14: Prototype Interface của Interbank

```

11 public class Interbank implements IInterbank {
12     /**
13      * reset card
14      */
15     @Override
16     public String reset(Card card) {
17         ResetTransaction resetTransaction = new ResetTransaction(card);
18         return resetTransaction.patch();
19     }
20
21     /**
22      * process transaction
23      */
24     @Override
25     public String processTransaction(Card card, int amount, String content, String createdAt, String command)
26     {
27         ProcessTransaction processTransaction = new ProcessTransaction(card, amount, content, createdAt, command);
28         return processTransaction.patch();
29     }
30 }

```

Hình 5-15: Triển khai của lớp thực thi Interface IInterbank

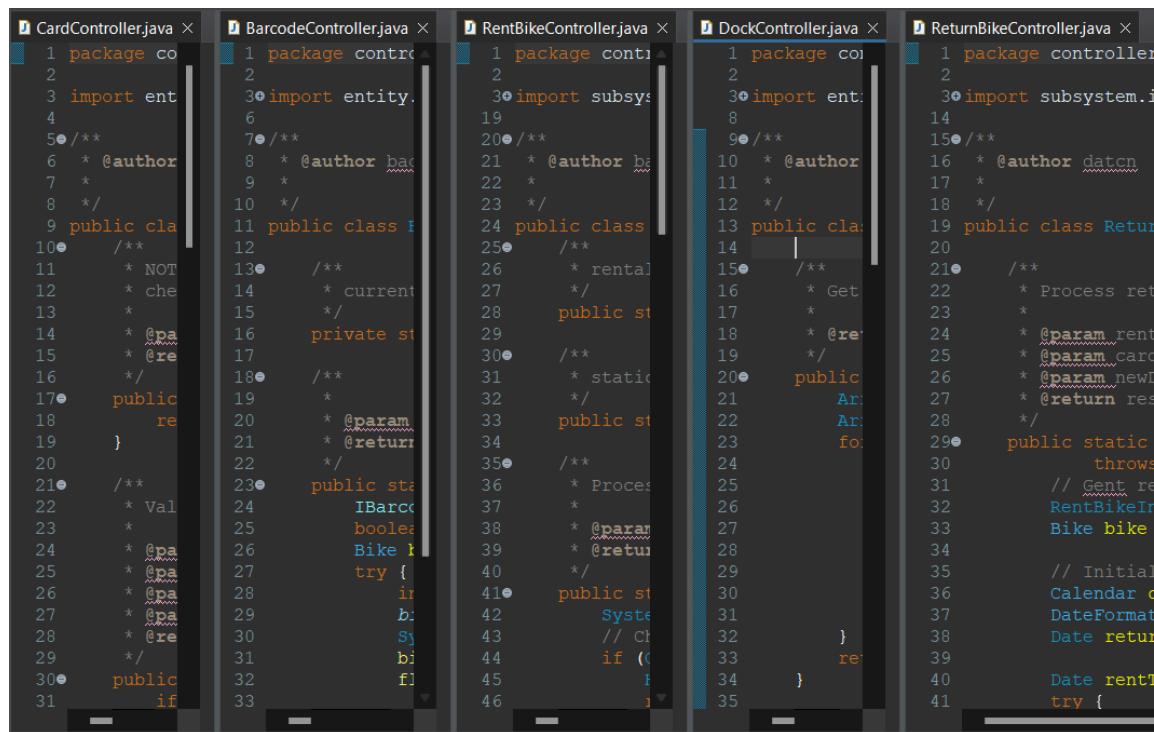
Có thể thấy, các lớp ResetTransaction và ProcessTransaction đều phục vụ chung cho mục đích của subsystem.

Related Modules	Description	Improvement
subsystem	Các hệ thống con triển khai đều có các lớp thành phần hướng đến mục đích chung của nó	Mức cohesion này chấp nhận được

5.5 Các nguyên lý trong thiết kế

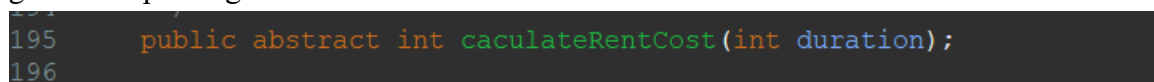
5.5.1 Single Responsibility Principle

Đối với nguyên lý mỗi lớp chỉ nên thực hiện một chức năng, nhóm luôn lấy nguyên lý này để xây dựng các lớp trong gói controller.



Hình 5-16: Mỗi controller trong gói controller phụ trách duy nhất một chức năng

Ví dụ, CardController phụ trách nhiệm vụ về kiểm tra thẻ đang được sử dụng hay chưa, kiểm tra tính hợp lệ của thẻ, lớp BarcodeController phụ trách nhiệm vụ chuyển đổi barcode, Tuy nhiên, ta thấy rằng, trong gói entity thì có các lớp Bike đều triển khai phương thức ghi đè lên phương thức abstract đó là:



Hình 5-17: Phương thức calculateRentCost của các lớp kế thừa Bike

Điều này làm cho các lớp kế thừa Bike sẽ đảm nhiệm hai trách nhiệm, một là trừu tượng hóa những đối tượng thực tế, hai là phải thực hiện trách nhiệm tính toán giá thuê xe.

Related Modules	Description	Improvement
controller	Các lớp được thiết kế để phụ trách một chức năng duy nhất	Chấp nhận được
entity.Bike, ...	Các lớp vừa thực hiện chức năng trừu tượng hóa các thực thể bên	Cần phải tách ra thêm làm các lớp phụ trách chức năng tính

	ngoài, vừa làm chức năng tính toán giá thuê xe	toán giá thuê xe, nên đặt các lớp này ở controller
--	--	--

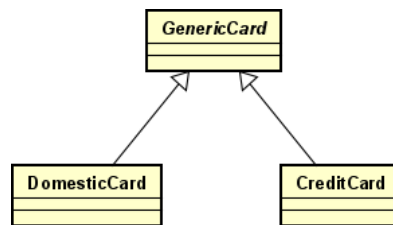
Kết quả đạt được: Tương tự như mẫu Strategy sẽ được trình bày ở mục 5.6.

5.5.2 Open/Closed Principle

Có thể thấy rằng, trong tương lai, rất có thể lớp Card sẽ bị thay thế bởi hệ thống chấp nhận thanh toán bằng nhiều loại thẻ hơn, chính vì vậy, cần phải tạo ra các lớp trừu tượng để có thể khi cần phải thêm một loại thẻ mới, chúng ta chỉ cần thêm các lớp để triển khai kế thừa lớp Card đó. Bên cạnh đó, có thể thấy rằng, thiết kế các lớp Bike đang thỏa mãn nguyên lý này, lớp Bike là lớp trừu tượng và việc thêm các loại xe khác trong tương lai sẽ dễ dàng hơn.

Related Modules	Description	Improvement
entity.Card	Rất khó khăn cho việc mở rộng thêm các loại thẻ	Cần phải thay thế bởi các lớp trừu tượng Card để có thể mở rộng dễ dàng hơn trong tương lai
entity.Bike	Đã sử dụng lớp trừu tượng, việc mở rộng trong tương lai đơn giản hơn	Chấp nhận được

Kết quả đạt được:



Hình 5-18: Thay thế Card bởi lớp trừu tượng

5.5.3 Liskov Substitution Principle

Nguyên lý này nói lên rằng, các lớp con có thể thay thế được cho các lớp cha. Có thể thấy rằng, các lớp kế thừa Bike như SingleNormalBike, DoubleNormalBike, ElectricBike đều có thể thay thế lớp cha là Bike.

Related Modules	Description	Improvement
entity.Bike, ...	Các lớp con kế thừa Bike đều có thể thay thế được lớp cha	Chấp nhận được

5.5.4 Interface Segregation

Nguyên lý này nói lên rằng, chúng ta nên để các Interface chỉ phục vụ 1 số chức năng nhất định, vì vậy, chúng ta nên cân nhắc lớp Interbank như sau:

```
IInterbank.java ×
9  */
10 public interface IInterbank {
11     /**
12      * process transaction
13      * @param card
14      * @param amount
15      * @param content
16      * @param createdAt
17      * @param command
18      * @return
19      */
20     String processTransaction(Card card, int amount, String content, String createdAt, String command);
21
22     /**
23      * reset card
24      * @param card
25      * @return
26      */
27     String reset(Card card);
28 }
```

Hình 5-19: Các phương thức trong IInterbank

Có thể thấy rằng, hai phương thức processTransaction và reset cũng không thực sự liên quan tới nhau, reset chỉ được sử dụng trong quá trình phát triển phần mềm và không phục vụ liên quan đến nghiệp vụ người dùng. Tuy nhiên, theo ý kiến chủ quan, chúng ta vẫn hoàn toàn có thể để hai phương thức này trong một giao diện, lý do bởi chúng phù hợp với Functional Cohesion, đó là hai phương thức này cùng thực hiện chức năng chung đó là cung cấp một lớp biên có các hành vi giao tiếp với API bên ngoài.

Related Modules	Description	Improvement
subsystem	Cần xem xét để tách các chức năng khỏi các Interface	Tuy nhiên, ở mức độ phát triển phiên bản phần mềm này là chấp nhận được.

5.5.5 Dependency Inversion

Nguyên lý này nói lên rằng, các thành phần chỉ nên phụ thuộc vào các thành phần trừu tượng khác, ta có thể thấy rằng, hiện tại InterbankTransaction và PaymentTransaction đều chưa làm được điều này.

```

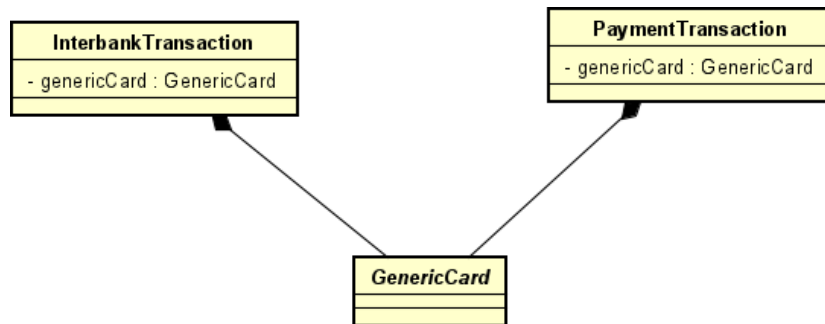
InterbankTransaction.java x
46     * @param amount
47     * @param createdAt
48     */
49     public InterbankTransaction(Card card, String command, {
50         this.setCardCode(card.getCardCode());
51         this.setOwner(card.getOwner());
52         this.setCvvCode(card.getCW());
53         this.setDateExpired(card.getExpiredDate());
54         this.setCommand(command);
55         this.setTransactionContent(content);
56         this.setAmount(amount);
57         this.setCreatedAt(createdAt);
58     }

```

Hình 5-20: InterbankTransaction phụ thuộc vào lớp Card

Related Modules	Description	Improvement
entity	InterbankTransaction, PaymentTransaction phụ thuộc vào lớp Card	Cần phải thiết kế để phụ thuộc vào lớp trừu tượng

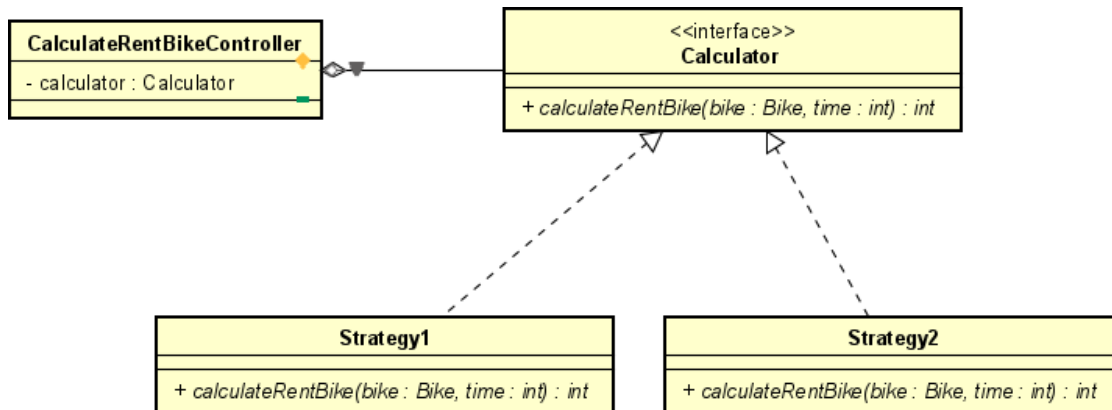
Kết quả đạt được:



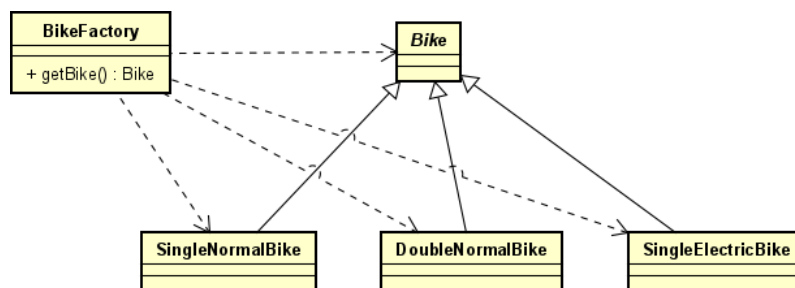
Hình 5-21: Thay sự kết nối Card bằng GenericCard

5.6 Các mẫu thiết kế (Design Pattern)

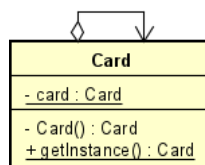
Hiện tại, bản thiết kế của nhóm không sử dụng các mẫu thiết kế, đây là điều thiếu sót lớn vì việc không tận dụng các mẫu thiết kế sẽ khiến cho thiết kế đôi khi trở nên thiếu coupling và cohesion cần thiết. Tuy nhiên, để sử dụng mẫu thiết kế đòi hỏi kinh nghiệm cũng như hiểu biết tường tận về vấn đề này. Có thể trong tương lai, một số mẫu thiết kế sẽ được tận dụng để giải quyết những hạn chế như đã trình bày ở hai mục trước đó. Dưới đây là một số ví dụ về các Design Pattern có thể được sử dụng:



Hình 5-22: Strategy để thực hiện tính giá thuê xe



Hình 5-23: Factory với lớp Bike



Hình 5-24: Singleton với lớp Card

Đối với lớp Card, trong khuôn khổ phát triển phần mềm phiên bản này, ta chỉ cần quan tâm đến một thẻ duy nhất được cung cấp, chính vì vậy, để đảm bảo thẻ không bị thay đổi trong suốt quá trình sử dụng, ta có thể cân nhắc thêm Singleton.