**ITransactionBilling**

Xác định các phần tử thiết kế

**Thực hiện bởi**

**Nguyễn Đăng Quang  
Vũ Thị Thành Vinh  
Cao Tiến Thắng  
Hoàng Văn Quyền  
Nguyễn Vũ Thanh Tùng**

**OOAD Nhóm 3**

**21 tháng 4 năm 2024**

**Mục lục**

[**1. Giới thiệu**](#_heading=h.3znysh7) **3**

[1.1 Mục đích](#_heading=h.2et92p0) 3

[1.2 Người đọc tài liệu và hướng dẫn đọc](#_heading=h.tyjcwt) 3

[1.3 Phạm vi dự án](#_heading=h.3dy6vkm) 4

[1.4 Tham chiếu](#_heading=h.4d34og8) 4

[2.1.1 Subsystem Diagram](#_heading=h.yfjz5rocu7ii) 5

[2.1.2 Subsystem Interface Description](#_heading=h.vg60mw73fvuv) 5

# Giới thiệu

## Mục đích

Đây là tài liệu **Xác định các phần tử thiết kế** cho môn học **Phân tích và thiết kế hướng đối tượng - 2324II\_INT3110\_3**.

Báo cáo này được viết dựa trên chuẩn “IEEE Std 830-1998 IEEE Recommended Practice

for Software Requirements Specifications”. Các nội dung trong báo cáo đã được các thành viên trong nhóm đồng thuận vào cuộc họp Online tối ngày 22 tháng 4 năm 2024.

## Người đọc tài liệu và hướng dẫn đọc

Tài liệu này dành cho:

* Quản lý dự án: Người quản lý dự án quản lý và chịu trách nhiệm về chất lượng của hệ thống. Quản lý dự án nên đọc toàn bộ tài liệu để lập kế hoạch và phân công công việc.
* Nhà phát triển: Nhà phát triển là người triển khai hệ thống từ thiết kế và tài liệu thành một phiên bản có thể chạy được. Nhà phát triển phải đọc toàn bộ tài liệu để triển khai hệ thống đúng cách.
* Người kiểm thử: Người kiểm thử nên đọc chi tiết để viết kiểm thử đơn vị đặc biệt.
* Người viết tài liệu: Những người sẽ viết tài liệu trong tương lai (báo cáo, biên bản cuộc họp). Nhà viết tài liệu nên đọc để hiểu phần Biểu đồ Sử dụng chính.

Nội dung của báo cáo bao gồm 5 phần chính:

* **Part 1 Subsystem Context Diagrams**: Các Sơ đồ Ngữ cảnh của hệ thống Phân tích được sử dụng để thiết kế các phụ hệ thống (Subsystem) để đóng gói hành vi trong một "gói" (package) cung cấp các giao diện rõ ràng và hình thức, và (theo quy ước) không tiết lộ bất kỳ nội dung nội bộ nào. Nó được sử dụng như một đơn vị hành vi trong hệ thống, cung cấp khả năng đóng gói hoàn toàn các tương tác của một số lớp và/hoặc phụ hệ thống. Khả năng "đóng gói" của các phụ hệ thống thiết kế được so sánh với Artifacts: Design Package, không thực hiện giao diện và có thể tiết lộ nội dung được đánh dấu là "public". Các gói được sử dụng chủ yếu cho quản lý cấu hình và tổ chức mô hình, trong đó các phụ hệ thống cung cấp các ngữ nghĩa hành vi bổ sung.
* **Part 2 Analysis Class to Design Element Map**
* **Part 3 Design Element to Owning Package Map**
* **Part 4 Architectural Layers and Their Dependencies**: Kiến trúc hóa cung cấp một phân chia logic của các phụ hệ thống thành một số tập hợp, với các quy tắc nhất định về cách tạo ra mối quan hệ giữa các lớp. Kiến trúc hóa cung cấp một cách để hạn chế sự phụ thuộc giữa các phụ hệ thống, kết quả là hệ thống được liên kết lỏng hơn và dễ bảo trì hơn.
* **Part 5 Packages and Their Dependencies**: Một gói thiết kế và nội dung của nó là trách nhiệm của một Vai trò duy nhất: Nhà thiết kế. Các phần tử trong gói có thể phụ thuộc vào các phần tử được chứa bởi các gói khác; điều này dẫn đến sự phụ thuộc giữa các gói. Sự phụ thuộc giữa các gói có thể được sử dụng như một công cụ để phân tích tính linh hoạt của mô hình thiết kế: một mô hình có các gói phụ thuộc chéo sẽ ít linh hoạt đối với sự thay đổi.

## Phạm vi dự án

Ban đầu, người dùng của phần mềm là các khách hàng và lái xe. Phần mềm sẽ tạo ra môi trường để khách hàng có thể đặt xe để di chuyển hoặc để ship hàng.

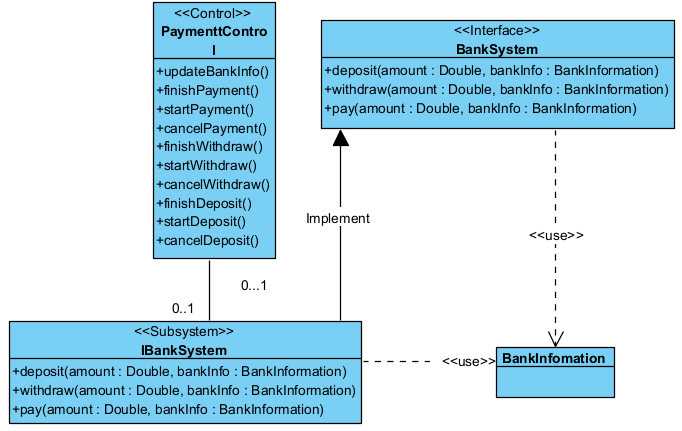
Bên cạnh đó, phần mềm cũng hướng tới những người muốn kiếm thêm thu nhập bằng cách đăng ký làm tài xế trên hệ thống, nhận chở khách và ship hàng theo yêu cầu của khách hàng.

## Tham chiếu

[1] Form of presentation IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

1. **Subsystem Context Diagrams**
   1. **Payment System Subsystem**

## Subsystem Diagram

****

## Subsystem Interface Description

IbankSystem: Quản lý sự giao tiếp giữa các hệ thống banking

deposit: nạp tiền với thông tin tài khoản và ngân hàng trong BankInformation

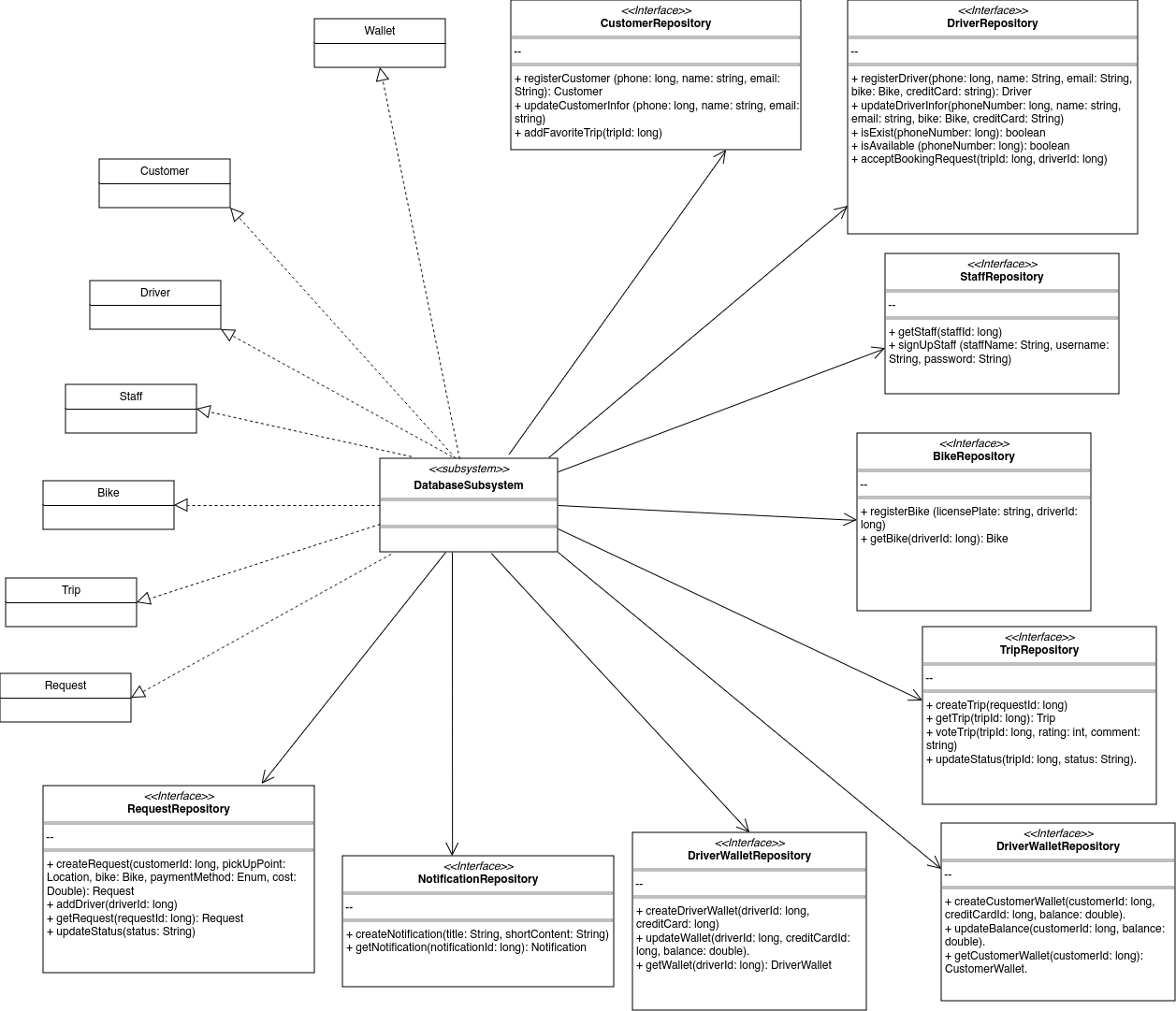
withdraw: rút tiền với thông tin tài khoản và ngân hàng trong BankInformation

pay: trả tiền với thông tin tài khoản và ngân hàng trong BankInformation

* 1. **Database Subsystem**

Database Subsystem cung cấp hỗ trợ cho cơ sở dữ liệu quan hệ được viết bằng ngôn

ngữ SQL. Hệ thống con được thiết kế như sau:

****

1. **Analysis-Class-to-Design-Element Map**

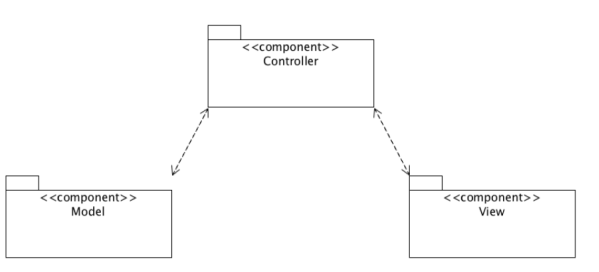
| **STT** | **Lớp phân tích** | **Phần tử thiết kế** |
| --- | --- | --- |
| 1 | DatabaseSystem | DatabaseSubsystem |
| 2 | DatabaseSubsystemsubsCustomer | ICustomerDatabase  IDriverDatabase  IStaffDatabase  IRequestDatabase  ITripDatabase  IWalletDatabase  INotificationDatabase |
| 3 | Customer | Customer, DatabaseSubsystem |
| 4 | Driver | Driver, DatabaseSubsystem |
| 5 | Staff | Staff, DatabaseSubsystem |
| 6 | Request | Request, DatabaseSubsystem |
| 7 | Trip | Trip, DatabaseSubsystem |
| 8 | WalletSystem | IWalletSystem |
| 9 | DriverWallet | DriverWallet  WalletSystem  DatabaseSystem |
| 10 | CustomerWallet | CustomerWallet  WalletSystem  DatabaseSystem |
| 11 | Notification | Notification, DatabaseSubsystem |
| 12 | TransactionBilling | TransactionBilling  BillingSystem |
| 13 | TripBilling | TripBilling  BillingSystem |
| 14 | BillingSystem | IBillingSystem  ITransactionBilling |
| 15 | MapAPI | MapAPI |
| 16 | LoginController | LoginController |
| 17 | BookingController | BookingController |
| 18 | RouteTrackingController | RouteTrackingController |
| 19 | FeedbackController | FeedbackController |
| 20 | ManageWalletController | ManageWalletController |
| 21 | AcceptRequestController | AcceptRequestController |
| 22 | PaymentController | PaymentController |
| 23 | ViewNotificationController | ViewNotificationController |
| 24 | ViewTripHistoryController | ViewTripHistoryController |
| 25 | ManageDriverInfoController | ManageDriverInfoController |
| 26 | ManageCustomerInfoController | ManageCustomerInfoController |
| 27 | CreateNotificationController | CreateNotificationController |
| 28 | LoginForm | LoginForm |
| 29 | CustomerLoginForm | CustomerLoginForm |
| 30 | BookingForm | BookingForm |
| 31 | CustomerRouteTrackingForm | CustomerRouteTrackingForm |
| 32 | DriverRouteTrackingForm | DriverRouteTrackingForm |
| 33 | FeedbackForm | FeedbackForm |
| 34 | ManageDriverWalletForm | ManageDriverWalletForm |
| 35 | ManageCustomerWalletForm | ManageCustomerWalletForm |
| 36 | AcceptRequestForm | AcceptRequestForm |
| 37 | CustomerPaymentForm | CustomerPaymentForm |
| 38 | DriverPaymentForm | DriverPaymentForm |
| 39 | ViewNotificationForm | ViewNotificationForm |
| 40 | ViewTripHistoryForm | ViewTripHistoryForm |
| 41 | ManageCustomerInfoForm | ManageCustomerInfoForm |
| 42 | ManageDriverInfoForm | ManageDriverInfoForm |
| 43 | CreateNotificationForm | CreateNotificationForm |

1. **Design-Element-to-Owning-Package Map**

| **STT** | **Lớp phân tích** | **Gói sở hữu** |
| --- | --- | --- |
| 1 | Customer | Model::Objects |
| 2 | TransactionBilling | Model::Objects |
| 3 | Staff | Model::Objects |
| 4 | Request | Model::Objects |
| 5 | Trip | Model::Objects |
| 6 | DriverWallet | Model::Objects |
| 7 | CustomerWallet | Model::Objects |
| 8 | Notification | Model::Objects |
| 9 | TransactionBilling | Model::Objects |
| 10 | TripBilling | Model::Objects |
| 11 | WalletSystem | WalletSystem |
| 12 | BillingSystem | BillingSystem |
| 13 | MapAPI | Model::API |
| 14 | LoginController | Controller |
| 15 | BookingController | Controller |
| 16 | RouteTrackingController | Controller |
| 17 | FeedbackController | Controller |
| 18 | ManageWalletController | Controller |
| 19 | AcceptRequestController | Controller |
| 20 | PaymentController | Controller |
| 21 | ViewNotificationController | Controller |
| 22 | ViewTripHistoryController | Controller |
| 23 | ManageDriverInfoController | Controller |
| 24 | ManageCustomerInfoController | Controller |
| 25 | CreateNotificationController | Controller |
| 26 | LoginForm | View::Staff |
| 27 | CustomerLoginForm | View::Customer |
| 28 | BookingForm | View::Customer |
| 29 | CustomerRouteTrackingForm | View::Customer |
| 30 | DriverRouteTrackingForm | View::Driver |
| 31 | FeedbackForm | View::Customer |
| 32 | ManageCustomerWalletForm | View::Customer |
| 33 | ManageDriverWalletForm | View::Driver |
| 34 | AcceptRequestForm | View::Driver |
| 35 | CustomerPaymentForm | View::Customer |
| 36 | DriverPaymentForm | View::Driver |
| 37 | ViewNotificationForm | View::Customer |
| 38 | ViewTripHistoryForm | View::Customer |
| 39 | ManageCustomerInfoForm | View::Staff |
| 40 | ManageDriverInfoForm | View::Staff |
| 41 | CreateNotificationForm | View::Staff |
| 42 | ICustomerDatabase | Model::DatabaseInterfaces |
| 43 | IDriverDatabase | Model::DatabaseInterfaces |
| 44 | IStaffDatabase | Model::DatabaseInterfaces |
| 45 | IRequestDatabase | Model::DatabaseInterfaces |
| 46 | ITripDatabase | Model::DatabaseInterfaces |
| 47 | IWalletDatabase | Model::DatabaseInterfaces |
| 48 | INotificationDatabase | Model::DatabaseInterfaces |
| 49 | IBillingSystem | Model::BillingInterfaces |
| 50 | ITransactionBilling | Model::BillingInterfaces |
| 51 | IWalletSystem | Model::WalletInterfaces |

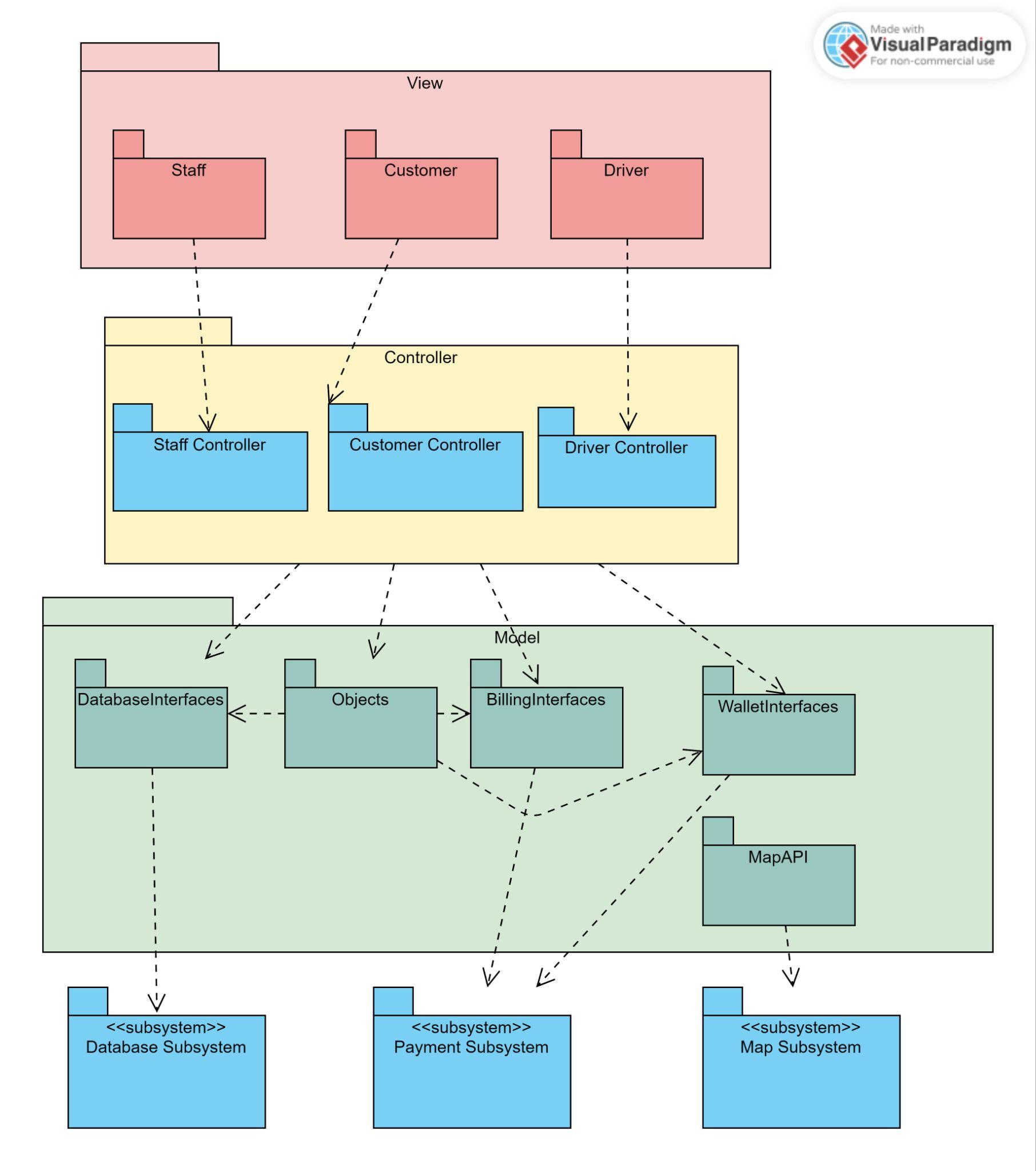
# 

1. **Architectural MVC and Their Dependencies**
   1. **MVC Architecture**



* 1. **Descriptions**
     1. **Model:** Model là các thành phần của ứng dụng tương ứng với tất cả logic liên quan đến miền dữ liệu (data domain), hoặc nói ngắn gọn đây là phần back-end chứa tất cả logic dữ liệu của ứng dụng. Dữ liệu ở đây có thể là dữ liệu đang được truyền giữa các thành phần View và Controller hoặc bất kỳ dữ liệu nào khác liên quan đến logic của doanh nghiệp.  
          
        Nếu trạng thái của dữ liệu này thay đổi thì Model thường sẽ thông báo cho View (để màn hình có thể thay đổi khi cần) và đôi khi là Controller (nếu cần logic khác để cập nhật View). Thông thường, các đối tượng Model có thể truy xuất từ cơ sở dữ liệu, thao tác và lưu trữ trạng thái Model trong cơ sở dữ liệu.
     2. **View:** View là các thành phần hiển thị giao diện người dùng (UI) của ứng dụng. Thông thường, giao diện người dùng này được tạo từ dữ liệu Model.
     3. **Controller:** Controller là các thành phần xử lý tương tác của người dùng để làm việc với Model (cập nhật logic dữ liệu) hoặc/ và với View (cập nhật hiển thị giao diện người dùng).  
          
        Trong ứng dụng MVC, Controller xử lý các giá trị chuỗi truy vấn và chuyển các giá trị này cho Model, từ đó Model sẽ truy vấn cơ sở dữ liệu bằng cách sử dụng các giá trị đó. View hiển thị thông tin do Controller xử lý và phản hồi đầu vào từ tương tác của người dùng.

1. **Packages and Their Dependencies**
   1. **Package Dependencies Diagram**

**Customer**

* 1. **Package Description**

Biểu đồ trên mô tả mối quan hệ phụ thuộc của các gói ở cùng một lớp và giữa các lớp. Một mối quan hệ phụ thuộc tồn tại giữa hai gói nếu sự thay đổi của một gói có thể dẫn tới sự thay đổi của gói còn lại.

* View: Gói chịu trách nhiệm nhận yêu cầu đầu vào của người dùng và hiển thị phản hồi của hệ thống đến với người dùng.
* Controller: Xử lý các yêu cầu và điều khiển luồng của ứng dụng. Nhiệm vụ chính của Controller là nhận các yêu cầu từ người dùng thông qua giao diện người dùng (View), xử lý logic ứng dụng và tương tác với Model để truy xuất dữ liệu cần thiết, sau đó trả về kết quả tương ứng cho View.
* Model: Model quản lý dữ liệu và logic ứng dụng. Model đại diện cho các đối tượng và dữ liệu trong ứng dụng và cung cấp các phương thức để truy cập và cập nhật dữ liệu.
* View::Staff: Nhận yêu cầu đầu vào của Staff và hiển thị phản hồi của hệ thống đến với Staff.
* View::Customer: Nhận yêu cầu đầu vào của Customer và hiển thị phản hồi của hệ thống đến với Customer.
* View::Driver: Nhận yêu cầu đầu vào của Driver và hiển thị phản hồi của hệ thống đến với Driver.
* Model::DatabaseInterfaces: Interface chịu trách nhiệm thao tác tới Cơ sở dữ liệu cho những chức năng chung nhất của hệ thống: Ví dụ như đăng nhập, đăng kí, …
* Model::BillingInterfaces: Interface chịu trách nhiệm thao tác tới Cơ sở dữ liệu (Thêm, sửa, xóa, đọc) chỉ đối với đối tượng Billing (thao tác trên bảng Billing)
* Model::WalletInterfaces: Interface chịu trách nhiệm thao tác tới Cơ sở dữ liệu (Thêm, sửa, xóa, đọc) chỉ đối với đối tượng Wallet (thao tác trên bảng Wallet)
* Database Subsystem: Hệ thống con đại diện cho các thao tác tương tác tới Cơ sở dữ liệu của hệ thống, các thao tác tới Cơ sở dữ liệu được thực hiện thông qua các Interface của hệ thống con này.
* Map Subsystem: Đóng gói cho việc giao tiếp với hệ thống cung cấp bản đồ điện tử.
* Payment Subsystem: Đóng gói việc giao tiếp với các hệ thống thanh toán.