KTPM1 – Group 06

E-commerce Web Application
Software Architecture Document

Version <3.0>

Revision History

Date	Version	Description	Author
01/12/2020	1.0		Phạm Vũ Duy, Hồ Nguyễn Huy Hoàng
09/12/2020	2.0	Add use case diagram and specification V2.0	Phạm Vũ Duy, Hồ Nguyễn Huy Hoàng
16/12/2020	3.0	Add component class diagram, deployment diagram and implementation View	Phạm Vũ Duy, Hồ Nguyễn Huy Hoàng

Table of Contents

Software Architecture Document

1. Introduction

"Dang's Company" is an E-commerce Web Application, its aim is to bring the buyers and the sellers together.

This document elaborates the software architecture document for the "Dang's Company E-commerce Web Application". The system architecture is abstracted into many view and components which will be explain in this document.

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture the significant architectural decisions which have been made on the system.

1.2 Scope

The software architecture document applies to each static and dynamic aspect of the system.

Under the static behavior of the system, the document discusses the class diagram and other static architecture designs. Dynamic aspects of the system are elaborated using case realizations.

1.3 Definition, Acronyms and Abbreviations

MVC – Model View Control architecture

DWA - Dang's Company E-commerce Web Application

DB – DataBase

1.4 Reference

https://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm#Definitions,%20Acronyms%20and%20Abbreviations

https://www.slideshare.net/PasinduTennage/sample-software-architecture-document

https://sceweb.uhcl.edu/helm/RationalUnifiedProcess/process/artifact/ar_sadoc.htm

1.5 Overview

This document will present a detailed analysis of the architecture of Dang's Company E-commerce Web Application. The further section will cover the architectural goals including the architectural constraints.

2. Architectural Goals and Constraints

2.1 Server side

The DWA will be hosted at "Heroku" JSP server. Mongo DB will be used as central database server. All communication between server and client will using HTTP/HTTPS (free SSL come along with Heroku) – a standard communication protocol.

2.2 Client side

User will be able to access DWA only online. Users/Clients are expected to use a modern web browser which

can support Bootstrap 4.0 to get full experience:

Recommend mobile devices browser

	Google Chrome	**Firefox**	**Safari 9.0 +**	**Android Browser 4.4+**
Android	✓	✓	N/A	✓
iOS	✓	✓	✓	N/A

Desktop

	Chrome 45.0+	**Firefox 38.0+**	**IE 10.0+**	**Microsoft Edge 12.0+**	**Opera 30.0+**	**Safari 9.0+**
Mac	✓	✓	N/A	N/A	✓	✓
Windows	✓	✓	✓	✓	✓	N/A

2.3 Security and privacy

The central security will be handled by the member of the development team. They will be given the full access not only in the application level but also in database level. Creating account for the staff and the owner of the shop are done by administrator. When creating an account, user can choose their password and this password can be changed anytime by them. All the password is encrypted both on database or on communication process between client and server in order to ensure high level of security. The user information will only be seen by the shop owner and the administrator.

2.4 Reliability and Availability

The system will be subjected to several testing step (Unit testing, Integration testing, System testing including Security and Performance testing) before being released to ensure that the system is reliable and worked as intended. The DB Central which is placed on Amazon, ensure both the security with TLS/SSL encryption and performance such as low latency and response time.

2.5 Performance

The server responds to any request from client within the web script timeouts (30 seconds), also the system performance can depend on available hardware, networks and internet connection capabilities. Therefore, the actual performance can be determined only after the system is deployed and tested. Our aim is to make the loading time on client side become ideal which is lower than 2 seconds.

2.6 Portability and Reuse

The DWA is design to be a complete cosmetic website. But can be extend to sale many kinds of product. In order to maintain reusability, the web using Handlebar Template Engines which can be reuse. Best practice of RUP during development combine with Mongo DB make the structured is well layered.

2.7 Development tools

The project is the combination of many tool:

Programing tool: Visual Code

Database: PostgreSQL

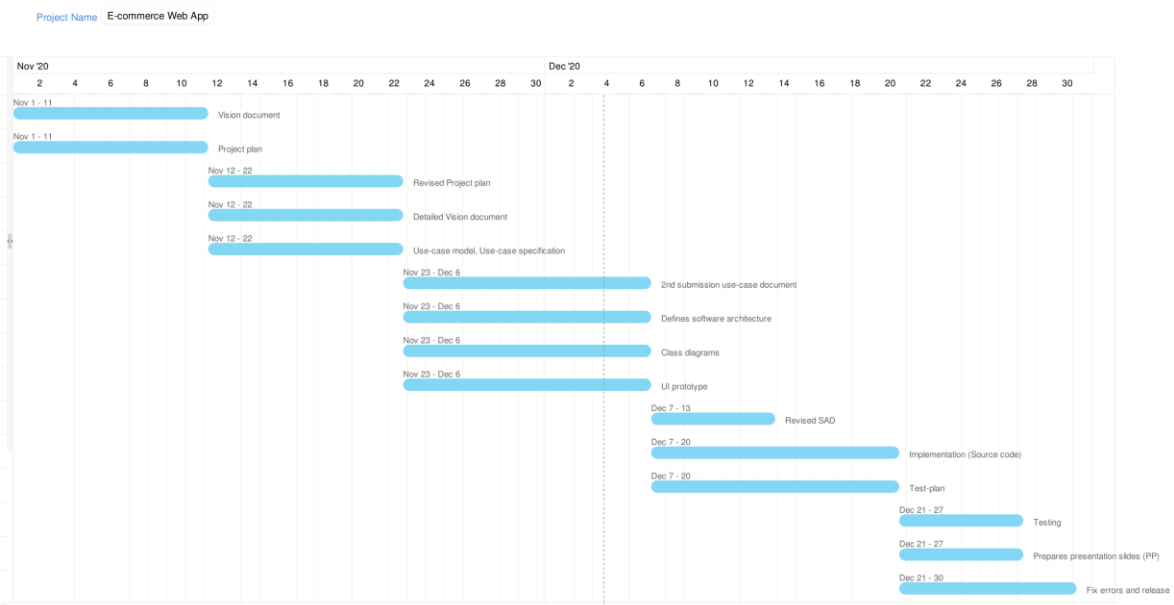
UI Prototype: figma

Meeting platform: Slack, Discord

Schedule: Trello

2.8 Schedule

The development process is follows by the combination of Agile and RUP workflow. There are six sprints, each has their own workload and document:



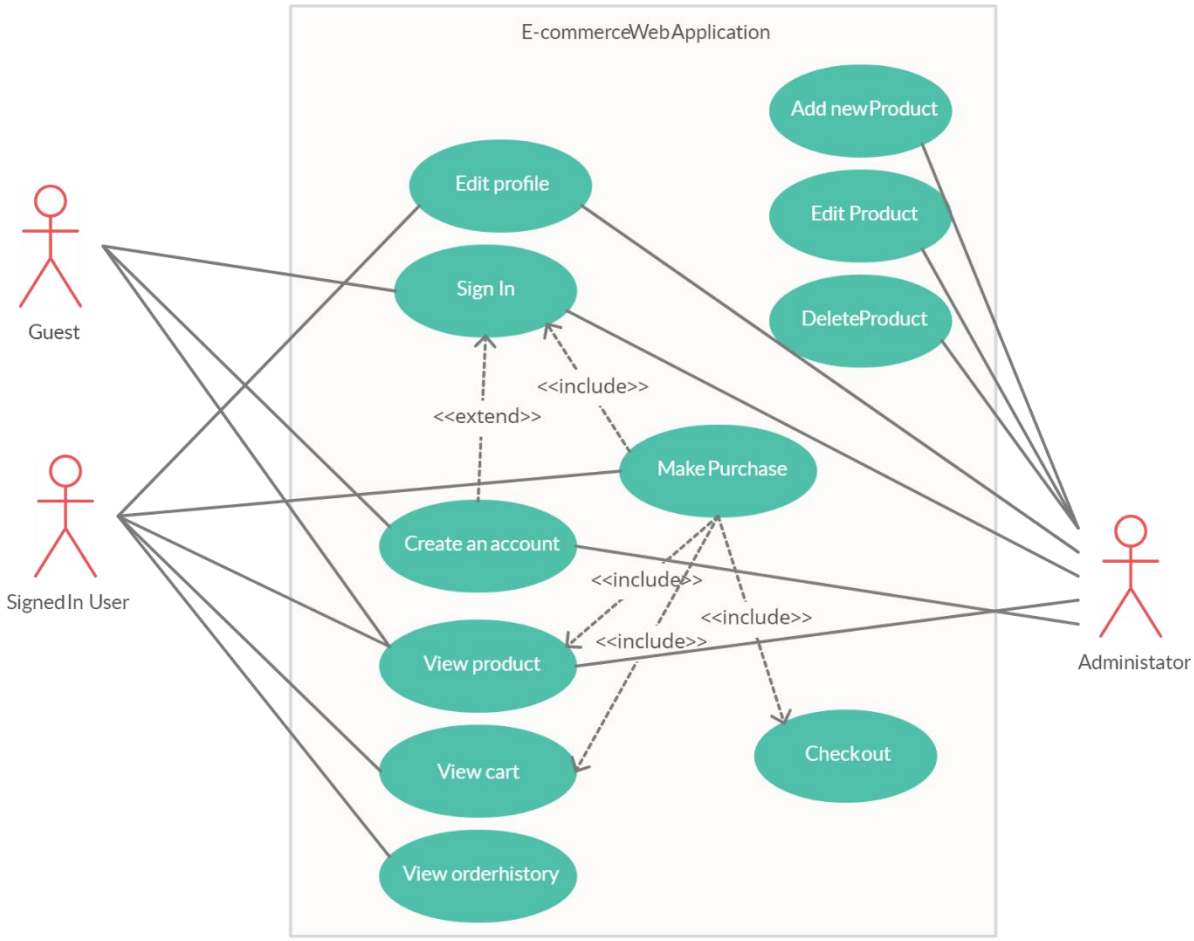
*Due to the limitation of Zoho Free-trial version that only allow maximum 3 member in each project. So, we are not able to show the specific assignment for each member in the Gantt chart above. This is the schedule with the specific assignment for each team member:

Phases	Interation No.	Tasks and Artifacts	Assignee	Start Date	End Date
Inception	1	- Vision document	Nguyễn Phúc Thịnh	1/11/2020	11/11/2020
		- Project plan	Huỳnh Nhật Nam		
Elaboration	1	- Revised project plan	Huỳnh Nhật Nam	12/11/2020	22/11/2020
		- Detailed vision document	Nguyễn Phúc Thịnh		
		- Use-case model, use-case specification	Huỳnh Nhật Nam, Nguyễn Phúc Thịnh		
	2	- 2nd submission use-case document	Huỳnh Nhật Nam, Nguyễn Phúc Thịnh	23/11/2020	6/12/2020
		- Defines software architecture	Phạm Vũ Duy, Hồ Nguyễn Huy Hoàng		
		- Class diagrams	Phạm Vũ Duy, Hồ Nguyễn Huy Hoàng		
Construction	1	- Revised SAD	Phạm Vũ Duy, Hồ Nguyễn Huy Hoàng	7/12/2020	13/12/2020
		- UI prototype	Phạm Vũ Duy, Hồ		

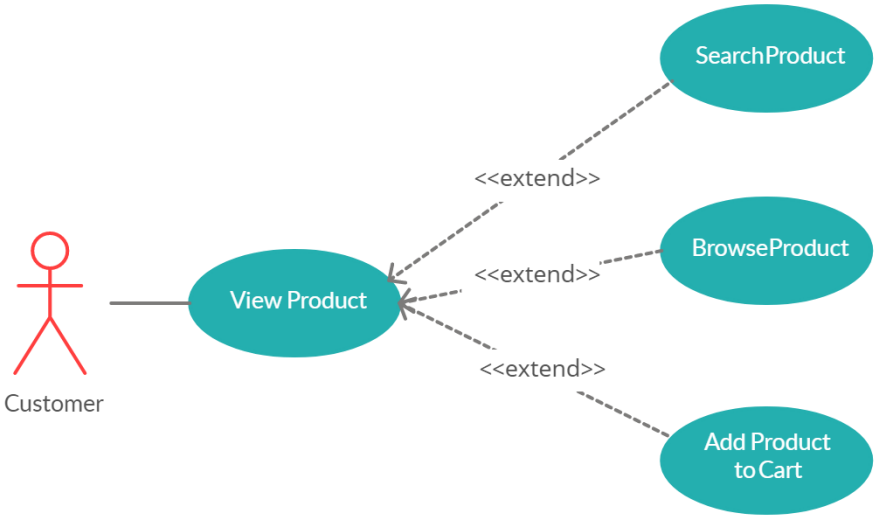
			Nguyễn Huy Hoàng		
		- Implementation (Source code)	All team members	7/12/2020	20/12/2020
		- Test-plan	Mai Đăng Khánh		
		- Release (See 4.2.3 for better details)	Hồ Nguyễn Huy Hoàng	14/12/2020	20/12/2020
	2	- Testing	Mai Đăng Khánh	21/12/2020	27/12/2020

3. Use-Case Model

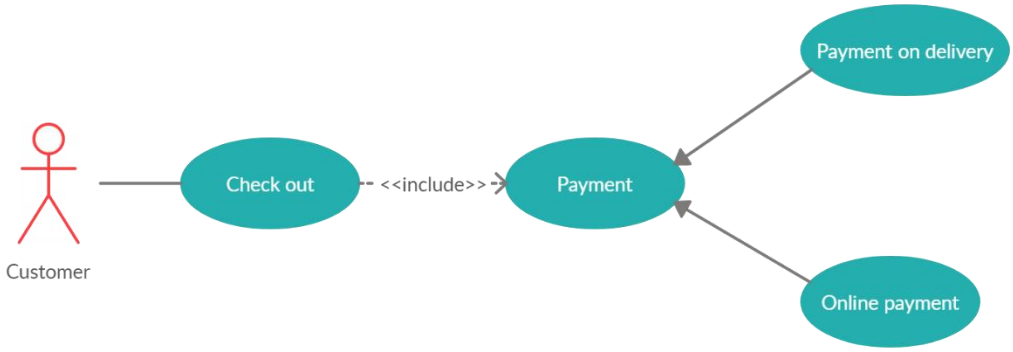
3.1 Use-case diagram



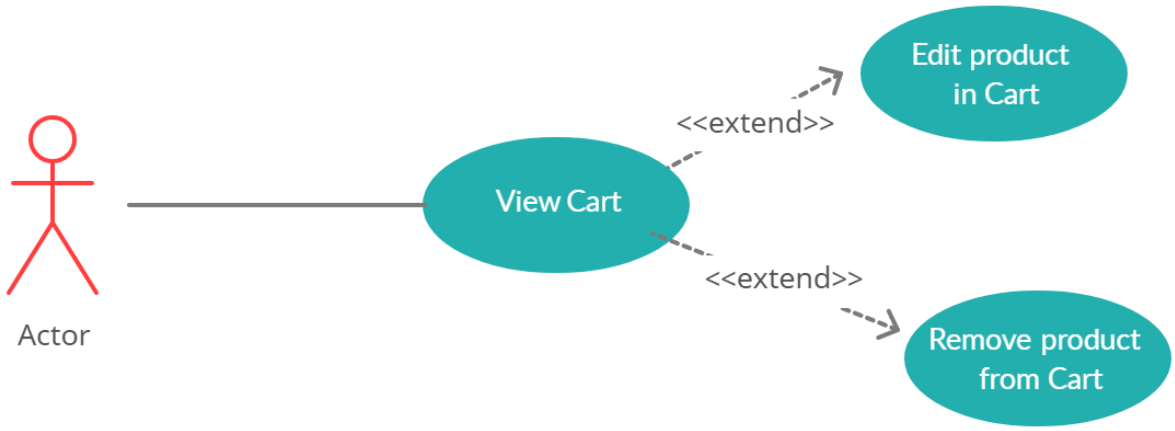
- View product model:



- Check Out model:



View Cart model:



4. Logical View

4.1 Overview

4.1.1 The subsystem

The DWA can be divided into 3 main sub-system.

1. Web store
2. Order management subsystem
3. Statical data management subsystem

4.1.1.1 Web store

This subsystem provides all the functionalities that is related to user. The main use cases of this subsystem include

1. User login / Shop owner login
2. Create new user
3. Change password
4. Edit profile
5. Searching for product
6. Add product to cart
7. Make comment and rating

Depending on the level of account (access level), some action may be not be allowed.

4.1.1.2 Order management subsystem

The Order Management System (OMS) is play an important role as a median subsystem between the Web Store and the Data manager subsystem. The OMS main functionalities relate to all the customer ordering task.

- Order management
- Checkout handling

4.1.1.3 Statical data manager subsystem

This subsystem is the main system in charge of managing the central database. The subsystem involved in data access and processing operation which require special algorithms and processing capabilities. Only a few required data are fetched from the database

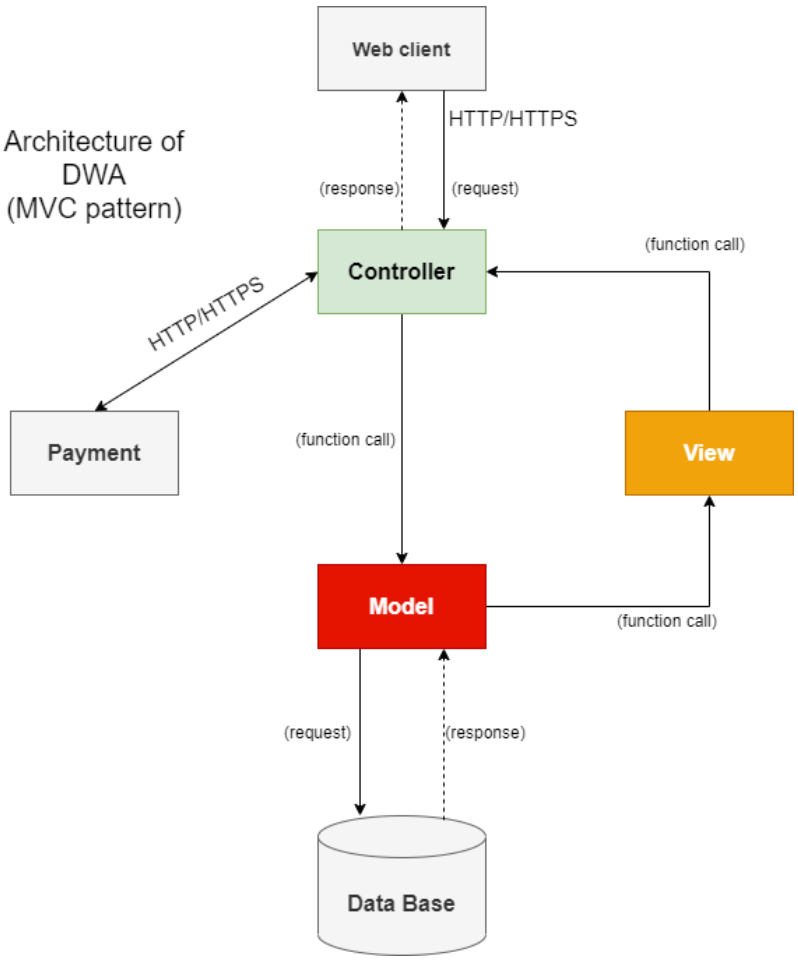
1. Numbers of product in stock
2. User information
3. Report about shop's monthly income
4. Change product status (add, remove, modify)
5. Comment and rating management

4.2 MVC model

The DWA (Dynamic Web Application) is divided into 3 main components: *Controller*, *Model* and *View*. Those components connected with each other in a strict rule, each of them has a specification job. The reason why this pattern has been chosen is that relate to each specification job of each main components, implementers can work parallel on several component to reduce the developing time. Following that, implementers easy to update and debug each component separately. However, there are also contrast, MVC pattern is hard to deeply understand and it must have strict rules on method to avoid errors and mixed structured.

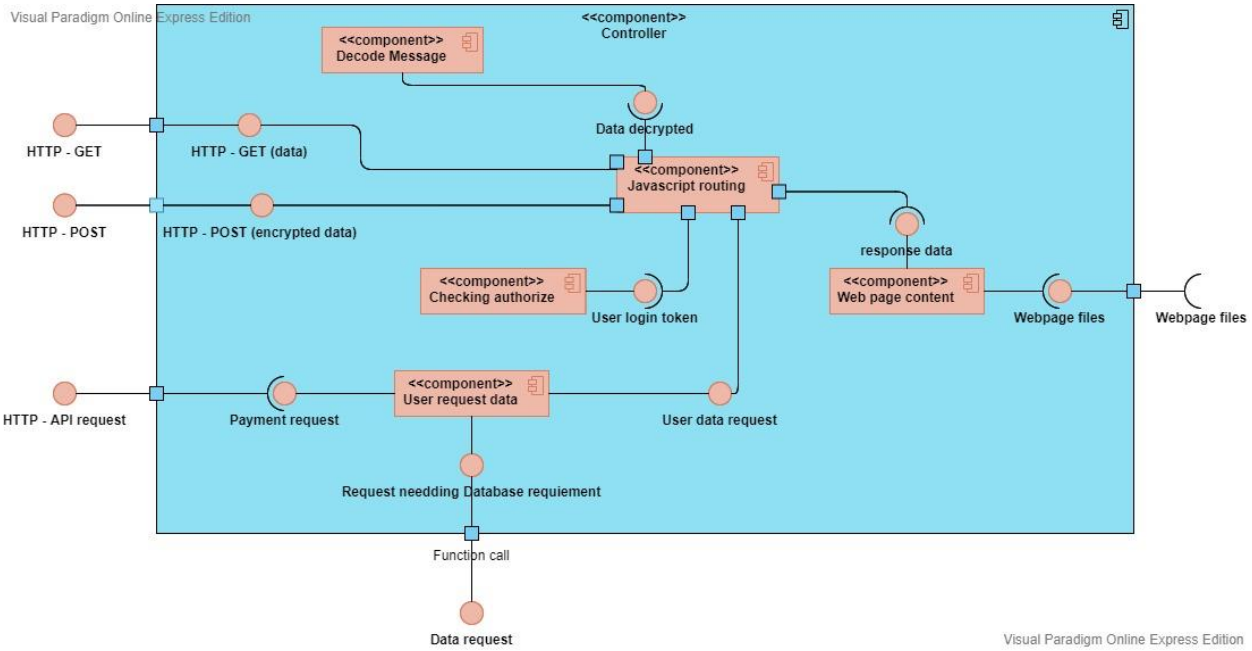
- The Controller handle the request of the client and make function calls to model to get the corresponding data.
- The Model handle the database, that means it make a request to the database such as get data, insert, update, delete. And its also send the data it has got to the view.
- The View in this case is import data to Html, Javascript and stick css style then send back to controller in order to respond the client request.

The pattern is present bellow to explain the communication method between those main components themself and other components



The MVC pattern for DWA architecture

4.2.1 Controller (additional)



In controller, client request and response by 2 main HTTP method, they are controlling by Javascript framework in order to receive and response data to client effectively:

- ***HTTP - GET:*** this is the method used to receive or send un-sensitive data from and to client.
- ***HTTP – POST:*** this is the method used to receive and send sensitive data such as User information (username, password, email…), or send webpage content to client.

The main sub-component inside Controller is Javascript routing, its job is to receive the HTTP request, and also response the Webpage content to client through HTTP/HTTPs protocol.

When POST data received, the routing catching that request and decode the message in its, then it pass the data to checking authorize:

- First it can make a request to User request data to get data for checking login success or register criteria.
- Secondly, if client request an authorize action the sub-component will check by using JSON Web Tokens to consider that user is login or not. Then response corresponding to that user action (restrict an un-authorized client from doing authorized action or passing successfully authorized client's request to User request data to go to Model).

The GET request does not need to decrypt, this request is pass through to User request data component to go to Model component.

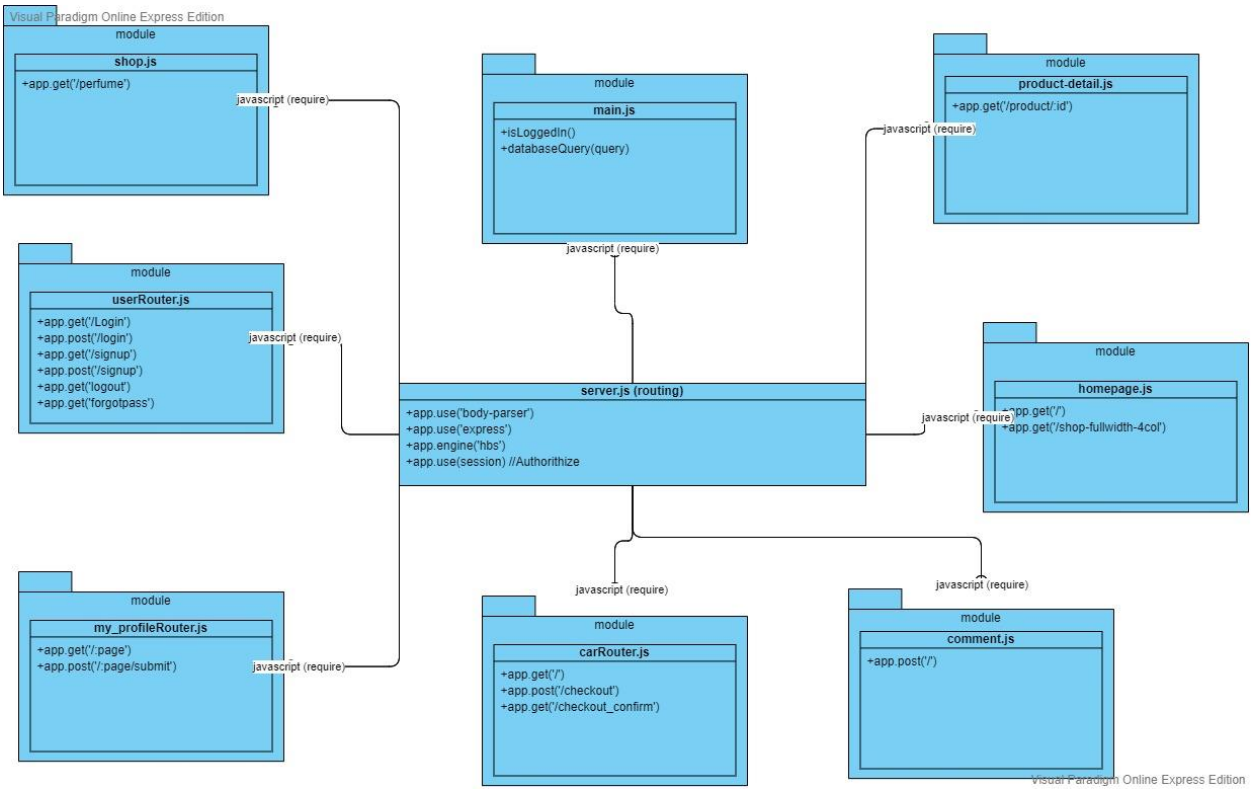
The User request sub-component is used as a bridge to pass request corresponding to third-party or to Model.

The webpage receive the web content from view to response the client.

The main programing language in this component is: Javascript.

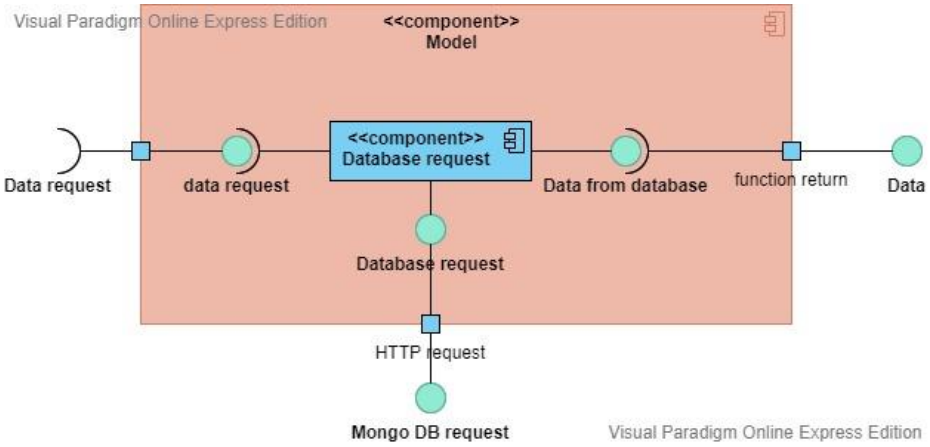
Framework: ExpressJS.

Here is the class diagram of it



It presents the file structure for javascript code, the function name, they split to javscript files and export the functions. Then the server.js – routing file for the DWA will 'require' or can be call as import to its.

4.2.2 Model (additional)

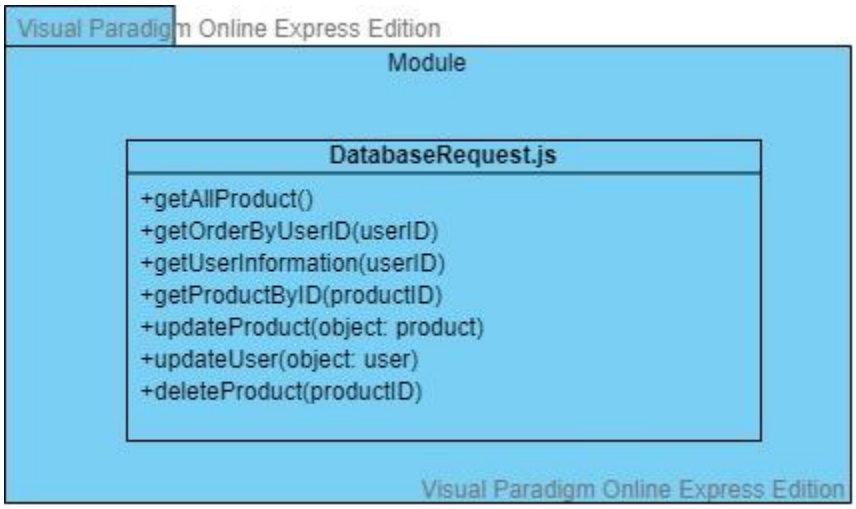


In Model, the database request is control by Database request component, base on request it make a HTTP call to PostgreSQL's database for: get data, insert data, update data, delete data. Then the response from the PostgreSQL's

will pass to the View component as javascript object.

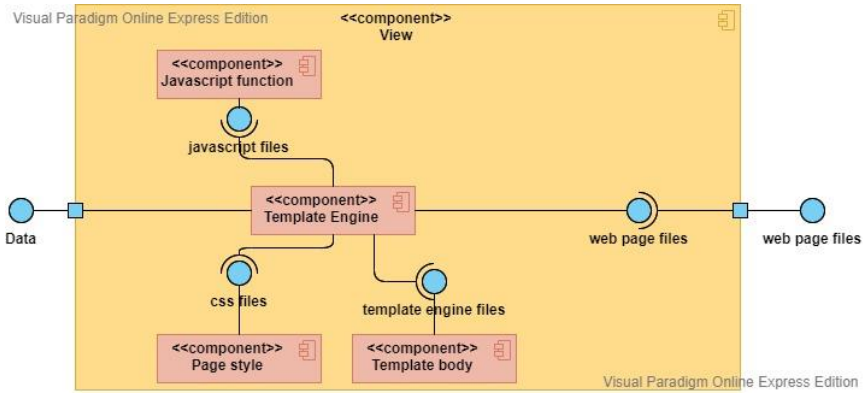
The main programing language in this component is: Javascript.

Here is the class diagram for it



The class diagram presents the export module for functions doing the job with database.

4.2.3 View (additional)



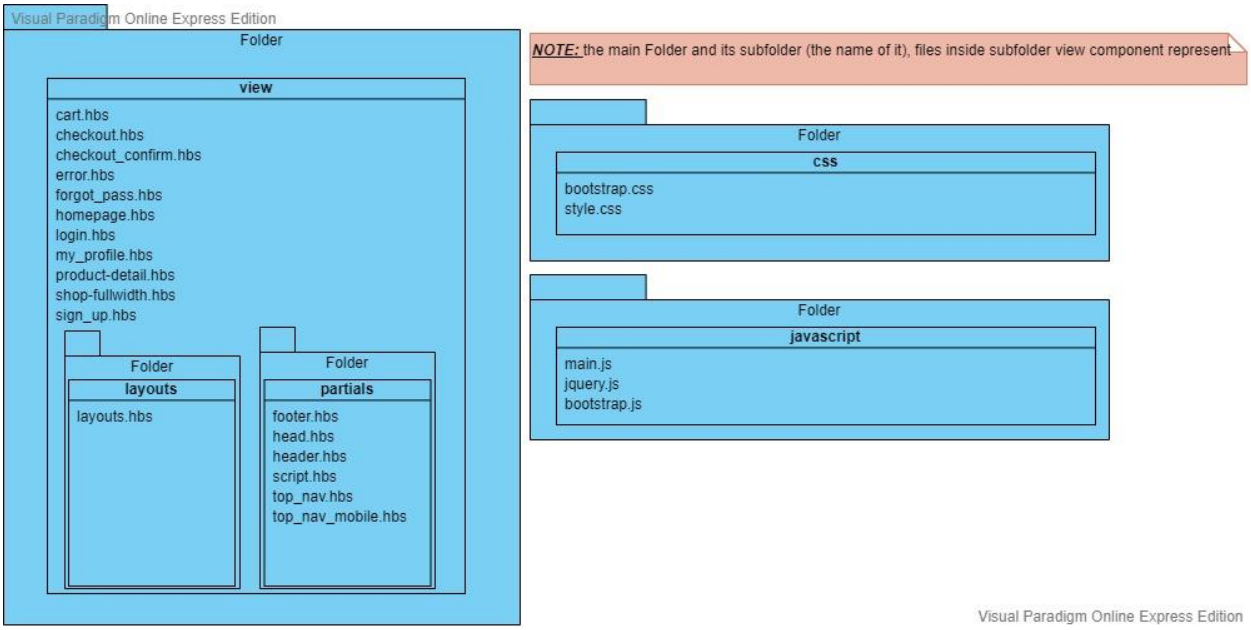
In View, using Template Engine which contain main layout of the web page, and then combine: the css style for that page, the javascript function for each button click or web page effect, and the template body of particular page (for example, product information page, home page, login page…). The Template Engine then convert all combination to HTML file, CSS files and Javascript files.

Page style may use ***Scss*** and then convert to ***css*** optimize.

Tempalte Engine: Handlebars

Javascript: ES6 structure.

Here is its class diagram:



The class diagram show the folder structure for View component, these file will be define in server routing javascript files to return to user.

4.2.4 Payment

In Payment, the DWA might use API provide by:

- Momo
- Zalo pay

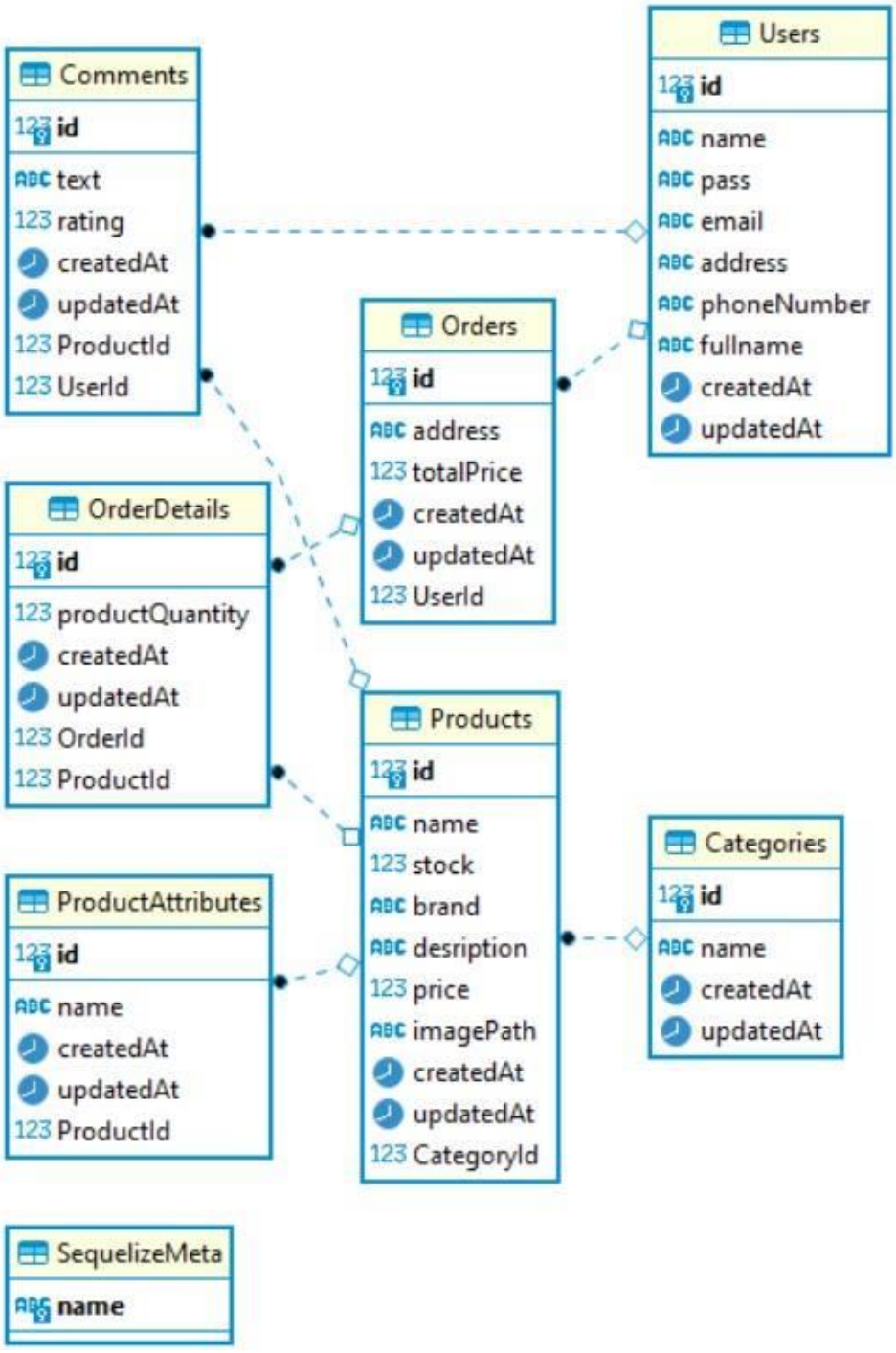
Due to those third-party payment organization can support for Vietnamese people, and themselves also have many vouchers to encourage customer/ user to buy more products.

Especially, Zalo pay support user make a purchase through their ATM card.

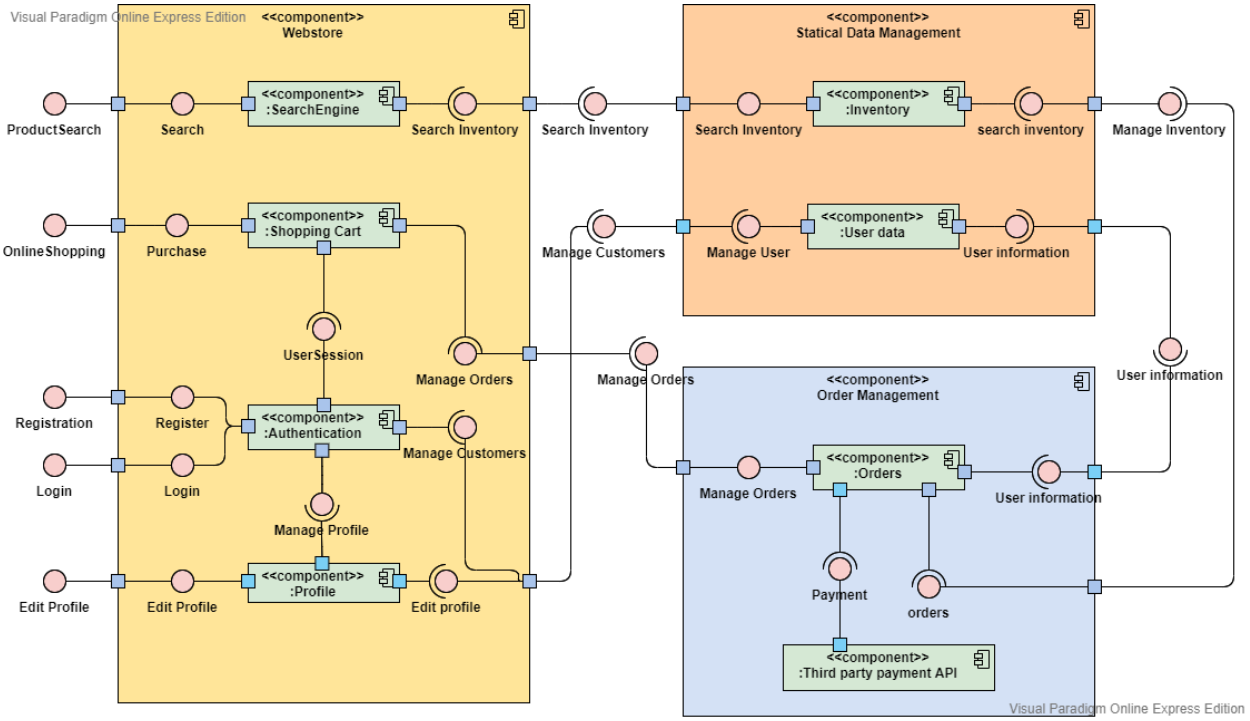
4.2.5 Database (changed)

Postgres provide by Heroku helps to maximize the data instead of spending time on database setup and maintenance. Test new schema migrations, manage database access levels and protect queries, scale horizontally, and allow the DWA to quickly access data. Postgres is a relationship sql. here its design.

The Database server is the server of Heroku, the same server with DWA.


Comments
id
text
rating
createdAt
updatedAt
ProductId
UserId
Users
id
name
pass
email
address
phoneNumber
fullname
createdAt
updatedAt
Orders
id
address
totalPrice
createdAt
updatedAt
UserId
OrderDetails
id
productQuantity
createdAt
updatedAt
OrderId
ProductId
Products
id
name
stock
brand
desription
price
imagePath
createdAt
updatedAt
CategoryId
Categories
id
name
createdAt
updatedAt
ProductAttributes
id
name
createdAt
updatedAt
ProductId
SequelizeMeta
name

4.3 UML (Overview)



The UML – Component diagram show the overview of DWA to make the MVC model architecture above more explicit, this diagram is an overview stand at Customer view such as customer request to search a product or login, register, edit profile… Split DWA into 3 subsystem in order to easy understand at a customer point in the traditional way the real shop work. Each UML Overview component and interface are reference to MVC – component (Controller, Model, View, Database, Payment).

As MVC-View component is do the front-end job so in this UML Overview there is no MVC-View as an UML component, in this diagram we can say that the response from Statical Data Management and Order Management is the front-end files provided by MVC – View.

4.3.1 Webstore UML explain:

In Webstore subsystem, it handles the requests from client such as *ProductSearch*, *OnlineShoping*, *Registration*, *Login* and *Edit Profile*. So, Webstore UML component is MVC – Controller.

The Authentication is the *checking authorize* of MVC – controller.

4.3.2 Statical Data Management explain:

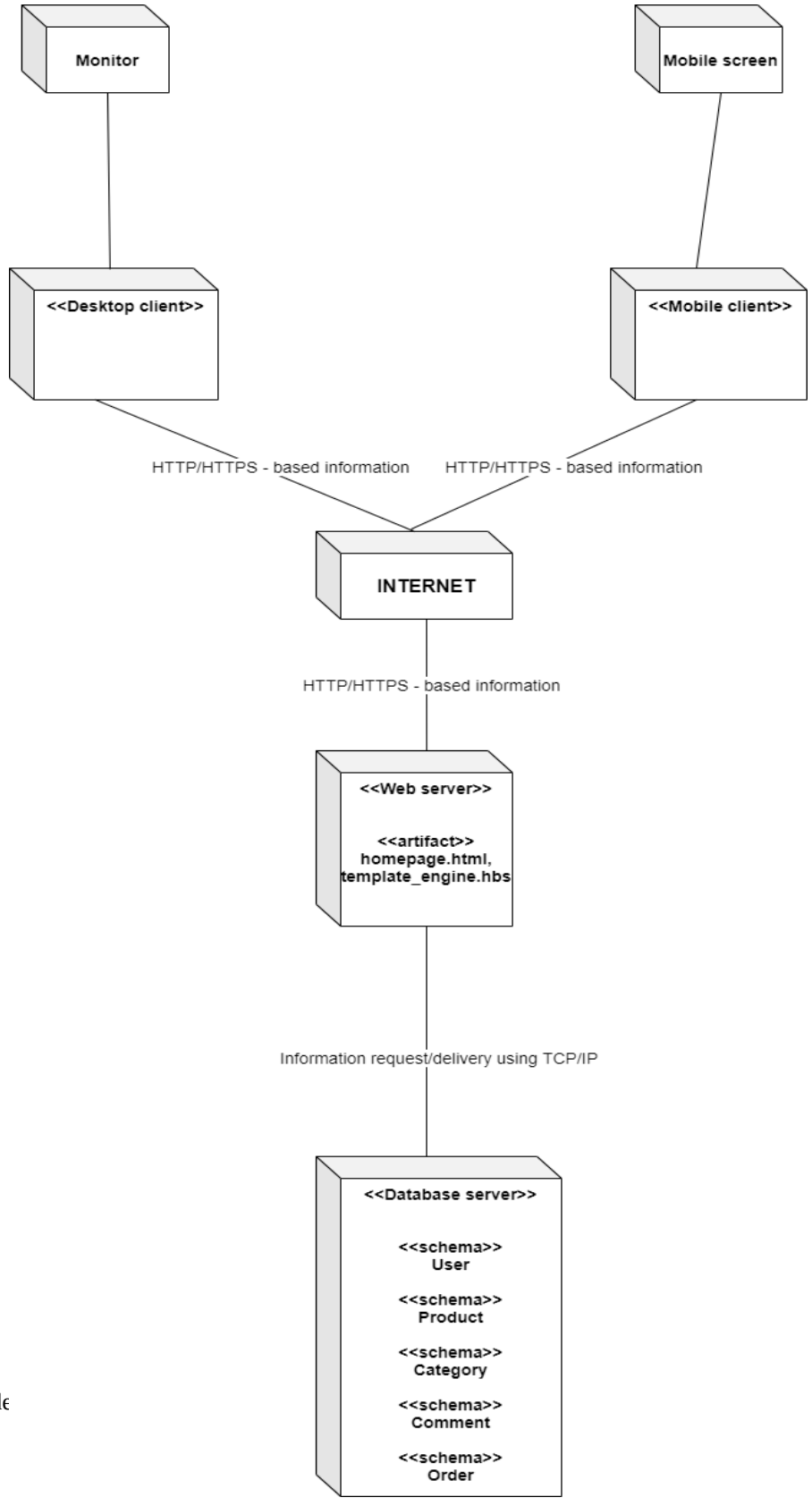
In Statical Data Management, it does the request job to Database. So, Statical Data Management is the MVC – Model and MVC – Database.

4.3.3 Order Management explain:

In Order Management, it is the MVC – Payment (the Third party payment API - UML component), and a part of MVC – controller (Orders - UML component).

5 Deployment

Here is the deployment diagram:


Monitor
Mobile screen
<<Desktop client>>
<<Mobile client>>
HTTP/HTTPS - based information
HTTP/HTTPS - based information
INTERNET
HTTP/HTTPS - based information
<<Web server>>
<<artifact>>
homepage.html,
template_engine.hbs
Information request/delivery using TCP/IP
<<Database server>>
<<schema>>
User
<<schema>>
Product
<<schema>>
Category
<<schema>>
Comment
<<schema>>
Order

The diagram includes 4 main nodes: Webserver, desktop client, mobile client and DB center. They are connected to each other through Internet using HTTP/HTTPS protocol.

+ Desktop client, mobile client: They sent request to the Web server. As the response, Web server will send them .html and template engine page. Client now will need Browser to render these files and showing them to the user using computer monitor or mobile screen.

+ Web server: The main task of server is to response to the client request. In order to satisfy the user, it must in charge of the others task is to communicate with the PostgreSQL server. When the client sends a request, it will response with the suitable file and the necessary data which stored in the DB center. To do that, Web server must communicate or get the appropriate data from PostgreSQL with the help of Sequelize and put it in the handlebars template engine before send to client to display.

+ DB center: In charge of holding and providing the Web server with suitable data. These data is classified into tables: OrderDetail, Category, User, Product, Comment, Order. The record will be updated and justified with command from Web server

6. Implementation View:

-Tree structure in the Web server: