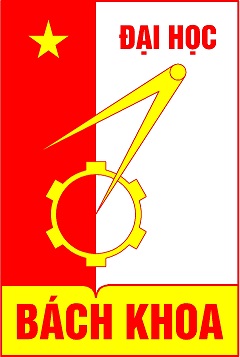
uwj **Đại Học Bách Khoa Hà Nội**

**Viện Công Nghệ Thông Tin và Truyền Thông**



**Báo cáo cuối kỳ**

**Lập Trình Cấu Trúc**

**Giáo viên hướng dẫn: TS. Vũ Thị Hương Giang**

**Nhóm 5 AS K57**

Thành viên:

Nguyễn Hữu Sơn 20122349

Trần Bảo Huy

Đào Mạnh Nhất 20124239

MỤC LỤC

MỤC LỤC 2

LỜI NÓI ĐẦU 3

PHÂN CÔNG THÀNH VIÊN TRONG NHÓM 4

CHƯƠNG 1.MÔ TẢ YÊU CẦU 5

1.1.Mô tả yêu cầu bài toán 5

1.2.Mô tả các user story 5

CHƯƠNG 2.THIẾT KẾ CHI TIẾT 7

2.1.Thiết kế Cơ sở dữ liệu 7

CHƯƠNG 3.DEMO CẤU TRÚC 8

3.1.Cấu trúc chương trình 8

CHƯƠNG 4.KIỂM THỬ 10

4.1.Nguyên tắc solid 10

4.2.Kiểm chứng cách trình bày báo cáo 10

4.3.Kiểm chứng thiết kế và kỹ năng lập trình 10

4.4.Kiểm thử tích hợp 10

CHƯƠNG 5.HƯỚNG DẪN CÀI ĐẶT VÀ CHƯƠNG TRÌNH MINH HỌA 12

TÀI LIỆU THAM KHẢO 13

LỜI NÓI ĐẦU

Việc ứng dụng máy tính để quản lý thông tin đã phát triển mạnh ở các nước tiên tiến từ những thập niên 70. Còn ở Nước ta hiện nay, vấn đề áp dụng Tin học để xử lý thông tin trong công tác quản lý đã trở thành nhu cầu bức thiết, nhất là trong thời đại "mở cửa" với các nước khác trên thế giới.

Hiện nay,mỗi cá nhân , tập thể đều có rất nhiều đồ cũ , đồ không dùng đến những cũng còn rất tốt . Vì thế việc tạo ra một diễn đàn để trao đổi đồ cũ là một vấn đề cần thiết .

Để hoàn thành được bài tập lớn này, nhóm chúng em xin được gửi lời cảm ơn chân thành đến cô giáo hướng dẫn đề tài Cô Vũ Thị Hương Giang, Giảng viên Khoa Công nghệ Thông tin Trường Đại học Bách Khoa Hà Nội - đã hết lòng giúp đỡ, hướng dẫn, chỉ dạy tận tình để nhóm em hoàn thành được đề tài này.

***Hà Nội, tháng 11 năm 2015***

**Nhóm 5**

**PHÂN CÔNG THÀNH VIÊN TRONG NHÓM**

|  |  |  |  |
| --- | --- | --- | --- |
| Họ tên | Công việc | Đóng góp | Hoàn thành |
| Nguyễn Hữu Sơn | * Thực hiện trao đổi * Tìm kiếm đồ * Viết báo cáo | 35% | 100% |
| Trần Bảo Huy | * Hiển thị, đăng và chỉnh sửa đồ. * Chỉnh sửa thông tin cá nhân | 35% | 100% |
| Đào Mạnh Nhất | * Quản lý của admin * Viết báo cáo | 30% | 100% |

**CHƯƠNG 1. MÔ TẢ YÊU CẦU**

## **Mô tả yêu cầu bài toán**

Nhu cầu về trao đổi những đồ đã qua sử dụng là không nhỏ.

Cần có một hệ thống phục vụ nhiều người dùng giúp người dùng có thể trao đổi với những người có cùng nhu cầu.

* 1. **Mô tả các user story**

**User story: Hiển thị đồ và đăng đồ**

* Là một người dùng hệ thống, tôi muốn quan sát được tất cả các đồ của người dùng khác đã đăng lên.
* Khi chọn vào một đồ bất kì thì tôi có thể quan sát được cụ thể các thông tin hơn, nếu là sản phẩm của tôi thì có thể chỉnh sửa thông tin.
* Khi có đồ cần trao đổi thì tôi có thể đăng được đồ của mình lên hệ thống

**User story: Thực hiện trao đổi**

* Là một người sử dụng hệ thống, tôi muốn thực hiện trao đổi vật phẩm với người sở hữu vật phẩm mà tôi quan tâm.
* Sau khi nhấn vào đồ của người dùng khác, tôi có thể chọn trao đổi và chuyển đến trang tất cả các đồ tôi đang sở hữu để lựa chọn đồ cần trao đổi
* Người dùng còn lại khi lựa chọn vào phần yêu cầu gửi sẽ thấy được thông báo trao đổi và từ đó là xác nhận trao đổi hoặc không .
* Tôi có khả năng xem trạng thái yêu cầu mà tôi đã gửi đã được người bên kia chấp nhận hay chưa.

**User story: Quản lý vật phẩm**

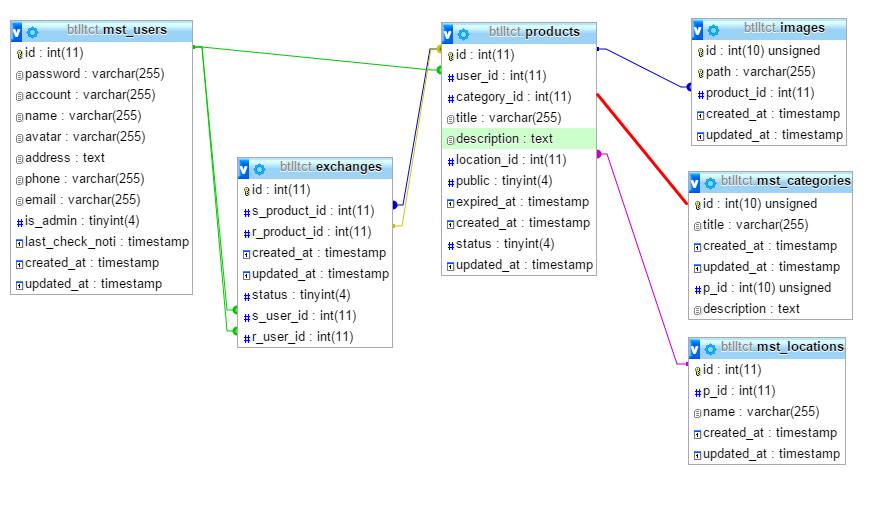
* Là một người sử dụng hệ thống, tôi muốn quản lý các đồ đăng lên của mình
* Sau khi chọn vào đồ của tôi đã đăng tôi có thể chỉnh sửa đồ đó, và tôi cũng có thể xóa bỏ đồ đó khỏi hệ thống khi mà tôi không còn sử dụng vật phẩm đó nữa.Trước khi xóa vật phẩm khỏi hệ thống, hệ thống cần phải xác nhận lại xem tôi có thực sự muốn xóa hay không.

**User story: Quản lý của admin**

* Là một admin của hệ thống, tôi muốn có thể quản lý các user , các category, các đồ và các địa điểm .
* Sau khi lựa chọn vào mỗi phần thì tôi có thế sửa hoặc xóa . Với các category và địa điểm thì tôi có thể tạo mới .

**CHƯƠNG 2. THIẾT KẾ CHI TIẾT**

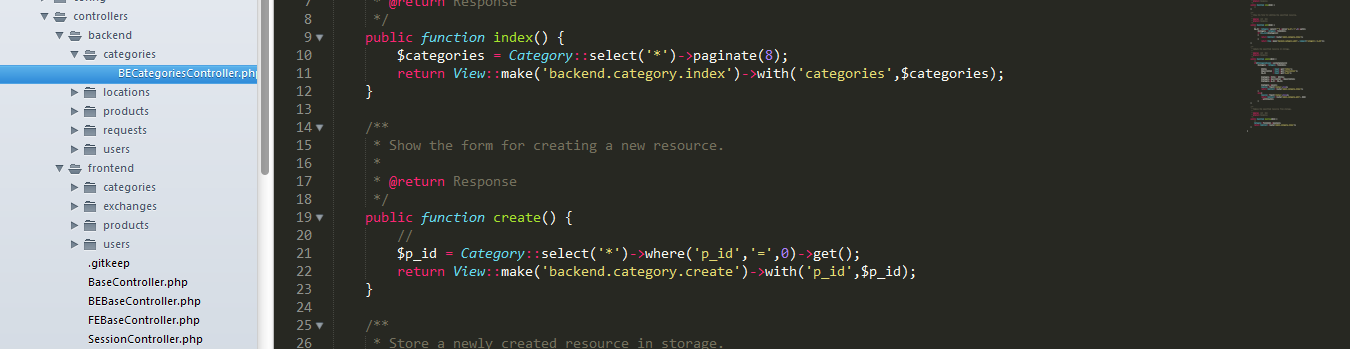
## **Thiết kế cơ sở dữ liệu.**



**CHƯƠNG 3. DEMO CẤU TRÚC**

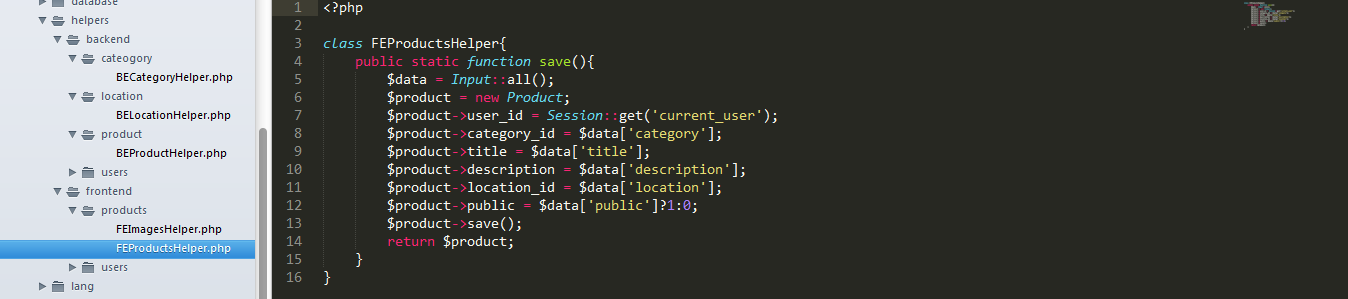
## **Cấu trúc chương trình – chương trình có cấu trúc tốt.**

## Đã có sự độc lập rất lớn :

* Controller: Được chia làm 2 phần lớn : Backend và Frontend . Trong mỗi phần có nhiều package chứa các controller hoàn toàn độc lập với nhau 🡪 rất tiện cho việc mở rộng nếu hệ thông tiếp tục phát triển.
* View: Được chia làm 2 phần lớn : Backend và Frontend tương đương với các phần như ở Controller.

*Hình ảnh minh họa tính độc lập của chương trình*

* Những file thêm vào có cấu trúc tốt:
* Controller: Tất cả các Controller ở phần backend đều extend một Controller là *BEBaseController* tương tự thì tất cả các Controller ở phần frontend đều extend một Controller là *FEBaseController* . Và 2 Controller base này đều được tự xây dựng kế thừa từ BaseController và có thể kiểm soát ở mức tối đa.
* Helper: Nhóm chúng em đã xây dựng nên các class helper độc lập với nhau, phân chia rõ ràng theo backend , frontend và các package để dễ nhận biết . Nhiệm vụ của các Helper là để giúp cho các Controller có sự độc lập hoàn toàn với nhau . Khi cần sự liên kết thì sẽ liên kết qua các hàm static của helper . Cũng như giúp cho chương trình áp dụng đúng nguyên tắc SRP, mỗi controller chỉ thao tác với 1 resource tương ứng.

*Hình ảnh minh họa các file helper*

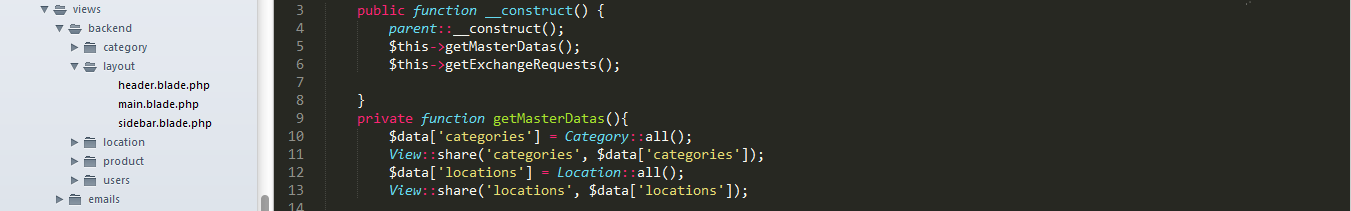
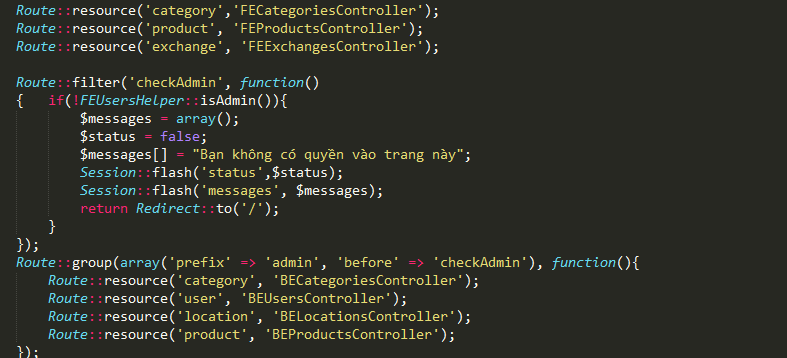
* Quan hệ các thành phần dựa trên thuộc tính bất biến:

Sử dụng user\_id làm thuộc tính bất biến trong phiên sử dung.

*Vd*: Người dùng truy cập bằng đường dẫn đến trang admin , nếu không phải admin thì sẽ chuyển về trang Home

Hay như là , dựa vào user\_id để khi view product , nếu như là người đăng sản phẩm thì sẽ chuyển đến trang edit product , còn không sẽ chuyển đến trang show.

Hoặc dựa vào số lượng product trên 1 page để gọi ajax giúp khi view sản phẩm chỉ cần scroll mà không cần load lại trang.

* Tính tích hợp: Project của chúng em sử dụng tích hợp cả ở Controller và ở View .
* Controller: Project được chia làm 2 phần frontend và backend. Mỗi phần chia làm các module thực hiện các chức năng riêng biệt như thao tác với products, thao tác với exchanges. Các controllers trong các module đều kế thừa từ FEBaseController(Front End Controller) hoặc BEBaseController(Base End Controller), và 2 controller này lại kế thừa từ BaseController.
* View: Tương tự Controller , phần view cũng được chia làm 2 phần frontend và backend . Mỗi phần chia làm các module thực hiện các chức năng riêng biệt . Các view này đều dùng chung một layout gồm có header , sidebar và main.
* Sử dụng resfull
  + Các tài nguyên trong project được ứng với các controller tương ứng. Như vậy tính hướng đối tượng đã được thể hiện trong project.
* Khả năng mở rộng : Khi cần phát triển thêm một chức năng mới , thì chỉ cần vào Controller tạo thêm một package mới với tạo controller ở trong package đó . Đồng thời tạo một package mới trong view ( có thể trùng tên với package ở controller để dễ phân biệt) chứa các hiển thị của chức năng mới .

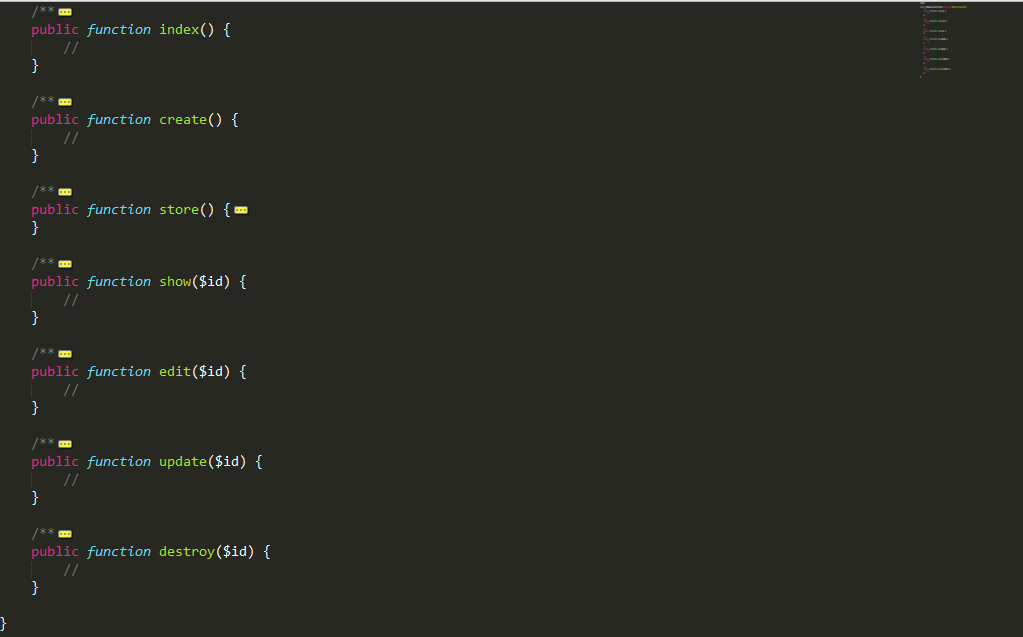
Các module này có thể kế thừa từ các Base Class để tái sử dụng các hàm có sẵn.

**CHƯƠNG 4. KIỂM THỬ**

## **Các nguyên tắc SOLID.**

* *Single Responsibility Principle (SRP)*

Mỗi class(và method trong mỗi class) chỉ nên thực hiện một mục đích, một trách nhiệm duy nhất.

* + FEUsersController: Thực hiện đăng ký người dùng.
  + FEProductsController: Thực hiện hiển thị đồ và đăng đồ
  + FERequestsController: Thực hiện quản lý trao đổi đồ.
  + BEUsersController: Thực hiện việc quản lý người dùng của admin.
  + BEProductsController: Thực hiện quản lý đồ của admin.
  + BECategoriesController:Thực hiện quản lý các category
  + Các hàm trong từng Controller:
    - Index():
    - create():
    - store():
    - show():
    - edit():
    - update():
    - destroy():

### **Single responsibility principle:** Project được chia làm 4 phần chính, model, controller , view và helper. Model có nhiệm vụ truy xuất từ DB, view để hiện thị cho người dung, controller để điều hướng và helper để xử lý các logic phát sinh.

### **Open/closed principle**

Các Controller đều kế thừa từ các BaseController, các view kế thừa từ master view, các Helper kế thừa từ class helper. Khi muốn mở rộng chương trinh thì viết các helper và controller kế thừa từ các helper và controller có sẵn.

* Liskov Substitution Principle

Các class trong model kế thừa class Eloquent nhưng vẫn có thể sử dụng các hàm của class Eloquent như save, update.

**CHƯƠNG 5. HƯỚNG DẪN CÀI ĐẶT VÀ CHƯƠNG TRÌNH MINH HỌA**

## **5.1 Hướng dẫn cài đặt**

1. Download project source code

Giải nén, đổi tên thư mục giải nén được thành btlltct, copy thư mục này vào thư mục apache root.

2. Cấu hình project mới

- Tạo một cơ sở dữ liệu tên btlltct.

- Import file btlltct.sql trong thư mục download vào cơ sở dữ liệu btlltct vừa tạo.

- DB: Mở file app/config/database.php

Sửa phần

'mysql' => array(

'driver' => 'mysql',

'host' => 'localhost',

'database' => 'btlltct',

'username' => 'root',

'password' => '12345678',

'charset' => 'utf8',

'collation' => 'utf8\_unicode\_ci',

'prefix' => '',

),

đúng với cấu hình apache hiện tại.

- Key secure

Mở file app/config/app.php

Nếu chưa có từ khóa ‘key’ thì tạo thêm 1 từ khóa ‘key’ với giá trị là 32 kí

tự bất kì.

- Lưu ý : Trên Unix OS, cần cung cấp quyền write cho thư mục bootstrap,

app/storage bằng cách vào thư mục root của project chạy lệnh

o sudo chmod –R 777 app/storage

o sudo chmod –R 777 bootstrap

3. Chạy thử project

Chạy thử project theo đường dẫn

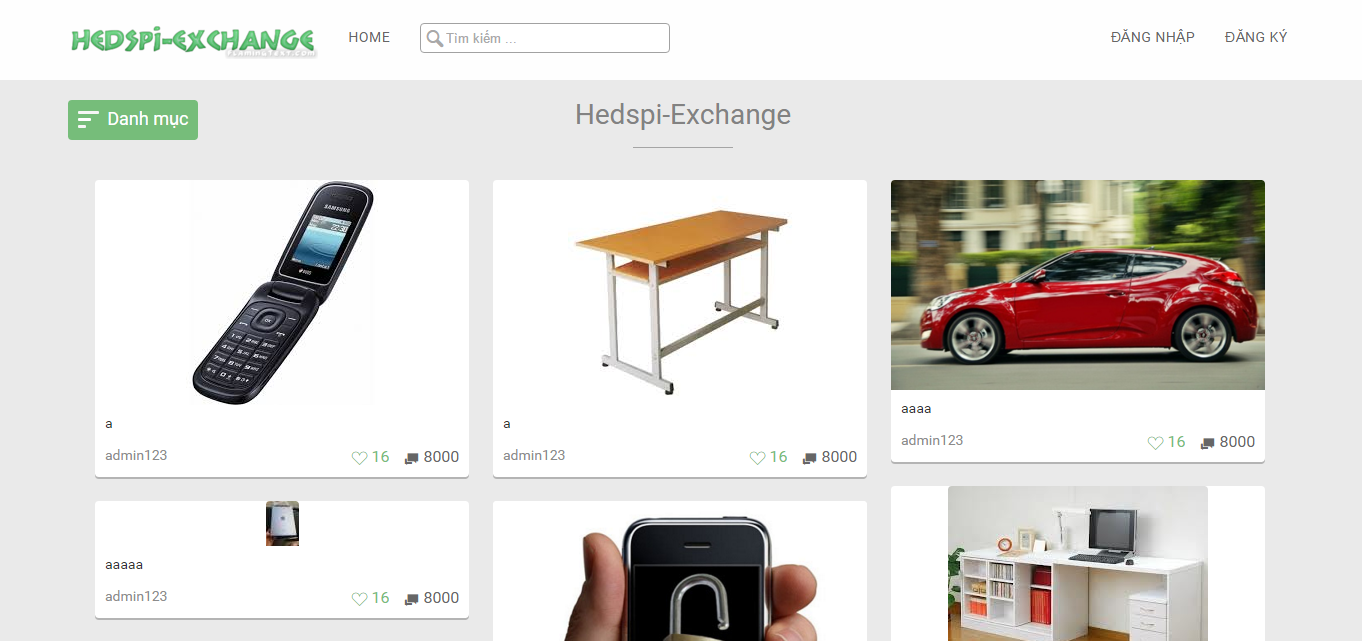
http://localhost/btlltct

Nếu hiển thị trang chủ của project là thành công.

Có thể thực hiện tìm kiếm sản phẩm theo tiêu đề.

Hoặc hiển thị các sản phẩm của category nhất định.

## **5.1 Minh họa chương trình**



*Giao diện trang chủ của chương trình*