

# 目录

1、实现环境.....	2
1、软件环境 .....	2
2、硬件环境 .....	2
2、系统结构图 .....	2
3、基本表的定义，主外码等完整性约束定义，索引的定义 .....	3
1.    comments 表格.....	3
2.    favorites 表格.....	4
3.    orders 表格.....	4
4.    Products 表格.....	4
5.    Shops 表格 .....	5
6.    UserInfos 表格.....	5
7.    Users 表格.....	6
4 系统的安全性设计，不同人员的外模式及相关权限 .....	7
1.    用户的信息保障机制 .....	7
2.    不同用户权限控制机制.....	7
3.    购物车的外模式约束 .....	7
5 存储过程、触发器和函数的代码说明.....	7
1.    存储过程 .....	7
2.    函数与触发器 .....	10
6 实现过程中主要技术论述.....	11
1.    ruby on rails 为主.....	11
2.    前端技术 .....	11
3.    后台技术 .....	11
7 若干展示系统功能的运行实例.....	12
1.    界面首页 .....	12
2.    未登录用户及普通用户 product 界面 .....	12
3.    管理员登陆的 product 界面 .....	13
4.    模糊搜索 iphone .....	13
5.    具体的 product 信息界面，产品效果图.....	14
6.    单个产品功能 .....	14
7.    产品信息修改.....	16
8.    注册登录界面 .....	16
9.    购物车.....	17
10.   商店.....	18
8 源程序简要说明 .....	19
8.1 文件说明 .....	19
8.2 运行说明 .....	19
9 收获和体会 .....	19

## 1、实现环境

### 1、软件环境

数据库：postgre(关于为什么不选择 mysql,这是篇关于 mysql 与 postgre 的对比  
详见这篇知乎, <http://www.zhihu.com/question/20010554>)

账户：psql 密码：psql 库名：appname\_application

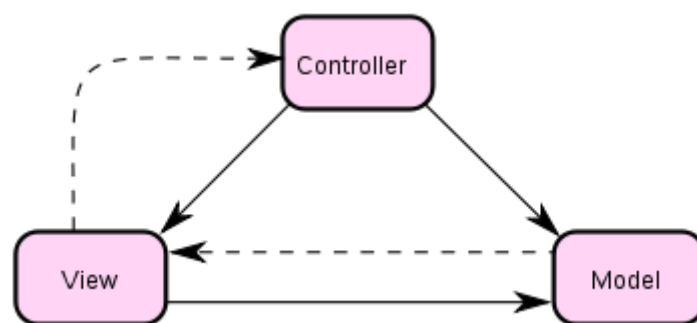
运行系统：Ubuntu

运行环境：Ruby+Rails+rbenv+vim 编辑器

### 2、硬件环境

普通 PC 机、支持 Ubuntu/Linux 系统

## 2、系统结构图



本系统使用的是 MVC 的框架,

下图的 XXX 为表示对象名, YYY 表示存储过程名, Z 为若是需要时的详细的参数。

其中 XXX 包括：

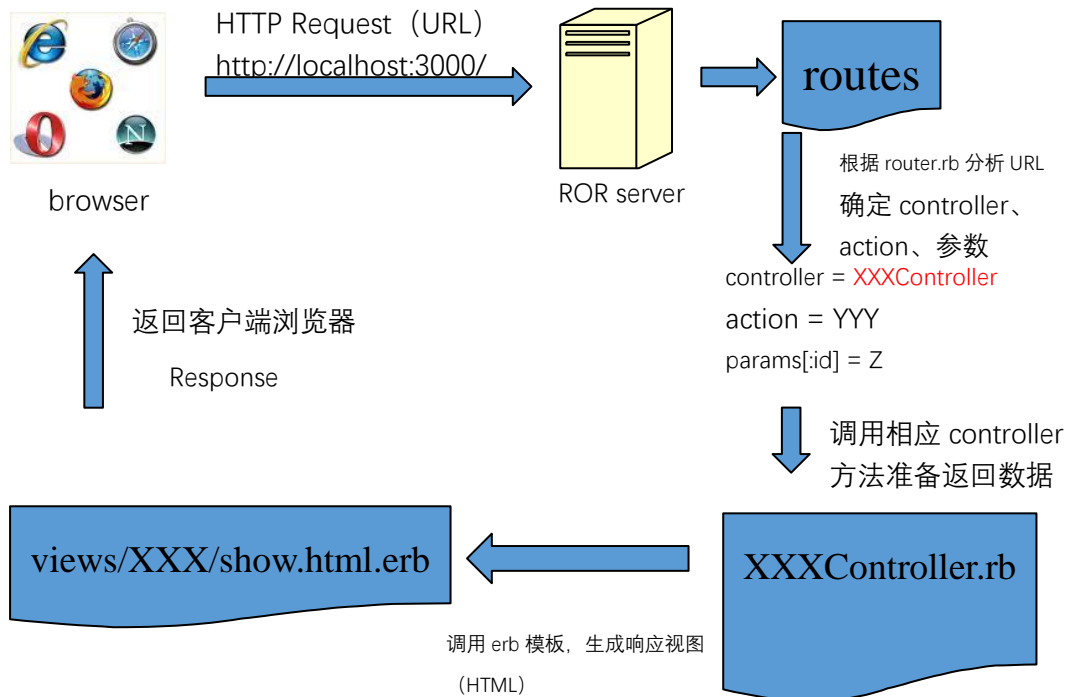
Appllication,Products,Comments,Favorites,Orders,Shops,Static\_page,UserInfos,Users

YYY 包括七个过程：

index,new,create,edit,destroy,show,update

Z 表示为：

browser 通过路由传值给 controller 的参数 id 值



### 3、基本表的定义，主外码等完整性约束定义，索引的定义

关于数据类型的定义是通过 ruby 语言来创建表格的，参见 db/schema.rb 以及 db/migrate 文件夹

#### 1. comments 表格

Column	Type	Modifiers
id	integer	not null default nextval('comments_id_seq'::regclass)
user_id	integer	
body	text	
rating	integer	
product_id	integer	
created_at	timestamp without time zone	
updated_at	timestamp without time zone	

Indexes:

"comments\_pkey" PRIMARY KEY, btree (id)  
 "index\_comments\_on\_product\_id" btree (product\_id)  
 "index\_comments\_on\_user\_id" btree (user\_id)

## 2. favorites 表格

Column	Type	Modifiers
<b>id</b>	integer	not null default nextval('favorites_id_seq'::regclass)
<b>rating</b>	integer	
<b>user_id</b>	integer	
<b>product_id</b>	integer	
<b>created_at</b>	timestamp without time zone	not null
<b>updated_at</b>	timestamp without time zone	not null

Indexes:

"favorites\_pkey" PRIMARY KEY, btree (id)  
"index\_favorites\_on\_product\_id" btree (product\_id)  
"index\_favorites\_on\_user\_id" btree (user\_id)

## 3. orders 表格

Column	Type	Modifiers
<b>id</b>	integer	not null default nextval('orders_id_seq'::regclass)
<b>user_id</b>	integer	
<b>product_id</b>	integer	
<b>total</b>	double precision	

Indexes:

"orders\_pkey" PRIMARY KEY, btree (id)  
"index\_orders\_on\_product\_id" btree (product\_id)  
"index\_orders\_on\_user\_id" btree (user\_id)

## 4. Products 表格

Column	Type	Modifiers
<b>id</b>	integer	not null default nextval('products_id_seq'::regclass)
<b>name</b>	character varying	
<b>description</b>	text	
<b>image_url</b>	character varying	

<b>price</b>	integer	
<b>created_at</b>	timestamp without time zone	
<b>updated_at</b>	timestamp without time zone	

Indexes:

"products\_pkey" PRIMARY KEY, btree (id)

Triggers:

update\_prod\_price BEFORE INSERT ON products FOR EACH ROW EXECUTE  
PROCEDURE trg\_update\_prod\_price()

## 5. Shops 表格

Column	Type	Modifiers
<b>id</b>	integer	not null default nextval('shops_id_seq'::regclass)
<b>product_id</b>	integer	
<b>name</b>	character varying	
<b>telephone</b>	character varying	
<b>location</b>	text	
<b>created_at</b>	timestamp without time zone	not null
<b>updated_at</b>	timestamp without time zone	not null

Indexes:

"shops\_pkey" PRIMARY KEY, btree (id)

"index\_shops\_on\_product\_id" btree (product\_id)

## 6. UserInfos 表格

Column	Type	Modifiers
<b>id</b>	integer	not null default nextval('userinfos_id_seq'::regclass)
<b>name</b>	character varying	
<b>age</b>	integer	
<b>idcard</b>	character varying	
<b>user_id</b>	integer	
<b>created_at</b>	timestamp without time	not null

	zone	
<b>updated_at</b>	timestamp without time zone	not null

Indexes:

- "userinfos\_pkey" PRIMARY KEY, btree (id)
- "index\_userinfos\_on\_user\_id" btree (user\_id)

## 7. Users 表格

Column	Type	Modifiers
<b>id</b>	integer	not null default nextval('users_id_seq'::regclass)
<b>first_name</b>	character varying	
<b>last_name</b>	character varying	
<b>created_at</b>	timestamp without time zone	
<b>updated_at</b>	timestamp without time zone	
<b>email</b>	character varying	not null default "::character varying"
<b>encrypted_password</b>	character varying	not null default "::character varying"
<b>reset_password_token</b>	character varying	
<b>reset_password_sent_at</b>	timestamp without time zone	
<b>remember_created_at</b>	timestamp without time zone	
<b>sign_in_count</b>	integer	default 0
<b>current_sign_in_at</b>	timestamp without time zone	
<b>last_sign_in_at</b>	timestamp without time zone	
<b>current_sign_in_ip</b>	character varying	
<b>last_sign_in_ip</b>	character varying	
<b>admin</b>	boolean	not null default false

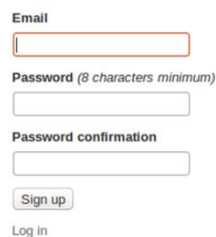
Indexes:

- "users\_pkey" PRIMARY KEY, btree (id)
- "index\_users\_on\_email" UNIQUE, btree (email)
- "index\_users\_on\_reset\_password\_token" UNIQUE, btree (reset\_password\_token)

## 4 系统的安全性设计，不同人员的外模式及相关权限

### 1. 用户的信息保障机制

- (1) 用户的密码通过 MD5 加密算法转换成长度为 64 位的字符串；
- (2) 用户申请新账号必须通过邮箱进行验证（暂时禁用，因为本地服务器的缘故）；
- (3) 用户可以通过邮箱找回密码；
- (4) 用户注册时格式有进行限制，其中账号必须要是以邮箱@的形式，密码必须要至少保证 8 个字符以上；



The image shows a user registration form. It consists of three input fields: 'Email', 'Password (8 characters minimum)', and 'Password confirmation'. Below the 'Password confirmation' field is a 'Sign up' button. At the bottom of the form is a 'Log in' link.

### 2、不同用户权限控制机制

仅有管理员能够在外模式下对商品信息、商店信息、用户评价的数据库中的数据进增删改查操作，一般用户及未登陆用户仅有查看的功能。

此部分设计为第一个申请的账号为管理员账号，并通过 cancan 进行权限管理，以此来更改 user 下的 admin 属性。

### 3、购物车的外模式约束

为防止一次用户由于点击出错或者因为要刷商品等行为，对每次购买的商品的数量进行了限制，在本系统中数量限制为 1-5

## 5 存储过程、触发器和函数的代码说明

以 product 为例说明实现的存储过程、触发器、函数三个部分。

### 1. 存储过程

在 ruby on rails 中，为适应敏捷开发的需要，将存储过程进行了封装。

封装出的存储过程由如下七个：

## 1. index

函数进行索引

```
def index
  if params[:q]
    search_term = params[:q]
    @products = Product.where("name LIKE ?", "%#{search_term}%")
  else
    @products = Product.all
  end
end
```

## 2. show

调用相关的实体对象获取数据

```
# GET /products/1
# GET /products/1.json
def show
  @comments = @product.comments.order("created_at DESC").paginate(:page =>
params[:page], :per_page => 5)
end
```

## 3. new

为在内存中创建一个实体对象

```
# GET /products/new
def new
  @product = Product.new
end
```

## 4. edit

为编辑该实体对象

```
# GET /products/1/edit
def edit

end
```



## 5. create

为在数据库中创建一个实体对象，即是一条记录

```
# POST /products
# POST /products.json
def create
  @product = Product.new(product_params)
  respond_to do |format|
    if @product.save
      format.html { redirect_to @product, notice: 'Product was successfully
created.' }
      format.json { render :show, status: :created, location: @product }
    else
      format.html { render :new }
      format.json { render json: @product.errors,
status: :unprocessable_entity }
    end
  end
end
```

## 6. update

为更新数据

```
# PATCH/PUT /products/1
# PATCH/PUT /products/1.json
def update
  respond_to do |format|
    if @product.update(product_params)
      format.html { redirect_to @product, notice: 'Product was successfully
updated.' }
      format.json { render :show, status: :ok, location: @product }
    else
      format.html { render :edit }
      format.json { render json: @product.errors,
status: :unprocessable_entity }
    end
  end
end
```

## 7. destroy

删除某一个实体对象

```

# DELETE /products/1
# DELETE /products/1.json
def destroy
  @product.destroy
  respond_to do |format|
    format.html { redirect_to products_url, notice: 'Product was successfully
destroyed.' }
    format.json { head :no_content }
  end
end
end

```

## 2. 函数与触发器

见 db/migrate/product.rb, 实现功能是对商家新添的商品, 价格若为负数, 则在插入数据库以前将其值改为 0, 以免出现价格为负数的情况。

如下为返回触发器的函数定义,

```

execute %q{
  CREATE FUNCTION trg_update_prod_price()
    RETURNS trigger AS
    $BODY$
    BEGIN
      IF NEW.price < 0 THEN
        NEW.price:=0;
        return NEW;
      END IF;
      RETURN NEW;
    END;
    $BODY$
  LANGUAGE plpgsql VOLATILE; }

```

如下为触发器的定义, 调用了函数,

```

execute %q{ CREATE TRIGGER update_prod_price
  BEFORE INSERT
  ON products
  FOR EACH ROW
  EXECUTE PROCEDURE trg_update_prod_price();}

```

## 6 实现过程中主要技术论述

### 1. ruby on rails 为主

主要开发语言为 ruby+html+js+css+sql。  
ruby on rails 为整个系统的框架结构。其中  
routes.rb 负责控制访问资源路径；  
db/migrate 负责数据库迁移；  
app/...负责整个网站的运行逻辑。

### 2. 前端技术

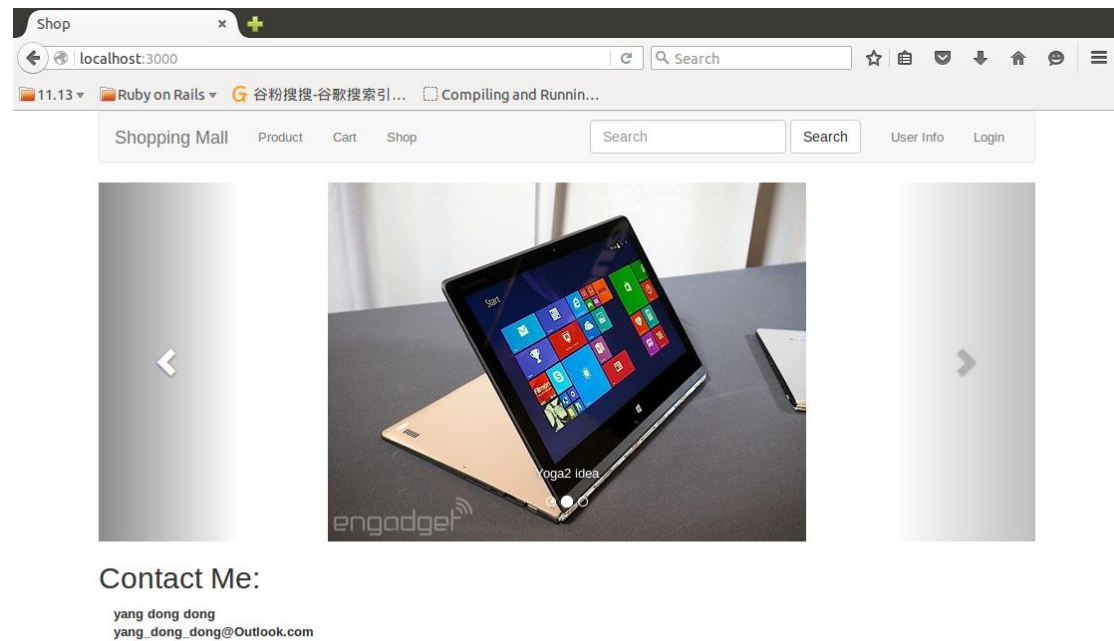
1. bootstrap.js (美化)
2. angular.js (ng-model 的前后台数据相互依赖的库)
3. raty.js (星级评价的 js 库)
4. jquery.js

### 3. 后台技术

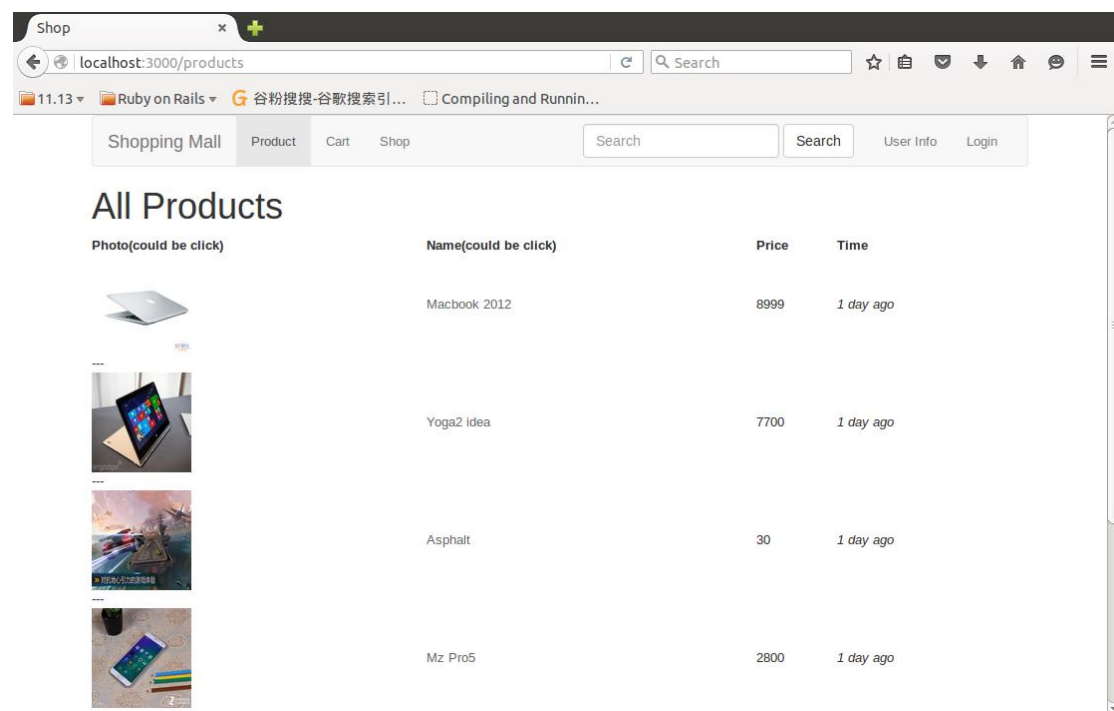
sql 语句以及 rails 的 model 相关的连接、验证、约束关系, 以及 controller 相关的实现与 views、model 层的交互控制。

## 7 若干展示系统功能的运行实例

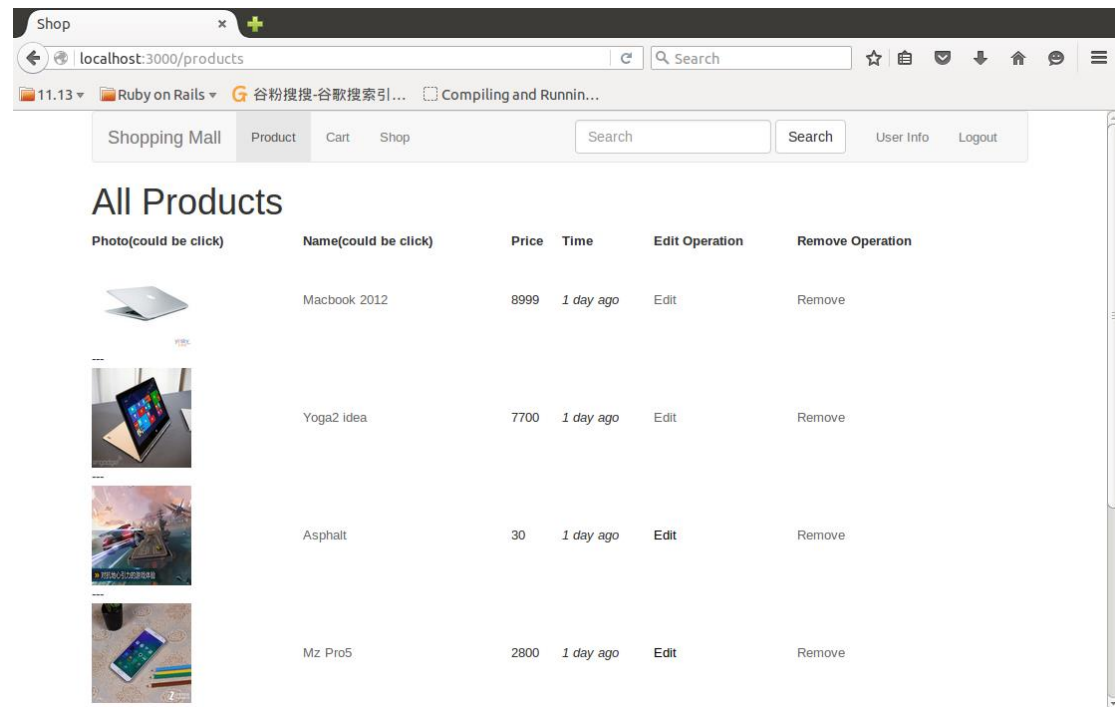
### 1. 界面首页



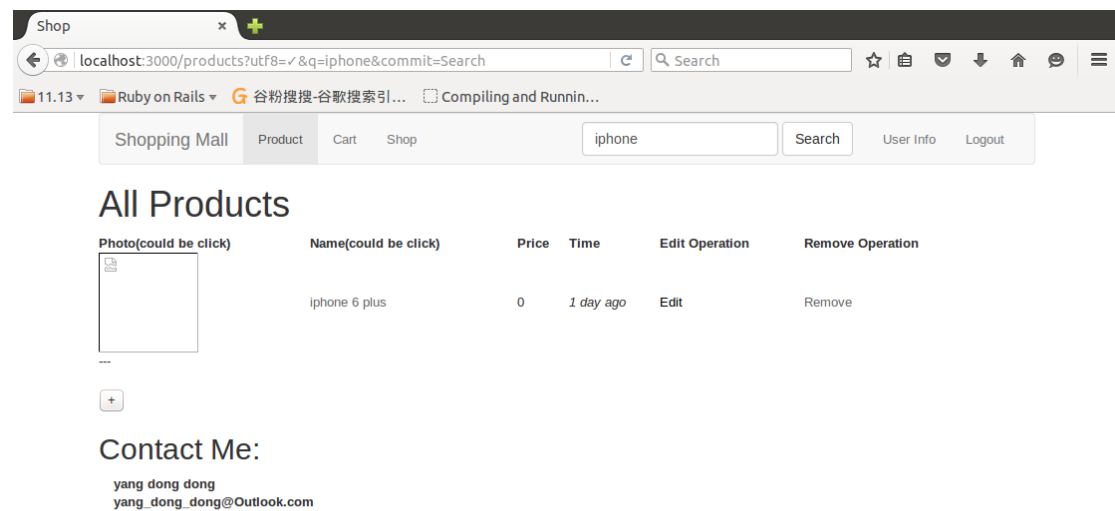
### 2. 未登录用户及普通用户 product 界面



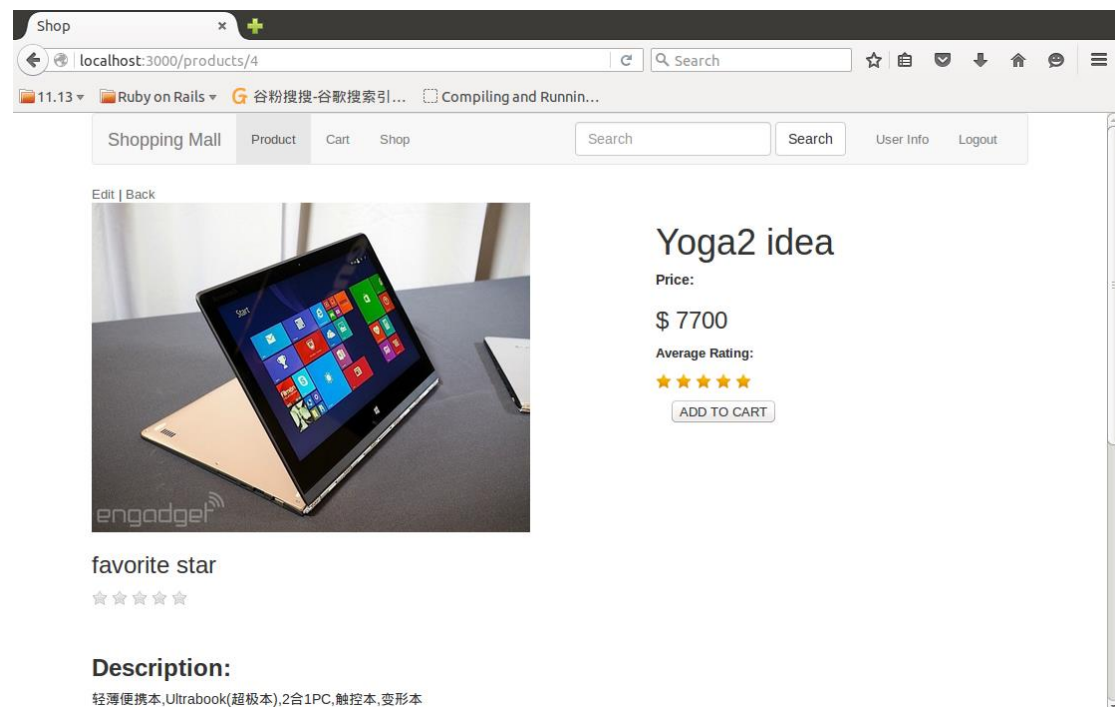
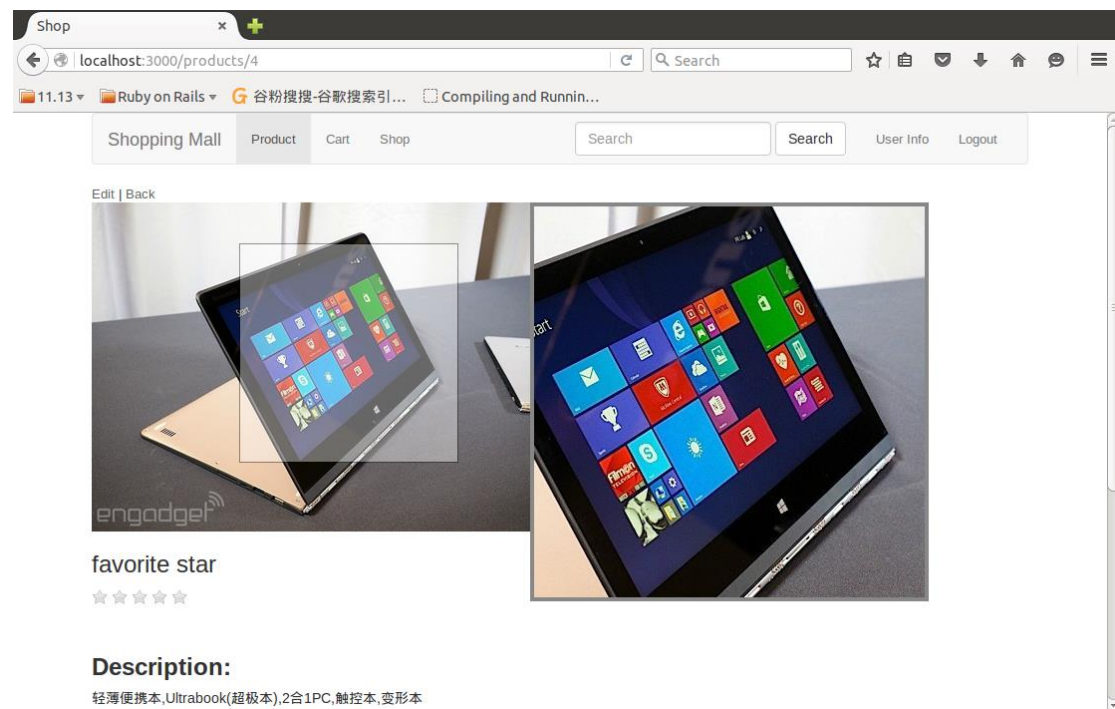
### 3. 管理员登陆的 product 界面



### 4. 模糊搜索 iphone



## 5. 具体的 product 信息界面，产品效果图



## 6. 单个产品功能

产品的星级评价、未登录不能评价、登陆后可评价、管理员能删除评价等一系列功能

Shop

localhost:3000/products/4

Search

11.13 ▾ Ruby on Rails ▾ 谷粉搜搜-谷歌索引... Compiling and Runnin...

favorite star

☆☆☆☆☆

**Description:**  
轻薄便携本,Ultrabook(超极本),2合1PC,触控本,变形本

**Comments:**  
body  
  
☆☆☆☆☆

✕

test1

☆☆☆☆☆


**Contact Me:**  
yang dong dong  
yang\_dong\_dong@Outlook.com

Shop

localhost:3000/products/4

Search

11.13 ▾ Ruby on Rails ▾ 谷粉搜搜-谷歌索引... Compiling and Runnin...



Average Rating:

☆☆☆☆☆

favorite star

☆☆☆☆☆

**Description:**  
轻薄便携本,Ultrabook(超极本),2合1PC,触控本,变形本  
(you dont log in , please log in and you will be able to comment)

**Comments:**  
test1  
☆☆☆☆☆

**Contact Me:**  
yang dong dong  
yang\_dong\_dong@Outlook.com

## 7. 产品信息修改

Shop

localhost:3000/products/4/edit

Search

11.13

Ruby on Rails

谷粉搜搜-谷歌索引...

Compiling and Runnin...

Shopping Mall

Product

Cart

Shop

Search

Search

User Info

Login

### Editing product

**Name**

Yoga2 idea

**Description**

轻薄便携本,Ultrabook(超极本),2合1PC,触控本,变形本

**Image url**

http://img2.utuku.china.com/t

**Price**

7700

Update Product

Show | Back

### Contact Me:

yang dong dong

yang\_dong\_dong@Outlook.com

## 8. 注册登录界面

Shop

localhost:3000/login

Search

11.13

Ruby on Rails

谷粉搜搜-谷歌索引...

Compiling and Runnin...

Shopping Mall

Product

Cart

Shop

Search

Search

User Info

Login

### Log in

**Email**

**Password**

☐ Remember me

Log in

Sign up

Forgot your password?

### Contact Me:

yang dong dong

yang\_dong\_dong@Outlook.com



Shop

localhost:3000/sign\_up

11.13Ruby on Rails谷粉搜搜-谷歌索引...Compiling and Runnin...

Shopping MallProductCartShop

SearchSearch

User InfoLogin

## Sign up

Email

Password (8 characters minimum)

Password confirmation

Sign up

Log in

## Contact Me:

yang dong dong

yang\_dong\_dong@Outlook.com

## 9. 购物车

Shop

localhost:3000/orders

11.13Ruby on Rails谷粉搜搜-谷歌索引...Compiling and Runnin...

Shopping MallProductCartShop

SearchSearch

User InfoLogout

## Cart

Number

Select a product

+

Number	Product	Unit Price	Total
2	Asphalt	30	<div></div>
5	Macbook 2012	8999	<div></div>

## Contact Me:

yang dong dong

yang\_dong\_dong@Outlook.com

## 10. 商店

Shop

localhost:3000/shops

Search

11.13 ▾ Ruby on Rails ▾ 谷粉搜搜-谷歌索引... Compiling and Runnin...

Shopping MallProductCartShop

SearchSearchUser InfoLogout

Shop was successfully created.

### Listing Shops

Name	Telephone	Location	Product_id		
test1	1	bj	3	Edit	Destroy

+

### Contact Me:

yang dong dong  
yang\_dong\_dong@Outlook.com

Shop

localhost:3000/userinfos/1

Search

11.13 ▾ Ruby on Rails ▾ 谷粉搜搜-谷歌索引... Compiling and Runnin...

Shopping MallProductCartShop

SearchSearchUser InfoLogout

Userinfo was successfully created.

Name: yang  
Age: 20  
Idcard: 1  
Edit | Back

### Contact Me:

yang dong dong  
yang\_dong\_dong@Outlook.com

## 8 源程序简要说明

### 8.1 文件说明

源程序文件	说明
/db/schema.rb	创建表格及相关约束
/db/migrate/	本层目录下所有文件各自单独建立一个实体，并建立函数及触发器
/app/views	ruby on rails 的 MVC 框架 V 层
/app/models	ruby on rails 的 MVC 框架 M 层
/app/controllers	ruby on rails 的 MVC 框架 C 层
/app/helpers	为每一个控制器创建一个调用的 helper 模板
/app/assets	每次初始化程序都会运行的本地 js 库
/app/mailers	发送邮件验证及取回密码的 ruby 代码

### 8.2 运行说明

- (1) ubuntu 下配置好 ruby+rails 环境；
- (2) 安装 postgres，创建 appname\_development 为名的数据库，账号密码都为 postgres
- (3) 在根目录下命令行输入以下命令：  
rake db:migrate  
(进行数据库迁移，如果遇到冲突，则将 db 目录下的 schema.rb 文件删除)  
rake db:seed  
(加载初始化数据)  
rails server  
(运行服务器)
- (4) 访问浏览器 localhost:3000

## 9 收获和体会

通过这次的数据库大作业，深刻理解了触发过程、函数、触发器的原理，在只是看课本的阶段只就对它们感觉很陌生，明明很简单却不知道怎么做。我深刻学习了数据库的相关知识，在原本已经熟悉 mysql 的情况下，尝试了商业型的、更加稳定的 postgres 数据库，这个助教们还有我们别的同学都没听过这个这点倒是让我很是意外。

以及花费时间最多的前端，为了美化前端，花费了大量的时间去调界面。所涉及到的技术，还好是在别的工程也有学过以及平常零散学习学到，所以理解起来并不难。

但是感觉最多的还是 ruby on rails 这个框架，为了好好用好 rails，几乎把大半本《rails 高级编程》看完了，不过还是有很多地方不懂的，特别是关于循环嵌套资源的那一部分。