学号： 24142401322

**南湖学院**

**毕业设计**

题目： 小型自行车店在线租赁系统的设计与实现

作 者 郭峻 届 别 2018

系 别 机械与电子工程系 专 业 计算机科学与技术

指导老师 丁卓平 职 称 副教授

完成时间 2018年5月15日

**摘要**

本系统基于现在骑行的愈发火热，以及互联网+的思维提出了小型自行车租赁系统，通过对自行车的集中管理开展租赁服务。

对于现在的大学生来说，骑行成为了一种越来越受欢迎的娱乐方式了。骑自行车几乎可以说是每个大学生都会的技能，所以对现在的大学生来说游玩自己大学所在的城市骑行可以说是最经济实惠的方式。骑行不像坐公交地铁，人多的时候会非常拥挤，可以说在拥挤的公交或地铁上连游玩的好心情都损失了一大半，它也比坐出租或滴滴更加绿色和实惠。而且骑行也不止以上所说的好处，骑行它更加的自由，从你骑行上路时就开始领略所在城市的风光了，而且它还能锻炼你的身体。综上所述，骑行对于越来越宅的大学生来说无论是身体还是精神都能带来莫大的好处。

现在的市面上同样也存在租赁自行车的服务，像是ofo，摩拜等等，但是这些自行车大多千篇一律，对于现在越来越追求潮流的大学生来说无疑是最大的不喜之处。所以我开发了这样一个自行车租赁系统，它可以提供各种各样的名牌自行车，款式新颖潮流，让大学生们花极少的金钱就可以骑着那些好看的名牌自行车好好的欣赏风景。并且本系统为门槛极低，专为小型商家服务，旨在服务大众。

**关键词**：自行车，租赁，系统设计；

Abstract

The system is based on the increasingly fierce cycling experience and the idea of Internet+ to propose a small-scale bicycle rental system to carry out rental services through the centralized management of bicycles.

For current college students, cycling has become an increasingly popular form of entertainment. Cycling can be said to be almost every college student's skill, so for today's college students to play in the city riding where their university is located can be said to be the most economical way. Riding is not like riding a subway. When people are crowded, it can be very crowded. It can be said that the good mood of playing in a crowded bus or subway is lost. It is also more green and affordable than sitting or renting. And riding is not only the above mentioned benefits, riding it more freedom, from the time you ride on the road to start to appreciate the scenery of the city, and it can also exercise your body. In summary, cycling can bring great benefits to both college and university students.

There are also services for renting bicycles on the market today, such as ofo, Mobike. However, most of these bikes are the same, and it is undoubtedly the biggest disappointment for college students who are increasingly pursuing trends. So I developed a bicycle rental system that can provide a variety of brand-name bicycles, fashionable style, so that college students can spend a very small amount of money on the good-looking brand-name bicycles to enjoy the scenery. And this system is very low threshold, designed for small businesses, designed to serve the public.

**Keywords: bicycle, lease, system design**

**目录**

**第一章 绪论----------------------------------------------------------------6**

**1.1课题背景**-----------------------------------------------------------------------**6**

**1.2内容以及意义-----------------------------------------------------------6**

**第二章 系统所用技术----------------------------------------------------7**

**2.1 开发语言-----------------------------------------------------------------7**

**2.2 开发框架-----------------------------------------------------------------7**

**2.3 缓存数据库--------------------------------------------------------------7**

**2.4 部署服务器--------------------------------------------------------------7**

**2.5 主数据库-----------------------------------------------------------------7**

**2.6 异步任务器--------------------------------------------------------------7**

**2.7 搜索引擎-----------------------------------------------------------------7**

**2.8 全文搜索框架-----------------------------------------------------------7**

**第三章 总体设计-----------------------------------------------------------8**

**3.1可行性分析---------------------------------------------------------------8**

**3.1.1经济可行性---------------------------------------------------------8**

**3.1.2技术可行性---------------------------------------------------------8**

**3.1.3系统功能可行性---------------------------------------------------8**

**3.2 任务目标-----------------------------------------------------------------8**

**第四章 系统设计---------------------------------------------------------10**

**4.1 系统概述--------------------------------------------------------------10**

**4.2 系统流程--------------------------------------------------------------11**

**4.3 数据库详情----------------------------------------------------------11**

**4.4 数据库设计----------------------------------------------------------12**

**第五章 系统详细设计与实现------------------------------------------15**

**5.1系统环境-----------------------------------------------------------------15**

**5.1.1开发环境----------------------------------------------------------15**

**5.1.2运行环境----------------------------------------------------------15**

**5.2界面设计-----------------------------------------------------------------15**

**5.3数据模型-----------------------------------------------------------------18**

**5.4视图控制-----------------------------------------------------------------21**

**5.4.1用户模块----------------------------------------------------------22**

**5.4.2自行车实体模块-------------------------------------------------29**

**5.4.3租赁管理模块----------------------------------------------------35**

**5.4.4租赁订单模块----------------------------------------------------42**

**第六章 总结---------------------------------------------------------------44**

**第七章 参考文献---------------------------------------------------------45**

**第八章 致谢---------------------------------------------------------------46**

**第一章．绪论**

**1.1课题背景**

现在是时代已经可以说已经步入了互联网的大时代，几乎生活中任何事情都可以通过互联网来完成。在这样的时代潮流下互联网+的思维俨然成为了现在的主流思想，无论是国家还是商业活动都无比重视，甚至可以说脱离了互联网很多事情都无法很好的完成。而现在的共享思维就是互联网+的一种大潮流，共享单车就是其中之一。虽然共享单车的商家众多，但是旗下的自行车都是千篇一律，对于追求个性与潮流的年轻人来说这无疑是最大的缺憾。所以为了迎合年轻人的追求个性以及潮流的特性，在系统上拥有各式各样的单车，以及各大潮流品牌自行车无疑是一种很大的商机。所以相信小型自行车租赁系统的出现能够吸引年轻人的眼光，填补这一市场空白。

**1.2内容以及意义**

网络的出现可以说是从各个角度，各个方面彻底地改变我们的生活，从思维方式到生活习惯。当我们要解决某事或者达到某些目的时，我们一般都会想去通过网络。现在共享思维的火热也让租赁行业得到很大的发展。对于现在的人来说网上租赁变得越来越平常了。而骑自行车也是一场非常吸引人的锻炼方式，不仅可以锻炼身体，而且还能愉悦心情，在欣赏风景同时还得到了锻炼，可以说是一举多得。相信在不久的未来，租赁自行车会更加的火热以及日常化。

集中式的平台，可以根据商家的具体情况以及需求把控系统的开发，系统可定制化程度高，可以满足不同商家的不同需求。在设计与实现系统中学习并且实践自己所学的思想与技术，不断充实和提高自身，增强自己的专业能力，让自身可以经受住社会市场的考验。

**1.3 拟解决的问题**

论文的研究内容依赖于年轻人对于骑行的热爱，致力于现在租赁行业的火热，弥补现在租赁行业的一些不足之处。提高用户对骑行的热爱，增强用户的骑行体验。

根据以上所述，本文论述的小型自行车店在线租赁系统，当中的自由选择自行车的款式弥补了现在自行车的租赁的千篇一律，在时尚潮流以及用户方便这些地方下功夫，让用户可以满意骑行，商家扩展商机。对两者西信息的收集和分析，自主开发的系统关键点如下：

1. 分析商家与用户的需求。
2. 了解租赁的流程。
3. 对技术的选择，按照现有情况与要求选择最合适的技术。
4. 对系统进行基础的搭建和流程的编写。

**第二章．系统所用主要技术**

**2.1 开发语言**

本系统采用的开发语言是python。python可以说是2017的风云语言了，它在17年获得了计算机年度语言的桂冠，在市场上的分量也是越来越大。而在今年，python同样搅动了不小的风云，在国内关注python的目光也越来越多了。Python诞生在1989年，是由荷兰人吉多·范罗苏姆创造，并且在1991发布了第一个开源版本。由于python开源的缘故，出现了大量的喜爱python的大牛为其保驾护航。由于这种原因，python拥有大量且广泛的第三方模块应有于各种各样的领域，不论是web，爬虫，区块链，连今年最火的人工智能都是以python为主。Python拥有众多的优点，比如，开发简洁，同样的程序它的代码只有java的一半，比如，代码规范，等等。

**2.2 开发框架django**

Django是python用于开发web的框架，它的功能十分强大，自己定义了基于mvc的mvt结构，强调代码的复用。

**2.3 缓存数据库redis**

Redis是NoSql数据存储的数据，拥有更快的读写速度，用来做缓存无疑是非常合适的

**2.4 部署服务器nginx**

Nginx是由俄罗斯人开发一款轻量级的web服务器。因为Nginx它的并发能力非常不错，而且稳定，低资源消耗，所以它也被受国内众多大型互联网公司青睐。

**2.5 主数据库mysql**

Mysql是一款免费开源的数据库，它体积小，速度快，在国内的中小型公司中非常受欢迎，也是本系统存放各类信息的数据库

**2.6 异步任务器Celery**

Celery是python开发的分布式任务调度，它可以让任务脱离主程序甚至主机。当我们遇到一些耗时耗资源的操作时可以用它完成异步任务，避免主程序的阻塞，可以极大优化web服务器的性能

**2.7 搜索引擎Whoosh**

Whoosh是python开发的全文搜索组件，易用性很高，相对与其他语言开发的组件对开发人员更加友好，而且功能完善，速度也比较快，稳定性也相当不错，不会莫名其妙的崩溃。

**2.8 全文搜索框架Haystack**

Haystack是一个全文搜索框架，对于一个搜索引擎来说框架无疑是非常重要的。框架与搜索引擎的关系可以类比汽车与引擎的关系。

**第三章．总体设计**

**3.1 可行性分析**

现在的共享思维是互联网主思维之一，所以可以说网上租赁看似有很大市场，但是因为其火热的状况，所以想要设计出吸引人，能在市场上站稳脚跟的良好系统无疑不是件简单的事情，所以做好万全的准备是重中之重。我们需要从各个角度分析好处境，把握好每一步才能设计出一个合格的系统。为了达到此目的我们需要做好技术可行性分析，经济可行性分析，系统功能分析。

**3.1.1 经济可行性分析**

现在的大学生对于骑行的热爱程度可以说是越来越高了。对于大学生来说尽情的游览自己所在大学的城市风光，骑行可以说是最佳方式，无论是从经济还是潮流，甚至是游览体验。而拥有众多品牌样式的自行车商城对年轻人来说是非常吸引人了。可以让用户在商城上自主选择自己喜欢的自行车款式进行租赁。虽然说现在共享单车的无商家模式是主流，但是还是有一定的局限性的。因为共享单车的停放地点不一定，而且数量上也不一定可以满足要求，再加上经常有损坏的自行车。而本系统采用的固定地点，可以很好的解决此问题。

**3.1.2 技术可行性**

良好的系统设计是保证用户体验的基本要求，而想要做好系统设计就必须结合现在的技术实现程度才能最大限度的完善。本系统需要做好自行车的商家管理以及用户的登陆注册，下单，浏览处理，以及商品搜索功能，以及用户自主一些操作，以及多用户下单并行处理。

**3.1.3 系统功能分析**

租赁功能是自行车租赁系统的基本，本系统提供给用户登录注册，自行车浏览，自行车搜索，自行车详情查看，以及最主要的下单功能。本系统的主要模块如下：

登陆注册模块：

用户在自定义自己信息在系统完成注册以及登陆

主页展示模块：

在主页面上用户进行自行车的大致浏览，查看详情

租赁管理模块：

进入购物车模块，锁定好自己的心仪的自行车

订单模块：

结算自行车

**3.2 任务设计目标**

（1）项目目的

降低喜欢骑行的同学的出行成本，加大现在大学生出行的热情；

商家对对自行车有效管理。将自行车展现给用户，并让用户可以查看自行车，租赁自行车，搜索自行车

（2） 项目功能

自行车一编号形式进行增删改查；

用户进行注册，登入，查询，租赁操作；

设立系统版面，是商家拥有对本系统的所有权限；

（3） 系统特点：

本系统具有以下特点

模块化：本系统采用的是基于mvc结构的MTV结构。将html页面代码，数据处理部分代码，数据哭交互部分代码模块化，以便于代码的复用，以及系统的升级，修改等。

安全化：本系统为了系统安全采取了一系列措施来防止用户的错误操作，sql注入，系统错误。能保证用户和商家的信息安全和财产安全。

人性化：该系统的操作简单，界面友好，可以极大的增加用户体验。

**第四章．系统设计**

**4.1 系统概述：**

系统开发采用的是web开发的b/s三层结构，基于django的mtv架构，M模型，构建数据库模型，连接数据库；T模板，放置html页面；V视图，系统逻辑事务处理，根据浏览器请求渲染模板，连接数据库。

B/s结构是浏览器对应服务器的结构，由于现在互联网的高速发展而兴起的一种结构，虽然就安全性而言会比c/s结构有所不如，但是由于其易维护性，系统更新的方便性，所以这种结构的成本更加低，得到了市场的青睐。

系统模块：本系统主要包括两个模块，个人用户模块以及系统管理员模块，详细描述如图4.1，4.2

小型自行车租赁系统

系统用户

个人用户模块

租赁管 理

登陆

注册

页面

自行车租赁页面

库存管理

个人信息界面

自行车展示

图4.1

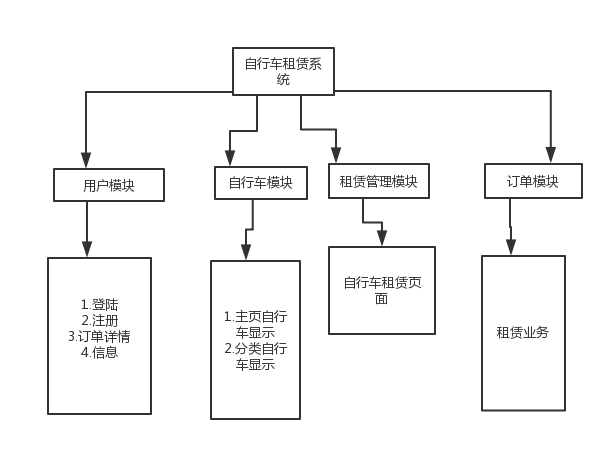


图4.2

用户：注册，登陆，显示个人信息，显示订单信息，显示浏览记录

系统：后台管理功能

**4.2 系统流程：**

用户在注册页面完成注册，登陆页面完成登陆功能。

主页面无差别对待用户进行展示商品，接入自行车详情页面。

对没有登陆的游客限制进入租赁页面，订单页面，以及用户个人中心页面。

**4.3 数据库详情：**

本系统按照系统的模块化以及信息需求设计了七张数据表，分别是User表，存放用户的账号信息，管理用户的登陆以及注册的数据；UserInfo表，存放用户的基本信息，比如昵称，手机号等；goods表，自行车实体的主类，存放自行车的基本信息，goodsInfo表，自行车信息类，存放自行车的款式信息；brand表，存放自行车品牌信息，可以进一步的分类并管理自行车；order表，订单表，记录用户租赁的自行车，以便结算；orderInfo表，订单信息表，存放单个商品的租赁信息，以便用户租赁结束后恢复商品的库存。

系统为了提高程序的性能以及系统维护的便利性，放弃了数据表外键的设置。

以下为数据库的详细设计。

**4.4 数据库设计**

系统E-R图：

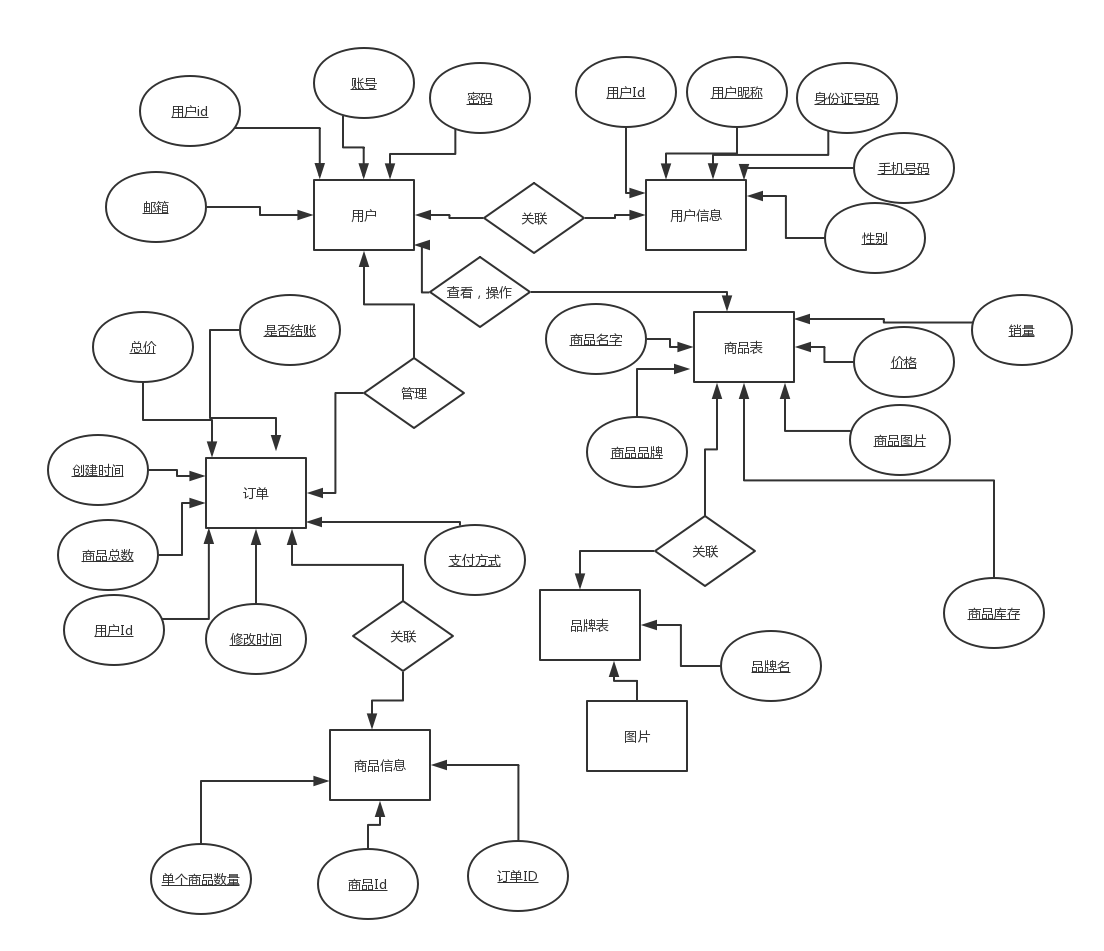


图4.3

用户表

继承了django自带的用户类，自带有账号，密码，邮箱字段。

用户信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Id | Int（11） | No | PRI | Null | Auto\_increment |
| userId | Int（11） | No |  | Null |  |
| NickName | Varchar（20） | No |  | Null |  |
| Phone | Varchr（11） | No |  | Null |  |
| Gender | Boolean | No |  | False |  |
| Idcard | vachar（20） | No |  | Null |  |

表4.1

订单表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Id | Int（11） | No | PRI | Null | Auto\_increment |
| time | datetime | No |  | Null |  |
| CarId | Int（11） | No |  | Null |  |
| UserId | Varchar（20） | No |  | Null |  |
| LastTime | Datetime | No |  | Null |  |
| IsOver | Boollean | No |  | False |  |
| GoodsNum | Int（11） | No |  | Null |  |
| Paymethod | Varchar（11） | No |  | Null |  |
| amount | Varchar（16） | No |  | Null |  |

表4.2

订单详情表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Id | Int（11） | No | PRI | Null | Auto\_increment |
| OrderId | Int（11） | No |  | Null |  |
| SkuId | Int（11） | No |  | Null |  |
| SkuNum | Int（11） | No |  | Null |  |

表4.3

自行车实体表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Id | Int（11） | No | PRI | Null | Auto\_increment |
| StyleId | Int（11） | No |  | Null |  |
| Num | Int（11） | No |  | Null |  |
| Price | Int（11） | No |  | Null |  |
| Img | Image | No |  | Null |  |
| BrandId | Int（11） | No |  | Null |  |
| Sales | Int（11） | No |  | Null |  |
| Goods | Varchar（20） | No |  | Null |  |

表4.3

自行车种类表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Id | Int（11） | No | Pri | Null | Auto\_increment |
| Color | Varchar | No |  | Null |  |
| Brand | Varchar | No |  | Null |  |
| Style | Varchar | No |  | Null |  |

表4.4

品牌表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Id | Int（11） | No | Pri | Null | Auto\_increment |
| Brand | Varchar（20） | No |  | Null |  |
| Img | Image | No |  | Null |  |

表4.5

收益表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| Id | Int（11） | No | Pri | Null | Auto\_increment |
| Income | Int（11） | No |  | Null |  |
| Time | Datatime | No |  | Null |  |

表4.6

**第五章．系统详细设计与实现**

**5.1 系统环境：**

开发环境：

IDE：PYCHARM

主数据库：MYSQL

系统：linux ubuntu

运行环境：

系统：linux ubuntu

浏览器：ie6及以上版本的web浏览器

**5.2 界面设计**

主界面

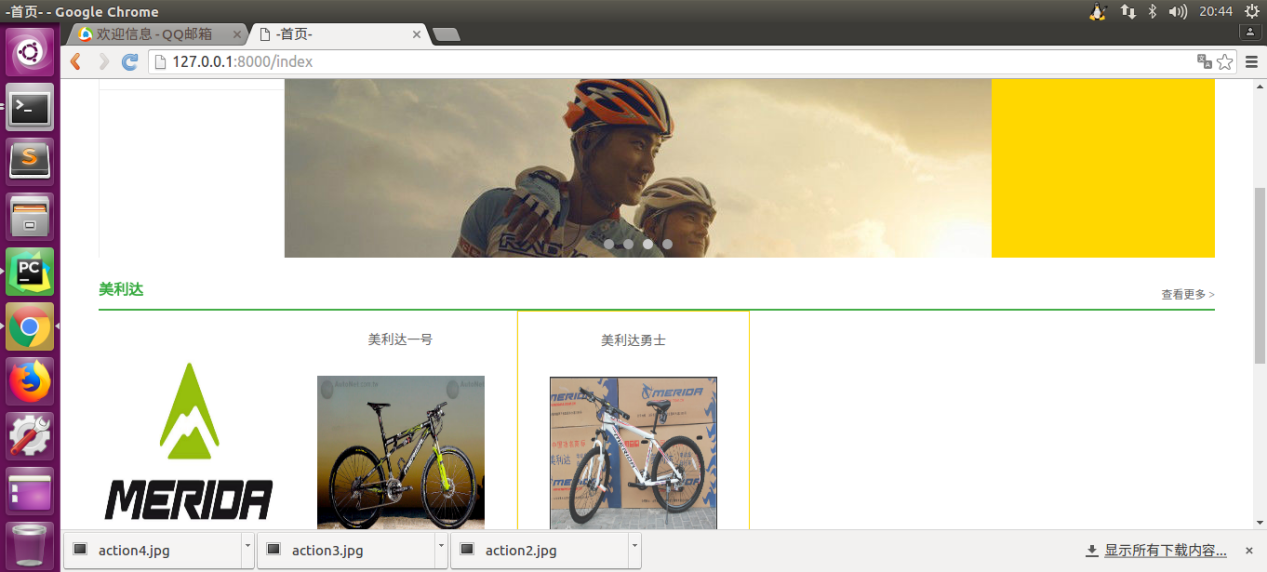


图5.1

列表页

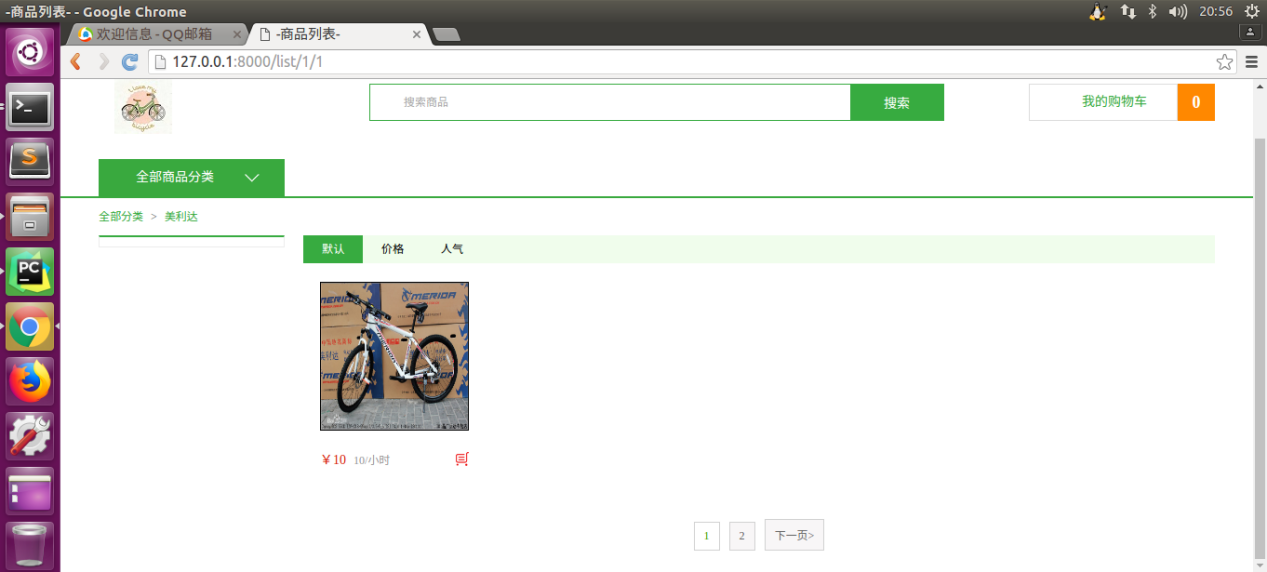


图5.2

注册登陆页面



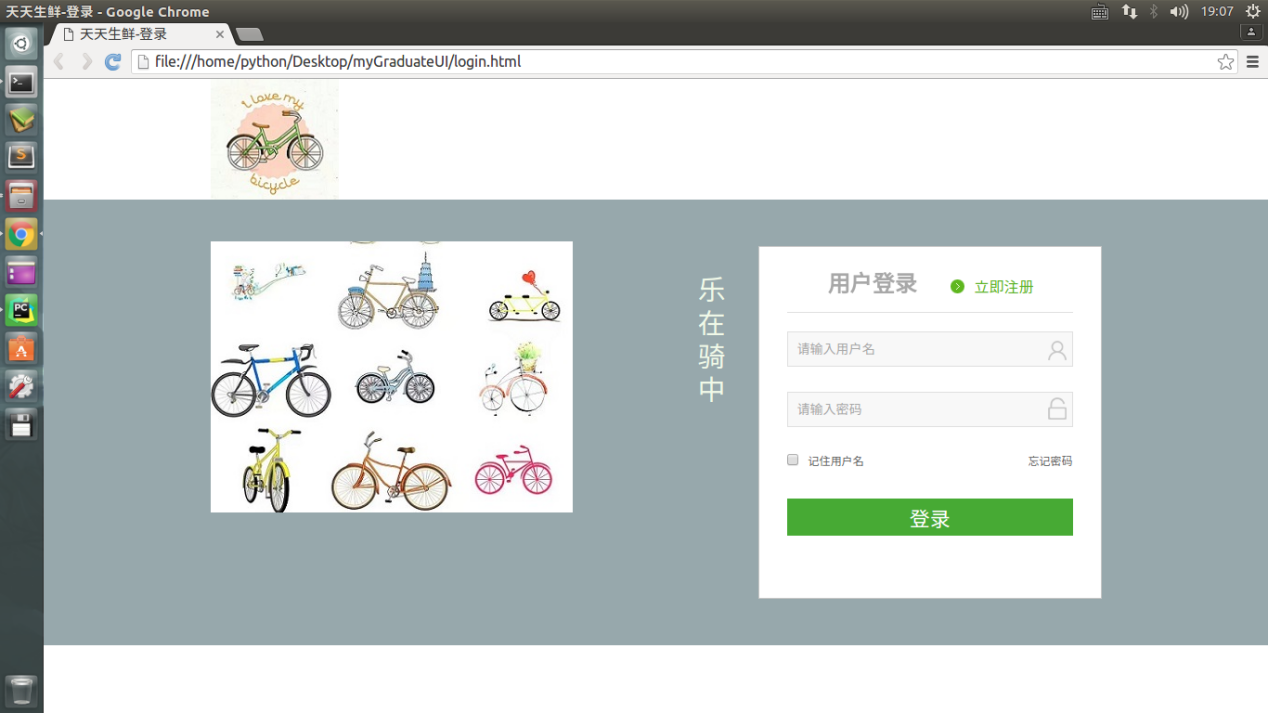


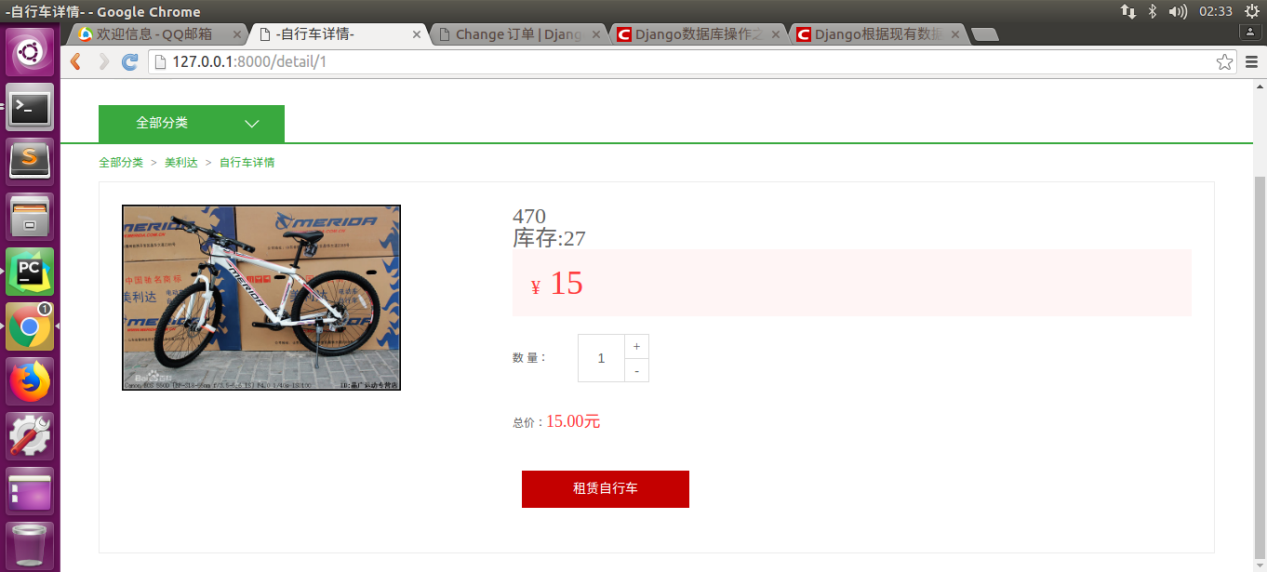
图5.3

用户中心页面



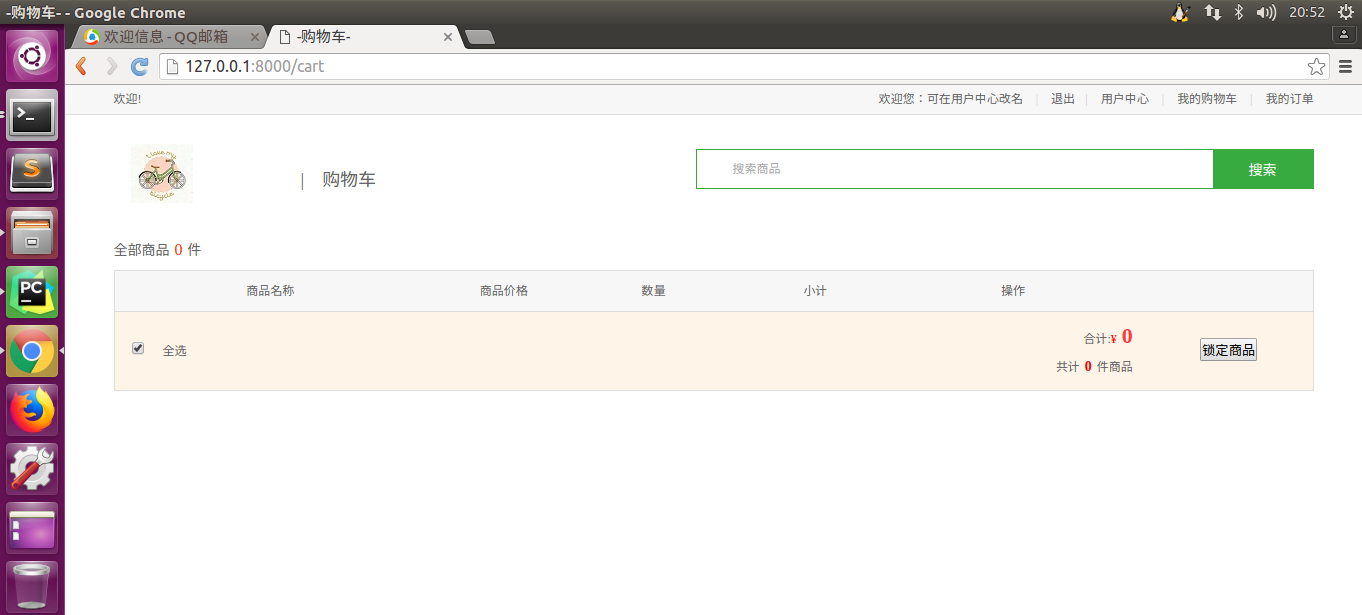
自行车详情页

图5.5



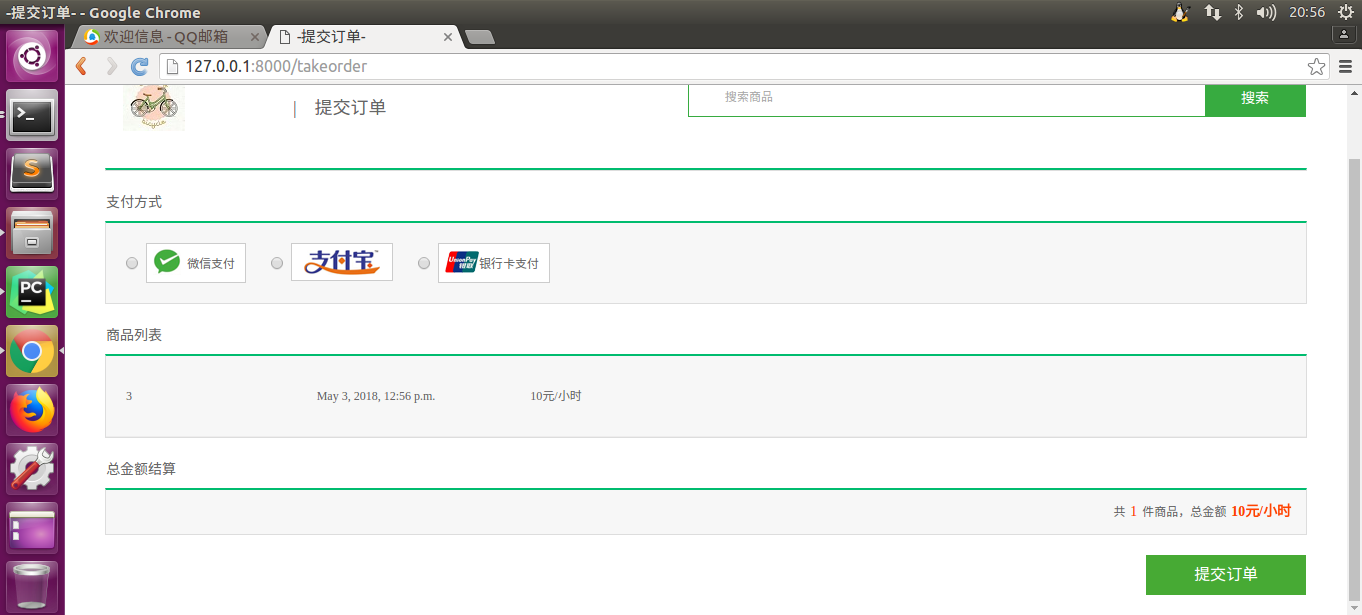
管理订单页

图5.6



支付页

图5.7



**5.3 模型数据**：

模型类是用来控制系统和数据库的交互的模块。

from django.db import models

from django.contrib.auth.models import AbstractUser

# Create your models here.

class User(AbstractUser, models.Model):

'''user模型类，继承AbstractUser类，是注册时候必填的信息'''

# AbstractUser类中自定义了用户的三个要素username，password，email，并且实现了django自带的认证系统

class Meta:

db\_table = 'User'

verbose\_name = '用户'

verbose\_name\_plural = verbose\_name

class UserInfo(models.Model):

'''用户信息表，用户注册成功后在用户中心补充信息'''

# 对应的用户Id，为了增删改查的方便和系统效率没有设置外键

UserId = models.IntegerField()

# 用户昵称

NickName = models.CharField(max\_length=20)

# 用户手机号码

Phone = models.CharField(max\_length=11)

# 用户性别

Gender = models.BooleanField(default=False)

# 用户身份证信息

IdCard = models.CharField(max\_length=20)

class Meta:

db\_table = 'UserInfo'

verbose\_name = '用户信息'

verbose\_name\_plural = verbose\_name

class Order(models.Model):

'''订单模型'''

# auto\_now\_add 表示创建时间

time = models.DateTimeField(auto\_now\_add=True)

# 商品数量

goodsNum = models.IntegerField()

# 对应的用户

userId = models.CharField(max\_length=20)

# auto\_now 表示修改时间

lastTime = models.DateTimeField(auto\_now=True)

# 表示订单是否结账

isOver = models.BooleanField(default=False)

# 支付方式

payMethod = models.CharField(max\_length=11)

# 商品总价

amount = models.CharField(max\_length=16)

class Meta:

# 定义订单表

db\_table = 'Order'

verbose\_name = '订单'

verbose\_name\_plural = verbose\_name

class OrderInfo(models.Model):

'''提交订单后将商品的库存更新的数据'''

# 对应订单表id

orderId = models.IntegerField()

# 对应商品id

skuId = models.IntegerField()

# 对应商品数量

skuNum = models.IntegerField()

class Goods(models.Model):

'''自行车商品模型'''

# 商品名字

goods = models.CharField(max\_length=20)

# 对于品牌ID

brandId = models.IntegerField()

# 对应型号ID

styleId = models.IntegerField()

# 商品数量

goodsNum = models.IntegerField()

# 当图片上传成功了之后，项目文件下会添加一个media的文件夹，并且图片也会自动放在该文件夹下upload\_to设置的文件夹路径下

# 图片

img = models.ImageField(upload\_to='theSystem')

# 价格

price = models.IntegerField()

# 销量

sales= models.IntegerField()

class Meta:

# 定义商品表

db\_table = 'Goods'

verbose\_name = '商品'

verbose\_name\_plural = verbose\_name

def \_\_str\_\_(self):

return self.goods

class GoodsInfo(models.Model):

# 商品颜色

color = models.CharField(max\_length=10)

# 商品品牌

brand = models.CharField(max\_length=16)

# 商品型号

type = models.CharField(max\_length=16)

class Meta:

# 定义商品种类表名称

db\_table = 'GoodsInfo'

verbose\_name = '商品信息'

verbose\_name\_plural = verbose\_name

def \_\_str\_\_(self):

return self.id

class Brand(models.Model):

"""自行车品牌表"""

# 商品品牌

brand = models.CharField(max\_length=10)

# 品牌图片

img = models.ImageField(upload\_to='theSystem')

class Meta:

# 定义商品种类表名称

db\_table = 'BrandInfo'

verbose\_name = '商品品牌'

verbose\_name\_plural = verbose\_name

def \_\_str\_\_(self):

return self.brand

**5.4 视图控制**：

处理客户端传来请求，完成与数据库的交互，渲染模板，并将模板文件发送回浏览器。视图共分为四个模块，分别是用户模块，商品模块，购物车模块，以及订单模块

from datetime import datetime # 时间模块函数

from django.shortcuts import render,redirect # 模板重定向以及渲染

from django.core.urlresolvers import reverse # 模板的方向解析

from django.core.paginator import Paginator # 模板的分页函数

from django.db import transaction # 数据库事务

from django.views.generic import View # 类视图

from django.contrib.auth import authenticate, login, logout # django的认证系统

from utils.mytool import LoginRequiredMixin # 登陆状态验证

from django.http import HttpResponse # 应答

from zCelery\_Tool.tasks import send\_register\_active\_email # 激活邮件发送

from itsdangerous import TimedJSONWebSignatureSerializer as Serializer # 加密

from itsdangerous import SignatureExpired # 密码失效抛出

from django\_redis import get\_redis\_connection # python与redis交互

from django.core.cache import cache # 缓存设置

from django.http import JsonResponse # ajax数据返回

from django.conf import settings # 静态设置

from theSystem.models import \* # 模型类

import re # 正则匹配

# Create your views here.

**5.4.1 用户模块：**

用户模块由七个函数组成Register类，LoginView类，LogoutView类，ActivateView类，UserInfoView类，OrderView类，SiteView类，下面是详细解释：

Register类：用户在注册页面填写账号，密码，以及邮箱，在经过系统的认证之后通过注册就进入到主页面。但是为了无良账号，本系统设置了激活程序，未激活的账号依然不能进行正常操作。所以需要进行激活操作。

ActivateView类：本系统的激活程序，在点击登陆后后自动执行此函数，为了用户良好体验，此函数是交由celery异步任务器执行，不会让用户堵塞在激活程序中。只需用户在一小时之中进入邮箱进行激活即可。

LoginView类：用户在登陆页面填写账号密码，点击登陆后，经过系统验证，跳转到主页。

UserInfoView类：展现用户个人信息，以及用户浏览过的自行车。

OrderView类：展现用户的订单信息，进行订单进入订单的结账页面

SiteView类：自定义用户的一些基本信息，身份证，昵称，手机号码，性别等

# 注册

class Register(View):

'''注册功能实现'''

'''from django.views.generic import View 实现类视图'''

def get(self, request):

return render(request,'register.html')

def post(self, request):

username = request.POST.get('user\_name')

password = request.POST.get('pwd')

email = request.POST.get('email')

allow = request.POST.get('allow')

# idCard = request.POST.get('IDcard')

if not all([username, password, email]):

return render(request, 'register.html', {'errmsg':'数据不完整'})

If not re.match(r'^[a-z0-9][\w.\-]\*@[a-z0-9\-]+(\.[a-z]{2,5}){1,2}$', email):

return render(request, 'register.html', {'errmsg': '邮箱格式不正确'})

if allow != 'on':

return render(request, 'register.html', {'errmsg': '请同意协议'})

try:

user = User.objects.get(username=username)

except User.DoesNotExist:

# 用户名不存在

user = None

if user:

# 用户名已存在

return render(request, 'register.html', {'errmsg': '用户名已存在'})

# 进行业务处理: 进行用户注册

user = User.objects.create\_user(username, email, password)

user.is\_active = 0

user.save()

# # 加密用户的身份信息，生成激活token

serializer = Serializer(settings.SECRET\_KEY, 3600)

info = {'confirm':user.id}

token = serializer.dumps(info) # bytes

token = token.decode()

# 发邮件

send\_register\_active\_email.delay(email, username, token)

# 返回应答, 跳转到首页

return redirect(reverse('system:index'))

# return HttpResponse('ok')

# 登陆

class LoginView(View):

'''登陆功能实现'''

def get(self, request):

if 'username' in request.COOKIES:

username = request.COOKIES.get('username')

checked = 'checked'

else:

username = ''

checked = ''

# 使用模板

return render(request, 'login.html', {'username':username, 'checked':checked})

def post(self,request):

'''登录校验'''

# 接收数据

username = request.POST.get('username')

password = request.POST.get('pwd')

# 校验数据

if not all([username, password]):

return render(request, 'login.html', {'errmsg': '数据不完整'})

# 业务处理:登录校验

user = authenticate(username=username, password=password)

if user is not None:

# 用户名密码正确

if user.is\_active:

# 用户已激活

# 记录用户的登录状态

login(request, user)

# 获取登录后所要跳转到的地址

# 默认跳转到首页

next\_url = request.GET.get('next', reverse('system:user'))

# 跳转到next\_url

response = redirect(next\_url) # HttpResponseRedirect

# 判断是否需要记住用户名

remember=request.POST.get('remember')

if remember == 'on':

# 记住用户名

response.set\_cookie('username', username, max\_age=7 \* 24 \* 3600)

else:

response.delete\_cookie('username')

# 返回response

return response

else:

# 用户未激活

return render(request, 'login.html', {'errmsg': '账户未激活'})

else:

# 用户名或密码错误

return render(request, 'login.html', {'errmsg': '用户名或密码错误'})

# 登出

class LogoutView(View):

'''退出登录'''

def get(self, request):

'''退出登录'''

# 清除用户的session信息

logout(request)

# 跳转到首页

return redirect(reverse('system:index'))

# 激活账户

class ActiveView(View):

'''用户激活'''

def get(self, request, token):

'''进行用户激活'''

# 进行解密，获取要激活的用户信息

serializer = Serializer(settings.SECRET\_KEY, 3600)

try:

info = serializer.loads(token)

# 获取待激活用户的id

user\_id = info['confirm']

# 根据id获取用户信息

user = User.objects.get(id=user\_id)

user.is\_active = 1

user.save()

# 跳转到登录页面

return redirect(reverse('system:login'))

except SignatureExpired as e:

# 激活链接已过期

return HttpResponse('激活链接已过期')

'''

celety 服务器启动

celery -A zCelery\_Tool.task worker -l info'''

# 用户中心

class UserInfoView(LoginRequiredMixin, View):

'''用户中心-信息页'''

def get(self, request):

'''显示'''

# Django会给request对象添加一个属性request.user

# 如果用户未登录->user是AnonymousUser类的一个实例对象

# 如果用户登录->user是User类的一个实例对象

# 如果页面发啦ajax请求,最好用request对象的user

# request.user.is\_authenticated()

# 获取用户的个人信息

user = request.user

try:

userInfo = UserInfo.objects.get(UserId=user.id)

except UserInfo.DoesNotExist as e:

userInfo = None

# 获取用户的历史浏览记录

# from redis import StrictRedis

# sr = StrictRedis(host='172.16.179.130', port='6379', db=9)

con = get\_redis\_connection('default')

history\_key = 'history\_%d'%user.id

# 获取用户最新浏览的5个商品的id

sku\_ids = con.lrange(history\_key, 0, 4) # [2,3,1]

print(sku\_ids)

# 从数据库中查询用户浏览的商品的具体信息

# goods\_li = GoodsSKU.objects.filter(id\_\_in=sku\_ids)

#

# goods\_res = []

# for a\_id in sku\_ids:

# for goods in goods\_li:

# if a\_id == goods.id:

# goods\_res.append(goods)

# 遍历获取用户浏览的商品信息

goods\_li = []

for id in sku\_ids:

goods = Goods.objects.get(id=id)

print(goods)

goods\_li.append(goods)

print(goods\_li)

# 组织上下文

context = {'page':'user',

'userInfo':userInfo,

'goods\_li':goods\_li}

# 除了你给模板文件传递的模板变量之外，django框架会把request.user也传给模板文件

return render(request, 'user\_center\_info.html', context)

# 订单中心

class OrderView(LoginRequiredMixin, View):

'''用户中心-订单页'''

def get(self, request, page=None):

'''显示'''

# 获取用户的订单信息

user = request.user

orders = Order.objects.filter(userId=user.id, isOver=False).order\_by('-time')

# 分页

paginator = Paginator(orders, 1)

# 获取第page页的内容

try:

page = int(page)

except Exception as e:

page = 1

if page > paginator.num\_pages:

page = 1

# 获取第page页的Page实例对象

order\_page = paginator.page(page)

# todo: 进行页码的控制，页面上最多显示5个页码

# 1.总页数小于5页，页面上显示所有页码

# 2.如果当前页是前3页，显示1-5页

# 3.如果当前页是后3页，显示后5页

# 4.其他情况，显示当前页的前2页，当前页，当前页的后2页

num\_pages = paginator.num\_pages

if num\_pages < 5:

pages = range(1, num\_pages + 1)

elif page <= 3:

pages = range(1, 6)

elif num\_pages - page <= 2:

pages = range(num\_pages - 4, num\_pages + 1)

else:

pages = range(page - 2, page + 3)

# 组织上下文

context = {'order\_page': order\_page,

'pages': pages,

'orders': orders}

# 使用模板

return render(request, 'user\_order\_info.html', context)

# 信息中兴

class SiteView(LoginRequiredMixin, View):

'''用户中心-地址页'''

def get(self, request):

'''显示'''

# 获取登录用户对应User对象

user = request.user

userInfo = UserInfo.objects.filter(UserId=user.id)

# 使用模板

return render(request, 'user\_center\_site.html', {'page':'userInfo', 'userInfo':userInfo})

def post(self, request):

'''地址的添加'''

# 接收数据

nickname = request.POST.get('nickname')

idcard = request.POST.get('idcard')

gender = request.POST.get('gender')

phone = request.POST.get('phone')

# 校验数据

if not all([nickname, idcard, phone, gender]):

return render(request, 'user\_center\_site.html', {'errmsg':'数据不完整'})

# 校验手机号

if not re.match(r'^1[3|4|5|7|8][0-9]{9}$', phone):

return render(request, 'user\_center\_site.html', {'errmsg':'手机格式不正确'})

# 获取登录用户对应User对象

user = request.user

userInfo = UserInfo.objects.filter(UserId=user.id)

if gender:

gender = True

else:

gender = False

if userInfo:

is\_default = False

else:

is\_default = True

# 添加地址

UserInfo.objects.create(UserId=user.id,

NickName=nickname,

Phone=phone,

Gender=gender,

IdCard=idcard

)

# 返回应答,刷新地址页面

return redirect(reverse('system:site')) # get请求方式

**5.4.2 自行车实体模块**

主页面模块由四个函数组成：Index类，Detail类，ListView类，CartAddView类

Index类：本系统的主页面，主要功能是系统分类展示自行车商品，按照品牌将自行车分类，将主要自行车展现给用户观看，并将自行车的基本信息显示。页面中包含搜索功能，并通过模板的继承将从功能传递给主页相关页面。主页拥有自己的一套缓存，可以有效的抵御DDOS攻击，提高系统性能。

Detail类：本系统的商品详情页面，将商品的详细信息展现出来，包括商品的价格，库存，名字。用户可以在此页面决定租赁此商品的数量。通过前端js函数监测商品数量的数据格式，若是数据格式错误则不通过ajax将数据传入后台。用户进入详情页面后，系统会将此记录存入redis数据库中，用户中心中会显示此浏览记录。

CartAddView类：接受商品详情页面通过ajax请求传来的商品数量，验证商品的库存，要是库存不足则返回错误信息。

ListView类：将选定品牌的所有商品展现给用户。用户可以在页面上指定商品的排序方式，后台接受到数据后按照用户指定的排序方式将商品排列，展示给用户。

# 首页

class Index(View):

def get(self, request):

'''显示首页'''

# 尝试从缓存中获取数据

context = cache.get('index\_page\_data')

if context is None:

# 缓存中没有数据

# 获取商品的种类信息

types = Brand.objects.all()

for type in types:

good = Goods.objects.filter(brandId=type.id)[:4]

# name = good.goods

# good\_id = good.id

# goods = {'good':name, 'id':good\_id}

type.good = good

context = {'types': types,

}

# 设置缓存

# key value timeout

cache.set('index\_page\_data', context, 3600)

# 获取用户购物车中商品的数目

user = request.user

cart\_count = 0

if user.is\_authenticated():

# 用户已登录

nickname = UserInfo.objects.filter(UserId=user.id, id=1)

conn = get\_redis\_connection('default')

cart\_key = 'cart\_%d' % user.id

cart\_count = conn.hlen(cart\_key)

if nickname:

user.nickname = nickname.NickName

else:

user.nickname = ''

# 组织模板上下文

context.update(cart\_count=cart\_count)

# 使用模板

return render(request, 'index.html', context)

''' 启动nginx

需要切换到/usr/local/nginx/sbin

sudo ./nginx -c /usr/local/nginx/conf/nginx.conf

'''

# 商品详情

class Detail(LoginRequiredMixin, View):

'''商品详情页面'''

def get(self, request, goods\_id):

'''显示详情页'''

try:

sku = Goods.objects.get(id=goods\_id)

except Goods.DoesNotExist:

# 商品不存在

return redirect(reverse('system:index'))

# 获取商品的分类信息

brand = Brand.objects.get(id=sku.brandId)

types = Brand.objects.all()

# 获取用户购物车中商品的数目

# 获取新品信息

user = request.user

cart\_count = 0

if user.is\_authenticated():

# 用户已登录

conn = get\_redis\_connection('default')

cart\_key = 'cart\_%d' % user.id

cart\_count = conn.hlen(cart\_key)

# 添加用户的历史记录

conn = get\_redis\_connection('default')

history\_key = 'history\_%d' % user.id

# 移除列表中的goods\_id

conn.lrem(history\_key, 0, goods\_id)

# 把goods\_id插入到列表的左侧

conn.lpush(history\_key, goods\_id)

# 只保存用户最新浏览的5条信息

conn.ltrim(history\_key, 0, 4)

# 组织模板上下文

context = {'sku': sku, 'types': types,

"brand":brand,

'cart\_count': cart\_count}

# 使用模板

return render(request, 'detail.html', context)

# 商品列表

class ListView(View):

'''列表页'''

def get(self, request, type\_id, page):

'''显示列表页'''

# 获取种类信息

try:

type = Brand.objects.get(id=type\_id)

except Goods.DoesNotExist:

# 种类不存在

return redirect(reverse('system:index'))

# 获取商品的分类信息

types = Brand.objects.all()

# 获取排序的方式 # 获取分类商品的信息

# sort=default 按照默认id排序

# sort=price 按照商品价格排序

# sort=hot 按照商品销量排序

sort = request.GET.get('sort')

if sort == 'price':

skus = Goods.objects.filter(brandId=type.id).order\_by('price')

elif sort == 'hot':

skus = Goods.objects.filter(brandId=type.id).order\_by('-sales')

else:

sort = 'default'

skus = Goods.objects.filter(brandId=type.id).order\_by('-id')

# 对数据进行分页

paginator = Paginator(skus, 1)

# 获取第page页的内容

try:

page = int(page)

except Exception as e:

page = 1

if page > paginator.num\_pages:

page = 1

# 获取第page页的Page实例对象

skus\_page = paginator.page(page)

# 进行页码的控制，页面上最多显示5个页码

# 1.总页数小于5页，页面上显示所有页码

# 2.如果当前页是前3页，显示1-5页

# 3.如果当前页是后3页，显示后5页

# 4.其他情况，显示当前页的前2页，当前页，当前页的后2页

num\_pages = paginator.num\_pages

if num\_pages < 5:

pages = range(1, num\_pages + 1)

elif page <= 3:

pages = range(1, 6)

elif num\_pages - page <= 2:

pages = range(num\_pages - 4, num\_pages + 1)

else:

pages = range(page - 2, page + 3)

# 获取用户购物车中商品的数目

user = request.user

cart\_count = 0

if user.is\_authenticated():

# 用户已登录

conn = get\_redis\_connection('default')

cart\_key = 'cart\_%d' % user.id

cart\_count = conn.hlen(cart\_key)

# 组织模板上下文

context = {'type':type, 'types':types,

'skus\_page':skus\_page,

'cart\_count':cart\_count,

'pages': pages,

'sort':sort}

# 使用模板

return render(request, 'list.html', context)

# ajax发起的请求都在后台，在浏览器中看不到效果

# /add

class CartAddView(View):

'''购物车记录添加'''

def post(self, request):

'''购物车记录添加'''

user = request.user

if not user.is\_authenticated():

# 用户未登录

return JsonResponse({'res':0, 'errmsg':'请先登录'})

# 接收数据

sku\_id = request.POST.get('sku\_id')

count = request.POST.get('count')

# 数据校验

if not all([sku\_id, count]):

return JsonResponse({'res':1, 'errmsg':'数据不完整'})

# 校验添加的商品数量

try:

count = int(count)

except Exception as e:

# 数目出错

return JsonResponse({'res':2, 'errmsg':'商品数目出错'})

# 校验商品是否存在

try:

sku = Goods.objects.get(id=sku\_id)

except Goods.DoesNotExist:

# 商品不存在

return JsonResponse({'res':3, 'errmsg':'商品不存在'})

# 业务处理:添加购物车记录

# 将购物车信息记录到redis数据库中

conn = get\_redis\_connection('default')

cart\_key = 'cart\_%d'%user.id

# 先尝试获取sku\_id的值 -> hget cart\_key 属性

# 如果sku\_id在hash中不存在，hget返回None

cart\_count = conn.hget(cart\_key, sku\_id)

if cart\_count:

# 累加购物车中商品的数目

count += int(cart\_count)

# 校验商品的库存

if count > sku.goodsNum:

return JsonResponse({'res':4, 'errmsg':'商品库存不足'})

print(7)

# 设置hash中sku\_id对应的值

# hset->如果sku\_id已经存在，更新数据， 如果sku\_id不存在，添加数据

conn.hset(cart\_key, sku\_id, count)

# 计算用户购物车商品的条目数

total\_count = conn.hlen(cart\_key)

# 返回应答

return JsonResponse({'res':5, 'total\_count':total\_count, 'message':'添加成功'})

**5.4.3 租赁管理页面**

购物车页面由四个模块组成：CartInfoView类，CartUpdateView类，CartDeleteView类，OrderPlaceView类。

CartInfoView类：展现购物车页面

CartUpdateView类：用户任然可以在购物车页面中修改自己的租赁自行车数量，通过全段的js函数监测数据的格式，决定是否传入后台，后台则监测自行车相关数据监测。

CartUpdateView类：用户可以删除此订单。

OrderPlaceView类：当用户点击页面中锁定商品按钮时候进行操作。此模块采用了事务的乐观锁，有效的解决了事务并发问题，避免商家和用户的利益受损。当用户锁定商品后进入订单页面。租赁开始计费。

# /cart

class CartInfoView(LoginRequiredMixin, View):

'''购物车页面显示'''

def get(self, request):

'''显示'''

# 获取登录的用户

user = request.user

# 获取用户购物车中商品的信息

conn = get\_redis\_connection('default')

cart\_key = 'cart\_%d'%user.id

# {'商品id':商品数量}

cart\_dict = conn.hgetall(cart\_key)

skus = []

# 保存用户购物车中商品的总数目和总价格

total\_count = 0

total\_price = 0

# 遍历获取商品的信息

for sku\_id, count in cart\_dict.items():

# 根据商品的id获取商品的信息

sku = Goods.objects.get(id=sku\_id)

# 计算商品的小计

amount = sku.price\*int(count)

# 动态给sku对象增加一个属性amount, 保存商品的小计

sku.amount = amount

# 动态给sku对象增加一个属性count, 保存购物车中对应商品的数量

sku.count = count

# 添加

skus.append(sku)

# 累加计算商品的总数目和总价格

total\_count += int(count)

total\_price += amount

# 组织上下文

context = {'total\_count':total\_count,

'total\_price':total\_price,

'skus':skus}

# 使用模板

return render(request, 'cart.html', context)

class CartUpdateView(View):

'''购物车记录更新'''

def post(self, request):

'''购物车记录更新'''

user = request.user

if not user.is\_authenticated():

# 用户未登录

return JsonResponse({'res': 0, 'errmsg': '请先登录'})

# 接收数据

sku\_id = request.POST.get('sku\_id')

count = request.POST.get('count')

# 数据校验

if not all([sku\_id, count]):

return JsonResponse({'res': 1, 'errmsg': '数据不完整'})

# 校验添加的商品数量

try:

count = int(count)

except Exception as e:

# 数目出错

return JsonResponse({'res': 2, 'errmsg': '商品数目出错'})

# 校验商品是否存在

try:

sku = Goods.objects.get(id=sku\_id)

except Goods.DoesNotExist:

# 商品不存在

return JsonResponse({'res': 3, 'errmsg': '商品不存在'})

# 业务处理:更新购物车记录

conn = get\_redis\_connection('default')

cart\_key = 'cart\_%d'%user.id

# 校验商品的库存

if count > sku.goodsNum:

return JsonResponse({'res':4, 'errmsg':'商品库存不足'})

# 更新

conn.hset(cart\_key, sku\_id, count)

# 计算用户购物车中商品的总件数 {'1':5, '2':3}

total\_count = 0

vals = conn.hvals(cart\_key)

for val in vals:

total\_count += int(val)

# 返回应答

return JsonResponse({'res':5, 'total\_count':total\_count, 'message':'更新成功'})

# 删除购物车记录

# 采用ajax post请求

# 前端需要传递的参数:商品的id(sku\_id)

# /cart/delete

class CartDeleteView(View):

'''购物车记录删除'''

def post(self, request):

'''购物车记录删除'''

user = request.user

if not user.is\_authenticated():

# 用户未登录

return JsonResponse({'res': 0, 'errmsg': '请先登录'})

# 接收参数

sku\_id = request.POST.get('sku\_id')

# 数据的校验

if not sku\_id:

return JsonResponse({'res':1, 'errmsg':'无效的商品id'})

# 校验商品是否存在

try:

sku = Goods.objects.get(id=sku\_id)

except Goods.DoesNotExist:

# 商品不存在

return JsonResponse({'res':2, 'errmsg':'商品不存在'})

# 业务处理:删除购物车记录

conn = get\_redis\_connection('default')

cart\_key = 'cart\_%d'%user.id

# 删除 hdel

conn.hdel(cart\_key, sku\_id)

# 计算用户购物车中商品的总件数 {'1':5, '2':3}

total\_count = 0

vals = conn.hvals(cart\_key)

for val in vals:

total\_count += int(val)

# 返回应答

return JsonResponse({'res':3, 'total\_count':total\_count, 'message':'删除成功'})

# /place

class OrderPlaceView(LoginRequiredMixin, View):

'''锁定商品'''

@transaction.atomic

def post(self, request):

'''提交订单页面显示'''

# 获取登录的用户

user = request.user

# 获取参数sku\_ids

sku\_ids = request.POST.getlist('sku\_ids') # [1,26]

# 校验参数

if not sku\_ids:

# 跳转到购物车页面

return redirect(reverse('system:cart'))

print(sku\_ids)

# 保存商品的总件数和总价格

total\_count = 0

total\_price = 0

save\_id = transaction.savepoint()

try:

# todo: 向df\_order\_info表中添加一条记录

order = Order.objects.create(userId=user.id,

goodsNum=total\_count,

amount=total\_price,

payMethod= 0,

)

# todo: 用户的订单中有几个商品，需要向df\_order\_goods表中加入几条记录

conn = get\_redis\_connection('default')

cart\_key = 'cart\_%d' % user.id

# sku\_ids = sku\_ids.split(',')

for sku\_id in sku\_ids:

print(type(sku\_id))

for i in range(3):

# 获取商品的信息

try:

sku = Goods.objects.get(id=sku\_id)

except:

# 商品不存在

transaction.savepoint\_rollback(save\_id)

return JsonResponse({'res': 4, 'errmsg': '商品不存在'})

# 从redis中获取用户所要购买的商品的数量

count = conn.hget(cart\_key, sku\_id)

# todo: 判断商品的库存

if int(count) > sku.goodsNum:

transaction.savepoint\_rollback(save\_id)

return JsonResponse({'res': 6, 'errmsg': '商品库存不足'})

# todo: 向df\_order\_goods表中添加一条记录

print('ok')

OrderInfo.objects.create(orderId=order.id,

skuId=sku\_id,

skuNum=count,)

print('bad')

# todo: 更新商品的库存和销量

orgin\_stock = sku.goodsNum

new\_stock = orgin\_stock - int(count)

new\_sales = sku.sales + int(count)

# 返回受影响的行数

res = Goods.objects.filter(id=sku\_id, goodsNum=orgin\_stock).update(goodsNum=new\_stock, sales=new\_sales)

if res == 0:

if i == 2:

# 尝试的第3次

transaction.savepoint\_rollback(save\_id)

return JsonResponse({'res': 7, 'errmsg': 2})

continue

# todo: 累加计算订单商品的总数量和总价格

amount = sku.price \* int(count)

total\_count += int(count)

total\_price += amount

# 跳出循环

break

# todo: 更新订单信息表中的商品的总数量和总价格

# 再订单表新增商品总单价以及商品总数

order.goodsNum = total\_count

order.amount = total\_price

order.save()

except Exception as e:

transaction.savepoint\_rollback(save\_id)

return JsonResponse({'res': 9, 'errmsg': '下单失败'})

# 提交事务

transaction.savepoint\_commit(save\_id)

# todo: 清除用户购物车中对应的记录

conn.hdel(cart\_key, \*sku\_ids)

# 使用模板

# 组织上下文

order\_id = order.id

context = {

'order':order,

'total\_count': total\_count,

'total\_price': total\_price,

'order\_id': order\_id}

return render(request, 'take\_order.html', context)

# 前端传递的参数:地址id(addr\_id) 支付方式(pay\_method) 用户要购买的商品id字符串(sku\_ids)

# mysql事务: 一组sql操作，要么都成功，要么都失败

# 高并发:秒杀

# 订单页面生成

**5.4.4 租赁管理页面**

模块由两个函数组成：OrderShow类，OrderCommitView类。

OrderShow类：将订单详情显示给用户

OrderCommitView类：结算商品，通过小时的计费，计算价格。将支付成功提示展现给商家后，商家通过管理员登陆管理员页面，确认订单。

class OrderShow(View):

'''订单页面生成'''

def get(self, request, order\_id):

user = request.user

if not user.is\_authenticated():

return redirect(reverse('system:index'))

order = Order.objects.get(id=order\_id, userId=user.id)

context = {'order':order}

# datetime.strftime('%H')将时间按想要的时间段取出

print(order.time.strftime('%H'))

return render(request, 'take\_order.html', context)

class OrderCommitView(View):

'''订单创建'''

def post(self,request):

user = request.user

if not user.is\_authenticated():

return redirect(reverse('system:index'))

order\_id = request.POST.get('order\_id')

order\_method = request.POST.get('pay\_method')

now = datetime.now().strftime('%H')

# 判断支付方式

paymethod = ''

if order\_method == '1':

paymethod = '微信支付'

elif order\_method == '2':

paymethod ='支付宝'

else:

paymethod = '银行卡支付'

print(paymethod)

order = Order.objects.get(userId=user.id, id=order\_id)

order.isOver=True

order\_time = Order.objects.get(userId=user.id, id=order\_id).time.strftime('%H')

# thetime = order\_time.time.strftime('%H')

# 将时间相减乘以单价得出支付金额

time = int(now) - int(order\_time)

money = int(order.amount) \* time

# 将库存更新

# 将该订单表中的所有商品id以及商品购买数量取出,形成更新

Infos = OrderInfo.objects.filter(orderId = order.id)

for info in Infos:

skuId = info.skuId

goods = Goods.objects.get(id=skuId)

skuNum = info.skuNum+goods.goodsNum

goods.goodsNum = skuNum

print('ok')

order.save()

print(2)

return JsonResponse({'res': 5, 'paymethod': paymethod, 'money': money})

**第六章．总结**

这个系统的开发花费了我近四个月的时间，在项目的最开始一直在苦恼整个系统的架构。这让我不得不承认，程序开发最难的一步就是系统设计了。从系统架构，到数据库设计，看似简单几步却让我寸步难行。可以说我80%的脑力都花费在了这个上面。而其中最让我感到困难的就是数据库的设计了，当我不断的完善系统功能时，就不断发现自己数据库设计的漏洞所在。从最开始简简单单的三张数据表，在系统的一步一步编写下扩增到了七张数据表。虽然困难，但是却也乐在其中，这也算是苦中作乐吧。同时这也明确了我自己对未来职业的一个规划，能在公司中做一名架构师。

在开发过程中，也并非自己想象中那般的一帆风顺。毕业设计是自己一个人的征途，系统的前后端都得靠自己一手包并。前端的页面设计让我头疼不已，前端的js数据监测以及和后端数据交互也是烧死我不少脑细胞，而最难的是html的css格式设计。我对于前端方面的知识并没有后端那么熟练，所以这个设计对我来说实在是有很大的挑战。但所幸在老师的帮助下还是勉勉强强设计出来了。

这次毕业设计也让我深刻了解到了时间是检验真理的唯一标准。在不断实践毕业设计时，我也不断了解到了自己的知识点的漏洞所在。对于完善自身专业能力，这无疑是极大的帮助。

经过几个月的努力下，成品也总算是出来了。虽然累，但是我也有着同等的收获，人生也是如此，唯有付出才有回报。然后我也非常感谢老师以及同学帮助，在他们的帮助下我也是少走了很多的弯路。所以最后，衷心感谢我的老师和同学。

**第七章．参考资料**

[1] 王春玲，徐瑶. 公用自行车租赁计费系统设计[J].科技视界.2017.9：36-37.

[2] Python学习手册 (美国)[Mark Lutz](https://baike.baidu.com/item/Mark Lutz" \t "https://baike.baidu.com/item/Python%E5%AD%A6%E4%B9%A0%E6%89%8B%E5%86%8C/_blank) . 出版时间 2009年08月.

[3] 资源博客Mybridge AI 值得收藏的45个python优质资源 云栖社区 2017.

[4] 知新文化. HTML 完全手册与速查辞典[M].北京：科学出版社，2007.

[5] 温谦. HTML+CSS网页设计与布局从入门到精通[M].人民邮电出版社, 2008.

[6] HTML/CSS/JavaScript标准教程实例版第3版 电子工业出版社，2011.

[7] 杜波依斯. MySQL技术内幕[M] 第4版 人民邮电出版社，2011.

[8] 唐汉明,翟振兴,关宝军,王洪权,黄潇,深入浅出MySQL[M],人民邮电出版社,2014.

[9] 董付国 Python程序设计 清华大学出版社，2015.

[10] 资源博主 山哥在这里 Python系列干货之Python与设计模式 云栖社区 2018

**第八章．致谢**

在最后我要真诚的感谢老师在我毕业设计期间的帮助。因为现在python火热让我感觉到python的市场，希望在毕业后找到一个python开发的工作，在新市场中就业，所以我选择了python作为我毕业设计的开发语言。但是python是我并未在大学接触过的语言，自然而然会遇到很多问题，当我遇到困难的时候是丁卓平老师的建议以及提点让我顺利走出了困境，让我能够在首次接触python的时候可以顺利的使用它完成一个系统的设计。再次感谢丁卓平老师对我的帮助，感谢！