**武汉大学计算机学院**

**本科生实验报告**

**软件设计与体系结构**

专 业 名 称 ：计算机科学与技术

课 程 名 称 ：软件设计与体系结构

指 导 教 师 一：王健

团 队 成 员 一：方正(2018302110260)

团 队 成 员 二：晋硕(2018302110280)

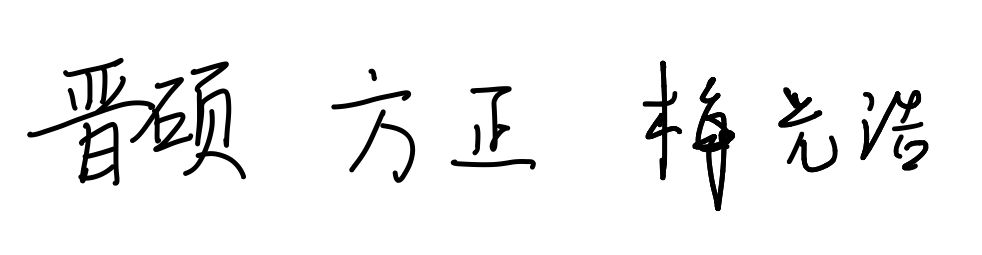
团 队 成 员 三：梅光浩(2018305231098)

二○二○年十一月

**郑 重 声 明**

本团队呈交的实验报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本实验报告不包含他人享有著作权的内容。对本实验报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本实验报告的知识产权归属于培养单位。

团队成员签名：



日期： 2020.11.20

摘 要

软件设计与体系结构课程设计实验的目的是用任意的编程语言实现一个SOAP或REST API并能用客户端进行调用。本实验在Django框架的基础上进行后端的编写同时实现了发布REST API，可以采用postman发送请求返回结果。前端部分采用Augular框架进行编写，与后端整合。

本实验基本实现了预计的商品预览，购物车，订单功能。在商品页面，能够将商品加入到购物车和查看商品详情，在购物车界面能够进行商品的增删改查，在订单界面能够查看总价并且跳转。

**关键词：**电子商城；Django；Augular；RESTful API

目录

[一、 实现功能 5](#_Toc56792014)

[1.1 选购商品 5](#_Toc56792015)

[1.2 对购物车中的商品进行修改 5](#_Toc56792016)

[1.3 提交订单 5](#_Toc56792017)

[二、 实现方法 6](#_Toc56792018)

[2.1 Angular 6](#_Toc56792019)

[2.2 Django 7](#_Toc56792020)

[2.3 RESTful API 9](#_Toc56792021)

[2.4 API接口 11](#_Toc56792022)

[2.5 Django路径实现 22](#_Toc56792023)

[三、 实现效果 24](#_Toc56792024)

[3.1 商品页面 24](#_Toc56792025)

[3.2 购物车页面 24](#_Toc56792026)

[3.3 订单页面 26](#_Toc56792027)

[3.4 全图 27](#_Toc56792028)

# 实现功能

## 选购商品

1. 在商品页面可以看到一系列商品信息
2. 通过添加按钮可以将商品加入购物车

## 对购物车中的商品进行修改

1. 在购物车页面可以添加某一个商品的数量
2. 在购物车页面可以删除某一个商品的数量
3. 在购物车页面可以删除某一个商品
4. 在购物车页面可以一键删除所有商品
5. 在购物车页面可以显示所有商品的总价
6. 在购物车页面可以将商品加入订单

## 提交订单

1. 显示订单的商品信息
2. 进行支付

# 实现方法

## 2.1 Angular

前端采用基于JavaScript的Angular框架。

Angular 是一个用 HTML 和 TypeScript 构建客户端应用的平台与框架。 Angular 本身就是用 TypeScript 写成的。它将核心功能和可选功能作为一组 TypeScript 库进行实现，你可以把它们导入你的应用中。

Angular 的基本构造块是 NgModule，它为组件提供了编译的上下文环境。 NgModule 会把相关的代码收集到一些功能集中。Angular 应用就是由一组 NgModule 定义出的。 应用至少会有一个用于引导应用的根模块，通常还会有很多特性模块。组件定义视图。视图是一组可见的屏幕元素，Angular 可以根据你的程序逻辑和数据来选择和修改它们。每个应用都至少有一个根组件。组件使用服务。服务会提供那些与视图不直接相关的功能。服务提供者可以作为依赖被注入到组件中，这能让你的代码更加模块化、更加可复用、更加高效。

模块、组件和服务都是使用装饰器的类，装饰器会标出它们的类型并提供元数据，以告知 Angular 该如何使用它们。组件类的元数据将组件类和一个用来定义视图的模板关联起来。 模板把普通的 HTML 和 Angular 指令与绑定标记（markup）组合起来，这样 Angular 就可以在渲染 HTML 之前先修改这些 HTML。服务类的元数据提供了一些信息，Angular 要用这些信息来让组件可以通过依赖注入（DI）使用该服务。应用的组件通常会定义很多视图，并进行分级组织。Angular 提供了 Router 服务来帮助你定义视图之间的导航路径。 路由器提供了先进的浏览器内导航功能。

## 2.2 Django

后端采用基于Python的Django框架。

Django是一个由Python编写的开放源代码的Web应用框架。Django本身基于MVC模型，MVC模型时软件工程中的一种软件架构模式，把软件系统分为三个基本部分：Model（模型），View（视图），Controller（控制器）。MVC模式使后续对程序的修改和扩展简化，并且使程序某一部分的重复利用成为可能。MVC以一种插件式的、松耦合的方式连接在一起。

模型部分编写程序应有的功能，负责业务对象与数据库的映射。视图部分是图形界面，负责与用户的交互。控制器负责转发请求并对请求进行处理。

MVC模式有如下的优势：

1. 低耦合
2. 开发快捷
3. 部署方便
4. 可重用性高
5. 维护成本低

Django基于MVC模式提出了MTV模式。MTV模式本质上和MVC模式相同，也是为了保持各组件间的松耦合关系，但是定义上稍有不同。M表示模型（Model），编写程序应有的功能，负责业务对象与数据库的映射。T表示模板（Template），负责如何把页面（html）展示给用户。V表示视图（View），负责业务逻辑，并在适当的时刻调用Model和Template。除此之后，MVT模式还需要一个URL分发器，将一系列的URL页面请求分发给不同的View处理，View再调用相应的Model和Template。

用户通过浏览器对服务器发起请求，这个请求回去访问视图函数。如果不涉及数据调用，那么视图函数直接返回一个模板（也就是一个网页）给用户。如果涉及到数据调用，视图函数调用模型，模型前往数据库中查找数据然后逐级返回。视图函数最后把返回的数据填充到模板中然后返回网页给用户。

Django除此之外，还有如下的特点：

1. 强大的数据库功能
2. 自带强大的后台功能
3. 优雅的网址

## 2.3 RESTful API

Rest全称Representation State Transfer。REST是一种软件体系结构样式，它定义了一组用于创建Web服务的约束。符合REST体系结构样式的Web服务（称为RESTful Web服务）提供Internet上计算机系统之间的互操作性。RESTful Web服务允许请求系统通过使用统一且预定义的无状态操作集来访问和操纵Web资源的文本表示。其他类型的Web服务（例如SOAP Web服务）公开其自己的任意操作集。

“网络资源”首先在万维网上定义为由其URL标识的文档或文件。但是，如今，它们有了更加通用和抽象的定义，涵盖了可以以任何方式在Web上进行标识，命名，寻址，处理或执行的所有事物，实体或动作。在RESTful Web服务，对资源的的请求URI将在在格式化的HTML，XML，JSON或其它格式引发一个响应有效载荷。响应可以确认已经对资源状态进行了某些更改，并且响应可以提供到其他相关资源的超文本链接。可用的操作（HTTP方法）是GET，HEAD，POST，PUT，PATCH，DELETE，CONNECT，OPTIONS和TRACE。

REST有如下的设计原则和规范：

1. 资源。首先要明确资源就是网络上的一个实体，可以是文本、图片、音频、视频。资源总是以一定的格式来表现自己。文本用txt、html；图片用JPG、JPEG等等。而JSON是RESTful API中最常用的资源表现格式。
2. 统一接口。对于业务数据的CRUD，RESTful 用HTTP方法与之对应。

|  |  |
| --- | --- |
| CRUD | HTTP Methods |
| Create | POST |
| Read | GET |
| Update | PUT |
| Delete | DELETE |

1. URI。统一资源标识符，它可以唯一标识一个资源。注意到，URL(统一资源定位符)是一种URI，因为它可以唯一标志资源。但URL != URI。应该说URL 是URI的子集。因为URL使用路径来唯一标识资源，这只是唯一标识资源的一种方式。还可以用一个唯一编号来标识资源。
2. 无状态。 所谓无状态是指所有资源都可以用URI定位，而且这个定位与其他资源无关，不会因为其他资源的变动而变化。这里引入一个幂等性的概念：无论一个操作被执行一次还是多次，执行后的效果都相同。比如对某资源发送GET请求，如果访问一次和访问十次获得的数据一样，那么就说这个请求具有幂等性。
3. URL中只能有名词，不能出现动词。这是因为在REST要求对资源的操作由HTTP 方法给出，而方法是由HTTP 请求报文头部给出的，自然不需要在URL中暴露操作方式。

## 2.4 API接口

1. http://localhost:8000/Product/detail

功能：查询数据库，在商品页面获取所有商品的信息

代码：

def detail(request):

    id, num, name, price, type = searchAll()

    results = {

            'id': id,

            'name': name,

            'price': price,

            'num': num,

            'type': type

        }

    results\_json = json.dumps(results, indent=4, ensure\_ascii=False)

return HttpResponse(results\_json, content\_type="application/json,charset=utf-8")

def searchAll():

    mydb = SqldbHelper()

    id = []

    num = []

    name = []

    price = []

    type = []

    all\_types = ["id", "name", "单价", "品牌", "口味", "保质期", "质量", "尺寸", "颜色", "材料", "性别",

                 "衣领", "类型", "体积", "摄像头", "电池", "网络类型", "CPU", "RAM", "频率", "核",

                 "threads", "storage"]

    table = "cart"

    for i in mydb.select(table, fields=["id"]):

        id.append(str(i[0]))

    for i in mydb.select(table, fields=["num"]):

        num.append(str(i[0]))

    table = "goods"

    for i in id:

        name\_i = mydb.select(table, cond\_dict={"id": str(i)}, fields=["name"])

        name.append(name\_i[0][0])

    for i in id:

        types = mydb.select(table, cond\_dict={"id": str(i)})

        types = types[0]

        print(types)

        one\_type = ""

        for j in range(len(all\_types)):

            if j == 0:

                continue

            if j == 1:

                continue

            if j == 2:

                price.append(types[j])

            if types[j] == "NONE":

                continue

            one\_type = one\_type + ", " + all\_types[j] + ": " + str(types[j])

        one\_type = one\_type[2:]

        type.append(one\_type)

    return id, num, name, price, type

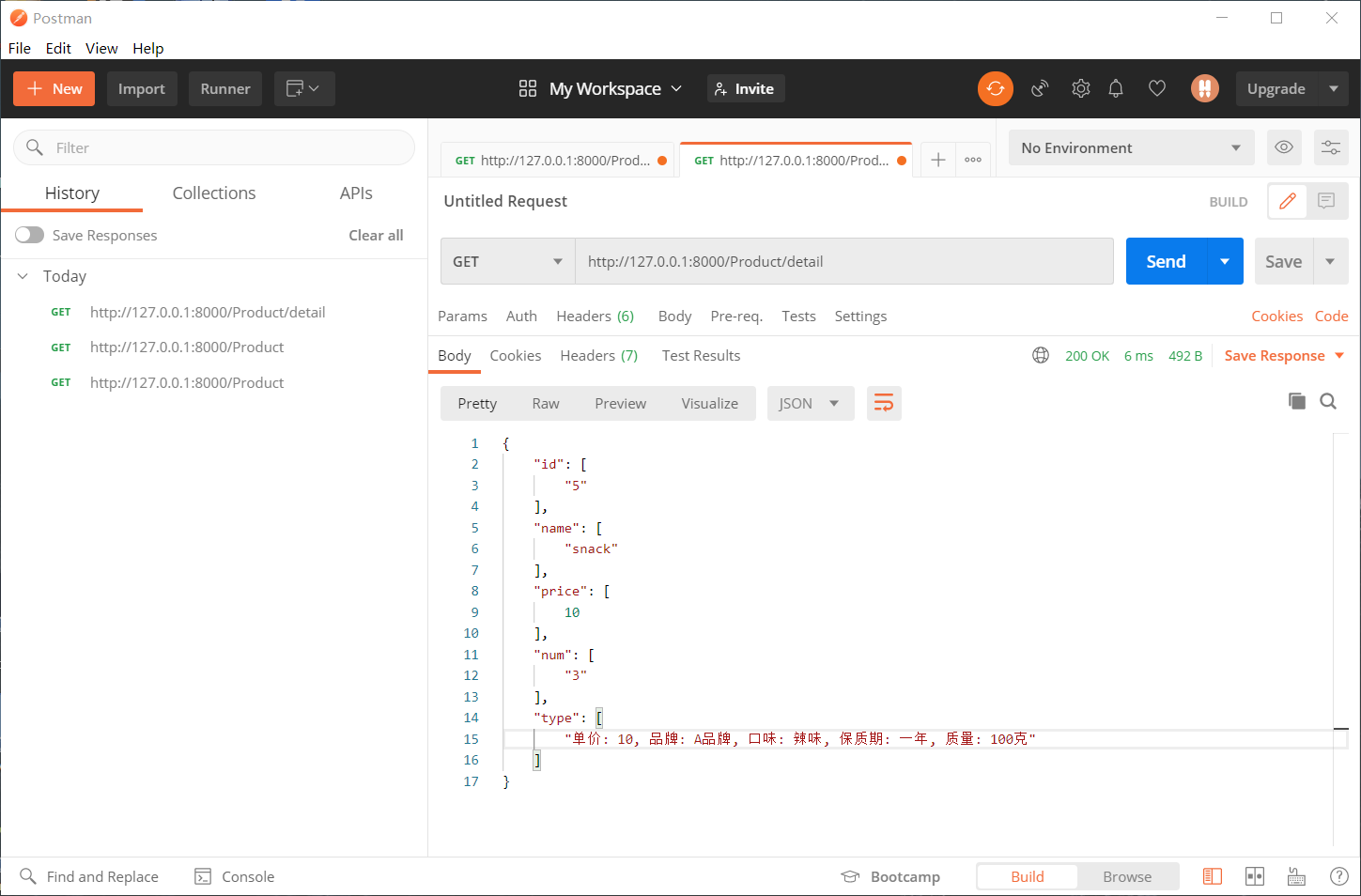


图2-4-1

1. http://localhost:8000/Product/<int:goodid>/add

功能：在商品页面添加某一商品进入购物车

def addToCart(request, goodid):

    mydb = SqldbHelper()

    table = "cart"

    cond\_dict = {"id": str(goodid)}

    if mydb.select(table, cond\_dict):

        # print(mydb.select(table, cond\_dict, fields=["num"]))  # [(1,)]

        num = mydb.select(table, cond\_dict, fields=["num"])

        num = int(num[0][0]) + 1

        params = {"id": str(goodid), "num": str(num)}

        mydb.update(table, params, cond\_dict)

    else:

        params = {"id": str(goodid), "num": "1"}

        mydb.insert(table, params)

    success = {

        'success': 1,

        'id':str(goodid)

    }

    success\_json = json.dumps(success, indent=4, ensure\_ascii=False)

    return HttpResponse(success\_json, content\_type="application/json,charset=utf-8")

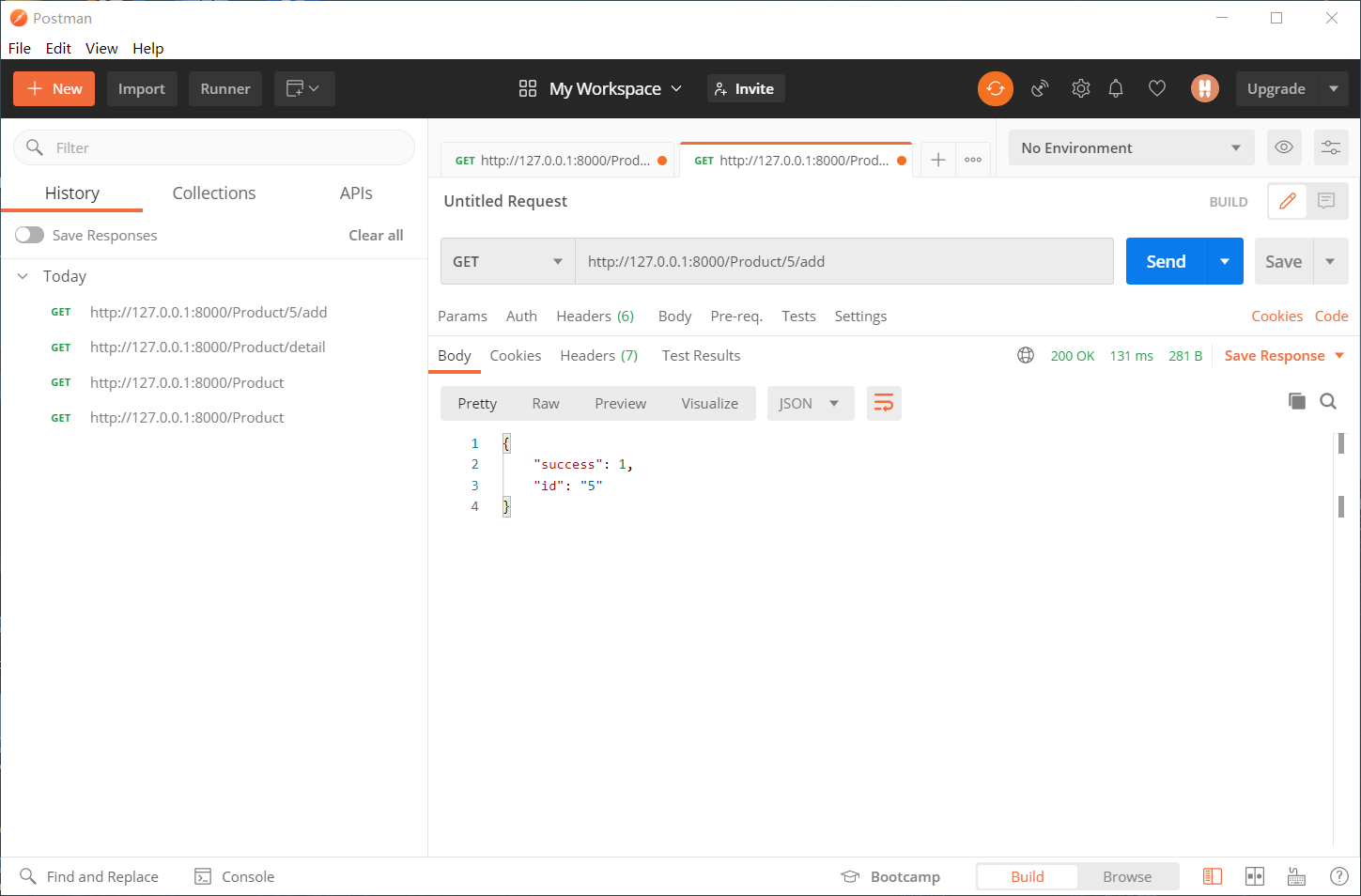


图2-4-2

1. http://localhost:8000/Cart/

功能：在购物车页面获取所有加入购物车的商品信息

def index(request):

    id, num, name, price, type = searchAll()

    results = {

            'id': id,

            'name': name,

            'price': price,

            'num': num,

            'type': type

        }

    results\_json = json.dumps(results, indent=4, ensure\_ascii=False)

    return HttpResponse(results\_json, content\_type="application/json,charset=utf-8")

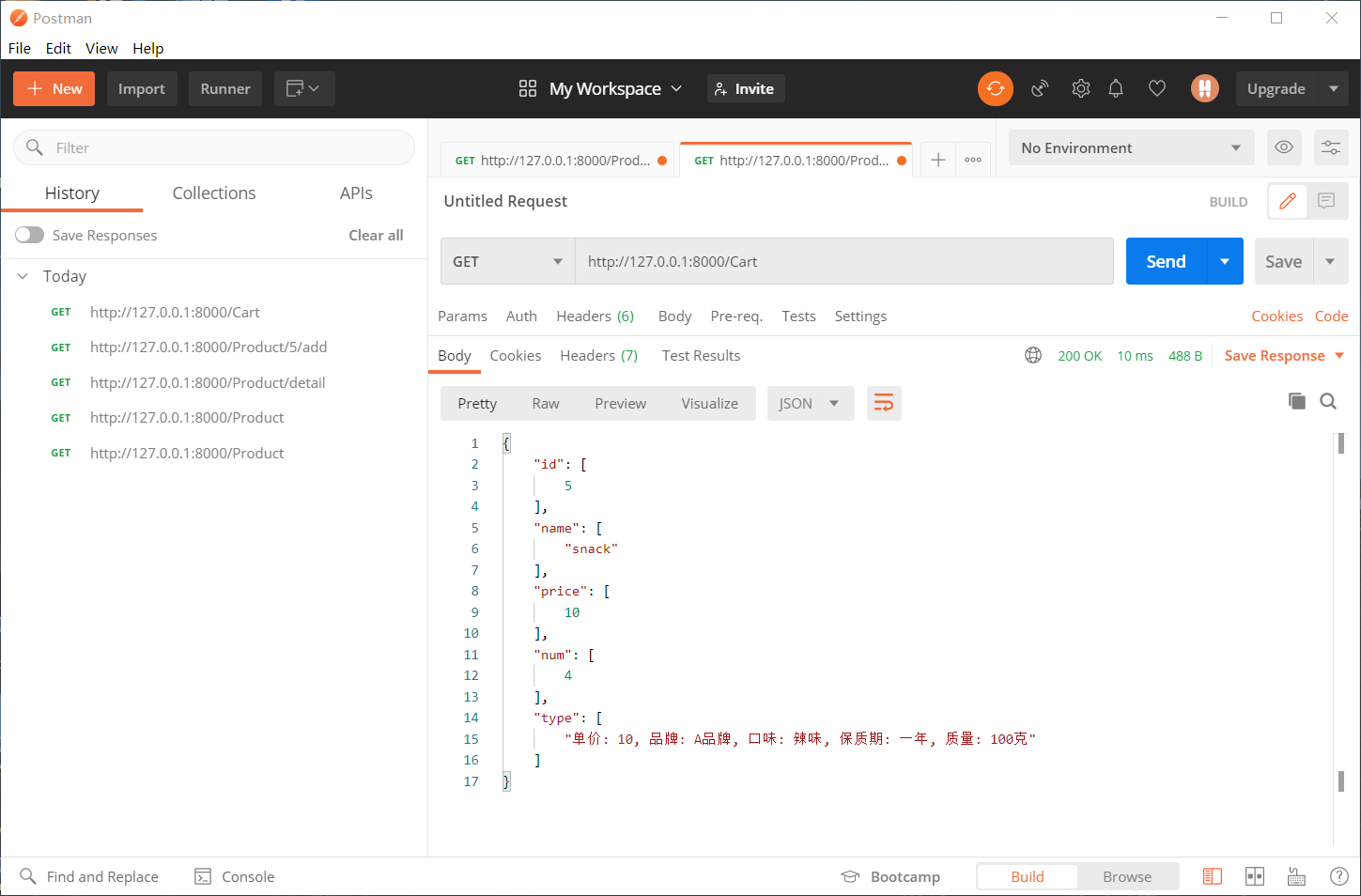


图2-4-3

1. http://localhost:8000/Cart/<int:goodid>/add

功能：在购物车中增加某一商品数量

def addToCart(request, goodid):

    mydb = SqldbHelper()

    table = "cart"

    cond\_dict = {"id": str(goodid)}

    if mydb.select(table, cond\_dict):

        num = mydb.select(table, cond\_dict, fields=["num"])

        num = int(num[0][0]) + 1

        params = {"id": str(goodid), "num": str(num)}

        mydb.update(table, params, cond\_dict)

    else:

        params = {"id": str(goodid), "num": "1"}

        mydb.insert(table, params)

    success = {

        'success': 1,

        'id':str(goodid)

    }

    success\_json = json.dumps(success, indent=4, ensure\_ascii=False)

    return HttpResponse(success\_json, content\_type="application/json,charset=utf-8")



图2-4-4

1. http://localhost:8000/Cart/<int:goodid>/delete

功能：在购物车中删除某一商品数量，若为0则从购物车中移除此商品

代码：

def deleteInCart(request, goodid):

    mydb = SqldbHelper()

    table = "cart"

    # params = {"id": str(id), "num": "1"}

    # mydb.insert(table, params)

    cond\_dict = {"id": str(goodid)}

    if mydb.select(table, cond\_dict):

        num = mydb.select(table, cond\_dict, fields=["num"])

        num = int(num[0][0])

        if num == 1:

            mydb.delete(table, cond\_dict)

        else:

            num = num - 1

            params = {"id": str(goodid), "num": str(num)}

            mydb.update(table, params, cond\_dict)

        sucess = {

            'sucess': 1,

            'id':str(goodid)

        }

        sucess\_json = json.dumps(sucess, indent=4, ensure\_ascii=False)

        return HttpResponse(sucess\_json,content\_type="application/json,charset=utf-8")

    else:

        sucess = {

            'sucess': 0

        }

        sucess\_json = json.dumps(sucess, indent=4)

        return HttpResponse(sucess\_json)

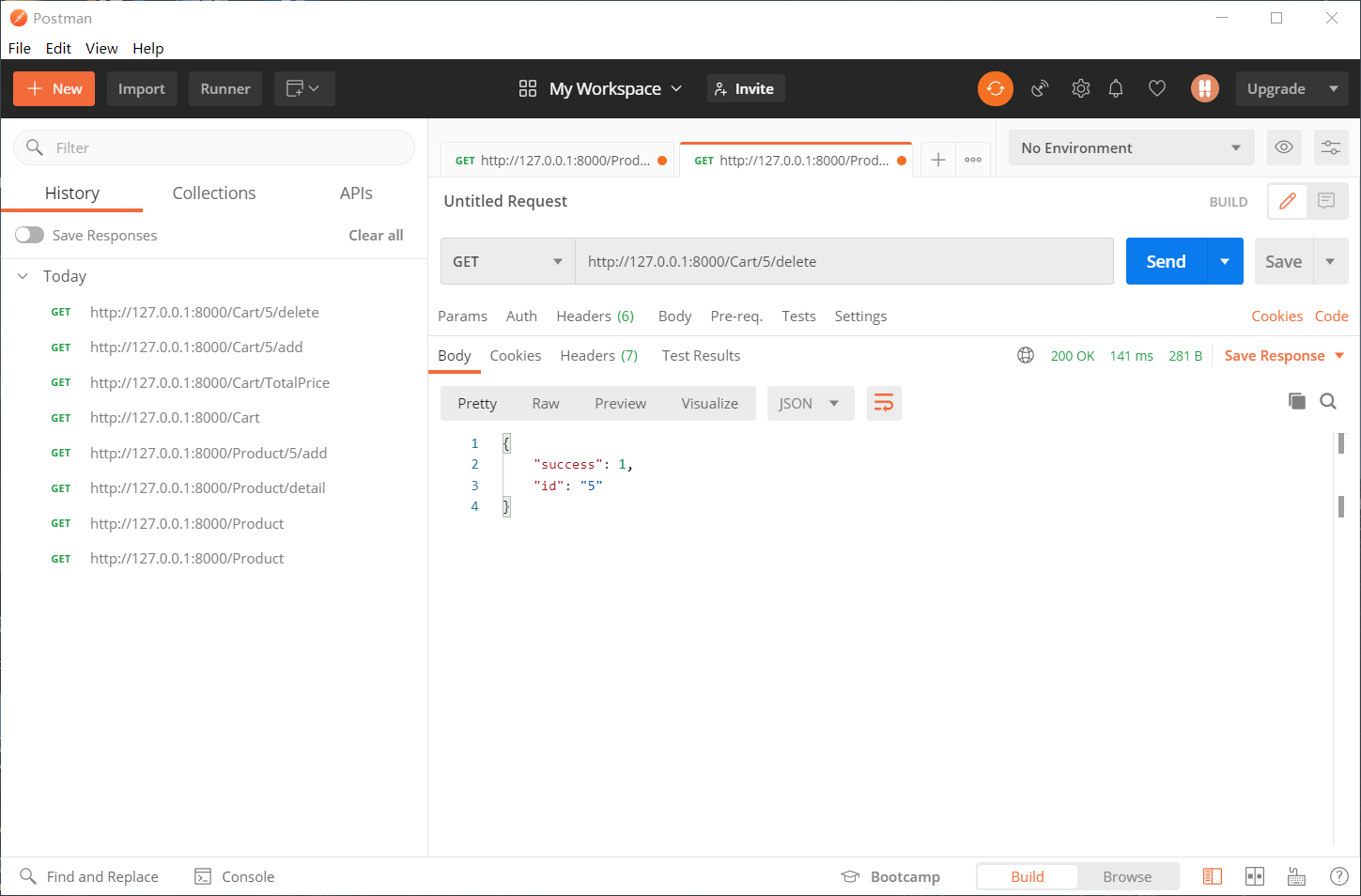


图2-4-5

1. http://localhost:8000/Cart/TotalPrice

功能：返回购物车中商品的总价

代码：

def getTotalPrice(request):

    \_, num, \_, price, \_ = searchAll()

    totalPrice = 0

    for i in range(0,len(num)):

        totalPrice += num[i] \* price[i]

    price = {

        'totalPrice' : totalPrice

    }

    price\_json = json.dumps(price, indent=4, ensure\_ascii=False)

    return HttpResponse(price\_json, content\_type="application/json,charset=utf-8")

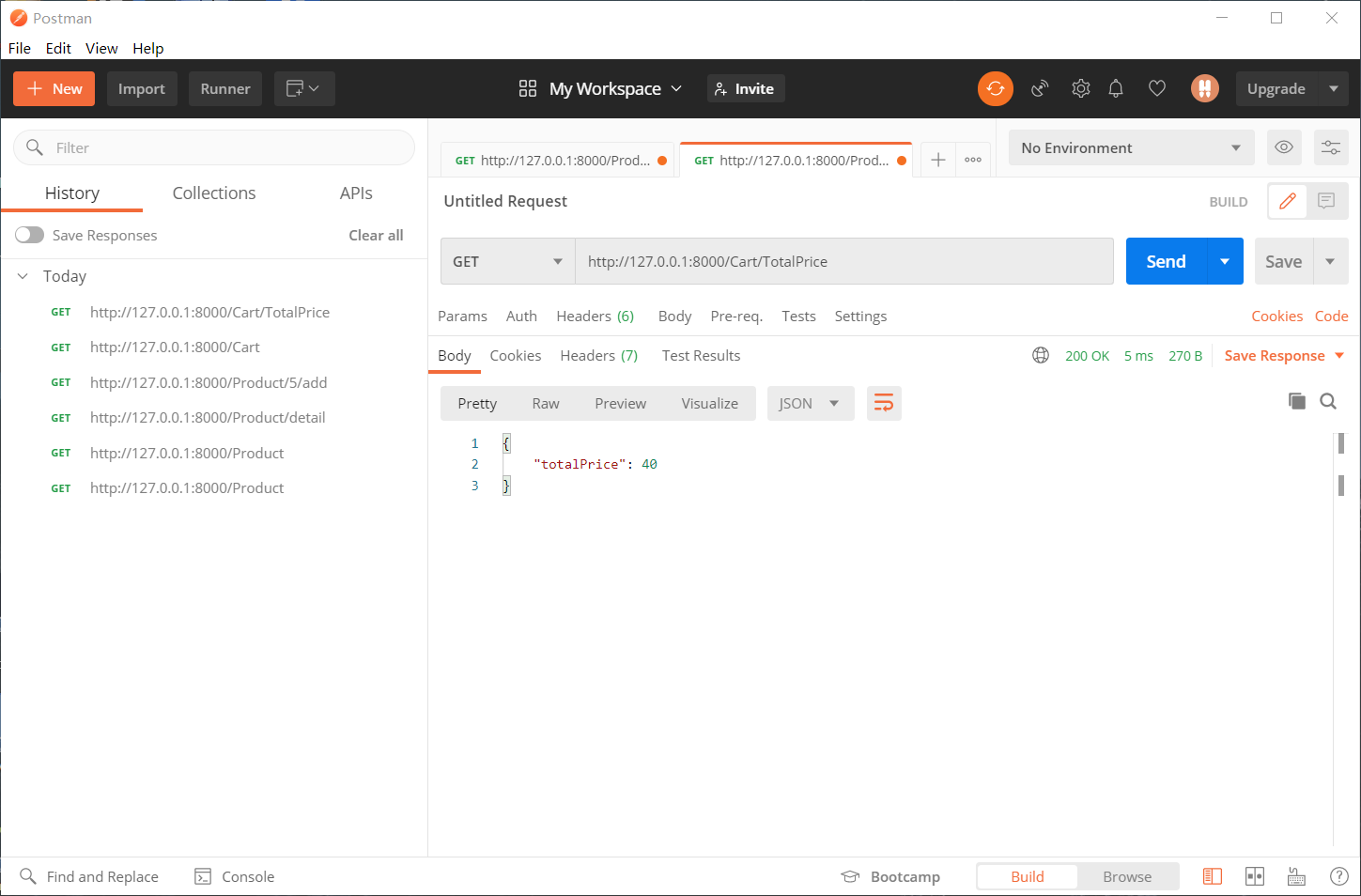


图2-4-6

1. http://localhost:8000/Cart/addToOrder

功能：将购物车中的商品加入订单

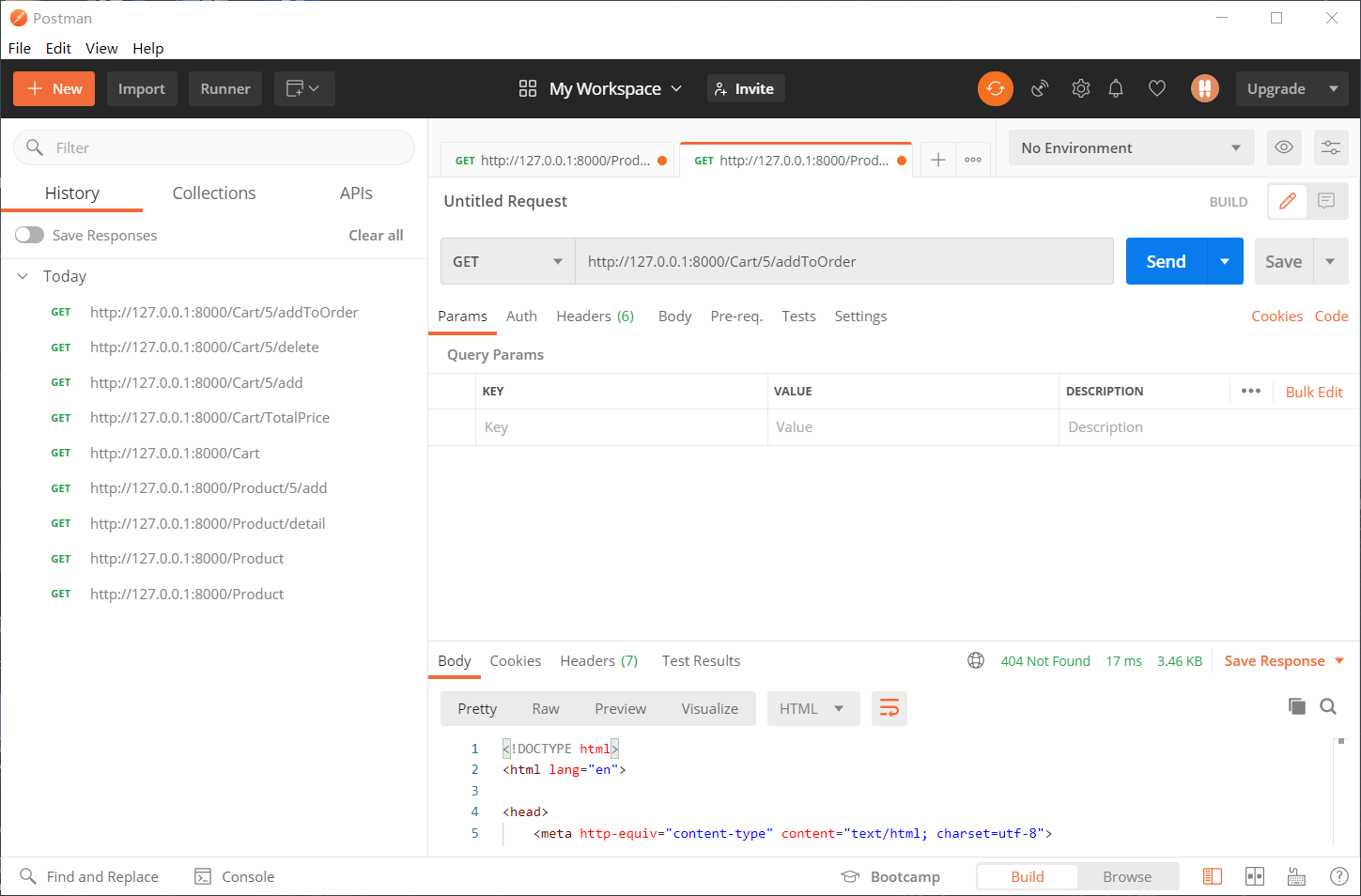
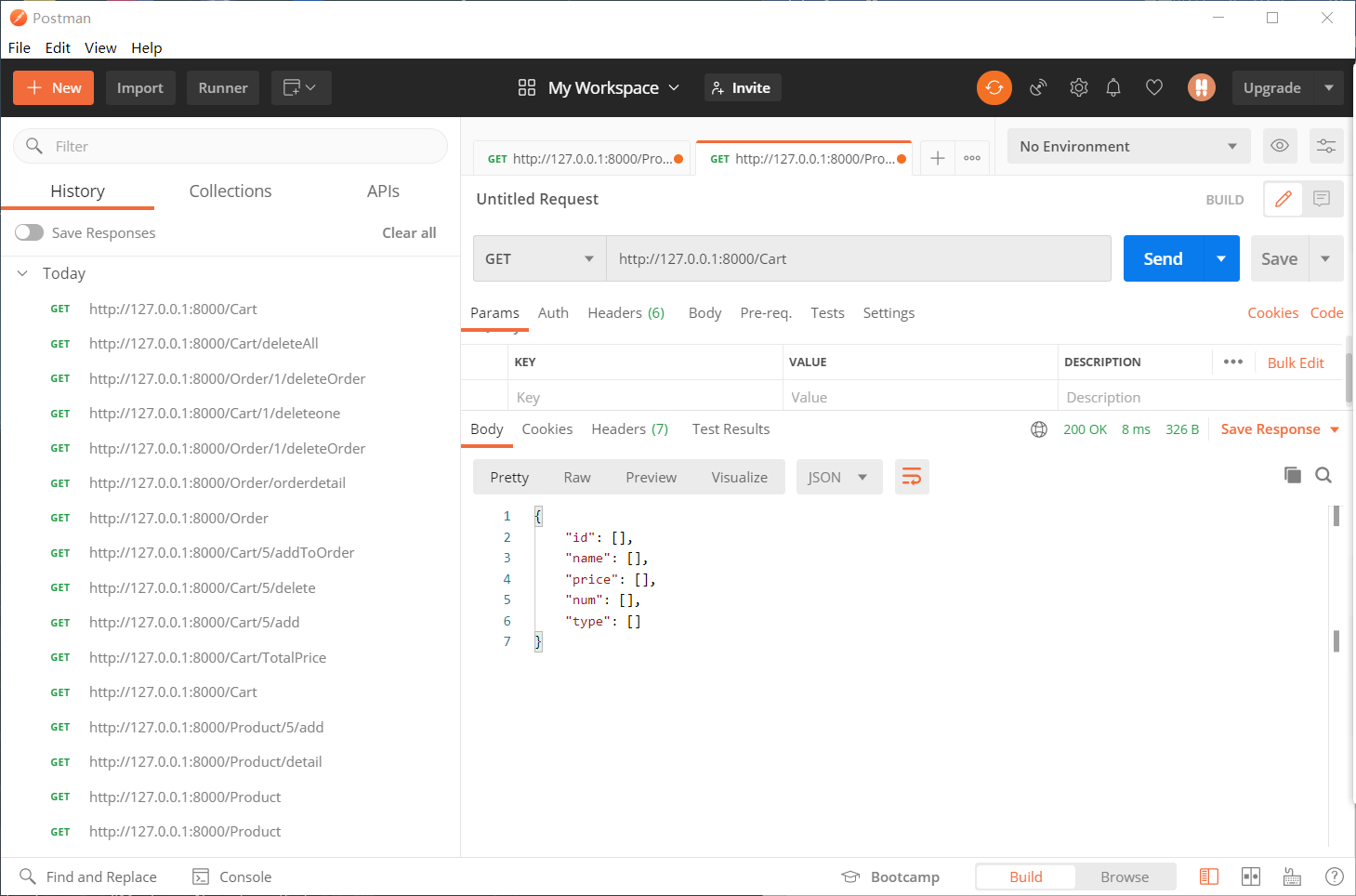
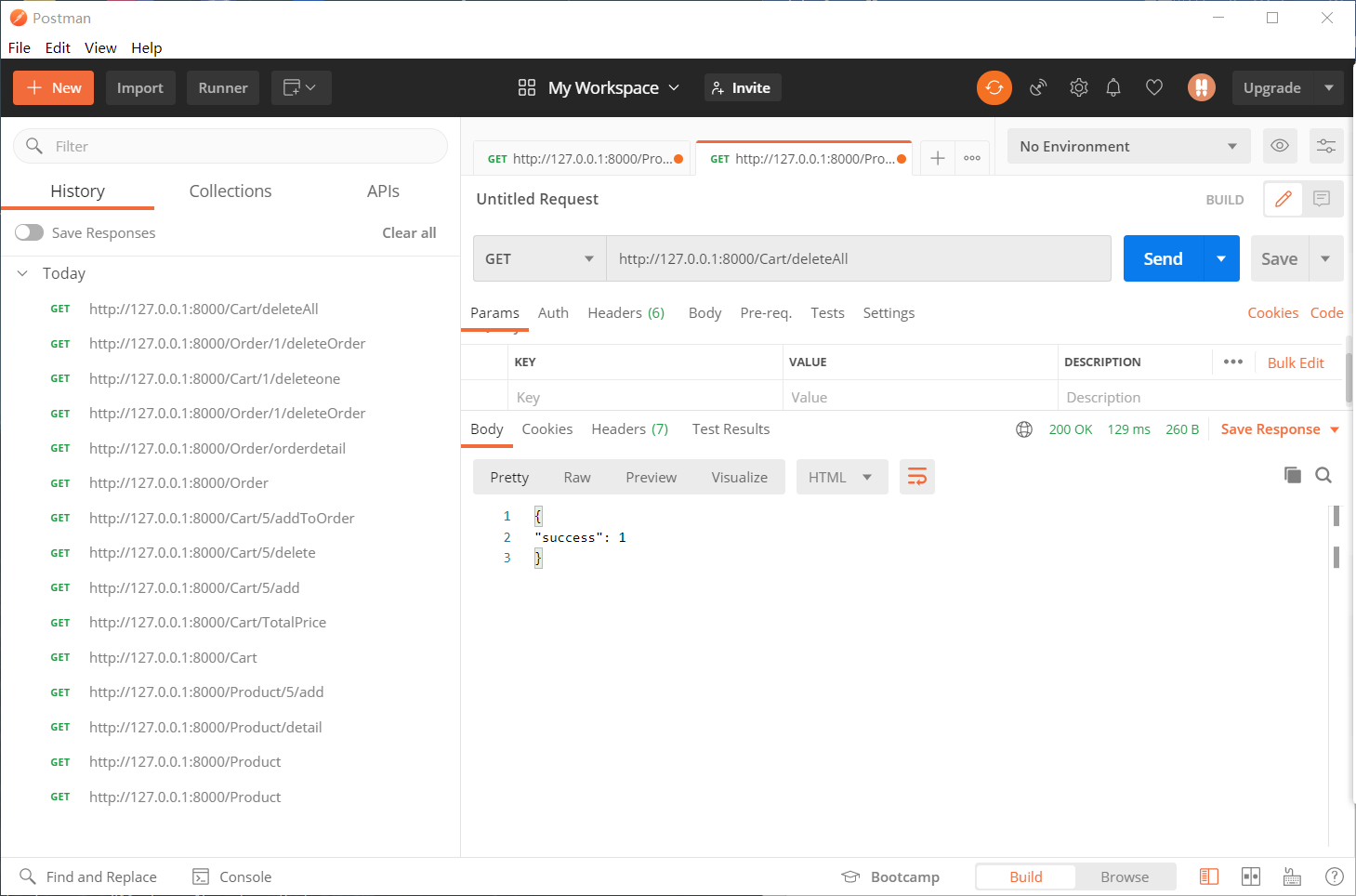


图2-4-7

1. http://localhost:8000/Cart/deleteAll

功能：删除购物车中所有商品



1. http://localhost:8000/Cart/deleteone

功能：删除购物车中某一个商品

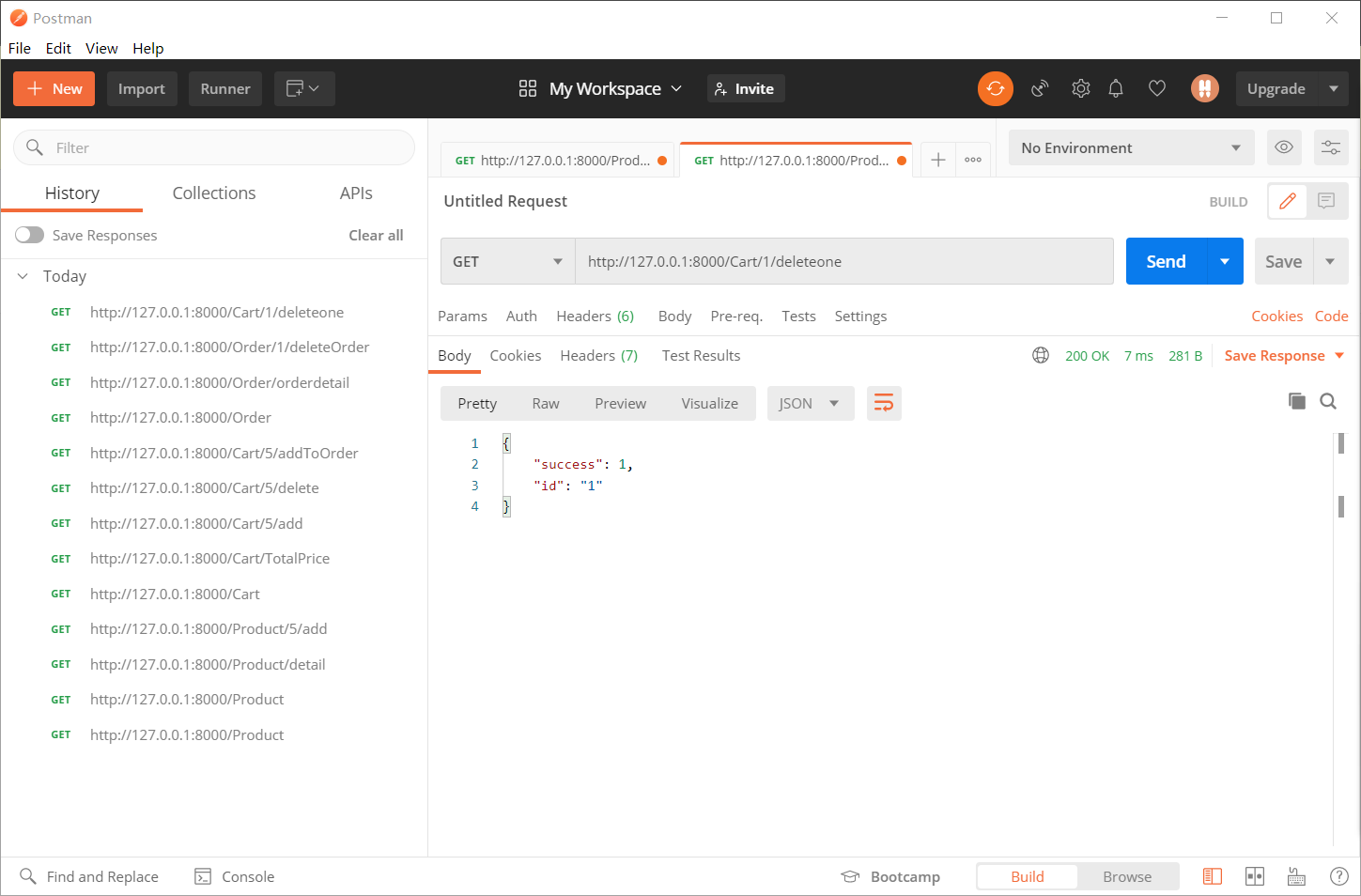


图2-4-8

1. http://localhost:8000/Order/orderdetail

功能：在订单页面返回订单中商品的相关信息

代码：

def getDetail(request):

    id, num, totalPrice, type = searchAll()

    detail = {

        'id': id,

        'num': num,

        'totalPrice': totalPrice,

        'detail': type

    }

    detail\_json = json.dumps(detail, indent=4, ensure\_ascii=False)

    return HttpResponse(detail\_json, content\_type="application/json,charset=utf-8")

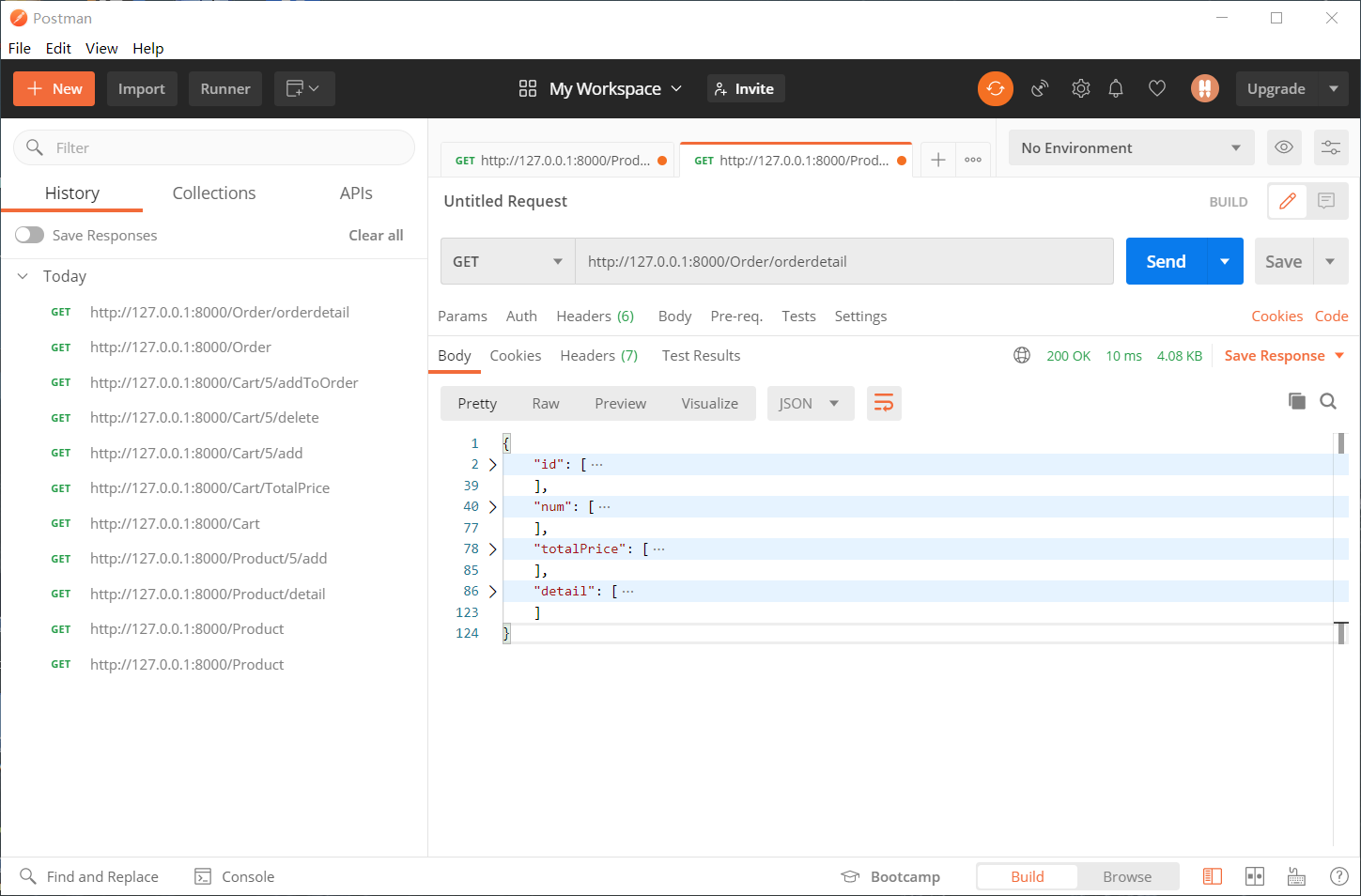


图2-4-9

1. http://localhost:8000/Order/payment

功能：成功支付订单之后调用此url然后重定向到商品页面

1. http://localhost:8000/Order/ <int:Orderid>/deleteOrder

功能：删除指定id的订单

## 2.5 Django路径实现

在Django中，我们先在项目文件中将不同应用的urlpatterns包含进来。从而根据不同的url进入不同应用的urls配置文件中进行进一步判断。

from django.contrib import admin

from django.urls import path, include

from Product import views

urlpatterns = [

    path('admin/', admin.site.urls),

    path('Order/',include('Order.urls')),

    path('Cart/',include('Cart.urls')),

    path('Product/',include('Product.urls')),

]

以Product/为例进行说明：

from django.urls import path

from . import views

app\_name = 'Product'

urlpatterns = [

    path('', views.index, name='index'),

    path('detail', views.detail, name='detail'),

    path('<int:goodid>/add', views.addToCart, name='addToCart')

]

从而能够将以Product/开头的url与Product这个应用中的函数一一关联起来。以Product/<int:goodid>/add为例进行说明，遇到这个url就会调用Product应用view文件中的addToCart函数：

def addToCart(request, goodid):

    mydb = SqldbHelper()

    table = "cart"

    cond\_dict = {"id": str(goodid)}

    if mydb.select(table, cond\_dict):

        # print(mydb.select(table, cond\_dict, fields=["num"]))  # [(1,)]

        num = mydb.select(table, cond\_dict, fields=["num"])

        num = int(num[0][0]) + 1

        params = {"id": str(goodid), "num": str(num)}

        mydb.update(table, params, cond\_dict)

    else:

        params = {"id": str(goodid), "num": "1"}

        mydb.insert(table, params)

    sucess = {

        'sucess': 1,

        'id':str(goodid)

    }

    sucess\_json = json.dumps(sucess, indent=4, ensure\_ascii=False)

    return HttpResponse(sucess\_json, content\_type="application/json,charset=utf-8")

# 实现效果

## 3.1 商品页面

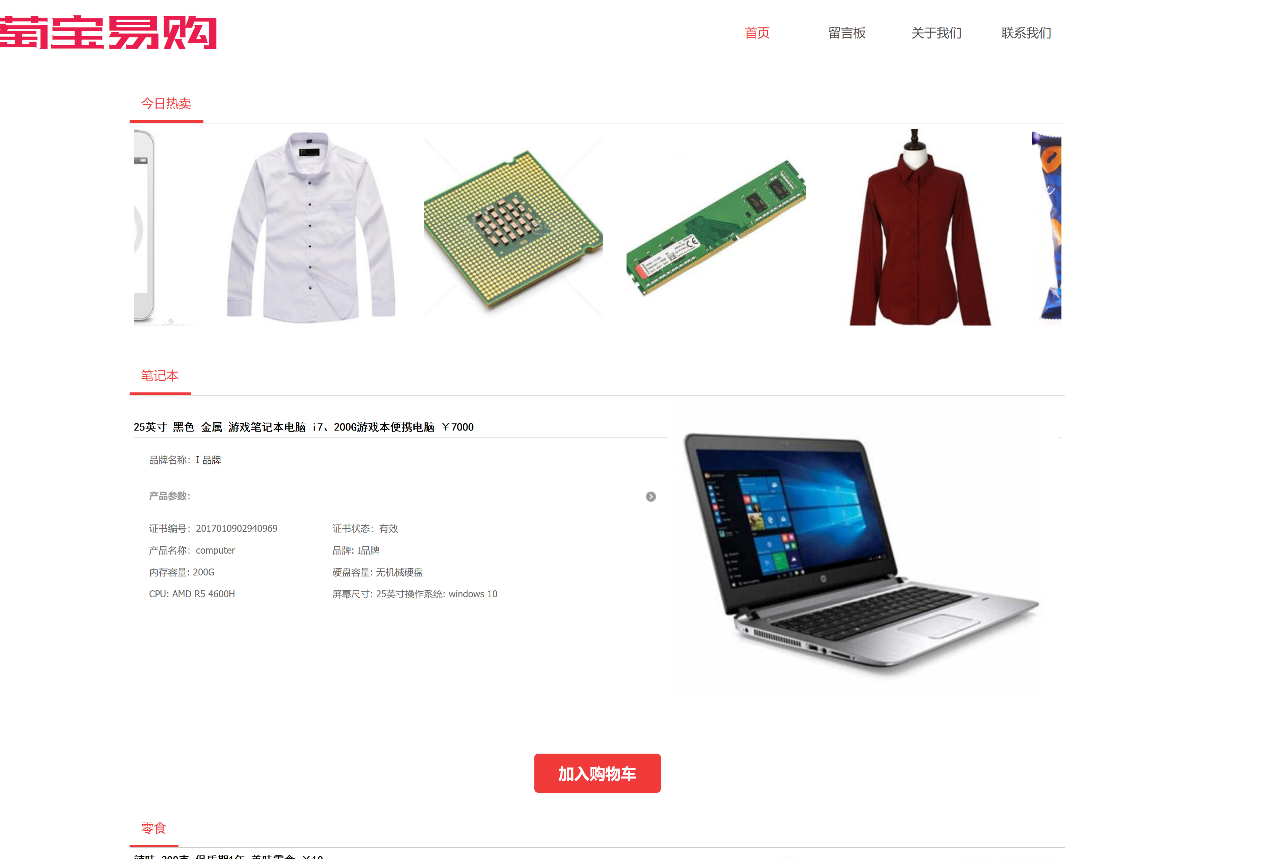


图3-1-1

## 3.2 购物车页面

1. 显示商品信息



图3-2-1

1. 增加数量



图3-2-2

1. 减少数量



图3-2-3

1. 删除商品

电脑商品被删除了，从而购物车中只剩下了五包零食。



图3-2-4

1. 显示总价和提交订单

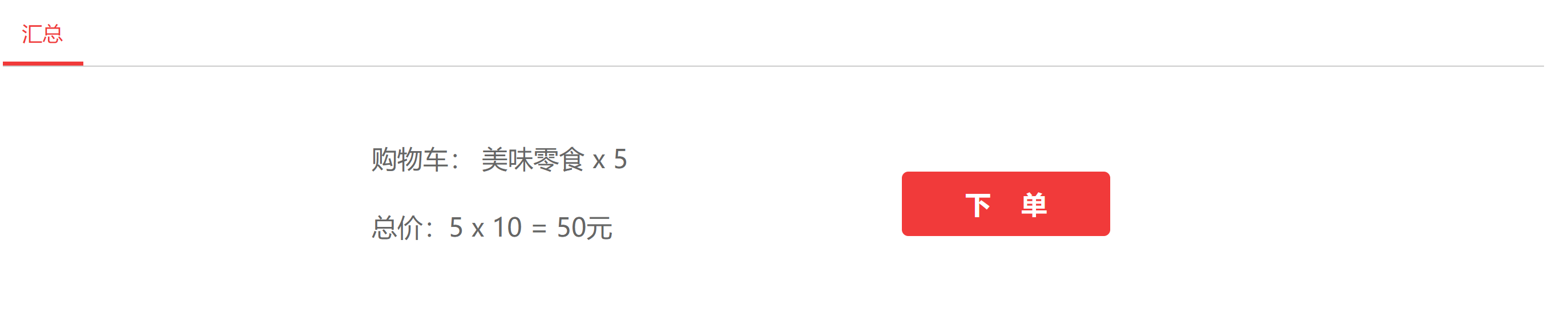


图3-2-5

## 3.3 订单页面

1. 显示商品信息

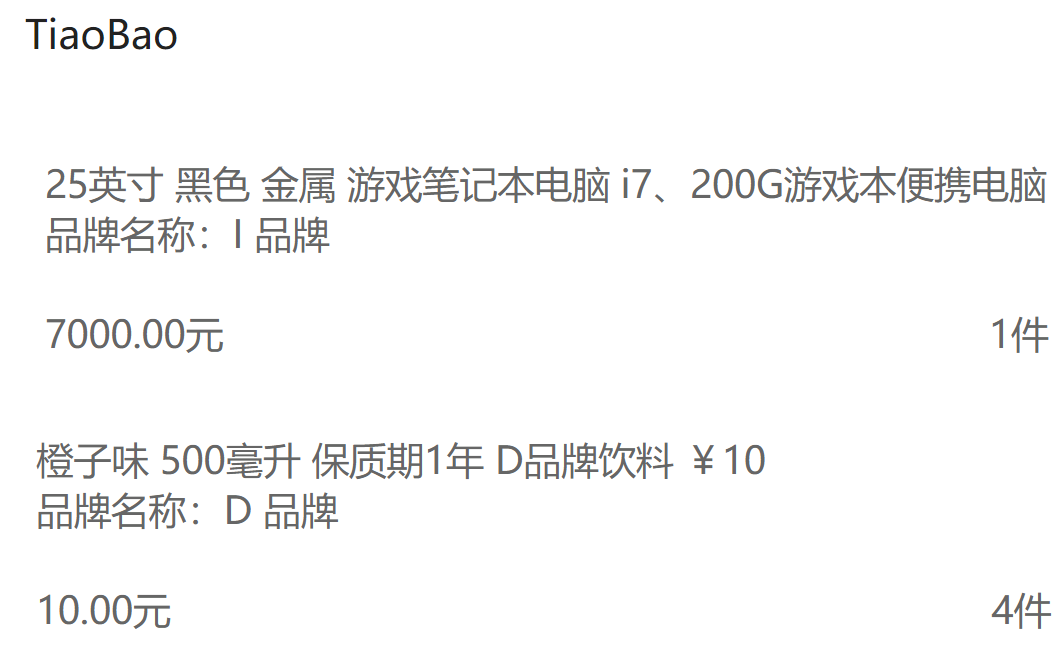


图3-3-1

1. 支付和显示商品总价



图3-3-2

## 3.4 全图

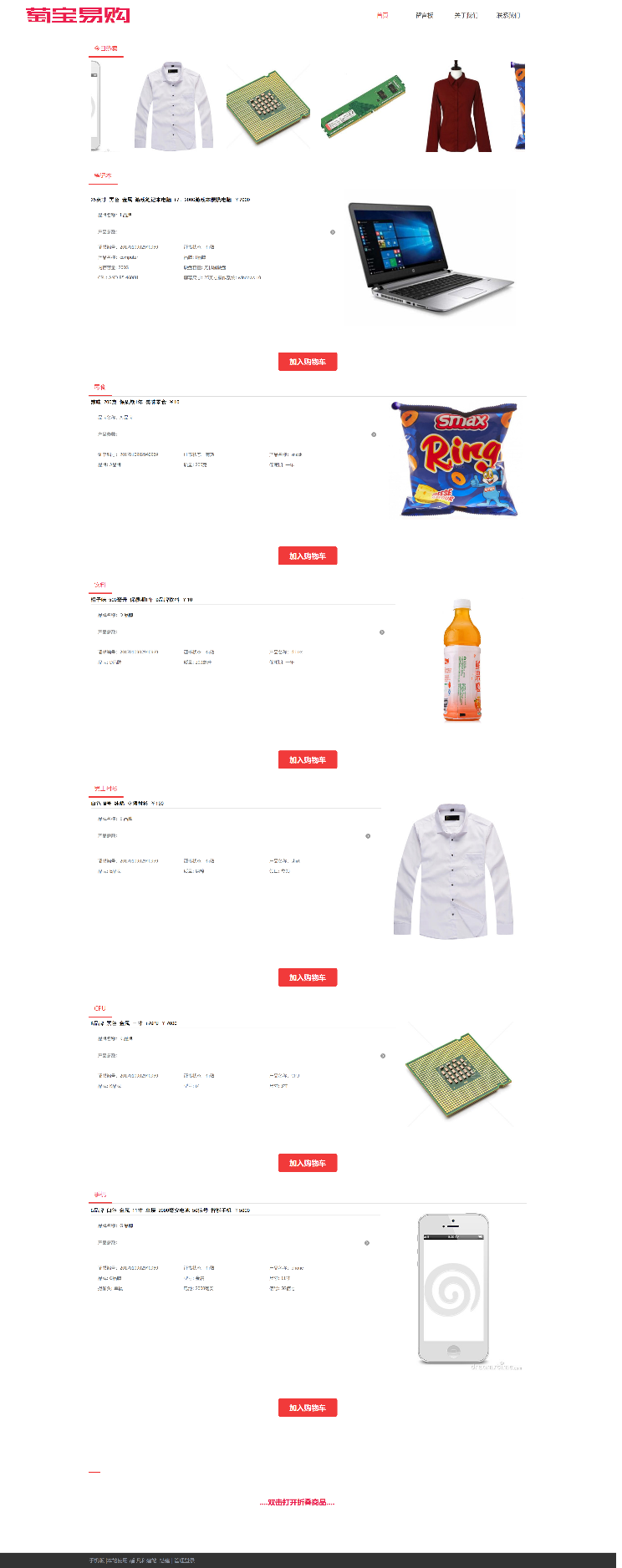


图3-4-1