目录

**[1． 项目设计目标及要求 1](#_Toc20374)**

**[2． 网站规划 2](#_Toc9872)**

[2.1 需求分析 2](#_Toc13741)

[2.2网站结构分析（站点目录） 2](#_Toc19748)

[2.3 网站风格分析 2](#_Toc30087)

[2.4 功能模块图 2](#_Toc14457)

**[3. 页面详细设计 2](#_Toc25092)**

[3.1 主页面设计 2](#_Toc23084)

[3.2 商品页面设计 2](#_Toc28553)

[3.3 品牌宣传页面设计 2](#_Toc24934)

[3.4 售后维护页面设计 2](#_Toc13219)

[3.5 用户登录页面设计 2](#_Toc19551)

[3.6 购物车板块页面设计 2](#_Toc32518)

[3.7 记事本页面设计 2](#_Toc12010)

**[4. 项目测试 2](#_Toc19556)**

**[5．设计总结及心得体会 2](#_Toc7802)**

**[6．参考文献 3](#_Toc21210)**

# 项目设计目标及要求

希望用响应式框架完成一个手机售卖平台，内容尽量贴切现代年轻人的审美，希望使得售后服务板块以及商品选择板块的体验性加强，因为商品效果的呈现，除了质量外，大部分是依靠营销以及用户持续的反馈来提升商品的市场地位。

# 网站规划

## 2.1 需求分析

我们在制作网站前首先要做的是要对网站内容进行一下具体的规划,确定一下制作的思路和目的,从而更方便我们的设计和制作。

创建新网站最佳方法是先建立原型系统,再进行详细设计,最后正式实施。

原型系统最主要的目标是学习、尝试采用不同的方法修改目标、更新形象。原型系统开发过程中要解决网站建设的一些基本问题。

例如：网站的结构;

网站整体风格的构思和素材的搜集;

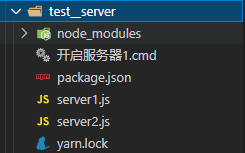
主页的制作;

* 存储信息的物理方法(采用数据库还是文件系统);
* 结构的完整性和一致性的维护方法。

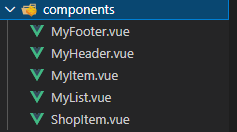
详细设计包括画面的布局、系统的内部结构、实现方法和维护方法等。

## 2.2网站结构分析

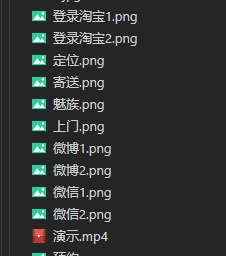
1、商品获取数据的测试服务器（项目开启前需手动执行下cmd，开启服务器）



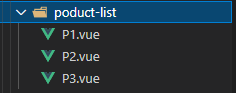
1. 实体组件（其中包括记事本模块以及购物车列表模块）



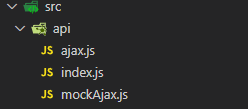
1. 项目存放的部分图片



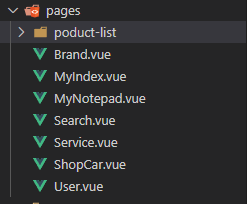
1. 商品详情的子组件



1. Api拦截器



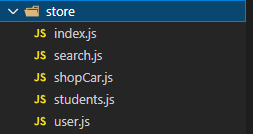
1. 各大模块路由组件



1. 路由表



1. Vuex仓库



1. 父组件



## 2.3 网站风格分析

简约风

## 2.4 功能模块图



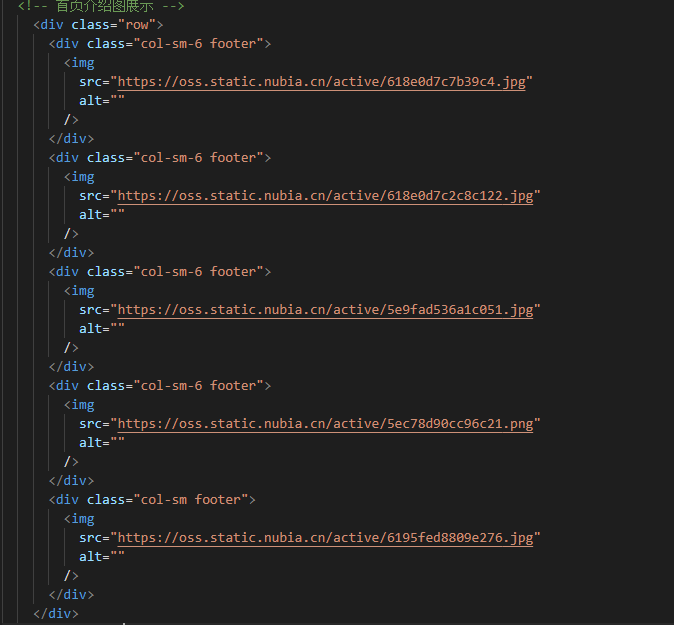
# 页面详细设计

## 3.1 主页面设计

1、主页采取轮播图+详情展示的效果，其中轮播图是采用的bootstrap4的组件



详情介绍图使用row均分的布局

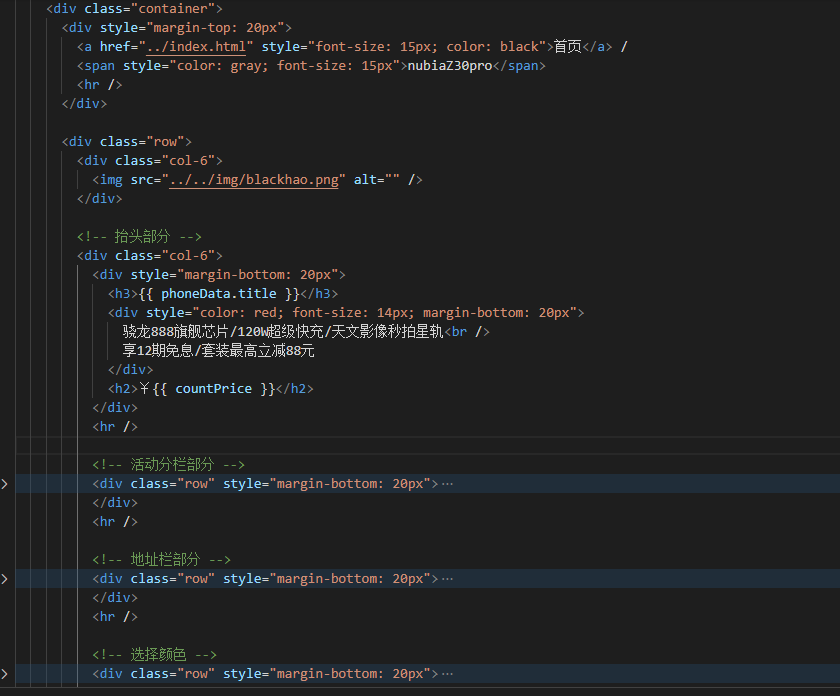


底部采用了卡片布局



## 3.2 商品页面设计

商品详情页左侧展示了产品图，右侧制定了基础模板，产品特定数据采用数据动态绑定渲染



1、接收App组件传递的商品数据，并动态渲染页面

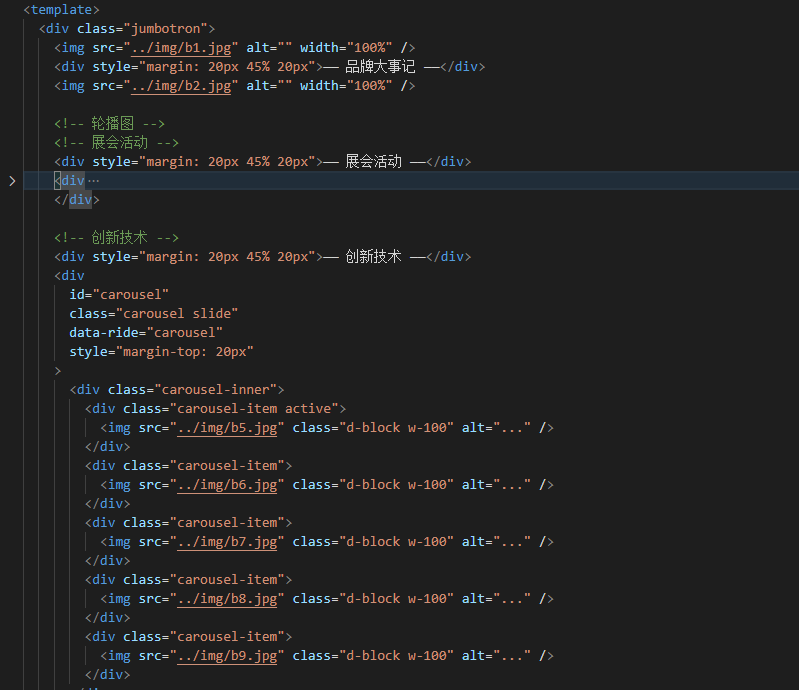
2、实现购物车商品数量动态的增加和减少

3、实现商品价格随数量的变动而变动



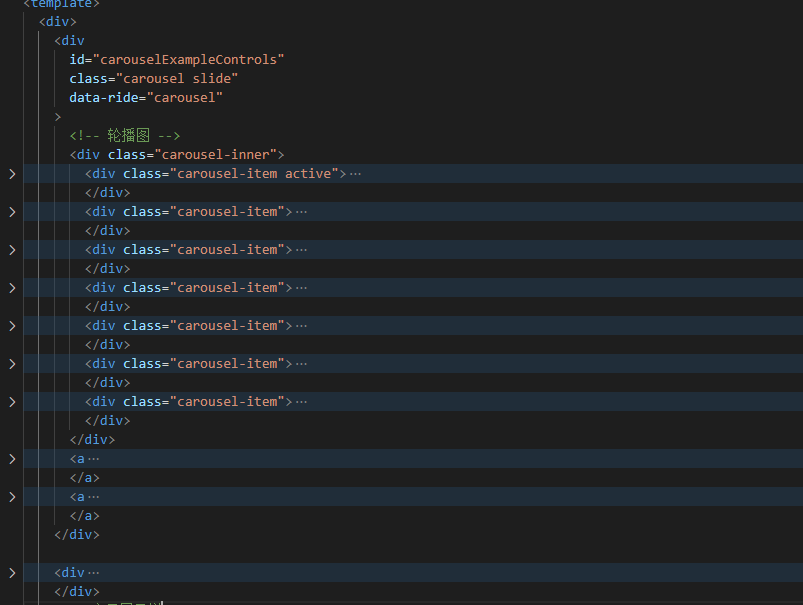
## 3.3 品牌宣传页面设计

使用轮播图展示商品信息



## 3.4 售后维护页面设计

轮播图展示手机新品信息



售后维修信息

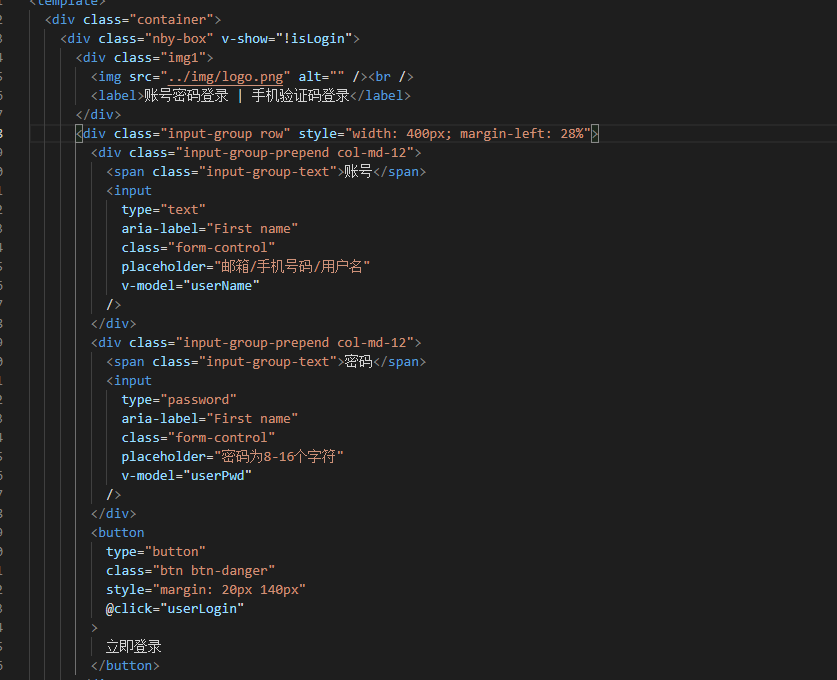


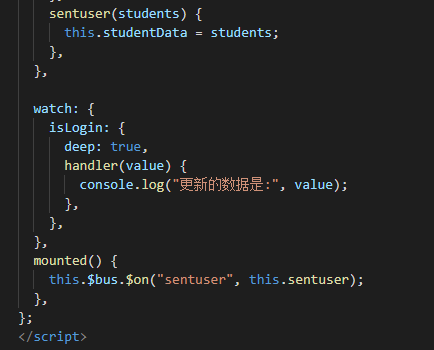
## 3.5 用户登录页面设计

1、用户名和密码动态绑定

2、App传送服务器用户数据给User页面进行数据对比（成功登录就跳转，失败则重现登录）

3、登录成功头像以及页面会同步更改





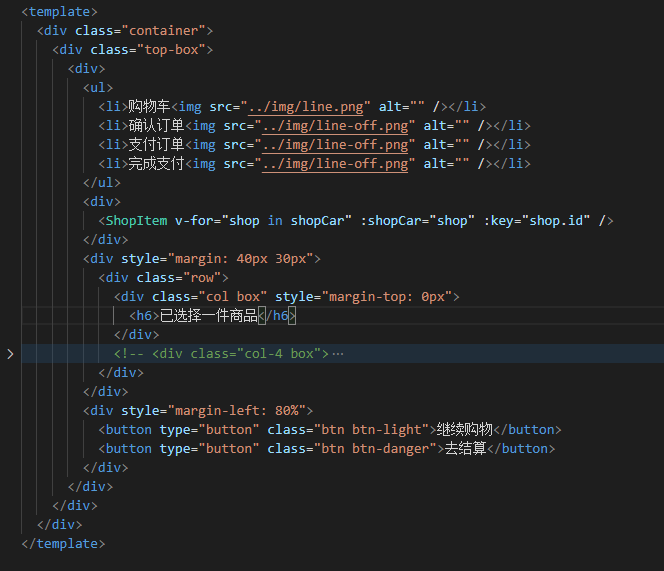


## 3.6 购物车板块页面设计

1、App组件提取服务器商品详情，

2、事件总线传递给购物车列表，

3、然后再遍历子组件并传递数组对象，渲染出数据



生命周期钩子中挂载接收的数据，在页面渲染前获取，便于后续的数据交互



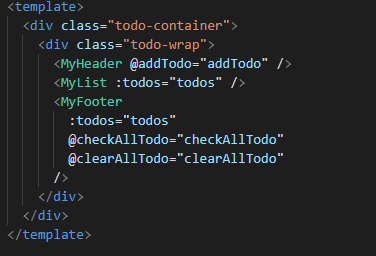
使用服务器里的商品标题以及价格，同步渲染页面并实现按钮对于商品数量的增加减少操作





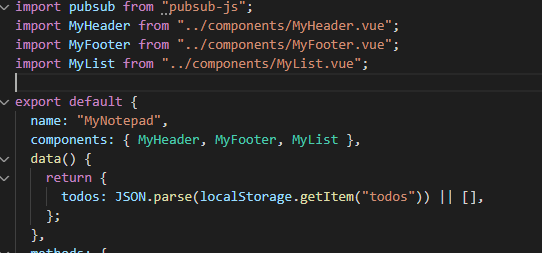
## 3.7 记事本页面设计

用父组件包裹其他三个子组件，方便数据的交互传输



引入外部资源库（消息订阅发布） 引入子组件，

将存储数据的数组写入浏览器的localStorage，实现页面刷新后数据仍然保留

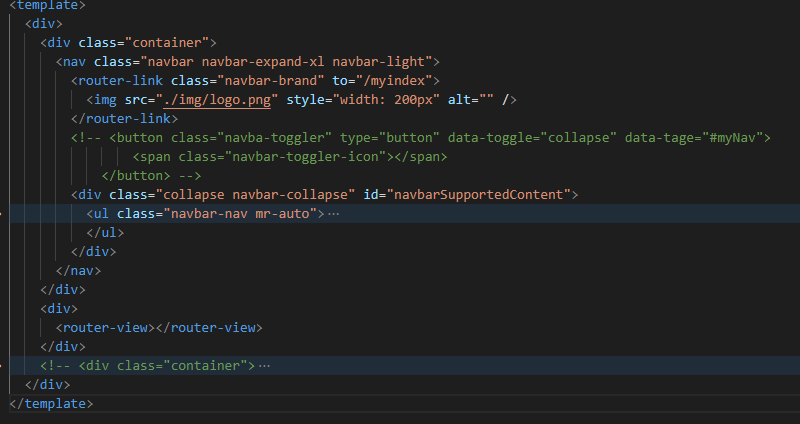






## **3.8 App组件设计**

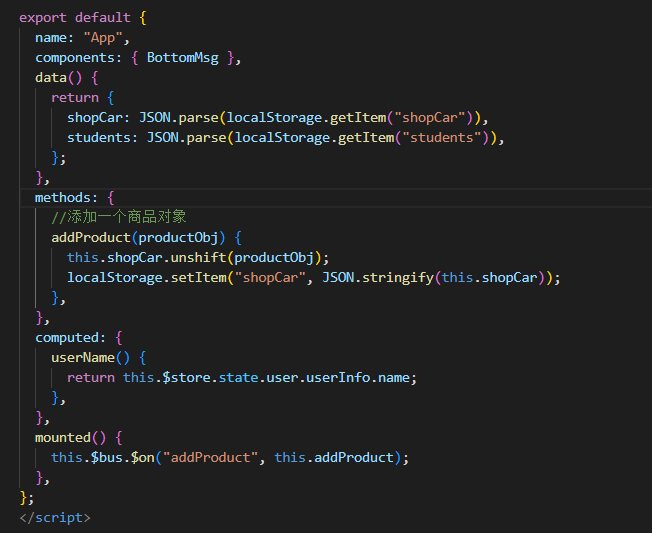
顶部采用导航栏的设计，中间加入了路由标签，使组件内容都在固定位置渲染



1、引入axios 以及消息订阅与发布库

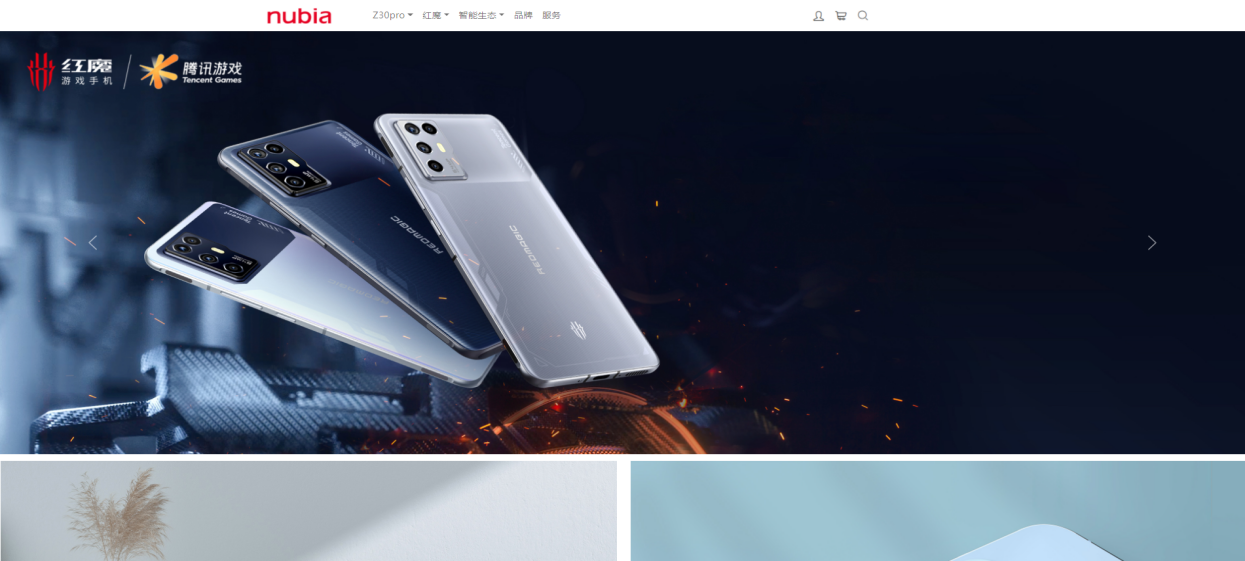
2、使用数组接收服务器接收的“学生”以及“用户”数据

3、使用事件总线传输数据给各个组件



# 项目测试

1. 首页展示









1. 商品页面展示

（通过首页的下拉菜单，路由到其他的页面）



手机展示页面1（商品名称和价格通过服务器数据进行渲染）



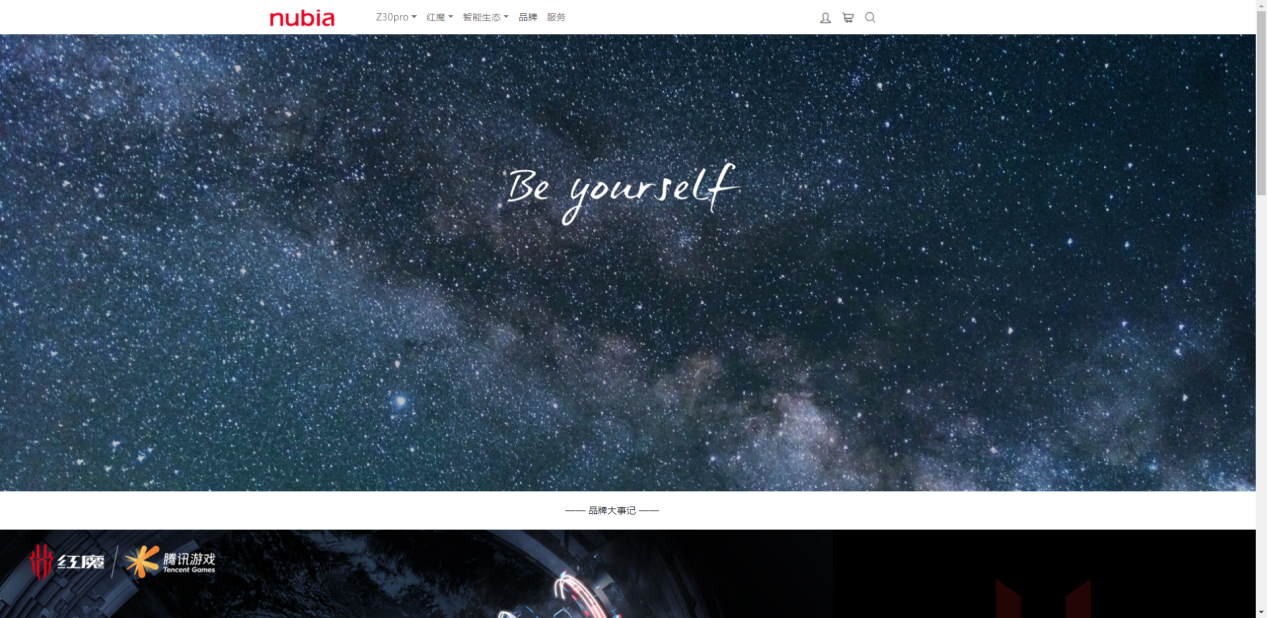
手机展示页面2（可根据需求增加或删减数量）

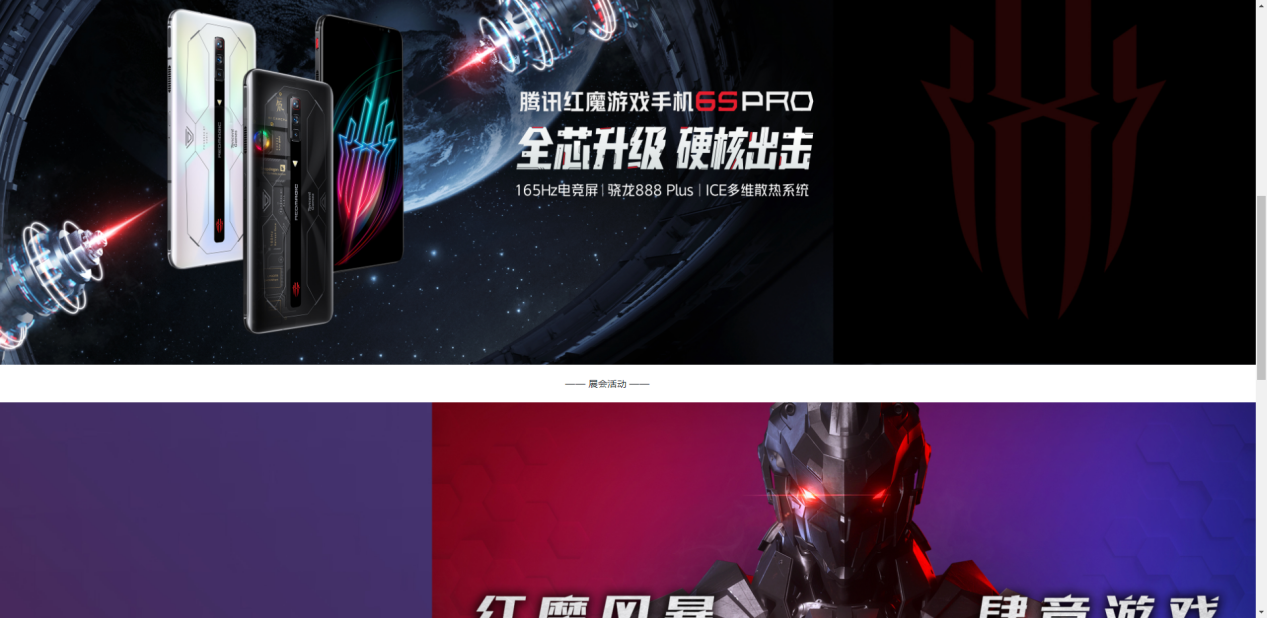


手机展示页面3（数量和价格实现了数据绑定，会同步更改）



3、产品推广页面演示（轮播图加商品展示）

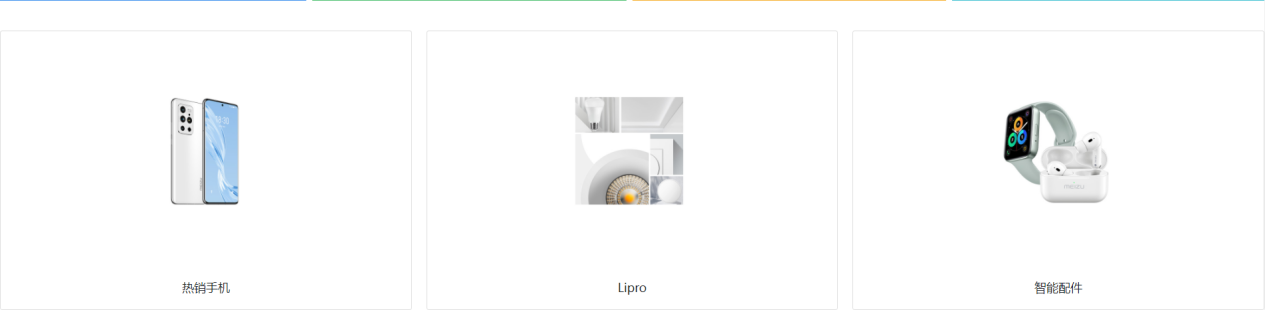








4、售后维护板块（轮播图加卡片布局以及浮动）



1. 用户登录页面（可以输入用户名密码登录）



（用户账号与密码检验成功后会出现弹框提示）

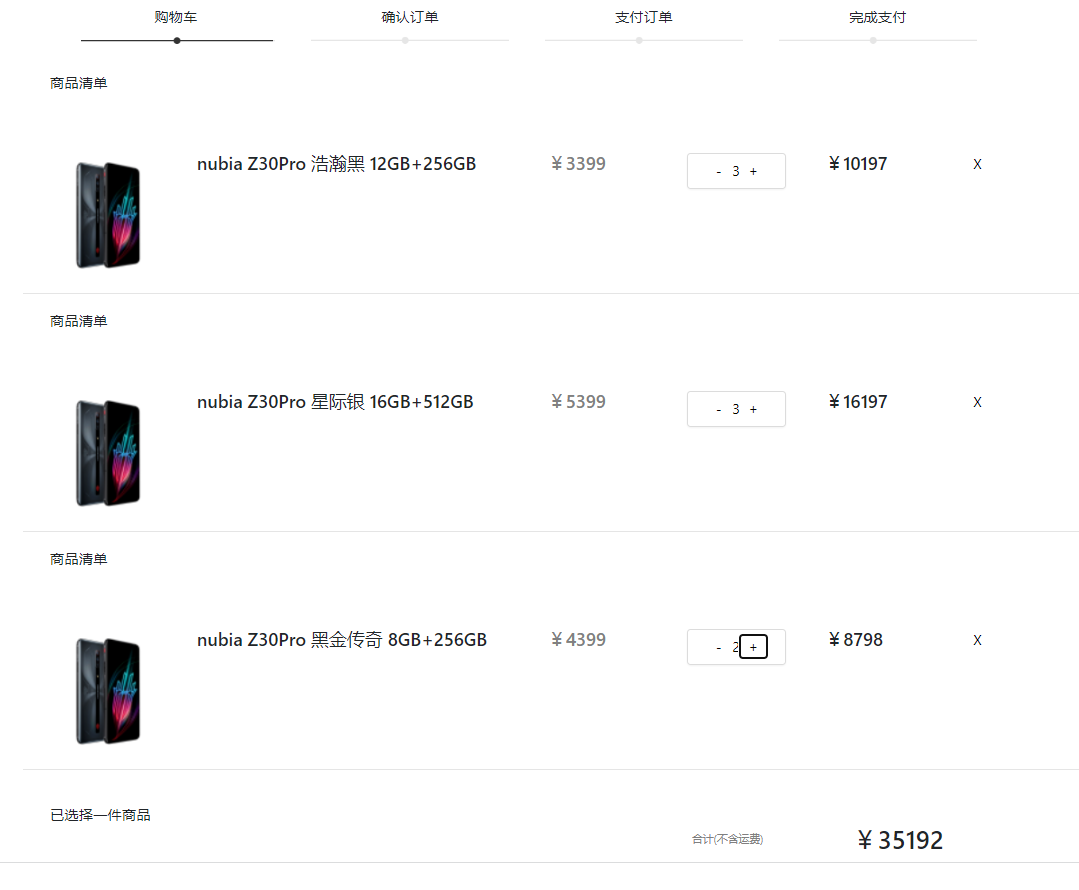


（如若重复注册则会提示：已被注册！）



（用户的头像和登录页的界面内容也会同步更改）

1. 购物车页面



（商品是通过请求服务器获取的数据，进而渲染到页面，通过子组件遍历出每一栏的数据）

1. 记事本功能

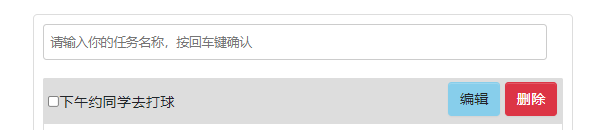


在个人网站的最后加入了记事本的模块，

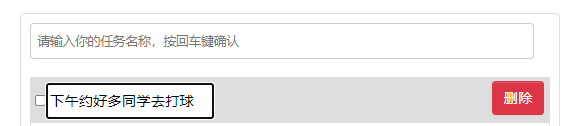
让深陷网购的自己想起还有哪些任务没做，进而起到警醒的作用。

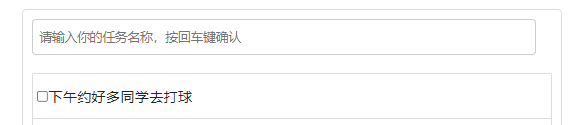
效果演示

修改编辑功能：



（点击编辑按钮组件切换状态）

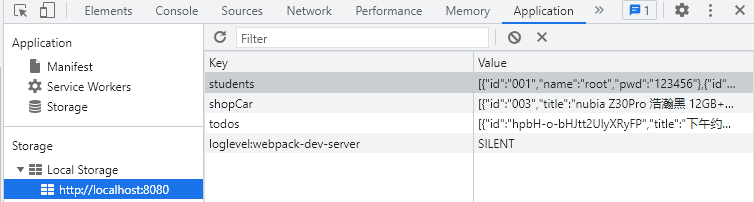




（编辑完成事件后，可选择“回车”或单击空白处完成逻辑修改）



（也可以实现全选事件删除，或者单一删除）



（数据首先会存储到浏览器的Local Storage中，实现刷新数据依旧保留的效果）

# 设计总结及心得体会

在这次设计中第一次体会到独立完成一个项目的难度，但是由于前端的页面构成借助了很多bootstrap的组件，已经里面的布局样式类，所以在页面构成这一块，效率提高了不少，从中也体会到老师说的，看书百次不如实践一次。这次我采用的是Vue+Bootstrap+Node.js+axios的方式来实现这个项目。

其中bootstrap用来构建页面的基础布局，实现一些轮播图，视频，图片的排列，其中的定位、浮动的问题也请教了同学不少，才使得自己有了些许清晰的理解。

而Vue则是用来完善里面功能模块的逻辑跳转，数据绑定，已经一些基本的事件处理，这也是我第一次使用组件化的方式来完成项目，也体验到了框架对于实际开发应用的重要性，虽然里面很多的功能都是只完善了一部分，但后续会继续朝着这个方向优化。

但是也有美中不足的点，就是由于没有springboot的基础所以无法连接数据库，所以才想到用node.js搭一个基础服务器，将数据存放在里面，然后再用Vue请求调用数据。

这次的页面传参也是使用的路由传参，代替了全局事件总线，使得数据可以高效的通过路由表进行传递，页面渲染的效果也会有大幅提升。

# 参考文献

Html+css

Bootstrap4

node.js

Vue.js