|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 题目名称 | 基于Vue前端框架技术设计的一个购物网站 | | | | |
| **姓名** |  | **班级** | **软件工程191** | **学号** | **102** |
|  | | | | | |
| 具体要求 | | | | | |
| **1.功能要求**  **1）前端功能**  商品分类展示;  商品详情  商品搜索  订单详情(购物车)  登录与注册  注意：注册用户可以购买商品，非注册用户仅可以浏览、查询商品  **2）后端功能（可选）**  提供商品查询接口，满足系统功能。  或者：使用模拟数据，不需要后端提供数据接口  **2.技术要求**  1）基于Vue前端框架技术，使用组件化开发方式。  2）知识点应用：[1] 项目创建 ；[2] Vue指令应用：插值、数据绑定、计算属性、方法、侦听器、事件监听等 ; [3] Vue组件创建和应用 [4] Vue路由使用；[5]前端基础知识的综合应用。  **3.设计文档**  1）内容  整体设计（功能描述、系统结构图、使用到的技术等）；数据来源及描述；详细设计(代码和界面)；小结；自我评价；  2）格式  参照任务书模版。要求格式良好。  提交：  1.项目源代码；  2.设计文档；  3.提交日期；2021-12-24 | | | | | |

目·录

[基于Vue前端框架技术设计的一个购物网站 3](#_Toc90751776)

[一、整体设计 3](#_Toc90751777)

[1.1系统结构 4](#_Toc90751778)

[1.2功能结构图 4](#_Toc90751779)

[二、数据来源及描述 4](#_Toc90751780)

[2.1数据存储说明 4](#_Toc90751781)

[三、详细设计 4](#_Toc90751782)

[3.1 主页功能模块设计 4](#_Toc90751783)

[3.2书籍详情设计 4](#_Toc90751784)

[3.3 购物车设计 4](#_Toc90751785)

[3.4登陆/注册设计 4](#_Toc90751786)

[3.5 数据获取设计 4](#_Toc90751787)

[3.6 刷新丢失数据设计 4](#_Toc90751788)

[四、结果展示 4](#_Toc90751789)

[五、设计小结 4](#_Toc90751790)

[六、自我评价 4](#_Toc90751791)

# 基于Vue前端框架技术设计的一个购物网站

## 一、整体设计

### 1.1系统结构

系统采取模拟数据的方式完成，通过在localStorage保存数据，通过维护该数据，不断地写入，以及读取，对登陆信息保存在sessionStorage；初始化页面时，若未登录，则只可以浏览。对登陆的用户可以浏览商品加下单，用户登陆之后，进行读取该用户的购物车信息，保存至store内，点击购物车时，直接展示出来。

使用多个路由视图切换实现页面跳转，对主页下面有多个分类的子路由，当需要登陆时整个app.vue进行切换，完成路由的切换。

### 1.2功能结构图



图1. 1功能结构图

## 二、数据来源及描述

### 2.1数据存储说明

书籍信息：书籍信息存放于data.js内，每个需要的页面可以直接引入，引入后就可以使用了，需要增删改查，只需要进行修改此处的数据即可。

用户信息：用户进行注册的时候，将数据模拟存储在localStorage对象内，使用setItem的方法进行设置，表名为“！！！user！！！”，数据采用对象{username：“”，password：“”}的方式进行模拟存储。

某用户的购物车信息：数据存储在localStorage对象内，表名为“！！！用户名！！！”，数据形式为{book：{}，num：X}，采用此形式，可以很方便的进行数据维护。

下单采用延时模拟的方法进行模拟的，并且对购物车的信息进行更改。

## 三、详细设计

### 3.1 主页功能模块设计

主页直接展示整个网页的头部和数据部分，点击书籍的图片，可以跳转到该书籍的详情页面。提供登陆/注销登陆、进入购物车、搜索书籍的功能。

展示的数据来源于data.js，登录信息来源于store内保存的数据。

主页代码

<template>

<div id="main">

<img id="img1" src="@/img/ad.f6f4649c.jpg">

<div id="login">

<div v-if="login">{{msg}} &nbsp; <button class="lo" @click="changelogin()">退出登陆</button> </div>

<div v-if="!login">{{msg}} &nbsp; <button class="lo" @click="changelogin()">请登陆</button> </div>

</div>

<div id="inpDiv">

<img id="img2" src="@/img/logo.9d476f20.png">

<input id="inp" placeholder="请输入书名..." v-model="input">

<button id="gobtn" @click="find">查找</button>

<div id="mycart" @click="entermycart()" >

我的购物车

</div>

</div>

<div id="nav">

<ul id='dh'>

<li @click="topath('/')">首页</li>

<li @click="topath('/renwen')">人文类</li>

<li @click="topath('/lishi')">历史类</li>

<li @click="topath('/yishu')">艺术类</li>

<li @click="topath('/keji')">科技类</li>

</ul>

</div>

<router-view/>

<br/>

<br/>

<br/>

<div style="text-align:center">-------------------------------我也是有底线的--------------------------------</div>

<br/>

<br/>

<br/>

<br/>

<br/>

<br/>

</div>

</template>

<script>

import {books} from '@/data/data.js'

export default {

name:'main',

data:function(){

return {

books,

input:'',

login:false

}

},

computed:{

msg:function(){

var msg=""

if(this.$store.state.username!=""){

msg = "欢迎您："+this.$store.state.username

}else{

msg="未登录"

}

return msg

}

},

mounted:function(){

if(this.$store.state.username!=""){

this.login=true

}else{

this.login = false

}

},

methods:{

find:function(){

this.$router.push({path:'/',query:{input:this.input}})

},

entermycart:function(){

this.$router.push('/mycart')

},

changelogin(){

if(this.login){

this.$store.state.username="" //全局变量改变

this.$store.state.carts=[]

sessionStorage.removeItem("logintag")

this.login=false

}else{

this.login = true

this.$router.push("/login")

}

},

topath(to){

this.$router.push(to)

}

}

}

</script>

### 3.2书籍详情设计

当点击书籍时，路由将发生跳转，对整个路由视图进行置换，传入书籍的信息，进行展示。详情内提供数量选择，并且可以加入购物车，加入购物车的时候弹出一个添加成功的标记。

代码：

<template>

<div id="book">

<div id='left'>

<img :src="book.img">

</div>

<div id='right'>

<p>书名:{{book.bookName}}</p>

<p>单价:￥{{book.price}}</p>

<br/>

<div>

<button class="change" @click="change(-1)">-</button>

<input id="binp" disabled :value="sum">

<button class="change" @click="change(1)">+</button>

</div>

<br/>

<button id="addCart" @click="add()">加入购物车</button>

</div>

<div id="msg">

<h2><br/><br/>详 细 介 绍 :</h2>

<p>&nbsp;&nbsp;{{book.msg}}<br/><br/></p>

</div>

</div>

</template>

<script>

import {books} from "../data/data.js"

export default {

data:function(){

return {

sum:1

}

},

computed:{

book:function(){

for(var i=0;i<books.length;i++){

if(books[i].id==this.$route.query.id){

return books[i]

}

}

return null

}

},

methods:{

//改变计数器数据

change:function(num){

if(this.sum<=1&&num==-1)return;

this.sum+=num;

},

add(){

if(sessionStorage.getItem("logintag")==null){

this.$router.push("/login")

return

}

this.$store.commit('addCart',{'book':this.book,'sum':this.sum})

alert("添加成功！")

}

}

}

</script>

### 3.3 购物车设计

购物车的设计，采用表格展示的方法，对传入的数据进行展示，再对数据进行计算总价，对数据可以提供支付的功能。

代码：

<template>

<div id="cart">

<h1 style="text-align:center;color:red">我 的 购 物 车</h1>

<div class="no-data" v-if="carts.length==0">购物车空空如也</div>

<table v-else>

<br/>

<tr>

<th>图片</th>

<th>书名</th>

<th>价格</th>

<th>数量</th>

<th>操作</th>

</tr>

<tr id="tr" v-for="(cart,index) in carts" :key="cart.id">

<td><img id="cart-img" :src="cart.book.img"></td>

<td>{{cart.book.bookName}}</td>

<td>￥{{cart.book.price}}</td>

<td>

<button @click="downCart(index)">-</button>

{{cart.sum}}

<button @click="upCart(index)">+</button>

</td>

<td><button @click="dele(index);settotal(index)">移除订单</button></td>

</tr>

<br/>

<br/>

<tr>

<td colspan="4" style="color:red">当前总价： ￥{{total}}元</td>

<td><button id="pay" @click="topay()">去支付</button></td>

</tr>

</table>

</div>

</template>

<script>

import {mapState} from 'vuex'

import {mapMutations} from 'vuex'

import {mapActions} from 'vuex'

export default {

name:"cart",

data:function(){

return{

}

},

computed:{

...mapState(['carts','total']),

total(){

var total=0

for(var i=0;i<this.carts.length;i++){

total += this.carts[i].book.price\*this.carts[i].sum

}

return total.toFixed(2);

}

},

methods:{

...mapMutations(['dele','downCart','upCart']),

//点击选择事件

...mapActions(['topay'])

}

}

</script>

### 3.4登陆/注册设计

#### 3.4.1 登陆

在点击加入购物车或者点击登陆的时候，若未登录的话，则直接跳转到登陆页面，在登陆界面，需要输入用户名和密码，提交的时候，会和后台的数据进行比对，如果一样的话，就会显示登陆成功，并且可以进入购物车等等。

登陆页面代码：

<template>

<div id="background">

<div class="container">

<form action="">

<h1>Login</h1>

<div class="form">

<div class="item">

<label>用户名：</label><input type="text" name="username" v-model="username" placeholder="请输入用户名">

<!-- v-model把输入的值传输给name变量 -->

<br/>

</div>

<div class="item">

<label>密码：</label><input type="password" name="password" v-model="password" placeholder="请输入密码">

<br/>

</div>

</div>

</form>

<button type="submit" @click.prevent="handlelogin">登录 </button>

<!-- v-on点击按钮触发handlelogin方法 -->

<button @click.prevent="handleregister">注册</button>

</div>

</div>

</template>

<script>

export default {

data(){

return{

username:"",//姓名，用v-model绑定监听，将输入的字符串赋值给name变量

password:""//密码

};

},

methods:{

//登录判断

handlelogin:function()

{

if(this.username==='')//名字为空

{

alert('用户名不为空');

}

else if(this.password==='')//密码为空

{

alert('密码不为空');

}

else{

var user = JSON.parse( localStorage.getItem("!!!user!!!"))

if(user==null||0==user.length-1){

alert('账号不存在，请注册后登录');//查无此号

this.password=""

return

}

for(var i=0;i<user.length;i++){

if(user[i].username==this.username){ //用户名存在

if(user[i].password==this.password){

sessionStorage.setItem("logintag",this.username)

//加载数据

var cart = JSON.parse(localStorage.getItem("!!!"+sessionStorage.getItem("logintag")+"!!!"))

if(cart!=null){

this.$store.state.carts=cart

}

this.$store.state.username = sessionStorage.getItem("logintag")

//跳转

this.$router.push('/')

return

}else{

alert('密码不正确')

this.password=""

return

}

}

}

}

},

handleregister:function()

{

this.$router.push('/register')//点击注册按钮，跳转至注册页面

}

}

};

</script>

#### 3.4.2注册

若无账号的情况，可以跳转到注册的界面，通过输入的信息，一旦数据正确，就可以写入数组中，再转到登陆的页面。

注册代码：

<template>

<div id="background">

<div id="contain">

<h1>Register</h1>

<div class="form">

<label>用户名：</label><input type="text" v-model="name"><br/>

</div>

<div class="form">

<label>密码：</label><input type="password" v-model="password"><br/>

</div>

<div class="form">

<label>邮箱：</label><input type="email" v-model="mail"><br/>

</div>

<div class="form">

<label>手机号：</label><input type="tel" v-model="tel"><br/>

</div>

<button @click.prevent="handlefinish">提交</button>

</div>

</div>

</template>

<script>

export default {

name:'register',

data:function(){

return{

name:"",

password:"",

mail:"",

tel:""

}

},

methods:{

//点击完成按钮触发handlefinish

handlefinish:function()

{

if(this.name==='')

{

alert("用户名不能为空");

}

else if(this.password==='')

{

alert("密码不能为空");

}

else{//将新用户信息存储到localStorage

var user = JSON.parse(localStorage.getItem('!!!user!!!'))

for(var i=0;user!=null&&i<user.length;i++){

if(user[i].username===this.name){

alert("用户名已存在")

return;

}

}

if(user==null){

user=[]

}

user.push({'username':this.name,'password':this.password,'mail':this.mail,'tel':this.tel})

localStorage.setItem('!!!user!!!',JSON.stringify(user))

alert("注册成功");

this.$router.push('/Login');//完成注册后跳转至登录页面

}

}

}

};

</script>

### 3.5 数据获取设计

数据的入口都在store的对象内，包括登陆名字和购物车，通过不同的人登陆，进行加载的数据都保存在此，因此这一部分的数据特别重要。

Store的设计

export default new Vuex.Store({

state: {

carts:[],

username:""

},

mutations: {

//添加商品

//加入购物车

addCart:function(state,poyload){

//poyload=[{book:{},sum:x]

//未登录

if(!sessionStorage.getItem("logintag")){

this.$router.push('/login')

return

}

for(var i=0;i<state.carts.length;i++){

if(state.carts[i].book.id==poyload.book.id){

state.carts[i].sum+=poyload.sum;

break;

}

if(i==state.carts.length-1){

state.carts.push(poyload)

break;

}

}

if(state.carts.length==0){

state.carts.push(poyload)

}

localStorage.setItem("!!!"+sessionStorage.getItem("logintag")+"!!!",JSON.stringify(state.carts))

},

//删除商品

dele(state,index){

//删除购物车数据

state.carts.splice(index,1)

//删除模拟数据本地数据

localStorage.setItem("!!!"+sessionStorage.getItem("logintag")+"!!!",JSON.stringify(state.carts))

},

//购物车--

downCart(state,index){

//如果数量为1，不进行修改

if(state.carts[index].sum==1)return

//购物车数据进行修改

state.carts[index].sum--

//修改模拟本地数据信息

localStorage.setItem("!!!"+sessionStorage.getItem("logintag")+"!!!",JSON.stringify(state.carts))

},

//购物车++

upCart(state,index){

//购物车

state.carts[index].sum++

//修改模拟本地数据信息

localStorage.setItem("!!!"+sessionStorage.getItem("logintag")+"!!!",JSON.stringify(state.carts))

}

},

//加载数据

createCart(){

},

actions: {

//模拟支付

topay({state}){

setTimeout(function(){

state.carts=[]

localStorage.setItem("!!!"+sessionStorage.getItem("logintag")+"!!!",JSON.stringify(state.carts))

alert("支付完成")

},1000)

}

},

modules: {

}

})

### 3.6 刷新丢失数据设计

在刷新的时候，会使得整个路由进行刷新的，此时的购物车数据store清空了，因此在路由页面重新加载时需要在APP.vue内的created钩子函数内，书写加载的代码

代码：

created:function(){

//刷新页面时进行数据获取

//已登录

if(sessionStorage.getItem("logintag")){

var cart = JSON.parse(localStorage.getItem("!!!"+sessionStorage.getItem("logintag")+"!!!"))

if(cart==null)return

this.$store.state.carts=cart

this.$store.state.username = sessionStorage.getItem("logintag")

}

}

## 四、结果展示

打开网页的状态，此时是登陆状态，因此显示用户名，并且有退出登陆的按钮。

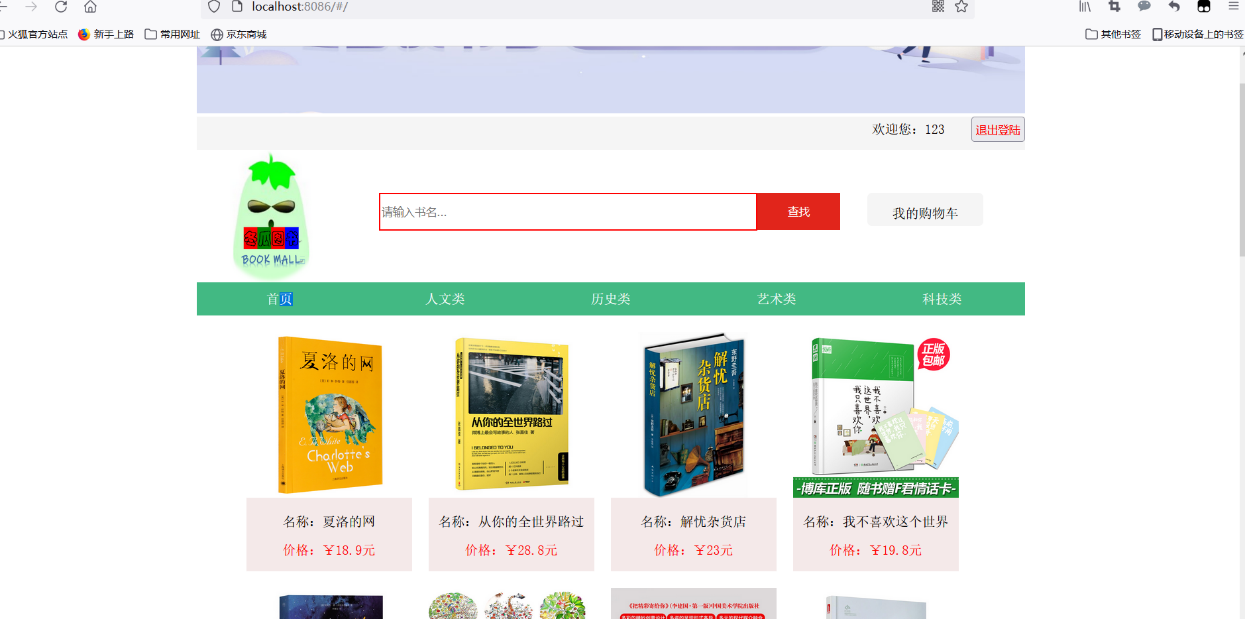


图4. 1首页展示

点击商品，显示信息



图4. 2商品展示



图4. 3商品展示

点击加入购物车，加入多个数量，将会提示加入成功

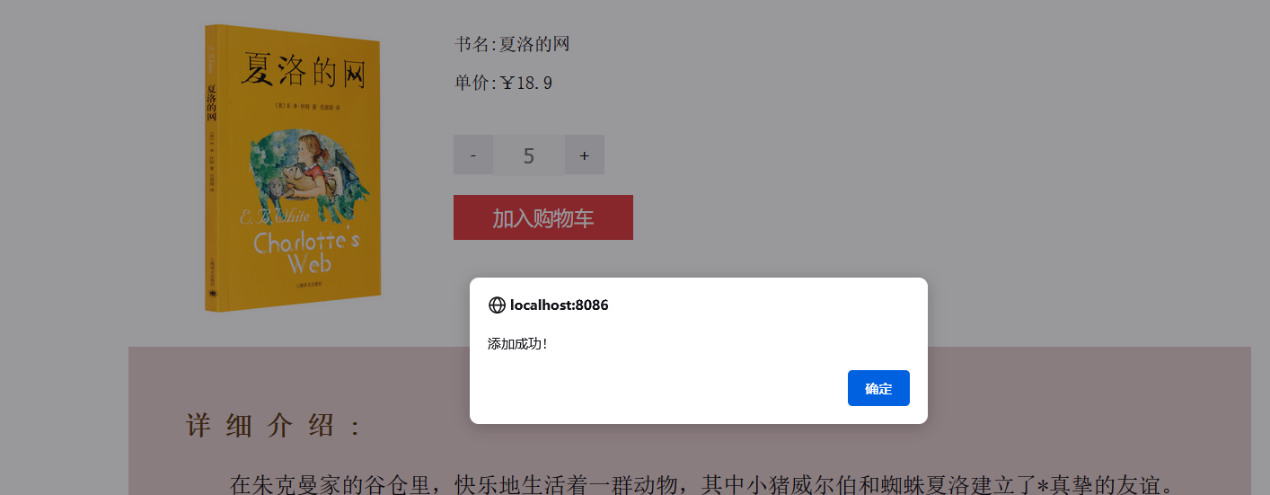


图4. 4添加成功

打开购物车，将数据展示出来



图4. 5购物车展示

此时可以进行数量的增减，和移除订单，最后在最底下有总价格和支付的功能。



图4. 6支付功能

当需要登陆的时候，跳转到登陆界面



图4. 7登陆功能展示

注册页面的展示



图4. 8注册页面展示

## 五、设计小结

本次期末的设计使用了本学期学习的内容，例如各种数据绑定和路由的切换，在掌握了相关的课程知识之后，利用vue的内容进行本次设计，我觉得还是挺不错的。

在一开始的设计之初，先对本次设计要求进行构思，最后还是决定制作本次这个冬瓜书城的网站，对于该网站，我的想法是通过不一样的路由切换实现书籍的分类展示，最后加上一些购物车的逻辑信息，以及对书籍的描述，最后做出了这个系统。

设计的过程中，相对而言学到的东西还是挺多的，尤其是涉及到全局的路由管理，在这一块我还是很薄弱的，最后我通过查看ppt以及网上资料的方法，最后把其学通了。从而实现了模拟数据的设计。如今可以实现分用户的购物车管理和用户状态管理。

## 六、自我评价

本次的设计实验，我觉得做得不是很好，虽然有完成所要求的所有功能，毕竟时间紧迫，因此本次设计我选择了模拟数据，没有进行后端的数据获取，再接下来的日子，在学习新技术的时候，会对旧知识进行整理，最后完善这些功能的。

综上所述，我觉得本次设计的成绩可以达到良好的。