第8章 “微商城”前台开发文档

8.1 准备工作

8.1.1 导入项目

（1）创建 D:\vue\chapter08 目录。

（2）从配套源代码中，将项目模板“my-shop-template”文件夹复制到 chapter08 目录，并将其重命名为“my-shop”。

（3）使用命令提示符打开 D:\vue\chapter08\my-shop 目录，安装依赖。

```
yarn
yarn add axios@1.2.2 --save
yarn add less@4.1.3 --save
yarn add pinia@2.0.27 --save
yarn add pinia-plugin-persist@1.0 --save
yarn add vue-router@4 --save
yarn add vant@4.0 --save
```

（4）打开 index.html 修改标题。

```
<title>微商城</title>
```

（5）禁止双击缩放。

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no" />
```

说明：本文档中标注红的代码为当前步骤新增或修改的代码。

8.1.2 定义路由

（1）创建 src\router\index.js，具体代码如下。

```
import { createRouter, createWebHistory } from 'vue-router'
const router = createRouter({
  history: createWebHistory(),
  routes: [
    {
      path: '/',
```

```
        redirect: '/home',
        meta: { title: '首页' }
      },
      {
        path: '/home',
        component: () => import('../pages/Home.vue'),
        name: 'home',
        meta: { title: '首页', name: 'home', isTab: true }
      },
      {
        path: '/category',
        component: () => import('../pages/Category.vue'),
        name: 'category',
        meta: { title: '分类', isTab: true, isShowNav: true }
      },
      {
        path: '/message',
        component: () => import('../pages/Message.vue'),
        name: 'message',
        meta: { title: '消息', isTab: true, isShowNav: true }
      },
      {
        path: '/cart',
        component: () => import('../pages/Cart.vue'),
        name: 'cart',
        meta: { title: '购物车', isTab: true, isShowNav: true, isShowBack:
true }
      },
      {
        path: '/user',
        component: () => import('../pages/User.vue'),
        name: 'user',
        meta: { title: '我的', isTab: true }
      },
    ]
  })
  router.beforeEach((to, from, next) => {
    const title = to.meta && to.meta.title
```

```
  if (title) {
    document.title = title + ' - 微商城'
  }
  next()
})
export default router
```

解释

isTab：是否显示底部 Tabbar 标签栏，true 表示显示，false 表示不显示。

isShowNav：是否显示 NavBar 导航栏，true 表示显示，false 表示不显示。

isShowBack： 是否显示导航栏中的返回箭头，true 表示显示，false 表示不显示。

（2）创建 src\pages\Home.vue，具体代码如下。

```
<template>
  Home
</template>
```

（3）创建 src\pages\Category.vue，具体代码如下。

```
<template>
  Category
</template>
```

（4）创建 src\pages\Message.vue，具体代码如下。

```
<template>
  Message
</template>
```

（5）创建 src\pages\Cart.vue，具体代码如下。

```
<template>
  Cart
</template>
```

（6）创建 src\pages\User.vue，具体代码如下。

```
<template>
  User
</template>
```

（7）修改 src\App.vue 中的模板部分，具体代码如下。

```
<template>
  <router-view></router-view>
</template>
```

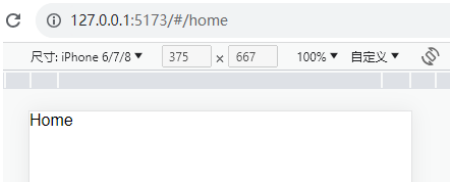
（8）修改 src\main.js 中的所有内容，具体代码如下。

```
import { createApp } from 'vue'
import App from './App.vue'
```

```
import Vant from 'vant'
import 'vant/lib/index.css'
import router from './router'
import { createPinia } from 'pinia'
import piniaPluginPersist from 'pinia-plugin-persist'
const pinia = createPinia()
pinia.use(piniaPluginPersist)

const app = createApp(App)
app.use(Vant)
app.use(pinia)
app.use(router)
app.mount('#app')
```

（9）启动项目，具体命令如下。

```
yarn dev
```



8.1.3 实现底部导航栏

（1）创建 src\components\TabBar.vue，具体代码如下。

```
<template>
  <van-tabbar route fixed placeholder border>
    <van-tabbar-item replace :to="{ name: 'home' }" icon="home-o">首页</van-tabbar-item>
    <van-tabbar-item replace :to="{ name: 'category'}" icon="apps-o">分类</van-tabbar-item>
    <van-tabbar-item replace :to="{ name: 'message'}" icon="chat-o" badge="4">消息</van-tabbar-item>
    <van-tabbar-item replace :to="{ name: 'cart'}" icon="shopping-cart-o">购物车</van-tabbar-item>
    <van-tabbar-item replace :to="{ name: 'user'}" icon="user-o">我的</van-tabbar-item>
```

```
  </van-tabbar>
</template>

<style scoped>
.van-tabbar-item {
  --van-tabbar-item-active-color: #FF8000;
}
</style>
```

（2）修改 src\App.vue 中的所有内容，具体代码如下。

```
<template>
  <router-view></router-view>
  <tab-bar v-if="isShowTabbar"></tab-bar>
</template>

<script setup>
import TabBar from './components/TabBar.vue'
import { useRoute } from 'vue-router'
import { ref, watch } from 'vue'

const route = useRoute()
const isShowTabbar = ref(true)

// 监听路由中的 isTab 是否为 true，如果为 true，展示底部 TabBar
watch(
  () => route.meta,
  val => {
    isShowTabbar.value = val.isTab
  }
)
</script>

<style>
#app {
  color: #2c3e50;
  line-height: 24px;
}
</style>
```

（3）访问测试，此时能看到页面底部的导航栏，效果如下。

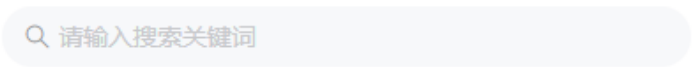
8.2 首页开发

8.2.1 实现首页搜索框

（1）创建 src\pages\Home.vue，具体代码如下。

```
<template>
  <van-search
    shape="round"
    v-model="value"
    placeholder="请输入搜索关键词"
    @search="onSearch"
    @cancel="onCancel"
  >
  </van-search>
</template>

<script setup>
import { ref } from 'vue'
import { showToast } from 'vant'

const value = ref('')
const onSearch = val=> showToast(val)
const onCancel = () => showToast('取消')
</script>
```

（2）访问测试，搜索框的效果如下。



8.2.2 实现首页轮播图

（1）创建 src\components\HomeSwiper.vue，具体代码如下。

```
<template>
```

```
    <div class="home-swiper">
      <van-swipe :autoplay="3000" lazy-render indicator-color="#FF8000">
        <van-swipe-item v-for="item in banner" :key="item">
          <img :src="item">
        </van-swipe-item>
      </van-swipe>
    </div>
</template>
<script setup>
const banner = [
  '/images/banner1.jpg',
  '/images/banner2.jpg',
]
</script>
<style lang="less" scoped>
.home-swiper {
  width: 100%;
  img {
    width: 100%;
  }
}
</style>
```

（2）修改 src\pages\Home.vue，导入组件，具体代码如下。

```
<template>
  ……（原有代码）
  <!-- 轮播图 -->
  <home-swiper></home-swiper>
</template>

<script setup>
import HomeSwiper from '../components/HomeSwiper.vue'
……（原有代码）
<script>
```

（3）查看轮播图效果，具体如下。

8.2.3 实现首页功能按钮区

（1）创建 src\components\HomeGrid.vue，具体代码如下。

```
<template>
  <div class="home-grid">
    <van-grid :column-num="5" square :gutter="5">
      <van-grid-item v-for="list in menulist" :key="list">
       <van-image :src="list.url" />
        <span>{{ list.text }}</span>
      </van-grid-item>
    </van-grid>
  </div>
</template>

<script setup>
import menu1 from '../assets/images/menu1.png'
import menu2 from '../assets/images/menu2.png'
import menu3 from '../assets/images/menu3.png'
import menu4 from '../assets/images/menu4.png'
import menu5 from '../assets/images/menu5.png'
import menu6 from '../assets/images/menu6.png'
import menu7 from '../assets/images/menu7.png'
import menu8 from '../assets/images/menu8.png'
import menu9 from '../assets/images/menu9.png'
import menu10 from '../assets/images/menu10.png'

const menulist = [
  { text: '今日爆款', url: menu1 },
  { text: '好物分享', url: menu2 },
  { text: '推荐购买', url: menu3 },
```

```
    { text: '购物心得', url: menu4 },
    { text: '直播专区', url: menu5 },
    { text: '签到中心', url: menu6 },
    { text: '值得购买', url: menu7 },
    { text: '每日优惠', url: menu8 },
    { text: '充值中心', url: menu9 },
    { text: '我的客服', url: menu10 }
  ]
</script>

<style lang="less" scoped>
.home-grid {
  .van-image {
    width: 55%;
  }
  span {
    font-size: 12px;
  }
}
</style>
```

（2）修改 src\pages\Home.vue，具体代码如下。

```
<template>
  ……（原有代码）
  <!-- 功能按钮区 -->
  <home-grid></home-grid>
</template>

<script setup>
import HomeSwiper from '../components/HomeSwiper.vue'
import HomeGrid from '../components/HomeGrid.vue'
……（原有代码）
<script>
```

（3）查看功能按钮区效果，具体如下。

8.2.4 实现首页商品信息展示区

（1）创建 src\components\HomeProduct.vue，具体代码如下。

```
<template>
  <div class="home-product">
    <ul>
      <li v-for="item in brandList" :key="item.id">
        <img :src="item.pic_url" alt="">
        <h4>{{ item.name }}</h4>
      </li>
    </ul>
  </div>
</template>

<script setup>
const brandList = [
  { id: 1, name: '直播', pic_url: '/images/product1.png' },
  { id: 2, name: '推荐', pic_url: '/images/product2.png' },
  { id: 3, name: '补贴', pic_url: '/images/product3.png' },
  { id: 4, name: '分享', pic_url: '/images/product4.png' }
]
</script>

<style lang="less" scoped>
.home-product > ul {
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  li {
    width: 49.5%;
```

```
    position: relative;
    img {
      width: 100%;
    }
    h4 {
      font-size: 14px;
      position: absolute;
      left: 2px;
      top: -13px;
      background-color: red;
      color: #fff;
      border-radius: 10%;
      padding: 1px 3px;
    }
  }
}
</style>
```

（2）修改 src\pages\Home.vue，具体代码如下。

```
<template>
  ……（原有代码）
  <!-- 商品信息展示区 -->
  <home-product></home-product>
</template>

<script setup>
import HomeSwiper from '../components/HomeSwiper.vue'
import HomeGrid from '../components/HomeGrid.vue'
import HomeProduct from '../components/HomeProduct.vue'
……（原有代码）
<script>
```

（3）查看商品信息展示区效果，具体如下。

8.2.5 实现每周上新

（1）创建 src\components\HomeNew.vue，具体代码如下。

```
<template>
  <div class="home-new">
    <div class="home-new-title">
      <h3>每周上新</h3>
    </div>
    <ul>
      <li v-for="item in newList" :key="item.id">
        <img :src="item.list_pic_url" alt="" />
        <p>{{ item.name }}</p>
        <p><span>￥</span>{{ item.retail_price }}</p>
      </li>
    </ul>
  </div>
</template>

<script setup>
const newList = [
  { name: '懒人小沙发', list_pic_url: '/images/new1.jpg', retail_price: '128.00' },
```

```
      { name: '减压弹力球', list_pic_url: '/images/new2.jpg', retail_price: '8
9.00' },
      { name: '简约一字夹发夹', list_pic_url: '/images/new3.jpg', retail_price:
 '12.8' },
      { name: '毛线小兔子耳朵发夹', list_pic_url: '/images/new4.jpg', retail_pr
ice: '9.9' }
    ]
    </script>

    <style lang="less" scoped>
    .home-new {
      .home-new-title {
        text-align: center;
        font-size: 16px;
        margin-top: 1.6rem;
        height: 50px;
        h3 {
          width: 50%;
          border-top: 2px solid #ccc;
          padding-top: 8px;
          margin: 0 auto;
        }
      }
      ul {
        display: flex;
        justify-content: space-between;
        flex-wrap: wrap;
        padding: 1rem 0 0;
        background-color: #f9f9f9;
        li {
          width: 49.5%;
          img {
            width: 100%;
          }
          p {
            text-align: center;
            margin: 0.5rem 0;
          }
```

```
      span {
        color: #FF8000;
        font-size: 12px;
      }
    }
  }
}
</style>
```

（2）修改 src\pages\Home.vue，具体代码如下。

```
<template>
  ……（原有代码）
  <!-- 每周上新 -->
  <home-new></home-new>
</template>

<script setup>
import HomeSwiper from '../components/HomeSwiper.vue'
import HomeGrid from '../components/HomeGrid.vue'
import HomeProduct from '../components/HomeProduct.vue'
import HomeNew from '../components/HomeNew.vue'
……（原有代码）
<script>
```

（3）查看每周上新效果，具体如下。

8.2.6 实现人气推荐

（1）创建 src\components\HomeTop.vue，具体代码如下。

```
<template>
  <div class="home-top">
    <h3>人气推荐</h3>
    <div class="content">
      <van-card
        v-for="item in goodsList"
        :key="item.id"
        :tag="item.tag"
        :price="item.retail_price"
        :origin-price="item.origin_price"
        :desc="item.goods_brief"
        :title="item.name"
        :thumb="item.list_pic_url"
      >
      </van-card>
    </div>
```

```
  </div>
</template>

<script setup>
const goodsList = [
  {
    retail_price: '299.00',
    name: '蚕丝被 正品桑蚕丝',
    goods_brief: '一级桑蚕丝，轻盈、透气、柔软',
    list_pic_url: '/images/top1.jpg',
    tag: 'TOP1'
  },
  {
    retail_price: '88.00',
    origin_price: '98.00',
    name: '儿童摇摇马',
    goods_brief: '安全、不会侧翻、爸妈放心',
    list_pic_url: '/images/top2.jpg',
    tag: 'TOP2'
  },
  {
    retail_price: '128.00',
    origin_price: '168.00',
    name: '可躺可睡休闲懒人沙发',
    goods_brief: '轻松看书、社交、办公、舒适放松',
    list_pic_url: '/images/top3.jpg',
    tag: 'TOP3'
  },
  {
    retail_price: '199.00',
    origin_price: '205.00',
    name: '儿童积木 拼装玩具',
    goods_brief: '大颗粒 家长更放心 不易吞咽、安全性高',
    list_pic_url: '/images/top4.jpg',
    tag: 'TOP4'
  },
  {
    retail_price: '89.00',
```

```
    origin_price: '99.00',
    name: '扭扭车 1——3 岁男女宝宝',
    goods_brief: '儿童扭扭车万向轮 防侧翻大人新款摇摆扭扭车',
    list_pic_url: '/images/top5.jpg',
    tag: 'TOP5'
  }
]
</script>

<style lang="less" scoped>
.home-top {
  h3 {
    font-size: 22px;
    line-height: 30px;
    text-align: center;
    margin: 0.5rem 0;
  }
  .content {
    --van-tag-primary-color: #FF8000;
    --van-card-font-size: 16px;
    --van-card-background: #f9f9f9;
    background: var(--van-card-background);
    :deep(.van-card) {
      margin-top: 0;
      .van-card__title {
        padding: 10px 0 5px;
      }
      .van-card__price-currency {
        font-size: var(--van-card-font-size);
      }
    }
    &::after {
      content: '';
      display: block;
      height: 3rem;
    }
  }
}
```

```
</style>
```

（2）修改 src\pages\Home.vue，具体代码如下。

```
<template>
  ……（原有代码）
  <!-- 人气推荐 -->
  <home-top></home-top>
</template>

<script setup>
import HomeSwiper from '../components/HomeSwiper.vue'
import HomeGrid from '../components/HomeGrid.vue'
import HomeProduct from '../components/HomeProduct.vue'
import HomeNew from '../components/HomeNew.vue'
import HomeTop from '../components/HomeTop.vue'
……（原有代码）
<script>
```

（3）查看人气推荐效果，具体如下。

8.3 消息页开发

（1）先实现页面顶部的导航栏，修改 src\App.vue，具体代码如下。

```
<template>
  <van-nav-bar
    :title="$route.meta.title"
    v-show="$route.meta.isShowNav"
```

```
    @click-left="onClickLeft"
    :left-arrow="$route.meta.isShowBack"
    fixed
    placeholder
    style="height: 46px"
  />
  ……（原有代码）
</template>

<script setup>
import TabBar from './components/TabBar.vue'
import { useRoute } from 'vue-router'
import { ref, watch } from 'vue'
import { useRouter } from 'vue-router'

const route = useRoute()
const isShowTabbar = ref(true)

const router = useRouter()
const onClickLeft = () => {
  if (history.length > 1) {
    router.back()
  } else {
    router.push({ name: 'home' })
  }
}
……（原有代码）
</script>

<style>
#app {
  color: #2c3e50;
  line-height: 24px;
  --van-nav-bar-background: #ff8000;
  --van-nav-bar-title-text-color: #fff;
  --van-nav-bar-icon-color: #fff;
}
</style>
```

页面顶部的导航栏效果，具体如下。

（2）修改 src\pages\Message.vue，具体代码如下。

```
<template>
  <van-cell-group v-for="item in lists" :key="item">
    <van-cell center :icon="item.img" :title="item.title" :value="item.va
lue" :label="item.label" />
  </van-cell-group>
</template>

<script setup>
const lists = [
  {
    img: '/images/avatar1.jpg',
    title: '食品旗舰店',
    value: '星期一',
    label: "您有一条店铺消息"
  },
  {
    img: '/images/avatar2.jpg',
    title: '水果旗舰店',
    value: '星期二',
    label: "亲爱的果粉："
  },
  {
    img: '/images/avatar3.png',
    title: '订阅号消息',
    value: '星期日',
    label: "水果旗舰店:【新到水果新品————粑粑柑、砂糖橘】"
  },
  {
    img: '/images/avatar4.png',
    title: '消息号内容',
    value: '星期一',
    label: "食品旗舰店：大量新品到货，速来选购"
  }
]
```

```
</script>

<style lang="less" scoped>
:deep(.van-cell) {
  .van-cell__left-icon {
    width: 40px;
    height: 40px;
    .van-icon__image {
      width: 100%;
      height: 100%;
    }
  }
  .van-cell__title {
    .van-cell__label {
      white-space: nowrap;
      text-overflow: ellipsis;
      overflow: hidden;
      width: 165px;
    }
  }
}
</style>
```

（3）消息页面的效果，具体如下。

8.4 用户登录和注册

8.4.1 实现登录页面

（1）修改 src\router\index.js，添加登录页面的路由，具体代码如下。

```
    routes: [
      ……（原有代码）
      {
        path: '/login',
        component: () => import('../pages/Login.vue'),
        name: 'login',
        meta: { title: '登录', isTab: true, isShowNav: true, isShowBack: true
 }
      },
    ]
```

（2）创建 src\pages\Login.vue，具体代码如下。

```
    <template>
      <van-form @submit="submitForm" @failed="onFailed" ref="ruleFormRef" :m
odel="form">
```

```
      <van-cell-group>
        <van-field
          v-model="form.username"
          label="账号："
          placeholder="请输入账号"
          clearable
          name="username"
          :rules="usernameRules"
        ></van-field>
      </van-cell-group>
      <van-cell-group>
        <van-field
          v-model="form.password"
          label="密码："
          placeholder="请输入密码"
          name="password"
          clearable
          type="password"
          :rules="passwordRules"
        ></van-field>
      </van-cell-group>
      <van-cell-group>
        <van-button block round type="primary" native-type="submit">登录</va
n-button>
      </van-cell-group>
    </van-form>
  </template>

  <script setup>
  import { ref, reactive } from 'vue'
  const form = reactive({
    username: 'demo1',
    password: '123456'
  })
  const ruleFormRef = ref()
  // 定义验证规则
  const usernameRules = ref([
    { required: true, message: '用户名不能为空' },
```

```
    { pattern: /^\w{3,16}$/, message: '用户名长度为 3-16 个字符' }
  ])
  const passwordRules = ref([
    { required: true, message: '密码不能为空' },
    { pattern: /^\w{6,24}$/, message: '密码必须为 6-24 位英文字母或数字' }
  ])
  // 表单提交函数
  const submitForm = async values => {
    console.log(values)
  }
  const onFailed = errorInfo => {
    console.log('failed', errorInfo)
  }
  </script>

  <style lang="less" scoped>
  button {
    position: fixed;
    top: 200px;
  }
  </style>
```

（3）查看登录页效果，具体如下。

8.4.2 封装网络请求

（1）服务器端 API，创建 src\config.js，具体代码如下。

```
export default {
```

```
  baseURL: 'http://127.0.0.1:8360'
}
```

（2）存储服务器返回的 token，创建 src\stores\token.js，具体代码如下。

```
import { defineStore } from 'pinia'
import { ref } from 'vue'

const useToken = defineStore('token', () => {
  const token = ref(null)
  const updateToken = val => token.value = val
  const removeToken = () => token.value = null
  return { token, updateToken, removeToken }
}, {
  persist: {
    enabled: true,
    strategies: [
      {
        key: 'token',
        storage: localStorage
      }
    ]
  }
})
export default useToken
```

（3）创建 src\utils\request.js，具体代码如下。

```
import axios from 'axios';
import useToken from '../stores/token'
import config from '../config'
import router from '../router'
import { showLoadingToast, showToast, closeToast} from 'vant'

const baseURL = config.baseURL
const service = axios.create({ baseURL })

// 请求拦截器
service.interceptors.request.use(config => {
  const { token } = useToken()
  showLoadingToast({
    message: '加载中...',
```

```
    forbidClick: true,
    loadingType: 'spinner'
  })
  if (token) {
    config.headers.jwt = token
  }
  return config
})

// 响应拦截器
service.interceptors.response.use(
  response => {
    closeToast()
    const { errno, data, errmsg } = response.data
    if (errno === 0) {
      if (errmsg !== '') {
        showToast({
          message: errmsg,
          type: 'success'
        })
      }
      return data || true
    }
    showToast({
      message: errmsg,
      type: 'error'
    })
    if (errno === 2) {
      router.push({ name: 'login' })
    }
    return false
  },
  error => {
    closeToast()
    showToast ({
      message: '请求失败',
      type: 'fail'
    })
```

```
    console.log(error)
  }
)

export default service
```

（4）封装 API，创建 src\api\index.js，具体代码如下。

```
import request from '../utils/request'
// 登录接口
export function login(data) {
  return request.post('/home/login', data)
}
```

8.4.3 实现登录功能

（1）修改 src\pages\Login.vue，具体代码如下。

```
<script setup>
import { ref, reactive } from 'vue'
import { login } from '../api'
import useToken from '../stores/token'

const { updateToken } = useToken()
……（原有代码）
</script>
```

（2）定义表单提交函数，具体代码如下。

```
const submitForm = async values => {
  const data = await login(values)
  if (data) {
    updateToken(data.token)
  }
}
```

（3）存储用户信息。

修改 src\api\index.js，调用用户信息接口，具体代码如下。

```
export function getUser() {
  return request.get('/home/user')
}
```

创建 src\stores\user.js，存储用户数据，具体代码如下。

```
import { defineStore } from 'pinia'
```

```
import { reactive } from 'vue'

const useUser = defineStore('user', () => {
  const defaultUser = {
    isLogin: false,
    username: '',
    avatar: ''
  }
  const user = reactive(Object.assign({}, defaultUser))
  const updateUser = options => {
    Object.assign(user, options)
    return user
  }
  const removeUser = () => {
    Object.assign(user, defaultUser)
    return user
  }
  return { user, updateUser, removeUser }
}, {
  persist: {
    enabled: true,
    strategies: [
      {
        key: 'user',
        storage: localStorage
      }
    ]
  }
})
export default useUser
```

（4）更新用户信息

修改 src\pages\Login.vue，具体代码如下。

```
import { login, getUser } from '../api'
import useToken from '../stores/token'
import useUser from '../stores/user'

const { updateToken } = useToken()
const { updateUser } = useUser()
```

```
if (data) {
  updateToken(data.token)
  const user = await getUser()
  updateUser({
    isLogin: true,
    username: user.username,
    avatar: user.avatar
  })
}
```

（5）登录成功后，跳转到“我的”页面，具体代码如下。

```
import { useRouter } from 'vue-router'

const { updateToken } = useToken()
const { updateUser } = useUser()
const router = useRouter()
```

```
if (data) {
  ……（原有代码）
  router.push({ name: 'user' })
}
```

（6）查看登录页面效果，具体如下。

8.4.4 检查登录状态

修改 src\App.vue，具体代码如下。

```
import { ref, watch, onMounted } from 'vue'
import { getUser } from './api'
import useUser from './stores/user'

const { user, updateUser } = useUser()

onMounted(() => {
  if (user.isLogin) {
    loadUser()
  }
})

const loadUser = async () => {
  const data = await getUser()
  updateUser({
    isLogin: true,
    username: data.username,
    avatar: data.avatar
  })
}
```

8.4.5 实现注册功能

（1）修改 src\router\index.js，添加注册页面的路由，具体代码如下。

```
routes: [
  ……（原有代码）
  {
    path: '/register',
    component: () => import('../pages/Register.vue'),
    name: 'register',
    meta: { title: '注册', isTab: true, isShowNav: true, isShowBack: true
}
```

```
    },
  ]
```

（2）修改 src\api\index.js，调用注册接口，具体代码如下。

```
export function register(data) {
  return request.post('/home/register', data)
}
```

（3）创建 src\pages\Register.vue，具体代码如下。

```
<template>
  <van-form @submit="submitForm" @failed="onFailed" ref="ruleFormRef" :model="form">
    <van-cell-group>
      <van-field
        v-model="form.username"
        label="账号："
        placeholder="请输入账号"
        clearable
        name="username"
        :rules="usernameRules"
      ></van-field>
    </van-cell-group>
    <van-cell-group>
      <van-field
        v-model="form.password"
        label="密码："
        placeholder="请输入密码"
        clearable
        type="password"
        name="password"
        :rules="passwordRules"
      ></van-field>
    </van-cell-group>
    <van-cell-group>
      <van-field
        v-model="form.confirmPassword"
        label="确认密码："
        placeholder="请再次输入密码"
        clearable
        type="password"
```

```
          name="confirmPassword"
          :rules="comfirmPasswordRules"
        ></van-field>
      </van-cell-group>
      <van-cell-group>
        <van-button block round type="primary" native-type="submit">注册</va
n-button>
      </van-cell-group>
    </van-form>
    <div class="tip">注册成功后的用户可用于登录</div>
  </template>

  <script setup>
  import { ref, reactive } from 'vue'
  import { getUser, register } from '../api'
  import useToken from '../stores/token'
  import useUser from '../stores/user'
  import { useRouter } from 'vue-router'

  const { updateToken} = useToken()
  const { updateUser } = useUser()

  const form = reactive({
    username: '',
    password: '',
    confirmPassword: ''
  })
  const ruleFormRef = ref()
  const router = useRouter()

  // 定义验证规则
  const usernameRules = ref([
    { required: true, message: '用户名不能为空', trigger: 'onBlur' },
    { pattern: /^\w{3,16}$/, message: '用户名长度为 3-16 个字符', trigger: 'onB
lur' }
  ])
  const passwordRules = ref([
    { required: true, message: '密码不能为空', trigger: 'onBlur' },
```

```
      { pattern: /^\w{6,24}$/, message: '密码必须为 6-24 位英文字母或数字', trigge
r: 'onBlur' }
    ])
    const comfirmPasswordRules = ref([
      { required: true, message: '密码不能为空', trigger: 'onBlur' },
      { pattern: /^\w{6,24}$/, message: '密码必须为 6-24 位英文字母或数字', trigge
r: 'onBlur' },
      { validator: value => {
          if (value !== form.password) {
            return '两次输入的密码不一致'
          }
          return true
        }
      }
    ])

    // 表单提交函数
    const submitForm = async () => {
      const data = await register(form)
      if (data) {
        updateToken(data.token)
        const user = await getUser()
        updateUser({
          isLogin: true,
          username: user.username,
          avatar: user.avatar !== '' ? config.baseURL + '/' + user.avatar : ''
        })
        router.push({ name: 'user' })
      }
    }
    const onFailed = errorInfo => {
      console.log('failed', errorInfo)
    }
    </script>

    <style lang="less" scoped>
    button {
      position: fixed;
```

```
    top: 270px;
  }
  .tip {
    position: fixed;
    top: 330px;
    text-align: center;
    width: 100%;
    font-size: 14px;
    color: #666;
  }
  </style>
```

（4）查看注册页面效果，具体如下。

8.5 “我的”页开发

（1）创建 src\pages\User.vue，具体代码如下。

```
  <template>
    <!-- 已登录 -->
    <van-row v-if="user.isLogin" class="user-info">
      <van-image v-if="user.avatar" round width="100" height="100" src="{{
user.avatar }}" />
      <van-image v-else round width="100" height="100" :src="avatar_default
" />
      <span class="user-info-name">{{ user.username }}</span>
      <van-button plain type="danger" size="mini" @click="onLogout">退出</v
an-button>
```

```
    </van-row>
    <!-- 未登录 -->
    <van-row v-else class="user-info">
        <van-image round width="100" height="100" :src="avatar_default" />
        <router-link :to="{ name: 'login' }">
          <span class="user-info-name">登录  |</span>
        </router-link>
        <router-link :to="{ name: 'register' }">
          <span class="user-info-name">注册</span>
        </router-link>
    </van-row>
    <van-row class="user-links">
      <van-col span="6">
        <van-icon name="pending-payment" />
        待付款
      </van-col>
      <van-col span="6">
        <van-icon name="records" :badge="user.isLogin ? '7' : ''" />
        待收货
      </van-col>
      <van-col span="6">
        <van-icon name="tosend" :badge="user.isLogin ? '40' : ''" />
        待评价
      </van-col>
      <van-col span="6">
        <van-icon name="logistics" :badge="user.isLogin ? '1' : ''" />
        退换/售后
      </van-col>
    </van-row>
    <van-cell-group class="user-group my-title">
      <van-cell icon="records" title="全部订单" is-link />
    </van-cell-group>
    <van-cell-group  class="my-title">
      <van-cell icon="points" title="我的积分" is-link />
      <van-cell icon="gold-coin-o" title="我的优惠券" is-link />
      <van-cell icon="gift-o" title="我的红包" is-link />
    </van-cell-group>
  </template>
```

```
<script setup>
import avatar_default from '../assets/images/avatar_default.png'
import router from '../router/index'
import useToken from '../stores/token'
import useUser from '../stores/user'
import { showToast } from 'vant'

const { removeToken } = useToken()
const { user, removeUser } = useUser()

// 退出登录
const onLogout = async () => {
  removeToken()
  removeUser()
  router.push({ name: 'user' })
  showToast({
    message: '退出成功',
    type: 'success'
  })
}
</script>

<style lang="less" scoped>
.user-info {
  padding: 15px;
  background: url(../assets/images/user_head_bg.png) no-repeat;
  background-size: 100%;
}
.user-info button {
  margin: 40px 0 0 10px;
}
.user-info-name {
  display: inline-block;
  color: #fff;
  padding: 40px 0 0 10px;
  font-size: 20px;
}
```

```
:deep(.van-badge--top-right) {
  top: 4px;
  right: 35px;
  transform: translate(50%, -50%);
}
.user {
  &-group {
    margin-bottom: 15px;
  }
  &-links {
    padding: 15px 0;
    font-size: 12px;
    text-align: center;
    .van-icon {
      display: block;
      font-size: 24px;
    }
  }
}
</style>
```

（2）查看“我的”页面效果，具体如下。

（3）单击“登录”链接，进入登录页面，登录成功后页面效果，具体如下。

（4）退出登录后页面效果，具体如下。

（5）单击“注册”链接，跳转到注册页面，输入信息后，单击“注册”按钮，注册成功则直接登录，具体如下。

8.6 分类页开发

8.6.1 加载分类数据

（1）修改 src\api\index.js，调用分类接口，具体代码如下。

```
export function getCategoryList() {
  return request.get('/home/category/list')
}
```

（2）修改 src\pages\Category.vue，具体代码如下。

```
<script setup>
import { onMounted } from 'vue'
import { getCategoryList } from '../api'

onMounted(() => {
  loadCategoryList()
})

// 获取分类数据
const loadCategoryList = async () => {
  let data = await getCategoryList()
  console.log(data)
```

```
}
</script>
```

（3）在控制台打印接口返回的数据，具体如下。

```
▶ 0: {id: 1, name: '潮流女装', picture: 'http://127.0.0.1:8360/', pid: 0}
▶ 1: {id: 2, name: '羽绒服', picture: 'http://127.0.0.1:8360/static/image/category/clothes/jackets.png', pid: 1}
▶ 2: {id: 3, name: '毛呢大衣', picture: 'http://127.0.0.1:8360/static/image/category/clothes/overcoat.jpg', pid: 1}
▶ 3: {id: 4, name: '连衣裙', picture: 'http://127.0.0.1:8360/static/image/category/clothes/dress.png', pid: 1}
▶ 4: {id: 5, name: '食品', picture: 'http://127.0.0.1:8360/', pid: 0}
▶ 5: {id: 6, name: '休闲零食', picture: 'http://127.0.0.1:8360/static/image/category/foods/biscuit.jpg', pid: 5}
▶ 6: {id: 7, name: '生鲜果蔬', picture: 'http://127.0.0.1:8360/static/image/category/foods/tomato.jpg', pid: 5}
▶ 7: {id: 8, name: '饮料汽水', picture: 'http://127.0.0.1:8360/static/image/category/foods/drinks.jpg', pid: 5}
▶ 8: {id: 9, name: '四季茗茶', picture: 'http://127.0.0.1:8360/static/image/category/foods/tea.jpg', pid: 5}
▶ 9: {id: 10, name: '粮油调味', picture: 'http://127.0.0.1:8360/static/image/category/foods/oil.jpg', pid: 5}
▶ 10: {id: 11, name: '珠宝配饰', picture: 'http://127.0.0.1:8360/', pid: 0}
▶ 11: {id: 12, name: '时尚饰品', picture: 'http://127.0.0.1:8360/static/image/category/jewelry/ornaments.jpg', pid: 11}
▶ 12: {id: 13, name: '毛呢大衣', picture: 'http://127.0.0.1:8360/static/image/category/jewelry/watch.jpg', pid: 11}
▶ 13: {id: 14, name: '连衣裙', picture: 'http://127.0.0.1:8360/static/image/category/jewelry/diy.jpg', pid: 11}
▶ 14: {id: 15, name: '日用百货', picture: 'http://127.0.0.1:8360/', pid: 0}
▶ 15: {id: 16, name: '居家日用', picture: 'http://127.0.0.1:8360/static/image/category/store/towel.png', pid: 15}
▶ 16: {id: 17, name: '个人清洁', picture: 'http://127.0.0.1:8360/static/image/category/store/paper.png', pid: 15}
▶ 17: {id: 18, name: '盆碗碟筷', picture: 'http://127.0.0.1:8360/static/image/category/store/bowl.png', pid: 15}
▶ 18: {id: 19, name: '茶杯茶具', picture: 'http://127.0.0.1:8360/static/image/category/store/cup.png', pid: 15}
▶ 19: {id: 20, name: '收纳整理', picture: 'http://127.0.0.1:8360/static/image/category/store/box.png', pid: 15}
▶ 20: {id: 21, name: '手机数码', picture: 'http://127.0.0.1:8360/', pid: 0}
▶ 21: {id: 22, name: '手机', picture: 'http://127.0.0.1:8360/static/image/category/phone/phone.png', pid: 21}
▶ 22: {id: 23, name: '笔记本', picture: 'http://127.0.0.1:8360/static/image/category/phone/computer.png', pid: 21}
▶ 23: {id: 24, name: '数码配件', picture: 'http://127.0.0.1:8360/static/image/category/phone/headset.png', pid: 21}
▶ 24: {id: 25, name: '平板', picture: 'http://127.0.0.1:8360/static/image/category/phone/flat.png', pid: 21}
▶ 25: {id: 26, name: '相机', picture: 'http://127.0.0.1:8360/static/image/category/phone/camera.png', pid: 21}
▶ 26: {id: 27, name: '户外运动', picture: 'http://127.0.0.1:8360/', pid: 0}
▶ 27: {id: 28, name: '运动鞋', picture: 'http://127.0.0.1:8360/static/image/category/motion/shoes.jpg', pid: 27}
▶ 28: {id: 29, name: '球类运动', picture: 'http://127.0.0.1:8360/static/image/category/motion/ball.png', pid: 27}
▶ 29: {id: 30, name: '垂钓用品', picture: 'http://127.0.0.1:8360/static/image/category/motion/fishingrod.png', pid: 27}
▶ 30: {id: 31, name: '运动服', picture: 'http://127.0.0.1:8360/static/image/category/motion/clothes.png', pid: 27}
▶ 31: {id: 32, name: '骑行装备', picture: 'http://127.0.0.1:8360/static/image/category/motion/bicycle.png', pid: 27}
▶ 32: {id: 33, name: '电动车', picture: 'http://127.0.0.1:8360/static/image/category/motion/vehicle.png', pid: 27}
```

8.6.2 数组格式转换

（1）修改 src\pages\Category.vue，具体代码如下。

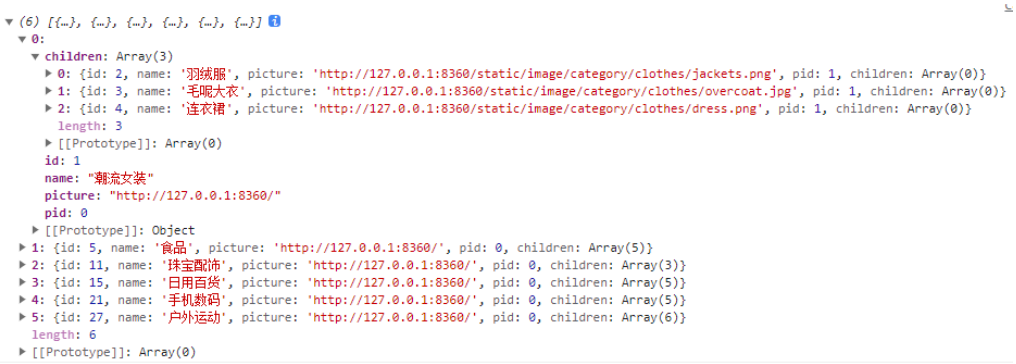
```
<script setup>
……（原有代码）
// 将一维数组转换成树形结构的方法
const convertToTree = data => {
  const treeData = []
  const map = {}
  // 遍历一维数组数据，建立节点映射表
  for (const item of data) {
    map[item.id] = { ...item, children: [] }
  }
  // 遍历映射表，将节点添加到父节点的 children 中
  for (const item of data) {
```

```
      const node = map[item.id]
      if (item.pid === 0) {
        treeData.push(node)
      } else {
        const parent = map[item.pid]
        parent.children.push(node)
      }
    }
    return treeData
  }
</script>
```

（2）将数据进行转换，具体代码如下。

```
const loadCategoryList = async () => {
  let data = await getCategoryList()
  // 将一维数组数据转换为树形结构
  const treeData = convertToTree(data)
  console.log(treeData)
}
```

（3）在控制台打印转换后的数据，具体如下。



8.6.3 显示分类数据

（1）修改 src\pages\Category.vue，具体代码如下。

```
<script setup>
import { onMounted, ref } from 'vue'
import { getCategoryList } from '../api'

const menus = ref([])
```

（2）将转换后的数据赋值给 menus 数组，具体代码如下。

```
const loadCategoryList = async () => {
  let data = await getCategoryList()
  // 将一维数组数据转换为树形结构
  const treeData = convertToTree(data)
  // 将转换后的数据赋值给 menus
  menus.value = treeData
}
```

（3）遍历 menus 数组，渲染页面，具体代码如下。

```
<template>
  <div class="menu">
    <div class="menu-left">
      <ul>
        <li class="menu-item" v-for="(menu, index) in menus" :key="index">
          <p class="text">{{ menu.name }}</p>
        </li>
      </ul>
    </div>
      <div class="menu-right">
      <!-- 显示二级分类 -->
      <ul>
        <li class="cate" v-for="(menu, index1) in menus" :key="index1">
          <h4 class="cate-title">{{ menu.name }}</h4>
          <ul class="cate-item">
            <li v-for="(item, index2) in menu.children" :key="index2">
              <router-link class="cate-item-wrapper" to="">
                <div class="cate-item-img">
                  <img :src="item.picture" alt="">
                </div>
                <span>{{ item.name }}</span>
              </router-link>
            </li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</template>
```

```
<style lang="less" scoped>
ul {
  margin: 0;
  padding: 0;
}
li {
  list-style: none;
}
.menu {
  display: flex;
  position: absolute;
  text-align: center;
  top: 46px;
  bottom: 50px;
  width: 100%;
  overflow: hidden;
  .menu-left {
    flex: 0 0 80px;
    width: 80px;
    background: #f3f5f7;
    line-height: 54px;
    .menu-item {
      height: 54px;
      width: 100%;
      border-bottom: 1px solid #e1e1e1;
      .text {
        width: 100%;
        margin: 0;
      }
    }
    .current {
      width: 100%;
      background: #fff;
      .text {
        color: red;
      }
    }
  }
```

```
.menu-right {
  flex: 1;
  background: #fff;
  .cate {
    height: 100%;
    .cate-title {
      margin: 0;
      text-align: left;
      font-size: 14px;
      color: #333;
      font-weight: bold;
      padding: 10px;
    }
    .cate-item {
      padding: 7px 10px 10px;
      display: flex;
      overflow: hidden;
      flex-flow: row wrap;
      li {
        width: 33.3%;
        .cate-item-wrapper {
          .cate-item-img {
            width: 100%;
            img {
              width: 70px;
              height: 70px;
            }
          }
          span {
            display: inline-block;
            font-size: 14px;
            color: #333;
          }
        }
      }
    }
  }
}
```

```
    }
    </style>
```

（4）查看数据渲染页面后的分类页，具体如下。

8.6.4 单击左侧菜单项获取右侧菜单对应位置

（1）修改 src\pages\Category.vue，给左侧菜单项添加一个单击事件，具体代码如下。

```
    <li class="menu-item" v-for="(menu, index) in menus" :key="index" @click
="clickList(index)">
```

（2）获取单击当前菜单项的索引值，具体代码如下。

```
    <script setup>
    ……（原有代码）
    // 单击左侧菜单项
```

```
const clickList = index => {
  console.log(index)
}
</script>
```

（3）定义一个 rightLiTops 数组，用于存储所有分类头部位置，具体代码如下。

```
const menus = ref([])
const rightLiTops = ref([])
```

（4）导入 watch 和 nextTick，调用 initRightHeight()监听右侧菜单列表高度，具体代码如下。

```
import { onMounted, watch, nextTick, ref } from 'vue'

onMounted(() => {
  loadCategoryList()
})
// 监听
watch(menus, () => {
  nextTick(() => {
    initRightHeight()
  })
})
```

（5）为右侧列表添加一个 ref 属性，属性值为 itemList，具体代码如下。

```
const menus = ref([])
const rightLiTops = ref([])
const itemList = ref()
```

为元素绑定一个 ref 属性，且属性值为 itemList，具体代码如下。

```
<div class="menu-right" ref="itemList">
```

（6）定义 initRightHeight()方法，具体代码如下。

```
<script setup>
……（原有代码）
// 初始化右边菜单的高度
const initRightHeight = () => {
  const itemArray = []
  let top = 0
  itemArray.push(top)
  const allList = itemList.value.getElementsByClassName('cate')
  Array.prototype.slice.call(allList).forEach(li => {
    top += li.clientHeight
    itemArray.push(top)
```

```
    })
    rightLiTops.value = itemArray
  }
</script>
```

（7）单击当前菜单项时，通过 index 索引会得到右侧菜单每一块<li>标签的高度，具体代码如下。

```
// 单击左侧菜单项
const clickList = index => {
  console.log(rightLiTops.value[index])
}
```

（8）在控制台打印，当单击左侧菜单项时输出对应的右侧菜单的高度，例如，0 表示单击左侧第一个菜单项时，对应的右侧菜单的高度，依次类推，具体如下。

8.6.5 初始化 better-scroll

（1）安装插件 better-scroll 插件，具体命令如下。

```
yarn add @better-scroll/core@2.5 --save
```

（2）在 Category.vue 中导入 better-scroll 插件，具体代码如下。

```
import BScroll from '@better-scroll/core'
```

（3）调用 initBScroll()，监听左右菜单的滚动，具体代码如下。

```
const rightLiTops = ref([])

let leftBscroll = null
let rightBscroll = null

// 监听
watch(menus, () => {
  nextTick(() => {
```

```
    initBScroll()
    initRightHeight()
  })
})
```

（4）定义 initBScroll()，初始化左菜单和右菜单，具体代码如下。

```
<script setup>
……（原有代码）
// 初始化 BScroll
const initBScroll = () => {
  // 初始化左菜单
  leftBscroll = new BScroll('.menu-left', {
    click: true,
    mouseWheel: true
  })
  // 初始化右菜单
  rightBscroll = new BScroll('.menu-right', {
    click: true,
    mouseWheel: true,
    probeType: 3 // 实时派发 scroll 事件
  })
}
</script>
```

（5）监听右侧滚动事件，当右侧菜单滚动的时候计算出滚动的距离，具体代码如下。

```
const scrollY = ref(0)  // 右侧列表滑动的 y 轴坐标

let rightBscroll = null


// 初始化 BScroll
const initBScroll = () => {
  ……（原有代码）
  rightBscroll.on('scroll', pos => {
    scrollY.value = Math.abs(pos.y)
  })
}
```

（6）单击左侧菜单项，右侧菜单滚动到相应位置，具体代码如下。

```
// 单击左侧菜单项
const clickList = index => {
```

```
  scrollY.value = rightLiTops.value[index]
  rightBscroll.scrollTo(0, -scrollY.value)
}
```

（7）单击左侧菜单项，右侧菜单滚动到相应位置，具体如下。

8.6.6 修复单击左侧底部菜单项页面跳转问题

（1）添加一个<li>标签，设置元素的高度，具体代码如下。

```
<script setup>
……（原有代码）
const RightHeightFix = () => {
  let bottom = itemList.value.getElementsByClassName('cate-bottom')[0]
  bottom.style.height = itemList.value.clientHeight / 1.2 + 'px'
}
</script>

<div class="menu-right" ref="itemList">
  <!-- 显示二级分类 -->
  <ul>
```

```
      <li class="cate" v-for="(menu, index1) in menus" :key="index1">
        <h4 class="cate-title">{{ menu.name }}</h4>
        <ul class="cate-item">
          <li v-for="(item, index2) in menu.children" :key="index2">
            <router-link class="cate-item-wrapper" to="">
              <div class="cate-item-img">
                <img :src="item.picture" alt="">
              </div>
              <span>{{ item.name }}</span>
            </router-link>
          </li>
        </ul>
      </li>
      <li class="cate-bottom"></li>
    </ul>
  </div>
```

（2）组件挂载后立即执行 RightHeightFix()，具体代码如下。

```
onMounted(() => {
  loadCategoryList()
  RightHeightFix()
})
```

（3）单击左侧菜单的最后一项时，右侧菜单可以滚动到相应的位置，具体如下。

8.6.7 右侧菜单滚动激活左侧菜单项

（1）右菜单滚动时，左菜单联动，具体代码如下。

```
const initLeftScroll = index => {
  const menu = menuList.value
  let el = menu[index]
  leftBscroll.scrollToElement(el, 300, 0, -100)
}
```

参数：scrollToElement(el, time, offsetX, offsetY, easing)，用于滚动到指定的目标元素。easing 缓动函数，一般不建议修改。

el：目标元素

300：滚动动画执行的时长，单位毫秒

0：相对于目标元素的横轴偏移量

-100：相对于目标元素的纵轴轴偏移量

（2）动态绑定 class 样式，激活左侧菜单项，具体代码如下。

```
const rightLiTops = ref([])
const menuList = ref()
```

```
<li class="menu-item" v-for="(menu, index) in menus" :key="index" :class="{ current: index === currentIndex }" @click="clickList(index)" ref="menuList">
```

（3）使用 computed 计算 currentIndex，具体代码如下。

```
import { onMounted, watch, nextTick, ref, computed } from 'vue'

<script setup>
……（原有代码）
const currentIndex = computed(() => {
  return rightLiTops.value.findIndex((top, index) => {
    if (index === rightLiTops.value.length - 2) {
      return true
    }
    if (scrollY.value >= top && scrollY.value < rightLiTops.value[index + 1]) {
      initLeftScroll(index)
      return true
    }
  })
```

```
})
</script>
```

8.7 商品列表页开发

8.7.1 单击分类跳转到商品列表页

（1）修改 src\router\index.js，添加商品列表页面的路由，具体代码如下。

```
routes: [
  ……（原有代码）
  {
    path: '/goodslist/:category_id',
    component: () => import('../pages/GoodsList.vue'),
    props: true,
    name: 'goodslist',
    meta: { title: '商品列表', isTab: true, isShowNav: true, isShowBack: tr
ue }
  },
]
```

（2）创建 src\pages\GoodsList.vue，具体代码如下。

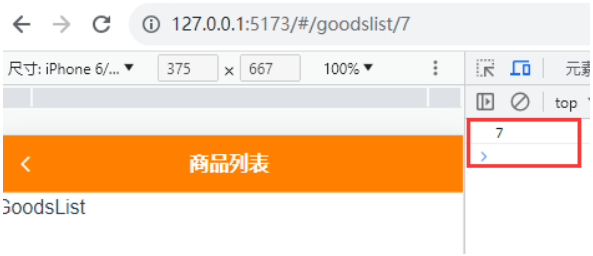
```
<template>
  GoodsList
</template>

<script setup>
import { onMounted } from 'vue'

const props = defineProps({
  category_id: String
})

onMounted(() => {
  loadGoodList()
})

const loadGoodList = async () => {
```

```
    console.log(props.category_id)
  }
  </script>
```

（3）修改 src\pages\Category.vue，具体代码如下。

```
  <router-link class="cate-item-wrapper" :to="{ name: 'goodslist', params:
{ category_id: item.id } }">
```

（4）测试能否接收到参数，具体如下。



8.7.2 加载商品列表数据

（1）修改 src\api\index.js，调用商品列表接口，具体代码如下。

```
// 商品列表接口
export function getGoodsList(params) {
  return request.get('/home/goods/list', { params })
}
```

（2）修改 src\pages\GoodsList.vue，具体代码如下。

```
import { onMounted, ref } from 'vue'
import { getGoodsList } from '../api'

const goodsList = ref([])
```

```
let last_id = '0'
```

（3）请求接口返回的数据，具体代码如下。

```
const loadGoodList = async () => {
  let params = {
    last_id,
    category_id: props.category_id,
    pagesize: 4
  }
  const data = await getGoodsList(params)
  if (data.length > 0) {
    goodsList.value = goodsList.value.concat(data)
    last_id = data[data.length - 1].id
  } else if (goodsList.value.length > 0) {
    // 已经到达底部
  } else {
    // 列表为空
  }
}
```

8.7.3 显示商品列表数据

（1）在页面中渲染数据，具体代码如下。

```
<template>
  <div class="goods-list">
    <div class="goods-item" v-for="item in goodsList" :key="item.id">
      <router-link to="">
        <van-image
          width="150"
          height="150"
          :src="item.picture"
        />
        <h1 class="title">{{ item.name }}<span class="small">{{ item.spec
}}</span></h1>
        <p class="info">
          <span class="price">¥{{ item.price }}</span>
          <span class="sell">剩余 {{ item.stock }} 件</span>
```

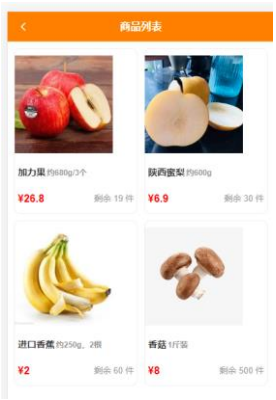
```
          </p>
        </router-link>
      </div>
    </div>
  </template>
```

（2）编写样式，具体代码如下。

```
<style lang="less" scoped>
.goods-list {
  display: flex;
  flex-wrap: wrap;
  padding-left: 10px;
  clear: both;
  .goods-item {
    width: calc(calc(100% / 2) - 12px);
    margin: 10px 10px 0 0;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    border-radius: 10px;
    border: 1px solid #efeff4;
    padding: 10px 0;
    clear: both;
    .title {
      text-align: left;
      font-size: 14px;
      color: #333;
      margin: 10px 0 0;
      padding: 0 5px;
      .small {
        font-size: 12px;
        padding-left: 2px;
        color: #999;
      }
    }
    .info {
      display: flex;
      justify-content: space-between;
      margin-bottom: 0;
```

```
        padding: 0 5px;
        .price {
          color: red;
          font-weight: bold;
          font-size: 16px;
        }
        .sell {
          font-size: 13px;
          color: #999;
        }
      }
    }
    .more {
      margin: 40px 20px 40px 10px;
      font-size: 14px;
    }
  }
  </style>
```

（3）查看商品列表，具体如下。



8.7.4 实现加载更多

（1）定义 is_last 默认为 false，具体代码如下。

```
const goodsList = ref([])
const is_last = ref(false)
```

（2）添加“加载更多”按钮，具体代码如下。

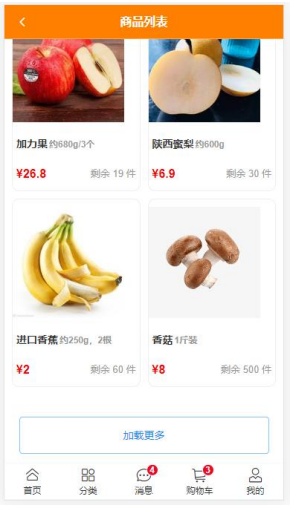
```
<div class="goods-list">
```

```
      ……（原有代码）
      <van-button class="more" :disabled="is_last" v-if="goodsList.length !
= 0" size="large" type="primary" plain hairline @click="getMore">加载更多</va
n-button>
    </div>
```

（3）定义 getMore 事件，具体代码如下。

```
    <script setup>
    ……（原有代码）
    const getMore = () => {
      loadGoodList()
    }
    </script>
```

（4）查看加载更多效果，具体如下。



（5）导入 showToast 组件，具体代码如下。

```
    import { showToast  } from 'vant'
```

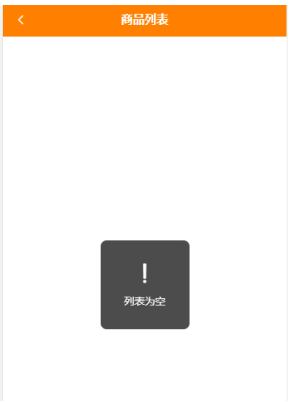
（6）根据数据的长度进行条件判断，给出不同的提示，具体代码如下。

```
    } else if (goodsList.value.length > 0) {
      showToast({
        message: '已经到达底部',
        type: 'fail'
      })
      is_last.value = true
    } else {
      showToast({
        message: '列表为空',
```

```
      type: 'fail'
    })
  }
```

（7）查看已经到达顶部的效果，具体如下。

（8）查看列表为空的效果，具体如下。



8.8 商品详情页开发

8.8.1 单击商品跳转到商品详情页

（1）修改 src\router\index.js，添加商品详情页的路由，具体代码如下。

```
routes: [
  ……（原有代码）
```

```
      {
        path: '/goodsDetail/:id',
        component: () => import('../pages/GoodsDetail.vue'),
        props: true,
        name: 'goodsDetail',
        meta: { title: '商品详情', isTab: false, isShowNav: true, isShowBack: t
rue }
      },
    }
```

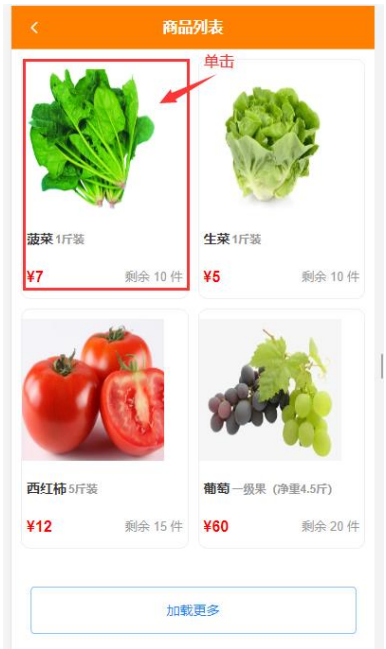
（2）创建 src\pages\GoodsDetail.vue，具体代码如下。

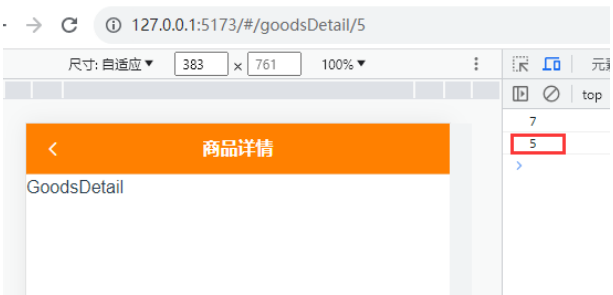
```
    <template>
      GoodsDetail
    </template>

    <script setup>
    import { onMounted } from 'vue'

    const props = defineProps({
      id: String
    })

    onMounted(() => {
      loadGoodsDetail()
    })

    // 加载商品详情
    const loadGoodsDetail = async () => {
      console.log(props.id)
    }
    </script>
```

（3）修改 src\pages\GoodsList.vue，具体代码如下。

```
    <router-link :to="{ name: 'goodsDetail', params: { id: item.id } }">
```

（4）测试能否接收到参数，单击商品，可以传递过去参数，具体如下。





8.8.2 加载商品详情数据

（1）修改 src\api\index.js，具体代码如下。

```
// 商品相册接口
export function getGoodsAlbum(params) {
  return request.get('/home/goods/album', { params })
}

// 商品详情接口
export function getGoodsDetail(params) {
  return request.get('/home/goods', { params })
}
```

（2）修改 src\pages\GoodsDetail.vue，具体代码如下。

```
import { reactive, ref, onMounted } from 'vue'
import { getGoodsAlbum, getGoodsDetail } from '../api'

const goods = reactive({})
const album = ref([])
const isNotFound = ref(false)
```

（3）加载商品详情数据和轮播图数据，具体代码如下。

```
// 加载商品详情
const loadGoodsDetail = async () => {
  const data1 = await getGoodsDetail({ id: props.id })
  if (!data1.id) {
    isNotFound.value = true
    return
  }
  const data2 = await getGoodsAlbum({ goods_id: props.id })
  if (data2.length === 0 && data1.picture !== '') {
    data2.push({ picture: data1.picture })
  }
  Object.assign(goods, data1)
  album.value = data2
}
```

8.8.3 显示商品详情数据

（1）修改 src\pages\GoodsDetail.vue，具体代码如下。

```
<template>
  <div class="goods" v-if="!isNotFound">
    <van-swipe class="goods-swipe" :autoplay="3000">
      <van-swipe-item v-for="item in album" :key="item.id">
        <img :src="item.picture" height="414">
      </van-swipe-item>
    </van-swipe>
    <van-cell-group>
      <van-cell>
        <template #title>
          <span class="goods-top">新品</span>
          <div class="goods-price">
            <span class="small">￥</span>
            {{ goods.price }}
            <span class="spec">{{ goods.spec }}</span>
          </div>
          <div class="goods-title">
            <span class="small"> {{ goods.name }}</span>
          </div>
```

```
        </template>
      </van-cell>
      <van-cell class="goods-express">
        <template #title>
          <van-col span="10">运费：10</van-col>
          <van-col span="14">剩余：{{ goods.stock }}</van-col>
        </template>
      </van-cell>
    </van-cell-group>
    <van-cell-group class="goods-cell-group">
      <van-cell>
        <template #title>
          <span class="van-cell-text">发货  陕西宝鸡</span>
        </template>
      </van-cell>
      <van-cell>
        <template #title>
          <span class="van-cell-text">保障  坏单包赔 • 假一赔四 • 极速退款</span>
        </template>
      </van-cell>
      <van-cell>
        <template #title>
          <span class="van-cell-text">参数  品牌：枝纯  价格：100-200</span>
        </template>
      </van-cell>
    </van-cell-group>
    <div class="goods-cell-title">
      —— 宝贝详情 ——
    </div>
    <div class="goods-description" v-html="goods.description"></div>
    <!-- 底部按钮-->
    <van-action-bar>
      <van-action-bar-icon icon="chat-o" text="客服" />
      <van-action-bar-icon icon="cart-o" text="购物车" />
      <van-action-bar-button type="warning" text="加入购物车" />
      <van-action-bar-button type="danger" text="立即购买" />
    </van-action-bar>
  </div>
```

```
    <div class="goods-not-found" v-else>商品不存在</div>
  </template>
```

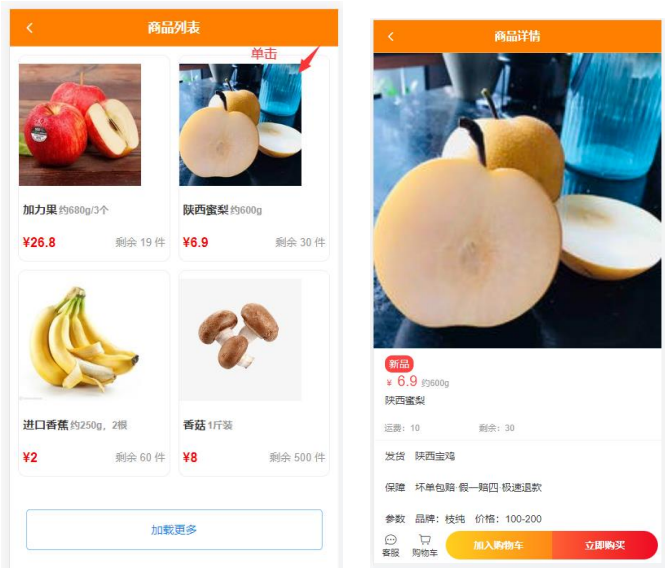
（2）编写样式，具体代码如下。

```
  <style lang="less" scoped>
  .goods {
    text-align: center;
    padding-bottom: 50px;
    .goods-swipe {
      img {
        width: 100%;
        display: block;
      }
    }
    .goods-top {
      display: block;
      width: 30px;
      padding: 0 5px;
      border-radius: 10px;
      color: #fff;
      background-color: #f44;
    }
    .goods-title {
      text-align: left;
      .small {
        font-size: 14px;
      }
    }
    .goods-price {
      color: #f44;
      text-align: left;
      font-size: 20px;
      .small {
        font-size: 12px;
      }
      .spec {
        font-size: 12px;
        color: #999;
      }
```

```
    }
    .goods-cell-title {
      padding: 20px 0;
    }
    .goods-description {
      width: 95%;
      margin: 0 auto;
      padding-bottom: 20px;
      font-size: 14px;
      :deep(img) {
        max-width: 100%;
        height: auto;
        display: block;
      }
    }
    &-express {
      color: #999;
      font-size: 12px;
      padding: 5px 15px;
      :deep(.van-col) {
        float: left;
      }
      :deep(.van-col--14){
        width: 58%;
      }
    }
   .goods-cell-group {
      :deep(.van-cell__title span){
        float: left;
      }
    }
    :deep(.van-cell:after) {
      border: none;
    }
  }
  .goods-not-found {
    padding-top: 48px;
    text-align: center;
```

```
    font-size: 28px;
  }
  </style>
```

（3）查看商品详情页，具体如下。



8.8.4 完成底部按钮功能

（1）修改 src\pages\GoodsDetail.vue，添加底部按钮区域，具体代码如下。

```
  <van-action-bar-icon icon="chat-o" text="客服" @click="sorry" />
  <van-action-bar-icon icon="cart-o" text="购物车" @click="onClickCart" />
  <van-action-bar-button type="warning" text="加入购物车" @click="addCart"
/>
  <van-action-bar-button type="danger" text="立即购买" @click="sorry" />
```

（2）导入路由和 showToast，具体代码如下。

```
  import { showToast } from 'vant'
  import { useRouter } from 'vue-router'

  const router = useRouter();
```

（3）处理逻辑，具体代码如下。

```
  <script setup>
  ……（原有代码）
  const sorry = () => {
    showToast('暂无后续逻辑~')
```

```
}

const onClickCart = () => {
  router.push({ name: 'cart' })
}

const addCart = () => {
  showToast('暂无后续逻辑~')
}
</script>
```

（4）单击“购物车”跳转到购物车页面，具体如下。

8.9 购物车页开发

8.9.1 添加到购物车

（1）创建 src\store\cart.js，具体代码如下。

```
import { defineStore } from 'pinia'
import { ref } from 'vue'

const useCart = defineStore('cart', () => {
  const cart = ref([])
  const addToCart = goods => {
    const item = cart.value.find(item => goods.id == item.id)
    if (item) {
      item.num += goods.num
    } else {
      cart.value.push(goods)
```

```
      }
    }
    const removeCart = id => {
      cart.value.forEach((item, index) => {
        if (item.id == id) {
          cart.value.splice(index, 1)
        }
      })
    }
    const cartCount = () => {
      let sum = 0
      cart.value.forEach(item => {
        sum += item.num
      })
      return sum || undefined
    }
    return { cart, addToCart, removeCart, cartCount }
  }, {
    persist: {
      enabled: true,
      strategies: [
        {
          key: 'cart',
          storage: localStorage
        }
      ]
    }
  })
  export default useCart
```

（2）底部导航栏显示购物车中的商品数量

修改 src\components\TabBar.vue，具体代码如下。

```
  <van-tabbar-item replace :to="{ name: 'cart' }" icon="shopping-cart-o" :
badge="cartCount()">购物车</van-tabbar-item>
```

```
  <script setup>
  import useCart from '../stores/cart'

  const { cartCount } = useCart()
```

```
</script>
```

底部导航栏的购物车中商品数量效果，具体如下。

（3）商品详情页显示购物车中的商品数量

修改 src\pages\GoodsDetail.vue，具体代码如下。

```
import useCart from '../stores/cart'


const { cartCount, addToCart } = useCart()
```

绑定 cartCount()，具体代码如下。

```
<van-action-bar-icon icon="cart-o" :badge="cartCount()" @click="onClickCart" text="购物车" />
```

（4）添加到购物车，具体代码如下。

```
const addCart = () => {
  addToCart({ id: props.id, num: 1, checked: true })
  showToast({
    message: '添加成功'
  })
}
```

（5）查看添加购物车效果，具体如下。

8.9.2 加载购物车数据

（1）修改 src\api\index.js，具体代码如下。

```
// 购物车接口
export function getCartList(params) {
  return request.get('/home/goods/cart', { params })
}
```

（2）修改 src\pages\Cart.vue，具体代码如下。

```
<script setup>
import { ref, onMounted } from 'vue'
import { getCartList } from '../api'
import useCart from '../stores/cart'

const { cart } = useCart()

const goodsList = ref([])

onMounted(() => {
```

```
    loadCart()
    console.log(goodsList)
  })

  // 加载购物车数据
  const loadCart = async () => {
    const ids = cart.map(item => item.id)
    goodsList.value = await getCartList({ ids: ids.join(',') })
    goodsList.value.forEach(goods => {
      goods.cart = cart.find(item => goods.id == item.id)
    })
  }
  </script>
```

（3）购物车数据，具体如下。

8.9.3 显示购物车页面

（1）导入空购物车图片，具体代码如下。

```
import cartEmptyImage from '../assets/images/cart_empty.png'
```

（2）渲染页面数据，具体代码如下。

```
<template>
  <div class="cart">
    <div class="cart-container">
      <van-empty v-show="goodsList.length == 0" description="购物车目前还没有商品" :image="cartEmptyImage">
        <router-link :to="{ name: 'category' }">
          <van-button round type="danger" class="bottom-button">去购物</van-button>
        </router-link>
      </van-empty>
      <!-- 购物车列表 -->
```

```
        <div v-for="item in goodsList" :key="item.id" class="list">
          <van-swipe-cell>
            <!-- 复选框 -->
            <div class="checkbox">
              <van-checkbox :name="item" v-model="item.cart.checked" checked
-color="#f11a27"></van-checkbox>
            </div>
            <!-- 商品图片 -->
            <div class="image">
              <router-link :to="{ name: 'goodsDetail', params: { id: item.id
 } }">
                <van-image width="50" height="50" :src="item.picture" />
              </router-link>
            </div>
            <!-- 商品信息 -->
            <div class="goods-info">
              <div>{{ item.name }}</div>
              <div class="bottom">
                <div class="price"><span>￥</span>{{ item.price }}</div>
                <van-stepper  v-model="item.cart.num" theme="round" button-s
ize="22" disable-input />
              </div>
            </div>
            <!-- 左滑删除 -->
            <template #right>
              <van-button aquare icon="delete" type="danger" class="delete-b
utton" />
            </template>
          </van-swipe-cell>
        </div>
      </div>
    </div>
  </template>
```

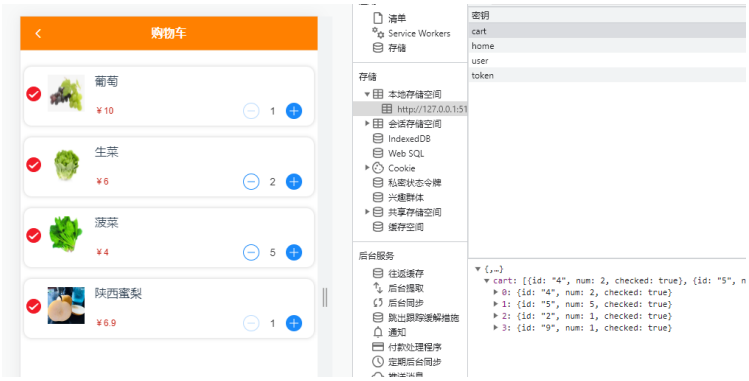
（3）编写样式代码，具体代码如下。

```
  <style lang="less" scoped>
  .cart {
    margin: 0.3rem;
    padding: .05rem 0 3rem 0;
```

```
.cart-container {
  margin-top: 1rem;
  .list {
    position: relative;
    height: 5rem;
    border-radius: 10px;
    box-shadow: 0px 0px 5px #ccc;
    margin-bottom: 1rem;
    .checkbox {
      position: absolute;
      top: 1.7rem;
      left: .2rem;
    }
    .image {
      position: absolute;
      top: .7rem;
      left: 2rem;
    }
    .goods-info {
      height: 5rem;
      display: flex;
      justify-content: space-around;
      flex-direction: column;
      padding: 0 1rem 0 6rem;
      .bottom {
        display: flex;
        justify-content: space-between;
        align-items: center;
        .price {
          color: #c82519;
          font-size: .45rem;
        }
      }
    }
    .delete-button {
      width: 2rem;
      height: 100%;
    }
```

```
      }
      .bottom-button {
        width: 7rem;
        height: 2rem;
      }
    }
    .settlement {
      margin-bottom: -0.1rem;
    }
  }
  :deep(.van-submit-bar) {
    bottom: 3.5rem;
  }
  :deep(.van-swipe-cell) {
    width: 100%;
  }
  </style>
```

（4）查看购物车数据，具体如下。



8.9.4 删除（左滑）购物车中的商品

（1）修改 src\pages\Cart.vue，具体代码如下。

```
  <van-button aquare icon="delete" type="danger" class="delete-button" @cl
ick="onDelete(item)" />
```

（2）导入 removeCart，具体代码如下。

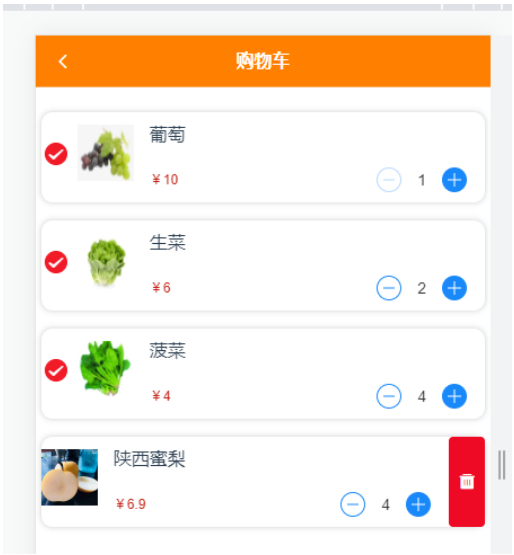
```
  const { cart, removeCart } = useCart()
```

（3）处理删除逻辑，具体代码如下。

```
  <script setup>
```

```
    ……（原有代码）
    // 删除商品
    const onDelete = item => {
      const id = item.id
      goodsList.value.forEach((item, index) => {
        if (item.id == id) {
          goodsList.value.splice(index, 1)
        }
      })
      removeCart(id)
    }
    </script>
```

（4）测试是否可以删除成功，左滑“陕西蜜梨”，将其删除，具体如下。



8.9.5 实现购物车结算功能

（1）实现结算部分的结构，具体代码如下。

```
    <div class="cart">
      ……（原有代码）
      <!-- 结算 -->
      <van-submit-bar v-show="goodsList.length" :price="total * 100" button-
text="去结算" button-type="primary" @submit="onSubmit" class="settlement">
        <van-checkbox v-model="allChecked" checked-color="#f11a27" @click="o
```

```
nCheckAll">全选</van-checkbox>
      </van-submit-bar>
    </div>
```

（2）定义 allChecked 初始为 false，具体代码如下。

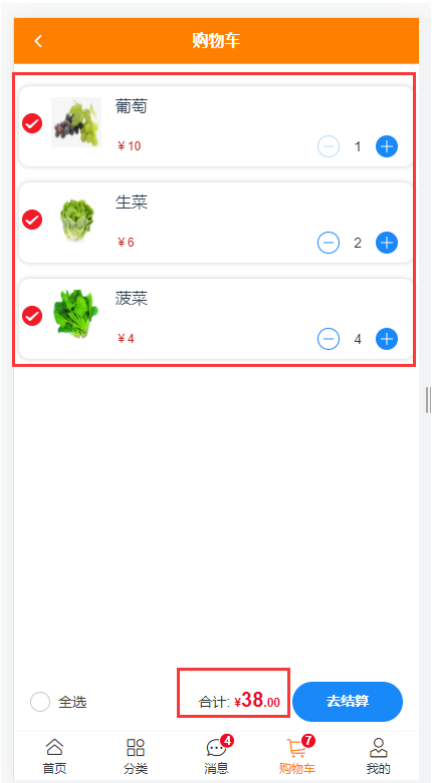
```
const goodsList = ref([])
const allChecked = ref(false)
```

（3）查看结算区域的效果，具体如下。

（4）实现“总金额”和“全选”按钮的逻辑，具体代码如下。

```
import { ref, onMounted, computed } from 'vue'

// 总金额
const total = computed(() => {
  let sum = 0
  goodsList.value.forEach(item => {
    if (item.cart.checked) {
      sum += item.price * item.cart.num
    }
  })
  return sum
})

// 全选
const onCheckAll = () => {
  goodsList.value.forEach(el => {
    el.cart.checked = allChecked.value
  })
}
```

（5）查看合计处的金额是否正确，具体如下。



（6）实现“去结算”逻辑，具体代码如下。

```
import { showToast } from 'vant'

// 去结算
const onSubmit = () => {
  showToast('暂无后续逻辑~')
}
```

（7）商品都被选中时，全选按钮高亮，具体代码如下。

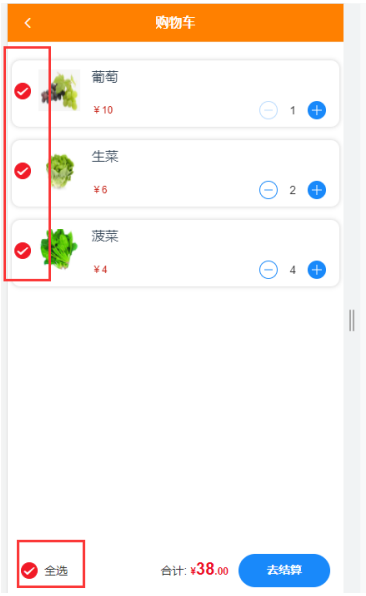
```
<van-checkbox :name="item" v-model="item.cart.checked" @change="onCheck"
checked-color="#f11a27"></van-checkbox>

// 商品都被选中时，全选按钮高亮
const onCheck = () => {
  allChecked.value = goodsList.value.every(el => el.cart.checked)
}
```

（8）页面打开后自动全选，具体代码如下。

```
onMounted(async () => {
  await loadCart()
  onCheck()
```

```
})
```

（9）商品全部选中时，全选按钮勾选，具体如下。



（10）商品非全选状态时，全选按钮不勾选，具体如下。