## 前言

本章节我们进入了前端部分的尾声，还剩下「我的订单」页面的制作，订单页面分几个状态，分别是全部、待付款、待确认、待发货、已发货、交易完成几个状态。

所以顶部是一个可拖动的 tab-bar ，页面需要做成上拉加载更多的翻页形式，下面我们来制作订单页面。

## 我的订单页面制作

在此之前，我们需要把上一章节的购物车创建订单流程跑通，我们先来添加订单相关的接口，在 service 文件夹下新建 order.js 文件，内容如下：

**import** axios **from** '../utils/axios'

*// 创建订单***export** **function** **createOrder**(params) {

**return** axios.**post**('/saveOrder', params);

}

*// 获取订单列表***export** **function** **getOrderList**(params) {

**return** axios.**get**('/order', { params });

}

*// 根据 id 获取订单详情***export** **function** **getOrderDetail**(id) {

**return** axios.**get**(`/order/${id}`);

}

*// 取消订单***export** **function** **cancelOrder**(id) {

**return** axios.**put**(`/order/${id}/cancel`);

}

*// 确认订单***export** **function** **confirmOrder**(id) {

**return** axios.**put**(`/order/${id}/finish`)

}

*// 支付接口***export** **function** **payOrder**(params) {

**return** axios.**get**('/paySuccess', { params })

}

我们来收尾之前购物车流程，点击「生成订单」按钮，调用 createOrder 接口，跳转至「我的订单」页面，代码如下：

<template>

<div class="create-order">

<s-header :name="'生成订单'" @callback="deleteLocal"></s-header>

<div class="address-wrap">

<div class="name" @click="goTo">

<span>{{ address.userName }} </span>

<span>{{ address.userPhone }}</span>

</div>

<div class="address">

{{ address.provinceName }} {{ address.cityName }} {{ address.regionName }} {{ address.detailAddress }}

</div>

<van-icon class="arrow" name="arrow" />

</div>

<div class="good">

<div class="good-item" v-for="(item, index) in cartList" :key="index">

<div class="good-img"><img :src="`//lmall.xinfeng.site${item.goodsCoverImg}`" alt=""></div>

<div class="good-desc">

<div class="good-title">

<span>{{ item.goodsName }}</span>

<span>x{{ item.goodsCount }}</span>

</div>

<div class="good-btn">

<div class="price">¥{{ item.sellingPrice }}</div>

</div>

</div>

</div>

</div>

<div class="pay-wrap">

<div class="price">

<span>商品金额</span>

<span>¥{{ total }}</span>

</div>

<van-button @click="createOrder" class="pay-btn" color="#1baeae" type="primary" block>生成订单</van-button>

</div>

<van-popup

closeable

:close-on-click-overlay="false"

v-model="showPay"

position="bottom"

:style="{ height: '30%' }"

@close="close"

>

<div :style="{ width: '90%', margin: '0 auto', padding: '50px 0' }">

<van-button :style="{ marginBottom: '10px' }" color="#1989fa" block @click="payOrder(1)">支付宝支付</van-button>

<van-button color="#4fc08d" block @click="payOrder(2)">微信支付</van-button>

</div>

</van-popup>

</div>

</template>

<script>

import sHeader from '@/components/SimpleHeader'

import { getCart, getByCartItemIds } from '../service/cart'

import { getDefaultAddress, getAddressDetail } from '../service/address'

import { createOrder, payOrder } from '../service/order'

import { setLocal, getLocal } from '@/common/js/utils'

import { Toast } from 'vant'

export default {

components: {

sHeader

},

data() {

return {

cartList: [],

address: {},

showPay: false,

orderNo: '',

cartItemIds: []

}

},

mounted() {

this.init()

},

methods: {

async init() {

Toast.loading({ message: '加载中...', forbidClick: true });

const { addressId, cartItemIds } = this.$route.query

const \_cartItemIds = cartItemIds ? JSON.parse(cartItemIds) : JSON.parse(getLocal('cartItemIds'))

setLocal('cartItemIds', JSON.stringify(\_cartItemIds))

const { data: list } = await getByCartItemIds({ cartItemIds: \_cartItemIds.join(',') })

const { data: address } = addressId ? await getAddressDetail(addressId) : await getDefaultAddress()

if (!address) {

this.$router.push({ path: 'address' })

return

}

this.cartList = list

this.address = address

Toast.clear()

},

goTo() {

this.$router.push({ path: `address?cartItemIds=${JSON.stringify(this.cartItemIds)}` })

},

deleteLocal() {

setLocal('cartItemIds', '')

},

async createOrder() {

const params = {

addressId: this.address.addressId,

cartItemIds: this.cartList.map(item => item.cartItemId)

}

const { data, resultCode } = await createOrder(params)

setLocal('cartItemIds', '')

this.orderNo = data

this.showPay = true

},

close() {

this.$router.push({ path: 'order' })

},

async payOrder(type) {

Toast.loading

await payOrder({ orderNo: this.orderNo, payType: type })

this.$router.push({ path: 'order' })

}

},

computed: {

total: function() {

let sum = 0

this.cartList.forEach(item => {

sum += item.goodsCount \* item.sellingPrice

})

return sum

}

}

}

</script>

<style lang="less" scoped>

@import '../common/style/mixin';

.create-order {

background: #f9f9f9;

.address-wrap {

margin-bottom: 20px;

background: #fff;

position: relative;

margin-top: 44px;

font-size: 14px;

padding: 15px;

color: #222333;

.name, .address {

margin: 10px 0;

}

.arrow {

position: absolute;

right: 10px;

top: 50%;

transform: translateY(-50%);

font-size: 20px;

}

&::before {

position: absolute;

right: 0;

bottom: 0;

left: 0;

height: 2px;

background: -webkit-repeating-linear-gradient(135deg, #ff6c6c 0, #ff6c6c 20%, transparent 0, transparent 25%, #1989fa 0, #1989fa 45%, transparent 0, transparent 50%);

background: repeating-linear-gradient(-45deg, #ff6c6c 0, #ff6c6c 20%, transparent 0, transparent 25%, #1989fa 0, #1989fa 45%, transparent 0, transparent 50%);

background-size: 80px;

content: '';

}

}

.good {

margin-bottom: 120px;

}

.good-item {

padding: 10px;

background: #fff;

display: flex;

.good-img {

img {

.wh(100px, 100px)

}

}

.good-desc {

display: flex;

flex-direction: column;

justify-content: space-between;

flex: 1;

padding: 20px;

.good-title {

display: flex;

justify-content: space-between;

}

.good-btn {

display: flex;

justify-content: space-between;

.price {

font-size: 16px;

color: red;

line-height: 28px;

}

.van-icon-delete {

font-size: 20px;

margin-top: 4px;

}

}

}

}

.pay-wrap {

position: fixed;

bottom: 0;

left: 0;

width: 100%;

background: #fff;

padding: 10px 0;

padding-bottom: 50px;

border-top: 1px solid #e9e9e9;

>div {

display: flex;

justify-content: space-between;

padding: 0 5%;

margin: 10px 0;

font-size: 14px;

span:nth-child(2) {

color: red;

font-size: 18px;

}

}

.pay-btn {

position: fixed;

bottom: 7px;

right: 0;

left: 0;

width: 90%;

margin: 0 auto;

}

}

}

</style>

init() 方法内，我们首先拿到浏览器地址传进来的 cartItemIds 参数，此参数为购物车页面生成的购物车商品 id，通过 JSON.parse(cartItemIds) 将字符串转化为对象形式，再将购物车存到本地，因为某些操作可能会导致浏览器地址参数丢失，所以如果取不到购物车参数的话，通过 JSON.parse(getLocal('cartItemIds')) 去本地获取购物车参数。

再通过 getByCartItemIds 方法拿到购物车商品列表，以及通过 addressId 拿到对应的地址数据用于展示该笔订单的详情信息。

点击生成订单按钮，调用 createOrder 接口之后，弹出底部弹窗进行支付操作，目前支付接口只需传生成的订单 id 以及支付方式，便能完成支付。完成整个购买流程之后，调转到「我的订单」页面。

接下来我们需要新建「我的订单」页面，在 views 目录下新建文件 Order.vue ，代码如下所示：

<template>

<div class="order-box">

<s-header :title="'我的订单'" :back="'/user'"></s-header>

<van-tabs @change="onChangeTab" :color="'#1baeae'" :title-active-color="'#1baeae'" class="order-tab" v-model="status">

<van-tab title="全部" name=''></van-tab>

<van-tab title="待付款" name="0"></van-tab>

<van-tab title="待确认" name="1"></van-tab>

<van-tab title="待发货" name="2"></van-tab>

<van-tab title="已发货" name="3"></van-tab>

<van-tab title="交易完成" name="4"></van-tab>

</van-tabs>

<van-pull-refresh v-model="refreshing" @refresh="onRefresh" class="order-list-refresh">

<van-list

v-model="loading"

:finished="finished"

finished-text="没有更多了"

@load="onLoad"

@offset="300"

>

<div v-for="(item, index) in list" :key="index" class="order-item-box" @click="goTo(item.orderNo)">

<div class="order-item-header">

<span>订单时间：{{ item.createTime }}</span>

<span>{{ item.orderStatusString }}</span>

</div>

<van-card

v-for="one in item.louMallOrderItemVOS"

:key="one.orderId"

:num="one.goodsCount"

:price="one.sellingPrice"

desc="全场包邮"

:title="one.goodsName"

:thumb="`http://lmall.xinfeng.site${one.goodsCoverImg}`"

/>

</div>

</van-list>

</van-pull-refresh>

</div>

</template>

<script>

import sHeader from '@/components/SimpleHeader'

import { getOrderList } from '../service/order'

export default {

data() {

return {

status: '',

loading: false,

finished: false,

refreshing: false,

list: [],

page: 1

}

},

components: {

sHeader

},

async mounted() {

// this.loadData()

},

methods: {

async loadData() {

// 获取订单列表

const { data, data: { list } } = await getOrderList({ pageNumber: this.page, status: this.status })

this.list = this.list.concat(list)

this.totalPage = data.totalPage

this.loading = false;

if (this.page >= data.totalPage) this.finished = true

},

onChangeTab(name, title) {

this.status = name

this.onRefresh()

},

goTo(id) {

this.$router.push({ path: `order-detail?id=${id}` })

},

goBack() {

this.$router.go(-1)

},

onLoad() {

// 加载更多

if (!this.refreshing && this.page < this.totalPage) {

this.page = this.page + 1

}

if (this.refreshing) {

this.list = [];

this.refreshing = false;

}

this.loadData()

},

onRefresh() {

this.refreshing = true

this.finished = false

this.loading = true

this.page = 1

this.onLoad()

},

}

}

</script>

<style lang="less" scoped>

@import '../common/style/mixin';

.order-box {

.order-header {

position: fixed;

top: 0;

left: 0;

z-index: 10000;

.fj();

.wh(100%, 44px);

line-height: 44px;

padding: 0 10px;

.boxSizing();

color: #252525;

background: #fff;

border-bottom: 1px solid #dcdcdc;

.order-name {

font-size: 14px;

}

}

.order-tab {

margin-top: 44px;

position: fixed;

left: 0;

z-index: 1000;

width: 100%;

}

.order-list-refresh {

margin-top: 68px;

.van-card\_\_content {

display: flex;

flex-direction: column;

justify-content: center;

}

.van-pull-refresh\_\_head {

background: #f9f9f9;

}

.van-list {

min-height: calc(100vh - 88px);

background: #f9f9f9;

margin-top: 20px;

}

.order-item-box {

margin: 20px 10px;

background-color: #fff;

.order-item-header {

padding: 10px 20px 0 20px;

display: flex;

justify-content: space-between;

}

.van-card {

background-color: #fff;

margin-top: 0;

}

}

}

}

</style>

这里再细说一下加载更多的处理方法，我采用的是 Vant 组件库自带的 van-pull-refresh 组件和 van-list 组件，当页面下拉触发 van-pull-refresh 组件的 refresh 事件时，重新获取列表数据；当页面下滑至滚动条与底部距离小于 offset 时，便会触发 onLoad 事件，加载下一页的数据拼接在之前的数据后面。

浏览器展示效果如下图所示：



Card 组件需要去 main.js 文件下全局注册，否则代码会报错，vue-cli 对项目报错非常友好，基本上一眼能看出是什么错，遇到组件没有添加的情况，同学们可以去查找一下，进行全局注册便可。

## 完善用户信息设置页面

修改个人信息页面以及退出登录等操作，我们统一把入口放在「我的」页面的「账号管理」，我们在 src/views 目录下新建文件 Setting.vue ，代码如下：

<template>

<div class="seting-box">

<s-header :name="'账号管理'"></s-header>

<div class="input-item">

<van-field v-model="nickName" label="昵称" />

<van-field v-model="introduceSign" label="个性签名" />

<van-field v-model="password" type='password' label="修改密码" />

</div>

<van-button class="save-btn" color="#1baeae" type="primary" @click="save" block>保存</van-button>

<van-button class="save-btn" color="#1baeae" type="primary" @click="logout" block>退出登录</van-button>

</div>

</template>

<script>

import sHeader from '@/components/SimpleHeader'

import { getUserInfo, EditUserInfo, logout } from '../service/user'

import { setLocal } from '@/common/js/utils'

import { Toast } from 'vant'

export default {

components: {

sHeader

},

data() {

return {

nickName: '',

introduceSign: '',

password: ''

}

},

async mounted() {

const { data } = await getUserInfo()

this.nickName = data.nickName

this.introduceSign = data.introduceSign

},

methods: {

async save() {

const params = {

introduceSign: this.introduceSign,

nickName: this.nickName,

passwordMd5: this.$md5(this.password)

}

const { data } = await EditUserInfo(params)

Toast.success('保存成功')

},

async logout() {

const { resultCode } = await logout()

if (resultCode == 200) {

setLocal('token', '')

window.location.href = '/'

}

}

}

}

</script>

<style lang="less" scoped>

.seting-box {

.input-item {

margin-top: 44px;

}

.save-btn {

width: 80%;

margin: 20px auto ;

}

}

</style>

在 services/user.js 下添加编辑用户信息接口：

**export** **function** **EditUserInfo**(params) {

**return** axios.**put**('/user/info', params);

}

修改 User.vue 页面，添加跳转方法：

<template>

<div class="user-box">

<s-header :name="'我的'"></s-header>

<div class="user-info">

<div class="info">

<img src="//s.weituibao.com/1583583975067/user-graduate%20(1).png"/>

<div class="user-desc">

<span>昵称：{{ user.nickName }}</span>

<span>登录名：{{ user.loginName }}</span>

<span class="name">个性签名：{{ user.introduceSign }}</span>

</div>

</div>

</div>

<ul class="user-list">

<li @click="goTo('order')">

<span>我的订单</span>

<van-icon name="arrow" />

</li>

<li @click="goTo('setting')">

<span>账号管理</span>

<van-icon name="arrow" />

</li>

<li @click="goTo('address?from=mine')">

<span>地址管理</span>

<van-icon name="arrow" />

</li>

<li @click="goTo('about')">

<span>关于我们</span>

<van-icon name="arrow" />

</li>

</ul>

<nav-bar></nav-bar>

</div>

</template>

<script>

import navBar from '@/components/NavBar'

import sHeader from '@/components/SimpleHeader'

import { getUserInfo } from '../service/user'

export default {

components: {

navBar,

sHeader

},

data() {

return {

user: {}

}

},

async mounted() {

const { data } = await getUserInfo()

this.user = data

},

methods: {

goBack() {

this.$router.go(-1)

},

goTo(r) {

this.$router.push({ path: r })

}

}

}

</script>

<style lang="less" scoped>

@import '../common/style/mixin';

.user-box {

.user-header {

position: fixed;

top: 0;

left: 0;

z-index: 10000;

.fj();

.wh(100%, 44px);

line-height: 44px;

padding: 0 10px;

.boxSizing();

color: #252525;

background: #fff;

border-bottom: 1px solid #dcdcdc;

.user-name {

font-size: 14px;

}

}

.user-info {

width: 94%;

margin: 10px;

height: 115px;

background: linear-gradient(90deg, @primary, #f19f7e);

box-shadow: 0 2px 5px @primary;

border-radius: 6px;

margin-top: 50px;

.info {

position: relative;

display: flex;

width: 100%;

height: 100%;

padding: 25px 20px;

.boxSizing();

img {

.wh(60px, 60px);

border-radius: 50%;

margin-top: 4px;

}

.user-desc {

display: flex;

flex-direction: column;

margin-left: 10px;

line-height: 20px;

font-size: 14px;

color: #fff;

span {

color: #fff;

font-size: 14px;

padding: 2px 0;

}

}

.account-setting {

position: absolute;

top: 10px;

right: 20px;

font-size: 13px;

color: #fff;

.van-icon-setting-o {

font-size: 16px;

vertical-align: -3px;

margin-right: 4px;

}

}

}

}

.user-list {

padding: 0 20px;

margin-top: 20px;

li {

height: 40px;

line-height: 40px;

border-bottom: 1px solid #e9e9e9;

display: flex;

justify-content: space-between;

font-size: 14px;

.van-icon-arrow {

margin-top: 13px;

}

}

}

}

</style>

浏览器展示效果如下图所示：



修改个人信息、密码、个性签名、退出登录，都在这个页面完成。

注意，新建页面的同时，要前往 router.js 添加相应的路由，否则跳转路由会找不到相对应的页面组件。

## 总结

行文至此，前端部分基本接近尾声，还有一些琐碎的页面，比如关于我们、联系客服等大家自行在源码中查阅，都是些静态页面，没有具体的功能，在此不赘述，如果在编码过程中遇到问题，建议大家对照源码边学边看，源码是可以正常运行的，若是有 bug 大家可以到 Github 仓库里给我提意见。