

第六天

订单模块 + 项目上线



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

填写订单 - 渲染基本信息

静态结构(分包)



封装请求API



初始化调用



类型声明



界面渲染

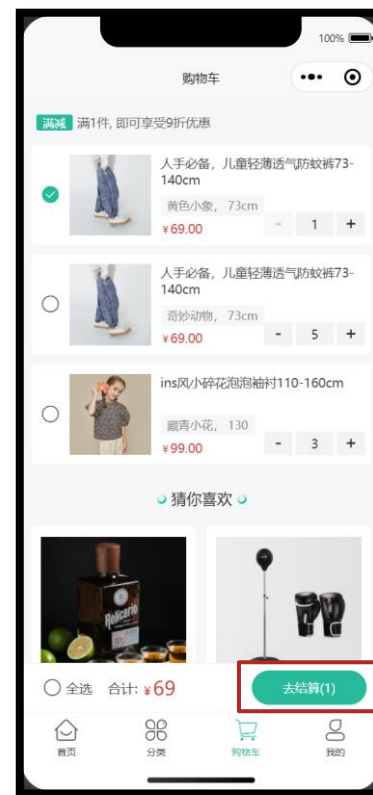
```
/**
 * 填写订单-获取预付订单
 */
export const getMemberOrderPreAPI = () => {
  return http<OrderPreResult>({
    method: 'GET',
    url: '/member/order/pre',
  })
}
```

```
/** 获取预付订单 返回信息 */
export type OrderPreResult = {
  /** 商品集合 [ 商品信息 ] */
  goods: OrderPreGoods[]
  summary: {
    totalPrice: number
    postFee: number
    totalPayPrice: number
  }
  /** 用户地址列表 [ 地址信息 ] */
  userAddresses: AddressItem[]
}
```

```
// 获取预付订单
const orderPre = ref<OrderPreResult>()
const getMemberOrderPreData = async () => {
  const res = await getMemberOrderPreAPI()
  orderPre.value = res.result
}

onLoad(() => {
  getMemberOrderPreData()
})
```

```
<!-- 商品信息 -->
<view class="goods">
  <navigator
    v-for="item in orderPre?.goods"
    :key="item.skuId"
    :url="`/pages/goods/goods?id=${item.id}`"
    class="item"
  >
    ...省略
  </navigator>
</view>
<!-- 支付金额等 -->
```



填写订单 - 收货地址

计算默认收货地址



地址列表页



修改收货地址



收货地址Store



选中收货地址

```
// src/pagesOrder/create/create.vue
const selectedAddress = computed(() => {
  return addressStore.selectedAddress || orderPre.value?.userAddresses.find((v) => v.isDefault)
})
```

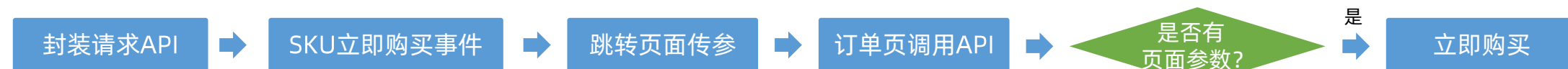
```
// src/stores/modules/address.ts
export const useAddressStore = defineStore('address', () => {
  // 选中的收货地址
  const selectedAddress = ref<AddressItem>()
  // 修改选中地址
  const changeSelectedAddress = (val: AddressItem) => {
    selectedAddress.value = val
  }
  return { selectedAddress, changeSelectedAddress }
})
```

```
// src/pagesMember/address/address.vue
// 选择收货地址
const onSelectAddress = (item: AddressItem) => {
  // 修改选中地址
  const addressStore = useAddressStore()
  addressStore.changeSelectedAddress(item)
  // 返回上一页
  uni.navigateBack()
}
```

```
// src/pagesOrder/create/create.vue
<view @tap="onChangeAddress(item)">
  ...省略
  <navigator
    @tap.stop="() => {}"
  >
    修改
  </navigator>
</view>
```



填写订单 - 立即购买



```
/** 填写订单-获取立即购买订单 */
export const getOrderPreNowAPI = (data: {
  skuId: string
  count: string
  addressId?: string
}) => {
  return http<OrderPreResult>({
    method: 'GET',
    url: '/member/order/pre/now',
    data,
  })
}
```

```
// src/pages/goods/goods.vue
<vk-data-goods-sku-popup
  @add-cart="onAddCart"
  @buy-now="onBuyNow"
/>
```

```
// src/pages/goods/goods.vue
// SKU立即购买事件
const onBuyNow = async (ev: SkuPopupEvent) => {
  // 跳转页面并传参
  uni.navigateTo({ url: `/pagesOrder/create/create?skuId=${ev._id}&count=${ev.buy_num}` })
  // 关闭 SKU 组件
  isShowSku.value = false
}
```

```
// 订单页获取页面参数
const query = defineProps<{
  skuId?: string
  count?: string
}>()
const orderPre = ref<OrderPreResult>()
const getMemberOrderPreData = async () => {
  if (query.skuId && query.count) {
    // 立即购买
    const res = await getMemberOrderPreNowAPI({
      count: query.count,
      skuId: query.skuId,
    })
    orderPre.value = res.result
  } else {
    // 预付订单
    const res = await getMemberOrderPreAPI()
    orderPre.value = res.result
  }
}
```

是否有
页面参数?

否

预付订单



页面
传参



填写订单 - 提交订单

封装请求API



类型声明文件



提交按钮事件



调用接口成功



跳转订单详情

无收货地址交互

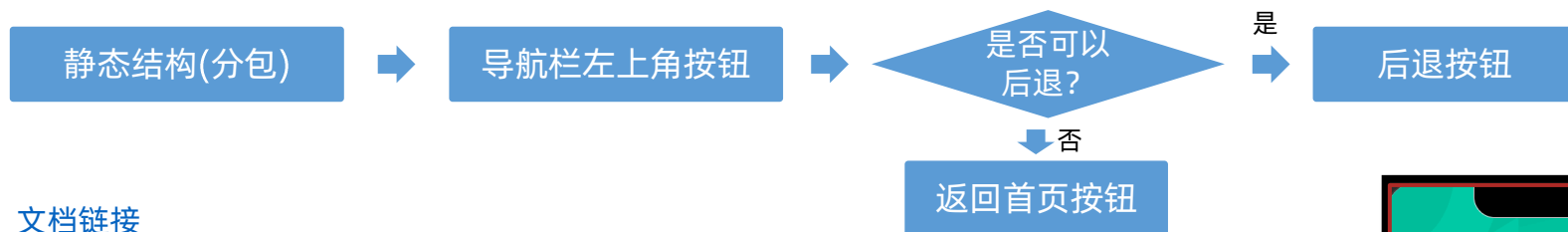
```
/**
 * 提交订单
 * @param data 请求参数
 */
export const postMemberOrderAPI = (data: OrderCreateParams)
=> {
  return http<{ id: string }>({
    method: 'POST',
    url: '/member/order',
    data,
  })
}
```

```
/** 提交订单 请求参数 */
export type OrderCreateParams = {
  addressId: string
  deliveryTimeType: number
  buyerMessage: string
  goods: {
    count: number
    skuId: string
  }[]
  payChannel: 1 | 2
  payType: 1 | 2
}
```

```
// 提交订单
const onOrderSubmit = async () => {
  // 无收货地址提醒
  if (!selectedAddress.value?.id) return uni.showToast({ icon: 'none', title: '请选择收货地址' })
  // 发送请求
  const res = await postMemberOrderAPI({
    addressId: selectedAddress.value!.id,
    buyerMessage: buyerMessage.value,
    deliveryTimeType: activeDelivery.value.type,
    goods: orderPre.value!.goods.map((v) => ({ count: v.count, skuId: v.skuId })),
    payChannel: 2,
    payType: 1,
  })
  // 关闭当前页,跳转订单详情页,并传递订单id
  uni.redirectTo({ url: `/pagesOrder/detail/detail?id=${res.result.id}` })
}
```



订单详情 - 自定义导航栏交互



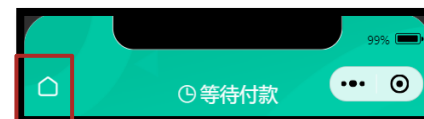
文档链接

滚动驱动动画

获取小程序页面实例

```
// 获取页面栈集合
const pages = getCurrentPages()
// 获取当前页面实例(最后一项)
const pageInstance = pages.at(-1) as any
// 页面渲染完成，绑定关键帧动画
onReady(() => {
  // 自定义导航栏
  pageInstance.animate(
    '.navbar',
    [
      { backgroundColor: 'transparent' },
      { backgroundColor: '#f8f8f8' }
    ],
    1000,
    {
      scrollSource: '#scroller',
      timeRange: 1000,
      startScrollOffset: 0,
      endScrollOffset: 50,
    },
  ),
  // ...省略
})
```

```
<!-- 自定义导航栏 -->
<view class="navbar">
  <view class="wrap">
    <navigator
      v-if="pages.length > 1"
      open-type="navigateBack"
      class="back icon-left"
    ></navigator>
    <navigator
      v-else
      url="/pages/index/index"
      open-type="switchTab"
      class="back icon-home"
    ></navigator>
    <view class="title">
      订单详情
    </view>
  </view>
</view>
<!-- 滚动容器 -->
<scroll-view id="scroller">
  ...省略
</scroll-view>
```



订单详情 - 订单状态渲染

封装请求API

初始化调用

类型声明文件

渲染订单状态

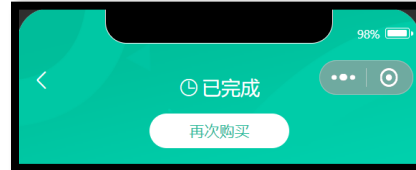
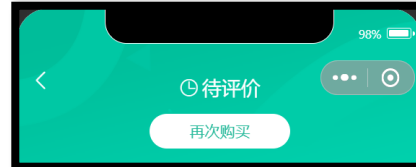
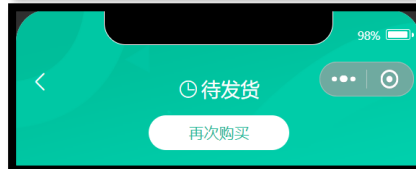
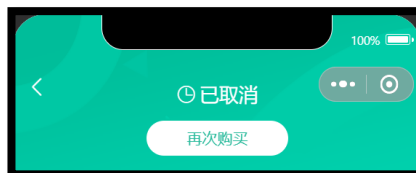
订单状态常量

```
/** 获取订单详情 */
export const getMemberOrderByAPI = (id: string) => {
  return http<OrderResult>({
    method: 'GET',
    url: `/member/order/${id}`,
  })
}
```

```
// src/types/order.d.ts
import type { OrderState } from '@services/constants'
/** 订单详情 返回信息 */
export type OrderResult = {
  id: string
  orderState: OrderState
  ...省略
}
```

```
// src/services/constants.ts
/** 订单状态枚举 */
export enum OrderState {
  DaiFuKuan = 1,
  DaiFaHuo = 2,
  DaiShouHuo = 3,
  DaiPingJia = 4,
  YiWanCheng = 5,
  YiQuXiao = 6,
}
/** 订单状态列表 */
export const orderStateList = [
  { id: 0, text: '' },
  { id: 1, text: '待付款' },
  ...省略
]
```

```
<view class="overview">
  <!-- 待付款状态:展示倒计时 -->
  <template v-if="order.orderState === OrderState.DaiFuKuan">
    <view class="status icon-clock">等待付款</view>
    <view class="button">去支付</view>
  </template>
  <!-- 其他订单状态:展示再次购买按钮 -->
  <template v-else>
    <view class="status">{{ orderStateList[order.orderState].text }}</view>
    <navigator: url="/pagesOrder/create/create?orderId=${query.id}">再次购买</navigator>
  </template>
</view>
```



订单详情 - 待付款 - 倒计时

倒计时组件



组件属性绑定



倒计时结束事件



修改订单状态

```
<template v-if="order.orderState === OrderState.DaiFuKuan">
  <view class="status icon-clock">等待付款</view>
  <view class="tips">
    <text class="money">应付金额: ¥ {{ order.payMoney }}</text>
    <text class="time">支付剩余</text>
    <uni-countdown
      color="#fff"
      splitter-color="#fff"
      :showDay="false"
      :show-colon="false"
      :second="order.countdown"
      @timeup="onTimeup"
    />
  </view>
  <view class="button">去支付</view>
</template>
```

```
// 倒计时时间到触发事件
const onTimeup = () => {
  // 支付超时修改订单状态为: 已取消
  order.value!.orderState = OrderState.YiQuXiao
}
```

uni-countdown 组件

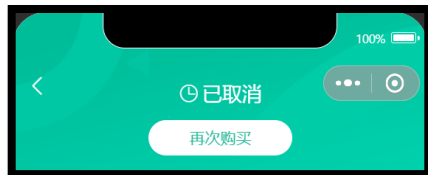
API

Countdown Props

属性名	类型	默认值	说明
backgroundColor	String	#FFFFFF	背景色
color	String	#000000	文字颜色
splitorColor	String	#000000	分割符号颜色
day	Number	0	天数
hour	Number	0	小时
minute	Number	0	分钟
second	Number	0	秒
showDay	Boolean	true	是否显示天数
showColon	Boolean	true	是否以冒号为分隔符
start	Boolean	true	是否初始化组件后就开始倒计时

Countdown Events

事件称名	说明	返回值
@timeup	倒计时时间到触发事件	-



订单详情 - 待付款 - 订单支付

生产环境(PROD)



获取支付参数API



发起微信支付



跳转支付结果页

开发环境(DEV)



模拟支付API



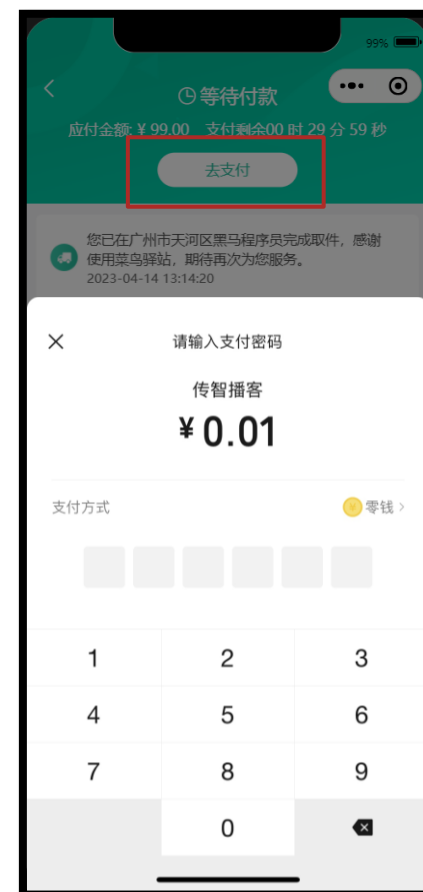
跳转支付结果页

注意事项：开发环境的模拟订单支付，更新订单状态为**待发货**，仅开发和学习使用。

```
/** 获取微信支付参数 */
export const getPayWxPayMiniPayAPI = (data: {
  orderId: string
}) => {
  return http<WechatMiniprogram.RequestPaymentOption>({
    method: 'GET',
    url: '/pay/wxPay/miniPay',
    data,
  })
}
/** 模拟支付-内测版 */
export const getPayMockAPI = (data: {
  orderId: string
}) => {
  return http({
    method: 'GET',
    url: '/pay/mock',
    data,
  })
}
```

```
// 订单支付
const onOrderPay = async () => {
  // 支付订单
  if (import.meta.env.DEV) {
    // 开发环境：模拟支付-更新订单支付状态
    await getPayMockAPI({ orderId: query.id })
  } else {
    // 生产环境：根据订单号获取微信支付所需参数
    const res = await getPayWxPayMiniPayAPI({ orderId: query.id })
    // 发起微信支付
    await wx.requestPayment(res.result)
  }
  // 支付成功，关闭当前页面，跳转到支付成功页
  uni.redirectTo({
    url: `/pagesOrder/payment/payment?id=${query.id}`
  })
}

<view @tap="onOrderPay">去支付</view>
```



订单详情 - 待发货 - 模拟发货

封装请求API



条件渲染



事件绑定



调用API成功



更新订单状态

注意事项：在 **DEV环境** 下使用，仅在订单状态为**待发货**时，可模拟发货，调用后订单状态修改为**待收货**，包含模拟物流。

```
/** 模拟发货-内测版 */
export const getMemberOrderConsignmentByIdAPI = (id: string) => {
  return http({
    method: 'GET',
    url: `/member/order/consignment/${id}`,
  })
}
```

```
<view v-if="isDev && order.orderState === OrderState.DaiFaHuo" @tap="onOrderSend">
  模拟发货
</view>

// 获取开发环境
const isDev = import.meta.env.DEV
// 点击模拟发货
const onOrderSend = async () => {
  // 仅在开发环境下使用，打包到生产环境会剔除以下代码 (tree shaking 树摇优化)
  if (isDev) {
    // 调用模拟发货
    await getMemberOrderConsignmentByIdAPI(query.id)
    uni.showToast({ icon: 'success', title: '模拟发货完成' })
    // 手动修改订单状态为待收货
    order.value!.orderState = OrderState.DaiShouHuo
  }
}
```



订单详情 - 待收货 - 确认收货

封装请求API



条件渲染 & 事件绑定



二次确认弹窗



调用API成功



更新订单状态

注意事项：仅在订单状态为**待收货**时，可确认收货。

```
/** 确认收货 */
export const putMemberOrderReceiptByIdAPI = (id: string) => {
  return http<OrderResult>({
    method: 'PUT',
    url: `/member/order/${id}/receipt`,
  })
}
```

```
<view v-if="order.orderState === OrderState.DaiShouHuo" @tap="onOrderConfirm">
  确认收货
</view>

// 确认收货
const onOrderConfirm = () => {
  // 二次确认弹窗
  uni.showModal({
    content: '为保障您的权益，请收到货并确认无误后，再确认收货',
    success: async (success) => {
      if (success.confirm) {
        const res = await putMemberOrderReceiptByIdAPI(query.id)
        uni.showToast({ icon: 'success', title: '确认收货成功' })
        order.value = res.result
      }
    },
  })
}
```



订单详情 - 订单物流

封装请求API



获取订单详情后



判断订单状态



获取物流信息



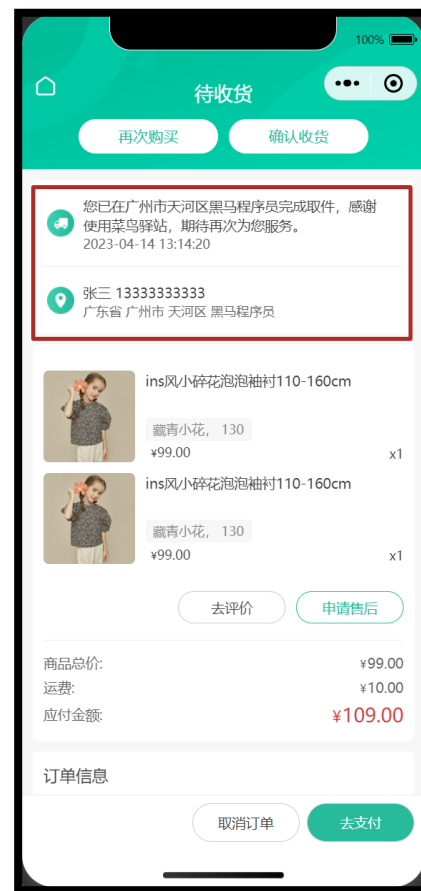
渲染物流信息

注意事项：仅在订单状态为**待收货**，**待评价**，**已完成**时，可获取物流信息。

```
/** 获取订单物流 */
export const getMemberOrderLogisticsByIdAPI = (id: string) => {
  return http<OrderLogisticResult>({
    method: 'GET',
    url: `/member/order/${id}/logistics`,
  })
}
```

```
// 获取订单详情
const order = ref<OrderResult>()
const getMemberOrderByIdData = async () => {
  const res = await getMemberOrderByIdAPI(query.id)
  order.value = res.result
  // 仅在订单状态为待收货，待评价，已完成时，可获取物流信息。
  if ([OrderState.DaiShouHuo, OrderState.DaiPingJia, OrderState.YiWanCheng].includes(order.value.orderState)) {
    getMemberOrderLogisticsByIdData()
  }
}
```

```
// 获取物流信息
const logisticList = ref<LogisticItem[]>([])
const getMemberOrderLogisticsByIdData = async () => {
  const res = await getMemberOrderLogisticsByIdAPI(query.id)
  logisticList.value = res.result.list
}
```



订单详情 - 删除订单

封装请求API



条件渲染 & 事件绑定



二次确认弹窗



调用API成功



跳转到订单列表

注意事项：仅在订单状态为**待评价**，**已完成**，**已取消**时，可删除订单。

```
/** 删除订单 */
export const deleteMemberOrderAPI = (data: { ids: string[] }) => {
  return http({
    method: 'DELETE',
    url: `/member/order`,
    data,
  })
}
```

```
<view v-if="order.orderState >= OrderState.DaiPingJia" @tap="onOrderDelete">
  删除订单
</view>

const onOrderDelete = () => {
  // 二次确认弹窗
  uni.showModal({
    content: '你确定要删除该订单?',
    success: async (res) => {
      if (res.confirm) {
        await deleteMemberOrderAPI({ ids: [query.id] })
        // 关闭当前页面，跳转到订单列表页
        uni.redirectTo({ url: '/pagesOrder/list/list' })
      }
    },
  })
}
```



订单列表 - Tabs 滑动切换

静态结构



Tabs 文字渲染



点文字高亮切换



swiper 滑动切换

```
// 高亮下标
const activeIndex = ref(0)

<!-- tabs -->
<view class="tabs">
  <text
    class="item"
    v-for="(item, index) in orderTabs"
    :key="item.title"
    @tap="activeIndex = index"
  >
    {{ item.title }}
  </text>
<!-- 游标 -->
<view class="cursor" :style="{ left: activeIndex * 20 + '%' }"></view>
</view>
<!-- 滑动容器 -->
<swiper class="swiper"
  :current="activeIndex"
  @change="activeIndex = $event.detail.current"
>
  ...省略
</swiper>
```

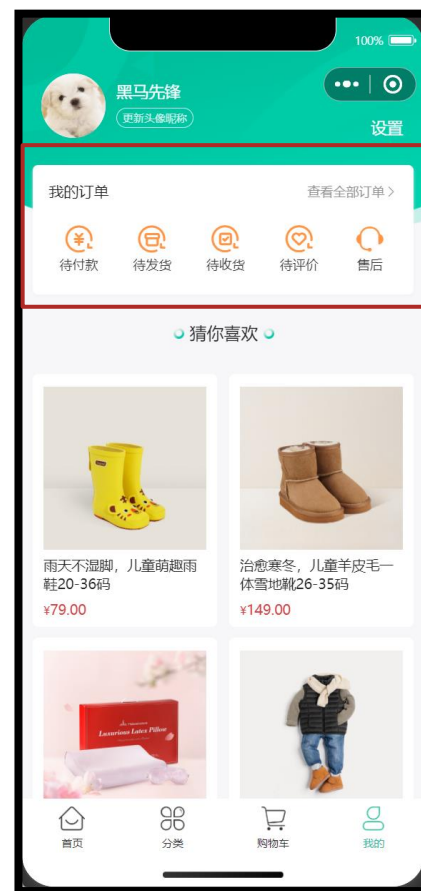


订单列表 - Tabs 页面跳转高亮



```
<navigator url="/pagesOrder/list/list?type=0">  
  查看全部订单  
</navigator>
```

```
// 接收页面参数  
const query = defineProps<{  
  type: string  
  
// 高亮下标  
const activeIndex = ref(  
  orderTabs.value.findIndex((v) => v.orderState === Number(query.type))  
)
```



页面
传参
→



订单列表 - 列表渲染

封装列表组件



订单状态父传子



封装请求API



准备请求参数



初始化调用



页面渲染

```
/** 获取订单列表 */
export const getMemberOrderAPI = (data: OrderListParams) => {
  return http<OrderListResult>({
    method: 'GET',
    url: `/member/order`,
    data,
  })
}
```

```
// 组件 props
const props = defineProps<{
  orderState: number
}>()
// 请求参数
const queryParams = {
  page: 1,
  pageSize: 5,
  orderState: props.orderState,
}
// 获取订单列表
const orderList = ref<OrderItem[]>([])
const getMemberOrderData = async () => {
  const res = await getMemberOrderAPI(queryParams)
  orderList.value = res.result.items
}
onMounted(() => {
  getMemberOrderData()
})
```

获取订单列表

GET /member/order

请求参数

Query 参数

page integer 可选

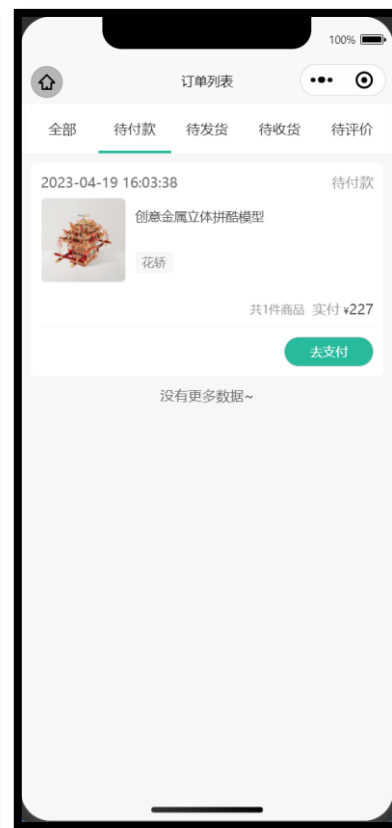
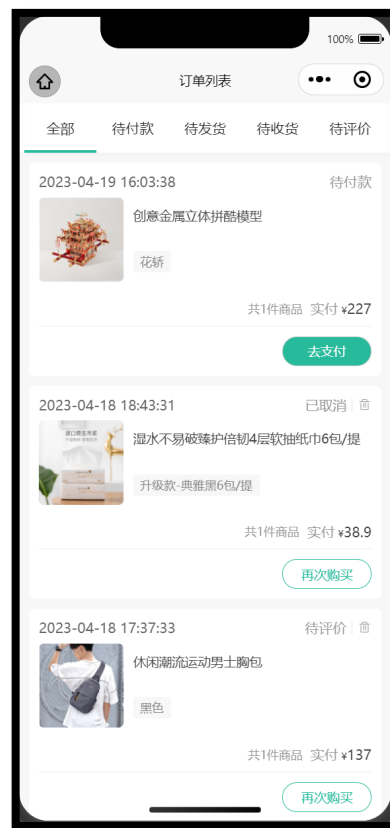
页码

pageSize integer 可选

页容量

orderState integer 可选

订单状态



订单列表 - 订单支付

按钮条件渲染



事件绑定



支付成功



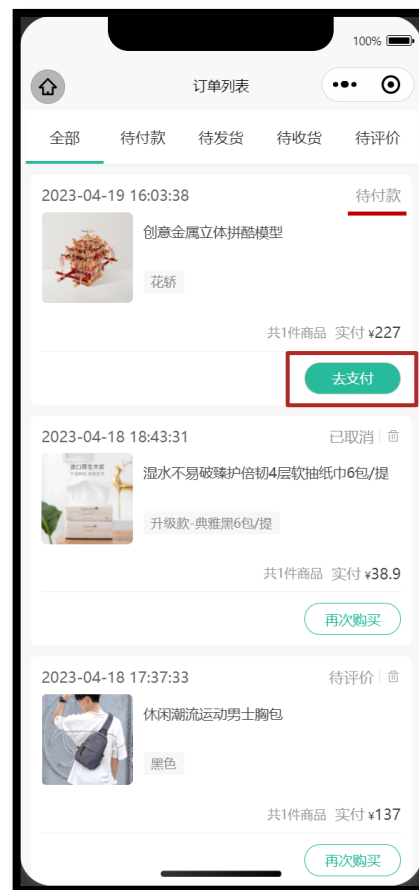
成功提示



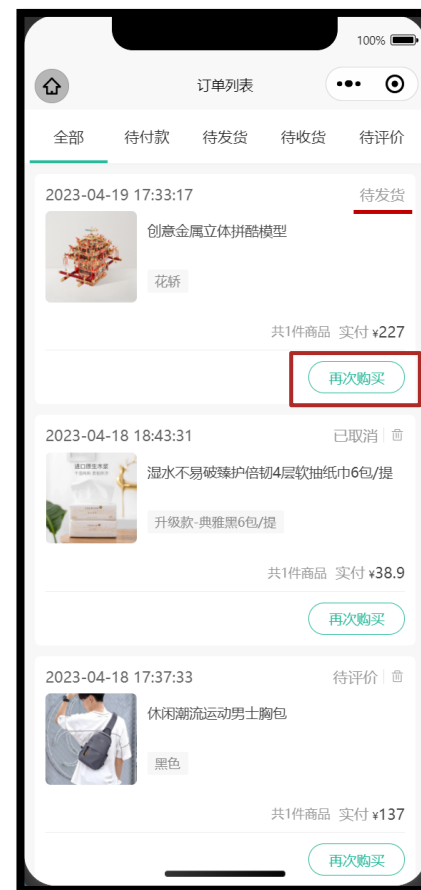
更新订单状态

```
<view @tap="onOrderPay(order.id)">去支付</view>

// 订单支付
const onOrderPay = async (id: string) => {
  if (import.meta.env.DEV) {
    // 开发环境模拟支付
    await getPayMockAPI({ orderId: id })
  } else {
    // 正式环境微信支付
    const res = await getPayWxPayMiniPayAPI({ orderId: id })
    await wx.requestPayment(res.result)
  }
  // 成功提示
  uni.showToast({ title: '支付成功' })
  // 更新订单状态
  const order = orderList.value.find((v) => v.id === id)
  order!.orderState = OrderState.DaiShouHuo
}
```



状态更新



项目打包 - 微信小程序端发布上线


运行打包命令

导入微信开发者工具

小程序代码上传

提交审核

正式发布

 Visual Studio Code

终端 调试控制台

```
PS E:\uni-app\小程序项目实战\heima-shop> pnpm build:mp-weixin

> uni-app-xiaotuxian@0.0.0 build:mp-weixin E:\uni-app\小程序项目实战\heima-shop
> uni build -p mp-weixin

正在编译中...
DONE Build complete.
运行方式: 打开 微信开发者工具, 导入 dist\build\mp-weixin 运行。
```

 微信开发者工具

 上传

 版本管理

 详情

 消息

温馨提示: 也通过开发辅助工具 [miniprogram-ci](#) 上传

 微信公众平台

<https://mp.weixin.qq.com>

版本号

开发者

黑马程序员

提交审核

1.0.1

提交时间

2023-05-20 13:14:20

[体验版](#)

项目备注



项目打包 - 条件编译和网页端打包

常见问题：按照 uni-app 规范开发可保证**多平台兼容**，但每个平台有自己的一些特性，该如何处理？

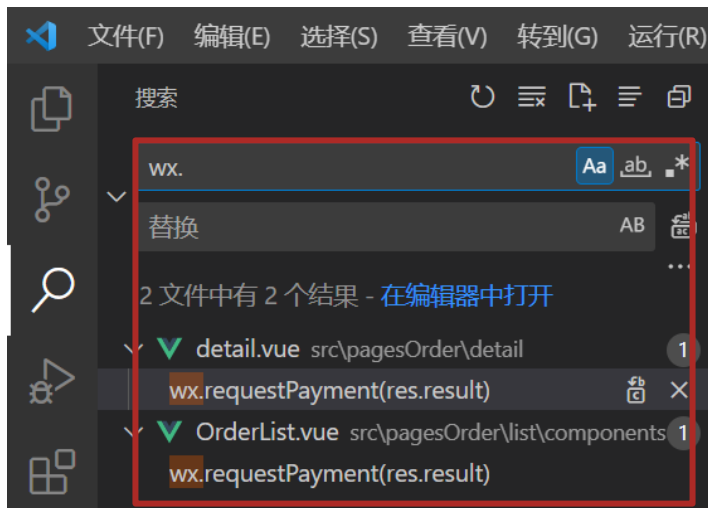
注意事项：网页端不支持微信平台授权登录等功能，可通过**条件编译**，让代码按条件**编译到不同平台**。

条件编译语法：通过特殊注释，以 **#ifdef** 或 **#ifndef** 加 **平台名称** 开头，以 **#endif** 结尾。

```
<script setup lang="ts">
// #ifdef MP-WEIXIN
wx.login()
wx.requestOrderPayment()
// #endif
</script>
```

```
<template>
<!-- #ifdef MP-WEIXIN -->
<button open-type="openSetting">授权管理</button>
<button open-type="feedback">问题反馈</button>
<button open-type="contact">联系我们</button>
<!-- #endif -->
</template>
```

支持：vue, ts, js, scss, css, pages.json 等文件



技巧：用 **wx.** 和 **open-type** 关键词快速搜索

```
// manifest.json
{
  /* 网页端特有配置 */
  "h5": {
    "router": {
      // 基础路径。默认为 /
      "base": "./"
    }
  },
  /* 小程序特有相关 */
  "mp-weixin": {
    ...省略
  },
  "vueVersion": "3"
}
```

打包成网页端：配置路由基础路径

温馨提醒：如果是打包成 App 端，需要把项目代码导入 HbuilderX 开发工具，由 HbuilderX 运行到真机或模拟器，提供原生App云打包服务。



传智教育旗下高端IT教育品牌