



# 廣東工業大學

## 移动智能平台系统 大作业

题    目	仿美团外卖
学    院	计算机学院
专业班别	计科 4 班
学    号	3120005084
学生姓名	王宁
指导教师	杨宝瑶

2023 年 5 月 13 日

# 目录

1	大作业题目 .....	1
1.1	制作《仿美团外卖》APP .....	1
2	开发环境及主要依赖库 .....	3
2.1	开发环境.....	3
2.2	主要依赖库.....	3
3	功能实现过程 .....	4
3.1	店铺功能实现.....	4
3.1.1	标题栏布局 .....	4
3.1.2	水平滑动的广告栏 Banner .....	4
3.1.3	店铺页面布局 .....	6
3.1.4	店铺列表条目项 .....	7
3.1.5	引入 okhttp、 gson、 glide 依赖库 .....	7
3.1.6	店铺列表界面 .....	8
3.2	店铺详细功能实现.....	10
3.2.1	店铺详细信息及购物车列表条目 .....	10
3.2.2	菜单列表条目界面 .....	11
3.2.3	清空购物车功能 .....	12
3.2.4	店铺详细信息列表适配器 .....	12
3.2.5	购物车列表适配器 .....	13
3.2.6	店铺详细信息功能小结 .....	13
3.3	菜品功能实现.....	14
3.3.1	菜品详情界面 .....	14
3.3.2	菜品数据显示 .....	14
3.4	订单功能实现.....	15
3.4.1	订单界面 .....	15
3.4.2	订单列表条目项 .....	16
3.4.3	支付订单功能 .....	17
3.5	启动应用功能.....	18
3.5.1	启动 Tomcat 服务器 .....	18
3.5.2	模仿欢迎界面 .....	18
3.5.3	项目启动入口 .....	19
4	实现效果展示 .....	20
4.1	启动项目 .....	20
4.2	店铺界面功能演示.....	20

4.2.1 广告 Banner .....	20
4.3 店铺详情界面.....	21
4.3.1 菜品信息 .....	21
4.3.2 购物车功能 .....	21
4.3.3 返回至店铺界面 .....	22
4.4 订单界面.....	23
4.4.1 支付结算 .....	23
4.4.2 收货地址 .....	23
4.4.3 支付二维码 .....	24
4.4.4 返回至店铺详情界面 .....	25
5 心得体会 .....	25

# 1 大作业题目

## 1.1 制作《仿美团外卖》APP

- 程序启动后，首先会进入店铺界面，该界面展示的是一些由店铺信息组成的列表与一个滑动的广告栏。



- 点击店铺列表中任意一个条目或广告栏中的任意一张图片，程序都会跳转到对应的店铺详情界面，该界面展示的是店铺的公告信息、配送信息、菜单列表信息以及购物车信息。点击菜单列表条目右侧的“加入购物车”按钮可以将菜品添加到购物车中，在界面左下角可以看到购物车中添加的菜品数量。



3. 已选商品列表的右上角有一个“清空”按钮，点击该按钮会弹出一个确认清空购物车的对话框。



4. 在店铺详情界面中，点击菜单列表的任意一个条目，程序都会跳转到菜品详情界面，菜品详情界面是一个对话框的样式。



5. 在店铺详情界面中，点击“去结算”按钮会跳转到订单界面，该界面通过一个列表展示购物车中的菜品信息，点击“去支付”按钮，程序会弹出一个显示支付二维码的对话框。



## 2 开发环境及主要依赖库

### 2.1 开发环境

操作系统: Windows 10

IDE: Android Studio 2021.3.1

服务器: Tomcat 9.0

JDK: jdk-11.0.11

Android SDK: 32 (minSdk 23)

Gradle: gradle-7.4-bin

Android Gradle Plugin Version: 7.3.0

Device Manager: Pixel 3a XL API 33

### 2.2 主要依赖库

glide:4.13.0

okhttp:3.12.0

gson:2.8.5

io.github.youth5201314:banner:2.2.2

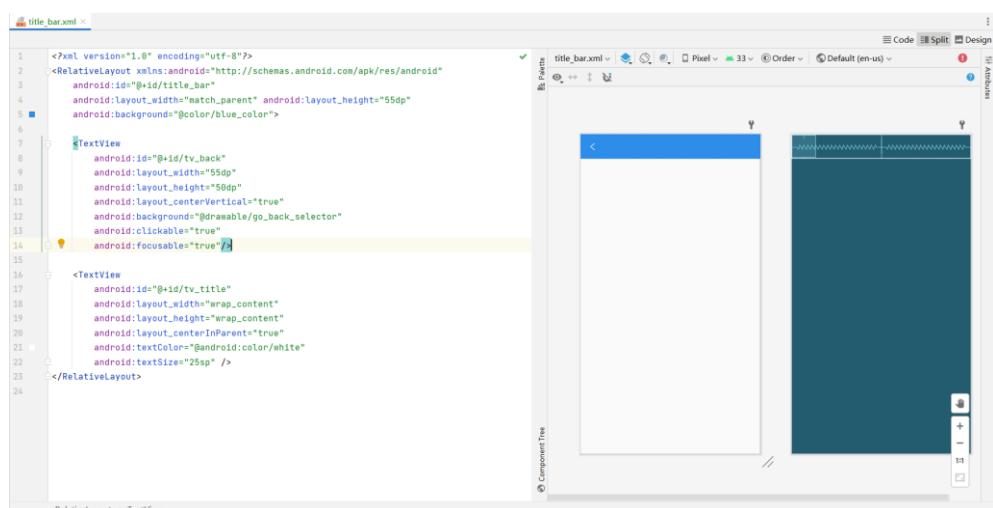
### 3 功能实现过程

#### 3.1 店铺功能实现

当打开《仿美团外卖》项目时，程序会直接进入主界面，也就是店铺列表界面。店铺列表界面从上至下分为标题栏、水平滑动广告栏和店铺列表三部分。其中，广告栏与店铺列表的数据是通过网络请求从服务器上获取的 JSON 数据解析而来。

##### 3.1.1 标题栏布局

(1) 标题栏布局由 2 个 TextView 控件放置在 RelativeLayout 布局中。其中 1 个 TextView 处理点击返回上一页事件，另 1 个 TextView 居中显示界面标题信息。



(2) 在 themes.xml 中去掉默认标题栏。



##### 3.1.2 水平滑动的广告栏 Banner

(1) 引入 banner 依赖库。

```
/* banner */  
implementation 'io.github.youth5201314:banner:2.2.2'
```

(2) 创建 AdBannerBean 类，并提供获取广告图片 Banner 列表方法；创建 ImageHolder 类来复用 Banner 中的 ImageView。

```

1 package com.wangning.order.bean;
2
3 import com.wangning.order.R;
4
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class AdBannerBean {
9     public Integer imageRes;
10
11     public AdBannerBean(Integer imageRes) { this.imageRes = imageRes; }
12
13     /**
14      * 获取广告 Banner 列表
15      *
16      * @return List<AdBannerBean>
17     */
18     public static List<AdBannerBean> getTestData() {
19         List<AdBannerBean> list = new ArrayList<>();
20         list.add(new AdBannerBean(R.drawable.banner_1));
21         list.add(new AdBannerBean(R.drawable.banner_2));
22         list.add(new AdBannerBean(R.drawable.banner_3));
23         return list;
24     }
25
26 }
27
28

```

```

1 package com.wangning.order.viewholder;
2
3 import ...
4
5 public class ImageHolder extends RecyclerView.ViewHolder {
6     public ImageView imageView;
7
8     public ImageHolder(@NonNull View view) {
9         super(view);
10        this.imageView = (ImageView) view;
11    }
12
13 }
14
15
16

```

(3) 创建 ImageAdapter 类自定义 Banner 适配器。

```

1 package com.wangning.order.adapter;
2
3 import android.view.ViewGroup;
4 import android.widget.ImageView;
5
6 import com.wangning.order.bean.AdBannerBean;
7 import com.wangning.order.viewholder.ImageHolder;
8 import com.youth.banner.adapter.BannerAdapter;
9
10 import java.util.List;
11
12 /**
13  * 自定义 Banner
14  */
15 public class ImageAdapter extends BannerAdapter<AdBannerBean, ImageHolder> {
16
17     /**
18      * 设置数据，也可以调用 banner 提供的方法，或者自己在 adapter 中实现
19      *
20      * @param mDatas mDatas
21     */
22     public ImageAdapter(List<AdBannerBean> mDatas) { super(mDatas); }
23
24     /**
25      * 创建 ViewHolder，可以用 viewType 这个字段来区分不同的 ViewHolder
26      *
27      * @param parent ViewGroup
28      * @param viewType viewType
29      * @return ImageHolder
30     */
31     ...
32 }
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58

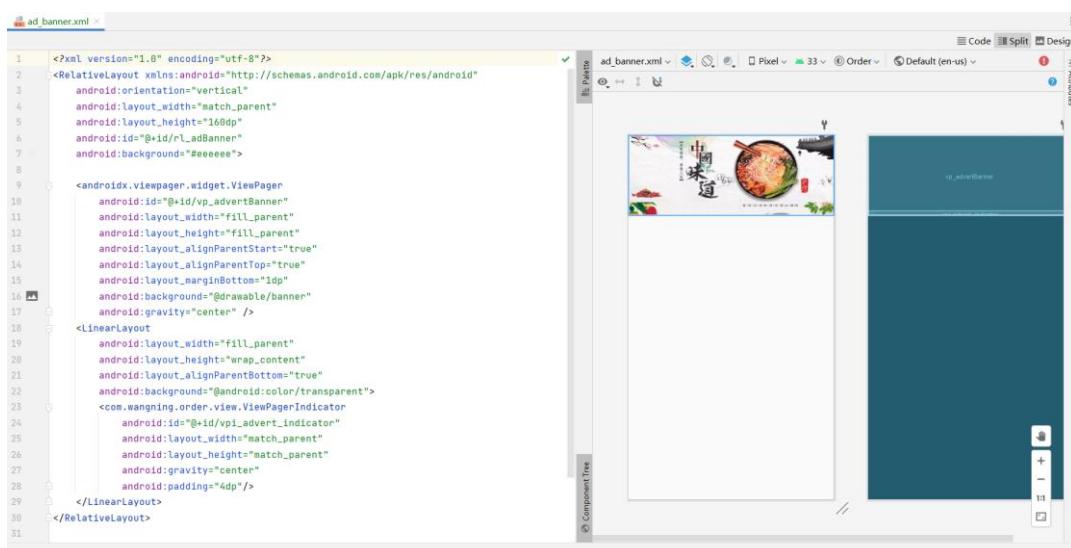
```

```

1 package com.wangning.order.adapter;
2
3 import android.view.View;
4 import android.widget.ImageView;
5
6 import com.wangning.order.bean.AdBannerBean;
7 import com.wangning.order.viewholder.ImageHolder;
8 import com.youth.banner.adapter.BannerAdapter;
9
10 import java.util.List;
11
12 /**
13  * 自定义 Banner
14  */
15 public class ImageAdapter extends BannerAdapter<AdBannerBean, ImageHolder> {
16
17     /**
18      * 设置数据，也可以调用 banner 提供的方法，或者自己在 adapter 中实现
19      *
20      * @param mDatas mDatas
21     */
22     public ImageAdapter(List<AdBannerBean> mDatas) { super(mDatas); }
23
24     /**
25      * 创建 ViewHolder，可以用 viewType 这个字段来区分不同的 ViewHolder
26      *
27      * @param parent ViewGroup
28      * @param viewType viewType
29      * @return ImageHolder
30     */
31     ...
32 }
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58

```

(4) 创建广告栏界面的布局文件 ad\_banner.xml。放置 1 个 ViewPager 控件，1 个 ViewPagerIndicator 控件。



(5) 创建 ViewPagerIndicator 类并继承 LinearLayout 类，设置 Banner 图内小圆点的行为。



```
1 package com.wangning.order.view;
2
3 import android.content.Context;
4 import android.util.AttributeSet;
5 import android.view.Gravity;
6 import android.widget.ImageView;
7 import android.widget.LinearLayout;
8
9 import com.wangning.order.R;
10
11 /**
12 * banner 下面的小圆点
13 */
14
15 public class ViewPagerIndicator extends LinearLayout {
16
17     private int mCount; //小圆点的个数
18     private int mIndex; //当前小圆点的位置
19     private Context context;
20
21     public ViewPagerIndicator(Context context) { this(context, null); }
22
23     public ViewPagerIndicator(Context context, AttributeSet attrs) {
24         super(context, attrs);
25         this.context = context;
26         setGravity(Gravity.CENTER); //设置小圆点布局中
27     }
28
29     this.context = context;
30     setGravity(Gravity.CENTER); //设置小圆点布局中
31 }
32
33 // 设置滑动到当前位置小圆点和其他小圆点的位置
34 public void setCurrentPosition(int currentIndex) {
35     mIndex = currentIndex; //设置当前小圆点
36     removeAllViews(); //移除界面上存在的view
37     int pex = 5;
38     for (int i = 0; i < mCount; i++) {
39         ImageView imageView = new ImageView(context);
40         if (index == i) { //设置当前圆点为选中
41             //白色圆点
42             imageView.setImageResource(R.drawable.indicator_on);
43         } else {
44             //灰色小圆点
45             imageView.setImageResource(R.drawable.indicator_off);
46         }
47         imageView.setPadding(pex, top: 0, pex, bottom: 0);
48         addView(imageView);
49     }
50 }
51
52 // 设置小圆点的数目
53 public void setCount(int count) { this.mCount = count; }
```

### 3.1.3 店铺页面布局

(1) 创建 `ShopListView` 类继承 `ListView` 类, 准备存放店铺列表数据。

```
ShopListView.java

1 package com.wangning.order.view;
2
3 import android.content.Context;
4 import android.util.AttributeSet;
5 import android.widget.ListView;
6
7 public class ShopListView extends ListView {
8     public ShopListView(Context context) { super(context); }
9
10    public ShopListView(Context context, AttributeSet attrs) { super(context, attrs); }
11
12    public ShopListView(Context context, AttributeSet attrs, int defStyleAttr) {
13        super(context, attrs, defStyleAttr);
14    }
15
16    public ShopListView(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
17        super(context, attrs, defStyleAttr, defStyleRes);
18    }
19
20
21    /**
22     * view 30 位: 取 Int 最大值 32 位 >> 2. 模式是所有子组件高度
23     *
24     * @param widthMeasureSpec widthMeasureSpec
25     * @param heightMeasureSpec heightMeasureSpec
26     */
27    @Override
28    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
29        int heightSpec = MeasureSpec.makeMeasureSpec( size: Integer.MAX_VALUE >> 2, MeasureSpec.AT_MOST );
30        super.onMeasure(widthMeasureSpec, heightSpec);
31    }
32
33
34
35
36 }
```

(2) 在 activity\_shop.xml 中引入 title\_bar.xml 文件，放置 2 个自定义控件 Banner 与 ShopListView，外层的 ScrollView 实现内容滑动。

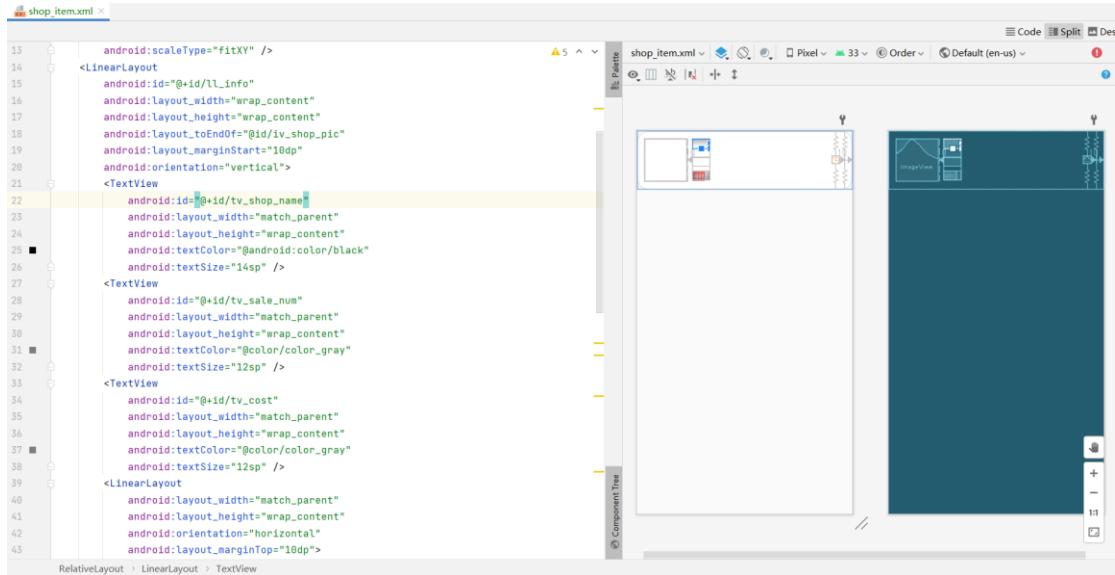
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activity.ShopActivity"
    android:orientation="vertical">
<include layout="@layout/title_bar"/>

<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">

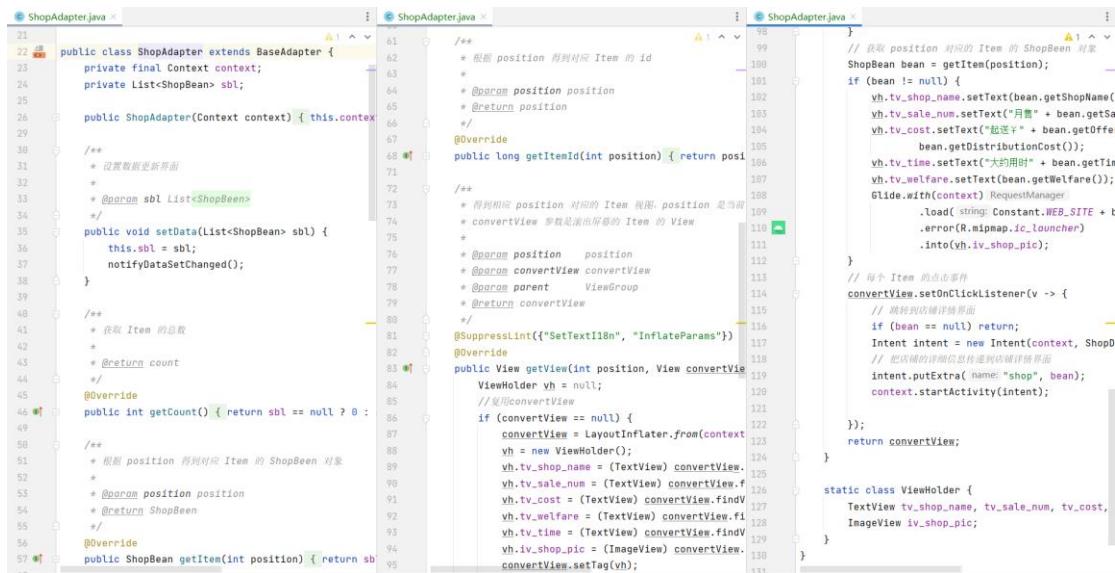
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <!-- Banner -->
        <com.youth.banner.Banner
            android:id="@+id/banner"
            android:layout_width="match_parent"
            android:layout_height="180dp" />
        <!-- ShopListView -->
        <com.wanping.order.view.ShopListView
            android:id="@+id/lv_shop_list"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_margin="10dp"
            android:divider="#FAFAFA"
            android:dividerHeight="10dp" />
    </LinearLayout>
</ScrollView>
</LinearLayout>
```

### 3.1.4 店铺列表条目项

(1) 店铺列表条目布局 shop\_item.xml 由 2 个 ImageView、6 个 TextView 控件组成。



(2) 店铺列表适配器 ShopAdapter 类，实现数据与 ShopListView 控件的适配。



### 3.1.5 引入 okhttp、gson、glide 依赖库

(1) 引入 okhttp、gson、glide 依赖库。

```
/* glide */
implementation 'com.github.bumptech.glide:glide:4.13.0'
annotationProcessor 'com.github.bumptech.glide:compiler:4.13.0'
/* okhttp */
```

```

implementation 'com.squareup.okhttp3:okhttp:3.12.0'
/* gson */
implementation 'com.google.code.gson:gson:2.8.5'

```

- (2) 本地 Tomcat 服务器使用非安全的 http 连接，需要配置 network security config，在 res/xml 下建立 network\_security\_config.xml。

```

<network-security-config>
    <!-- 默认运行所有网址使用非安全连接 -->
    <base-config cleartextTrafficPermitted="true" />
</network-security-config>

```

- (3) 在 AndroidManifest.xml 中申请网络权限并设置 network security config。

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="饿了么"
        android:networkSecurityConfig="@xml/network_security_config"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Order"
        tools:targetApi="31">
    
```

### 3.1.6 店铺列表界面

- (1) 创建 Constant 类设置常量，其中 WEB\_SITE 为请求内网接口，REQUEST\_SHOP\_URL 为请求列表接口。

```

package com.wangning.order.utils;

public class Constant {
    /**
     * 内网接口
     * <p>
     * <b>TODO: 修改 IP:PORT</b>
     */
    public static final String WEB_SITE = "http://192.168.3.185:8080/order";

    /**
     * 店铺列表接口
     */
    public static final String REQUEST_SHOP_URL = "/shop_list_data.json";
}

```

(2) 创建 JsonParse 工具类实现对 JSON 数据的解析，提供了从 JSON 文本解析 ShopBeen 列表的功能，并且使用单例模式使项目中只存在一个 JsonParse 实例。

(3) 界面所在的 ShopActivity 实现对界面控件的初始化、事件捕获、异步线程访问网络获取 JSON 文本后调用 JsonParse 完成解析，最后对数据进行展示。

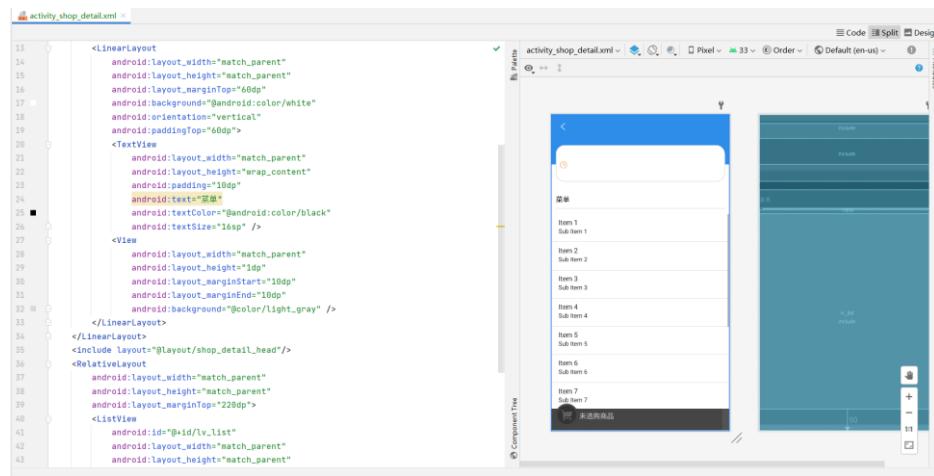
The screenshot shows three instances of the same Java file, `ShopActivity.java`, displayed in separate tabs of an Android Studio code editor. Each instance highlights the same line of code, which is annotated with a yellow arrow pointing upwards. The code on this line is as follows:

```
    .addBannerLifecycleObserver(this); //添加生命周期观察者
```

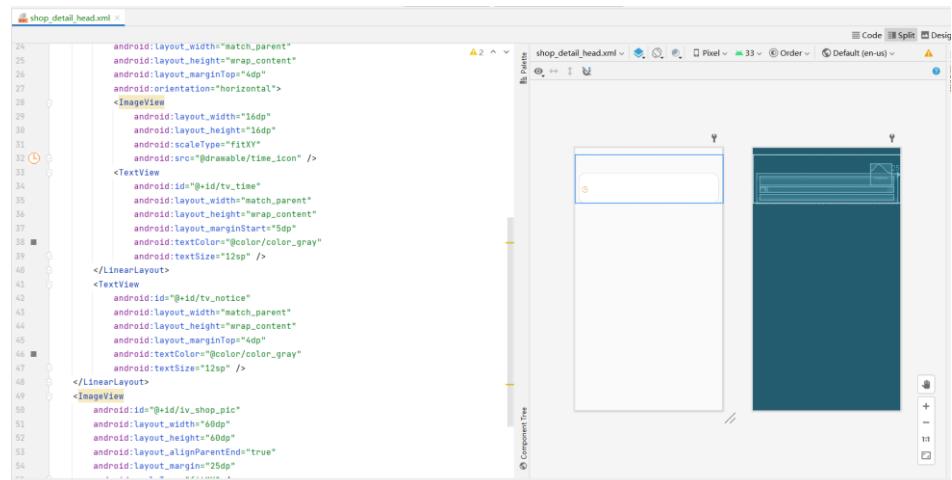
## 3.2 店铺详细功能实现

### 3.2.1 店铺详细信息及购物车列表条目

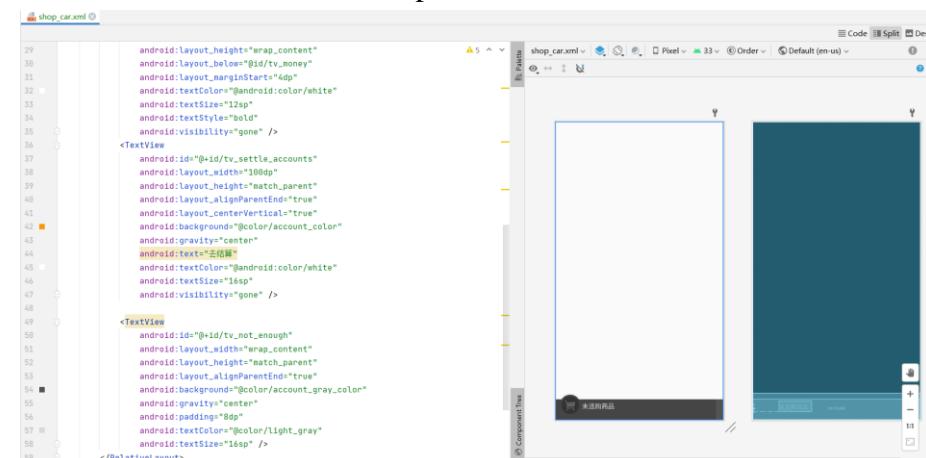
- (1) 创建 ShopDetailActivity，指定布局文件名为 activity\_shop\_detail.xml，放置 1 个 TextView 控件、1 个 View 控件、1 个 ListView 控件。



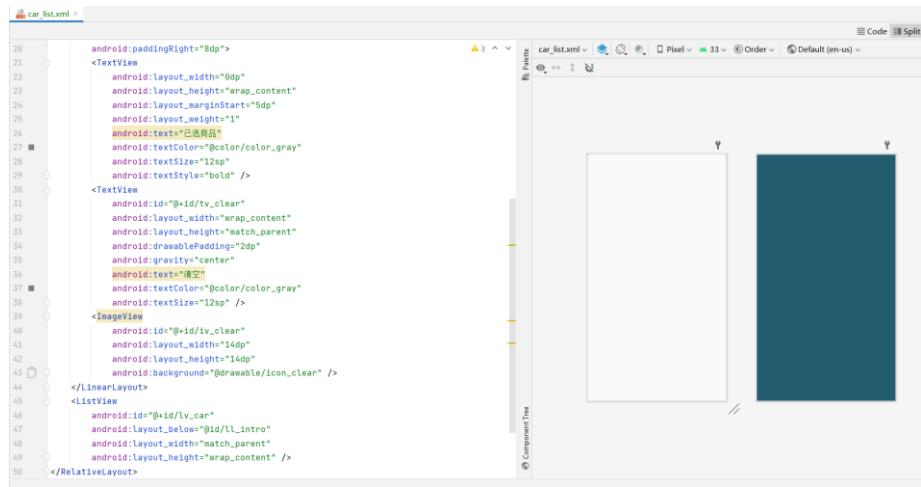
- (2) 头部布局文件 shop\_detail\_head.xml。



- (3) 页面下方购物车栏布局文件 shop\_car.xml。



#### (4) 购物车内商品列表布局文件 car\_list.xml。

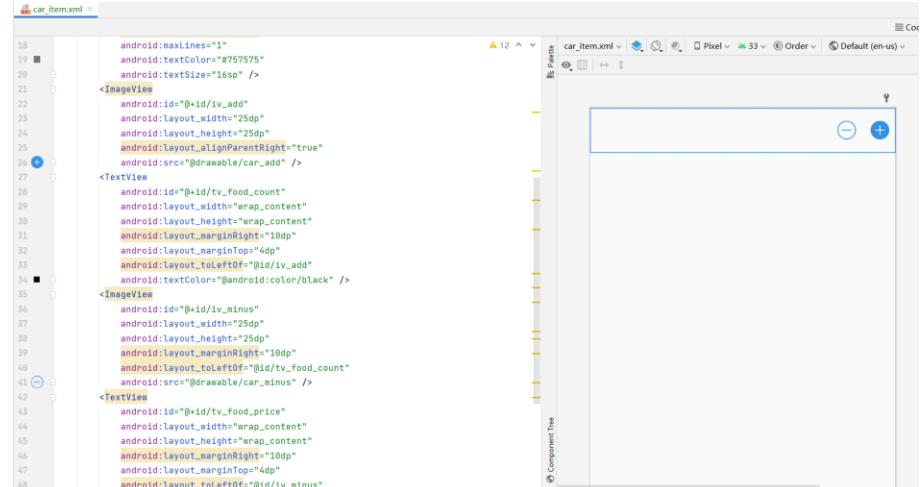


```

<LinearLayout
    android:id="@+id/lv_car"
    android:layout_below="@+id/lL_intro"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
<Listview
    android:id="@+id/lv_car"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
<RelativeLayout>
    <TextView
        android:id="@+id/tv_clean"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:drawablePadding="2dp"
        android:gravity="center"
        android:text="清空"
        android:textColor="@color/color_gray"
        android:textSize="12sp"
        android:textStyle="bold" />
    <Imageview
        android:id="@+id/iv_clean"
        android:layout_width="14dp"
        android:layout_height="14dp"
        android:background="@drawable/icon_clean" />
</RelativeLayout>

```

#### (5) 购物车内商品列表项布局文件 car\_item.xml。



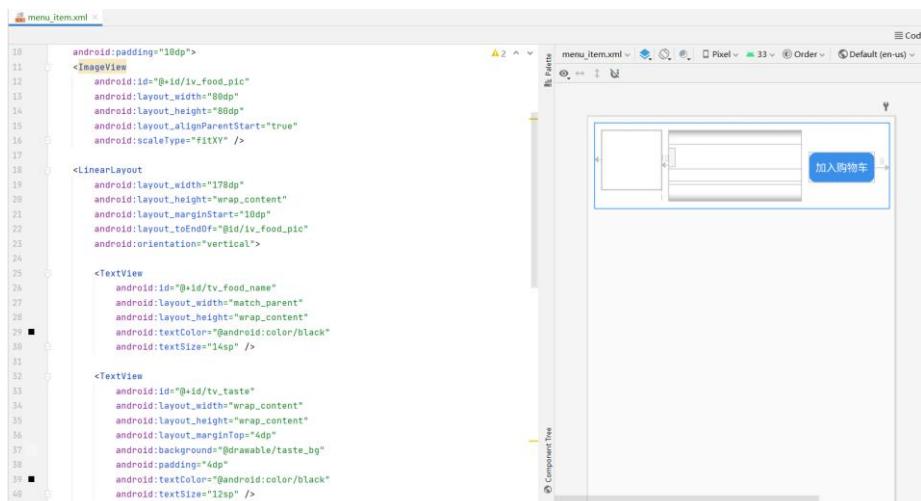
```

<Imageview
    android:id="@+id/iv_add"
    android:layout_width="25dp"
    android:layout_height="25dp"
    android:layout_alignParentRight="true"
    android:src="@drawable/car_add" />
<Textview
    android:id="@+id/tv_food_count"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginRight="10dp"
    android:layout_marginTop="4dp"
    android:layout_toLeftOf="@+id/iv_add"
    android:textColor="@android:color/black" />
<Imageview
    android:id="@+id/iv_minus"
    android:layout_width="25dp"
    android:layout_height="25dp"
    android:layout_marginRight="10dp"
    android:layout_toLeftOf="@+id/tv_food_count"
    android:src="@drawable/car_minus" />
<Textview
    android:id="@+id/tv_food_price"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginRight="10dp"
    android:layout_marginTop="4dp"
    android:layout_toLeftOf="@+id/iv_minus"
    android:textColor="@android:color/black" />

```

### 3.2.2 菜单列表条目界面

(1) 菜单列表条目项布局文件 menu\_item.xml 由 4 个 TextView、1 个 ImageView、1 个 Button 控件组成。



```

<Imageview
    android:id="@+id/iv_food_pic"
    android:layout_width="80dp"
    android:layout_height="80dp"
    android:layout_alignParentStart="true"
    android:scaleType="fitXY" />
<LinearLayout
    android:layout_width="178dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_toEndOf="@+id/iv_food_pic"
    android:orientation="vertical" />
<Textview
    android:id="@+id/tv_food_name"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@android:color/black"
    android:textSize="14sp" />
<Textview
    android:id="@+id/tv_taste"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="4dp"
    android:background="@drawable/taste_bg"
    android:padding="4dp"
    android:textColor="@android:color/black"
    android:textSize="12sp" />
<Button
    android:id="@+id/btn_add"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:background="@drawable/btn_bg"
    android:text="加入购物车" />

```

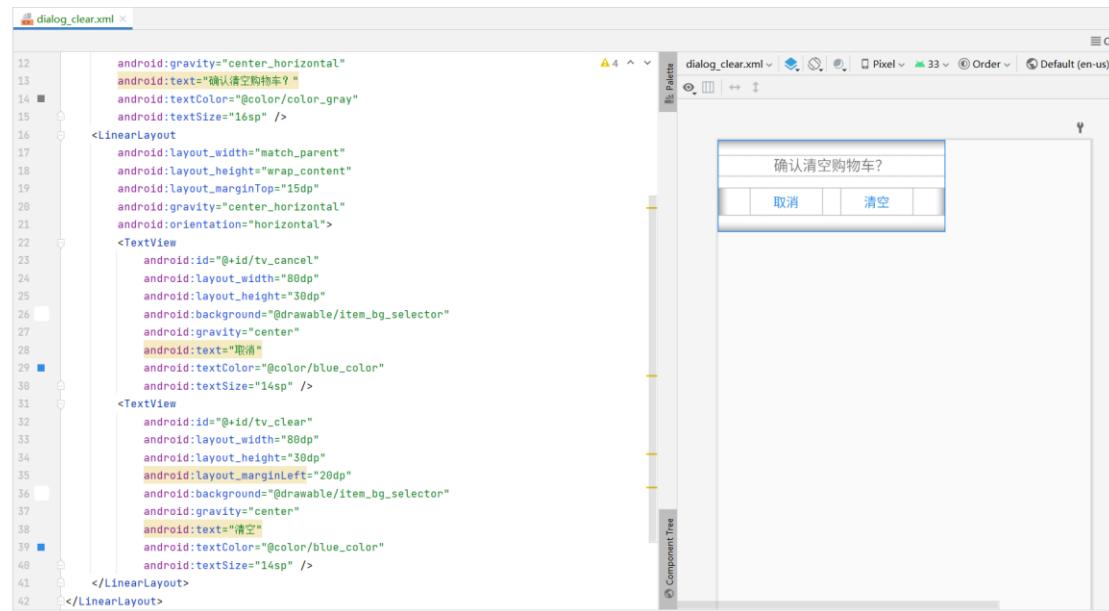
### 3.2.3 清空购物车功能

(1) 在 themes.xml 文件中添加 Dialog 样式。



```
24
25 <style name="Dialog_Style" parent="@android:style/Theme.Dialog">
26     <!-- 设置界面无标题栏-->
27     <item name="android:windowIsFloating">true</item>      <!-- 对话框浮在Activity之上-->
28     <item name="android:windowIsTranslucent">true</item> <!-- 设置对话框背景为透明-->
29     <item name="android:windowNoTitle">true</item>        <!-- 设置界面无标题-->
30     <!-- 设置窗体背景透明-->
31     <item name="android:windowBackground">@android:color/transparent</item>
32     <!-- 设置对话框内容背景透明-->
33     <item name="android:background">@android:color/transparent</item>
34     <!-- 设置对话框背景有半透明遮障层-->
35     <item name="android:backgroundDimEnabled">true</item>
36 </style>
37
38 <style name="Theme.ActivityDialogStyle" parent="Theme.AppCompat.Light.NoActionBar">
39     <item name="android:windowIsTranslucent">true</item>    <!-- 设置对话框背景为透明-->
40     <!-- 设置对话框背景有半透明遮障层-->
41     <item name="android:backgroundDimEnabled">true</item>
42     <item name="android:windowContentOverlay">@null</item> <!-- 设置窗体内容背景-->
43     <!-- 点击对话框外的部分关闭该界面-->
44     <item name="android:windowCloseOnTouchOutside">true</item>
45     <item name="android:windowIsFloating">true</item> <!-- 浮在Activity之上-->
46 </style>
```

(2) 清空购物车布局文件 dialog\_clear.xml，设置 Dialog\_style 样式。



### 3.2.4 店铺详细信息列表适配器

由于店铺详情界面中的菜单列表是用 ListView 控件展示的，所以需要创建一个数据适配器 MenuAdapter 对 ListView 控件进行数据适配。

```

private List<FoodBean> fbl; //菜单
private OnSelectListener onSelectListener; //加入购物车按钮的监听器
public MenuAdapter(Context context, OnSelectListener onSelectListener) {
    this.mContext = context;
    this.onSelectListener = onSelectListener;
}
//设置数据更新界面
public void setData(List<FoodBean> fbl) {
    this.fbl = fbl;
    notifyDataSetChanged();
}
//获取Item的总数
@Override
public int getCount() { return fbl == null ? 0 : fbl.size(); }
//根据position得到对应Item的对象
@Override
public FoodBean getItem(int position) { return fbl == null ? null : fbl.get(position); }
//根据position得到对应Item的id
@Override
public long getItemId(int position) { return position; }

```

```

    /**
     * 根据 position 对应的 Item 视图 position 是当前Item的View
     */
    public View getView(final int position, View convertView, ViewGroup parent) {
        final ViewHolder vh;
        if (convertView == null) {
            vh = new ViewHolder();
            convertView = LayoutInflater.from(mContext).inflate(R.layout.item_food, parent, false);
            vh.tv_food_name = (TextView) convertView.findViewById(R.id.tv_food_name);
            vh.tv_taste = (TextView) convertView.findViewById(R.id.tv_taste);
            vh.tv_sale_num = (TextView) convertView.findViewById(R.id.tv_sale_num);
            vh.tv_price = (TextView) convertView.findViewById(R.id.tv_price);
            vh.btn_add_car = (Button) convertView.findViewById(R.id.btn_add_car);
            vh.iv_food_pic = (ImageView) convertView.findViewById(R.id.iv_food_pic);
            convertView.setTag(vh);
        } else {
            vh = (ViewHolder) convertView.getTag();
        }
        //根据position对应的Item的数据对象
        final FoodBean bean = getItem(position);
        if (bean != null) {
            vh.tv_food_name.setText(bean.getFoodName());
            vh.tv_taste.setText(bean.getTaste());
            vh.tv_sale_num.setText("月售" + bean.getSale());
            vh.tv_price.setText("¥" + bean.getPrice());
            Glide.with(mContext).RequestManager
                .load(string.Constant.WEB_SITE + bean.getImage())
                .error(R.mipmap.ic_launcher)
                .into(vh.iv_food_pic);
        }
        return convertView;
    }
}

class ViewHolder {
    public TextView tv_food_name, tv_taste, tv_sale_num, tv_price;
    public Button btn_add_car;
    public ImageView iv_food_pic;
}

public interface OnSelectListener {
    void onSelectAddCar(int position); //处理加入购物车事件
}

```

### 3.2.5 购物车列表适配器

由于店铺详情界面中的购物车列表是用 ListView 控件展示的，所以需要创建一个数据适配器 CarAdapter 对 ListView 控件进行数据适配。

```

private Context mContext;
private List<FoodBean> fbl;
private OnSelectListener onSelectListener;
public CarAdapter(Context context, OnSelectListener onSelectListener) {
    this.mContext = context;
    this.onSelectListener = onSelectListener;
}
//设置数据更新界面
public void setData(List<FoodBean> fbl) {
    this.fbl = fbl;
    notifyDataSetChanged();
}
//获取Item的总数
@Override
public int getCount() { return fbl == null ? 0 : fbl.size(); }
//根据position得到对应Item的对象
@Override
public FoodBean getItem(int position) { return fbl == null ? null : fbl.get(position); }
//根据position得到对应Item的id

```

```

    /**
     * 根据 position 对应的 Item 的数据对象
     */
    @Override
    public long getItemId(int position) { return position; }
    /**
     * 根据 position 对应的 Item 视图 position 是当前Item的View
     */
    @Override
    public View getView(final int position, View convertView, ViewGroup parent) {
        final ViewHolder vh;
        if (convertView == null) {
            vh = new ViewHolder();
            convertView = LayoutInflater.from(mContext).inflate(R.layout.item_food, parent, false);
            vh.tv_food_name = (TextView) convertView.findViewById(R.id.tv_food_name);
            vh.tv_food_count = (TextView) convertView.findViewById(R.id.tv_food_count);
            vh.tv_food_price = (TextView) convertView.findViewById(R.id.tv_food_price);
            vh.iv_add = (ImageView) convertView.findViewById(R.id.iv_add);
            vh.iv_minus = (ImageView) convertView.findViewById(R.id.iv_minus);
            convertView.setTag(vh);
        } else {
            vh = (ViewHolder) convertView.getTag();
        }
        //根据position对应的Item的数据对象
        final FoodBean bean = getItem(position);
        if (bean != null) {
            vh.tv_food_name.setText(bean.getFoodName());
            vh.tv_food_count.setText(bean.getCount() + "");
            BigDecimal count = BigDecimal.valueOf(bean.getCount());
            vh.tv_food_price.setText("¥" + bean.getPrice());
            vh.iv_add.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    onSelectListener.onSelectAdd(position, tv_food_price);
                }
            });
            vh.iv_minus.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    onSelectListener.onSelectMis(position, tv_food_price);
                }
            });
        }
        return convertView;
    }
}

class ViewHolder {
    public TextView tv_food_name, tv_food_count, tv_food_price;
    public ImageView iv_add, iv_minus;
}

public interface OnSelectListener {
    void onSelectAdd(int position, TextView tv_food_price);
    void onSelectMis(int position, TextView tv_food_price);
}

```

### 3.2.6 店铺详细信息功能小结

店铺详情界面主要是展示店铺信息、菜单列表信息以及购物车信息，其中在菜单列表中可以点击“加入购物车”按钮，将菜品添加到购物车中。此时点击购物车图片会从界面底部弹出一个购物车列表，该列表显示的是购物车中添加的菜品信息，这些菜品信息在列表中可以进行增加和删除。点击购物车列表右上角的“清空”按钮，程序会弹出一个确认清空购物车的对话框，点击对话框中的“清空”按钮会清空购物车中的数据。

### 3.3 菜品功能实现

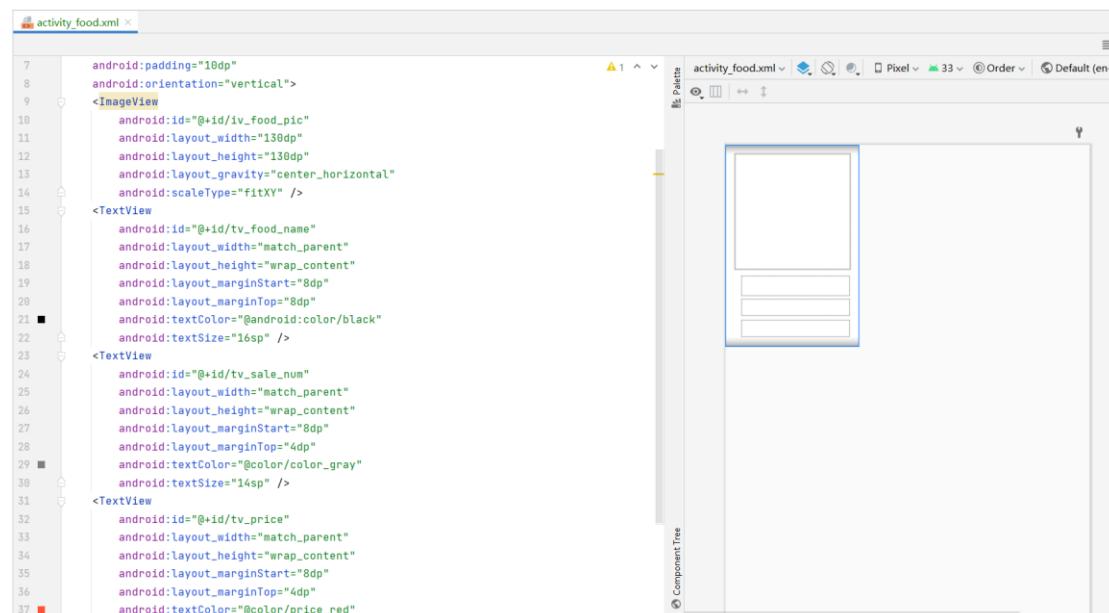
#### 3.3.1 菜品详情界面

(1) 在 themes.xml 文件中创建对话框 Theme.ActivityDialogStyle 样式。



```
24
25 <style name="Dialog_Style" parent="@android:style/Theme.Dialog">
26     <!-- 设置界面无标题栏-->
27     <item name="android:windowIsFloating">true</item>      <!-- 对话框浮在Activity之上-->
28     <item name="android:windowIsTranslucent">true</item>  <!-- 设置对话框背景为透明-->
29     <item name="android:windowNoTitle">true</item>        <!-- 设置界面无标题-->
30     <!-- 设置窗体背景透明-->
31     <item name="android:windowBackground">@android:color/transparent</item>
32     <!-- 设置对话框内容背景透明-->
33     <item name="android:background">@android:color/transparent</item>
34     <!-- 设置对话框背景有半透明遮层-->
35     <item name="android:backgroundDimEnabled">true</item>
36 </style>
37
38 <style name="Theme.ActivityDialogStyle" parent="Theme.AppCompat.Light.NoActionBar">
39     <item name="android:windowIsTranslucent">true</item>      <!-- 设置对话框背景为透明-->
40     <!-- 设置对话框背景有半透明遮层-->
41     <item name="android:backgroundDimEnabled">true</item>
42     <item name="android:windowContentOverlay">@null</item>    <!-- 设置窗体内容背景-->
43     <!-- 点击对话框外的部分关闭该界面-->
44     <item name="android:windowCloseOnTouchOutside">true</item>
45     <item name="android:windowIsFloating">true</item>        <!-- 浮在Activity之上-->
46 </style>
```

(2) 创建 FoodActivity，布局文件为 activity\_food.xml，由 3 个 TextView、1 个 ImageView 控件组成。



#### 3.3.2 菜品数据显示

获取从店铺详情界面传递过来的菜品数据，将数据展示在菜品详情界面上。

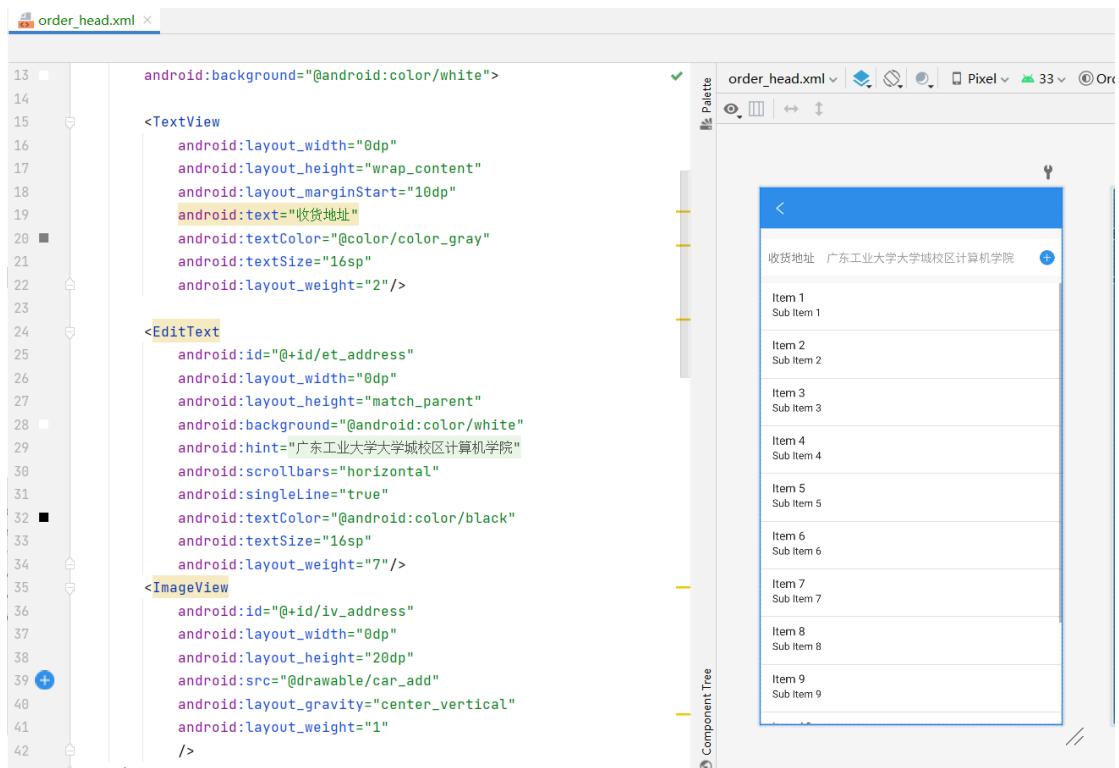
```
FoodActivity.java
1 package com.wangning.order.activity;
2
3 import ...
4
5 public class FoodActivity extends AppCompatActivity {
6
7     private FoodBean bean;
8     private TextView tv_food_name, tv_sale_num, tv_price;
9     private ImageView iv_food_pic;
10
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_food);
16         // 从标题详情界面传递过来的数据
17         bean = (FoodBean) getIntent().getSerializableExtra("name: food");
18
19         initView();
20         setData();
21     }
22
23
24     /**
25      * 初始化界面控件
26     */
27
28     private void initView() {
29         tv_food_name = (TextView) findViewById(R.id.tv_food_name);
30         tv_sale_num = (TextView) findViewById(R.id.tv_sale_num);
31         tv_price = (TextView) findViewById(R.id.tv_price);
32         iv_food_pic = (ImageView) findViewById(R.id.iv_food_pic);
33     }
34
35
36     /**
37      * 设置界面数据
38     */
39
40     private void setData() {
41
42         if (bean == null) return;
43         tv_food_name.setText(bean.getFoodName());
44         tv_sale_num.setText("月售" + bean.getSaleNum());
45         tv_price.setText("¥" + bean.getPrice());
46         Glide.with(activity: this) RequestManager
47             .load(string: Constant.WEB_SITE + bean.getFoodPic()) RequestBuilder<Drawal
48             .error(R.mipmap.ic_launcher)
49             .into(iv_food_pic);
50     }
51
52
53
54
55
56 }
```

### 3.4 订单功能实现

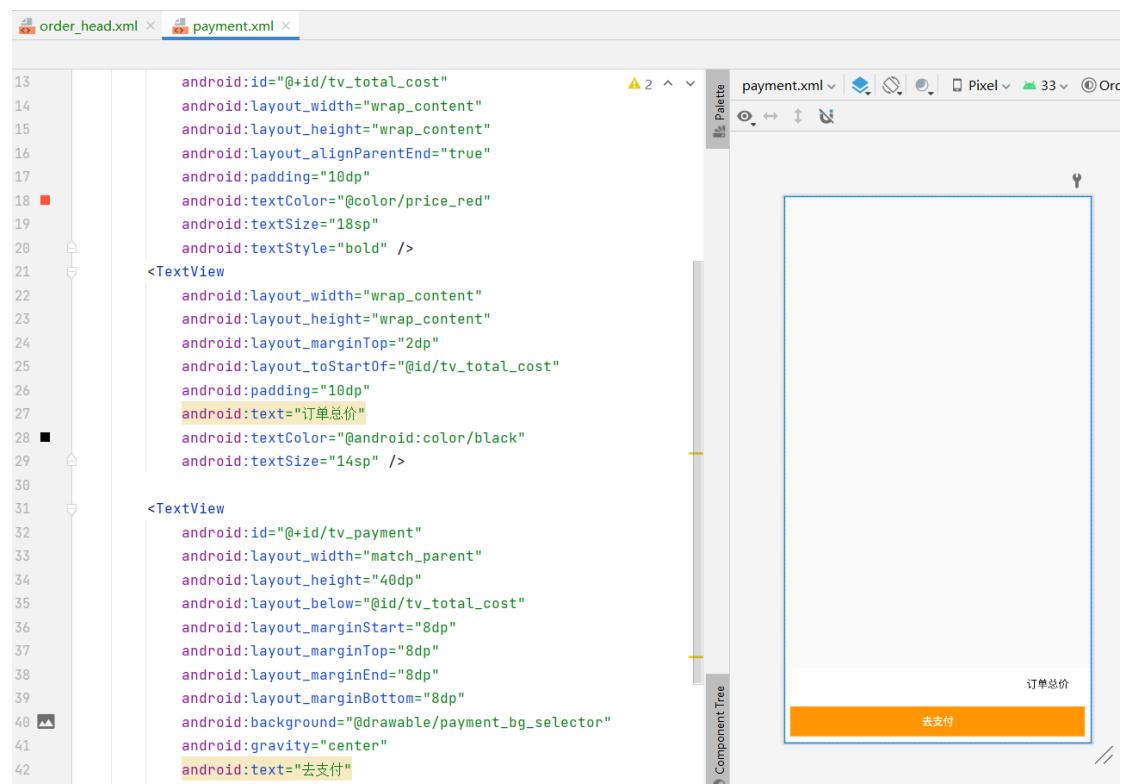
在店铺详情界面，点击“去结算”按钮，程序会跳转到订单界面，订单界面主要展示的是收货地址、订单列表、小计、配送费以及订单总价与“去支付”按钮，该界面的数据是从店铺详情界面传递过来的。点击“去支付”按钮，程序会弹出一个二维码支付界面供用户付款（此处二维码不属于真实业务场景）。

### 3.4.1 订单界面

(1) 订单列表 order\_head.xml。

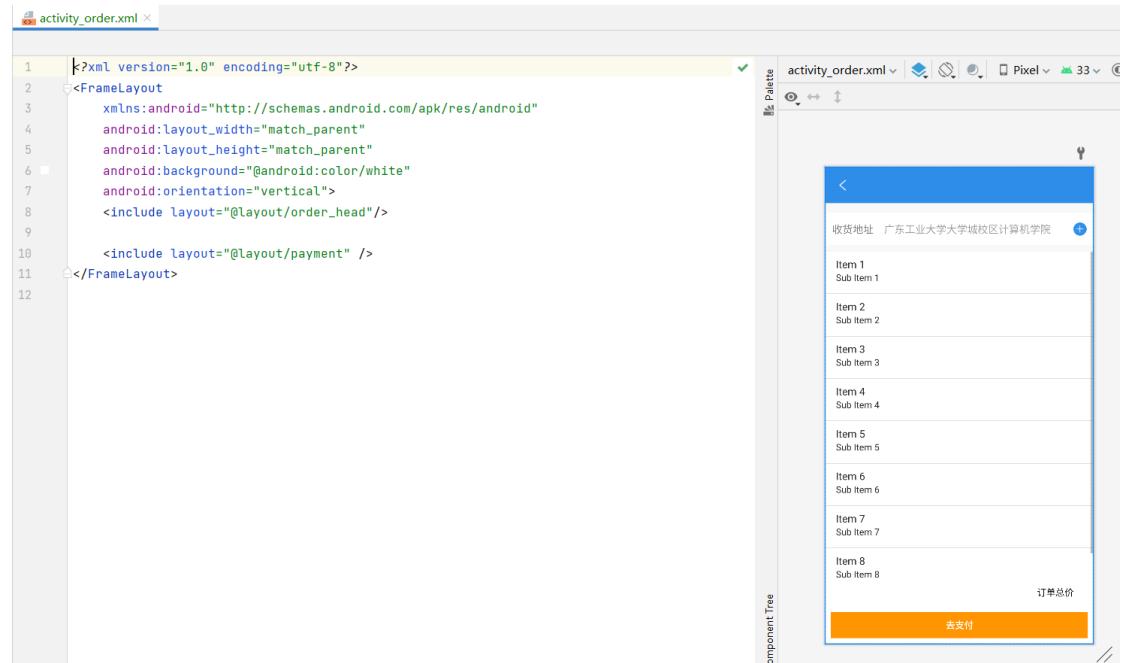


(2) 下方为“去支付”界面 payment.xml。



```
13     android:id="@+id/tv_total_cost"
14     android:layout_width="wrap_content"
15     android:layout_height="wrap_content"
16     android:layout_alignParentEnd="true"
17     android:padding="10dp"
18     android:textColor="@color/price_red"
19     android:textSize="18sp"
20     android:textStyle="bold" />
21
22     <TextView
23         android:layout_width="wrap_content"
24         android:layout_height="wrap_content"
25         android:layout_marginTop="2dp"
26         android:layout_toStartOf="@+id/tv_total_cost"
27         android:padding="10dp"
28         android:text="订单总价"
29         android:textColor="@android:color/black"
30         android:textSize="14sp" />
31
32     <TextView
33         android:id="@+id/tv_payment"
34         android:layout_width="match_parent"
35         android:layout_height="40dp"
36         android:layout_below="@+id/tv_total_cost"
37         android:layout_marginStart="8dp"
38         android:layout_marginTop="8dp"
39         android:layout_marginEnd="8dp"
40         android:layout_marginBottom="8dp"
41         android:background="@drawable/payment_bg_selector"
42         android:gravity="center"
43         android:text="去支付"
```

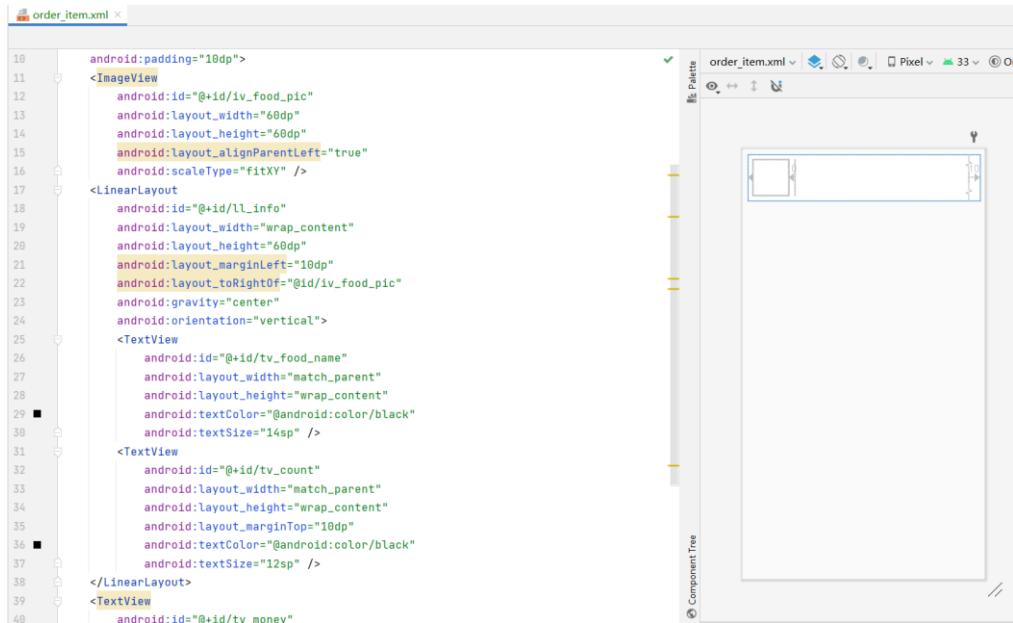
(3) 布局文件 activity\_order.xml 引入 order\_head.xml、payment.xml。



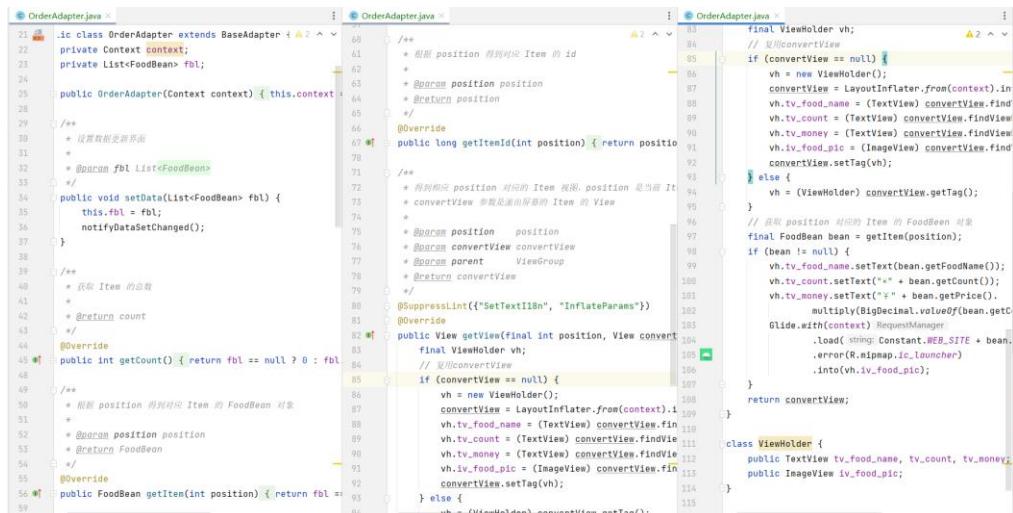
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <FrameLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:background="@android:color/white"
7      android:orientation="vertical">
8      <include layout="@layout/order_head"/>
9
10     <include layout="@layout/payment" />
11 </FrameLayout>
```

### 3.4.2 订单列表条目项

(1) 条目项布局文件 order\_item.xml 由 1 个 ImageView、3 个 TextView 控件组成。

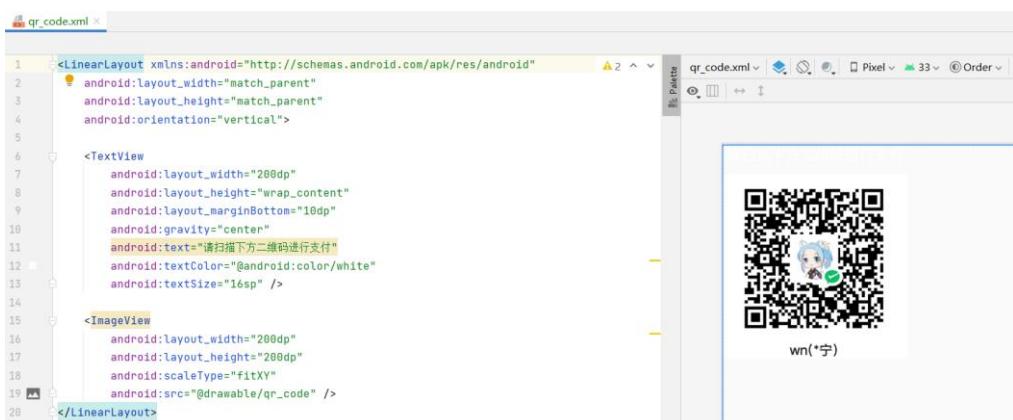


## (2) 订单列表适配器 OrderAdapter 对 ListView 控件进行数据适配。



### 3.4.3 支付订单功能

#### (1) 布局文件 qr\_code.xml 用于放置收款码。



(2) 订单界面从店铺详情界面获取数据，再将数据显示在界面上，实现对订单的支付。

```
OrderActivity.java
21     private ListView lv_order;
22
23     private OrderAdapter adapter;
24     private List<FoodBean> carFoodList;
25     private TextView tv_title, tv_back, tv_distribution_cost, tv_total_cost,
26     tv_cost, tv_payment;
27     private BigDecimal money, distributionCost;
28     private EditText et_address;
29     private ImageView iv_address;
30
31     @Override
32     protected void onCreate(Bundle savedInstanceState) {
33         super.onCreate(savedInstanceState);
34         setContentView(R.layout.activity_order);
35         carFoodList = (List<FoodBean>) getIntent().getSerializableExtra("carFoodList");
36         money = new BigDecimal(getIntent().getStringExtra("totalMoney"));
37         distributionCost = new BigDecimal(getIntent().getStringExtra("distributionCost"));
38         initView();
39         setData();
40     }
41
42     /**
43      * 初始化界面控件
44      */
45     private void initView() {
46         tv_title = (TextView) findViewById(R.id.tv_title);
47         tv_title.setText("订单");
48         tv_back = (TextView) findViewById(R.id.tv_back);
49         lv_order = (ListView) findViewById(R.id.lv_order);
50         tv_distribution_cost = (TextView) findViewById(R.id.tv_distribution_cost);
51         tv_total_cost = (TextView) findViewById(R.id.tv_total_cost);
52         tv_cost = (TextView) findViewById(R.id.tv_cost);
53         tv_payment = (TextView) findViewById(R.id.tv_payment);
54
55         et_address = (EditText) findViewById(R.id.et_address);
56         et_address.setHorizontalScrollBaring(true);
57         iv_address = (ImageView) findViewById(R.id.iv_address);
58
59         // 选择地址
60         iv_address.setOnClickListener(v -> {
61             et_address.setText("广东工业大学大学城校区计算机学院");
62         });
63
64         // 返回键的点击事件
65         tv_back.setOnClickListener(v -> finish());
66         tv_payment.setOnClickListener(view -> {
67             Log.v("tag", et_address.getText().toString().trim());
68             if (!et_address.getText().toString().trim().equals("")) {
69                 Dialog dialog = new Dialog(context: OrderActivity.this, R.style.Dialog);
70                 dialog.setContentView(R.layout.qr_code);
71                 dialog.show();
72             } else {
73                 Toast.makeText(context: this, text: "请输入您的地址", Toast.LENGTH_SHORT).show();
74             }
75         });
76
77         /**
78          * 设置界面数据
79          */
80         private void setData() {
81             adapter = new OrderAdapter(context: this);
82             lv_order.setAdapter(adapter);
83             adapter.setData(carFoodList);
84             tv_cost.setText("¥" + money);
85             tv_distribution_cost.setText("¥" + distributionCost);
86             tv_total_cost.setText("¥" + (money.add(distributionCost)));
87         }
88     }
89 }
```

### 3.5 启动应用功能

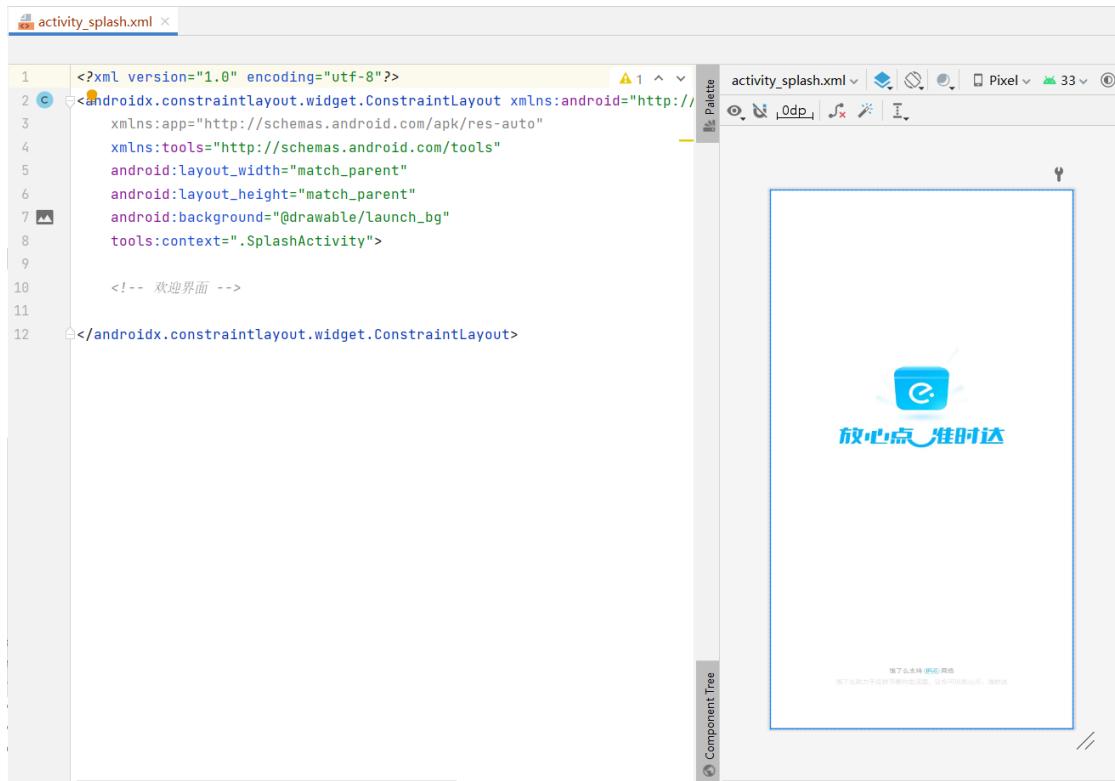
### 3.5.1 启动 Tomcat 服务器

将资源文件 img、shop\_list\_data.json 放入\tomcat9\webapps\ROOT\order 目录下。启动\tomcat9\bin\startup.bat。即已启动本地 Tomcat 服务器传递数据。Ctrl+C 则关闭 Tomcat。

Tomcat  
13-May-2023 05:27:02.696 潤℃他 [main] org.apache.catalina.startup.Catalina.load 鏈嶅姪鍇ㄥ濬[451]姣 錢哩濮嬪嬪  
13-May-2023 05:27:02.756 潤℃他 [main] org.apache.catalina.core.StandardService.startInternal 姝 e濬鍚 姮鍊嶅姪[Catalin  
a]  
13-May-2023 05:27:02.757 潤℃他 [main] org.apache.catalina.core.StandardEngine.startInternal 姮 e濬鍚 姮 Servlet 察晨捄  
銆佹Apache Tomcat/9.0.54]  
13-May-2023 05:27:02.768 潤℃他 [main] org.apache.catalina.startup.HostConfig.deployDirectory 鍣安eb 塞旂妝紓嬪簷閭ㄧ讲  
銆扮泊寢:[G:\Dev\_wn\study\_wn\gdut-android\project-order\tomcat9\webapps\docs]  
  
13-May-2023 05:27:03.077 瑞～愰 [main] org.apache.catalina.util.SessionIdGeneratorBase.createSecureRandom 浃跨妝[SHA1PRN  
G]鍊夾缓浼氱瘞ID鑿燔垚鑽鑻嶄垂浜呰132]姣 鎗?  
  
13-May-2023 05:27:03.100 潤℃他 [main] org.apache.catalina.startup.HostConfig.deployDirectory Web塞旂妝紓嬪簷鍊 綽[G:\D  
ev\_wn\study\_wn\gdut-android\project-order\tomcat9\webapps\docs]鑽勸儼緝插凡鏩\_332]姣 錢哩畲?  
13-May-2023 05:27:03.101 潤℃他 [main] org.apache.catalina.startup.HostConfig.deployDirectory 鍄安eb 塞旂妝紓嬪簷閭ㄧ讲  
銆扮泊寢:[G:\Dev\_wn\study\_wn\gdut-android\project-order\tomcat9\webapps\manager]  
  
13-May-2023 05:27:03.151 潤℃他 [main] org.apache.catalina.startup.HostConfig.deployDirectory Web塞旂妝紓嬪簷鍊 綽[G:\D  
ev\_wn\study\_wn\gdut-android\project-order\tomcat9\webapps\manager]鑽勸儼緝插凡鏩\_50]姣 錢哩畲?  
  
13-May-2023 05:27:03.157 潤℃他 [main] org.apache.catalina.startup.HostConfig.deployDirectory 鍄安eb 塞旂妝紓嬪簷閭ㄧ讲  
銆扮泊寢:[G:\Dev\_wn\study\_wn\gdut-android\project-order\tomcat9\webapps\ROOT]  
  
13-May-2023 05:27:03.179 潤℃他 [main] org.apache.catalina.startup.HostConfig.deployDirectory Web塞旂妝紓嬪簷鍊 綽[G:\D  
ev\_wn\study\_wn\gdut-android\project-order\tomcat9\webapps\ROOT]鑽勸儼緝插凡鏩\_22]姣 錢哩畲?  
  
13-May-2023 05:27:03.184 潤℃他 [main] org.apache.coyote.AbstractProtocol.start 察€濮嬪岡璁 鍔喟莽鍛刑"http-nio-8080  
"]  
13-May-2023 05:27:03.312 潤℃他 [main] org.apache.catalina.startup.Catalina.start [615]姣 錫窟涓鏁” 嫵鍚

### 3.5.2 模仿欢迎界面

模仿饿了么的欢迎界面实现 activity\_splash.xml。



### 3.5.3 项目启动入口

SplashActivity 为项目主活动，项目由此启动，启动欢迎界面 3 秒后跳转至 ShopActivity。

```

    package com.wangning.order;

    import ...

    public class SplashActivity extends AppCompatActivity {

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_splash);
            init();
        }

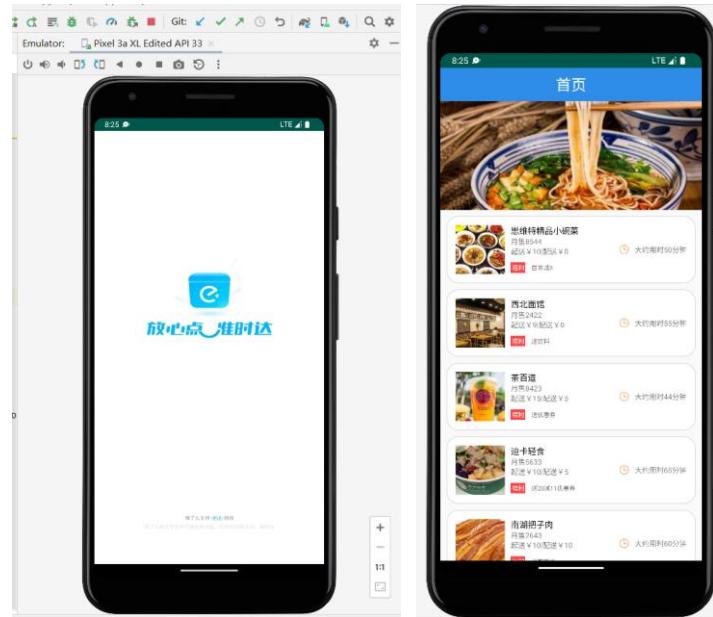
        /**
         * 启动欢迎界面展示 3 秒后跳转至 ShopActivity
         */
        private void init() {
            // 利用 Timer 让此界面延迟 3 秒后跳转，timer 中有一个线程，这个线程不断执行 task
            Timer timer = new Timer();
            // TimerTask 实现 Runnable 接口，TimerTask 类表示一个在指定时间内执行的 task
            TimerTask task = () -> {
                Intent intent = new Intent(packageContext: SplashActivity.this, ShopActivity.class);
                startActivity(intent);
                SplashActivity.this.finish();
            };
            timer.schedule(task, delay: 3000);
        }
    }

```

## 4 实现效果展示

### 4.1 启动项目

由 `SplashActivity.java` 启动项目，显示欢迎界面，3 秒钟后跳转至店铺界面。



### 4.2 店铺界面功能演示

点击店铺进入店铺详情界面。

#### 4.2.1 广告 Banner

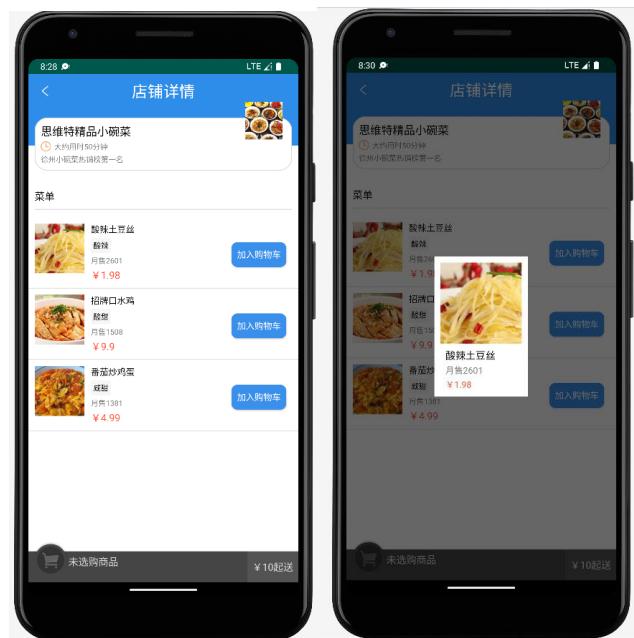
广告 Banner 本身在自动轮播，也可手动滑动。



## 4.3 店铺详情界面

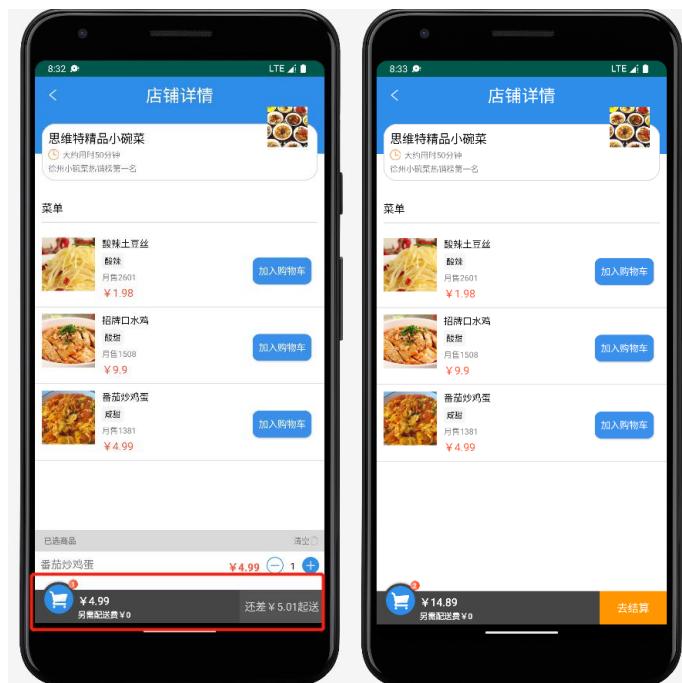
### 4.3.1 菜品信息

点击单个菜品列表项会弹出菜品信息对话框显示。

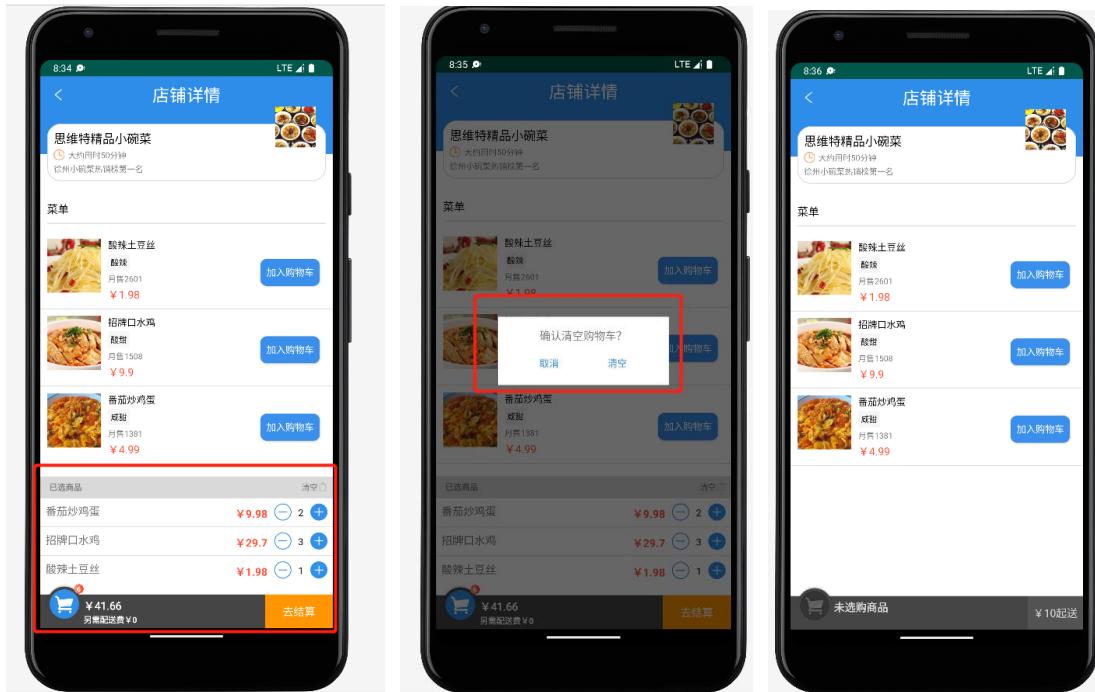


### 4.3.2 购物车功能

点击加入购物车会自动加入到下方购物车栏。当总价低于起送费时，不可支付；当总价高于起送费时，可支付。

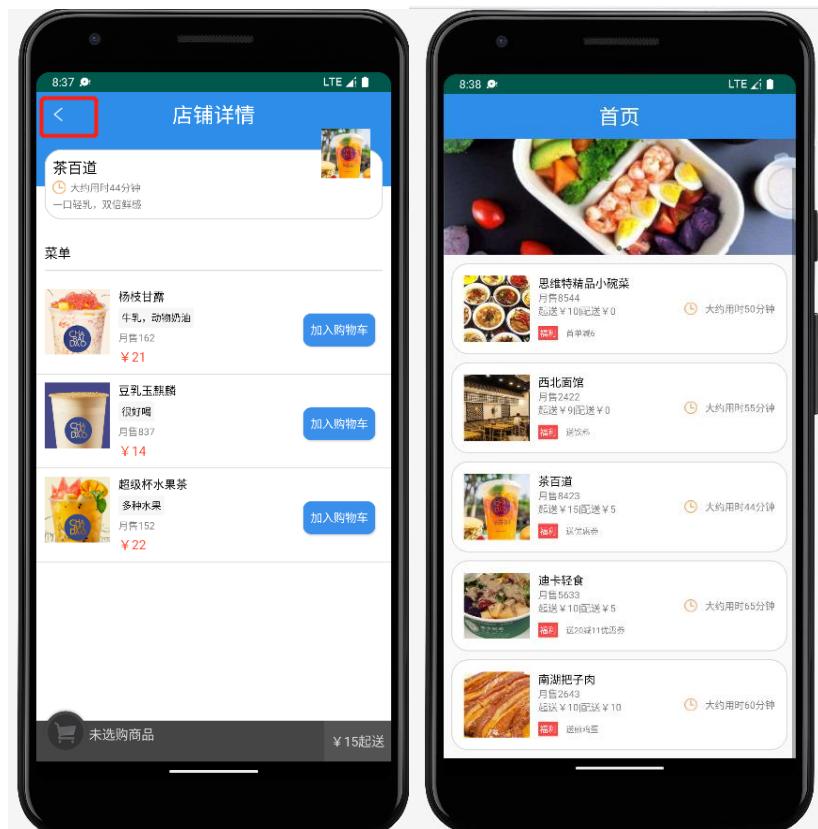


点击购物车，会显示订单信息。购物车栏里也可加减商品，自动计算总价，也可清空。点击清空弹出清空提示框，点击清空则清空购物车。



#### 4.3.3 返回至店铺界面

点击标题栏返回按钮则返回至店铺界面。



## 4.4 订单界面

### 4.4.1 支付结算

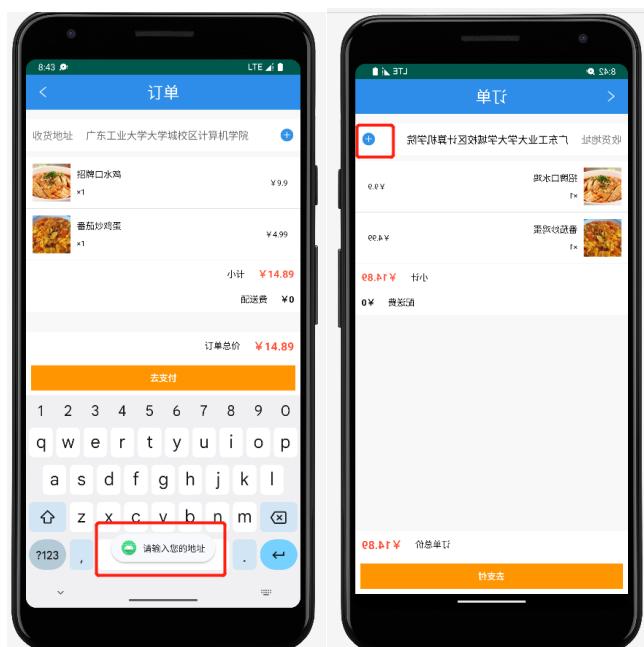
点击购物车右方“去支付”，跳转至订单界面。



### 4.4.2 收货地址

若未填写收货地址，会 Toast 一条提示信息。

点击收货地址栏右方“+”，则会自动填写“广东工业大学大学城校区计算机学院”作为收货地址。



#### 4.4.3 支付二维码

在订单界面点击“去支付”弹出支付二维码对话框。



点击二维码对话框外任意空白位置会取消支付。



#### 4.4.4 返回至店铺详情界面

点击标题栏返回按钮则返回至店铺界面。



## 5 心得体会

在实验和大作业实现过程中，我深刻体会到界面和逻辑分离思想的重要性，能更加清晰的书写代码逻辑。平时可以积累一些高质量轮子，也可以使用 GitHub 里开源的组件库，能更加快捷、美观地实现界面和逻辑。

这次课程让我对 Android 客户端开发有了一定的了解，也许未来不会从事客户端开发，但是课程对我的代码复现能力、逻辑能力有一定的提升帮助，十分感谢。