**移动互联网应用开发课程设计个人报告**

2020-2021学年 第二学期，2019级本科，计算机科学与技术

**项目名称：MOTA学习魔塔**

**姓名：陈语奇**

**学号：1906010232**

**承担任务：UI制作**

**评分：**

**1.任务概述**

本人主要负责这个APP的UI实现，实现整个APP的UI界面以及部分页面跳转的效果，根据UI界面设计，优化并完成所给设计的所有UI组件及界面。UI的实现为后端的实现提供了可视化的界面，为后端提供了方便，同时也是整个APP的门面。

**2.设计思路**

初期界面设计由其他组员提供，根据其他组员提供的设计，实现并完善UI界面。

2.1标题界面、登陆界面、注册界面

这三个界面的实现较为简单，如下图2-1，2-2，2-3所示，考虑采用LinearLayout嵌套布局实现。

|  |  |  |
| --- | --- | --- |
| 2-1标题界面 | 2-2登陆界面 | 2-3 注册界面 |

2.2 APP主界面

主界面一共有五个，如下图2-4至2-8所示，由于界面的上端标题栏、用户信息以及下方底部的按钮栏目变动较小，考虑使用一个Activity，采取在当前Activity上才去切换Fragment的方式来实现页面切换的功能。顶部标题工具栏可以使用自定义的toolbar实现。每个页面的条目显示由于会有很多条，需要通过滑动显示全部，我考虑使用ListView控件来实现。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2-4习惯页面 | 2-5待办事项页面 | 2-6每日任务页面 | 2-7奖励页面 | 2-8菜单页面 |

2.3成就页面、账单页面

成就和账单页面较为类似，如下图2-9和2-10所示考虑使用LinearLayout，滑动条目仍然使用ListView控件实现。

|  |  |
| --- | --- |
| 2-9 成就页面 | 2-10 账单页面 |

2.4 新建条目页面

四个新建条目界面都比较类似，由于是在Activity上方显示，所以使用Dialog类来实现显示操作，布局仍然使用LinearLayout嵌套布局。

|  |  |  |  |
| --- | --- | --- | --- |
| 2-11 新建习惯 | 2-12 新建待办事项 | 2-13 新建每日任务 | 2-14 新建奖励 |

**3.技术路线与方法**

由于本人主要负责UI实现，所以下面介绍每个页面具体的布局设计。其中的图片文件由阿里巴巴矢量图标库以及好友友情提供。

3.1标题界面、登陆界面、注册界面

3.1.1标题界面

标题界面主要是由一个图标，APP的标题，以及登录和注册按钮构成，如下左图3-1所示。LinearLayout主要布局如下面右侧代码所示。每个控件的layout\_gravity值设置的均为center使得控件在整个布局居中显示。

|  |  |
| --- | --- |
| 3-1 标题界面 | <**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"  android:background="@color/main\_background"  android:orientation="vertical"  tools:context=".Login"**>  <**ImageView ...** >  <**TextView ...** >  <**TextView ...** >  <**Button ...** >  <**Button ...** > </**LinearLayout**> |

3.1.2 登陆界面

登陆界面和标题界面布局类似，如下左图3-2所示，同样是使用LinearLayout，登录按钮使用了圆角，实际上是自己重新写了一个btn\_shape.xml的样式，如下右侧代码所示，设置了背景色，圆角以及文字与按钮四周的间隙。后面的注册界面也同样运用了这个样式。

记住密码采用的是CheckBox控件，下面的忘记密码也是一个按钮，不过我将背景更换为了透明色（android:background="@android:color/transparent"），并且在string.xml中增加了下划线（<string name="forget\_psd"><u>忘记密码</u></string>）呈现出这种效果（遗憾的是这个忘记密码暂未实现其功能）。

|  |  |
| --- | --- |
| 3-2登陆界面 | *<?***xml version="1.0" encoding="utf-8"***?>* <**shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle"**>  *<!-- 设置背景颜色 -->* <**solid android:color="@color/white"** />  *<!-- 设置圆角 -->* <**corners android:radius="20dp"**/>  *<!-- 设置文字与按钮的四周间隙 -->* <**padding  android:left="10dp"  android:right="10dp"  android:top="10dp"  android:bottom="10dp"** /> </**shape**> |

3.1.3注册界面

同样的，与前两个界面类似，仍然使用的是LinearLayout布局，如下图3-3所示，不同的是在手机号/邮箱那一处使用了布局的嵌套，使得Button与EditText组成一行。布局嵌套是之后经常要使用到的一种方法，后面不再过多描述。

|  |
| --- |
| 3-3注册界面 |

3.2APP主界面

主界面只由一个Activity构成基本结构，如下图和代码所示。

|  |
| --- |
| 3-4 主界面结构 |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"**>  <**androidx.appcompat.widget.Toolbar  android:id="@+id/app\_toolbar"  android:layout\_width="match\_parent"  android:layout\_height="50dp"**>  </**androidx.appcompat.widget.Toolbar**>  <**FrameLayout  android:id="@+id/frame\_content"  android:layout\_weight="1"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:background="@color/app\_background"  android:layout\_below="@+id/app\_toolbar"  android:layout\_above="@+id/app\_bottom"**>  </**FrameLayout**>  <**include  layout="@layout/bottom"  android:id="@+id/app\_bottom"  android:layout\_height="50dp"  android:layout\_width="match\_parent"  android:layout\_alignParentBottom="true"**>  </**include**> </**RelativeLayout**> |

在主界面Activity中，使用了相对布局，即RelativeLayout。最上方用了Toolbar控件，用来自定义顶部工具栏。最下方用了自己写的bottom.xml布局文件，再使用include导入。由于之前使用LinearLayout布局时，在输入文字的时候会导致底部菜单栏被键盘挤压到上方，所以采用了相对布局，使得底部菜单栏能够保持在最下方。中间部分嵌套了一个FrameLayout，以便于后期在五个界面使用Fragment的来回切换，只需要在这个给定的FrameLayout上进行渲染就可以，十分方便且快捷。

3.2.1主界面的自定义布局

1.底部菜单栏

底部菜单栏保存在bottom.xml文件中，具体结构如下图3-5所示。

|  |
| --- |
| 3-5底部菜单栏结构 |

该底部菜单栏实现较为简单，由五个ImageView控件水平线性布局排列而成。由于之后要设置监听所以必不可少的代码是要对这五个ImageView设置可获取焦点（android:focusable="true"），否则无法设置监听。

2.顶部菜单栏

顶部菜单栏定义在menu文件下app\_menu.xml当中，实现的效果如下左图3-6所示，代码如下右所示

|  |  |
| --- | --- |
| 3-6顶部菜单栏 | *<?***xml version="1.0" encoding="utf-8"***?>* <**menu xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"**>  <**item  android:id="@+id/action\_search"  android:icon="@drawable/search\_icon"  app:showAsAction="always" app:actionViewClass="androidx.appcompat.widget.SearchView"  android:title="Search"** />  <**item  android:id="@+id/menu\_add"  android:icon="@drawable/add\_icon"  android:title="Add"  app:showAsAction="always"** /> </**menu**> |

两个item分别为查找按钮及增加按钮，其中showAsAction=”always”属性，表示一直显示。运用自定义的menu可以在后端代码中使用自带的类更加方便的调用，使得UI更加美观。

下面是后端中对自定义menu的初始化以及按钮点击功能的实现，使用setSupportActionBar()方法实现对toolbar的初始化。然后重写onCreateOptionsMenu(Menu menu)方法实现渲染toolbar样式以及对toolbar按钮上的监听。后端的成员只需要再重写监听方法就可以实现效果了。

|  |
| --- |
| *//toolbar初始化*  **toolbar** = (Toolbar) findViewById(R.id.***app\_toolbar***); setSupportActionBar(**toolbar**); getSupportActionBar().setHomeButtonEnabled(**false**); getSupportActionBar().setDisplayHomeAsUpEnabled(**false**); |
| @Override **public boolean** onCreateOptionsMenu(Menu menu) {  *//toolbarOptionmenu 监听 inflate* getMenuInflater().inflate(R.menu.***app\_menu***,menu);  **mySearch** = menu.findItem(R.id.***action\_search***);  **add\_btn** = menu.findItem(R.id.***menu\_add***);  SearchView mySearchView = (SearchView) **mySearch**.getActionView();  mySearchView.setQueryHint(**"Search"**);  *// toolbar查找* mySearchView.setOnQueryTextListener(**new** SearchView.OnQueryTextListener() {  @Override  **public boolean** onQueryTextSubmit(String query) { **return false**; }  @Override  **public boolean** onQueryTextChange(String newText) {**return false**; }  });  *//toobar 增加* **add\_btn**.setOnMenuItemClickListener(**new** MenuItem.OnMenuItemClickListener() {  @Override  **public boolean** onMenuItemClick(MenuItem item) {**return true**; }  });  **return super**.onCreateOptionsMenu(menu); } |

3.2.2fragment布局定义

本APP一共有五个需要切换的页面，分别为习惯fragment、待办事项fragment、每日任务fragment、奖励fragment和菜单fragment，每个fragment基本组成相差无几，以习惯fragment为例，如下图3-7所示。

|  |
| --- |
| 3-7 习惯fragment结构 |

每个fragment都是用帧布局嵌套线性布局，即外部是FrameLayout嵌套内部的LinearLayout，其中上面是一个自定义的布局文件user\_info,xml，用来显示用户的个人信息下面是一个ListView控件，用来显示各种各样的条目（菜单fragment除外）。代码如下。

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**FrameLayout 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=".habit"**>  *<!--* ***TODO: Update blank fragment layout*** *-->* <**LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"  android:paddingTop="5dp"**>  <**include layout="@layout/user\_info"**></**include**>  <**ListView  android:id="@+id/habit\_lv"  android:layout\_marginTop="10dp"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:layout\_marginLeft="10dp"  android:layout\_marginRight="10dp"**>  </**ListView**>  </**LinearLayout**> </**FrameLayout**> |

user\_info.xml布局文件用于显示用户信息，我们需要显示用户的头像、用户名、等级、HP（血量）、EXP（经验值）以及所拥有的钻石数量。如下图3-8所示。

|  |
| --- |
| 3-8 用户信息（user\_info.xml） |

再这个布局文件中由于需要显示HP、EXP等的占比值，所以使用了进度条ProgressBar控件，下面具体介绍进度条这一控件，以HP值为例，布局文件如下所示。

|  |
| --- |
| <**ProgressBar  android:id="@+id/hp\_progressBar"  android:layout\_width="210dp"  android:layout\_height="15dp"  android:layout\_marginLeft="10dp"  android:max="100"  android:progress="100"  android:progressDrawable="@drawable/hp\_progress"  style="@android:style/Widget.ProgressBar.Horizontal"**></**ProgressBar**> |

这是进度条的基本属性展示，其中android:max属性指的是进度条的最大值，android:progress属性指的是当前进度条的值，然后style属性这里我使用的是横向的进度条所以是android:style/Widget.ProgressBar.Horizontal，android:progressDrawable属性则是我自定义了一个xml文件用于渲染进度条的渐变色，使得整体UI更加的美观，下面具体介绍自定义的渲染xml文件。代码如下所示。

|  |
| --- |
| <**layer-list xmlns:android="http://schemas.android.com/apk/res/android"** >  <**item android:id="@android:id/background"**>  *<!--形状-->* <**shape**>  <**corners android:radius="20dp"** />  <**solid android:color="@color/progress\_bar\_background"**></**solid**>  </**shape**>  </**item**>  <**item android:id="@android:id/progress"**>  <**clip**>  <**shape**>  <**corners android:radius="20dp"** />  <**gradient  android:angle="0"  android:startColor="#80F44336"  android:endColor="#CBFF0000"**/>  </**shape**>  </**clip**>  </**item**> </**layer-list**> |

首先是最外层的layer-list，内部有两个item一个定义形状，我设置了进度条的圆角为20dp，另一个定义渐变色。<gradiet>下的属性startColor为起始的颜色endColor属性为终止的颜色。

最后单独介绍菜单fragment，菜单fragment的下方控件与其他的不同，仅仅由两个ImageButton和TextView组成，如下图所示3-9所示。

|  |
| --- |
| 3-9 菜单fragment |

如何将按钮显示为原型只需要在其按钮背景布局中将其圆角设置为原先按钮长度的一半，就可以达到圆形得到效果。

3.2.3ListView中的item设计

由于前四个fragment的结构类似，这里不做过多的赘述。这部分将着重介绍对于每个ListView中显示的条目即item样式。

1.习惯item

习惯item由三个部分组成左右两边是两个ImageButton，用于点击记录完成次数。中间是一个TextView，用于显示习惯的内容和标签。如下图3-10所示。

|  |
| --- |
| 3-10习惯item |

习惯item布局文件依旧采取了最简单的水平线性布局，并且额外设置了分割线属性（android:outlineAmbientShadowColor="@color/black"）使得界面更加美观，中间的Content和Label则是又嵌套了纵向线性布局，并对其颜色大小进行了适当的修改，需要注意的是，由于ImageButton要显示在两侧，所以外部的线性布局的宽度必须要设置为match\_parent。（android:layout\_width="match\_parent"）。再对每个控件设置layout\_weight达到这样的效果。部分代码如下所示。

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="horizontal"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:paddingLeft="10dp"  android:paddingRight="10dp"  android:paddingTop="5dp"  android:paddingBottom="12dp"  android:outlineAmbientShadowColor="@color/black"  android:gravity="center\_vertical"**>  <**ImageButton ...>**  <**LinearLayout  android:layout\_width="wrap\_content"  android:layout\_height="50dp"  android:orientation="vertical"  android:layout\_weight="1"  android:background="@color/rice\_white"**>  <**TextView ...>**  <**TextView ...>**  </**LinearLayout**>  <**ImageButton ...>** </**LinearLayout**> |

2.待办事项、每日任务item

这两个item由于样式相同，所以只写了一个xml文件。由于存在完成任务会变灰的需求，所以写了一个未完成的样式以及一个已完成的样式，由于版本更新的缘故，仍然保留了只打勾不变色的版本，可以随时更换。如下图3-11、3-12所示。

|  |  |
| --- | --- |
| 3-11 每日任务item | 3-12 待办事项item |

这个item的布局文件与上面所写的类似，只是缺少了右侧的ImageButton，所以左侧的背景仍然沿用选择了habit\_item\_btn\_left.xml这一布局文件（android:background="@drawable/habit\_item\_btn\_left"），只对其中的图片文件进行了修改（android:src="@drawable/unfinished"），由于找到的图片会过大或过小，我在对其属性设置了适应大小（android:scaleType="fitCenter"）。

在灰色item中你可以看到，左侧的Button与右侧的TextView中存在一条竖线，如果仅仅修背景颜色，即background属性的值，是没有办法做到的，所以我对其背景又重写了一个xml文件并调用。代码如下。

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**layer-list xmlns:android="http://schemas.android.com/apk/res/android"**>  <**item**><**shape** >  <**solid android:color="@color/light\_gray"** />  <**corners  android:topLeftRadius="0dp"  android:topRightRadius="20dp"  android:bottomRightRadius="20dp"  android:bottomLeftRadius="0dp"**/>  </**shape**></**item**>  <**item android:bottom="-4dp" android:right="-4dp" android:top="-4dp"**>  <**shape**><**stroke android:width="1dp" android:color="@color/dark\_gray"**/></**shape**>  </**item**> </**layer-list**> |

仍然是外部一个layer-list，第一个item设置的圆角属性（<coners>标签），我只需要右侧的两个为圆角，所以android:topRightRadius，android:bottomRightRadius设置为20dp。下面一个item我需要只显示左边框，我就必须将上下右边先设置为-4dp（item android:bottom="-4dp" android:right="-4dp" android:top="-4dp"），然后对左边设置边框线条（<stroke>标签）。

3.奖励item

这个item与上面的也比较类似，原先打算沿用左侧一个TextView，右侧一个ImageButton的格式，但是后来发现，右侧还需要显示文字，于是放弃使用ImageButton，考虑使用最基础的Button控件。效果图如下图3-13所示。

|  |
| --- |
| 3-13 奖励item |

右侧的按钮背景由于与习惯item右侧按钮背景一致，仍然沿用了其xml文件。左侧的显示界面背景仍然是自定的一个xml文件，由于这次是左侧需要圆角，所以只需稍微修改一些属性的值便可以实现效果。

关于右侧如何既显示图片又显示文字，代码如下所示。

|  |
| --- |
| <**Button  android:layout\_width="50dp"  android:layout\_height="50dp"  android:background="@drawable/habit\_item\_btn\_right"  android:paddingLeft="5dp"  android:drawableLeft="@drawable/diamond\_small"  android:text="10"  android:layout\_gravity="right"**> </**Button**> |

从中我们可以看出，只需要增加android:drawableLeft这一属性，便可以使图片添加在文字的左侧。

在user\_info.xml中的钻石图标与这个相同，我没有使用diamond.png而是又重新下载了一个更小的png图片，是由于过大的png图片文件显示在按钮中会显示的很大，由于本人能力原因没有办法像ImageView中调节其scale属性。所以我选择了另一个方法找到一个更小的图片。

3.3 成就界面、账单界面

成就界面和账单界面布局类似，也是由上下两个部分构成，最上方是一个ImageButton构成的返回按钮，中上方显示图片，数据等，下方是ListView显示条目。如下图3-14，3-15所示。

|  |  |
| --- | --- |
| 3-14成就界面 | 3-15 账单界面 |

由于实现起来较为容易，且与之前布局类似，这里不再过多赘述。主要将介绍两个部分的item样式。

3.3.1成就item样式

与之前的待办事项item类似，成就item也拥有原色与灰色两种形式。如下图3-16所示。

|  |
| --- |
| 3-16 成就item |

成就item的结构与之前的待办事项item结构极其类似，只需要更换背景颜色，图片文件即可。

3.3.2账单item

账单item与习惯item相类似，左侧是ImageButton，右侧更换为了TextView。由于存在收入和支出，需要写两个不同的xml文件。如下图3-17所示。

|  |
| --- |
| 3-17 账单item |

对于背景，都是自定义的xml文件，在xml文件中直接修改，方便且快捷。前面已经写过很多类似的，到这里已经是轻车熟路。

3.4 用来增加条目的dialog

在习惯、每日任务、待办事项、奖励中都需要增加新的条目（点击右上角的加号），屏幕上会弹出一个对话框用来填写增加的信息。对于每一个增加的界面，都需要一个xml来自定义dialog样式。

增加的页面基本都比较相似，这里以增加dialog为例具体说明，效果如下图3-18所示。

|  |
| --- |
| 3-18 增加习惯dialog |

整体上仍然是才去垂直的线性布局，这个页面对布局的嵌套要求较高，考虑到美观性与和谐性的统一，需要对每个控件的位置进行排版，其中padding、margin、layout\_weight等属性需要熟练使用。难度不大，但是需要细心和耐心。

剩余的dialog页面如下图3-19、3-20、3-21所示，其本质与上面所示没有差别。结构也都基本类似，在这里不再做过多赘述。

|  |  |  |
| --- | --- | --- |
| 3-19 增加每日任务dialog | 3-20 增加待办事项dialog | 3-21 增加奖励dialog |

与其他的不同的是在增加待办事项dialog中最下方的选择实时间应当是点击之后出现日期选择器然后用户选择日期后显示在EditText中，这一点将会在后文说道。

3.5页面之间的转换

|  |
| --- |
| 3-22页面之间的转换 |

页面间的转换如上图所示，目前采取Intent执行页面之间的相互转化，后期可以使用putExtra()方法增加页面之间传递的参数。

**4.实现效果及测试用例**

在虚拟机上运行时UI界面较为完好，在手机上运行时，会出现尺寸不匹配显示不全的问题，在对代码进行尺寸修改后，最终在手机APP上显示的效果如下所示。

|  |  |  |
| --- | --- | --- |
| 标题界面 | 登陆界面 | 注册界面 |

当APP打开时，显示的是标题界面，此时点击登录后，出现登陆界面，登陆界面可以选择记住密码，下次登陆时可以不用输入密码（后端，小组成员工作）。忘记密码按钮此版本暂未实现。在标题界面如果点击注册按钮，则会进入注册界面，输入用户名、手机号/邮箱、验证码和密码。所有的密码设置了inputText属性为textPassword，显示时为圆点。

页面的切换采用intent，代码如下所示。

|  |
| --- |
| *// 从标题界面跳转至登陆界面* Intent intent = **new** Intent(Main.**this**,Login.**class**); startActivity(intent); |

登陆成功后，首先显示的是习惯的界面，点击下方底部栏的按钮可以更改页面中的fragment，此时点击的按钮会变为橙色，原先的按钮变为黑色，实现页面切换的效果。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 习惯界面 | 每日任务界面 | 待办事项界面 | 奖励界面 | 菜单界面 |

切换fragment的代码如下。

|  |
| --- |
| *//fragment 切换 index为切换的界面* **private void** InitFragment(**int** index){  *//使用getSupportFragmentManager获取* FragmentManager fragmentManager = getSupportFragmentManager();  *//启动事务* FragmentTransaction transaction = fragmentManager.beginTransaction();  *//将所有的Fragment隐藏* hideAllFragment(transaction);  **switch** (index){  *// 习惯界面* **case** 1:  **if** (**habit\_f** == **null**){  **habit\_f** = **new** habit();  transaction.add(R.id.***frame\_content***,**habit\_f**);  }**else**{  transaction.show(**habit\_f**);  }  *// 修改图标以及标题*  **habit\_iv**.setImageResource(R.drawable.***habit\_icon\_selected***);  setTitle(**"习惯"**);  **break**;  *//待办事项界面* **case** 2:  **if** (**matter\_f**== **null**){  **matter\_f** = **new** matter();  transaction.add(R.id.***frame\_content***,**matter\_f**);  }  **else**{  transaction.show(**matter\_f**);  }  **break**;  *//每日任务界面* **case** 3:  **if** (**task\_f**== **null**){  **task\_f** = **new** task();  transaction.add(R.id.***frame\_content***,**task\_f**);  }  **else**{  transaction.show(**task\_f**);  }  **break**;  *//奖励界面* **case** 4:  **if** (**reward\_f**== **null**){  **reward\_f** = **new** reward();  transaction.add(R.id.***frame\_content***,**reward\_f**);  }  **else**{  transaction.show(**reward\_f**);  }  **break**;  *//菜单界面* **case** 5:  **if** (**menu\_f**== **null**){  **menu\_f** = **new** menu();  transaction.add(R.id.***frame\_content***,**menu\_f**);  }  **else**{  transaction.show(**menu\_f**);  }  **break**;  }  *//提交事务* transaction.commit(); } |

底部按钮的切换fragment监听，由于代码类似，只展示其中一部分。

|  |
| --- |
| *//fragment切换监听*  @Override  **public void** onClick(View v) {  **switch**(v.getId()){  **case** R.id.***habit\_iv***:**habit\_iv**.setImageResource(R.drawable.***habit\_icon\_selected***);  *// 切换fragment* InitFragment(1);  setTitle(**"习惯"**);  **mySearch**.setVisible(**true**);  **add\_btn**.setVisible(**true**);  *// fragment标记*  **fragment\_flag** = 1;  **break**;  *// 剩余的case都类似，这里省略*  **case ...**:  } |

切换时，需要修改顶部的按钮是否显示，底部的标题，以及一个flag变量记录当前fragment是哪一个。

每个页面中存在ListView的item显示，这里我使用了自定义的ItemAdapter继承自BaseAdapter，这里以待办事项ItemAdapter为例，代码如下。

|  |
| --- |
| **public class** MatterItemAdapter **extends** BaseAdapter {  *//测试代码 list* **private** List<String> **data**;  *//测试代码 list* **private** LayoutInflater **layoutInflater**;  **private** Context **context**;  **public** MatterItemAdapter(Context context, List<String> data) {  *//传入的data，就是我们要在listview中显示的信息* **this**.**context** = context;  **this**.**data** = data;  **this**.**layoutInflater** = LayoutInflater.*from*(context);  }  @Override  **public int** getCount() {**return data**.size();}  @Override  **public** Object getItem(**int** position) {**return null**; }  @Override  **public long** getItemId(**int** position) {**return** 0; }  @Override  **public** View getView(**int** position, View convertView, ViewGroup parent) {  *//测试代码 listview inflate* **if**(position == 2){  View v = LayoutInflater.*from*(**context**).inflate(R.layout.***matter\_or\_task\_item\_finished***, **null**);  **return** v;  }  *//测试代码 listview inflate* View v = **layoutInflater**.inflate(R.layout.***matter\_or\_task\_item***, **null**);  **return** v;  } } |

继承BaseAdpater，需要重写其中的五个方法，在getView(int position, View convertView, ViewGroup parent)方法中渲染ListView的Item样式，同时通过控制position的值，可以修改不同的样式。其余的ItemAdapter与这个类似，这里不再赘述。

|  |  |  |  |
| --- | --- | --- | --- |
| 增加习惯 | 增加每日任务 | 增加待办事项 | 增加奖励 |

点击加号后显示增加item的dialog，如上图所示，不同的fragment会显示不同的增加样式，是由于上面提到的fragment\_flag标记记录当前的fragment。具体代码如下。

|  |
| --- |
| *//增加按钮的监听*  **add\_btn**.setOnMenuItemClickListener(**new** MenuItem.OnMenuItemClickListener() {  @Override  **public boolean** onMenuItemClick(MenuItem item) {  **switch** (**fragment\_flag**){  **case** 1:  showAddHabitDialog();  **break**;  **case** 2:  showAddMatterDialog();  **break**;  **case** 3:  showAddTaskDialog();  **break**;  **case** 4:  showAddRewardDialog();  **break**;  }  **return true**;  } }); |

对于这四个dialog，每一个都需要写一个类继承Dialog类，由于四个类的实现都比较类似，这里用增加习惯dialog举例。代码如下。

|  |
| --- |
| **public class** AddHabitDialog **extends** Dialog {  **public** AddHabitDialog(Context context) {**super**(context);}  @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  *//界面初始化* **this**.setContentView(R.layout.***dialog\_add\_habit***);  *//设置背景透明* **this**.getWindow().setBackgroundDrawable(**new** ColorDrawable(Color.***TRANSPARENT***));  } } |

setContentView()方法用来渲染dialog，setBackgroundDrawable()方法用来使背景变为透明的。

特别的在增加待办事项的时候，最下方的选择日期EditText需要实现点击后让用户选择日期然后显示的效果，如下图所示。

|  |
| --- |
|  |

这个功能在java代码中实现。需要注意的是在dialog类中调用findViewById()方法时，一定是使用this. findViewById()，否则没有办法获取到该控件，也没有办法设置监听。代码如下。

|  |
| --- |
| *//日期选择* **date\_select\_et** = **this**.findViewById(R.id.***matter\_date\_et***); **date\_select\_et**.setOnClickListener(**new** View.OnClickListener() {  @Override  **public void** onClick(View v) {getDate();} }); |

在这之中的getDate()方法是我自定义的方法，用于获取用户选择的日期。具体代码如下。

|  |
| --- |
| **private void** getDate(){  *//editview 显示选择日期* Calendar calendar = Calendar.*getInstance*();  *//显示日历的dialog*  DatePickerDialog dialog = **new** DatePickerDialog(**context**, **new** DatePickerDialog.OnDateSetListener() {  *//选择日期的监听*   @Override  **public void** onDateSet(DatePicker view, **int** year, **int** month, **int** dayOfMonth) {  **int** mYear = year;  **int** mMonth = month;  **int** mDay = dayOfMonth;  *// 更新EditText控件日期 小于10加前面0* **date\_select\_et**.setText(**new** StringBuilder()  .append(mYear)  .append(**"-"**)  .append((mMonth + 1) < 10 ? **"0"** + (mMonth + 1) : (mMonth + 1))  .append(**"-"**)  .append((mDay < 10) ? **"0"** + mDay : mDay));  }  },calendar.get(Calendar.***YEAR***), calendar.get(Calendar.***MONTH***), calendar.get(Calendar.***DAY\_OF\_MONTH***));  dialog.show(); } |

使用系统自带的类Calender以及DatePickerDialog实现此效果，通过用户的日期选择，EditText自动显示所选择对应的日期格式。

成就界面与菜单界面的效果如下图所示，仍然是使用自定义的ItemAdapter渲染ListView中的item，左上角返回按钮可以回到之前的程序主界面（不是标题界面）。

|  |  |
| --- | --- |
| 成就界面 | 账单界面 |

以下是这次UI基本文件结构。

|  |  |
| --- | --- |
| java文件 | drawable文件 |
| layout文件 | menu文件 |

**5.个人小结**

在这次的Android开发过程中，遇到了许多困难，也学习了很多新知识。在开始做UI之前，对其了解仅仅局限于每个控件的样式以及部分布局。在制作过程中，为了达到其他组员给我的设计需求，我查阅了许多网上的资料，阅读别人的布局代码，有的借鉴了，有的也提炼出自己想要的结果。一开始的时候，连最简单的两个控件在两边一个控件放在中间都做不好。经过不断的摸索，对每个布局的属性了解更加深刻了。有的时候为了贴合组员给的设计，仅仅是很小的一部分，比如只显示左侧的边框，我也在网上找了各种代码，并进行不断地尝试。同时，改bug也是不可或缺的一部分，在写dialog的监听时，由于之前从来没有用过dialog类，所以对于在dialog类中通过findViewById()方法寻找控件时，尝试了各种各样的方法，直接找报空指针错误，通过view找报运行错误，后来经过不懈的尝试，发现了要用this.findViewById()，这小小的四个字母的bug改了我一个晚上加一个早晨。不过最终还是成功的做完了所有的UI，虽然UI制作看起来简单，但是当你真正去做的时候，你会发现其实这也是一门学问，如何做的好看，美观，十分重要。如果以后我还有机会参加Android开发项目，我希望我可以尝试一下后端的开发，毕竟什么方面都需要勇敢的去尝试一下嘛。

**本人承诺该课程报告中不存在抄袭、剽窃或其他学术不端行为，并愿意承担因此类行为可能导致的一切后果。**

**承诺人（签名）：陈语奇**