

# 移动互联网技术及应用

## 大作业报告

### BUPT 充电系统 设计与实现

选题：应用系统设计实现类（Android）

姓名	班级	学号
马天成	2020211305	2020211376

2023.5

# 目录

0. 引言 .....	3
1. 相关技术.....	3
1.1 开发工具.....	3
1.2 http 协议包——okhttp .....	3
1.3 全局变量处理——SharedPreferencesW .....	3
1.4 http 协议请求进程——NetworkTask .....	4
1.5 动态表单控制——adapter&listview .....	4
2. 系统功能需求.....	4
2.1 加载动画.....	4
2.2 登录验证.....	4
2.3 获取订单信息.....	5
2.4 动态表单展示.....	5
3. 系统设计与实现.....	5
3.1 总体设计.....	6
3.2 系统组成.....	6
3.2.1 入场动画.....	6
3.2.2 登录窗口.....	6
3.2.3 查询界面.....	8
3.3 各层模块设计.....	9
3.3.1 Main 模块 .....	10
3.3.2 Splash 模块 .....	10
3.3.3 Login 模块 .....	10
3.3.4 Order 模块 .....	12
3.4 关键代码解释.....	13
3.4.1 一次 http 请求和相应处理 .....	13
3.4.2 order 处理 listview.....	14
4. 系统可能的扩展.....	15
5. 总结体会.....	16

# 0. 引言

郑重声明，本项目白手起家，完全没有照搬任何现成代码框架，是借由网络搜索资料，源码阅读，以及 chatgpt 修改 bug 完成的。

这次实验，我们主要对于我们 go 语言开发的后端进行一个配套前端安卓程序的开发。（此外其实我们已经有了一个 vue 开发的前端页面，其项目的核心是客户提供充电请求，然后在现实时间内服务，已开发的客户端中客户可以使用请求登录注册一系列的功能）。但是为了解决使用当客户使用客户端申请了充电请求时，不能随时随地查看当前充电的状态，我们提出：开发一个安卓软件，客户能方便地获得当前车辆的信息，以便于及时获得当前车辆状态来进行提车或者是其他的操作。

## 1. 相关技术

### 1.1 开发工具

相关技术工具用的是安卓开发开发工具用的是 android Studio。首先我们在前几次作业中熟悉了按着 studio 的操作以及开发方法。总的来说，难度不大，因为他符合 jetbrain 的使用习惯。

### 1.2 http 协议包——okhttp

首先安卓中的网络开发架构我们尝试了 PDF 中给的几种方法但很难成功。最后在老师和助教老师的帮助下，我借助老师发的 http 文件以及 chatgpt 的帮助，使用 okhttp 框架进行开发解决了我们一直以来不能申请成功的问题，成功跑通了 get 和 post 服务。

在此基础上我们进行了服务的对接和操作，其中 post 请求主要针对登录操作：post 方法帐号和密码以获得他的登录成功信息，并在 response 中获取 token 存到全局变量。get 方法根据用户当前的 token 存到 header 里。并且提供按钮点击的方法，传入参数 parameter 加载到 url 问号后的参数列表里，来查询当前订单或者是历史订单。

### 1.3 全局变量处理——SharedPreferences

在这个过程中我们也需要用到全局变量 token。因为我们在登陆后会返回给用户加密后的长串 token，这样的话我们需要在登陆时，我们使用 sharedPreferences 变量，存储 token 到其中，以便后续操作的全局使用：直接在后续的操作中把 token 从 sharedPreferences 中取出然后插入到我们的申请 request header 的这样就可以成功进行后面的请求。



我们需要 request 一个 post 请求，将用户名密码发送给后端。后端返回 response 报文，可以在其中提取数据。如果登陆成功，那么我们会返回一个 token，如果未登陆成功，我们会返回一个错误状态码，来进行提示。

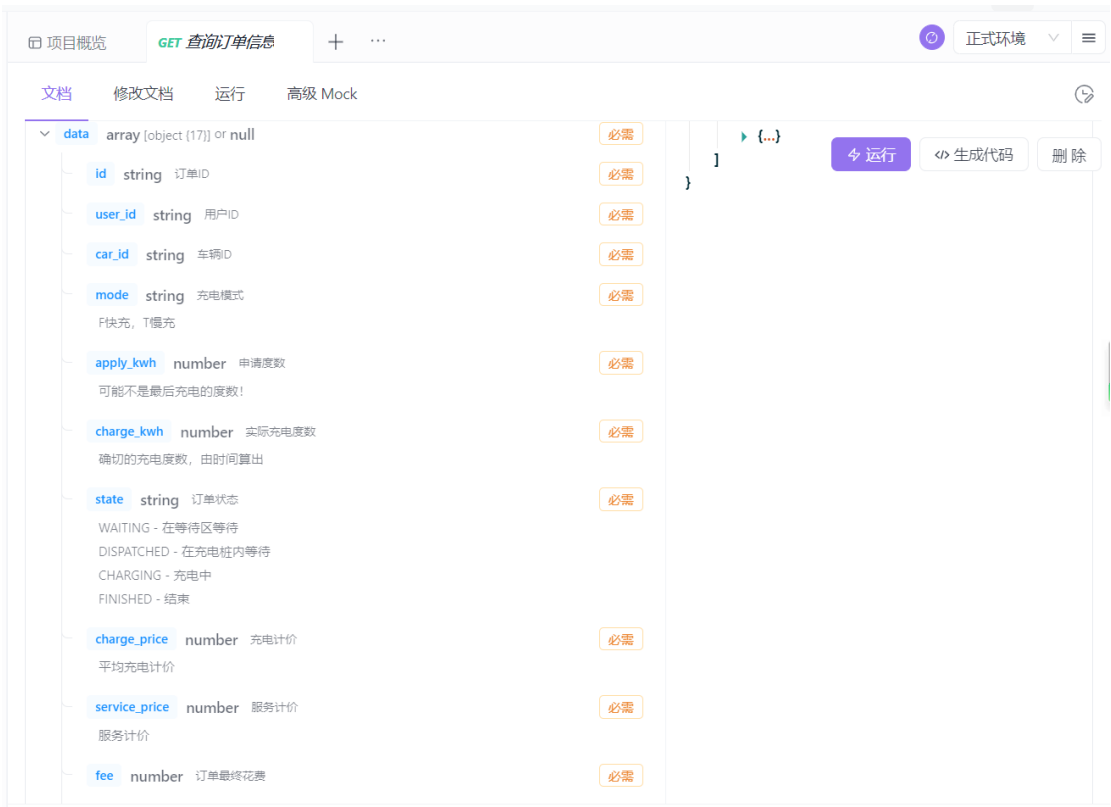
此外，用户点击按钮后，我们也需要展示提示信息，比如登录成功或失败。

## 2.3 获取订单信息

在主界面中，我们需要两个按钮，分别查看当前订单和历史订单。在背后，我们也需要处理这些订单列表数据，获取数据，展示数据。

## 2.4 动态表单展示

我们最核心的问题就是：**如何获取数组信息来动态的，非固定 size 的展示到屏幕上！**因为 order 订单又一些数据，是长这样的：（下图为 APIFOX 接口设计）



所以我们需要设计一个 order 类进行数据的接收和对控件的数据绑定。

## 3. 系统设计与实现

（此句阅后删除： 应包括 总体设计、系统组成、各层模块设计， 关键代码的解释）

## 3.1 总体设计

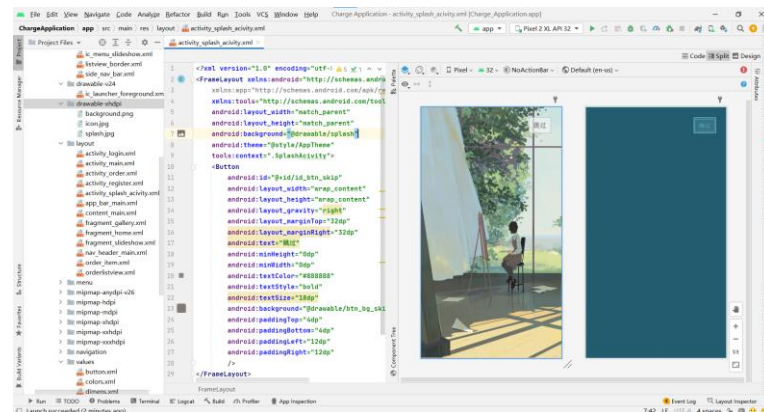
总体上我们分成三个模块：

- 加载动画：加载图片，三秒自动跳过，右上角按钮可主动跳过
- 登录窗口，提交 post 请求并处理
- 订单查询窗口，提交 get 请求并处理到控件上

## 3.2 系统组成

### 3.2.1 入场动画

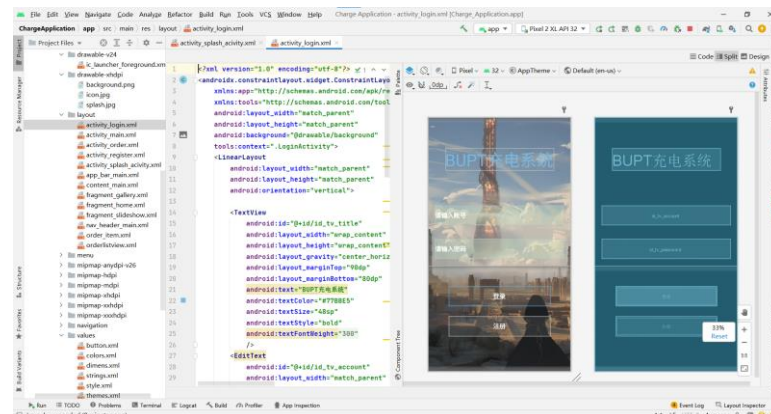
在初始化的时候我们会一个图片进行展示。其中右上角有跳过可以按，及时的跳转到登录窗口。这里就是一个计时器的设置。在三秒后我们会自动跳转，但是按下跳过键后我们会及时进行跳转。

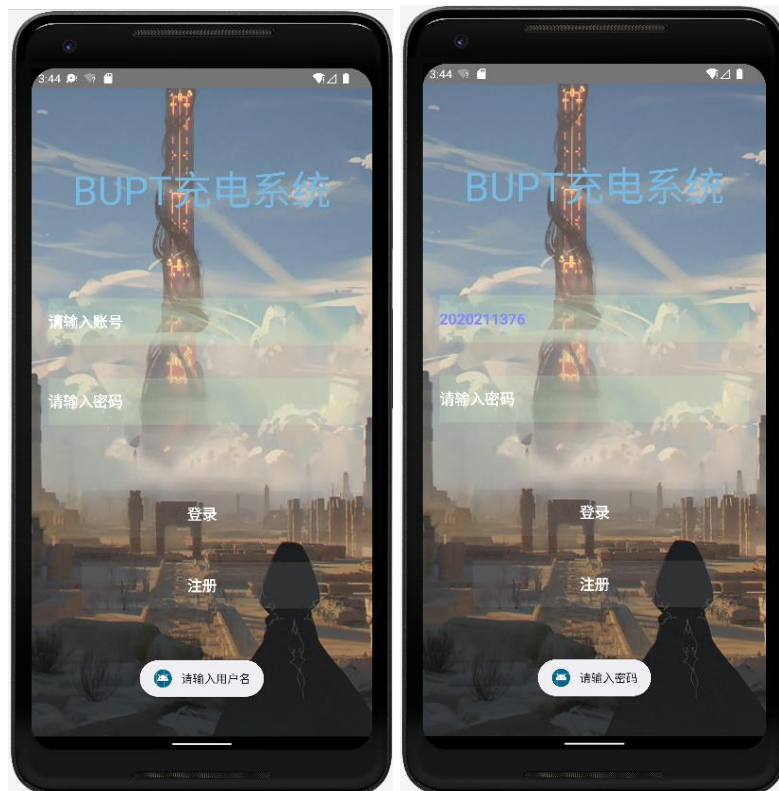


### 3.2.2 登录窗口

在登录窗口中我们有几个控件：一个是账号输入的 edittext，有一个是输密码的 edittext。输入密码的时候，我们的信息展示是可以屏蔽密码信息的，有安全性的考虑。

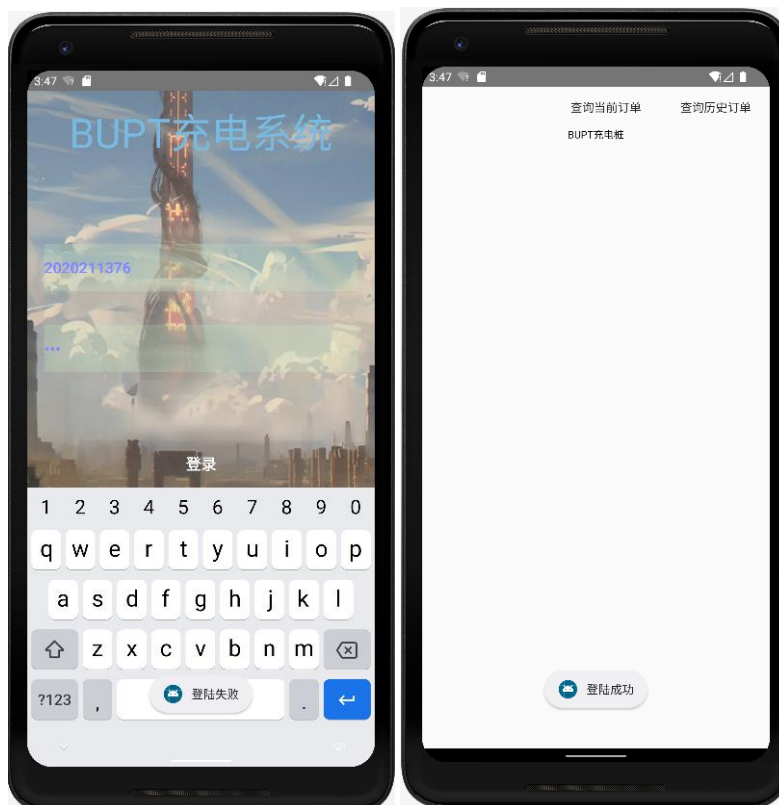
在登录的时候下面有按钮可以登录。登录后，假如输入帐号为空，我们会提示请输入用户名，如果输入密码为空那么会提示请输入密码。





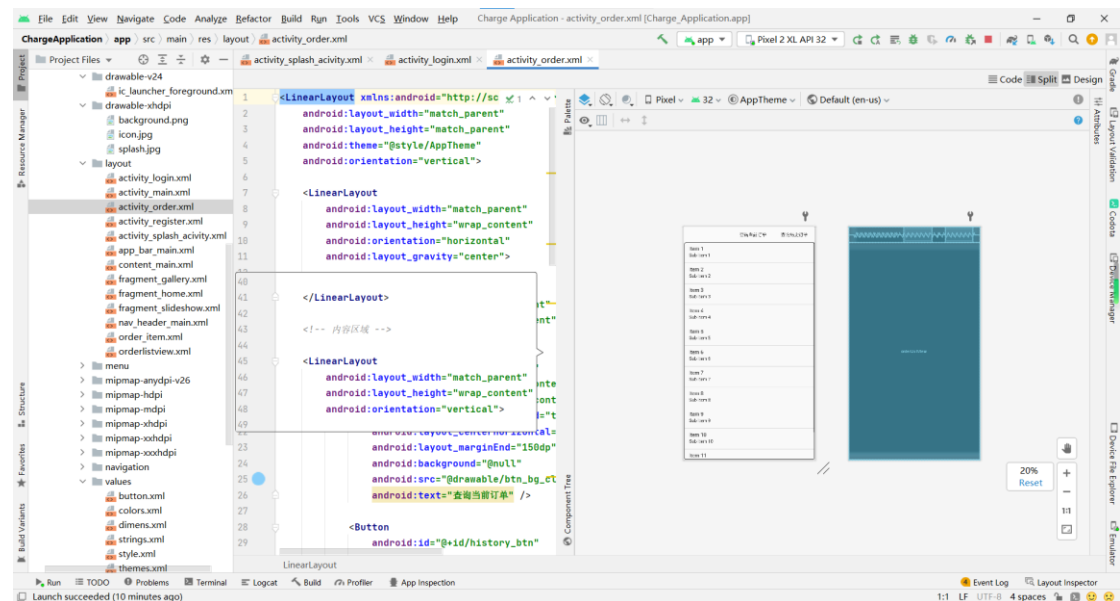
当两者都不为空时我们会提交这次 request 请求，向后端发送。后端会回一个 reponse 来提示你是否拥有登录权限。

- code=0 我们成功判断，并且将 token 存到一个全局变量中，跳转到主页面
- code=-1 我们判断错误，需要重新输入，来进行信息提交



全局变量使用的技术是 sharedpreference。它存入 token 后我们可以很方便的取出使用。

### 3.2.3 查询界面



这其实是一个很简单的 listview 控制器。我们会使用一个 adapter 来进行数据的绑定，将数据动态的展示到其中。

展示效果：

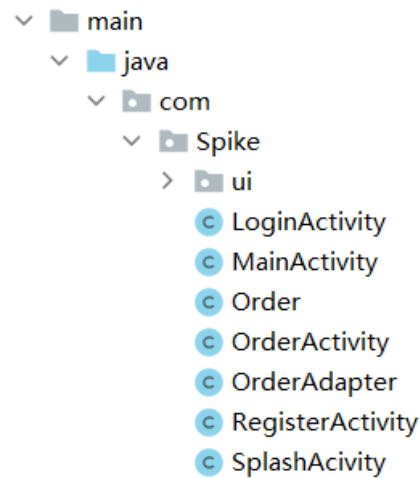


（我刚申请了一个订单）可以看到，他能及时的获取订单信息。历史订单中，也能通过滑动查看所有的订单信息。

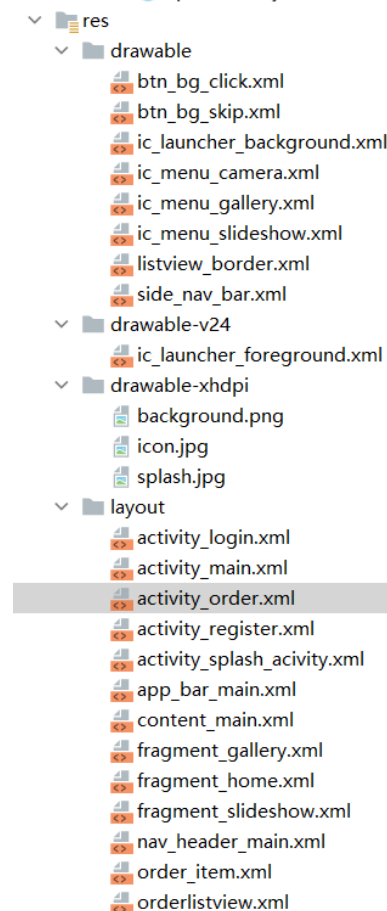


### 3.3 各层模块设计

首先介绍一下我们的 com.Spike 模块:



- Main Activity: 背后运行逻辑
- Splash Activity: 加载动画模块
- Login Activity: 登录窗口模块
- Order Activity: 订单查询模块
- Order: order 类, 负责获取数据
- Order Adapter: 负责绑定 order 数组和控件



然后介绍以西 res。我们写了很多的对应控件，以及一些复用的控件。此外还有图片资源。我们对背景和 icon 都进行了替换。

### 3.3.1 Main 模块

第 1 个就是我们的 mainactivity。在这个 activity 中我们有最主要的初始化的一些操作，这里就不过多赘述

### 3.3.2 Splash 模块

第 2 个是我们的 Splash activity 在这个 activity 中我们有过场动画的展示他会有一个展示。以及右上角有一个按钮跳转，在这个按钮中只要点击事件发生我们就可以跳转到下面的登陆页面。

这里边其实就是一些按钮的绑定 id 和按钮事件的覆写。还是比较简单的。主要就是一个 callbacks 的问题，似乎可能是会出现泄露啥的，需要注意写法。



```
11 public class SplashActivity extends AppCompatActivity {
12
13     private Button eBtnSkip;
14     private Handler eHandler = new Handler();
15     private Runnable eRunnable = new Runnable() {
16         @Override
17         public void run() { toLoginActivity(); }
18     };
19
20
21
22     @Override
23     protected void onCreate(Bundle savedInstanceState) {
24         super.onCreate(savedInstanceState);
25         setContentView(R.layout.activity_splash_activity);
26         initView();
27         initEvent();
28         eHandler.postDelayed(eRunnable, delayMillis: 3000);
29     }
30
31     private void initEvent() {
32         eBtnSkip.setOnClickListener(new View.OnClickListener() {
33             @Override
34             public void onClick(View view) {
35                 toLoginActivity();
36                 eHandler.removeCallbacks(eRunnable);
37             }
38         });
39     }
40
41     private void initView() {
42         eBtnSkip = findViewById(R.id.id_btn_skip);
43     }
44
45     public void toLoginActivity(){
46         Intent intent = new Intent(packageContext: this.LoginActivityv.class);
```

### 3.3.3 Login 模块

第 3 个是我们的 Login activity.在这个 activity 中我们可以填写用户名和密码，然后通过一个 button 的 click 事件据发送一个请求 request。这个 request 是一个 post 请求，它包含了我们的两个键值对：分别是用户名和密码。

在点击事件后我们会新开一个线程去处理整个 request 然后一步获取 response 然后再分析报文去看 code 是否等于零。零的话就代表成功然后我们可以取出键值对 token 存到 sharedpreferences。

```

11 public class SplashActivity extends AppCompatActivity {
12
13     private Button eBtnSkip;
14     private Handler eHandler = new Handler();
15     private Runnable eRunnable = new Runnable() {
16         @Override
17         public void run() { toLoginActivity(); }
18     };
19
20
21
22     @Override
23     protected void onCreate(Bundle savedInstanceState) {
24         super.onCreate(savedInstanceState);
25         setContentView(R.layout.activity_splash_activity);
26         initView();
27         initEvent();
28         eHandler.postDelayed(eRunnable, delayMillis: 3000);
29     }
30
31     private void initEvent() {
32         eBtnSkip.setOnClickListener(new View.OnClickListener() {
33             @Override
34             public void onClick(View view) {
35                 toLoginActivity();
36                 eHandler.removeCallbacks(eRunnable);
37             }
38         });
39
40     private void initView() {
41         eBtnSkip = findViewById(R.id.id_btn_skip);
42     }
43
44     public void toLoginActivity(){
45         Intent intent = new Intent( PackageContext: this, LoginActivity.class);
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64     private void initEvent() {
65         eBtnLogin.setOnClickListener(new View.OnClickListener() {
66             @Override
67             public void onClick(View view) {
68                 // authentic
69                 String account = eEtAccount.getText().toString().trim();
70                 String password = eEtPassword.getText().toString().trim();
71                 if (TextUtils.isEmpty(account)) {
72                     Toast.makeText( context: LoginActivity.this, text: "请输入用户名", Toast.LENGTH_SHORT).show();
73                     return;
74                 } else if (TextUtils.isEmpty(password)) {
75                     Toast.makeText( context: LoginActivity.this, text: "请输入密码", Toast.LENGTH_SHORT).show();
76                     return;
77                 }
78                 // 执行提交
79                 new NetworkTask().execute(account, password);
80             }
81         });
82
83         eBtnRegister.setOnClickListener(new View.OnClickListener() {
84             @Override
85             public void onClick(View view) { toRegisterActivity(); }
86         });
87     }
88
89
90
91     private class NetworkTask extends AsyncTask<String, Void, JSONObject> {
92         @Override
93         protected JSONObject doInBackground(String... params) {
94             String account = params[0];
95             String password = params[1];
96
97             String API_URL = "http://121.43.119.64:8848/client/login";
98             MediaType mediaType = MediaType.parse("application/json");
99             JSONObject requestJson = new JSONObject();
100             try {
101                 requestJson.put( name: "account", account);
102                 requestJson.put( name: "password", password);
103             } catch (JSONException e) {
104                 e.printStackTrace();
105             }
106             requestBody = RequestBody.create(requestJson.toString(), mediaType);
107             request = new Request.Builder()
108                 .url(API_URL)
109                 .post(requestBody)
110                 .build();
111             call = okHttpClient.newCall(request);
112             try {
113                 response = call.execute();
114                 if (response.isSuccessful()) {
115                     responseBody = response.body();
116                     if (responseBody != null) {
117                         return new JSONObject(responseBody.string());
118                     }
119                 }
120             } catch (IOException | JSONException e) {
121                 e.printStackTrace();
122             }
123             finally {
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

其中主要注意 **NetworkTask** 的覆写问题。他暗含了一个线程，是异步处理整个请求过程的。所以我们只需要简简单单的 **excute** 即可完成。然后就是信息的处理和异常捕捉。在这里不得不感叹 **java** 的异常处理机制，真的是很严格。

此外，这里也有提示，当用户输入或者点击的时候，我们会用 **toast** 输出相应的信息到屏幕上进行提示。

### 3.3.4 Order 模块

**Login activity** 后就是我们的 **order activity**。这个模块关联的有：**order activity**，还有我们的 **order** 和 **order adapter** 这两个类。

其中 **order activity** 的是主要的展示页。这个 **activity** 中主要封装了两个不同的 **click** 事件。当我们查询当前订单的时候我们会申请一个 **get** 请求，并将它们的参数 **CURRENT** 赋值到我们的 **URL** 里去进行一个请求。当我们点击查询历史订单时我们会触发一个 **click** 事件将我们的 **HISTORY** 放到我们的 **URL** 参数里，然后进行一个 **get** 请求。这两个请求和 **login** 一样，也是异步处理的。

拿到 **response** 之后，最重要的是：我们如何处理这个数组。请求因为他给我们的的是一个订单数组。就算我取出 **data** 里的数据，他也是一个 **json array** 的类型，我们不能控制个数的去显示 **text** 信息。所以我们只能通过一个可扩展的 **List view** 来显示我们的信息。

关于 **ListView**：这里就用到我们的 **order adapter** 和我们的 **order** 类来解析我们获得的 **data** 中的每一个值。新建完这个 **order** 后我们，会将这个 **order** 存到一个数组中 **orders** 会存储所有的 **order** 的信息，相当于转化的本地。在此之后我们会将 **orders** 转给当前 **activity** 的这个 **adapter** 就是我们的 **orderadapter** 会将他的信息关联到我们的 **textview** 上，形成我可以动态的多个订单，并且可以滑动来展示所有信息。

```
131 // 进行数据分析和控件写入数据
132 try {
133     // 遍历JSON数组并将每个项添加到适配器中
134     List<Order> orders = new ArrayList<>();
135     for (int i = 0; i < data.length(); i++) {
136         JSONObject order = data.getJSONObject(i);
137         // 创建order
138         Order item = new Order(
139             order.getDouble( name: "apply_kwh"),
140             order.getString( name: "car_id"),
141             order.getString( name: "charge_id"),
142             order.getDouble( name: "charge_kwh"),
143             order.getDouble( name: "charge_price"),
144             order.getString( name: "create_time"),
145             order.getString( name: "dispatch_time"),
146             order.getDouble( name: "fee"),
147             order.getString( name: "finish_time"),
148             order.getInt( name: "front_cars"),
149             order.getString( name: "id"),
150             order.getString( name: "mode"),
151             order.getDouble( name: "service_price"),
152             order.getString( name: "start_time"),
153             order.getString( name: "state"),
154             order.getString( name: "user_id")
155         );
156         System.out.println(item);
157         orders.add(item);
158     }
159     adapter = new OrderAdapter( context: OrderActivity.this, orders);
160     listView.setAdapter(adapter);
161 } catch (JSONException e) {
```

## 3.4 关键代码解释

重点代码已标出底纹。

### 3.4.1 一次 http 请求和相应处理

---

```
private void initEvent() {
    eBtnLogin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // authentic
            String account = eEtAccount.getText().toString().trim();
            String password = eEtPassword.getText().toString().trim();
            if (TextUtils.isEmpty(account)) {
                Toast.makeText(LoginActivity.this, "请输入用户名",
                    Toast.LENGTH_SHORT).show();
                return;
            } else if (TextUtils.isEmpty(password)) {
                Toast.makeText(LoginActivity.this, "请输入密码",
                    Toast.LENGTH_SHORT).show();
                return;
            }
            // 执行提交
            new NetworkTask().execute(account, password);
        }
    });
    . . . .
}
```

---

```
private class NetworkTask extends AsyncTask<String, Void, JSONObject>
{
    @Override
    protected JSONObject doInBackground(String... params) {
        String account = params[0];
        String password = params[1];

        String API_URL = "http://121.43.119.64:8848/client/login";
        MediaType mediaType = MediaType.parse("application/json");
        JSONObject requestJson = new JSONObject();
        try {
            requestJson.put("account", account);
            requestJson.put("password", password);
```

```

        } catch (JSONException e) {
            e.printStackTrace();
        }

        requestBody = RequestBody.create(requestJson.toString(),
mediaType);

        request = new Request.Builder()
            .url(API_URL)
            .post(requestBody)
            .build();

        call = okHttpClient.newCall(request);

        try {
            response = call.execute();
            if (response.isSuccessful()) {
                responseBody = response.body();
                if (responseBody != null) {
                    return new JSONObject(responseBody.string());
                }
            }
        } catch (IOException | JSONException e) {
            e.printStackTrace();
        } finally {
            if (responseBody != null) {
                responseBody.close();
            }
        }

        return null;
    }

}

@Override
protected void onPostExecute(JSONObject result) {
    . . . .
}
}

```

---

这些其实都是一个 post 请求的来回过程。先进性 network 变量的 excute，去一部进行申请数据和拿到返回值。可以看到我们拿到返回值之后，进行了处理，我们在成功报文中将 token 取出，放到 sharedpreference 中。

### 3.4.2 order 处理 listview

```

// 进行数据分析和控件写入数据
try {
    // 遍历 JSON 数组并将每个项添加到适配器中

```

```

List<Order> orders = new ArrayList<>();
for (int i = 0; i < data.length(); i++) {
    JSONObject order = data.getJSONObject(i);
    // 创建 order
    Order item = new Order(
        order.getDouble("apply_kwh"),
        order.getString("car_id"),
        order.getString("charge_id"),
        order.getDouble("charge_kwh"),
        order.getDouble("charge_price"),
        order.getString("create_time"),
        order.getString("dispatch_time"),
        order.getDouble("fee"),
        order.getString("finish_time"),
        order.getInt("front_cars"),
        order.getString("id"),
        order.getString("mode"),
        order.getDouble("service_price"),
        order.getString("start_time"),
        order.getString("state"),
        order.getString("user_id")
    );
    orders.add(item);
}
adapter = new OrderAdapter(OrderActivity.this, orders);
listView.setAdapter(adapter);
} catch (JSONException e) {
    e.printStackTrace();
}

```

---

这里主要是进行了一个 order 从 jsonarray 中的提取和适配给 adapter。然后将 adapter 给 listView 就形成了我们的动态表单展示。

## 4. 系统可能的扩展

系统可能够拓展当然是把一些客户端的东西拿过来做了。比如注册提交订单删除订单增加车辆信息删减车辆信息更改车辆信息等。我们还有一个管理员端前端！甚至也可以考虑移植过来做移动客户端。

当然，在 UI 上，我们还是有非常非常非常大的进步空间的。我开发出来的这玩意儿他实在是比我预想中丑太多了（emoji）

我手机的使用状况：



## 5. 总结体会

在这次实验中我所要遇到的第一个问题就是 HTTP 的使用。因为它的涉及到很多网络协议栈，可能不同的安卓版本不同的协议往它都有不同的需求，对于不同的配置也有不同的写法要求。在这个过程中我花了近两天的时间来解决 HTTP 的问题。其中我不得不提我们的后端后端因为接受了跨域请求所以基本上是没有问题的（这也我们测过很多遍）但是前端的话我们使用了很多框架但最终只有 `okhttp` 是在我们的安卓版本上面使用的。`API 32` 的版本在这个版本中我们用 `okhttp` 去请求 `http` 不加密的这种网络请求我们才可以成功联通。

此外就是我们在开发过程中遇到了很多各种各样的控件问题。比如说它的样是不能成功渲染，格式不能成功展示，位置不能随意放置。这些过程中我们主要针对 `xml` 文件进行了很多修改和审查，并且在互联网的帮助下我们成功得到了我们想要的样式。

最后就是我们如何进行一个动态数据的绑定。在这个过程中我们使用了很多方法比如



`scrollview` 比如 `recycleview`。我们最后采用了 `textview` 这种比较简单的方法进行数据的展示因为其他方法或多或少上在编写 `adapter` 或者是数据绑定的时候都有一些问题，或者是很难编写的地方。（`scrollview` 的写法会完全不一样）。

我们在最终的裁决方案中我们也对单个 `ordertext` 进行了一些格式的编写样式的设计，以便我们在展示某一个单个数据的时候风格还是说得过去的。

总之在这次开发中我获得了各种各样的经验，以及体会了安卓的开发风格。它是类似一个前端开发，不过封装了一些比较灵活的方式，比如他的控件 `ID`，这个很巧妙的设计对于整个安卓开发来的有很强的风格。此外我也对安卓技术有了一个更强的了解。

最后我们其实在导出 `apk` 包的时候其实也对签名有了一个比较深入的了解。虽然最终并没有签名成功因为我们并不能申请到一个合规的签名，但是在这个正式发布的过程中我们还是积累了经验，真正去了解了一些以前可能接受不到的知识。