



中南大學
CENTRAL SOUTH UNIVERSITY

计算机应用基础实践 实验报告 (医院看病预约)

院 系	计算机院
专业班级	计算机科学与技术 1707
姓 名	罗家璇
学 号	0902170512
指导教师	杨希

2019 年 1 月 15 日

摘 要

本文主要实现了医院看病预约的应用，报告从下面十点阐述——问题描述、需求分析、设计思想、概要设计、详细设计、程序模块功能、运行结果、调试报告、思考感悟和参考文献。

代码实现主要分为三大模块，前端的Android界面，后台的Jsp与java，和数据库MySQL。用户在注册登录进入应用后，可以通过城市定位或者通过输入关键字自己搜索所在城市，应用会通过所定位的城市给出医院列表，用户选择医院列表，并基于此选择相应科室后，应用会展示出该科室下的医生列表，然后用户可以点击医生列表查看其日程表和详细信息介绍，点击日程表值班项可以提出预约挂号申请，预约成功并缴费后订单会显示在主界面，该应用满足用户一人多次挂号预约，所有的订单都会一列排开在主界面。

关键词：医院挂号系统；移动应用开发；MVC 模式；MySQL

目 录

摘要	I
1 问题描述	1
1.1 实验目的	1
1.2 课程设计内容	1
2 需求分析	2
2.1 用户背景	2
2.2 用户需求分析	2
2.3 功能需求分析	3
3 设计思想	4
3.1 Android 前端	4
3.2 JAVA 与 JSP 后端	4
3.3 MySQL 数据库	5
4 概要设计	6
4.1 前端安卓界面	6
4.2 Jsp 后台	6
4.3 数据库表结构	7
5 详细设计	8
5.1 Android 前端	8
5.2 Jsp 后台	15
5.3 MySql 数据库	17
6 程序模块功能	18
6.1 Index.java:	18
6.2 Agenda.java	23
6.3 Reserve.java:.....	27
7 运行结果	31
7.1 登陆/注册界面	31
7.2 选择城市	32
7.3 选择医院	33
7.4 选择科室	34

7.5	展示医生列表	34
7.6	展示医生详细信息	35
7.7	展示医生日程表	36
7.8	点击 20 日下午出勤预约	37
7.9	点击确认预约	38
8	调试报告及遇到的问题	39
8.1	访问后端失败	39
8.2	未在 web.xml 指定映射导致报错	39
8.3	Android.support 和 Androidx 混用报错	40
8.4	ViewPager 总是吃掉下面的组件	41
8.5	申请高德 API 时申请 SHA1 口令	41
9	收获与感悟	42
	参考文献	43

1 问题描述

1.1 实验目的

通过该课程设计，让学生学会综合运用所学过的《计算机程序设计基础》、《数据结构》、《算法》、《Java 语言与系统设计》、《数据库》、《Web 技术》、《移动应用开发》等课程的基础知识，从而能够熟练掌握开发市面上比较流行的移动应用开发的基本技能，设计和开发出具有一定规模的 Android 客户端应用+JSP（PHP）服务器端编程+MySQL 数据库的典型移动互联网应用。

1.2 课程设计要求

项目 4 旅行信息分享应用

基本内容与要求

- 1) 编写 Android 客户端应用，能够在旅行前、后，以及过程中分享旅行信息。
- 2) 用 JSP 或 PHP 开发旅行信息分享 Web 服务器。
- 3) 用 MySQL 做旅行信息分享 Web 服务器的后台数据库。
- 4) 旅游前的攻略收集和筛选、队伍组建。
- 5) 旅游中的路线云备份和队伍管理。
- 6) 旅游后资源分享和浏览、好友聊天互动等。

2 需求分析

2.1 用户背景

该应用主要为患者提供更方便更快捷的看病预约挂号功能，主要的服务对象是希望挂号看病的患者和后台管理人员。

2.2 用户需求分析

- 用户需要注册、登录自己的账号以进入看病预约应用。
- 用户在登录账号后，可以点击按钮预约挂号，依次选择想要挂号的医院、科室后，可以看到对应的医生列表
- 用户可以查看该科室下的医生简要介绍，可以点击该医生图标进入详细介绍，并且查看该医生的日程安排表
- 选择合适的时间和医生，点击预约挂号按钮，可以弹出并浏览患者挂号信息
- 确认支付后，主页出现预约挂号单
- 能多次挂号且能在主页生成所有预约挂号单

2.3 功能需求分析

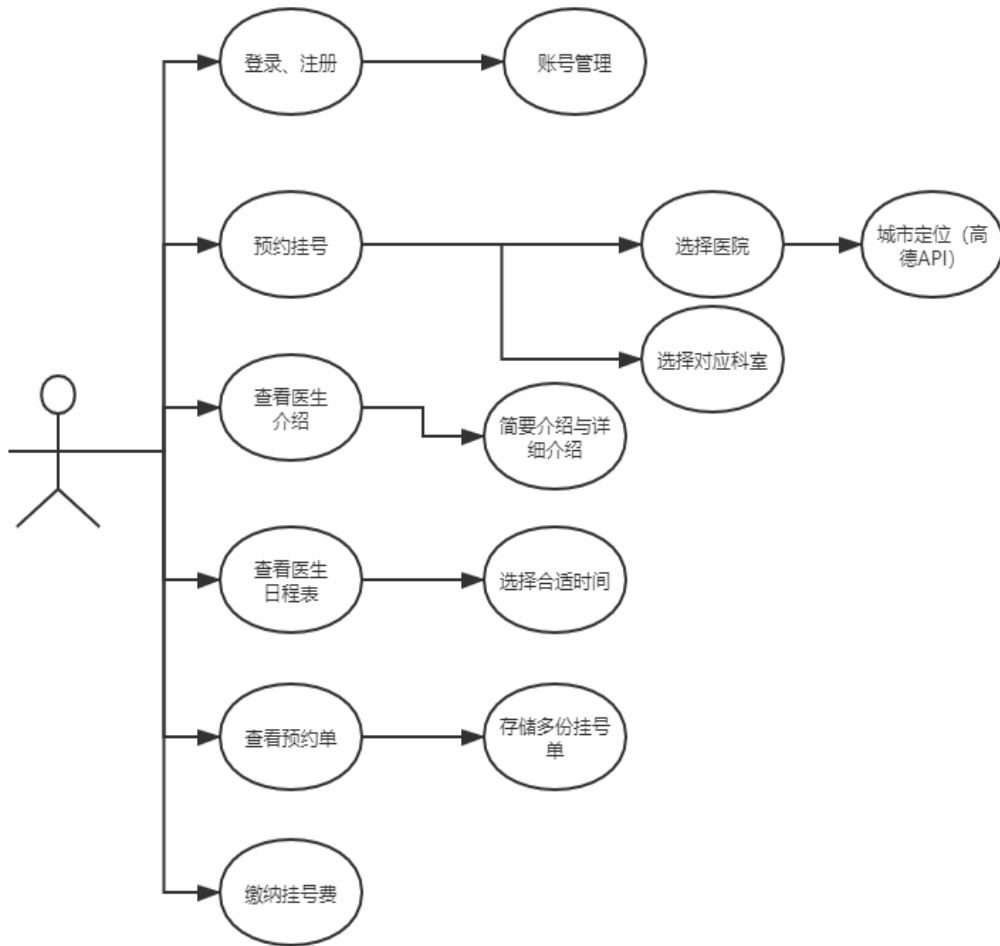


图 1-1 用户用例图

3 设计思想

设计思想主要从三个方面来阐述，即：Android 前端、Java 与 jsp 后端，MySQL 数据库

3.1 Android 前端

Android 采用 API 29 版本，前端界面切换使用 Activity 通信，与后端 jsp 交互主要通过继承 AsyncTask 类来实现异步前后端数据交互。

布局采用 xml 文件布局，登录、注册等简单界面采用静态组件如 TextView 等布置，而需要动态生成的组件比如从数据库中调出该城市所有医院，则需采用 java 文件动态调用方式生成。

前端与后端的交互采用的数据格式：简单的数据用 io 流直接交互，复杂的数据采用 JSON 格式，通过 Google 封装的 gson 实现 String 和 json 的相互转换。

3.2 JAVA 与 JSP 后端

本实验后端使用 Tomcat 服务器建立后台服务器应用，主要分为两大部分，一部分是封装好的对数据库统一操作接口，即 DAO 类，另一部分是连接 Android 前端，与前端进行数据交互，通过继承 HttpServlet，重载 doGet 和 doPost 方法来实现与前端交互，通过 request.getParameter 获得前端传来的数据，通过 response.getWriter()函数写入返回数据。

JSP 在代码形式上约等于在 HTML 上嵌入 java 代码，它放在服务端最终会以 class 文件形式存在，将数据处理如 DAO 等要与数据库交互频繁的部分写入 java 文件中，以类的形式封装好，方便调用，而在 jsp 中调用 java 封装的类，从而让代

码显得更有层次和简洁明了。

在后台管理 pc 端而言，后端仍然使用 jsp 处理数据，只不过前端由移动端变为 PC 网页端。

3.3 MySQL 数据库

数据库采用 mysql 进行开发，后台以 root 身份登入数据库，在 jsp 代码中展现为以 JDBC 驱动调用数据库，mysql 服务器中专门开了一个名为 hospital 的数据库，把所有数据表都存在该数据库中，为了实现所有的功能，我一共建了 7 张表，分别是 admin——管理员表，存放管理员身份信息，agenda——日程表，存放了医生排班信息，doctor——医生表，存放了医生个人信息，hospital——医院表，对于每个城市而言，涵盖了该城市的所有医院，reserve——即预约挂号表，存放了所有支付成功后生成的预约挂号单信息，subject——科室表，对于每个医院来说科室表都不完全一致，users——患者表，记录了患者的个人信息和账号信息。

每张表要确定主键和唯一、非空等约束条件，防止出错。

4 概要设计

4.1 前端安卓界面

前端的 java 文件一共有 19 个，包括 activity 和 model 文件，布局文件共有 16 个 xml 文件，分别对应了：登录注册界面、预约挂号罗列清单、选择所在城市、选择想预约的医院、选择要预约的科室、查看医生个人介绍、查看医生排班日程表、提交预约挂号单、支付等界面。

顺序说来，患者首先免费注册一个账号，然后通过该账号登录看病预约系统，然后可以查看到自己挂了哪些预约单，点击按钮“点击挂号”开始预约，首先选择所在城市，这里调用了高德地图 api 可以定位到当前城市，然后前端会显示数据库中对应于该城市的各个医院信息，用户可以点击感兴趣的医院进入详情，通过选择合适的科室查看对应的医生团体，选择合适的医生后可以提交预约单，确认无误并线上缴费后则挂号成功，此时回到主界面可以看到预约单已经生效。

4.2 Jsp 后台

分为三个 package 和一些 jsp 文件，package 中有数据类，如 Doctor、Hospital 等实现数据表中的每行数据实例，有 DAO 类，通过该类将数据表与数据类联系在一起，之后前端想要数据库数据，直接从 DAO 获取即可。

Jsp 继承 HttpServlet 来实现对网络处理，由于 android 严格禁止在主线程中运行网络交互，所以必须要新建一个子线程来进行网络操作。

另外，为了保证访问失败时，程序不至于崩溃，所以可以在子线程中封装好两个接口，一个用于访问失败的接口，另一个用于访问成功接口。

在访问成功的接口中读取传送的有效信息，在发送数据时通过键值对数据格式封装。

4.3 数据库表结构

一共 7 张表。

管理员表：管理员 id，管理员密码 apasswd，管理员姓名，aname

日程表：存放了医生的排班安排，即医生姓名 dname，医生编号 did，排班的日期 aday，出勤的具体时间段 astate，是否出勤 aduty。

医生表：存放了医生的个人信息，有 dintro 基本介绍，dname 医生名字，所在医院名字和所在科室名。

医院表：该应用不只布置在一个地方，而是很多个城市，所以一张医院表来存放每个城市的医院信息，包括 hname 医院名和 cname 城市名。

科室表：展现了医生所在的科室，有 sname 即科室名称，hname 即医院名称。

预约订单表：最复杂的表，涵盖了医院名称、科室、医生、病患联系方式等等数据，也是最后挂号打印出来的清单。

用户表：也就是存放用户注册账号登录该应用的基本信息，比如 uname 用户名，upasswd 用户密码。

5 详细设计

5.1 Android 前端

5.1.1 Android 项目结构

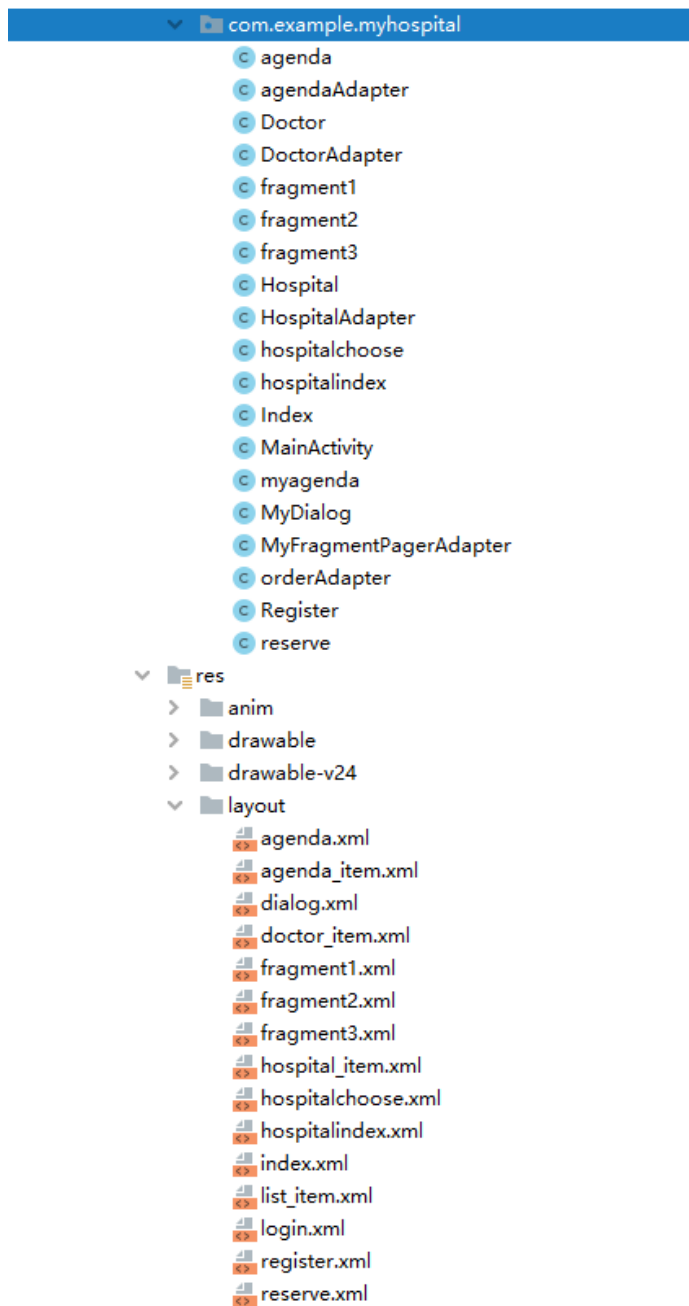


图 5-1 android 项目结构

由 Manifest 文件可知，我共有六个 activity，基本对应于六个功能，如下所示：

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="hospital"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:usesCleartextTraffic="true"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>

            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
    <activity android:name=".Register"></activity>
    <activity android:name=".Index"></activity>
    <activity android:name=".agenda"></activity>
    <activity android:name=".hospitalindex"></activity>
    <activity android:name=".hospitalchoose" android:theme="@style/DefaultCityPickerTheme"
        android:screenOrientation="portrait"
        android:windowSoftInputMode="stateHidden|adjustPan">
    </activity>
    <activity android:name=".reserve"></activity>
</application>
```

图 5-2 manifest 结构

5.1.2 登录界面

对应于 manifest 的 MainActivity 文件，在登陆界面，放置两个输入框和两个按钮，两个输入框即输入账号和密码登录，同时有个较小的蓝色按钮即注册通道，当用户没有已注册的账号时，可以点击免费注册，同时在 activity 通过 Intent 通信发送给 Register activity 从而实现页面的切换。



The image shows a login interface with a light gray background. At the top, there is a text input field with the placeholder text "输入您的名字" (Enter your name) in gray. Below it is another text input field with the placeholder text "输入您的密码" (Enter your password) in gray. In the center, there is a gray button with the text "登陆" (Login). At the bottom right, there is a blue link with the text "免费注册" (Free registration).

图 5-3 登录界面

5.1.3 注册界面

注册界面和登陆界面有些类似，不同的是，要输入更完整的用户信息，比如用户姓名和用户电话号码，并且为了尽可能避免用户注册时输入错误的密码，所以采取两次密码输入比较一致性来判断用户是否手误打错密码。

与登录不同的地方还体现在数据库方面，登录仅仅需要在数据库中查询数据表中是否存在该行数据，如果有则登录，如果没有则登陆失败，而注册则是需要在数据表中添加数据，因此需要更加谨慎，有可能导致数据库遭受损失，要做好错误异常处理。

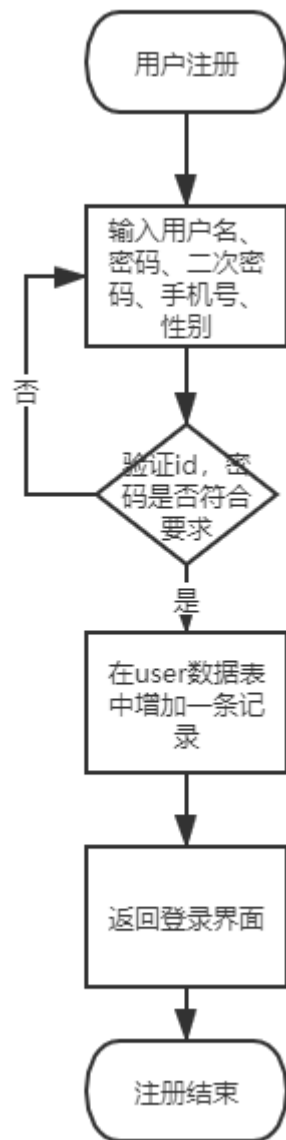


图 5-4 用户注册流程图

5.1.4 逻辑主界面

用户登录后，点击预约挂号，则应该利用 `intent` 通信跳转到城市定位界面，通过 GPS 定位或者用户自己选择城市获得所在地，然后通过 `RecyclerView` 弹出医院列表，用户点击想要的挂号的医院，则会触发在 `HospitalAdapter` 的监听器，即引起页面携带者被点击的医院名通过 `intent` 启动 `DepartActivity`，即获得该医院对应的

科室列表。

接着进一步点击科室列表，运用同样的组件——RecyclerView 通过注册 DoctorAdapter 获得被点击的 depart 名字对应下的所有医生，这些医生会陈列为单向列表，展示名字和基本介绍，考虑到排版的美观，通过 substr 截断过长的个人介绍，截断后的字符串使用省略号代替，如果用户对该医生感兴趣，那么可以点击该医生的 item，则会进入医生的详细介绍，此时将会有足够的空间展示医生的详细个人介绍和日程表，日程表可以使用 Calendar 组件来实现周日程，每天有三个时间段分别是上午、下午和晚上，每个时间段都有出勤和休息两种状态，用户可以自行挑选合适的医生出勤时间段，通过点击 item 则会通过 intent 携带着医生 id 等信息进入预约挂号提交单。

上面会展示患者的个人信息和通过不断点击最终确定的医院、科室、医生、时间和费用，最后用户可以点击确认预约，并缴费后完成预约挂号，此时应用的主页面会由空空无一物出现该用户的订单，用户仍然可以继续挂号和别的操作，最终实现一人多次挂号。

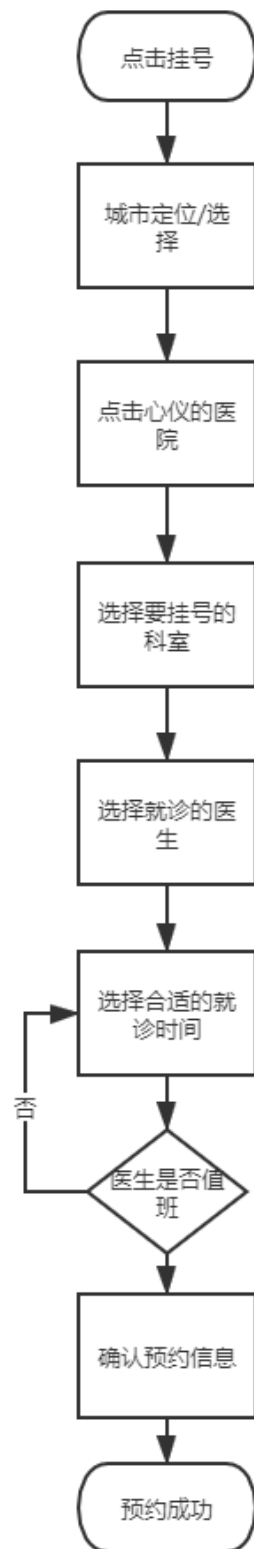


图 5-5 核心业务流程图

5.2 Jsp 后台

后端服务采用类 MVC 结构, src 分为三个 package, 其中 bean 代表了 model 文件, 将数据库中的数据通过封装类的形式表示, 通过 setter 和 getter 方法保护数据的安全性。

DAO 类都是进行数据操作的类, 是对于数据库中的数据做增删改查等操作的代码, 其实主要工作就是在后台对数据库数据进行处理, 主要有根据字段查找所需数据然后返回给前端等工作。

最后一个包——hospital 承担了一部分 jsp 的工作, 在 web.xml 中指明映射关系就可以直接在前端加上映射名访问 java 文件, 然后通过重载的 doGet 方法进行数据传输。

另外, 后台代码大部分是 jsp, 每个 jsp 都是先用 request.getParameter 获得前端传输的数据, 然后通过 response 并通过 PrintWriter 写回给前端。

前后端之间的数据格式大部分为 json, 通过 Gson 包的封装函数, 可以进行 string 和 json 的互相转换, 总是把 json 转换成 string 以方便进行数据传输, 传输接收到后, 再把 string 通过 gson 转成 json, 方便读取和校正。

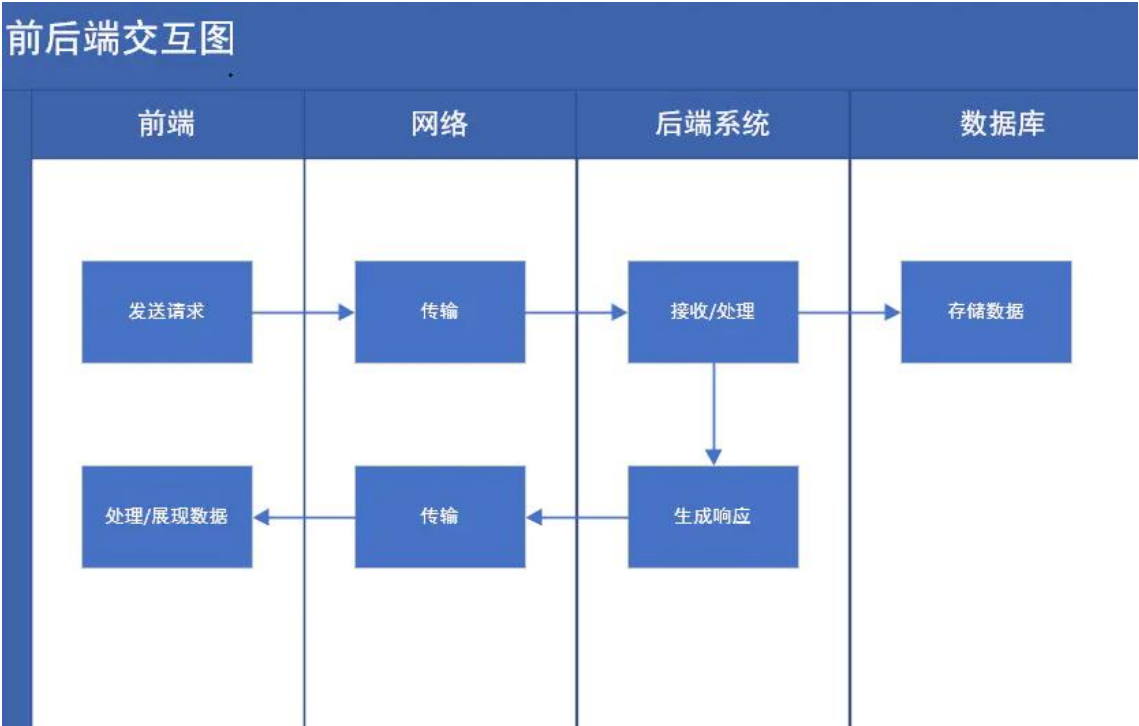


图 5-6 后端系统与网络和数据库交互图

5.3 MySQL 数据库

如图所示，打开 mysql 服务器界面查看所有表结构：



did	dname	dintro	hname	sname
0001	姜西月	内科副主任医师。大学本科，医学学士。糖尿病专科门诊主治医师，从事内科临床工作20余年。曾在四川省人民...	湘雅医院	内科
0002	罗玲	内科主任医师。大学本科，医学学士。从事内科临床工作30年，曾在四川大学华西医院等三甲医院多次进修学习...	湘雅医院	内科
0003	张小明	内科主任医师。大学本科，医学学士。从事内科临床工作30年，曾在四川大学华西医院等三甲医院多次进修学习...	湘雅医院	内科

图 5-7 数据表图

数据库可以先建表再写前后端，也可以先写前后端再在测试时写入数据表，感觉有时候并不是很清楚到底需要几个表，需要什么样的字段，所以建议先写逻辑代码，根据逻辑代码来确定数据段和数据表。

6 程序模块功能

由于代码量比较大，逻辑代码加上布局文件，全部贴上非常笨拙，所以仅在这里展示部分模块核心代码：

6.1 Index.java:

//作用：从数据库拿数据，动态生成了 departs 科室，以及点击任意一个科室会动态刷新出该科室下的所有医生

```
1. package com.example.myhospital;
2.
3. import android.content.Intent;
4. import android.os.AsyncTask;
5. import android.os.Bundle;
6. import android.text.Html;
7. import android.view.View;
8. import android.widget.LinearLayout;
9. import android.widget.TextView;
10. import android.widget.Toast;
11.
12. import androidx.appcompat.app.AppCompatActivity;
13. import androidx.recyclerview.widget.LinearLayoutManager;
14. import androidx.recyclerview.widget.RecyclerView;
15.
16. import java.io.BufferedReader;
17. import java.io.IOException;
18. import java.io.InputStream;
19. import java.io.InputStreamReader;
20. import java.lang.reflect.Type;
21. import java.net.HttpURLConnection;
22. import java.net.MalformedURLException;
23. import java.net.ProtocolException;
24. import java.net.URL;
25. import java.util.ArrayList;
26. import java.util.List;
27.
28. import com.google.gson.Gson;
29. import com.google.gson.GsonBuilder;
30. import com.google.gson.reflect.TypeToken;
```

```
31.
32. public class Index extends AppCompatActivity {
33.     private LinearLayout departs;
34.     //private ArrayList<String> departsList = new ArrayList<String>();
35.     private LinearLayout doctorsline;
36.     private TextView depart1;
37.     private TextView depart2;
38.     private TextView depart3;
39.     private String str_departs[];
40.     private Gson gson;
41.     private GsonBuilder builder;
42.     private Doctor doctor;
43.     private List<Doctor> doctors;
44.     private DoctorAdapter adapter;
45.     private RecyclerView recyclerView;
46.
47.     @Override
48.     protected void onCreate(Bundle savedInstanceState) {
49.         super.onCreate(savedInstanceState);
50.         getSupportActionBar().hide();
51.         setContentView(R.layout.index);
52.         //获得 intent 的值
53.         System.out.println(getIntent().getStringExtra("Hname"));
54.         String hname=getIntent().getStringExtra("Hname");;
55.         String hid=getIntent().getStringExtra("Hid");
56.         getdeparts(hname,hid);
57.         departs = (LinearLayout) findViewById(R.id.departs);
58.         //
59.
60.         //初始化
61.         doctors = new ArrayList<Doctor>(); //初始化医生列表
62.         recyclerView = (RecyclerView) findViewById(R.id.recycler_view)
63.         ;
64.         LinearLayoutManager layoutManager = new LinearLayoutManager(this);
65.         recyclerView.setLayoutManager(layoutManager);
66.         builder = new GsonBuilder();
67.         gson = builder.create(); //创建一个 gson
68.     }
69.     public void getinfo(String dname,String hname) {
70.         String path = MainActivity.basepath + "getdoctors.jsp";
71.         new Index.getTask().execute(dname,hname,path);
```

```
72.     }
73.
74.     class getTask extends AsyncTask {
75.         @Override
76.         protected Object doInBackground(Object[] params) {
77.             //依次获取用户名，密码和路径
78.             String sname = params[0].toString();
79.             String hname=params[1].toString();
80.             String path = params[2].toString();
81.             try {
82.                 //获取网络上 get 方式提交的整个路径
83.                 URL url = new URL(path + "?sname=" + sname+"&hname="+h
name);
84.                 //打开网络连接
85.                 HttpURLConnection conn = (HttpURLConnection) url.openC
onnection();
86.                 //设置提交方式
87.                 conn.setRequestMethod("GET");
88.                 //设置网络超时时间
89.                 conn.setConnectTimeout(5000);
90.                 if (conn.getResponseCode() == 200) {
91.                     //用 io 流与 web 后台进行数据交互
92.                     InputStream is = conn.getInputStream();
93.                     //字节流转字符流
94.                     BufferedReader br = new BufferedReader(new InputSt
reamReader(is));
95.                     //读出每一行的数据
96.                     String str = br.readLine();
97.                     System.out.println(str);
98.                     //返回读出的每一行的数据
99.                     return str;
100.                }
101.            } catch (MalformedURLException e) {
102.                e.printStackTrace();
103.            } catch (ProtocolException e) {
104.                e.printStackTrace();
105.            } catch (IOException e) {
106.                e.printStackTrace();
107.            }
108.            return null;
109.        }
110.
111.        @Override
112.        protected void onPostExecute(Object o) {
```



```

113.         super.onPostExecute(o);
114.         //获取 android 与 web 数据交互获得的值
115.         String s = (String) o;
116.         //解析 json, 生成医生列表
117.         System.out.println("输出医生列表:"+s);
118.         Type type1 = new TypeToken<List<Doctor>>() {
119.         }.getType();
120.         doctors = gson.fromJson(s, type1);//把 json 数据的集合转换成
            实体类的集合
121.         adapter = new DoctorAdapter(Index.this, doctors);//对
            doctors 集合创建一个适配器
122.         recyclerView.setAdapter(adapter);
123.         //adapter.notifyDataSetChanged();
124.         //显示医生列表至 view
125.         for(Doctor d: doctors){
126.             System.out.println(d.getDname());
127.             //doctors.add(d);
128.             //doctors 集合中每有一个元素, 就 new 一个 textView
129.             //TextView tv = new TextView(Index.this);
130.             //把信息设置为文本框内容
131.             //String totalinfo="姓名:"+d.getName();
132.         }
133.
134.         //toast 弹出
135.         //Toast.makeText(Index.this,s,Toast.LENGTH_SHORT).show();
136.         //System.out.println(s);
137.     }
138. }
139.
140. class getDepart extends AsyncTask {
141.     @Override
142.     protected Object doInBackground(Object[] params) {
143.         String path = params[0].toString();
144.         String hname=params[1].toString();//获得医院名称
145.         String hid=params[2].toString();
146.         try {
147.             //获取网络上 get 方式提交的整个路径
148.             URL url = new URL(path+"?hname="+hname+"&hid="+hid);
149.             //打开网络连接
150.             HttpURLConnection conn = (HttpURLConnection) url.openConnection();
151.             //设置提交方式
152.             conn.setRequestMethod("GET");

```

```
153.         //设置网络超时时间
154.         conn.setConnectTimeout(5000);
155.         if (conn.getResponseCode() == 200) {
156.             //用 io 流与 web 后台进行数据交互
157.             InputStream is = conn.getInputStream();
158.             //字节流转字符流
159.             BufferedReader br = new BufferedReader(new InputStrea
mReader(is));
160.             //读出每一行的数据
161.             String str = br.readLine();
162.             System.out.println(str);
163.             //返回读出的每一行的数据
164.             return str;
165.         }
166.     } catch (MalformedURLException e) {
167.         e.printStackTrace();
168.     } catch (ProtocolException e) {
169.         e.printStackTrace();
170.     } catch (IOException e) {
171.         e.printStackTrace();
172.     }
173.     return null;
174. }
175.
176. @Override
177. protected void onPostExecute(Object o) {
178.     super.onPostExecute(o);
179.     //获取 android 与 web 数据交互获得的值
180.     String s = (String) o;
181.     System.out.println("输出科室列表:"+s);
182.     //解析交互值获得 departs
183.     str_departs = s.split(",");
184.     final String temphname=str_departs[0];//医院名
185.     for (int i = 1; i <str_departs.length;i++){
186.         final TextView t= new TextView(Index.this);
187.         String str="<u>"+str_departs[i]+"</u>";
188.         t.setText(Html.fromHtml(str));
189.         t.setTextSize(18);
190.         t.setClickable(true);
191.         t.setPadding(5,0,5,0);
192.         t.setOnClickListener(new View.OnClickListener(){
193.             @Override
194.             public void onClick(View view){
195.                 new Thread(new Runnable() {
```

```

196.         @Override
197.         public void run() {
198.             getinfo(t.getText().toString(),temphname)
199.             ;//传入当前科室和所在医院
200.         }
201.     }).start();
202. }
203.     departs.addView(t);
204. }
205. }
206. }
207.     public void getdeparts(String hname,String hid){
208.         String path=MainActivity.basepath+"getdeparts.jsp";
209.         new getDepart().execute(path,hname,hid);
210.     }
211. }

```

由上面代码可知,首先在 android 前端的 onCreate 主线程创建一个网络子线程,然后通过该子线程传入查询数据库所需的字段关键词,然后在后端通过 doGet 接收到传入参数,进数据表查询后把结果通过 response 返回给前端,然后前端再通过 RecyclerView 组件通过销毁重建 list 而动态刷新页面,使其不需要经过 intent 等通信跳转 activity,同时,再 adapter 适配器中可以设置监听器,可以在被点击的 item 处进行一系列逻辑处理。

6.2 Agenda. java

Agenda, 顾名思义, 这个类是拿来处理日程表的, 我在数据表中建的日程表中,

dname	did	aday	astate	aduty
罗玲	0002	2020-1-20	上午	休息
罗玲	0002	2020-1-20	下午	出勤
罗玲	0002	2020-1-20	晚上	出勤

有着如上图的五个字段, 分别是医生名字, 医生编号, 日期, 时间段(上午、下午、晚上), 是否值班五大属性, 在 agenda 中, 我先是通过 xml 文件使用 calendar 组件完成界面美观友好的日程表样式, 然后通过权重的设置, 固定下方的出勤情况表, 然后通过点击在 calendar 上的日期, 与数据表中的 aday 相比较, 如果 aday 相同并且 dname 相同说明是同一医生的当天日程表, 则通过 RecyclerView 显示出勤情况,

然后让用户点击出勤 item，弹出预约界面。

```
1. package com.example.myhospital;
2.
3. import android.content.Intent;
4. import android.graphics.Color;
5. import android.os.AsyncTask;
6. import android.os.Bundle;
7. import android.text.Html;
8. import android.view.LayoutInflater;
9. import android.view.View;
10. import android.widget.*;
11.
12. import androidx.annotation.NonNull;
13. import androidx.appcompat.app.AlertDialog;
14. import androidx.appcompat.app.AppCompatActivity;
15. import androidx.recyclerview.widget.LinearLayoutManager;
16. import androidx.recyclerview.widget.RecyclerView;
17. import com.google.gson.Gson;
18. import com.google.gson.GsonBuilder;
19. import com.google.gson.reflect.TypeToken;
20.
21. import java.io.BufferedReader;
22. import java.io.IOException;
23. import java.io.InputStream;
24. import java.io.InputStreamReader;
25. import java.lang.reflect.Type;
26. import java.net.HttpURLConnection;
27. import java.net.MalformedURLException;
28. import java.net.ProtocolException;
29. import java.net.URL;
30. import java.sql.Date;
31. import java.util.ArrayList;
32. import java.util.List;
33.
34. public class agenda extends AppCompatActivity {
35.     private String did;//获取医生 id
36.     private String dname;//医生名字
37.     private String aday;
38.     private String astate;//表示时间段，上午还是下午还是晚上
39.     private String aduty;//表示出勤与否，默认休息
40.     private String dintro;//详细介绍
41.     private CalendarView calendarView;
42.     private RecyclerView recyclerView;
```

```

43.     private TextView completeinfo;
44.     private TextView docname;
45.     private ImageView docimage;
46.     private Gson gson;
47.     private GsonBuilder builder;
48.     private agenda agenda_item;
49.     private List<myagenda> agendas;
50.     private agendaAdapter adapter;
51.     protected static String hname;
52.     protected static String sname;
53.     private MyDialog dialog;
54.     static final int COLOR1 = Color.parseColor("#ffffcc99");
55.
56.     @Override
57.     protected void onCreate(Bundle savedInstanceState) {
58.         super.onCreate(savedInstanceState);
59.         getSupportActionBar().hide();
60.         setContentView(R.layout.agenda);
61.         //初始化
62.         completeinfo=(TextView)findViewById(R.id.completeinfo);
63.         docname=(TextView)findViewById(R.id.docname);
64.         docimage=(ImageView)findViewById(R.id.doctor_image);
65.         agendas = new ArrayList<myagenda>();//初始化医生列表
66.         recyclerView = (RecyclerView) findViewById(R.id.duty_view);
67.         LinearLayoutManager layoutManager = new LinearLayoutManager(this);
68.         recyclerView.setLayoutManager(layoutManager);
69.         builder = new GsonBuilder();
70.         gson = builder.create();//创建一个 gson
71.
72.         //获取 intent 通信的值
73.         dname=getIntent().getStringExtra("dname");
74.         did=getIntent().getStringExtra("did");
75.         dintro=getIntent().getStringExtra("dintro");
76.         sname=getIntent().getStringExtra("sname");
77.         hname=getIntent().getStringExtra("hname");
78.         //设置 textView
79.         completeinfo.setText(dintro);
80.         String name="<b>医生: </b>"+dname;
81.         docname.setText(Html.fromHtml(name));
82.         //设置图片
83.         docimage.setImageResource(R.mipmap.doctor2);
84.         calendarView=(CalendarView)findViewById(R.id.calendarview);//获得日历控
    件

```

```

85.         calendarView.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {
86.             @Override
87.             public void onSelectedDayChange(@NonNull CalendarView view, int year, int month, int dayOfMonth) {
88.                 month+=1; //month 要加 1, 因为从 0 开始计算
89.                 String theday=year+"-"+month+"-"+dayOfMonth;
90.                 getinfo(theday,dname);
91.             }
92.         });
93.
94.         //获得 agenda 表
95.     }
96.     public void getinfo(String theday,String dname){
97.         String path=MainActivity.basepath+"agenda.jsp";
98.         new getTask().execute(path,theday,dname);
99.     }
100.    class getTask extends AsyncTask {
101.        @Override
102.        protected Object doInBackground(Object[] objects) {
103.            String path = objects[0].toString();
104.            String theday = objects[1].toString();
105.            String dname=objects[2].toString(); //获得医生名字
106.            try {
107.                URL url = new URL(path + "?aday="+theday+"&dname="+dname); //
                获取路径
108.                HttpURLConnection conn = (HttpURLConnection) url.openConnection(); //打开连接
109.                //设置提交方式
110.                conn.setRequestMethod("GET");
111.                //设置超时
112.                conn.setConnectTimeout(5000);
113.                if (conn.getResponseCode() == 200) {
114.                    //用 io 流与 web 后台进行数据交互
115.                    InputStream is = conn.getInputStream();
116.                    //字节流转字符流
117.                    BufferedReader br = new BufferedReader(new InputStreamReader(is));
118.                    //读出每一行的数据
119.                    String str = br.readLine();
120.                    System.out.println(str);
121.                    //返回读出的每一行的数据
122.                    return str;
123.                }

```

```
124.         } catch (
125.             MalformedURLException e) {
126.             e.printStackTrace();
127.         } catch (
128.             ProtocolException e) {
129.             e.printStackTrace();
130.         } catch (
131.             IOException e) {
132.             e.printStackTrace();
133.         }
134.         return null;
135.     }
136.     //回调函数
137.     @Override
138.     protected void onPostExecute(Object o){
139.         super.onPostExecute(o);
140.         //获取 android 与 web 数据交互获得的值
141.         String s=(String) o;
142.         //解析 json, 生成 agenda 列表
143.         System.out.println("输出日程列表:"+s);
144.         Type type1 = new TypeToken<List<myagenda>>() {
145.             }.getType();
146.         agendas = gson.fromJson(s, type1);//把 json 数据的集合转换成实体类的
集合
147.         adapter = new agendaAdapter(agenda.this, agendas);//对 doctors 集合
创建一个适配器
148.         recyclerView.setAdapter(adapter);
149.     }
150. }
151. }
```

6.3 Reserve.java:

这个是最用户得到的预约挂号信息单，也就是如果预约后去医院打印出来的票据。

```
1. package com.example.myhospital;
2.
3. import android.content.Intent;
4. import android.os.AsyncTask;
5. import android.os.Bundle;
6. import android.text.Html;
```

```

7. import android.view.View;
8. import android.widget.Button;
9. import android.widget.EditText;
10. import android.widget.TextView;
11. import androidx.appcompat.app.AppCompatActivity;
12.
13. import java.io.BufferedReader;
14. import java.io.IOException;
15. import java.io.InputStream;
16. import java.io.InputStreamReader;
17. import java.net.HttpURLConnection;
18. import java.net.MalformedURLException;
19. import java.net.ProtocolException;
20. import java.net.URL;
21.
22. public class reserve extends AppCompatActivity {
23.     private Button reserve;
24.     private TextView reslist;
25.     private String totallist;
26.     private String[] infos;
27.     @Override
28.     protected void onCreate(Bundle savedInstanceState) {
29.         super.onCreate(savedInstanceState);
30.         getSupportActionBar().hide();
31.         setContentView(R.layout.reserve);
32.         //获取控件
33.         reserve = (Button) findViewById(R.id.reserve);
34.         reslist = (TextView) findViewById(R.id.reslist);
35.         //点击预约
36.         reserve.setOnClickListener(new View.OnClickListener() {
37.             @Override
38.             public void onClick(View view) {
39.                 //跳转到最开始的 index 界面并且联系后台传数据到服务器
40.                 sendinfo();
41.             }
42.         });
43.         //打印清单
44.
45.         //先接受前一个 activity 传来的参数
46.         infos = getIntent().getStringArrayExtra("reserve");
47.         String[] mytime = infos[3].split("-");
48.
49.         totallist = "<h2>姓    名:    " + infos[6] + "</h2>" +
50.             "<h2>电话号码:    " + infos[7] + "</h2>" +

```



```

51.         "<h2>医    院:  " + infos[1] + "</h2>" +
52.         "<h2>科    室:  " + infos[0] + "</h2>" +
53.         "<h2>医    生:  " + infos[2] + "</h2>" +
54.         "<h2>日    期:  " + mytime[0] + "年" + mytime[1] + "月
    " + mytime[2] + "日</h2>" +
55.         "<h2>具体时间:  " + infos[4] + "</h2>" +
56.         "<h2>缴    费:  " + infos[5] + "</h2>";
57.         reslist.setText(Html.fromHtml(totallist));
58.     }
59.     class sendTask extends AsyncTask{
60.         @Override
61.         protected Object doInBackground(Object[] params) {
62.             String path = params[0].toString();
63.             String totalinfo=params[1].toString();//获得清单信息
64.             try {
65.                 //获取网络上 get 方式提交的整个路径
66.                 URL url = new URL(path+"?totalinfo="+totalinfo);
67.                 //打开网络连接
68.                 HttpURLConnection conn = (HttpURLConnection) url.openConnection();
69.                 //设置提交方式
70.                 conn.setRequestMethod("GET");
71.                 //设置网络超时时间
72.                 conn.setConnectTimeout(5000);
73.                 if (conn.getResponseCode() == 200) {
74.                     //用 io 流与 web 后台进行数据交互
75.                     InputStream is = conn.getInputStream();
76.                     //字节流转字符流
77.                     BufferedReader br = new BufferedReader(new InputStreamReader(is));
78.                     //读出每一行的数据
79.                     String str = br.readLine();
80.                     System.out.println(str);
81.                     //返回读出的每一行的数据
82.                     return str;
83.                 }
84.             } catch (MalformedURLException e) {
85.                 e.printStackTrace();
86.             } catch (ProtocolException e) {
87.                 e.printStackTrace();
88.             } catch (IOException e) {
89.                 e.printStackTrace();
90.             }
91.             return null;

```

```
92.     }
93.
94.     @Override
95.     protected void onPostExecute(Object o) {
96.         super.onPostExecute(o);
97.         //获取 android 与 web 数据交互获得的值
98.         String s = (String) o;
99.         System.out.println("插入完成:"+s);
100.        //接下来处理订单存入 index
101.        Intent i =new Intent(reserve.this,hospitalindex.class);
102.        i.putExtra("totalinfos",totallist);
103.        startActivity(i);
104.    }
105. }
106. public void sendinfo(){
107.     String path=MainActivity.basepath+"reserve.jsp";
108.     String totalinfo=infos[0];
109.     for(int i=1;i<infos.length;i++){
110.         totalinfo+=","+infos[i];
111.     }
112.     new sendTask().execute(path,totalinfo);
113. }
114. }
```

7 运行结果

7.1 登陆/注册界面



图 7-1 登陆界面



图 7-2 注册界面

7.2 选择城市



图 7-3 选择城市界面



图 7-4 选择城市定位

7.3 选择医院



图 7-5 医院列表

7.4 选择科室



图 7-6 科室列表

7.5 展示医生列表



图 7-7 医生列表

7.6 展示医生详细信息



图 7-8 某医生的详细信息

7.7 展示医生日程表



图 7-9 某医生第 20 日出勤情况

7.8 点击 20 日下午出勤预约



图 7-10 预约罗玲医生清单

7.9 点击确认预约



图 7-11 主页成功出现预约单



图 7-12 再次预约晚上出现两张预约单

8 调试报告及遇到的问题

8.1 访问后端失败

```
D/NetworkSecurityConfig: No Network Security Config specified, using platform default
W/System.err: java.io.IOException: Cleartext HTTP traffic to 10.0.72.187 not permitted
    at com.android.okhttp.HttpHandler$ClearTextURLFilter.checkURLPermitted(HttpHandler.java:124)
    at com.android.okhttp.internal.huc.HttpURLConnectionImpl.execute(HttpURLConnectionImpl.java:462)
    at com.android.okhttp.internal.huc.HttpURLConnectionImpl.getResponse(HttpURLConnectionImpl.java:411)
    at com.android.okhttp.internal.huc.HttpURLConnectionImpl.getResponseCode(HttpURLConnectionImpl.java:542)
```

图 8-1 未加密的数据传输被拒绝

看错误知道，它说是因为明文 http 传输不被允许，那么采用加密是比较好的做法，但是我只是做一个登录，感觉特地去调库用加密比较没必要，所以网上搜了搜如何取消 google 的明文传输禁令，发现

`android:usesCleartextTraffic="true"`

可以这样允许不加密传输。

8.2 未在 web.xml 指定映射导致报错

之前没有在 web.xml 指定 servlet 的名字和类名，导致前端访问后端时找不到对应的 doLogin 文件，因为 doLogin.java 是 java 文件，不像 jsp 会主动转换成 class 存在服务器端，java 文件必须手动在 web.xml 指明映射。

这也是我经常漏掉的地方，因为不常去看 web.xml 文件，不够关注。

如下图所示就可以了。

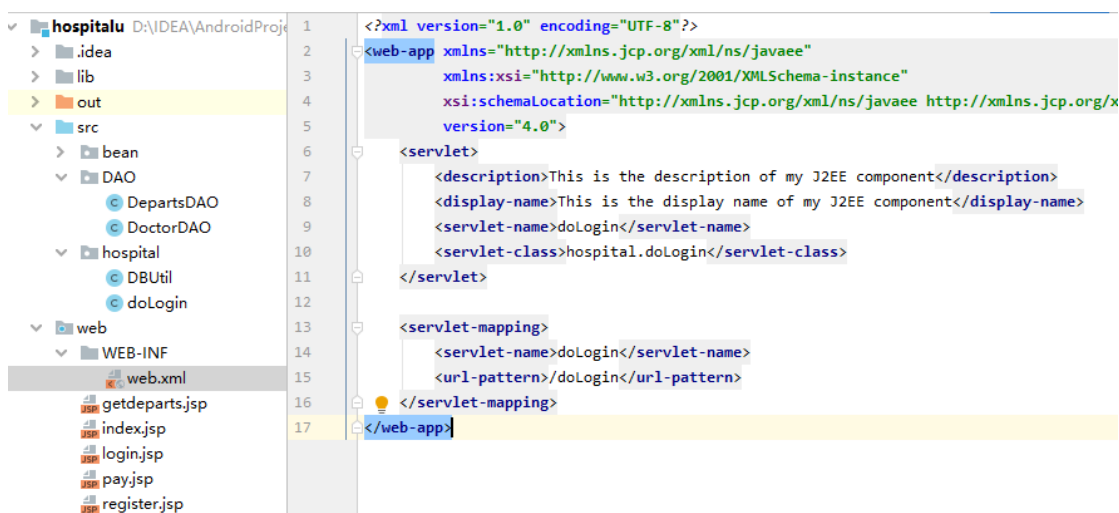


图 8-2 在 web.xml 指定 java 映射

8.3 Android.support 和 Androidx 混用报错

```
E/AndroidRuntime: FATAL EXCEPTION: main
Process: com.example.myhospital, PID: 5993
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.example.myhospital/com.example.myhospital.hosp
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3270)
    at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:3409)
    at android.app.servertransaction.LaunchActivityItem.execute(LaunchActivityItem.java:83)
    at android.app.servertransaction.TransactionExecutor.executeCallbacks(TransactionExecutor.java:135)
    at android.app.servertransaction.TransactionExecutor.execute(TransactionExecutor.java:95)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2016)
    at android.os.Handler.dispatchMessage(Handler.java:107)
    at android.os.Looper.loop(Looper.java:214)
    at android.app.ActivityThread.main(ActivityThread.java:7356) <1 internal call>
```

图 8-3 支持库混用报错

因为谷歌用 AndroidX 取代了支持库，并做了映射。

Android Support 或 AndroidX 不能同时存在

所以在 xml 中的 android.support 都得改成 androidx，因为我的是 API29，比较新的 android 版本

```
<android.support.design.widget.TabLayout
    android:id="@+id/tabLayout"
    android:layout_width="match_parent"
    android:layout_height="100dp">
</android.support.design.widget.TabLayout>
<android.support.v4.view.ViewPager
    android:id="@+id/page"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</android.support.v4.view.ViewPager>
```

图 8-4 支持库混用

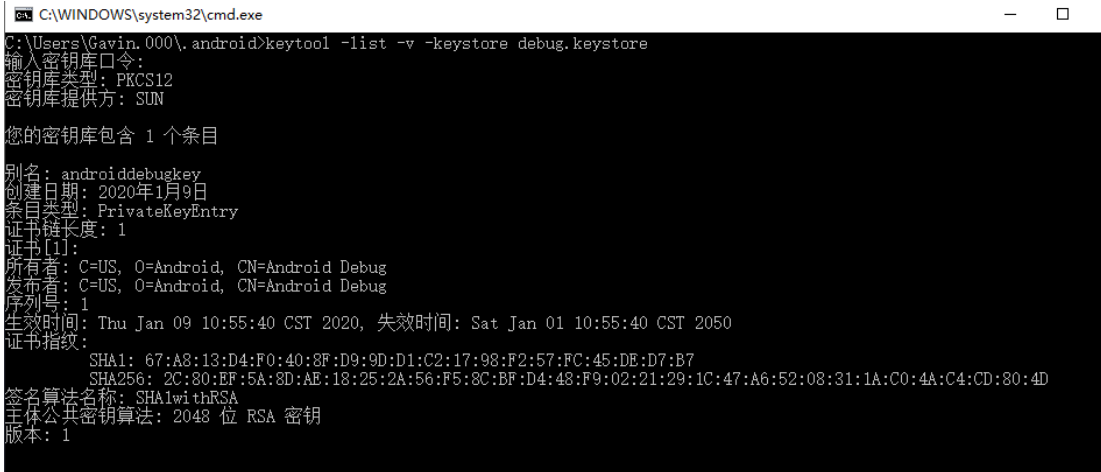
```
<com.google.android.material.tabs.TabLayout
    android:id="@+id/tabLayout"
    android:layout_width="match_parent"
    android:layout_height="100dp">
</com.google.android.material.tabs.TabLayout>
<androidx.viewpager.widget.ViewPager
    android:id="@+id/page"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</androidx.viewpager.widget.ViewPager>
```

图 8-5 支持库正确用法

8.4 Viewpager 总是吃掉下面的组件

在做 index 界面时，想在顶部栏做一个选择框，下面用 viewpager 最后下面添加一个“点击挂号”的 button，但是发现由于 viewpager 的影响，因为它是 view 组件，就算设置为 wrap_content，它也会把剩下的屏幕占满，我又不想固定 viewpager 的高度，这会降低 app 机型适应性，查找书本发现可以用两个 linearlayout 分隔开上下，把 button 放到底部的 linearLayout 并且固定高度，然后上面的 linearLayout 设置权重为 1 浮动即可。

8.5 申请高德 API 时申请 SHA1 口令



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Gavin.000\.android>keytool -list -v -keystore debug.keystore
输入密钥库口令:
密钥库类型: PKCS12
密钥库提供方: SUN

您的密钥库包含 1 个条目

别名: androiddebugkey
创建日期: 2020年1月9日
条目类型: PrivateKeyEntry
证书链长度: 1
证书 [1]:
所有者: C=US, O=Android, CN=Android Debug
发布者: C=US, O=Android, CN=Android Debug
序列号: 1
生效时间: Thu Jan 09 10:55:40 CST 2020, 失效时间: Sat Jan 01 10:55:40 CST 2050
证书指纹:
  SHA1: 67:A8:13:D4:F0:40:8F:D9:9D:D1:C2:17:98:F2:57:FC:45:DE:D7:B7
  SHA256: 2C:80:EF:5A:8D:AE:18:25:2A:56:F5:8C:BF:D4:48:F9:02:21:29:1C:47:A6:52:08:31:1A:C0:4A:C4:CD:80:4D
签名算法名称: SHA1withRSA
主体公共密钥算法: 2048 位 RSA 密钥
版本: 1
```

找了好几种方法才实现，因为我的时jdk13，版本较高，网上的老版本都不合适了。

9 收获与感悟

这次课设，把我们的移动应用开发，数据库，web，前后端都揉到了一起，在完全是自己独立完成的情况下，我确实学到了不少的东西，代码敲得更熟练了，但这并不是最大的收获，最大的收获是我在课程压力下完成了很多小小的突破，比如我之前从没用过 RecyclerView，想尝试用用，结果被网上参差不齐的教程，和复杂的机制——要有 item，要有 bean 数据类，要有 adapter 适配器，还有一堆支持库问题，让我失望想要放弃这个新组件，转而去用 listview，但是想了想都看了不少 RecyclerView 的源码了，半途而废太亏，然后就压住烦闷的心情，再自己摸索一段，最后还是把 RecyclerView 变成了我自己的知识库一员。

这次课设不是像以前那样有着充足而扎实的理论基础，这次课设要用的东西我很多都是一知半解，硬着头皮边看文档边敲代码实现的，感觉比平时顺风顺水地做完获得了更大的收获，也提高了我的自信，只要沉下心来，很多看似有难度的问题，实则不过如此。

另外，我发现 java 确实比 C++ 更容易让人有成就感，毕竟 java 移动开发，代码写出来了，很多安卓机就能用，而且界面，逻辑都还过得去，也不是很难实现，而反观 C++，换台操作系统可能就会运行出错，而且没有自带的图形界面，一般还是拿来写短而精悍的算法题更拿手。

另外，马上就要大三下了，大学生活就快要结束了，我们更应该沉下心来，多看多练多实战，计算机毕竟是一门工科，不论之后是读研还是工作，代码复现能力都非常重要。

参考文献

[1]<https://blog.csdn.net/tuike/article/details/95937072>.

[2]《深入Android应用开发：核心技术解析与最佳实践》，苗忠良、曾旭、宛斌著，第一版，机械工业出版社.

[3]《Android编程权威指南》，Bill Phillips Brian Hardy著，第三版，人民邮电出版社.