移动应用开发期末
大作业报告

姓　　名：_____李飞飞_____

班　　级：_____软工 2206_____

题　　目：____King of Bots____

手 机 号：____17857329420____

队伍编号：_____33_____

二零二三年五月制

一、功能说明

由贪吃蛇游戏改编而来，是一个前后端分离的游戏项目。游戏为两名玩家在线联机对战。由两个后端和两个前端，两个前端分别为 Vue 编写的网页端和原生安卓编写的客户端。两个后端分别为提供主服务的 backendcloud 和提供敏感词屏蔽功能的 SensitiveWord。其中 backendcloud 由 backend, MatchingSystem, BotRunningSystem 三个微服务组成

比赛为在线对战回合制 1v1 游戏，玩家操控一条蛇躲避障碍物并设法堵住对方以取得胜利。玩家可以上传由 java 编写的 Ai 来代替自己出战并赢取天梯积分。支持赛后查看对局录像，并实时公布天梯排名。

程序模拟了游戏引擎，对局所有元素继承自 GameObject, 拥有 start(), update(), destroy(), render()等多个函数，通过状态机记录游戏类，并且每秒刷新 60 次自身状态最终实现了游戏效果。

程序封装了数据持久层接口，使得访问与记录数据变得十分简便。玩家每次对战记录会被保存在自己本地数据库中供玩家分析也可在线查看所有的比赛对局回放。同时通过 SharedPreferences 来记录各项数据，实现了 JWT 验证登录、快捷登录、快速开启游戏、主题切换的功能。

程序封装了大量网络接口，屏蔽了各种数据不一致导致的问题，实现了前后端分离的效果，使得移动端的玩家可以和网页端玩家同台

竞技。

在主要的游戏功能外海提供了一个敏感词屏蔽的功能，玩家输入文本后会自动过滤其中的隐私、身份信息和其它危险内容。通过调用 ChatGPT 接口实现此功能，APP 调用 backendcloud 的接口，backendcloud 再通过 RestTemplate 调用 SensitiveWord 接口，SensitiveWord 通过 https 请求实现对 ChatGPT 的调用。

本 APP 还提供了切换主题色、快速开始游戏等功能

二、需求分析

2.1 用户用例图

用户可以登录注册、与其他玩家匹配游戏、查看自己/所有对局回放列表、查看天梯排行、上传，编辑自己的代码、使用敏感词屏幕模块、切换主题、快速开始游戏

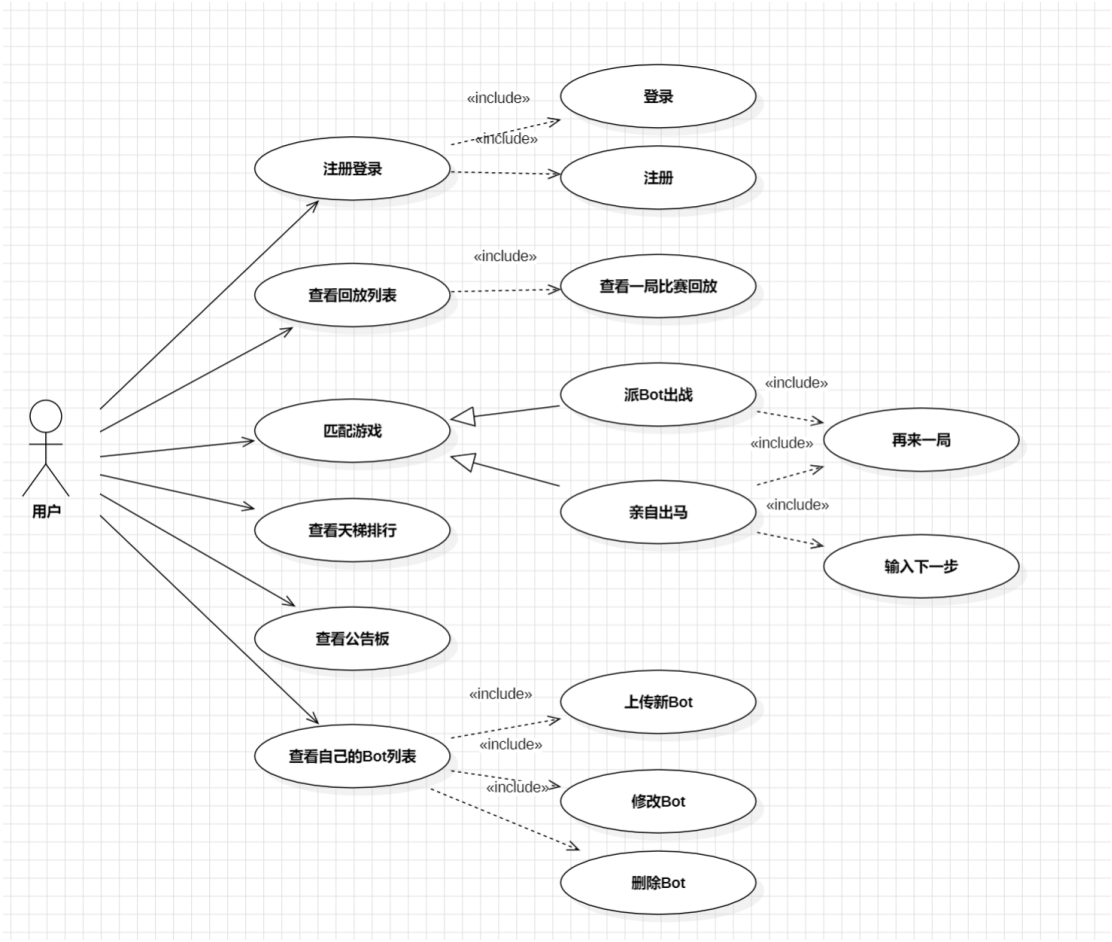


图 二-1 用户用例图

2.2 用户用例分析

用例一：用户注册 **用例描述：如题** **参与者：用户** **前置条件：**用户首次使用 APP
1、输入账号密码 2、再次确认密码 3、确认注册 4、注册成功 5、自动登录并跳转主页面

用例二：　用户登录 **用例描述：如题** **参与者：用户** **前置条件：**用户首次使用 APP 或 token 过期
1、输入账号密码 2、确认登录 3、登录成功 4、自动登录并跳转主页面

用例三： 用户查看回放列表 **用例描述：如题** **参与者：用户** **前置条件：用户已登录**
1、用户点击“对局列表” 2、出现所有对局记录

用例四： 用户查看一局对战回放 **用例描述：如题** **参与者：用户** **前置条件：用户已登录**
1、用户点击“对局列表” 2、出现所有对局记录 3、用户点击按钮“查看回放” 4、开始播放对应的对局

用例五： 用户开始一局游戏并派 Bot 出战 **用例描述：如题** **参与者：用户** **前置条件：用户已登录，用户已有 Bot**

1、用户点击“对战” 2、点击“亲自出马”旁边下拉选项，选择一个 Bot 3、点击开始匹配 4、等待匹配成功 5、匹配成功，开始游戏

用例六： 用户开始一局游戏并亲自出马 **用例描述：如题** **参与者：用户** **前置条件：用户已登录**
1、用户点击“对战” 2、点击开始匹配 3、等待匹配成功 4、匹配成功，开始游戏 5、手动输入操作

用例七： 用户查看天梯排行 **用例描述：如题** **参与者：用户** **前置条件：用户已登录**

1、用户点击“排行榜” 2、出现天梯排行榜

用例八： 用户查看公告板 **用例描述：如题** **参与者：用户** **前置条件：用户已登录**
1、用户点击“公告板” 2、出现公告栏目

用例九： 用户上传新 Bot **用例描述：如题** **参与者：用户** **前置条件：用户已登录**
1、点击右上角昵称 2、点击“我的 Bot” 3、出现所有 Bot 列表 4、点击“创建 Bot” 5、输入 Bot 名称，简介，代码 6、点击“确认”按钮

7、创建成功，列表里出现新 Bot

用例十：　用户修改 Bot **用例描述：如题** **参与者：用户** **前置条件：用户已登录，登录用户已有 Bot**
1、点击右上角昵称 2、点击“我的 Bot” 3、出现所有 Bot 列表 4、点击要修改的 Bot 5、输入新的 Bot 名称，简介，代码 6、点击“确认”按钮 7、修改成功

用例十一：　用户删除 Bot **用例描述：如题** **参与者：用户** **前置条件：用户已登录，登录用户已有 Bot**
1、点击右上角昵称 2、点击“我的 Bot”

3、出现所有 Bot 列表 4、点击要删除的 Bot 6、点击“确认”按钮 7、删除成功，Bot 从列表中移除

用例十二： 用户退出登录 **用例描述：如题** **参与者：用户** **前置条件：用户已登录**
1、点击右上角昵称 2、点击“退出” 3、退出成功

三、用户页面设计

本项目包括 14 个用户页面：

1. 匹配页面
2. 对局页面
3. 排行榜页面
4. 全局回放页面
5. 我的回放页面
6. 回放页面

7. 个人信息页面

8. 我的 Bot 页面

9. 修改、添加 Bot 页面

10. 铭感词检测页面

11. 登录页面

12. 注册页面

13. 主题切换后页面

14. 快速开始游戏页面

3.1 用户界面的内部功能介绍

1. 匹配页面

登录后默认所在页面，玩家可以再次选择自己出战的 Bot 并开始游戏，也可亲自出马。开始匹配后会优先匹配天梯积分接近的玩家，若匹配过久也可能匹配到积分差距较大的对手。匹配成功后会展示对手身份信息并跳转到对局页面。

玩家进入页面即开启 WebScoket 连接与后端持续通信，直至此页面不可见时断开

2. 对局页面

对局页面玩家如果是亲自出马对战就可以操作自己的游戏角色，每三回合蛇会伸长一格。玩家要在确保自身存货情况下堵住对方的去路以此取得胜利。对局结束后会把本次对局记录在本地

数据库中供玩家查看。

玩家每次操作会通过 WebScoket 通知后端，后端也通过 WebScoket 连接返回对方的操作与本轮操作的结果

3. 排行榜页面

系统会根据玩家的天梯积分为所有玩家排名，初始默认积分 1500，每次获胜+5，失败-3（游戏操作步数小于 5 步的对局将被忽略）。玩家通过创造更强的 Bot 来达到更高的排名。

此处展示玩家排名、玩家积分、玩家头像、玩家昵称。数据通过访问 getRankList 结果获取，再根据 RankItemAdapter 与条目适配后呈现。

4. 全局回放页面

此页面展示所有有被记录的对局，玩家点击即可查看对应比赛的回放。本页面与“我的回放页面”通过 TabView 与 ViewPage2 实现了滑动切换的效果

此处展示对局的两名玩家头像、昵称与对局时间。数据通过访问 getRecordList 结果获取，再根据 RecordItemAdapter 与条目适配后呈现。

5. 我的回放页面

此页面展示当前登录玩家的在此设备的游戏记录，玩家可以查看回放或获取数据库数据做分析用以改进自己的 Bot 代码

6. 回放页面

玩家在“全局回放页面”或“我的回放页面”点击对局条目后可跳转到此页面。此页面会解析回放数据并在模拟的游戏引擎中依次执行以达到重现对局的效果

如果时从“全局回放页面”进入则数据通过网络请求获取。如果是通过“我的回放页面”进入则数据通过访问 SQLite 获取

7. 个人信息页面

此页面展示用户信息，包括头像，昵称，天梯积分等信息，此页面也是跳转“我的 Bot” 页面和“敏感词检测”页面的入口。用户也能在此页面切换主题或是退出登录。

数据通过访问 getInfo 接口获取

8. 我的 Bot 页面

此页面展示用户所有 Bot 列表,展示的信息包括 Bot 名称、创建时间。用户也可以修改、删除、新建 Bot。

数据通过 getBotList 接口走网络请求获取

9. 修改、添加 Bot 页面

增删改分别走对应的接口，此处展示 Bot 的名称、简介与代码内容。其中代码内容为空，如果没有填写代码则无法新建 Bot

10. 敏感词检测页面

玩家输入文本后会自动过滤其中的隐私、身份信息和其它危险内容。通过调用 ChatGPT 接口实现此功能，APP 调用 backendcloud 的接口，backendcloud 再通过 RestTemplate 调用 SensitiveWord 接口，SensitiveWord 通过 https 请求实现对 ChatGPT 的调用。还提供了一键复制脱敏文本的功能。

11. 登录页面

本项目的登录模式为 JWT 鉴权登录。Token 是用户的凭证。此页面要求玩家输入账号密码,确认登录会走 login 的接口，若后端认定登录成功会分配一个 token，后续 token 是用户的唯一凭证，token 过期后强制下线重新登录。

登录成功后会通过 SharedPreferences 记录数据，后续用户点击即可进入游戏，无需再次登录。

玩家密码在后端是加密存储的，每次登录判断用户输入密码与加密密码是否对应。

登录成功后会情况当前 Activity 栈并跳转到主界面，用户不退出登录无法回到此页面。

12. 注册页面

类似登录页面，用户输入昵称、密码、确认密码后若昵称无重复则注册成功。注册成功后会自动登录，用户无需再额外操作。

13. 主题切换后页面

APP 提供了两种主题：红色与蓝色，在 Application 初始化时加载本次主题，用户切换主题后再次打卡 APP 即可体验新主题。

14. 快速开始游戏页面

本页面玩家长按图标即可启动，也可以将快捷方式注册在页面上即可一次点击启动。点击后若 token 未过期则直接登录并且派上一次对战的 Bot 出战，若为第一次游戏则亲自出马

3.2 用户界面的截图

1. 匹配页面

图 三-1 匹配页面

2. 对局页面

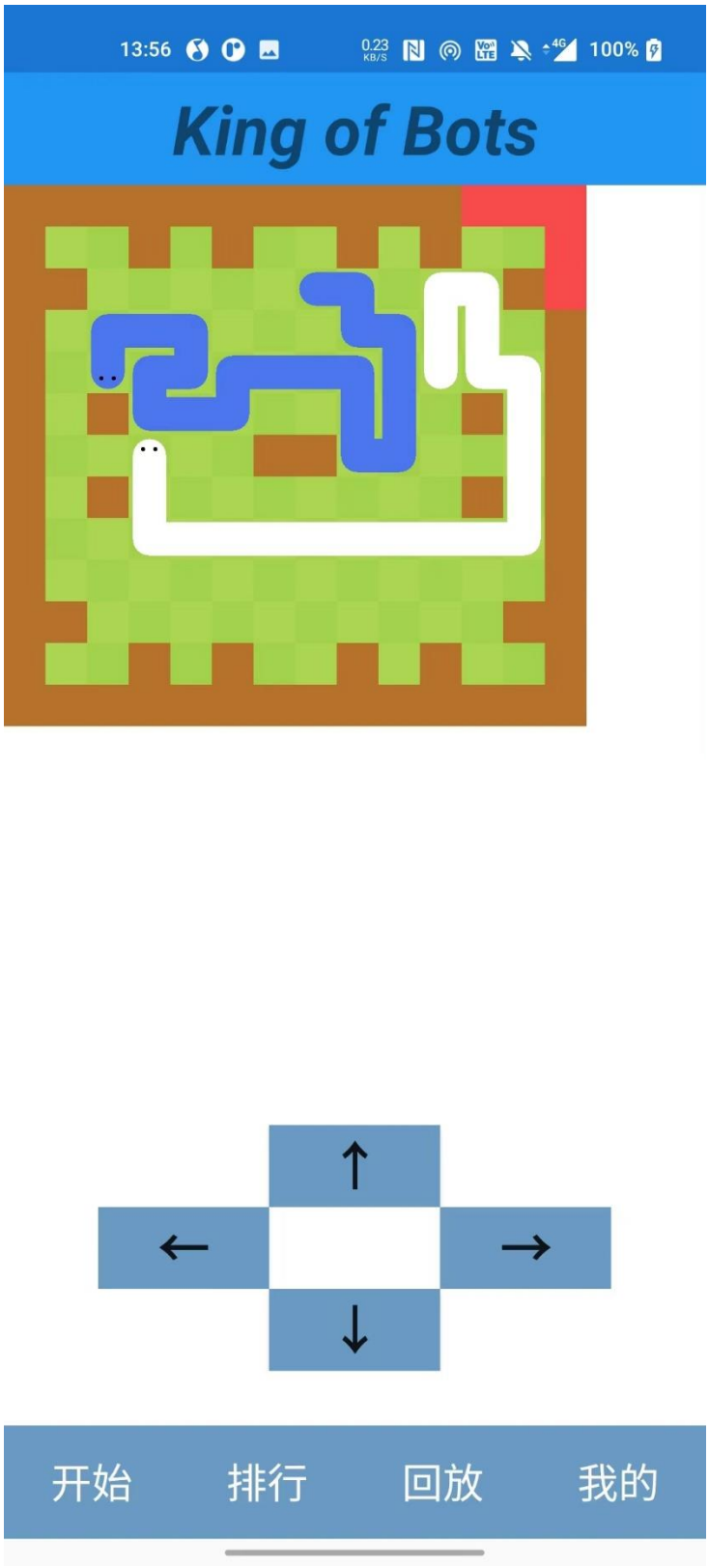


图 三-2 游戏页面

3. 排行榜页面

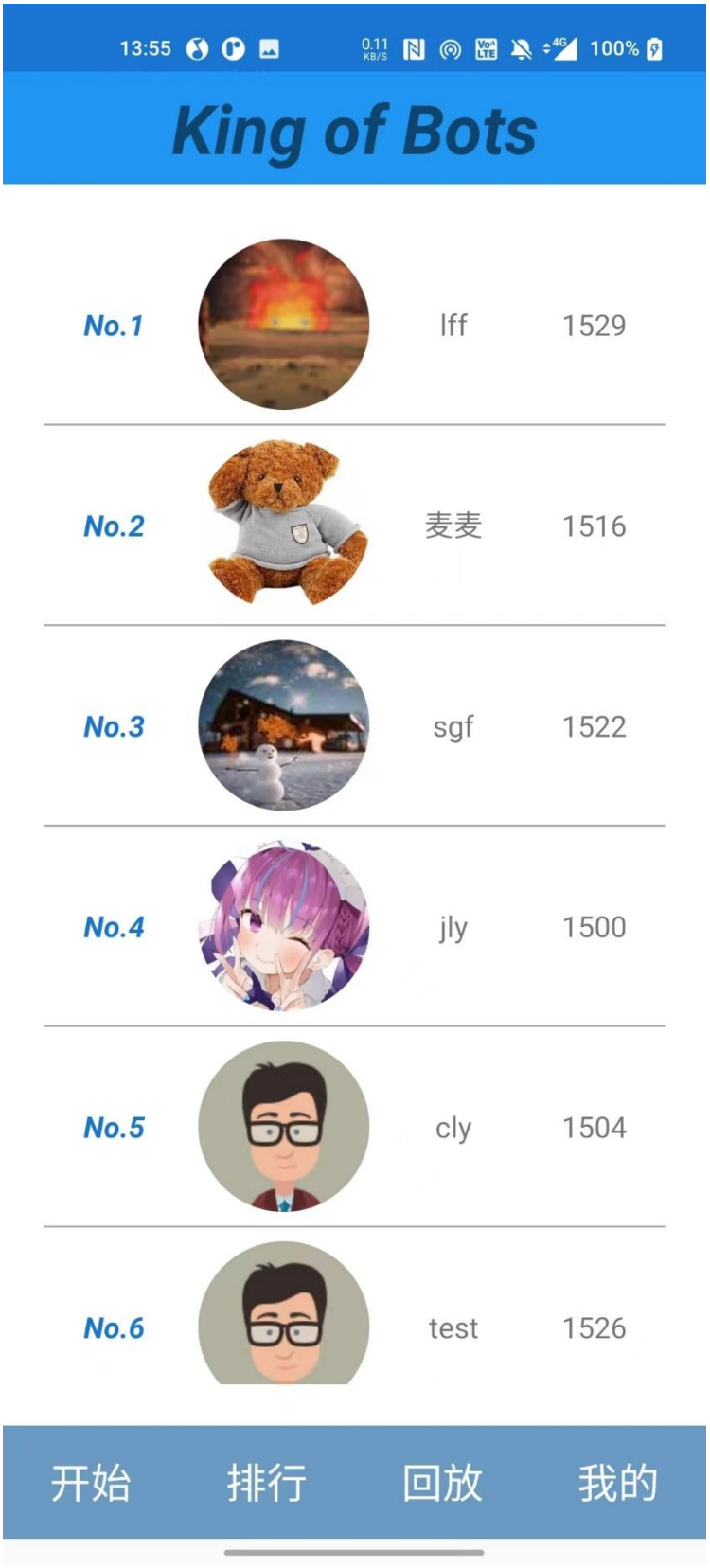


图 三-3 排行页面

4. 全局回放页面

图 三-4 所有回放页面

5. 我的回放页面

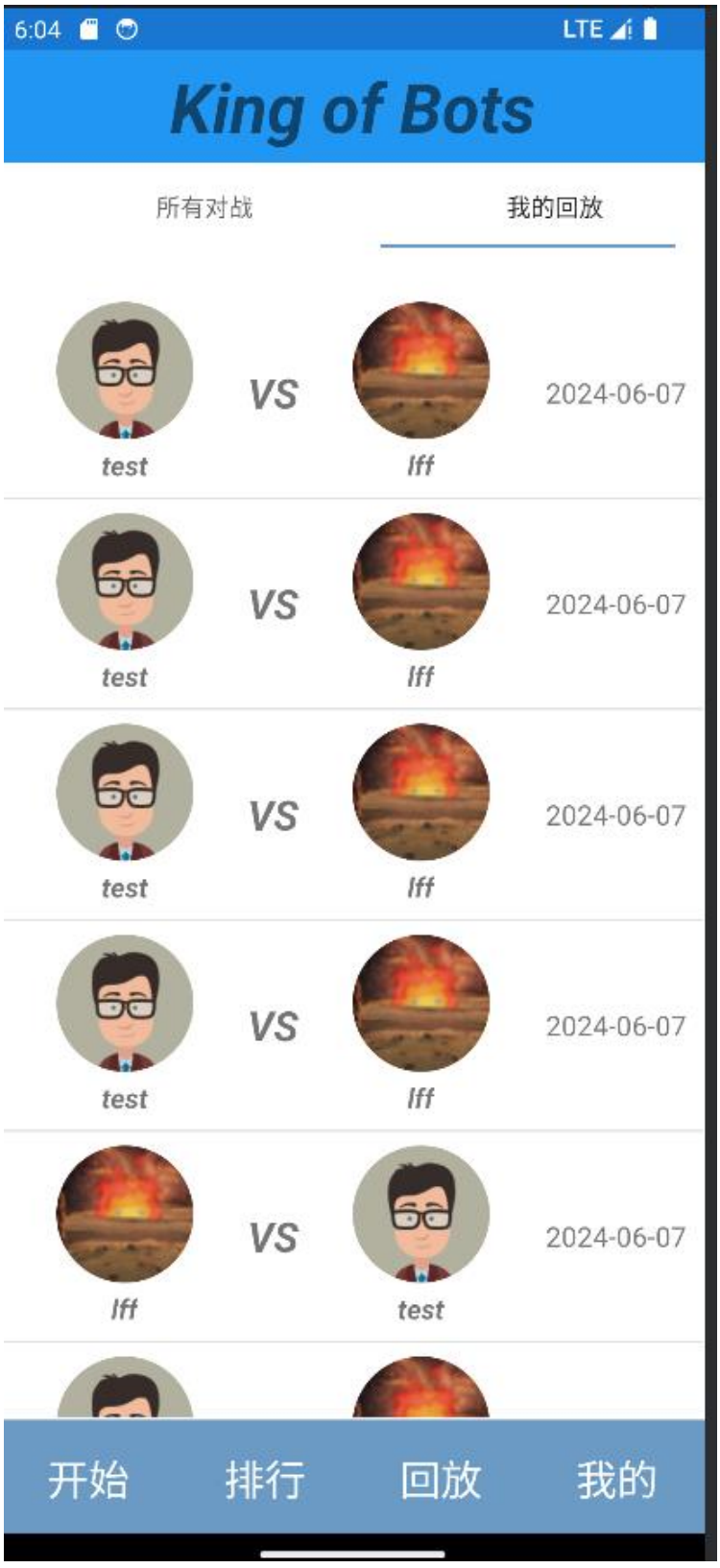


图 三-5 我的回放页面

6. 回放页面

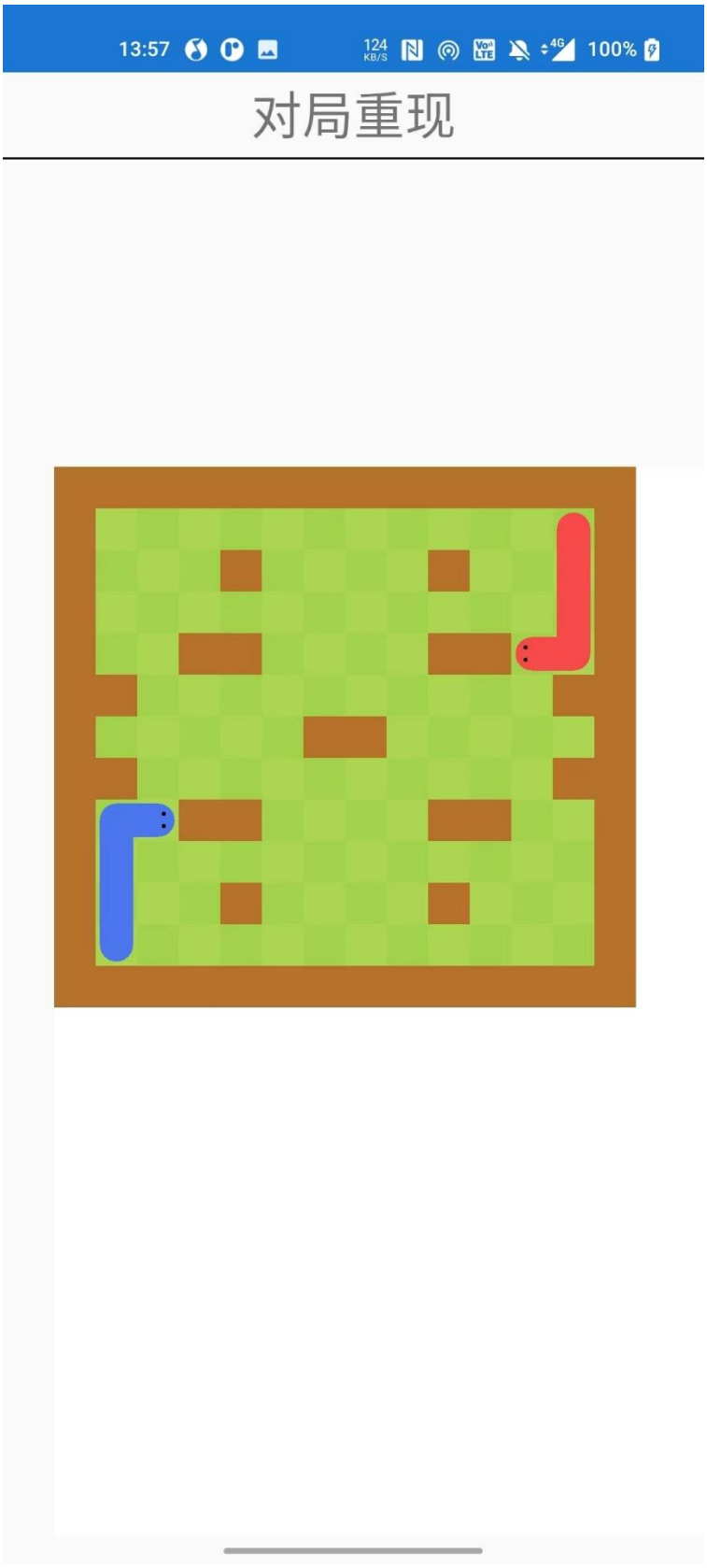


图 三-6 回放页面

7. 个人信息页面

图 三-7 个人信息页面

8. 我的 Bot 页面

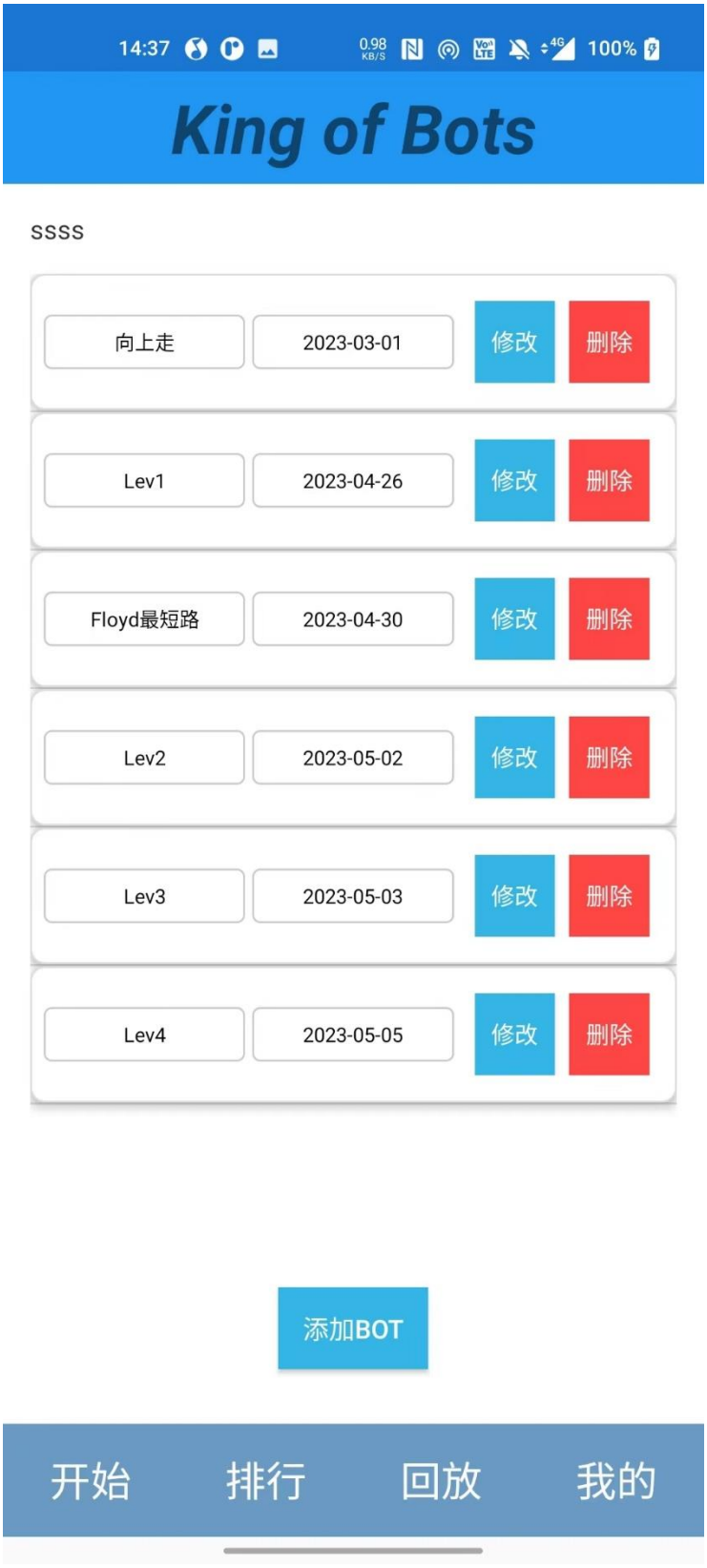


图 三-8 Bot 列表页面

9. 修改、添加 Bot 页面

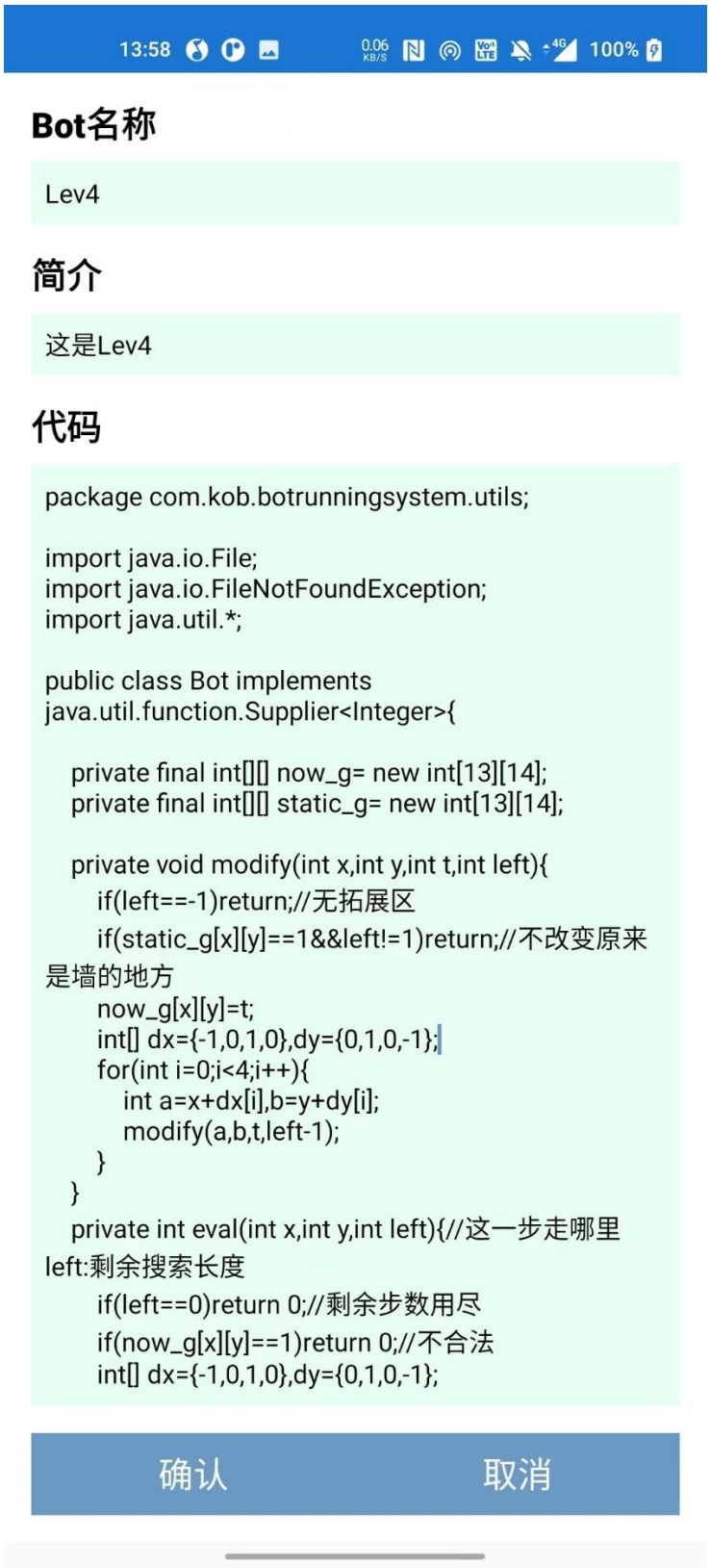


图 三-9 编辑 Bot 页面

10. 敏感词检测页面

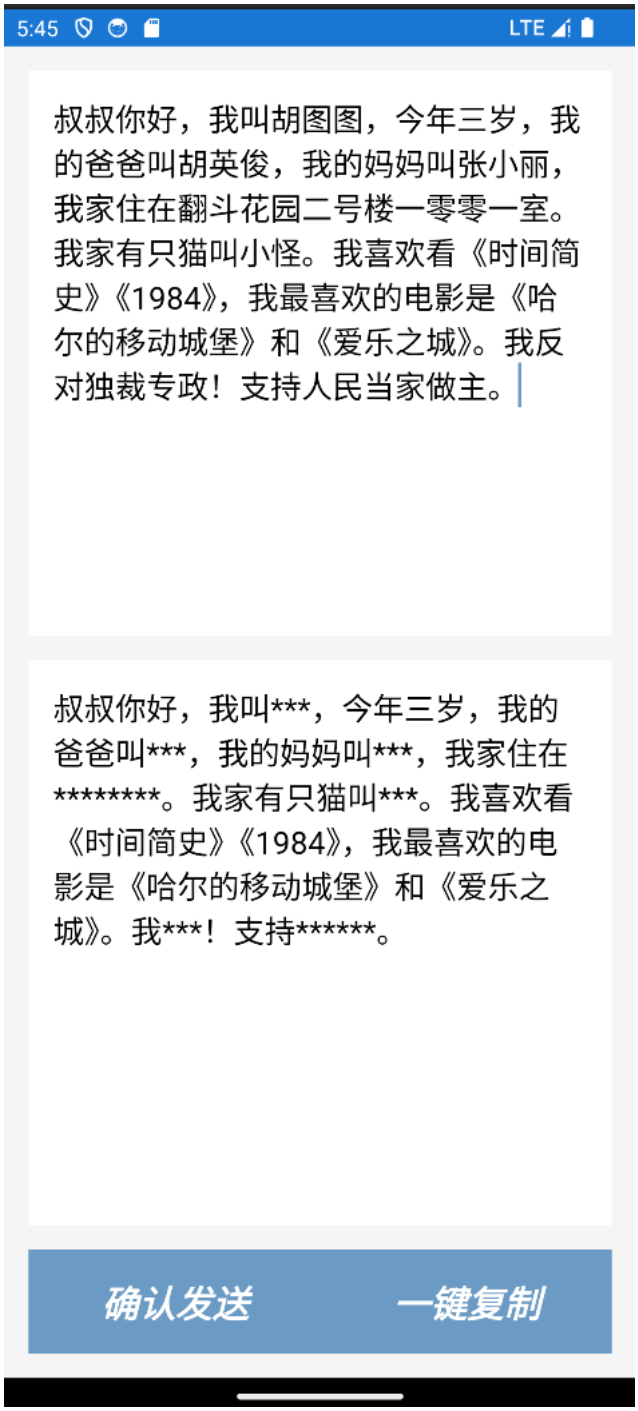


图 三-10 敏感词屏蔽页面

11. 登录页面

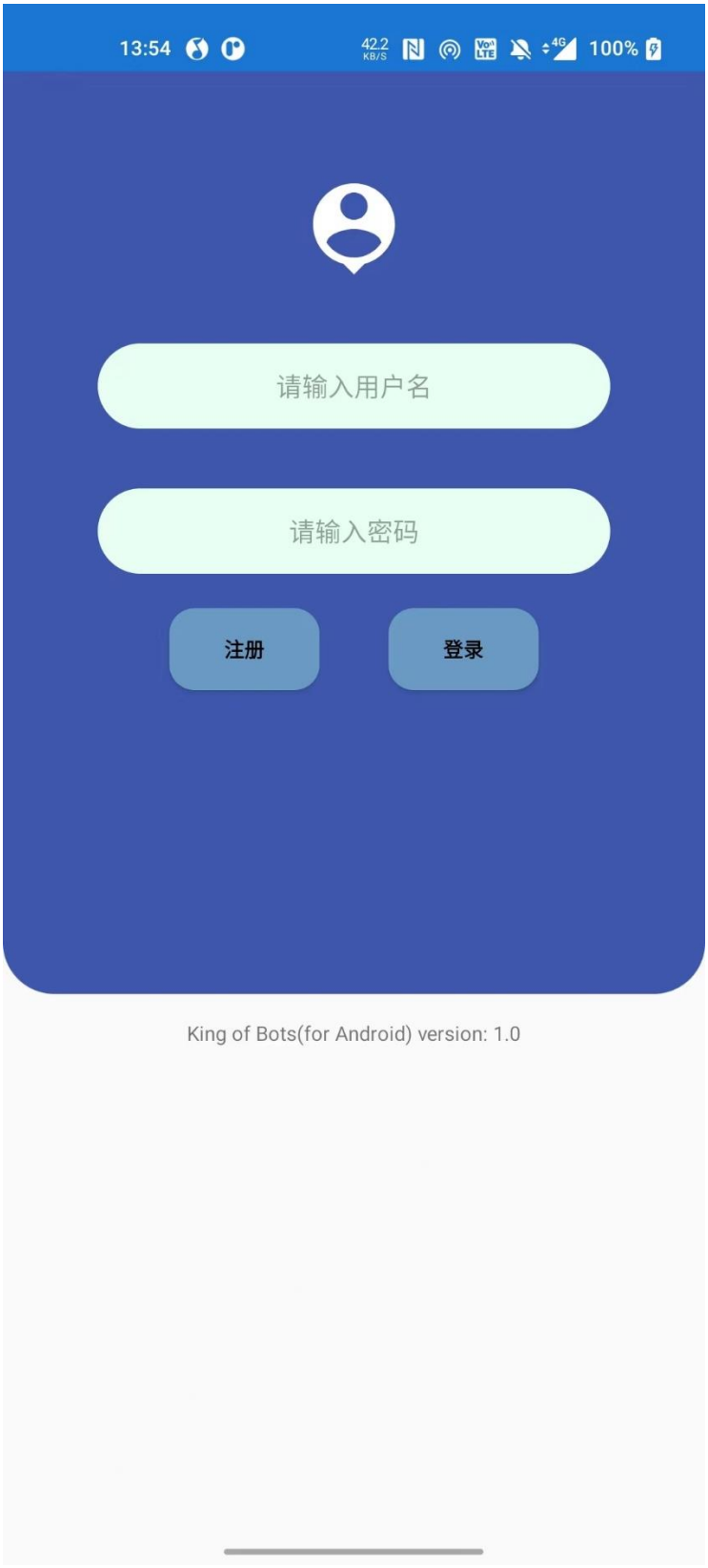


图 三-11 登录页面

12. 注册页面

图 三-12 注册页面

13. 主题切换后页面

图 三-13 切换主题页面


13:58
2.57 KB/S
100%
King of Bots
lff
1529
我的Bot
退出登录
修改主题色
敏感词检测
开始
排行
回放
我的

图 三-14 红色主题页面

14. 快速开始游戏页面

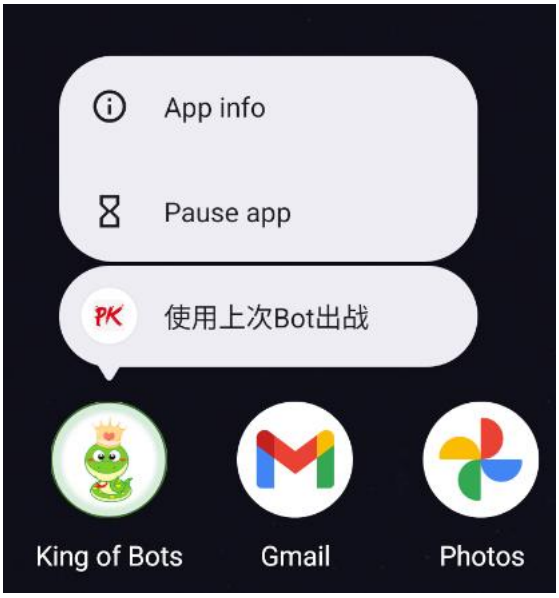


图 三-15 logo 与快速开战

四、数据库设计

4.1 数据功能介绍

本应用主要存储两种数据：

1）配置信息：因为配置信息的数据量很小，从 Android 支持的存储方式上分析，可以保存在 SharePreference、文件或 SQLite 数据库中

2）本地回放信息：回放条目数据量会随着用户的使用而逐渐增多，且格式固定所以用 SQLite 存

4.2 数据格式

现将数据格式说明如下：

1）配置信息中主要保存用户数据（昵称、头像、积分、id）、token、当前主题、上次出战 BotId(默认-1)

配置信息的数据库表结构

属性	数 据 类型	说明
userName	string	用户昵称
photo	string	用户头像
rating	integer	天梯积分
id	integer	玩家 Id
theme	string	当前主题，默认蓝色
botId	integer	上次出战 Bot Id
token	string	JWT 登录凭证

2）对局回放数据

回放服务保存了两位玩家的昵称、头像、对局结果以及 JSON 格式下的回放数据。

JSON 格式回放数据中由包含了两位玩家的起始 id, 玩家操作序列、地图等内容, 游戏运行模块通过解析反序列化后的 record 按操作一步步移动即可展示对应的历史对局

玩家对局回放的数据库表结构

属性	数据类型	说明
a_photo	string	玩家 a 的头像
a_username	string	玩家 a 的昵称
b_photo	string	玩家 b 的头像
b_username	string	玩家 b 的昵称
record	string	JSON 格式的回放信息
result	string	对局结果

五、程序模块设计

整个应用程序划分为 4 个模块，分别是用户界面、游戏运行模块、数据库模块、网络请求模块、后端（BackendCloud 与 SenstiveWord）

模块结构图：

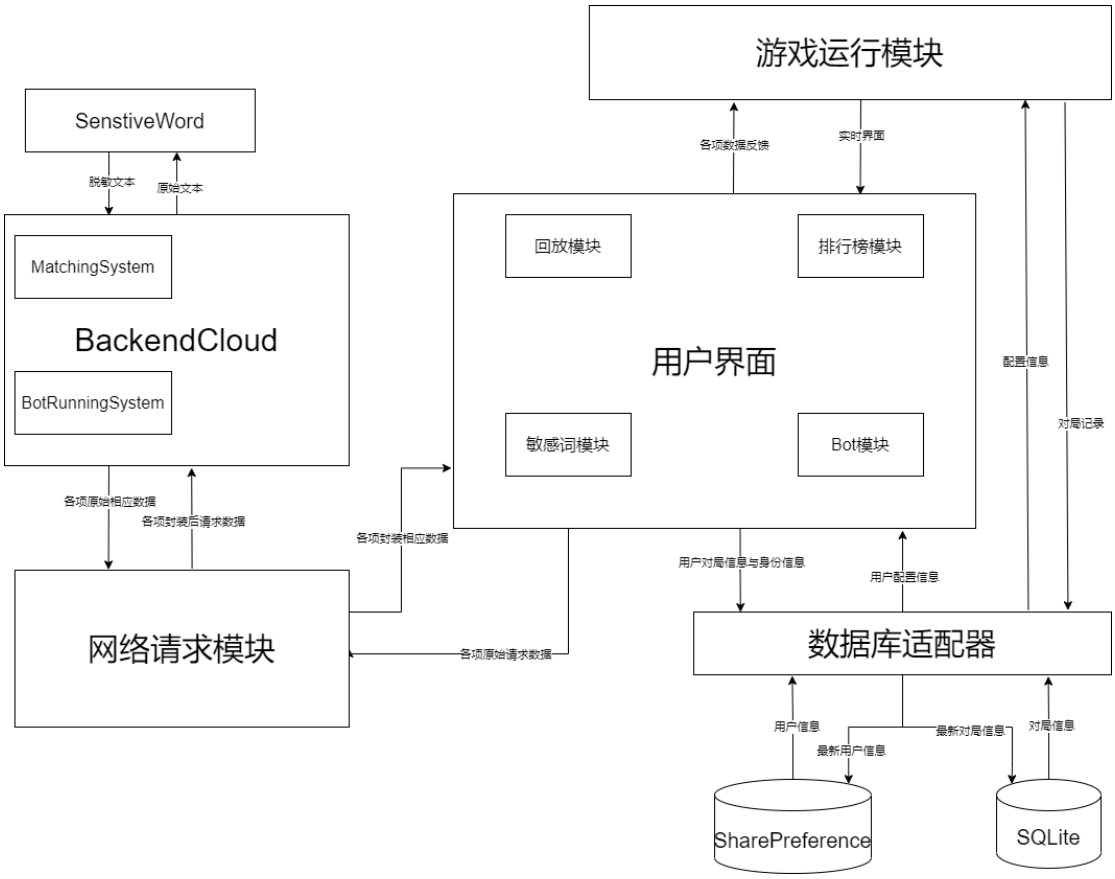


图 五-1 程序模块图

由模块结构图中可知，用户界面是整个应用程序的核心，主要包含四个子模块，1.回放模块，用于展示历史对局；2.敏感词模块，用于文本脱敏；3.排行榜模块，查看最新排行；4.Bot 模块，用于对自己的 Bot 增删改查

游戏运行模块由匹配或回放启动，不断从用户界面或数据库获取数据并解析，实时渲染在画面上

网络请求模块屏蔽了大量的网络接口格式、鉴权、返回数据格式的问题，为用户界面提供方便的数据获取途径

后端由 BackendCloud(含 Backend 、 BotRunningSystem 、 MatchingSystem 三部分)和 SenstiveWord 两个后端组成，Backend 中会调用 SenstiveWord 接口，SenstiveWord 再调用 openai 的接口为用户界面提供各种数据

数据库适配器封装了所有对 SQLite 数据库与 SharePreference 操作的方法，用户界面和游戏运行模块会调用它实现数据库和 SharePreference 操作

六、程序逻辑实现

6.1 游戏引擎实现

程序模拟了游戏引擎，游戏基类 GameObject 派生了 Wall Snake GameMap 三个类，GameObject 中定义 start(), update(), destroy(), render() 个私有函数，分别用于执行游戏类创建时、更新时、销毁时、渲染在画布时的行为。GameObject 还有两个静态方法供 TimeThread 调用，每秒调用 60 次以更新和渲染游戏类

```
1 个用法  Casssifa(loacl)
public static void step(long timeStamp) {
    for (GameObject now : list) {
        if (!now.hasCalled) {
            now.hasCalled = true;
            now.start();
        } else {
            now.timeDelta = timeStamp - lastTimeStamp;
            now.update();
        }
    }
    GameObject.lastTimeStamp = timeStamp;
}

1 个用法  Casssifa(loacl)
public static void renderAll(Canvas canvas) {
    //保证蛇在顶层
    for (GameObject now : GameObject.list)
        if (!(now instanceof Snake))
            now.render(canvas);
    for (GameObject now : GameObject.list)
        if (now instanceof Snake)
            now.render(canvas);
}
```

图 六-1 游戏类刷新代码

通过重写 start(), update(), destroy(), render()不同的类就能实现不同的行为与动画效果

GameObject 由一个静态数组用于存放自身的所有实例,如 wall、snake、gameMap

GameObject 的两个静态方法分别为：step()和 rendrAll()。

- step()会更新一次所有数组中的实例状态
- renderAll 会把根据实例当前状态把所有实例全部渲染一次

6.2 贪吃蛇运动

小蛇由一串联通的 canvas 画出的正方形构成，头尾为圆形，头部有眼睛作标识。继承自游戏基类，每秒刷新 60 次，每次清除之前图像再渲染新图像以达到动画效果。每次移动会向前推进头尾格

5.2.2 生命周期：

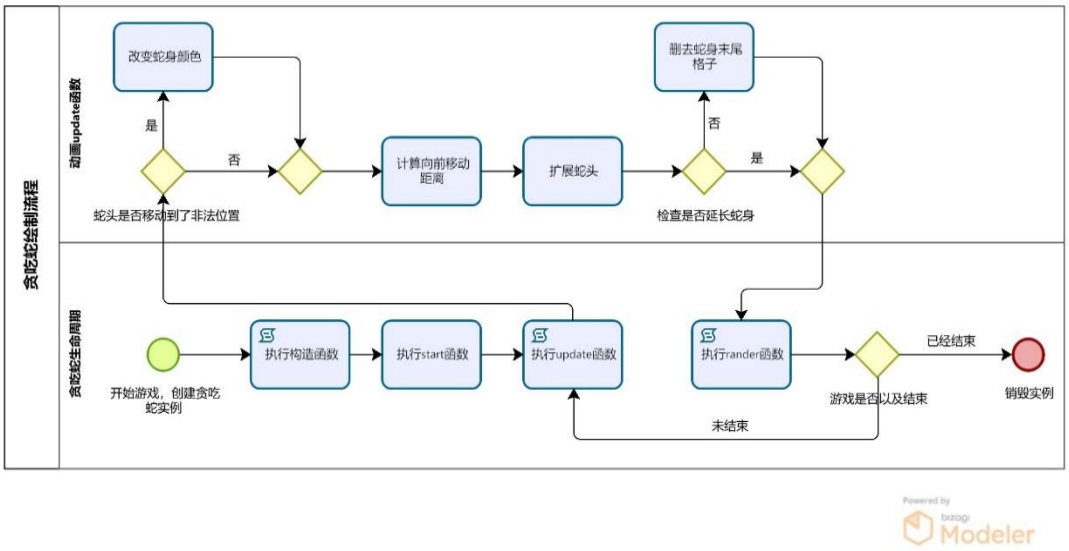


图 六-2 贪吃蛇生命周期泳道图

6.3 网络请求封装

封装了 hilt，需要使用时通过@Inject 注入。在发送请求前通过请求拦截器在请求头加入当前 token

```
public class TokenInterceptor implements Interceptor {

    Casssifa(loacl)
    @NonNull
    @Override
    public Response intercept(Chain chain) throws IOException {
        Request originalRequest = chain.request();

        // 添加 Authorization 头部
        Request.Builder requestBuilder = originalRequest.newBuilder();
        if (UserSharedPreferences.getInstance().getToken() != null)
            requestBuilder.header( name: "Authorization",  value: "Bearer " + UserSharedPreferences.getInstance().getToken());

        Request requestWithAuth = requestBuilder.build();

        // 继续请求链
        Response response = chain.proceed(requestWithAuth);

        return response;
    }
}
```

图 六-3 集成 hilt

6.4 JWT 登录验证与持久化登录、自动登录

每次打开会尝试通过 SharePreference 保存的 token 请求个人信息，如果没有 token 或请求失败则要求重新登录，否则直接进入 MainActivity，跳过 LoginActivity。

注册成功后会携带 intent 跳转 LoginActivity，在 LoginActivity 中判断是否有注册页面传来的 intent，如果有则直接执行登录逻辑

```
Casssifa(loacl)
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //如果是从注册跳转来的
    Intent intent = getIntent();
    if (intent != null && intent.getStringExtra( name: "username") != null) {
        String username = intent.getStringExtra( name: "username");
        String password = intent.getStringExtra( name: "password");
        if (username != null && password != null) {
            tryLogin(username, password);
        }
    } else {
        //使用存储数据尝试登录
        if (UserSharedPreferences.getInstance().getToken() != null) {
            updateUserInfo();
        } else {
            //没有token
            setContentView(R.layout.activity_login);
            initView();
        }
    }
}
```

图 六-4 快速登录

6.5 快速开始对战

快速开始对战会携带 intent 进入 MainActivity, MainActivity 解析 intent 并传递给 playGroundFragment，同样 playGroundFragment 也会先解析 intent, 如果为快速开始则直接读取 SharePreference 数据并开始匹配

6.6 主题切换

用户切换主题后会修改 SharePreference 中记录，每次启动 Applacation 时提前设置主题

```
Casssifa(loacl)
@Override
//APP启动时
public void onCreate() {
    UserSharedPreferences.initInstance( application: this);
    theme = UserSharedPreferences.getInstance().getTheme();
    if (theme.equals("red")) setTheme(R.style.RedTheme);
    else if (theme.equals("blue")) setTheme(R.style.BlueTheme);

    super.onCreate();
    mApp = this;
    //初始化数据
    token = UserSharedPreferences.getInstance().getToken();
    user = UserSharedPreferences.getInstance().getUser();
}
```

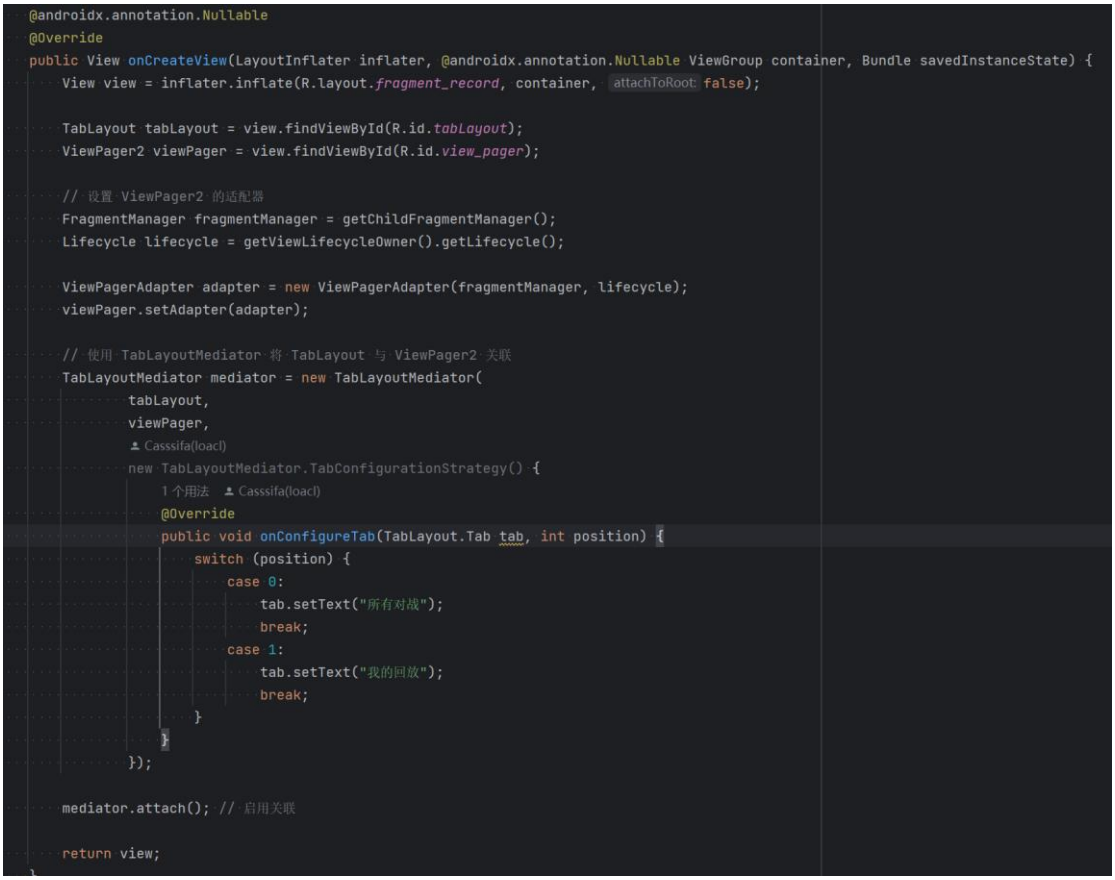
图 六-5 初始化主题

6.7 ChatGPT 调用

APP 调用 backendcloud 的接口，backendcloud 再通过 RestTemplate 调用 SensitiveWord 接口，SensitiveWord 通过 https 请求实现对 ChatGPT 的调用。Token 是从 OpenAI 购买的

6.8 滑动切换页面

集成 TabView 和 ViewPage2 库，实现效果。这里是掉了谷歌提供的库



```
@androidx.annotation.Nullable
@Override
public View onCreateView(LayoutInflater inflater, @androidx.annotation.Nullable ViewGroup container, Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_record, container,  attachToRoot: false);

    TabLayout tabLayout = view.findViewById(R.id.tabLayout);
    ViewPager2 viewPager = view.findViewById(R.id.view_pager);

    // 设置 ViewPager2 的适配器
    FragmentManager fragmentManager = getChildFragmentManager();
    Lifecycle lifecycle = getViewLifecycleOwner().getLifecycle();

    ViewPagerAdapter adapter = new ViewPagerAdapter(fragmentManager, lifecycle);
    viewPager.setAdapter(adapter);

    // 使用 TabLayoutMediator 将 TabLayout 与 ViewPager2 关联
    TabLayoutMediator mediator = new TabLayoutMediator(
            tabLayout,
            viewPager,
            Casssifa(loacl)
            new TabLayoutMediator.TabConfigurationStrategy() {
                1 个用法  Casssifa(loacl)
                @Override
                public void onConfigureTab(TabLayout.Tab tab, int position) {
                    switch (position) {
                        case 0:
                            tab.setText("所有对战");
                            break;
                        case 1:
                            tab.setText("我的回放");
                            break;
                    }
                }
            });

    mediator.attach(); // 启用关联

    return view;
}
```

图 六-6 集成 TabView 与 ViewPage2

6.9 ConvertView 实现资源复用

对与上百条条目只会创建等同于屏幕可见范围内数据的条目对象，大大减少内存开销

```
@SuppressLint("SetTextI18n")
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder viewHolder;
    if (convertView == null) {
        convertView = LayoutInflater.from(mContext).inflate(R.layout.item_rank, root: null);
        viewHolder = new ViewHolder();

        viewHolder.userRank = convertView.findViewById(R.id.rankItemUserRank);
        viewHolder.userImage = convertView.findViewById(R.id.rankItemUserImage);
        viewHolder.userName = convertView.findViewById(R.id.rankItemUserName);
        viewHolder.userRating = convertView.findViewById(R.id.rankItemUserRating);
        convertView.setTag(viewHolder);
    } else {
        viewHolder = (ViewHolder) convertView.getTag();
    }
    User user = userRankList.get(position);
    viewHolder.userRank.setText(user.getPassword());
    viewHolder.userName.setText(user.getUsername());
    viewHolder.userRating.setText(user.getRating().toString());
    Constant.setHttpImg(viewHolder.userImage, user.getPhoto(), mContext);
    return convertView;

}
```

```
3 个用法  Casssifa(loacl)
public final class ViewHolder {
    2 个用法
    public TextView userRank;
    2 个用法
    public ImageView userImage;
    2 个用法
    public TextView userName;
    2 个用法
    public TextView userRating;
}
```

图 六-7 集成 ConvertView

6.10 地图生成

为了防止玩家作弊，地图生成及操作合法性的检验均在后端进行，每轮移动后后端会将结果通过 WebSocket 链接回传给两位玩家。GameMap 类初始化时会随机生成一个中心对称地图，并 DFS 判断两名玩家是否联通,直至生成联通地图向前端返回地图结果

地图生成流程图

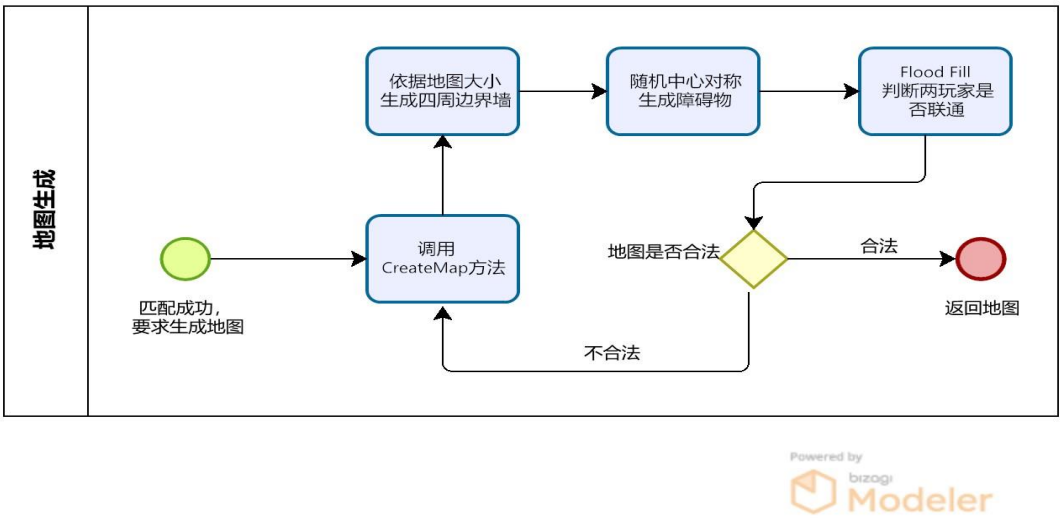


图 六-8 地图生成流程示意图

6.11 玩家匹配

Backend 会将玩家 id、AI id、玩家天梯分传给 MatchingSystem,系统会将玩家加入匹配池并每秒尝试匹配一次所有玩家，每个玩家有可最大可容忍天梯分差值，随时间推移最大可容忍差值会逐步扩大。匹配过程会优先匹配加入早的玩家

MatchingSystem 暴露的接口：

```
0 个用法
@PostMapping("/player/add/")
public String addPlayer(@RequestParam MultiValueMap<String,String> data){
    Integer userId=Integer.parseInt(Objects.requireNonNull(data.getFirst( k: "user_id")));
    Integer rating=Integer.parseInt(Objects.requireNonNull(data.getFirst( k: "rating")));
    Integer botId=Integer.parseInt(Objects.requireNonNull(data.getFirst( k: "bot_id")));
    return matchingService.addPlayer(userId,rating,botId);
}

0 个用法
@PostMapping("/player/remove/")
public String removePlayer(@RequestParam MultiValueMap<String,String> data){
    Integer userId=Integer.parseInt(Objects.requireNonNull(data.getFirst( k: "user_id")));
    Integer rating=Integer.parseInt(Objects.requireNonNull(data.getFirst( k: "rating")));
    return matchingService.removePlayer(userId,rating);
}
```

图 六-9 匹配接口

用户数据流图：

顶层图

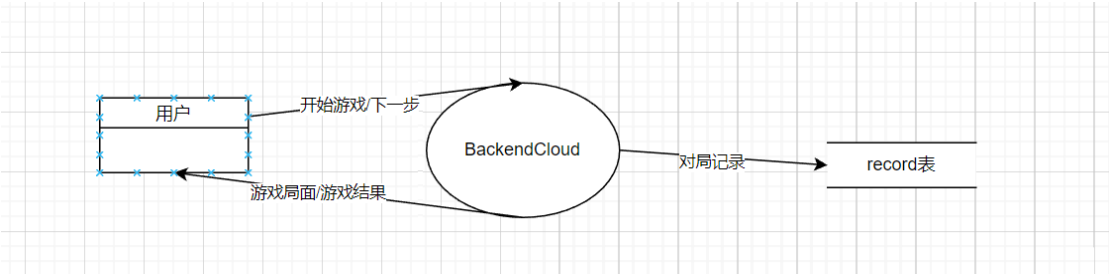


图 六-10 数据流图-顶层图

0 层图

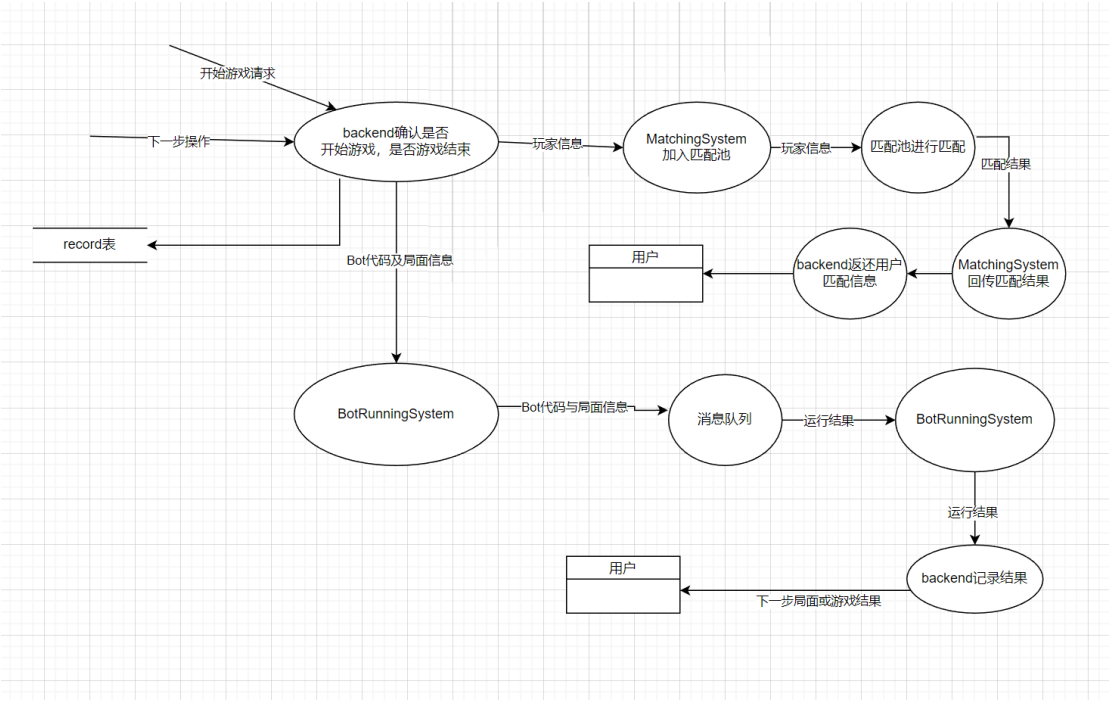


图 六-11 数据流图-0 层图

七、文件结构与用途

7.1 文件结构

- 代码部分

King of Bots 源代码文件结构

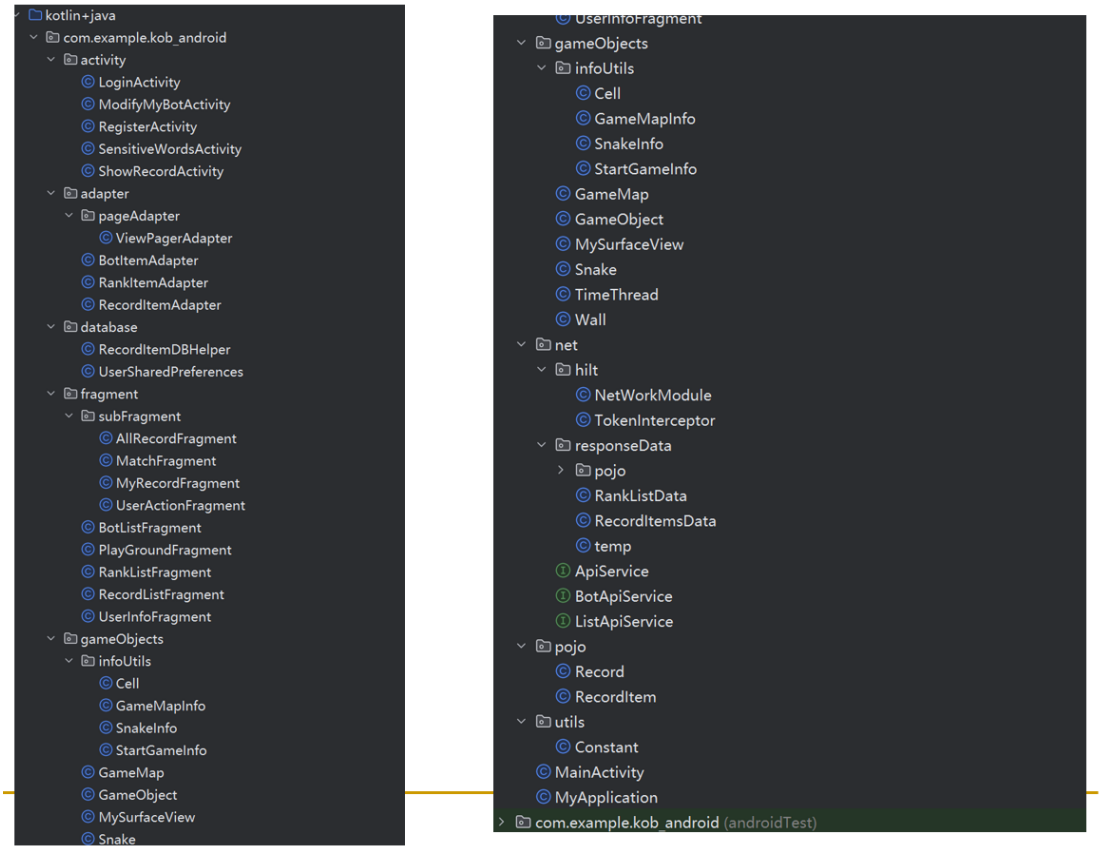


图 七-1 代码文件结构

- 资源部分

King of Bots 资源与配置文件文件结构

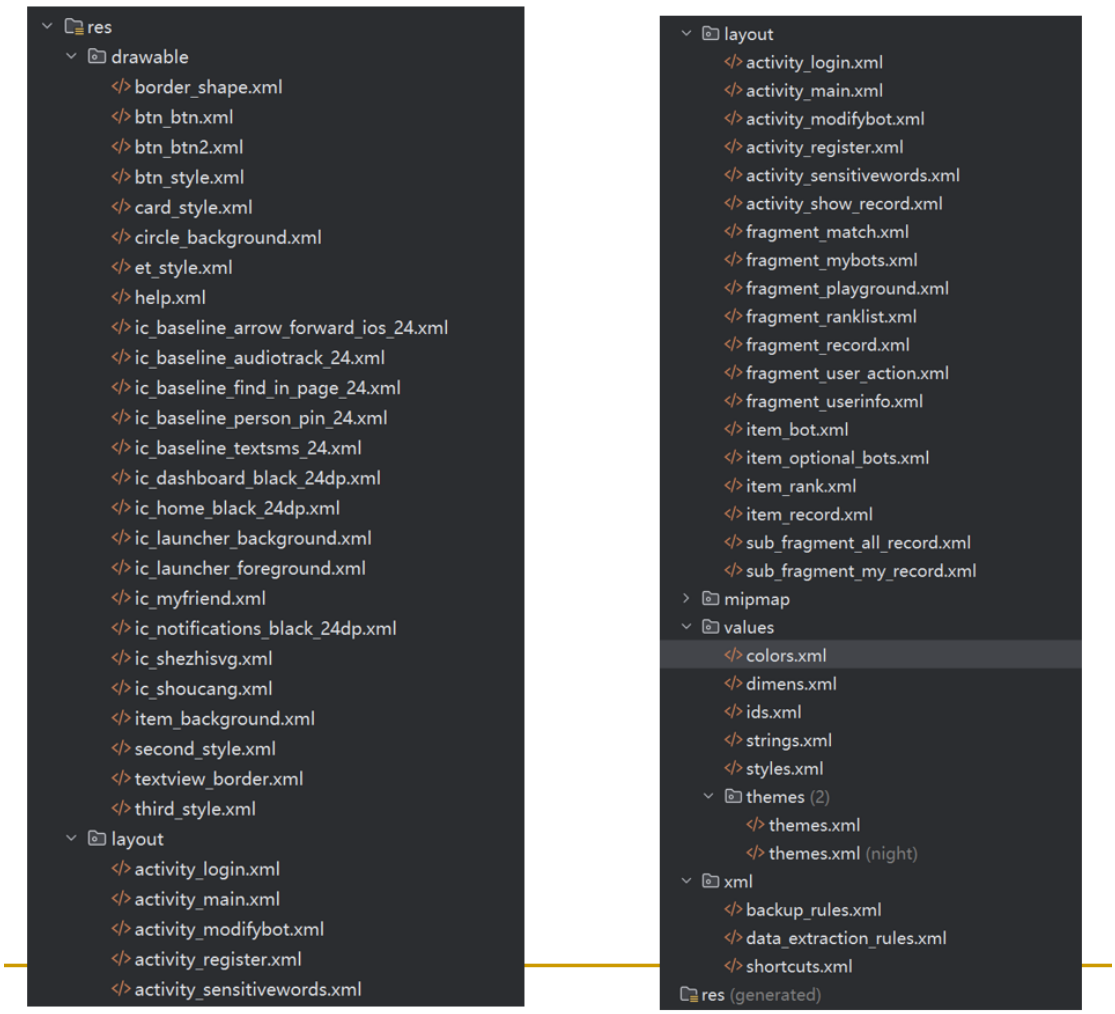


图 七-2 资源文件文件结构

7.2 文件用途说明

- 代码部分

包名称	子包名	文件名	说明
kob_android.Activity		LoginActivity	登录页面
		MainActivity	主页面，几乎所有 Fragment 挂靠
		ModifyMyBotActivity	修改 Bot 的界面
		RegisterActivity	注册界面

		SenstiveWordActivity	敏感词屏蔽界面
		ShowRecordActivity	播放回放页面
kob_android.adapter	pageAdapter	ViewPagerAdapter	Page 适配器
		BotItemAdapter	Bot 条目适配器
		RankItemAdapter	排行条目适配器
		RecordItemAdapter	回放条目适配器
kob_android.database		RecordItemDBHelper	封装 SQLite
		UserSharedPreferences	封装 SharedPreferences
kob_android.Fragment	subFragment	AllRecordFragment	展示所有回放的 Fragment
		MatchFragment	匹配组件的 Fragment
		MyRecordFragment	展示我的回放 Fragment
		UserActionFragment	用户移动蛇的 Fragment
		BotListFragment	Bot 列表的 Fragment
		PlayGroundFragment	游戏页面的 Fragment
		RankListFragment	排行页面的 Fragment
		RecordListFragment	回放列表的 Fragment
		UserInfoFragment	用户信息的 Fragment
kob_android.gameObject	infoUtils	Cell	蛇身体单元
		GameMapInfo	当前对局信息
		SnakeInfo	蛇的配置信息
		StartGameInfo	开启游戏时初始数据
		GameMap	游戏地图配件
		GameObject	游戏基类，所有游戏类继承此类
		MySurfaceView	游戏界面类，展示游戏动画
		Snake	蛇类，执行蛇行为的发出者
		TimeThread	线程类，配合界面类控制所有游戏类
		Wall	墙类
kob_android.net	hilt	NetWorkModule	网络配置文件
		TokenInterceptor	为所有请求添加 token
	responseData	RankListData	排行条目数据格式适配器
		RecordItemsData	回放条目适配器
		ApiService	一系列接口 api
		BotApiService	一系列接口 api
		ListApiService	一系列接口 api
kob_android.pojo		Bot	Ai 类
		Record	回放类
		RecordItem	回放条目适配器
		User	用户类
kob_android.utils		Constant	工具类

kob_android		MyApplication	Application,APP 初始化工具

- 资源部分

资源过多，这里只详细介绍 layout 目录下的

资源目录名	作用
drawable	可绘制组件，主要用来实现登录页面效果
layout	布局文件
mipmap	图片资源，存启动图标与快速开始图标
values	字符串、颜色、值资源，供其它地方引用
xml	快速启动配置文件

Layout 资源资源目录:

资源名称	功能
activity_login	登录页面资源文件
activity_main	主页面资源文件
activity_modifybot	编辑 Bot 页面资源文件
activity_register	注册页面资源文件
activity_sensitivewords	敏感词页面资源文件
activity_show_rocord	回放页面资源文件
fragment_match	匹配页面组件资源文件
fragment_mybots	我的 Bot 页面组件资源文件
fragment_playground	游戏页面组件资源文件
fragment_ranklist	排行页面组件资源文件
fragment_record	回放页面组件资源文件
fragment_userinfo	用户信息组件资源文件
fragment_user_action	用户移动组件资源文件
item_bot	Bot 条目资源文件
item_rank	排行条目资源文件
item_record	回放条目资源文件
sub_fragment_all_record	所有回放条目资源文件
sub_fragment_my_record	我的回放条目资源文件

八、 创新点及开发时所用与本课程相关技术

8.1 创新点

1. 前后端分离，几乎所有数据都走网络请求，数据通过接口请求，实现了不同设备数据同步，网页端与移动端一同游戏（见 6.3）
2. 利用原生 JAVA 模拟实现了游戏引擎的功能（见 6.1）
3. 通过 Intent 传递及多重逻辑判断实现，快速开始游戏功能（见 6.5）
4. 调用大语言模型实现需求（见 6.7）

8.2 课程相关技术点

1. 各种组件之间嵌套实现多样的界面效果
   - 几乎所有地方都是 Activity 套 Fragment
   - Activity 套 Fragment 再套 TabView 再套 ViewPage2 实现回放功能的界面滑动切换的效果
2. 利用 Application 实现主题切换
3. 利用 OKHttp、Hilt 封装了大量网络接口，屏蔽数据不一致的问题，方便对接后端。集成 WebScoket 服务
4. 封装大量 Adapter 适应多种数据条目
5. 封装 SQLite 与 SharePreference
6. 大量的多线程、回调、安卓组件生命周期函数的运用

7. 对于素材资源的运用,抽离了多种 Drawable 组件实现资源复用
8. 使用 MSurfaceView 实现贪吃蛇动画效果
9. 使用 ConvertView 实现资源复用
10. Activity 启动模式的运用，比如登录页面进主页面时会销毁 Activity 栈再启动