中山大学数据科学与计算机学院本科生实验报告（2017年秋季学期）

> 课程名称：移动应用开发
>
> 任课教师：刘宁

- 年级：2015级
- 学号：15331436
- 专业（方向）：软件工程（计算机应用方向）
- 姓名：**朱楚**
- 电话：18675101884
- Email：me@zyuco.com
- 开始日期：2017年10月25日
- 完成日期：2017年10月26日

实验题目

安卓期中项目，三国武将电子词典。

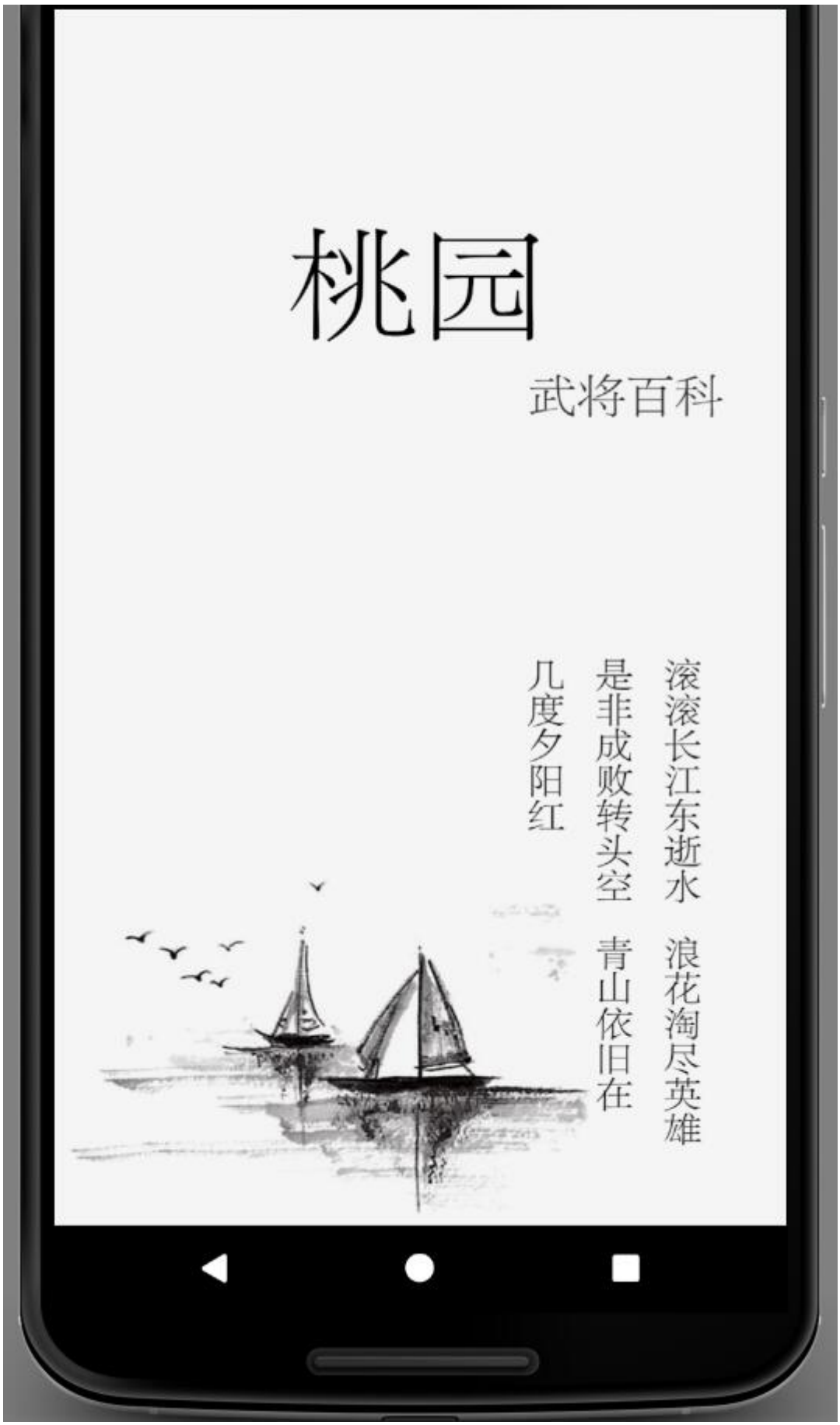
负责部分

- 初始应用数据爬取与处理
- 产品功能设计
- 产品美工
- 整体架构设计实现
- 数据库表格设计、接口实现
- 主页功能
- 关于页功能

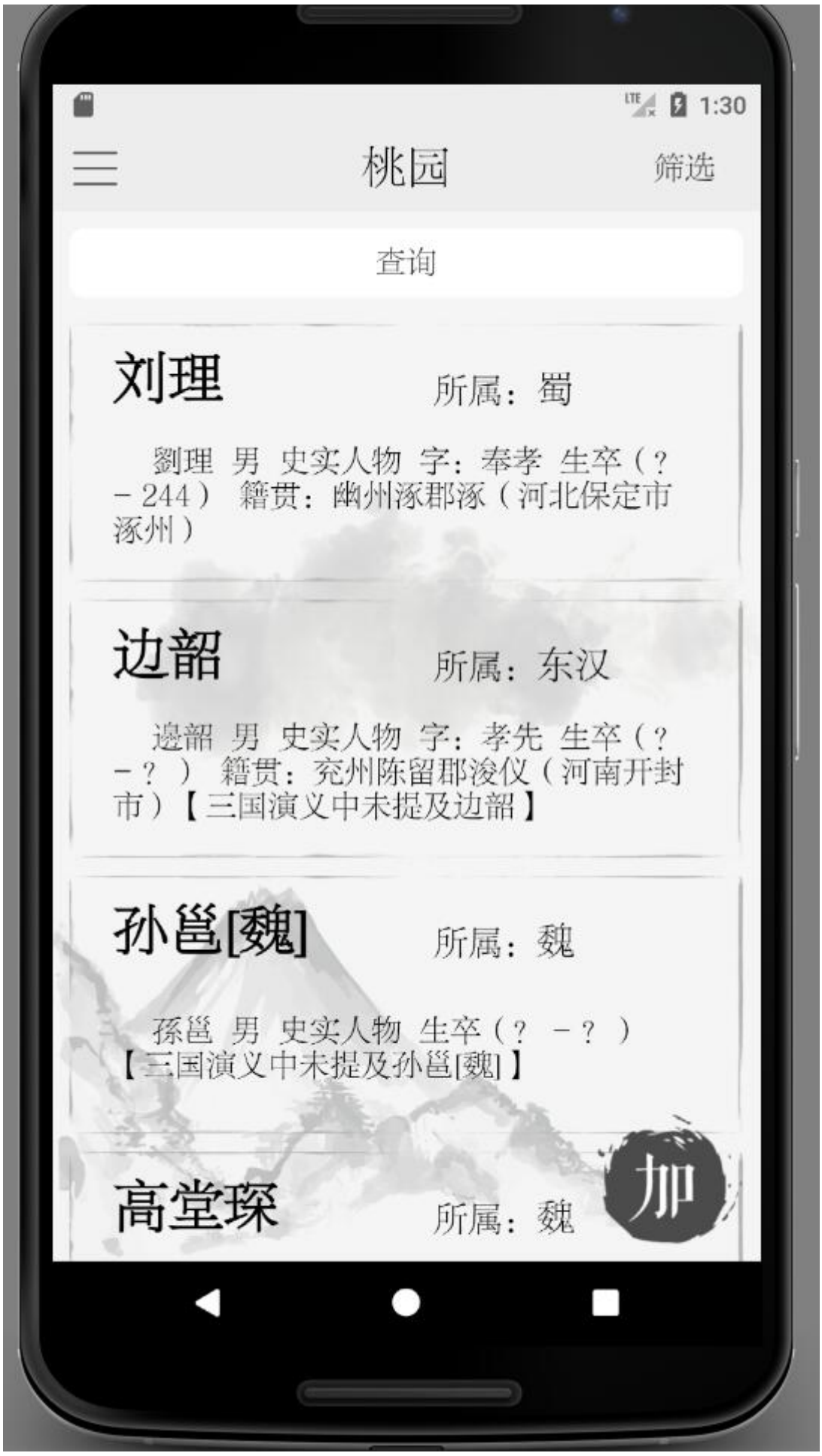
实验结果

实验截图

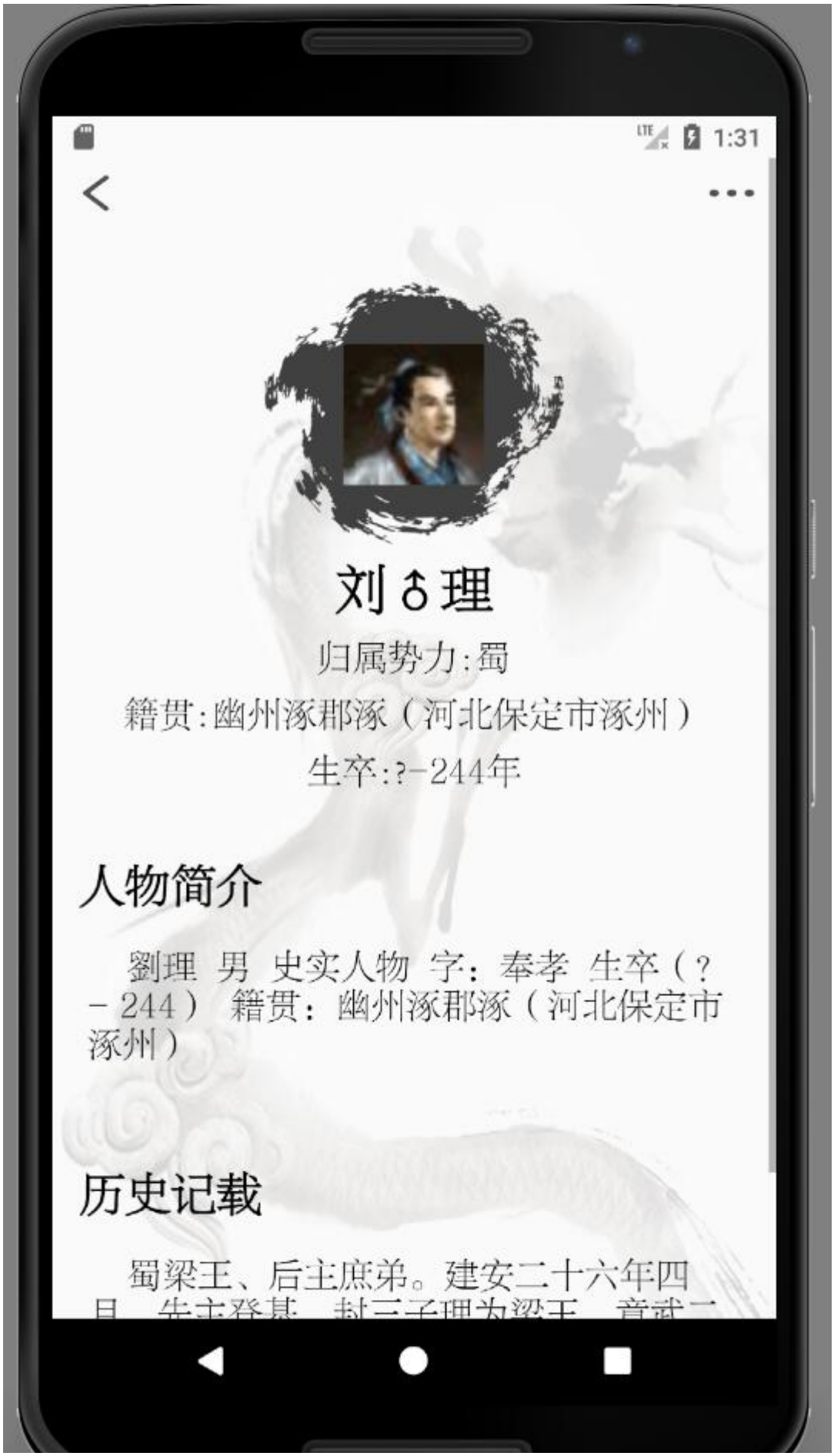
启动Splash页：



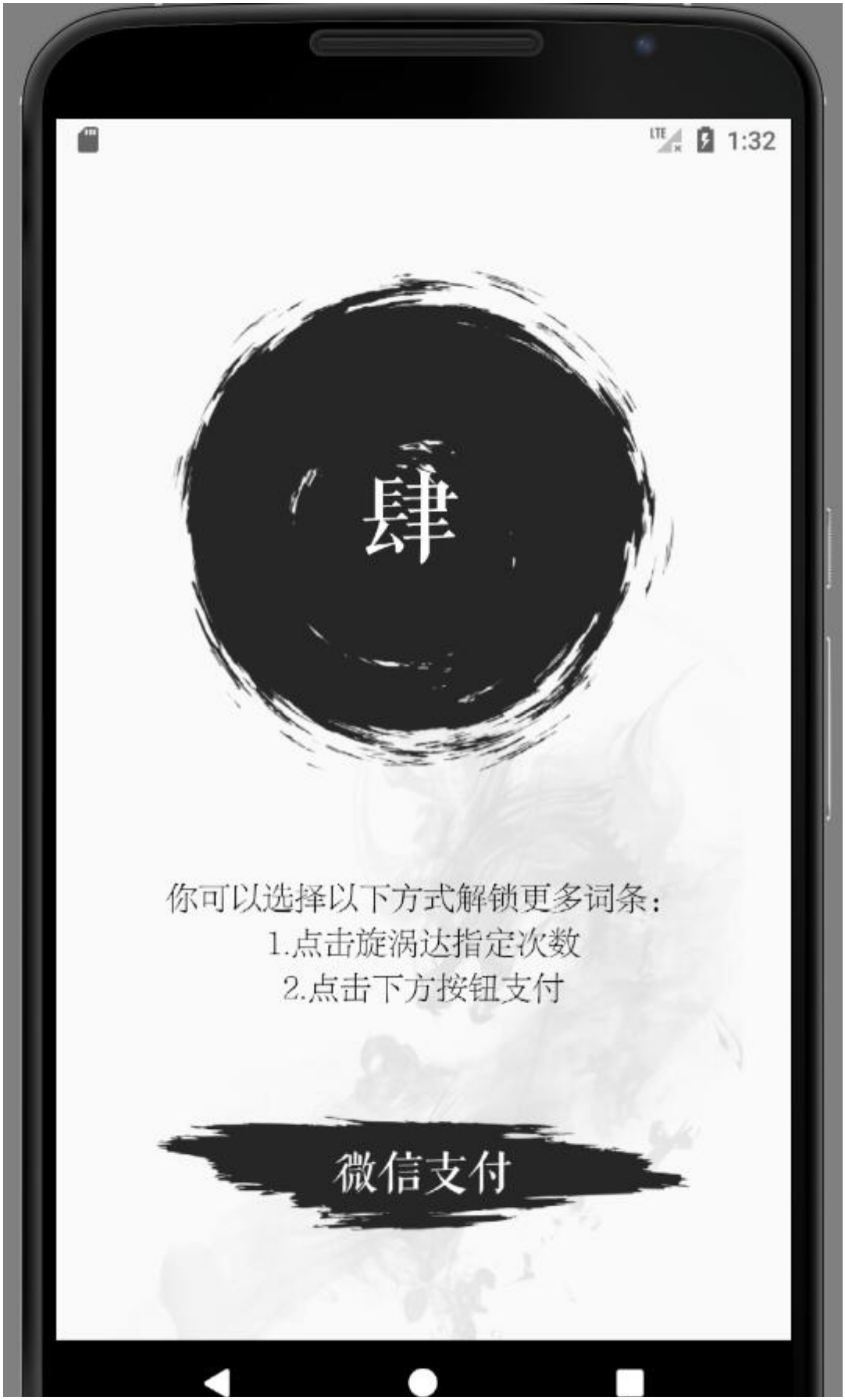
主页：


1:30
桃园
筛选
查询
刘理
所属：蜀
劉理 男 史实人物 字：奉孝 生卒（？－244） 籍贯：幽州涿郡涿（河北保定市涿州）
边韶
所属：东汉
邊韶 男 史实人物 字：孝先 生卒（？－？） 籍贯：兖州陈留郡浚仪（河南开封市）【三国演义中未提及边韶】
孙邕[魏]
所属：魏
孫邕 男 史实人物 生卒（？－？）【三国演义中未提及孙邕[魏]】
高堂琛
所属：魏
加

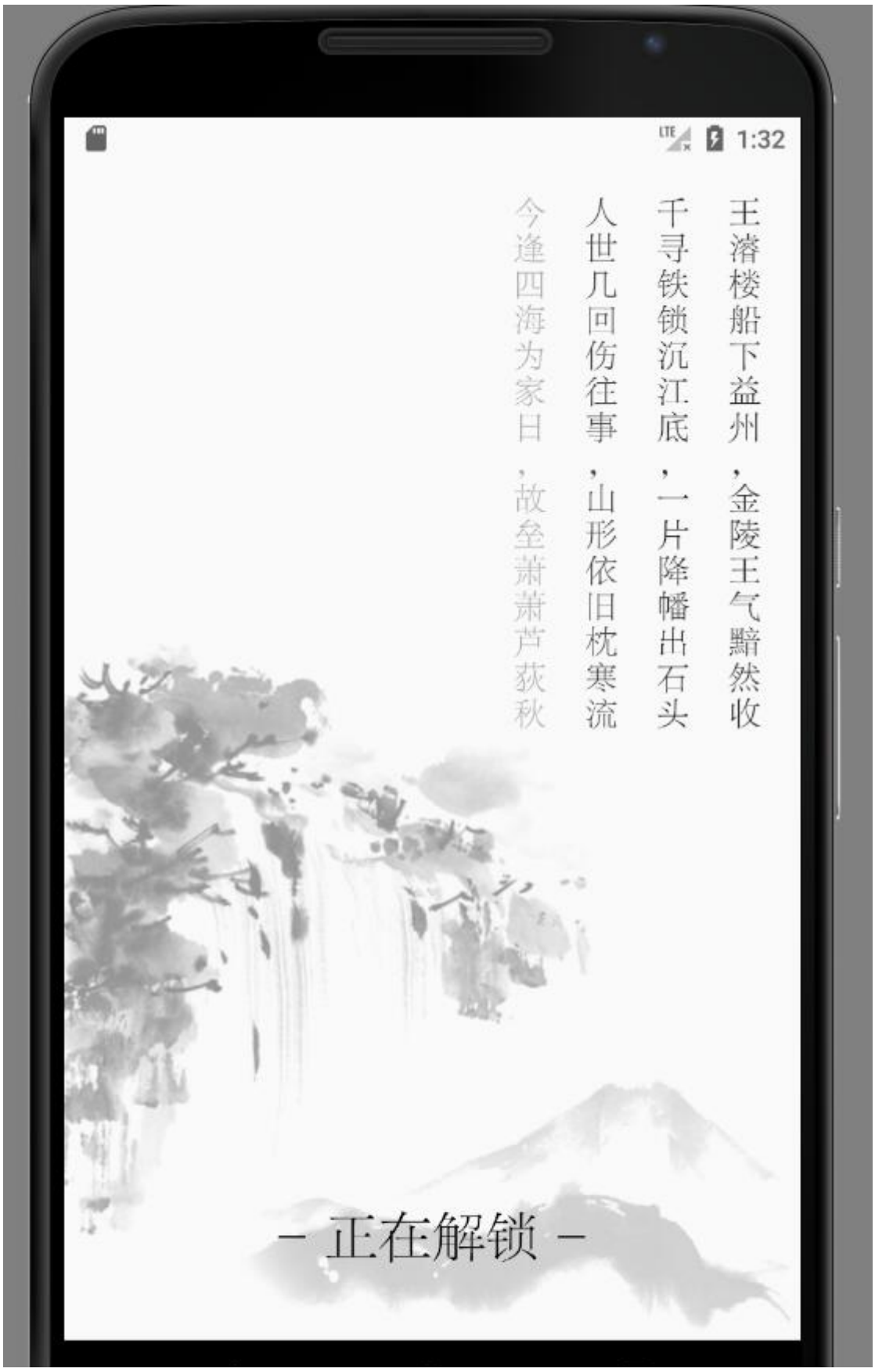
点击进入详情:

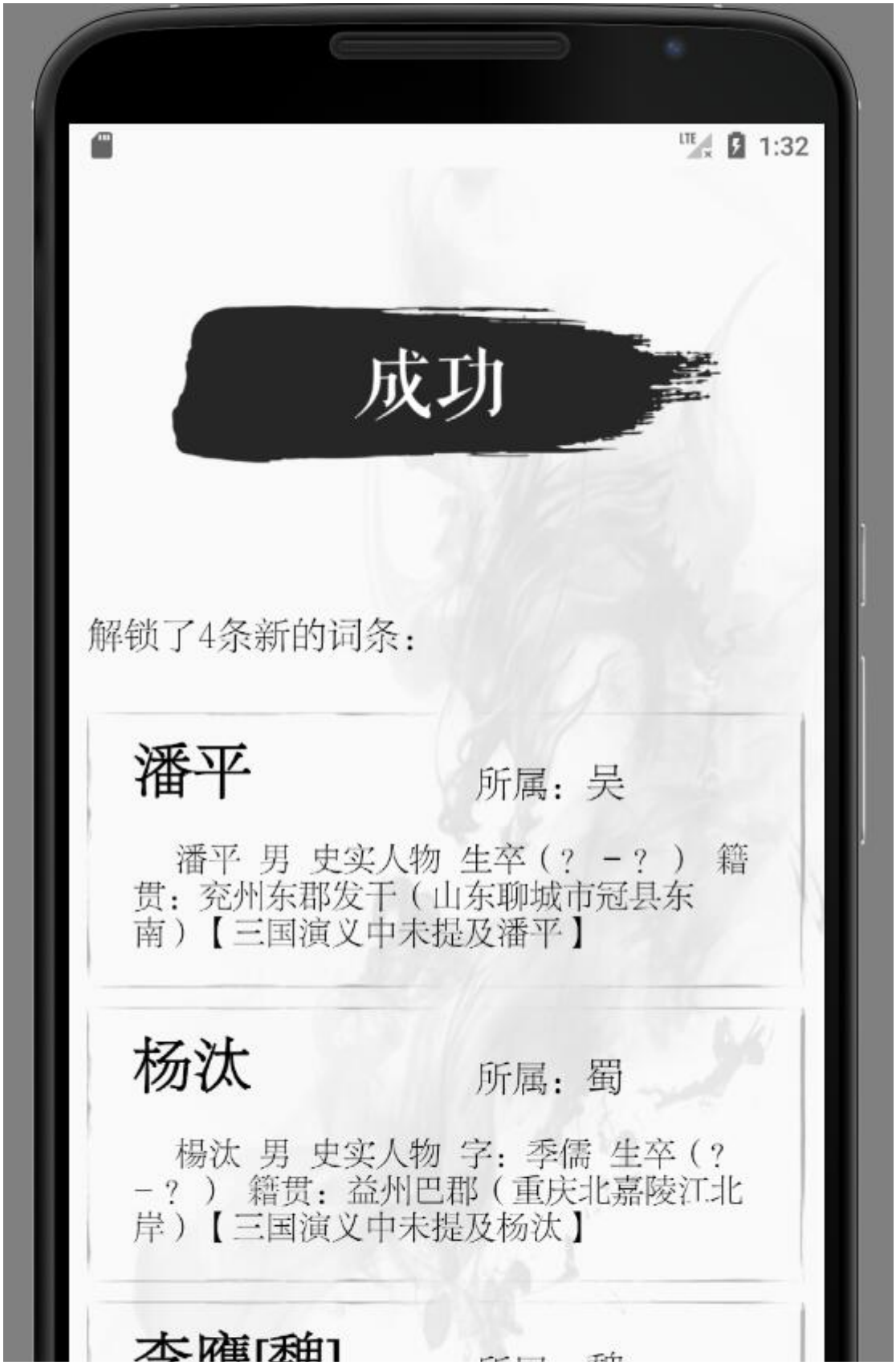

1:31
刘𠂤理
归属势力:蜀
籍贯:幽州涿郡涿（河北保定市涿州）
生卒:?-244年
人物简介
劉理 男 史实人物 字：奉孝 生卒（？－244） 籍贯：幽州涿郡涿（河北保定市涿州）
历史记载
蜀梁王、后主庶弟。建安二十六年四

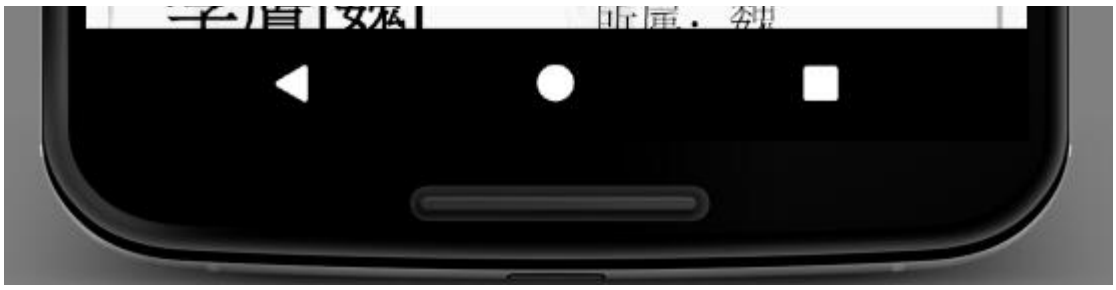
抽取更多:



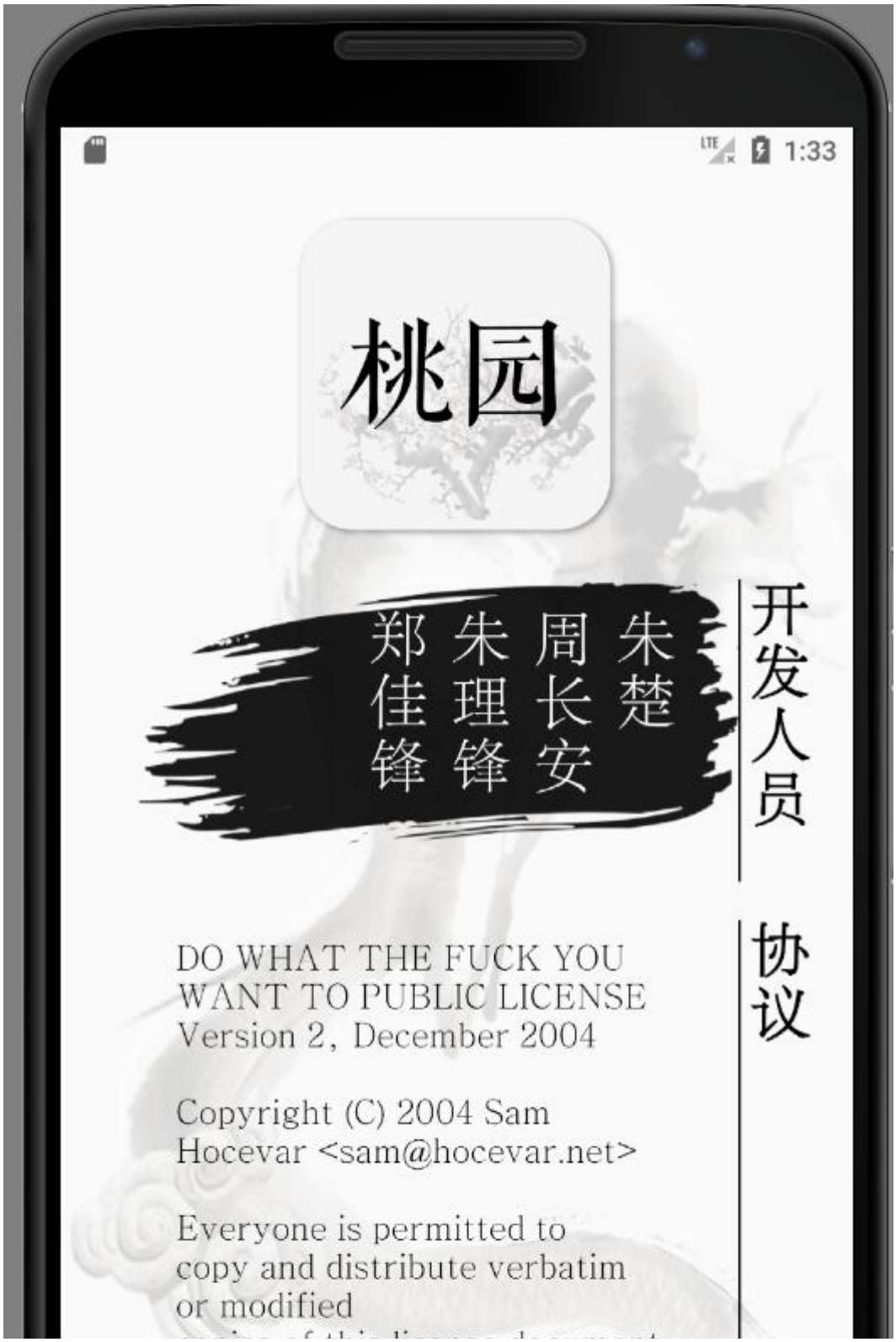
解锁成功：

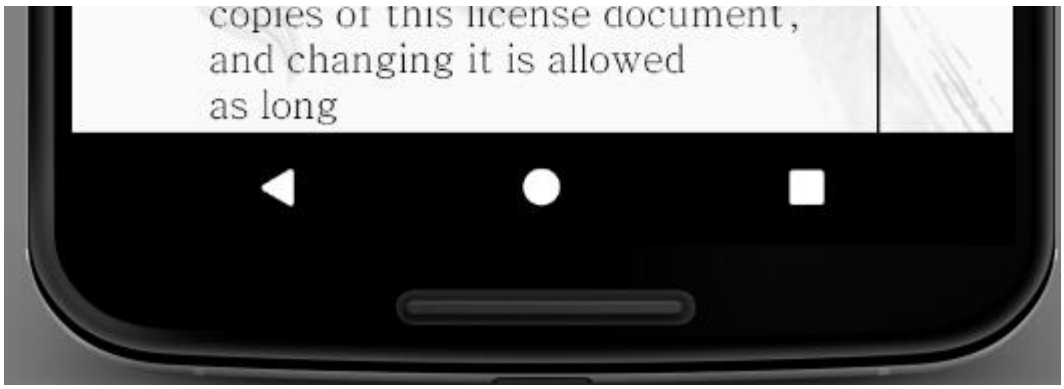

1:32
王濬楼船下益州，金陵王气黯然收
千寻铁锁沉江底，一片降幡出石头
人世几回伤往事，山形依旧枕寒流
今逢四海为家日，故垒萧萧芦荻秋
- 正在解锁 -


1:32
成功
解锁了4条新的词条：
潘平
所属：吴
潘平 男 史实人物 生卒（？ －？） 籍贯：兖州东郡发干（山东聊城市冠县东南）【三国演义中未提及潘平】
杨汰
所属：蜀
楊汰 男 史实人物 字：季儒 生卒（？ －？ ） 籍贯：益州巴郡（重庆北嘉陵江北岸）【三国演义中未提及杨汰】



关于页面：


1:33
桃园
开发人员
朱楚
周长安
朱锋
郑佳锋
协议
DO WHAT THE FUCK YOU
WANT TO PUBLIC LICENSE
Version 2, December 2004
Copyright (C) 2004 Sam
Hocevar <sam@hocevar.net>
Everyone is permitted to
copy and distribute verbatim
or modified



实验步骤以及关键代码

- 数据爬取

使用Node.js并发爬取e3ol的数据，然后处理成json作为安卓项目的raw资源。

```
(async () => {
  const queue = new Queue(10);
  queue.push(...Array(GeneralNum)
    .fill(null)
    .map((_, idx) => idx + 1)
    .map(id => {
      const fn = fetch.bind(null, id);
      fn.id = id;
      return fn;
    })
  );
  let count = 0;
  queue.on('error', (err, job) => {
    if (!err.message.match(/timeout/)) {
      console.error(err);
    }
    queue.push(job); // retry
  });
  queue.on('complete', job => {
    console.log(`${job.id} checked. [${++count}/${GeneralNum}]`);
    if (count % 1000 === 0) {
      fs.writeJson('./result/all.json', result.sort((o1, o2) => o1.id > o2.id ? 1 : -1), {
spaces: 2 }); // save per stage
    }
  });

  await queue.wait();
  console.log(`all done, ${result.length} in total`);
  await fs.writeJson('./result/all.json', result.sort((o1, o2) => o1.id > o2.id ? 1 : -1), {
spaces: 2 });
})()
  .catch(console.error);
```

- 引入数据库：

使用DbHelper封装数据库的创建等基本操作，再向上封装了一层DbWriter与DbReader提供武将的CRUD操作。

```
class DbHelper extends SQLiteOpenHelper {
    private static DbHelper instance;

    private static final String TAG = "PeachGarden.DbHelper";

    private static final int DB_VERSION = 1;
    public static final String DB_NAME = "PeachGarden.db";

    // table name
    static final String TABLE_CHARACTER = "characters";
    static final String TABLE_OWN = "own";

    // create tables
    private static final String CREATE_TABLE_CHARACTERS = "" +
        "CREATE TABLE IF NOT EXISTS " + TABLE_CHARACTER +
        " ( _id INTEGER PRIMARY KEY AUTOINCREMENT, " + Character.COL_NAME + " TEXT NOT NULL,
" + Character.COL_PINYIN + " TEXT, " + Character.COL_AVATAR + " TEXT, " +
Character.COL_ABSTRACT + " TEXT, " + Character.COL_DESCRIPTION + " TEXT, " +
Character.COL_GENDER + " INT, " + Character.COL_FROM + " INT, " + Character.COL_TO + " INT,
" + Character.COL_ORIGIN + " TEXT, " + Character.COL_BELONG + " TEXT, " +
Character.COL_BELONG_ID + " INT );";

    private static final String CREATE_TABLE_OWN = "" +
        "CREATE TABLE IF NOT EXISTS " + TABLE_OWN +
        " ( _id INTEGER PRIMARY KEY AUTOINCREMENT," +
        "   character_id INTEGER," +
        "   FOREIGN KEY(character_id) REFERENCES " + TABLE_CHARACTER + "(" + Character._ID +
")" +
        ");";

    private static final int INIT_OWN_SIZE = 10;

    private Context context;

    @Override
    public void onCreate(SQLiteDatabase db) {
        Log.i(TAG, "db created");
        db.execSQL(CREATE_TABLE_CHARACTERS);
        db.execSQL(CREATE_TABLE_OWN);

        Character[] data = readInitCharacters();
        insertInitialData(db, data);
        generateInitOwn(db, data, INIT_OWN_SIZE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int i, int i1) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_CHARACTER);
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_OWN);
        onCreate(db);
    }

    private DbHelper(Context context) {
```

```
48          super(context, DB_NAME, null, DB_VERSION);
49          this.context = context;
50      }
51
52      private void insertInitialData(SQLiteDatabase db, Character[] data) {
53          db.beginTransaction();
54          for (Character ch : data) {
55              ch._id = db.insert(TABLE_CHARACTER, null, ch.toContentValues());
56          }
57          db.setTransactionSuccessful();
58          db.endTransaction();
59      }
60
61      private void generateInitOwn(SQLiteDatabase db, Character[] data, int count) {
62          List<Character> list = Arrays.asList(data);
63
64          int toGen = Math.min(count, data.length);
65          Collections.shuffle(list);
66          List<Character> gen = list.subList(0, toGen);
67
68          db.beginTransaction();
69          for (Character ch : gen) {
70              ContentValues values = new ContentValues();
71              values.put("character_id", ch._id);
72              db.insert(TABLE_OWN, null, values);
73          }
74          db.setTransactionSuccessful();
75          db.endTransaction();
76      }
77
78      private Character[] readInitCharacters() {
79          try (InputStream stream = context.getResources().openRawResource(R.raw.characters))
    {
80              BufferedReader reader = new BufferedReader(new InputStreamReader(stream,
    "UTF8"));
81              Gson gson = new Gson();
82              return gson.fromJson(reader, Character[].class);
83          } catch (IOException e) {
84              Log.e(TAG, "readInitCharacters: ", e);
85              return new Character[]{};
86          }
87      }
88
89      public static synchronized DbHelper getInstance(Context context) {
90          if (instance == null) {
91              instance = new DbHelper(context.getApplicationContext());
92          }
93          return instance;
94      }
95  }
```

```java
/**
 * DbWriter is an interface to manipulate (write-only) Database
 */
public class DbWriter {
    private static DbWriter instance;

    private SQLiteDatabase db;

    private DbWriter(Context context) {
        db = DbHelper.getInstance(context).getWritableDatabase();
    }

    public void addCharacters2Own(List<Character> list) {
        db.beginTransaction();
        for (Character ch : list) {
            ContentValues values = new ContentValues();
            values.put("character_id", ch._id);
            db.insert(DbHelper.TABLE_OWN, null, values);
        }
        db.setTransactionSuccessful();
        db.endTransaction();
    }
    public void addCharacters(List<Character> list) {
        db.beginTransaction();
        for (Character ch : list) {
            ch._id = db.insert(TABLE_CHARACTER, null, ch.toContentValues());
        }
        db.setTransactionSuccessful();
        db.endTransaction();
    }

    public void deleteOwnedCharacter(Character ch) {
        db.delete(DbHelper.TABLE_OWN, Character._ID + " = ?", new String[]
{String.valueOf(ch._id)});
    }
    public void deleteCharacter(Character ch) {
        db.delete(DbHelper.TABLE_CHARACTER , Character._ID + " = ?", new String[]
{String.valueOf(ch._id)});
    }

    public static synchronized DbWriter getInstance(Context context) {
        if (instance == null) {
            instance = new DbWriter(context);
        }
        return instance;
    }
}
```

实验遇到困难以及解决思路

- 爬取网站数据的时候并发开太高被ban了IP，后来尽量降低了并发，并通过约20个代理节点进行爬取。

- Android虚拟机中的SQLite信息不好观察：后来通过`debugCompile ';com.amitshekhar.android:debug-db:1.0.1';`使得在debug期在8080端口暴露一个DB的网页操作端，并通过`adb forward tcp:8080 tcp:8080`将对应端口映射到开发机上通过浏览器打开即可方便地观察数据。

实验思考及感想

比起之前的小作业来说，期中项目需要在开发之前便计划好一个大致的架构并严格按照这个架构开发，否则项目代码会越写越难看。