软件设计文档

一、项目概述

本项目为安卓端 2048 游戏项目，包括常规 2048 游戏与倒计时模式，并架构了简单网络数据库与后台接口，用于联网保存排行榜数据。

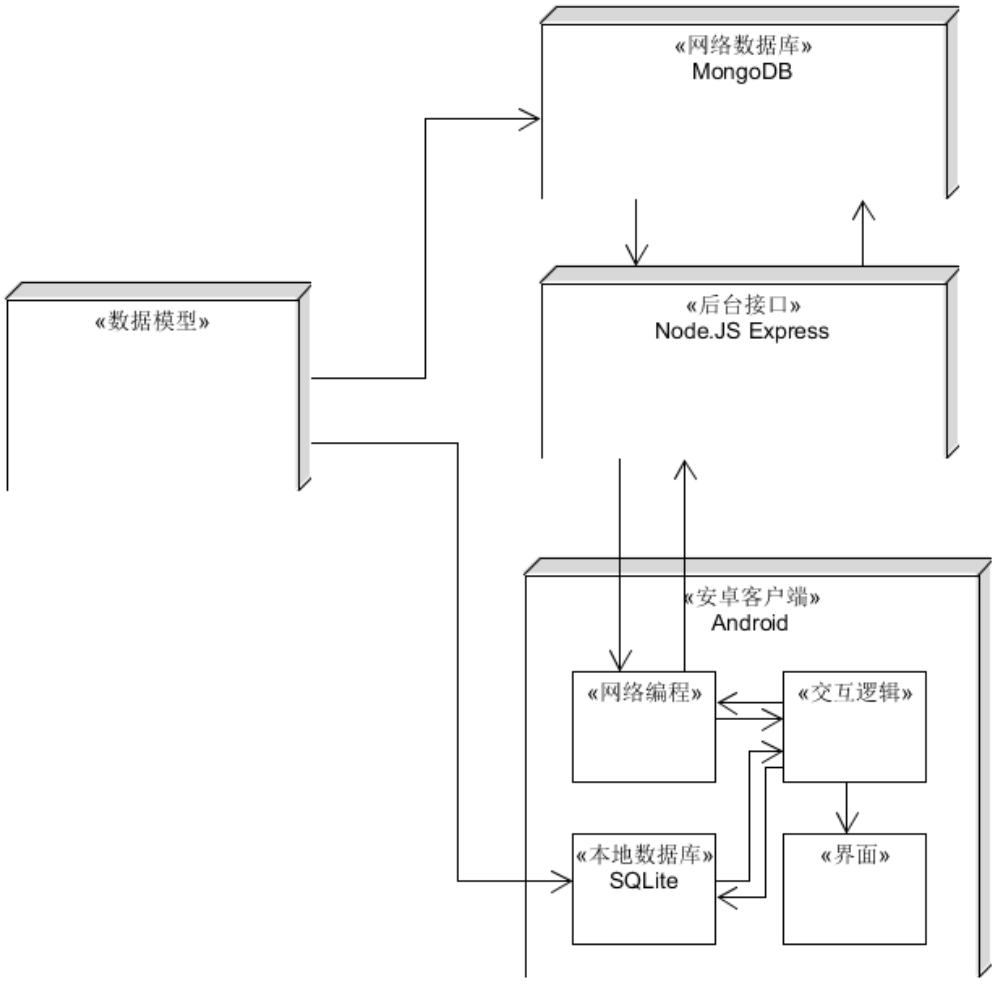
二、技术选型

游戏主体运行在安卓客户端，适用 SDK 版本 16~25，使用此平台基于代码编写硬件需求小，平台覆盖面广，代码可维护性与合作编程便捷性较高。使用到的安卓端技术包括：多线程用以处理计时与动画、SharedPreference 用以保存属性数据、SQLite 用以保存游戏数据、网络接口编程用以联网保存与获取用户排行榜数据。

网络后台使用 nodejs 搭建，nodejs 采用事件驱动，适合处理高并发的操作，对于多人游戏频繁访问服务器处理有较高的性能。采用的库包括 express，mongoose 等，能够明显减少工作量和提高开发成本。

网络数据库使用 MongoDB，此为一个 NoSQL 轻量级数据库，并且具有无模式、查询与搜索方式灵活等优点，在处理多 I/O 操作方面具有较好的可靠性。

三、架构设计



软件设计遵从 MVC 架构。

Model 数据模型，软件中数据来源主要有三个。安卓端本地数据，数据模型保存于类 My2048Data 中，包括如下成员：

```
private int id;
private int[] numbers;
private TimeUtil time;
private int step;
private String username;
private int score;
```

此数据模型定义了一次游戏的全部数据，并在本地分别针对普通模式最高分 top10、计时模式最高分 top10、正在进行的游戏数据进行保存。使用此种数据模型，将与 SQLite 实现完美对接。

安卓端本地属性，则使用 SharedPreference 保存，数据包括本地普通模式最高分、本地计时模式最高分、是否可以继续游戏、是否显示计时模式提示框等属性数据。

网络端数据简化为 username 与 score，排行榜仅展现这两项关键数据。存储与 MongoDB 中，保存为 json 格式。

View 视图，体现在安卓界面与交互中，软件共包括主页面、游戏页面、排行榜三个页面。

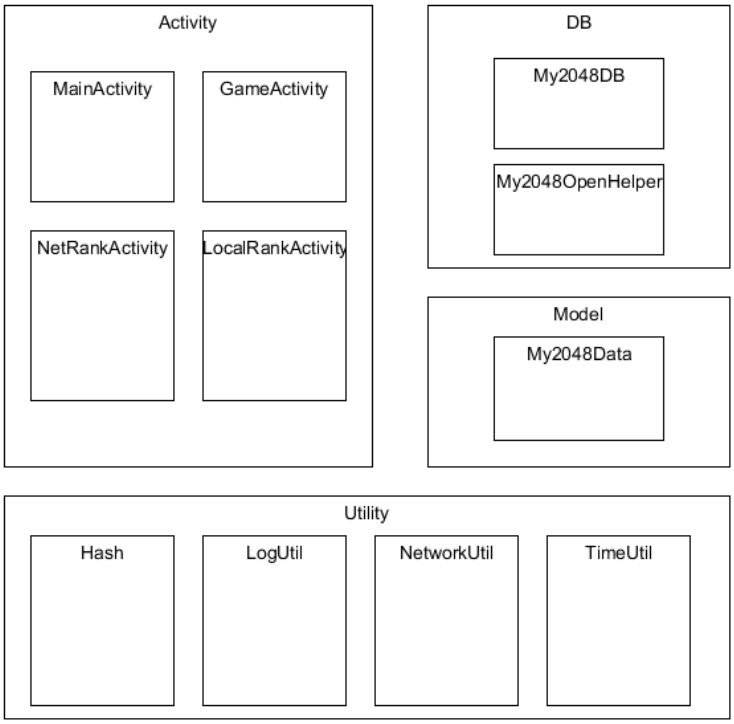
Controller 控制器，针对三种数据模型分别有三套对应获取与保存方式。安卓端本地数据保存于 SQLite 中，通过重用安卓内置类与方法，建立 My2048DB 与 My2048OpenHelper 类处理数据的增删改查操作，并与视图进行交互，以显示、更改数据。

SharedPreference 操作更为简单，使用安卓内置方法即可对数据进行操作。

对于网络端数据的获取与上传，在安卓中建立了 NetworkUtil 静态类，包括对 json 的编码解码、http 连接建立与数据获取、返回于安卓界面的交互等。网络端则使用 Express 编写接口，处理 http 请求，并对 MongoDB 中保存的数据进行修改操作。

四、模块划分

1. 安卓端



安卓端根据 java 类文件主要分为如下大模块：活动 Activity，数据库 DB，数据模型 Model，应用类 Utility。

其中活动即软件呈现的界面，包括主页面 MainActivity，游戏页面 GameActivity，网络排行榜页面 NetRankActivity 与本地排行榜 LocalRankActivity。

数据库用来处理 SQLite 数据存储与交互部分。包括 My2048DB 类，处理更新、插入删除等操作；My2048OpenHelper 类，处理数据库创建、版本控制等。

数据模型中 My2048Data 类存储了游戏进程的数据模型，保存 id、分数、步数、当前游戏状态等信息。

应用类中包含四个类：Hash 类包括一些静态常量与方法，用来转换如游戏方块资源名称、游戏方块分数与索引关系、游戏方块在屏幕上位置等信息，为了便于编程中此类转换。LogUtil 静态类管理了安卓编程中会用到的注释信息，能够允许某种级别以上的注释信息得到显示，通过此种控制方法，能够轻松对发布版程序规避经常有注释信息输出，减慢软件运行效率的情况进行控制。NetworkUtil 类为静态类，包含对网络数据的获取与上传。TimeUtil 定义了游戏中需要用到的计时系统，包括分、秒、toString()、正向计时、倒计时等实用字段与方法。

2. 网络后端

网络后台主要根据处理的功能分为两大块：接收数据和发送数据。

在接收数据方面，存在两种模式，一种接收用户计时模式下的成绩，一种是接收用户在普通模式下的成绩，两种模式都应该的采用 post 方法想后台传送数据。同时两种模式下，数据库只保存前十名里面的信息。

发送数据，采用的是 get 方法，同样存在两种模式，用于发送用户在不同的游戏模式下的排名信息。

五、软件设计技术

1. 面向对象程序设计

   软件安卓客户端使用 Java 语言编写，全部以类为主导。其中每个页面的创建都是对应活动类的实例化，并调用其 onCreate()函数实现的。游戏进程中的数据保存成为 My2048Data 类的一个实例，当游戏进程的分数改变、时间流逝等出现时，都针对改实例进行修改操作。当游戏主动或被动退出时，改实例将被保存在 SQLite 数据库中，继续游戏时，将从数据库取出，并重新构建对象实例。又比如 Hash.java、NetworkUtil.java 等静态实用类的应用，均使用面向对象的程序设计方式。通过面向对象的设计，更便于程序可维护与数据交互、数据转换等操作的进行。

2. 高度代码复用的设计

   软件为了编程效率、后续可维护性与协作编程简易性，高度复用代码。比如两种游戏模式对应的活动均为 GameActivity，由于两种模式的游戏内核代码基本一致，因此绝不应该重写两份核心代码。因此使用 Intent 为游戏活动传入游戏模式参数，代码中通过此参数实现两种模式的不同点，比如决定时钟正数还是倒数，代码如下：

```
if (gameType == MODE_NORMAL) {
    my2048Data.tickTime(1);
    scoreTimeTxt.setText(my2048Data.getTime().toString());
} else if(gameType == MODE_COUNTDOWN) {
    my2048Data.tickTime(-1);
    scoreTimeTxt.setText(my2048Data.getTime().toString());
    if(my2048Data.getTime().isNil()) {
        gameOver();
    }
}
```

   代码复用的设计还体现在 Hash.java 中，将游戏方块对应的资源位置、对应的分数与方块索引值保存成静态数组，方便游戏进程中不断调用。

3. 模块化设计

   软件中，所有与游戏操作无关的操作全部放入对应的数据库或应用类等模块中进行，一方面提高复用性，一方面维持代码简介，便于维护。如与网络编程相关的操作全部放入 NetworkUtil 类中，游戏主进程仅需调用其接口方法，便可轻松拉取或上传数据，使代码高度可维护。

4. 多线程编程

   为了实现方块的动画效果，使用多线程编程模式。主要体现在 timer 与 handler 的使用。比如动画效果为耗时操作，要保证动画操作结束后再进行新方块的创建、游戏是否结束的判断等，例移动方块方法 moveBlock()中：

```
final Handler handler = handleMessage(msg) -> {
        if (msg.what == 0x789) {
            setRandomNewPiece();
        }
        if (msg.what == 0xabc) {
            LogUtil.d("Over", "Ready to Check");
            checkGameOver();
        }
};
new Timer().schedule(() -> {
        handler.sendEmptyMessage(0x789);
}, 100 * maxTimes);
new Timer().schedule(() -> {
        handler.sendEmptyMessage(0xabc);
}, 100 * maxTimes + 150 );
```

又比如游戏视图中的时钟时间不断改变，其计时操作在副线程中进行，而只有主线程可以执行 UI 操作，因此也需要 handler 与主线程进行交互。时钟计时器 handler 代码：

```
// handler for timer
timerHandler = (Handler) handleMessage(msg) -> {
        if (msg.what == 0x666) {
            if (gameType == MODE_NORMAL) {
                my2048Data.tickTime(1);
                scoreTimeTxt.setText(my2048Data.getTime().toString());
            } else if(gameType == MODE_COUNTDOWN) {
                my2048Data.tickTime(-1);
                scoreTimeTxt.setText(my2048Data.getTime().toString());
                if(my2048Data.getTime().isNil()) {
                    gameOver();
                }
            }
        }
};
isPause = false;
startTimer();
```

网络编程模块中同样用到了多线程，因为 http 请求的发送为异步操作，使用新线程，获取到数据后调用 handler：

```
public class NetworkUtil {
    static private Context context;
    static private final int SUBMIT_SUCCEED = 0x111;
    static private final int SUBMIT_FAIL = 0x222;
    static private final int GET_SUCCEED = 0x333;
    static private final int GET_FAIL = 0x444;
    static private Handler handler = handleMessage(msg) -> {
            switch (msg.what) {
                case SUBMIT_SUCCEED:
                    Toast.makeText(context, "上传成功", Toast.LENGTH_SHORT).show();
                    break;
                case SUBMIT_FAIL:
                    Toast.makeText(context, "上传失败", Toast.LENGTH_SHORT).show();
                    break;
                case GET_SUCCEED:
```