**1.课程实践介绍**

1.1 项目的设计任务。(说明到底做什么？)

这个项目的题目为个人助理，是一款帮助用户管理日程安排的应用。主要的设计任务是解决多个本地服务器与一个后台服务器的信息同步问题。

1.2 项目方案的选择。（说明所采用的方案）

这个项目采用的架构是C/S架构；采用的通信协议为Http；客户端与服务器之间的主要信息封装成Json字符串进行交互；

在同步实现方面，采用数据在本地数据库的最后一次更新时间即timestamp来比较判断数据是否是最新；每条记录采用App分配的id来作为记录的主键（将来将会弃用这一方案，改为由服务器为每条记录分配唯一的id进行唯一标定一条记录，App端的id只能从服务器获取，不可在本地生成）。

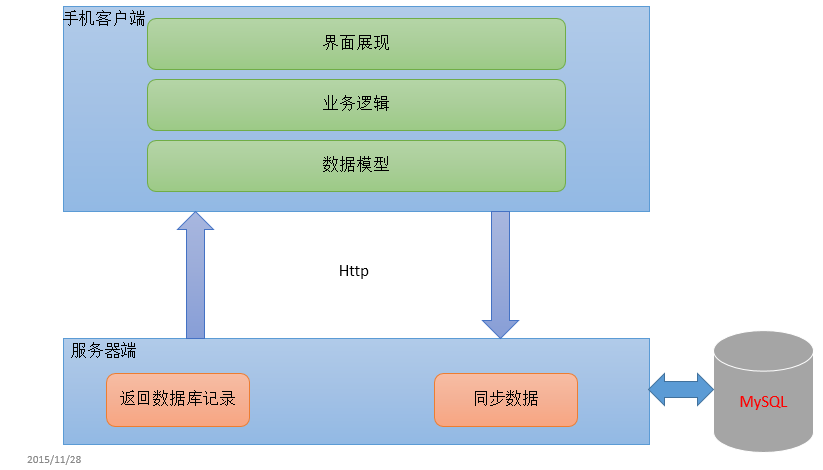
1.3 项目的功能。（此项目实现后有什么功能）

* 新增日志（可以设定日志内容、日志的日期和时间、是否提醒、是否重复提醒、重复提醒的时间间隔），并可以同步其他手机的客户端以及后台数据库；
* 更新日志并并可以同步其他手机的客户端以及后台数据库，更新的结果与最晚进行更新操作的客户端保持一致；
* 删除日志并可以同步其他手机的客户端以及后台数据库；
* 日志按提醒时间先后进行排列，方便查看下一个要完成的日志；
* 日志到点响铃提醒功能

**2.概要设计**

划分出各个软件模块，画出整体的软件设计框架图，对软件的各个模块进行说明

整体上C/S架构。



**3.详细设计**

3.1 **安卓端软件模块**

（1）**主要功能**。

* 1. **添加提醒**

可以添加以下内容：

1. 内容（文本类型）
2. 日期（日/月/年，字符串类型）
3. 时间(时：分，字符串类型)
4. 是否重复提醒（布尔类型）
5. 提醒间隔（整形数字）
6. 提醒类型（Minute、Hour、Day、Week、Month，字符串类型）
   1. **删除提醒**

删除特定的提醒记录

* 1. **修改提醒**

可以修改以下内容：

1. 内容（文本类型）
2. 日期（日/月/年，字符串类型）
3. 时间(时：分，字符串类型)
4. 是否重复提醒（布尔类型）
5. 提醒间隔（整形数字）
6. 提醒类型（Minute、Hour、Day、Week、Month，字符串类型）
   1. **同步数据到服务器数据库**

分为三个步骤：

1. **从服务器获取所有提醒记录（PUll）：**

这是第一次网络请求。

1. **手机进行数据处理并更新该App的SQLite数据库：**

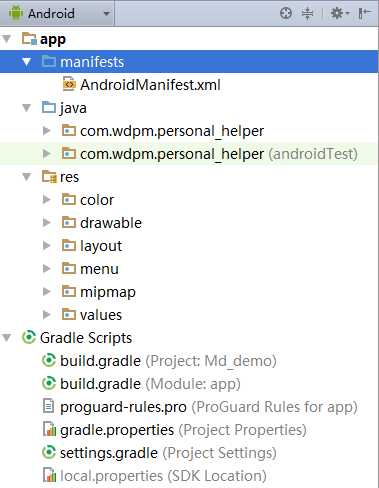
* 如果服务器返回的提醒记录为空（FALSE为标记），则只进行手机App中SQLite数据库的删除提醒的操作；
* 如果服务器返回的提醒记录不为空，则进行手机App中SQLite数据库的增删改查操作：具体分为以下流程：

先使用Gson解析出服务器的所有提醒记录，然后使用一个数组idInServer[] 记录所有在服务器的提醒记录的id，然后对SQLite中提醒记录进行轮询：

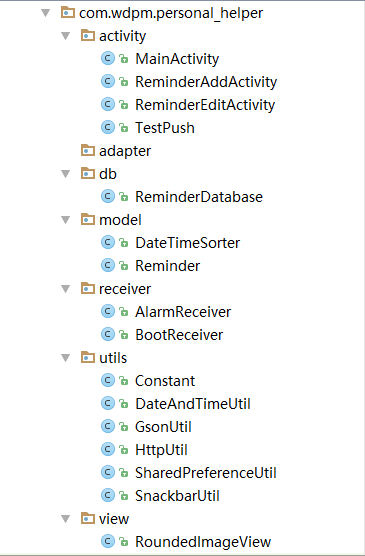
1. 如果该记录存在，如果服务器的这条记录的时间戳大于SQLite中这条记录的时间戳，则更新这条记录为服务器版本，否则不做改变。
2. 如果该记录不存在，如果服务器的这条记录的时间戳大于上一次同步时间，则向SQLite插入服务器的这条记录，否则不做改变。
3. 接下来，读取SQLite中所有提醒记录，对服务器所有提醒记录作轮询，找到不存在于服务器所有提醒记录中但存在于SQLite中的记录，如果SQLite中这条提醒记录时间小于等于上一次同步时间，则删除SQLite中这条提醒记录，否则不做改变。
4. **将更新后的SQLite数据库的所有提醒记录推到服务器（PUSH）：**

这是第二次网络请求。

（2）主要结构。



这是Android Studio中项目整体上目录结构。



这是代码目录的结构，具体说明如下：

* activity：
  + - MainActivity:主界面活动，长按提醒条目可以选择删除提醒记录
    - ReminderAddActivity：添加提醒活动
    - ReminderEditActivity：修改提醒活动
    - TestPush：临时测试活动
* adapter：暂为空，应该从MainActivity分离出来一部分
* db：
* ReminderDatabase：对SQLite增删改查操作的封装、建表、更新
* model：
* DateTimeSorter：日期比较的实体类
* Reminder：提醒的实体类
* receiver：
* AlarmReceiver：闹钟广播接收器
* BootReceiver：开机启动广播接收器
* utils：
* Constant：该App全局常量，例如服务器通信的地址
* DateAndTimeUtil：日期和时间相关函数封装
* GsonUtil：解析Json字符串的函数封装
* HttpUtil：Http网络请求的函数封装
* SharedPreferenceUtil：共享设置变量的函数封装
* SnackbarUtil：Snackbar控件的函数封装
* view：
* RoundedImageView：视图层的一个类，功能是自定义导航头像，将正方形截取为圆型

（3）每个软件函数说明。

* activity：
  + - MainActivity:
* onCreate
* configViews //设置侧滑栏
* onNavgationViewMenuItemSelected //设置NavigationView中menu的item被选中后要执行的操作
* onCreateContextMenu //设置长按事件的上下文菜单
* selectReminder //点击一个提醒项目
* onActivityResult
* onResume //重新回收recyclerView ，展示新建的提醒条目
* getLayoutManager //布局回收视图管理器
* getDefaultItemCount
* onCreateOptionsMenu
* onOptionsItemSelected
* onRefresh //处理下拉刷新事件
* onStop //记录是否为第一次使用该App的标记
* SimpleAdapter //提醒记录的适配器
  + - ReminderAddActivity：
* setTime //设置时间
* setDate //设置日期
* onTimeSet//从时间选择器获得时间并显示
* onDateSet//从日期选择器获得时间并显示
* selectFab1//选择响铃
* selectFab2//选择不响铃
* onSwitchRepeat//根据是否响铃显示相应文本
* selectRepeatType//选择重复提醒的类型
* setRepeatNo//设置重复提醒的间隔
* saveReminder//保存提醒记录
  + - ReminderEditActivity：
* updateReminder//更改提醒的信息

（注：该java文件其他函数方法基本和ReminderAddActivity相同，不再赘述）

* + - TestPush：//略
* adapter：
* db：
* ReminderDatabase：
* ReminderDatabase//带上下文参数的构造函数
* onCreate//含建表语句
* onUpgrade//更新数据库
* addReminder//增加记录
* getReminder//根据id查询特定记录
* getAllReminders//获取所有记录
* getRemindersCount//获得记录总条数
* updateReminder//更改记录
* deleteReminder//删除记录
* isReminderAlreadyExists//判断特定id的记录是否存在
* model：
* DateTimeSorter：
* public int mIndex //索引
* public String mDateTime//日期和时间
* （set 和 get方法）
* Reminder：
  + - * + private int id;//事务
        + private String title;//内容
        + private String date;//年月日
        + private String time;//时分
        + private String is\_repeat;//是否重复提醒
        + private String repeat\_no;//重复提醒的时间间隔
        + private String repeat\_type;//重复提醒的时间单位
        + private String active;//是否响铃
        + private String time\_stamp;//时间戳
* （set 和 get方法）
* receiver：
* AlarmReceiver：
* onReceive//处理闹钟意图，开启倒计时
* setAlarm//设置闹钟
* setRepeatAlarm//设置重复闹钟
* cancelAlarm//取消闹钟
* BootReceiver：
* onReceive//处理开机意图，继续闹钟倒计时
* utils：
* Constant：
* DateAndTimeUtil：
* stringToInt//提取字符串中所有数字，返回int类型
* getCurrentTimeStamp//得到系统目前时间，返回字符串类型
* getTime//将字符串类型时间转化为时间戳
* GsonUtil：
* remindersToJson//将SQLite中所有提醒记录转化为Json格式
* HttpUtil：
* get //get请求
* post // post请求
* getClient //得到静态实例化的异步请求客户端
* SharedPreferenceUtil：
* SnackbarUtil：
* SharedPreferenceUtil//带上下文参数的构造函数
* setKeyValue//设置键值
* getKeyValue//根据键获得值
* getFirstInFlag//获得是否第一次使用该App的标记
* setFirstInFlag//设置是否第一次使用该App的标记
* view：
* RoundedImageView：
* RoundedImageView//多个构造函数
* onDraw//绘制函数
* getCroppedBitmap //获得裁剪后的图片

**3.2 服务器php模块**

（1）主要功能。

服务器端（PHP）主要功能：

将数据库的记录封装成Json字符串返回给App；

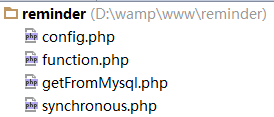
将App传来的记录与服务器数据库进行同步。

（2）主要结构。

服务器端使用MySQL建立数据库gerenzhuli；数据库里中含有表reminder，用于存储详细信息。

（3）每个软件函数说明。

服务器端（PHP）文件截图：



**功能说明**：

1.config.php 主要记录一些配置信息，比如数据库名字、访问密码等；

2.function.php 里面仅含有一个函数connectDb(), 用于连接数据库；

3.getFromMysql.php 将服务器数据库的所有记录封装成Json字符串返回给App；

4.synchronous.php 接受App端的数据并将其同步更新后台数据库；同步成功后将返回成功信息。

**3.3 服务器数据库模块**

具体见附件中MySQL建表语句文件。

**4.测试和截图**

4.1 系统测试。（描述出整个项目测试的操作过程，即一个操作手册）

* 1. 网络连接
     1. 含后台数据库的电脑连接实验室的WiFi，并用该电脑发射WIFI
     2. 手机连接电脑发射的WIFI即可进入下一步测试

1.2 同步测试（使用两台Android手机）

|  |  |  |  |
| --- | --- | --- | --- |
|  | 手机1（华为） | 手机2（中兴） | 预测结果 |
| 操作 | 新增名字分别为‘A’、‘B’、‘C’的三条日志，下拉刷新 | 下拉刷新 | 手机1和手机2以及数据库的记录完全一致。 |
| 删除记录‘A’，保持记录‘B’不变，把记录’C‘改为’Cnew‘，增加记录’D‘，下拉刷新 | 下拉刷新 | 手机1和手机2以及数据库的记录完全一致，即手机1更改后的版本。 |
| 先于手机2把记录‘B‘改为‘Bnew1’ | 后于手机1把记录‘B’改为‘Bnew2’，下拉刷新 | 手机1下拉刷新后，手机1和手机2以及数据库的记录完全一致，即手机2更改后的版本。 |

4.2 效果截图。（对实现和测试的结果进行截图说明）

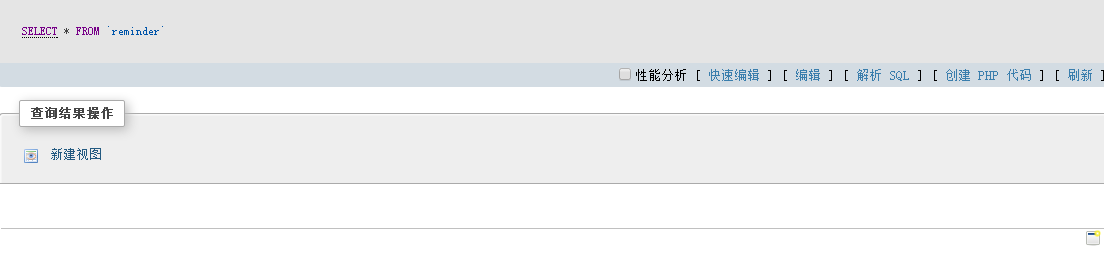
1. 初始化

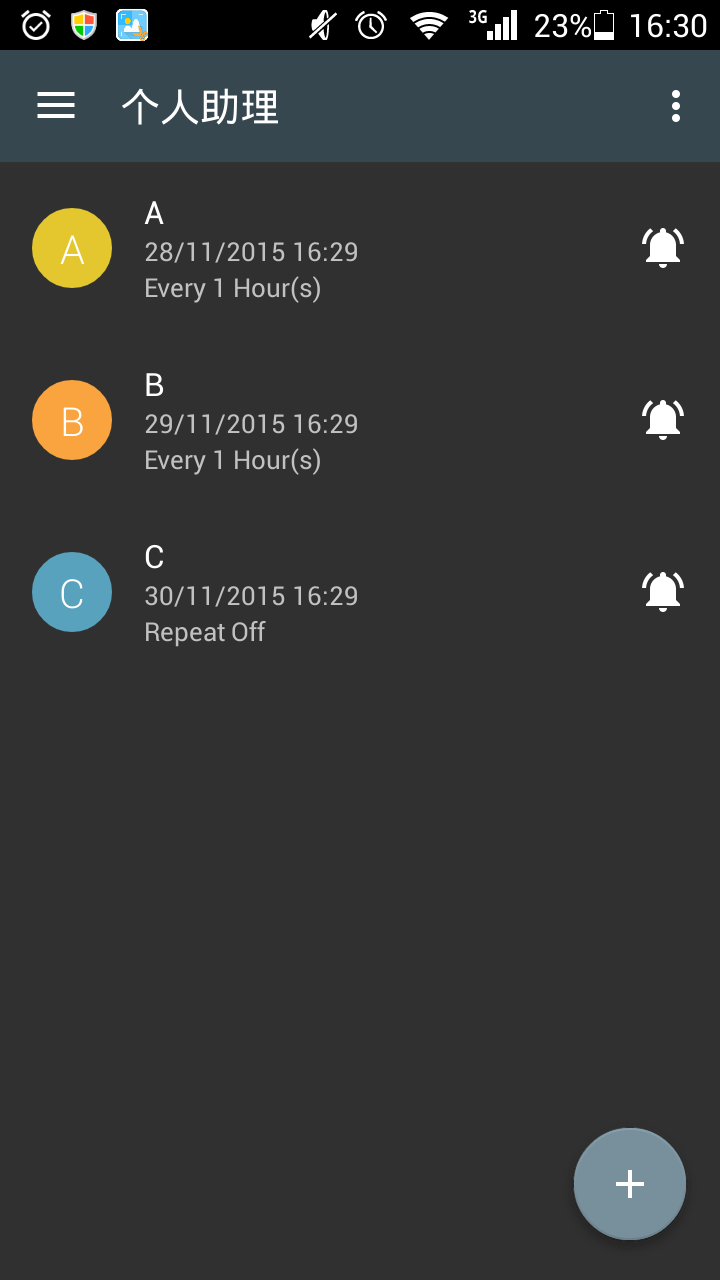
**手机2**

**手机1**

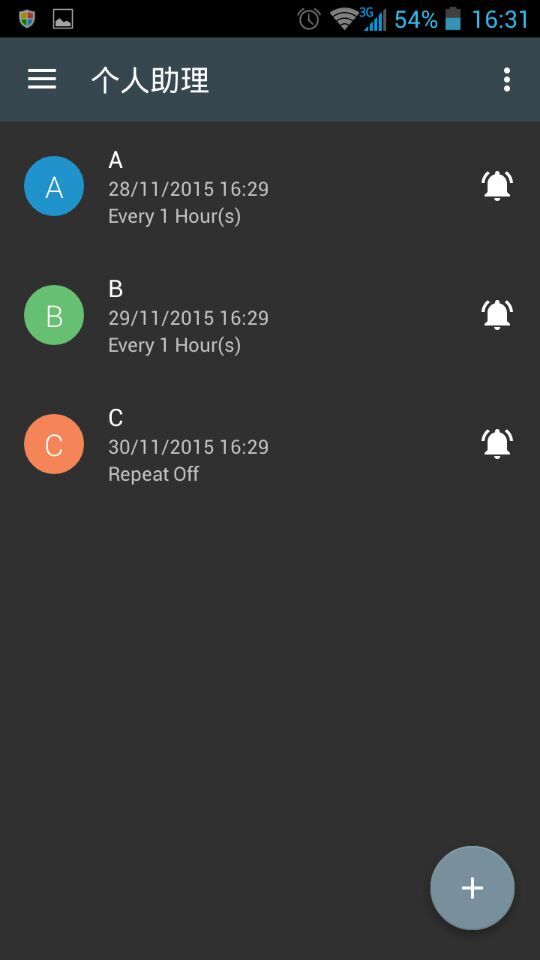
**后台数据库：**



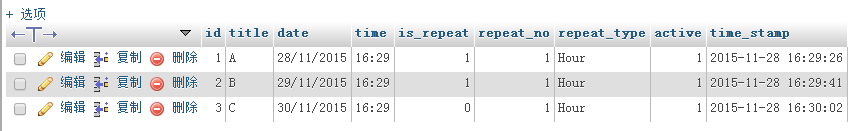
1. 手机1新增3条记录，分别为‘A’、‘B’、‘C’，下拉刷新



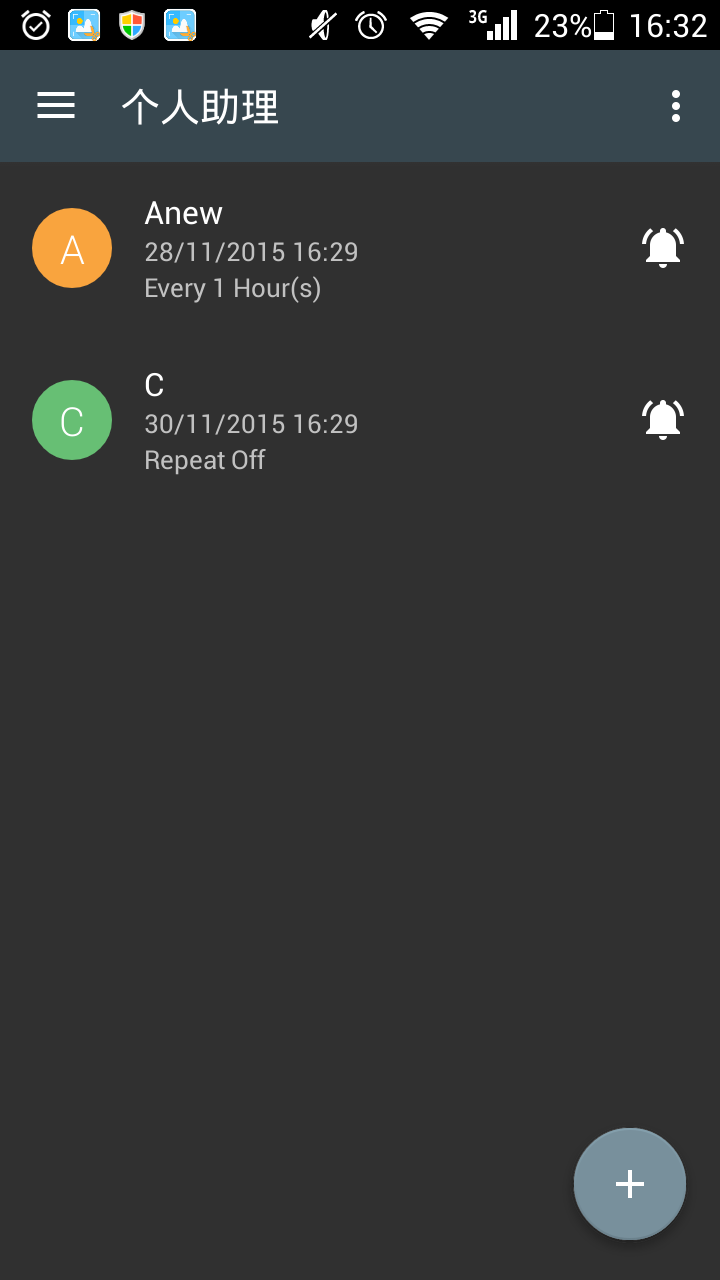
1. 服务器数据库与手机2进行刷新，结果与手机1一样



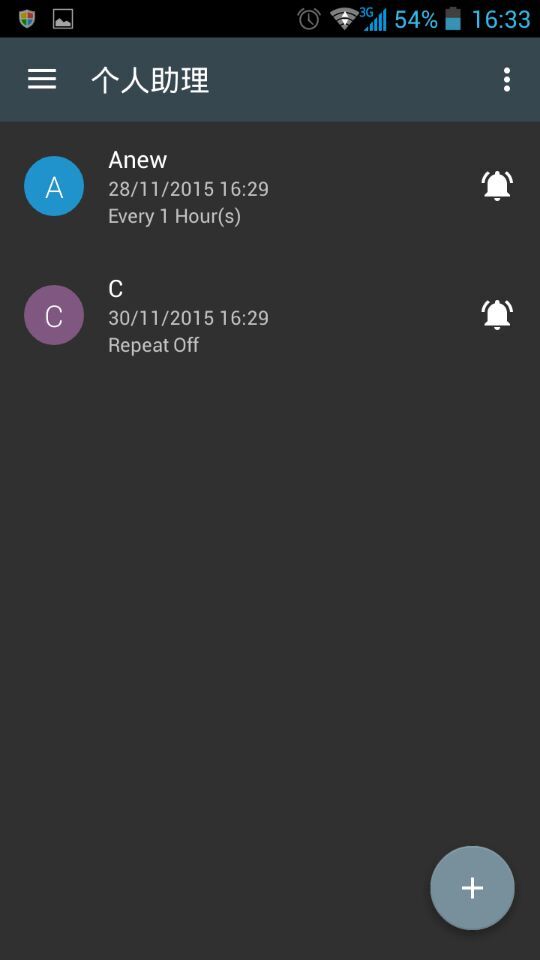
**后台数据库：**



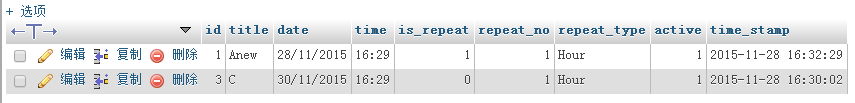
1. 手机1把记录‘A’改为‘Anew’ ， 删除记录‘B’，下拉刷新



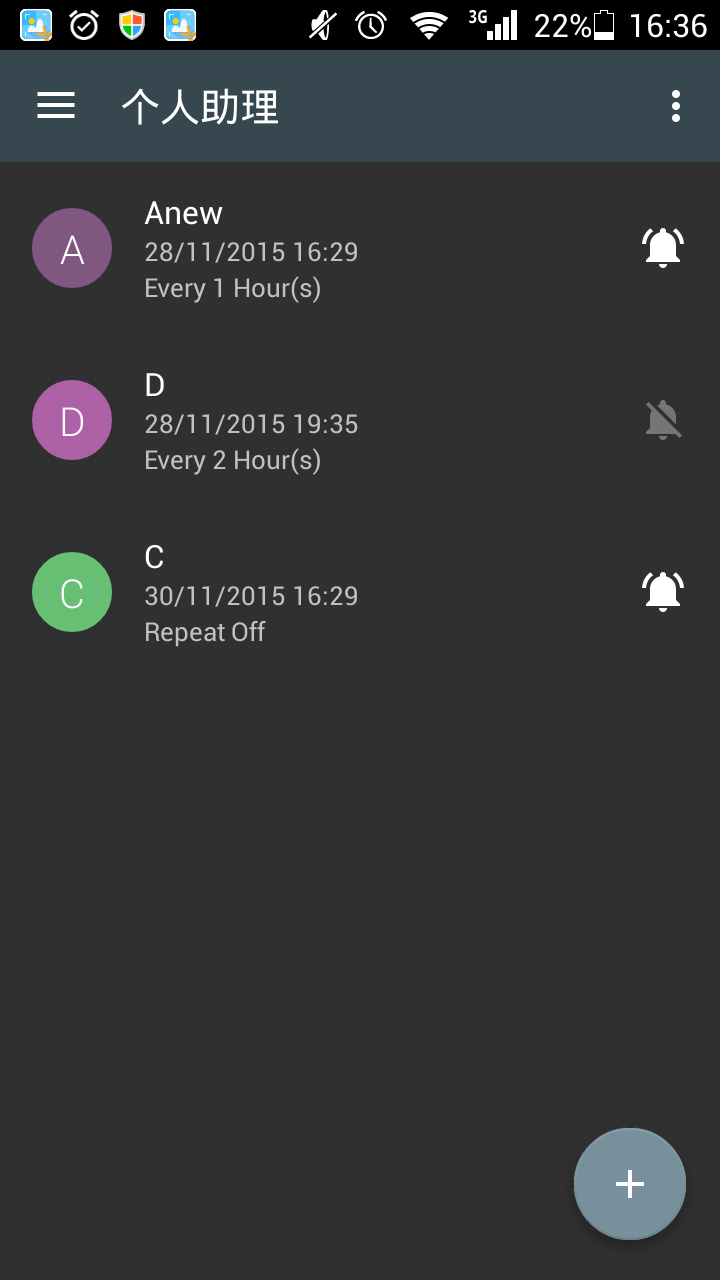
1. 服务器数据库与手机2进行刷新，结果与手机1一样



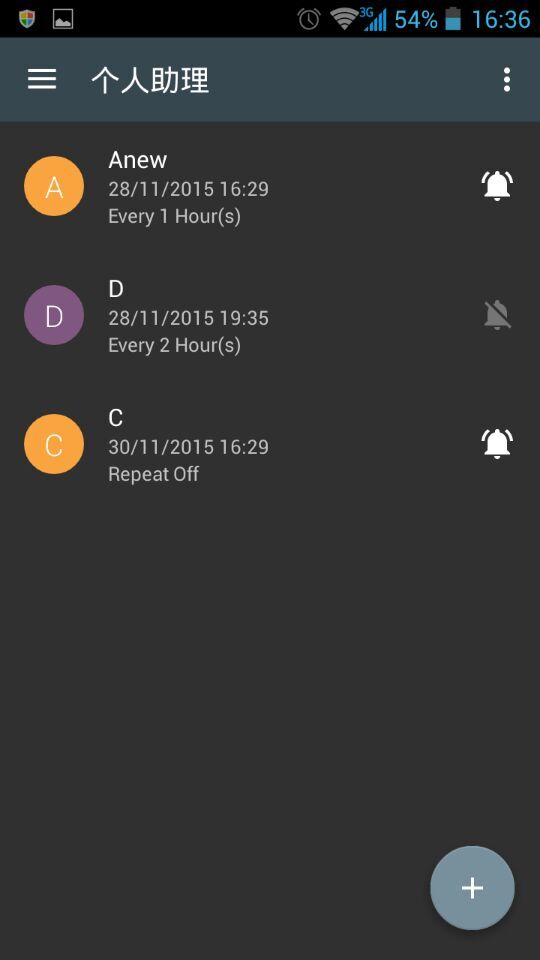
**后台数据库：**



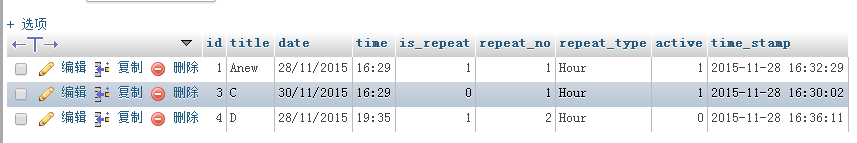
1. 手机1新增一条记录’D‘ ， 下拉刷新



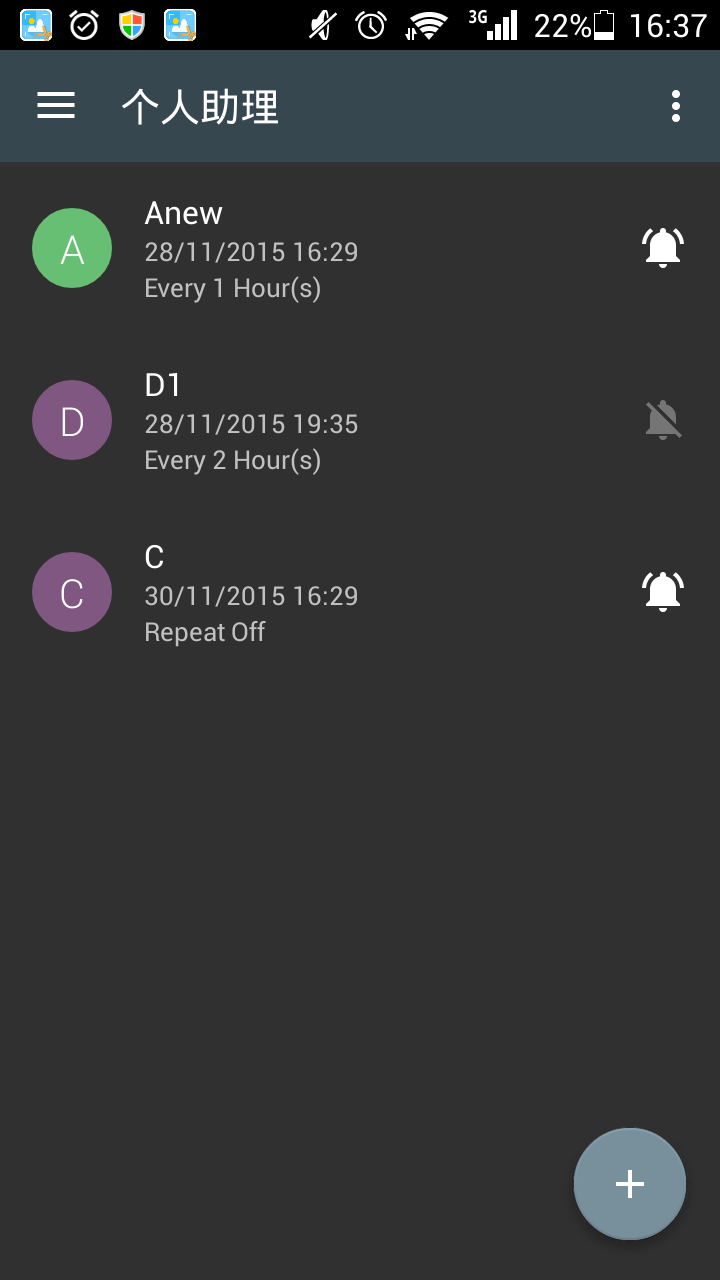
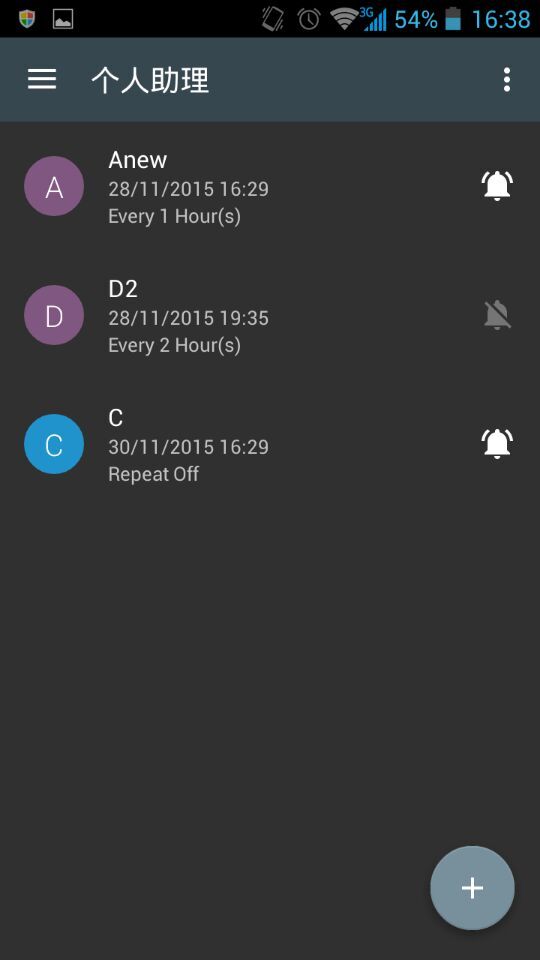
1. 服务器数据库与手机2进行刷新，结果与手机1一样



**后台数据库：**



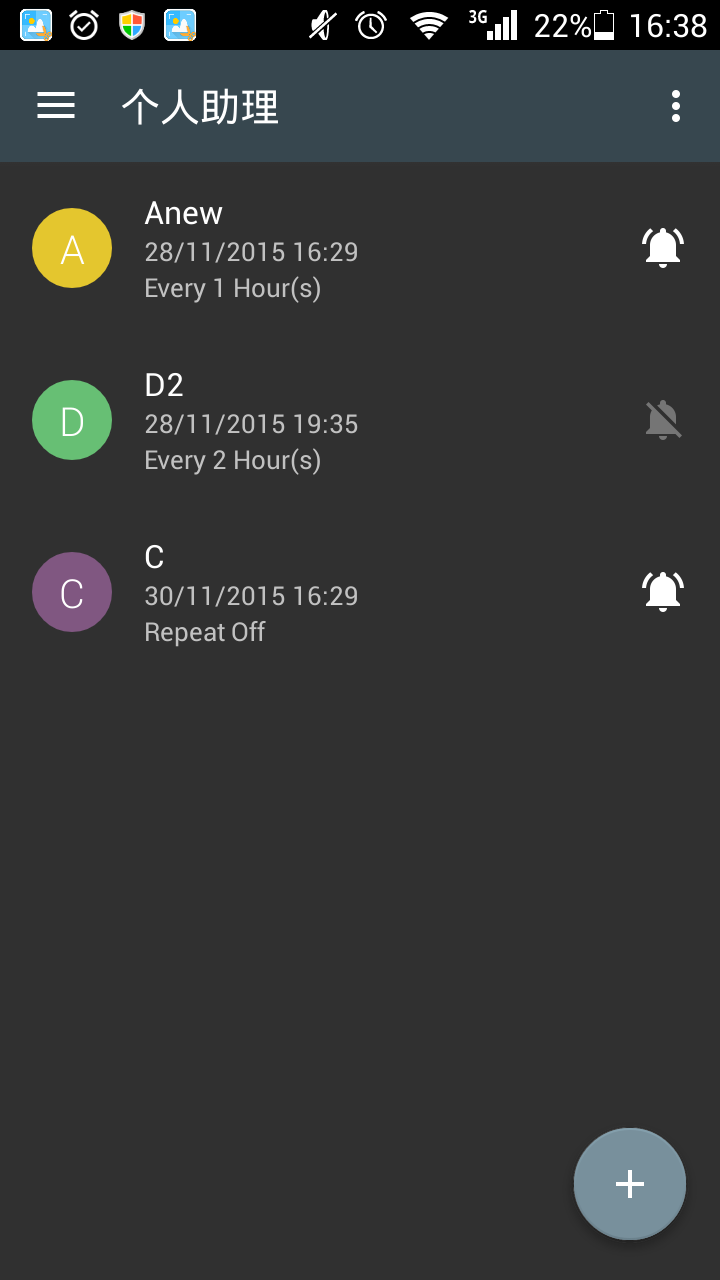
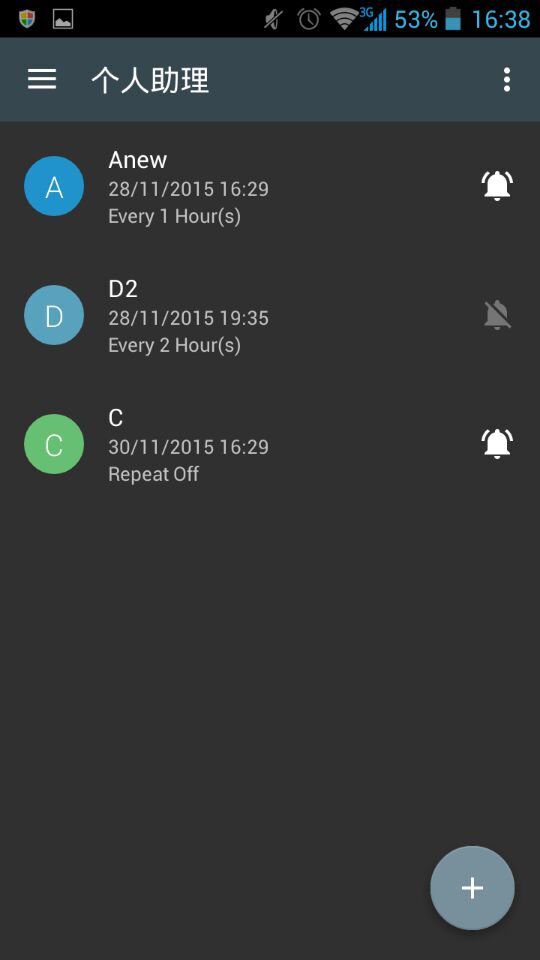
1. 手机1先把记录‘D‘改为’D1‘ ，然后手机2把记录’D‘改为’D2‘

**手机2**

**手机1**

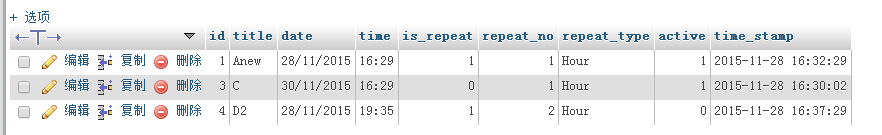
1. 刷新数据库，手机1和手机2，结果和手机2不变（最新），数据库与手机1和手机2一样；

**手机2**

**手机1**

**后台数据库：**



**5.总结**

1实现过程中遇到什么技术难点？

实验中遇到的主要难点为解决多个本地数据库数据同步逻辑，在调试过程中由于一些初始化问题造成应用多次崩溃，但最终都成功解决了。解决时间同步的方案主要是采用时间进行比较。

2实现中的关键点有哪些？

手机同步数据到服务器数据库，或者服务器数据库同步数据到手机，然后让手机App的数据库和服务器数据库保持一致。

3实现中采用什么方法实现该关键点？

通过id轮询，提醒记录的时间之间的对比，或者提醒记录的时间和上一次同步时间的对比进行处理。

4实现效果如何？

可以实现信息同步，但是还没有解决冲突。未来将采用服务器为每个日志分配id，App端不可自动生成id的方案解决冲突问题。

**6.附件**

* Android端源代码
* PHP源代码：php文件夹
* 数据库：mysql文件夹