基于 Android 的护眼应用客户端部分功能的开发

摘 要

随着电子产品特别是智能手机和电脑的普及，人们的眼部疲劳问题越来越突出。很多电子产品都具有高辐射、蓝光伤眼等问题，本课题正是针对这些问题，开发一款用于科学护眼的 Android 客户端软件。

该款软件实现了包括眼部问题检测、护眼建议及文章推送、日常用眼统计、专家解答等多种功能，对于用户的护眼问题具有很强的针对性，有效地缓解了眼部疲劳问题，引导了人们树立一种正确的用眼习惯。

开发整个过程，主要采用了目前主流的 Android 客户端+java 服务端的组合，客户端和后端均以 java 作为主要开发语言，运用了很多目前比较主流的前后端框架，整个系统的架构清晰简洁，充分运用了分层思想，使得系统各个模块高度解耦，易于扩展。系统的运行结果较为良好，服务端程序连续正常运行超过 20 天，客户端的连续运行 7 天均无异常。

关键字：android， java web， web 框架，mysql， redis

Abstract

opularization of electronic products, especially smart phones and computers, people's eye fatigue has become increasingly prominent. Many electronic products have high radiation, blue-ray hurting eyes and other problems, this project exactly addresses these issues so that develop an Android client software for scientific eye protection.

This software implements a variety of functions including eye problem detection, eye recommendation and article pushing, daily eye statistics, and expert answers. It is highly targeted for users' eye problems and effectively relieves eye fatigue and has guided people to establish a correct eye habit.

The whole process of development mainly uses the combination of the current mainstream Android client and java server. Both the client and the backend use java as the main development language. Many current mainstream front-end and back-end frameworks are used. The architecture of the entire system is clear and concise. Full use of layered thinking, making the system modules highly decoupled, easy to expand. The operating results of the system are relatively good. The server-side program runs continuously for more than 20 days. There is no abnormality in the client's continuous operation for 7 days.

Key Words：Android, Java web,Web Framework, Mysql, Redis

目 录

第一章 绪论

1.1 课题背景和意义

随着电子产品的普及，工作、学习压力的日趋增大，人们眼睛疲劳过度问题愈加严重。每位手机用户平均每天查看 150 次手机。换言之,除了休息时间外,每人平均每 6 分半钟查看一次手机。截至 2014 年底,中国手机网民规模达到 5 亿。在所有的"视力杀手"中,电子产品的"杀伤力"排名第一[1]。正因如此，护眼问题日益重要，如何科学地用眼、护眼，让自己的眼睛健康、明亮起来，也日益重要。

调查显示，蓝光危害是由波长介于 622 纳米与 822 纳米的辐射照射后引起的光化学作用，通常会存在导致视网膜损伤的潜能。如果照射时间超过 10s，这种损害机理起主要作用，而且是热损害机理的数倍之多[2]。因此，电子产品特别是针对手机的防蓝光措施尤其重要。

另外，很多人对自己的眼部存在的的问题认识不足，没有及时针对自己的眼部问题进行有效的补救和资料措施，日常生活也缺少科学用眼的意识，更没有专门地针对眼部问题请教相应的专业医生，从而导致眼部问题越来越严重。调查显示，近视是青少年中常见的多发病,发病率高达 40%[3]，而大部分近视一方面是由于电子产品使用的不合理，另一方面也是由于很多人缺乏科学的用眼方法，导致眼部问题越来越突出。因此，研究一款科学护眼的应用，从而提高人们科学用眼的意识，让人们树立良好的用眼习惯，对于缓解目前突出的眼部问题显得尤为有意义。

本应用试图通过专业的而科学的方法为用户提供一种科学合理的眼部护理方案，前期通过应用的眼部测试功能，运用云端技术，自动检测用户眼部问题，提供专业的方案供用户进行眼睛视力（包括辨色能力、明视能力、识光能力等）的测试，并根据结果，采用云端技术为用户提供合理的专家建议以及护眼方案，并根据用户目前用眼情情况，利用相关程序技术动态调整电子产品亮度、过滤蓝光，为用户提供一个科学、方便的护眼平台。

互联网的发展，催生出越来越多云端产品，而在 web 端编程，java 作为当之无愧的首选编程语言，它拥有众多的开源框架、丰富的文档资料以及活跃的编程社区，这使得利用 java 进行 web 开发尤其方便、高效且稳定。另外，android 系统作为移动端用户最多的操作系统[4]，它开源、稳定，加上 Google 等科技巨头的支持，android 应用开发的类库、组件丰富且成熟，开发 android 应用非常便捷、高效，因此，在开发本系统中，我们的应用选用了 android 作为客户端，java 作为服务端的搭配选择。

1.2 国内外相关产品研究和对比

目前，国内外市场上充斥着各种护眼应用，例如比较著名的护眼宝就是其中较为优秀的代表。这些应用大部分都有蓝光过滤、各种护眼模式选择等功能，对于蓝光过滤能起到一定的作用。 然而，在调查、使用了 10 款的护眼应用之后，发现很多护眼应用功能较为单一，功能上仅仅停留在蓝光过滤等较为表面的护眼功能实现上，而且很多应用充斥着商业广告，用户体验非常不好。另外，很多应用对于用户的用眼问题也没有很好地提出相应的护眼意见，更没有相应的眼部问题测试功能，护眼作用非常有限。

和其他产品相比，本应用的功能更具有针对性。本应用除了提供市场上护眼应用普遍有的蓝光过滤、护眼模式选择等手机护眼基本功能外，还有相应的测试模块、护眼知识模块、眼睛使用情况模块等，用户可以使用应用进行眼部问题的科学测试之后，再进行针对性的护眼措施。另外，应用本身针对用户眼部问题，提供相应的提问模块入口，用户可以就自己护眼方面的问题向后台的专家进行提问，把自己在用眼、护眼中存在的问题向专家请教，从而使护眼更具有针对性和科学性。

1.3 系统客户端功能模块介绍

客户端模块按应用的功能分为以下几大模块：视力测试模块、用户模块、知识库模块、蓝光护眼模块、用眼数据模块。

视力测试模块主要提供色盲、散光、视力测试和敏感度测试。主要

测试方式是通过图片素材，根据用户的正确率判断患病可能性。其中，敏感度测试主要是测试用户分辨相近物体的能力，这里主要用相近的颜色块对用户进行测试。在用户测试完毕，可以选择把测试结果提交到云端进行报保存，方便专家对其用眼情况进行分析。下图是该模块的相关截图：

图 1-1

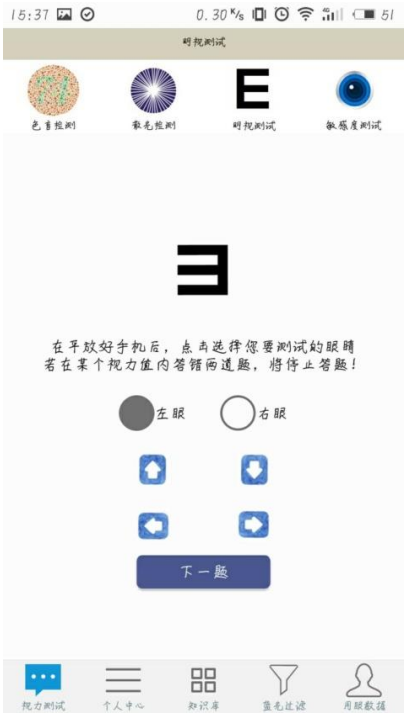


图 1-2

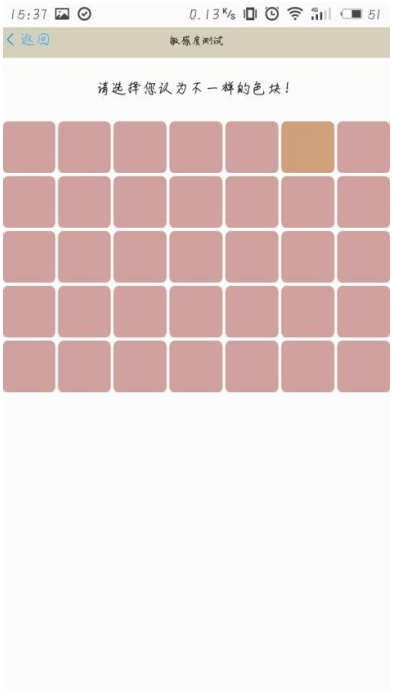


图 1-3　　图 1-4

个人中心主要用于用户相关的数据浏览，主要包括修改个人资料、当天的用眼情况、眼部测试数据、专家交流记录和消息中心。其中，专家交流记录主要记录和用眼专家的聊天记录，用户可以在这里向专家进行用眼咨询。

知识库主要是用眼相关的文章以及建议，用户可以在这里查看专家们的用眼文章。文章类型有饮食习惯、护眼博客文章，同时也会发布一些线上的讲座信息，另外，用户也可以对专家进行专门的提问。下图是详细的界面图。

图 1-5　　图 1-6

蓝光过滤模块主要用于调节手机屏幕达到护眼效果，主要有几种模式选择：夜间模式、护眼模式和正常模式，另外，用户也可以根据自己的喜好，选择其他的背景光调节。下面是详细的效果图：

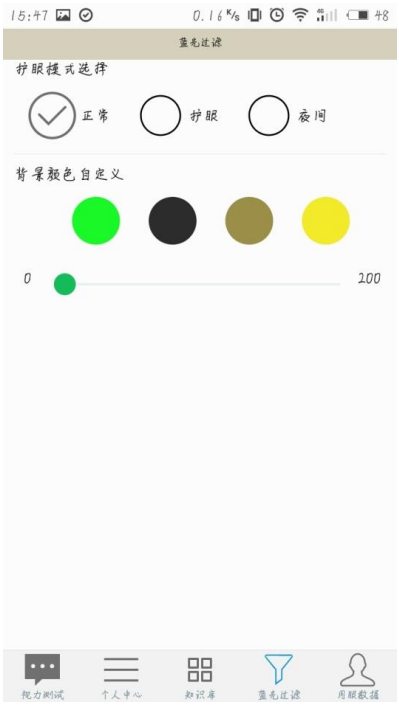


图 1-7

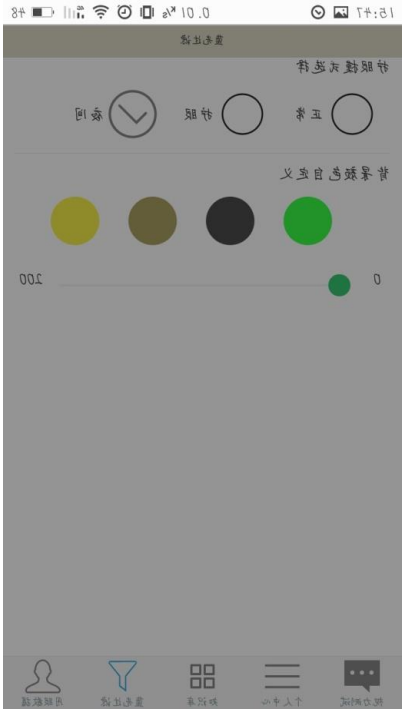


图 1-8

图 1-9

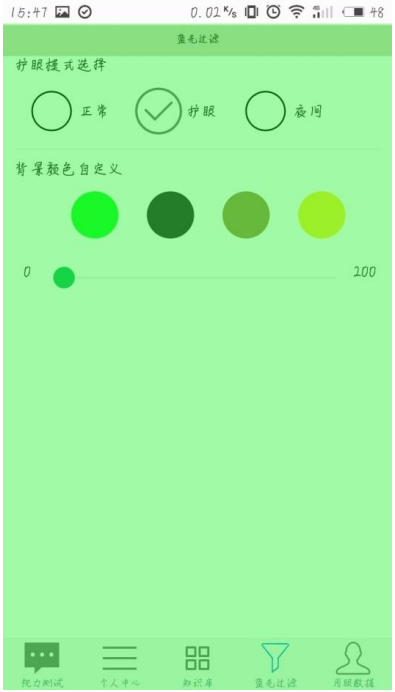


图 1-10

用眼数据主要记录用户使用手机相关的次数，例如今天开屏次数、连续使用手机时间、本次开屏使用手机时间等等。效果详细如下：

图 1-11

下面是主要的应用的功能分布简略图：

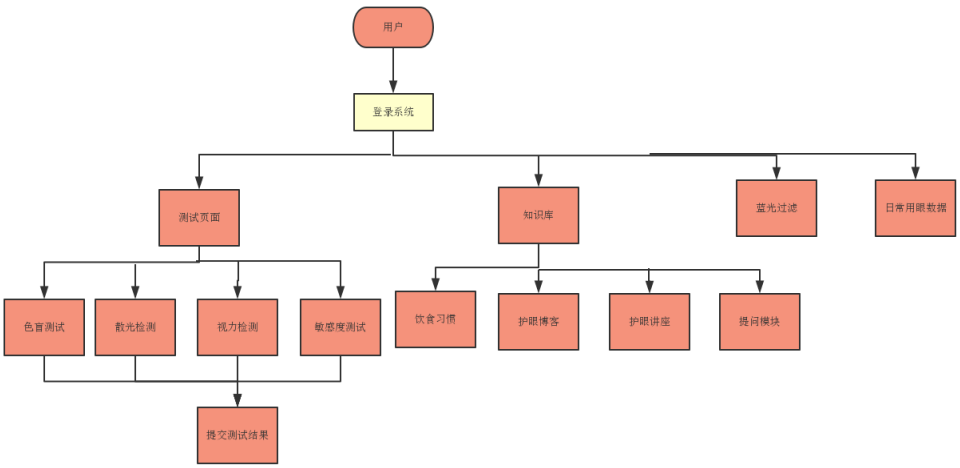


图 1-12

后端的功能模块主要分为：用户模块，测试模块、文章模块，模块对应前端客户端的各个模块，后端对应模块对前端的模块提交的数据进行处理、存储等后台操作。

第二章 可行性分析

2.1 技术可行性

目前，web 开发技术领域总，java web 体系是主流的技术，很多开源框架可以免费使用，同时开源社区也很多现成的技术资料可以参考。我们系统基于 Android+java 后端的开发，技术上已经非常成熟，开发集成环境也有很多选择，各种 java 类库应有尽有。另外，开源的数据库系统、ORM 框架也应有尽有，技术成熟度较高，开发门槛也比较低，所以，开发难度上完全是可控的。

另外，服务端利用 linux 作为操作系统，加上 tomcat 的容器，对于本应用而言，运行环境完全是很好搭建的。

2.2 经济可行性分析

经济上而言，开发软件除了人力外，基本无经济支出，加上阿里云的廉价学生服务器和开源社区上应有尽有的开源产品，开发系统更多是脑力层面的成本支出，所以，开发该系统的经济成较低，有很强的可行性。

第三章 相关技术简单介绍

3.1 Android 操作系统

Android 是应用于智能手机或者平板电脑上的一款操作系统。Android 是 Google 公司在 2007 年 11 月 5 日公布的基于 linux 内核的操作系统。早期由 Google 开发,后由开放手持设备联盟(Open Handset Alliance)开发。它采用了软件堆层的架构,主要分为三部分。底层 linux 内核只提供基本功能,其他的应用软件则由各公司自行开发,部分程序以 Java 编写。[5]

Android 作为目前移动端使用最多的操作系统，具有代码开源、应

用程序易于开发、特性丰富等优点。Android 操作系统内核其实是基于 Linux 内核进行开发的，支持多任务、多种网络制式（包括 WiFi/蓝牙/移动数据等等）。另外，由于 Android 基本用 java 进行开发，而 java 具有很强的面向对象特性，加上 Google 丰富的 Android 类库、插件和强大的集成开发环境， Android 应用的开发是非常低门槛、高效率的。

利用 Android 集成开发环境，特别是 Google 官方推荐 AndroidStudio，它很好地封装了整套针对 Android 的开发组件，加上先进的 Gradle 依赖管理系统，开发的过程让人有种赏心悦目的感觉，高效而优雅。

在 Android 开发中，优秀的 Google 大神把 Android 的类库封装得非常优雅，利用 xml 进行布局文件的设置，通过 java 语言进行页面动态操作，分离了界面和业务，降低了耦合度，灵活易于扩展，这种 Android 应用开发方式也是一种非常了不起的编程设计思想。

3.2 MVC 模型

MVC 模式最早由 Trygve Reenskaug 在 1978 年提出，是施乐帕罗奥多研究中心（Xerox PARC）在 20 世纪 80 年代为程序语言 Smalltalk 发明的一种软件架构[6]。MVC 的分层思想是软件设计中最为重要的解耦思想之一，通过 MVC 的模型思想，可以大大降低软件各个模块（界面模块、业务模块等）的耦合度，增加代码复用度，提高代码模块的可读性。

MVC 其实是下列三个英文的缩写：Model、View、Controller，分别代表的意思是模型、视图和控制器。Model 代表的是编程里面的对象，可以理解为数据以及逻辑的集合，对应业务中现实的具体模型和行为；View 其实是界面，我们平常可以看得到的软件界面、浏览器界面都可以理解为软件架构中的 View；Controller 其实是业务代码的入口，它负责把 View 传过来的数据进行分发，交给不同的业务逻辑代码进行处理，并返回前端处理完成的结果，可以理解为一个路由器。总的来说，Model 是数据，View 代表界面，而 Controller 代表逻辑代码的处理入口。

MVC 的架构思想，很好地解决了传统软件开发中界面和逻辑代码

混在一起的问题。在软件开发一开始，人们并没有 MVC 的思想，而是把所有代码放在一个地方，页面实现代码、逻辑代码混在一起，加大了系统扩展的难度，也不利于代码复用，而 MVC 设计思想的出现，很好地解决了上述问题。

下面是 MVC 的一般架构思路原理图：

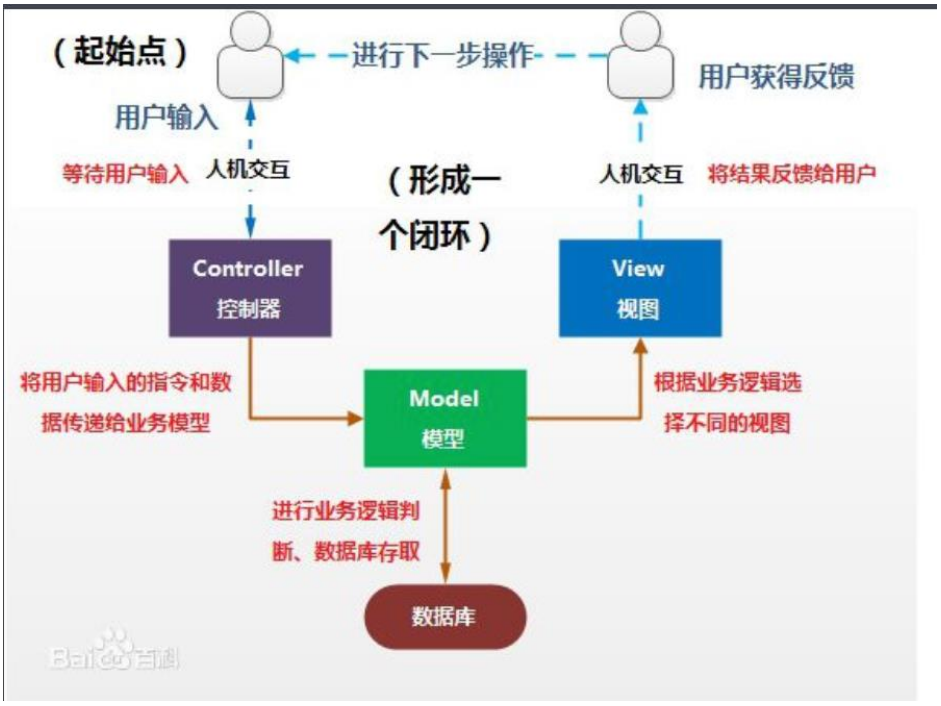


图 3-1

在上图中，Controller 负责接收用户的数据，进行数据分发；Model 层是核心的数据层；View 是视图，负责展示数据。

3.3 数据库 sqlite

SQLite 遵守 ACID，实现了大多数 SQL 标准。它使用动态的、弱类型的 SQL 语法[7]，常常被用在客户端中作为本地数据的存储。它不是服务端的数据库系统，而是一种较为小型的客户端数据库嵌入到相应的操作系统中去。在 Android 系统内部就内嵌了 sqlite 数据库，可以非常方便地存储对应的本地数据。总的来说，sqlite 非常轻便，操作数据时也很方便，用作客户端数据库非常有优势。

3.4 java 编程语言

Java 语言是一门非常强大的 web 服务端开发语言，具有很多非常优秀的特性：跨平台性、内存自动回收特性、易于使用等。Java 几乎是任何网络应用的基础，也是开发和提供嵌入式应用、游戏、Web 内容和企业软件的全球标准。 Java 在全球拥有超过 900 万名开发人员[8]。

Java 语言除了拥有完善的体系、强大的平台性外，Java EE 平台 (Java Platform Enterprise Edition) 是企业 Java 计算的行业标准[9]。Java EE 是 web 端最为流行的平台技术，开源框架众多，各种类库工具也应有尽有。另外，如今 java 也是 Android 应用开发主要程序语言，在 Android 中，针对 Android 应用设计的类库插件非常丰富，应用开发门槛较低，是的 Android 软件的开发具有很强的便捷性和高效性。

3.5 C/S 架构

主从式架构 (英语：Client–server model) 也称客户端-服务器（Client/Server）架构、C/S 架构，是一种网络架构，它把客户端 (Client)（通常是一个采用图形用户界面的程序）与服务器 (Server) 区分开来。每一个客户端软件的实例都可以向一个服务器或应用程序服务器发出请求[10]。C/S 架构具有很强的优势，服务端和客户端的分离使得服务端程序可以复用，不同客户端完全可以使用同一个后端程序，通过网络协议的通讯，达到前后端分离的目的，Web 应用很多都是基于 C/S 架构进行开发组织的。

3.6 http 网络请求框架、json 以及 json 处理框架

网络请求主要采用 http 的请求协议，在 Android 客户端主要使用 okhttp 这个网络框架进行请求发起和数据接收，okhttp 是一个非常强大的 http 网络请求框架，支持异步发起请求以及多种请求方式。Okhttp 在网络状况不佳时，会持续地继续请求，同时，它也支持备用地址设置、这对于当前的 IPV4+IPV6 和数据冗余中心托管服务是必要的[11]。Okhttp 这个网络请求框架广泛用于客户端、服务端的 http 通讯服务中，本系统正是利用该框架对后端发起 http 请求的。

在和后端交互数据中，主要采用 json 字符串的形式进行交互。JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式，易于人阅读和编写。同时也易于机器解析和生成，它基于 JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999 的一个子集[12]。Json 字符串是一种非常简洁、轻便的数据表现形式，主要是以 key-value 的形式对数据进行序列化，继承了 xml 的可读性良好的同时，降低了字符通讯的开销，高效、轻量而便捷。客户端接收到 json 字符后，通过对应 json 处理框架便可把数据提取出来，显示在页面上。

目前对 java、php、javascript 等 web 端语言都有丰富的 json 处理接口和框架，json 的序列化和反序列处理非常方便、高效。本系统主要是利用 java 处理 json 字符串中的优秀框架

Fastjson 是阿里巴巴下的一个开源的 json 处理框架，具有很快的处理速度，轻量级且易于使用。它也可以用来将 JSON 字符串转换为等效的 Java 对象。 Fastjson 可以使用任意 Java 对象，包括没有源代码的预先存在的对象[13]。

3.7 Gradle 依赖管理和 Android Studio 继承开发环境

Gradle 是一个基于 Apache Ant 和 Apache Maven 概念的项目自动化建构工具，它使用一种基于 Groovy 的特定领域语言来声明项目设置，而不是传统的 XML[14]。Gradle 与传统的依赖管理工具不一样，它可以通过 java、Scala 等语言进行依赖逻辑的书写，大大满足了开发人员对于日益复杂的应用系统的构建开发。

Android Studio 是 Google 专门为 Android 开发者定制的开发集成环境。由于专门针对 Android 开发者进行定制，所以它具有很多种针对 Android 开发的功能，Android Studio 利用优秀的代码编辑、调试、性能工具、一套灵活的构建系统以及一套即时构建/部署系统[15]，大大提高了 Android 应用开发的便捷性，使得工程师只需要集中解决应用开发的业务问题。

第四章 系统设计原理

4.1 系统总体架构

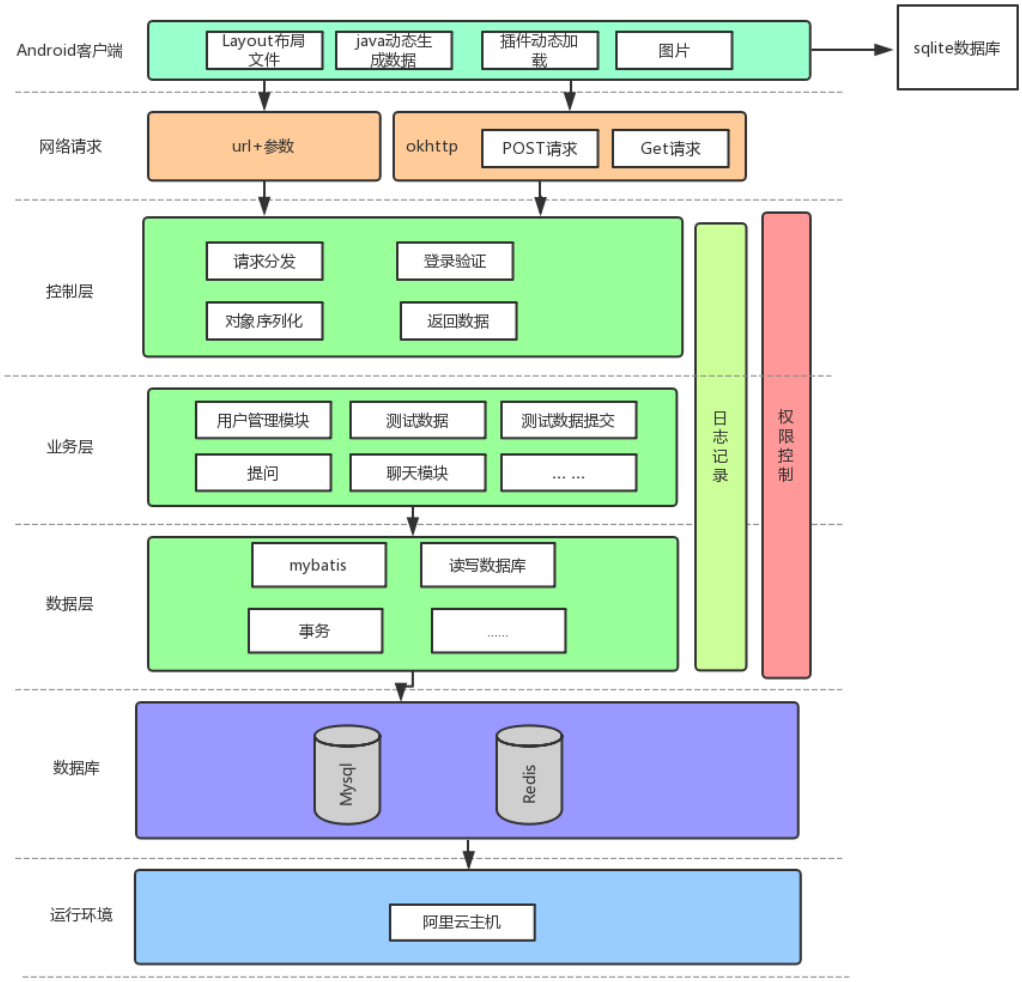


图 4-1

系统的总体架构图如上。总的请求过程：Android 客户端，在用户进行输入操作并提交数据以及发起请求之后，Android 端通过 okhttp 这个 http 请求框架发起网络请求或者请求本地的 sqlite 数据库查询数据。

如果是需要请求后端数据，则直接发起 http 请求，把数据提交到 web 容器。后端系统入口是 springMVC 这个框架所实现的 Controller 层，数据经过 service、DAO 之后最终到达数据库，在进行数据库查询、增

加等操作之后，操作结果的数据集会沿路返回，在 Controller 调用序列化相关的框架对数据进行序列化处理后以 json 字符串的形式，通过网络输出流返回给客户端。

如果请求无需请求后台系统，直接操作 android 本地数据库即可的话，那么，客户端将会直接调用封装好的本地数据库操作工具类进行本地数据库操作。

4.2 客户端模块架构

客户端的架构图大概如下：

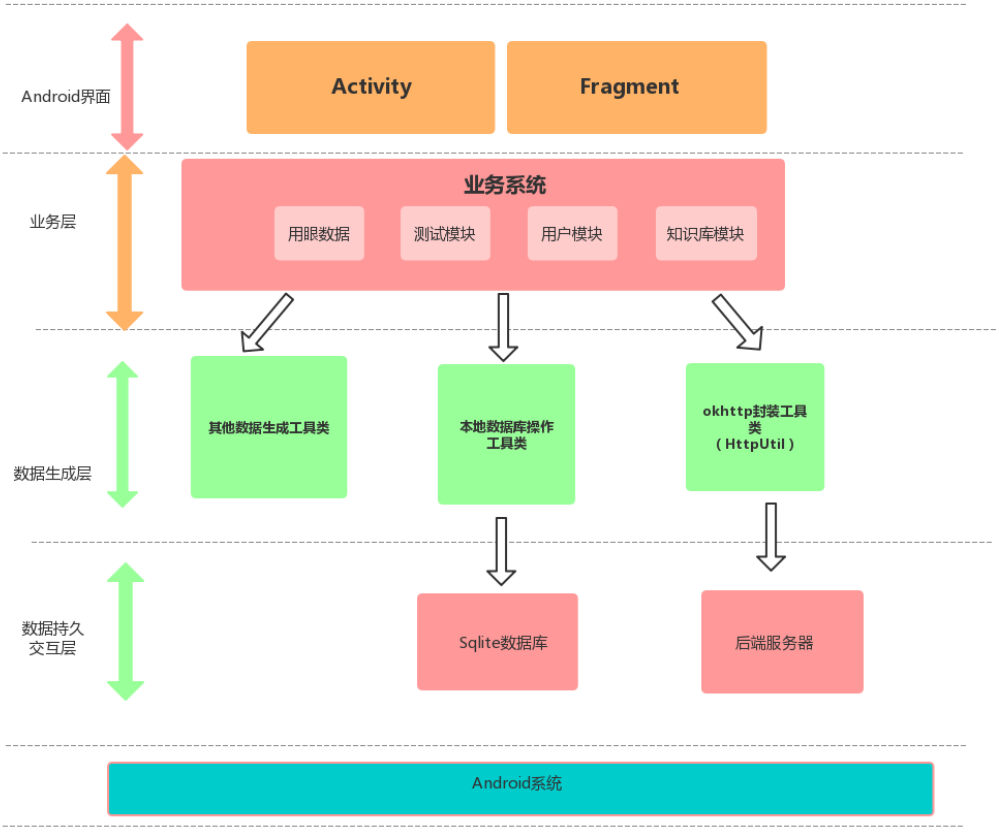


图 4-2

客户端的总体架构较为简单，界面主要通过 Activity 和 Fragment 进行数据的动态展示，在开发时，用到一些动态加载数据的组件实现下拉刷新、向上拉去加载更多等动态功能。

在客户端的业务层，主要有三种方式获取数据：工具类生产，本地

数据库读取以及向服务端请求。对于敏感度测试等模块，主要通过工具类自动生成测试所需数据；而一些色盲测试、用户数据相关以及知识库等等则需要请求后端进行数据的读取；当然，对于当天的用眼情况统计，主要存储在本地的 sqlite 数据库，一般通过封装好的一个工具类进行直接数据操作即可。

第五章 模块具体实现分析

5.1 视力测试模块

5.1.1 色盲测试模块

下图是色盲测试模块的截图，其他模块类似：

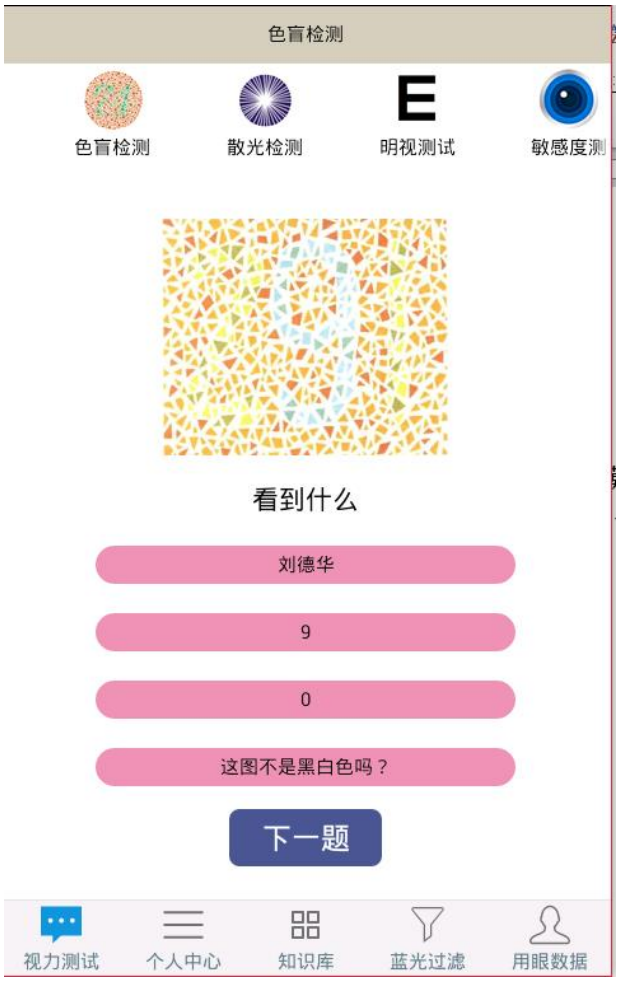


图 5-1

测试模块中，色盲测试和敏感度测试均是通过读取后端的数据获取

测试题目的数据。测试数据的后端表结构如下图：

名称	类型	空	默认值	属性	备注(C)
主索引(P)	Id			unique	
Id	int(11)	no	<auto_increme...		
title	varchar(255)	yes	<空>		问题的标题
img_url	varchar(100)	yes	<空>		题目涉及到的图片路径/如果题目没有图片，为空
option1	varchar(100)	yes	<空>		题目选项1
option2	varchar(100)	yes	<空>		题目选项2
option3	varchar(100)	yes	<空>		题目选项3
option4	varchar(100)	yes	<空>		题目选项4
correct_option	int(2)	yes	<空>		取值从1-4，代表正确选项
type	char(2)	yes	<空>		测试问题类型：色盲/明视/敏感度/散光

图 5-2

题目的构成主要有一下部分：题目素材（一般是一个图片素材），题目内容、各个选项以及对应的正确选项，type 是代表不同类型的题目。测试相关的数据动态地从后端读取，在打开 Android 页面的时候，便进行数据初始化了，系统自动一次性地缓存当次测试的所有数据。具体的初始化数据代码大概如下：

```
ArrayList<TextQuestion> questions;//这是存储问题数据的列表
int nowAnswerQuestion = 0;//存储当前答题的序号
private void init(){
    //前面初始化控件过程已经省略
    TestService service = new TestService();
    //初始化数据
    try {
        questions = service.getTestQuestions(QUESTION_NUM,
                GlobalConst.TEST_TYPE_COLORBIND);
    } catch (HttpException e) {
    }
    //初始化第一个答题页面
    showNewQuestion(questions.get(nowAnswerQuestion));
}
```

总的来说，从后端读取数据的过程，其实是调用服务层（Service 相关的类）对数据进行读取的，而数据主要以一个 ArrayList<TextQuestion> 列表来进行存储的。在读取完数据之后，会进行数据的展示，调用的是

showNewQuestion()这个方法。在服务端读取数据的核心代码大概如下：

```
public List<TestQuestionVO> getTestQuestions(int num, String type)
    throws HttpException {
    //前面还有一些参数正确性的判断，这里省略掉

    List<TestQuestionVO> questions = new ArrayList<>();
    Map<String,String> params = new HashMap<>();
    params.put("type",type);
    params.put("questionsize",num + "");
    String result = HttpUtil.synGet(GET_QUESTION_PATH,params);
    Map<String,Object> resultMap =
            (Map<String,Object>)JavaBeanUtil.jsonToObj(result);
    int status = (Integer) resultMap.get("status");
    if(status==10){
        throw new UserException(GlobalConst.REMIND_NOT_LOGIN);
    }
    if(status==1){
        throw new
    BackstageException(GlobalConst.REMIND_BACKSTAGE_ERROR);
    }
    List<Map<String,Object>> data =
(List<Map<String,Object>>)resultMap.get("data");
    for(Map<String,Object> m : data){
        questions.add(mapToQuestion(m));
    }
    return questions;
    }
```

在读取数据中，核心代码是这个：String result = HttpUtil.synGet(GET_QUESTION_PATH,params);这段段代码是发起 http

请求的通用代码，主要是调用 HttpUtil 这个封装好的工具类进行请求的发起，请求成功之后，会返回一个 json 字符串（result），猴面再对字符串进行反序列化过程，将之变为一个对象即可得到对应的对象数据。由于具体的 http 请求代码比较多，这里就省略了。总的来说，色盲测试模块的请求过程可以用如下的图表示：

5.1.2 散光测试模块

功能实现过程和色盲测试基本一致，略。

5.1.3 明视距离测试模块

明视距离测试的页面图如下：

图 5-3

在进行明视距离测试时，首先需要用户输入测试眼睛和手机之间的距离，系统会根据用户输入的数字，自动计算需要显式图标的大小。总的数据请求过程和色盲测试部分基本一致，只是视力相关的测试数据无

需发起后台请求，直接在本地生成即可。具体的核心代码调用如下：

```
questions = service.getVisionQuestions(testDistance,VISION_NUM);
```

总的来说，就是调用对应的查询方法获取数据，下面具体分析下生成用眼数据的相关算法过程，核心代码如下：

```
public static TestVisionQuestionVO getTestVisionQuestionVO(float
    distance,float vision){
    //省略参数逻辑判断代码
    TestVisionQuestionVO question = new TestVisionQuestionVO();
    //随机获取方向
    int randomInt = (int)(Math.random() * 4);
    char direction = '左';
    if(randomInt<1){
        direction = '左';
    }else if(randomInt<2){
        direction = '右';
    }else if(randomInt<3){
        direction = '上';
    }else {
        direction = '下';
    }
    question.setDirection(direction);
    //5.0 标准分辨大小距离
    float normalVisionSize = distance * EYE_MIN_ANGLE;     int
multiple = (int)((5.0-vision)/0.1);//计算 vision 值和 5.0 视力值的倍率关系
    float size = (float)(normalVisionSize *
(Math.pow(EYE_ANGLE_INCREASE_RATE,(float)multiple)));//计算对应
vision 的尺寸
//标准视力表是五分最小视角，所以乘以 5，乘以 10 是因为 question
//的 size 返回值是毫米
    question.setSize(size * 10 *5);
```

```
    return question;
}
```

明视距离核心算法是，把距离乘以人的最小分辨角便是最小的可分辨距离，另外，对于视力 4.0-5.0 这个区间的视力值和图标大小的计算，主要遵循以下的计算原则：在标准视力中，视力之间和图标的大小满足一定的倍率关系，标准视力表 5.0 视力是 5 分最小视角，具体公式如下：

$$\frac{\text{实际图标大小}}{\text{5.0标准图标大小}} = \frac{\text{5.0-视力值}}{0.1}$$

图 5-4

而对于不同的距离的 5.0 最小可分辨距离可以直接如下公式计算：

$$\text{5.0最小可分辨大小} = \text{最小可分辨视角} * 5 * \text{距离}$$

在以上计算中，最小可分辨视角是一个常量，由于国际标准对一标准视力（5.0）的规定是 5 倍的最小可分辨视角，所以公式需要乘以 5。

在实际调整图标大小的时候，还需要结合手机屏幕的尺寸大小，对图片的像素大小进行计算，然后，再按比例进行图像像素的设置，保证图标的实际大小和实际测试的效果相近。

5.1.4 敏感度测试模块

敏感度测试页面如下：

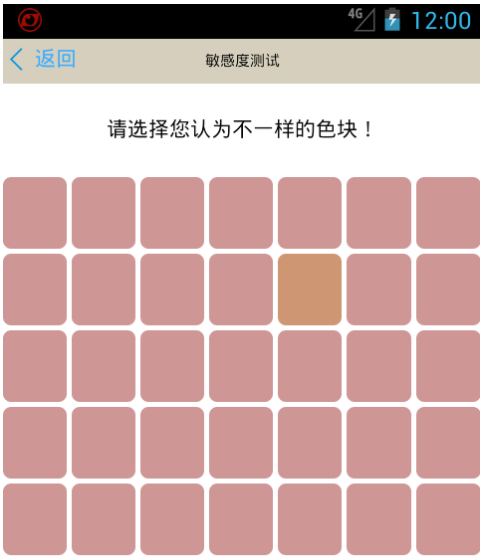


图 5-5

敏感度测试主要采取的原理是：利用程序动态生成 16 进制相近的颜色块，用户需要选择颜色不一样的颜色块。在测试的题目中， 难度会依次地递增，颜色块的颜色会越来越接近，具体的核心算法如下：

```
public static List<TestSensitivityQuestionVO>
    defaultCreateQuestions(){
     int defaultQuestionNum = 10;
     int maxDifficulty = 35;
     int minDifficulty = 3;
     int gradient = (maxDifficulty - minDifficulty) /
                (defaultQuestionNum - 1);
     List<TestSensitivityQuestionVO> questions = new ArrayList<>();
     for(int i=0;i<defaultQuestionNum;i++){
          questions.add(createQuestion(maxDifficulty));
          maxDifficulty-=gradient;
     }
     return questions;
}
```

在色块生成中，默认问题个数是 10，最大难度数为 50，最小为 10，每个题目的难度系数计算公式如下：

$$题目难度系数 = \frac{最大难度系数 - 最小难度系数}{题目个数} * 题目题目序号$$

图 5-6

另外，在生成颜色相关数据之后，关于哪个按钮显示正确的颜色，我们也采取简单的随机生成的策略，算法实现不难，代码省略。

5.2 知识库模块

5.2.1 饮食习惯模块

饮食习惯界面图如下：

图 5-7

界面的数据查询流程和色盲测试的测试数据获取流程基本一致，这

里不详细展开。这个页面主要实现的难点是下拉刷新以及上拉动态加载更多这个过程，下面详细展开这 两部分的实现过程。

总的来说，展示的文章列表利用 Android 动态展示组件 ListView，这个组件支持动态地添加、展示数据。在动态加载数据过程中，主要采用了 SwipeRefreshLayout 这个安卓组件。当然，由于需要判断下拉的距离，所以我继承了该类并进行了相应方法的重写过程，下拉距离判断的核心代码如下：

```
public boolean dispatchTouchEvent(MotionEvent ev) {
    switch (ev.getAction()) {
        case MotionEvent.ACTION_DOWN:
            // 移动的起点
            mDownY = ev.getY();
            break;
        case MotionEvent.ACTION_MOVE:
            // 移动过程中判断时候能下拉加载更多
            if (canLoadMore()) {
                // 加载数据
                Log.d(TAG, "------->  加载更多数据");
                loadData();
            }
            break;
        case MotionEvent.ACTION_UP:
            // 移动的终点
            mUpY = ev.getY();
            break;
    }
    return super.dispatchTouchEvent(ev);
}
```

上面方法主要获取两个距离：手指按下准备滑动时的初始位置和滑

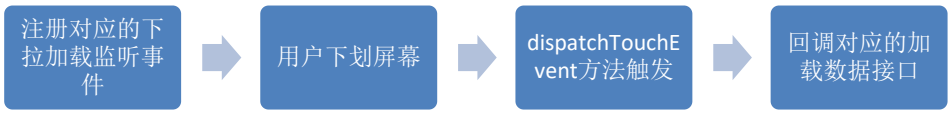
动停止抬起的位置，对应地两个距离会在 canLoadMore 这个方法中判断能否进行动态加载，canLoadMore 核心代码如下：

```
    private boolean canLoadMore() {
        // 1. 是上拉状态
        float dinstance = mDownY - mUpY;
        boolean condition1 = dinstance >= mScaledTouchSlop;
        if (condition1) {
            Log.d(TAG, "------->   是上拉状态");
        }

        // 2. 当前页面可见的 item 是最后一个条目,一般最后一个条目
        //位置需要大于第一页的数据长度
        boolean condition2 = false;
        if (mListView != null && mListView.getAdapter() != null) {
            if (mItemCount > 0) {
                int adaptorCount = mListView.getAdapter().getCount();
                if (adaptorCount < mItemCount) {
                    // 第一页未满，禁止下拉
                    condition2 = false;
                }else {
                    int vPostion = mListView.getLastVisiblePosition();
                    int cItem = mListView.getAdapter().getCount();
                    condition2 = (vPostion >= cItem - 1);
                }
            } else {
            // 未设置数据长度，则默认第一页数据不满时也可以上拉
                condition2 = mListView.getLastVisiblePosition() ==
(mListView.getAdapter().getCount() - 1);
            }
```

```
        }

        if (condition2) {
            Log.d(TAG, "------->  是最后一个条目");
        }
        // 3. 正在加载状态
        boolean condition3 = !isLoading;
        if (condition3) {
            Log.d(TAG, "------->  不是正在加载状态");
        }
        return condition1 && condition2 && condition3;
    }
```

该方法中，通过判断手指按下和抬起的位置判断用户是否要进行下拉加载更多数据的过程。当然，能否进行加载更多数据，处理用户手指滑动屏幕位置距离外，还有另外两个条件：当前是否是该页的底端以及是否正在加载数据中，前者防止用户下拉数据时也对数据进行加载，后者避免层面加载数据。

总的来说，动态加载的流程图如下：



在实现该模块过程，除了实现下拉加载更多这个功能外，另一个关键功能便是下拉刷新的功能。在下拉刷新这个功能中，借助了 Google 的 Android 组件 SwipeRefreshLayout 组件，核心的接口注册事件代码如下所示[]：

```
    swipeRefreshLayout.setOnRefreshListener(new
SwipeRefreshLayout.OnRefreshListener() {
        @Override
```

```
        public void onRefresh() {
            new Handler().postDelayed(new Runnable() {
                @Override
                public void run() {
                    refresh();
                    swipeRefreshLayout.setRefreshing(false);
                }
            }, 0);
        }
    });
```

该组件实现下拉刷新是一个异步的过程，主要是新开一个线程任务进行后台数据加载，第二个参数代表是延迟多久加载数据，0 代表立即刷新加载数据。在 refresh 刷新方法中，刷新完成会有一个通知 ListView 刷新数据过程，refresh 核心代码如下：

```
    private void refresh(){
        switch (paperType) {
            case PAPER_TYPE_DEFAULT :
                //清空数据
                clearPaperList();
                //重新初始化
                initPapers();
                //刷新页面
                listViewAdaptor.notifyDataSetChanged();
                //回到顶部
                papersList.setSelectionAfterHeaderView();
                break;
            case PAPER_TYPE_BLOG :
                changePaperType(PAPER_TYPE_BLOG);
                break;
            case PAPER_TYPE_LECTURE :
```

```
                changePaperType(PAPER_TYPE_LECTURE);
                break;
            case PAPER_TYPE_EATINGHABIT :
                changePaperType(PAPER_TYPE_EATINGHABIT);
        }
    }
```

红色字样是核心代码，它保证了 ListView 在加载数据完成之后对页面进行刷新操作，把加载完成的数据呈现出来。总的来说，下拉刷新的逻辑流程大概如下：

5.2.2 提问模块

用户根据自己的测试结果、用眼疑惑进行提问操作，吧问题提交到云端，相应地会有专家对用户的问题进行解答。主要的截图如下：

图 5-8

5.3 用眼数据

用眼数据界面效果图如下:

图 5-9

该功能主要实现的是对用户使用手机的数据记录：例如开屏次数、当天使用手机时长和手机待机时长等数据，这些数据主要存储在本地的数据库中，每天进行更新。下面详细介绍该部分功能实现的原理细节。

在实现客户使用手机相关数据统计过程，采用了一个定时器进行定时地计时和监听用户行为，定时器核心代码如下：

```
//设置定时器，定时刷新数据
if(timerTask==null){
    timerTask = new TimerTask() {
        @Override
```

```
            public void run() {
                //判断日期是否是今天，防止和昨天重复计算
                if(!RegularUtil.dateIsSame(eyedata.getDate(),new Date())){
                    //如果已经过了一天,更新数据库并重新初始化 eyedata
                    updateEyedata();
                }

        eyedata.setOpenScreenTimeCountRecent(eyedata.getOpenScreenTimeCo
            untRecent() + 1);
        eyedata.
            setOpenScreenTimeCountToday(eyedata.
                    getOpenScreenTimeCou   ntToday() + 1);
                handler.post(refreshUI);//刷新界面
                if(timerCounter>=AUTO_SUBMIT_DATA_TIME){
                    //每隔  AUTO_SUBMIT_DATA_TIME  秒往数据库保存用眼
数据，              //默认是五分钟
                    timerCounter=0;
                    service.updateEyedata(getContext(),eyedata);
                }
            }
        };
    }
    if(timer==null){
        timer = new Timer();
        timer.schedule(timerTask,1000,1000);//每隔一秒更新数据
    }
```

位了提高性能，定时器设置每秒运行一次，进行计时，根据手机当前状态是待机休眠还是用户正在使用来进行分情况计时。另外，定时器每隔五分钟会吧记录的数据插入到数据库当中去。具体的定时器工作流程如下：

在监控用户开屏、锁屏行为中，采取了 Android 本身的一个广播监听组件 BroadcastReceiver，当系统有特定的动作例如锁屏、开屏等全局性事件时，该监听组件可以接受这些时间的通知，具体的代码如下：

```
private class ScreenBroadcastReceiver extends BroadcastReceiver {
    private String action = null;

    @Override
    public void onReceive(Context context, Intent intent) {
        action = intent.getAction();
        if (Intent.ACTION_SCREEN_ON.equals(action)) { // 开屏
            mScreenStateListener.onScreenOn();
        } else if (Intent.ACTION_SCREEN_OFF.equals(action)) { // 锁屏
            mScreenStateListener.onScreenOff();
        } else if (Intent.ACTION_USER_PRESENT.equals(action)) { // 解锁
            mScreenStateListener.onUserPresent();
        }
    }
}
//回调接口
public interface ScreenStateListener {// 返回给调用者屏幕状态信息
    public void onScreenOn();

    public void onScreenOff();

    public void onUserPresent();
}
```

上述的 ScreenBroadcastReceiver 是一个自己实现的内部类，重写了 onReceiver 方法，在接收到广播的时候，根据事件类型进行接口的回调。总的来说，客户端程序只需要实现 ScreenStateListener 接口即可达到监听锁屏开屏的目的，在对应的方法里面实现锁屏、开屏次数的数据库记录操作即可。

具体的调用核心代码如下：

```
creenObserver.setScreenStateListener(new
    ScreenObserver.ScreenStateListener() {
    @Override
    public void onScreenOn() {
        Log.d("开屏事件：","------>开屏");
        //更新用眼数据
        openScreenRefreshEyedata();
    }

    @Override
    public void onScreenOff() {
        Log.d("闭屏事件：","------>关闭屏");
        //更新用眼数据
        closeScreenRefreshEyedata();
    }

    @Override
    public void onUserPresent() {
    //解锁触发的方法
    }
});
```

上述代码在对应的锁屏、开屏回调方法调用了对应的记录方法 openScreenRefreshEyedata 和 closeScreenRefreshEyedata，两个方法会更新数据库、内存中关于开屏、锁屏次数的记录。总的开屏、锁屏统计的实现流程大概如下：

5.4 个人中心、蓝光过滤

该部分是另外一个成员完成，这里不对其详细实现过程进行展开。

第六章 系统总结及展望

6.1 系统研究、开发总结

本次的护眼应用系统开发工作基本完成，所有业务需求均在客户端、服务端实现，通过测试在各个机型均可成功安装、正常运行。最长测试连续运行时间超过 7 天，且请求数据速度、普通功能切换速度较为流畅。

通过这次系统的开发编程工作，掌握熟悉了 Android 平台开发的相关技术，提高了自己的编程技巧和能力，锻炼了产品能力、编程思维。在开发过程通过解决各种程序 bug，搭建项目到投入测试的过程让自己接触到了客户端开发的前沿技术，对软件工程的编程模式、编程思维也有了更深的理解。

6.2 系统展望

总的来说，系统相比于软件市场上的其他护眼应用，功能已较为丰富。但是，软件还可以结合图像识别技术，通过对用户的眼部进行疲劳度识别，迅速、自动化地判断出用户目前的用眼情况，从而更高程度地自动化推荐用户相应的用眼方案和建议，从这方面来说，本系统还有待完善和改进的地方。

当然，我们应该相信，在不久的将来，更为自动化的护眼应用定会被更多的开发人员研究、开发出来，届时，人们的眼睛也会得到更多的保护，科学用眼的观念也会深入人心，目前的眼睛疲劳和眼部疾病问题也定会得到很大程度的缓解。

致谢

在系统开发过程中，很多人对我的开发工作提供了很多帮助。最应该感谢的是我们的指导老师陈长水老师，他对我们的开发工作以及产品功能提出了很多建设性的意见，对于我们的不足和错误，他也以一种鼓励的方式积极引导我们，没有他我们是不可能成功把系统开发出来的。另外，感谢团队其他开发人员，感谢李彦鹏、郑泳智在服务端开发过程中提供了清晰、明了的调用接口，感谢他们对于客户端调用需求的满足；感谢魏奇笙对于客户端开发工作的配合，我们分工开发客户端的过程非常愉快，正是他默契的配合使得我们可以高效、迅速地完成了 Android 客户端的开发工作。

参考文献

[1] 江南时报. "低头族"变"屏奴":每天查看 150 次手机[J]. 江南时报, 2014.

[2] 戴锦晖. 蓝光与眼健康[J]. 中国眼镜科技杂志, 2017(11):94-96.

[3] 孙莹, 林梅, 胡央红,等.青少年近视调查分析及预防对策[J].社区医学杂志.2008, 6(20):60-61.

[4] Statcounter GlobalStats.2017 Press Release.Android overtakes Wind-ows for f-irst time.http://gs.statcounter.com/press/android-overtakes-windows-for-first-ti-me

[5] Martinez, Jennifer. Corrected: Update 2: More mobile phone makers back Google's Android. Reuters (Thomson Reuters). 2008-12-10

[6] Reenskaug, Trygve. THING-MODEL-VIEW-EDITOR: an Example from a planning system (PDF).[2017.04.06].http://heim.ifi.uio.no/~trygver/2007/

MVC_Originals.pdf

[7] Owens, Michael. Chapter 4: SQL. (编) Gilmore, Jason; Thomas, Keir. The Definitive Guide to SQLite. D. Richard Hipp (foreword), Preston Hagar (t-echnical reviewer). Apress. 2006: 133 [30 December 2014]. ISBN 978-1-59059-673-9.

[8]-[9]java 官方网站.甲骨文中国.技术板块. [2018.04.06].java 技术 https://www.oracle.com/cn/java/technologies/index.html

[10] 维基百科.主从式架构字条.https://zh.wikipedia.org/wiki/主从式架构

[11] Github 开源社区.square 开源区域.okhttp 框架.[2018.04.06].https://github.com/square/okhttp

[12] json 技术官网.中文介绍. [2018.04.06].http://www.json.org/json-zh.html

[13] Github 开源社区.alibaba 开源区域.fastjson. [2018.04.06].https://github.com/alib-aba/fastjson

[14] Gradle Build Tool. [2018.04.06].https://gradle.org/.

[15] 谷歌 Android 开发者社区.Android Studio Introduction. [2018.04.06].https://developer.andro-id.com/studio/index.html