**班级**

**学号**

西安电子科技大学

**本科毕业设计论文**



题 目 多选择强隐私的个人博客系统的设计与实现

学 院 计算机学院

专 业 计算机科学与技术

学生姓名 张世欣

学生学号 03121311

导师姓名 黄伯虎老师

目 录

[第一章 绪 论](#_Toc200355145)

[1.1 研究背景及意义](#_Toc200355146)

[1.2 国内外研究现状](#_Toc200355147)

[1.3 本文内容及章节安排](#_Toc200355148)

[第二章 相关技术简介](#_Toc200355150)

[2.1 项目框架概述](#_Toc200355151)

[2.2 安卓MVP架构简介](#_Toc200355155)

[2.3 数据库greenDAO简介](#_Toc200355156)

[2.4 服务端LeanCloud简介](#_Toc200355156)

[2.5 安卓材料设计规范简介](#_Toc200355156)

[2.6 Markdown简介](#_Toc200355156)

[第三章 个人博客系统的设计](#_Toc200355157)

[3.1 系统需求分析](#_Toc200355158)

[3.1.1 需求数据分析](#_Toc200355159)

[3.1.2 需求分析报告](#_Toc200355160)

[3.2 系统概要设计](#_Toc200355158)

[3.2.1 系统流程设计](#_Toc200355159)

[3.2.2 系统功能设计](#_Toc200355160)

[3.2.3 系统架构设计](#_Toc200355160)

[3.3 系统详细设计](#_Toc200355158)

[3.3.1 服务端设计](#_Toc200355159)

[3.3.2 数据库设计](#_Toc200355160)

[3.3.3 客户端设计](#_Toc200355160)

[3.4 界面设计](#_Toc200355158)

[3.4.1 服务端界面设计](#_Toc200355159)

[3.4.1 客户端界面设计](#_Toc200355159)

[第四章 个人博客系统的实现](#_Toc200355157)

[4.1 文章模块的实现](#_Toc200355163)

[4.1.1 编写普通文章的实现](#_Toc200355164)

[4.1.2 MarkDown语法编写文章的实现](#_Toc200355165)

[4.2 待做事项模块的实现](#_Toc200355163)

[4.2.1 编写待做事项的实现](#_Toc200355164)

[4.2.2 待做事项锁屏提醒的实现](#_Toc200355165)

[4.2.3 待做事项定时提醒的实现](#_Toc200355165)

[4.3 推荐模块的实现](#_Toc200355165)

[4.3.1 推荐模块定时更新的实现](#_Toc200355164)

[4.3.2 推荐视频播放的实现](#_Toc200355165)

[4.4 云同步模块的实现](#_Toc200355165)

[4.4.1 本地文章、待做上传服务器的实现](#_Toc200355164)

[4.4.2 服务器数据同步到本地的实现](#_Toc200355165)

[4.5 导入导出模块的实现](#_Toc200355165)

[4.5.1 文章、待做导出为文件的实现](#_Toc200355164)

[4.5.2 本地指定文件导入系统的实现](#_Toc200355165)

[4.5 设置模块的实现](#_Toc200355165)

[4.5.1 文件保存相关设置的实现](#_Toc200355164)

[4.5.2 同步相关设置的实现](#_Toc200355165)

[4.5.1 安全相关设置的实现](#_Toc200355164)

[4.5.2 提醒相关设置的实现](#_Toc200355165)

[4.6 数据库的实现](#_Toc200355165)

[4.6.1 数据库、相关表格的创建](#_Toc200355164)

[4.6.2 数据库升级及迁移的实现](#_Toc200355165)

[4.7 服务端的实现](#_Toc200355165)

[4.7.1 服务端接口的实现](#_Toc200355164)

[4.7.2 服务端数据库的实现](#_Toc200355165)

[第五章 个人博客系统的测试](#_Toc200355157)

[5.1 测试用例的设计](#_Toc200355168)

[5.2 测试情况及结果分析](#_Toc200355168)

[5.3 问题及解决方案](#_Toc200355169)

[第六章 总结](#_Toc200355170)[与展望](#_Toc200355170)

[6.1 总 结](#_Toc200355171)

[6.2 展 望](#_Toc200355172)

[致 谢](#_Toc200355173)

[参 考 文 献](#_Toc200355174)

1. 绪论

1.1 研究背景及意义

1. 相关技术简介

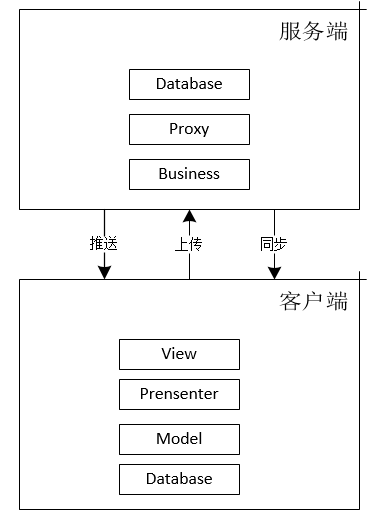
本章主要介绍本课题项目的框架结构、项目中Android端使用的架构 - MVP架构的简介、项目中Android端使用的数据库ORM（对象关系映射）框架 - GreenDao的简介、 项目中服务端采用的BAAS（后端即服务）平台 - LeanCloud的简介、项目中Android端采用的Google官方UI设计规范 - MaterialDesign简介、项目中Android端提供的编辑方式之一Markdown简介。

2.1 项目框架概述

本项目主要采用Client/Server架构，即客户端/服务端架构，通过合理分配客户端和服务端的功能，可以使系统耦合度更低、通讯成本降低。客户端负责收集用户行为、保存用户创建内容、展示推荐内容等，服务端负责持久化保存用户信息和用户内容，为项目实现信息持久化、跨平台同步提供可能。

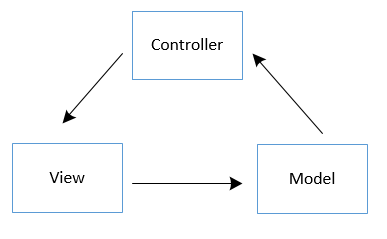
服务端主要由Business层（业务层）、Proxy层（代理层）和Database层（数据库）组成。业务层主要负责处理业务，比如用户注册、登录、文章上传备份、推荐内容更新等；代理层的作用是层隔离，隔断业务层与底层数据库层，这样任何一个层的修改都不会影响到其他层，同时还可以对业务层的操作做一些过滤，避免直接操作数据库导致问题。

客户端主要由View层、Presenter层、Model层、Database层组成，通过使用MVP架构来解耦界面与数据，具体细节下节介绍。



2.2 安卓MVP架构简介

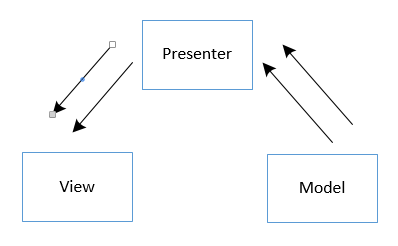
本项目中安卓客户端采用当前比较流行的MVP(Model-View-Presenter)框架。介绍MVP之前要先了解下之前被广泛使用的MVC(Model-View-Controller)架构，Model指的是业务模型，View是指用户操作的界面，Controller是指控制器。使用MVC架构的目的是为了将Model和View实现代码分离，从而使程序的耦合性更低。



然而在过去的使用过程中，太多的程序中Model被理解成提供View要展示的数据的实体类，而忽略了Model更重要的功能：处理业务逻辑。这样处理业务逻辑的代码就都放到了Controller(也就是Activity、Fragment)中，导致Activity、Fragment最后代码十分臃肿，维护困难，不利于重用。Model其实是MVC中非常重要的一部分，它提供展示的数据，同时负责实现业务逻辑。而Controller仅仅起到一个“桥梁”的作用就好了，它负责把View层接收到的用户行为转发给Model，然后再把Model的处理结果通知给View显示。所以Controller是用来解耦用户界面与业务逻辑的，准确的说就是让UI与逻辑隔离开。

为了解决MVC中Controller臃肿的问题，出现了MVP架构。在MVP中，Presenter完全把Model和View隔离开来，两者间没有直接交互。主要业务逻辑在Presenter中实现。MVP中的View代表的是将界面抽象出的接口。Presenter不操作具体的View，而是View的实现类，这样可以实现View有修改时Presenter几乎不收到影响，同时任何实现了该接口的界面都可以复用对应的Presenter。除此外，我们还可以针对View写对应的测试用例，模拟用户的行为，从而测试Presenter，即单元测试。

MVP真正意义上做到了View与业务逻辑的解耦。这样的好处是可以把整个的、复杂的项目拆分成多个模块，每个模块间相互独立，及时修改、替换其中一个模块，其他模块也不会受到影响，这样就实现了项目的可拓展、可维护性。每个模块实现特定的功能，内部聚合的同时也方便了单独测试。这样拆分后可以便于分工合作，原来需要一个人做六个月的工作量，采用MVP架构后可以交给六个人来做，一个月的时间就能做完。模块划分地够清晰后，每个人只需要专注于实现自己的那部分，完成后进行模块对接就好，这对于大项目，或者工期紧的项目是非常重要的。不过还要注意掌握好度，对于十分简单的功能，如果也分的稀碎，就有点过度设计之嫌了。解耦另一方面提高了程序的维护性，即容易区分边界，一旦出了问题，能够立刻定位是哪个模块出了问题。然后让相关责任人去修改。



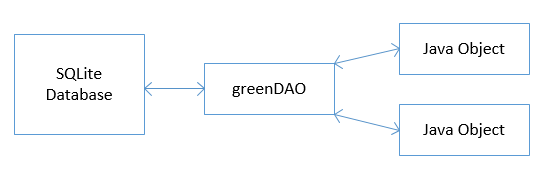
2.3 数据库greenDAO简介

本项目中数据库操作使用的是greenDAO。greenDAO是一个为Android开发者提供便捷的、有效率的SQLite操作数据接口的开源框架，使用它后开发者就不需要与底层数据事务打交道，可以节省很多开发时间。虽然SQLite是一个优秀的轻量级安卓内嵌关系型数据库，然而写SQL语句、解析查询结果还是很费时间。greenDAO通过提供Java对象与数据库中表的映射关系（通常称为关系对象模型）让你免于这些操作。这样你就可以使用面向对象接口来进行Java对象的增删改查。

greenDAO有以下几点特性：

1. 性能最优化(可能是Android最快的ORM框架)；
2. 数据的覆盖关系、加入接口使用方便，同时功能也很强大；
3. 耗费内存极少；

(4)本身体积小，让你编译地更快，同时也能避免“65k方法溢出”。



2.4 服务端LeanCloud简介

如今市场上有很多BaaS服务平台(后端即服务:Backend as a Service)，为移动应用开发者提供了很多功能的SDK，其中包括数据统计分析、推送、支付、定位、数据存储以及托管环境等服务。BaaS作为应用开发的省时省力助手，近一步促进了移动应用开发的成本，推动了应用市场的繁荣。

本项目使用的LeanCloud的存储功能就是一种为开发者提供便捷高效的数据存储服务的。它配合 Node.js 云端服务器环境，为开发者提供一站式的解决方案。

我只要把我的代码部署到LeanCloud的云端来执行，不用自己筹备服务器，同时也不必担心流量跟压力，可以轻松应对多种需求，LeanCloud提供了很成熟的命令行工具，部署、发布、回滚都十分方便。客户端调用时也很方便，只需要满足JSONObject这一基本格式就可轻松访问。

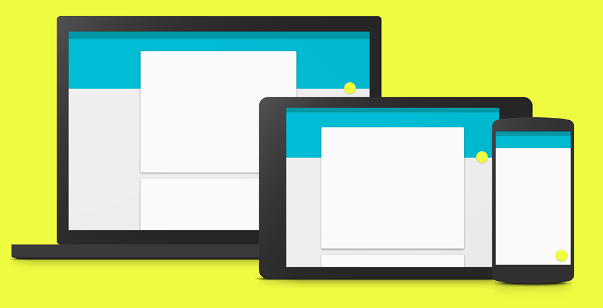
除此外LeanCloud还有以下特点：

1. 支持定时任务，beforeSave 等 Hook 调用；
2. 支持在线编辑，本地部署，方便调试；
3. 提供离线数据分析：使用 SQL 语句分析应用数据；
4. 文件存储采用国内全网加速 CDN，支持多种类型文件；
5. 全平台 SDK 支持：iOS、Android、Windows Phone、Unity 3D、JavaScript等。

2.5 安卓材料设计规范简介

本项目中Android端的设计规范是采用的Google官方UI设计规范 - Material design(又名材料设计)。Material design的核心思想就是把我们在物理世界得到的体验带进屏幕上去，保留物理世界中最原始纯净的形态、空间关系与过渡动画效果，同时需要去掉现实中的杂质和不确定性，然后配合虚拟世界的灵活特性，为用户创造最贴近真实感受的体验，达到简洁与美观的体验。

Material design的设计目标是让应用在不同平台、不同屏幕上都有一致的体验。因此它的规范十分严格细致，保证它在各个平台上使用体验高度一致。



本项目中用到的Material design的特性主要有：

1. 风格：限制主色调数量、图片规格、文字尺寸等；
2. 布局：凸出与省略、三维层级合理分布、适当运用阴影等；
3. 动画：模拟真实动画，考虑材质、加速度等，过渡动画的优美使用等；
4. 组件：合理使用材料风格的卡片、按钮、列表、网格、开关、提示和对话框等。

2.6 Markdown简介

本项目客户端端提供了一种简洁优雅的编辑方式 - Markdown。

http://www.chinaz.com/manage/2015/0413/397997.shtml需求分析