**四川师范大学本科毕业设计**

|  |
| --- |
| 基于iOS的美妆电商类APP实现 |

|  |  |
| --- | --- |
| **学生姓名** | **＊＊＊** |
| **院系名称** | **计算机科学学院** |
| **专业名称** | **软件工程** |
| **班 级** | **2＊＊＊级＊班** |
| **学 号** | **＊＊＊** |
| **指导教师** | **李贵洋** |
| **完成时间** | **2016年5月6日** |

**基于iOS的美妆电商类APP实现**

学生：＊＊＊ 指导教师：李贵洋

内容摘要：此为内容摘要，大概300字

关键词：**iOS开发 移动智能客户端 电子商务**

**THE IMPLEMENTATION OF ELECTRICITY BUSINESS APPLICATION FOR IOS**

**Abstract:** say something lalalalala

**Key Words: iOS development Mobile intelligent terminal Electricity business**

# 目录

1. 概述 ………………………………………………………………………………
   1. 研究的目的和意义 ………………………………………………………
   2. 研究的背景 ………………………………………………………………
      1. 国内现状 …………………………………………………………
      2. 国外现状 …………………………………………………………
   3. 主要贡献 …………………………………………………………………
   4. 文章的结构 ………………………………………………………………

2. 预备知识及原理说明 ……………………………………………………………

2.1 Cocoa Touch …………………………………………………………………

2.2开发语言及开发工具 ………………………………………………

2.3设计开发模式MVC …………………………………………

2.4 CocoaPods

2.4 iOS 第三方库 ………………………………………………………………

3. 基于iOS的电商美妆app的系统分析 ………………………………………….

4．基于iOS的电商美妆app的设计和实现 ………………………………………

5. 基于iOS的电商美妆app系统测试和运行 ……………………………………

6. 成果与展望 ………………………………………………………………………

7. 致谢 ………………………………………………………………………………

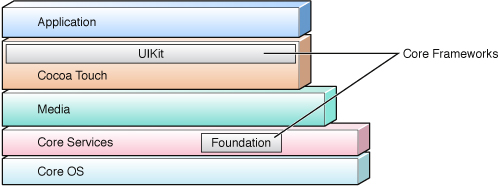
参考文献 ………………………………………………………………………………

**基于＊＊＊＊＊**

# 1 概述

# 2预备知识及原理说明

**2.1 Cocoa Touch**

****

**iOS 系统架构**

Cocoa不是一种编程语言（它可以运行多种编程语言），它也不是一个开发工具（通过命令行我们仍然可以开发Cocoa程序），它是创建Mac OS X和IOS程序的原生面向对象API，为这两者应用提供了编程环境。

Cocoa Touch 是针对于iOS的应用开发环境，它包括Objective-C 运行环境和两个核心框架：foundation框架和ApplicationKit（UIKit）框架。框架的功能类似于动态库，既可以在运行时动态的载入应用程序的地址空间，但框架作为一个对计算机的捆绑，而非独立文件，除了可执行代码外，也包含了资源，头文件和文档。

foundation框架作为通用的面向对象函数库，主要定义了一些基础类，提供了字符串，数值的管理，容器及其枚举，以及一些其他的与图形用户界面没有直接关系的功能；UIKit框架提供了在屏幕上绘制的机制，捕获事件，和创建通用用户界面元素。UIKit也通过管理显示在屏幕上的组件来组织复杂的项目。使用UIKit可以：构建和管理你的用户界面，捕获触摸和基于移动的事件，呈现文字和web内容，优化你的多任务程序，创建定制的用户界面元素。

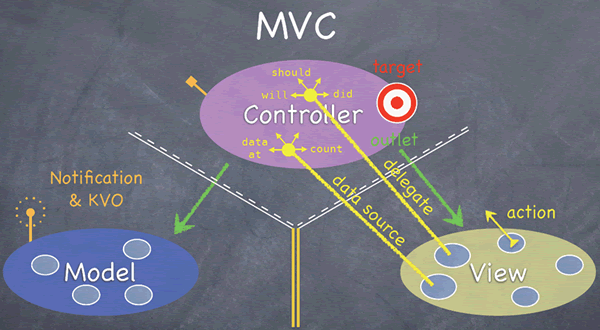
**2.2 开发语言及开发工具**

20世纪八十年代初，Brad Cox在其公司Stepstone发明Objective-C，它以一种叫做SmallTalk-80的语言为基础。Objective-C建立在C语言之上，意味着它是在C语言基础上添加了扩展而创造出来的能够创建和操作对象的一门新的程序设计语言。对Objective-C最主要的描述是他1986年出版的《 Object-oriented Programming, An Evolutionary Approach》。1988年，NeXT Software公司获得了Objective-C语言的授权，并开发出了Objective-C的语言库和一个名为NEXTSTEP的开发环境。1992年，自由软件基金会的GNU开发环境增加了对Objective-C的支持。1994年，NeXT Computer公司和升阳微系统（Sun Microsystem）联合发布了一个针对 NEXTSTEP系统的标准典范，名为OPENSTEP。OPENSTEP在自由软件基金会的实现名称为GNUstep。1996年12月20日，苹果公司宣布收购NeXT Software公司，NEXTSTEP/OPENSTEP环境成为苹果操作系统下一个主要发行版本OS X的基础。这个开发环境的该版本被苹果公司称为Cocoa。（互动百科）

苹果公司于2008年3月6日发布了iPhone和iPod touch的应用程序开发包,其中包括Xcode开发工具、iPhone SDK 和iPhone手机模拟器。第一个Beta版本是iPhone SDK 1.2b1(build 5A147p),发布后立 能使用, 是同时 出 的App Store所需要的 件更新 到2008年7月11日 发布。编写本书时,iOS SDK 7.0.4版本 经发布。  iOS开发工具主要是Xcode。自从Xcode 3.1发布以后,Xcode 成为iPhone 件开发工具包的开发 。Xcode 可以开发Mac OS X和iOS应用程序,其版本是与SDK相 对应的。例如,Xcode 3.2.5与iOS SDK 4.2对应,Xcode 4.1 与iOS SDK 4.3对应,Xcode 4.2与iOS SDK 5对应,Xcode 4.5和Xcode 4.6与iOS SDK 6对应,Xcode 5与iOS SDK 7 对应。  在Xcode 4.1之前,还有一个配 使用的工具Interface Builder, 是Xcode 件的一部分,用来设计 体和视 图,通过 可以 所见 所得 地 控件并定 事件等,其数据以XML的形式 存储在xib文件中。在Xcode 4.1 之后, Interface Builder成为了Xcode的一部分,与Xcode集成在一 。 （iOS开发指南：从零基础到App Store上架（第2版 ）

**2.3设计开发模式MVC**

MVC，全称是 Model View Controller，是模型 (model)－视图 (view)－控制器 (controller) 的缩写。它表示的是一种常见的客户端软件开发框架。MVC 的概念最早出现在二十世纪八十年代的施乐帕克实验室中，当时施乐帕克为 Smalltalk 发明了这种软件设计模式。现在，MVC 已经成为主流的客户端编程框架，在 iOS 开发中，系统为我们实现好了公共的视图类：UIView，和控制器类：UIViewController。大多数时候，我们都需要继承这些类来实现我们的程序逻辑，因此，我们几乎逃避不开 MVC 这种设计模式。



iOS系统下的MVC关系图

Paul Hegarty，斯坦福大学公开课：iOS 7应用开发[EB/OL]. 网易公开课.2014.

Controller能够访问Model和View，Model和View不能互相访问；当View与用户交互产生事件时，使用target-action方式来处理；当View需要处理一些特殊UI逻辑或获取数据源时，通过delegate或data source方式交给Controller来处理；Model不能直接与Controller通信，当Model有数据更新时，可以通过Notification或KVO (Key Value Observing)来通知Controller更新View。

另外由MVC还衍生出了一些其他的架构：MVVM，MVCS，VIPER等。

MVVM是基于胖model的架构思路建立，在胖model中拆出两部分：model和viewModel，viewModel让Controller只需关注于数据调配，它去负责数据加工并通过通知机制让view响应viewModel的改变，总之不管是胖model还是viewModel都是给controller减负。

（PS：胖model：即包含了部分业务逻辑，controller从其拿到数据之后不用做额外操作或很少操作就能使用，瘦model：只负责数据的表达，然后配套helper类或方法来对弱业务做抽象，其他一律扔给controller。独立性强。但一定程度上违背了DRY，即Don’t repeat yourself）

MVCS是从controller中把负责数据存储部分提取出来，交给另外一个对象去做，这个对象就是store。

VIPER，即视图 (View)，交互器 (Interactor)，展示器 (Presenter)，实体 (Entity) 以及路由 (Routing)。将一个应用程序的逻辑结构划分为不同的责任层。这使得它更容易隔离依赖项 (如数据库)，也更容易测试各层间的边界处的交互。

**2.4 iOS 第三方库**

# 3 系统分析

## 3.1 问题定义

**3.2 可行性研究**

**3.3 需求分析**

**系统需求如下：**

1. **用户通过第三方登录或注册，以便创建以及定制自己的信息数据**
2. **创建出门前（等其它）检查列表，以及修改删除**
3. **通过图库或拍照获取图样创建物品语音备忘，以及更新和删除**

**3.4 概要设计**

**3.5 详细设计**

**3.6**