基于iOS的物流监控系统移动平台的设计与实现

**摘要**

**随着手机用户规模的扩大和智能手机用户比例的提高，手机业务的发展有了庞大的用户**

**群基础。移动互联网的快速发展，带动着移动互联网应用的不断创新，通过计算机、互联网、手机等物品的自动识别和信息的互联与共享，提高了生产和管理的效率。现代物流是移动互联网极具现实意义的应用领域之一，积极探索移动互联网在物流领域的应用具有重要的现实意义。**

**本课题研究了基于iOS的物流监控系统移动平台的设计与实现。系统分为远程数据监控、iOS客户端以及简单的服务器管理员Web界面，分别在Xcode和IntelliJ IDEA平台上开发。通过使用HTTP最新的RESTful API交互技术，实现了iOS客户端远程访问服务端并操作数据库的功能。RESTful API交互是整个系统的“神经中枢”，通过频繁地API交互，达到实时同步快递员信息到服务器的功能，主要功能有快件位置监控、快件状态监控，快件收揽派件信息验证。iOS客户端，使得快递员可以通过手机客户端扫描二维码的方式快速揽件，并在派件时接受到服务器发来的位置错误警告，而收件者也可以实时通过iOS客户端同步到服务器的精确GPS位置信息监控到自己的快件位置和预计到达时间。本系统充分利用现代信息技术，特别是移动互联网有关技术，对传统物流业务进行优化整合，以达到降低物流成本、提高物流服务水平的目的。**

**关键字:** 移动互联网、iOS、物流监控、物流管理、物流信息化

目录

[第一章 绪论 2](#_Toc482781783)

[1.1背景和意义 2](#_Toc482781784)

[1.2物流信息化国内外研究现状 3](#_Toc482781785)

[1.2.1国外研究现状分析 3](#_Toc482781786)

[1.2.2国内研究现状分析 3](#_Toc482781787)

[1.2.3物流信息化的优势 3](#_Toc482781788)

[1.2.4物流信息化发展趋势 4](#_Toc482781789)

[第二章 开发环境和相关技术介绍 4](#_Toc482781790)

[2.1 iOS简介（本APP的开发平台） 4](#_Toc482781791)

[2.1.1 iOS操作系统介绍 4](#_Toc482781792)

[2.1.2 iOS操作系统架构 4](#_Toc482781793)

[2.1.3 iOS开发工具Xcode 7](#_Toc482781794)

[2.1.4 iOS 开发语言Swift 7](#_Toc482781795)

[2.1.5CocoaPod概述 8](#_Toc482781796)

[2.2 RESTful API 概述（APP与服务器进行交互的技术） 8](#_Toc482781797)

[2.2.1RESTful API 概念介绍 8](#_Toc482781798)

[2.2.2RESTful 架构定义 9](#_Toc482781799)

[2.3 JHipster概述（APP所需的服务器端技术） 10](#_Toc482781800)

[2.2.1 JHipster概念介绍 10](#_Toc482781801)

[2.2.2JHipster后端应用 Spring boot 11](#_Toc482781802)

[第三章 需求分析与设计 18](#_Toc482781803)

[3.1实体-关系图 18](#_Toc482781804)

[3.2状态图 19](#_Toc482781805)

[3.2.1 总状态 19](#_Toc482781806)

[3.2.2收揽货物 20](#_Toc482781807)

[3.2.3派件中 20](#_Toc482781808)

[3.3服务器端数据库ER图 22](#_Toc482781809)

[第四章 实现 22](#_Toc482781810)

[4.1工作流程的状态切换 22](#_Toc482781811)

[4.2保存状态，保证手机重启之后软件的可恢复性 24](#_Toc482781812)

[4.3 与服务器的信息交互 25](#_Toc482781813)

[4.4 保存时的错误处理与警告 26](#_Toc482781814)

[4.5信息同步到服务器上 27](#_Toc482781815)

[4.6QR码读取 28](#_Toc482781816)

[第五章 测试 29](#_Toc482781817)

[第六章 总结 29](#_Toc482781818)

# 第一章 绪论

## 1.1背景和意义

随着信息技术的不断发展，互联网的不断普及，使用网络购物的人数越来越多。因此，人们对物流依赖程度越来越高。快捷、安全性快递服务凭借先进的计算机网络、先进的通讯网络及最快的运输工具将商业函件和包裹送到世界各个角落，快捷的需求使快递公司必须具备快速运输工具和负责集散货物、分发、派送的方法，利用科技实现快递物品的门到门或桌到桌服务，必须及时、准确地接受到顾客信息，快速处理订单，实现对快件的全程跟踪，随时提供查询服务等。为了实现这些功能，快递公司必须配备先进的计算机网络、先进的通讯网络以及能够满足和实现各种个性化的、特殊的快递服务需求的特有软件和硬件。UPS在电子技术方面共投资了110亿美元，实现了对包裹运送的每一步紧密跟踪，从而使其在电子商务大潮中占据了有利地位。由于快递业的服务表现形式是桌到桌、门到门，它需要对快件实行全程跟踪，对于客户则希望能够实时查询快件的情况，所以快递公司必须在优质服务的基础上才能留住客户。

配送服务是物流活动的关键环节之一。智能城市交通的物流监控及调度平台可以将物流信息共享，从而减少成本，最大化地提升物流运输配送的效率， 帮助物流企业实现更多的营业收入。同时，目前的物流监控及调度平台无法基于位置、实时地、动态地调度。

所以本课题打算做这样一个设计，通过现在每个手机自带的定位系统，实时定位快递员在物流派送时的位置信息。通过近几年兴起的好用的二维码技术，让快递员实现一键揽货。再使用智能城市交通的物流监控及调度平台将物流信息实时共享给用户，并同时通过物流监控及调度平台将对快递员位置的需求等一系列信息实时返回给快递员。这样既能改善用户对快递的体验，也能提高快递员的效率。

## 1.2物流信息化国内外研究现状

### 1.2.1国外研究现状分析

根据柳贺在第一情报(信息产业)中的描述，美国作为物流理念的发源地，其物流研究、设计和技术开发一直处于世界前沿，有着十分成熟的物流管理经验，特别是商贸流通和生产制造企业十分重视现代物流的开发。从20世纪50年代物流发展初期的“实物配送”阶段，到80年代的“物流”阶段，再到今天的供应链管理阶段，一直将物流战略作为企业战略的核心组成部分，并予以高度重视。美国企业纷纷将物流信息化作为物流合理化的一个重要途径：1、普遍采用条形码技术和射频识别技术，提高信息采集效率和准确性；采用基于互联网的电子数据交换技术进行企业内外的信息传输，实现订单录入、处理、跟踪、结算等业务处理的无纸化。2、通过与供应商和客户的信息共享，实现供应链的透明化，实现供应链伙伴之间的协同商务，以便“用信息代替库存”，降低供应链的物流总成本，提高供应链的竞争力。3、借助网上采购辅助材料、网上销售多余库存等电子商务手段来降低物流成本。

日本的物流概念于20世纪50年代从美国引进，随后发展非常迅速，物流现代化和生产现代化是日本战后经济发展的两个车轮。无论是政府对物流的重视程度、企业对物流的管理方面，还是物流基础设施、现代化物流发展水平方面，其水平均不亚于欧美。日本己成为现代物流管理的先进国家，主要特点表现在以下几个方面：1、先进的电子信息技术加快物流现代化的进程。几乎所有的专业物流企业都是通过计算机信息管理系统来处理和控制物流信息；2、发达的交通运输业是物流业的强大支柱；3、具有国际领先水平的物流基础设施；4、高效的企业管理和多样化的服务内容；5、具有健全的政策保障。近年来，日本政府又积极调整物流发展战略，倡导高附加值物流，并将物流信息技术作为重点发展对象。

### 1.2.2国内研究现状分析

目前我国物流业整体发展水平还比较低，物流费用在GDP中的比重相对很高，物流行业信息化应用的整体水平并不高，信息系统的业务功能不完善，远程通信能力低，缺乏必要的决策能力。相比之下，欧美发达国家的物流费用一般占该国GDP比重则较低，空载率也较低。国内专家通过对美国和欧洲一些发达国家考察发现：在差不多十几年前，美国公路的空驶率在20％左右，后来降到了10％以下，其中的关键就在于他们应用了信息管理技术，尤其是ERP技术、GIS技术、GPS技术和通信技术。而在我国有超过60％的企业还处在单机应用阶段、基础网络建设和系统建设阶段。在国外物流企业得到广泛使用的条码技术、RFID、GPS／GIS和EDI技术在中国物流企业的应用还不够理想。

当前我国物流业的发展和物流信息化市场正进入一个加速发展的时期。其基本特点为：物流管理软件将趋于更加专业化、信息化意识正在逐步提高、建设步伐加快。相关调查显示，我们大中型企业物流及第三方物流企业信息化意识普遍提高，大约有74％的企业已经建立了信息管理系统，77％的企业已有自己的网站。物流企业对现代通信技术的接受程度逐渐提高，开始积极采用GPS、GIS等先进技术来提高企业运营水平和综合实力。

### 1.2.3物流信息化的优势

物流信息化是现代物流发展的灵魂，是现代物流发展的必然要求和基石。正确的物流信息化战略能够促进物流信息化的发展，提高物流效率、降低物流成本、整合优化各类物流信息资源、实现供应链信息的无缝接轨和物流信息的网络化、实时化传输，真正达到物畅其流的效果。

### 1.2.4物流信息化发展趋势

物流系统是一个大跨度系统。物流活动范围、流动速度也进入一个前所未有的发展阶段，物流业正向全球化、网络化和信息化方向发展，EDI技术与互联网的应用，使物流效率的提高更多地取决于信息管理技术。电子计算机的普及和条形码技术的普遍应用，则提供了更多的需求和库存信息，提高了信息管理的科学水平。信息化己成为物流活动的核心，成为物流创新的动力。通过前两个小节的分析可以看出，我国的物流发展是相当有潜力的，我们必须不断创新，紧跟时代脚步，在政府的支持和大家共同努力的基础上，走一条属于中国特色的现代物流信息化发展道路。

# 第二章 开发环境和相关技术介绍

## 2.1 iOS简介（本APP的开发平台）

### 2.1.1 iOS操作系统介绍

iOS是由苹果公司开发的移动操作系统。苹果公司最早于2007年1月9日的Macworld大会上公布这个系统，最初是设计给iPhone使用的，后来陆续套用到iPod touch、iPad以及Apple TV等产品上。iOS与苹果的Mac OS X操作系统一样，属于类Unix的商业操作系统。原本这个系统名为iPhone OS，因为iPad，iPhone，iPod touch都使用iPhone OS，所以2010WWDC大会上宣布改名为iOS（iOS为美国Cisco公司网络设备操作系统注册商标，苹果改名已获得Cisco公司授权）。

iOS用户界面基于直接操作，使用多点触控手势。接口控制元件由滑块，开关和按钮组成。与操作系统的交互包括手势，如滑动，点击，捏和反向夹，所有这些手段都在iOS操作系统及其多点触控界面的上下文中具有特定的定义。一些应用程序使用内部加速度计来响应设备的抖动（一个常见的结果是撤销命令）或三维旋转（一个常见的结果是在纵向和横向模式之间切换）。苹果已被大量赞誉，将完整的辅助功能纳入iOS，使视力和听力障碍用户正确使用其产品。

主要版本的iOS每年都会发布。目前的版本，iOS 10，于2016年9月13日发布。它适用于iPhone 5及更高版本的iPhone型号，第四代iPad，iPad Air和iPad Air 2，iPad Pro，iPad Mini 2和后来的iPad Mini型号，以及第六代iPod Touch。在iOS中，有四个抽象层：Core OS，Core Services，Media和Cocoa Touch图层。

### 2.1.2 iOS操作系统架构

iOS的系统架构分为四个层次：核心操作系统层（Core OS layer）、核心服务层（Core Services layer）、媒体层（Media layer）和可触摸层（Cocoa Touch layer）

1、Core OS是位于iOS系统架构最下面的一层是核心操作系统层，它包括内存管理、文件系统、电源管理以及一些其他的操作系统任务。它可以直接和硬件设备进行交互。作为app开发者不需要与这一层打交道。

2、Core Services是核心服务层，可以通过它来访问iOS的一些服务。

3、Media是媒体层，通过它我们可以在应用程序中使用各种媒体文件，进行音频与视频的录制，图形的绘制，以及制作基础的动画效果。

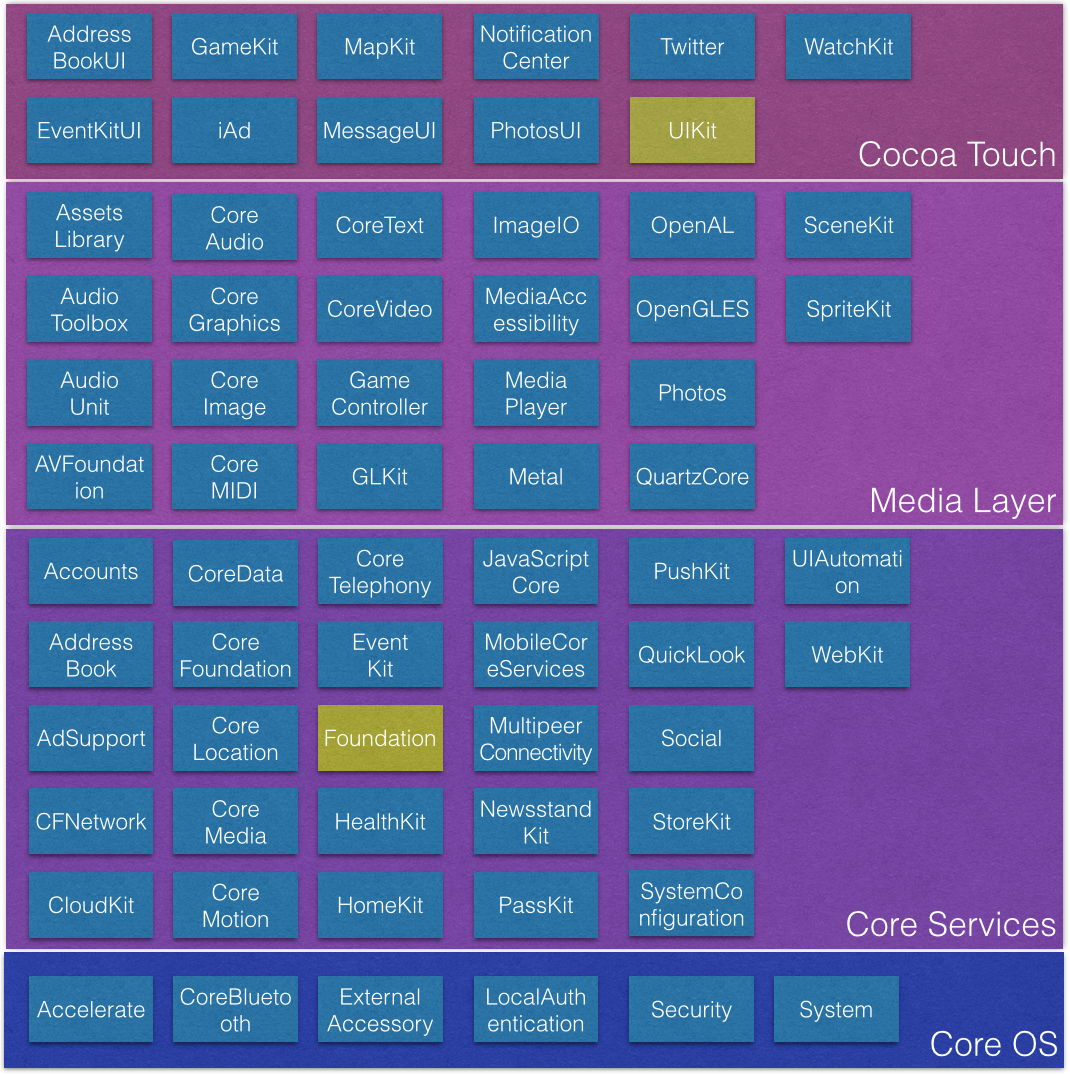
4、Cocoa Touch是可触摸层，这一层为我们的应用程序开发提供了各种有用的框架， 并且大部分与用户界面有关，本质上来说它负责用户在iOS设备上的触摸交互操作。

我们通常称呼iOS的框架为Cocoa Touch框架，Cocoa Touch是一个框架的集合，里面包含了众多的子框架。每一个子框架都是一个目录，包含了共享资源库，用于访问该资源库中储存的代码的头文件，以及图像、声音文件等其他资源，共享资源库定义应用程序可以调用的函数和方法。框架中的类相互依赖构成一个整体，提供完善的某一方面的服务或解决方案，多个框架一起实现整个应用程序的结构。由于应用程序的结构是通用的，开发者通过框架提供的函数和方法，做细致个性化的处理，从而满足不同应用的不同需求。开发一个应用程序就是将需求细致化的代码插入到框架提供的设计中来组合成一个整体完成最终的应用设计。

框架的结构是泛型结构，为应用程序提供一个模板。就像开发商开发好的毛坯房，你需要用应用程序代码来布置不同的家具地板门窗，这才让不同的房屋有不同的格调，不同的幸福。当然，有时候，做装修的时候你需要简单的改装你的房屋，但承重墙是不能改造的。就像我们之前说的，多个框架一起实现了整个应用程序的结构，我们必须接受它所定义好的应用程序结构，设计的时候让我们的应用适合该结构。

在Android开发中，采用模板模式来实现应用程序的一些特性行为，Android提供了Activity,Service,Content providers,Broadcast receivers四大组件默认功能，应用通过继承这些组件根据需要覆盖组件的一些方法来完成应用程序开发。在iOS中则采用代理和协议模式来实现应用的特性行为。例如Cocoa Touch框架集合中的UIKit框架的UIApplication对象，它负责整个应用程序生命周期的事件分发。是应用最核心的一个对象，Android的设计中就需要对其子类化，覆盖父类的方法，iOS中则交给UIApplication的代理AppDeleagte来处理应用程序的各种状态改变相关事件(AppDelegate需要实现UIApplicationDelegate协议) 。在iOS的框架中，大量的使用代理和协议。

iOS提供的许多可使用的框架，构成了iOS操作系统的层次结构，从下到上依次是:Core OS、Core Ssevices、MediaLayer、Cocoa Touch共四层。下图为iOS8.3系统的框架架构图。



iOS8.3系统框架架构图

Core OS Layer, 系统核心层包含大多数低级别接近硬件的功能，它所包含的框架常常被其它框架所使用。Accelerate框架包含数字信号，线性代数，图像处理的接口。针对所有的iOS设备硬件之间的差异做优化，保证写一次代码在所有iOS设备上高效运行。CoreBluetooth框架利用蓝牙和外设交互，包括扫描连接蓝牙设备，保存连接状态，断开连接，获取外设的数据或者给外设传输数据等等。Security框架提供管理证书，公钥和私钥信任策略，keychain,hash认证数字签名等等与安全相关的解决方案。

Core Services Layer, 系统服务层提供给应用所需要的基础的系统服务。如Accounts账户框架，广告框架，数据存储框架，网络连接框架，地理位置框架，运动框架等等。这些服务中的最核心的是CoreFoundation和Foundation框架，定义了所有应用使用的数据类型。CoreFoundation是基于C的一组接口，Foundation是对CoreFoundation的OC封装。

Media Layer, 媒体层提供应用中视听方面的技术，如图形图像相关的CoreGraphics,CoreImage,GLKit,OpenGL ES,CoreText,ImageIO等等。声音技术相关的CoreAudio,OpenAL,AVFoundation,视频相关的CoreMedia,Media Player框架，音视频传输的AirPlay框架等等。

Cocoa Touch Layer, 触摸层提供应用基础的关键技术支持和应用的外观。如NotificationCenter的本地通知和远程推送服务，iAd广告框架，GameKit游戏工具框架，消息UI框架，图片UI框架，地图框架，连接手表框架，自动适配等等

在上面所有的框架中，最重要也最经常使用的就是UIKit和Foundation框架。Foundation框架提供许多基本的对象类和数据类型，使其成为应用程序开发的基础,为所有应用程序提供最基本的系统服务，和界面无关。 UIKit框架提供的类是基础的UI类库，用于创建基于触摸的用户界面，所有 iOS 应用程序都是基于 UIKit，它提供应用程序的基础架构，用于构建用户界面，绘图、处理和用户交互事件，响应手势等等。UIKit通过控制器对象管理屏幕上显示的内容，界面的跳转，来组织应用程序。没有UIKit框架就没有iOS应用程序。

### 2.1.3 iOS开发工具Xcode

Xcode是一个用于macOS的集成开发环境，包含由Apple开发的一套用于开发macOS，iOS，watchOS和tvOS软件的软件开发工具。Xcode 具有统一的用户界面设计，编码、测试、调试都在一个简单的窗口内完成。

The Xcode suite 包含有GNU Compiler Collection自由软件 （GCC、 apple-darwin9-gcc-4.0.1 以及 apple-darwin9-gcc-4.2.1, 默认的是第一个），并支持 C语言、C++、Fortran、Objective-C、Objective-C++、Swift、Java、AppleScript、Python以及Ruby，还提供Cocoa、Carbon以及Java等编程模式。协力厂商更提供了 GNU Pascal，Free Pascal, Ada, C#, Perl, Haskell 和 D语言。Xcode套件使用 GDB作为其后台调试工具。

由于Mach-O可执行格式，允许包含多种体系结构的大二进制文件，Xcode可以构建通用的二进制文件，允许软件在PowerPC和基于Intel（x86）平台上运行，并且可以包括32 位和64位代码。 使用iOS SDK，Xcode也可用于编译和调试在ARM体系结构处理器上运行的iOS应用程序。

Xcode还包含一系列带有GUI的测试监控工具，它运行在由Sun Microsystems创建并作为OpenSolaris一部分发布的动态跟踪框架DTrace之上。

### 2.1.4 iOS 开发语言Swift

Swift是Apple在WWDC2014所发布的一门编程语言，用来撰写OS X和iOS应用程序。在设计Swift时．就有意和Objective-C共存，Objective-C是Apple操作系统在导入Swift前使用的编程语言.

Swift旨在与苹果的Cocoa和Cocoa Touch框架以及为Apple产品编写的大量Objective-C（ObjC）代码。它是使用开源的LLVM编译器框架构建的，自版本6被包含在Xcode中。在Linux之外的平台上，它使用Objective-C运行时库，它允许C，Objective-C，C ++和Swift代码在一个程序中运行[10]

Swift旨在比Objective-C更有弹性的错误代码（“更安全”），更简洁。然而，它支持与Objective-C相关联的许多核心概念;特别是动态调度，广泛的后期绑定，可扩展编程和类似功能。为了安全起见，Swift有助于解决常见的编程错误，如空指针，并提供语法糖，以避免否则可能导致的厄运金字塔。更基本的是，Swift增加了协议可扩展性的概念，这是一种可应用于类型，结构和类的可扩展性系统。苹果宣称这是他们所谓的“面向协议的程序设计”的编程范例的一个真正变化。[11]

Swift在苹果2014年全球开发者大会（WWDC）上推出。[12]它在2014年升级到版本1.2，并在WWDC 2015上升级为Swift 2。虽然刚开始是一个闭源语言，版本2.2在2015年12月3日在Apache许可证2.0下被声明为用于苹果平台和Linux的开源语言。

### 2.1.5CocoaPod概述

CocoaPods是Objective-C，Swift以及在Objective-C运行时运行的任何其他语言的应用程序级别依赖管理器，例如RubyMotion ，提供了管理外部库的标准格式。它由EloyDurán和Fabio Pelosin开发，他们在许多其他人的帮助和贡献下继续管理该项目。他们于2011年8月开始发展，并于2011年9月1日发布了第一个公开发行。CocoaPods受RubyGems和Bundler的Ruby项目的强大启发。

CocoaPods专注于基于源代码的第三方代码分发和自动整合到Xcode项目中。

CocoaPods从命令行运行，并且还集成在JetBrains的AppCode集成开发环境中。它通过依赖性的规定安装依赖关系（例如库），而不是手动复制源文件。除了从许多不同的源安装，许多开放源代码库的“主”规范存储库的元数据被保留为Git存储库，并在GitHub上托管。目前有超过3000个库可供使用。

## 2.2 RESTful API 概述（APP与服务器进行交互的技术）

### 2.2.1RESTful API 概念介绍

表征状态转移（REST）或RESTful Web服务是在因特网上的计算机系统之间提供互操作性的一种方式。符合REST的Web服务允许请求系统使用统一和预定义的一组无状态操作来访问和操作Web资源的文本表示。存在其他形式的Web服务，它们暴露了自己的任意操作集合，如WSDL和SOAP。

“Web资源”首先在万维网上定义为由其URL标识的文档或文件，但是今天他们有更通用和抽象的定义，涵盖可以识别，命名，寻址或处理的任何东西或实体无论如何，在网上。在RESTful Web服务中，向资源URI发出的请求将引发可能是XML，HTML，JSON或其他定义格式的响应。响应可以确认已经对存储的资源进行了一些改变，并且它可以提供与其他相关资源或资源集合的超文本链接。使用HTTP，最常见的是可用的操作类型，包括由HTTP动词GET，POST，PUT，DELETE等预定义的操作。

通过利用无状态协议和标准操作，REST系统旨在通过重新使用可以管理和更新的组件来快速实现性能，可靠性和增长能力，而不影响整个系统，即使它正在运行。

代表性的状态转移转移一词在2000年由Roy Fielding博士论文引入和定义。 Fielding使用REST来设计HTTP 1.1和统一资源标识符（URI） 该术语旨在唤起设计良好的Web应用程序行为的形象：它是Web资源（虚拟状态机）的网络，用户通过选择诸如/ user / tom之类的链接来执行应用程序，并且诸如GET或DELETE（状态转换）的操作导致下一个资源（表示应用的下一个状态）被传送给用户供它们使用。

### 2.2.2RESTful 架构定义

定义RESTful系统有六个指导约束 [10]这些约束限制了服务器可以处理和响应客户端请求的方式，以便通过在这些约束内操作，服务获得所需的非功能属性，如性能，可扩展性，简单性，可修改性，可见性，可移植性和可靠性。 如果服务违反任何所需的约束，则不能被认为是RESTful的。

正式的REST约束如下：

客户端 - 服务器[编辑]

另请参阅：客户端 - 服务器模型

加入我们混合风格的第一个约束是客户端 - 服务器架构风格。担心的分离是客户端 - 服务器约束背后的原则。通过将用户界面问题与数据存储问题分开，我们通过简化服务器组件来提高跨多个平台的用户界面的可移植性，并提高可扩展性。然而，对于Web来说，最重要的是分离可以使组件独立演进，从而支持多个组织领域的互联网规模要求。

无状态[编辑]

另请参见：无状态协议

客户端 - 服务器通信受到在请求之间没有客户机上下文存储在服务器上的约束。任何客户端的每个请求都包含服务请求所需的所有信息，会话状态保存在客户端。会话状态可以由服务器转移到另一个服务（如数据库），以保持持续状态一段时间并允许认证。当客户端准备好转换到新状态时，客户端开始发送请求。虽然一个或多个请求未完成，但客户端被认为正在转换。每个应用程序状态的表示包含下一次客户端选择启动新的状态转换时可以使用的链接。[11]

在万维网上，客户端和中介可以缓存响应。因此，响应必须隐含地或明确地将自己定义为可缓存的或不能阻止客户端响应进一步的请求重复使用陈旧的或不适当的数据。良好管理的缓存部分或完全消除了一些客户端 - 服务器交互，进一步提高了可扩展性和性能。

分层系统

参见：分层系统

客户端通常不会将其直接连接到终端服务器，还是直接连接到中间人。中介服务器可以通过启用负载平衡和提供共享缓存来提高系统的可扩展性。他们也可以执行安全政策。

需求代码（可选）[编辑]

另请参阅：客户端脚本

服务器可以通过传输可执行代码临时扩展或定制客户端的功能。其示例可能包括编译的组件，如Java小程序和客户端脚本（如JavaScript）。

统一界面[编辑]

统一接口约束是任何REST服务设计的基础它简化和解耦架构，使每个部分都能独立发展。这个统一接口的四个约束是

识别资源

在请求中标识个别资源，例如在基于Web的REST系统中使用URI。资源本身在概念上与返回给客户端的表示形式分开。例如，服务器可以将数据从数据库发送为HTML，XML或JSON，这些都不是服务器的内部表示。

通过表示操纵资源

当客户端持有包含所有元数据的资源的表示时，它具有足够的信息来修改或删除资源。

自我描述信息

每个消息都包含足够的信息来描述如何处理消息。例如，调用哪个解析器可以由Internet媒体类型（以前称为MIME类型）来指定。

超媒体作为应用状态引擎（HATEOAS）

访问REST应用程序的初始URI - 类似于访问网站主页的人类Web用户，REST客户端应该能够动态地使用服务器提供的链接来发现所需的所有可用操作和资源。随着访问的进行，服务器以文本的形式进行响应，其中包含超链接到当前可用的其他操作。客户端无需使用有关REST服务的结构或动态的信息进行硬编码。

## 2.3 JHipster概述（APP所需的服务器端技术）

### 2.2.1 JHipster概念介绍

JHipster或“Java Hipster”是一个方便的应用程序生成器，将为您创建一个Spring Boot（即后端Java部分）和AngularJS（这是前端js部分）应用程序。

JHipster致力于通过使用大量Spring技术生成具有Java后端的高质量应用程序; Spring Boot，Spring Security，Spring数据，Spring MVC（为Websockets提供框架，REST和MVC）等。一个Angular.js前端和一套预配置的开发工具，如Yeoman，Maven，Gradle，Grunt， Gulp.js和Bower。 JHipster创建一个完全配置的Spring引导应用程序，其中包含一组用于用户管理，监视和记录的预定义屏幕。生成的Spring Boot应用程序是专门为使Angular.js工作更加平滑的体验而量身定做的。

JHipster提供了更新，管理和打包生成的应用程序的工具。运行mvn package -Pprod来触发使用Spring Boot Maven插件创建单个可执行文件.war文件这种生成的war包已经内置了tomcat，所以在没有tomcat的机器上也能直接用java –jar 指令直接运行，特别适合在APP开发过程中搭建一个便捷的服务端，并且自动生成并使用Grunt或Gulp.js任务来测试，最小化和优化JavaScript，HTML和CSS代码。

### 2.2.2JHipster后端应用 Spring boot

2.2.2.1 Spring 框架概念介绍

Spring Framework 是一个开源的Java／Java EE全功能栈（full-stack）的应用程序框架，以Apache许可证形式发布，也有.NET平台上的移植版本。该框架基于 Expert One-on-One Java EE Design and Development（ISBN 0-7645-4385-7）一书中的代码，最初由Rod Johnson和Juergen Hoeller等开发。Spring Framework提供了一个简易的开发方式，这种开发方式，将避免那些可能致使底层代码变得繁杂混乱的大量的属性文件和帮助类。

Spring中包含的关键特性：

1.强大的基于JavaBeans的采用控制反转（Inversion of Control，IoC）原则的配置管理，使得应用程序的组建更加快捷简易。

2.一个可用于从applet到Java EE等不同运行环境的核心Bean工厂。

3.数据库事务的一般化抽象层，允许宣告式（Declarative）事务管理器，简化事务的划分使之与底层无关。

4.内建的针对JTA和单个JDBC数据源的一般化策略，使Spring的事务支持不要求Java EE环境，这与一般的JTA或者EJB CMT相反。

5.JDBC 抽象层提供了有针对性的异常等级（不再从SQL异常中提取原始代码），简化了错误处理，大大减少了程序员的编码量。再次利用JDBC时，你无需再写出另一个'终止'（finally）模块。并且面向JDBC的异常与Spring通用数据访问对象（Data Access Object）异常等级相一致。

6.以资源容器，DAO实现和事务策略等形式与Hibernate，JDO和iBATIS SQL Maps集成。利用众多的翻转控制方便特性来全面支持，解决了许多典型的Hibernate集成问题。所有这些全部遵从Spring通用事务处理和通用数据访问对象异常等级规范。

灵活的基于核心Spring功能的MVC网页应用程序框架。开发者通过策略接口将拥有对该框架的高度控制，因而该框架将适应于多种呈现（View）技术，例如JSP、FreeMarker、Velocity、Tiles、iText以及POI。值得注意的是，Spring中间层可以轻易地结合于任何基于MVC框架的网页层，例如Struts、WebWork或Tapestry。

7.提供诸如事务管理等服务的面向方面编程框架。

在设计应用程序Model时，MVC模式（例如Struts）通常难于给出一个简洁明了的框架结构。Spring却具有能够让这部分工作变得简单的能力。程序开发员们可以使用Spring的JDBC抽象层重新设计那些复杂的框架结构。

2.2.2.2Spring框架中内含的模块

Spring框架包含几个提供一系列服务的模块：

Spring Core Container：这是Spring的基本模块，并提供Spring容器（BeanFactory和ApplicationContext）。

面向方面的程序设计：能够实现跨领域的关注。

认证和授权：通过Spring Security子项目（以前称为Acegi Security System for Spring）支持一系列标准，协议，工具和实践的可配置安全过程。

通过配置：Spring Roo模块提供了一个基于Spring的企业应用程序的快速应用程序开发解决方案

数据访问：使用JDBC和对象关系映射工具和NoSQL数据库在Java平台上使用关系数据库管理系统

控制容器的反转：Java对象的应用程序组件和生命周期管理的配置，主要通过依赖注入完成

消息传递：通过JMS从消息队列中透明消息消息的消息侦听器对象的配置注册，通过标准JMS API改进消息发送

模型视图控制器：基于HTTP和Servlet的框架，为Web应用程序和RESTful Web服务的扩展和定制提供了钩子。

远程访问框架：支持RMI，CORBA和HTTP协议（包括Web服务（SOAP））的网络上的Java对象的配置RPC风格编组

事务管理：统一多个事务管理API并协调Java对象的事务

远程管理：通过JMX进行本地或远程配置的Java对象的配置曝光和管理

测试：用于编写单元测试和集成测试的支持类

控制容器的反转（依赖注入）

Spring框架的核心是控制（IoC）容器的反转，它提供了使用反射配置和管理Java对象的一致手段。容器负责管理特定对象的对象生命周期：创建这些对象，调用它们的初始化方法，并通过将它们连接在一起来配置这些对象。

由容器创建的对象也称为受管对象或bean。可以通过加载XML文件或检测配置类上的特定Java注释来配置容器。这些数据源包含提供创建bean所需信息的bean定义。

可以通过依赖关系查找或依赖注入来获取对象。[13]依赖关系查找是一种模式，其中调用者向容器对象询问具有特定名称或特定类型的对象。依赖注入是容器通过名称向其他对象通过构造函数，属性或工厂方法传递对象的模式。

在许多情况下，在使用Spring Framework的其他部分时，不需要使用该容器，尽管使用它可能会使应用程序更易于配置和自定义。 Spring容器提供了一致的机制来配置应用程序，并将其与几乎所有的Java环境集成，从小型应用程序到大型企业应用程序。

容器可以通过Pitchfork项目变成部分兼容的EJB 3.0容器。有些人批评Spring框架不符合标准[14]然而，SpringSource并没有将EJB 3的遵守视为主要目标，并声称Spring框架和容器允许更强大的编程模型。[15]您不要创建一个对象，而是通过在Spring配置文件中定义它们来描述如何创建它们。您不要调用服务和组件，而是通过在Spring配置文件中定义它们来告诉哪些服务和组件必须被调用。这使得代码易于维护，易于通过IoC进行测试。

面向方面的编程框架[编辑]

Spring框架有自己的面向方面的程序设计（AOP）框架，它将交叉关切的方面模块化。创建单独的AOP框架的动机来自于相信可以提供基本的AOP功能，而无需设计，实现或配置中的太多复杂性。 Spring AOP框架还充分利用了Spring容器。

Spring AOP框架是基于代理模式的，并且在运行时配置。这不需要编译步骤或加载时间编织。另一方面，拦截仅允许在连接点处对现有对象进行公共方法执行。

与AspectJ框架相比，Spring AOP功能较弱，但并不复杂。 Spring 1.2包括支持在容器中配置AspectJ方面。 Spring 2.0增加了与AspectJ的更多集成; 例如，切入点语言被重用，可以与基于Spring AOP的方面混合使用。此外，Spring 2.0添加了一个使用AspectJ的Spring Aspects库，通过AspectJ编译时或加载时编织提供常见的Spring功能，如声明式事务管理和依赖注入。 SpringSource还在Spring Roo和Spring Insight等其他Spring项目中使用AspectJ AOP，Spring Security还提供了一个基于AspectJ的方面库。

Spring AOP旨在使其能够在Spring框架内处理跨部门问题。使用Spring AOP可以丰富由容器创建和配置的任何对象。

Spring框架在内部使用Spring AOP进行事务管理，安全性，远程访问和JMX。

从2.0版框架开始，Spring为AOP配置提供了两种方法：

基于模式的方法[16]和@基于AspectJ的注释样式

春季团队决定不引入新的AOP相关术语;因此，在Spring参考文档和API中，诸如方面，连接点，建议，切入点，引用，目标对象（建议对象），AOP代理和编织等术语与大多数其他AOP框架（特别是AspectJ ）。

数据访问框架

Spring的数据访问框架解决了在应用程序中处理数据库时开发人员面临的常见困难。为Java中所有流行的数据访问框架提供支持：JDBC，iBatis / MyBatis，Hibernate，JDO，JPA，Oracle TopLink，Apache OJB和Apache Cayenne等。

对于所有这些支持的框架，Spring提供了这些功能

资源管理 - 自动获取和释放数据库资源

异常处理 - 将数据访问相关异常转换为Spring数据访问层次结构

交易参与 - 透明地参与持续交易

资源解包 - 从连接池包装器检索数据库对象

BLOB和CLOB处理的抽象

使用Spring为每个支持的框架提供的模板类时，所有这些功能都可用。批评者们认为这些模板类是非常有侵入性的，而且比直接使用（例如）Hibernate API没有任何优势。[18] [不在引用中]作为回应，Spring开发人员可以直接使用Hibernate和JPA API 。然而，这需要透明的事务管理，因为应用程序代码不再承担获取和关闭数据库资源的责任，并且不支持异常转换。

与Spring的事务管理一起，其数据访问框架为处理数据访问框架提供了灵活的抽象。 Spring Framework不提供通用的数据访问API;相反，支持的API的全部功能保持不变。 Spring框架是Java中提供的唯一可用于在应用程序服务器或容器之外提供托管数据访问环境的框架。

在使用Spring进行事务管理与Hibernate时，可能需要配置以下bean：

像com.mchange.v2.c3p0.ComboPooledDataSource或org.apache.commons.dbcp.BasicDataSource这样的数据源

一个SessionFactory，如org.springframework.orm.hibernate3.LocalSessionFactoryBean，具有DataSource属性

一个HibernateProperties如org.springframework.beans.factory.config.PropertiesFactoryBean

一个TransactionManager，如org.springframework.orm.hibernate3.HibernateTransactionManager，具有SessionFactory属性

其他配置点包括：

AOP配置切割点。

AOP咨询的交易语义[澄清]。

交易管理框架

Spring的事务管理框架为Java平台带来了抽象机制。它的抽象能力是：

使用本地和全局事务（本地事务不需要应用程序服务器）

使用嵌套事务

使用保存点

在Java平台的几乎所有环境中工作

相比之下，JTA仅支持嵌套事务和全局事务，并且需要一个应用程序服务器（在某些情况下也是在应用程序服务器中部署应用程序）。

Spring Framework为许多事务管理策略提供了PlatformTransactionManager：

在JDBC连接上管理的事务

在对象关系映射工作单元上管理的事务

通过JTA TransactionManager和UserTransaction管理的事务

在其他资源上管理的事务，如对象数据库

除了这种抽象机制之外，该框架还提供了两种向应用程序添加事务管理的方式：

以编程方式，通过使用Spring的TransactionTemplate

通过使用元数据（如XML或Java注释）（@Transactional等）进行配置，

与Spring的数据访问框架（它集成了事务管理框架）一起，可以通过配置来设置事务系统，而不必依赖于JTA或EJB。事务框架还与消息传递和缓存引擎集成。

模型视图控制器（MVC）框架

Spring框架具有自己的MVC Web应用程序框架，而不是最初的计划。 Spring开发人员决定编写自己的Web框架作为对他们所认为的（当时）流行的Jakarta Struts Web框架的不良设计的反应[19]以及其他可用框架的缺陷。特别地，他们认为呈现和请求处理层之间以及请求处理层和模型之间的分离不足。[20]

像Struts一样，Spring MVC是一个基于请求的框架。框架定义了现代基于请求的框架必须处理的所有责任的策略接口。每个接口的目标是简单明了，所以Spring MVC用户很容易编写自己的实现，如果他们这样选择的话。 MVC为更清洁的前端代码铺平了道路。所有接口与Servlet API紧密耦合。有些人认为与Servlet API的紧密联系是Spring开发人员为基于Web的应用程序提供高级抽象的一个失败。然而，这种耦合确保了开发人员可以使用Servlet API的特性，同时提供了一个高抽象框架来轻松使用所述API。

DispatcherServlet类是框架的前端控制器[21]，负责在HTTP请求的执行阶段将控制委托给各种接口。

Spring MVC定义的最重要的接口及其职责如下：

控制器：在模型和视图之间管理传入请求并重定向到适当的响应。它作为引导传入信息的门。它在进入模型或视图之间切换。

HandlerAdapter：执行处理传入请求的对象

HandlerInterceptor：拦截传入的请求可比较，但不等于Servlet过滤器（使用是可选的，不受DispatcherServlet控制）。

处理程序映射：根据这些请求内部或外部的任何属性或条件，选择处理传入请求（处理程序）的对象

LocaleResolver：解析并可选地保存单个用户的区域设置

MultipartResolver：通过打包传入的请求，方便处理文件上传

视图：负责向客户端返回响应。有些请求可以直接查看，而不去模型部分;其他人都可以通过这三个。

ViewResolver：根据视图的逻辑名称选择一个视图（使用不是严格要求的）

以上每个战略界面在总体框架中都有重要的责任。这些接口提供的抽象是强大的，所以为了允许其实现中的一组变体，Spring MVC附带了所有这些接口的实现，并且一起在Servlet API之上提供了一个功能集。然而，开发人员和供应商可以自由地编写其他实现。 Spring MVC使用Java java.util.Map接口作为模型的面向数据的抽象，其中键将被预期为字符串值。

测试这些接口的实现的容易性似乎是Spring MVC提供的高级抽象的一个重要优点。 DispatcherServlet与控制容器的弹簧反转紧密耦合，用于配置Web应用程序层。但是，Web应用程序可以使用Spring Framework的其他部分（包括容器），并选择不使用Spring MVC。

远程访问框架

Spring的远程访问框架是用于处理Java平台上可用的各种基于RPC的技术，用于客户端连接和服务器上的编组对象的抽象。该框架提供的最重要的功能是通过组合控制和AOP的反转来尽可能地简化这些技术的配置和使用。

该框架还提供故障恢复（连接失败后自动重新连接）和客户端使用EJB远程无状态会话bean的一些优化[阐明]。

Spring提供支持这些协议和产品开箱即用

基于HTTP的协议

Hessian：二进制序列化协议，由基于CORBA的协议开源和维护

RMI（1）：使用RMI基础架构但是Spring特有的方法调用

RMI（2）：使用符合常规RMI使用的RMI接口的方法调用

RMI-IIOP（CORBA）：使用RMI-IIOP / CORBA的方法调用

企业JavaBean客户端集成

本地EJB无状态会话bean连接：连接到本地无状态会话bean

远程EJB无状态会话bean连接：连接到远程无状态会话bean

肥皂

与Apache Axis Web服务框架集成

Apache CXF提供与Spring Framework的集成，用于在服务器端对象进行RPC风格的导出。

在Spring Core容器中配置了Spring Remote访问框架（Apache Axis支持除外）支持的所有RPC风格协议和产品的客户端和服务器设置。

Spring Framework中包含一个Remoting子系统的替代开源实现（Cluster4Spring），它旨在支持各种远程处理方案（1-1，1-many，动态服务发现）

1. 调研基于iOS的物流监控系统移动平台的设计与实现的基本方法，并从理论和实际角度分析这些方法的效率、实用性及可操作性。

2. QR码扫描功能：QR Code码，是由Denso公司于1994年9月研制的一种矩阵二维码符号，它具有一维条码及其它二维条码所具有的信息容量大、可靠性高、可表示汉字及图象多种文字信息、保密防伪性强等优点。

3. HTTP通信：HTTP协议即超文本传送协议，是web联网的基础，也是手机联网的常用协议之一，HTTP协议是建立在TCP协议之上的一种应用。其显著特点是客户端每次请求都需要服器响应，在请求结束后主动释放连接。

4. GPS定位：GPS是英文Global Positioning System（全球定位系统）的简称。GPS起始于1958年美国军方的一个项目，1964年投入使用。手机定位请求是由手机主动发起，主动性掌握在手机。手机不想请求定位即关闭GPS即可终止

为完成该系统的设计实现，主要采用Swift语言编写，结合UIKit、Foundation、Alamofire、SwiftyJSON框架以及服务器的Spring Boot开发技术实现 。采用技术简要介绍如下：

1. Foundation：它是苹果平台开发软件的核心，他的存在就相当于C语言中的stdlib.h一样重要，不可或缺，基本包含了Swift中所需要用到的所有库函数。

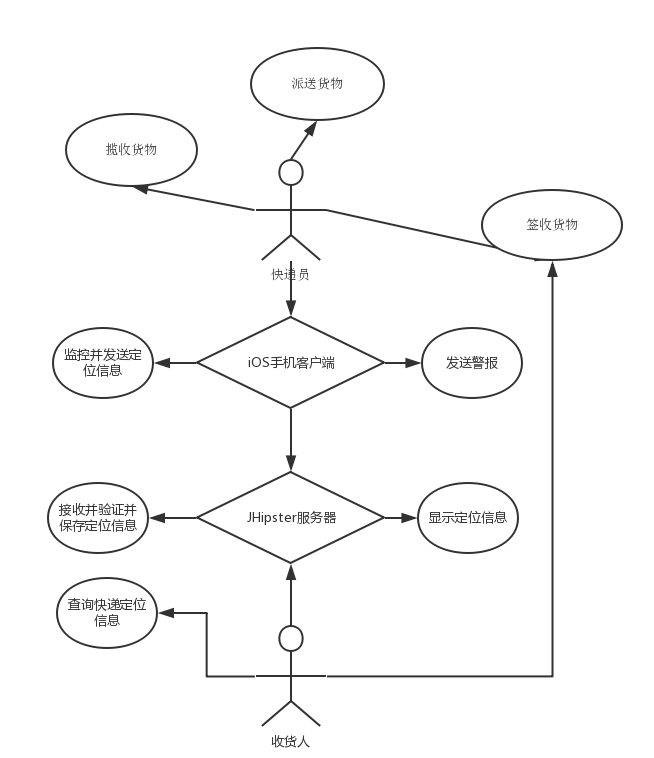
2. UIKit：它是用于开发移动端苹果设备的界面框架，是为了解决iOS应用程序开发复杂性而创建的。框架的主要优势之一就是其内置的MVC架构，使用了UIKit之后，Xcode会自动在项目中生成一个storyboard界面文件，和一个ViewController.swift文件。ViewController即为自动生成的这个界面的控制器。

3. Alamofire：它是一个把NSUrlSession包装成RESTful形式的http包开源项目，通过这个包我们就可以很方便的进行使用RESTful API的iOS软件开发。

4. Jhipster：它是一个一键生成服务端的开源项目。通过JHipster可以一键生成一个使用Spring Boot做后端，Angular做前端，并且后端嵌入了JPA的服务器。对快速生成一个RESTful API的项目有极大的帮助。

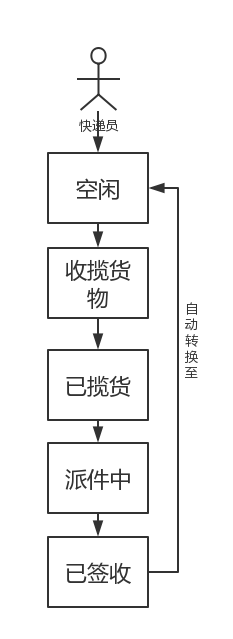
# 第三章 需求分析与设计

## 3.1实体-关系图

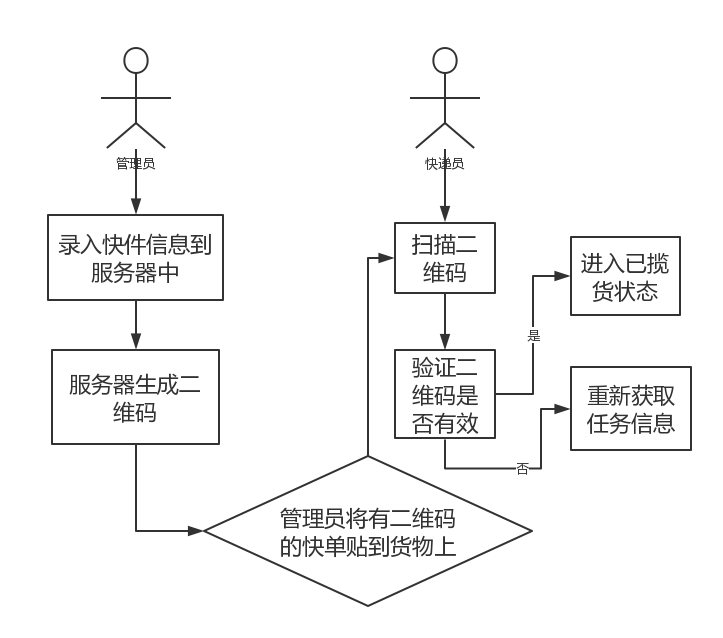


## 3.2状态图

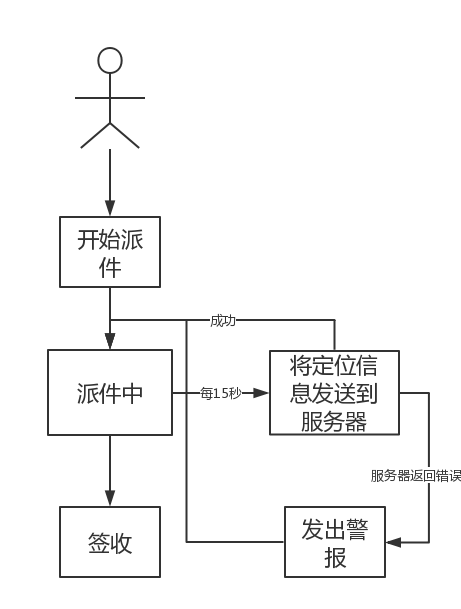
### 3.2.1 总状态



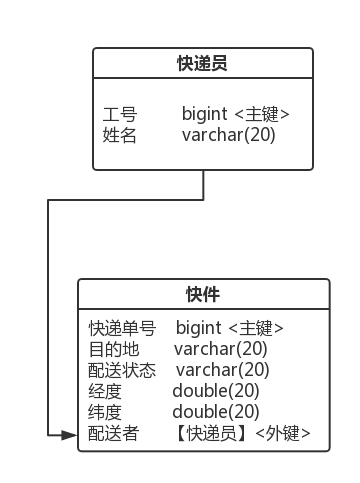
### 3.2.2收揽货物



### 3.2.3派件中



## 3.3服务器端数据库ER图



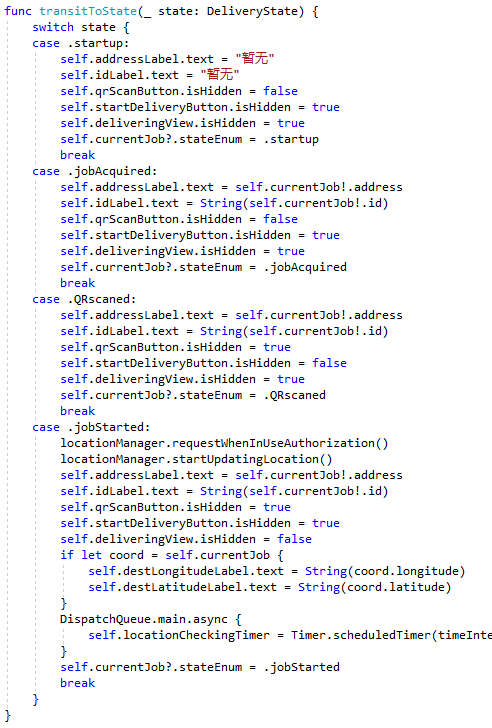
# 第四章 实现

本课题要求开发一款运行在ios系统上的物流监控系统移动端，与服务器端配合使用。

## 4.1工作流程的状态切换

主要工作流程：揽收货物、运输途中、签收货物

代码中有一个State变量表示当前的流程，每种不同的流程UI界面上显示不同的按钮。当进入下一个流程时State变量会改变，State变量的改变会触发transitState函数的执行，该函数会修改界面上显示的按钮以及标签。

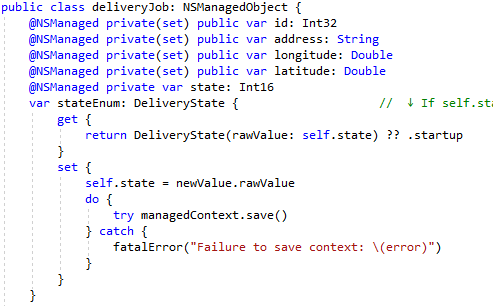


## 4.2保存状态，保证手机重启之后软件的可恢复性

特别需要考虑移动端可能异常关机，所以任务信息需要保存到移动端本地数据库中，保证异常关机后，重新开机时任务状态的恢复。

而最好的方式便是把软件的信息保存在本地数据库中。移动应用一般使用的是SQlite数据库．特点是轻量级。而ios核心框架中，CoreData框架对Sqlite进行 了封装，直接调用CoreDala即可。 CoreData是一个Cocoa框架，用于为管理对象图提供基础实现，以及为多种文件格 式的持久化提供支持。管理对象图包含的工作如撤销(undo)和重做(redo)、有效性检查、 以及保证对象关系的完整性等。对象的持久化意味着Core Data可以将模型对象保存到持 久化存储中，并在需要的时候将它们取出。Core Data应用程序的持久化存储(也就是对象 数据的最终归档形式)的范围可以从XML文件到SQL数据库。Core Data用在关系数据库 的前端应用程序是很理想的，但是所有的Cocoa应用程序都可以利用它的能力。 Core Data的核心概念是托管对象。托管对象是由Core Data管理的简单模型对象， 但必须是NSManagedObject类或其子类的实例。可以用一个称为托管对象模型的结构 (Schema)来描述Core Data应用程序的托管对象(Xcode中包含一个数据建模工具，可以帮 助您创建这些结构)。托管对象模型包含一些应用程序托管对像(也称为实体)的描述。每 个描述负责指定—个实体的属性、它与其它实体的关系、以及像实体名称和实体表示类这 样的元数据。 在一个运行着的Core Data程序中，有一个称为托管对象上下文的对象负责管理托管 对象图。图中所有的托管对象都需要通过托管对象上下文来注册。该上下文对象允许在图 中加入或删除对象，以及跟踪图中对象的变化，并因此可以提供撤销(undo)和重做(redo) 的支持。当准备好保存对托管对象所做的修改时，托管对象上下文负责确保那些对象处 于正确的状态。当Core Data应用程序希望从外部的数据存储中取出数据时，就向托管对象上下文发出—个取出请求．也就是一个指定—组条件的对象。在自动注册之后．上下文 对象会从存储中返回与请求相匹配的对象。

在创建Xcode项目时便启用了iPhone内置的数据库Core Data，然后在货物信息或是当前工作流程(State变量)发生变化时，进行一次保存操作。



如图，将保存操作直接写入state变量的set函数里，可以保证state在有任何变化时都能被实时写入数据库保证安全性。

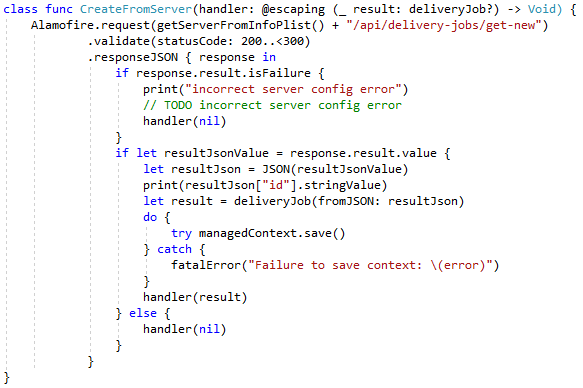
## 4.3 与服务器的信息交互

揽收货物时，从服务器获取待执行的任务信息，扫描货物对应的二维码与任务中的信息比对校验，完成后发送验证信息到服务器等待验证结果。

软件启动时会自动通过一次HTTP GET请求从服务器获取新任务，在揽收货物之前用户也可以通过获取新任务按钮获取新任务。服务器揽收货物按钮按下时会启动摄像头让用户扫描二维码，二维码中的信息正确解析之后，通过一次HTTP POST请求将信息发送给服务器请求服务器再次验证信息。

HTTP(HyperText Transfer Protocol)是一套计算机通过网络进行通信的规则。HTTP通信的一次完整的通信过程：建立TCP连接，在HTTP工作开始之前，手机端APP首先要通过网络与服务器建立连接，一般TCP连接的端口号是80； 手机端APP向服务器发送请求命令；手机端APP发送请求头信息；服务器应答，客户机向服务器发出请求后，服务器会客户机回送应答； 服务器发送应答头信息； 服务器向手机端APP发送数据，服务器向手机端APP发送头信息后，它会发送一个空白行来表示头信息的发送到此为结束； 服务器关闭TCP连接，一般情况下，一旦服务器向浏览器发送了请求数据，它就要关闭TCP连接，然后如果手机端APP或者服务器在其头信息加入了这行代码Connection:keep-alive，TCP连接在发送后将仍然保持打开状态 。

Alamofire是 Swift 语言的 HTTP 网络开发工具包,相当于Swift实现AFNetworking版本。当然,AFNetworking非常稳定,在Mac OSX与iOS中也能像其他Objective-C代码一样用Swift编写。不过Alamofire更适合Swift语言风格习惯(Alamofire与AFNetworking可以共存一个项目中,互不影响) Alamorefire相比较于原生iOS网络函数的优点莫过于对JSON类型的完美支持。项目中使用JSON传输信息可以传输多类型多种信息在一个请求中。大大增高效率。



## 4.4 保存时的错误处理与警告

验证成功后装车出发，验证不通过，根据服务器返回验证消息中的正确任务信息更新任务，重新扫描；

验证通过后，软件会切换到运输途中状态。验证失败会有提示, 并通过服务器返回的正确信息重新修改软件上的UI显示。

在任何与服务器的信息交互中，处理服务器返回值时，都会首先判断返回值中是否包含有警告或者错误信息，比如定位错误信息。软件则会把信息以一个有警告声的提示的方式显示出来。

系统声音服务(System Sound Services)提供了一个接口，用于播放不超过30秒的声音。它支持的文件格式有限，具体地说只有CAF、AIF和使用PCM或IMA/ADPCM数据的WAV文件。由于这些函数没有提供操纵声音和控制音量的功能，所以当你为多媒体或者游戏创建专门的配乐时，不要使用系统声音服务。

iOS使用系统声音服务来支持三种不同的通知：

1. 声音：立刻播放一个简单的声音文件。如果手机被设置为静音，用户什么也听不到

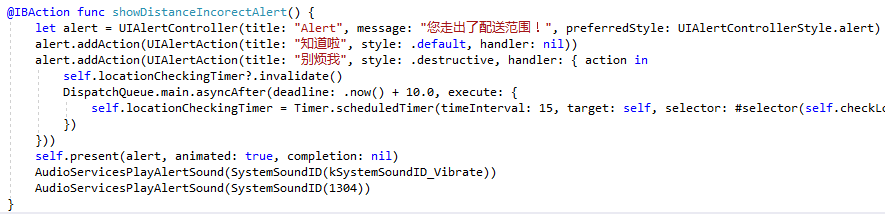
2. 提醒：播放一个声音文件，如果手机被设置为静音或震动，将通过震动提醒用户

3. 震动：震动手机，而不考虑其他设置

系统声音服务并非是通过类实现的，而是使用传统的C语言函数调用来触发播放操作。要播放音频，需要使用的两个函数是AudioServicesCreateSystemSoundID 和 AudioServicesPlaySystemSound。还需要声明一个类型为SystemSoundID的变量，它表示要使用的声音文件。下面演示了如何加载并播放声音：

然后在需要打开声音的地方添加代码：AudioServicesPlaySystemSound(audioID);

需要注意的是，震动也有一个自己的SystemSoundID，也就是说，我们可以直接使用声音函数来使手机振动。



## 4.5信息同步到服务器上

揽收完成后出发，运输途中每隔一段时间（15s）使用GPS定位一次，并且发送定位信息到服务器。

iOS 7 的 四种定位服务

1.GPS卫星 优点最为准确，缺点 耗电量大，不能遮挡（建筑物内不行）

2.WI-FI通过WIFI路由器定位 优点比较省电，经济实惠

3.蜂窝式移动电话基站。通过移动运营商基站定位 误差比较‑大

4.iBeacon微定位。 苹果公司自己研发的iBeacon，使用低功耗的蓝牙。

iOS 定位服务开发不需要指定定位途径（除了微定位外）ios 会根据设备的情况采用最佳方案，如果能接受GPS优先采用GPS定位，次之是Wi-Fi ,最后是蜂窝式基站定位；

iOS 定位服务使用Core Location框架

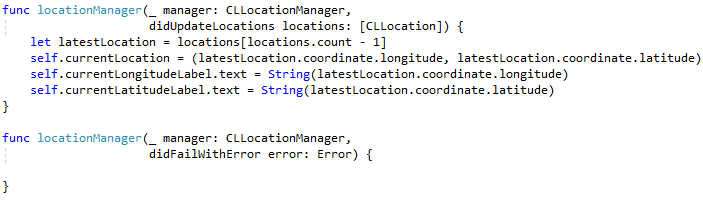
CLLocationManager 用于定位管理的类，他提供位置信息和高度信息，也可以监控设备进入或者离开某个区域，还可以获得设备的运行方向；

CLLocationManagerDelegate。CLLocationManager类的委托协议；

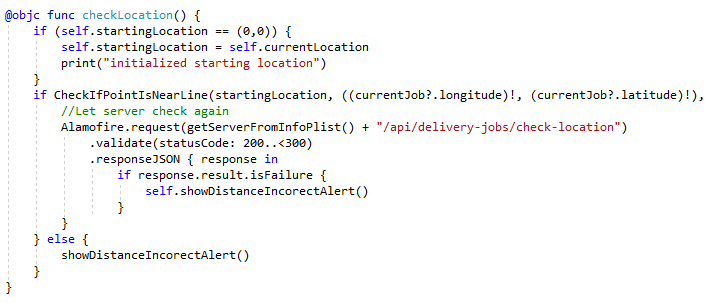
CLLocation 封装了位置和高度信息；

第一次请求定位信息时，系统会提示用户是否允许开启定位服务。

询问用户得到GPS使用权限之后，CLLocationManagerDelegate接口里的didUpdateLocations方法在每次GPS信息改变时会被调用。通过这个我们可以让这个函数实时修改经度和纬度两个变量。然后以这两个变量来作为当前的实时位置信息。



通过Foundation库中已经内置了的NSTimer方法可以完美的达到每隔15s执行一次函数的功能。在该函数中获取当前的经纬度值并将该值，通过一次判断是否在路线上，如果成功则再通过一次HTTP PUT请求发送给服务器，若判断失败或者是服务器发送了任何警告信息，则会跳出一个有声音震动的提醒。



## 4.6QR码读取

IOS7之前，开发者进行扫码编程时，一般会借助第三方库。常用的是 ZBarSDKa 和 ZXingObjC ，IOS7之后，系统的AVMetadataObject类中，为我们提供了解析二维码的接口。经过测试，使用原生API扫描和处理的效率非常高，远远高于第三方库。



# 第五章 测试

# 第六章 总结