

**本 科 毕 业 设 计（论文）**



**题目: 基于iOS的物流监控系统移动平台的设计与实现**

**姓 名 阿尔斯兰·阿布力克木**

**学 院 计算机学院**

**专 业 计算机科学与技术**

**班 级 2013211308**

**学 号 2013211392**

**班内序号 21**

**指导教师 姚文斌**

**2017年 5月**

基于iOS的物流监控系统移动平台的设计与实现

**摘 要**

近年来手机已经普及到了人手一台智能机的地步，通过手机APP来加快业务效率已成一大趋势。二维码技术的快速发展，带动着移动互联网应用的在手机与实物之间的信息交互，通过计算机、互联网、手机等物品的二维码生成与识别和信息的互联与共享，提高了生产和管理的效率。随着现代物流对物流监控的要求越来越高，积极探索使用手机进行物流信息的互联共享，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iOS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:** 二维码、iOS、物流监控、物流管理、物流信息化

Design and Implementation of Mobile Terminal Based on iOS's Logistics Monitoring System

**ABSTRACT**

In recent years, mobile phones have been popular to the point of the staff of an intelligent machine, through the mobile phone APP to speed up business efficiency has become a major trend. The rapid development of two-dimensional code technology, driven by mobile Internet applications in the mobile phone and physical information exchange between the computer, the Internet, mobile phones and other items of two-dimensional code generation and identification and information interconnection and sharing, Management efficiency. With the modern logistics requirements for logistics monitoring more and more high, and actively explore the use of mobile phone logistics information sharing, in the field of logistics applications have important practical significance.

  This paper studies the design and implementation of mobile platform for logistics monitoring system based on iOS. The system is divided into remote data monitoring, iOS client and simple server administrator Web interface, respectively, in the Xcode and IntelliJ IDEA platform development. Through the use of HTTP latest RESTful API interactive technology, to achieve the iOS client remote access server and operate the database function. RESTful API interaction is the whole system of "nerve center", through frequent API interaction, real-time synchronization courier information to the server function, the main function of express mail location monitoring, express status monitoring, express mail acceptance information verification. IOS client, so that the courier can scan the two-dimensional code through the mobile phone client quickly pick and send the server to send the location of the error warning, and the recipient can also be real-time through the iOS client to The server's precise GPS location information is monitored to its own express location and the expected arrival time. The system makes full use of modern information technology, especially iOS technology, the traditional logistics business to optimize the integration, in order to achieve lower logistics costs, improve the level of logistics services.

**Key Words:** Quick Response Code, iOS, logistics monitoring, logistics management, logistics information

目录

[第一章 绪论 6](#_Toc483408925)

[1.1背景和意义 6](#_Toc483408926)

[1.2物流信息化国内外研究现状 6](#_Toc483408927)

[1.2.1国外研究现状分析 6](#_Toc483408928)

[1.2.2国内研究现状分析 6](#_Toc483408929)

[第二章 开发环境和相关技术介绍 8](#_Toc483408930)

[2.1 iOS简介（本APP的开发平台） 8](#_Toc483408931)

[2.1.1 iOS操作系统介绍 8](#_Toc483408932)

[2.1.2 iOS操作系统架构 8](#_Toc483408933)

[2.1.3 iOS开发工具Xcode 10](#_Toc483408934)

[2.1.4CocoaPod概述 10](#_Toc483408935)

[2.2 iOS 开发语言Swift 11](#_Toc483408936)

[2.2.1Swift 概念介绍 11](#_Toc483408937)

[2.2.2 Swift的功能与优点 11](#_Toc483408938)

[2.3 RESTful API 概述（APP与服务器进行交互的技术） 13](#_Toc483408939)

[2.3.1RESTful API 概念介绍 13](#_Toc483408940)

[2.3.2RESTful 架构定义 14](#_Toc483408941)

[2.4 JHipster概述（APP所需的服务器端技术） 15](#_Toc483408942)

[2.4.1 JHipster概念介绍 15](#_Toc483408943)

[2.4.2JHipster后端应用 Spring boot 15](#_Toc483408944)

[2.5 QR码概述 19](#_Toc483408945)

[2.5.1 QR码概念介绍 19](#_Toc483408946)

[第三章 需求分析与设计 21](#_Toc483408947)

[3.1实体-关系图 21](#_Toc483408948)

[3.2状态图 22](#_Toc483408949)

[3.2.1 总状态 22](#_Toc483408950)

[3.2.2收揽货物 23](#_Toc483408951)

[3.2.3派件中 23](#_Toc483408952)

[3.3服务器端数据库ER图 25](#_Toc483408953)

[3.4安全性设计 25](#_Toc483408954)

[3.4.1 JWT介绍 25](#_Toc483408955)

[3.4.2 JWT在项目中的应用 26](#_Toc483408956)

[3.5算法设计 26](#_Toc483408957)

[3.5.1计算快递员是否走出范围 26](#_Toc483408958)

[第四章 实现 28](#_Toc483408959)

[4.1工作流程的状态切换 28](#_Toc483408960)

[4.2保存状态，保证手机重启之后软件的可恢复性 29](#_Toc483408961)

[4.3 与服务器的信息交互 30](#_Toc483408962)

[4.4 保存时的错误处理与警告 31](#_Toc483408963)

[4.5信息同步到服务器上 32](#_Toc483408964)

[4.6QR码读取 33](#_Toc483408965)

[第五章 测试 35](#_Toc483408966)

[5.1单元测试 35](#_Toc483408967)

[5.2性能测试 35](#_Toc483408968)

[5.3测试用例 36](#_Toc483408969)

[第六章 总结 37](#_Toc483408970)

# 第一章 绪论

## 1.1背景和意义

随着信息技术的不断发展和互联网的不断普及，越来越多的人使用网上购物。因此，人们越来越依赖于物流。很快，有先进的计算机网络安全服务，先进的通信网络和最快的商业信件和包裹运输到世界的各个角落，快速的快递公司要求必需有一个快速的运输，负责货物的配送，配送方式，利用技术的门到门或桌服务快递物品实施，必须准确及时，客户信息的快速处理订单，跟踪快递的实施，提供查询服务。[[1]](#footnote-1)UPS投资110亿美元的电子技术，使它能够跟踪包装交付的每一步，使其能够利用电子商务的浪潮。服务传递行业形成表格到餐桌，门到门，它需要实施跟踪来表达，为客户表达的实时查询，所以快递公司必须以优质的服务留住客户。

配送服务是物流活动的对客户体验影响最大的环节。使用智能城市交通的物流监控及调度平台将物流信息共享，不仅能减少成本，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，而且能提高客户对物流的体验。

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。[1] 美国很多企业都将物流信息化作为物流合理化的重要途径，比如：

1、采用条码技术和RFID技术，提高工作效率和信息采集的准确性；内部和外部的基于互联网的信息传输电子数据交换技术，实现无纸化订单录入、处理、跟踪、结算处理。

2、通过与供应商和客户的信息共享，实现供应链的透明度，实现供应链伙伴之间的协同商务，以“用信息取代库存，供应链，降低物流总成本，提高供应链的竞争力。

3、借助网上采购材料，网上销售过剩库存等电子商务手段，降低物流成本。

然而，虽然在美国日本等发达国家国内的物流信息化已经达到世界前沿水平，国际物流却仍处在混乱的状态。全球都在对对铁路、公路、水运、航空等基础设施的增加投入，初步缓解了交通运输的全面紧张状况。但与满足国际物流服务需求的能力相比，交通运输总体规模目前仍然不高，交通运输矛盾仍然比较突出。[2] 国际物流节点功能混乱，不同的国家有不同的物流信息节点，导致国际物流节点功能层次混乱。

### 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，2014年我国的物流成本占GDP18%。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。[3]

当前我国物流业的发展和物流信息化市场正在加快进度飞速发展。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。[4] 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广泛提高，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。尤其是近年来，快递公司越来越多，这些快递公司也通过各种手机端APP，网页，开放接口等多种方式实现物流信息的共享。当前，我国物流业正处于重要的战略机遇期。2015年我国社会物流总额达到219.2万亿元，“十二五”时期年均增长8.7%，社会物流需求规模持续扩大。

# 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。[5] 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）

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，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，这也是Java中，C++中默认的方法。而在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) 。iOS中的这种运行时覆盖方法，大大降低了代码的编写难度。也加快了代码的运行速度，虽然造成了更大的内存消耗，但是iOS的内存管理完美解决了这一点问题。

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。

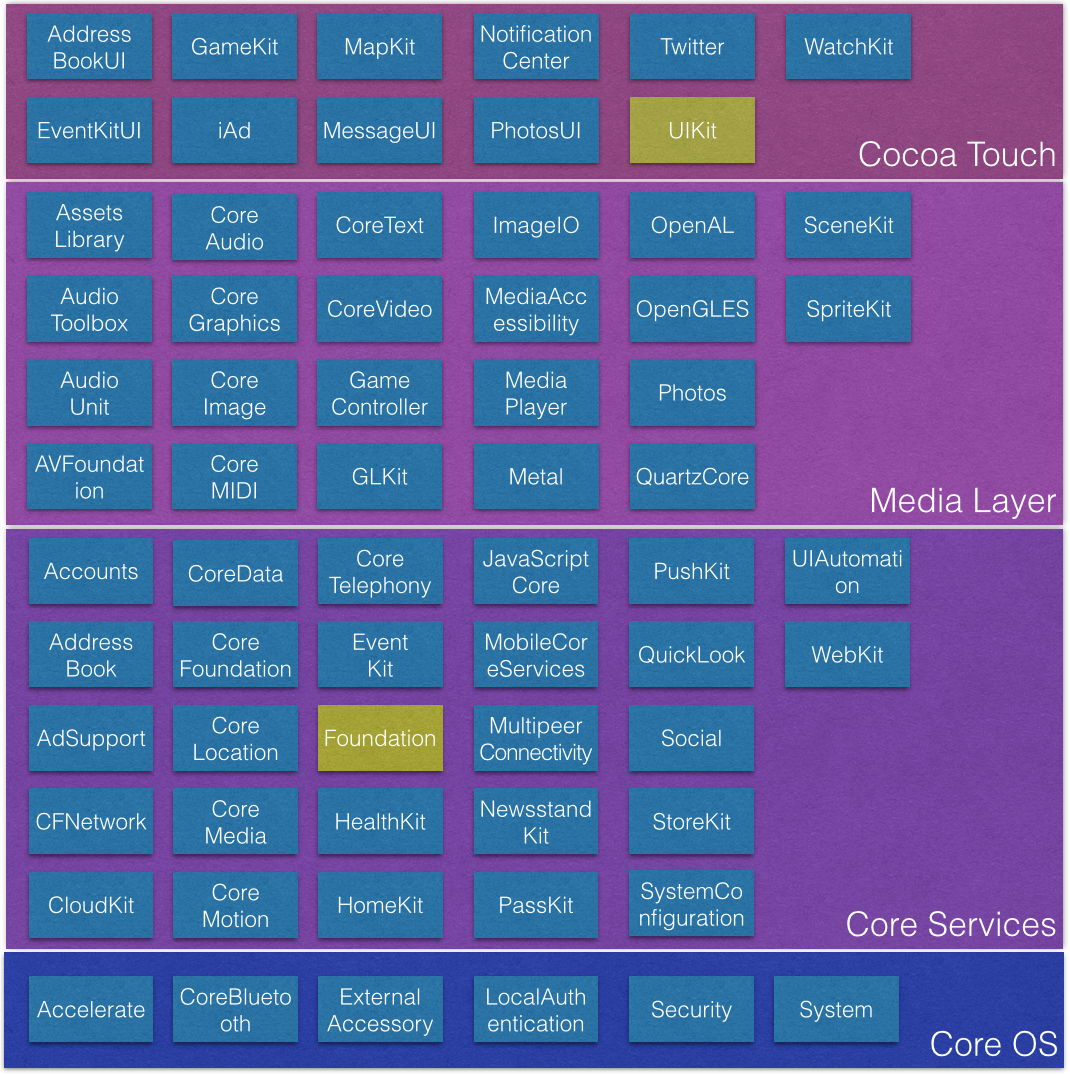


图2-1 iOS8.3系统框架架构图iOS系统框架概述 - 简书

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。

Core Services Layer, 系统服务层提供给应用所需要的基础的系统服务。其中包括最核心的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。相当于C语言中的stdlib.h。

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。其中最主要的便是UIKit。

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，就相当于C语言中的stdlib.h一样重要。 UIKit框架顾名思义，就是基础的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。

### 2.1.4CocoaPod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 iOS 开发语言Swift

### 2.2.1Swift 概念介绍

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

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。

Swift在苹果2014年全球开发者大会（WWDC）上推出。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。

Swift的缺点在于对旧设备的兼容性，Swift语言无法对iOS8以下设备进行编译，但是Swift的优点在于对语言的兼容性，Swift代码可以与Objective-C代码和C，C++代码进行混用，也就是说，之前用Objective-C写的项目完全不必要进行升级和改动，可以直接向项目中添加Swift文件。

而且Swift更有最新语言的特性，比如JS的模糊变量类型功能，你就可以在Swift语言中声明变量时不去声明他的类型。函数式编程可以让你把一个函数想变量一样作为参数和函数的返回值，并且也支持lambda语句。

### 2.2.2 Swift的功能与优点

Swift是Objective-C语言的替代方法，它采用现代编程语言理论概念，力求提供更简单的语法。他被称为“没有C的Objective-C”。

默认情况下，Swift不会公开指针并不允许其他不安全的访问器，与Objective-C相反，Objective-C使用指针来引用对象实例。此外，Objective-C中使用的类似Smalltalk的语法进行方法调用，在Swift中已被其他常见的面向对象（OO）语言（如Java或C＃）的程序员所熟悉的点符号样式和命名空间系统所取代。 Swift引入了真正的命名参数，并保留了关键的Objective-C概念，包括协议，闭包和类，大部分用更清晰的版本替换以前的语法，并允许将这些概念应用于其他语言结构，如枚举类型（Enum）

语法糖

在Cocoa和Cocoa Touch环境下，许多常见的类是Foundation Kit库的一部分。这包括NSString字符串库（使用Unicode），NSArray和NSDictionary集合类等。 Objective-C提供了各种句法糖，以允许在语言中即时创建这些对象中的某些对象，但一旦创建，这些对象就需要调用类函数来进行操作。例如，在Objective-C连接两个NSStrings所需的类似于这样的方法调用：

NSString \* str = @“hello”;

str = [str stringByAppendingString：@“world”];

在Swift中，这些基本类型中的许多已经被提升全局函数定义，可以直接操纵。例如，字符串不可见地桥接到NSString（当导入过Foundation库时），现在还可以直接调用+运算符连接，大大简化了语法;先前的例子变成了

var str =“hello”，

str + =“world”

文件处理

Swift支持符号的五个访问控制级别：open，public，internal，fileprivate和private。与许多面向对象的语言不同，这些访问控制忽略继承层次结构：private表示符号只能在类范围内访问，fileprivate表示只能从文件内部访问，internal表示可以在包含它的模块中访问public表示它可以从任何模块访问，并且open（仅可用于类和open类中的的方法）表示该类可以被模块之外类继承。

Optional和链接

Swift中的一个重要的新功能是Optional类型，允许引用或值以类似于C中的模式的方式操作，其中指针可以引用值或可以为null。这意味着非Optional类型不会导致空指针错误;编译器会确保非Optional类型永远不为null。

Optional类型使用Optional机制创建，要创建一个值可能为null的Int变量，你必须特别声明变量为Optinal<Int>，有点类似于Java8和C#中的Optional。为了使代码书写方便简洁，Swift对Optional类型也专门有语法糖，允许通过在类型名称后面添加一个问号来指示变量是可null的，例如 var optionalInteger：Int？标记为可选的变量或常量具有声明的类型的值或为null。可选类型包装基类型，会被视为不同的类型。比如String和String？是根本不同的类型，因为String？其实是Optional<String>的简写。

要访问Optional类型里面的值，假设它不是nil，它必须被打开以将它里面的值暴露出来。

anOptionalInstance！.someMethod()

在这种情况下，！会强制解开Optional，允许在其上进行方法调用。如果anOptionalInstance为nil，则会发生空指针错误。这在实践中很令人讨厌，所以Swift还包括可选链接的概念，以测试包含的值是否为null，然后如果非null则则将其解开：

    anOptionalInstance？.someMethod()

在这种情况下，只有在anOptionalInstance不为null的情况下才会调用someMethod，从而抑制错误。但是没有调用someMethod的话我想执行另一个语句怎么办呢？

Swift 2会为代码停止执行的情况引入了新的关键字guard：

guard let dongxi = anOptionalInstance？.someMethod() {

}

guard其实就是一种省略了if的if else语句，如果anOptionalInstance为null则会执行大括号中的语句。

ObjC是弱类型，允许任何方式在任何对象上被调用。 如果方法调用失败，则在运行时中有一个默认处理程序返回nil。 这意味着不需要解包或测试，ObjC中的等效语句：

     leaseStart = [[[aBuilding tenantList：5] leaseDetails] startDate]

将返回null，这可以测试。 然而，这也要求所有方法调用都是动态的，这引起了很大的开销。 Swift对可选项的使用提供了类似的机制，用于测试和处理null，但这样做的方式允许编译器使用静态调度，因为在定义的线程调用解包操作，而不是在实时多线程中。

性能

Swift引入的许多功能也具有众所周知的性能和安全性缺点。 苹果投入了大量精力进行积极的优化，可以减少方法调用和使用访问器，以减少对性能的消耗。

## 2.3 RESTful API 概述（APP与服务器进行交互的技术）

### 2.3.1RESTful 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。

### 2.3.2RESTful 架构定义

定义RESTful系统有六个指导约束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

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

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。

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。[6]

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。

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

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。

## 2.4 JHipster概述（APP所需的服务器端技术）

### 2.4.1 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。

### 2.4.2JHipster后端应用 Spring boot

2.3.2.1 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。[7]

2.3.2.2Spring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。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，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。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

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。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。

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。

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，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。

## 2.5 QR码概述

### 2.5.1 QR码概念介绍

QR码的“QR”是Quick Response 的缩写。是二维码的一种，这种二维码能够快速读取。与之前的一维条形码相比， 二维QR 码能存储更丰富的信息，包括对文字、URL 地址和其他类型的数据加密。QR码1994年由日本Denso-Wave公司发明，QR码的标准JIS X 0510在1999年1月发布，而其对应的ISO国际标准ISO/IEC18004，则在2000年6月获得批准。根据Denso Wave公司的网站资料，QR码是属于开放式的标准，QR码的规格公开，而由Denso Wave公司持有的专利权益，则不会被执行。[8] 除了标准的QR码之外，也存在一种称为“微型QR码”的格式，是QR码标准的缩小版本，主要是为了无法处理较大型扫描的应用而设计。微型QR码同样有多种标准，最高可储存35个字元。

与一维码不同，二维码不再使用线性扫描的方式工作，而是使用红外光增强的摄像头工作，直接对镜头拍摄到的图像中的QR码图像进行软件识别，所以对反射角度的要求大大降低。二维码扫描器甚至能对液晶屏幕上显示的条码进行“扫描”与识别，所以可以直接扫描到手机等屏幕上显示的条码。

QR 码呈正方形，只有黑白两色。在4个角落的其中3个，印有较小，像“回”字的的正方图案。这 3 个是帮助解码软件定位的图案，使用者不需要对准，无论以任何角度扫描，资料仍可正确被读取。[8]

# 第三章 需求分析与设计

## 3.1实体-关系图

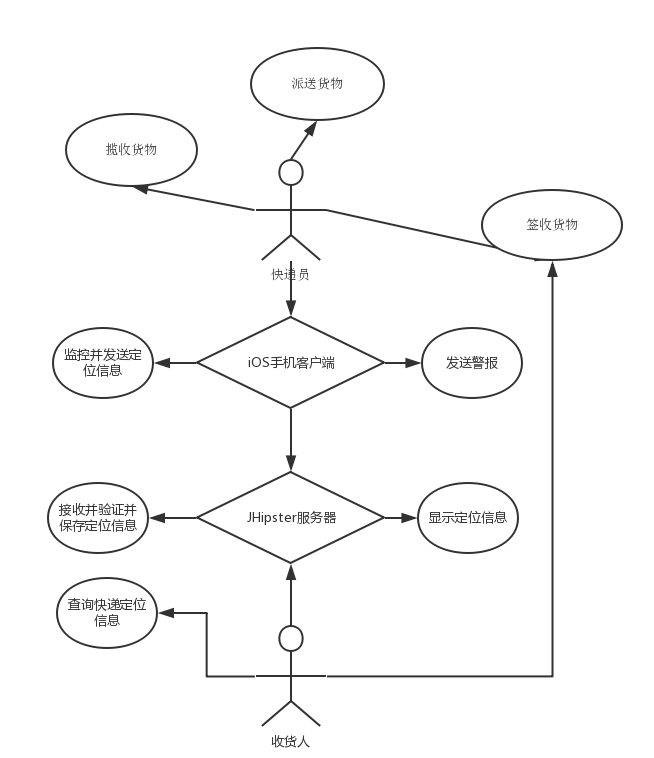


图3-1实体-关系图

## 3.2状态图

### 3.2.1 总状态

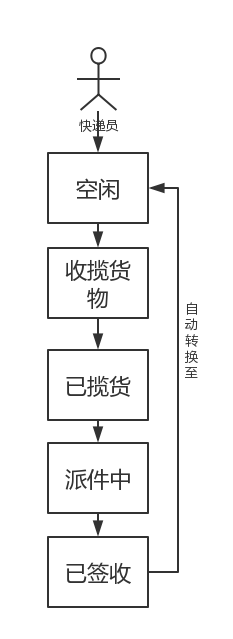


图3-2 总状态图

### 3.2.2收揽货物

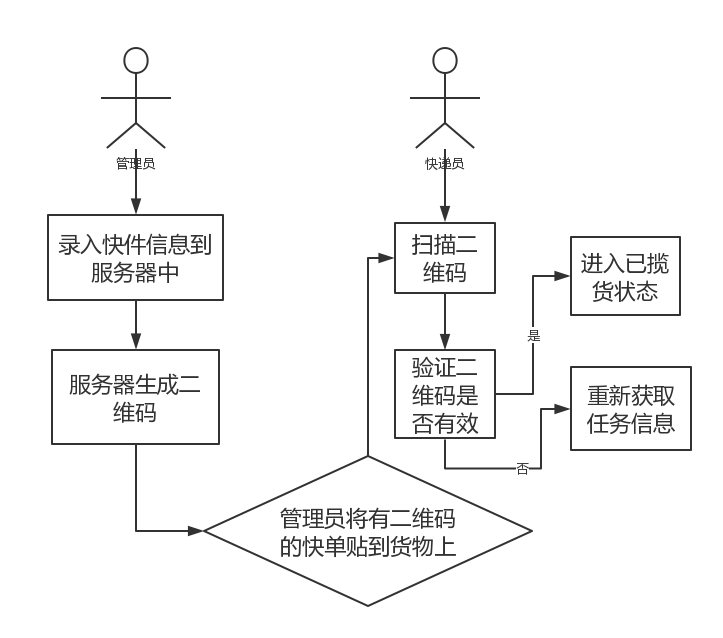


图3-3 收揽货物状态图

### 3.2.3派件中

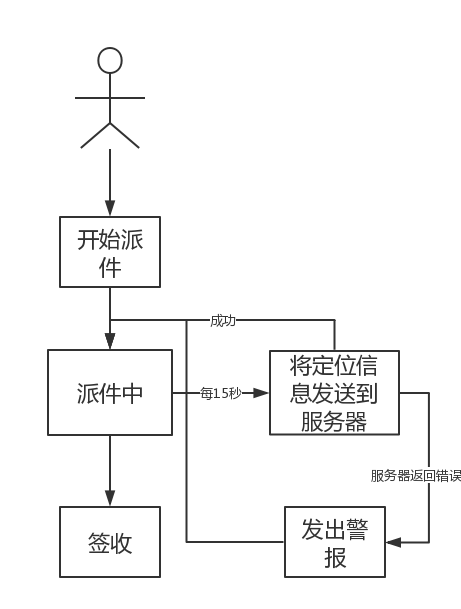


图3-4 派件中状态图

将定位信息发送到服务器其实也属于派件中状态，但是因为其包含很多重要的逻辑和算法，所以将其单独列出来。

## 3.3服务器端数据库ER图

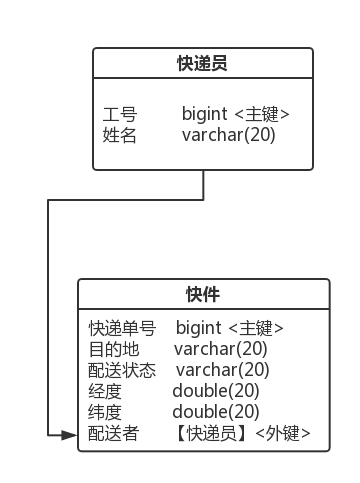


图3-5 服务器端数据库ER图

## 3.4安全性设计

### 3.4.1 JWT介绍

JWT全称为JSON Web Token是一种开放性的标准（RFC7519）.他定义了一种安全的传输JSON信息的方法。JWT可以用HMAC或者RSA方式加密。

### 3.4.2 JWT在项目中的应用

因为考虑到APP只有快递员会使用，所以不需要注册登录功能。但是仍需要一种辨别每个APP使用者的方法。所以引入了JWT。

每个APP在获取新任务时，都会根据快递员ID和姓名，在服务端生成一个JWT，JWT是一串不可读的乱码字符串，服务器内的JHipster库中已经包含JWT的加密解密器。然后APP端在收到获取到的新任务信息是，也会同时收到服务器发给他的JWT。

APP然后则通过Alamofire自带的Headers功能，在每次向服务器发送消息时将JWT作为请求的Authentication请求头发出去。服务器在收到请求时，会先对JWT进行解密，解密之后会从数据库中寻找是否有匹配的快递员ID和姓名，如果找到了则执行请求，如果未找到，则会拒绝请求，这样便做到了服务器对APP的辨别以及对非法请求的阻止。

## 3.5算法设计

### 3.5.1计算快递员是否走出范围



图3-6 计算点到直线的距离

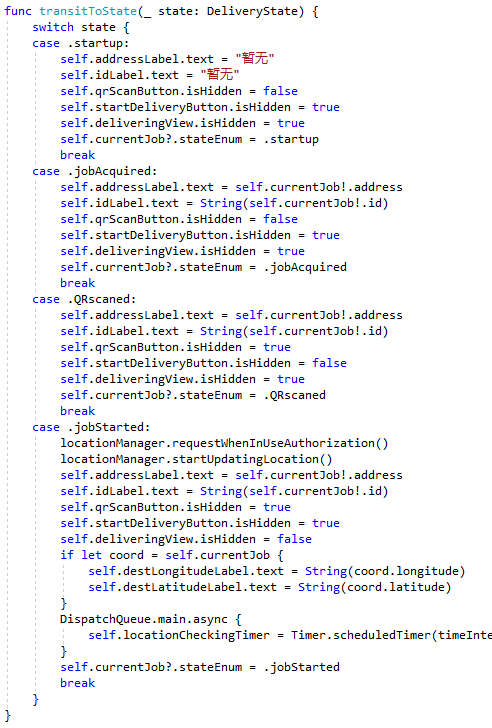
如图所示，若快递员位置（T）到揽货点（A）的距离为D1，快递员位置（T）到收货点（B）的距离为D2，则快递员位置（T）在揽货点（A）和收货点（B）的连接线上的映射 到揽货点的距离为D1+(D2-D1)/2。映射，快递员位置（T），以及揽货点（A）三点形成一个直角三角形，通过勾股函数可以算出快递员到揽货点和收货点的连接线的距离。

# 第四章 实现

## 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。



如图所示，工作流程的状态转换其实并没有数据的变化，变化的只有保存状态的state变量，和UI界面的改变。唯一不同的是在转换到派送中状态时会触发软件获取位置信息的权限，并在新线程中开启计时器，计时器每15s同步信息到服务器一次。（图中每个case最后改变的stateEnum变量会触发state变量的改变，这么处理的原因是因为我们要保存状态到本地CoreData数据库，而本地CoreData数据库无法保存枚举类型的变量，所以我们的state变量是Int类型，而stateEnum是枚举类型，这样我们就可以做到既能使用Swift强大的枚举变量的功能，又可以使其保存在本地数据库上。stateEnum变量触发state变量的改变的实现详见图4-2）

图4-1 软件中状态转换的实现

## 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轻量级的数据库SQlite。而ios核心框架中，CoreData框架对Sqlite进行 了封装，直接调用CoreData即可。 CoreData是一个Cocoa数据库框架，用于在本地保存数据，支持多种数据类型和文件类型。

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。

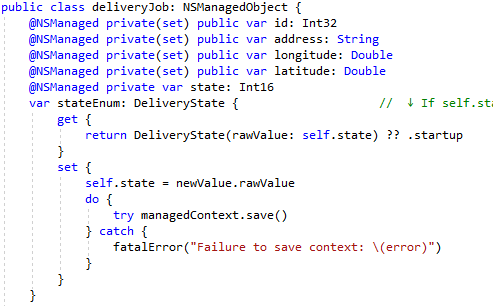


图4-2 软件中受管理对象的实现

如图，将保存操作直接写入stateEnum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。并且在这里可以体现出了Swift语言中优越的变量权限设计，与Java等语言不同，Swift中无需写getter和setter，变量可以分别设置他的读权限和写权限，图中 id, address, longitude,latitude都是外部可读不可写的变量，state是外部不可读写的变量，stateEnum是有特定的getter和setter的变量。StateEnum的getter会将state（Int类型）转换成枚举然后返回，setter会将枚举转换成Int类型保存到state变量中，并触发一次数据库保存，保证state变量被修改的时候都会有一次保存，保证软件的可恢复性。

## 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。[10]

但是当JSON中包含太多嵌套对象时，使用Swift的原生JSON解析语句，每层嵌套解析急需要写一个 guard，大大降低代码的可读性，所以在此项目中又引入了SwiftyJSON库，SwiftyJSON完美兼容原生Swift以及Alamofire，SwiftyJSON的用处便是优雅的解析并创建JSON。

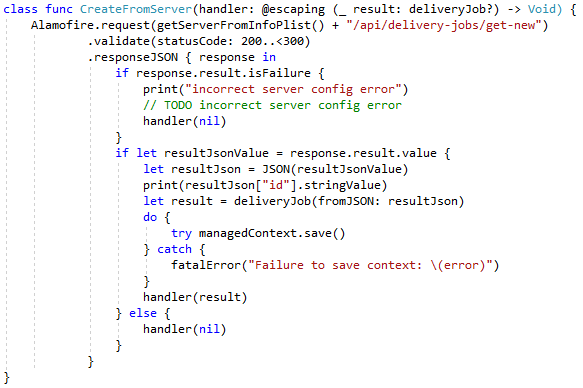


图4-3 软件中RESTful式交互的实现

## 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。

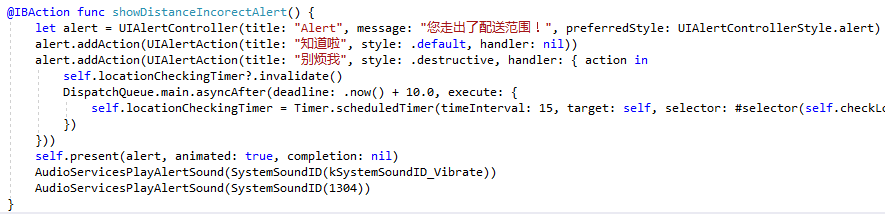


图4-4 软件中警报声的实现

## 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。

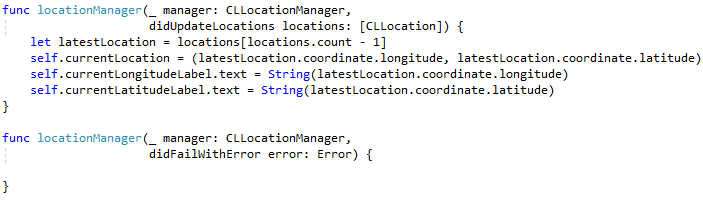


图4-5 软件中获取经纬度信息的实现

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。

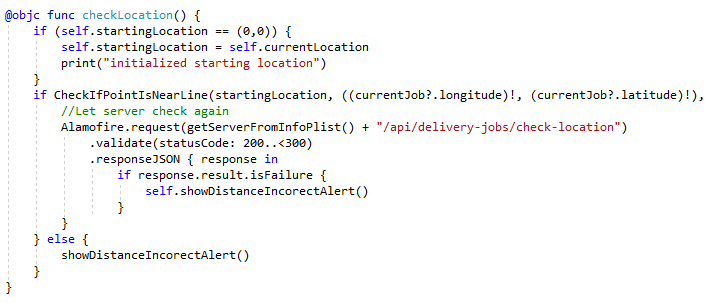


图4-6 软件中同步经纬度信息的实现

## 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。

在这里使用了QRCodeReader.swift包，进一步封装了QR读取，主要是为了能让代码简洁。[9]



图4-7 软件中二维码读取的实现

如代码中所示，QRCodeReader.swift在解析到QR码之后会立即触发链接的scanAction函数，函数将读取到的内容发给服务器验证，无论是否验证成功都会退出QR码扫描界面。如果

# 第五章 测试

## 5.1单元测试

XCTest是苹果开发测试的框架，XCTest和其他语言的单元测试库一样，都是通过比较函数的输出值是否等于预期值的方式来判断。

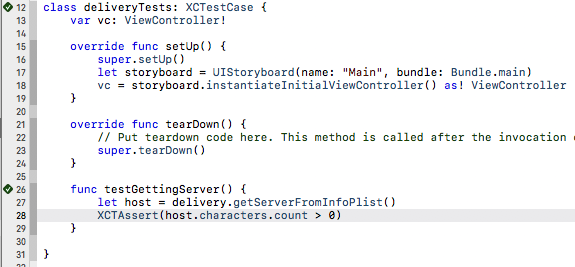


图5-1 软件单元测试

## 5.2性能测试

Xcode开发工具集的一部分, Instruments是一个多方位的的性能分析和测试工具,它被设计

Instruments包含许多的工具，在本次项目中最常用的便是时间事件查看器工具(Time Profiler)，时间事件查看器工具可以实时显示现在软件在执行哪个函数，已经之前执行过的函数的执行时长，以及软件在时间线上的CPU使用率。这在找出软件速度慢的原因时尤为方便。

除了时间事件查看器工具，Instruments 还有很多实用的工具，比如：

Core Data：监测读取、缓存未命中、保存等操作，能直观显示是否保存次数远超实际需要。

Cocoa Layout：观察约束变化，找出布局代码的问题所在。

Network：跟踪 TCP / IP和 UDP / IP 连接。

Automations：创建和编辑测试脚本来自动化 iOS 应用的用户界面测试。

因为本次项目比较简单所以测试只用到了时间事件查看器

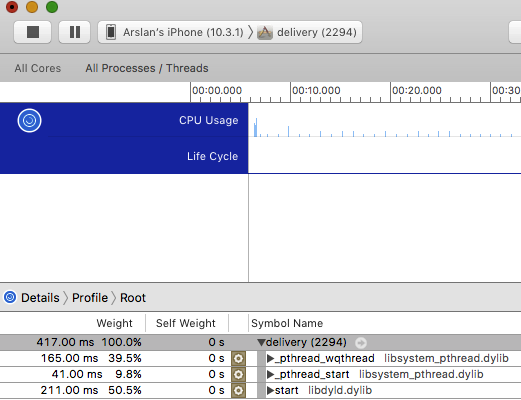


图5-2 软件性能测试

## 5.3测试用例

表5-1 测试用例

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 输入/动作 | 期望的输出/相应 | 实际情况 |
| 1 | 按下接收新任务按钮 | 界面上显示新的任务信息 | 服务端可以正常连接的情况下符合期望。  若无法连接服务器则退出程序。 |
| 2 | 按下揽收货物按钮 | 打开摄像头并显示QR码扫描框 | 符合期望 |
| 3 | 扫描到正确的QR码 | 揽收货物按钮变为开始派送按钮 | 符合期望 |
| 4 | 按下开始配送按钮 | 界面上开始显示经纬度信息，开始派送按钮变为签收按钮 | 符合期望 |
| 5 | 按下签收货物按钮 | 在经纬度信息对的情况下，软件状态返回到初始状态，并自动重新获取新任务 | 符合期望 |

# 第六章 总结

3月初我便开始了设计工作，因为当时不是很清楚有做服务端的人，所以我自己便设计了一个以我最熟悉的RESTful API为基础的服务器，然后围绕这个RESTful 交互方式 设计了APP端的网络交互，并决定使用开源库Alamofire。

4月，写完开题报告便着手APP的开发，在APP开发过程中遇到了许多问题，我便一个一个在网上搜集资料和解决方法。我将这些宝贵的资料全部记在笔记本上，尽量使我的资料完整、精确、数量多，这有利于论文的撰写。

4月底APP的开发基本而完成，在导师和学姐的指导下，顺利完成了中期报告，并开始修补APP的bug并开始写测试。

5月APP已经完成，开始着手论文的编写。当我终于完成了所有设计、编写代码、写论文、排版、校对的任务后整个人都很累，但同时看着电脑荧屏上的毕业设计稿件我的心里是甜的，我觉得这一切都值了。这次毕业论文的制作过程是我的一次再学习，再提高的过程。在论文中我充分地运用了大学期间所学到的知识。这次项目是我第一次接触到iOS开发，所以我也学习到了很多新东西。

APP虽然已经完成了所有功能，但任然让我有些缺憾，APP仍然有几个问题需要更做进步：

APP的UI有些单调，但是因为iOS对矢量图的支持欠佳，做不到不会出现图片错位变形的UI。而不使用图片的话，使用StoryBoard内置的修改功能比较繁琐。况且我对APP美观性的把握不佳，总是不好看。我想如果我的APP不是使用iOS原生语言，而是使用React Native之类的高级语言，应该会对UI有很大的改观。并且也能运行在Android机器上。

同样因为iOS系统的限制，我因为没有购买苹果开发者账号，导致我无法开启软件的许多后台功能，所以APP现在在使用过程中不能在后台发出警报。是一大缺憾。

**参考文献**

1. **刘智勇.** **基于Android手机的物流运输监控系统[D].** **东南大学.2013**
2. **李松庆.** **我国物流基础设施的建设策略分析.《商业研究》.** **2006年4期.** **P168—P170**
3. **林屹.** **物流企业信息管理系统的设计与实现[D].** **电子科技大学.2013**
4. **王虎.** **铁路物流管理信息系统设计及关键技术研究[D].** **中国铁道科学研究院.2013**
5. **甘泉 宫蕾 李刚.** **iOS 封闭性研究.《电子测试》.** **2013年8期**
6. **代强.** **CStore云存储系统客户端的设计与实现[D].** **电子科技大学.2013**
7. **楼明珠.** **基于SSH框架技术的网上花店系统的设计与实现[D].** **南昌大学.2011**
8. **吴福城.** **基于二维码的电子优惠券系统的设计与实现[D]. 厦门大学.2011**
9. **Simon Ng， Building a QR Code Reader in Swift. AppCoda，2014.12.09** [**http://www.appcoda.com/qr-code-reader-swift/**](http://www.appcoda.com/qr-code-reader-swift/)
10. **Alamofire. Alamofire介绍与使用方法https://github.com/Alamofire/Alamofire**

**致 谢**

随着论文渐渐地接近尾声，我的本科生生涯也将落下帷幕。在本次项目开发和论文撰 写过程中，得到了许多老师、同事、同学和家人的大力支持，在此，对你们深表感谢!

衷心感谢我的导师姚文斌教授和黄芬芬学姐。对于我的每一次的提问都不厌其烦 的一一回复。从论文的选题到结构规划，导师都进行了精心的指导，再一次向您表示真心谢意。

感谢北京邮电大学计算机学院的所有老师，感谢你们在我四年的学习、生活中给予的关怀和 帮助，感谢你们给我提供的良好学习环境。

特别感谢多年以来在背后一直默默的关心我的父母，一路走来，离不开父母的支持。 最后，感谢所有帮助过我和关心我成长的人。

1. [↑](#footnote-ref-1)