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。随着现代网上购物的普及，人们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。

本课题研究并设计实现的基于iOS的物流监控系统移动平台，使用了最新的Swift语言，并结合了RESTful API技术。本系统可以进行揽收货物、运输途中、签收货物三个主要功能模块。揽收货物模块可以从服务器获取最新的派件任务，并在拿到货物的时候通过扫描货物上的二维码的方式，验证是否揽收了正确的货物。运输途中模块则会自动调用机器上的GPS定位系统，每15秒将定位信息发送至服务器，并接受服务器发来的定位错误信息。这个物流监控系统通过自动化监控信息的上传，来提高了用户对物流监控的体验。

本文从现阶段我国物流系统的现状入手，阐述了一个好的电子化、互联网化的物流系统的重要性。借鉴结合许多材料，实现了一个基于iOS的物流监控系统移动平台。本文首先阐述了物流监控系统的开发背景、实现意义和研究现状，然后详细介绍了整套系统所用到的几个关键技术。然后从功能、需求方面进行分析，设计了一个简单的逻辑并对其进行了代码实现。最后进行了单元测试和性能测试。

**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. With the popularity of modern online shopping, people increasingly demanding the logistics monitoring, and actively explore the use of mobile phone logistics information sharing, sharing in the field of logistics has important practical significance.

This project research and design of the iOS-based logistics monitoring system mobile platform, using the latest Swift language, combined with RESTful API technology. The system can be carried out to collect goods, transport, sign the goods three main functional modules. The collection module can obtain the latest delivery task from the server and verify that the correct shipment is checked by scanning the two-dimensional code on the goods. In transit, the module will automatically call the GPS positioning system on the machine, every 15 seconds will be sent to the server positioning information, and accept the server sent to the positioning error message. This logistics monitoring system through the automated monitoring information upload, to improve the user experience of logistics monitoring.

This paper begins with the present situation of China's logistics system, and expounds the importance of a good electronic-based and Internet-based logistics system. Drawing a combination of many materials, to achieve a iOS-based logistics monitoring system mobile platform. This paper first elaborates the development background, the realization meaning and the research present situation of the logistics monitoring system, and then introduces the key technology used in the whole system. And then from the function, the needs of the analysis, the design of a simple logic and its code to achieve. Finally, unit testing and performance testing were performed.

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

目录

[第一章 绪论 1](#_Toc484164937)

[1.1背景和意义 1](#_Toc484164938)

[1.2物流信息化国内外研究现状 1](#_Toc484164939)

[1.2.1国外研究现状分析 1](#_Toc484164940)

[1.2.2国内研究现状分析 2](#_Toc484164941)

[1.3 本章小结 2](#_Toc484164942)

[第二章 开发环境和相关技术介绍 3](#_Toc484164943)

[2.1 iOS简介 3](#_Toc484164944)

[2.1.1 iOS操作系统介绍 3](#_Toc484164945)

[2.1.2 iOS开发工具Xcode 3](#_Toc484164946)

[2.1.3 iOS 开发语言Swift 4](#_Toc484164947)

[2.1.4 CocoaPods概述 8](#_Toc484164948)

[2.2 RESTful API 概述 8](#_Toc484164949)

[2.2.1 RESTful API 概念介绍 8](#_Toc484164950)

[2.2.2 RESTful 架构定义 9](#_Toc484164951)

[2.3 JHipster概述 10](#_Toc484164952)

[2.3.1 JHipster概念介绍 10](#_Toc484164953)

[2.3.2 JHipster后端应用 Spring boot 11](#_Toc484164954)

[2.4本章小结 12](#_Toc484164955)

[第三章 需求分析与设计 13](#_Toc484164956)

[3.1 需求分析 13](#_Toc484164957)

[3.1.1 可行性分析 13](#_Toc484164958)

[3.1.2需求分析 13](#_Toc484164959)

[3.2安全性设计 14](#_Toc484164960)

[3.2.1 JWT介绍 14](#_Toc484164961)

[3.2.2 JWT在项目中的应用 14](#_Toc484164962)

[3.3算法设计 14](#_Toc484164963)

[3.3.1计算快递员是否走出范围 14](#_Toc484164964)

[3.4 功能模块设计图 16](#_Toc484164965)

[3.5用例图 17](#_Toc484164966)

[3.6状态图 17](#_Toc484164967)

[3.6.1 总状态 17](#_Toc484164968)

[3.6.2收揽货物 18](#_Toc484164969)

[3.6.3派件中 19](#_Toc484164970)

[3.7数据模型概念结构 21](#_Toc484164971)

[3.7.1 快递员实体 21](#_Toc484164972)

[3.7.2 快件实体 21](#_Toc484164973)

[3.8 本章小结 22](#_Toc484164974)

[第四章 物流监控系统移动平台的实现 23](#_Toc484164975)

[4.1 开发平台的运行环境和搭建 23](#_Toc484164976)

[4.1.1 iOS开发平台的运行环境 23](#_Toc484164977)

[4.1.3 CocoaPods的安装与使用 23](#_Toc484164978)

[4.1.4 JHipster 开发环境的搭建 24](#_Toc484164979)

[4.2 各功能实现 24](#_Toc484164980)

[4.2.1工作流程的状态切换 24](#_Toc484164981)

[4.2.2保存状态，保证手机重启之后软件的可恢复性 25](#_Toc484164982)

[4.2.3 与服务器的信息交互 26](#_Toc484164983)

[4.2.4 保存时的错误处理与警告 27](#_Toc484164984)

[4.2.5信息同步到服务器上 29](#_Toc484164985)

[4.2.6 QR码读取 31](#_Toc484164986)

[4.3 本章小结 33](#_Toc484164987)

[第五章 测试 34](#_Toc484164988)

[5.1单元测试 34](#_Toc484164989)

[5.2性能测试 35](#_Toc484164990)

[5.3测试用例 36](#_Toc484164991)

[5.4 本章小结 36](#_Toc484164992)

[第六章 总结 37](#_Toc484164993)

# 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。举个例子，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。目前常用的快派送方式是通过写在快单上的快递单号进行分拣，派送。并且快递在运输过程中并没有可靠的监控系统，都是快递员手动在快递到达每个分拣中心时录入信息的，耗时耗力。

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[1]，美国是物流理念的发源地，并且其物流研究、设计和技术开发都是世界顶尖水平，它有着最成熟的物流管理理念，尤其是商贸流通和生产制造企业也十分重视现代物流的开发。美国很多企业都认为物流合理化的最重要途径便是物流信息化，比如：

1、机器录入信息：采用条形码、RFID等机器录入信息的技术，无需人工读取录入，提高工作效率和信息采集的准确性，并且是信息存储数字化，互联网化，实现无纸化订单录入、处理、跟踪、结算处理。不仅提高工作效率也能加快信息的共享。

2、信息共享：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、利用电商降低资源浪费：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。[2]

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。全球虽然都在对铁路、公路、水运、航空等基础设施的增加投入，并初步缓解了交通运输的全面紧张状况。但交通运输矛盾仍然比较突出，交通运输总体规模目前仍然很低，无法满足国际物流服务需求。[3]并且国际物流节点功能混乱，因为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。[4] 虽然在国内物流业已经是非常成熟的系统化规模化产业，但也无法抵挡人工成本上升的压力。成本控制越严格的行业，对人工成本上升的容错率越底，一件快递相隔千里，需要揽件，运输，分拣，派送等多个环节，每个环节的成本都要被精确控制。

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。[5]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快速发展时期。数据显示在2015年我国社会物流总额达到了219.2万亿元，“十二五”期间物流总额年均增长8.7%，社会物流需求规模持续扩大。

## 1.3 本章小结

本章介绍了物流监控系统的研究背景、研究意义与国内外研究现状。尤其是近年来，各种电子购物平台异军突起，像淘宝、京东等平台每年举行的六一八，双十一等狂欢活动，极大地推动了人们网上购物的热情，并对物流系统造成了前所未有的压力。而目前的物流管理方式远无法应对这种流量，并出现在这种活动前后物流速度减慢的情况，物流公司和物流管理越来越费力，急需智能化方式调高效率。所以使用新的技术，改良现有的物流监控系统及软件，尽可能的用自动化的方式来减少快递员的工作量，提高工作效率，是很重要的。

# 第二章 开发环境和相关技术介绍

## 2.1 iOS简介

### 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。[6]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。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。

Xcode 内置了GCC编译器，所以很显然Xcode支持C语言、C++语言等GCC可编译的语言，然而Xcode支持的语言远不止这些，它还支持Fortran、Objective-C、Objective-C++、Swift、Java、AppleScript、Python以及Ruby，并且Xcode除了简单的编译语言外还提供了Cocoa、Carbon以及Java以及其他几种编程模式。与苹果合作的协力厂商也还为Xcode提供了 GNU Pascal，Free Pascal, Ada, C#, Perl, Haskell 和 D语言的编译工具。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，这种二进制文件既可以在在PowerPC（然而因为PowerPC架构过老在2011年的Xcode4 版本中取消了对PowerPC支持）上也可以在基于Intel（x86以及amd64）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。

### 2.1.3 iOS 开发语言Swift

#### 2.1.3.1 Swift 概念介绍

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旨在做出比Objective-C语言更有弹性，更安全的错误代码（有点类似于Java8 和JS中的错误处理方式），并且错误处理变得更简洁。然而，它仍然支持与Objective-C语言中的许多核心概念，比如动态调度，广泛的后期绑定，可扩展编程和其他Objective-C语言中的类似功能，然而这些老的方式只是为了能让Objective-C语言的程序员能更好的上手Swift，在不久的将来这些老特性很可能会被新方式取代。为了安全起见，Swift有助于解决常见的编程错误，如空指针，并提供了各种语法糖，以避免产生像老式语言C Java那种链式崩溃的出现。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，通过这个系统你可以让一个已经定义好的类继承一个你自定义的协议，大大方便了程序开发。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。

#### 2.1.3.2 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，并且通过减少使用指针来减少性能开销，这与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）

1. 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。

然而，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。

1. 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。

1. 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。在这种情况下，！解开符就可以强制解开Optional，允许在其上进行方法调用。但是如果变量为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（？解开符），以测试包含的值是否为null，然后如果非null则则将其解开。在这种情况下，只有在变量不为null的情况下才会调用对他进行操作的类函数，从而抑制错误。但是没有调用对他进行操作的类函数的话我想执行另一个语句怎么办呢？

Swift 2为代码停止执行的情况引入了新的关键字guard：

guard其实就是一种省略了if的if else语句，如果变量为null则会执行guard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中如果有返回值的函数无法正确计算出返回值，则会直接返回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。

1. 储值类型

在许多面向对象语言中，对象在计算机内部分为两部分。对象的值放置在堆上的数据块存储，而该对象的名称（或“句柄”）则是一个指针。方法之间通过复制指针的值来传递对象，允许任何具有副本的任何人访问堆上相同的数据。恰恰相反，基本类型(char Int Double)的对象,句柄的值就是数据，而不是指向它的指针，并且数据通过复制直接传递给方法。这种访问对象为对象时的方法称为传引用调用，而对于基本类型来说，这些类型的值为传值调用。

这两个概念都有其优点和缺点。当数据较大时，对象存储很有用，如窗口的描述或文档的内容。在这些情况下，通过复制32位或64位指针地址，而不是复制整个数据结构，就可以访问该数据。然而，像整数这样的较小的值与指针地址的大小相同（通常都是一个字），因此传递指针没有任何优势。此外，传引用调用固有地需要一个取消引用操作，这可以在一些操作中产生显着的性能开销，尤其是在传引用调用对象是个数字类型的小值的时候。

与C＃类似，与大多数其他面向对象语言相比，Swift使用传引用调用或传值调用语义来提供对对象的内置支持，前者使用class声明，后者使用struct。 Swift中的struct几乎具有与class相同的功能、方法、实现协议以及使用扩展机制。因此，Apple将所有数据通常视为实例，而不是对象或值。然而，唯一的不同点是，结构不支持继承。

程序员可以自由地选择哪种语义更适合应用程序中的每个数据结构。较大的结构像windows将被定义为class，允许它们作为指针传递。像2D点一样，较小的结构可以被定义为struct，它们将被传值调用，并允许直接访问其内部数据，而无需引用。传值调用概念固有的性能改进使得Swift对于几乎所有常见的数据类型（包括Int和Double）以及通常由对象（如String和Array）表示的类型使用传值调用。所以程序员在程序中使用会被传值调用的类型能显著提高程序速度。

为了确保甚至最大的struct在它们被调用时都不会导致性能损失，Swift使用写入复制方法，以便只有当程序尝试更改其中的值时才会复制对象。这意味着各种访问器具有指向同一数据存储器的指针，但这在计算机内存管理单元（MMU）中远远低于语言级别。所以当所有数据都被物理存储在内存中的同一个实例时，在应用程序级别，这些值是看上去是分开的，并且只有在需要时写入时才会进行复制。

1. 面向协议的编程

Objective-C语言中的一个关键特性是它支持category，一种可以在运行时给类添加扩展函数的方法。category允许在当前项目中扩展类以添加新功能，而不需要继承这个类，甚至在少数情况下扩展的函数可以访问原始类中的源代码。举个例子，你可以给基础的NSString类添加中文拼写检查函数，这意味着你的应用程序中所有NSString的所有实例都可以获得中文拼写检查功能。该系统也被广泛用作组织技术，允许相关代码被收集到类库扩展中。 Swift继续保留了这个特性，尽管它们现在被称为扩展（extension，比category贴切了好多），并以关键字extension声明。与Objective-C语言不同，Swift还可以为现有实例添加新的属性访问方法，类型和枚举。

Objective-C语言中的另一个关键特性是其使用的协议，在大多数现代语言中被称为接口。协议保证继承了这个协议的类一定会实现这个协议中的所有方法，这意味着系统中的其他对象可以在支持该协议的任何对象上调用这些方法。虽然功能并不是完全相同，但这种方法通常用于现代面向对象语言中作为多重继承的替代。 Cocoa中协议的一个常见例子是NSCopying协议，它定义了一种在对象上实现深复制的方法copyWithZone。

在Objective-C语言和大多数其他实现协议概念的语言中，由程序员来确保在每个类中实现所需的方法。 Swift添加了使用扩展（extension）添加这些方法的能力，并使用泛型来实现它们。综合起来，这些允许协议被编写一次并支持各种各样的实例。此外，扩展机制可用于在其定义中没有继承该协议的对象，在他的扩展中继承该协议。

1. 库，运行时和开发

Swift使用与现有Objective-C系统相同的运行时系统，但Swift需要iOS 7或MacOS 10.9或更高版本才可编译使用。Swift和Objective-C代码可以在一个程序中使用，通过扩展的方式（将函数作为一个扩展库使用）Swift也可以和C代码在一个程序中使用。然而，C ++代码不能直接从Swift代码中调用，但是C++代码可以包装到C或者Objective-C代码中，然后Swift 可以调用这个包装函数。对于Objective-C代码，Swift可以很好地访问对象模型，可以继承Objective-C中的类，也可以对Objective-C中的类进行扩展。然而Objective-C却不能很好的访问Swift中的类，也不能继承Swift中的类。

为了帮助开发此类包含多个语言的程序以及继续使用现有代码，Xcode 6提供了一个半自动化系统，可以构建并维护一个带有桥接功能的头文件，以将Objective-C代码公开给Swift。这个额外的头文件简单地定义或导入项目的Swift代码所需的所有Objective-C代码中的类型，函数和变量。然后，Swift就可以引用在这些导入中声明的类型，函数和变量，就好像它们是用Swift写的。 Objective-C代码也可以通过导入这个头文件直接使用Swift代码。然而，并不是所有的符号都可以通过这种机制使用 - 使用特定于Swift的功能，如通用类型，非对象可选类型，复杂枚举，甚至是Unicode标识符，可能会导致Objective-C无法访问这个符号。

Swift对属性也有一些有限的支持，由开发环境读取的元数据，并不一定是编译代码的一部分。像Objective-C一样，属性使用@语法，但是当前可用的部分还很少。举个例子@IBOutlet属性，它将代码中的给定的值标记为可在Interface Builder（界面文件）中使用的连接（有点像QT中的信号和槽）。

1. 调试以及其他

Swift语言系统的一个关键元素是它可以使用读入-计算-输出循环（REPL）在开发环境中进行清晰的调试和运行，给它一种像Python这种脚本语言一样的交互式能力。 REPL的概念进一步被Xcode的Playground增强了。 这些是在Xcode环境中运行的可以在运行时响应代码或调试器更改的交互式视图工具。 如果某些代码随着时间的推移或其他范围内的输入值发生变化，则该视图可以与Timeline Assistant一起使用，以动画方式演示输出。

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。

### 2.1.4 CocoaPods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

### 2.2.1 RESTful 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。

### 2.2.2 RESTful 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：

1. 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。

1. 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。[9]

1. 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）。

1. 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

1. 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。

1. 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。

1. 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。

1. 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

### 2.3.1 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。

### 2.3.2 JHipster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，将服务器的底层代码与功能代码分开，大大提高了代码的可读性并降级了书写难度，程序员不再需要知道底层的网络传输知识，而是可以专注他们的经历在写代码上。[10]

#### 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：

1. Spring Core Container：这是Spring的基本模块，并提供Spring容器（BeanFactory和ApplicationContext）。
2. 面向方面的程序设计：能够实现跨领域的关注。
3. 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。
4. 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
5. 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
6. 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
7. 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
8. 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。
9. 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
10. 事务管理：统一多个事务管理API并协调Java对象的事务
11. 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
12. 测试：用于编写单元测试和集成测试的支持类

## 2.4本章小结

本章简单介绍了物流监控系统需要的技术，包含了苹果软件开发所需的iOS系统知识、Xcode编译器、Swift编程语言以及包管理器CocoaPods，并且简单介绍了RESTful API和使用了RESTful API的服务器框架JHipster，后端框架Spring boot。Xcode编译器是苹果软件开发的核心，这也是唯一一个可以对苹果系统进行有图形界面编译的编译器（其他所有编译器，如Qt等，都需要依赖Xcode才可编译），而Swift语言是目前为止最先进的开发语言，使用Swift编写苹果应用可以大大降低编写的难度。而服务器则使用Jhipster快速生成，后端RESTful API已经生成并可用，并且生成了可以进行简要管理的前端网页，在这里不详细阐述了。

# 第三章 需求分析与设计

## 3.1 需求分析

### 3.1.1 可行性分析

1. 技术可行性:

本次项目使用了Jhipster自动生成服务端框架，并使用Intellij IDEA编译器进行服务端Java代码的编写。客户端则使用了最新版Xcode用Swift编写原生的iOS软件，并使用iOS自带的CoreData数据库。除了Xcode需要有苹果系统开发环境的限制外，本项目在技术上完全可行。

2. 经济可行性:

开发一个物流监控软件的成本不大。软件编写所用的Intellij IDEA 有免费版可以使用。Xcode也是只要有苹果开发环境就能免费下载使用。而且本项目所使用的RESTful API使用JSON交互的方式也能大大减少软件对手机流量的使用。因此，本项目具备经济可行性。

3. 法律可行性:

本项目纯粹用于毕业设计所用，不用于任何商业用途。本项目中使用的Alamofire、SwiftyJSON、QRReader均为Apache 2.0许可下的开源库。因此，本项目具备法律可行性。

### 3.1.2需求分析

1. 功能需求

客户端应在揽货时可以扫描货物上的二维码，并通过服务端对扫描到的二维码进行验证。

验证信息成功后才可以开始派送。若验证失败，则根据服务器返回的正确任务信息更新任务，并需要重新扫描货物二维码。

派件途中客户端需要每15秒定位一次，并通过服务端对定位到的位置进行验证。

若验证失败，则需要通过语音和震动的方式提示快递员。

2. 可靠性和可用性需求

通过将信息存储在移动端本地数据库的方式，保证软件在手机异常关机、软件被退出等等情况下，仍能保证任务状态的恢复。

3. 出错处理需求

保证在软件出错，或者是服务端返回错误结果时，都能有语音及震动提醒。

4. 信息的采集输入

管理员/录入员：货物的收货地址、收货地址经纬度

快递员：货物的二维码

## 3.2安全性设计

### 3.2.1 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。

### 3.2.2 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。

## 3.3算法设计

### 3.3.1计算快递员是否走出范围



图3-1 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。 式3-1

## 3.4 功能模块设计图

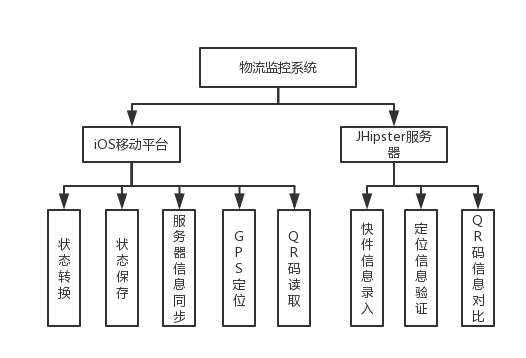


图3-2功能模块设计图

## 3.5用例图

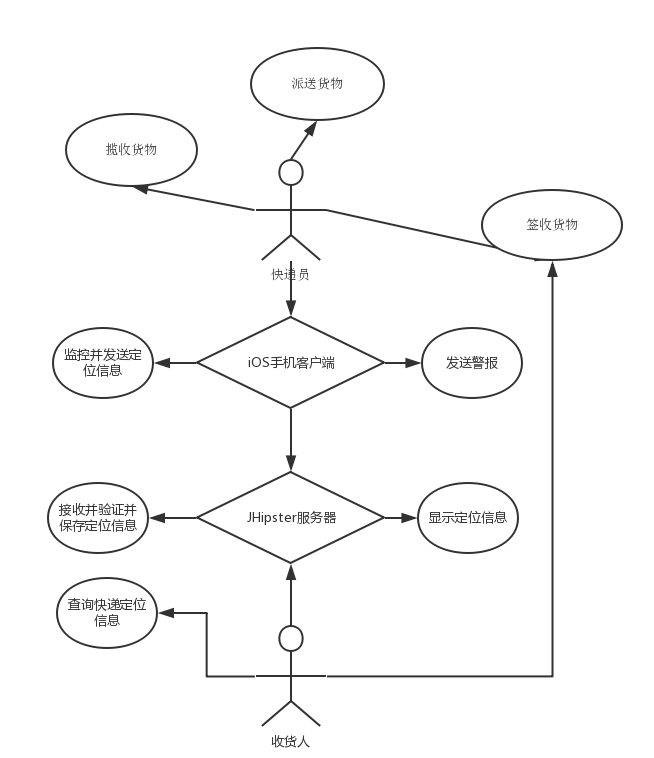


图3-3用例图

## 3.6状态图

### 3.6.1 总状态

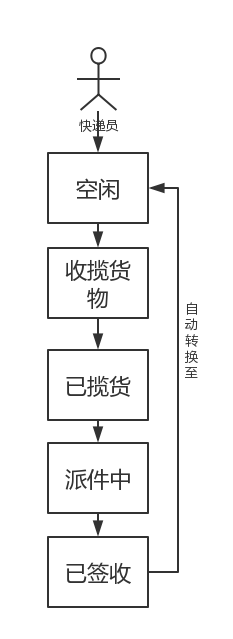


图3-4 总状态图

### 3.6.2收揽货物

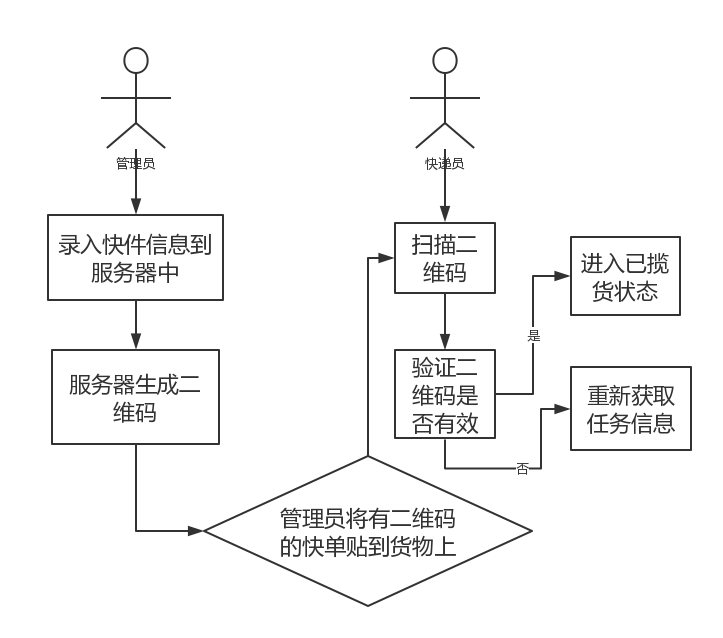


图3-5 收揽货物状态图

### 3.6.3派件中

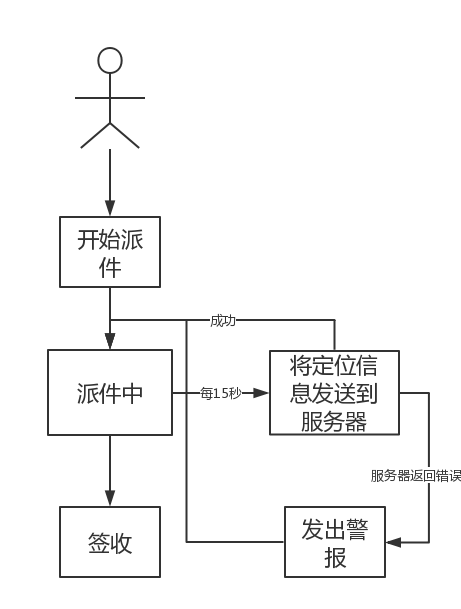


图3-6 派件中状态图

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。

## 3.7数据模型概念结构

数据模型库概念结构设计具体如下图所示：

### 3.7.1 快递员实体

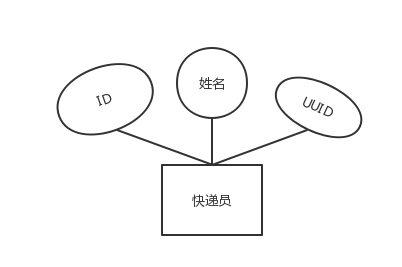


图3-7 快递员实体图

表3-1 快递员表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 说明 | 数据类型 | 宽度 |
| id | 数据库编号（主键） | Long | 无 |
| name | 姓名(可为空) | Char | 250 |
| uuid | 标识符 | Char | 250 |

因为姓名在本此设计中没有用到，所以可为空。

本次设计中，直接将手机的UUID作为辨别快递员的标识，没有制作登录功能，所以添加了UUID这一项。若要编写快递员登陆功能，则可将name改为不可为空，将UUID替换为password。

### 3.7.2 快件实体

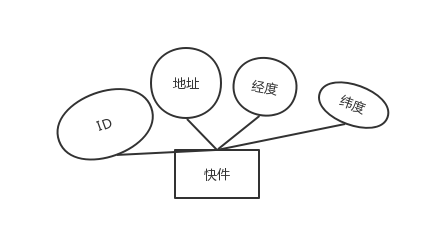


图3-8 快件实体图

表3-2 快件表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 说明 | 数据类型 | 宽度 |
| id | 数据库编号（主键） | Long | 无 |
| address | 地址 | Char | 250 |
| longitude | 经度 | Double | 无 |
| latitude | 纬度 | Double | 无 |

在验证手机定位信息时，手机定位的精度可能远高于运维人员向数据库中录入的经纬度的精度，所以在服务器中验证时一般是通过计算两者的差值小于10的-6次方来判定是否相等。

## 3.8 本章小结

本章对系统的技术、经济和法律进行了可行性分析。提出对系统的需求分析，其中包括了对系统的功能需求、可靠性和可用性需求、出错处理需求和信息的采集输入。并简要介绍了对达到安全性需求打算采用的算法和检验快递员位置的算法。并根据对系统设计的想法画出了系统ER图。再通过ER图设计出了软件状态流程和服务端数据库表。

# 第四章 物流监控系统移动平台的实现

## 4.1 开发平台的运行环境和搭建

### 4.1.1 iOS开发平台的运行环境

表4-1 iOS开发平台的运行环境需求表

|  |  |
| --- | --- |
| 需求名称 | 详细要求 |
| CPU | Core I5及以上 |
| 内存 | 4G（不使用模拟器），8G(使用模拟器) |
| 硬盘 | 64G |
| 显示器 | 720P及以上，（使用模拟器建议1080P及以上） |
| 网络 | 8M以上 |
| 操作系统 | macOS 或 Mac OS X 10.11及以上 |
| 开发工具 | Xcode |

**4.1.2 Jhipster服务端的运行环境**

表4-2Jhipster服务端运行环境需求表

|  |  |
| --- | --- |
| 需求名称 | 详细要求 |
| CPU | Core I5及以上(开发)，单核及以上（发布） |
| 内存 | 4G（开发），1G(发布) |
| 硬盘 | 16G（开发），1G(发布) |
| 显示器 | 720P及以上（开发），只需要有可连接终端(发布) |
| 网络 | 8M以上 |
| 操作系统 | 任何可运行Intellij IDEA的电脑（开发），任何可运行JAVA的电脑（发布） |
| 开发工具 | Intellij IDEA |

### 4.1.3 CocoaPods的安装与使用

#### 4.1.3.1 CocoaPods的安装

CocoaPod是用Ruby写出来的所以，安装CocoaPods非常简单，因为Mac系统里面已经内置了Ruby以及Ruby的包管理器。所以终端中只要执行下面一句话即可。

sudo gem install cocoapods

#### 4.1.3.2 CocoaPods在项目中的使用

要使用CocoaPods创建一个新项目，按照以下简单步骤操作就可以了：

按照通常的方式在Xcode中创建一个新项目。

打开一个终端窗口，$ cd进入你的项目目录。

创建一个Podfile文件。 这也可以通过运行$ pod init做到。

打开你的Podfile 第一行应该指定支持的平台和版本。

platform :ios, '9.0'

这样，在安装包依赖时，CocoaPods会检测你要安装的包是否兼容你要编译的版本。如果包没有兼容这个平台的版本则会报错提示你。

为了使用CocoaPods，您需要定义Xcode目标来将它们链接起来。 所以，例如，如果你正在编写一个iOS应用程序，这将是您的应用程序的名称。例如，若果你的项目名称为MyApp 并且需要安装ObjectiveSugar库到你的项目里，则要写

target 'MyApp' do

pod 'ObjectiveSugar'

end

保存你的Podfile之后就可以运行$ pod install来安装你的包依赖。

打开创建的MyApp.xcworkspace工作区文件便可以使用刚才安装的库。

因为每一个库都是一个单独的项目文件，不属于你的项目，所以为了能在你的项目中找到这些库，你需要打开工作区文件再想你的项目中写代码，而不是项目文件。

### 4.1.4 JHipster 开发环境的搭建

使用JHipster之前需要安装Java和Nodejs。Java和Nodejs都可以上相应的网站官网下载安装包一键安装，所以不再赘述。这里主要解释一下JHipster的安装。

安装过了Java和Nodejs之后。还需要安装git。苹果操作系统如果已经安装过Xcode，那么电脑里就已经有Git了。Windows可以上Git for Windows网站下载安装包。Linux操作系统可以在终端中执行

sudo apt(或者是yum,视系统而定) install git

来安装git。

上述软件全部安装好之后，确保nodejs的环境变量已经正确配置。然后通过nodejs来安装yarn。

npm –g install yarn

yarn包管理器安装好之后就可以通过yarn安装yo和JHipster

yarn global add yo

yarn global add generator-jhipster

然后就可以通过yo调用jhipster生成器了

yo jhipster

然后，jhipster就打开了，jhipster会在当前的终端中打开一个仿GUI界面一步一步指导您在当前文件夹下生成一个JHipster服务器。

## 4.2 移动平台各功能实现

### 4.2.1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）

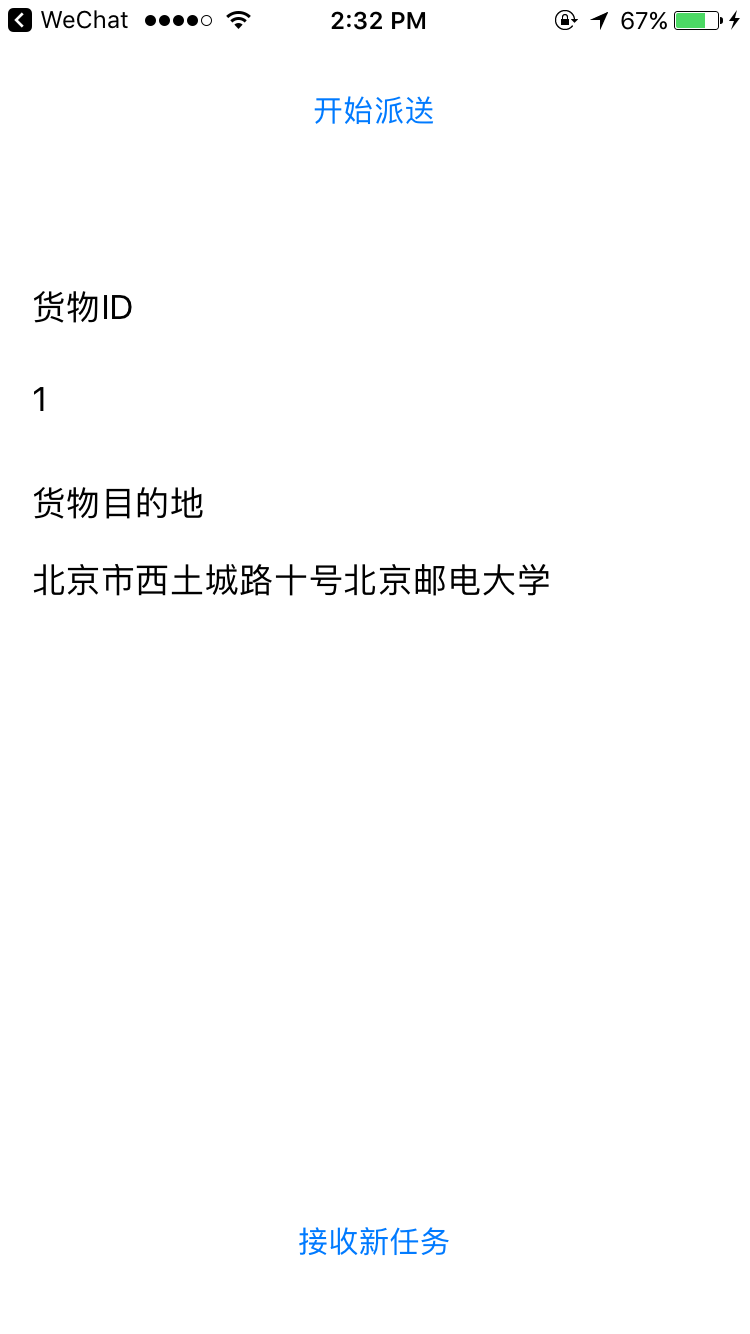
 

图4-1 软件中状态转换的实现

### 4.2.2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。

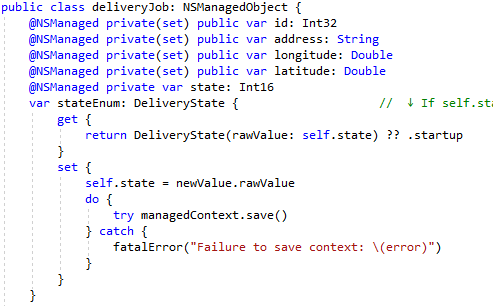


图4-2 软件中受管理对象的实现

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。

### 4.2.3 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。

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。[15]

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。

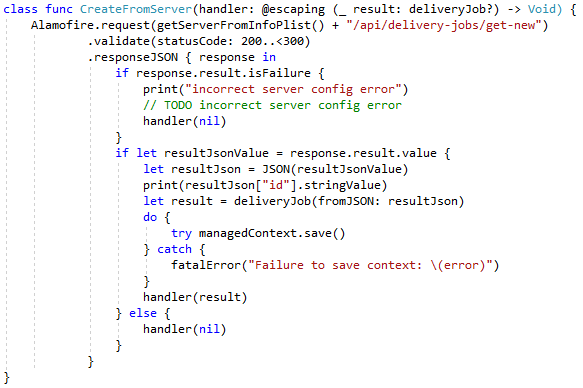


图4-3 软件中RESTful式交互的实现

### 4.2.4 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。

而警报声我们则可以用系统声音服务(System Sound Services)提供的AudioServicesPlayAlertSound函数，系统声音服务可以用于播放不超过30秒的声音文件，并且他支持的文件格式也非常有限，几乎只能播放系统内置声音以及CAF、AIF和 使用PCM或IMA/ADPCM数据的WAV文件 ，但是因为我们只是要做简单的警报功能，所以这个函数完全够用了。

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。在这里，我们要播放警报声音，需要使用的两个函数是SystemSoundID和 AudioServicesPlayAlertSound。我们通过声音ID用SystemSoundID获取声音文件，再用AudioServicesPlayAlertSound播放。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。

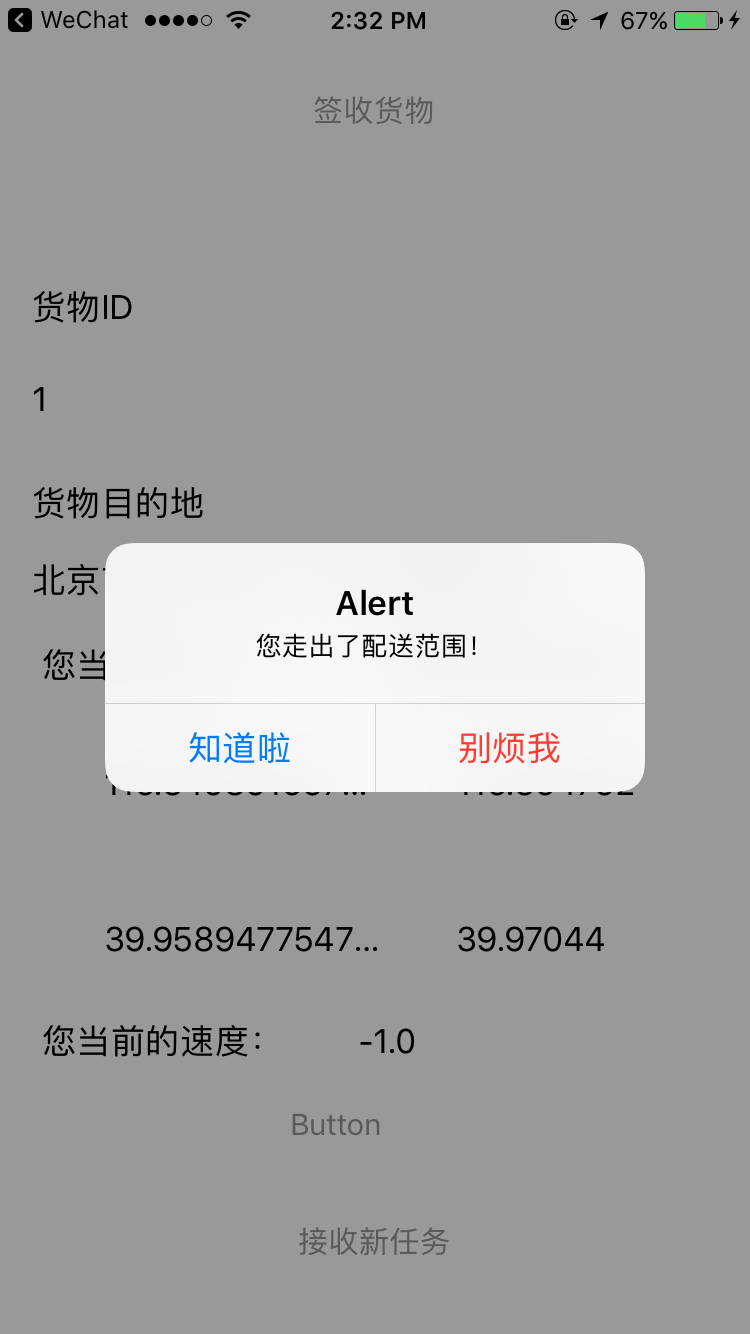


图4-4 软件中警报声的实现

为了不使软件反复提醒，影响使用体验，警报弹出之后，若点击确认，应该留出5分钟或者更长时间的空闲时间，然后再次启用监控。在这里需要注意的是NSTimer是无法被暂停的，所以为了达到暂停监控的效果，最好的方法便是，终止当前的监控NSTimer进程然后在新线程计时5分钟之后重新开启NSTimer。

### 4.2.5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，其中CLLocationManager类就是用于定位管理的类，他提供经纬度信息和高度信息，并且可以监控设备进入或者离开某个区域，也可以获取设备的运动方向。

我们在项目中继承了CLLocationManagerDelegate类。CLLocationManagerDelegate是CLLocationManager类的委托协议类，该类中的CLLocation值 封装了设备的位置和高度信息；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。

然后需要注意的是，iPhone内置的速度监测接口，在手机静止不动，或者运动时间过短时，会返回默认值 -1。



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。

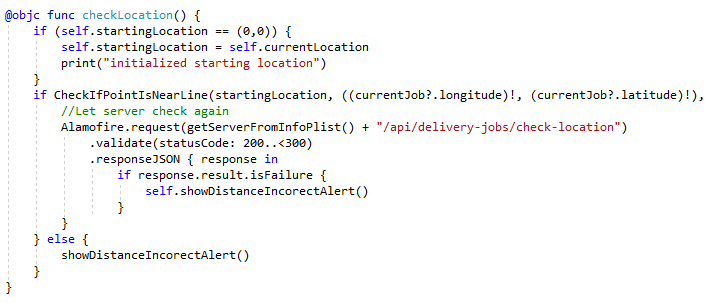


图4-6 软件中同步经纬度信息的实现

### 4.2.6 QR码读取

针对iOS7及之前的设备，若要实现扫描二维码功能，需要自己编写扫描算法，一般程序员都会借助第三方写好的库。常用的是 ZBarSDKa 和 ZXingObjC ，但是IOS7之后，系统的AVMetadataObject类中，苹果已经为我们提供了解析二维码的接口。因为项目中使用的是最新的Swift，我们也可以直接使用这个库，使用原生API扫描和处理的效率非常高，远远高于之前的那些第三方库。

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。[14]

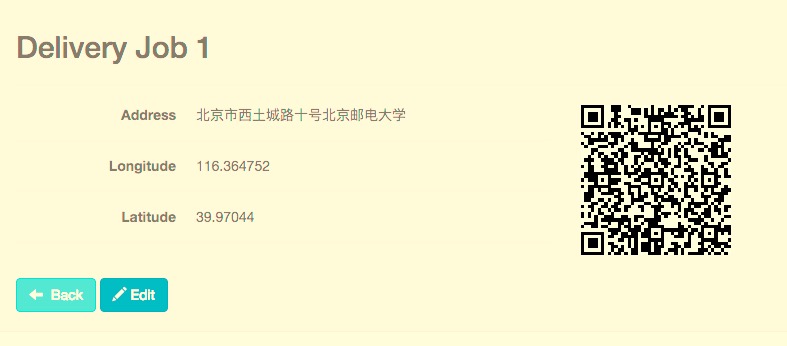


图4-7 服务器中二维码生成的实现

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。

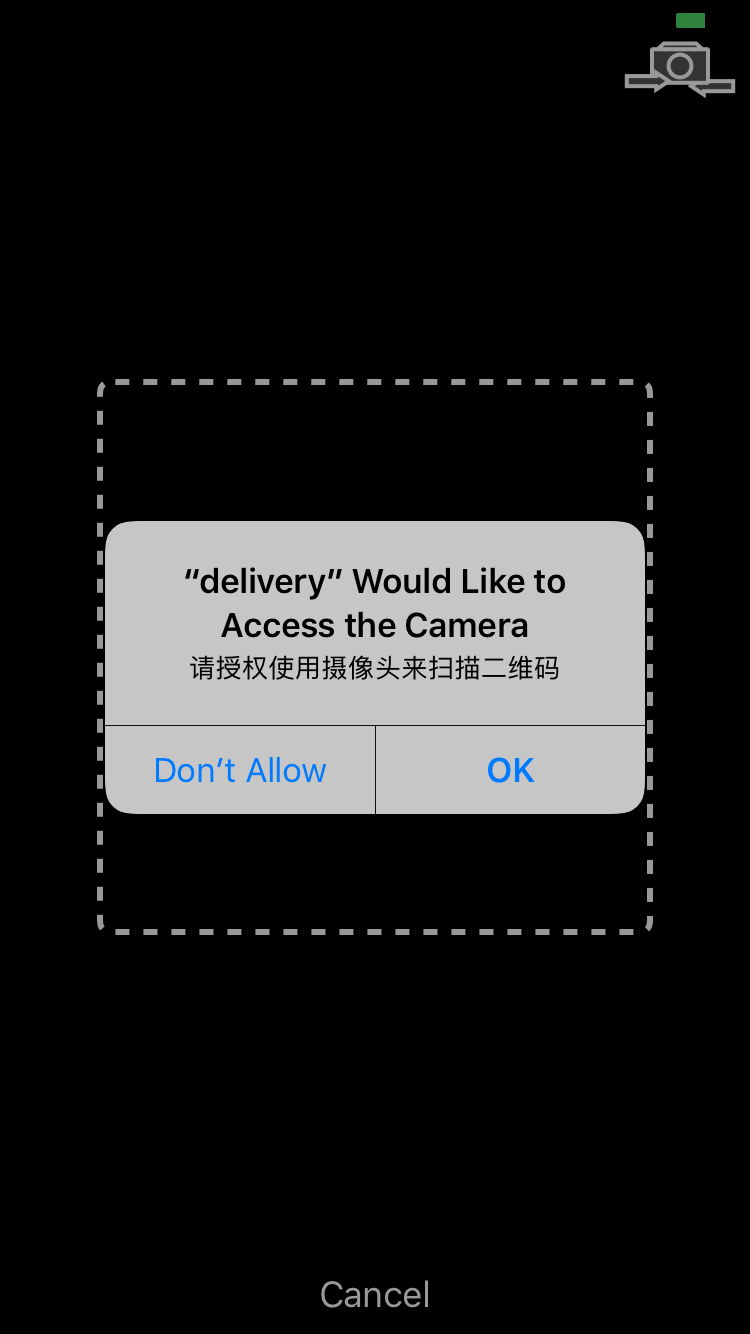
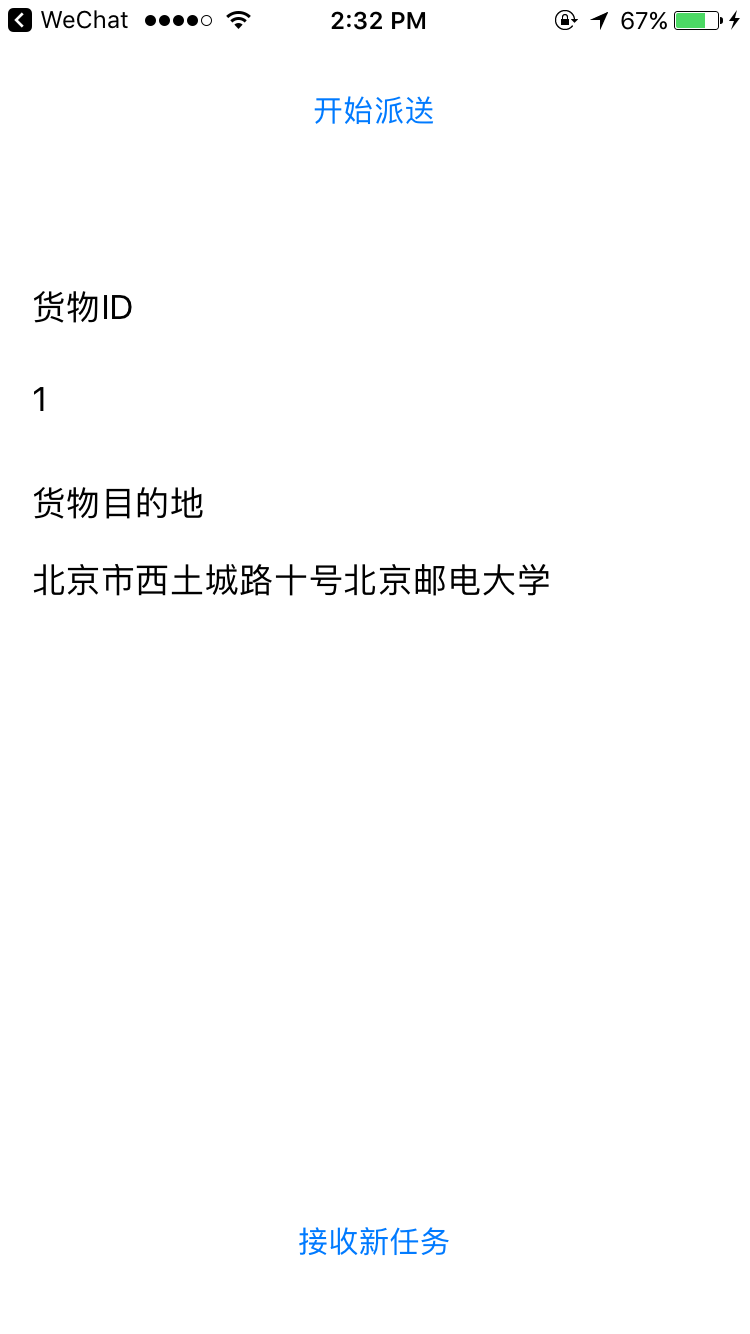
 

图4-8 软件中二维码读取的实现

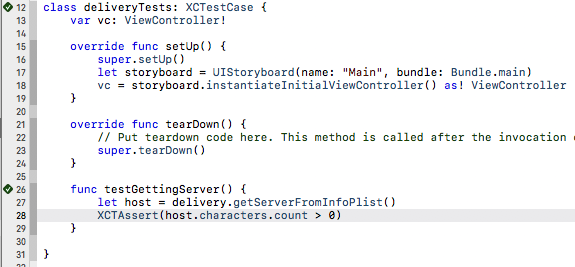
## 4.3 本章小结

本章重点描述了物流监控系统移动平台的实现，每一节解释了软件的一个功能模块的的实现方式以及实现效果。

# 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。



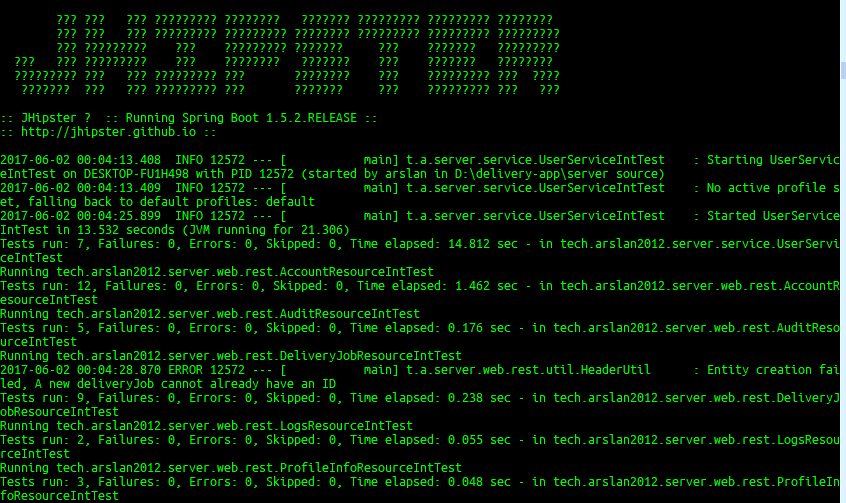


图5-1 软件单元测试

## 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

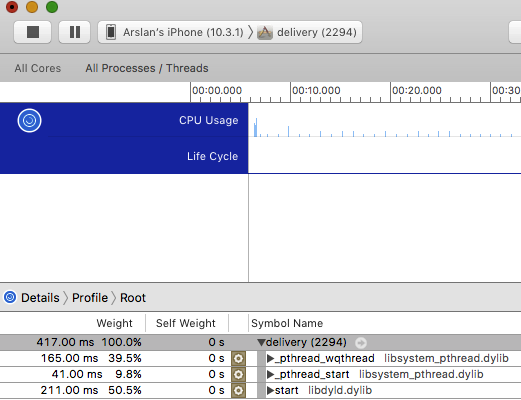


图5-2 软件性能测试

## 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 |

## 5.4 本章小结

本章主要介绍了物流监控系统系统移动平台的测试环节。列出了测试内容以及展示测试结果。对部分结果进行了截图展示。

# 第六章 总结

通过对系统进行需求分析，找出物流监控问题的各个要素以及约束条件，提出物流监控问题的求解难点和优化目标，并设计出物流监控系统所需要的功能结构以及操作流程。并分析了各个功能所需要的技术，使用CocoaPods导入所需要的开源库，并使用Swift语言在Xcode编译器中编写代码，实现iOS的平台物流监控系统，再通过测试结果对软件进行改进。

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。而不使用矢量图片的话，使用StoryBoard内置的UI设计功能操作过程比较繁琐。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. **柳贺.国内外物流信息化发展.2011.http://www.istis.sh.cn/list/list.aspx?id=7157**
2. **刘智勇.** **基于Android手机的物流运输监控系统[D].** **东南大学.2013**
3. **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**
4. **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**
5. **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**
6. **甘泉 宫蕾 李刚.** **iOS 封闭性研究.《电子测试》.** **2013年8期**
7. **张云鹏. 浅谈iOS系统的使用心得.《通讯世界》. 2016年8期**
8. **pican.** **iOS系统框架概述.间书.** **http://www.jianshu.com/p/0f2ab4b09e8b**
9. **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**
10. **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**
11. **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**
12. **于弼君 谭裴.基于二维码的网络流媒体播放系统设计与实现.《电信工程技术与标准化》.2011年2期**
13. **王子毅.基于uClinux和WEB的无线监控系统研究[D]. 武汉理工大学.2007**
14. **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/)
15. **Alamofire. Alamofire介绍与使用方法.https://github.com/Alamofire/Alamofire**

**致 谢**

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。

而现在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。