题目：基于互联网环境下的车辆管理系统设计

（1、2）中英文摘要（750字）：

随着互联网技术以及数字化城市建设的发展，城市规划的作用更显的尤为重要。由于我国的城市化进度不断加快，人口向城市的大量集中带来的不仅仅是经济的发展，社会资源的合理分配带来的问题也更加突出。

经济的发展带来的是居民的生活水平的提高，公共交通已经无法满足日常人们的出行要求，家用型汽车便随着市场需求的增加销量更是逐年增加。然而车辆的增加给城市的交通带来的运输压力也是不容小觑，城市道路拥堵、交通安全等问题日益突出。解决这些车辆与交通问题的方式中，车联网的出现便体现出了数字化与物联网+的优势。

本文主要表述了现代化城市交通中，基于互联网环境下的车辆综合管理云平台系统，该系统包含了车辆自身、车辆与车辆、车辆与互联网之间构成的一个巨大的信息交互平台。车辆通过自身的传感器、摄像头、车载雷达、车机电脑等装置将车辆自身的数据进行采集并通过互联网上传到系统中，通过云计算以及大数据处理将车辆的信息、状态进行分析，从而得到一个较为科学的、合理的使用结果，最终减少车辆的安全以及驾驶等问题。

该系统面向的用户群体主要是涉及到运输行业的企业，如快递、货运等。该系统可将企业下所有的车辆进行管理，并对车辆的信息进行分析后，得出一个可视化的分析结果。管理人员可随时随地对企业下的车辆进行调度分配，也可以查看车辆的各种运营风险和已发生的、正在发生的风险行为，为提高车辆的使用效率以及驾驶员的安全意识提供了极大的帮助。

该系统采用手机APP的形式开发，在iOS操作系统上运行使用，使用MVC开发模式与Swift开发语言进行软件系统开发。

关键字：车联网，车辆管理，iOS，Swift

Abstract

With the development of Internet technology and digital city construction, the role of urban planning becomes more and more important. Due to the acceleration of urbanization in China, the large concentration of population into cities not only brings about economic development, but also causes more prominent problems in the rational distribution of social resources.

With the development of economy, the living standard of residents has been improved. Public transport has been unable to meet the daily travel requirements of people, and the sales of family cars have increased year by year with the increase of market demand. However, the increase of vehicles brings more and more pressure to urban traffic, urban road congestion, traffic safety and other problems are becoming more and more prominent. Among the solutions to these vehicles and traffic problems, the emergence of Internet of vehicles reflects the advantages of digitalization and Internet of Things +.

This paper mainly describes the vehicle integrated management cloud platform system based on the Internet environment in modern urban traffic, which contains a huge information interaction platform composed of the vehicle itself, vehicles and vehicles, and vehicles and the Internet. Vehicles through own sensors, cameras, automotive radar, car machine computer device will of its own data are collected and uploaded to the system, via the Internet through cloud computing and big data processing, the information of vehicle, the state is analyzed, so as to get a more scientific and reasonable use as a result, eventually reduce the vehicle's safety and driving.

The users of this system are mainly enterprises involved in the transportation industry, such as express delivery, freight, etc. The system can manage all the vehicles under the enterprise and obtain a visual analysis result after analyzing the vehicle information. Managers can dispatch and assign vehicles under the enterprise at any time and any place, and can also check various operation risks and existing and ongoing risk behaviors of vehicles, which provides great help for improving the use efficiency of vehicles and drivers' safety awareness.

This system is developed in the form of mobile phone APP and is used on iOS operating system. MVC development mode and Swift development language are used for software system development.

Key words: vehicle networking, vehicle management, iOS, Swift.

绪论

随着社会经济与互联网的高速发展，传统企业也受到了不小的冲击。各企业在互联网的推动下纷纷改革创新，车联网更是首当其冲。经济的发展带来的是消费能力的不断上涨，各个车企的销量更是连年创新。车联网从刚开始的物联网到如今的智能车载系统的发展过程，充分的体现出了车企+互联网不断改革与创新。

（4）研究背景说明（1500字）：

传统的车联网定义是指装载在车辆上的电子标签通过无线射频等识别技术，实现在信息网络平台上对所有车辆的属性信息和静、动态信息进行提取和有效利用，并根据不同的功能需求对所有车辆的运行状态进行有效的监管和提供综合服务的系统。而如今的智能车载系统的车联网是以车内网、车际网和车载移动互联网为基础，按照约定的通信协议和数据交互标准，在车-X（X：车、路、行人及互联网等）之间，进行无线通讯和信息交换的大系统网络，是能够实现智能化交通管理、智能动态信息服务和车辆智能化控制的一体化网络，是物联网技术在交通系统领域的典型应用。

车联网系统的开发实际上是集智能硬件、智能软件、车辆云服务为一体的多面型服务系统，是各车企进步发展的必经之路。车联网系统不仅能提升用户的驾驶感受，更充分体现了互联网技术的多方面应用。

随着企业的业务扩张与人员的不断流动，企业的用车成本也在不断增加。当一个企业员工需要使用车辆时，通过人工手动查询、登记、调度，以及车辆使用结束后的信息统计与归纳等一套流程，既费时费力又容易出错，或者出现车辆信息不同步导致用车请求无法完成。手续流程复杂漫长、车辆使用效率低下、业务拓展难度增加、车辆使用汇总不明确不清晰等，这些问题在无形中限制着企业的各项发展。车辆管理系统的开发与投入，大大减少了企业的各项成本，提高了各车辆的使用效率。

车辆管理系统主要面向车企、运输公司以及涉及到因公使用企业车辆等政企。欲在帮助企业对车辆使用的管理控制。并且随着5G的到来，工业和信息化部更是印发了《车联网（智能网联汽车）直连通信使用5905-5925MHz频段管理规定（暂行）》文件，规划了5905-5925MHz频段共20MHz带宽的专用频率资源，用于基于LTE演进形成的V2X智能网联汽车的直连通信技术，为车联网的进一步发展提供了便利。

基于互联网的特性，车联网从诞生起就具有较为丰富的产业链，由上游的设备原件系列，到中游的汽车、软件、终端的相关企业，再到下游的车机、媒体运营等企业。位于中游的车联网软件可以说起到了承上启下的作用。随着智能网联汽车的逐步发展，汽车各部件也有了较高的增长。根据企查查平台数据，经营辅助驾驶、地图导航、语音服务等业务的公司企业已经超过了1000家。其中，有广东、上海、浙江等珠三角、长三角地区分布最多。

车联网产业在相关政策、技术创新、基础设施建设等因素的影响下，将保持快速增长。由于我国的人口红利，汽车市场有着巨大的市场，技术的换代使得新车搭载车联网系统的比例在不断提升，成联网的渗透率也在不断的提升。随着技术、经济的发展，用户对车联网的功能付费意愿也在不断的提高。同时由于5G技术的推广及应用，用户增值付费提升等因素，市场将迎来爆发式的增长。

本文的组织结构

本文对车辆管理系统的设计与开发过程进行了详细的描述，包括开发使用的技术、需求分析、系统设计、系统实现、系统测试等几个方面。

本文主要分为以下几个章节：

1. 绪论。通过描述城市发展中遇到的各种社会性问题，引出当前城市日常生活中常见的交通问题，从而表现出来车联网以及车辆管理系统存在的必要性。
2. 文献综述。介绍了车辆管理系统的开发环境以及开发工具与平台等。
3. 需求分析。
4. 系统设计。主要描述了车辆管理系统的设计思想、系统架构、功能模块等。
5. 系统实现。

（5）文献综述（1500字）：

iOS软件开发

iOS是由苹果公司开发的移动操作系统，最早于2007年1月9日的MacWorld大会上公布。到目前为止，iOS软件开发已成为最热的IT行业之一。依托于苹果公司的闭环生态环境，iOS软件开发所用的开发语言以及开发工具都具有唯一性，这种封闭的方式既保证了iOS系统下应用程序的安全性，又能有效的管理应用程序市场不受恶意软件的扰乱。在2014年之前，软件开发语言为Objective-C，也就是面向对象的C语言，在C语言的基础上引入了面向对象的编程思想，也称OC、ObjC。在2014年，苹果公司推出了Swift开发语言，Swift 是一种非常好的编写软件的方式，无论是手机，台式机，服务器，还是其他运行代码的设备。它是一种安全，快速和互动的编程语言，将现代编程语言的精华和苹果工程师文化的智慧，以及来自开源社区的多样化贡献结合了起来。

OC开发语言简介：

2007年，苹果公司推出Objective-C 2.0，并将此作为iOS、MacOS平台的基础开发语言。由于iPhone手机的推出并大受欢迎，iOS开发者的人数也跟着呈现爆发式增长。因此OC语言与2011年登上了TIOBE平台年度最热门开发语言的奖台。OC作为一门面向对象的编程语言，同样也具有面向对象的语言的特性：继承、封装、多态，与Java语言有很多共通处。

Swift简介：

Swift开发语言是苹果公司与2016年推出的一门全新的开发语言，不同于Objective-C，Swift完全为苹果公司独立开发完成的一门编程语言。相比Objective-C来说，Swift更快、更安全，也更方便。Swift 对于初学者来说也很友好。它是一门满足工业标准的编程语言，但又有着脚本语言般的表达力和可玩性。它支持代码预览（playgrounds），这个革命性的特性可以允许程序员在不编译和运行应用程序的前提下运行 Swift 代码并实时查看结果。Swift 代码被编译和优化，以充分利用现代硬件。语法和标准库是基于指导原则设计的，编写代码的明显方式也应该是最好的。安全性和速度的结合使得 Swift 成为从“Hello，world！”到整个操作系统的绝佳选择。

Swift 将强大的类型推理和模式匹配与现代轻巧的语法相结合，使复杂的想法能够以清晰简洁的方式表达。因此，代码不仅更容易编写，而且易于阅读和维护。而且经过多年的发展，Swift逐渐趋于完善与稳定，并且随着新特性和功能的出现，Swift大有替换Objective-C的趋势。

Xcode简介：

Xcode 是运行在MacOS操作系统上的集成开发工具（IDE），由苹果公司开发。Xcode是开发MacOS和iOS应用程序的最快捷的方式。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前身是NeXT的Project Builder，并且Xcode支持多种开发语言，如：C、C++、Objective-C、Java、AppleScript、Python以及Ruby，还提供Cocoa、Carbon以及Java等编程模式。所以，无论程序员使用什么开发语言，都可以使用Xcode轻松完成编程任务。

CocoaPods简介：

CocoaPods 是一个负责管理 iOS 项目中第三方开源库的工具。CocoaPods 的项目源码在 GitHub（ https://github.com/CocoaPods ）上管理。开发 iOS 项目不可避免地要使用第三方开源库，在使用第三方库时，除了需要导入源码，集成这些依赖库还需要我们手动去配置，还有当这些第三方库发生了更新时，还需要手动去更新项目，这就显得非常麻烦。而 CocoaPods的出现使得我们可以节省设置和更新第三方开源库的时间，通过CocoaPods，我们可以将第三方的依赖库统一管理起来，配置和更新只需要通过简单的几行命令即可完成。

在使用 CocoaPods 之前，开发项目需要用到第三方开源库的时候，我们需要：

1. 把开源库的源代码复制到项目中。
2. 添加一些依赖框架和动态库。
3. 设置-Objc，-fno-objc-arc等参数。
4. 管理他们的更新。

在使用 CocoaPods 之后，我们只需要把用到的开源库放到一个名为 Podfile 的文件中，然后执行 pod update 就可以了，CocoaPods 就会自动将这些第三方开源库的源码下载下来，并且为我们的工程设置好相应的系统依赖和编译参数。

CocoaPods 的原理是将所有的依赖库都放到另一个名为 Pods 的项目中，然后让主项目依赖 Pods 项目，这样就把源码管理工作从主项目移到了 Pods 项目中。

面向对象简介：

面向对象程序设计（Object Oriented Programming，OOP）是一种计算机编程架构。OOP的一条基本原则是计算机程序由单个能够起到子程序作用的单元或对象组合而成。OOP达到了软件工程的三个主要目标：重用性、灵活性和扩展性。OOP=对象+类+继承+多态+消息，其中核心概念是类和对象。

当要解决一个问题时，面向对象会把事物抽象成对象的概念，就是说这个问题里面有哪些对象，然后给对象赋一些属性和方法，然后让每个对象去执行自己的方法，问题得到解决。

MVC简介：

MVC全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。

第三章 需求分析

（6）研究路线技术路线以及系统实现过程（10500字）：

需求分析是在系统开发前的一项重要内容，同时也是软件开发方向的一个总体指南。在需求分析的阶段需要完成的工作实在结合实际与需求的情况下，理论上总结出该系统具体需要实现什么功能，确认后完整清晰的记录下来，以便在日后开发过程中，按照实际的需求逻辑莱开发。做好需求分析，也就是做好了系统开发的良好开端。

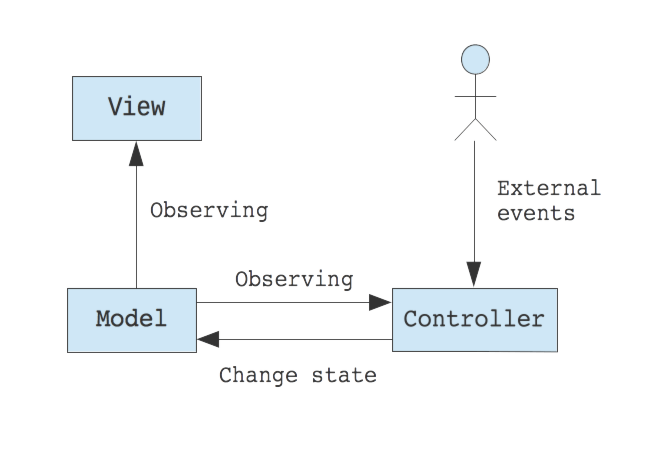
可行性分析

可行性分析是对系统开发是否可行的一种有效分析，主要在技术、经济、社会这三个方面进行分析。

在分析产品技术可行性时要逐项分析产品技术指标，在开发一个系统之前，首先要明确要做的是一个什么样的系统，市场上是否已经有同类产品被投放使用了，我们要做的系统与其他类似的产品有什么差异或特点；在实现该系统的功能过程中，有没有什么技术上的风险，有的话该如何规避；该系统在开发完成后，是否方便使用，是否对用户的使用有门槛限制；该系统开发完成后，需不需要依赖其他的平台环境；这些问题都是我们需要在系统开发之前就要确定好的，只有确定了这些问题，才能更好的完成我们需要的产品。

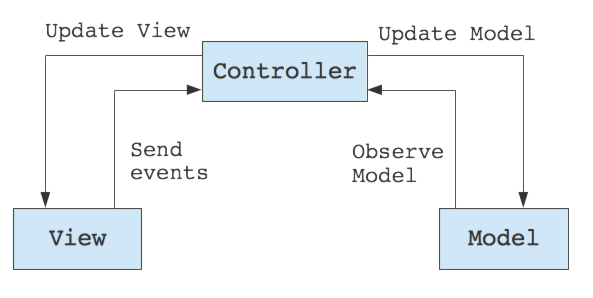
技术可行性分析

本系统采用的架构模式为苹果的MVC模式，这种模式与传统的MVC模式还是有一些区别的。传统MVC的缺点之一是组件之间的强互连性，这使单元测试变得复杂。 在现代程序中，控制器，视图和视图模型的层次结构愈发复杂，它们被认为是基于MVC的应用程序，因此实际上无法进行单元测试。业务模型的“增厚”。为什么会这样呢？ 视图可以具有复杂的状态。 例如，文本输入框的输入字段验证的逻辑及其取决于验证结果的文本颜色的设置，此时视图的状态不能直接保存在视图模型的字段中，也不能在IDE中进行设置。那么，在哪里“转移”这种状态呢？模型和控制器中可以吗？在传统的MVC中，控制器不应保存视图的状态，因此这些复杂的状态需要在Model中实现。因此，除了域模型之外，该模型还包括部分文本输入ViewModel。



为了适应传统MVC并解决其缺点，苹果重新构建了MVC架构，实际上是在传统的MVC的基础上构建了Cocoa和CocoaTouch框架。 在苹果的MVC下，模型与传统MVC中的模型相同，并且是主动模型（即在观察者的帮助下通知其状态的变化）。

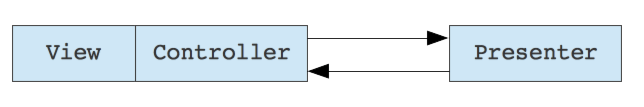
为此，在Cocoa和CocoaTouch框架中，可以方便地使用NSNotificationCenter和KVO，而不必了解其他组件。 视图也类似于来自MVC的视图（可以是组件的层次结构）。 为了减少类的互连性，View无法直接访问Model。



优于传统MVC的优势

在Apple的MVC模式下，视图和模型之间不再存在直接的连接，视图的状态和数据表示的处理逻辑也在控制器中，在当前情况下，这种职责分工更为合适。

这种模式的缺点是Controller包含View状态的一部分和几乎所有View逻辑，而且由于Controller还充当View和Model之间的中介者，因此它成为应用程序逻辑适应的一个非常着重的地方。实际上，UIViweController类变得过于庞大。通常，由于Controller和视图之间的紧密关系，它们被视为表示层的组成部分。



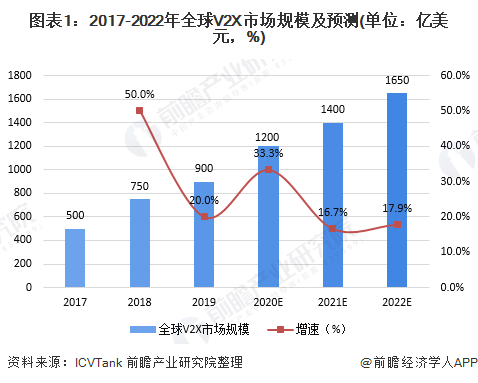
因此，在开发过程中，应尽量避免向Controller层中注入大量的业务逻辑，采用封装的形式将可复用的功能代码打包起来，方便在其他地方使用。采用通知或者观察者模式对Controller层进行逻辑业务分拣，保证业务逻辑流程正常的情况下还有较好的代码可读性。

关于车辆信息的对接，采用TBox系统进行车辆信息的获取。T-BOX（Telematics BOX）即车载远程信息处理器，依托无线语音、数字通信、人造卫星的GPS/北斗系统和CAN总线集成等，实现T-BOX向驾驶员和乘客提供道路交通信息、导航信息、应付紧急情况的对策、远距离车辆诊断、车联网远程控制以及互联网娱乐服务等。

综合技术层的分析，该系统从技术上是独立的一个iOS 系统的app，并且大部分技术都来源于苹果的软件生态。因此技术层面的可行性是完全可以的。

经济的可行性

车联网的发展总体来看大致可以分为三个阶段，第一阶段为具备基本网联能力的车载信息阶段;第二阶段为智能网联汽车阶段，通过V2X技术，车路开始协同;第三阶段为未来的智慧出行阶段，车路协同在智能交通和高级自动驾驶中广泛应用。目前，车联网发展处于第二阶段。5G技术的迅速推广带动物联网应用的落地和普及，车联网作为物联网高速领域行业成熟度最高且连接数量最多的领域，车联网行业快速渗透，行业规模不断扩大。根据ICVTank公布的数据显示，2019年，全球V2X市场规模达900亿美元，预计到2022年，全球V2X市场规模有望突破1650亿美元。



中国的5G技术走在全球前列，在5G时代，中国的车联网发展有望实现弯道超车，同时，自2017年以来，国家出台了一系列政策支持车联网的发展，2019年中国V2X市场规模达200亿美元，预计到2022年，中国V2X市场规模有望达到500亿美元，增速高于全球增速。中国车联网的发展与全球车联网发展基本同步，当前中国车联网处于发展阶段，受汽车传感技术、网线等限制，稍落后于欧美日，但在国家政策的强力支持下，发展形势预计与全球同步。2016年后传统互联网巨头，比如BATH等纷纷介入，美国高通的数据显示，从2018年开始，60%的新车将通过移动技术联网，到2020年中国车联网用户数将超过4000万。

因此，车联网系统在今后的互联网发展中具有很大的经济上升空间，本系统从经济性上也是完全可行的。

社会的可行性

本系统的研发工作，完全是积极响应了国家政府政策，体现了2020年2月国家发改委等11部委发出的《智能汽车创新发展战略》文件中提到的，“推进智能化道路基础设施规划建设”政策方针。对城市智能交通的发展、智慧道路以及新一代国家交通控制网的建立起到了积极的作用。因此，本系统从社会性上来说，是完全可行的。

系统设计原则

本系统的设计原则应从这几方面考虑：用户界面以及交互设计、系统的操作实用、可扩展可维护。

用户界面以及交互设计

对于手机而言，屏幕空间资源显得非常珍贵，为了提升屏幕空间的利用率，界面布局应以内容为核心，而提供符合用户期望的内容是移动应用获得成功的关键。如何设计和组织内容，使用户能快速理解移动应用所提供的内容，使内容真正有意义，这是非常关键的。

关于用户交互，点击操作是PC时代交互的基础，在触摸屏设备上基于手指的手势操作已经代替了鼠标的点击操作。所以，设计时应以简单自然的交互为主，在情境中引导用户学习。

系统的操作实用

在移动产品的操作过程中会碰到多种多样的情况——找不到目标、不知道该怎么操作、操作后没有及时反馈，等等，这都会对产品的流畅性造成影响。在移动产品的设计中主要从三个方面来考虑产品的流畅性：手指及手势的操作流、用户的注意流、转场流畅或者自然，不能牵强。

对于移动产品，提倡的是简单、直接的操作，倾向于清晰地表达产品目标和价值。让用户快速学会使用，尽量不要让他们查看帮助文档。界面架构简单，明了，导航设计清晰易理解，操作简单可见，通过界面元素的表意的和界面提供的想说就能让用户清晰地知道操作方式。只有这样的设计，才能让用户的学习使用没有负担，而不是通过帮助系统来教会用户操作。

可扩展可维护

一个系统不仅仅只在一个时间段内活跃，为了适应将来以及系统的健康发展。在开发过程中，必须具有良好的扩展性和维护性。

车辆管理系统说明：

该车辆管理系统采用较为灵活方便的移动设备app的形式开发，主要面向iOS操作系统的智能终端。APP的开发工具使用的是苹果公司向开发人员提供的集成开发环境Xcode，该环境可用来研发macOS、iOS、WatchOS和tvOS平台上的应用程序。开发语言采用目前流行的Swift开发语言，代码的版本管理采用Xcode自带的git。

该系统的数据存储采用本地应用沙盒的方式存储，应用沙盒是指每个iOS应用都有的、一个独立的文件夹，于其他系统文件夹独立。且每个应用都必须在自己的沙盒文件里，不能访问其他沙盒文件。采用沙盒文件的存储方式可以方便快捷的进行数据交互。

车辆管理系统面向的用户群体是旗下有较多货运车辆的企业，企业的管理人员通过该系统实时查看、派遣所有受到管理的车辆。因此，该系统的用户群体都为企业下的员工，或者与企业合作的第三方公司员工。基于这种限制，登录页面的设计无需考虑注册功能，用户的账号皆由公司企业通过后台管理平台自行创建。为了方便起见，账号采用手机号加手机验证码的方式登录。虽然使用该系统受限于用户所在的企业，会天然的限制用户量的规模，但是可以将该系统进行模块化处理。这样一来，该系统所提供的功能可以根据企业的具体业务来进行调整，从而通过扩展企业的数量实现用户量的增加。

App运行流程以及功能介绍

由于使用限制，app在启动后必须要求用户进行登录操作。登录页面为两个内容输入框加两个按钮，输入框可以输入手机号和手机短信验证码，按钮用来实现获取短信验证码与登录的功能。用户登录之后，会在手机内存中保存一个随机字符串作为用户的token，一方面可以用来判断用户的权限。另一方面，当用户再次使用app的时候就不用再获取手机验证码登录了，省掉登录的操作即可以提高用户使用体验，也可以满足系统的业务要求，这种方式目前较为流行。登录页面如图：

用户登录后APP以地图为主要展示内容。地图中展示用户当前所处位置，该功能通过获取手机的物理经纬度，并在地图上将经纬度绘制成地图的标记点。

地图上除了显示用户的所在位置之外，还会展示当前账号权限下能看到的车辆定位。同理，车辆经纬度坐标由车辆中的车机电脑上传到服务器，然后由应用程序转换成的地图标记点，并展示在地图上。首页效果如图：

考虑到手机的性能以及用户体验，车辆数据并不是一次性全部获取到，而是通过获取地图的中心点坐标加上一定的半径距离，筛选后得到的数据进行展示。尽管如此，还是有可能会导致地图要渲染大量标记点而使得手机硬件消耗过大，会出现发热的现象，因此引入了四叉树的概念。

为了保证规律性和平均分配，采用区域划分方法显示标记点。简单来说就是把屏幕分割成若干个区域，每个区域最多显示一个标记点，然后根据地图缩放比例动态的设置这些区域的大小以达到最佳的用户体验。如果有100条数据，我们可以嵌套循环找到合适的点放入相应的区域，循环10000次，如果有10000条数据，那很有可能会因为app消耗了大量内容而被系统认为是垃圾软件而被系统终止运行。这种计算方法是低效的，时间复杂度至少为O(n^2)。因此，引入四叉树的概念是必要的，它很适合当前的需求场景。

用户拖动地图时，此时地图的中心点发生改变，app需要重新获取地图的中心点再去获取车辆信息，通过上述的展示逻辑再次刷新地图的标记点。而当用户点击地图上的任一点时，app会判断这个标注点是否表示是被聚合起来的标记，如果是的话则将地图进一步放大，展示所有被聚合起来的车辆标记点。如果不是，则在手机屏幕的下部展示一个选中的车辆的信息概览内容。即该车辆最近一次的行驶信息，如车牌号、驾驶员姓名等。

在app首页中的顶部有搜索框和前往个人中心页面的按钮，点击搜索框时会进入到一个新页面。该页面会展示搜索到的结果，用户可以查看或点击搜索结果，点击后进入到详情页展示所要查看的信息，如驾驶员信息、车辆行驶信息、车辆状态报告等。

点击用户信息按钮会跳转到个人中心页面，在该页面中，用户可以修改一些app的设置，如app推送功能、清理app占用的缓存空间、退出登录等；

在app首页的右上侧，垂直排列两个按钮，表示筛选和刷新。点击筛选按钮后，会通过侧边栏的方式展示一个筛选页面，筛选条件为想要查看的某个车队，或者车辆的某种状态。用户选择筛选条件后回到首页，将筛选条件当作入参通过请求接口获得对应的数据，并刷新、展示地图上的标记点。具体刷新条件可以根据企业的业务规则进行后台适配。

点击刷新按钮表示重新刷新、展示地图上的标记点。

在app首页的底部并排水平排列两个按钮，消息通知以及查看报表。点击消息通知按钮跳转到风险列表页面。点击查看报表按钮跳转到企业报表页面。

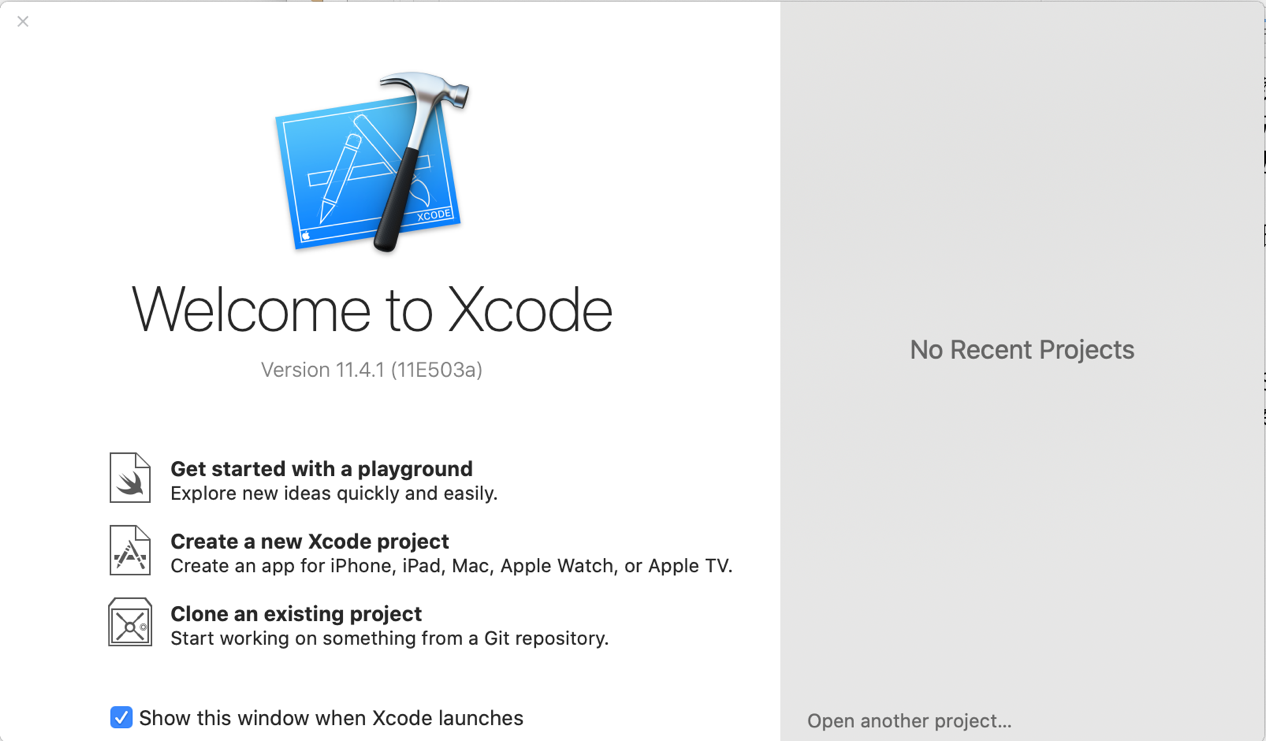
在风险列表页中，用户可以查看不同风险类型下的风险事件，比如说车辆超速、车辆交通违章、维修通知、保险到期通知等。根据这些功能提供的数据，管理者可以从多个方面、多个角度管控车队，并对自己所管理的车队的安全驾驶情况有一个详细的了解。通过这些数据的统计，管理人员可以对驾驶员的一些危险驾驶行为进行约束，从而提高安驾驶员的全行驶意识，保障企业与人员的财产和人身安全。

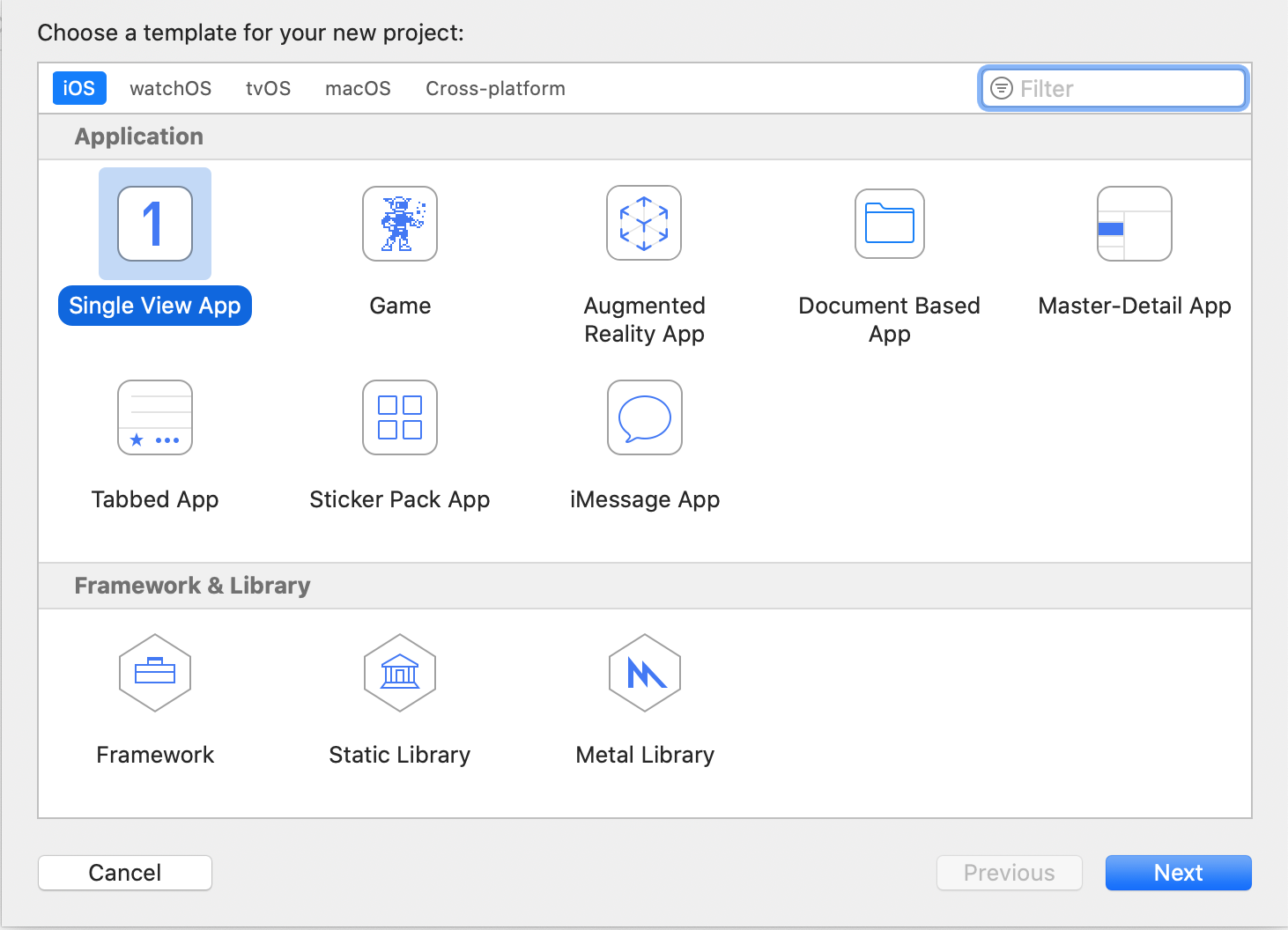
点击查看报表按钮跳转到当前账户所管理的车队总体运营报表页面。该页面可查看各个时间段下车辆的行驶里程，车辆的各种状态，如：行驶时长、出勤率、涉及到的风险事件等。

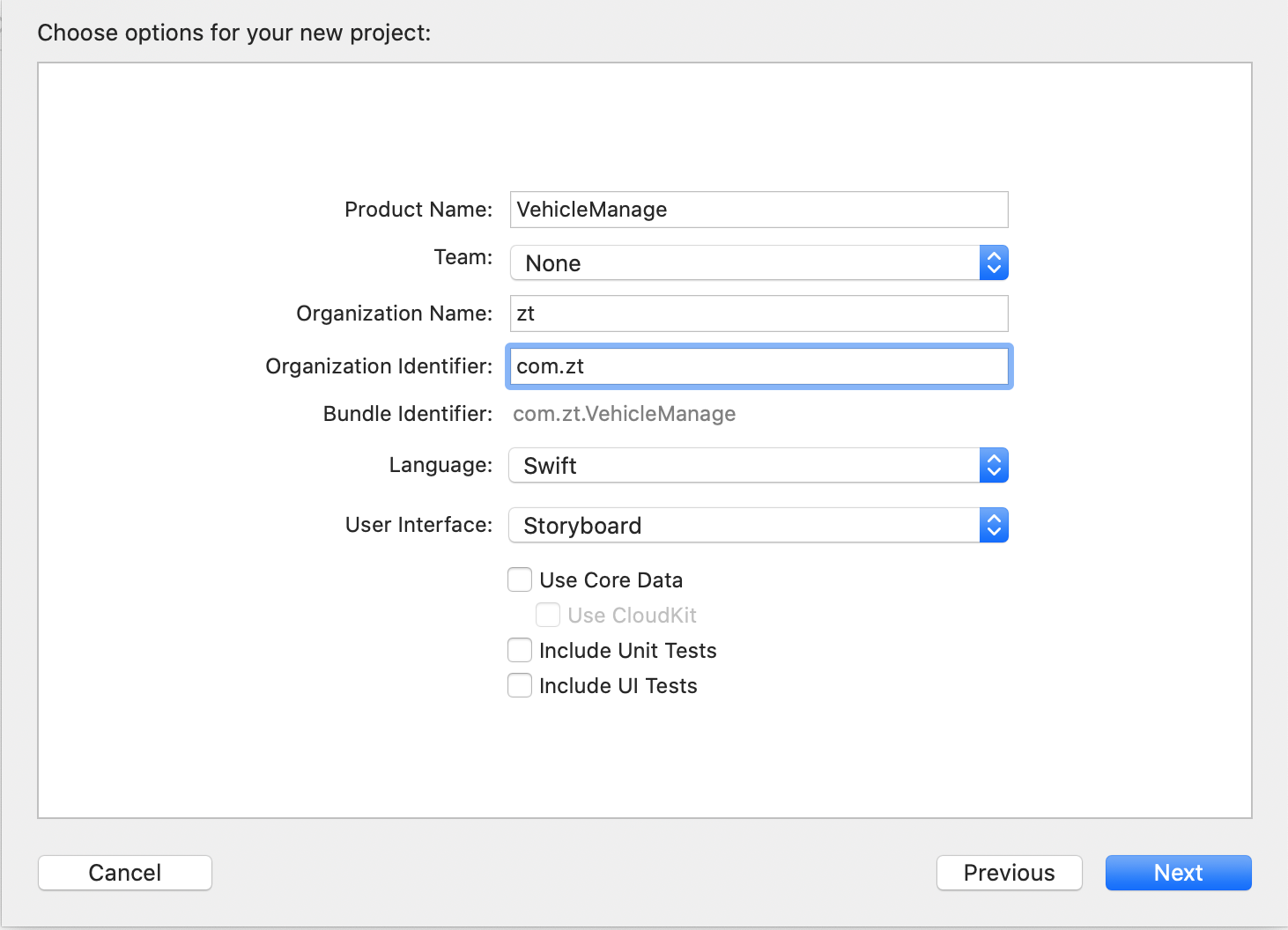
系统实现

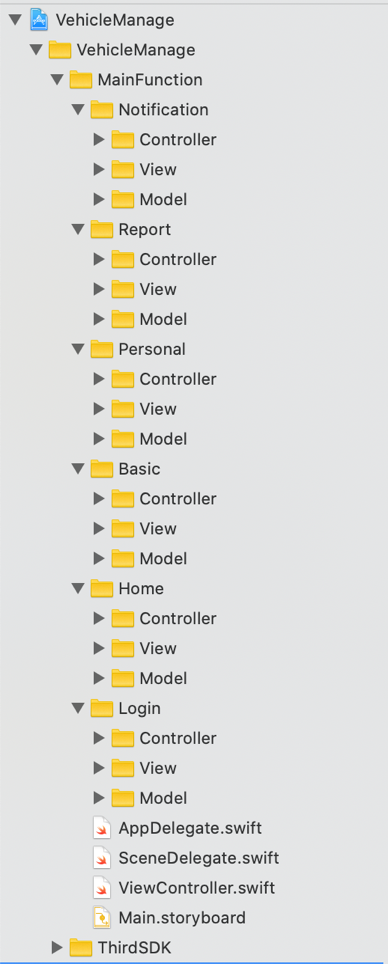
开发环境搭建及配置

首先准备一台Mac操作系统的电脑，通过Mac中的 App Store下载Xcode开发工具。下载成功后打开Xcode，并选择 Create a new Xcode project 来创建一个的工程。如图。在接下来的页面中，选择Single View App表示创建一个带有View视图的工程，点击Next跳转到工程名称以及包名填写页面。在Product Name 处填写“VehicleManage”，Organization Name 处表示开发者所在组织、团体的名称，个人开发者的话可根据喜好填写，此处填写姓名缩写。在Organization Identifier 处填写包名的前缀，此处填写“com.zt”，填写好后工程会根据报名前缀+工程名称的方式自动生成最终包名。此包名会在最后项目开发完成后，提交到App Store时需要作为app的唯一标识符来使用。









工程创建完成后开始app的结构搭建，首先要添加app在开发过程中用到的一些第三方SDK（Software Development Kit，软件开发工具包）。目前主流的方式是通过CocoaPods工具来统一对第三方SDK进行管理，使用终端进入到工程所在的路径下，输入该命令来创建一个管理配置文件“vim Podfile”。在文件中添加自己用到的SDK，添加完成后执行“pod install”命令来安装这些SDK文件。在后续开发过程中，如果需要添加或删除某个SDK文件，可以通过修改Podfile文件中的内容来实现。

项目工程环境搭建

SDK文件配置好后，再配置工程的文件结构，由于该项目是基于MVC开发模式进行的，而且为了项目文件的可读性，因此整体结构划分为：

1. 功能模块、第三方头文件的引用以及资源文件作为1级目录。其中第三方文件目录下的二级目录为SDK的名称，二级目录下则是SDK的文件，且不设置第三级目录资源文件不设置二级、三级目录。
2. 功能模块下，每个功能模块都设为二级目录。
3. 每个二级目录下，都只设置三个三级目录，对应MVC三个模块。并且以“controller”、“view”、“model”三个名称作为三级目录的命名。
4. Controller路径下存放当前功能模块中所有的类文件，也是主要功能的路径。
5. View路径下存放当前功能模块中，展示在手机屏幕上的UI元素。
6. Model路径下存放当前功能模块中的数据模型。

最终文件目录结构效果如图。从图中可以看到本系统的主要实现逻辑是在“MainFunction”目录下，这样的目录结构使得项目有较高的扩展性和后期维护性。接下来，通过表格的形式罗列出各个功能模块下的是类以及类的作用。

首先是消息通知模块：

|  |  |  |
| --- | --- | --- |
| 二级目录名称 | 类名 | 作用 |
| Notification | NotificationViewController | 用来切换展示的内容 |
| RiskListViewController | 展示各种风险事件 |
| TrafficViolationViewController | 车辆交通违章相关内容 |
| AccidentViewController | 车辆事故通知内容 |
| MaintenanceViewController | 车辆维修内容 |
| InsuranceViewController | 车辆保险相关内容 |
| RiskDetailsViewController | 风险事件详情内容 |
| TrafficViolationDetailsViewController | 车辆违章详情内容 |
| AccidentDetailsViewController | 车辆事故详情内容 |
| MaintenanceDetailsViewController | 维修详情内容 |
| InsuranceDetailsViewController | 保险详情内容 |
| TimeSelectorViewController | 选择时间控件 |

然后是报表模块：

|  |  |  |
| --- | --- | --- |
| 二级目录名称 | 类名 | 作用 |
| Report | EnterpriseReportVC | 报表查看 |
| VehicleManagerVC | 车辆信息查看管理 |
| DriverListViewController | 驾驶员列表 |
| DriverDetailsViewController | 驾驶员安全运营报告 |
| EnterReportFilterVC | 条件筛选 |

接着是用户个人中心模块：

|  |  |  |
| --- | --- | --- |
| 二级目录名称 | 类名 | 作用 |
| Personal | MyController | 展示用户信息 |
| NotificationSettingVC | 设置推送消息 |
| AboutUs | 关于我们介绍 |

基础类模块：

|  |  |  |
| --- | --- | --- |
| 二级目录名称 | 类名 | 作用 |
| Basic | WRNavigationBar | 导航栏基础控件 |
| WRCustomNavigationBar | 导航基础控件 |
| RootController | 根控制器 |
| BaseVC | 根视图 |

主页模块：

|  |  |  |
| --- | --- | --- |
| 二级目录名称 | 类名 | 作用 |
| Home | HomeVC | 展示地图主页面 |
| FilterVC | 筛选条件页面 |
| MarkToastVC | 展示车辆信息面板 |

登录模块：

|  |  |  |
| --- | --- | --- |
| 二级目录名称 | 类名 | 作用 |
| Login | LoginVC | 登录页面 |

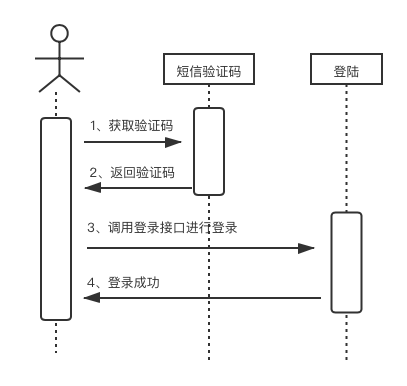
详细设计

用户登陆模块

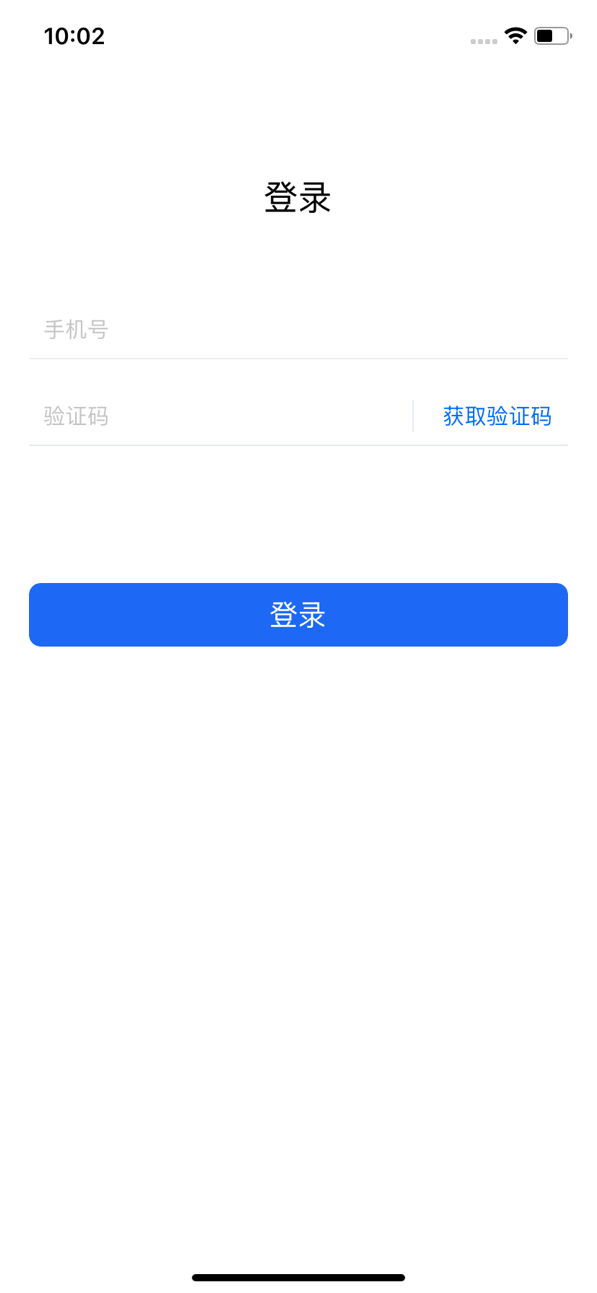
根据需求，app启动后由登录页面作为主页面，且用户必须登录之后再进入到功能页，因此在登陆页面中，不应设置返回按钮。并且在app启动时要对本次启动进行判断，判断用户是否已经有过登录行为，如果有的话则无需进入到登陆页面，直接展示主功能页面即可。用户登录的流程为：

1. 输入手机号码；
2. 点击获取短信验证码按钮。此时程序中需要做手机格式校验，目的是为了防止输入无意义的手机号进行无效的网络请求。校验的标准为：输入长度为11位，并且为纯数字的内容表示手机号码格式正确。否则app端应弹出提示框，提示用户手机号格式有误，请重新输入；
3. 输入短信验证码；
4. 点击登陆按钮调用登陆接口，将用户手机号与验证码作为请求参数发送请求。此时app端也要对手机号码的格式进行校验。短信验证码的格式可以不校验，因为app端无法知道当前系统对接的短信服务商是哪一家，也就无法判断什么是正确的短信格式。
5. 根据登陆接口返回的数据，判断用户是否登陆成功，如果是的话，将接口返回的用户信息进行保存，并将主页面展示出来。如果不是，根据接口返回的错误信息，提示用户进行对应的操作。

登陆页面时序图如图：



登陆页面效果如图：



登录逻辑判断核心代码：

func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

// Override point for customization after application launch.

IQKeyboardManager.shared.enable = true

IQKeyboardManager.shared.shouldResignOnTouchOutside = true

AMapServices.shared().apiKey="97750fcfe046cecd6adf37c915c20189"

AMapServices.shared().enableHTTPS = true

let entity = JPUSHRegisterEntity();

entity.types = Int(JPAuthorizationOptions.alert.rawValue) | Int(JPAuthorizationOptions.sound.rawValue) | Int(JPAuthorizationOptions.badge.rawValue);

JPUSHService.register(forRemoteNotificationConfig: entity, delegate: self as JPUSHRegisterDelegate);

JPUSHService.setup(withOption: launchOptions, appKey: "107c971e50962b7179d8cddb", channel:"" , apsForProduction: true);

if isLogin() {

let navi = BaseNavigationController.init(rootViewController: HomeVC())

self.window?.rootViewController = navi

}else {

let navi = BaseNavigationController.init(rootViewController: LoginVC())

self.window?.rootViewController = navi

}

self.requestVersion()

self.window?.makeKeyAndVisible()

return true

}

登录逻辑核心代码：

func loginHandle() {

self.phoneStr = self.phoneTf.text ?? ""

if self.phoneStr.count == 0 {

self.view.makeToastMid(message: "请填写手机号")

return

}else if !isMobileNumber(phone: self.phoneStr) {

self.view.makeToastMid(message: "手机号格式不正确")

return

}

if self.codeTf.text?.count != 4 {

self.view.makeToastMid(message: "验证码长度不对")

return

}

self.requestLogin()

}

func requestGetCode() {

let param = ["phone":self.phoneStr]

HttpRequest.loadData(target: InterfaceAPI.getVerCode(param: param), needCache: false, cache: nil, success: { (datas) in

self.getCodeBtn.isCounting=true

}) { (stateCode, message) in

self.view.makeToastMid(message: message)

}

}

func requestLogin() {

let param = ["phone":self.phoneStr,"verCode":self.codeTf.text!]

NSLog("%@", param)

HttpRequest.loadData(target: InterfaceAPI.login(param: param), needCache: false, cache: nil, success: { (datas) in

let json = JSON(datas)

let token = json["data"].description

setToken(token:token)

setIsLogin(isLogin: true)

setPhone(phone: self.phoneStr)

JPUSHService.setAlias(self.phoneStr, completion: { (iResCode, iAlias, seq) in

print(iResCode)

print(iAlias!)

}, seq: 1)

let app = UIApplication.shared.delegate as!AppDelegate

let navi = BaseNavigationController.init(rootViewController: HomeVC())

app.window?.rootViewController = navi

}) { (stateCode, message) in

self.view.makeToastMid(message: message)

}

}

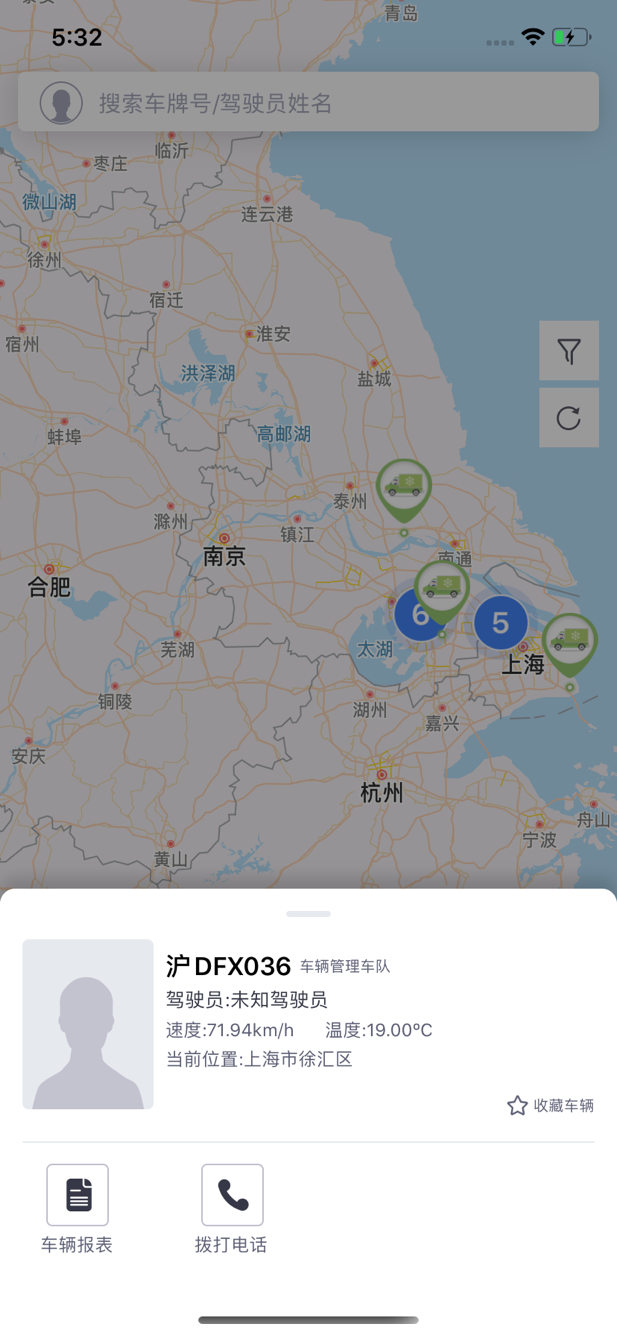
用户登陆成功后，进入到主页面，主页面展示的主要内容为地图界面。以用户当前所在的位置为圆心，半径为5公里，将这两个数据作为主要参数调用接口查询附近的车辆。将接口返回的数据信息进行处理，最终以地图标记点的形式展示到地图上，如图：



主页面模块：

主页面提供的功能有：快速搜索、条件过滤刷新、个人中心入口、通知功能入口、报表功能入口、车辆信息面板展示；在主页面中，地图上展示的点为该坐标点处的车辆数量，当数量大于1时，根据四叉树的逻辑将标记点合并为一个展示到地图上，标记点的数字为所包含的车辆数量。

点击标记点时，需要做一次判断，逻辑为：点用户点击的点为聚合点时，需要将地图以点击的位置为中心点进行放大操作，放大后的地图应将聚合的点拆分开，并展示出来，此时的标记点为带一个车辆图片的点。用户点击的是车辆标记点时，在屏幕下发的区域展示一个该车辆的信息面板，如图



车辆详情面板中展示该车辆最近一次行驶的记录信息和车辆报表，还有驾驶员相关信息。并且点击拨打电话按钮可以直接和驾驶员进行沟通。在车辆信息面板页面的上方提供一块灰色透明遮罩，用户点击遮罩会把信息面板移除。

首页中的筛选按钮提供了一些快捷条件筛查的功能，用户可以根据不同的公司、车队来查找符合的数据。点击按钮后，从手机屏幕右侧弹出一个新页面，新页面分为两个区域，左侧是灰色半透明的可点击区域，右侧为筛选条件列表。点击灰色区域隐藏该页面并退回展示地图主页面。其中，筛选条件为单选，选中某个条件后隐藏条件筛选页面，回到地图主页面并根据选中的筛选条件重新发起请求获取数据。如果没有查询到对应的数据则弹一个提示框提示用户没有数据，如果有的话则展示到地图上。筛选页面效果如图：



点击搜索框时拦截键盘弹出的事件，改为跳转到搜索页面。在搜索页面中，主要展示车辆信息列表，在手机屏幕上方有一个输入框，对输入框中的文本进行搜索，搜索得到的结果在列表中展示。列表内容包括最近驾驶过该车辆的三名驾驶员、车辆行驶里程、最近行驶日期以及行驶状态。页面中带有排序功能按钮，点击之后会对搜索结果进行驾驶历程的排序。

排序功能核心代码：

func createFilterView() {

self.rightFlag = !self.rightFlag

if self.rightFlag == true {

self.view.addSubview(transparentView)

filterView = VehicleChooseView.init(frame: CGRect(x: 0, y: 0, width: KW, height: KH - statusHeight - 90),orgArr:self.orgArr,vehicleGroupArr:self.vehicleGroupArr)

transparentView.addSubview(filterView)

filterView.passValue = { (index,id,isFocus) in

self.handleFilterValue(index: index, id: id)

}

}else {

self.rightFlag = false

self.filterView.removeFromSuperview()

self.transparentView.removeFromSuperview()

}

filterView.closeClick = {

self.rightFlag = false

self.filterView.removeFromSuperview()

self.transparentView.removeFromSuperview()

}

}

func handleFilterValue(index:Int,id:Int64) {

if index == 1 {

orgId = id

vehGroupId = nil

focusVeh = nil

}else if index == 2 {

orgId = nil

vehGroupId = id

focusVeh = nil

}else if index == 3 {

orgId = nil

vehGroupId = nil

focusVeh = true

}else if index == 0 {

orgId = nil

vehGroupId = nil

focusVeh = nil

}

let defaults = UserDefaults.standard

defaults.set(orgId, forKey: "orgId")

defaults.set(vehGroupId, forKey: "groupId")

defaults.set(focusVeh, forKey: "focusVeh")

self.filterView.removeFromSuperview()

self.transparentView.removeFromSuperview()

self.rightFlag = false

self.requestVehicle()

}

搜索页面如图



点击任意一行列表，跳转到对应车辆的运营报告页面中，该页面展示的内容为车辆的运营数据，具体有三个模块：行驶里程运营情况交通状况。三个模块的数据都是根据时间区间来展示，比如说展示当天或者7天内以及30天内的行驶数据。

在行驶里程中，展示了车辆累计行驶公里数、日间行驶公里数、夜间行驶公里数以及百公里高/中风险数量。这些数据可以体现出车辆的整体运营状态，当车辆在夜间行驶公里数较多时，管理人员应多注意车辆和人员的行驶安全，尽量调节车辆和人员的行驶时间。

在运营情况中，展示了车辆的运营天数、运营率、日均行驶公里和行驶时长。这些数据可以让管理者实时掌握车辆的调用使用情况，避免车辆闲置过久造成的资源的浪费。日均行驶数据可以体现出车辆的损耗情况，以便管理者及时的对车辆进行保养维护，防止车辆损耗过快。

在交通状况中，展示了车辆的违章次数和交通事故次数。该数据可以追踪到事故出现时的具体驾驶情况，对事故责任的划分起到了一定的作用。也能对事故种类进行分析，在以后的运营中提前发出风险预警，防止同类事故再次发生。

该报表页面中，还提供了车辆定位和历史轨迹回放的功能。车辆定位会将车辆的位置进行上报，并在地图中展示出来，当车辆发生丢失、抛锚或失联等危险情况时，可以提供精确的位置帮助，这对车辆的救援起到了重要的作用。历史轨迹回放功能是在地图上画出车辆的行驶路线，并播放车辆从起点到终点的行驶轨迹，同时记录了车辆在行驶过程中的车厢温度和时速表的变化情况。这对车辆的维护保养以及道路信息的规划提供了很大的帮助。







通知页面

通知页面由5个子标签页组成，分别是：风险事件、交通违章、事故通知、维保通知、保险即将到期。这5种类型都与车辆有关，当车辆在运营过程中触发了某项的风险项，就会在对应的通知标签页中展示出来。通知页面的数据可以对各项车辆涉及到的驾驶风险进行一个归类总结，管理者可以看到哪些风险行为是经常发生的，从而对驾驶员进行约束。这样既能有效的帮助驾驶员提高安全驾驶意识，降低交通违章、事故的发生概率，也能实时对驾驶员的人身安全有一个详细的掌握，防止事故发生时对车内人员的受伤情况不了解。

（7、8）设计心得、致谢、参考文献（750字）：