摘要

随着国家在战略和应用层面广泛的对农业产业的扶持，扶贫工作的大力推进，对于全国各地广大农村的扶贫工作来说，至关重要的是农村现代化信息化建设。对于广大农村地区的发展和建设工作而言，建立发达且现代化的农业生产体系，扶持广大农民群众获得先进的农业生产生活知识，对解决三农问题至关重要。而随着互联网技术的不断发展以及信息技术的不断进步，农村人口也成为了中国网民的重要组成部分，这部分群体天然的适合接受农业信息化带来的便利，于是一个方便快捷的农业科技综合服务应用对广大农村群众而言便会是一个易于接受的解决方案，为三农问题的相关建设作出贡献。

本系统以Nodejs为基础，结合Eggjs企业级应用解决方案，使用JavaScript作为系统前后端的主要实现语言，前端使用Vue-cli集成化项目骨架，采用MVVM架构实现高性能的前端服务，后端使用基于Koa的Nodejs框架Egg，数据库相关操作使用MySql的ORM应用Sequelize，开发一个具有展示多种基本农业应用中可用信息以及WebGis功能的农技百科系统。系统涵盖了常用的农业相关信息查询如良种肥料病害的查询，月度农事信息以及农事相关天气灾害信息的发布，食品安全相关法律法规的普及信息以及一个测土配方的用肥指南功能。

经过实际的测试，本系统相对完善的实现了所有设计的功能。Nodejs作为近年来非常流行的新兴技术框架，其并发高响应快且轻量的特点使它在许多Web应用情景中具有显著的优势。本文所作的研究对今后使用Nodejs技术在相关应用系统的开发方面做了积极地探索，对相关的工作有一定的参考作用。

关键词：三农，农技百科，JavaScript，Nodejs，Vue， WebGIS

# 第一章 绪论

## 1.1研究的背景及意义

我国农村广而分散，尤其是中西部广大农村，距城市较远、交通不便、信息闭塞，满足农业人口的生产和生活需求，一直是。互联网的基本功能之一就是可将农民分散的个性化供给和需求积聚起来，通过各种网络平台形成规模效应，从而为农民服务。尤其是随着移动互联网的发展，千家万户的小农户与千变万化的大市场距离越来越近，分散的农户在销售自家生产的农产品、购买农业生产投入品或社会化服务时， 通过互联网或移动终端就可以实现。而且减少了中间环节，降低了经营成本，增加了收入,增加了农产品定价的话语权。

近十年以来，云计算、物联网、专家系统的飞速发展，改变了人们的生产生活方式，信息技术与多学科的发展，将不同产业与信息技术有效结合，促进了各大产业的发展。但是，农业产业仍未与信息产业全方面的结合，发展方式过于单一。可见，我国在农业信息化方面还存在巨大的发展潜力，相关的工作还有待进一步的推进。

## 1.2研究的主要内容

本文实现的农技百科系统，其主要的用途在于把专业化的农业相关知识通过一个web信息系统作为信息发布和更新的平台，使互联网成为农民朋友的好帮手，且其易于操作易于理解的特点使得它容易被掌握和使用。同时本系统在开发环境上使用的是Nodejs和Vue等较为新潮的技术架构，可以为之后相关的研发工作提供一点参考。

## 1.3本文的组织结构

Nodejs作为目前比较火热的服务端环境，比较适合快速响应高性能的微服务开发，也是目前互联网项目的前沿发展方向。本论文共分为六章：

第一章，对论文的研究背景和研究意义作概括性阐述。

第二章，对农技百科系统用到的Nodejs及相关的应用框架进行了详细的介绍。

第三章，对项目中相对独立同时又重要且值得关注的技术进行介绍，主要包括响应式web设计，数据库ORM工具等。

第四章，对农技百科系统的系统设计，主要包括系统的架构，各个模块的设计以及数据库的设计。

第五章，农技百科系统功能的实现，在之前的基础上，完成了农技百科系统的开发与实现，包括一些基本功能。

结论与展望，总结了本文的主要工作所取得的成果，并展望了今后仍需进一步研究的问题。

# 第二章 Nodejs与MVVM相关理论技术

## 2.1在服务端运行的JavaScript-Nodejs

### 2.1.1 Nodejs简介

Web应用程序架构基于Node.js的Web应用程序主要遵循Java Web应用程序架构。主要区别在于客户端请求处理。使用Java，每个客户端请求将由单独的线程同步处理，因此它是多线程的，但在Node.js应用程序的情况下客户端请求将只由单个线程，由异步方式来处理。

总的来说，Node.js是一个事件驱动I/O服务端JavaScript环境，基于Google的V8引擎，V8引擎执行Javascript的速度非常快，性能非常好。

### 2.1.2 Nodejs相比传统服务端开发环境的特点与优势

Nodejs最主要特点与优势有二，为高并发和适合I/O型密集调用。

1.高并发

以主流的Javaweb或类似的多线程模型为例，这种多线程以调节服务端线程的数量来提高对并发请求的接收和响应，但并发量高的时候，请求仍然需要等待，它有个更明显的问题，我们以客户端请求与服务端通讯的过程为例：

与服务端通信的过程：



带来的问题：



服务端与客户端每建立一个连接，都要为这个连接分配一套配套的资源，主要体现为系统内存资源，以PHP为例，维护一个连接可能需要20M的内存，这就是为什么一般并发量变大，就需要多开服务器。

而Nodejs则是单线程，异步，事件驱动模型。以一个简单的类比来描述这种机制：

假设服务器是食堂，请求是顾客，一个顾客对食堂需求的完整服务为：排队，点餐，取餐。并发相当于大量的顾客涌入食堂，Java多线程相当于开多个服务窗口，顾客需要在窗口点餐并在窗口，等待，饭上齐了然后取餐，所以用多开窗口（多线程）来解决客流大的问题。而Nodejs则是单窗口叫号的模式，每个顾客点餐后取号并离开窗口，等待叫号取餐。这个叫号的行为就是异步编程的Callback回调，能等到事件（I/O）完成后继续执行之后的逻辑（吃饭）。整个过程没有阻塞新用户的连接（点餐），也不需要维护已经点餐的用户与窗口的连接。

基于这样的机制，理论上陆续有用户请求连接，NodeJS都可以进行响应，因此NodeJS能支持比Java、PHP更高的并发量。

2. 非阻塞I/O

总的来说这个优点仍然归功于异步编程的特点，换句话说它仍是JS语言本身的特点带来的优势。

假如有两个I/O请求a和b，从发出请求到得到数据各需要1秒，需要a和b都执行完毕才能继续之后的逻辑。

如果使用Java或PHP，用普通的串行写法，则总用时2秒。但用Nodejs的普通写法，总用时仍是1秒左右。因为Nodejs本身的特点，一个I/O请求相当于一个异步函数，会直接创建一个队列去执行它，因而a和b本身就是并发执行，所以总时间仍是1秒左右。

虽然Java和PHP也能够用子线程并行请求，但需要在代码上作额外的处理，而Nodejs不用任何特殊的处理就能用回调函数这种JS语言机制自然地完成并发。

Nodejs也有一些缺点：

1. 不适合CPU密集型应用；CPU密集型应用给Node带来的挑战主要是：由于JS单线程的原因，如果有长时间运行的计算，将会导致CPU时间片不能释放，使得后续I/O无法发起；解决方案：分解大型运算任务为多个小任务，使得运算能够适时释放，不阻塞I/O调用的发起；  
 2. 只支持单核CPU，不能充分利用CPU  
 3. 可靠性，一旦代码某个环节崩溃，整个系统都崩溃。解决方案：（1）使用pm2，nodemon等常用插件，负载均衡，绑定多个端口；（2）使用多个进程监听同一个端口，使用cluster模块。  
 4. 开源组件库质量参差不齐，更新快，常出现兼容性问题。解决方案：随着行业寡头们共同的开源工作，这个问题已经基本解决。  
 5. Debug不方便，错误没有stack trace。解决方案：随着TypeScript的普及，这个问题也已解决。

总的来说，任何语言都能实现任何应用，我们只需考虑在具体条件下它适合与否。

### 2.1.2 Koa及Eggjs简介

原生的Nodejs并没有自带的工程化架构，或者说任何企业或个人想用Nodejs搭建工程，自然不能只使用原生的接口函数，而是想办法封装再封装。与其他服务端环境的发展史类似，Nodejs也发展出了一些常用的框架。在很长一段时间Nodejs最常用的框架是Express，它在使用上也是最接近原生Nodejs的框架，就像JQuery之于原生DOM一样。

Nodejs的发展史自然也是JavaScript本身的发展史，JS的最大特点是回调函数（Callback），它最初在代码上有个致命问题，在于回调函数的嵌套。如果想控制回调函数a和b的执行顺序，即等a执行完后再执行b，只能把b再作为a的回调函数。如此在代码上很容易出现大量的多层次嵌套，使代码杂乱无章，这个问题被称为回调地狱。这个问题由微软的开源工作得到初步解决，即Promise函数，能以链式调用的形式安排回调函数的顺序，也是目前最广泛的解决方式。但Promise本身也有可能自我嵌套，直到2016年发布的新JS语言规范ES7的问世，加入了Async/await函数，这个问题才得到更完善的解决。而基于这个函数的特性，Koa诞生了。

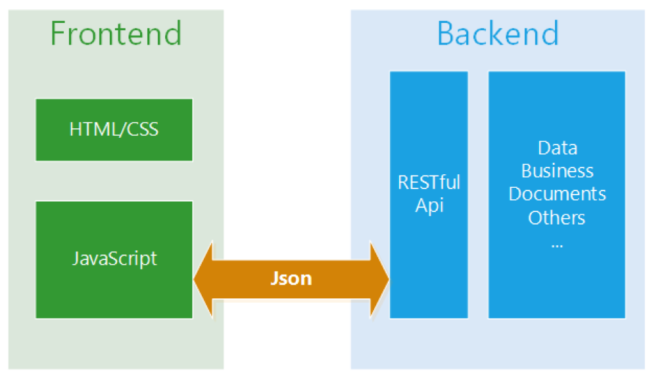
Koa是基于Node.js的下一代web框架，由Express团队打造，特点：优雅、简洁、灵活、体积小。几乎所有功能都需要通过中间件实现。它是最新一代的Nodejs应用框架，特点是洋葱型的处理过程，所有的请求经过一个中间件的时候都会执行两次，对比 Express 形式的中间件，Koa 的模型可以非常方便的实现后置处理逻辑。这个特点其实就是Async/await函数本身的特点，它使得连续的，看上去串行的函数有一个回溯执行的过程，可以在代码上不使用任何回调函数，因为Async/await本身就包含了回调的功能。

Egg.js是阿里公司的开源应用框架，它继承于Koa。如上述，Koa 是一个非常优秀的框架，然而对于企业级应用来说，它还比较基础。而 Egg 选择了 Koa 作为其基础框架，在它的模型基础上，进一步对它进行了一些增强，使它成为一个企业级的应用框架。

## 2.2 MVVM简介

### 2.2.1 前后端完全分离

传统的web开发在分工上原本并没有前后端之分。前后端分离的趋势之所以没有在更早的时候充分发展，是因为html的相关技术发展的较晚，而html相关技术的发展其实是由广泛的互联网网速提升引起的。十年前的网站页面和今天看上去差别甚大，应用的复杂程度更是不可同日而语,如今web应用即可替换掉传统的原生应用。由此，前后端的解耦，大型项目分工的细化是很自然的发展趋势。在此基础上，前后端所有的数据交互都是通过AJAX进行的，如下图：



但前后端分离并不只是为了考虑开发的难度或分工的细化，任何一项新技术得以流行都不会是因为它使用便捷，而是它确实有重要的性能优势。传统的服务端渲染如JSP技术，网页本身需要服务端传输，而前后端分离可以让前端静态化，整个请求服务的过程流量大幅减少，同时网页的表现性能也大幅提高，且前端因为静态化，一些页面攻击变得绝无可能，少量注入式攻击可以被很好的规避。

### 2.2.2 从MVC到MVVM

既然web相关开发技术是从前后端不分离发展到前后端分离的，在早期的发展历程里，前端本身也自然使用和后端相同的MVC结构。但从直观上看，如果使用前后端分离的结构，前端并没有采用MVC的必要，或者说在早期采用这种结构仅仅出于习惯使然。把代码逻辑嵌合到HTML中是JSP的遗留，在前后端不分离的前提下这么做的确是方便快捷的。但如果前后端分离了，前端工程使用MVC便会使代码显得复杂，往往一个单独组件就由不同代码目录里的四到五个文件组成，这显得只是在机械地模仿后端MVC结构。

这种 MVC 架构模式对于简单应用是可行的，也符合软件架构的分层思想。 但实际上，随着H5 的不断发展，人们更希望使用H5 开发的应用能和Native 媲美，或者接近于原生App 的体验效果，于是前端应用的复杂程度已不同往日，今非昔比。而MVVM能很好的满足新的应用需求。

MVVM最早由微软提出，它借鉴了桌面应用程序的MVC思想，在前端页面中，把Model用纯JavaScript对象表示，View负责显示，两者做到了最大限度的分离。把Model和View关联起来的就是ViewModel。ViewModel负责把Model的数据同步到View显示出来，还负责把View的修改同步回Model。

在一个MVC结构中，即便抛开视图后，模型和控制器处理的大部分业务逻辑，都是为视图服务的。所以，当我们抽象一些视图的基础模型，并通过VM将视图本身的（不需要与外界交互的）状态变迁逻辑封装在一个MVVM组的内部，对外只暴露必要的数据更新和消息回调接口。繁琐的视图逻辑就可以被限制在一个MVVM当中，这就是MVVM的基本结构。

## 2.3 新一代前端框架Vue简介

### 2.3.1 Vue介绍

Vue.js 是一套渐进式前端框架，基于MVVM模式。与其他重量级框架不同的是，Vue 采用自底向上增量开发的设计。Vue 的核心库只关注视图层，并且非常容易学习。Vue.js可以作为一个js库来使用，也可以用它全套的工具来构建系统界面，这些可以根据项目的需要灵活选择，所以说，Vue.js是一套构建用户界面的渐进式框架。

Vue.js的优点：（1） 简洁 （2） 轻量 （3）快速 （4） 数据驱动 （5） 模块友好 （6） 组件化

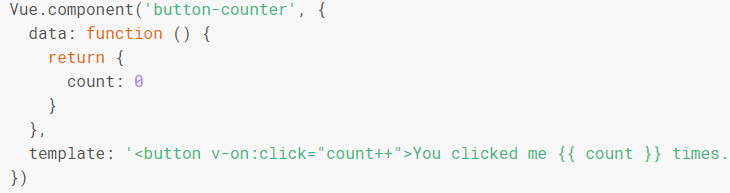
Vue的两大核心：数据绑定和视图组件。

Vue的数据驱动：数据改变驱动了视图的自动更新，传统的做法得手动改变DOM来改变视图，vuejs只需要改变数据，就会自动改变视图，这就是MVVM思想的实现。

视图组件化：把整一个网页的拆分成一个个区块，每个区块我们可以看作成一个组件。网页由多个组件拼接或者嵌套组成。

### 2.3.2 Vue作为MVVM架构的实现

MVVM可以拆分成：View --- ViewModel --- Model三部分，单个的Vue组件即包含这三个部分的实例。一个Vue组件实例如下：



data对应model，template对应View，data的平级项可以增添大量逻辑，对应ViewModel。

Vue.js 可以说是MVVM 架构的最佳实践，专注于 MVVM 中的 ViewModel，做到了数据双向绑定，而数据双向绑定也是Vue最核心的实现细节。下面是 Vue.js 关于双向绑定的实现细节：

Vue.js 是采用 Object.defineProperty 的 getter 和 setter，并结合观察者模式来实现数据绑定的。当把一个普通 Javascript 对象传给 Vue 实例来作为它的 data 选项时，Vue 将遍历它的属性，用 Object.defineProperty 将它们转为 getter/setter。用户看不到 getter/setter，但是在内部它们让 Vue 追踪依赖，在属性被访问和修改时通知变化。

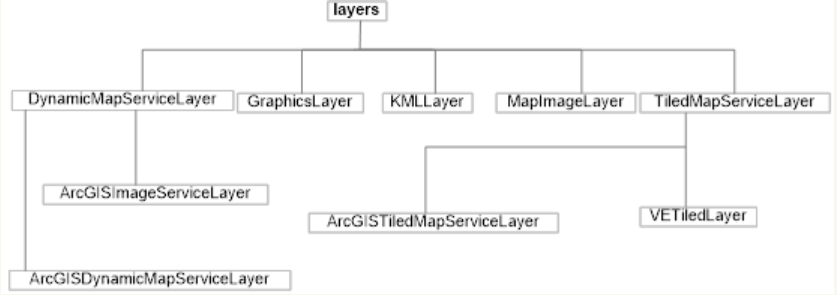
# 第三章 系统相关理论技术

## 3.1 ArcGIS API for JavaScript 简介

随着ArcGIS 9.3的发布，ArcGIS API for JavaScript是ESRI根据JavaScript技术实现的调用ArcGIS Server REST API接口的一组脚本。当前的最新版本是Version 4.9。通过ArcGIS API for JavaScript可以将ArcGIS Server提供的地图资源和其它资源(ArcGIS Online)嵌入到Web应用中。

### 3.1.1 基本概念

实现地图图层功能的主要是Map类、Layer类，Map类是最核心的类，其它控件或多或少的都将Map对象作为其参数，它主要用于呈现地图服务、影像服务等。一个地图对象需要通过一个DIV元素才可以添加到页面中，通常地图控件的宽度和高度是通过DIV容器初始化的。Map对象不仅仅用来承载地图服务和GraphicsLayer同时还可以监听用户在地图上的各种操作事件，并做出响应，Map对象提供了非常丰富的事件，使用这些事件，就可以让地图跟用户随心所欲的去交互。Layer类是图层类，用来添加相关图层，比如WMS服务、热点图（HeatMap）、Bing地图、OpenStreetMap、GeoRSS、KML数据等，其类继承结构如图：



其中，ESRI.ArcGIS.Client.GraphicsLayer是一种客户端图层，用于绘制基于矢量数据的图形，并指定符号进行渲染。DynamicMapServiceLayer为抽象基类，表示动态图层类型。如果想创建自定义的动态图层类型，通过指向在线服务的URL访问动态生成的图片，那么就得实现DynamicMapServiceLayer。ESRI.ArcGIS.Client.Toolkit.DataSources.KmlLayer也是一种GraphicsLayer，其图形几何体由基于XML语法的数据源返回，可以是KML、KMZ或服务。TiledMapServiceLayer为抽象基类，表示切片/缓存地图服务图层。当需要自定义一个切片图层类型，并且缓存图片可以通过简单的URL直接引用时，应实现TiledMapServiceLayer。总之，在非大型项目的实际应用中，照着文档说明行事基本能解决所有问题。

### 3.1.2 使用方法

ArcGIS API for JavaScript支持在线引用和离线引用两种方式。在下载API时需要注册ArcGIS用户。主要分为以下几个步骤：   
1. 配置nginx服务器,本地调试和远程部署都可使用；  
2. 修改API的两个JS文件，为init.js和dojo.js，将指定baseUrl修改为项目计划的地图路径，原理和百度地图api类似；   
3. 将修改好的API函数库复制到指定文件夹，即arcgis\_js\_api放入项目配置目录 ；   
4. 测试功能，新建一个空的html文件，配置相关api，能用即可。

## 3.2 ORM及Sequelize简介

### 3.2.1 ORM概述

ORM即Object-Relationl Mapping，它的作用是在关系型数据库和对象之间作一个映射，这样，我们在具体的操作数据库的时候，就不需要再去和具体的SQL语句打交道，只要像平时操作对象一样操作它就可以了 ，它主要是为了解决数据持久化的问题。

持久（Persistence），即把数据（如内存中的对象）保存到可永久保存的存储设备中（如磁盘）。持久化的主要应用是将内存中的数据存储在关系型的数据库中，当然也可以存储在磁盘文件中、XML数据文件中等等。持久层（Persistence Layer），即专注于实现数据持久化应用领域的某个特定系统的一个逻辑层面，将数据使用者和数据实体相关联。

关系型数据库中的数据基本都是以一行行的数据进行存取的，而程序运行却是一个个对象进行处理，而目前大部分数据库驱动技术（如JDBC）均是以行集的结果集一条条进行处理的。所以为解决这一困难，就出现 ORM 这一个对象和数据之间映射技术。

### 3.2.2 Sequelize简介

Sequelize是一款基于Nodejs功能强大的异步ORM框架。同时支持PostgreSQL, MySQL, SQLite 和 MSSQL等多种数据库，很适合作为Nodejs后端数据库的存储接口，为快速开发Nodejs应用奠定扎实、安全的基础。既然Nodejs的特点在于异步，没有理由不找一个强大的支持异步的ORM框架，与之配合。

目前Web应用的服务端最常用的环境为JDBC，因而相关最常用的ORM框架为MyBatis。MyBatis是轻量且简单的，它本质上就是编程化的使用sql语句，相比原始的sql语句，它的作用主要表现为让sql语句变得简洁，但从本质上来说结果集仍然需要手动定义，然后和对象一一对应，来完成前后端的对接。而在Nodejs环境里，因为运行的本来就是JS语言，直接操作JS的对象来对应ORM结果集则会显得更加便捷自然。比如使用MyBatis时是可能在使用接口时出现对象不匹配的错误，而使用Sequelize则不会出现这种情况。当然更重要的优点在于Sequelize契合Nodejs最主要的特点异步，在性能流畅度和系统稳定性上都有所提升。

### 3.2.3 Sequelize使用方法

Sequelize可以直接操作对象进行常规数据库操作，而不用直接使用任何sql语句。因而它不是一个轻量级的框架，所有数据库操作都经过了大量的封装，从而提高开发效率。

安装过程与普通Nodejs插件相同，使用npm直接添加。建立数据库连接与配置过程与相关常规操作无异，此处主要介绍它的数据库操作：

1.定义Model，与表对应字段与外键



2.在Service里使用，下面是一个多表级联查找的实例，可见用直观的配置即可完成复杂的sql语句操作：



## 3.3 响应式布局技术

响应式布局就是一个网站能够兼容多个终端，而不是为每个终端做一个特定的版本。这个概念随着移动设备的兴起而深入人心。页面的布局会会随着你拖动浏览器窗口大小变化而变化。响应式布局没有绝对的话好与坏，需要根据网站的性质来确定，比如的页面元素非常多，页面需要包括所有屏幕尺寸的样式显示不是很好操作，如果页面元素较少，反而放在一起方便维护。

实现的方法很简单，使用CSS3的媒体查询就行了。在使用时，因为现代浏览器在字体及页面元素缩放方面已经做得很完善，我们只用考虑可能的设备宽度，即不同宽度区间使用不同的布局样式即可。通用的解决方案即网格布局，比如在容器内每行有24个长度单位，设定某元素在不同设备所占的长度单位不同，就能达到响应式布局的效果，这种网格布局在所有的应用框架内都各有内置的实现。具体到本系统使用的Elementui使用，如下图所示：



本例中在设备为xl（宽度≥1920px）时占12个长度，即一半，在设备为sm（宽度≥768px）时占24个长度，即全部内容,所有的页面元素都做了响应式设计，也并非全部使用网格布局实现。

# 第四章 触摸屏农技百科系统设计

## 4.1系统需求分析

### 4.1.1 需求确定原则

信息化农业是指以农业信息科学为理论指导，农业信息技术为工具，用信息流调控农业活动的全过程，以信息和知识投入为主体的可持续发展的新型农业，是农业现代化的高级阶段。近年来，我国将信息化作为加快农业农村现代化发展的重要条件，同时大力开展农业物联网试验示范、信息进村入户、农民手机技能培训等工作，期间也是取得了较为良好的效果。但是，更多的农户关心的是如何实现农业农村信息化和智慧农业这一问题。基于这一现实需求，以新的技术为基石，搭建一个更加全面，实用，高效，且在维护成本与扩展成本上都符合轻量的需求，结合最新的互联网开发技术的农技百科系统，具有十分重要的意义。

在信息化农业时代，快捷方便地信息获取非常重要，对所有农业相关信息实时地获取会使相关农产品的生产过程事半功倍，也为农产品的利润提供了保障。因此，农技百科系统的设计应满足以下需求：

（1）基于B/S模式的信息系统，既方便了系统的升级维护，也避免了任何原生安装过程，更符合农业信息快速更新的特点。

（2）系统的信息展示有大量的图片，需要搭建独立的图片服务器CDN，且需要按特定生成规则管理。

（3）使用Echart等前沿插件库以更好的实现系统中数据的可视化，既方便用户使用，也适合在现有功能上作增补。

（4）系统在前后台的前端工程均使用Vue.js实现，因而有较好的模块化设计。系统在模块化的同时，在数据层的设计上应该降低耦合，因为这是一个初步且基础的系统，需要有良好的系统拓展性。

（5）完善的后台管理功能，能快速增删改查系统数据，做到农业信息快速更新。

### 4.1.2系统功能需求分析

本系统目前还是一个基础的系统，一方面受现有数据总量的限制，能展示的内容有限，因而现有功能有许多可拓展的地方。总体来说，本系统在总体架构上基于现有可展示数据的内容与层次，并在规范上契合企业级应用，因而整个系统架构上的复杂度远超过可展示的内容，为更多的拓展需求作了良好准备，并且为以后的基于Nodejs的农技系统开发工作提供参考。基于现有数据的功能设计如下：

1.后台管理系统

后台管理系统主要功能在于对所有网站信息进行增删改查，和发送广播信息。同时管理系统本身也有完备的用户鉴权功能，符合一般的管理系统相关规范。如图4.1所示

管理员界面

作物管理模块

作物相关管理模块

农事相关管理模块

后台用户管理模块

增删改查

多样化操作

图4.1 后台管理系统基本结构

2.后台模块详细功能

模块的分类本身基于现有信息的分类，主要分为作物相关信息和农事相关信息。

（1）作物管理模块，所有作物相关模块基于这个模块，需要先添加作物信息才能添加作物相关信息。作物信息包括作物名称，描述，标签以及图片，图片字段为CDN，由图片上传接口统一生成。

（2）作物相关管理模块，包括四个模块：病虫诊断，作物栽培，良种推荐，每月农事。每个模块对应的信息不同，主要包括各自的专家信息的不同数据库字段。病虫诊断模块包括特定作物病害的基本介绍，病害的发病特点与防治方法。作物栽培模块包括几种不同的技术介绍。良种推荐模块包括良种的名称查询，品质产量和栽培要点。每月农事模块包括作物在不同月份的气候，栽培要点和注意事项。

（3）农事相关管理模块，包括四个模块：常用农药，用肥指南，食品安全，天气灾害。常用农药包括农药基本介绍，使用时的注意事项以及农药的使用方法。用肥指南可查询不同种肥料各自的使用相关信息。食品安全内容主要是相关政策条款的学习，为一个普法栏目。天气灾害模块为显示常见的天气灾害信息以及应对预防的手段。

（4）用户管理模块，功能为管理员用户增删改查用户信息以及为前台广播系统消息。

3.前台展示系统功能

前台系统无需登录，用户只需要浏览相关的农技信息以及接受系统广播，前台系统在功能上只作为信息的接收和展示层存在。

## 4.2系统功能设计

### 4.2.1后台管理系统模块设计

1.后台管理员登录流程图，如下图4.1所示：

登录

根据角色生成Token

请求接口

退出登录

接收请求并且验证token有效性

token是否过期

token是否过期

N

Y

N

Y

图4.2 后台管理登录流程图

一个功能完备的信息系统需要良好的拓展性以及完备的后台系统，因此农技百科系统在后台管理系统的设计上有鉴权功能，同时也契合目前市面上常用的一套后台管理系统逻辑。

2.作物管理模块设计

作物管理关联着与作物有关的其他模块，比如添加良种信息就要选择相应已经存在的作物。结构如图4.3所示：

作物管理

病虫诊断管理

作物栽培管理

每月农事管理

良种推荐管理

图4.3 作物管理模块

作物本身的名称和描述为相关信息所共用，且在前台表现上，作物本身的图片作为引导页面的作物标题图，因而在图片CDN的实现上需要有良好的管理，以便于管理和更换图片的显示。

3.作物相关管理模块设计

病虫管理，作物栽培，每月农事，良种推荐四个模块功能类似，单条信息就包括了每个信息用以展示的三个分页的富文本。这些功能旨在帮助农户快速查询到相应作物的专家信息，对农作物的栽培信息，良种采购，农事查询以及病虫诊断。

管理员功能需要对相应信息进行增删改查，且相应信息也有用以展示用的题图，同样需要独立的图片服务相关接口进行管理。

4.农事相关管理模块设计

现有的模块包括常用农药，用肥指南，食品安全及天气灾害，这些数据在存储结构上没有与其他耦合，这样设计的目的是提高拓展性，比如一个非常完整的电商网站项目，它的数据结构仍然可以是低耦合的。理论上与这些模块相似的功能可以无限拓展，且增加更多的农业信息变得非常方便。

这些模块的功能主要是农技百科系统中与作物本身没有强关联的信息的展示。作为天气灾害信息的发布，相关食品安全信息的普及，以及提供一些推荐的农药及废料的购买信息的发布，这些模块可以在项目迭代产生新需求的过程中不断拓展，以实现更多的新添加的信息展示功能。

5.用户管理模块设计

用户管理采用一套通用的相关信息管理系统的实现方式。管理员可以增删改查用户信息，为创建的角色分配权限，为用户分配角色。本系统只有后台管理系统需要登录，前台使用信息不需要登录，可以收到系统的广播信息。

### 4.2.2系统前台功能设计

本系统的前台主要作信息展示查询的功能，主要体现在布局与用户体验设计上。

所有页面元素都为响应式设计，可以适配多种设备。

1.页面结构

页面使用头部+两栏布局，头部有全屏功能，以及查看系统广播信息按钮，和跳转到后台管理系统的连接按钮。左侧边栏即功能选择导航，列表形式列举系统所有可用功能，功能本身和后台管理系统的模块对应。

为了使系统接近原生应用，在页面头部区域下方有tab标签栏，使用过的功能会自动生成标签，可点击切换到页面或关闭标签。

2.首页功能

首页也存在系统所有现有功能的导航，并能实时更新。且首页同时显示系统的一些元信息如信息量等，且包含两个统计图表显示系统的使用情况。

3.模块功能

每个不同的信息模块拥有各自的展示路径。各自的展示效果与功能也随信息本身的特点而各不相同，增强用户体验。

## 4.3 系统数据库设计

本系统的服务端采用ORM框架Sequelize，因此在设计数据库时不用作物理结构设计，而是先设计出对象实体，再由逆向工程生成数据库表。数据库使用MySql，总体的E-R图如图4.4所示：

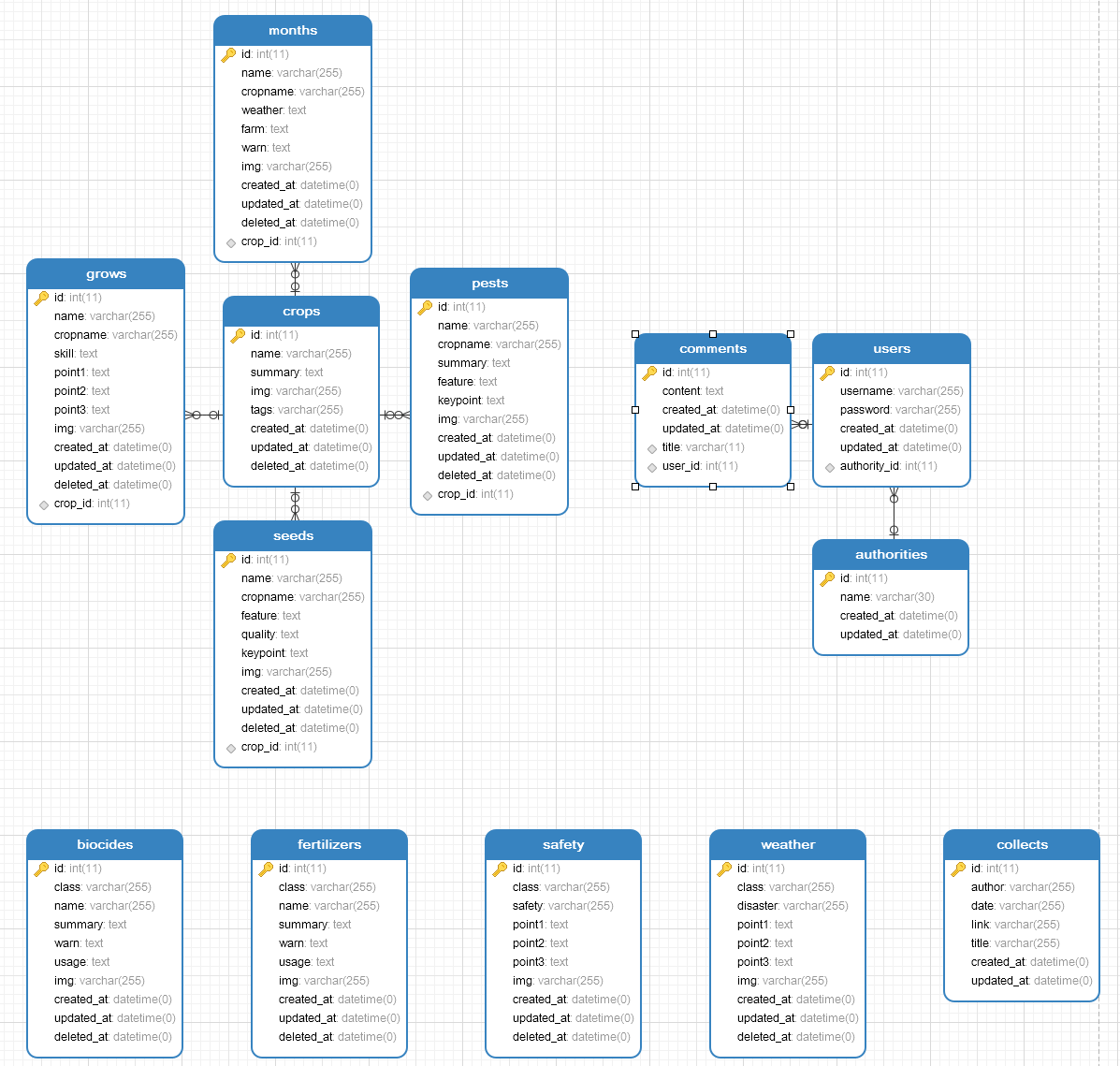


图4.4 总体E-R图

以下为部分对象实体信息，省略掉部分相似的实体。

（1）用户信息实体（User）

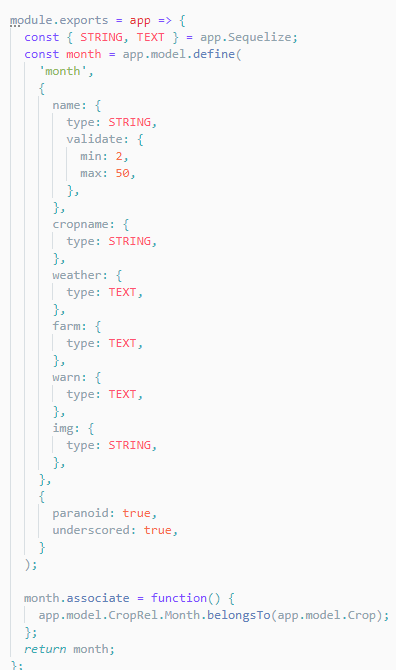
用户表主要存储用户信息，密码存储为MD5加密形式，且包含对应的角色信息id外键。



（2）作物信息实体



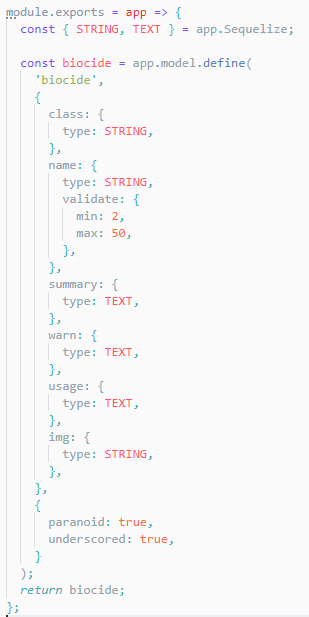
（3）月度农事实体



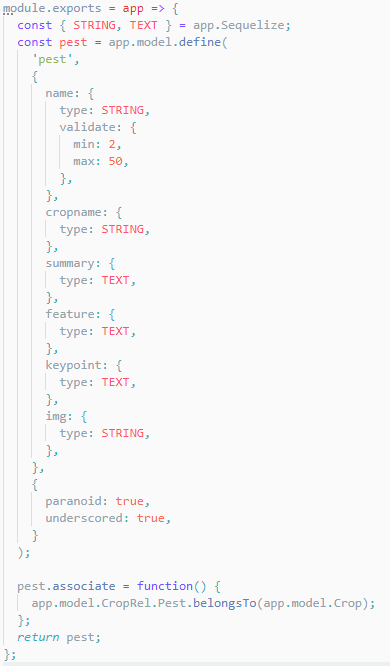
（4）系统消息信息实体



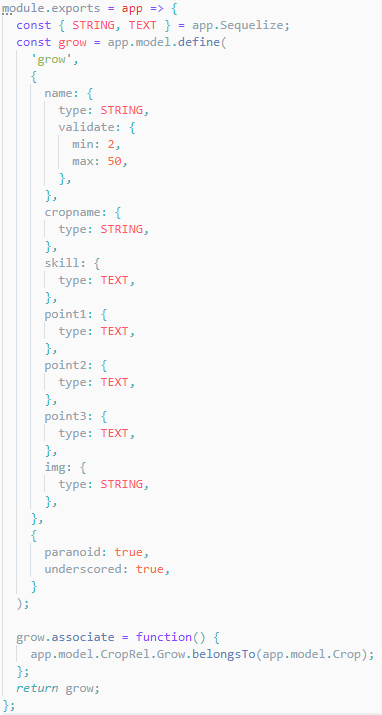
（5）常用农药实体



（6）病虫诊断实体



（7）作物栽培信息实体



（8）用户角色实体



## 4.4本章小结

本章内容在前面章节的基础上进行了拓展，首先分析了系统架构的设计原则即良好的拓展性，然后介绍了农技百科系统应有的功能，以及系统如何应用前面章节介绍的技术去实现，最后阐述了实体模型与数据库的设计方式。

# 第五章 触摸屏农技百科系统功能实现

## 5.1 后台管理系统功能

## 5.2 作物相关功能

## 5.3 农事相关功能

## 5.4 地图操作功能

# 结论与展望

## 结论

## 展望

# 参考文献