1. 绪论

1.1前言

1.2选题背景和意义

随着时代和互联网的发展，传统企业的业务运营方式渐渐变得繁琐，低效，运营成本高昂。为了降低企业的运营成本，提高运营效率以及降低人员成本成了必由之路。基于互联网的BOSS(Business & Operation Support System)业务运营支撑系统应运而生，通过整合传统的ERP系统，CRM系统，计费系统，来打通各系统间数据交互的通道，还可以通过数据聚合分析来发现潜在的商业价值，优化企业业务流程。

由于机缘巧合，本人母亲工作的挖掘机租赁公司仍然处于传统的基于人工的业务运营系统下，随着业务的发展壮大，迫切地需要一套业务支撑系统。在对行业的了解和沟通后，我发现设备租赁行业仍缺少定制化的BOSS系统。作为即将步入软件行业的一员，可以在同时磨练自己专业技能的同时创造社会价值当然是一举两得的选择，于是就萌生了自己开发一套基础的BOSS系统的想法，希望能在完善后可以应用在实际生产当中。

1.3现状和发展情况

1.3.1 市场现状

经过一定的市场调研（百度搜索引擎结果，几家大型设备的租赁商运营方式），发现大型工程设备的租赁管理系统尚有很大的空缺。传统的ERP软件开发商如用友，金蝶等软件开发商暂未进入该市场，当前工程机械租赁公司的业务支撑主要有一下几种方式：

1. 基于Excel表格的传统记账方式，使用纸质回执的方式来控制业务流程，在面对大量业务时会产生较高的人工成本，并且易出错，且极难统计业务数据，洞察市场趋势。
2. 使用传统的ERP软件进行管理，效率较高，但缺乏自定义业务流程的支持，数据流通性差，且缺乏资产定位能力。
3. 使用企业订购的资产追踪系统附带的业务管理功能，初步将资产定位和业务结合在一起，但对业务流程的支撑较差。

由此可见，当前较多租赁企业的业务运营都处于高度依赖人工的状态下尤其是业务数据的收集工作。因此，当前的设备租赁的BOSS系统具有较高的市场价值。

1.3.2 同类竞品分析

荣峰科技开发的“管租易”是一个比较完善的租赁管理解决方案，在业内有一定的市场占有率，其以后台WEB客户端为核心，构建出包含微信公众号，安卓APP，GPS资产定位，等业务实施终端的一体化业务支撑系统，并拥有一定的业务统计能力，但由于其使用成本较高（标准配置下约15万/年）， 而且系统较复杂，因此限制了一些客户对业务系统的忠重视程度。

综上所述，开发一个较简单，并可以与传统EXCEL表格进行交互以存续历史业务数据的业务支撑系统就有了实际的意义，也就是本人所努力的目标。

1.4主要研究内容

1.4.1 BOSS（Business & Operation Support System）

全称业务运营支持系统，通常来说分为四个组成成分：计费及结算系统，营业与账务系统，客户服务系统以及决策支持系统，将其整合在一起便于数据通信以及纵向和横向管理。

1.4.2 前后端分离&动静分离

在早期的WEB应用开发中，大多数实现技术（PHP,Servlet，JSP等）都将业务逻辑和视图控制编写在同一个文件中，每个页面对应一个文件，处理代码间错综复杂的依赖关系和代码维护一直是业界难题，随着技术的发展，MVC模式（Model-View-Controller）渐渐成为公认的优秀设计思想，其对数据模型，视图，控制器的分层逻辑对之后的软件开发产生了深远的影响。近年来流行的前后端分离正是这一思想的衍生产物。

前后端分离，顾名思义，前端页面与后端接口完全分离开发，前端（浏览器）负责用户操作逻辑以及页面渲染，并通过API接口与后端（服务器）进行数据交互。后端则专注于实现API接口里的具体业务逻辑。在这种思想下，前后端都拥有了各自独立的两套MVC架构来实现各自的逻辑，并基于XML、JSON等数据交换格式，以及序列化和反序列化技术来实现前后端通信。需要注意的是，在后端的MVC结构中。View已近由原来的HTML文档演化为前端可以直接解析的JSON视图。

在这种开发模式下，WEB应用实现了静态化，HTML、CSS、JS等需要在浏览器中运行的静态文件可以存放到高性能的静态资源服务器（如Nginx等），而Web服务器（如tomcat，weblogic等）则可以有更多的资源来处理用户需求，以此实现了动静分离以及负载分担，提高了网站的性能以及稳定性。

从开发的角度来讲，前后端分离使得前后端开发人员直接面向接口编程，前端不用关心API的实现方式，后端不用关心页面的布局以及交互逻辑，这极大地提高了企业的开发效率，降低了运营成本以及后续运营的成本，往后的迭代升级只需要修改API或面向API进行向下兼容即可。

1.4.3 MVVM模式

MVVM全称Model-View-ViewModel（模型-视图-视图模型绑定）是前端MVC模式的改进版。如果你是新手前端开发者，一定会遇到类似“都XXXX年了你怎么还在用Jquery?”。不可否认的是，Jquery是一个非常强大的JS库，使得你对DOM的操纵得心应手，然而思想是在不断进步的。

试想你正在实现一个简单的数据表格，当用户加了一个筛选条件后我们需要重新渲染表格，最开始大家想的很简单：把TABLE下的DOM全部丢掉，重新渲染一遍。Jquery可以轻松胜任这项工作，但当你的页面布局比较复杂或者加入了一些动画效果时，就会产生严重的卡顿和画面撕裂，影响用户体验，这肯定不行。于是人们想出了先筛选出不需要重新渲染的DOM节点再对其他的DOM节点进行处理，这样显著的提高了界面性能，然而如果用JS和JQuery实现这个将是一件复杂的工作。

久而久之，人们发现处理视图渲染的工作有很高的复用性，MVVM应运而生，现在只需要修改Model,View-model的双向绑定就会帮你处理视图渲染的问题，并使用特定的JSON对象来描述一个虚拟DOM节点，并以此为依据来渲染DOM，以及实现部分更新的效果。大量的优秀MVVM框架如雨后春笋般爆发，其中的代表便是Vue.js、Angular等。在大家的努力下，我们拥有了可以在前端运行的“JSP”，只需要将模型放到想放的位置即可，数据处理交给框架，前端开发者更关心的是页面美感，用户也得到了更好的体验。

第二章 技术选型

介绍相关技术以及他们的差别 技术选型的理由。

2.1 技术选型

2.1.1 Spring

Spring 自2004年发布以来十几年的发展，已经成为Java应用开发中最重要的轻量级框架之一。与Java官方定义的EJB3.0，由JBOSS,ORACLE提供开源的和商业的解决方案不同，Spring 面向的是轻量级的，非分布式的中小型企业应用开发，其具有如下几个技术特点：

1. 依赖注入：将类的实例化，装配工作交由Spring统一管理，从而实现各组件间的解耦以及性能优化。
2. 低侵入性：配置工作通过XML或注解进行，对旧项目的适配相对友好。
3. 独立于WEB服务器：Spring可以独立运行或者运行在各种web容器中，可以很轻易的移植，而不像EJB那样需要大量的适配工作。
4. 开放的ORM整合：Spring 可以和多种第三方ORM解决方案整合，来简化底层数据库的访问工作。
5. AOP特性：Spring AOP可以对一些通用任务如事务，安全，日志进行统一的集中管理，提高代码复用率。

基于以上几种技术特点，我认为Spring是当下本项目的最佳解决方案，其组织各模块的模式也非常值得学习和探讨，因此，此项技术将用于实现后端的Service（服务层）以及DAO（数据访问对象）。

2.1.2 Struts 2

Struts2 源自传统的Struts 和 Webwork发展而来，是一个非常经典并且优秀的MVC框架，在前后端未分离，JSP大行其道的年代非常流行，以至于非常多的项目依然在使用它，并且其基于filter（拦截器）的实现思路非常适合新手学习和理解WEB应用的架构。本项目在初期也准备使用该技术，但经过调查研究后发现了其与项目需求相冲突的地方，这也直接导致了项目的第一此重构：

1. 对前端广泛使用的JSON数据格式支持较差，参数的反序列化需要引入比较古老的JSON插件，序列化则需要直接处理response对象，对开发和后期维护产生了不利的影响。
2. 参数隔离和性能问题，在struts的action中，前端传入的参数默认将填充至Action对象的对应字段中，不同路由对应的API参数因此集中在一起，给数据安全带来了隐患。
3. 对跨域请求的支持很差，在动静分离的项目中，我们有两种方式来区分浏览器请求的是静态资源还是API接口，第一种是使用Nginx配置一个反向代理，将发送给API的请求转发给WEB容器。第二种是使用跨域请求，这种方式对用户的带宽要求较高（需要额外发送一次请求用于验证）。

因此，尽管struts2 是一个我比较熟悉的WEB框架，但是项目的需求更为重要，因此我放弃使用它。

2.1.3 SpringMVC

SpringMVC作为Spring全家桶中的一员，因其与Spring无缝集成，正逐渐取代Struts2成为主流的Web框架，这里简单列举其与本项目的需求的契合之处：

1. 跨域请求支持：当不准备使用反向代理时，静态资源和API是在不同的域下的，因此需要发送跨域请求来访问接口。
2. 安全的参数装载方式：与Struts2 将参数装载至字段中不同，SpringMVC将请求参数装载为方法参数，开发者可以处理其中的参数映射，并且由于其核心Servlet为单例模式，方法参数可以在保证线程安全的情况下进行并发操作。
3. 完整的JSON视图支持，SpringMVC将java对象和JSON中的序列化看作视图处理，只需要指定返回的视图为JsonView或者在方法上注解@ResponseBody 将其指定为返回体。

出于以上需求，我在最终实施的时候选择SpringMVC作为后端的MVC框架。

2.1.4 Hibernate

Hibernate作为一款成熟的ORM框架，其完整地实现了JPA方案。全映射作为其重要特性与JPA规范相结合使得开发人员不必关心对象和关系映射问题，直接面向对象编程，使得快速开发成为了可能，是个人开发小型系统的优秀选择。但其也存在着优化困难，效率低下等问题，关于如何解决这些问题大家都有各自的见解，我也将在下文中阐述我自己的解决方案。

2.1.5 Mybatis

Mybatis 也是老牌的ORM框架之一，但其实现方式却与Hibernate大相径庭，其实现逻辑更像是一个SQL语句的拼装器，开发人员可以直接操控SQL语句，通过各种条件来判断以实现动态地组装SQL语句，并根据映射返回一个包装完成的对象。其实现方式决定了其相较Hibernate拥有更好的性能，而且较易优化，所以在对性能要求较高的移动互联网领域非常流行。当然，其非全映射的特性也限制了其级联操作的能力，并且开发者即使在工具的帮助下仍然需要花费大量的精力去维护映射关系，这也就是我选择Hibernate作为持久层解决方案的原因之一。

2.1.6 MySQL

MySQL 是一种典型的轻量级关系数据库系统，对于中小型企业网站来说，MySQL可以说是数据持久化的最优解决方案，其开源免费，体积小，性能好，使用成本低等优点深受中小型企业和开发者的喜爱。跟其他的关系型数据库系统（Oracle，SQL server）等相比虽然在集群，海量存储上有一定的不足之处，但对于中小企业来说绰绰有余。由于MySQL支持多种数据库引擎，开发者根据其具体需求来对每个表选择相应的数据库引擎，本项目中大多使用InnoDB引擎，其支持的事务机制，外键支持，相对简单的自增长主键是我选择InnoDB引擎的主要原因。

2.1.7 JSP/静态HTML

JSP是一种以Servlet发展而来的动态网页技术。在支持JSP的web容器启动时，容器将根据JSP生成对应的Servlet来对用户请求进行处理，并将控制层传入的填入其中，最后返回给用户。在早期的web开发中，由于服务器性能，浏览器执行JS脚本的的性能，网页渲染速度等多方面因素，JSP的设计思想是非常合理的，随着技术的发展，以及对性能和用户体验的要求，人们注意到JSP的缺点：

1. 网页中静态部分占大多数，而Servlet几乎将所有的内容当作动态内容处理，将内容逐行输出，这显然造成了一部分不必要的性能开销，在有高并发的需求下代价非常高昂。
2. 数据模型缺失，在缺乏JS数组或对象作为前端的数据模型是，从前端收集数据就变得非常复杂且不稳定，使得一些常用组件（数据表格等）的实现变得非常复杂。在过去，这种限制使得用户界面往往存在功能有限，需要频繁刷新页面，给用户带来较差的体验。
3. JSP的调试必须在Web容器中进行，这使得前端开发人员需要一定的JAVA能力，而且给不明确的分工会给项目的开发带来不可预知的风险。

正是由于上面面种种局限性，除了老项目依然在使用JSP，目前大部分项目都使用了前后端分离的开发模式，本项目也模拟了这种模式。

2.1.8 Vue.js

对于后端程序员来说，这就是前端版的JSP，二者的逻辑相似到连{{}}标记都是那么亲切，这也是2016年至今最流行的前端MVVM模板框架之一。其中View-model实现的模型视图双向绑定都是以前的JSP无法想象的，基于Vue.js的前端网页只需要通过接口从后端获取数据，然后再修改vue托管的数据模型即可，在下文中我将详细描述其在项目中的应用。

2.1.9 Tomcat

当下流行的轻量级WEB容器，性能稳定并且开源免费，因此十分受个人开发者和小型企业的青睐。对于javaee的初学者来说，其相对简单的架构更适合学习JavaWeb的实现原理。通常来说，Tomcat可以被看作一个控制JSP/Servlet生命周期的容器，使用类似线程生命周期的模式实现了Servlet规范，对于性能要求不高的项目来说，tomcat是相当符合应用需求的，这也是我选用tomcat的原因之一。

2.2 技术选型总结

2.2.1 平衡用户体验以及性能要求

用户体验永远是产品最先考虑的组成成分之一，在实现上主要由两部分组成，其一是产品设计，其二是技术实现，这里只讨论技术实现。由于早期的电脑性能限制以及设计观念，从前的IE内核渲染DOM时使用单线程，这也就使得用户的其对JS操纵DOM的支持非常孱弱，性能低下，也就使得当时的网页大多是静态页面。

随着技术的发展以及谷歌的崛起，Chrome内核的浏览器渐渐占据了市场的半壁江山，其优秀的渲染性能以及稳定性使得单页面富应用的实现成为了可能，其提供的交互体验远胜于从前的IE，因此渐渐获得用户的认可。有了具体的技术支撑，我们开发者当然也要更进。

为了实现良好的用户体验，本系统抛弃了传统的iframe实现多页面的方案，完全使用JS实现单页面前端路由，并对其中的参数进行统一的管理，相关的技术实现将在后面的章节中阐述。

2.2.2 最终选型

经过多次项目重构后，我最终选择了下面的技术来实现本系统：

后端API/路由实现：SpringBoot(SpringMVC+Spring+Hibernate),Shiro权限验证。

前端路由实现：Vue.js，js,jquery

前端页面实现：mdui,layui。

WEB容器：Tomcat

持久化方案：MySQL.

以上是项目当前使用的解决方案，具体的业务实现主要使用JAVA以及Javascript 。

第三章 系统总体设计

3.1 用户需求分析

经过与客户的沟通后，我们大致得出一下几点用户需求：

3.1.1 订单管理

接单员接到客户的订单之后，将订单详情录入系统，主要包括下单人（工地老板）姓名及电话，工地地址，需要的机型，工程内容，价格，工期等。

财务或老板，通过查询系统即可了解信息，无需一个个过问知情人。

后续工作人员按照订单办事，机手可以按照对应的订单号上报日报，文员找到对应的订单号录入，当订单完工，则按订单生成对应的对账单，找客户对账收款，收到款后，财务再找对应的订单销账。

整个管理系统以订单为主线，严格区分不同订单，机手日报要对应订单号。工地完工并收款销账后，就关闭订单。

3.1.2客户管理

接单员在录入订单详情的时候也录入了下单员（工地老板）的姓名及电话，相应的工地施工员也录入在他工地老板的二级科目下。也就是查一个工地老板，他名下的施工员全都跑出来；查一个施工员，对应的老板也都跑出来。让工地老板和施工员都联系起来。方便客户管理及保存，并促进收款。

3.1.3派单管理

每个机手一个手机客户端，机手管理人员将工地老板联系电话，工地地址，机型机号、对应的订单号等信息，通过平台发送给相应的机手。

机手接到信息后，前往指定的地点工作，每天在手机端找到对应的订单号，上报当天的工作情况。

财务及相关信息使用人，可以通过系统查看到每个机手、每台机器分别在哪里干活。无需一一询问。

3.1.4租赁明细管理

每个机手一个手机客户端，直接找到对应的订单号，录入当天的工作情况，（或者由一个文员每天通过微信群获取机手汇报的日报，找到对应的订单号录入到系统中）。主要录入的信息有日期、机型机号、土方时间、破碎时间、运费、单据编号、机手等。

系统能根据这些信息再结合订单详情，自动生成对账单，能让我们知道应收金额、工地老板姓名电话及相关的租赁明细。

系统还需要有一个保存图片的功能，要保存的照片就是工时签收单的照片，以备后期查询并用于对账。

3.1.5对账单打印

工地完工后，系统自动生成对账单，任何人都可以操作打印出对账单。

3.1.6收款管理

我司工地负责人带着对账单去对账收款，收到款项后，财务根据收款信息核销账款，关闭订单。

结合客户管理功能，系统生成的对账单，只要前期数据录入无误，后期生成的对账单准确率是相对较高的。

收款查询功能要能以工地、老板、施工员为单位，搜索出对应的收款信息，如总应收多少，已收多少，优惠多少，未收多少等。方便我们进行收款管理。

3.1.7固定资产管理

固定资产的管理主要是对每台挖机的入账价值，机器折旧，修理修配等情况的核算，需要随时能查出该固定资产的现值、残值等。

其管理功能主要包括固定资产的新增、修改、退出、转移、删除、借用、归还、维修、计算折旧率及残值率等日常工作。

如果能知道每台机器在哪里状态如何，是否开工运行那就更好了。

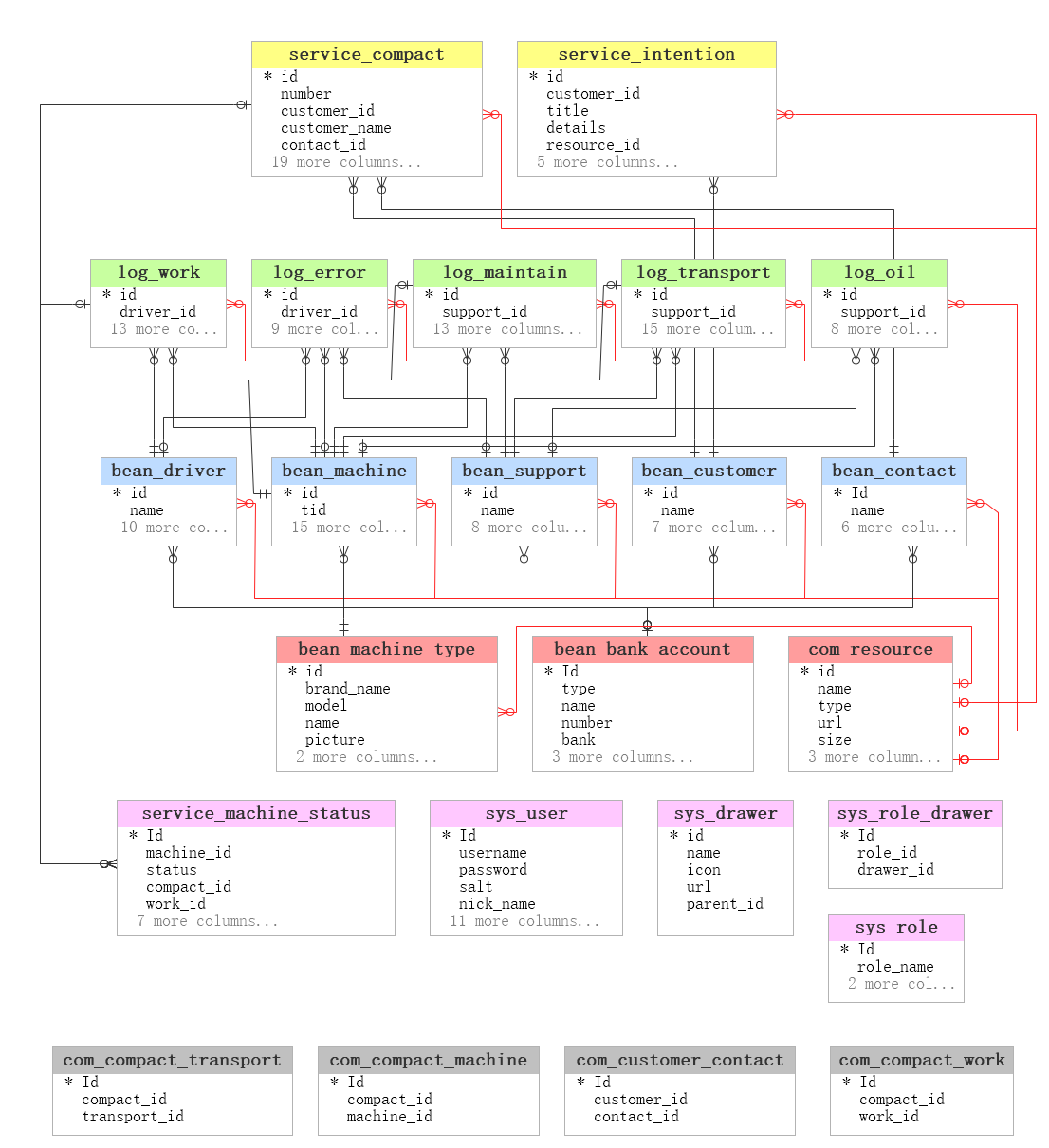
3.2 系统架构设计

通过上面的需求分析，我将本系统划分为如下图的几个模块：



其中，基础，日志中的每个模块都有一套CRUD（增删改查）操作来提供管理功能，业务模块中的挖机地图可以展示挖机当前的地理信息以及状态。资源系统中的模块提供系统内的公共资源（业务文件、银行账号）的支持，模块之间有关联操作以方便查询。

3.3 数据库设计



考虑到企业ERP系统的复杂程度，本系统在开发之初就对业务流程进行了一定程度的简化，因此本系统并不是全功能的ERP系统，整体更侧重于生产控制管理。我结合客户的实际需求和自己的理解，以及优化实体关系等，将数据库分成了以下几个层次：

1. 角色层：客户信息(bean\_customer)，客户联系人(bean\_contact)，挖掘机驾驶员(bean\_driver)，后勤人员(bean\_support)，挖掘机(bean\_machine)。
2. 日志层：异常处理(log\_error)，加油(log\_oil)，维修(log\_maintain)，转场(log\_transport)，机手工单(log\_work)。
3. 业务层：合同(service\_compact)，客户意向(service\_intention)。
4. 系统层：系统菜单管理(sys\_drawer)，系统用户管理(sys\_user,sys\_role,sys\_role\_drawer)，挖掘机状态机(service\_machine\_status)
5. 公共资源层：银行账号(bean\_bank\_account)，文件系统(com\_resource)，挖掘机资料(bean\_machine\_type )。

此外，数据库中还包含了一些用于处理多对多关系和系统后续升级预留的多对一关系，以及优化数据库性能的中间连接表，它们的表名都是以com开头的，以此表明他们的身份。

详细结构：

角色层：

bean\_customer 客户信息表

字段

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **id** | int(11) unsigned | <auto\_increment> | 客户标识 |
| name | varchar(64) |  | 客户公司名称 |
| type | varchar(11) | 普通 | 客户类型 |
| status | varchar(11) | 正常 | 客户状态 |
| detail | varchar(256) | <空> | 客户描述 |
| resource\_id | int(11) unsigned | <空> | 资源ID |
| bank\_account\_id | int(11) unsigned | <空> | 银行账户信息 |
| create\_time | timestamp | <INSERT-TimeStamp> |  |
| edit\_time | timestamp | 0000-00-00 00:00:00 |  |

约束：

|  |  |
| --- | --- |
| 名称 | 类型 |
| customer\_bank\_account | bank\_account\_id -> bean\_bank\_account.Id |
| customer\_resource | resource\_id -> com\_resource.id |

bean\_contact 客户联系人

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **Id** | int(11) unsigned | <auto\_increment> |  |
| name | varchar(30) | <空> | 姓名 |
| role | varchar(30) | <空> | 角色 |
| tel | varchar(50) | <空> | 电话 |
| person\_id | varchar(50) | <空> | 身份证号码 |
| detail | varchar(255) | <空> | 备注 |
| resource\_id | int(11) unsigned | <空> | 资源ID |
| bank\_account\_id | int(11) unsigned | <空> |  |

约束：

|  |  |
| --- | --- |
| 名称 | 类型 |
| contact\_bank\_account | bank\_account\_id -> bean\_bank\_account.Id |
| contact\_resource | resource\_id -> com\_resource.id |

客户-联系人连接表

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 默认值 |
| **Id** | int(11) unsigned | <auto\_increment> |
| customer\_id | int(11) unsigned | <空> |
| contact\_id | int(11) unsigned | <空> |

bean\_driver 驾驶员信息

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **id** | int(11) unsigned | <auto\_increment> | 机手ID |
| name | varchar(16) |  | 机手姓名 |
| tel | varchar(16) |  | 机手联系电话 |
| person\_id | varchar(32) |  | 身份证号码 |
| paper\_id | varchar(32) | <空> | 驾驶许可证号码 |
| status | varchar(11) | 空闲 | 机手状态 |
| type | varchar(11) | 普通 | 机手类型 |
| resource\_id | int(11) unsigned | <空> | 资源ID |
| bank\_account\_id | int(11) unsigned | <空> | 银行账号信息 |
| head\_pic | varchar(300) | resource | 头像地址 |
| create\_time | timestamp | <INSERT-TimeStamp> |  |
| edit\_time | timestamp | 0000-00-00 00:00:00 |  |

约束：

|  |  |
| --- | --- |
| 名称 | 类型 |
| driver\_bank\_account | bank\_account\_id -> bean\_bank\_account.Id |
| driver\_resource | resource\_id -> com\_resource.id |

Bean\_support 后勤人员信息

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **id** | int(11) unsigned | <auto\_increment> | 后勤ID |
| name | varchar(16) |  | 后勤姓名 |
| tel | varchar(16) |  | 后勤电话 |
| status | varchar(11) | 空闲 | 后勤状态 |
| type | varchar(11) | 普通 | 后勤种类 |
| bank\_account\_id | int(11) unsigned | <空> | 银行账号信息 |
| resource\_id | int(11) unsigned | <空> |  |
| head\_pic | varchar(200) | <空> | 头像地址 |
| create\_time | timestamp | <INSERT-TimeStamp> |  |
| edit\_time | timestamp | 0000-00-00 00:00:00 |  |

约束：

|  |  |
| --- | --- |
| 名称 | 类型 |
| support\_bank\_account | bank\_account\_id -> bean\_bank\_account.Id |
| support\_resource | resource\_id -> com\_resource.id |

Bean\_machine挖掘机信息

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **id** | int(11) unsigned | <auto\_increment> | 机器ID |
| tid | int(11) unsigned | 0 | 机器型号ID |
| nameplate | varchar(50) | <空> | 铭牌 |
| code | varchar(32) | <空> | 机器自编号 |
| owner | varchar(20) | <空> | 机主 |
| payway | varchar(20) | <空> | 购买方式 |
| price | int(11) | 0 | 参考价格 |
| m\_pay | int(11) | 0 | 月供 |
| status | varchar(11) | 空闲 | 状态 |
| worktime | float(7,2) unsigned | <空> | 工作时间 |
| xyz | varchar(64) | <空> | GPS坐标 |
| resource\_id | int(11) unsigned | <空> | 资源ID |
| detail | varchar(256) | <空> | 详细备注 |
| create\_time | timestamp | <INSERT-TimeStamp> |  |
| edit\_time | timestamp | 0000-00-00 00:00:00 |  |
| birth | timestamp | 0000-00-00 00:00:00 |  |
| dead\_line | timestamp | 0000-00-00 00:00:00 |  |

约束：

|  |  |
| --- | --- |
| 名称 | 类型 |
| machine\_machine\_type | tid -> bean\_machine\_type.id |
| machine\_resource | resource\_id -> com\_resource.id |

日志层：

Log\_oil 加油日志

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **id** | int(11) unsigned | <auto\_increment> | 加油ID |
| support\_id | int(11) unsigned | <空> | 后勤ID |
| machine\_id | int(11) unsigned | <空> | 机器ID |
| litre | double(11,2) | 0.00 | 加油量 |
| money | double(11,2) | 0.00 | 加油金额 |
| details | varchar(256) | <空> | 详细描述 |
| resource\_id | int(11) unsigned | <空> | 资源ID |
| log\_mode | varchar(11) | <空> | 日志状态 |
| create\_time | timestamp | <INSERT-TimeStamp> |  |
| edit\_time | timestamp | 0000-00-00 00:00:00 |  |

约束：

|  |  |
| --- | --- |
| 名称 | 类型 |
| oil\_machine | machine\_id -> bean\_machine.id |
| oil\_resource | resource\_id -> com\_resource.id |
| oil\_support | support\_id -> bean\_support.id |

Log\_transport 转场日志

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **id** | int(11) unsigned | <auto\_increment> | 转场单 |
| support\_id | int(11) unsigned | 0 | 后勤人员ID |
| support\_name | varchar(16) | <空> | 后勤人员姓名 |
| support\_tel | varchar(16) | <空> | 后勤人员电话 |
| machine\_id | int(11) unsigned | 0 | 机器ID |
| departure | varchar(256) | <空> | 起点 |
| destination | varchar(256) | <空> | 终点 |
| type | varchar(16) | <空> | 类型 |
| money | double(11,2) | <空> | 转场金额 |
| details | varchar(256) | <空> | 描述 |
| resource\_id | int(11) unsigned | <空> | 资源ID |
| log\_mode | varchar(11) | <空> | 状态 |
| working | varchar(11) | <空> | 执行状态 |
| create\_time | timestamp | <INSERT-TimeStamp> |  |
| edit\_time | timestamp | 0000-00-00 00:00:00 |  |
| start\_time | timestamp | 0000-00-00 00:00:00 |  |
| end\_time | timestamp | 0000-00-00 00:00:00 |  |

约束：

|  |  |
| --- | --- |
| 名称 | 类型 |
| transport\_machine | machine\_id -> bean\_machine.id |
| transport\_resource | resource\_id -> com\_resource.id |
| transport\_support | support\_id -> bean\_support.id |

Log\_error 异常处理日志

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **id** | int(11) unsigned | <auto\_increment> | 异常ID |
| driver\_id | int(11) unsigned | <空> | 提交机手 |
| support\_id | int(11) unsigned | <空> | 后勤ID |
| machine\_id | int(11) unsigned | <空> | 机器ID |
| type | varchar(11) | <空> | 异常种类 |
| title | varchar(128) | <空> | 异常标题 |
| details | varchar(256) | <空> | 异常描述 |
| resource\_id | int(11) unsigned | <空> | 资源ID |
| log\_mode | varchar(11) | <空> | 日志状态 |
| create\_time | timestamp | <INSERT-TimeStamp> |  |
| edit\_time | timestamp | 0000-00-00 00:00:00 |  |

约束：

|  |  |
| --- | --- |
| 名称 | 类型 |
| error\_driver | driver\_id -> bean\_driver.id |
| error\_machine | machine\_id -> bean\_machine.id |
| error\_resource | resource\_id -> com\_resource.id |
| error\_support | support\_id -> bean\_support.id |

Log\_maintain 维修记录表

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **id** | int(11) unsigned | <auto\_increment> | 维修单ID |
| support\_id | int(11) unsigned | 0 | 后勤人员ID |
| support\_name | varchar(16) | <空> | 后勤人员姓名 |
| support\_tel | varchar(16) | <空> | 后勤人员电话 |
| machine\_id | int(11) unsigned | 0 | 机器ID |
| type | varchar(16) | <空> | 维修类型 |
| money | double(11,2) | <空> | 维修金额 |
| details | varchar(256) | <空> | 维修详情 |
| resource\_id | int(11) unsigned | <空> | 资源ID |
| log\_mode | varchar(11) | <空> | 日志状态 |
| working | varchar(11) | <空> | 执行状态 |
| create\_time | timestamp | <INSERT-TimeStamp> |  |
| edit\_time | timestamp | 0000-00-00 00:00:00 |  |
| start\_time | timestamp | 0000-00-00 00:00:00 |  |
| end\_time | timestamp | 0000-00-00 00:00:00 |  |

约束：

|  |  |
| --- | --- |
| 名称 | 类型 |
| maintain\_machine | machine\_id -> bean\_machine.id |
| maintain\_resource | resource\_id -> com\_resource.id |
| maintain\_support | support\_id -> bean\_support.id |

Log\_work 工单记录

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **id** | int(11) unsigned | <auto\_increment> | 工单ID |
| driver\_id | int(11) unsigned | 0 | 司机ID |
| driver\_name | varchar(16) | <空> | 机手姓名 |
| driver\_tel | varchar(16) | <空> | 机手电话 |
| machine\_id | int(11) unsigned | 0 | 机器ID |
| type | varchar(16) | <空> | 类型 |
| money | double | <空> | 金额 |
| details | varchar(256) | <空> | 描述 |
| resource\_id | int(11) unsigned | <空> | 资源ID |
| log\_mode | varchar(11) | <空> | 状态 |
| working | varchar(11) | <空> | 执行状态 |
| create\_time | timestamp | <INSERT-TimeStamp> |  |
| edit\_time | timestamp | 0000-00-00 00:00:00 |  |
| start\_time | timestamp | 0000-00-00 00:00:00 |  |
| end\_time | timestamp | 0000-00-00 00:00:00 |  |

约束：

|  |  |
| --- | --- |
| 名称 | 类型 |
| work\_driver | driver\_id -> bean\_driver.id |
| work\_machine | machine\_id -> bean\_machine.id |
| work\_resource | resource\_id -> com\_resource.id |

业务层：

Service\_compact 合同表

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **id** | int(11) unsigned | <auto\_increment> | 合同ID |
| number | varchar(11) | 0 | 合同编号 |
| customer\_id | int(11) unsigned | 0 | 关联客户信息 |
| customer\_name | varchar(64) | <空> |  |
| contact\_id | int(11) unsigned | 0 | 客户联系人信息 |
| type | varchar(11) | <空> | 合同种类 |
| bucket | int(11) unsigned | 0 | 挖斗数量 |
| hammer | int(11) unsigned | 0 | 破碎锤数量 |
| limit\_hour | double(11,2) unsigned | 440.00 | 每月限制使用时间 |
| total\_hour | double(11,2) unsigned | <空> | 实际使用时间 |
| extra\_hour | double(11,2) unsigned | <空> | 总计超出时间 |
| money\_promise | double(11,2) unsigned | <空> | 保证金 |
| money\_rent | double(11,2) unsigned | <空> | 每月租金 |
| money\_total | double(11,2) unsigned | <空> | 总金额 |
| money\_delay | double(11,2) unsigned | <空> | 违约金 |
| money\_fact | double(11,2) unsigned | <空> | 实收金额 |
| money\_free | double(11,2) unsigned | <空> | 优惠金额 |
| dest | varchar(256) | <空> | 租用地点 |
| resource\_id | int(11) unsigned | <空> | 资源ID |
| status | varchar(11) | 未审核 | 合同状态 |
| create\_time | timestamp | <INSERT-TimeStamp> | 创建时间 |
| edit\_time | timestamp | 0000-00-00 00:00:00 | 更新时间 |
| enter\_time | timestamp | 0000-00-00 00:00:00 | 进场时间 |
| start\_time | timestamp | 0000-00-00 00:00:00 | 开工时间 |

约束：

|  |
| --- |
|  |
| 名称 | 类型 |
| compact\_contact | contact\_id -> bean\_contact.Id |
| compact\_customer | customer\_id -> bean\_customer.id |
| compact\_resource | resource\_id -> com\_resource.id |

Service\_intention 客户意向登记

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **id** | int(11) | <auto\_increment> | 唯一ID |
| customer\_id | int(11) unsigned | 0 | 客户ID |
| Title | varchar(128) | <空> | 标题 |
| details | varchar(256) | <空> | 描述 |
| resource\_id | int(11) unsigned | <空> | 相关资源 |
| service\_mode | int(11) | <空> | 服务状态 |
| create\_time | timestamp | <INSERT-TimeStamp> |  |
| edit\_time | timestamp | 0000-00-00 00:00:00 |  |
| start\_time | timestamp | 0000-00-00 00:00:00 |  |
| end\_time | timestamp | 0000-00-00 00:00:00 |  |

约束：

|  |  |
| --- | --- |
| 名称 | 类型 |
| intention\_customer | customer\_id -> bean\_customer.id |
| intention\_resource | resource\_id -> com\_resource.id |

系统层：

Sys\_drawer 系统菜单管理

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **Id** | int(11) | <auto\_increment> | 唯一ID |
| Name | varchar(64) |  | 导航名称 |
| Icon | varchar(128) | <空> | 导航图标 |
| url | varchar(128) | <空> | 导航URL |
| parent\_id | int(11) | <空> | 自关联 |

约束：

Parent\_id为自关联字段，存储上级菜单ID，若无上级菜单则为空，用于存储级联菜单。

公共资源层：

com\_resource 文件系统

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **Id** | int(11) unsigned | <auto\_increment> | 资源ID |
| Name | varchar(128) |  | 文件名称 |
| Type | varchar(16) |  | 文件类型 |
| url | varchar(512) |  | 对应相对路径 |
| Size | int(11) unsigned | 0 | 资源大小 |
| parent\_id | int(11) | <空> | 父资源ID |
| create\_time | timestamp | <INSERT-TimeStamp> |  |
| edit\_time | timestamp | 0000-00-00 00:00:00 |  |

约束：

在文件系统中，没有父节点的节点被视为根节点，当本系统内其他角色（挖掘机驾驶员等）或日志（如加油单）附带其他媒体资源（图片，表格，PDF ，视频）时将会创建一个对应的父节点通过树状结构来组织这些文件，并记录他们的存储位置以供下载或编辑。

Bean\_bank\_account 公共银行账号

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **Id** | int(11) unsigned | <auto\_increment> |  |
| Type | varchar(50) | <空> | 账户类型 |
| Name | varchar(50) | <空> | 开户名 |
| number | varchar(50) | <空> | 账号 |
| Bank | varchar(100) | <空> | 开户行 |
| Detail | varchar(255) | <空> | 描述 |
| create\_time | timestamp | <INSERT-TimeStamp> |  |
| edit\_time | timestamp | 0000-00-00 00:00:00 |  |

描述：

公共表，存储着各个角色的银行账户信息。

Bean\_machine\_type 机器图鉴

字段：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 备注 |
| **Id** | int(11) unsigned | <auto\_increment> | 机器型号ID |
| brand\_name | varchar(32) | <空> | 机器品牌 |
| model | varchar(32) |  | 型号 |
| Name | varchar(50) | <空> | 通俗名称 |
| picture | varchar(300) | <空> | 型号照片 |
| Detail | varchar(256) | <空> | 详细描述 |
| resource\_id | int(11) unsigned | <空> | 资源ID |

描述：

存储基本的机器图鉴及其详细信息 待定。

第四章 系统实现

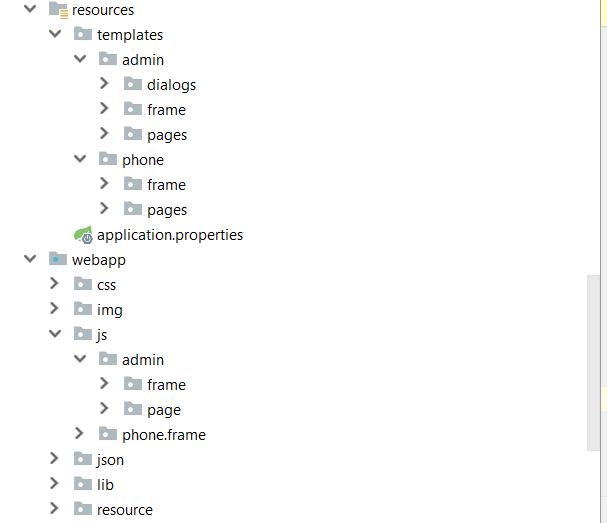
4.1 后端核心组件

4.2 前端核心组件及模板

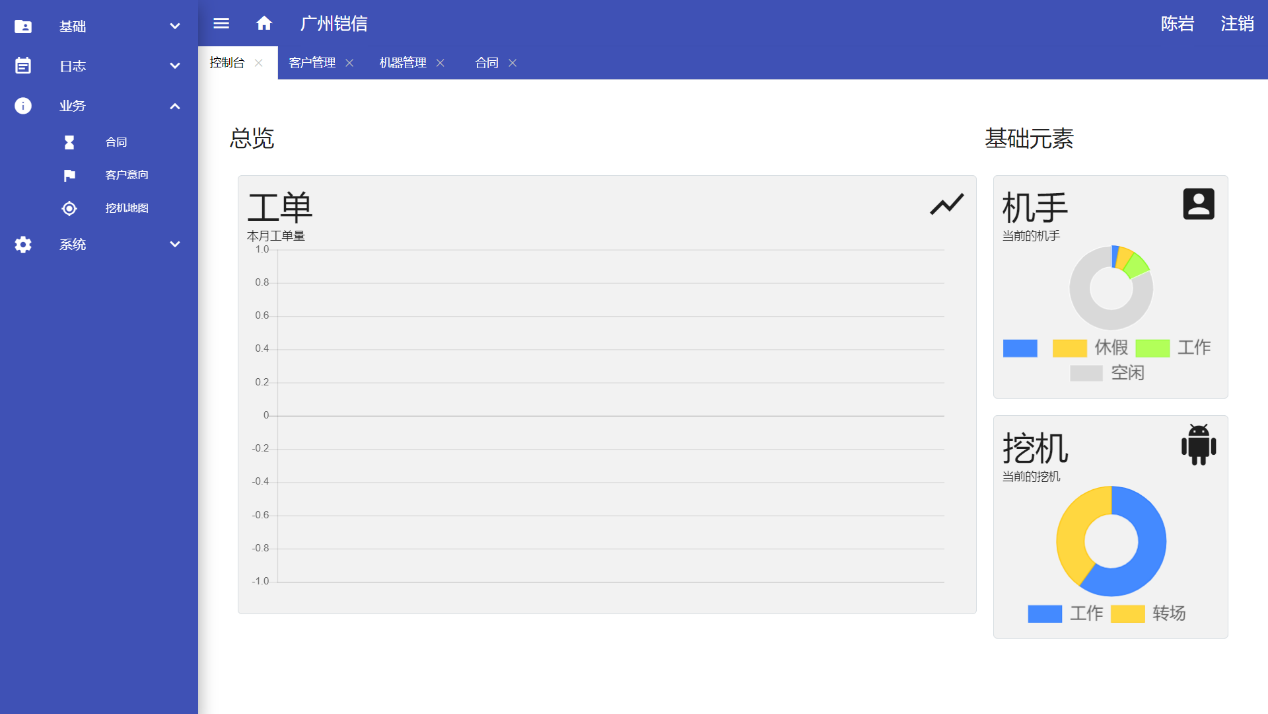
系统界面总体采用谷歌推出的Material Design设计风格，其界面类似安卓操作系统风格，搭配单页面设计能给用户良好的体验。

4.2.1 前端静态文件目录结构

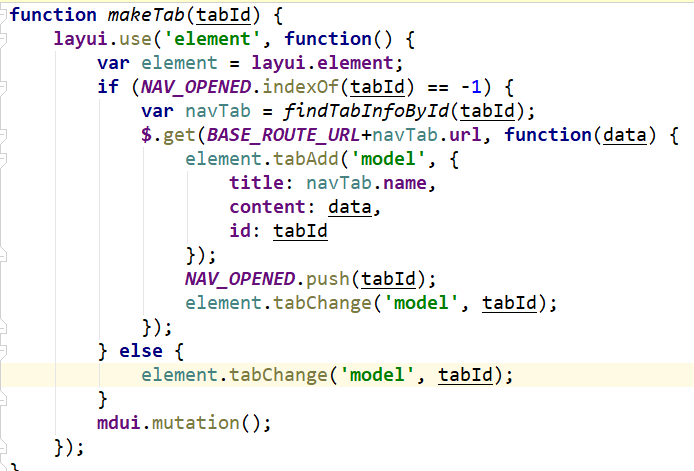
前端目录结构借鉴了微信小程序的目录结构，项目共用一个CSS文件，html文件都有相同文件名的js文件维护页面逻辑，通过AJAX实现动态加载来提升性能和用户体验。

如图所示，文件主要分为移动端和PC端网页两部分，每部分都包含了Dialog组件，frame主体框架，page子页面模板等目录。值得一提的是，所有的子页面模板都不包含完整的HTML结构（只包含一个外部的DIV），外部资源的引入都在frame框架中进行统一管理

4.2.2 主体框架设计

页面框架主要由三部分组成：左侧抽屉导航栏，顶部状态栏以及标签栏，右下角子页面容器。下面将介绍其具体实现方法：

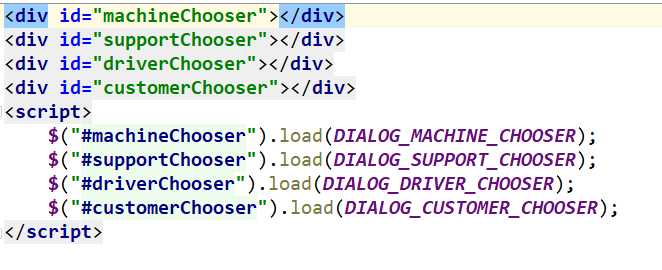
1. 左侧抽屉导航：HTML的布局方式为嵌套的ul标签，并通过vue.js中的v-for标签遍历一个二维数组渲染而成，该二维数组通过后端API获取，在数据库中的具体数据结构为一个带自关联外键字段的表，可以存储树状结构。
2. 顶部标签栏：顶部标签的标签项为ul，每个li对应一个div容器，通过获取被触发click事件的li标签中的id来查找对应的div容器并改变其display属性来控制子页面的显示和隐藏。每个页面都有其唯一ID,以此来实现左侧抽屉导航和标签栏之间的联动。具体实现如下图

每个左侧抽屉都关联了对应的页面初始化信息，当触发了点击事件时将从服务端申请模板进行渲染以及初始化工作，如页面在标签栏内未关闭，则切换到特定的页面。

1. 子页面容器：子页面容器将填充剩下的屏幕空间，所有已经加载的子页面将存放在容器中，容器将自动适应用户的缩放比例变化，并且限制变量作用域。

4.2.3 级联选择弹窗注册器

在dialogs目录下有一个名为dialogRegister.html的文件，文件内容如下图



在初始化框架时，该文件将被动态加载并执行其中的JS脚本来加载全局的选择弹窗，在需要进行级联选择的时候将弹出如下图的窗口



这个弹窗包含了一个快速的条件筛选，如后勤人员列表中可以通过后协勤人员姓名，电话来对结果进行筛选，选择的项会填入输入框中来起到限制输入的作用。

4.3 数据库优化

第五章 总结与期望

5.1 系统的缺陷及后续的升级计划

5.2 总结

2.1 相关技术