车辆租赁管理系统

**摘要：**随着互联网的发展，传统的车辆租赁行业的人工管理方式已经不再满足现阶段的需求。因此，利用现阶段的信息技术和系统管理技术对车辆租赁管理进行优化设计，对车辆租赁行业的发展有着重要意义。通过对车辆租赁行业的研究与对实际员工的操作了解，获得了车辆租赁管理系统的整体业务需求。通过对车辆租赁管理系统的设计与开发，使得车辆租赁公司的管理工作更加规范和清晰，大大缩减了企业的人力资源的开销和管理成本。

**关键字：**车辆租赁；数据库

**Car Rental Management System**

**Abstract：**With the development of Internet, the traditional manual management mode of vehicle rental industry no longer meets the current needs. Therefore, it is of great significance for the development of the vehicle rental industry to optimize the vehicle rental management by using the information technology and system management technology at the present stage. Through the study of the vehicle rental industry and the operation of the actual staff, the overall business needs of the vehicle rental management system are obtained. Through the design and development of the vehicle rental management system, the management of the vehicle rental company is more standardized and clear, which greatly reduces the cost of human resources and management.

**Keywords：**Car rental；SQL

**绪论**

由于近些年国内经济发展，导致国内汽车租赁行业快速发展。随着银行征信体系的完善和各大网络支付平台的信用体系的普及，以及中国公路交通基础设施建设的不断完善都为汽车出租行业在我国的迅猛发展提高了最坚实的基础。在我国汽车租赁产业还处在发展的初级阶段，目前我国市场上租车平台有一嗨租车、神州租车、悟空租车等等。汽车租赁公司在不断开发新的经营模式、提高服务质量、降低经营成本的同时，应充分有效地利用资源，实现信息化管理，提高工作效率，增强核心竞争力，这将是一项重大的课题，车辆租赁业作为朝阳产业，前景广阔但路途艰辛。

本次课题通过数据流程图设计、ER图设计进行初步的框架设计，再将设计好的ER图转换成SQL sever的语句来设计数据库，最后将设计好的数据库结合图形界面完成一个面向大众的车辆租赁管理系统。

汽车租赁业作为朝阳产业，前景广阔但路途艰辛。对一个汽车租赁公司来讲，摆在其面前需要首先解决的问题是对车辆与客户之间管理，本次课题便是解决对租赁车辆管理问题和客户租赁订单问题。

**1 车辆租赁管理系统需求分析**

* 1. **系统需求分析**

从系统需求层面来看，车辆租赁管理系统面向客户和企业员工，所以要在权限上设置不同对象。本次车辆租赁管理系统面对客户时要尽可能提供方便快捷的检索方式，而面对后台管理员时应提供严谨的操作准则。

* 1. **功能需求分析**

车辆租赁管理系统主要对车辆出租过程中的各项业务进行管理，通过对车辆出租过程的需求分析，整理出系统的六个基本功能模块。

根据实际的管理情况，系统的主要功能模块包含以下几个方面

车辆信息管理：车辆信息管理主要是实现对车辆的基本信息以及租赁状态的管理。包括对车辆信息的增、删、改、查等基本操作。

司机信息管理：司机信息管理主要实现对司机人员的基本信息以及出车状态的管理。其中按照人员管理的操作情况可以分为司机的添加、司机信息的编辑和司机删除等具体操作。

客户信息管理：对进行车辆租赁的客户实现基本信息管理和分类信息管理，主要包括添加、删除和修改客户租赁车辆的状态信息等功能

租赁信息管理：业务信息管理模块主要实现客户的租赁业务的管理，其中包括出租、续租、催车还交、还车结算等功能。

* 1. **性能分析**
     1. 可操作性

在车辆租赁管理系统中进行信息的编辑时，尽可能减少后台管理人员或者客户的操作步骤。这样便可以在一定程度上避免操作错误，从而导致查询效率的下降。

* + 1. 可靠性

车辆租赁系统管理系统的整体运行需要安全性和稳定性，所以定期要对SQL sever数据库进行检查，确保数据的正确性。

* + 1. 安全性

本系统需要对客户的权限做出严格的限制，避免客户进入后台的现象。

* + 1. 可移植性

本系统并非是专门为某一个汽车租赁公司设计开发，所以确保改系统具有良好的普适性和可移植性。

1. **开发环境介绍**

**2.1前端**

* + 1. Vue.js

Vue.js（读音 /vjuː/, 类似于 view）是一个构建数据驱动的 web 界面的渐进式框架。Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

* + 1. iView for Vue UI组件库

一套基于 Vue.js 的高质量UI 组件库

2.1.3 HTML

HTML 不是一种[编程语言](https://baike.baidu.com/item/%E7%BC%96%E7%A8%8B%E8%AF%AD%E8%A8%80/9845131)，而是一种标记语言 (markup language)，是网页制作所必备的。语言文档制作不是很复杂，但功能强大，支持不同[数据格式](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%A0%BC%E5%BC%8F/5198733)的文件镶入。

2.1.4 CSS

层叠样式表(英文全称：Cascading Style Sheets)是一种用来表现[HTML](https://baike.baidu.com/item/HTML)（[标准通用标记语言](https://baike.baidu.com/item/%E6%A0%87%E5%87%86%E9%80%9A%E7%94%A8%E6%A0%87%E8%AE%B0%E8%AF%AD%E8%A8%80/6805073)的一个应用）或[XML](https://baike.baidu.com/item/XML)（标准通用标记语言的一个子集）等文件样式的计算机语言。CSS不仅可以静态地修饰网页，还可以配合各种脚本语言动态地对网页各元素进行格式化。

**2.2后端**

2.2.1 Sql Sever

SQL Server是由Microsoft开发和推广的关系数据库管理系统（DBMS），其具有真正的客户机/服务器体系结构，图形化用户界面，使系统管理和[数据库管理](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86)更加直观、简单。丰富的[编程接口](https://baike.baidu.com/item/%E7%BC%96%E7%A8%8B%E6%8E%A5%E5%8F%A3)工具，为用户进行程序设计提供了更大的选择余地。SQL Server与Windows NT完全集成，利用了NT的许多功能，如发送和接受消息，管理登录安全性等。SQL Server也可以很好地与Microsoft BackOffice产品集成。

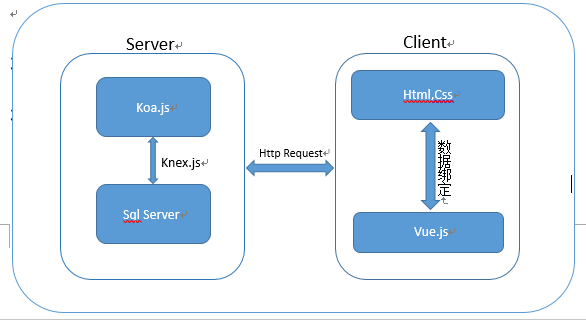
2.2.2 koa.js

Koa 是一个新的 web 框架，由 Express 幕后的原班人马打造， 致力于成为 web 应用和 API 开发领域中的一个更小、更富有表现力、更健壮的基石。 通过利用 async 函数，Koa 可丢弃回调函数，并有力地增强错误处理。 Koa 并没有捆绑任何中间件， 而是提供了一套优雅的方法，能够快速而愉快地编写服务端应用程序。

2.2.3 Knex.js

Knex.js是一个npm包,用于node.js下连接各种数据库并构建查询。

**2.3 项目模式示意图**



1. **系统设计**
   1. **功能模块设计**

通过对市面上的汽车租赁公司的管理模式进行研究分析，并将其结果与数据库结构相结合，从而为本次车辆租赁管理系统设计了六大模块。

* + 1. 系统信息管理模块

该模块包括系统维护，权限设置、更改密码等子功能模块

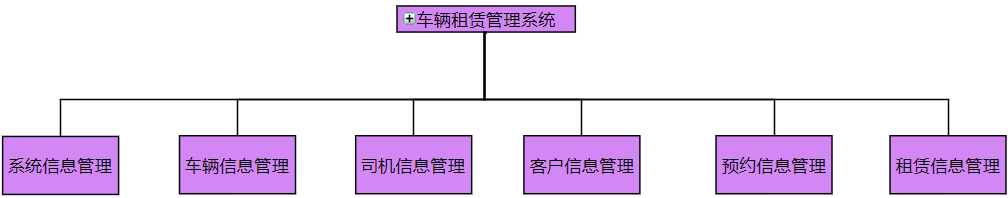
* + 1. 车辆信息管理模块

该模块是对租赁车辆的信息管理，包括对车辆的信息进行增、删、改、查等子功能模块

* + 1. 司机信息管理模块

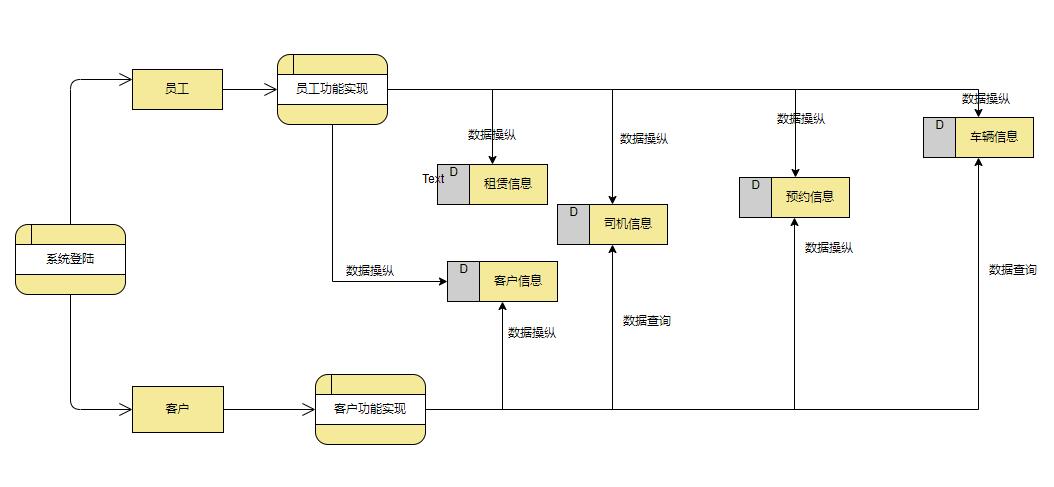
该模块是对司机的信息管理，包括对司机的信息的注册、注销、修改、查询等子功能模块

* + 1. 该模块是对进行车辆租赁的客户信息的管理工作，包括对客户信息的注册、注销、编辑、检索等子功能模块
    2. 租赁信息管理模块
    3. 该模块是对进行中的租赁订单的信息管理，包括对租赁订单的增、删、改、查等子功能模块
    4. 预约信息管理模块
    5. 该模块是对客户预约订单信息的管理和查询

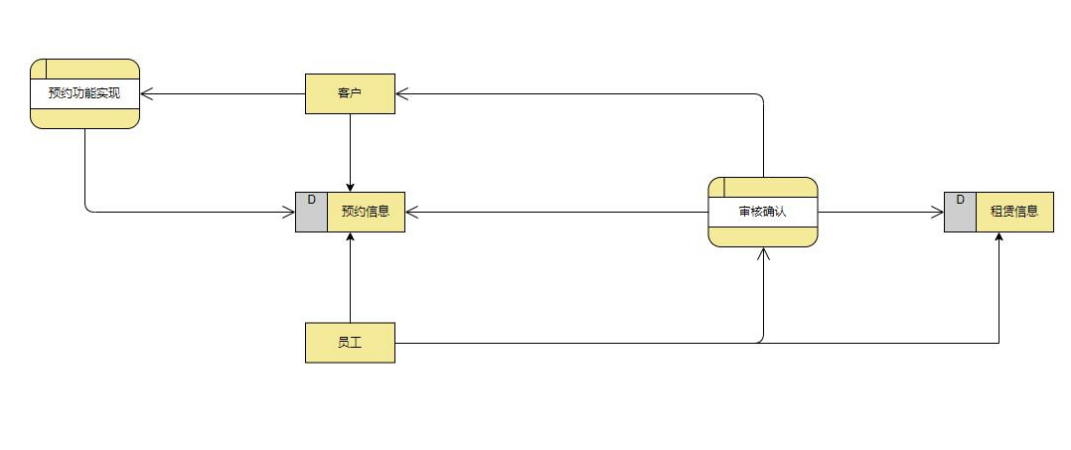


**图3.1 系统功能模块图**

* 1. **详细设计**

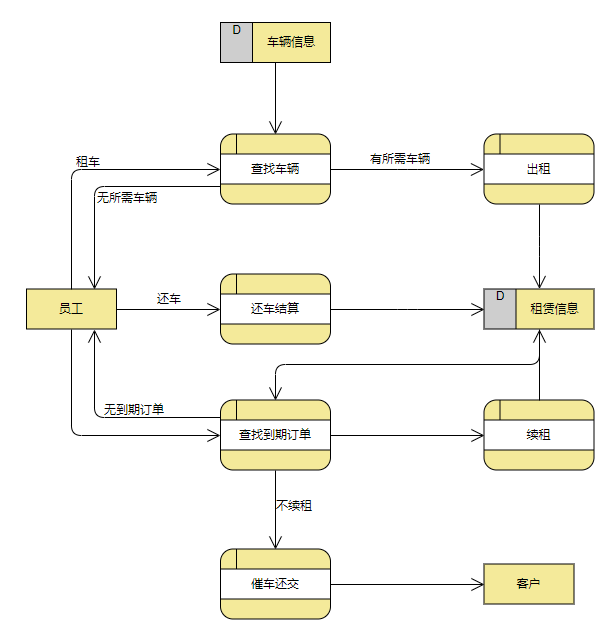
1. 系统数据流程图，如图3.2所示

**图 3.2 系统数据流程图**

****（2）预约车辆数据流程图，如图3.3所示

**图 3.3 预约车辆数据流程图**

（3）租赁业务流程图，如图3.4所示



**图 3.4租赁业务流程图**

* 1. **数据库设计**

数据库设计是整个系统设计中非常重要的一个环节。数据是所有系统的基础，如果在设计初期没有将其设计得完善、合理，在后期整个系统的开发过程，甚至到后期的功能扩展、功能变更和系统维护时会引起较多问题。

* + 1. 逻辑结构设计

本次车辆租赁管理系统的数据库各个表的实体属性如下：

客户信息：身份证号、姓名、手机号、年龄

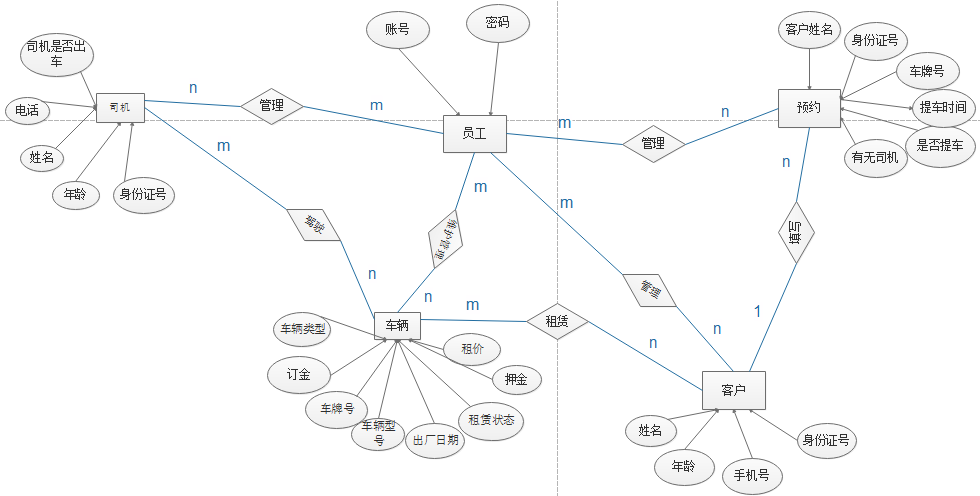
司机信息：身份证号、姓名、年龄、电话、司机是否出车

车辆信息：车牌号、车辆类型、车辆型号、出厂日期、租赁状态、租价、押金、订金

预约信息：车牌号、身份证号、租赁时长、提车时间、有无司机、是否提车

租赁信息：车牌号、身份证号、开始租赁时间、结束租赁时间、有无司机、是否还车、总租金

* + 1. ER图



**图 3.5 ER图**

* + 1. 物理结构设计

本节主要介绍SQL sever数据库的各表单信息及相关内容

表 3.6 车辆信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主要属性 | 含义说明 |
| Car\_number | Varchar(20) | 主键，不可为空 | 存储车辆牌照 |
| Car\_model | Varchar(30) | 不可为空 | 存储车辆型号 |
| Car\_date | Datetime(8) | 不可为空 | 存储车辆出产日期 |
| Car\_flag | Char(6) | 不可为空 | 租赁状态 |
| Car\_price | Int | 不可为空 | 一天租金 |
| Car\_type | Varchar(20) | 不可为空 | 车辆类型 |
| Earnest\_money | Int | 不可为空 | 订金 |
| Car\_deposit | Int | 不可为空 | 押金 |

表 3.7 司机信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主要属性 | 含义说明 |
| Driver\_id | Char(18) | 主键，不可为空 | 身份证号 |
| Driver\_name | Varchar(8) | 不可为空 | 姓名 |
| Driver\_age | Varchar(3) | 不可为空 | 年龄 |
| Driver\_phone | Varchar(11) | 唯一，不可为空 | 电话 |
| Driver\_flag | Char(6) | 不可为空 | 是否出车 |

表 3.8客户信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主要属性 | 含义说明 |
| Customer\_id | Char(18) | 主键，不可为空 | 身份证号 |
| Customer\_name | Varchar(8) | 不可为空 | 姓名 |
| Customer\_age | Vachar(3) | 不可为空 | 年龄 |
| Customer\_phone | Char(11) | 唯一，不可为空 | 手机号 |

表3.9 预约信息表

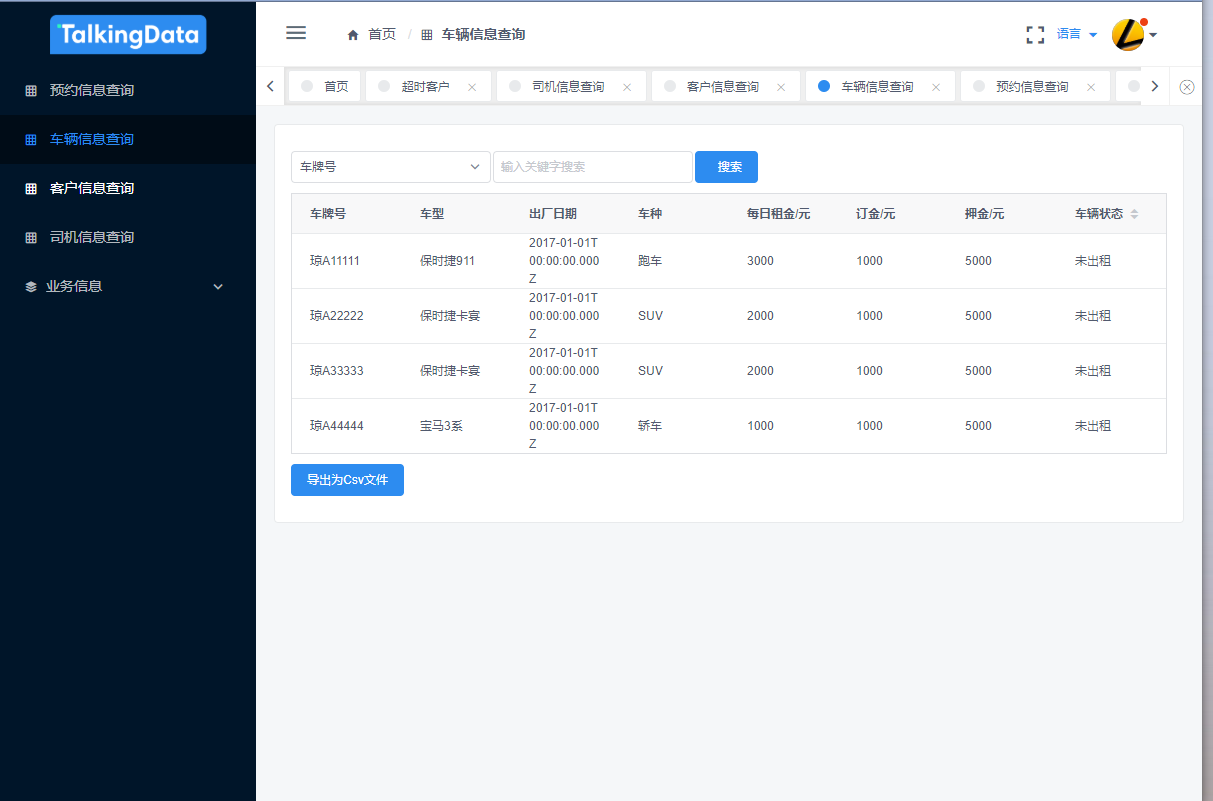
|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主要属性 | 含义说明 |
| Customer\_id | Char(18) | 主键，不可为空 | 身份证号 |
| Car\_number | Varchar(20) | 主键，不可为空 | 存储车辆牌照 |
| Pick\_time | Datetime | 不可为空 | 提车时间 |
| Lift\_car | Char(6) | 不可为空 | 是否提车 |
| Flag | Char(6) | 不可为空 | 有无司机 |

表4.0 租赁信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 主要属性 | 含义说明 |
| Customer\_id | Char(18) | 主键，不可为空 | 客户身份证号 |
| Car\_number | Varchar(20) | 主键，不可为空 | 存储车辆牌照 |
| Start\_time | Datetime | 不可为空 | 开始租赁时间 |
| End\_time | Datetime | 不可为空 | 结束租赁时间 |
| Rental | Int | 不可为空 | 总租金 |
| Return\_car | Char(6) | 不可为空 | 是否还车 |
| Flag | Char(6) | 不可为空 | 有无司机 |

1. **功能实现**

**4.1 车辆信息查询**



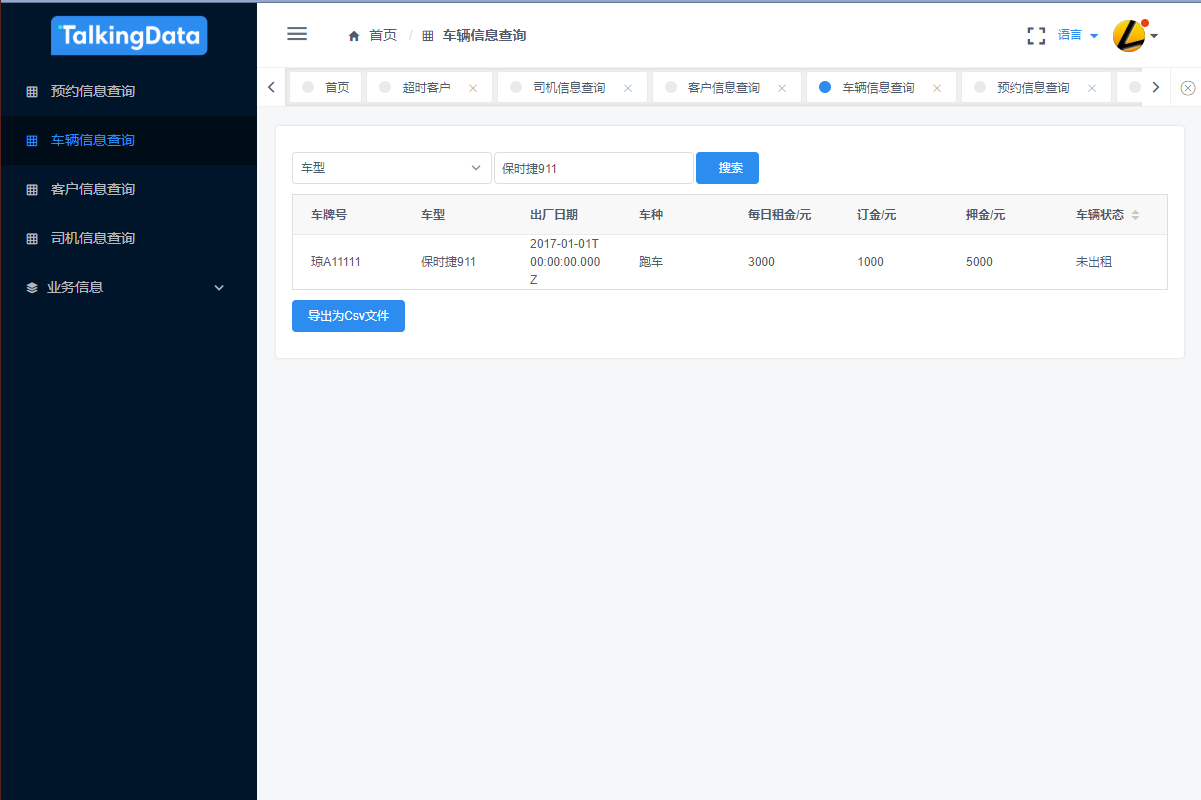
SQL语句：SELECT \* FROM CAR

Knex.js: knex. (**' CAR'**)

Koa.js API名称: **GetCarList**

请求方式：GET

* + 1. 通过车辆型号查询



SQL语句：

CREATE PROCEDURE [dbo].[PRO\_CAR\_MODEL] --通过车辆型号查询

@CAR\_MODEL VARCHAR(30)

AS

SELECT \*

FROM CAR

WHERE Car\_model=@CAR\_MODEL

GO

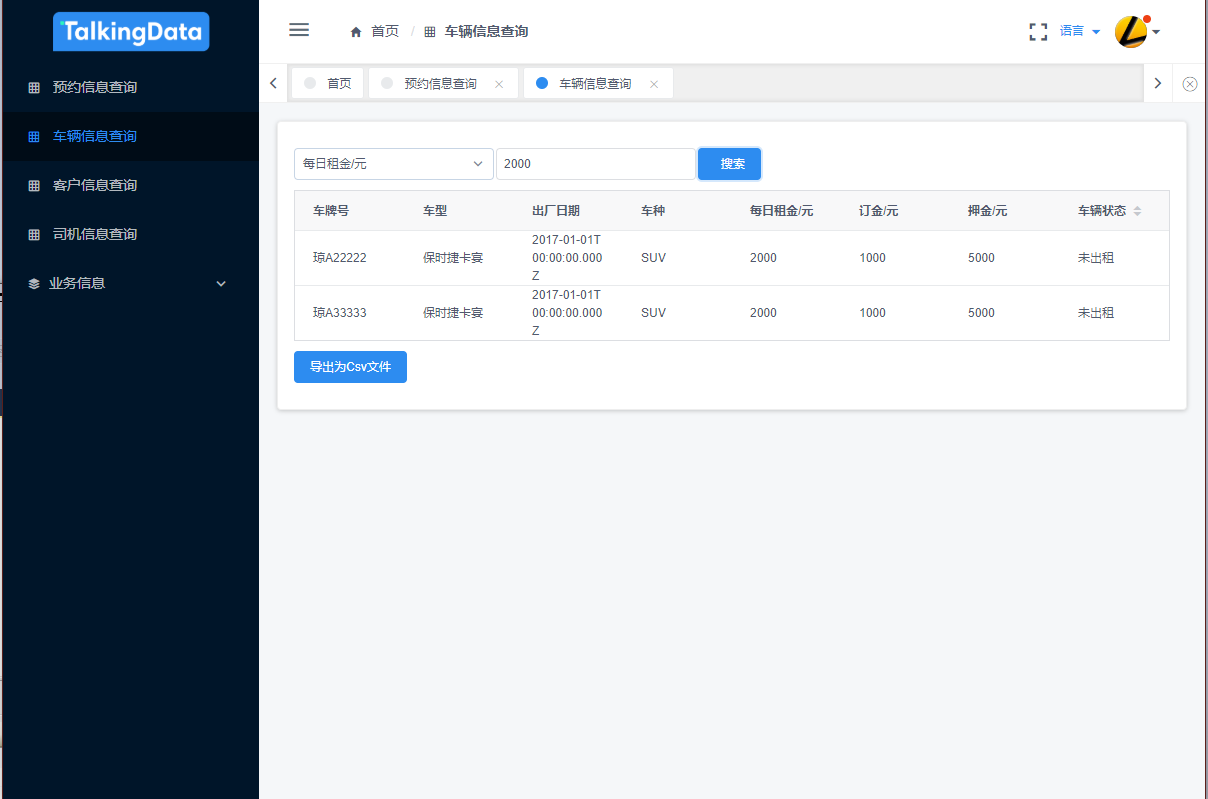
Knex.js:knex.**schema**.raw(**'EXEC PRO\_CAR\_MODEL' + 'CAR\_MODEL'**)

Koa.js API名称: GetCarListByModel

请求方式：POST

请求参数：Car\_model

4.1.2租金查询



SQL语句：

CREATE PROC [dbo].[PRO\_CAR\_PRICE] --通过租金查询

@CAR\_PRICE INT

AS

SELECT \*

FROM CAR

WHERE Car\_price<=@CAR\_PRICE

GO

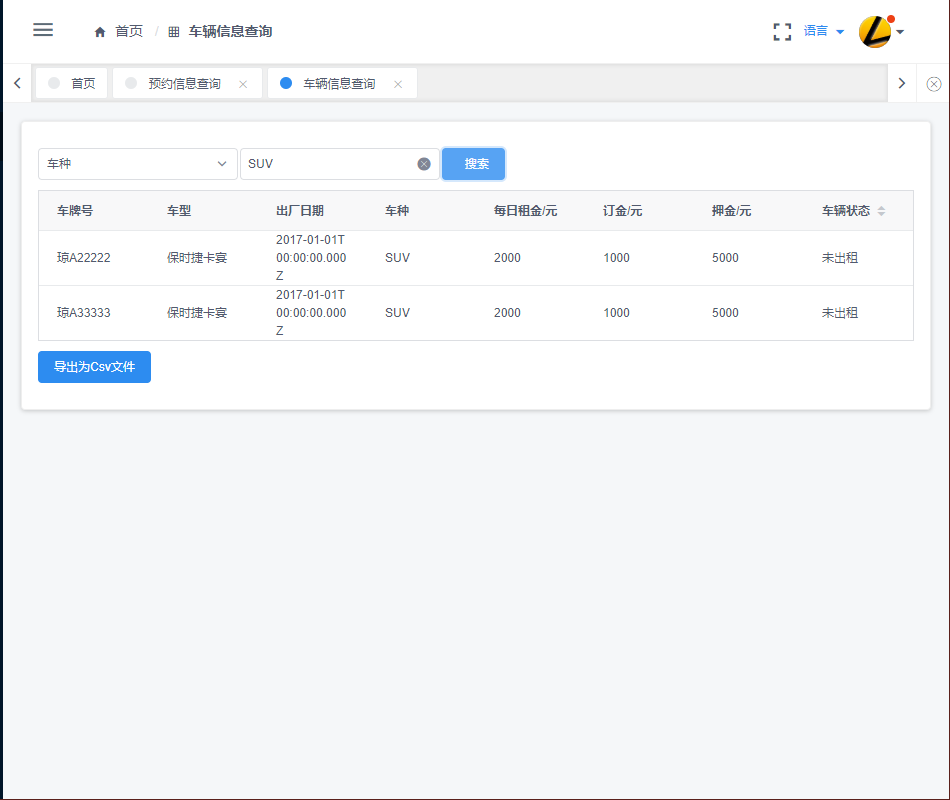
Knex.js:knex.**schema**.raw(**'EXEC PRO\_CAR\_PRICE' + 'CAR\_ PRICE '**)

Koa.js API名称: GetCarListByPrice

请求方式：POST

请求参数：Car\_price

4.1.3 通过车辆类型查询



SQL语句：

CREATE PROC [dbo].[PRO\_CAR\_TYPE] --通过车辆类型查询

@CAR\_TYPE VARCHAR(20)

AS

SELECT\*

FROM CAR

WHERE Car\_type=@CAR\_TYPE

GO

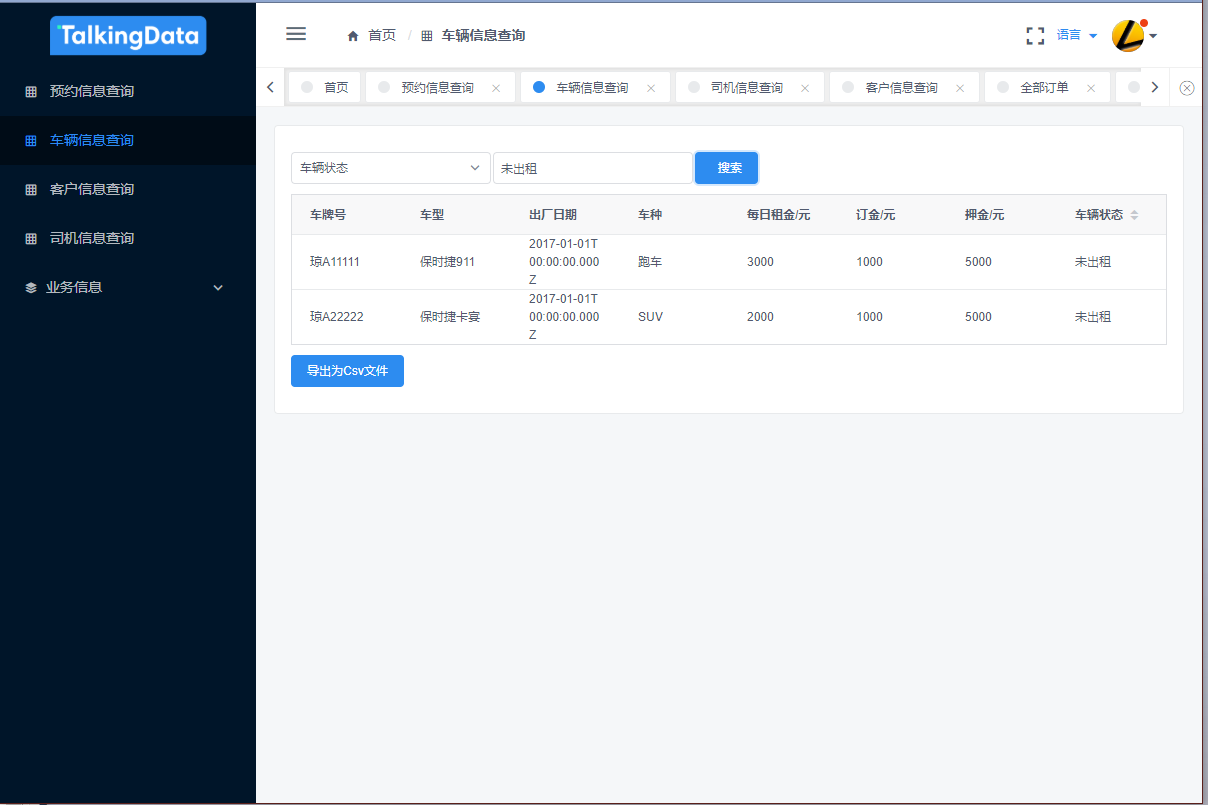
Knex.js: knex.**schema**.raw(**'EXEC PRO\_CAR\_TYPE' + 'CAR\_ TYPE '**)

Koa.js API名称: GetCarListByType

请求方式：POST

请求参数：Car\_type

* + 1. 通过租赁状态查询



SQL语句：

CREATE PROC [dbo].[PRO\_CAR\_FLAG] --通过租赁状态查询

@CAR\_FLAG CHAR(6)

AS

SELECT\*

FROM CAR

WHERE Car\_flag=@CAR\_FLAG

GO

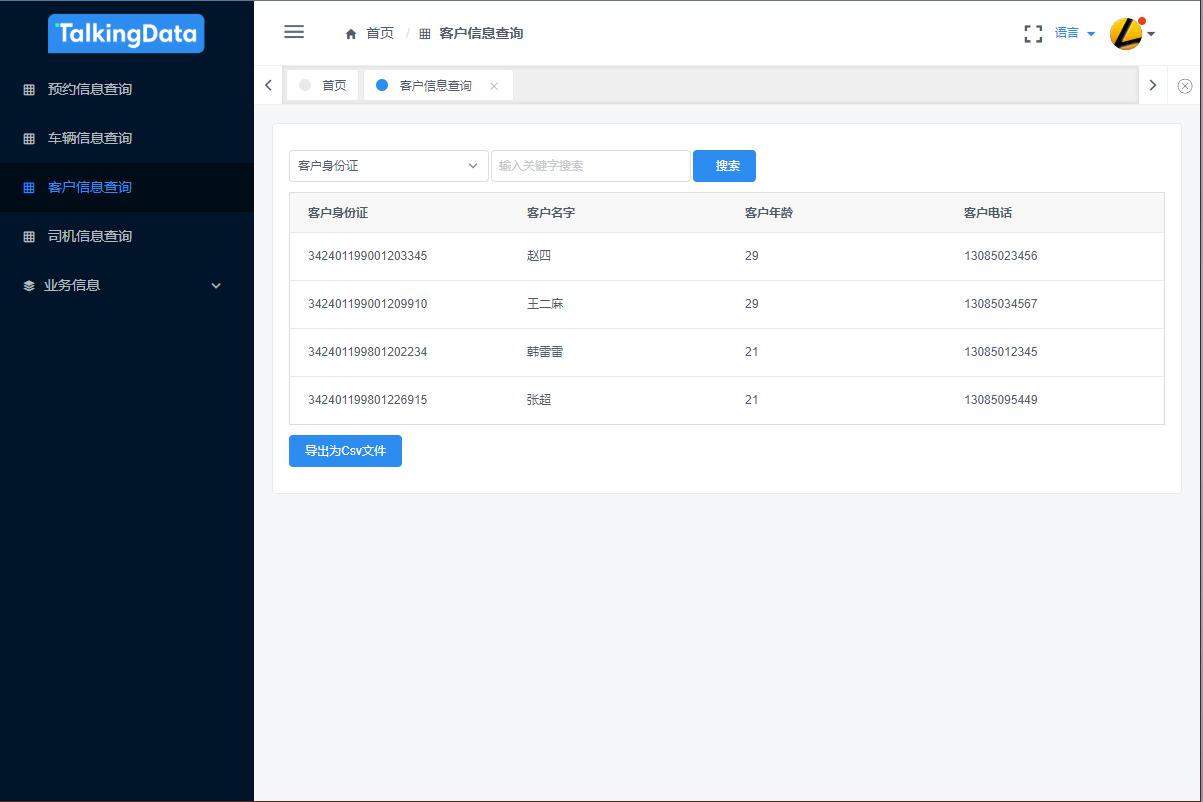
Knex.js:knex.**schema**.raw(**'EXEC PRO\_CAR\_FLAG' + 'CAR\_ FLAG '**)

Koa.js API名称: GetCarListByFlag

请求方式：POST

请求参数：Car\_flag

**4.2客户信息查询**



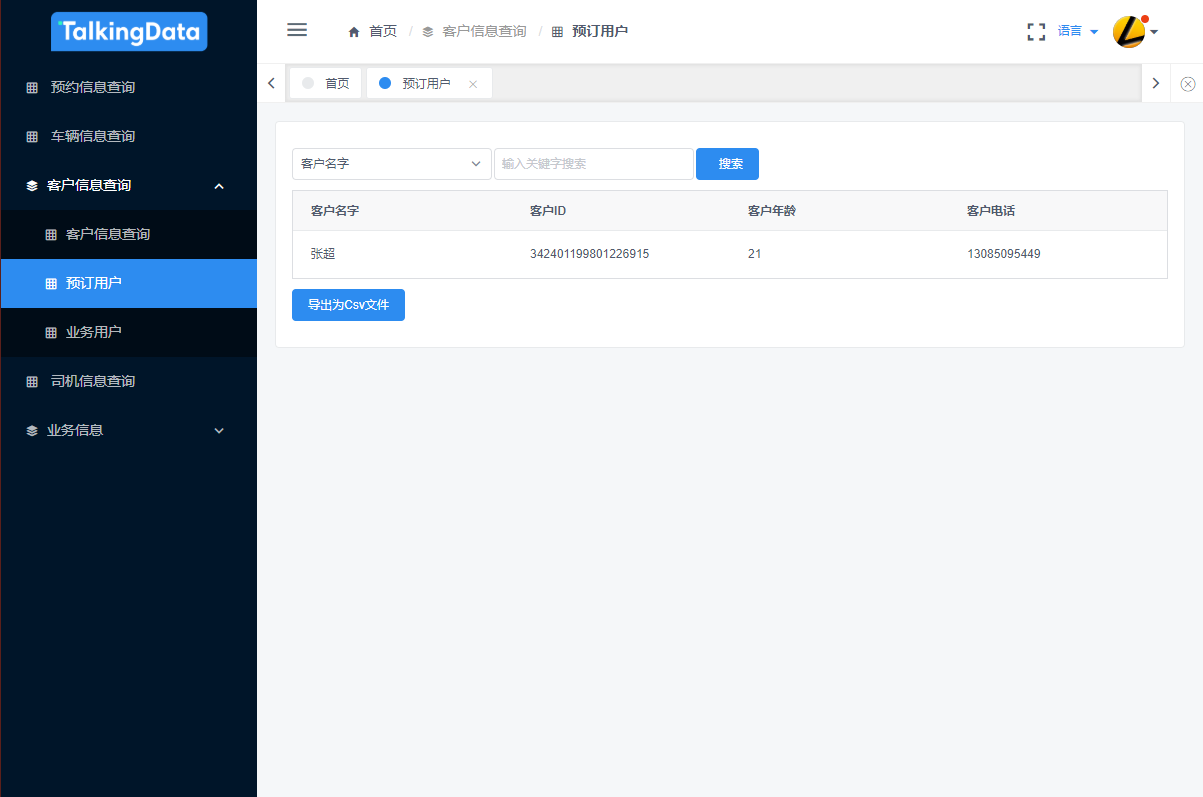
SQL语句：SELECT \* FROM CUSTOMER

Knex.js: knex. (**'CUSTOMER'**)

Koa.js API名称: **GetCustomerList**

请求方式：GET

4.2.1查询已预约的客户



SQL语句：

CREATE PROC [dbo].[PRO\_CUSTOMER\_BOOK] --查询已预约的客户

AS

SELECT CUSTOMER.Customer\_name,CUSTOMER.Customer\_id,

CUSTOMER.Customer\_age,CUSTOMER.Customer\_phone

FROM CUSTOMER,BOOK

WHERE CUSTOMER.Customer\_id=BOOK.Customer\_id AND BOOK.Lift\_car='未提车'

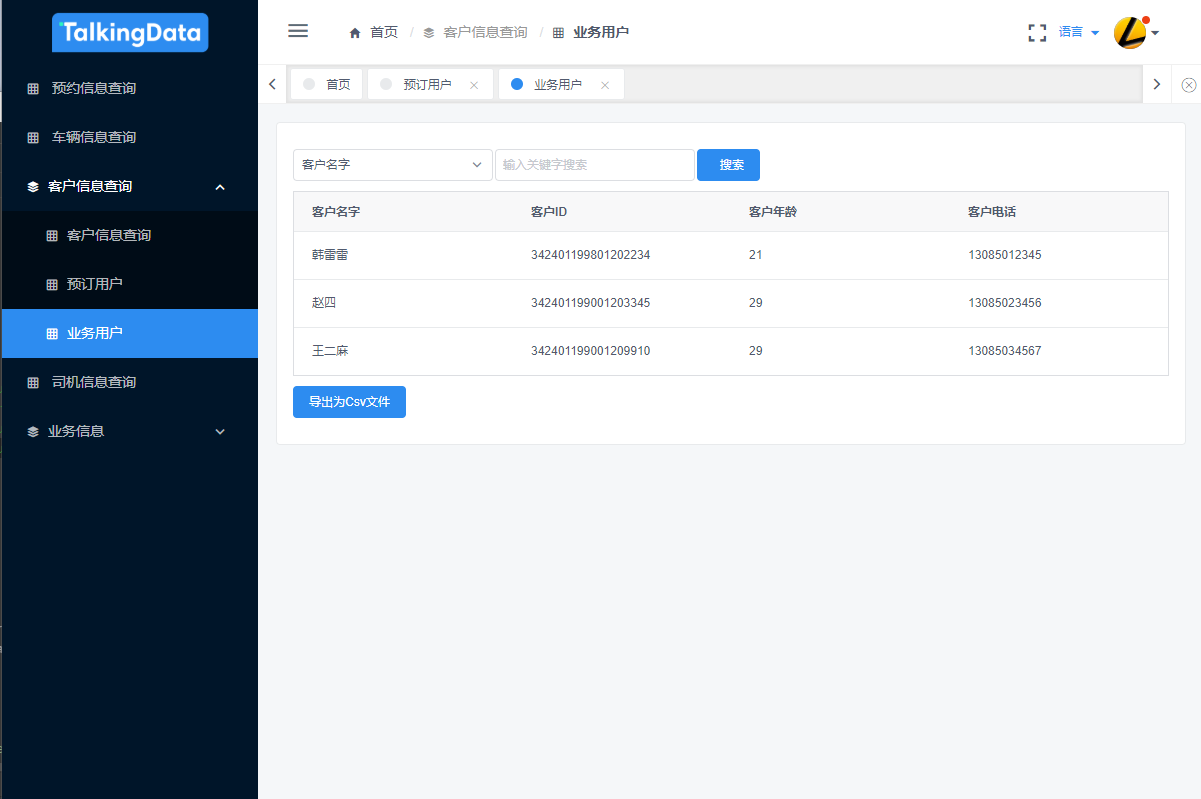
GO

Knex.js:knex.**schema**.raw(**'EXEC PRO\_CUSTOMER\_BOOK'** )

Koa.js API名称: GetBookedCustomerList

请求方式：POST

4.2.2查询租赁车辆的客户



SQL语句：

CREATE PROC [dbo].[PRO\_CUSTOMER\_LEASE] --查询租赁车辆的客户

AS

SELECT CUSTOMER.Customer\_name,CUSTOMER.Customer\_id,

CUSTOMER.Customer\_age,CUSTOMER.Customer\_phone

FROM CUSTOMER,LEASE

WHERE CUSTOMER.Customer\_id=LEASE.Customer\_id

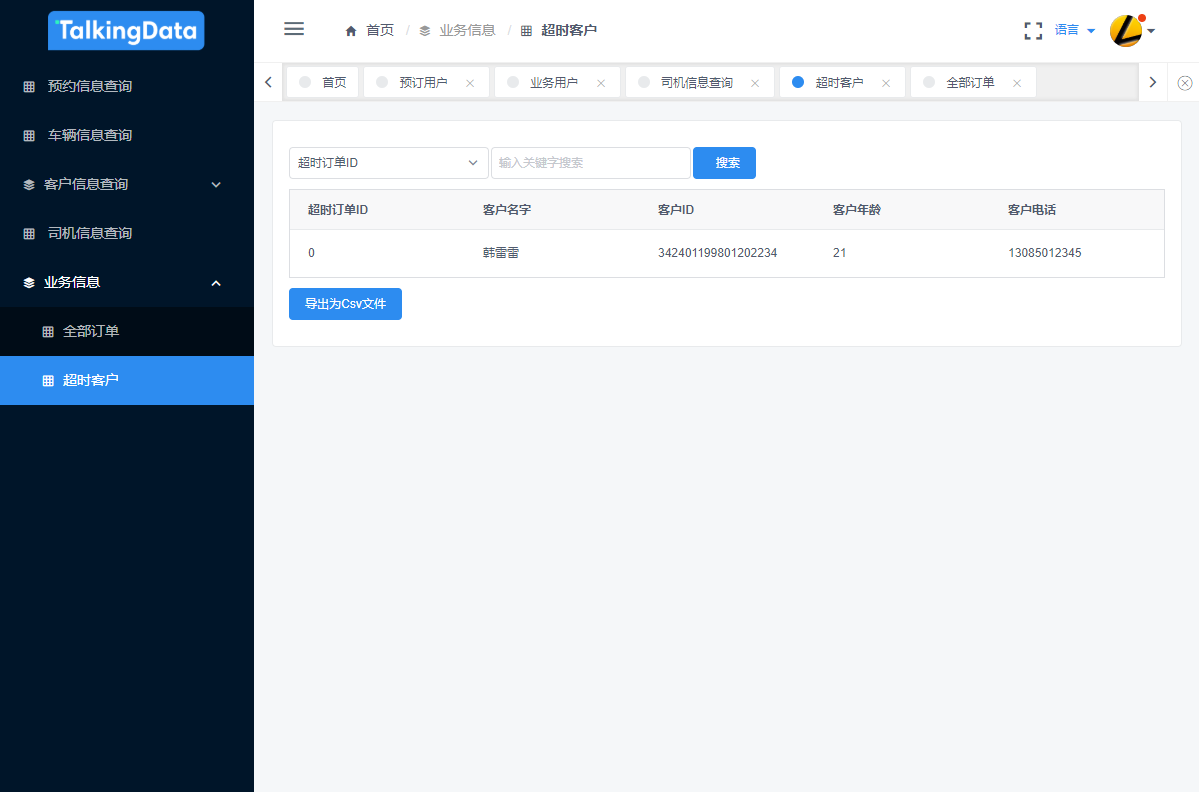
GO

Knex.js:knex.**schema**.raw(**'EXEC PRO\_CUSTOMER\_LEASE '** )

Koa.js API名称: GetLeasedCustomerList

请求方式：POST

4.2.3查询租赁时间到期但并未还车的客户，进行催车还交



SQL语句：

CREATE PROC [dbo].[PRO\_CUSTOMER\_DELAY] --查询租赁时间到期但并未还车的客户，进行催车还交

@CUSTOMER\_DELAY DATETIME

AS

SELECT LEASE.Lease\_id,CUSTOMER.Customer\_name,CUSTOMER.Customer\_id,

CUSTOMER.Customer\_age,CUSTOMER.Customer\_phone

FROM CUSTOMER,LEASE

WHERE CUSTOMER.Customer\_id=LEASE.Customer\_id AND

LEASE.Return\_car='未还车'AND LEASE.End\_time<@CUSTOMER\_DELAY

GO

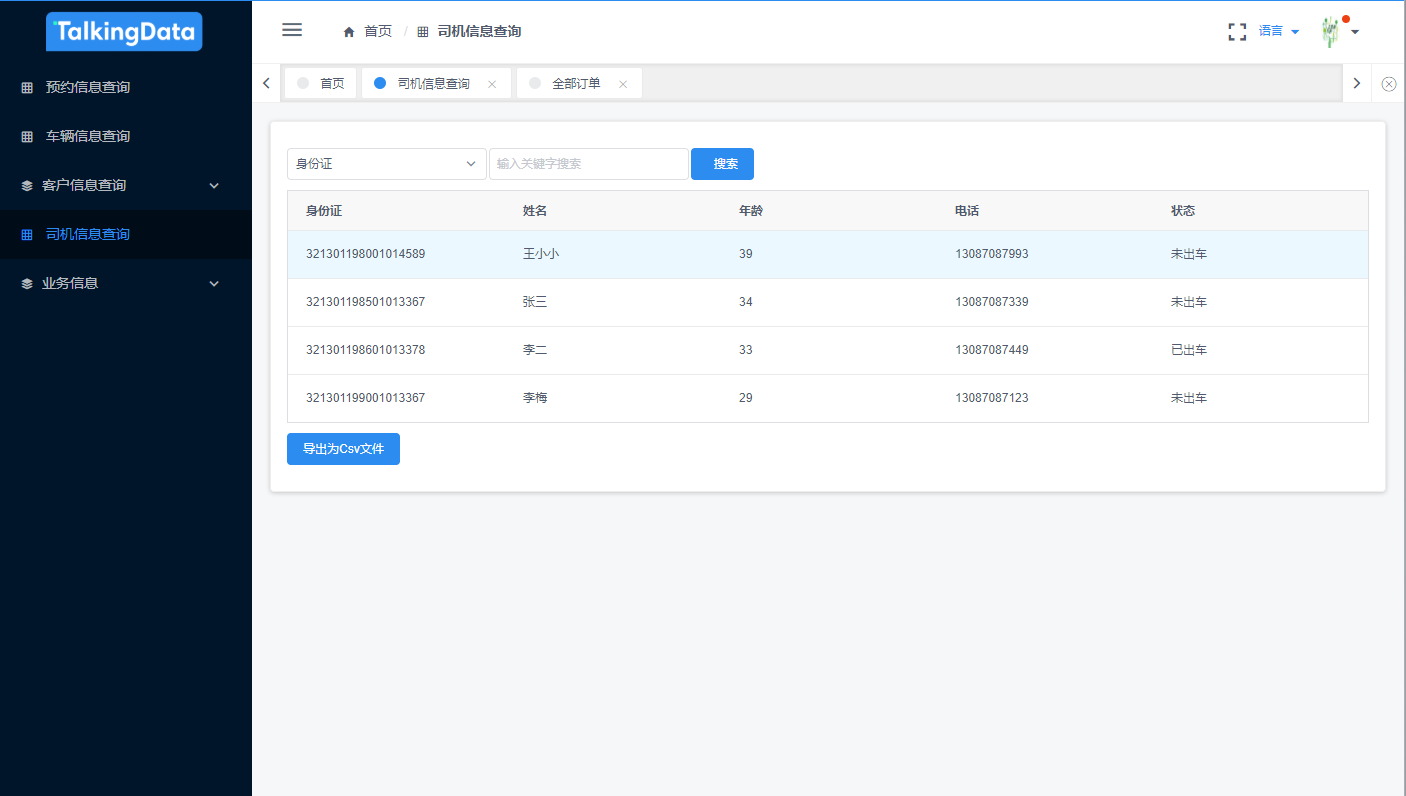
Knex.js:knex.**schema**.raw(**'EXEC PRO\_CUSTOMER\_DELAY' +'DATETIME'** )

Koa.js API名称: GetLeaselist

请求方式：POST

请求参数：DATETIME

**4.3司机信息查询**



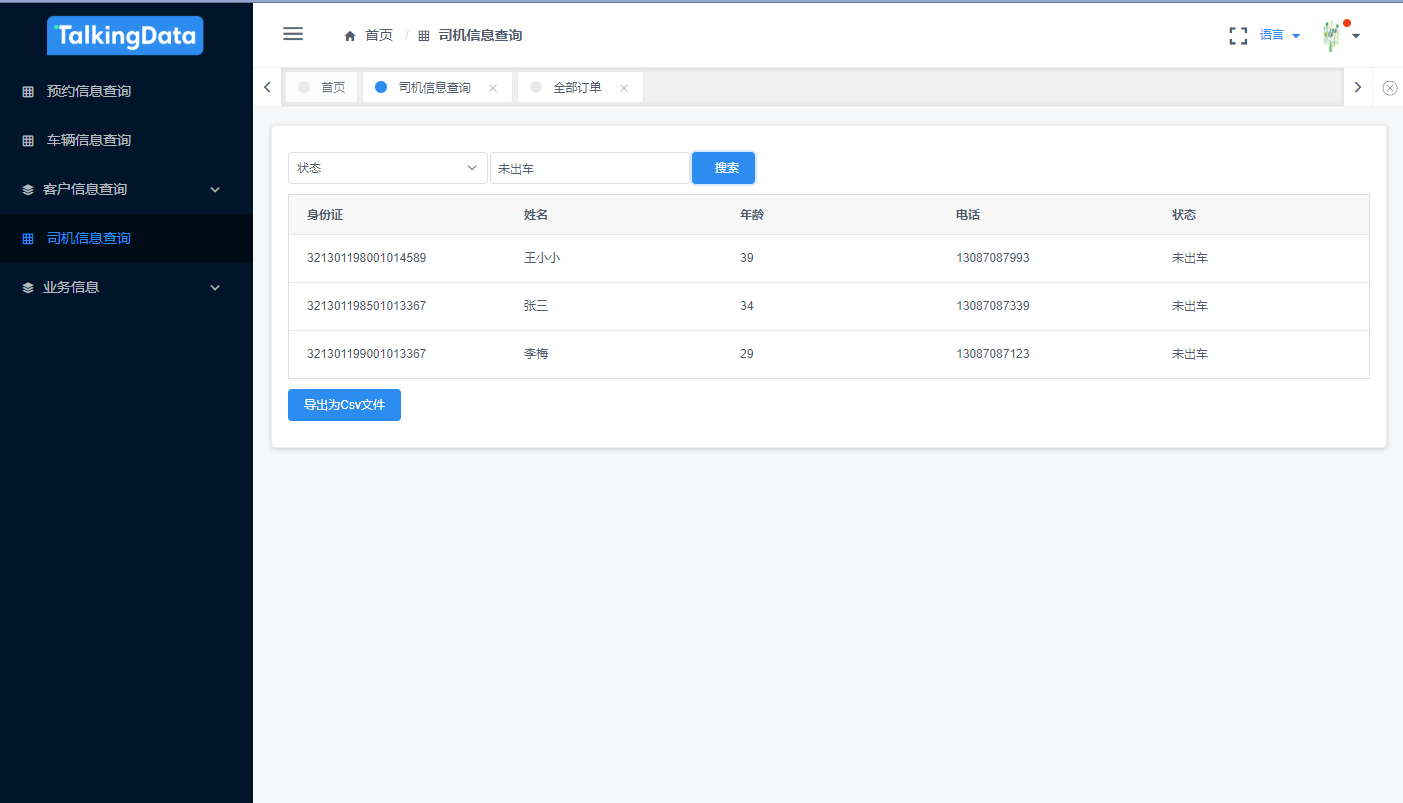
SQL语句：SELECT \* FROM BOOK

Knex.js: knex. (**'BOOK'**)

Koa.js API名称: **GetBookList**

请求方式：GET

4.3.1通过出车状态查询司机

SQL语句：

CREATE PROC [dbo].[PRO\_DRIVER\_FLAG] --通过出车状态查询司机

@DRIVER\_FLAG CHAR(6)

AS

SELECT \*

FROM DRIVER

WHERE DRIVER.Driver\_flag=@DRIVER\_FLAG

GO

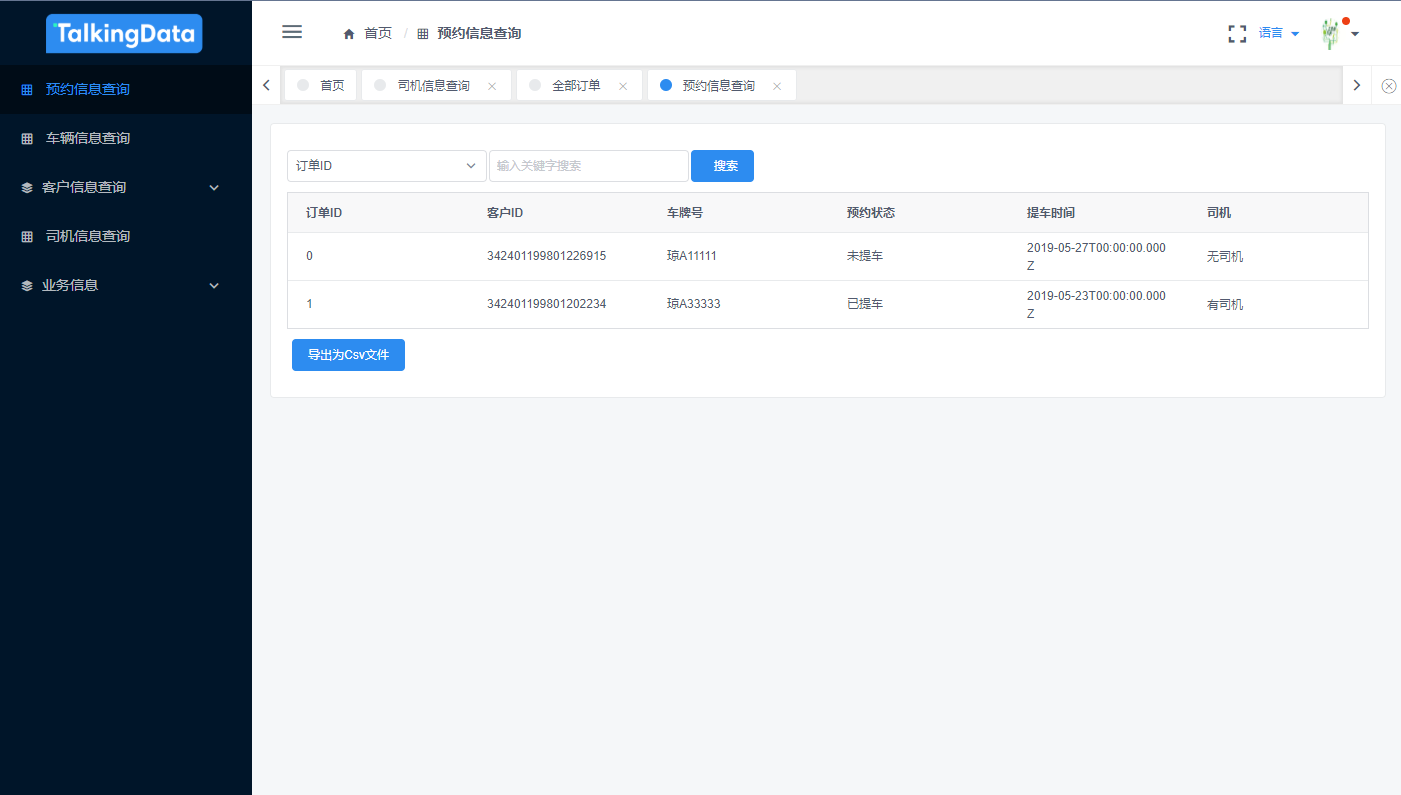
Knex.js:knex.**schema**.raw(**'EXEC PRO\_DRIVER\_FLAG '** )

Koa.js API名称: GetDriverListByFlag

请求方式：POST

**4.4业务管理**

4.4.1 租赁信息查询



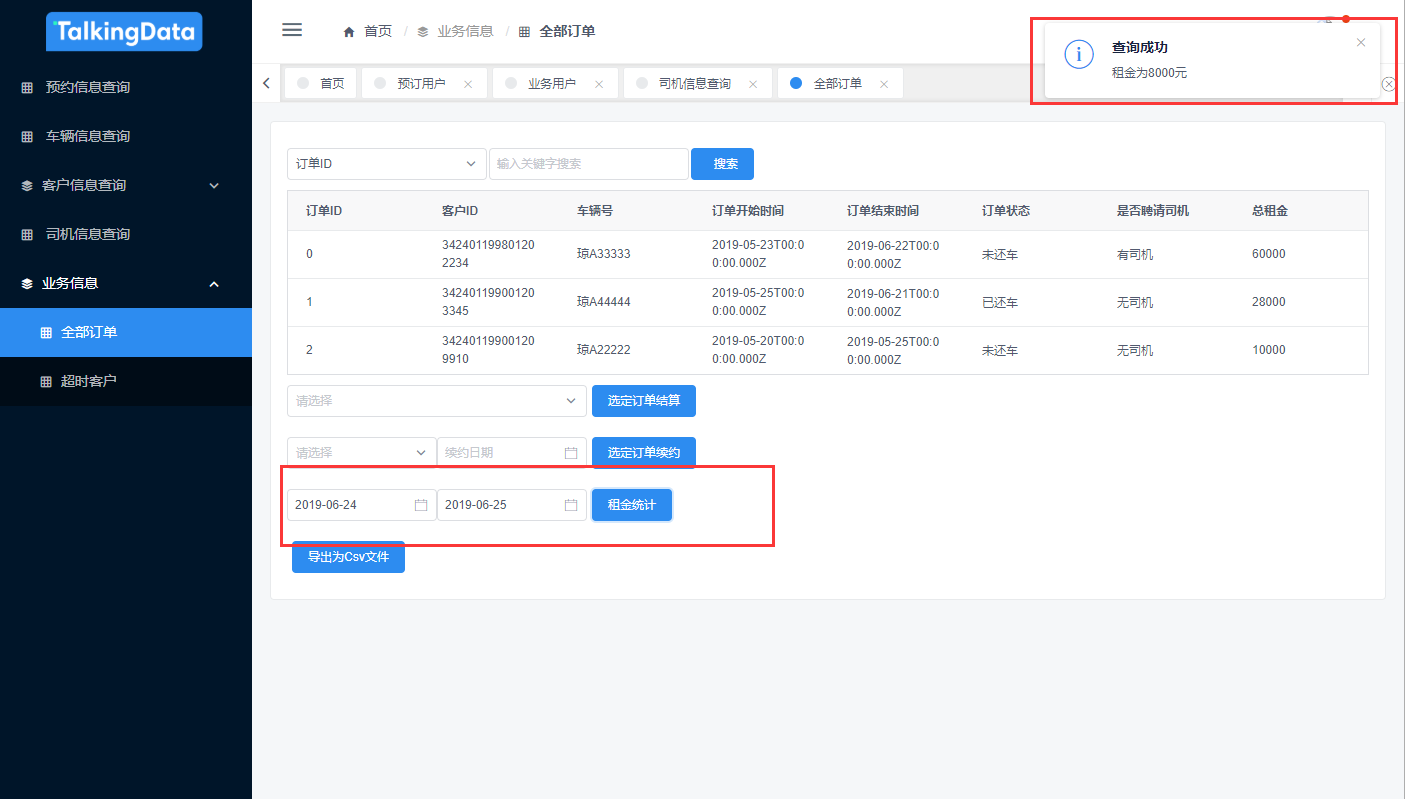
SQL语句：SELECT \* FROM LEASE

Knex.js: knex. (**'LEASE'**)

Koa.js API名称: **GetLeaseList**

请求方式：GET

4.4.2 对某天所得租金进行统计



SQL语句：

CREATE PROC [dbo].[PRO\_LEASE\_RENTAL] --对某天所得租金进行统计

@LEASE\_RENTAL DATETIME

AS

SELECT SUM(CAR.Car\_price)

FROM CAR,LEASE

WHERE CAR.Car\_number=LEASE.Car\_number AND

(End\_time>@LEASE\_RENTAL OR Return\_car='未还车')

GO

Knex.js:

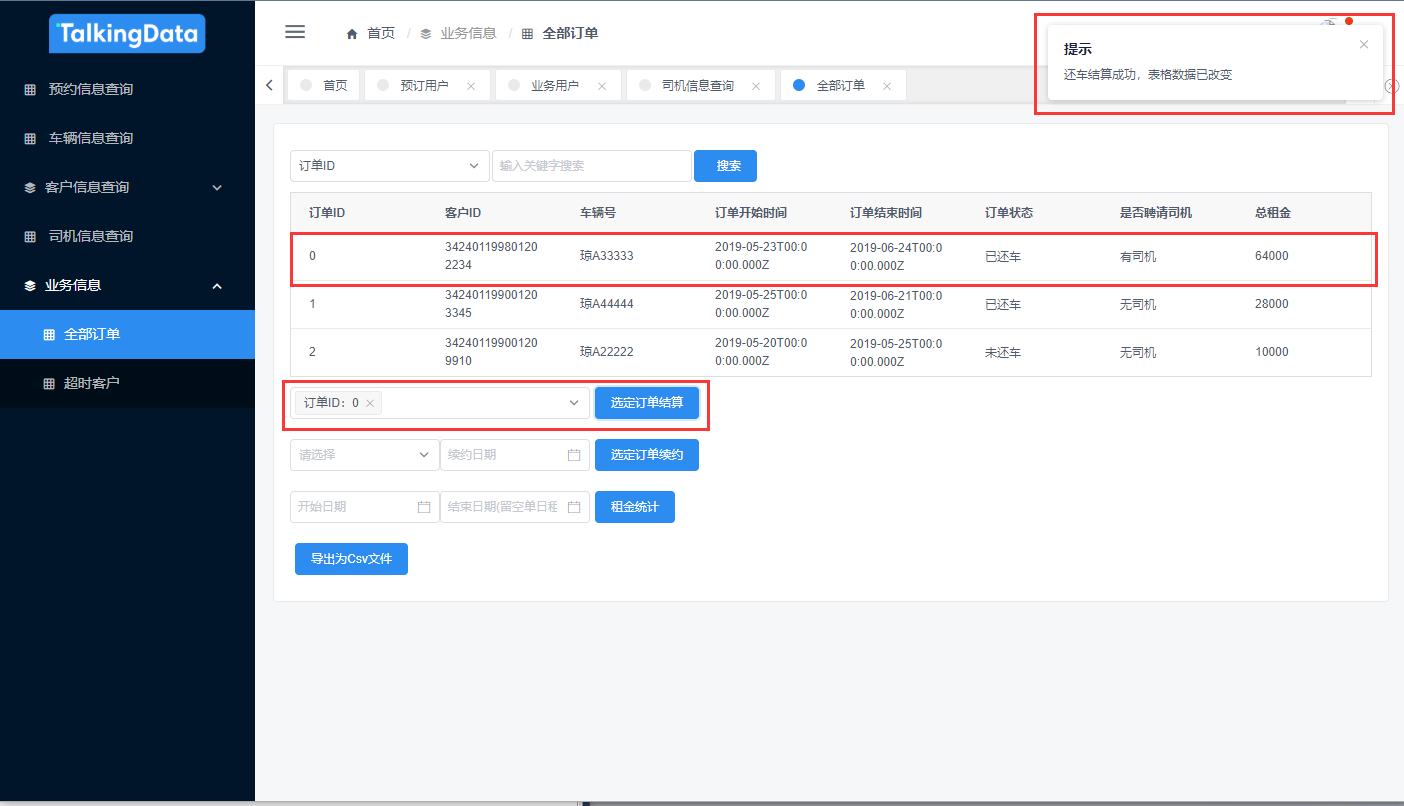
knex.**schema**.raw(**'PRO\_LEASE\_RENTAL'+ 'DATETIME'** )

Koa.js API名称: LeaseRental

请求方式：POST

请求参数：DATETIME

4.4.3 还车结算



SQL语句：

CREATE PROC [dbo].[PRO\_LEASE\_END] --还车结算

@LEASE\_END\_ID INT, --需还车的订单号

@LEASE\_END\_DATE DATETIME --还车的日期

AS

UPDATE LEASE --对还车的租赁表进行修改

SET Return\_car='已还车',End\_time=@LEASE\_END\_DATE,

Rental=(

SELECT DATEDIFF(DAY,LEASE.Start\_time,@LEASE\_END\_DATE)\*CAR.Car\_price --计算订单总租金

FROM LEASE,CAR WHERE LEASE.Lease\_id=@LEASE\_END\_ID AND LEASE.Car\_number=CAR.Car\_number

)

WHERE LEASE.Lease\_id=@LEASE\_END\_ID

UPDATE CAR --对所还车辆的出租状态进行修改

SET Car\_flag='未出租' FROM CAR,LEASE WHERE LEASE.Lease\_id=@LEASE\_END\_ID AND LEASE.Car\_number=CAR.Car\_number

SELECT \* FROM LEASE

GO

Knex.js:

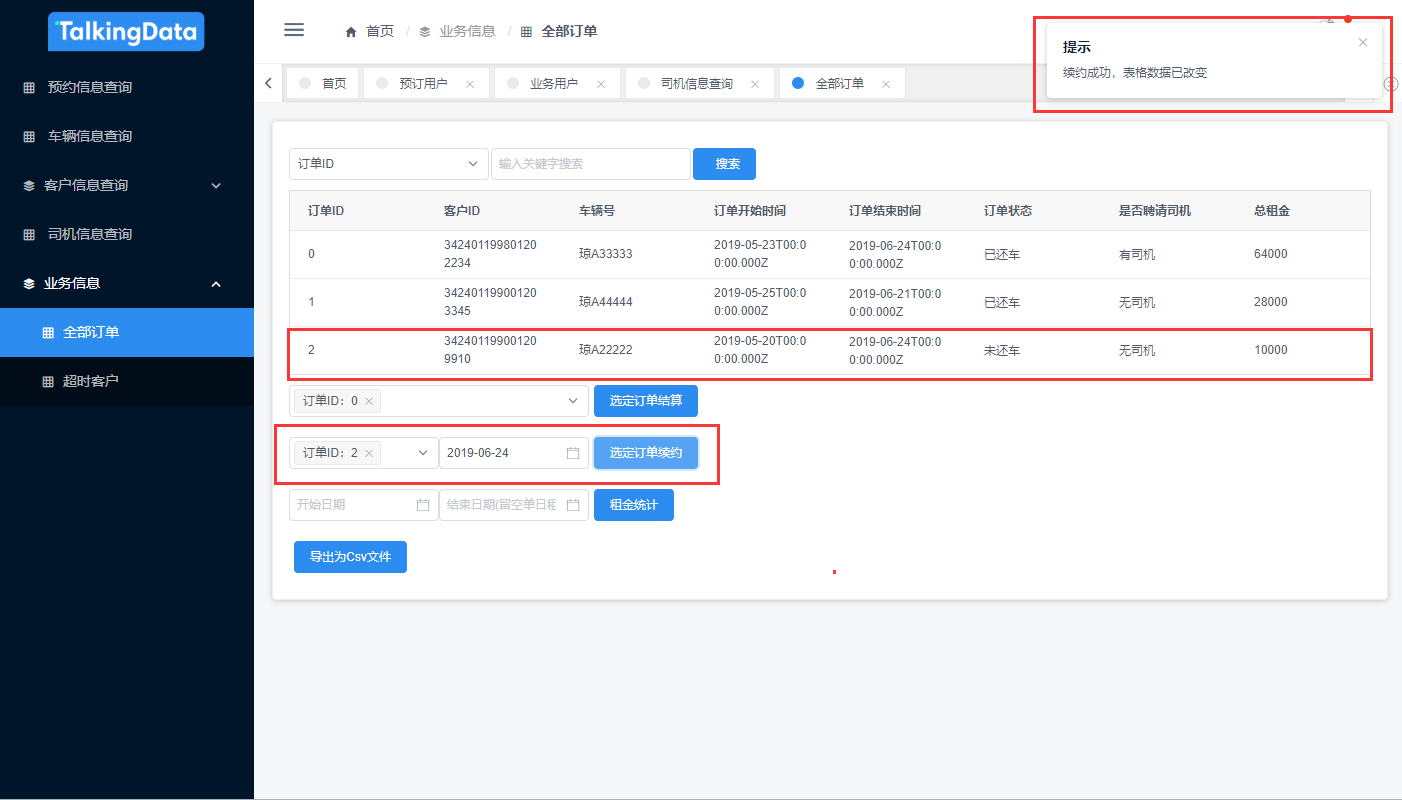
knex.**schema**.raw(**'** **PRO\_LEASE\_END '+ 'DATETIME'** )

Koa.js API名称: EndLeaseCalc

请求方式：POST

请求参数：DATETIME

4.4.4 续约



SQL语句：

CREATE PROC [dbo].[PRO\_LEASE\_RENEW] --续约

@LEASE\_RENEW\_ID INT,

@LEASE\_RENEW\_DATE DATETIME

AS

UPDATE LEASE

SET LEASE.End\_time=@LEASE\_RENEW\_DATE

WHERE LEASE.Lease\_id=@LEASE\_RENEW\_ID

SELECT \* FROM LEASE

GO

Knex.js:

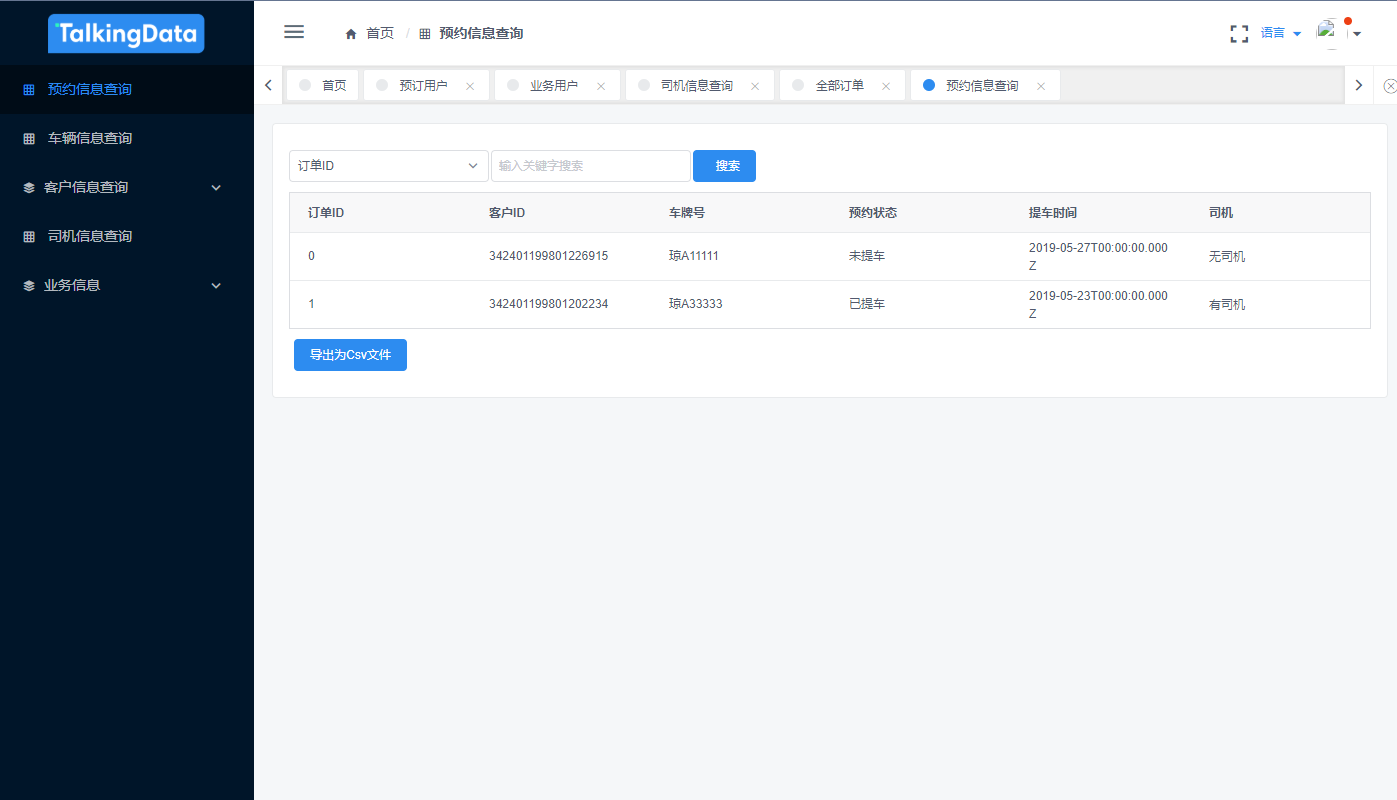
knex.**schema**.raw(**'** **PRO\_LEASE\_RENEW '+'Lease\_id' +'DATETIME'** )

Koa.js API名称: EndLeaseCalc

请求方式：POST

请求参数：DATETIME,Lease\_id

4.5预约信息查询



SQL语句：SELECT \* FROM BOOK

Knex.js: knex. (**'BOOK'**)

Koa.js API名称: **GetBookList**

请求方式：GET

**5 目录结构、部署方法与核心代码**

**5.1服务端**

**5.1.1服务端主要文件目录结构**

└── server //根文件

├── controllers API

├── token token管理

├── config 数据库配置信息

├── controller API路由管理

└── app.js Koa程序入口

**5.1.2服务端部署方法**

**1、修改根目录下config.js与app.js中的数据库信息和端口信息；**

**2、打开管理员模式下的Terminal，进入client目录，输入npm install补全项目组件；**

**3、运行命令node app.js即可启动服务端**

**5.1.3 核心代码**

**const** Koa = require(**'koa'**);  
**const** bodyParser = require(**'koa-bodyparser'**);  
**const** controller = require(**'./controller'**);  
**const** app = **new** Koa();  
**const** cors = require(**'koa2-cors'**);  
  
app.use(cors({  
 origin: **function** (ctx) {  
 **if** (ctx.url === **'/API/v1/Login'**) {  
 **return "\*"**;  
 }  
 **return '\*'**;  
 },  
 exposeHeaders: [**'WWW-Authenticate'**, **'Server-Authorization'**],  
 maxAge: 5,  
 credentials: **true**,  
 allowMethods: [**'GET'**, **'POST'**, **'DELETE'**],  
 allowHeaders: [**'Content-Type'**, **'Authorization'**, **'Accept'**],  
}));  
app.use(async (ctx, next) => {  
 console.log(**`Process** ${ctx.request.method} ${ctx.request.url}**...`**);  
 await next();  
});  
app.use(bodyParser());  
*// add controllers:*app.use(controller());  
app.listen(3000);  
console.log(**'app started at port 3000...'**);

**5.2客户端**

**5.2.1客户端主要文件目录结构**

└── client //根文件

├── config 开发相关配置

├── public 打包所需静态资源

└── src

├── api AJAX请求

└── assets 项目静态资源

├── icons 自定义图标资源

└── images 图片资源

├── components 业务组件

├── config 项目运行配置

├── directive 自定义指令

├── libs 封装工具函数

├── locale 多语言文件

├── router 路由配置

├── store Vuex配置

├── view 页面文件

└── tests 测试相关

**5.2.2客户端部署方法**

**1、修改client/src/config/index.js中的baseUrl为server地址；**

**1、安装Node.js最新的LTS版本，同时安装NPM（Node.js Package Manager）；  
 2、打开管理员模式下的Terminal，进入client目录，输入npm install补全项目组件；**

**3、安装结束后在client目录下使用命令npm run dev即可进入开发模式，使用命令npm run build 即可部署服务模式**

**4、使用3中任意一条命令并成功执行后，在浏览器输入localhost:端口号 即可成功访问。端口号会在执行命令后在Terminal中显示；**

**5、网站有super\_admin与admin两个默认账户，密码均为111111**

**5.2.3核心代码**

**import** Vue from **'vue'  
import** App from **'./App'  
import** router from **'./router'  
import** store from **'./store'  
import** iView from **'iview'  
import** i18n from **'@/locale'  
import** config from **'@/config'  
import** importDirective from **'@/directive'  
import** { directive as clickOutside } from **'v-click-outside-x'  
import** installPlugin from **'@/plugin'  
import './index.less'  
import '@/assets/icons/iconfont.css'  
import** TreeTable from **'tree-table-vue'  
import** VOrgTree from **'v-org-tree'  
import 'v-org-tree/dist/v-org-tree.css'***// 实际打包时应该不引入mock  
/\* eslint-disable \*/***if** (process.env.NODE\_ENV !== **'production'**) require(**'@/mock'**)  
  
Vue.use(iView, {  
 i18n: (key, value) => i18n.t(key, value)  
})  
Vue.use(TreeTable)  
Vue.use(VOrgTree)installPlugin(Vue)Vue.config.productionTip = **false**Vue.prototype.$config = configimportDirective(Vue)  
Vue.directive(**'clickOutside'**, clickOutside)**new** Vue({  
 el: **'#app'**,  
 router,  
 i18n,  
 store,  
 render: h => h(App)  
})

[1]koa.js官网文档[EB/OL]，2019-06-16 [2019-06-22]. <https://koa.bootcss.com/>

[2] Knex.js - A SQL Query Builder for Javascript [EB/OL]，2018-05-12 [2019-06-22]. <https://knexjs.org/>

[3] iView - A high quality UI Toolkit based on Vue.js [EB/OL]，2019-04-23 [2019-06-22]. <https://www.iviewui.com/>

[4] Vue.js [EB/OL]，2019-04-21 [2019-06-22]. <https://cn.vuejs.org/>