城市共享停车管理系统

体系结构文档

1.0

2018/12/16

白烨淞

软件工程导论

2018 秋季

修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **描述** | **作者** | **备注** |
| 2018/12/16 | 版本1.0 | 白烨淞 | 介绍与总体概述 |

文档批准

以下需求分析文档已经被以下机构人员批准并认可:

|  |  |  |  |
| --- | --- | --- | --- |
| **签名** | **打印签名** | **标题** | **日期** |
|  | 白烨淞 | 城市共享停车管理系统 | 2018/12/16 |
|  |  |  |  |
|  |  |  |  |

**目录**

[1. 项目相关信息 5](#_Toc532758285)

[1.1概述 5](#_Toc532758286)

[1.2 范围 5](#_Toc532758287)

[1.3主要功能 5](#_Toc532758288)

[1.4 术语与缩写定义 6](#_Toc532758289)

[1.5 引用 7](#_Toc532758290)

[2. 体系结构需求 7](#_Toc532758291)

[2.1 关键指标 7](#_Toc532758292)

[2.2 用户特征 7](#_Toc532758293)

[2.3体系结构用例 7](#_Toc532758294)

[2.3.2用户角色1-车主 8](#_Toc532758295)

[2.3.3用户角色2-停车系统管理员 9](#_Toc532758296)

[2.3.4用户角色3-车位实时信息提供者 9](#_Toc532758297)

[2.3.5用户角色5-档案 9](#_Toc532758298)

[2.4 各相关方对体系结构的要求 10](#_Toc532758299)

[2.6 非功能性需求 11](#_Toc532758300)

[2.6.1质量需求 11](#_Toc532758301)

[(1) 性能 11](#_Toc532758302)

[(4)密安性 12](#_Toc532758303)

[(6)可移植性 12](#_Toc532758304)

[2.6.2 工程需求 12](#_Toc532758305)

[(1) 逆向需求 12](#_Toc532758306)

[(3) 逻辑数据库需求 13](#_Toc532758307)

[3. 解决方案 14](#_Toc532758308)

[3.1 相关的体系结构模式 14](#_Toc532758309)

[3.1.1 共享数据仓库模式 14](#_Toc532758310)

[3.1.2 分层的C/S模式 14](#_Toc532758311)

[3.2 体系结构概述 15](#_Toc532758312)

[3.3 结构化视图 16](#_Toc532758313)

[3.3.1 概念级体系结构视图 16](#_Toc532758314)

[3.3.2模块级体系结构视图 16](#_Toc532758315)

[3.3.4运行级体系结构 18](#_Toc532758316)

[3.4 实现问题 18](#_Toc532758317)

[4. 系统的质量分析与评价 19](#_Toc532758318)

[4.1 场景分析 19](#_Toc532758319)

[4.1.1 用例场景 19](#_Toc532758320)

[4.1.2 增长性场景 19](#_Toc532758321)

[4.1.3 探索性场景 19](#_Toc532758322)

[4.2 原型分析 19](#_Toc532758323)

[4.3 风险 20](#_Toc532758324)

# 项目相关信息

## 1.1概述

城市共享停车管理系统是为了方便城市空闲车位的管理而开发的基于Web的数据库应用系统。目的是解决用户停车难、城市车位空忙不均等情况，同时推动单位大院、居民区等开放空车位的工作。

系统的用户分为三类：**车主**、**车位实时信息提供方**和**停车系统管理员**。该项目由城市交通局统一管理，车主、车位实时信息提供方和停车系统管理员可以登录系统进行对应操作，停车系统管理员负责管理使用权限。为了保证系统安全性，系统对车主、车位实时信息提供方和停车系统管理员三种用户角色设定了不同的权限，停车系统管理员能够管理车位实时信息提供方和车主的权限和账户信息，并且具有最高权限。

基本的业务流程是：车位实时信息提供者借助信息手段将所管辖区域内的空闲车位数量及详细地址信息上传至管理系统系统，并定期更新数据。停车系统管理者对信息提供者进行审批并进行认定，认定后的信息才可以发布到管理系统中，之后完成认证的车主可以通过移动端或PC端查询目的地附近的空闲车位情况，在完成空闲车位的预订工作后开始计费，管理系统将自动接入导航软件快速引导车辆停放。

该系统即可以完全独立，也可以作为交通管理系统的子系统。

## 1.2 范围

(1) 名称：城市共享停车管理系统（*Urban Shared Parking Management System*）

(2) 本系统将主要适用于普通城市的车位共享及管理，主要完成收集城市合法空闲车位信息、空闲车位查询、空闲车位预定、权限管理等业务，可作为城市交通信息管理系统的子模块。

有关车辆导航，驾驶人识别认证以及机动车认证等其他功能均属于其父系统城市交通管理系统的范畴，并不在本系统的功能涵盖范围之内。

## 1.3主要功能

城市共享停车管理系统可以实施整合城市空闲车位的相关信息，减去了人工核对造成的负担以及人为错误造成的损失，信息存储大管理方便。各角色可支持的功能有：

1. 车主
2. 通过登录手机APP或是PC程序，查询目的地址周边空闲车位情况
3. 通过系统进行空闲车位的预定，之后自动接入导航系统
4. 车位实时信息提供者
5. 每隔1天更新所管辖区域的车位的数量信息。
6. 查看所在地区的车位空闲数量信息。
7. 临时设置某车位的状态，包括空闲、占用、损坏和维修四种状态。
8. 停车系统管理员
9. 查看全部区域或者是制定某区域的车位实时信息，包括空闲车位数量、车位总数、占用比例、异常车位状态。
10. 审核某区域的可用车位情况。在车位实时信息提供者把车位信息录入完提交审核后，停车系统管理员负责审核信息真实性，如果发现无误则给予通过；如果发现成绩有误，则给相应车位实时信息提供者发送邮件提示更改信息重新上传。
11. 发布车位实时信息。在认定车位信息正确无误后，管理员可正式发布车位信息
12. 归档记录。将相应的车位信息、用户信息、错误记录进行整理归档。
13. 系统管理员
14. 管理账户信息。管理员可以对所有类型的用户进行权限的授予撤回以及账户的增删。

系统需要实现的功能有：

1. 车主在注册成功后之后，可通过手机APP或者PC端定位车辆实时所在地点，此外，车主还可以设置目的地，系统将自动查询目的地附近的空闲车位的可用车位比例及具体分布。
2. 系统在读取目的地信息之后可以接入导航系统。

## 1.4 术语与缩写定义

|  |  |
| --- | --- |
| ****术语/缩写**** | ****释义**** |
| 车位状态 | 包含空闲、占用、损坏和维修中四种状态 |
| 城市共享停车管理系统 | 即该系统的名称。 |
| **系统** | 为方便叙述，用“系统”来代指城市共享停车管理系统。 |
| SPEGAA | *Students Physical Education Grades Administration System* |
| C/S模式 | *Client/Server模式* |
| 用例图 | 用例图是指由参与者（Actor）、功能用例，以及他们之间的关系构成的图。其目的是描述系统功能的视图 |
| DFD（数据流图） | 数据流图从数据传递和加工角度，以图示的方式来表达系统的逻辑功能、数据在系统内部的逻辑流向和逻辑变换过程，是结构化系统分析方法的主要表达工具及用于表示软件模型的一种图示方法。 |
| 客户端 | 供车主使用的安装在手机上的客户端软件，是本系统的一个组成部分。 |

表-1 术语与缩写定义表

## 1.5 引用

1. 王安生，《软件工程化》[M].北京：清华大学出版社，2014

# 2. 体系结构需求

## 2.1 关键指标

|  |  |
| --- | --- |
| 吞吐量 | 一秒钟内支持用户量不小于1000000人 |
| 系统响应时间 | 正常情况：小于2秒 |
| 最大并发数 | 5000000人 |
| 最大用户数 | 100000000人 |

## 2.2 用户特征

|  |  |  |
| --- | --- | --- |
| 用户角色 | | 特征 |
| 车主 | 持有驾驶执照的中国公民，可以驾驶所对应车型的机动车辆。将目的地输入至系统。 | | |
| 车位实时信息提供者 | 录入并更新所管辖区域的车位信息 | | |
| 停车系统管理员 | 拥有最高权限，可以确认车位实时信息提供者上传的信息并发布。 | | |

## 2.3体系结构用例

**2.3.1用户角色**

该系统有四种用户角色（参与者）：车主、车位实时信息提供者、停车系统管理员和档案，图-1展示了各个用例和参与者的关系，以及用例，参与者本身之间的联系。

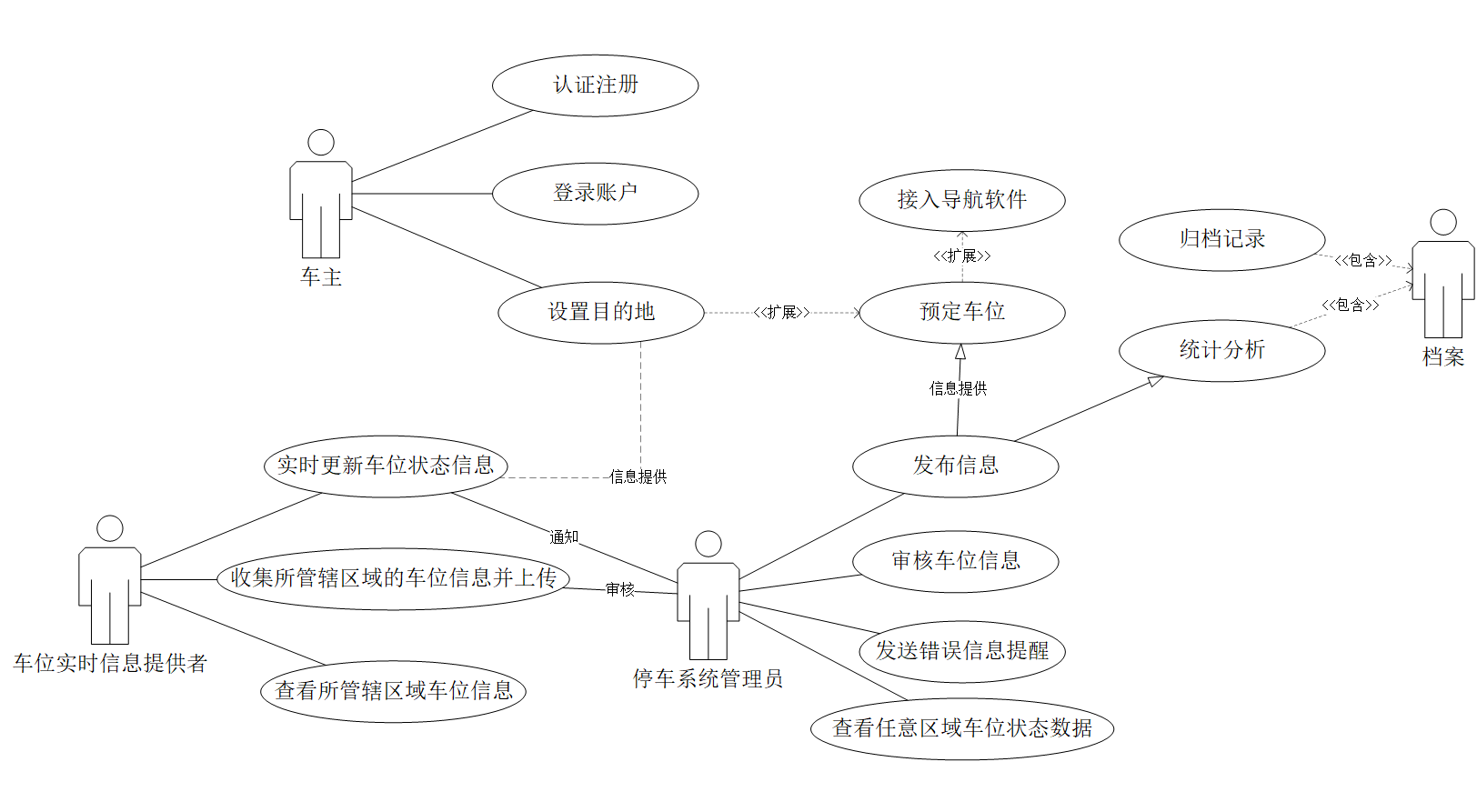


图-1 用户角色图

### 2.3.2用户角色1-车主

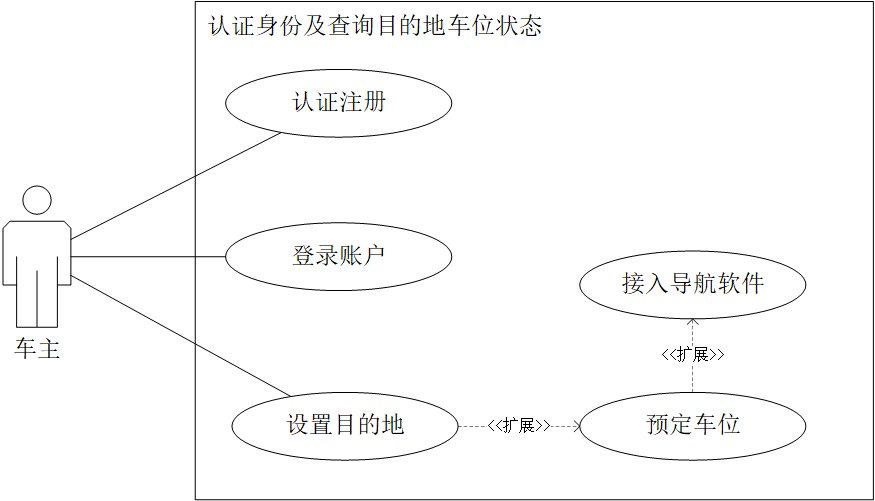


图-2 用户角色1-车主图

### 2.3.3用户角色2-停车系统管理员

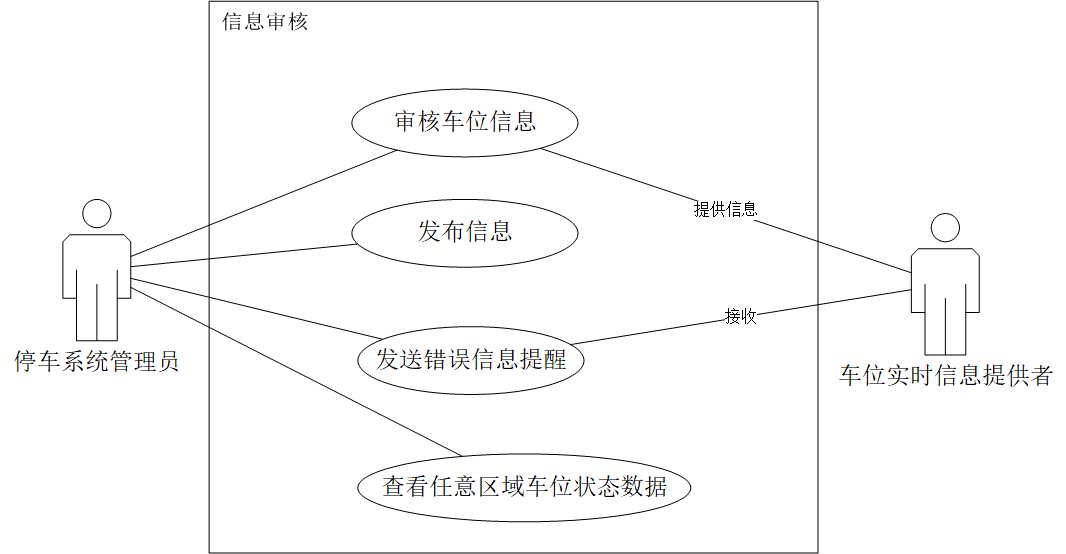


图-3 用户角色2-停车系统管理员图

### 2.3.4用户角色3-车位实时信息提供者

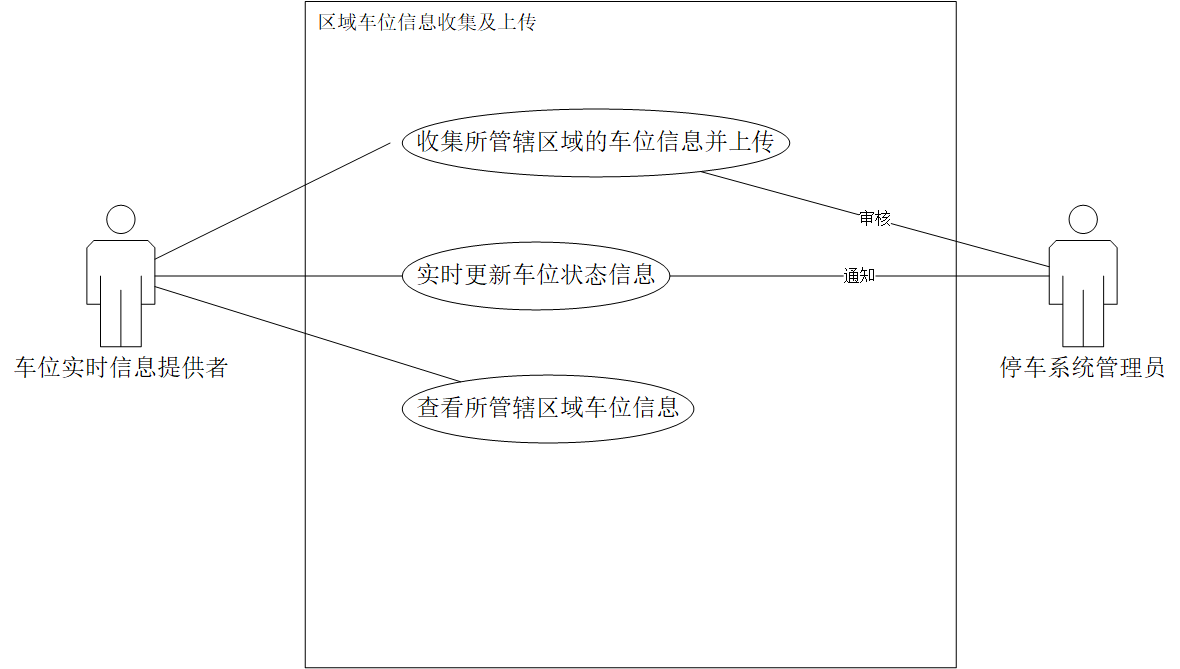


图-4 用户角色3-车位实时信息提供者图

### 2.3.5用户角色5-档案

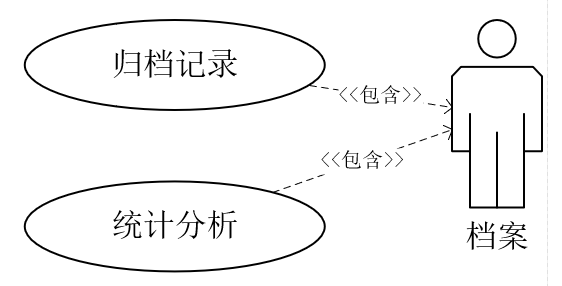


图-6 用户角色5-档案图

## 2.4 各相关方对体系结构的要求

1. 最终用户： 要求每一个角色对应的功能齐全，不同角色功能的特性要明显，并且逻辑合理，便于使用；
2. 开发人员：关注软件的管理。
3. 城市：系统能支持10000000个用户，最大并发量为500000人，正常情况响应时间小于2秒，最大并发量下响应时间控制在5秒以内；为保证系统安全，只能在校园网内使用，不允许丢失信息；系统需要保证24x365正常运行。

**2.5约束条件**

|  |  |
| --- | --- |
| 约束 | 描述 |
| 业务与项目约束条件 | |
| 项目进度要求 | 产品第一版需在600天以内交付 |
| 系统开发成本 | 1000000～1500000元 |
| 项目开发要求 | 使用Java EE编写，便于维护 |
| 业务约束 | 系统作为城市交通管理系统的子系统，导入父系统的数据，最终结果可被父系统使用 |
| 质量与可信赖性约束 | |
| 性能 | 正常情况响应时间小于2秒，最大并发量下响应时间控制在5秒以内 |
| 资源管理 | 服务器部件的内存必须具有40%以上的预留量 |
| 可使用性 | 系统需要保证24x365正常运行，可使用性达到99%以上 |
| 易用性 | 用户需安装客户端 |
| 可靠性 | 不允许丢失信息 |
| 密安性 | 所有的修改请求必须被授权，并使用认证过程进行加密 |

## 2.6 非功能性需求

## 2.6.1质量需求

### (1) 性能

1. 时间性能：
2. 本城市共享停车管理系统，客户端点击相关按钮后的响应时间应不超过3秒。
3. 按照目的地址检索信息时系统的反应时间不能超过5秒。
4. 当多用户同时访问系统时，不会出现服务器宕机的情况。根据城市总共由1126.8万名驾驶人，此人数定为900万。
5. 空间性能：
6. 数据库容量应能够存储至少600万机动车信息，1000万名驾驶人的信息。但考虑到可能的扩招，本系统的数据库容量应该能够存储700万机动车信息，1200万名驾驶人的信息。
7. 数据库容量能够存储至少1000万条车位记录，但考虑到可能的车位数量的增加和地区的扩张，本系统的数据库容量设置为能够存储1200万条车位记录。
8. 系统有足够大的缓存空间，保障系统运行流畅。

**(2)可靠性**

可靠性是指产品在规定条件下，规定时间内，完成规定功能的能力。

1. 本系统的可靠性需求具体体现在系统应能长时间下稳定运行。
2. 当用户在本系统内的各种输入不符合要求规范的时候，均不会引起系统的故障，并能提示用户错误操作。
3. 当设备故障时，系统需要具备一定的恢复能力，数据需要有至少一个备份。
4. 系统具有一定的容错和抗干扰能力，在非硬件故障或非通讯故障时，系统能够保证正常运行，并有足够的提示信息帮助用户有效正确地完成任务。

**(3)易用性**

1. 易理解性：系统的界面、使用说明、功能设计应该能让80%以上用户理解；
2. 易学性：对刚刚使用该系统的新用户，能在10分钟以内掌握整个系统的功能；
3. 易操作性：系统的75%以上操作应由鼠标能单独完成；
4. 吸引性：系统的颜色使用，图形化设计美观。

### (4)密安性

本系统的安全性十分重要。为了保证系统的安全性，在这里提出如下要求：

1. 严格保证未经认证的用户不能访问到关键信息。
2. 严格保证未经授权的操作不能被以外地、或通过不正常途径执行。
3. 除此之外，对实现方面也提出如下要求：
4. 用户的密码不得直接保存在数据库中，应当通过Hash再保存。
5. 尽量避免在业务代码中直接使用SQL访问数据库，应当使用ORM来代替。
6. 服务器除了80（HTTP）、443（HTTPS）不得对公网开放其它端口。
7. 所有表单加入CSRF验证
8. 多次登陆失败之后再次登陆要求输入验证码。

**(5)可维护性**

软件可被修改和维护的能力。在运行中，应当容易判断出系统的缺陷和失效原因。代码、设计和文档应当结构清晰，易于修改。同时，保证系统的稳定性，避免多次修改造成代码混乱，文档不清晰。

### (6)可移植性

本系统应当具有一定的跨平台和环境的能力。作为交通管理系统的子系统，应当容易与其在同一平台上运行，而不发生冲突。

1. 使用跨平台Java语言进行系统的编写，并使用开源库和开源架构。
2. 系统接口易于调用和改造，可以方便地移植到不同的设备上。

## 2.6.2 工程需求

### (1) 逆向需求

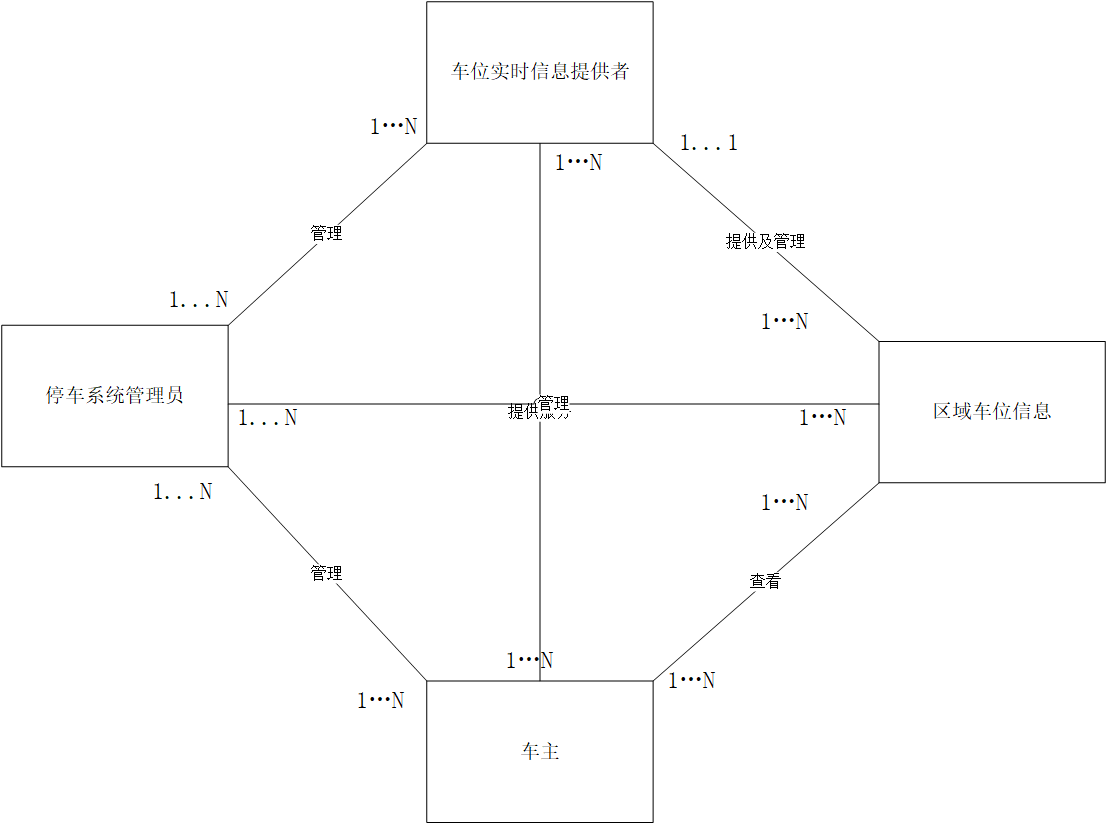
更新系统时，要求分析旧系统，在旧系统的基础上增加新功能，在添加新功能时，

尽量无需修改其他功能。

**(2) 设计约束**

|  |  |  |
| --- | --- | --- |
| 设计要素 | | 主要约束 |
| 运行环境 | 操作系统 | Windows 7/Linux 10及以上 |
| 数据库 | MySQL 14.0及以上 |
| Web服务器 | WebLogic |
| 用户端PC软件 | 操作系统 | Windows/OSX/Linux/Android/iOS |
| 浏览器 | Chrome/Edge/Firefox/Safari |
| 开发环境支持 | 操作系统 | Linux Ubuntu 16.04 |
| 开发工具 | myeclipse |
| Web服务器 | WebLogic |
| CPU | 3.4 GHz Intel Core i7 |
| 内存 | 16GB |

## (3) 逻辑数据库需求



**2.7 风险**

**2.7.1 风险一：系统并发量超过最大容忍度导致系统崩溃；**

**2.7.2 风险二：在车位实时信息提供者上传车位状态信息、车主在预定车位时，出现网络故障导致传输过程中信息丢失；**

**2.7.3 风险三：服务器的数据库异常，造成用户登录失败；**

# 3. 解决方案

## 3.1 相关的体系结构模式

### 3.1.1 共享数据仓库模式

•在共享仓库模式中，有一个部件作为数据仓储，被其它的独立部件访问。这样的仓库可以被多个部件同时访问；

•对共享仓库模式，必须提供一些机制，例如信息安全机制；

–有些系统提供更高层的访问机制，例如，数据库管理系统(DBMS)的SQL查询语言实现对数据库表的并行访问；

•共享仓库是对顺序性体系结构（例如分层模式和管道过虑器模式）的一种替代，优势在于实现了数据共享，以及不相邻的两个部件之间的信息交流。如果将其用到管道过滤器中，可以实现不同过滤器之间的数据共享；

–在共享仓库中，所有的客户端都是独立的部件，很像是C/S结构，数据仓储扮演了服务器的角色。也可以认为是两层的C/S结构；

### 3.1.2 分层的C/S模式

1）分割：

–表现层、业务层和数据处理逻辑被分割到不同的层面；

2）各层之间异步通信：

–层与层之间的通信是异步的“请求-响应(request-reply)”。请求是单方向的，从客户层开始，经过Web和业务逻辑层，再到数据管理层。每层都等待其它层的处理响应。

3）部署灵活：

–分层结构不限制多层应用的部署方式。所有层可以运行在一台机器上，或者，每层部署到一台独立的机器上。

4）标准化：

–分层可以实现各层的标准化，例如，在Web应用中，客户端可以使用不同厂家的浏览器，兼容地对不同厂家Web服务器访问。

采用分层分布式系统框架的C/S结构示意图：



## 3.2 体系结构概述

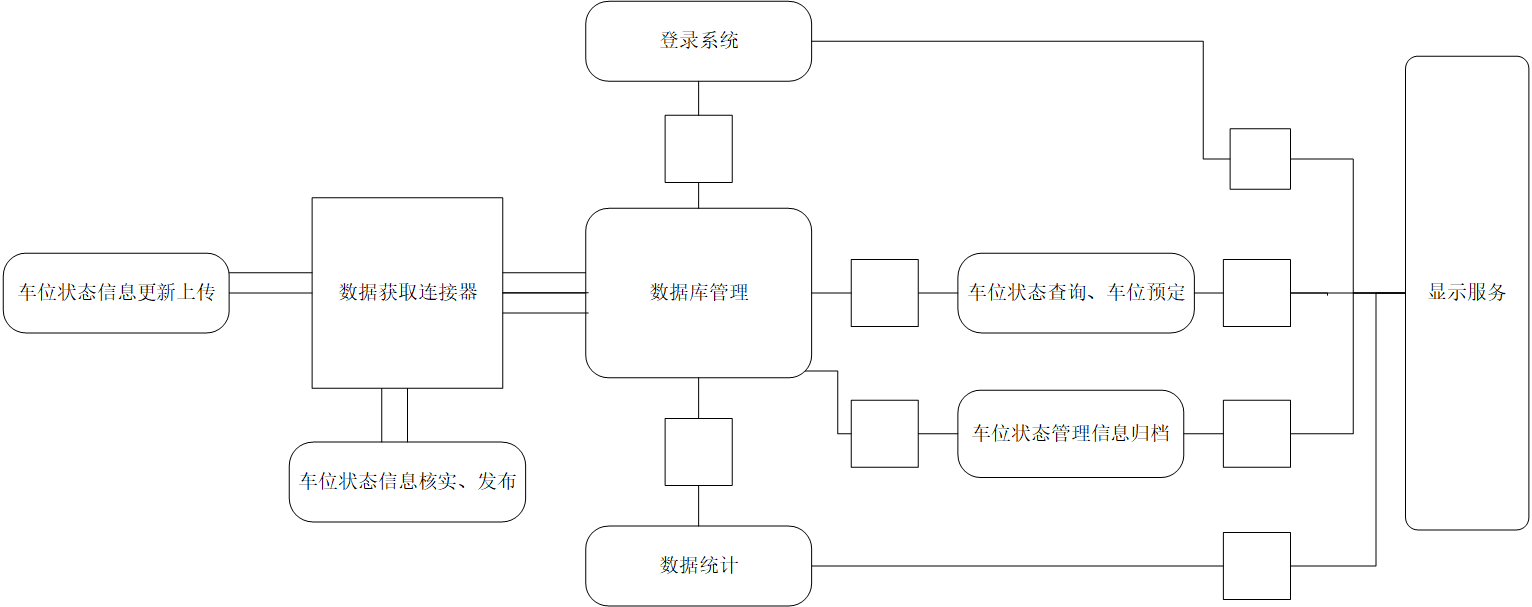
城市共享停车管理系统核心在于车主可以对目的地附近的车位状态进行查询和预订，车位实时信息提供者可以实时更新其所负责的区域的车位状态信息。

在系统运行过程中，各个角色均访问同一类数据，为了实现数据共享以及不同角色的部件之间信息交流，采取共享数据仓库模式。

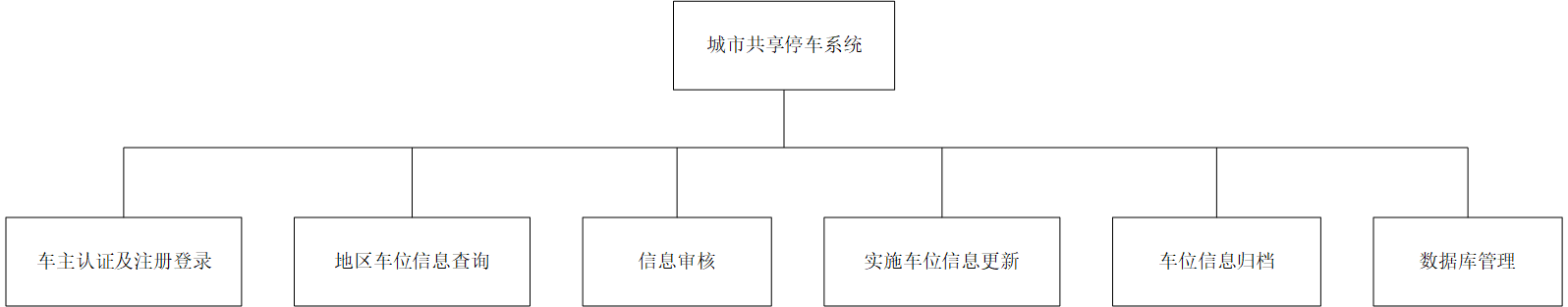
选用分层的C/S模式的原因： 系统需实现车位实时信息提供者上传车位状态数据的功能，需要读取终端的状态数据并上传，面临终端系统不同，软件版本控制较麻烦，使用客户端不容易统一，不利于成绩管理。使用分层的C/S模式可以通过浏览器调取车位状态数据上传，停车系统管理员也可使用浏览器查询车位状态信息。其他角色用户的功能同样适合分层的C/S模式开发，于是统一各个功能的结构模式，同时节约了开发成本。

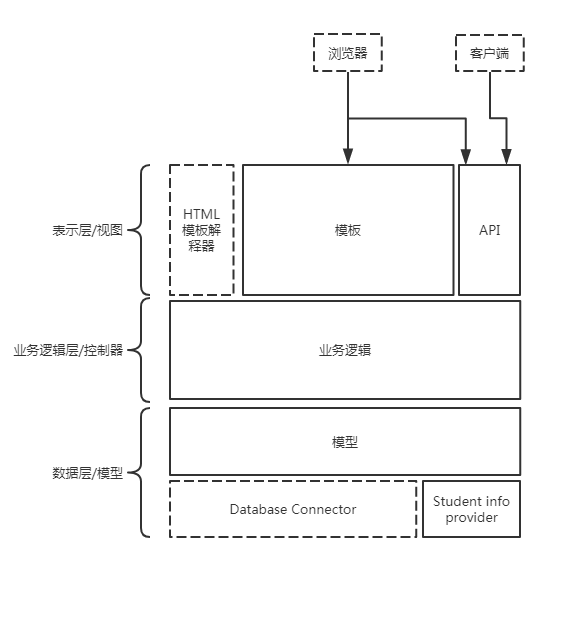
## 3.3 结构化视图

### 概念级体系结构视图



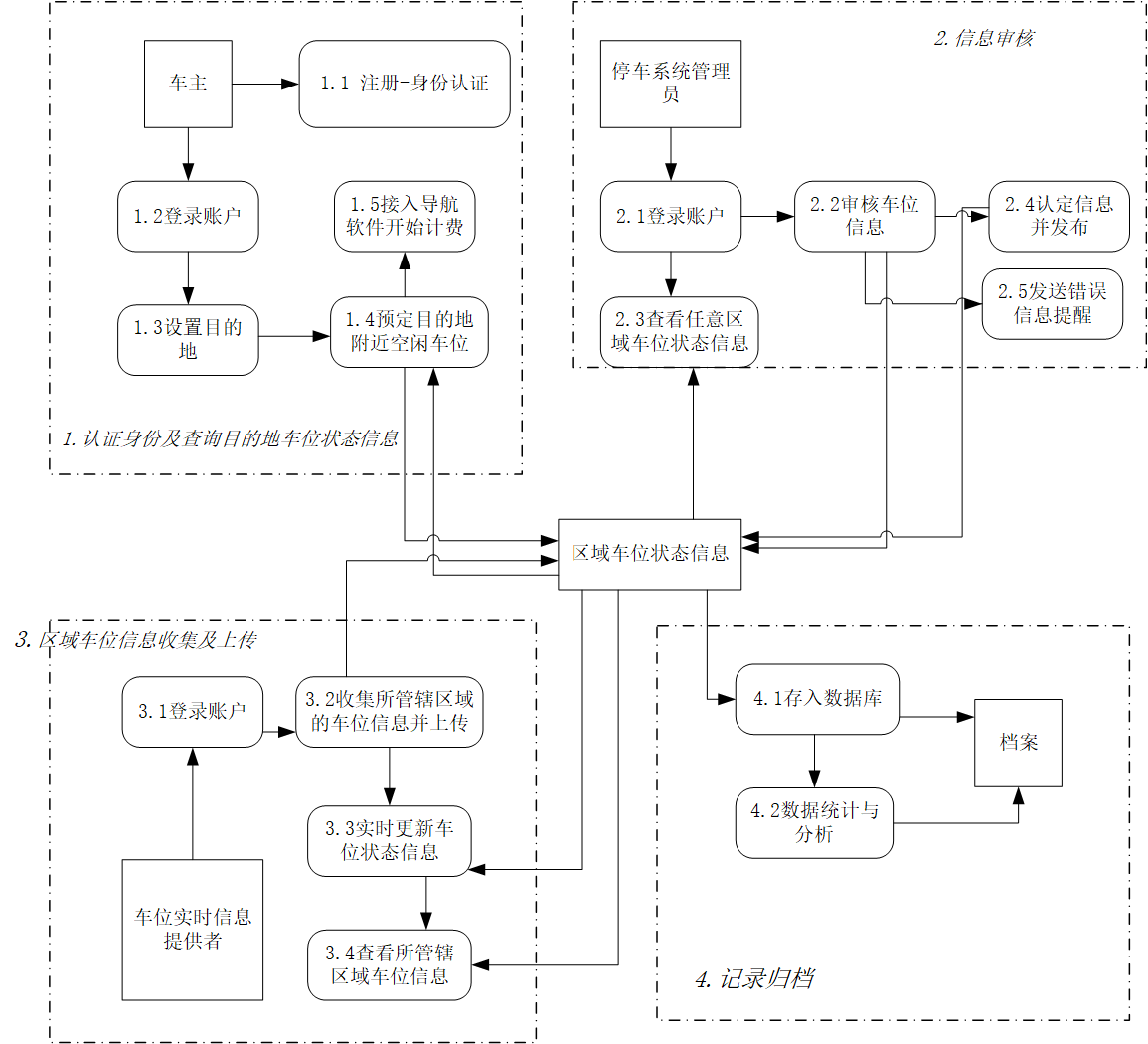
### 3.3.2模块级体系结构视图



3.3.3 纵向层次划分结构视图

纵向架构采用饱经历史考验的MVC三层架构。这有多方面的原因：其一，MVC架构在饱经历史的考验之后，如今依然广泛使用，正是得益于它提供的可维护性、可扩展性，比如，试图一层和业务逻辑解耦，因而可以非常方便地更换试图而无需关注下层的具体构造；其二，由于MVC架构应用相当广泛，因而对于我们当前的这个Web应用来说，有很多现有的MVC框架可以采用，大大节省了软件开发成本和减少了工期。上图中的HTML模板解释器和Database Connector通常在这些MVC框架中都会提供。

### 3.3.4运行级体系结构



**3.4 实现问题**

（1）软硬件条件的限制；

（2）软件系统设计人员对系统理解不到位；

（3）软件开发人员编程水平有限；

（4）时间限制。

# 4. 系统的质量分析与评价

## 4.1 场景分析

### 用例场景

4.1.1.1场景1：车主登录系统后，可以随时查询某区域的车位状态信息，系统可以调用终端的接 口读取地理位置信息上传到服务器，再结合对应的区域车位信息统计数据进行显示。该场景代表了用户期待系统易于使用，即易用性；

* + - 1. 场景2：发生数据异常时，系统要通知系统管理员，并提示修改。该场景代表了用户期望的可靠性；
      2. 场景3：车主用户可在手机端浏览器使用系统，车位实时信息提供者与停车系统管理员在电脑端使用系统。该场景代表了用户期望的可使用性；
      3. 场景4：用户在登录系统时，期望填写验证码防止账号密码泄漏。该场景代表了用户期望的可靠性；
      4. 场景5: 用户通过web获得数据报告的时间小于5秒。该场景代表了系统的性能要求；
      5. 场景6: 当处理器发生故障时，缓存系统要能在1秒中内从一个处理器切换到另一个处理器，该场景代表了系统可靠性要求。

### 增长性场景

4.1.2.1增长性场景1: 可以增加新的数据服务器，将用户通过Web访问时间从5秒降低到2.5秒，希望系统仅需增加一人周的工作量就能完成对系统的调整；

4.1.2.2增长性场景2:通过扩充现有数据库表的规模，把检索时间降低到平均1秒之内；

4.1.2.3增长性场景3:车主使用手机端浏览器访问系统时在一段时间内可以记住密码，更方便地访问查询。

### 探索性场景

* + - 1. 探索性场景1：改进系统的可使用性，从98% 提升到 99.999%；
      2. 探索性场景2：当城市车主数量在8000000人左右同时使用该系统时，系统可承受并且反应时间控制在5秒内；
      3. 探索性场景3：对车主用户指定的目的地附近区域的车位状态进行分析，平衡经济、道路拥堵状况、个人喜好等信息，为用户推荐最佳的停车位置。
      4. 探索性场景4：将系统做成微信小程序，方便手机端用户定位和查询车位信息。

## 原型分析

使用原型法进行需求分析的流程如下：

1. 快速分析，列出用户的基本信息需求：

城市共享停车系统的用户分为三类：车主、车位实时信息提供方和停车系统管理员。车位实时信息提供者借助信息手段将所管辖区域内的空闲车位数量及详细地址信息上传至管理系统系统，并定期更新数据。停车系统管理者对信息提供者进行审批并进行认定，认定后的信息才可以发布到管理系统中，之后完成认证的车主可以通过移动端或PC端查询目的地附近的空闲车位情况，在完成空闲车位的预订工作后开始计费，管理系统将自动接入导航软件快速引导车辆停放。

1. 构造原型，开发初始原型系统：

原型系统先考虑原型系统应必备的待评价特性，暂时不考虑系统的安全性、健壮性，建立基本满足车主、车位实时信息提供方和停车系统管理员使用要求的系统；

1. 用户和开发人员共同评价原型：

本步骤要求系统的使用者使用原型进行初步的评价，还要求开发人员对原型系统进行最基本的测试工作，从而找出系统需要改进的地方并进一步进行完善。

## 风险

1. 车位实时信息提供者在更新数据时，出现网络障碍，数据上传不完全却无法修改，导致系统车位状态与实际不符。
2. 车主用户终端出现网络故障，将导致定位信息不准确，出现超时计费等问题。
3. 大量车主同时使用系统功能导致系统响应变慢甚至崩溃；
4. 车位实时信息提供者在更新车位状态信息的同时有用户进行车位预定或查询行为，导致预定失败或者查询失败。
5. 城市共享停车系统的父系统——交通管理系统中信息变动，则可能会导致子系统信息未及时更新，成绩出差错。