<城市共享停车管理系统>

体系结构文档

<1.0>

<2019.12.06>

<原正阳>

<2017211959>

<软件学院504班>

# 版本历史

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **描述** | **作者** | **注释** |
| <2019.12.06> | <版本 1> | <原正阳> | <第一个版本> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 文件许可

以下的软件需求规格已被接受并认可：

|  |  |  |  |
| --- | --- | --- | --- |
| **签名** | **名称** | **标题** | **日期** |
|  | 打车软件系统 | 软件架构师 | 2019.12.06 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[版本历史 ii](#_Toc26556444)

[文件许可 ii](#_Toc26556445)

[1.引言 2](#_Toc26556446)

[1.1编写目的 2](#_Toc26556447)

[1.2范围 2](#_Toc26556448)

[1.3术语定义 2](#_Toc26556449)

[1.4参考文献 4](#_Toc26556450)

[1.5概述 4](#_Toc26556451)

[2.项目相关信息 5](#_Toc26556452)

[2.1项目概述 5](#_Toc26556453)

[2.2用户角色 5](#_Toc26556454)

[2.3项目功能 6](#_Toc26556455)

[2.3.1 用户注册/登录模块 6](#_Toc26556456)

[2.3.2 乘客交互模块 6](#_Toc26556457)

[2.3.3司机交互模块 6](#_Toc26556458)

[2.3.4第三方机构模块 7](#_Toc26556459)

[2.3.5抢单判定模块 7](#_Toc26556460)

[2.3.6业务检测员模块 7](#_Toc26556461)

[2.3.7系统管理员模块 7](#_Toc26556462)

[2.3.8系统功能结构图 8](#_Toc26556463)

[3.体系结构需求 9](#_Toc26556464)

[3.1关键指标 9](#_Toc26556465)

[3.2体系结构用例 9](#_Toc26556466)

[3.2.1司机的用例图 10](#_Toc26556467)

[3.2.2乘客的用例图 11](#_Toc26556468)

[3.2.3 第三方机构的用例图 12](#_Toc26556469)

[3.2.4 业务检测人员的用例图 13](#_Toc26556470)

[3.2.5 系统管理人员的用例图 13](#_Toc26556471)

[3.3各相关方对体系结构的要求 14](#_Toc26556472)

[3.4约束条件 14](#_Toc26556473)

[3.5设计遵循的标准 15](#_Toc26556474)

[3.5.1 实用性原则 15](#_Toc26556475)

[3.5.2 稳定性原则 15](#_Toc26556476)

[3.5.3 复用性原则 15](#_Toc26556477)

[3.5.4 扩展性原则 15](#_Toc26556478)

[3.5.5 安全性原则 15](#_Toc26556479)

[3.6非功能需求 16](#_Toc26556480)

[3.6.1质量要求 16](#_Toc26556481)

[3.6.2工程需求 18](#_Toc26556482)

[3.6.3其他需求 19](#_Toc26556483)

[3.7风险 19](#_Toc26556484)

[4.解决方案 19](#_Toc26556485)

[4.1相关的体系结构模式 19](#_Toc26556486)

[4.1.1三层B/S体系结构模式 19](#_Toc26556487)

[4.1.2三层B/S体系结构的优点 20](#_Toc26556488)

[4.1.3 三层B/S体系结构对质量属性的影响 21](#_Toc26556489)

[4.2体系结构概述 22](#_Toc26556490)

[4.2.1 概念级体系结构 22](#_Toc26556491)

[4.2.2 模块级体系结构 23](#_Toc26556492)

[4.2.3 运行级体系结构 24](#_Toc26556493)

[4.2.4 代码级体系结构 25](#_Toc26556494)

[4.3结构化视图 27](#_Toc26556495)

[4.3.1概念级体系结构视图 27](#_Toc26556496)

[4.3.2逻辑视图 27](#_Toc26556497)

[4.3.3运行级体系结构视图 28](#_Toc26556498)

[5.系统的质量分析与评价 28](#_Toc26556499)

[5.1场景分析 28](#_Toc26556500)

[5.1.1用例场景 28](#_Toc26556501)

[5.1.2增长性场景 29](#_Toc26556502)

[5.1.3探索性场景 29](#_Toc26556503)

[5.2原型分析 29](#_Toc26556504)

[5.3风险 31](#_Toc26556505)

[5.3.1 技术风险 31](#_Toc26556506)

[5.3.2 进度风险 31](#_Toc26556507)

[5.3.3 质量风险 32](#_Toc26556508)

# 1.引言

## 1.1编写目的

本体系结构设计文档是在对打车软件系统进行了全面细致的需求分析，明确了所要开发的系统应具有的功能、性能之后编写的文档，旨在阐述打车软件系统的总体结构，包括逻辑设计、物理结构，分析系统的体系结构需求，包括约束条件、设计遵循的标准、非功能性需求，给出体系结构设计的解决方案并分析建模，最后进行体系结构的质量分析和评估。

本体系结构设计文档作文产品立项和产品开发的参考文档，给出各用户详细的功能要求，系统功能块组成及联系，进程部署和硬件要求等，有益于提高软件开发过程中的能见度，便于软件开发过程中的控制与管理。此体系结构设计文档是进行软件项目设计开发的基础，也是编写测试用例和进行系统测试的主要依据，它对开发的后续阶段性工作起着指导作用。同时此文档也可作为软件用户、软件客户、开发人员等各方进行软件项目沟通的基础。

本文档的预期读者对象为：

1) 开发人员：根据本文档了解预期项目的功能，并据此进行系统设计与开发。

2) 项目经理：根据体系结构定义的构件结构制定项目的开发计划。

3) 测试人员：根据体系结构设计系统的总体测试框架。

4) 维护人员：根据本文档中确定的体系结构进行软件系统维护。

5) 用户：了解预期项目的功能和性能与整体结构。

6) 其他相关人员：如用户文档编写者、项目管理人员等。

## 1.2范围

（1）软件系统名称：打车软件系统（Taxi Software System,**TSS**）

（2）内容：本体系结构设计文档概括地描述了城市共享停车管理系统的主要功能，阐述了系统的总体结构，说明了系统的总体设计策略，给出了体系结构设计的解决方案并分析建模，最后进行体系结构的质量分析和评估

（3）范围：本软件体系结构设计文档适合于城市共享停车管理系统的总体应用结构，目的是满足系统的质量和可信赖性要求，以及未来的维护、运行和升级改造等要求。

## 1.3术语定义

|  |  |
| --- | --- |
|  |  |
| 术语 | 定义 |
| TSS系统 | 打车软件系统（Taxi Software System,**TSS**） |
| B/S结构 | B/S结构（Browser/Server，浏览器/服务器模式），是WEB兴起后的一种网络结构模式，WEB浏览器是客户端最主要的应用软件。这种模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用。 |
| 三层B/S结构 | 三层B/S体系结构是在B/S结构的基础上，在数据管理层(Server)和用户界面层(Client)增加了一层结构，称为中间件(Middleware)，使整个体系结构成为三层。 |
| C/S结构 | C/S 架构是一种典型的两层架构，其全程是Client/Server，即客户端服务器端架构，其客户端包含一个或多个在用户的电脑上运行的程序，而服务器端有两种，一种是数据库服务器端，客户端通过数据库连接访问服务器端的数据；另一种是Socket服务器端，服务器端的程序通过Socket与客户端的程序通信。 |
| 概念级体系结构 | 描述系统的主要设计元素和元素之间的关系。 |
| 模块级体系结构 | 按两两正交模块的结构组织系统，即，功能分解和层次分解。 |
| 运行级体系结构 | 描述系统的动态结构。 |
| ATAM | Architecture Tradeoff Analysis Method；体系结构折中分析方法。这是SEI开发的通过手工评估和确认体系结构的方法。 |
| 用户角色 | 用户角色是指按照一定参考体系划分的用户类型，是能够代表某种用户特征、便于统一描述的众多用户个体的集合。 |
| 功能需求 | 功能需求是对未来软件的服务功能进行的描述。功能需求更关注系统的输入、加工（业务流程）直到输出的情况 |
| 非功能需求 | 非功能需求是指功能以外的需求描述，主要是系统的一些关键的特性，例如可靠性、响应时间、存储空间、I/O设备的吞吐量、接口的数据格式等 |
| 用例图 | 用例图是指由参与者（Actor）、用例（Use Case）以及它们之间的关系构成的用于描述系统功能的视图。用例图（User Case）是被称为参与者的外部用户所能观察到的系统功能的模型图，呈现了一些参与者和一些用例，以及它们之间的关系，主要用于对系统、子系统或类的功能行为进行建模。 |

表1-术语定义

## 1.4参考文献

[1] 王安生. 软件工程化[M]. 北京：清华大学出版社,2015.

## 1.5概述

本文档将依次介绍：介绍、项目相关信息、体系结构需求、解决方案、系统质量分析和评价等。

即按照下图

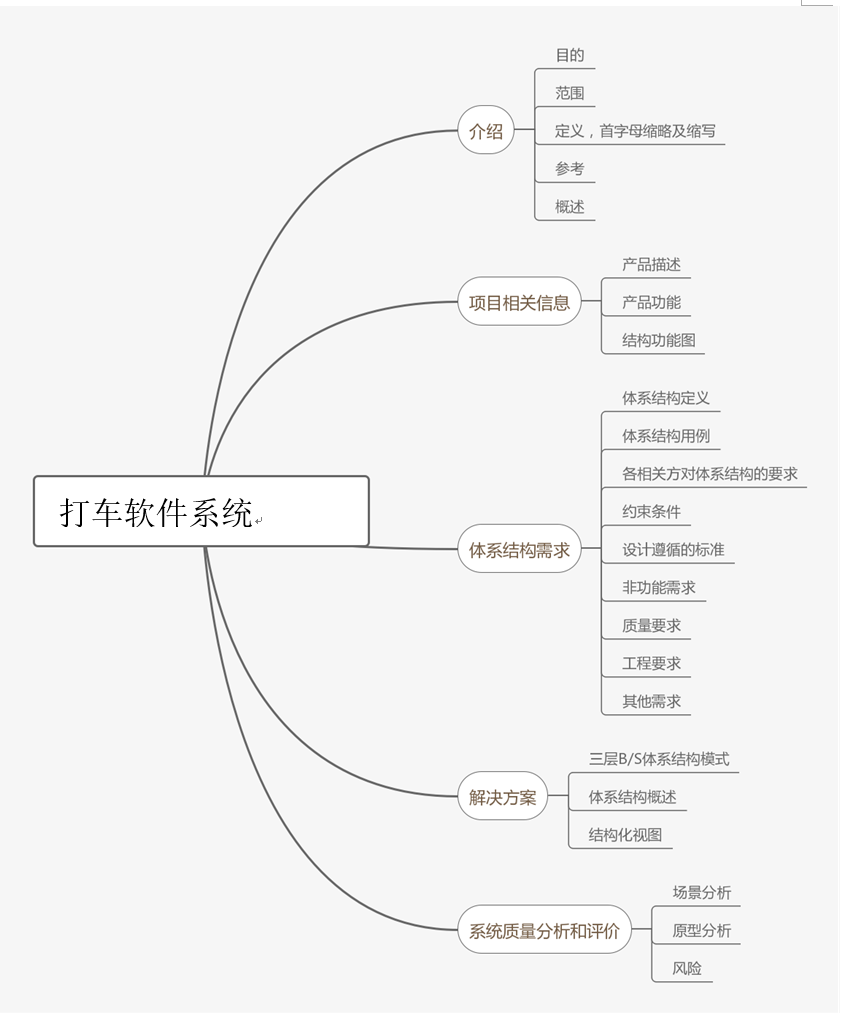


图 1 打车软件系统体系结构图

# 2.项目相关信息

在这一部分对打车软件系统的相关信息进行概括性的描述，包括项目概述、用户角色、项目功能。

## 2.1项目概述

（１）能实现主要功能

* 为乘客提供即时打车服务、预约打车服务和寻求代驾服务
* 为司机提供接单载客服务、路况信息服务和收听广播服务
* 为第三方提供广告推送服务、本地信息显示服务（包括餐饮娱乐及酒店宾馆）、广播服务、受付服务、地图服务、交通管理服务

（２）运行环境

* 硬件平台：智能手机等移动客户端
* 操作系统：安卓系统和IOS系统都可以
* 共存软件：地图软件、社交软件（如微信），可以嵌入到用户群体很大的如微信、支付宝、高德地图等软件中调用打车软件，或者在打车软件中调用地图API等。

（３）用户界面简洁、方便且快速。

（４）本系统初步设计为独立的系统，并且全部内容自含。

（５）在后期可以进行扩展，为其他的类似系统提供服务，如共享单车等。

（６）能从很大程度上解决“打车难”及“空载”的问题。

（７）从最初给乘客和司机提供免费、便利的打车服务从而积累用户，到最后通过软件增值服务、第三方支付平台、本地信息服务、入口价值等方式实现盈利

（８）系统保证密安性，不会不会泄露用户的信息，也不允许不具有特定权限的用户更改信息。

## 2.2用户角色

在打车软件系统中有五类用户：司机，乘客，第三方机构，业务检测员和系统管理员。这五类用户角色有不同的系统使用权限，可以对系统中的打车信息数据库、用户信息数据库和账单信息数据库进行不同的操作。不同用户角色对系统使用的需求不同，具体如下：

司机是本系统的主要使用者，司机使用账号和密码登录此系统进入司机端，进行接单，查看订单，客户管理，查看路况，收听广播，设置个人资料等功能。

乘客也是主要使用者使用账号和密码登录此系统，进入乘客端可以进行即时叫车，预约叫车，申请代驾，支付订单，查看空车，查看司机信息，评价，设置个人资料等功能。

第三方机构提供支付和地图接口，也可以在系统中投放广告。

业务监测人员用分配的账号和自定义的密码登录，可以查看订单交易信息以及司机和成乘客的信息，进行业务检测和数据分析。

系统管理是系统最高权限的拥有者，使用特殊账号和密码登录此系统，可以录入、查询和修改所有的数据库的信息，并处理司机和乘客提交的更新信息和举报信息。

## 2.3项目功能

### 2.3.1 用户注册/登录模块

为不同用户设计的。主要设计思想如下：

1. 不同用户可以注册各自不同类型账号
2. 用户可以用本模块进行登录
3. 可以管理个人账户资料
4. 可以修改密码

### 2.3.2 乘客交互模块

乘客交互模块是针对乘客设计的。主要设计思路

(1)系统要求系统管理员用户在进入系统之前输入用户名和密码进行信息的验证。 系统管理员具有当前系统的最高权限。

(2)系统要求当密码校验失败时，会提示错误，如果超过系统允许的最大连续 出错值，那么会对该用户进行锁定。

（3）为乘客提供即时打车服务、预约打车服务和寻求代驾服务

为第三方提供广告推送服务、本地信息显示服务（包括餐饮娱乐及酒店宾馆）、广播服务、受付服务、地图服务、交通管理服务

### 2.3.3司机交互模块

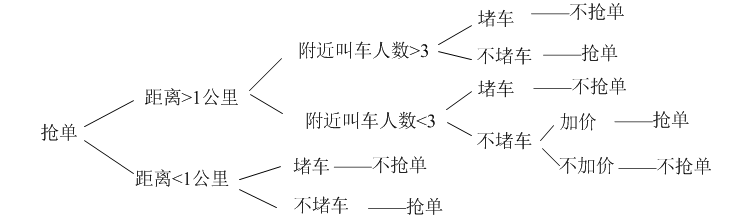
1. 系统要求系统管理员用户在进入系统之前输入用户名和密码进行信息的验证。 系统管理员具有当前系统的最高权限。
2. 系统要求当密码校验失败时，会提示错误，如果超过系统允许的最大连续 出错值，那么会对该用户进行锁定。
3. 为司机提供接单载客服务。
4. 路况信息服务和收听广播服务。

### 2.3.4第三方机构模块

1. 实现支付接口，如网银、支付宝、微信等第三方支付公司
2. 实现地图接口使用第三方地图显示位置信息、路况信息和本地服务信息等，如高德、百度等。
3. 为第三方提供广告推送服务、本地信息显示服务（包括餐饮娱乐及酒店宾馆）、广播服务、受付服务、地图服务、交通管理服务

### 2.3.5抢单判定模块

本模块用来判定乘客发出乘车需求后，由哪个司机进行接单。如下是抢单判定树



### 2.3.6业务检测员模块

业务检测员模块是针对具有专业知识的业务检测人员设计的。主要设计思想如下：

(1)系统要求业务检测用户在进入系统之前输入用户名和密码进行信息的验证。 系统管理员具有当前系统的最高权限。

(2)系统要求当密码校验失败时，会提示错误，如果超过系统允许的最大连续 出错值，那么会对该用户进行锁定。

(3)系统允许业务检测人员获得客户以及订单信息，进行业务检测与数据分析。

### 2.3.7系统管理员模块

系统管理员模块是针对具有专业知识的系统管理员设计的。主要设计思想如下：

(1)系统要求系统管理员用户在进入系统之前输入用户名和密码进行信息的验证。 系统管理员具有当前系统的最高权限。

(2)系统要求当密码校验失败时，会提示错误，如果超过系统允许的最大连续 出错值，那么会对该用户进行锁定。

(3)系统允许系统管理员对人员信息进行增删改查的工作。人员包括司机乘客。

### 2.3.8系统功能结构图

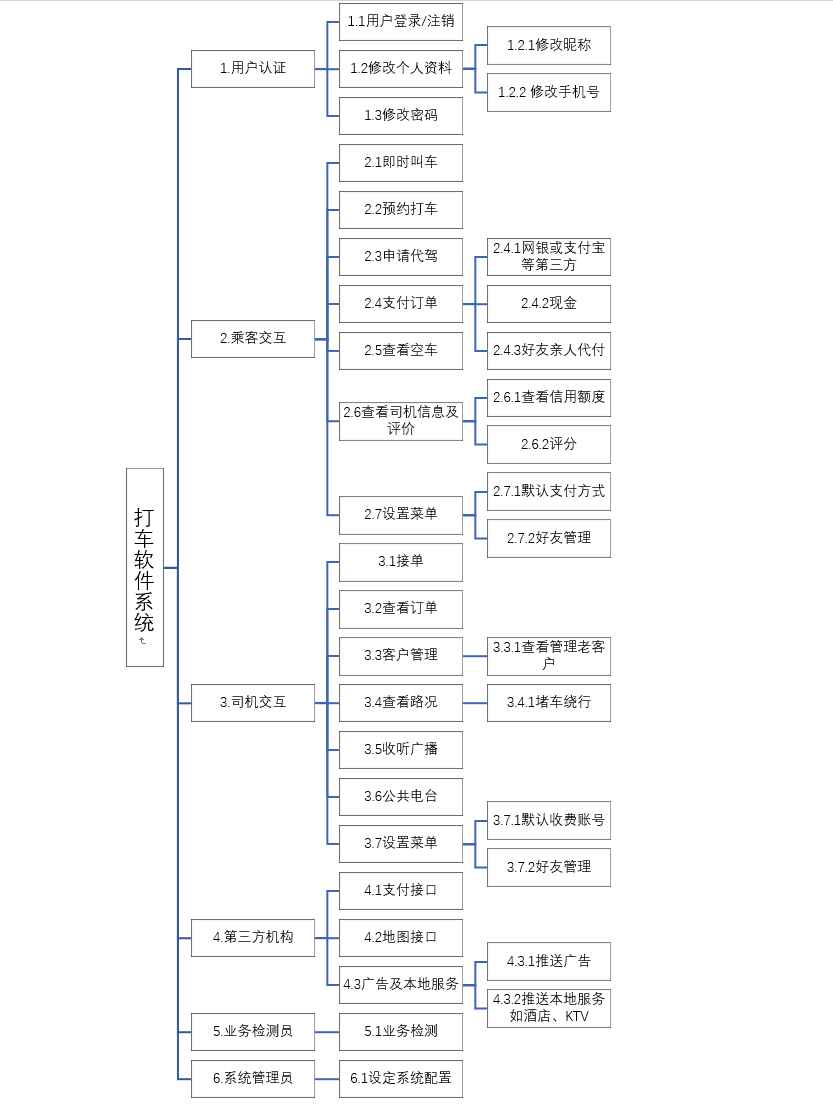


图 2 打车软件系功能结构图

# 3.体系结构需求

软件体系结构是具有一定形式的结构化元素，即构件的集合，包括处理构件、数据构件和连接构件。处理构件负责对数据进行加工，数据构件是被加工的信息，连接构件把体系结构的不同部分组组合连接起来。这一定义注重区分处理构件、数据构件和连接构件，这一方法在其他的定义和方法中基本上得到保持。由于软件系统具有的一些共通特性，这种模型可以在多个系统之间传递，特别是可以应用到具有相似质量属性和功能需求的系统中，并能够促进大规模软件的系统级复用。

## 3.1关键指标

本系统主要面向广大司机和乘客以及第三方机构，业务检测人员，系统管理人员，使用本系统，乘客可以很方便的发起打车订单去往要求的地方，并结算费用少去等待时间。司机也可以方便接单，收取费用，第三方机构可以投放广告，业务检测人员可以收集数据进行分析，系统管理人员可以对系统进行管理。并且同时，由于所储存的订单信息、用户信息和交易信息属于比较重要的信息，所以要求系统具有较高的稳定性和安全性，降低系统出错率，才能保证系统正常、高效、快速的使用，所以本系统在性能上还需要具备以下关键指标：

（1）系统应该至少支持250万个司机和6万个乘客使用。

（2）系统应该支持300万个司机同时查询订单记录进行接单，支持6万个乘客同时查询车辆信息、账单信息和结算费用。

（3）系统应该支持启动速度在0.5s以内

（4）系统应该支持启动之后各项操作反应速度0.1s以内（快速的反应有利于增加用户体验度）

（5）系统应该支持当多用户同时访问系统时，不会出现服务器宕机的情况。

（6）系统的数据库容量应能够存储至少500万个司机、6万个乘客、2亿条交易记录的信息。

（7）软件正常运行RAM占用25M以内。

（8）精简代码控制软件的bug量，连续运行一周不能出程序未响应或闪退情况，重要功能如打车功能一定要可靠稳定

（9）要易于使用、操作简洁，设置常用功能快捷键或快捷手势，复杂功能应放入菜单中，用户的操作体验很重要，后期要进行操作体验测试。

## 3.2体系结构用例

用例图是由参与者，功能用例以及它们之间的关系构成的图，其目的是描述系统功能，通过用例图呈现参与者和他们之间的关系构成的图，就可以更清晰地了解用户对系统、子系统或者各项功能的使用和行为。

本打车软件系统的用户角色包司机，乘客，第三方机构，业务检测人员，系统管理员。不同的用户对应着不同的功能，针对这些用户角色，可以画出如下的用例图。

### 3.2.1司机的用例图

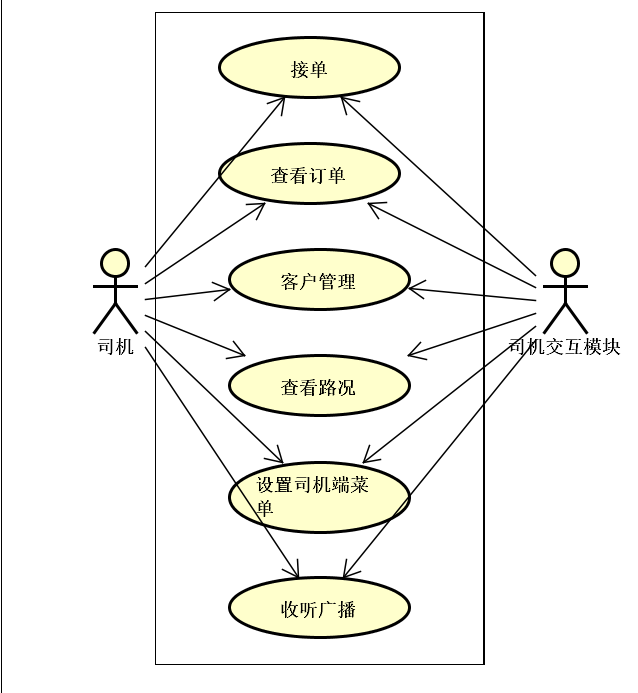


图3- 司机的用例图

分析司机用例的具体功能如下：

* 使用账号和密码登录系统
* 接单
* 查看订单
* 客户管理
* 查看路况
* 设置司机端菜单
* 收听广播

### 3.2.2乘客的用例图

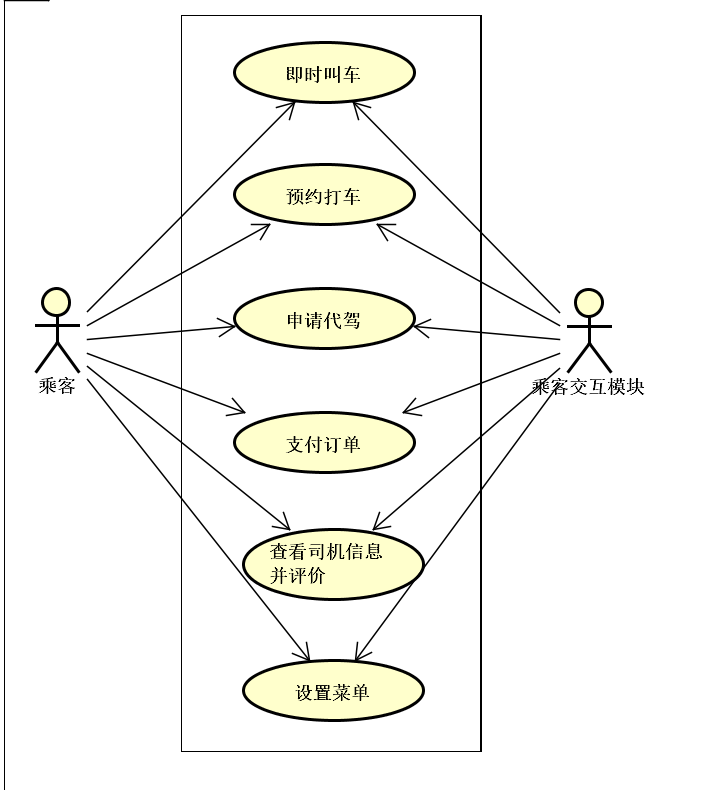


图4-乘客的用例图

分析乘客用例的具体功能如下：

* 使用账号和密码登录系统
* 即时叫车
* 预约打车
* 申请代驾
* 支付订单
* 查看司机信息并评价
* 设置菜单

### 3.2.3 第三方机构的用例图

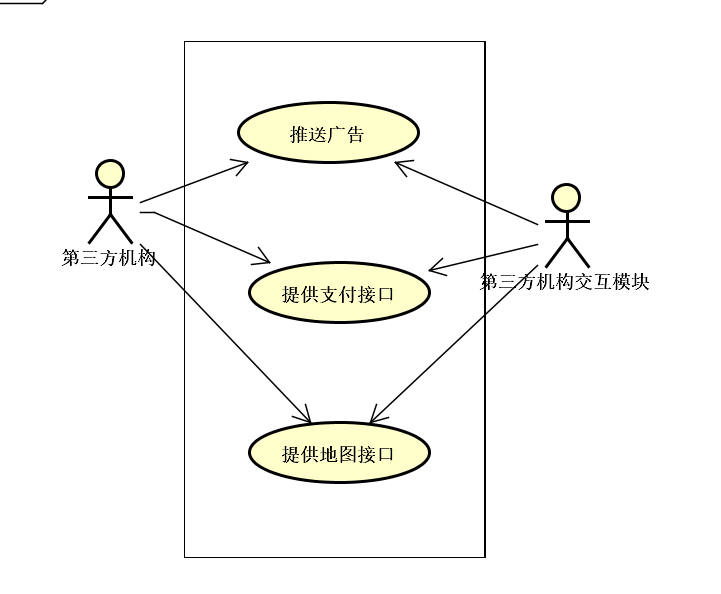


图5-第三方机构的用例图

分析第三方机构用例的具体功能如下：

* 推送广告
* 提供支付接口
* 提供地图接口

### 3.2.4 业务检测人员的用例图

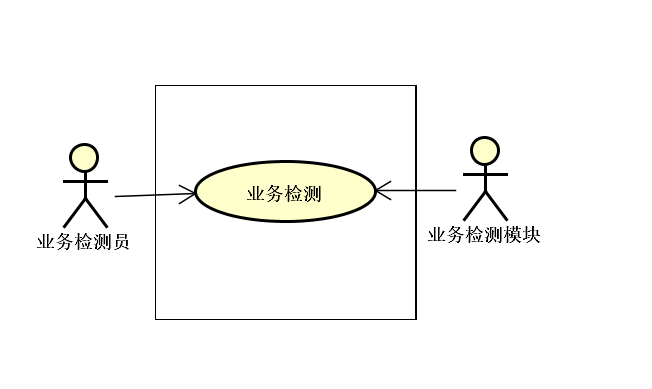


图6-业务检测人员的用例图

分析业务检测人员用例的具体功能如下：

* 业务检测

### 3.2.5 系统管理人员的用例图

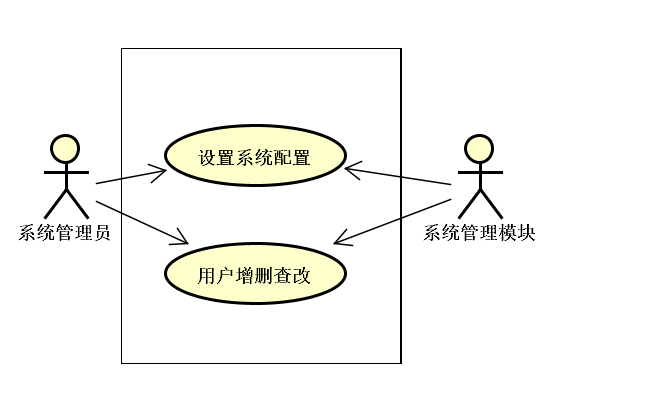


图7-系统管理人员的用例图

分析系统管理人员用例的具体功能如下：

* 设置系统配置
* 用户增删查改

## 3.3各相关方对体系结构的要求

本打车软件系统结构视角如图5-所示，该模型从4个角度（逻辑、实现、进程和部署）指出不同的相关利益方关心的事情，外加从使用者的角度对用例做观察，分析其影响系统的上下文和商业目标情况。

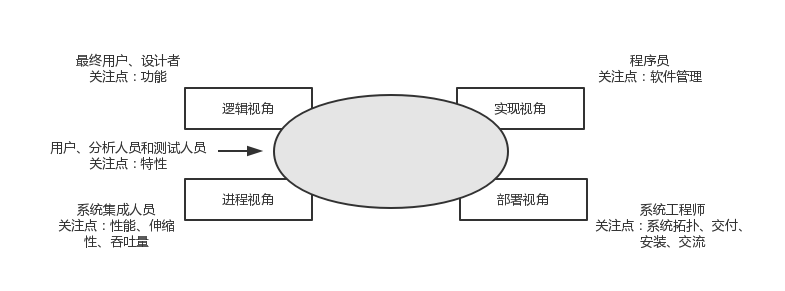


图8-软件体系结构视角

* 系统的最终用户和设计者要求系统可以实现需求分析阶段的基本功能，对于打车软件系统，要求系统能满足乘客、司机、第三方机构、业务检测人员、系统管理人员五种不同用户的功能需求，通过限制不同角色的使用权限，实现安全、快速、便捷的信息查询，打车预定，接单和支付结算功能，节省了人力物力资源，缓解了城市的达车压力。
* 用户、分析人员和测试人员要求系统满足需求分析阶段设想的系统特性，要求系统能正确可靠的运行，数据完整，系统可操作性强且用户体验良好。
* 系统集成人员要求系统满足需求分析阶段提出的性能指标，比如系统时间性能和空间性能上的要求。要求系统各个模块和层次之间尽量松耦合以方便各个部件之间的集成。
* 程序员要求系统软件各部分结构合理，方便进行修改、升级、测试和维护，要求系统接口设计合理，便于进行某些模块功能代码的重用，方便被当作模版框架来进行其他系统的二次开发。
* 系统工程师要求系统在规定时间内完成并交付，并在系统交付时对项目负责人进行相应的培训以便其正确使用系统。

## 3.4约束条件

1)Web服务器系统硬件配置需满足服务器能够高效稳定地与逆行。

需满足：

CPU:第三代以上双核

内存：8G以上

硬盘：1T以上

可配备磁带机等。

2）基于系统安全和保密性的考虑，系统的配置文件、数据存储文件等应进行加 密处理，采用国际通用的加密算法，防止意外泄露或恶意攻击。

3） 基于系统可靠性的考虑，系统的数据存储文件应进行冗余备份，比如磁盘冗 余阵列存储 RAID 等。

4）为了系统将来的可扩展性，系统硬件使用国际通用的硬件，不应使用具有针 对性的硬件。整个系统也应尽量减少各模块间的调用，尽量做到松耦合。

## 3.5设计遵循的标准

### 3.5.1 实用性原则

在打车软件系统的设计过程中，要充分考虑各个城市的实际情况进行业务逻辑的设计，以保证最终开发出来的软件系统切实可用，效率优异，能够良好的运行。切忌纸上谈兵，看似用了一些很炫酷的框架实则不具备良好的实用效果

### 3.5.2 稳定性原则

打车软件系统提供的 24×7小时不间断服务，要保证用户在任何时刻都能使用到我们的平台来他们所需的功能。因此，在设计时要考虑到任何可能导致的服务中断的原因。给用户提供稳定的服务。

### 3.5.3 复用性原则

组件复用是打车软件系统设计中必须遵守的原则。对可重用的组件进行统一包装，包括系统级的应用组件和应用级的服务组件。其中，系统级的应用组件主要考虑应用服务器和数据库的广泛适用性，对各种企业级服务进行重新包装，以便于提高架构的平台独立性。以使后续能够为不同的城市开发出适合他们实际情况的打车软件管理系统。

### 3.5.4 扩展性原则

打车软件系统必须具有良好地可扩展性。随着共享理念的兴起，越来越多的用户选择共享产品，因此，该打车软件管理系统必须在未来扩展到能够适配共享单车，共享汽车等，以满足用户的需求。打车软件系统必须能顺应这种业务的急剧扩展而快速进行相关的扩展开发。

### 3.5.5 安全性原则

打车软件系统带来了业务的扩展性，应用的延伸性、用户的方便性等优势的同时，也带来了风险的集中。打车软件系统架构的设计将在成熟稳定的硬件环境和应用软件基础上、通过网络、系统、应用、备份恢复、安全控制机制、运行管理监控等手段来保障系统的安全运行。

。

## 3.6非功能需求

### 3.6.1质量要求

#### 3.6.1.1性能

在CPU 1GHz且RAM 512M的安卓2.3系统上运行时，当系统有60%的空闲资源时：

* 启动速度在0.5s以内；
* 启动之后各项操作反应速度0.1s以内（快速的反应有利于增加用户体验度）；
* 软件正常运行RAM占用25M以内；
* 后台运行占用RAM资源少于6M，CPU少于1%。

#### 3.6.1.2可靠性

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，数据需要有至少两个个备份。

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。

#### 3.6.1.3可用性

因为此系统的使用用户是广大乘客和司机，其并不是专业人员，对于复杂系统操作不是很熟悉，为了让用户能够快速的掌握系统的使用方法，系统应当具有设计简洁的图形化界面，同时对于系统的操作应当标上文字已便于理解。同时系统的界面可能需要不断修改完善，我们要求采用MVC标准架构。同时系统应当具有一定的容错性，当用户做出一些非法操作时应当给出警告，并且当用户真的执行时不至于引起系统停机或数据混乱。系统必须24\*364运行，可使用性达到99%以上。

#### 3.6.1.4安全性

1. 系统提供用户登录功能(进行用户身份验证或使用微信号作为用户信息)，并且用户名和用户编号是唯一的。用户在登录界面上填写任意的用户名和用户密码(中文或英文)；系统提供登录过程中的出错处理机制和操作成功处理机制。
2. 在用户登录时会根据用户的用户名区分不同的用户类型（只可以查询还是既可以查询也可以预定），判断不同的用户身份。系统管理员会授予不同类型的用户不同的权限。
3. 系统对非法用户具有警告功能，例如：单用户表中不存在的用户企图登录系统，系统应该要求用户输入合法用户名和用户密码，并警告用户的操作。
4. 为了保障数据信息的安全性，应考虑防电磁辐射，重要的服务器硬件设备的电磁兼容性应满足国家相关标准的要求。
5. 操作系统的安全稳定是整个系统的核心，操作系统应具有防病毒措施。所以操作系统的安全水平应在不影响系统功能的情况下尽可能地考虑信息媒体的安全性。首先内部人员对数据的访问操作要进行控制,对用户、信息及操作进行分类授权,防止越权操作,避免数据遭到破坏。信息系统应具有防病毒措施。

6）建设一个成功的自动备份系统，来承担系统的数据备份，在数据库信息遭受意外丢失后可以及时恢复。

#### 3.6.1.5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。同时要求系统停机维护时间不超过8个小时

#### 3.6.1.6可移植性

本系统应当具有一定的跨平台和环境的能力。为了可以作为国家共享停车管理系统的子系统，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。

### 3.6.2工程需求

#### 3.6.2.1设计约束

|  |  |  |
| --- | --- | --- |
| 设计要素 | | 主要约束 |
| 用户端运行环境 | 操作系统 | Android/iOS |
| 数据库 | MySQL 14.0 及以上 |
| Web服务器 | WebLogic |
| 管理员端运行环境 | 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 |
| 开发环境支持 | 操作系统 | Linux Ubuntu 16.04 |
| 开发工具 | Android Studio、Eclipse |
| Web服务器 | WebLogic |
| CPU | 2.6 GHz Intel Core i9 X |
| 内存 | 128GB |

表1-设计约束表

#### 3.6.2.2逻辑数据库需求

要求使用关系型数据库来实现数据的存储与管理。由于SQL是开源免费的，所以从节约成本角度要求必须支持基于关系型数据库SQL Server，同时从未来可扩展的角度上也要求其支持DB2

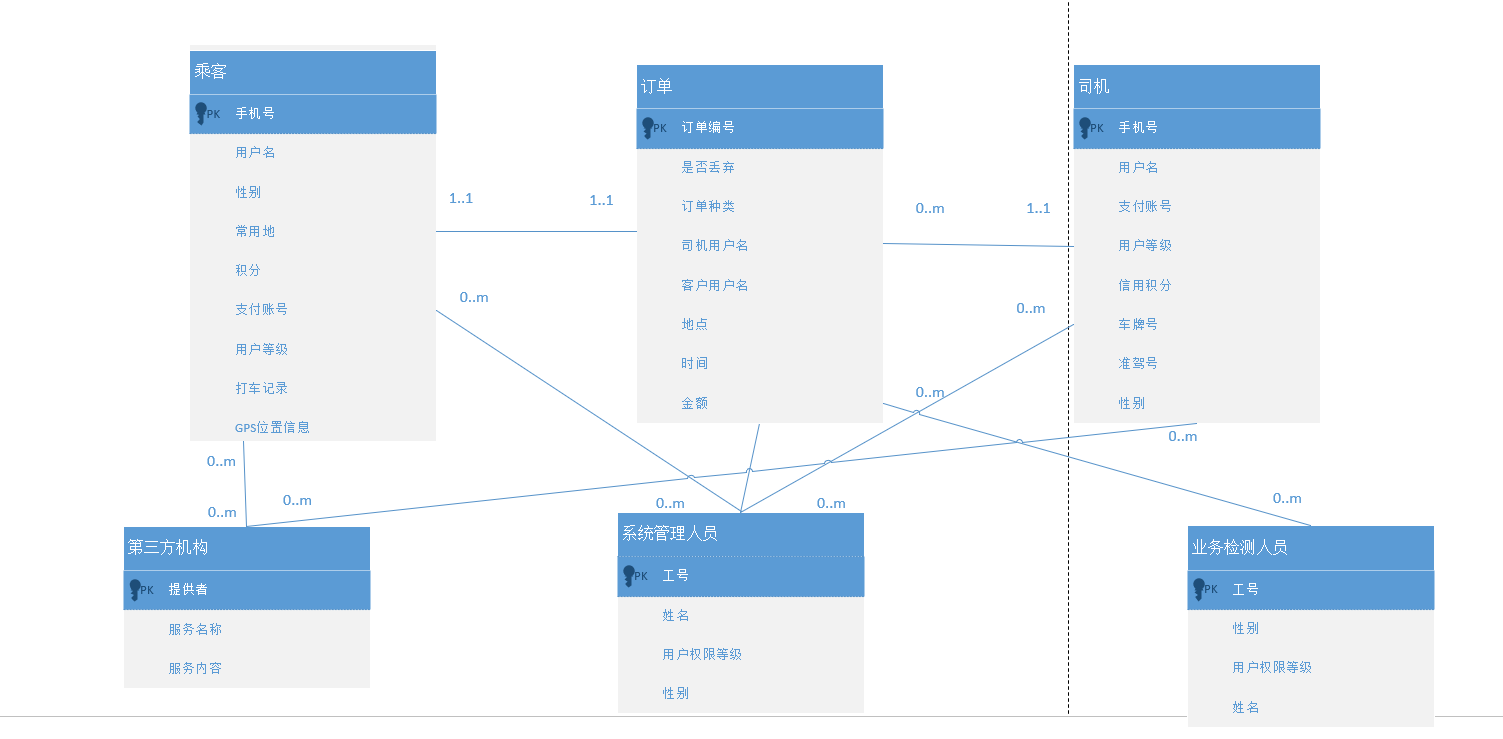
**

图9-TSS系统的E-R图

### 3.6.3其他需求

#### 3.6.3.1 界面需求

界面应有很好的交互体验，便于操作，对于没有计算机基础的司机来说可以很容易理解系统的各个功能。

#### 3.6.3.2 数据容量的需求

对于每个季度结束后的订单数据可以进行合并记录，留下汇总型数据，使得数据库有足够容量去处理新一季度的订单信息。

## 3.7风险

风险1：

如果系统使用人数在某个时候大幅增加，可能会导致服务器负载过重，而影响高并发时系统的性能和响应时间，甚至可能造成系统崩溃。

解决方案：添加备用的服务器，提升服务器对客户端请求的响应性能。

风险2：

如果系统的需求发生变更，可能会导致当前的体系结构无法再满足系统的需要。

解决方案：在进行体系结构设计时，尽量做到部件和模块之间的分离。这样，在将来需求分析发生变更的时候，能够尽量减少代码层面上的更改，而是从部件和模块层面上对系统进行改变。建立完善的需求变更管理体制，跟踪系统的需求变动轨迹，在确定需求的每一步中都要经过专人审核。

# 4.解决方案

## 4.1相关的体系结构模式

### 4.1.1三层B/S体系结构模式

B/S结构（Browser/Server，浏览器/服务器模式），是WEB兴起后的一种网络结构模式，WEB浏览器是客户端最主要的应用软件。这一模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用。

本系统客户端以手机APP的形式呈现，在APP页面嵌入浏览器，实现B/S体系结构模式。

本系统服务器安装MySQL数据库，客户端通过Web Server同数据库进行数据交互。

三层B/S体系结构是在B/S结构的基础上，在数据管理层和用户界面层增加了一层结构，称为中间件，使整个体系结构分为三层。

三层结构是伴随着中间件技术的成熟而兴起的，核心概念是利用中间件将应用分为表示层、业务逻辑层和数据存储层三个不同的处理层次，三个层次的划分是从逻辑上分的，具体的物理分法可以有多种组合。

表示层：客户端运行用户界面,向用户提供数据输入输出,响应用户动作并控制用户界面,把业务逻辑与用户界面分开,简化客户端的工作。

业务逻辑层：是系统实现业务逻辑与数据操作的核心部门，它的任务是接受用户的请求，首先执行扩展的应用程序并与数据库进行链接，通过SQL方式向数据库服务器提出数据处理申请，然后等到数据库服务器将数据处理的结果提交给Web服务器，再由Web服务器将结果传回给客户端。它提供所有的业务逻辑处理功能，整个系统中对数据库的操作都在这一层中完成。

数据层：数据库服务器运行数据引擎,负责数据存储。其任务是接受中间层对数据库操作的请求,实现对数据库的查询、修改、更新等功能,把运行结果返回给中间层。

图7-是三层B/S结构的示意图。

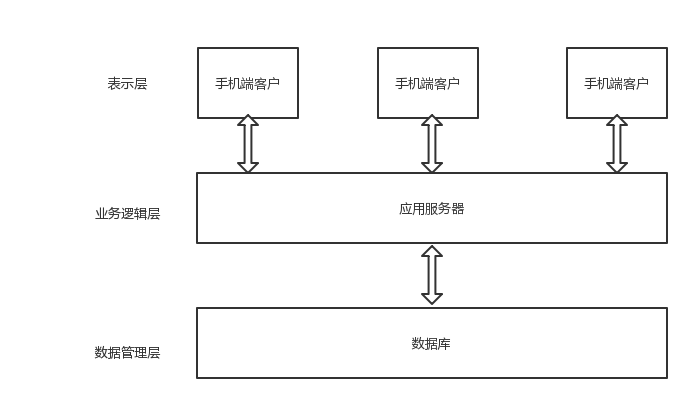


图10- 三层B/S体系结构示意图

### 4.1.2三层B/S体系结构的优点

三层B/S结构具有以下优点

1. 客户端无需安装，只需用浏览器即可访问，更加方便快捷。同时，由于客户端不再有业务逻辑，降低了步数和维护升级的成本以及实现的难度。客户端只需要提出请求，而相应的业务逻辑则都在业务逻辑层实现。客户端可以实现零维护，当对页面进行修改时，也不需要改动相应的业务逻辑。
2. BS架构可以直接放在广域网上，通过一定的权限控制实现多客户访问的目的，交互性较强。而且BS架构无需升级多个客户端，升级服务器即可。
3. 开发人员开发时只需要关注整体结构中的一层即可，大大降低了开发的难度。
4. 三层体系结构大大降低了层与层之间的依赖，当需要更新维护时，只需要用新的实现来替换原有层次的实现即可。同时分层还有利于各层逻辑的复用和整体的维护
5. 三层结构使系统具有更好地伸缩性。每个客户端不直接访问数据库，而是与业务逻辑层进行信息交互，再通过一个业务逻辑层与数据库进行数据的请求与获取，这样大大减少了与数据库建立连接资源消耗。
6. 业务逻辑层独立于客户端和数据库访问设计，提供了灵活性和可重用性。当药修改中间层的应用服务器时，不需要修改客户端的界面以及数据库访问层相应的代码，从而是得开发编程人员可以专注于核心业务逻辑的实现，大大简化了软件系统开发的难度。

### 4.1.3 三层B/S体系结构对质量属性的影响

表 2 三层B/S体系结构对质量属性的影响

|  |  |
| --- | --- |
| **质量属性** | **对质量的影响** |
| 性能 | 这种结构的性能已经得到了证明。关键的问题是要考虑每个服务器支持的并发线程数量、各层之间的连接速度以及数据传递的速度。对于分布式系统来说，降低了为完成每个需求所需的层与层之间的调用时间。 |
| 可靠性 | 通过中间层的缓冲，连接数据库的用户数大大减少。虽然增加了应用服务层，并不会使系统的性能和可靠性降低。因为在动态分布式计算系统中，客户端程序不必要确切指出应用服务的网络地址，如果应用服务器超负荷，通过统一的管理程序调度将请求转移到其他应用服务器上来消除瓶颈。 |
| 可用性 | B rowser／ Server 用户的界面都统一在浏览器上，浏览器易于使用、界面友好，不须再学习使用其它的软件，一劳永逸 的解决了用户的使用问题。 |
| 密安性 | B rowser／ Server 系统在客户机与数据库 服务器之间增加了一层 W eb 服务器，使两者不再直接相连，通过对中间层的用户编程可实现更加健全、灵活的安全机制。客户机无法直接对数据库操纵，有效地防止用户的非法 入侵。 |
| 可维护性 | B rowser／ Server 系统的三部分模块各自相对独立，其中一部分模块改变时， 其它模块不受影响，应用的增加、删减、更新不影响用户个数和执行环境，系统改进变 得非常容易，且可以用不同厂家的产品来组成性能更佳的系统。 |
| 可移植性 | Browser／ Server 系统使用HTML， HTML 是数据格式的一个开放标准，目前大多数流行的软件均 支持HTML，同时 MIME技术使得Browser可访问多种格式的文件。 |
| 可伸缩性 | Browser／ Server 所采用的TCP／ IP、 HTTP 等标准都是开放的、非专用的，是经过标准化组织所确定的而非单一厂商所制定，保证了其应用的通用性和跨平台性。同时，标准化使得B／ S模式 可直接接入Internet ，具有良好的扩展性、伸缩性，可从不同厂家选择设备和服务。  多个服务器可以有备份，多个服务运行在同一个或多个不同的服务器上，使体系结构的规模可以得到很好的提升。在实际中，数据管理层往往成为系统能力的瓶颈。 |

表 2 3层B/S结构对质量属性的影响

## 4.2体系结构概述

通过4.1部分的分析，可以知道三层B/S体系结构具有许多传统C/S体系结构不具备的优点。因此打车软件系统选择基于三层B/S结构设计的开发。

选用三层B/S模式的原因：

表示层：系统的客户端APP界面和Web界面友好，司机，乘客，业务检测人员，系统管理员可以在各自的用户界面对系统进行操作，系统响应用户操作，实现用户与系统的交互。

应用层：主要实现系统的核心功能，实现包括信息录入、查询、修改等业务逻辑。连接系统的前端和后端。

持久化层：该层主要根据用户操作与数据库进行信息交换和实现交互。

### 4.2.1 概念级体系结构

在概念级体系结构中，表现层负责向用户展示系统的页面及功能，应用层负责处理用户的请求并返回相应的数据，持久化层负责将用户的信息和数据保存在数据库中，并做必要的数据备份和恢复。结构示意图如下图所示：

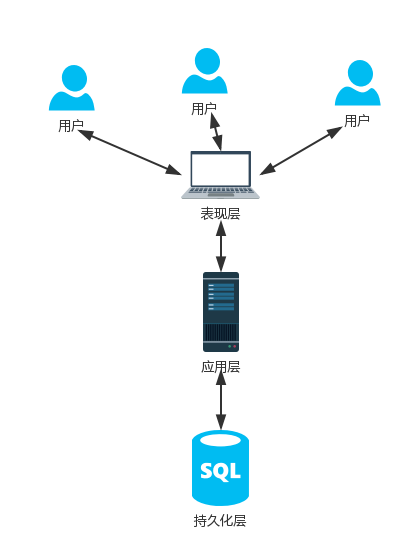


图 11 概念级体系结构示意图

### 4.2.2 模块级体系结构

系统的模块化设计是系统进行复用的关键。模块级体系结构反应了对软件代码实现时的期望。特别是对于程序规模较大的系统。下图是按系统的层次进行划分的系统模块化表示。体系结构的层次表达了每层的向上一个层面提供的功能和接口，以及需要使用下一层的功能。



图 12 模块级体系结构

### 4.2.3 运行级体系结构

每个用户都是一个独立的进程，可以独立地访问服务器发送请求并查看结果。

服务器为分布式架构，一台服务器作为Master服务器，管理Slave服务器的资源配置与负载均衡。

所有的Slave均访问同一个数据库，便于维护数据的一致性，同时定期对服务器进行备份，防止可能发送的故障和奔溃。

本打车软件系统的运行级体系结构如下图所示。

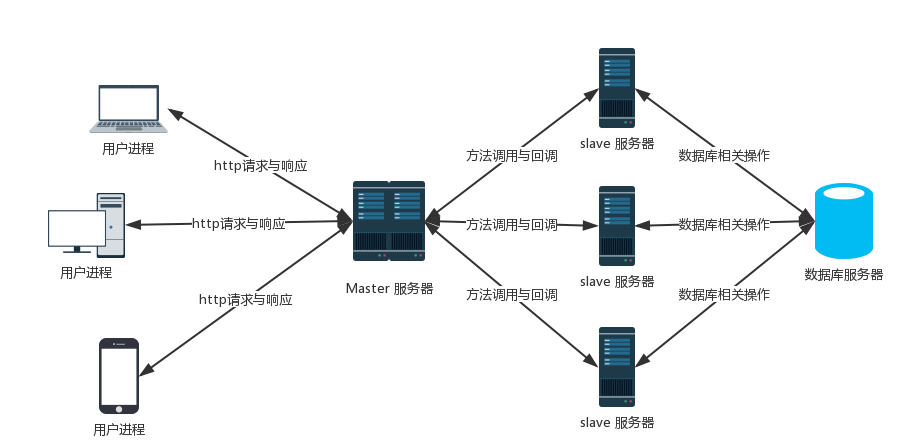


图 13 运行级体系结构示意图

### 4.2.4 代码级体系结构

代码体系结构为编程实现提供方便，在编程语言层面按模块、目录、文件、和库的形式组织源代码。如果说在模块化体系结构设计时不要过多地考虑所采用的编程语言的话，这个阶段必须考虑编程语言、开发工具和环境(例如，配置管理)，以及项目和企业的组织结构。从体系结构设计角度看，如何降低代码之间、子项目之间的相互依赖性，提升代码的可重用性等，成为体系结构设计阶段必须考虑的问题。本学生运动管理系统的代码级体系结构如下图所示。

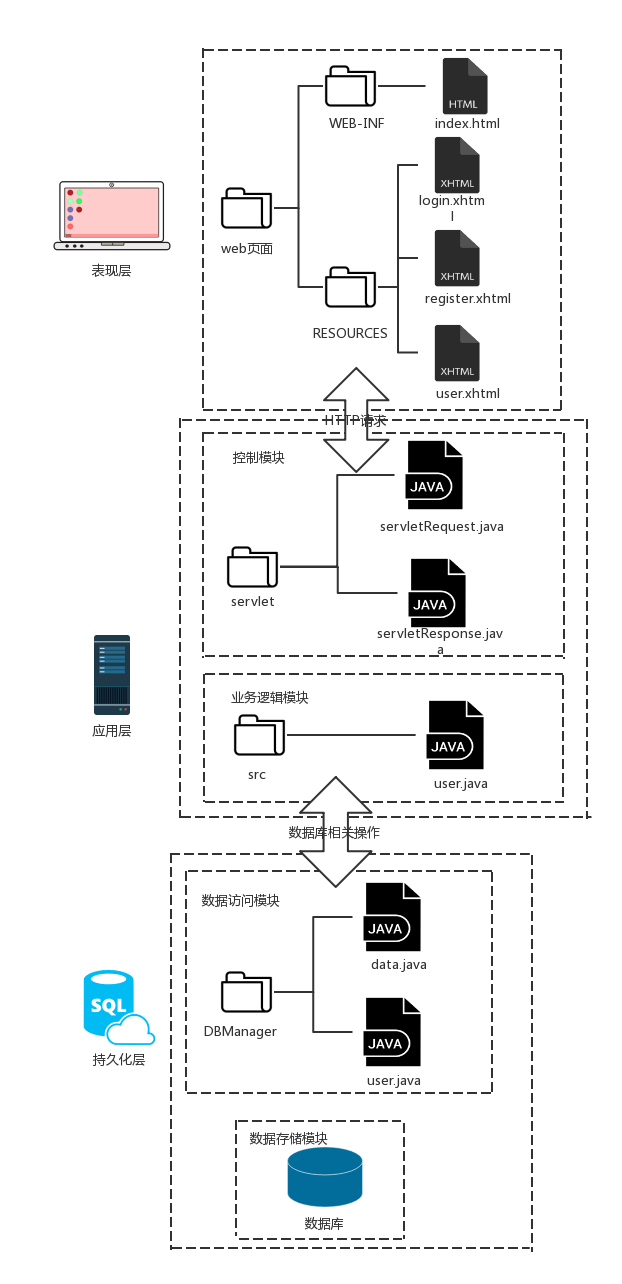


图 14 代码级体系结构示意图

## 4.3结构化视图

### 4.3.1概念级体系结构视图

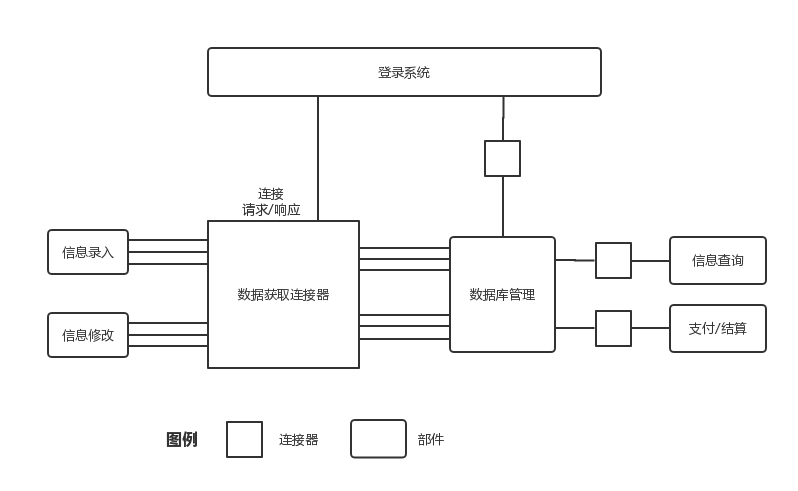


图15- TSS系统概念级体系结构视图

### 4.3.2逻辑视图

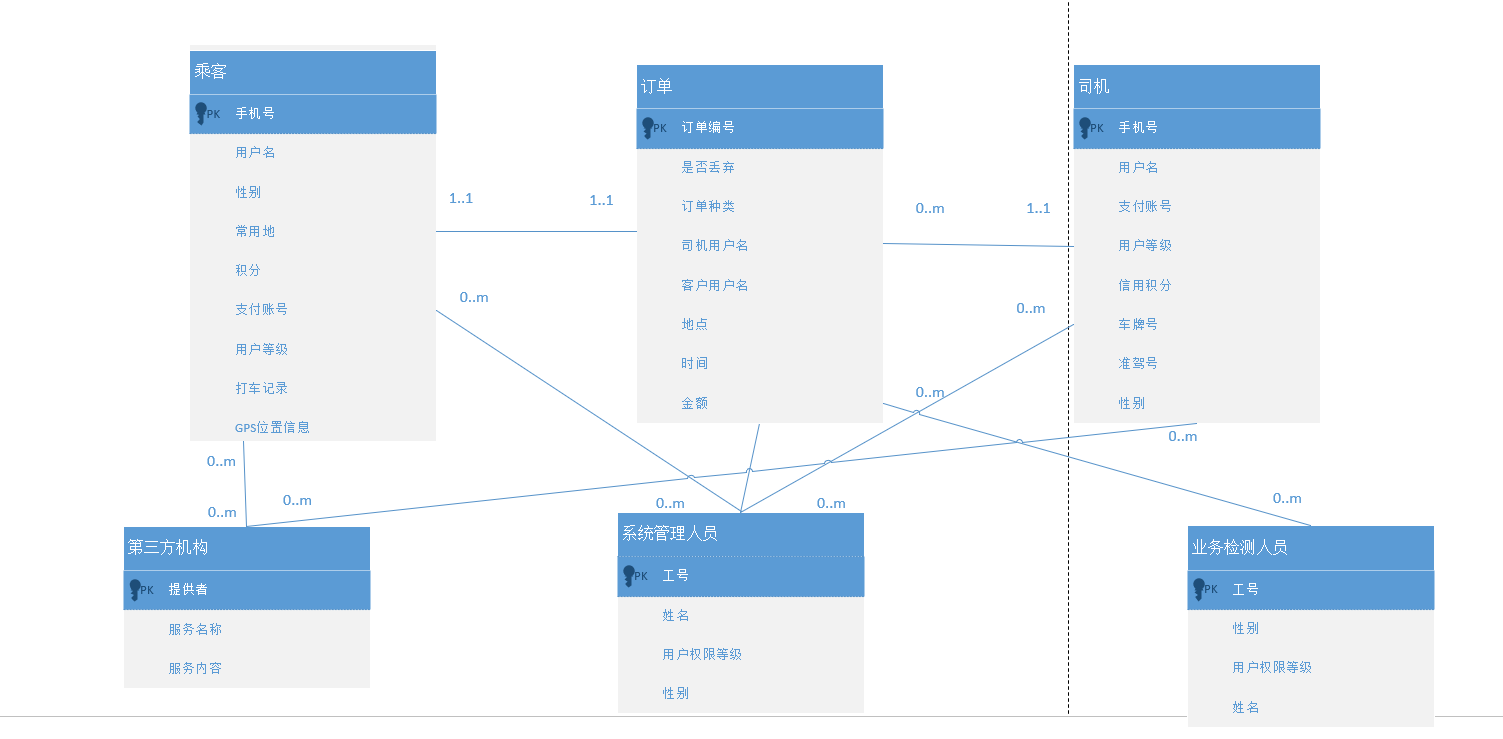
**

图16-TSS系统的逻辑结构视图

### 4.3.3运行级体系结构视图

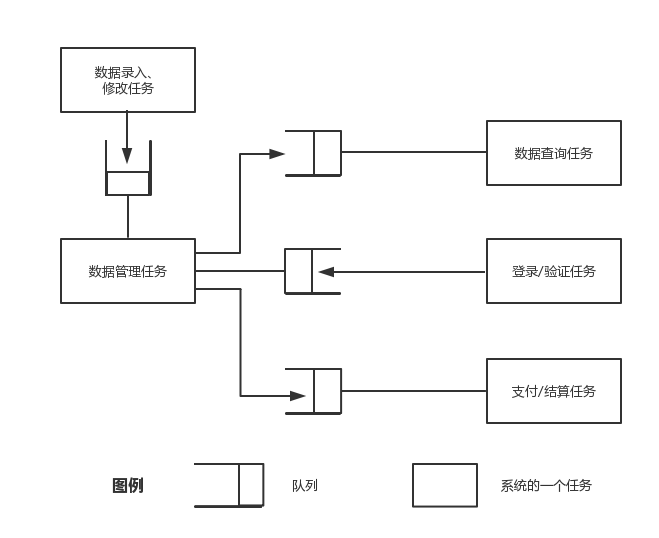


图17-TSS系统的运行级体系结构视图

# 5.系统的质量分析与评价

## 5.1场景分析

根据ATAM方法来对城市共享停车管理系统的体系架构进行分析。

### 5.1.1用例场景

场景1：

系统应该能够支持300万位用户的并发操作。此场景代表了用户所期望的系统性能和效率。

场景2：

用户登录系统的响应时间不超过5秒，用户对数据的录入、查询和修改的系统响应时间不超过5秒。此场景代表了用户所期望的系统性能和效率。

场景3：

系统应在用户输入目的地或者定位当前地理位置进行打车预约时，在2min内返回结果，并将结果存入换入。此场景代表了用户所期望的系统性能和效率。

场景4：

当服务器故障发生时，系统能够在3分钟内切换到备用服务器上继续工作，该场景代表了用户期望的可靠性。

场景5：

当数据库访问异常时或者用户进行错误操作时，均应有相应的错误提示，该场景代表了用户期望的可靠性。

场景6：

用户可方便快捷地使用系统，系统界面简洁美观，具有很好的人机交互性。此场景代表了用户所期望的系统易用性。

### 5.1.2增长性场景

场景1：

可以增加新的数据服务器，将用户通过APP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；

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5秒之内；

场景3：

用户使用APP访问系统时，可以在一段时间内保持登录状态，更方便用户的访问查询。

### 5.1.3探索性场景

场景1：

改进系统的可使用性，从98% 提升到到 99.999%；

场景2：

当全城350万司机同时使用该系统时，系统可承受并且反应时间控制在5秒内；

场景3：

对司机提交的举报信息进行数据分析，判定其真实度，确保真实可靠，并将反馈结果发送给系统管理员、举报者和被举报者。

## 5.2原型分析

使用效用树法进行原型分析，分析结果如下图所示

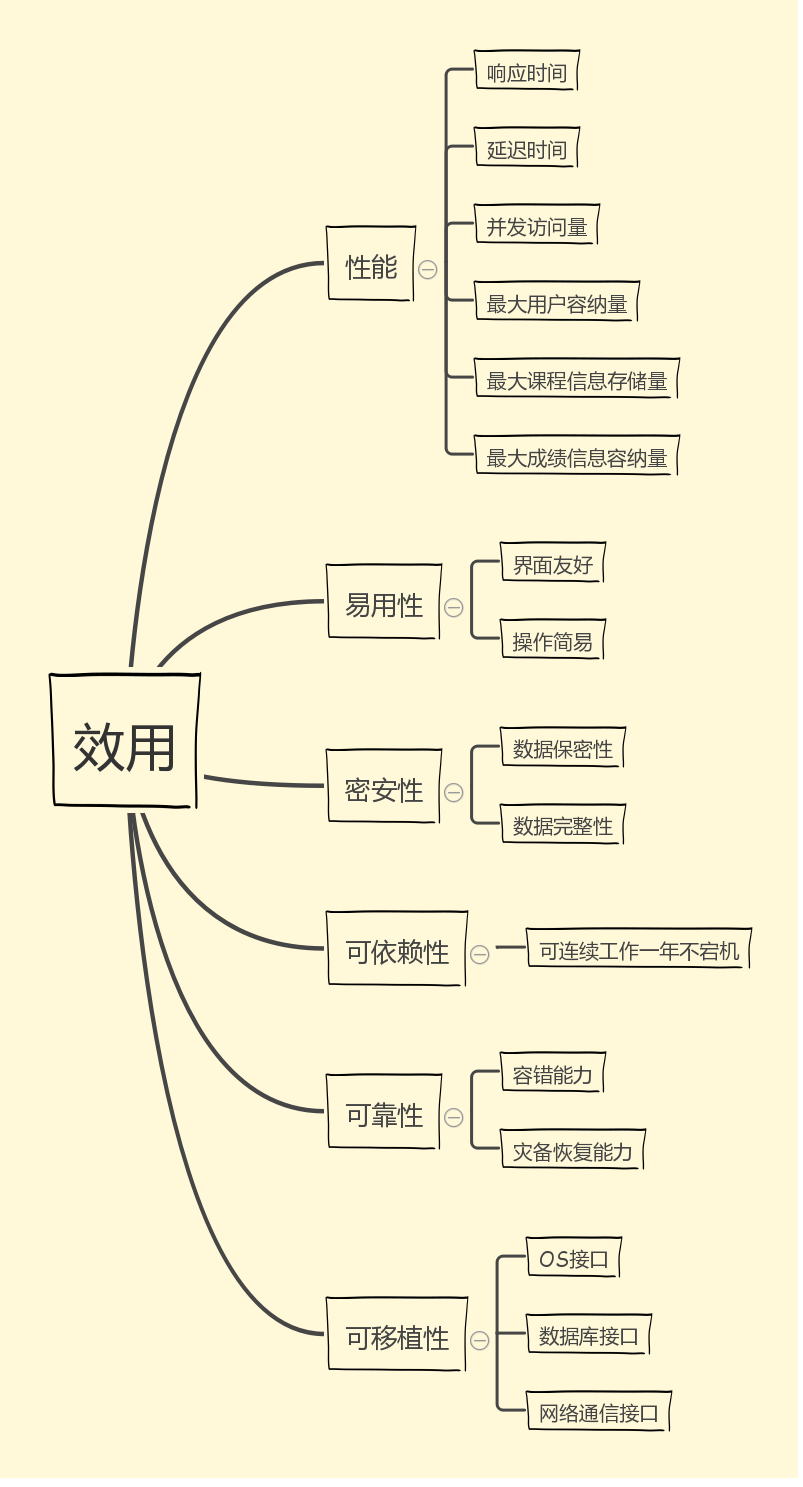


图18-TSS系统的效用树分析视图

## 5.3风险

在开发新的软件系统的过程中，由于存在许多不确定因素，软件开发失败的风险是客观存在的。在系统运行过程中，由于外部环境变化或者自身设计的不足，也可能产生各种风险。因此，风险分析对于软件项目管理是决定性的。

### 5.3.1 技术风险

在软件项目开发和建设的过程中，战略管理技术因素是一个非常重要的因素。项目组一定要本着项目的实际要求，选用合适、成熟的技术，千万不要无视项目的实际情况而选用一些虽然先进但并非项目所必须且自己又不熟悉的技术。如果项目所要求的技术项目成员不具备或掌握不够，则需要重点关注该风险因素。重大的技术风险包括：软件结构体系存在问题，使完成的软件产品未能实现项目预定目标;项目实施过程中才用全新技术，由于技术本身存在缺陷或对技术的在掌握不够深入，造成开发出的产品性能以及质量低劣。

预防这种风险的办法一般是经常和用户交流工作成果、品牌管理采用符合要求的开发流程、认真组织对产出物的检查和评审、计划和组织严格的独立测试等。软件质量的保证体系是软件开发成为可控制过程的基础，也是开发商和用户进行交流的基础和依据。所以制定卓有成效的软件质量监督体系，是任何软件开发组织必不可少的。

### 5.3.2 进度风险

软件的工期常常是制约软件项目的主要因素。软件项目工期估算是软件项目初期最困难的工作之一。很多情况下，软件用户对软件的需求是出于实际情况的压力，希望项目承担方尽快开发出软件来。在软件招标时，开发方为了尽可能争取到项目，对项目的进度承诺出已远远超出实际能做到的项目进度，使项目在开始时就存在严重的时间问题。软件开发组织在工期的压力下，往往放弃文档的编写与更新，结果在软件项目的晚期大量需要通过文档进行协调时，却拖累软件进度越来越慢。此外，由于用户配合问题、资源调配等问题也可能使软件项目不能在预定的时间内完成任务。软件项目过程中有自身的客观规律性，用户对软件项目的进度要求不能与软件开发过程的时间需要相矛盾。

对于这种风险解决方案一般是分阶段交付产品、增加项目监控的频度和力度、多运用可行的办法保证工作质量避免返工。在项目实施的时间进度管理上，需要充分考虑各种潜在因素，适当留有余地;任务分解要详细，便于考核;在执行过程中，应该强调项目按照进度执行的重要项，再考虑任何问题时，都要经保持进度作为先决条件;同时，合理利用赶工期及快速跟进等方法，充分利用资源。应该避免：某方面的人员没有到位，或者在多个项目的情况下某方面的人员中途被抽到其他项目，或身兼多个项目，或在别的项目中无法抽身投入本项目。为系统测试安排足够的时间，能使项目进度在改变之初就被发现，这对及时调整项目进度至关重要。在计划制定时就要确定项目总进度目标与分进度目标;在项目进展的全过程中，进行计划进度与实际进度的比较，及时发现偏离，及时采取措施纠正或者预防，协调项目参与人员之间的进度关系。

### 5.3.3 质量风险

任何软件项目实施过程中缺乏质量标准，或者忽略软件质量监督环节都将对软件的开发构成巨大的风险。有些项目，用户对软件质量有很高的要求，如果项目组成员同类型项目的开发经验不足，则需要密切关注项目的质量风险。矫正质量低下的不可接受的产品,需要比预期更多的测试、设计和实现工作。

预防这种风险的办法一般是经常和用户交流工作成果、品牌管理采用符合要求的开发流程、认真组织对产出物的检查和评审、计划和组织严格的独立测试等。软件质量的保证体系是软件开发成为可控制过程的基础，也是开发商和用户进行交流的基础和依据。所以制定卓有成效的软件质量监督体系，是任何软件开发组织必不可少的。