打车软件系统系统

体系结构设计文档

2018/12/16

姓名：金昊宸

学号：2017522045

班级：2017211502

# 修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **描述** | **作者** | **备注** |
| 2018/12/16 | 版本1.0 | 金昊宸 | 完成体系结构文档 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 文档批准

以下需求分析报告已经被以下机构人员批准并认可：

|  |  |  |  |
| --- | --- | --- | --- |
| **签字** | **打印姓名** | **标题** | **日期** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目录**

修订历史 2

文档批准 2

1. 介绍 7

**1.1 目的 7**

**1.2 项目相关信息 7**

**1.3 功能概述 7**

**1.4 术语定义 8**

**1.5 使用人群及阅读建议 9**

**1.6 参考文献 9**

2. 体系结构需求 9

**2.1 关键指标 9**

**2.1.1 体系结构 9**

**2.1.2基于本系统 10**

**2.2 体系结构用例 11**

**2.2.1 用户角色 11**

**2.2.2 用户功能用例图 13**

**2.2.2.1 用例图1-个人用户 13**

**2.2.2.2 用例图2-单位用户 14**

**2.2.2.3 用例图3-停车管理员 15**

**2.2.5 用例图4-系统管理员 16**

**2.2.3 具体功能用例图 17**

**2.2.3.1查询、预定车位用例图 17**

**2.2.3.2上传审核发布车位用例图 18**

**2.2.3.3车位收费缴费用例图 19**

**2.3 各相关方对体系结构的要求 19**

**2.3 约束条件 20**

**2.3.1 实现约束 20**

**2.3.2 设计约束 20**

**2.4 非功能需求 21**

**2.5.1 质量需求 21**

**2.5.1.1 可靠性 21**

**2.5.1.2 可使用性 21**

**2.5.1.3 密安性 22**

**2.5.1.4 可维护性 22**

**2.5.1.5 可移植性 22**

**2.5.1.6 性能 23**

**2.5.1.7 风险 23**

**2.5.2 工程需求 23**

**2.5.2.1 设计约束 23**

**2.5.2.2 逻辑数据库需求 24**

**2.5.3 其它需求 24**

**2.5.3.1 良好的人机交互能力 24**

**2.5.3.2 界面需求 25**

**2.5.3.3 数据容量需求 25**

3. 解决方案 25

**3.1 相关的体系结构模式 25**

**3.1.1 传统的信息管理系统开发模式(C/S) 25**

**3.1.2 B/S信息管理体统开发模式 26**

**3.1.3 体系结构选择 26**

**3.2 体系结构概述 27**

**3.3 结构化视图 27**

**3.3.1系统概念级体系结构设计 27**

**3.3.2模块级体系结构 29**

**3.3.2.1 登陆模块 30**

**3.3.2.2 车位信息管理模块 30**

**3.3.2.3 车位使用管理模块 31**

**3.3.2.4 用户信息管理模块 31**

**3.4 行为视图 32**

**3.4.1 逻辑视角 32**

**3.4.2 进程视角 33**

**3.4.2.1 个人用户进程 34**

**3.4.2.2 单位用户进程 35**

**3.4.2.3 停车管理员进程 36**

**3.4.3 实现视角 37**

**3.4.4 部署视角 38**

**3.5 实现问题 38**

4. 质量的分析与评价 39

**4.1 场景分析 39**

**4.1.1 用例场景分析 39**

**4.1.2 增长性场景分析 40**

**4.1.3 探索性场景分析 40**

**4.2 原型分析 41**

# 介绍

本文档是一个打车软件的体系结构分析报告。本文档主要通过对打车软件系统的质量需求进行分析，使用MySQL数据库管理系统进行数据信息的存储和访问，选择B/S结构为系统的体系结构模式，设计出该系统的总体体系结构，并进行分级。

## 目的

本文档通过对打车软件系统的需求进行分析，提供了一份从逻辑视角出发的体系结构文档。目的是为了使用户了解该软件系统的主要功能以及层次结构，便于用户与技术开发人员的理解与交流，熟练最终用户对系统的使用。

## 项目相关信息

本文档所分析的软件系统需求范围如下：

1. 该软件系统的名字为：打车软件系统(S Urban shared parking management system, 打车软件系统)。
2. 该系统不适用于以下情况：
3. 由于该系统只是打车软件系统的线上管理模块，所以该系统不包含的功能有：转账管理、物理设备设计以及导航系统接入设计。
4. 该系统属于在Web平台下的软件系统，因此在没有计算机网络且不提供位置权限许可的环境下，不能提供使用。
5. 该系统仅在地理位置比较精确的环境下使用。系统应用的目的是为设计出更好的打车软件，以更好地满足用户的需求。

## 功能概述

1. 乘客
2. 用户注册登录（第三方登录如：微信、QQ）
3. 填写个人信息（姓名（或昵称），通常打车地点，等）
4. 选择自己的出发地与目的地，系统会根据用户选择的出发地与目的地生成多条可供选择的到达路线，并根据路况生成大约金额，以供选择
5. 查看附近空闲司机状态（若周围空闲司机<2，则扩大选择范围以发现更多司机）
6. 若有紧急打车需求，系统会立即为乘客分配一名距离最近的司机，但金额会有所上涨
7. 查看司机基本信息（姓名，车牌号，车型，从业时间，行为分）
8. 选择一名司机或由系统推荐一名司机
9. 与司机联系，以详细确定上车时间，地点
10. 预约司机（至少提前1小时），与司机沟通（该司机有接送该乘客的历史），并确定预约地点与时间
11. 上车后：开始实时计费（计费与司机端同步）
12. 到达目的地后，结算金额，并缴费（支持多种缴费方式）
13. 若司机有任何不规范之处，乘客可立刻停止订单，并投诉司机
14. 司机
15. 用户注册登录（第三方登录如：微信、QQ）
16. 填写个人信息（姓名，年龄，车牌号，车型，等）
17. 可查看附近有打车需求的乘客的信息：出发地，目的地，路线，金额等
18. 选择改名乘客以作为该乘客司机，并立即与乘客联系
19. 若被系统安排为紧急打车，应立即与乘客联系，并赶往乘客所在地
20. 若为预约司机（至少提前1小时），与乘客联系，并确定预约地点与时间
21. 确认订单后，系统为司机导航至乘客所在地
22. 乘客与司机共同选择行进路线，并开始前往
23. 乘客上车后，开始实时计费（与乘客端同步）
24. 预约司机（提前1小时），与司机沟通（该司机有接送该乘客的历史）
25. 到达目的地后，结算金额，必须通过打车系统进行金额交易
26. 若乘客有立即停止订单需求，系统为司机立刻停止订单，并将此次失败订单的详细内容上传至服务器
27. 若司机的行为分<2，则收到任何乘客信息与打车需求，需等待时间，行为分缓慢增长后，才可再次拉客。

## 使用人群及阅读建议

本文档的主要内容共分为四部分，分别为：项目相关信息，体系结构需求，解决方案，系统的质量分析和评价。项目相关信息主要对系统的整体结构进行了大致的介绍；体系结构需求则描述了体系结构的相关信息；解决方案则提供了视图和解决的问题；系统的质量分析和评价则提供了场景和原型分析，以及本系统潜在的风险。

本文档面向多种读者对象：

（1）项目经理：项目经理可以根据该文档了解预期产品的功能，并据此进行系统设计、项目管理。

（2）设计员：对体系结构进行分析，并设计出系统，包括数据库的设计。

（3）程序员：配合《设计报告》，了解系统功能，编写《用户手册》。

（4）测试员：根据本文档编写测试用例，并对软件产品进行功能性测试和非功能性测试。

（5）用户：了解预期产品的功能和性能，并与分析人员一起对整个需求进行讨论和协商。

（6）其他人员：如单位领导等可以据此了解产品的功能和性能

## 参考文献

[1] 王安生.《软件工程化》[M]. 北京：清华大学出版社，2014:135-138.

[2] 王国华，薛福冰.《基于UML的高校科研管理系统分析与设计》[J]. 今日科技：2012(06):99-102.

# 体系结构需求

通过对该系统的质量和约束条件分析，该文档设计出以下的体系结构需求。

## 关键指标

### 2.1.1 体系结构

软件体系结构为软件系统提供了一个结构、行为和属性的高级抽象，由构成系统的元素描述、元素相互作用、指导元素集成的模式以及这些模式的约束组成。一个良性的软件体系结构应该有以下五个质量要素：

1. 体系结构应是适宜的。
2. 体系结构应是概念完整的。
3. 体系结构应是易于维护和升级的。
4. 体系结构应是便于移植的。
5. 体系结构应是理性化的。

这五个质量要素体现了体系结构作为早期设计决策对系统需求的支持、实现的约束，管理的组织。

这五个质量要素是比较抽象的，我们从需求、开发、项目管理三个角度细化体系结构的指标，可以划分出以下的表格：

表2‑1 关键指标分类

|  |  |
| --- | --- |
| 角度 | 包含指标 |
| 需求角度 | 功能性  可靠性  可用性  安全性 |
| 开发角度 | 可维护性  可扩展性  可移植性  可测试性  可重用性 |
| 项目管理角度 | 部件无关  风险性  可度量性 |

### 2.1.2基于本系统

本系统主要面对乘客、司机，可以明显减少乘客寻找司机的时间，以及司机对乘客订单的实时响应时间提升。同时，因为所存储的个人信息属于比较重要的信息，所以要求系统的稳定性和安全性得达到一定的高度，出错率降低到一定的水平。才能保证软件正常、快速、高效地使用，所以本系统在性能上应当具备以下关键指标：

1. 订单响应时间：

在同时有30000名乘客进行打车的情况下，系统响应时间不得大于5 s

在同时有10000名乘客进行打车的情况下，系统响应时间不得大于3 s

在同时有 5000名乘客进行打车的情况下，系统响应时间不得大于1 s

1. 订单响应时间：

在同时有30000名司机进行抢单的情况下，系统响应时间不得大于5 s

在同时有10000名司机进行抢单的情况下，系统响应时间不得大于3 s

在同时有 5000名司机进行抢单的情况下，系统响应时间不得大于1 s

1. 插入数据响应时间：

在同时有 30000 名用户上传数据的情况下，系统响应时间不得大于 3 s

在同时有 10000 名用户上传数据的情况下，系统响应时间不得大于 2 s

在同时有 5000 名用户上传数据的情况下，系统响应时间不得大于 1 s

1. 其他所有交互功能反应速度：

在系统的用户数目为8000 的情况下，响应时间不超过3 s

1. 对处理数据量的要求

对数据的处理量在G级别时，系统的响应时间应当在 5s 之内。

所以本系统在可靠性上应当：

1. 出错率：

在累计 100000 次服务请求的情况下，其出错率不得高于 3 次

在累计 1000000 次服务请求的情况下，其出错率不得高于 60 次

1. 稳定性

在 10000 用户的并发操作下，连续 5 个小时工作的情况下，系统不会出现停止服务的情况

1. 易恢复性

在停机进行恢复之后损失的数据不得大于1 h

## 体系结构用例

### 2.2.1 用户角色

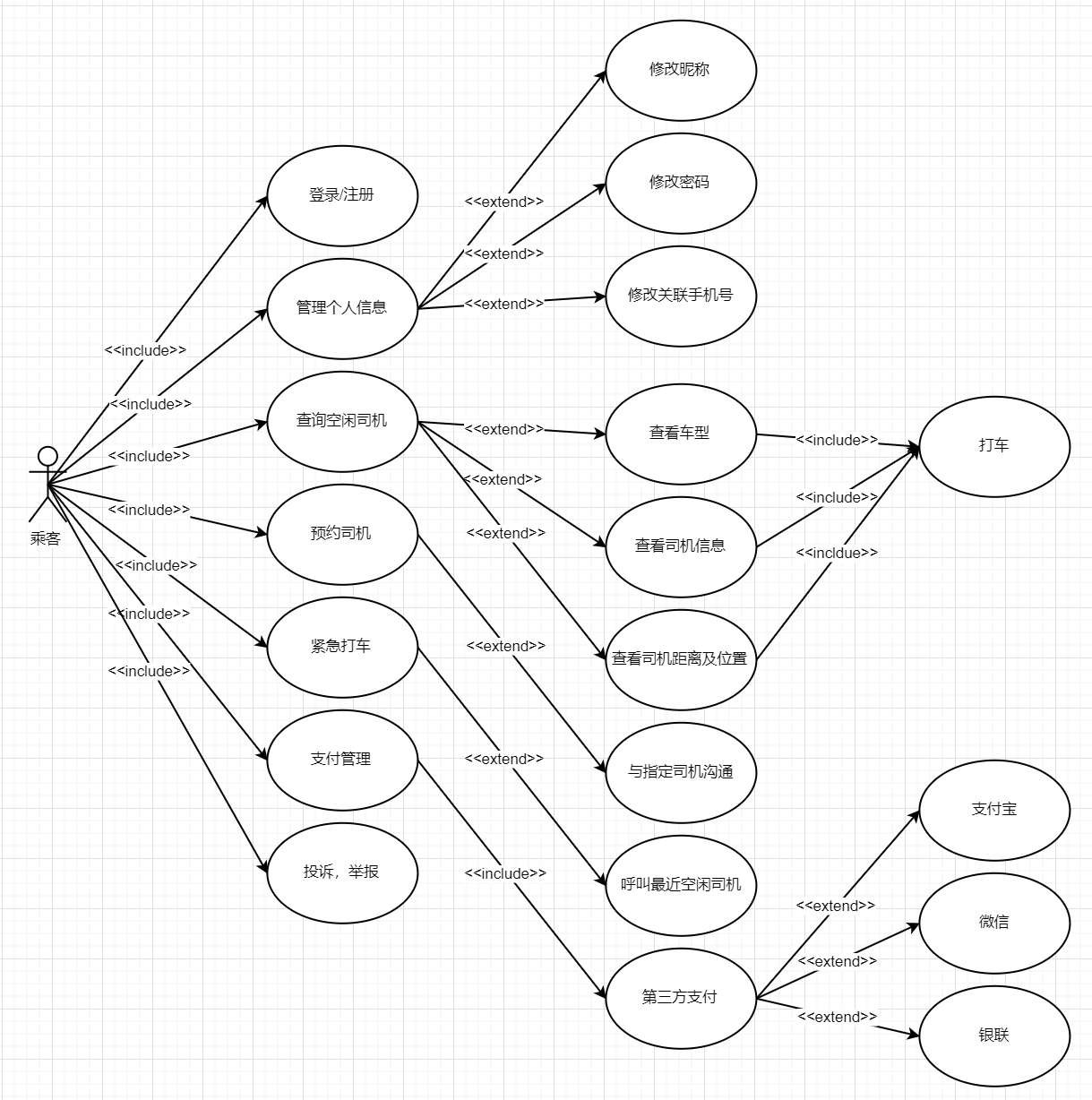
打车软件系统共有四种角色：乘客、司机、客服、管理员。下面本文将针对四种用户画出它们的用例图。

表2‑2 角色职责描述

|  |  |
| --- | --- |
| 角色 | 职责描述 |
| 乘客 | 登陆/注册，管理个人信息，查询空闲司机，预约司机，紧急打车，支付管理，投诉，举报 |
| 司机 | 登陆/注册，管理个人信息，查询需求乘客，预约司机，紧急打车，提现管理 |
| 客服 | 解决用户问题 |
| 管理员 | 处理系统问题，管理数据 |

### 2.2.2 用户功能用例图

2.2.2.1 乘客用例图

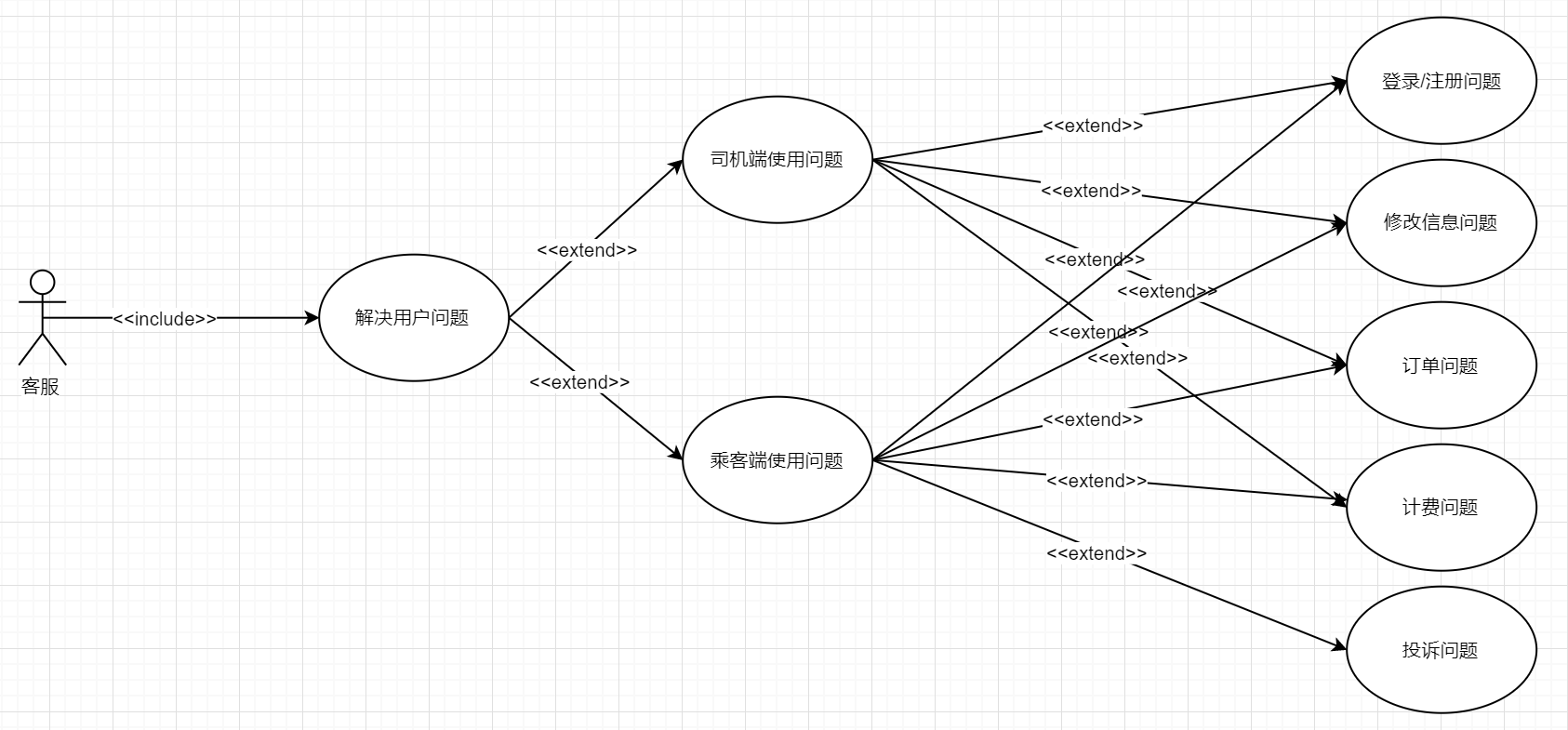


2.2.2.2 司机用例图

图片包含 文字

描述已自动生成

2.2.2.3 客服用例图



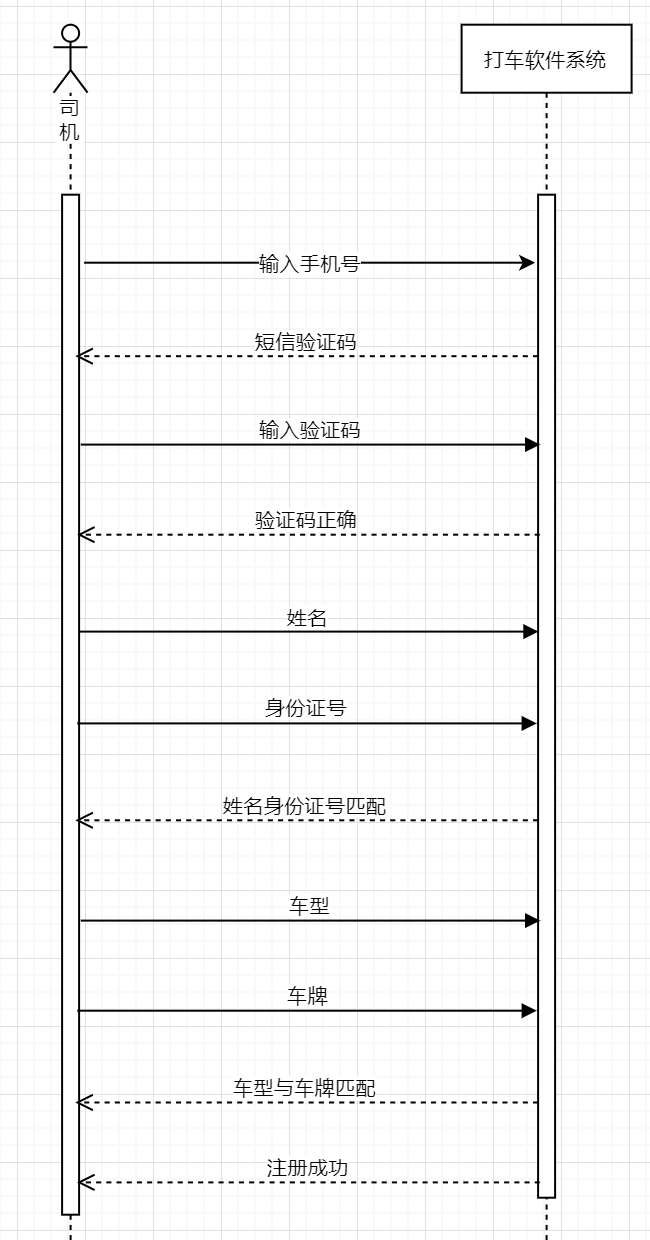
### 2.2.5 管理员用例图

图片包含 文字, 地图

描述已自动生成

### 2.2.3 主要使用流程

司机注册UML顺序图

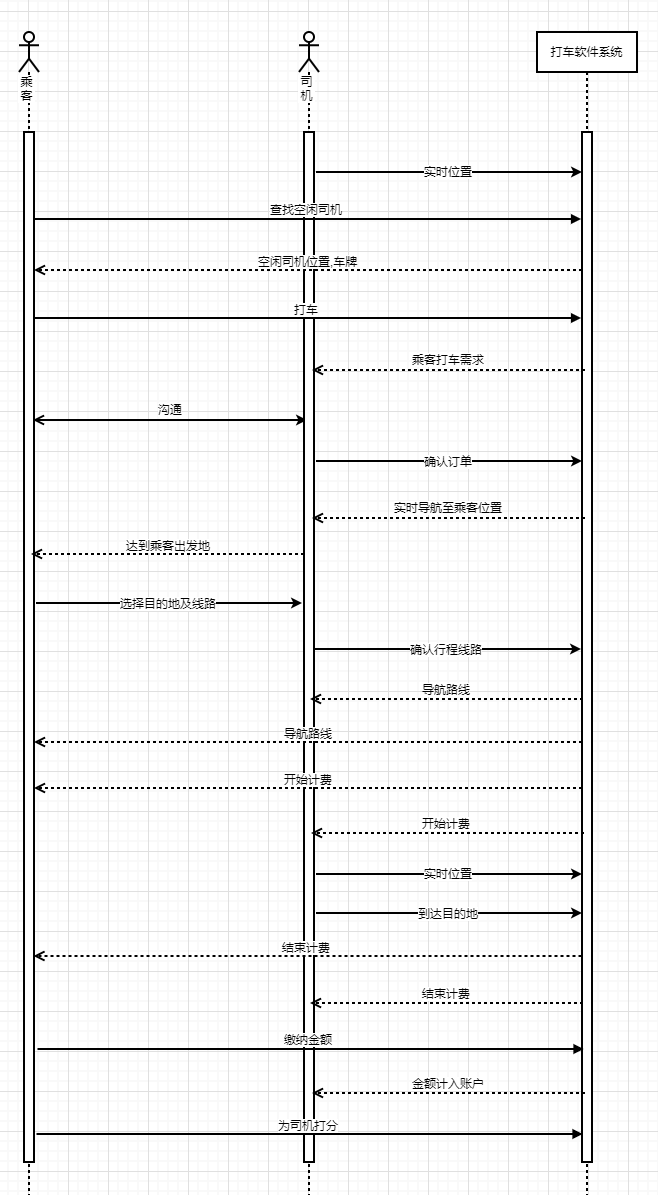


乘客注册UML顺序图

图片包含 文字

描述已自动生成

乘客打车UML顺序图



## 2.3 各相关方对体系结构的要求

在SGM系统的开发的过程中，存在的相关方有如下5类：

1. 最终用户、设计者，其关注点是功能
2. 程序员，其关注点是软件管理
3. 用户、分析人员和测试人员，其关注点是特性
4. 系统集成人员，其关注点是性能、伸缩性、吞吐量
5. 系统工程师，其关注点是系统拓扑；交付、安装、交流

根据以上的分析，可以看出：

最终用户对体系结构的要求是：功能清晰、完善；特性完备；

程序员对体系结构的要求是：层次清晰，易于软件的管理；

分析人员和测试人员对体系结构的要求是：测试模块鲁棒性高，体现出软件测试要考虑到的问题；

系统集成人员对体系结构的要求是：考虑到系统性能和约束条件对体系结构的影响；

系统工程师对体系结构的要求是：体系结构清晰，系统拓扑结构逻辑正确；包含系统最终交付安装以及跟踪的设计。

## 约束条件

### 2.3.1 实现约束

本系统在进行代码编写的时候要求给出详细的代码注释，每一个功能函数都要给出函数的执行功能，输入和返回值，便于后期进行代码的重用。系统要求应用Java语言进行编写，并且所有变量的命名规范符合Java语言命名规范。系统所有网页HTML语言都要符合XHTML 5.0标准。系统要求对数据信息进行加密，要求使用当下市面上已有系统的数据加密算法。

2.3.2 设计约束

系统的最终数据呈现上，一定要求带给用户良好的视觉体验性。约束系统的数据展示清晰、图表美观、相应的功能组件设计得体。

这就要求系统最开始的时候从用户的需求出发，设计出相应的应用逻辑可以满足用户的需求，之后根据业务逻辑设计出相应的业务模型开始进行系统的构建，构建的过程之中根据从数据库中获取到的数生成相应视图呈现在网页界面上，呈现给最终用户。

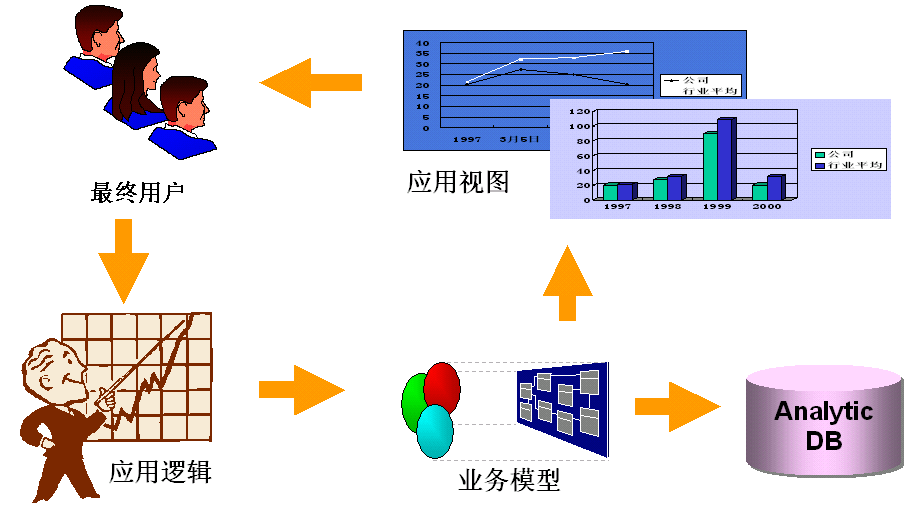


图2-9 设计约束图

## 非功能需求

所谓非功能性需求，是指软件产品为满足用户业务需求而必须具有且除功能需求以外的特性[[2]](#_1.4_参考文献_1)。非功能性需求在需求分析阶段常常被忽略或没有被足够重视。软件产品的非功能性需求包括系统的质量需求和工程需求。对于本系统，主要通过以下三个方面对本系统的非功能性需求进行描述。

### 2.5.1 质量需求

2.5.1.1 可靠性

可靠性是指产品在规定的条件下，规定的时间内，完成规定功能的能力。该系统的可靠性需求具体体现在系统能较长时间下稳定运行。同时，该系统需要具备一定的故障恢复能力，即有一定的容错能力。当用户的操作不当引起某些故障时，或者是由于操作系统或者网络发生故障时，系统需要具备一定的故障恢复能力。

1. 本系统测试时间至少为100小时，测试参与用户至少为200人。本系统应该避免因为故障而导致的失效的能力，在进行测试的过程中，本系统应该尽可能遍历所有的可能故障情况，保证系统的成熟。
2. 本系统在发生硬件或者软件异常时，应该具有依然具有服务能力。服务能力虽然下降，但不会达到系统崩溃的程度级别。
3. 本系统在发生停机故障之后，应该保证十分钟之内可以修复并且正常运行。

2.5.1.2 可使用性

1. 在接收了用户错误的数据输入之后，本系统能不死机、不崩溃，依然正常工作。本系统应该可以进行错误识别并且给出用户相应的提示信息。
2. 系统界面美观，吸引使用者。本系统的界面设计应该在保证基本功能的基础上做到布局美观友好，系统界面和市场面上大部分管理系统界面类似，用户的体验性在本系统的界面设计上因该充分提高。
3. 本系统易于学习。本系统应该保证新用户在初次使用时，在不超过3分钟的学习时间之内便可以明白整个系统的基本功能，在10分钟的学习时间之内该用户应该熟练掌握系统的全部功能。
4. 用户使用本系统工作效率得到提高。本系统应该加快了学生的成绩查询效率和教师的成绩录入效率，并且本系统应该提高了教务管理人员的成绩数据保存和发布效率。
5. 系统提供相应的操作指南文档。本系统提应该提供给使用者相应的文档使用手册，便于使用者获得系统功能的详细解读和使用。

2.5.1.3 密安性

1. 本系统应该对使用者的权限进行控制。本应同应该使用学生学号、教员教职工号、家长校方注册号进行系统的登录，不同的类型的使用者应该具有不同的系统使用权限。本系统通关系统管理员管理实现对于不同角色开放不同功能的行为，来防止不合法的操作的产生。
2. 本系统对于重要数据应该进行数据加密。初次登录系统时，应该输入相应的用户名和密码，对于密码的存储应该采取一定的加密措施，防止因为管理员不适当的信息泄露而产生不可挽回的错误。
3. 系统数据定期备份。为了本成绩管理系统的长远考虑，系统数据应该进行相应的备份，防止在系统停止工作时能保证进行数据恢复。

2.5.1.4 可维护性

软件的可维护性是指改进软件的难易程度。该系统的结构、接口、功能以及内部过程在开发以及跟踪阶段，容易被维护人员理解。同时，该系统有良好的测试和诊断系统错误的功能。当系统应用于不同学校的学生管理系统之下时，应该具备良好的适应性。不需要通过大幅度的接口与内部过程修改，就能使用户进行使用。

2.5.1.5 可移植性

1. 本系统应该保证在市面上常用的系统环境下便于进行安装使用，不会因为环境的不同而导致系统的崩溃或者时系统功能的缺失。
2. 本系统的接口设计合理，便于进行系统的改造和重用，应该方便被当作模板框架来进行相应其他系统的二次开发。
3. 本系统在和其他系统在同一个平台下面进行运行的时候，不会产生冲突，各自有条不紊的运行。

2.5.1.6 性能

1. 本系统应该存储至少2万人的信息的记录，用于不同身份背景的人进行系统的使用。
2. 在系统因为访问人数过多而发生宕机时，本系统应该在10分钟之内可以进行恢复。
3. 系统的一般操作相应时间应该不超过2秒，系统涉及数据连接的操作相应时间应该不超过5秒。

2.5.1.7 风险

1. 系统市场风险。本系统的市场风险主要存在于一下几个方面：系统对用户是否发挥了效率上的便利性；使用用户是否会接受并采用本系统提供的相关服务；是否存在其他比本系统更快、更好的打车软件系统等。
2. 系统财务风险。本系统的财务风险来源于这几个方面：开发组织是否有能力承担这一系统项目的开发；项目关系人的财务预算是否合适，防止在系统开发的过程中出现预算超支的现象；该系统能否给学校带来相关利益等。
3. 技术风险。本系统的技术风险来源于：该项目在技术上是否可行、成熟；软件、硬件和网络功能是否合适；相应的技术是否满足项目目标；在产品生产出来之前，该技术是否会过时；在系统开发的过程之中过于热衷前沿的先进技术，从而使项目冒太大的技术风险。
4. 系统集成风险。本系统集成需要根据用户的具体信息，系统地设计和实现系统。本系统集成的关键是要将计算机信息领域内不同方面的技术揉合在一起，为用户提供一个出色的整体解决方案,
5. 系统管理风险。本系统要求在校方进行引入之后，安排相关人员进行后续管理维护。重点管理维护工作包括：无用数据清理、数据定期备份、用户信息加密管理。

### 2.5.2 工程需求

2.5.2.1 设计约束

针对项目的基本要素进行分析，可以得到下面的设计约束表格。

表2-4 项目设计约束条件

|  |  |  |
| --- | --- | --- |
| 设计要素 | | 主要约束 |
| 运行环境软件 | 操作系统 | Windows 7/Linux 7.0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/Linux |
| 浏览器 | Chrome/Firefox/Opera/Safari |
| 开发环境支持 | 操作系统 | Mac10.15 |
| 开发工具 | Myeclipse |
| Web服务器 | Weblogic |
| CPU | 2.4 GHz Intel Core i7 |
| 内存 | 128GB |

2.5.2.2 逻辑数据库需求

数据库需求设计分为两部分：概念结构设计和逻辑结构设计。概念结构设计指的是画出E-R模型。将概念结构进一步转化为某一DBMS所支持的数据模型，然后根据逻辑设计的准则、数据的语义约束、规范化理论等对数据模型进行适当的调整和优化，形成合理的全局逻辑结构，并设计出用户子模式。这就是数据库逻辑设计所要完成的任务。

### 2.5.3 其它需求

2.5.3.1 良好的人机交互能力

由于系统的用户较多，且操作习惯、年龄阶段、接受事物能力都各不相同，所以应要求该系统具备良好的人机交互能力。系统提供的各种功能能够便于用户理解，操作简单。

2.5.3.2 界面需求

系统应该具有可以适用于多平台、多浏览器的界面。

2.5.3.3 数据容量需求

由于用户量较大，该系统应该能够支持大数据分析与处理。

# 3. 解决方案

## 3.1 相关的体系结构模式

本系统采用分层结构模式。分层系统体系结构有一下有点：

1. 支持基于抽象程度递增的系统设计。这允许设计者可以将一个复杂系统设计按递增的步骤进行分解。
2. 支持扩充。因为每层之多和与之相邻的上层和下层交互，所以，改变某层的功能最多只会影响与之相邻的其它两层。
3. 支持重用。与抽象数据类型一样，只要对相邻层提供同样的接口，每层可以有很多不同的实现方法，并且这些方法可以相互替代。

下面将对一些典型的层次体系结构模式进行介绍和分析。

### 3.1.1 传统的信息管理系统开发模式(C/S)

传统的基于客户机/服务器(Client/Server)模式的管理信息系统产生于上世纪70年代并创立了一种分布式应用标准。到目前为止大多数的企业仍然在使用此种模式的信息管理系统，它为企业信息管理系统的共享集成和分布式应用做出了巨大的贡献，但是传统信息管理系统的缺点也是比较明显的：

1. 安装、升级、维护工作量大。每个客户机安装一套应用软件，一旦出现一点微小的修改或版本升级就需要对每台机重装一次，这对客户端较多的大型系统来说，既费时有提高了软件成本。
2. 数据一致性差。在C/S结构软件的解决方案中，不同地域的用户需要安装区域服务器，然后再进行数据同步。局部网络故障或者人为因素都有可能造成个别数据库不能同步，即使同步上来，各服务器也不是一个时点上的数据，数据很难做到一致性。
3. 系统生命周期短，移植困难，升级麻烦。

### 3.1.2 B/S信息管理体统开发模式

B/S结构，即浏览器/服务器(Browser/Server)结构，是随着Internet技术的发展，对C/S结构的改进。在B/S结构中，客户端只需要安装Web浏览器软件，主要事务逻辑由服务器实现，用户通过浏览器向分布在网络上的服务器发出请求，服务器负责对客户端的请求进行分析、处理，并将反馈信息返回到浏览器。如下图所示。

因此可以看出，B/S体系结构简化了客户端的工作，在该模式中，客户端只需要装上操作系统和浏览器，就可以在服务器上进行所需要的软件开发、维护等工作，极大地提高了工作效率。

与传统的C/S两层结构相比，B/S应用体系的主要优势包括：

1. 升级维护方便。
2. 客户端负载轻。在客户端只需要标准的Web浏览器作为接入方式，而在各种平台上均有专门厂商提供的浏览器，从而简化了客户端配置。
3. 资源访问简单。系统信息和资源以HTML标准进行组织，通过统一资源定位(URL)方式进行访问，并且访问点单一，允许在不同的地方访问数据库。
4. 安全性高，应用逻辑和数据库由服务器实现，对客户端是非透明的，保证了系统的安全可靠性。

从上述对C/S和B/S开发模式的分析可以看出，基于B/S模式的信息管理系统系统将会成为信息管理系统发展的必然趋势。

### 3.1.3 体系结构选择

综合以上对C/S和B/S两种开发模式的分析，可以看出B/S模式具有适用范围广、异构和开放性强、平台技术稳定的特点。打车软件系统需要网上办公和对信息进行高效管理，结合以上分析，本系统结合B/S体系结构模式和MVC体系结构模式设计而成。B/S架构主要体现在使用者借助浏览器登录软件管理系统，开始进行相关数据的反馈，根据客户相关的业务逻辑，数据将会从数据库中取出同样显示在网页上面。MVC主要用于编写代码时的约束，使得设计思路清晰，便于未来系统的修改和重新使用。

## 3.2 体系结构概述

本章节将体系结构进行分层次描述，共分成了以下3种：

1. 概念级体系结构设计：MVC体系架构描述，SGMS系统体系结构
2. 模块级体系结构设计
3. 逻辑级体系功能结构设计

前两种采用结构化视图，后一种采用行为视图进行描述

## 3.3 结构化视图

### 3.3.1系统概念级体系结构设计

MVC体系架构主要体现在打车软件系统，一共分为三层：表示层、应用层、服务层。这三层最终都是基于B/S架构的浏览器和服务器实现的。这三层主要用于编写代码时的约束，使得设计思路清晰，便于未来系统的修改和重新使用。表3-1展示了MVC和B/S组织模型，图3-1显示了MVC架构图。

表3-1 MVC和B/S组织模型

|  |  |  |
| --- | --- | --- |
| MVC层次 | 理解性抽象 | 对应B/S架构 |
| 表示层(V) | 用户界面层 | 浏览器获取数据 |
| 应用层© | 用户身份验证层、信息检索和修改层 | 浏览器显示数据，服务器传送数据 |
| 服务层(M) | 数据库层 | 服务器计算数据 |

用户界面层是和用户最直接接触的一个层次。这个层次通过网页界面进行相应的展示，用户可以通过自己的浏览器界面进行相应的访问和查看。

用户身份验证层是进行用户身份类型的检查。对于不同身份的人系统提供的功能是不同的。在实际中，采用识别出相应的手机号、工号来完成身份的验证，因此用户身份验证层是不可缺少的。

信息检索和修改层主要针对的是车位信息的查询，上传审核发布车位信息，收集更新车位信息等功能。在这一层根据上一层获得的身份类型返回不同的结果。

数据库层主要完成数据信息的存储和备份，根据上一层获取的条件返回相应的查询结果，并且对数据进行相应的权限管理，防止非管理人员进行相应的数据访问。

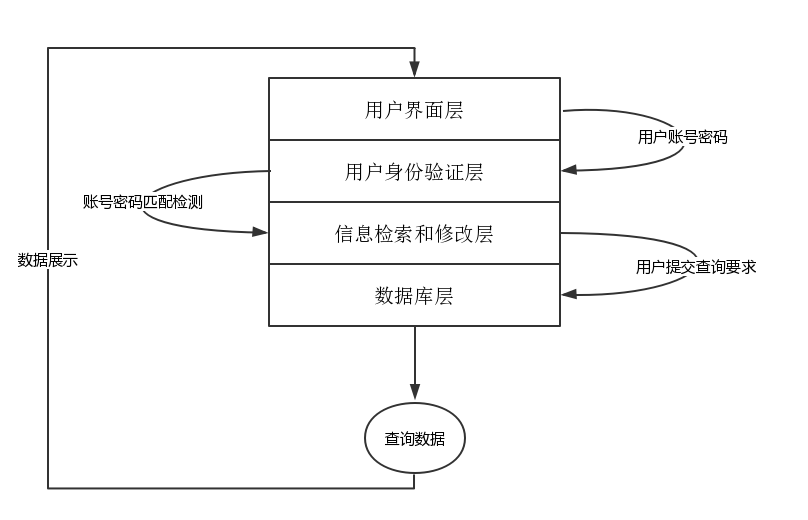


图3-1 MVC体系架构图

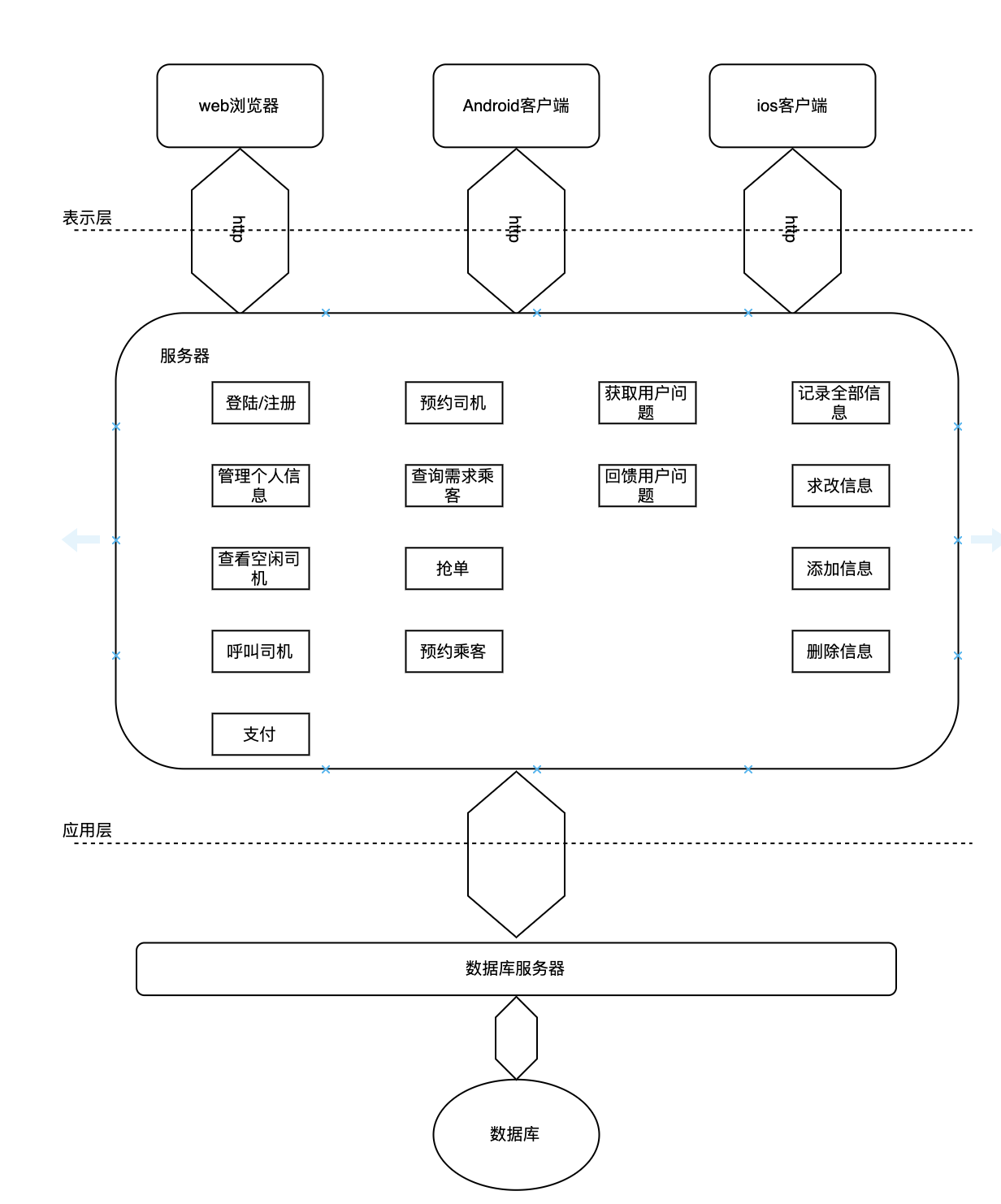


图3-2 三层B/S模式的系统体系结构模型

### 3.3.2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。模块级体系结构反映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。模块化表达有两种方式；(1) 将系统按功能从逻辑上分解为系统、模块以及程序单元；

(2) 按系统的层次进行划分。模块化分解方式很容易区分出概要设计和详细设计两个阶段。

本说明书将系统按功能从逻辑上分解，细化功能结构模块。

3.3.2.1 登陆模块

登陆系统主要为乘客与司机提供登陆功能。用户在浏览器中输入账号密码时，客户端会读入用户信息，并将数据信息传送到服务器，服务器将输入数据与后台数据库内容进行比对来查看用户名与密码是否匹配，将结果传送给浏览器向用户输出。

3.3.2.2 查询功能模块

查询功能主要为乘客和司机提供查询功能。用户选择要查询的信息，库护短获取用户选择，并请求服务器响应。服务器将处理数据，并将结果传送给浏览器，客户端通过该数据响应用户请求。

3.3.2.3 打车功能模块

打车功能主要完成乘客的打车请求功能。乘客与司机以达成打车协议，客户端取用户选择，并请求服务器响应，服务器将处理数据，并将结果传送给客户端，客户端通过该数据响应用户请求。

3.3.2.4 用户信息管理模块

信息管理模块主要是乘客和司机管理自身个人信息。该模块的主要功能是实现用户信息以及权限的修改。客户端读入用户要修改的信息并将数据与请求发送给服务器，服务器根据浏览器传送的信息更新数据库，并返回结果。

## 3.4 行为视图

本说明书使用 Philippe Kruchten提出的从不同角度勾画系统的蓝图，即“4+1”视图模型。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进行观察，分析其影响系统的上下文和商业目标情况.

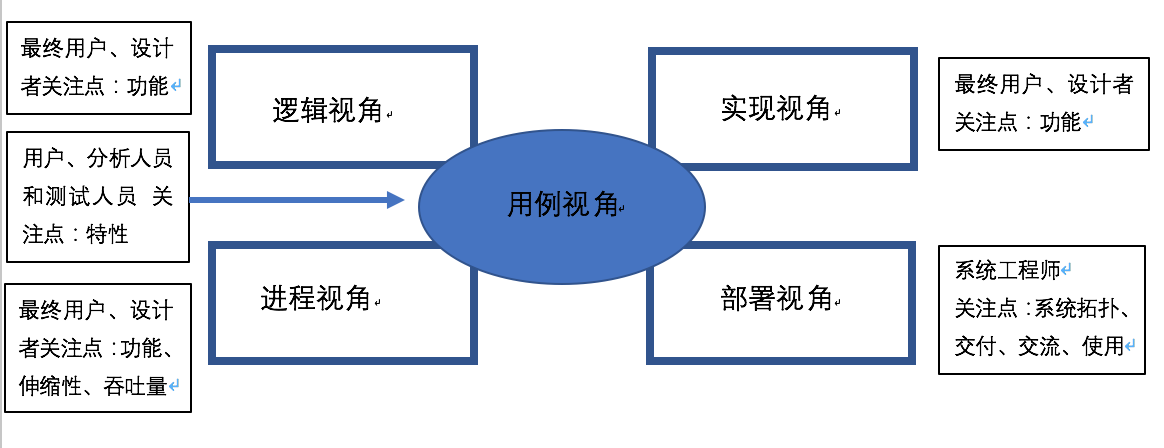


图3-7 “4+1”视图模型

### 3.4.1 逻辑视角

本系统在软件逻辑架构的设计工作上采用的是轻量级 Struts 框架，该框架是基于MVC模式的一种架构。下图为本文系统的软件逻辑架构示意图

在 Sturst 框架技术中，Action和 Actionservlet 具有控制器的功能，而 JavaBean 则是用来充当系统的控制模型的角色。具体实现的过程为:当用户提出操作请求之后，第一步会由 Tomcat 服务器负责接收用户发出的操作请求的信息，在将此信息请求传送到控制器Actionservlet 中，接下来由 Action来负责接收操作指令，然后从数据库 MySQL中找到的对应信息，之后再将数据访问结果返回给视图呈现给用户。

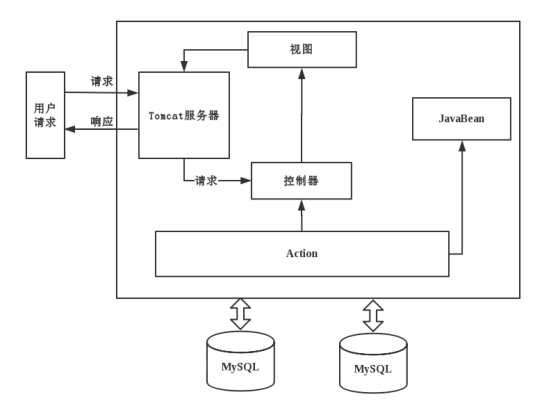


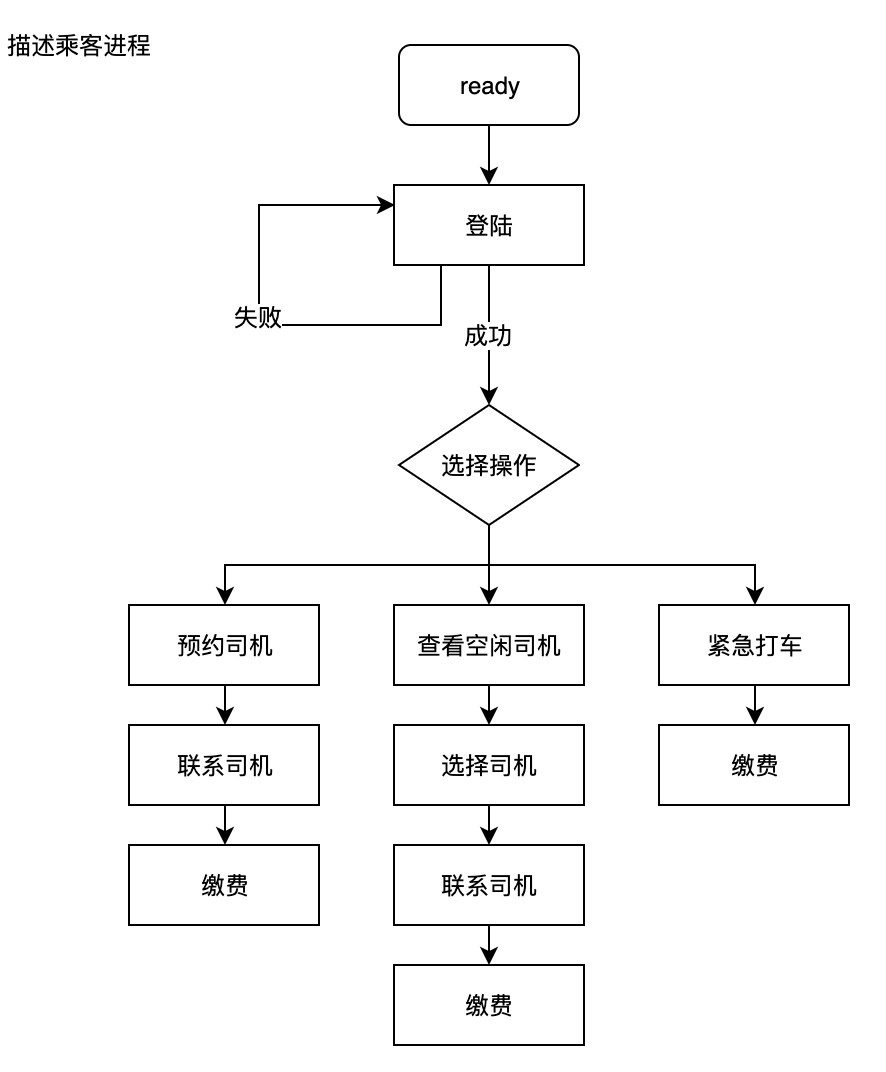
图3-8 系统逻辑视角

### 3.4.2 进程视角

本说明书使用 SDL语言对针对进程视角进行描述。主要描述在概念级的软件体系结构下，系统运行态的情况。描述系统在执行时，包括哪些进程（包括线程、进程、进程组），以及它们之间是如何进行通信的、如何进行消息传递、接口如何。并且来说明如何进行组织。

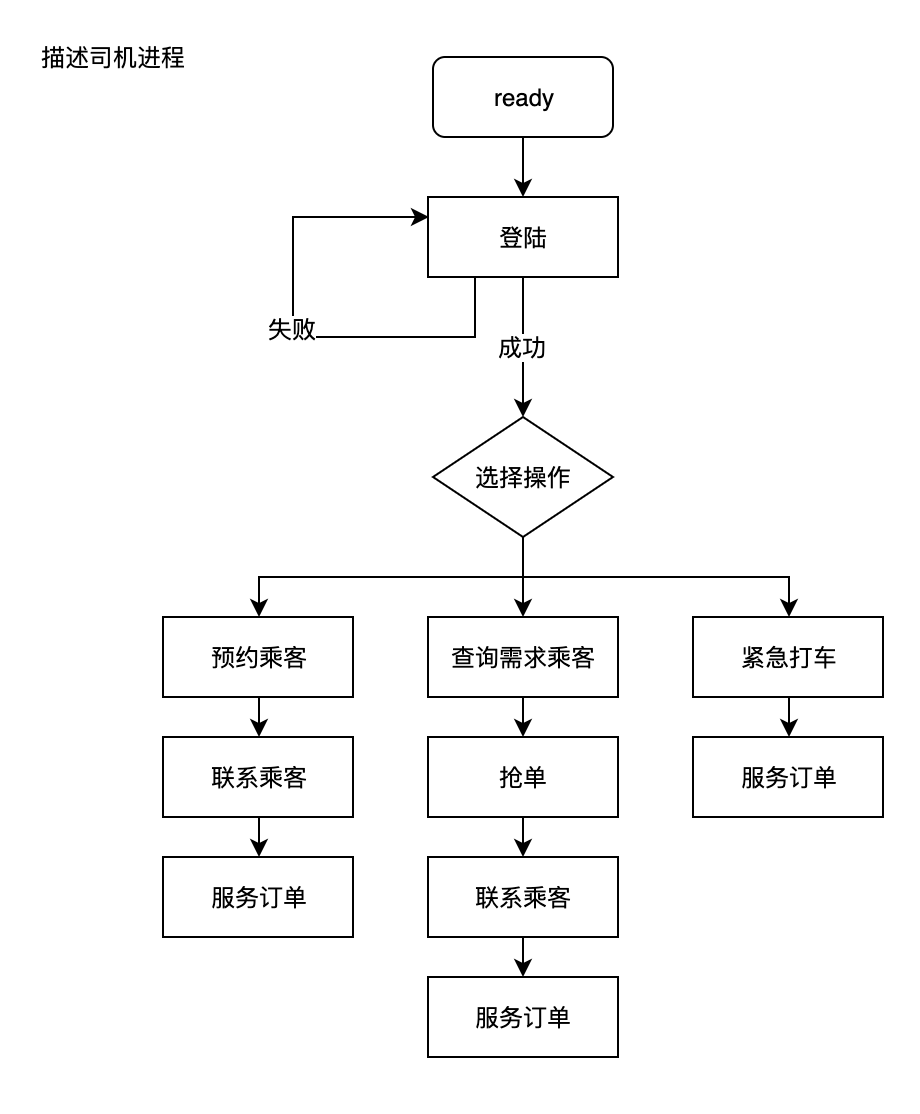
3.4.2.1 乘客进程

乘客登陆并选择自己的操作，即预约司机，产看空闲司机，紧急打车；在预约司机下还可使用联系司机，联系司机后缴费，查看空闲司机下可选择司机，选择后可联系司机，随后进行缴费，紧急打车下可直接进行缴费。



3.4.2.2 司机进程

司机登陆并选择自己的操作，即预约乘客，查询需求司机，紧急打车；在预约乘客下还可使用联系乘客，联系乘客后服务订单，查询需求司机下可选择抢单，选择后可联系乘客，服务订单，紧急打车下可直接进行服务订单。



### 3.4.3 实现视角

实现视角，又叫开发视图（Development View），描述了在开发环境中软件的静态组织结构，即关注软件开发环境下实际模块的组织，服务于软件编程人员。将软件打包成小的程序块（程序库或子系统），它们可以由一位或几位开发人员来开发。子系统可以组织成分层结构，每个层为上一层提供良好定义的接口。

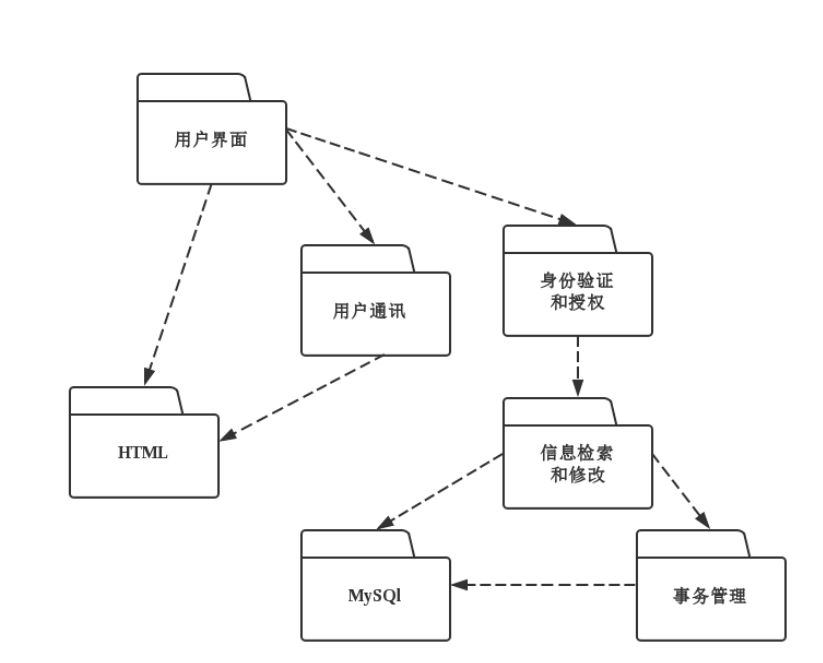


图3-12 架构开发视图

### 3.4.4 部署视角

从系统软硬件物理配置的角度，描述系统的网络逻辑拓扑结构。模型包括各个物理节点的硬件与软件配置，网络的逻辑拓扑结构，节点间的交互和讯关系等。同时还表达了进程视图中的各个进程具体分配到物理节点的映射关系。由于本系统采用的是 B/S架构，结合现实的网络拓扑结构，本文设计部署视角如下图：

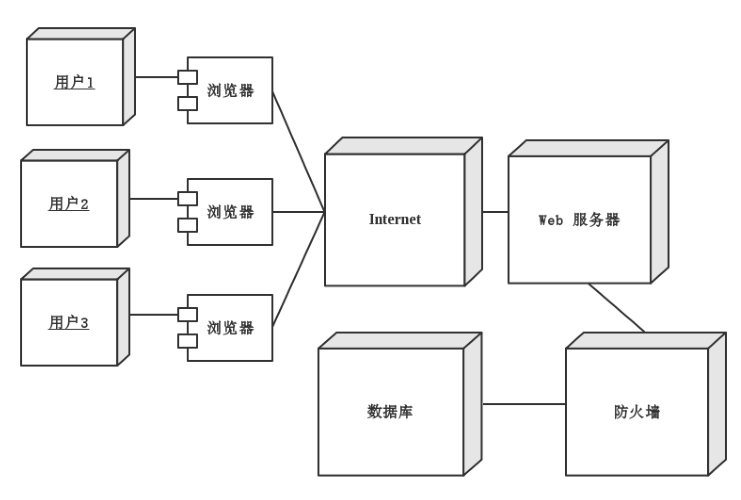


图3-13 系统部署视图

## 3.5 实现问题

针对风险分析，得出该体系结构可能存在的实现问题如下：

1. 相关人员应该对该系统所使用的开发结构、平台以及相关技术十分熟练
2. 当人数上升时，系统本身可以并发响应的用户有可能达不到此时实际的并发用户。我们不可能无上限地设置并发用户。因此，当发生这种情况时，我们应该设置相应的优先级队列进行处理，比如先到先处理，同时，对有一段时间没有响应处理的线程任务，我们应当将其暂时挂起。
3. 可能存在部分用户通过手机进行访问，这个时候应该考虑到要对相应的一些手机系统的浏览器进行接口完善，防止无法访问的情况发生。比如，在QQ中打开某一网址时，QQ会自动在网址后面添加一些参数，如果系统本身对参数没有处理，有可能导致手机用户无法访问。
4. 在系统开发的过程中，要对开发周期和开发预算有严格的限制，在必要的情况下可以对这些环境因素进行调整，但是并不能无节制的调整和延期。

# 4. 质量的分析与评价

## 4.1 场景分析

场景分析通过分析软件应用的场景，从用户的角度出发，从场景的角度来设计测试用例，是一种面向用户的测试用例设计方法。系统的使用场景与质量属性的要求是密切相关的，也是决定体系结构的重要依据。使用场景包括了系统需求工程中所提出的各种需求，通过场景可以很好地评估体系结构。

### 4.1.1 用例场景分析

用例场景是从使用者的角度出发，描述用户期望的与整个运行系统的交互：

表4-1 场景用例

|  |  |  |  |
| --- | --- | --- | --- |
| 场景编号 | 用户动作 | 系统结果 | 测试结果 |
| 场景1 | 用户登陆 | 登陆成功或密码错误或账号错误 |  |
| 场景2 | 用户修改密码 | 用户输入旧密码，要求输入两次新密码，验证之后提示修改成功 |  |
| 场景3 | 查询空闲司机 | 跳转至程序查询页面 |  |
| 场景4 | 预约司机 | 选择想要预约的司机及时间 |  |
| 场景5 | 紧急打车 | 程序提示紧急打车成功，并显示接受到紧急打车订单的司机与车牌号 |  |
| 场景6 | 乘客缴费 | 乘客在订单页面选择点击缴费，跳转至第三方安全支付平台 |  |
| 场景7 | 预约乘客 | 若出现乘客预约提示，跳转至预约乘客页面 |  |
| 场景8 | 选择需求乘客 | 显示地图，并显示有打车需求乘客，可点击选择有需求乘客 |  |
| 场景9 | 紧急打车（司机） | 若出现乘客紧急打车需求，跳转至紧急达成页面 |  |
| 场景10 | 获取收益 | 乘客支付完毕，由第三方平台金额转账 |  |
| 场景11 | 联系 | 乘客与司机进入联系界面，双方可发送信息，语音与图片。 |  |
| 场景12 | 订单反馈 | 乘客在订单界面如果对当前订单有任何疑问，可点击反馈向客服。 |  |

### 4.1.2 增长性场景分析

1. 未来可能会出现用户量剧增导致的服务器负载过重的问题，可以考虑使用分布式的服务器集群，均衡单个服务器的负载量，从而降低用户使用该软件系统的响应时间

2. 通过扩充现有数据库规模，新增数据库服务器，降低用户的检索时间

3. 通过对软件系统适配手机界面，使得用户可以在移动设备终端上访问学生成绩管理系统

### 4.1.3 探索性场景分析

探索性场景的目标是解释当前设计的边界条件的限制，揭露出可能隐含着的假设条件。探索性场景在某种程度上可以为未来学生成绩管理系统的修改给出更实际的需求。

探索性场景1：该管理系统可以做出移动客户端（Android平台、IOS平台以及其他平台）

探索性场景2：当数据库系统出现问题时候，可以通过日志来进行恢复，损失的数据恢复不超过5分钟

## 4.2 原型分析

本系统结合B/S体系结构模式和MVC体系结构模式设计而成。B/S架构主要体现在使用者借助浏览器登录软件管理系统，开始进行相关数据的反馈，根据客户相关的业务逻辑，数据将会从数据库中取出同样显示在网页上面。MVC主要用于编写代码时的约束，使得设计思路清晰，便于未来系统的修改和重新使用。

使用效用树法进行原型分析，分析结果如下图所示：

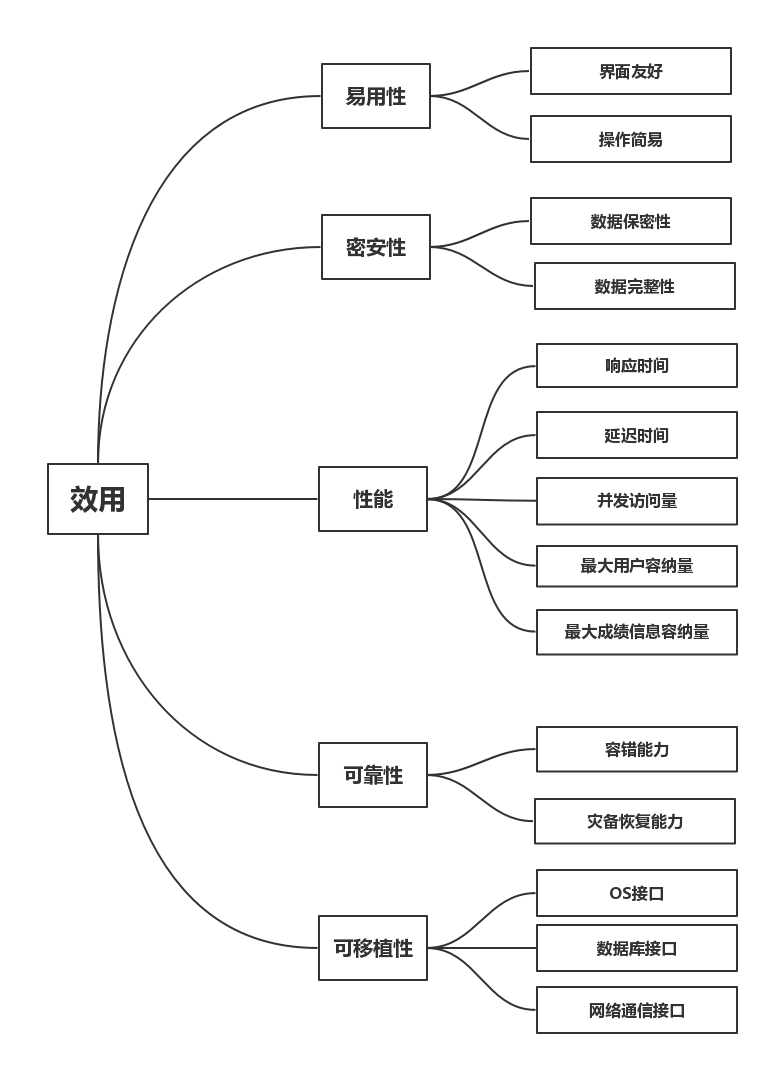


图3-13 系统部署视图