

网约车管理系统

体系结构设计文档

1.0

2018.12.11

王珑涵

软件学院

2017211905

修订历史记录

日期	描述	作者	说明

审批历史记录

签名	姓名	名称	日期

目录

修订历史记录.....	2
审批历史记录.....	2
1. 介绍	5
1.1 目的	5
1.2 范围	5
1.3 术语定义	6
1.4 参考文献	6
2. 项目相关信息	6
2.1 项目概述	6
2.2 用户角色	7
2.3 产品功能	7
3. 体系结构需求	8
3.1 关键指标	8
3.2 体系结构用例	9
3.3 各相关方对体系结构的要求	13
3.4 约束条件	13
3.5 非功能需求	14
4. 解决方案	16
4.1 相关的体系结构模式	16
4.1.1 传统的信息管理系统开发模式 (C/S)	17
4.1.2 B/S 信息管理体制系统开发模式	17
4.1.3 体系结构选择	18
4.2 体系结构概述	18
4.3 结构化视图	18
4.3.1 系统概念级体系结构设计	18
4.3.2 模块级体系结构	21
4.3.2.1 登陆模块	21
4.3.2.2 查询功能模块	21
4.3.2.3 打车功能模块	21
4.3.2.4 用户信息管理模块	21
4.4 行为视图	21

4.4.1 逻辑视角.....	22
4.4.2 进程视角.....	23
4.4.2.1 乘客进程	23
4.4.2.2 司机进程	24
4.4.3 实现视角.....	26
4.4.4 部署视角.....	26
4.5 实现问题.....	27
5. 质量的分析与评价.....	28
5.1 场景分析	28
5.1.1 用例场景分析.....	28
5.1.2 增长性场景分析	29
5.1.3 探索性场景分析	29
5.2 原型分析	29
5.2.1 效用树	29
5.1 风险	30
5.1.1 技术风险.....	30
5.1.2 进度风险.....	31
5.1.3 质量风险.....	32

1. 介绍

1.1 目的

本体系结构设计文档是在对网约车管理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，旨在阐述网约车管理系统的总体结构，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为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开发人员、测试人员、维护人员、软件客户、项目经理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) 软件部件开发人员：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：根据本文档理解系统中各部件之间的关系，并根据文档中确定的体系结构进行软件系统维护。
- 5) 系统用户：了解预期项目的功能和性能与整体结构，检查需求是否买描述满足未来需要，并说明将来可能的变更。
- 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

系统名称：网约车管理系统。

文档内容：本体系结构设计文档概括地描述了网约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。

应用范围：本软件体系结构设计文档适合于网约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，以及网约车管理系统未来的维护、运行和升级改造等要求。

1.3 术语定义

术语	定义
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。

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.

2. 项目相关信息

在这一部分对网约车管理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

产品设计背景：随着我国网约车数量和网约车平台的持续增加，网约车已经成为人们出行的一种重要交通方式。虽然网约车发展迅速，但网约车仍是一个新兴的行业，现有平台的管理系统尚有不完善之处，本文档将对网约车管理系统的功能需求进行详细分析。

系统应用目的：帮助司机尽可能多地接到订单，提高工作效率；帮助乘客选择优质、合适的网约车，并保证乘客安全、隐私等；帮助网约车管理者调度、管理网约车。

2.2 用户角色

在网约车管理系统中有三类用户：司机、乘客和管理员。这三类用户角色有不同的系统使用权限。不同用户角色对系统使用的需求不同，具体如下：

- **司机：**
 - 1) 一般在开车时使用系统，无法将过多注意力集中于系统上。
 - 2) 部分司机对电子产品接受度较低，未用过相似产品。
 - 3) 对行车收入较为关心
- **乘客：**
 - 1) 对电子产品接受度较高，一般使用过相似产品。
 - 2) 对乘车所耗时间、费用较为关心。
 - 3) 在意乘车体验和乘车安全。
- **管理员：**
 - 1) 一般为受过培训的专业人士，对复杂系统有着较高接受度。
 - 2) 关心乘客、司机的安全问题。
 - 3) 维持系统的正常工作。
 - 4) 关心产品的运营情况，如每日流水等。

2.3 产品功能

- **乘客端**
 - 1) LBS 定位功能：使用优化算法让乘客的定位更加精准，方便司机寻路和接送。
 - 2) 地图导航：选择上车地点开始，地图精准定位，附近司机位置实时显示，接单后司机位置实时更新，司机距离显示，司机预计到达时间。
 - 3) 费用估算功能：乘客确定出发地和目的地后，大概估算出需要花费的时间与费用。
 - 4) 司机与车辆信息：在为乘客分派司机后，乘客可以查看司机的相关信息，包括司机的手机号码、车牌号码、车辆型号、司机评价等。
 - 5) 即时计费功能：在司机到达起始地点，乘客上车后开始计费，用户可以在 APP 上实时查看此次行车已用时间费用等信息。
 - 6) 一键呼救功能：遇到危险情况时，乘客可以通过 APP 一键求救，后台管理人员将介入系统了解情况，根据情况选择处理方式。
- **司机端：**
 - 1) 系统推荐订单：系统会给司机推送附近的乘客及其目的地，司机可以选择接单或者拒单。
 - 2) 接受订单：司机接受订单后，可以看到乘客的联系方式。司机行至上车地点附近后可联系乘客确认具体上车地点。

- 3) 智能导航系统：自主研发或者调用百度、搞得的导航系统，过呢据出发地和目的地进行精确导航，确认最短或最快路线。
- 4) 收入查询：司机可以在“我的钱包”中看到载客所收报酬，可通过绑定银行卡、支付宝或微信，将收入提现。
- 管理员端：
 - 1) 根据权限查看订单并以次进行业务分析。
 - 2) 接收乘客的紧急呼救、生成解决方案并干预车辆行程。

3. 体系结构需求

3.1 关键指标

3.1.1 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：

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

3.1.2 质量要求

所储存的车位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) 网约车管理系统应能支持（M: 30000 D:100000 B:150000）位用户同时进行操作（即最低支持30000 位用户，期望正常支持100000 位用户，在支持150000 名用户时系统仍能正常使用）。
- 2) 本系统对各种请求的响应时间不超过1.5 秒。
- 3) 导航的响应延迟不应超过2秒。
- 4) 支付的响应延迟不应超过1 秒。
- 5) 在系统因并发访问人数过多而发生宕机时，系统应该在 6 分钟内能进行系统恢复。

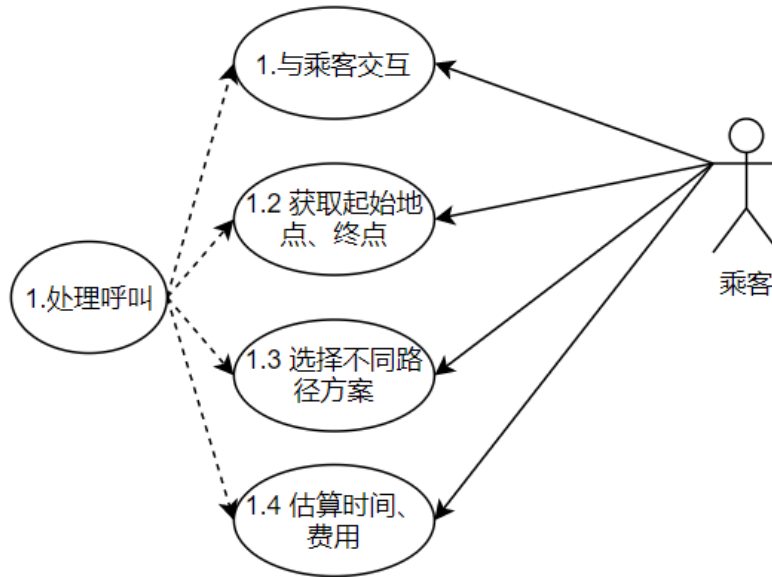
3.2 体系结构用例

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、子系统或各项功能的使用和行为。

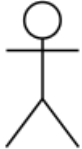
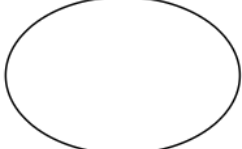
3.2.1 用户角色

网约车管理系统共有三种用户角色：司机、乘客、管理员。下面将针对三种用户画出他们的用例图

3.2.2 乘客用例



图例说明：

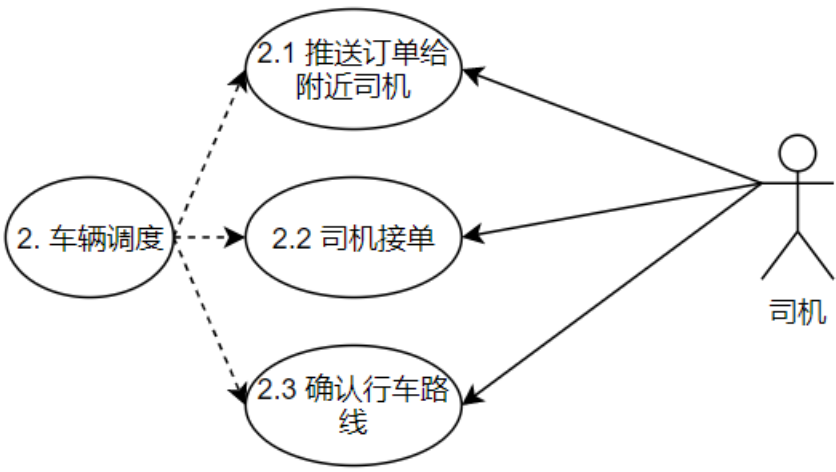
图形	含义
	参与者
	用例

设计论述：


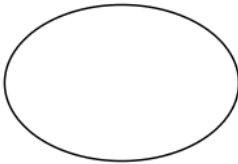
- 1) 乘客用户所使用的处理呼叫模块
- 2) 模块包含的子功能有：
 - a) 交互
 - b) 获取起始地点、终点
 - c) 选择不同路径方案
 - d) 估算时间、费用

3.2.3 司机用例

/



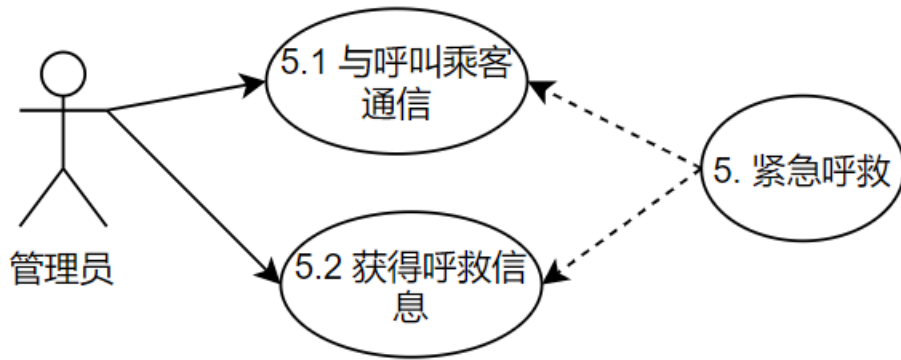
图例说明：

图形	含义
	参与者
	用例


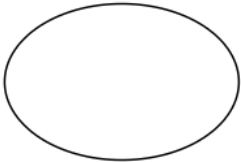
设计论述：

- 1) 乘客用户所使用的车辆调度模块
- 2) 模块包含的子功能有：
 - a) 推送订单
 - b) 接单
 - c) 确认行车路线

3.2.4 管理员用例



图例说明：

图形	含义
	参与者
	用例

设计论述：

- 1) 乘客用户所使用的紧急呼救模块
- 2) 模块包含的子功能有：
 - a) 与呼救乘客通信
 - b) 获取呼救信息

3.3 各相关方对体系结构的要求

网约车管理系统结构视角如图所示，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。

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，对于网约车管理系统，要求系统能满足司机、乘客、管理员三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、便捷地呼叫车辆、接受订单、支付费用等功能。

3.3.1 系统开发、测试、维护人员

体系结构的设计应与部件无关、低风险性、高可复用性高、可度量性、高可移植性、高可维护性。

与部件无关：是指体系结构的部件可相互独立地工作，无需了解其他部件的具体实现原理

低风险性：按此体系结构实施的技术和技能的风险度较低。

可复用性：体系结构设计抽象，且具有高通用性

可度量性高：按该体系结构实施，系统的开发进度，人力、资金、资源调配可以估计的能力。使项目管理人员能更好地进行项目进度的管理和安排

高移植性：统软件的接口易改造，可以比较容易地转移到其他计算机上使用；同时系统使用了跨平台的 Java 语言来编写并且使用了开源库和复用了一些开源项目的代码，因此可以比较方便地移植到不同平台上。

高可维护性：系统的构建结构合理，采用“高内聚、低耦合”的原则，可以比较方便地进行修改、升级、测试、维护。

3.3.2 用户

系统应具有高可用性

高可用性：

该具备容易操作的功能。

- 1) 提供针对不同用户的用户使用说明手册，方便用户学习使用。
- 2) 系统应提供在线帮助界面，方便用户学习操作。
- 3) 由于操作该系统的人员有很多，且操作习惯、受教育程度、年龄阶段、接受事物能力等都各不相同，这就要求系统具备良好的人机交互能力。系统提供的各种功能便于用户理解，操作简单，用户很容易掌握。
- 4) 系统的界面设计应简洁明了，使用户能够自己学会使用本系统。
- 5) 系统应具有一定的美观性，可参考目前大部分网站的扁平化设计。

3.4 约束条件

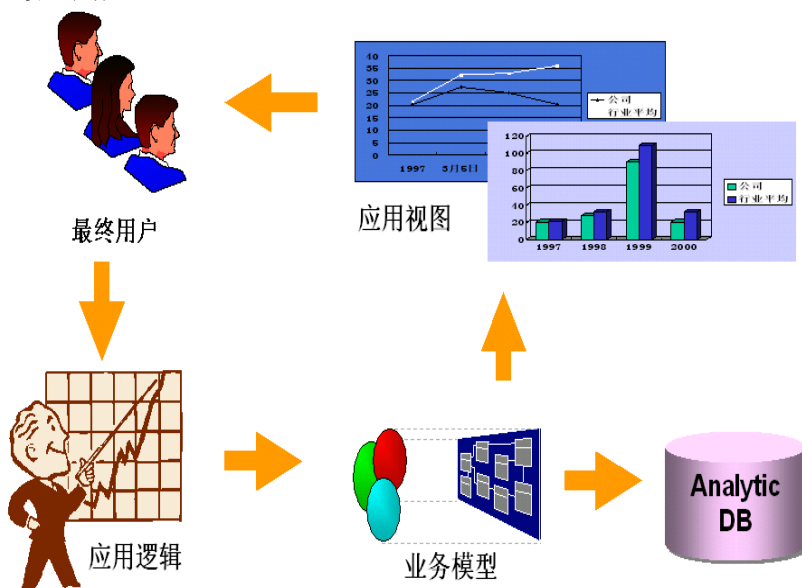
3.4.1 实现约束

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。

3.4.2 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。



3.5 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]。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。

3.5.1 质量需求

3.5.2 性能

网约车管理系统应能支持（M：30000 D:100000 B:150000）位用户同时进行操作（即最低支持30000 位用户，期望正常支持100000 位用户，在支持150000 名用户时系统仍能正常使用）。本系统对各种请求的响应时间不超过1.5 秒；导航的响应延迟不应超过2

秒；支付的响应延迟不应超过1 秒。

3.5.2.1 可靠性

网约车管理系统的用户数量庞大、用户类型多样，用户可能输入错误的信息或者进行错误的操作。本系统应具有较强的鲁棒性，能够捕捉用户操作引起的异常，

可在一定程度上修复用户引起的错误，并在可容忍错误的情况下保持稳定运行。本系统要求对用户的账户余额、收入等关键数据做到准确一致。同时应定期对数据进行备份，进一步提高网约车管理系统对意外情况的处理能力。

3.5.2.2 易用性

网约车管理系统同时面向专业认识和非专业人士，非专业人士（如司机、乘客）对复杂系统的接受能力较低。因此系统应的易用性应满足该类用户的需求，界面应设计的人性化、简洁、符合普通人的使用习惯。

3.5.2.3 安全性

网约车管理系统涉及到多种用户的重要记录，如行车记录、账户余额、收入等，系统中的数据必须要得到良好的保护，所哟数据应采取加密存储，并使用安全的连接保证传输过程的数据信息安全性。同时系统应当是能被重建的；应当能被有效控制，抗干扰能力强；系统用户的权限必须是可识别的。

3.5.2.4 可移植性

由于系统将被长期使用并在将来作出修改、维护、拓展，网约车管理系统需要具有较好的可维护性。在系统架构设计时要充分考虑业务流程中可能出现的对象，系统要能及时根据业务信息的变化进行相应的调整，系统要有对技术和业务需求变化有足够的支持能力，今后对系统的功能进行修改时，要以尽量少的代价适应变化。

3.5.2.5 可移植性

网约车管理系统的生命周期长，有可能改变运行环境。因此系统应具有较强的适应性和跨平台性。系统应能够容易地被卸载、升级、替换，系统应可以与其他软件共存于同一平台上，不产生冲突。

3.5.3 工程需求

3.5.3.1 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

3.5.3.2 逻辑数据库要求

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。

3.5.4 其他需求

3.5.4.1 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。

3.5.4.2 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。

3.5.4.3 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管理。

4. 解决方案

4.1 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以下特点：

- (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。

4.1.1 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,既费时又提高了软件成本。
- (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。

4.1.2 B/S 信息管理系统开发模式

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。

- i. 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将会成为信息管理系统发展的必然趋势。

4.1.3 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。

4.2 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

4.3 结构化视图

4.3.1 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端实现的。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 ^①	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。

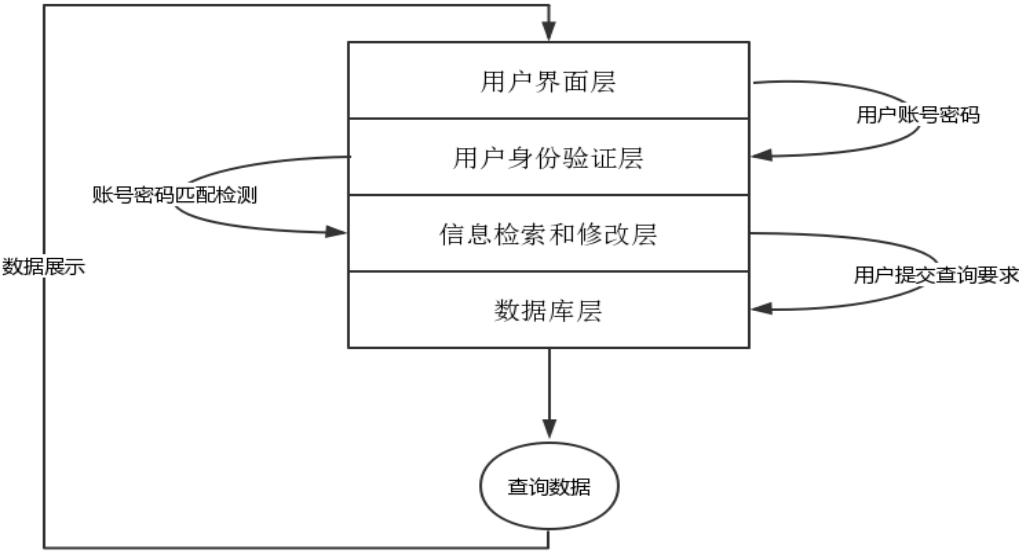


图 3-1 MVC 体系架构图

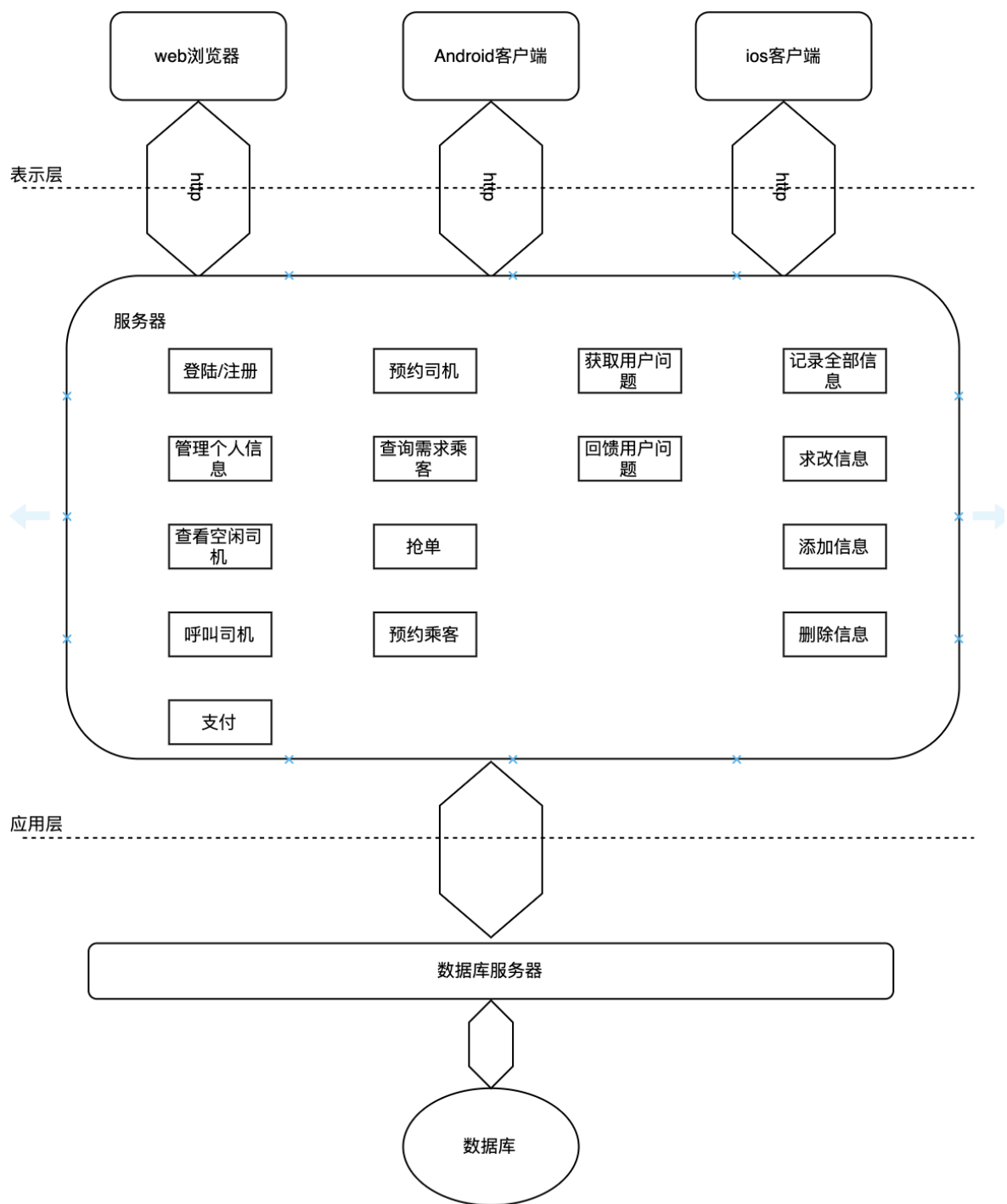


图 3-2 三层 B/S 模式的系统体系结构模型

4.3.2 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。

4.3.2.1 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。

4.3.2.2 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。

4.3.2.3 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。

4.3.2.4 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。

4.4 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。

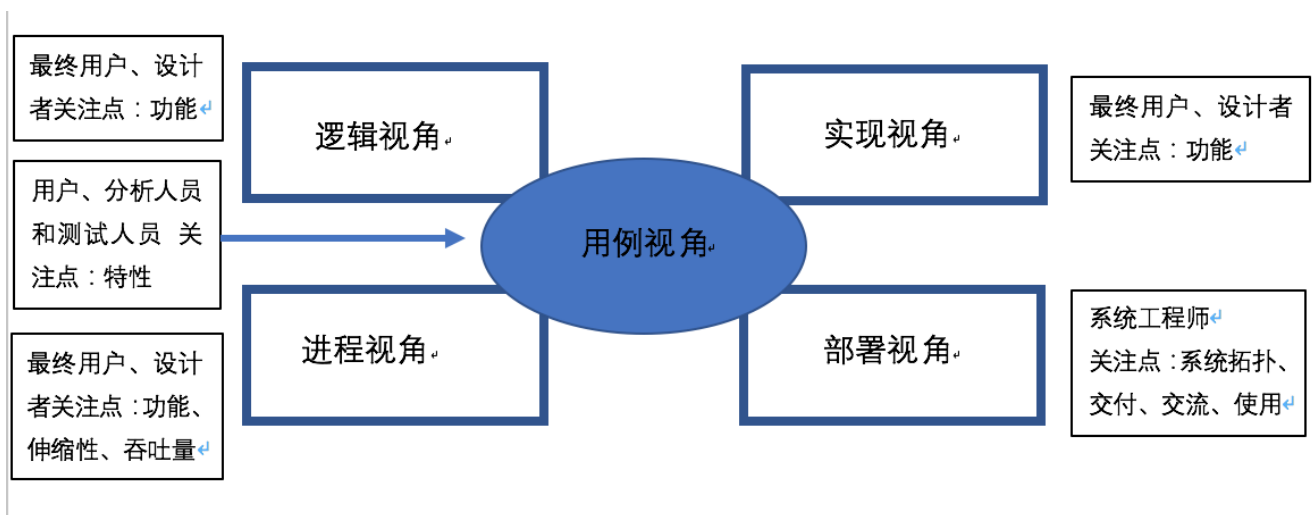
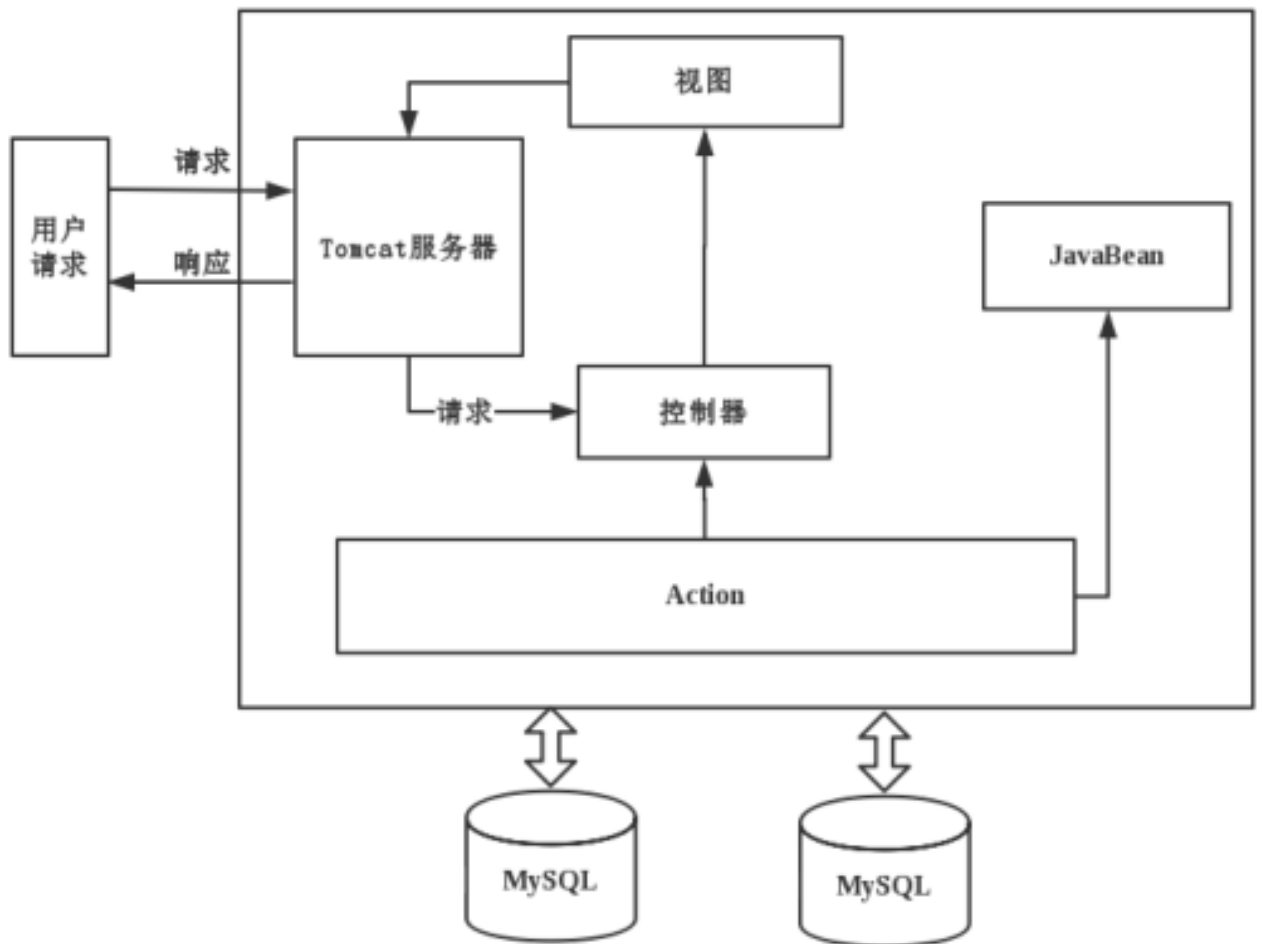


图 3-7 “4+1” 视图模型

4.4.1 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 Struts 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。



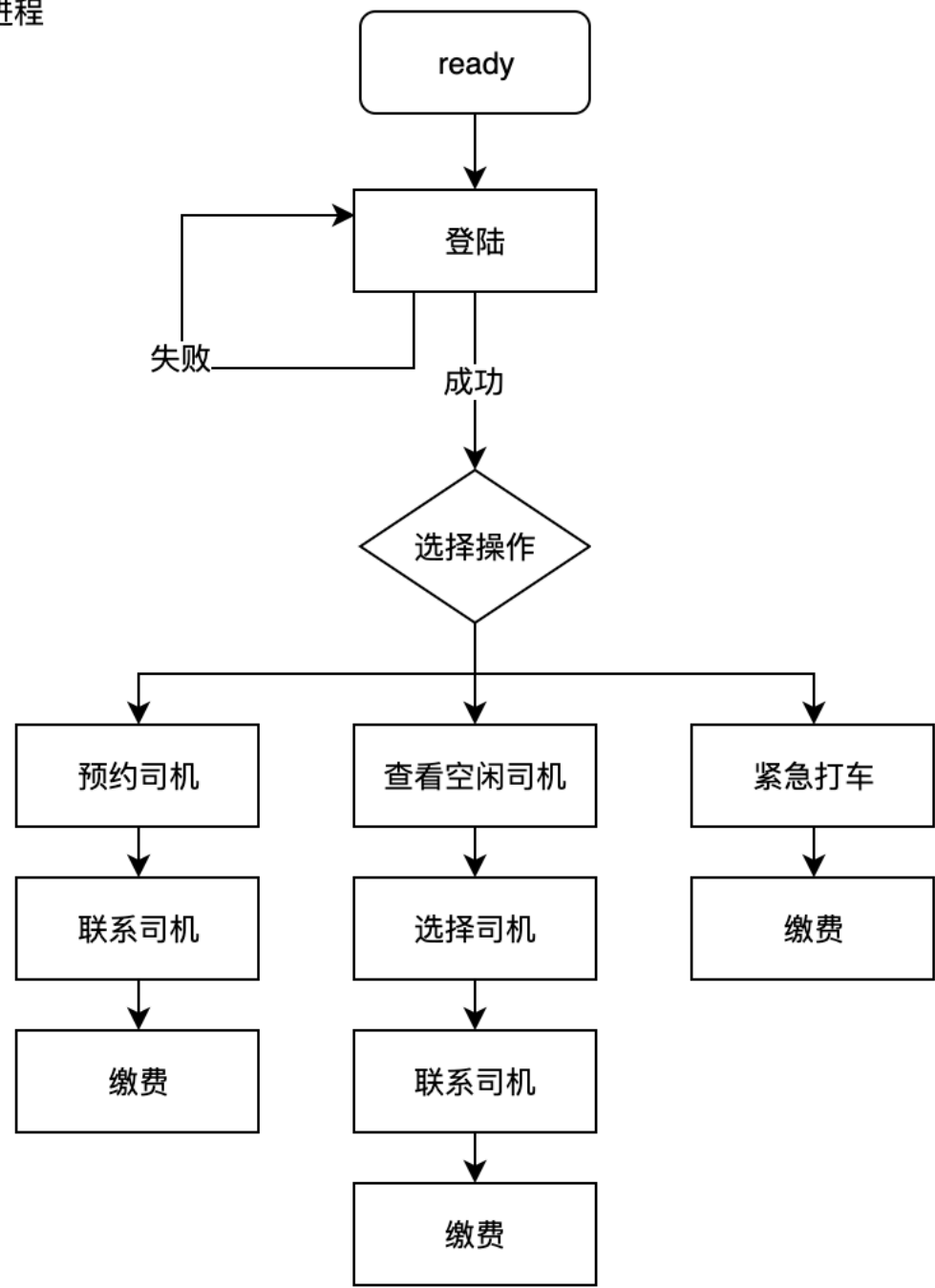
4.4.2 进程视角

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。

4.4.2.1 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。

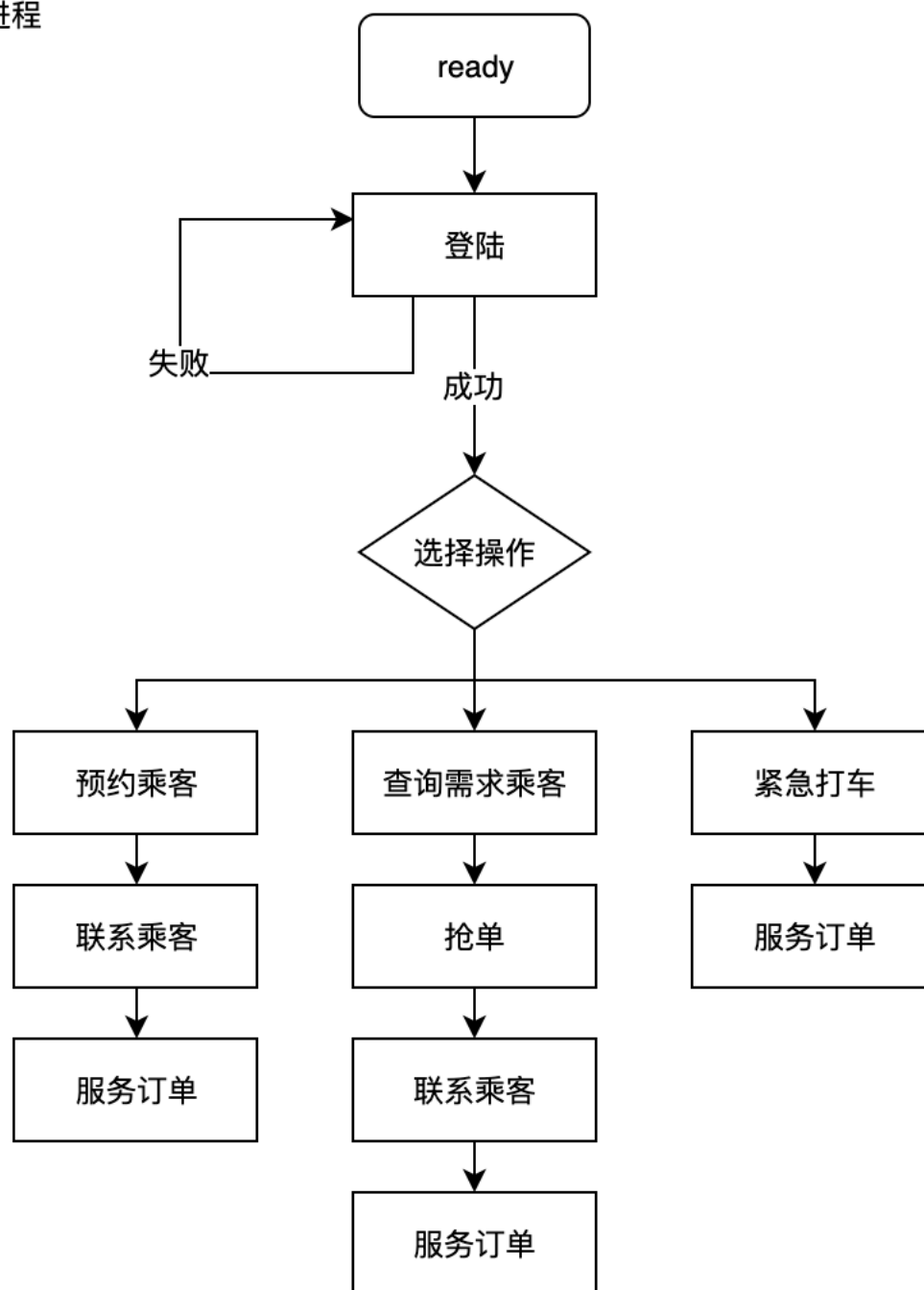
描述乘客进程



4. 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。

描述司机进程



4.4.3 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。

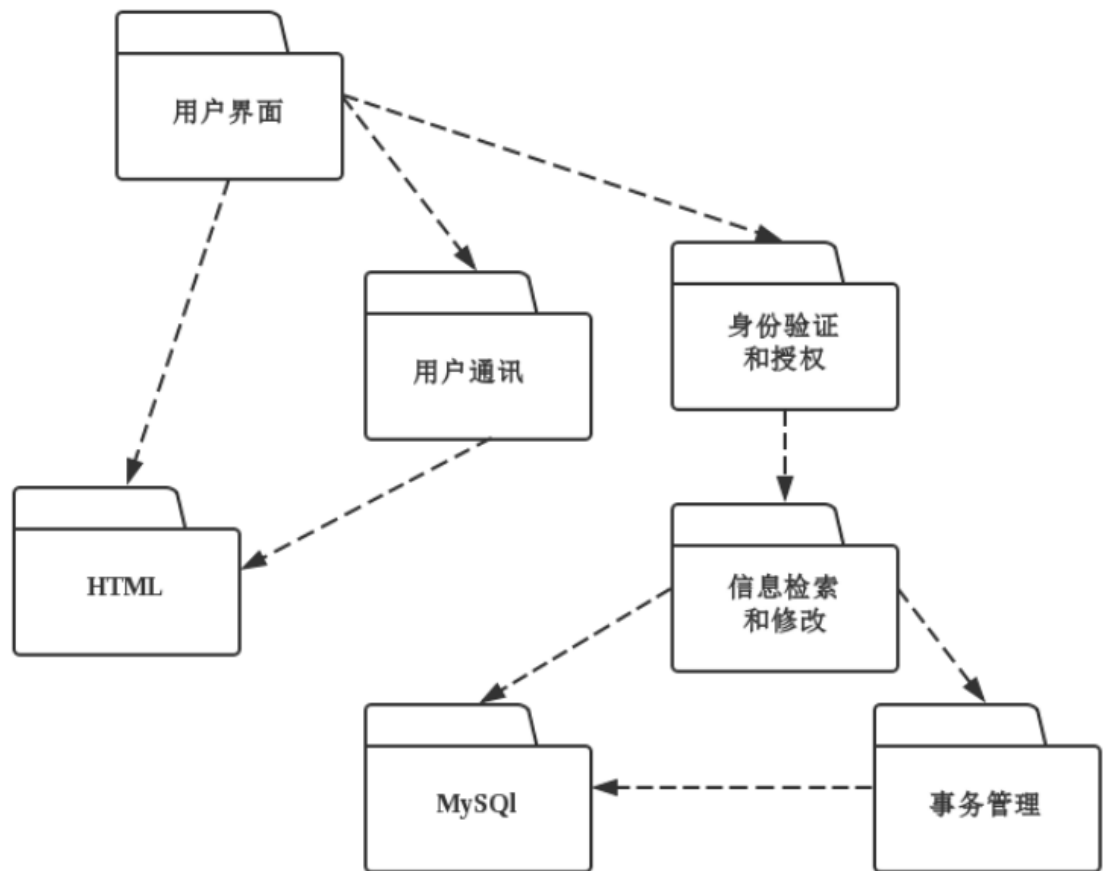
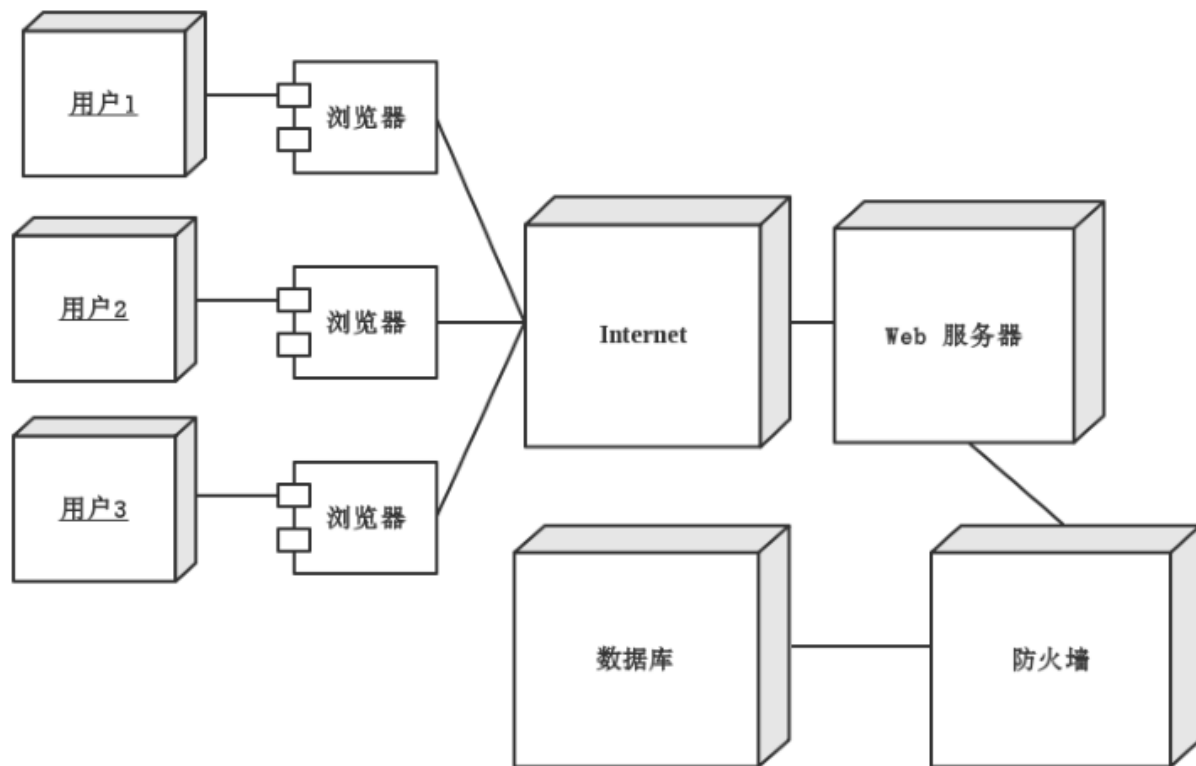


图 3-12 架构开发视图

4.4.4 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，节点间的交互和通讯关系等。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：



4.5 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。

5. 质量的分析与评价

5.1 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。

5.1.1 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：

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	输入起止地点	跳转至地图页面	
场景 4	选择司机	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 8	呼救（乘客）	若乘客发出呼救信息，跳转至人工服务	
场景 9	获取收益	乘客支付完毕，由第三方平台金额转账	
场景 10	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 11	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。	

5.1.2 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

5.1.3 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）

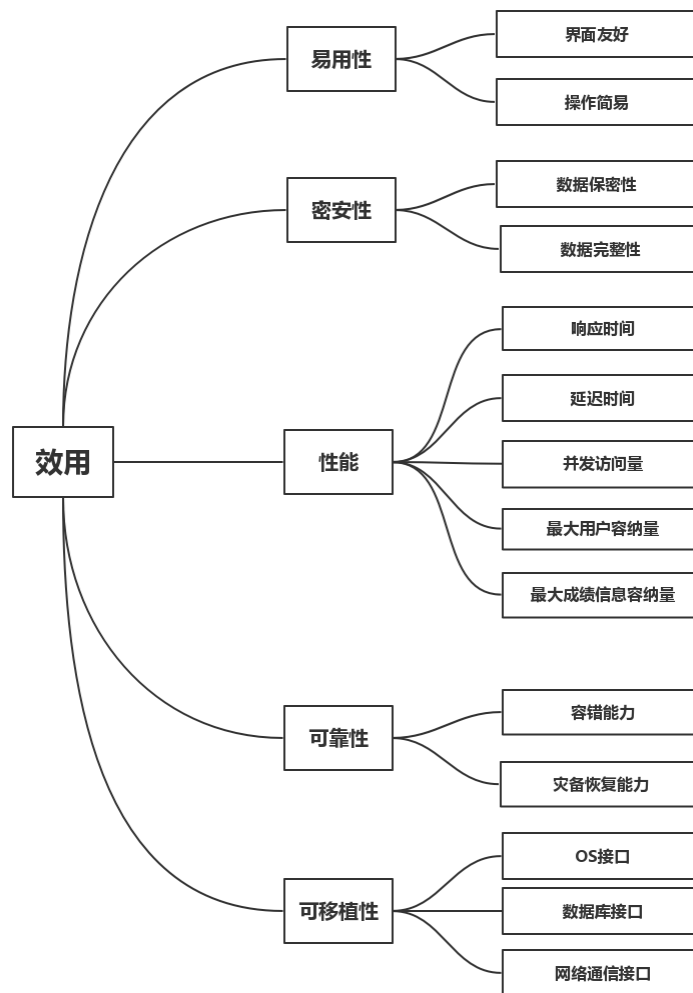
探索性场景 2：当数据库系统出现问题时候，可以通过日志来进行恢复，损失的数据恢复不超过 5 分钟

5.2 原型分析

5.2.1 效用树

效用树（utility trees）提供自顶向下的机制，直接和有效地将系统的业务具体分解到质量的属性场景，效用树的输出是场景的优先权列表，可供评估人员在短时间内发现体系结构中存在的问题。

本网约车管理系统的效用树见[错误!未找到引用源。](#)。



5.1 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。

5.1.1 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。

5.1.2 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。

5.1.3 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。