打车软件系统

体系结构设计文档

1.0

2019/12/11

颜加泉

软件学院

2017211967

软件工程导论

2019 秋

# 修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| **时间** | **版本** | **作者** | **备注** |
| <2019.12.11> | <版本1.0> | <颜加泉> | <第一版> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 文件批准许可

以下体系结构数据文档已经被以下机构人员批准并认可:

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

目录

[修订历史 2](#_Toc26967058)

[文件批准许可 2](#_Toc26967059)

[1. 介绍 6](#_Toc26967060)

[1.1 目的 6](#_Toc26967061)

[1.2 范围 6](#_Toc26967062)

[1.3 术语定义 7](#_Toc26967063)

[1.4 参考资料 7](#_Toc26967064)

[2. 体系架构概述 8](#_Toc26967065)

[2.1文档结构概述 8](#_Toc26967066)

[2.2 体系结构调研与选择 8](#_Toc26967067)

[3. 体系架构目标和约束 11](#_Toc26967068)

[3.1质量要求目标 11](#_Toc26967069)

[3.1.1影响系统运行质量指标要求 11](#_Toc26967070)

[3.1.2非运行质量的质量指标要求 12](#_Toc26967071)

[3.2约束条件 13](#_Toc26967072)

[3.2.1 设计约束 13](#_Toc26967073)

[3.2.2 逻辑数据库需求 13](#_Toc26967074)

[3.2.3业务约束 13](#_Toc26967075)

[3.3各相关方对体系结构的要求 14](#_Toc26967076)

[4. 用例视角 15](#_Toc26967077)

[4.1 体系结构用例 15](#_Toc26967078)

[4.1.1注册/登录 16](#_Toc26967079)

[4.1.2个人信息管理 16](#_Toc26967080)

[4.1.3呼叫/预约打车 17](#_Toc26967081)

[4.1.4在线支付 17](#_Toc26967082)

[4.1.5导航 18](#_Toc26967083)

[4.1.6投诉/投诉处理 18](#_Toc26967084)

[4.1.7业务记录 19](#_Toc26967085)

[4.1.8管理/备份数据库 19](#_Toc26967086)

[4.1.9处理系统运行异常 20](#_Toc26967087)

[5. 逻辑视角 21](#_Toc26967088)

[5.1 体系概述-包和子系统分层 21](#_Toc26967089)

[5.1.1交互层 22](#_Toc26967090)

[5.1.2逻辑层 24](#_Toc26967091)

[5.1.3服务层 25](#_Toc26967092)

[5.2 重要的类设计 26](#_Toc26967093)

[6. 进程视角 27](#_Toc26967094)

[6.1进程 27](#_Toc26967095)

[6.1.1司机交互进程 27](#_Toc26967096)

[6.1.2打车者交互进程 28](#_Toc26967097)

[6.1.3业务监测员交互进程 28](#_Toc26967098)

[6.1.4系统管理员交互进程 28](#_Toc26967099)

[6.1.5订单传输进程 28](#_Toc26967100)

[6.1.6投诉传输进程 29](#_Toc26967101)

[6.1.7实时导航进程 29](#_Toc26967102)

[6.1.6在线支付进程 29](#_Toc26967103)

[6.1.6数据库访问 29](#_Toc26967104)

[6.2进程的设计要素 30](#_Toc26967105)

[6.2.1订单数据缓存线程 30](#_Toc26967106)

[6.2.2投诉数据缓存线程 31](#_Toc26967107)

[6.2.3导航路径缓存线程 31](#_Toc26967108)

[6.2.4个人信息缓存线程 31](#_Toc26967109)

[6.3进程模型的设计模型依赖关系 32](#_Toc26967110)

[6.4进程实现 33](#_Toc26967111)

[6.4.1<sys/socket.h> 33](#_Toc26967112)

[6.4.1<pthread.h> 33](#_Toc26967113)

[7. 部署视角 34](#_Toc26967114)

[7.1 移动端 34](#_Toc26967115)

[7.2 PC 35](#_Toc26967116)

[7.3服务器 35](#_Toc26967117)

[7.4数据库 35](#_Toc26967118)

[7.5实时导航系统 35](#_Toc26967119)

[7.6在线支付系统 35](#_Toc26967120)

[8. 实现视角 36](#_Toc26967121)

[8.1实现视角图说明 36](#_Toc26967122)

[9. 规模与性能 37](#_Toc26967123)

[9.1时间性能要求： 37](#_Toc26967124)

[9.2空间性能要求： 37](#_Toc26967125)

[9.3规模要求： 37](#_Toc26967126)

[9.4本系统体系结构如何满足规模与性能 37](#_Toc26967127)

[10. 质量 38](#_Toc26967128)

[10.1可靠性 38](#_Toc26967129)

[10.2可使用性 38](#_Toc26967130)

[10.3密安性 38](#_Toc26967131)

[10.4可维护性 38](#_Toc26967132)

[10.5 可移植性 39](#_Toc26967133)

[10.7.1用例场景 39](#_Toc26967134)

[10.7.2增长性场景 40](#_Toc26967135)

[10.7.3探索性场景 40](#_Toc26967136)

**图目录**

[图 1 B/S(Browser Server)三层架构图 8](#_Toc26954640)

[图 2 C/S(Client Server)两层架构图 9](#_Toc26954641)

[图 3本系统架构图 10](#_Toc26954642)

[图 4用例视角用例图 15](#_Toc26954643)

[图 5注册/登录顺序图 16](#_Toc26954644)

[图 6个人信息管理顺序图 16](#_Toc26954645)

[图 7呼叫/预约打车顺序图 17](#_Toc26954646)

[图 8在线支付顺序图 17](#_Toc26954647)

[图 9导航顺序图 18](#_Toc26954648)

[图 10投诉/投诉处理 18](#_Toc26954649)

[图 11业务记录 19](#_Toc26954650)

[图 12管理/备份数据库 19](#_Toc26954651)

[图 13处理系统运行异常 20](#_Toc26954652)

[图 14体系概述-包和子系统分层图 21](#_Toc26954653)

[图 15类图 26](#_Toc26954654)

[图 16进程图 27](#_Toc26954655)

[图 17进程的设计要素图 30](#_Toc26954656)

[图 18进程模型的设计模型依赖关系图 32](#_Toc26954657)

[图 19进程实现图 33](#_Toc26954658)

[图 20部署视角图 34](#_Toc26954659)

[图 21实现视角图 36](#_Toc26954660)

**表目录**

[表 1术语定义表 7](#_Toc26968559)

[表 2 用例场景表 40](#_Toc26968560)

[表 3增长性场景 40](#_Toc26968561)

[表 4探索性场景 40](#_Toc26968562)

# 1. 介绍

本文档是一个对打车软件系统的体系结构分析报告。本文档主要通过分析影响系统运行质量和非运行质量的质量指标要求，并按不同的体系结构设计视角描述设计的体系架构以满足质量要求。本文档选择的体系结构视角模型为“4+1”视图模型，能从四个角度更好地描述出打车软件系统中不同的相关利益方关心的事情。

## 1.1 目的

本体系结构设计文档是在对打车软件系统编写了全面细致的需求分析设计文档，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，旨在阐述城打车软件系统的总体结构，包括对约束条件、设计遵循的标准的描述，从用例视角、逻辑视角、进程视角、部署视角等多个角度的视角描述体系结构的内容，包括逻辑设计、物理结构，分析系统的体系结构需求，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性能、质量的分析和评估。

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

产品名称：打车软件系统(TSS)

产品目标用户：打车者与司机

产品设计背景：随着我国交通出行的需求越来越大，选择打车出行的乘客数也越来越多，于是打车软件系统的用户市场极大。同时，出租车司机也拥有就近快速寻找乘客接单的需求，私家车司机也可以通过接单来利用空闲的车辆资源来赚取额外收入。通过完善的打车软件系统，可以极大地方便出租车司机与打车者之间交易的进行，缓解打车市场供不应求的压力。

系统应用目的：解决打车难的问题，让打车者能快速找到附近的空闲司机，司机能方便接单，同时保证打车交易过程中司机与打车者双方的权益。

系统适合场景：

1 保证打车者需要打车时能通过打车软件发出订单给附近的空闲司机。

2 保证司机空闲时能及时接收订单。

3 保证完成订单后的付款交易正确无误进行。

4 保证用户信息保密不被泄露

5 保证出现违规行为时的举报功能

6 建立打车者与司机的诚信系统，对于未按照约定完成订单的人员扣除诚信积分。

7 为打车者提供寻求代驾、共享单车、共享汽车等扩展功能。

## 1.3 术语定义

|  |  |
| --- | --- |
| **术语** | **定义** |
| TSS | 打车软件系统(TSS)，本文档负责的软件系统 |
| C/S架构 | 客户端/服务器结构 |
| B/S架构 | 浏览器/服务器结构 |
| 用例视角 | 用例视角，描述了主要的用例和场景，以及一些核心功能。 |
| 逻辑视角 | 逻辑视角，主要是整个系统的抽象结构表述，关注系统提供最终用户的功能需求。 |
| 进程视角 | 进程视角，关注系统动态运行时的进程以及相关并发、同步、通信等问题 |
| 部署视角 | 部署视角描述了最典型的平台配置的各种物理节点。还描述了任务（从流程视图）到物理节点的分配。 |
| 实现视角 | 实现视角，侧重于软件模块的组织和管理。 |

表 1术语定义表

## 1.4 参考资料

[1] 王安生.《软件工程化》[M]. 北京：清华大学出版社，2014.

[2] 《仓库管理系统的软件体系结构构模型》 百度文库

# 2. 体系架构概述

## 2.1文档结构概述

本文档描述了打车软件系统的体系架构，其内容主要包括：

1. 第1章节：对本系统和文档进行初步的介绍
2. 第2章节：对市场上的体系结构进行调研与选择，并分析原因
3. 第3章节：对体系架构的目标和约束的分析与描述. 分析了系统的影响系统运行质量和非运行质量的质量指标要求、以及一些设计约束和业务约束。
4. 第4、5、6、7、8章节：本文档使用Philippe Kruchten提出的“4+1”视图模型描述体系结构，将架构呈现为一系列视图：用例视角、逻辑视角、流程视角、部署视图和实现视角，对不同视角进行论述。
5. 第8、9章节：进行体系结构的性能、质量指标的分析，详细说明了本系统的体系架构如何满足这些要求。同时描述了可扩展性和可修改性，以及说明了哪些公性的软件部件可以抽象出形成公司的资产，外卖给同行，开源等。

## 2.2 体系结构调研与选择

**2.2.1 B/S(Browser Server)三层架构**

B/S架构的全称为Browser/Server，即浏览器/服务器结构。Browser指的是Web浏览器，极少数事务逻辑在前端实现，但主要事务逻辑在服务器端实现，Browser客户端，WebApp服务器端和DB端构成所谓的三层架构。B/S架构的系统无须特别安装，只有Web浏览器即可。

B/S架构中，显示逻辑交给了Web浏览器，事务处理逻辑在放在了WebApp上，这样就避免了庞大的胖客户端，减少了客户端的压力。因为客户端包含的逻辑很少，因此也被成为瘦客户端。

其架构流程主要有三层：

* 第一层表现层：主要完成用户和后台的交互及最终查询结果的输出功能。主要为客户端浏览器。
* 第二层逻辑层：主要是利用服务器完成客户端的应用逻辑功能。主要为WEB服务器
* 第三层数据层：主要是接受客户端请求后独立进行各种运算。主要为数据库服务器。

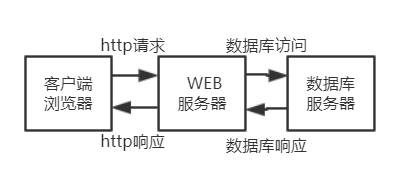


图 1 B/S(Browser Server)三层架构图

优点：

* 客户端无需安装，有Web浏览器即可。
* 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。
* BS架构无需升级多个客户端，升级服务器即可。

缺点：

* 在跨浏览器上，BS架构不尽如人意。
* 表现要达到CS程序的程度需要花费不少精力。
* 在速度和安全性上需要花费巨大的设计成本，这是BS架构的最大问题。
* 客户端服务器端的交互是请求-响应模式，通常需要刷新页面，这并不是客户乐意看到的。（在Ajax风行后此问题得到了一定程度的缓解）

**2.2.2 C/S(Client Server)两层架构**

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。

C/S 架构也可以看做是胖客户端架构。因为客户端需要实现绝大多数的业务逻辑和界面展示。这种架构中，作为客户端的部分需要承受很大的压力，因为显示逻辑和事务处理都包含在其中，通过与数据库的交互（通常是SQL或存储过程的实现）来达到持久化数据，以此满足实际项目的需要。

其架构流程主要有两层：

* 第一层：客户端——表示层（界面层与逻辑层）
* 第二层：服务器模式——数据库层

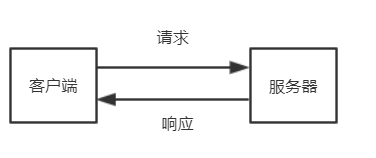


图 2 C/S(Client Server)两层架构图

优点：

* C/S架构的界面和操作可以很丰富。
* 安全性能可以很容易保证，实现多层认证也不难。
* 由于只有一层交互，因此响应速度较快。

缺点：

* 适用面窄，通常用于局域网中。
* 用户群固定。由于程序需要安装才可使用，因此不适合面向一些不可知的用户。
* 维护成本高，发生一次升级，则所有客户端的程序都需要改变。

**2.2.3 本系统的体系架构选择**

本系统选择的体系架构为C/S体系架构，在普通用户的移动端安装客户端app，以及管理人员可以在PC端使用的管理软件。在客户端处理逻辑，在需要在线支付、实时地图、订单消息传输时，利用服务器端进行消息的处理与传递。

选择理由：

* 打车软件系统具有导航、在线支付等功能，需要有较高的实时性，与密安性。
* C/S架构由于只有一层交互，逻辑处理可以在客户端上执行，只需要在需要部分远程数据时利用网络进行数据传递，能够保证数据传递得到简化，响应速度足够快，以满足实时发送位置信息，获取导航数据的功能。
* C/S架构能够通过客户端本身设置的一些安全性措施，例如防火墙，保证软件的安全性，使得用户的交易性，在线支付的数据不被窃取。可以实现多层认证。

其架构流程主要有两层：

* 第一层：客户端——表示层（界面层与逻辑层）

包括司机、打车者使用的移动端APP软件，管理人员使用的PC端管理软件。负责进行用户交互，以及大部分的逻辑处理（除了第三方子系统需要在服务器端处理），要求系统客户端软件界面友好、处理逻辑响应及时、数据传递及时。

* 第二层：服务器模式

负责与第三方在线支付系统，实时导航系统，数据库进行访问，在客户端请求进行在线支付、导航、数据库访问等操作时，能够从第三方系统输入和获取数据，并返回给客户端。

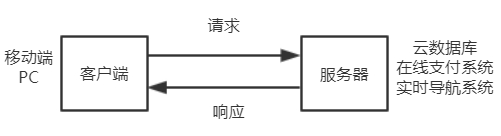


图 3本系统架构图

# 3. 体系架构目标和约束

## 3.1质量要求目标

### 3.1.1影响系统运行质量指标要求

#### 3.1.1.1性能

**时间性能：**

（1） 系统执行速度

打车软件系统应当及时完成操作响应，系统使用过程中的每个简单操作响应时间不超过0.5秒。

（2） 对于复杂操作，系统响应时间考虑到同时进行操作的用户人数，系统的响应时间应该达到如下标准比较合理：

a) 打车者发出约车消息，到附近司机能够显示消息的时间差不超过5s，系统应该支持同时10000人发出约车消息。

b) 司机使用导航功能时，实时更新路径信息时，每次更新路径不大于2s, 系统应该支持同时10000人使用导航功能。

c) 支付时，从打车者支付车费，到司机收到车费，时间不超过1分钟。系统应该支持同时10000人使用支付功能。

d) 每次操作数据库的响应时间不大于0.5秒。系统应该支持同时10000人操作数据库。

**空间性能：**

(1)系统使用实时导航功能，导航期间，占用内存不超过200MB

(2)系统处理约车消息时，占用内存不超过50MB

(3)系统操作数据库时，占用内存不超过60MB

#### 3.1.1.2可靠性

1)系统能在较长时间下稳定运行而不出现故障。

2)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。需要为系统管理员提供调试、检测错误的功能。

3)需要定期备份与维护数据库，保障数据库的可靠性，防止因为数据库的硬件故障导致数据丢失，保障及时恢复数据库的数据。

### 3.1.2非运行质量的质量指标要求

#### 3.1.2.1可使用性

1）该系统的可使用性体现在它可以支持多移动平台、操作系统运行，例如安卓、IOS。

2）同时该系统应该具备方便操作的功能。

a) 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。

b) 系统应提供在线帮助界面，方便用户学习操作。

c) 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。

d) 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。

#### 3.1.2.2密安性

（1）该系统应该通过设置防火墙确保数据传输的安全。使用可靠的操作系统 来保证系统的操作安全，确保系统在一个安全可靠的环境下运行。

（2）系统应保证用户信息不泄露，系统配置文件和数据库存储文件应当进行 加密处理。 （3）系统应保证不会因恶意攻击而崩溃，系统开发过程不存在明显漏洞。

（4）系统应当能够保证选取的开发方不存在商业竞争对手或类似的恶意对手。

（5）系统应该使用账号密码登录的方法使得不同类型的用户具有不同的使用权限。

#### 3.1.2.3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。当系统应用于不同城市的公共交通系统之下时，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。

本系统通过对不同部分的功能的系统分块。使得每个部分保持高内聚和低耦合。增强可维护性。

#### 3.1.2.4 可移植性

（1）易安装性：该系统能够跨平台移动运行，包括安卓平台和IOS平台

（2）共存性：系统应当能够和其他软件共存于一个平台上。

（3）已替换性：系统可被容易地卸载，也易被高版本的系统替换。

#### 3.1.2.5可扩展性

本系统项目具有一定的可扩展性，可将功能扩展到寻找附近的共享单车、共享汽车、寻求代驾等功能上，扩展功能时需要提交需求更变申请，根据规定审核，再进行扩展部分实现。系统代码结构应该低耦合，高内聚，使得扩展新需求时不容易造成结构混乱。

## 3.2约束条件

### 3.2.1 设计约束

1)软件开发选择的主要平台为Android平台和IOS平台

同时需要在PC的windows平台为管理人员设计管理用的软件。

2)Web服务器部署于 Linux 系统之上以保证提供稳定可靠的服务。

3)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

4）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。

5）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。

6）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。

### 3.2.2 逻辑数据库需求

1）基本需求

* 1. 系统推荐使用阿里云的云数据库，使用起来更加简单方便安全
  2. 若经费允许，可使用 Oracle 等大型数据库，可靠性较高，稳定性较高。
  3. 数据库的字段定义应具有一定的灵活性，保证一定的可修改性。
  4. 数据库表的设计应保证一致性、完整性，避免出现数据冗余，出现数据不一致 现象应进行及时的调整。
  5. 满足以上的条件下，数据库数据存储硬顶尽量节省空间。

2.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。

### 3.2.3业务约束

1. 打车软件系统必须能够以移动端app的形式让用户在线自主下载。
2. 打车软件系统必须保护所有注册者的个人信息与出行、交易记录，免受未经授权的访问。所有远程访问都受用户标识和密码控制。
3. 打车软件系统将作为客户/服务器系统实现，客户端部分部署于用户的移动端上，服务器部分部署在使用该系统的公司的UNIX服务器上。
4. 打车软件系统必须管理好用户账号的注册权限，只有具有一定驾驶资格的用户才能注册为司机，只有特定的内部人员可以注册为管理人员。

## 3.3各相关方对体系结构的要求

在打车软件系统的开发过程中，存在相关利益方有以下五类：

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

所以我们体系结构需要完成以下目标：

1. 最终用户对体系结构的要求是：功能清晰、完善；特性完备；
2. 程序员对体系结构的要求是：层次清晰，易于软件的管理；
3. 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；
4. 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；
5. 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。

# 4. 用例视角

阅读人群：用户、分析人员、测试人员

关注点：特性。

软件体系结构的用例视图的描述。用例视图是选择作为迭代焦点的场景集和用例集的重要输入。用例视图描述了一组场景和用例，它们表示一些重要的中心功能。用例视图还描述了一组场景和用例，这些场景和用例具有相当大的体系结构覆盖范围（可以使用许多体系结构元素），或者强调或说明体系结构的一个特性的关注点。

打车软件系统中主要有打车者、司机、业务监测员、系统管理人员这四个角色，我们通过用例图，呈现参与者与功能用例之间的关系。

打车软件系统用例包括：

**- 注册/登录 - 个人信息管理 - 呼叫/预约打车 - 在线支付 - 导航**

**- 投诉/投诉处理 - 业务记录 - 管理/备份数据库 - 处理系统运行异常**

## 4.1 体系结构用例

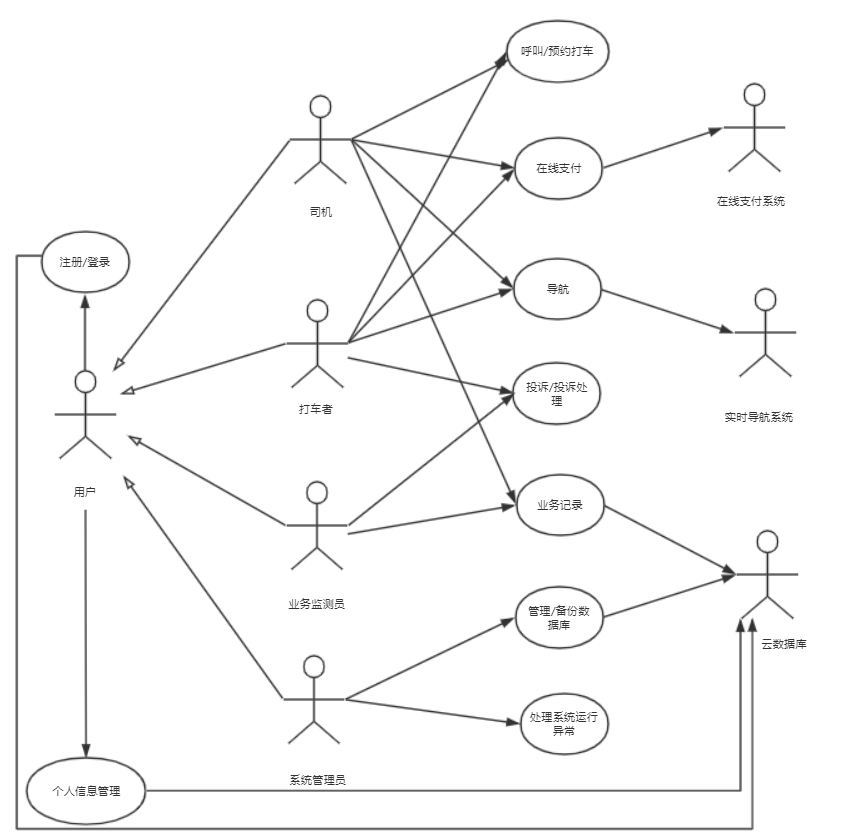


图 4用例视角用例图

### 4.1.1注册/登录

本系统一个主要有四种使用该系统的角色：司机、打车者、业务监测员、系统管理员。每种角色都需要注册账号以及登录账号，才能使用响应的功能。其主要目的是：1、区分不同账号之间的权限、每个账号只能访问和查看与自己有关的数据，而不能获取其他用户的数据。2、区分不同类型账号之间的权限，每种账号只能使用与账号类型相关的功能，司机、打车者、管理员所能使用的功能都不同。在注册司机时需要审核司机的驾驶资格，注册管理员时，需要公司内部的授权。

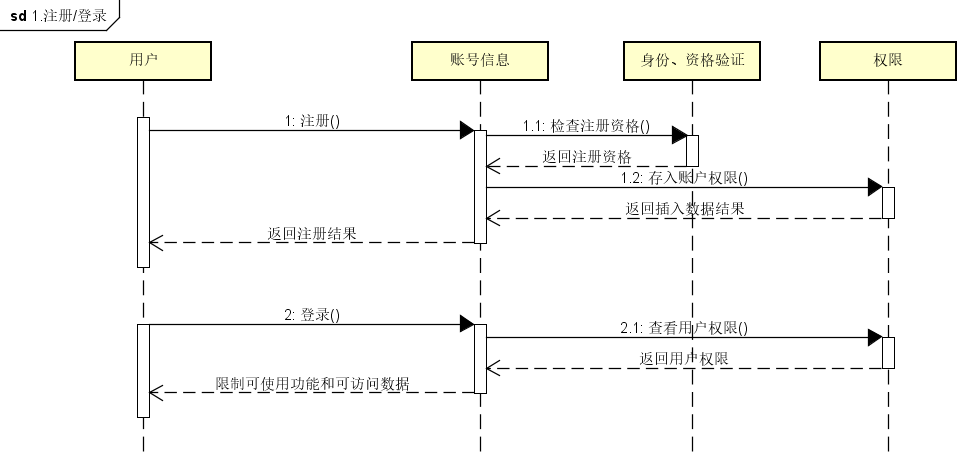


图 5注册/登录顺序图

### 4.1.2个人信息管理

每个用户可以管理自己的账户信息，修改数据库与自己相关的、且权限允许修改的数据。每次进行修改操作，需要进行身份认证、权限认证。系统应该保证每个用户账户信息的安全性，进行多层认证。

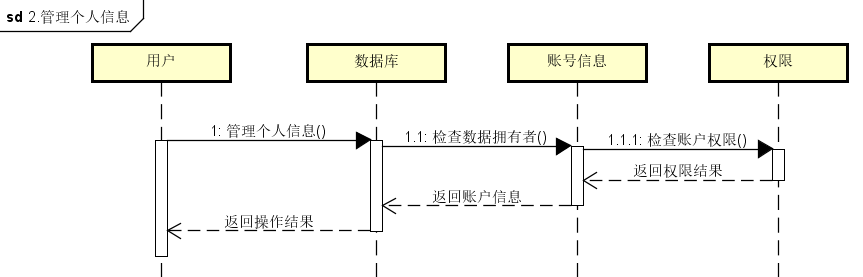


图 6个人信息管理顺序图

### 4.1.3呼叫/预约打车

**打车者**先通过打车软件系统呼叫或预约打车，将订单请求发送给司机。

而**司机**在打车软件系统中选择要接收的订单，完成接单操作。

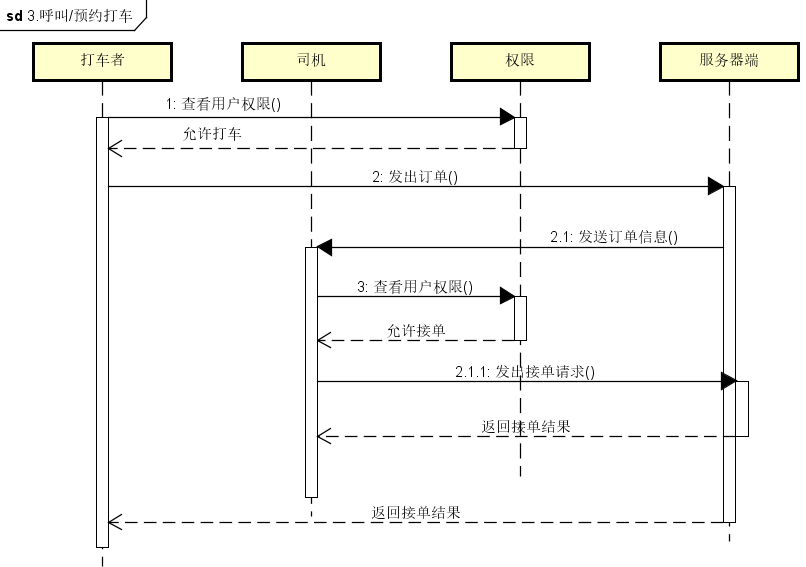


图 7呼叫/预约打车顺序图

### 4.1.4在线支付

**打车者**在完成订单后，通过**在线支付系统**模块，将车费支付给司机。

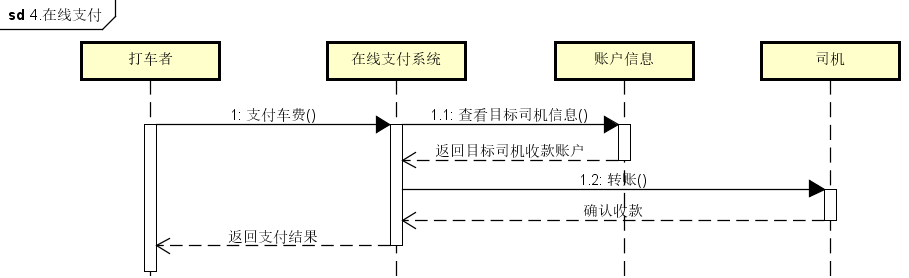
而**司机**同时通过**在线支付系统**，接收来自打车者的车费。  
 

图 8在线支付顺序图

### 4.1.5导航

**打车者**在进行约车操作时，同时将等车地点和目的地地点位置信息发送给**实时导航系统**，方便为司机进行导航服务。

**司机**根据**实时导航系统**传来的导航信息，按照导航路线，前往客户等车地点，并送往目的地。

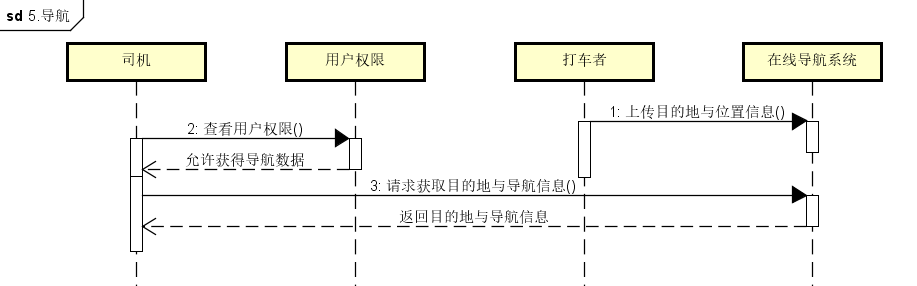


图 9导航顺序图

### 4.1.6投诉/投诉处理

**打车者**在进行打车的过程中，如果遇到不满意的情况，可以发送投诉请求给业务监测员处理。

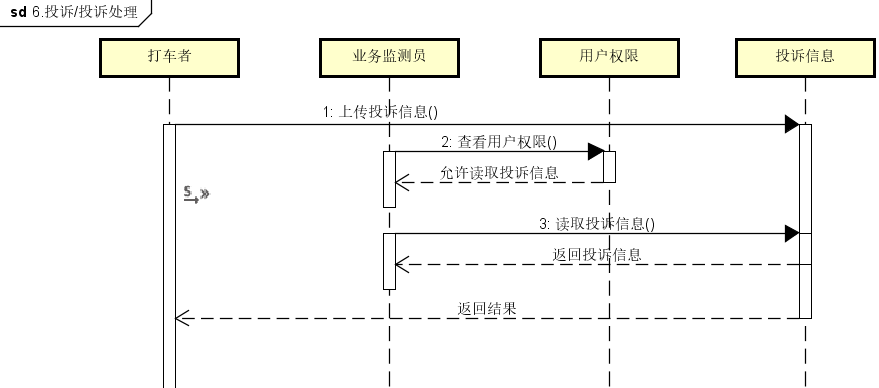
**业务监测员**可以通过打车软件系统结束打车者的投诉信息，根据情况进行投诉处理。 

图 10投诉/投诉处理

### 4.1.7业务记录

**司机**完成业务后，自动上传业务相关信息到**云数据库**保存。因为业务信息包括车费金额等信息，需要等到支付完成后上传。

**业务监测员**可以接收到的上传的业务记录中出现异常的业务记录，以便监测员并根据情况处理

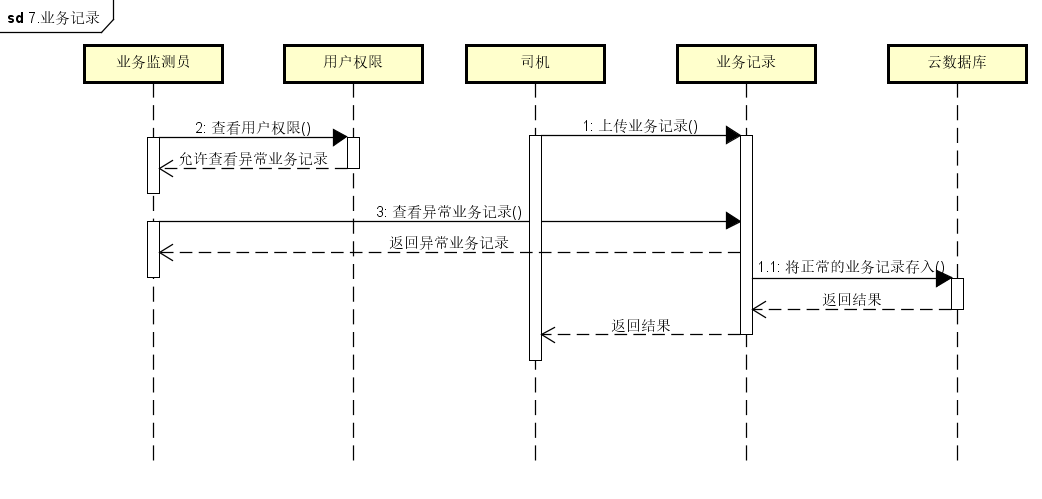


图 11业务记录

### 4.1.8管理/备份数据库

系统管理员需要根据公司要求，对数据库进行管理，或者为了保证数据安全进行数据库备份。

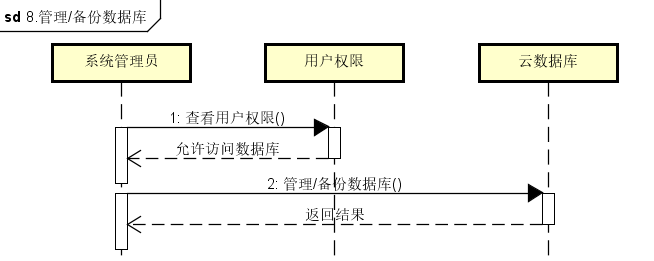


图 12管理/备份数据库

### 4.1.9处理系统运行异常

系统出现错误时，系统管理员及时处理错误，使得系统恢复正常运作。系统管理员可以对系统进行关闭，开启，源码修改等操作。

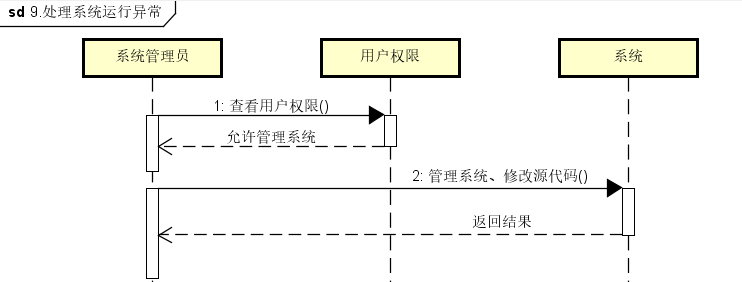


图 13处理系统运行异常

# 5. 逻辑视角

阅读人群：最终用户、设计者

关注点：功能

对体系结构的逻辑视图的描述。描述最重要的类，它们在服务包和子系统中的组织，以及这些子系统在层中的组织。还描述了最重要的用例实现，例如，体系结构的动态方面。类图可以用来说明体系结构上重要的类、子系统、包和层之间的关系。

## 5.1 体系概述-包和子系统分层

本系统的主要使用的是C/S架构。

* 客户端主要关注是与用户进行交互的交互层的模块，以及进行一些数据处理的逻辑层模块。
* 服务器端主要关注的是服务层的数据存储模块以及提供与其他第三方模块交互的接口。

客户端可以分为交互层和逻辑层，加上服务器端的服务层组成三层模块结构。每一层向上一层提供接口，向下一层发送数据与请求。

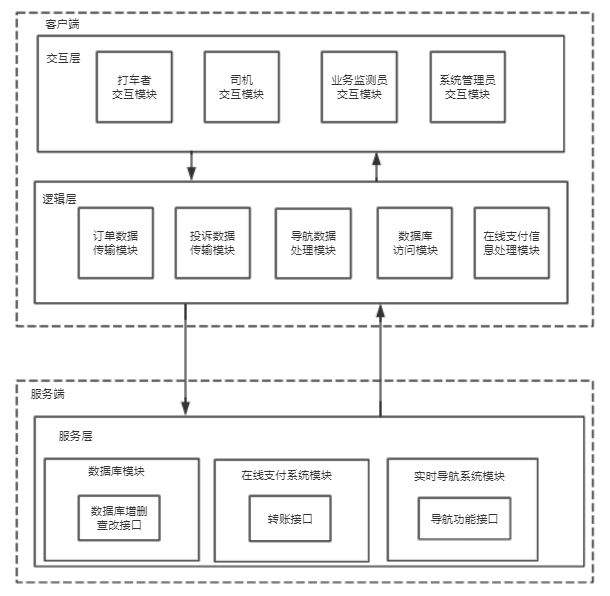


图 14体系概述-包和子系统分层图

### 5.1.1交互层

根据不同角色需要的进行的交互操作，封装成4个交互模块。

登录与个人信息管理的功能是每个交互模块都具有的功能，不重复说明，我们主要关注每个模块不同的功能。

1. **打车者交互模块：**

该模块主要任务是完成打车者的界面交互功能，实现打车者与系统之间的输入输出关系。并且将数据传给逻辑层进行处理。

1. 输入：
2. 打车者输入订单信息：在打车者需要发出打车订单时，读取打车者输入的订单信息。
3. 打车者输入投诉信息：在打车者需要投诉时，读取打车者输入的投诉信息。
4. 打车者输入在线支付信息：在打车者需要在线支付时，读取打车者输入的支付相关信息。
5. 输出：
6. 接单反馈：在有司机接收打车者发出的订单后，输出接单反馈给打车者
7. 投诉处理反馈：在业务监测员处理完投诉后，输出投诉处理反馈给打车者
8. 支付反馈：在在线支付系统完成在线支付功能后，输出支付反馈。
9. 调用下一层的接口：
10. 订单数据传输模块：使用该模块中的接口传输订单数据到服务器端，再分发给智能筛选出的司机
11. 投诉数据传输模块：使用该模块中的接口传输投诉数据到服务器端，再发送给业务监测员
12. 在线支付信息处理模块：使用该模块中的接口传输在线支付信息到服务器端，调用第三方在线支付系统提供的转账接口，转账给目标用户。
13. 导航数据处理模块：使用该模块中的接口将订单中的目的地和当前位置等信息传送给实时导航系统，方便生成导航数据提供给司机。
14. **司机交互模块：**

该模块主要任务是完成打车者的界面交互功能，实现司机与系统之间的输入输出关系。并且将数据传给逻辑层进行处理。

1. 输入：
2. 司机接单请求：在司机选择了想要接收的订单后，读取司机选择的订单信息。
3. 司机输入驾驶资格认证信息： 在司机需要注册司机账户或更新驾驶资格认证信息时，读取司机输入的驾驶资格认证信息。
4. 司机输入导航请求：在司机需要使用导航功能时，读取司机输入的导航请求，以便服务器端的实时导航系统返回导航数据。
5. 输出：
6. 可以选择的订单信息：司机可以看到当前可以选择的订单的信息
7. 收款反馈： 在司机完成订单并收到在线支付系统转来的车费时，输出收款反馈。
8. 驾驶资格认证反馈： 在司机注册或修改司机驾驶资格认证信息后，反馈驾驶资格认证反馈。
9. 实时导航信息：输出实时导航系统返回的导航数据。
10. 调用下一层的接口：
11. 订单数据传输模块：使用该模块中的接口接收来自服务器端的订单数据。
12. 导航数据处理模块：使用该模块中的接口获取来自服务器端的导航数据
13. 数据库访问模块：在进行驾驶资格认证时，使用该模块中的接口访问数据中认证信息，进行认证和修改。
14. 在线支付信息处理模块：使用该模块中的接口接收来自在线支付系统的车费。
15. **业务监测员交互模块**

该模块主要任务是完成业务监测员的界面交互功能，实现司机与系统之间的输入输出关系。并且将数据传给逻辑层进行处理。

1. 输入：
2. 查看投诉信息请求：业务监测员在需要查看投诉信息时读取该请求。
3. 查看异常业务记录信息请求：业务监测员在需要查看异常业务记录信息时读取该请求。

b. 输出：

1.投诉信息：在请求查看投诉信息后，输出投诉信息。

2. 异常业务记录信息: 在请求查看异常业务记录信息后，输出异常业务记录信息。

c. 调用下一层的接口：

1. 投诉数据传输模块：使用该模块中的接口接收来自打车者的投诉信息。
2. 数据库访问模块：使用该模块中的接口接收数据库中异常的业务记录信息
3. **系统管理员交互模块**

该模块主要任务是完成系统管理员的界面交互功能，实现司机与系统之间的输入输出关系。并且将数据传给逻辑层进行处理。

1. 输入：
2. 备份/管理数据库请求：在系统管理员需要备份/管理数据库时，读取该请求
3. 查看系统状态请求：在系统管理员需要检查系统错误时，读取查看系统状态请求
4. 修改系统：在系统管理员修复检查系统错误时，读取系统修改的操作。
5. 输出：
6. 备份/管理数据库反馈：在完成备份/管理数据库操作后，输出反馈。
7. 系统状态：在请求查看系统状态后，输出系统状态信息。
8. 系统修改结果反馈：在修改系统后，输出修改结果反馈

c. 调用下一层的接口：

1. 数据库访问模块：使用该模块中提供的数据库访问接口，访问服务器端的数据库。

### 5.1.2逻辑层

1. **订单数据传输模块：**

该模块主要任务是完成打车者到司机之间的订单信息传输的功能。打车者通过该模块，将订单数据传输给服务器端，服务器端再处理分发给筛选出的司机客户端，司机客户端的订单数据传输模块再接收订单信息，完成订单信息的传输任务。

1. 提供给上一层的接口：
2. 发送订单信息：主要由打车者调用，将参数中的订单信息，通过网络传输给服务器
3. 接收订单信息：主要由司机调用，接收从服务器通过网络传输来的订单信息。
4. 调用下一层的接口：
5. 开发工具包中的SOCKET接口：使用该接口进行远程数据传输。
6. **投诉数据传输模块：**

该模块主要任务是完成打车者到业务监测员之间的投诉数据传输的功能。打车者通过该模块，将投诉数据传输给服务器端，服务器端转发给业务监测员的客户端，业务监测员客户端的投诉数据传输模块再接收投诉数据，完成投诉数据的传输任务。

a. 提供给上一层的接口：

1. 发送投诉信息：主要由打车者调用，将参数中的投诉信息，通过网络传输给服务器

2. 接收投诉信息：主要由业务监测员调用，接收从服务器通过网络传输来的投诉信息。

b. 调用下一层的接口：

1. 开发工具包中的SOCKET接口：使用该接口进行远程数据传输。
2. **导航数据处理模块：**

该模块主要任务是完成发送导航需要的地址信息给服务器层的实时导航系统。以及在实时导航系统计算出的导航数据后，接收来自服务器端的传输。

1. 提供给上一层的接口：
2. 发送地址信息：主要由打车者调用，将订单信息中的地址信息发送给服务器端的实时导航系统。
3. 接收导航数据：主要由司机调用，接收服务器端由实时导航系统计算出的导航数据。
4. 调用下一层的接口：
5. 开发工具包中的SOCKET接口：使用该接口进行远程数据传输。
6. 实时导航系统模块：调用该模块中的接口进行导航路径规划。
7. **数据库访问模块**

该模块主要任务是完成对服务器端数据库进行访问的相关操作。包括司机的驾驶资格认证、系统管理员备份和管理数据库。

a. 提供给上一层的接口：

1. 修改驾驶资格认证信息：主要由司机调用，访问服务器端数据库中的资格认证数据，进行驾驶资格的审核验证。

2. 备份/管理数据库：主要由系统管理员调用，访问服务器端数据库中数据，进行备份、查看、修改等操作。

3. 上传业务记录：主要由司机调用，上传业务记录到数据库中

b. 调用下一层的接口：

1. 开发工具包中的SOCKET接口：使用该接口进行远程数据传输。

2. 数据库模块：调用该模块中的接口对数据库进行增删改查操作。

1. **在线转账模块**

该模块主要任务是从打车者到司机的转账功能

a. 提供给上一层的接口：

1. 支付车费：主要由打车者调用，支付本次订单的车费

2. 接收车费：主要由司机调用，接收本次订单的车费。

b. 调用下一层的接口：

1. 开发工具包中的SOCKET接口：使用该接口进行远程数据传输。

2. 在线支付系统模块：调用该模块中的接口接收来自打车者的车费，并将车费转账到对应司机账户中。

### 5.1.3服务层

1. 数据库模块

该模块的主要功能是提供远程访问数据库的接口，使得用户可以在客户端访问数据库中的内容。

其主要接口为对数据库进行增删改查的接口。

1. 在线支付系统模块

该模块的主要功能是提供在线支付的接口，并完成转账功能。使得用户可以在客户端完成在线支付转账的功能。

其主要接口为从指定账户收取车费，和存入车费到指定账户中。

1. 实时导航系统模块

该模块的主要功能是提供实时导航的接口，并完成导航路径计算，使得用户可以在客户端使用导航功能进行实时导航。

其主要接口为接收来自客户端的地址信息，将计算后的导航路线发送给指定的客户端。

## 5.2 重要的类设计

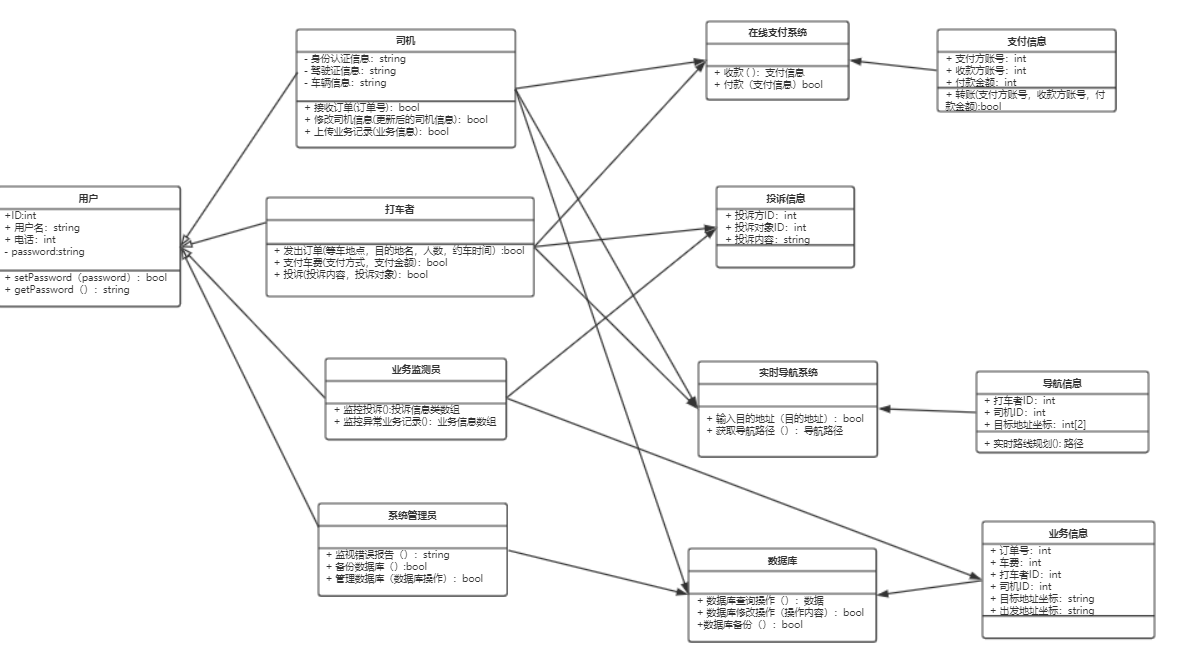


图 15类图

该类图描述了该系统中主要的类和功能，显示了该体系结果的静态结构。

在该逻辑视图中，司机、打车者、业务监测员、系统管理员都是通过用户类泛化来的。

每个角色也有着各自的操作，其分别需要使用到在线支付系统、实时导航系统、数据库等类提供的功能。它们之间的调用逻辑也在类图中表现出来。

# 6. 进程视角

阅读人群：系统集成人员

关注点：性能、伸缩性、吞吐量

体系结构的过程视图的描述。描述系统执行中涉及的任务（进程和线程）、它们的交互和配置。还描述对象和类到任务的分配。

在本系统中，主要的进程有用来与用户进行交互的进程，以及各个功能的进程。在本模块的进程视图描述中忽略了服务器与客户端之中数据传输的模块，只考虑客户端之间的数据传输，例如订单信息从打车者发送到司机。也就是将服务器和客户端看作一个整体来描述。

其中每个客户端会根据登录的账户的类型不同，来选择运行不同的角色交互进程，不同的角色交互进程的目的就是控制不同角色可以调用的功能和访问的权限不同。

## 6.1进程

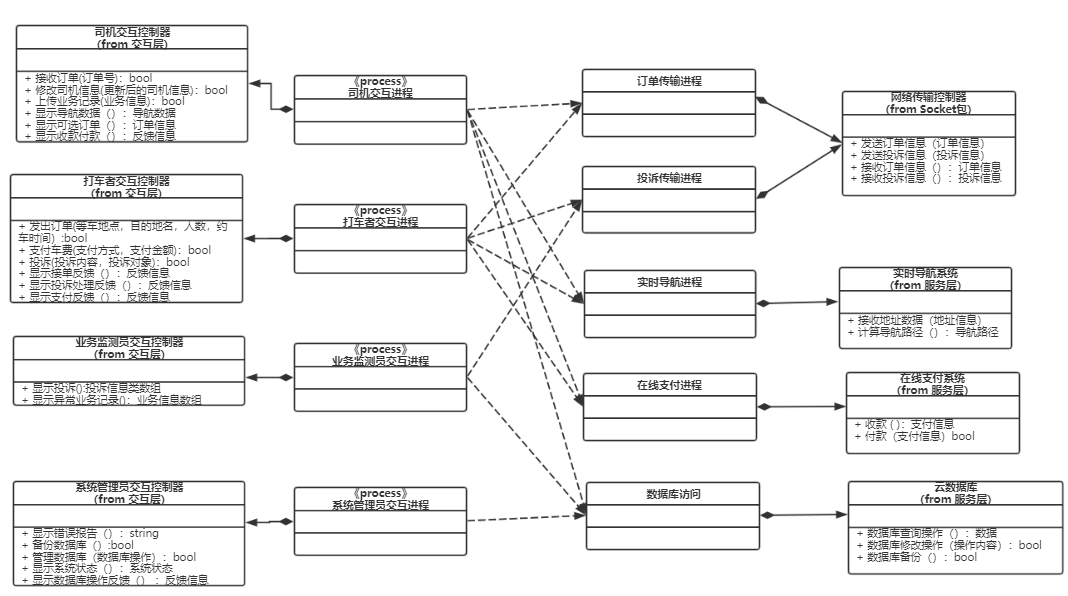


图 16进程图

### 6.1.1司机交互进程

该进程主要用于控制司机账户权限下，系统与用户的交互。调用司机交互控制器。对于每一个司机用户的客户端，都会有这样的一个进程。

其主要功能有：

输入：

+ 接收订单(订单号)：bool

+ 修改司机信息(更新后的司机信息)：bool

+ 上传业务记录(业务信息)：bool

输出：

+ 显示导航数据（）：导航数据

+ 显示可选订单（）：订单信息

+ 显示收款付款（）：反馈信息

### 6.1.2打车者交互进程

该进程主要用于控制打车者账户权限下，系统与用户的交互。调用打车者交互控制器。对于每一个打车者用户的客户端，都会有这样的一个进程。

输入：

+ 发出订单(等车地点，目的地名，人数，约车时间）:bool

+ 支付车费(支付方式，支付金额)：bool

+ 投诉(投诉内容，投诉对象)：bool

输出：

+ 显示接单反馈（）：反馈信息

+ 显示投诉处理反馈（）：反馈信息

+ 显示支付反馈（）：反馈信息

### 6.1.3业务监测员交互进程

该进程主要用于控制业务监测员账户权限下，系统与用户的交互。调用业务监测员交互控制器。对于每一个业务监测员用户的客户端，都会有这样的一个进程。

输出：

+ 显示投诉():投诉信息类数组

+ 显示异常业务记录()：业务信息数组

### 6.1.4系统管理员交互进程

该进程主要用于控制系统管理员账户权限下，系统与用户的交互。调用系统管理员交互控制器，对于每一个系统管理员用户的客户端，都会有这样的一个进程。

输入：

+ 备份数据库（）:bool

+ 管理数据库（数据库操作）：bool

输出：

+ 显示错误报告（）：string

+ 显示系统状态（）：系统状态

+ 显示数据库操作反馈（） ：反馈信息

### 6.1.5订单传输进程

该进程主要用于发送订单数据，主要时为了将打车者客户端发出的订单数据发送到司机的客户端。其主要调用的是开发工具包中的Socket包，来进行数据的网络传输。对于每一个客户端，都会有这样的一个进程，作为网络传输和订单数据处理的进程。

使用的接口：

+ 发送订单信息（订单信息）

+ 接收订单信息（）：订单信息

### 6.1.6投诉传输进程

该进程主要用于发送投诉信息，主要时为了将打车者客户端发出的投诉信息发送到业务监测员的客户端。其主要调用的是开发工具包中的Socket包，来进行数据的网络传输。对于每一个客户端，都会有这样的一个进程。作为网络传输和投诉数据处理的进程。

使用的接口：

+ 发送投诉信息（投诉信息）

+ 接收投诉信息（）：投诉信息

### 6.1.7实时导航进程

该进程主要任务是获取目的地地址信息，并通过该子系统的计算，得到最佳的导航路径规划，使得用户可以使用导航功能进行实时导航。调用了实时导航子系统的接口。

在客户端和服务器端都会有相关的进程，但客户端的只是充当一个适配器来提供接口去访问服务器端的实时导航子系统，服务器端的进程是真正用来进行计算导航数据的。

使用的接口：

+ 接收地址数据（地址信息）

+ 计算导航路径（）：导航路径

### 6.1.6在线支付进程

该进程主要用于实现在线支付车费和接收车费的任务，调用了在线支付子系统提供的接口。

在客户端和服务器端都会有相关的进程，但客户端的只是充当一个适配器来提供接口去访问服务器端的在线支付子系统，服务器端的进程是真正用来进行转账操作的。

使用的接口：

+ 收款 ( )：支付信息

+ 付款（支付信息）bool

### 6.1.6数据库访问

该进程主要任务是完成对数据库进行访问的相关操作。包括司机的驾驶资格认证、系统管理员备份和管理数据库。调用了数据库子系统的接口。

在客户端和服务器端都会有相关的进程，但客户端的只是充当一个适配器来提供接口去访问服务器端的数据库，服务器端的进程是真正用来进行数据库访问的。

使用的接口：

+ 数据库查询操作（）：数据

+ 数据库修改操作（操作内容）：bool

+ 数据库备份（）：bool

## 6.2进程的设计要素

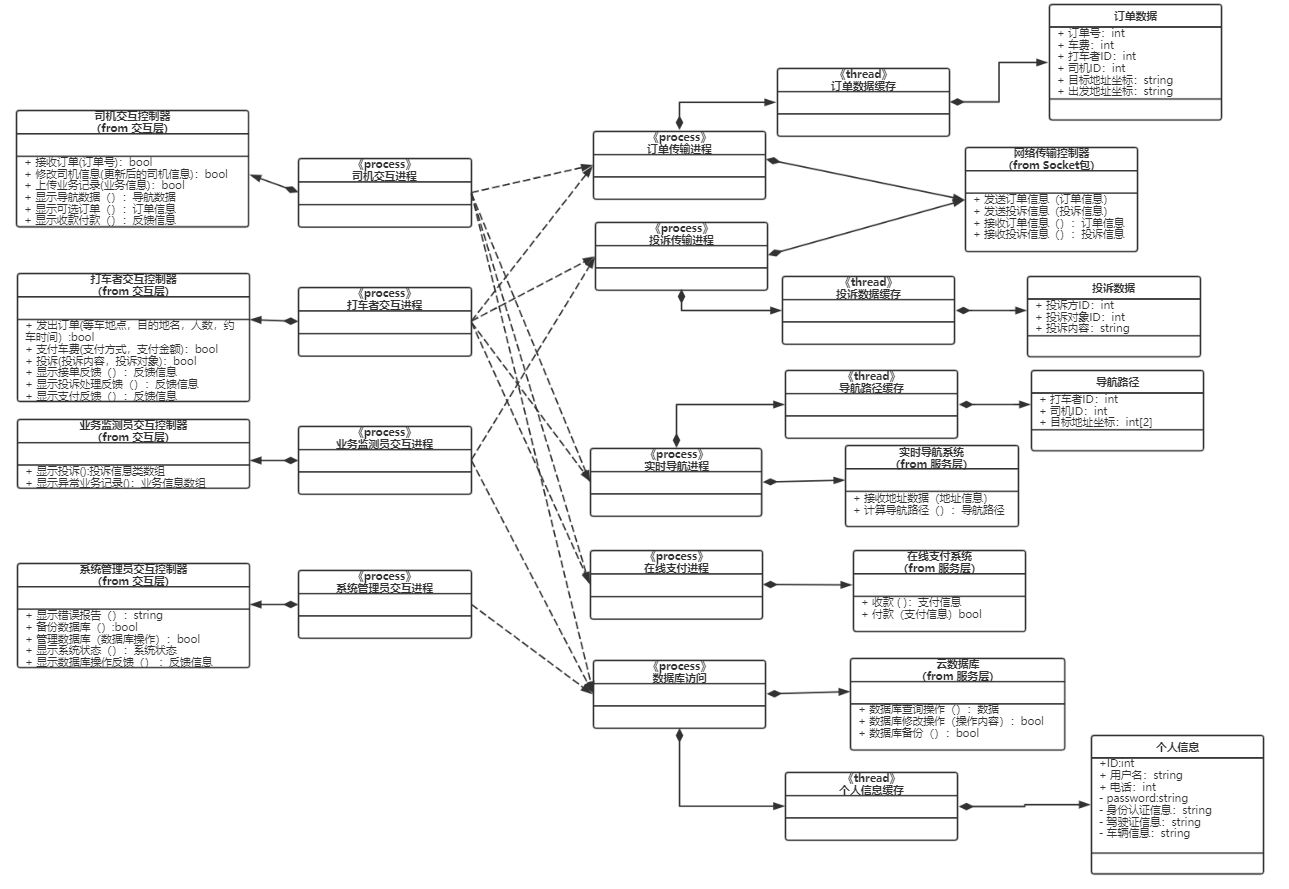


图 17进程的设计要素图

### 6.2.1订单数据缓存线程

该线程用于缓存订单数据。因为只有在完成订单后才会将订单数据作为业务记录的一部分存入数据库，在此之前需要缓存订单信息。且司机需要接收大量订单信息，需要使用该线程对这些订单数据进行管理。

订单信息中需要存储的信息有：

+ 订单号：int

+ 车费：int

+ 打车者ID：int

+ 司机ID：int

+ 目标地址坐标：string

+ 出发地址坐标：string

### 6.2.2投诉数据缓存线程

该线程用于缓存投诉数据。业务监测员需要接收大量投诉信息，需要使用该线程对这些投诉数据进行管理。

投诉信息中需要存储的信息有：

+ 投诉方ID：int

+ 投诉对象ID：int

+ 投诉内容：string

### 6.2.3导航路径缓存线程

该线程用于缓存订单数据。因为只有在完成异常订单后才会将订单数据作为业务记录的一部分存入数据库，在此之前需要缓存订单信息。且司机需要接收大量订单信息，需要使用该线程对这些订单数据进行管理。

导航路径信息中需要存储的信息有：

+ 打车者ID：int

+ 司机ID：int

+ 目标地址坐标：int[2]

### 6.2.4个人信息缓存线程

该线程用于管理数据库访问时产生的个人信息缓存。来提异步检索个人信息数据。

个人信息中需要存储的信息有：

+ID: int

+ 用户名：string

+ 电话：int

- password: string  
- 身份认证信息：string

- 驾驶证信息：string

- 车辆信息：string

## 6.3进程模型的设计模型依赖关系

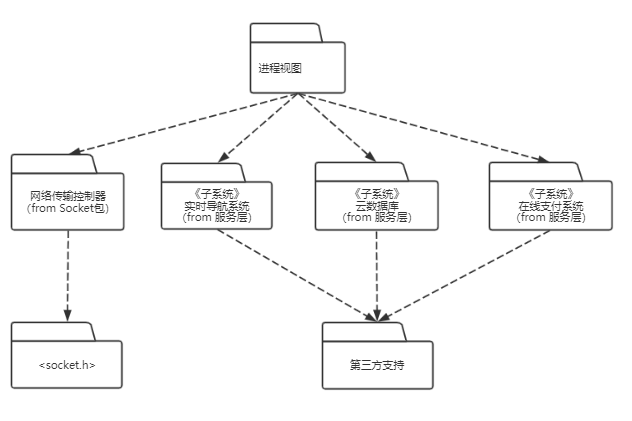


图 18进程模型的设计模型依赖关系图

本体系架构中的依赖关系主要有三个子系统：实时导航系统，数据库，在线支付系统。它们都是依赖与第三方公司的支持。

进行网络传输的模块需要依赖于开发工具中的socket包。

## 6.4进程实现

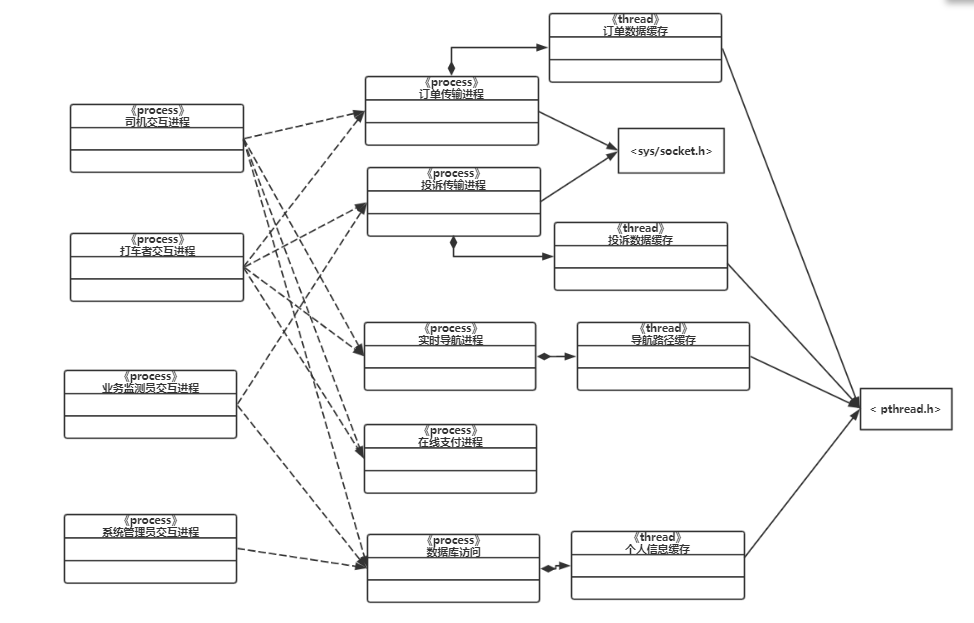


图 19进程实现图

### 6.4.1<sys/socket.h>

在订单传输进程与投诉传输进程中，需要进行数据通过Internet传输，必须包含<sys/socket.h>头文件，使用其中的调用来实现远程数据传输。所以需要进行远程数据传输的进程都需要调用此头文件。

### 6.4.1<pthread.h>

包含<pthread.h>头文件以使用线程的功能。

每个线程都有优先级。优先级较高的线程优先于优先级较低的线程执行。每个线程也可以标记为守护进程，也可以不标记为守护进程。当在某个线程中运行的代码创建一个新线程对象时，新线程的优先级最初设置为等于创建线程的优先级，并且仅当创建线程的进程是守护进程时才是守护进程线程。

# 7. 部署视角

阅读人群：系统工程师

关注点：交付、安装、交流

架构的部署视图描述了最典型的平台配置的各种物理节点。还描述了任务（从流程视图）到物理节点的分配。

本节按物理网络配置进行组织；每个这样的配置都由一个部署图进行说明，然后是进程到每个处理器的映射。

本体系结构的PC和移动端都是作为客户端来部署，但是其中运行的用户类型不一样。

由于本体系结构选择的是C/S架构，所以大部分的逻辑处理都运行在客户端，除了第三方子系统逻辑需要在服务器端和第三方系统处理。

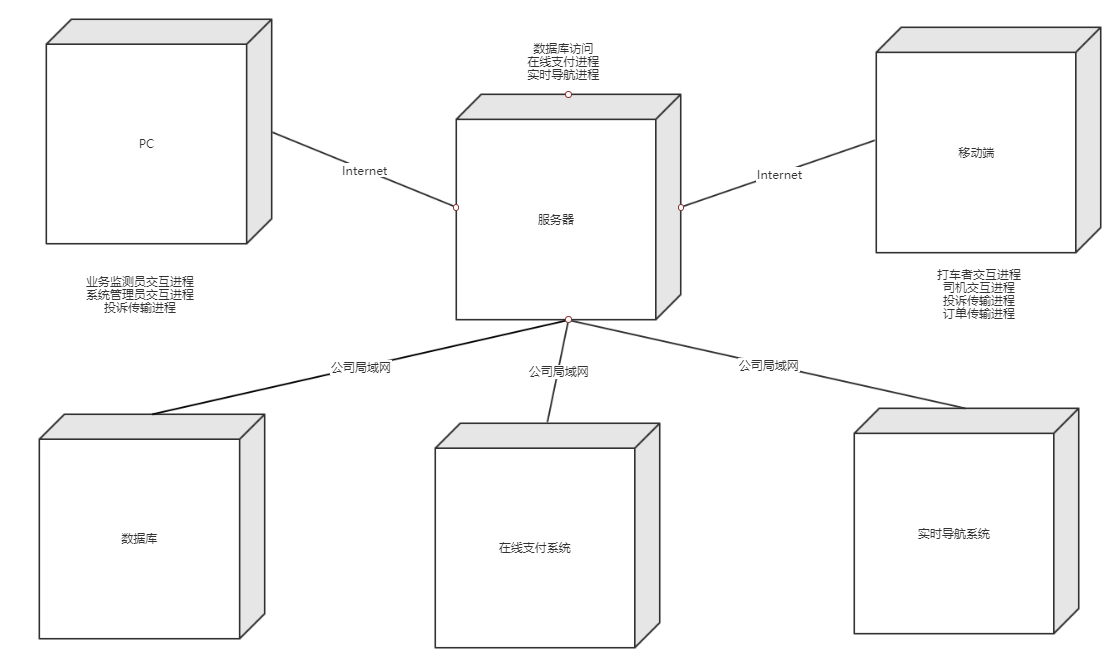


图 20部署视角图

## 7.1 移动端

打车者和司机用户通过移动端来使用打车、导航、在线支付等功能。

移动端通过互联网连接到服务器端。

大部分数据逻辑处理在本地进行。导航、在线支付、数据库存储需要调用服务器的功能。

其运行的主要进程有：

* 打车者交互进程
* 司机交互进程
* 投诉传输进程
* 订单传输进程

## 7.2 PC

业务监测员和系统管理员通过PC端来使用一些系统管理和业务处理的功能。

PC通过互联网连接到服务器端。

大部分数据逻辑处理在本地进行。导航、在线支付、数据库存储需要调用服务器的功能。

其运行的主要进程有：

* 业务监测员交互进程
* 系统管理员交互进程
* 投诉传输进程

## 7.3服务器

服务器主要部署在运营公司内部，且由公司内部的技术人员进行维护和管理。大部分用户需要通过互联网进行访问，公司内部维护人员也可以通过局域网进行访问。服务器访问数据库、实时导航系统、在线支付系统是通过公司内部的局域网进行访问，以保证稳定性和及时性。

其主要处理客户端之间的信息传输，以及响应客户端的请求调用子系统的功能，来实现导航、在线支付等功能。

其运行的主要进程有：

* 数据库访问
* 在线支付进程
* 实时导航进程

## 7.4数据库

是存储个人信息和业务记录的数据库，为用户提供查询修改数据库的接口，可以通过公司内部局域网被服务器访问。

## 7.5实时导航系统

为用户提供实时导航功能，可以通过公司内部局域网被服务器访问。

## 7.6在线支付系统

为用户提供在线支付功能，可以通过公司内部局域网被服务器访问。

# 8. 实现视角

阅读人群：程序员

关注点：软件管理

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

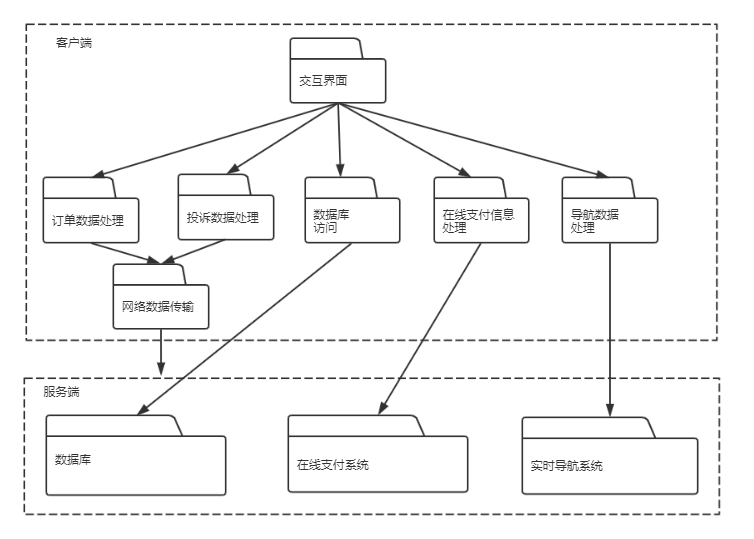


图 21实现视角图

## 8.1实现视角图说明

具体实现视角设计如上图所示，其中为了保证可扩展性和可修改性、可重用性，以下代码包是被封装为可重用的：网络数据传输、数据库访问、数据库、在线支付系统、实时导航系统。

在扩展新功能时，可以使用网络数据传输代码模块封装的接口来进行客户端之间的数据传输。可以使用数据库访问代码模块中封装的数据库访问接口来对服务器端的数据库进行存取、查询操作。

数据库、在线支付系统、实时导航系统可以抽象出作为公司的资产，当其他系统需要使用数据库、在线支付或者实时导航功能时，可以使用已经封装好接口，来使用这些功能。

# 9. 规模与性能

根据第二章节中分析的体系架构目标，可以得到本系统架构支持的关键规模和时间要求：

## 9.1时间性能要求：

1. 系统使用过程中的每个简单操作响应时间不超过0.5秒。
2. 打车者发出约车消息，到附近司机能够显示消息的时间差不超过5s。
3. 司机使用导航功能时，实时更新路径信息时，每次更新路径不大于2s。
4. 支付时，从打车者支付车费，到司机收到车费，时间不超过1分钟。
5. 每次操作数据库的响应时间不大于0.5秒。

## 9.2空间性能要求：

1. 系统使用实时导航功能，导航期间，占用内存不超过200MB
2. 系统处理约车消息时，占用内存不超过50MB
3. 系统操作数据库时，占用内存不超过60MB

## 9.3规模要求：

1. 系统应该支持同时10000人发出约车消息。
2. 系统应该支持同时10000人使用导航功能。
3. 系统应支持在任何给定时间针对中央数据库的最多10000个同时用户，以及在任何时间针对本地服务器的最多2500个同时用户。

## 9.4本系统体系结构如何满足规模与性能

本系统使用C/S架构进行部署。

在C/S架构下大部分逻辑在客户端进行，使得用户进行操作时，响应时间快。

在C/S架构下数据传输量较小，网络传输速度也加快.

在C/S架构下底层处理逻辑在客户端进行，服务器端需要处理压力变小，可以容纳更多用户的并发访问。

本系统通过封装各个功能的接口，且有进程管理每个功能的数据，通过高内聚低耦合的分层代码结构，防止因为调用关系过于复杂导致的占用内存过大。

# 10. 质量

根据第二章节中分析的体系架构目标，可以得到本系统架构支持质量需求：

由于在第二章节中已经详细分析了质量需求的内容，本章节只说明本体系架构如何满足这些需求

## 10.1可靠性

可靠性是保证系统能够长期稳定允许的质量要求。

本体系架构通过以下方法保证系统的可靠性：

1. 加强该系统的故障恢复能力，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，系统能够自动检测错误。同时公司委派系统管理员定期检测和及时修复系统。系统为系统管理员提供了调试、检测错误的功能，以及系统管理员修改系统的接口。
2. 公司委派系统管理员定期备份与维护数据库，保障数据库的可靠性，防止因为数据库的硬件故障导致数据丢失，能够及时恢复数据库的数据。

## 10.2可使用性

可使用性是保证系统能方便用户安装和使用的质量要求

本体系架构通过以下方法保证系统的可使用性：

1. 该系统在多个移动平台、操作系统进行开发，例如安卓、IOS、windows，可以供不同平台的用户进行安装。
2. 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。
3. 系统应提供在线帮助界面，方便用户学习操作。
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. 系统参考了目前大部分网站的扁平化设计，具有一定的美观性。

## 10.3密安性

密安性是保证信息安全性的质量要求

本体系架构通过以下方法保证系统的密安性：

1. 该系统在网络传输模块设置了防火墙，确保系统在一个安全可靠的环境下运行。
2. 该系统对系统配置文件和数据库存储文件进行加密处理，保证用户信息不被泄露。
3. 系统管理员会定期维护系统，及时修复漏洞，保证不会因恶意攻击而崩溃。
4. 系统应该使用账号密码登录的方法使得不同类型的用户具有不同的访问权限，通过多层认证的方法，保证每个用户的访问权限得到控制。

## 10.4可维护性

可维护性是指改进软件的难易程度的质量要求。

本体系架构通过以下方法保证系统的可维护性：

1. 本体系架构通过对不同部分功能的系统分块。使得每个部分保持高内聚和低耦合。增强可维护性。
2. 本体系架构对代码结构进行分层，将功能封装成接口，提供给上一层。抽象出很多可重用的代码。使得系统方便进行维护。

## 10.5 可移植性

可移植性是软件能在不同平台、不同软件环境下安装卸载的质量要求。

本体系架构通过以下方法保证系统的可移植性：

1. 该系统在多个移动平台、操作系统进行开发，例如安卓、IOS、windows，保证系统的易安装性。
2. 该系统通过高内聚的软件架构，保证系统不与其他软件冲突，保证系统的共存性。
3. 该系统通过清晰的分层软件结构，使得软件整体容易彻底卸载或升级为更高版本，保证系统的易替换性。

#### 10.6可扩展性

可扩展性是指系统可以很容易地修改，扩展功能，例如适应共享单车、共享汽车；扩展，接纳更多的司机、打车者，出租汽车公司等。

本体系架构通过以下方法保证系统的可扩展性：

本体系架构为了保证可扩展性，将以下代码包是被封装为可重用的：网络数据传输、数据库访问、数据库、在线支付系统、实时导航系统。

1. 在扩展新功能时，可以使用网络数据传输代码模块封装的接口来进行客户端之间的数据传输。可以使用数据库访问代码模块中封装的数据库访问接口来对服务器端的数据库进行存取、查询操作。
2. 数据库、在线支付系统、实时导航系统可以抽象出作为公司的资产，当其他系统需要使用数据库、在线支付或者实时导航功能时，可以使用已经封装好接口，来使用这些功能。

#### 10.7场景描述方法评价

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。

### 10.7.1用例场景

|  |  |  |
| --- | --- | --- |
| 场景编号 | 场景描述 | 质量评价 |
| 场景1 | 打车者发出约车消息，到附近司机能够显示消息的时间差不超过5s。系统处理约车消息时，占用内存不超过50MB。 | 该场景代表了系统性能要求。 |
| 场景2 | 司机使用导航功能时，实时更新路径信息时，每次更新路径不大于2s。导航期间，占用内存不超过200MB | 该场景代表了系统性能要求。 |
| 场景3 | 支付时，从打车者支付车费，到司机收到车费，时间不超过1分钟。 | 该场景代表了系统性能要求。 |
| 场景4 | 用户每次操作数据库的响应时间不大于0.5秒。系统操作数据库时，占用内存不超过60MB。 | 该场景代表了系统性能要求。 |
| 场景5 | 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，系统能够自动检测错误。同时系统管理员对错误进行检测以及修复系统。 | 该场景代表了系统中用户期望的可靠性。 |
| 场景6 | 系统管理员定期备份与维护数据库。 | 该场景代表了系统中用户期望的可靠性。 |
| 场景7 | 系统管理员定期维护系统 | 该场景代表了系统中用户的访问权限控制，即密安性。 |
| 场景8 | 用户注册需要填写认证信息，司机账户需要驾驶资格认证，业务监测员和系统管理员账户需要公司内部授权。 | 该场景代表了系统中用户的访问权限控制，即密安性。 |

表 2 用例场景表

### 10.7.2增长性场景

|  |  |  |
| --- | --- | --- |
| 场景编号 | 场景描述 | 质量评价 |
| 增长性场景1 | 系统可以扩展服务器的数据库。例如接纳更多的司机、打车者，出租汽车公司 | 该场景代表了系统的可扩展性 |
| 增长性场景2 | 系统可以增加新的功能。例如适应共享单车、共享汽车 | 该场景代表了系统的可扩展性 |

表 3增长性场景

### 10.7.3探索性场景

|  |  |  |
| --- | --- | --- |
| 场景编号 | 场景描述 | 质量评价 |
| 探索性场景1 | 系统可以在不同平台进行安装。 | 该场景代表了系统的可移植性。 |
| 探索性场景2 | 数据库、在线支付系统、实时导航系统可以抽象出作为公司的资产，当其他系统需要使用数据库、在线支付或者实时导航功能时，可以使用已经封装好接口，来使用这些功能。 | 该场景代表了系统的可复用性。 |

表 4探索性场景