# 《轩轩打车》打车软件系统

## 软件体系结构文档

2.0

2019/12/11

陈梦实

2017211960

2017211504 班

软件工程导论

2019 秋季学期

# 修订历史

| 时间 | 版本 | 作者 | 备注 |
|---|---|---|---|
| 2019.12.05 | 版本 1.0 | 陈梦实 | 文档框架编辑、目录调整 |
| 2019.12.08 | 版本 1.1 | 陈梦实 | 对软件体系结构进行总体构思设计，完成概要说明部分 |
| 2019.12.09 | 版本 1.2 | 陈梦实 | 细化体系结构设计 |
| 2019.12.10 | 版本 1.3 | 陈梦实 | 完成剩余部分 |
| 2019.12.11 | 版本 2.0 | 陈梦实 | 最终定稿，检查并上交文档 |

# 文件批准许可

以下需求分析报告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

# 1. 介绍

在这一部分，介绍了本体系结构设计文档的编写目的、范围、用到的参考文献与涉及到的术语以及本体系结构设计文档的组织结构。

## 1.1 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

**系统名称**:《轩轩打车》打车软件系统（Taxi-hailing Software System, TSS）

文档内容：

**文档应用范围**：本软件体系结构设计文档适合于《轩轩打车》打车软件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《轩轩打车》打

车软件系统未来的维护、运行和升级改造等要求。

**系统涉及功能：**

a) 打车者：能够查看当前地理位置信息、周边地图信息；能够选择出发地与目的地并发送订单；能够在行程中查看当前位置及导航；能够在行程结束后缴费并对司机作出评价。
b) 司机：能够查看当前地理位置信息、周边地图信息；能够接受派单；能够在行程中查看当前位置及导航；能够在行程结束后得到计价信息并收费。
c) 业务监测人员：能够查看行程详细信息；能够于系统中联系双方。
d) 系统管理人员：能够检测并处理订单错误；能够对用户信息进行管理。

**系统适用范围：**

系统适用于以下情况：

a) 本打车软件系统的性能最低满足 3 万个司机、3 万个乘客使用。
b) 系统应保持用户基本信息完整,打车者发送订单、司机接受订单、系统派单信息应一致无冗余。
c) 系统保证密安性，不会泄露用户的信息，不允许无特定权限的用户更改信息。
d) 系统推荐使用的 DBMS：MySQL

系统不适用于以下情况：

a) 系统无网络情况（该系统必须在连接网络的条件下才可使用）。
b) 系统无定位信息情况（该系统必须获知用户的位置信息）。

## 1.3 术语定义

| 术语 | 定义 |
| --- | --- |
| TSS | 打车软件系统 |
| o2o | 指将线下的商务机会与互联网结合，让互联网成为线下交易的平台的方式 |

表格 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. 体系结构文档概述

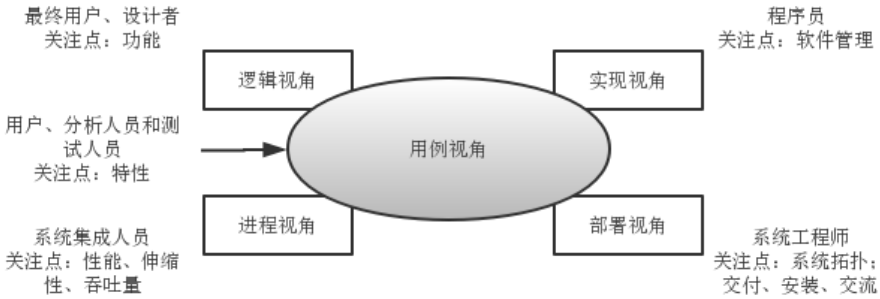
本文档以一系列视图的形式（用例视图，逻辑视图，进程视图，部署视图和实现视图）介绍了体系结构。

概括来说，本文档将按照图 1（即下图）来介绍《轩轩打车》打车软件系统的体系结构。


图表 1：打车软件系统体系结构文档概述图

## 2.1 体系结构视角

本打车软件系统结构文档使用 “4+1 视图模型”（如图 2 所示）进行分析。故本文档从 4 个角度（逻辑、进程、部署和实现）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。




**图表** 2 “4+1”**视图模型**

各方人员要求如下：

1）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，对于本打车软件系统，要求系统能满足打车者、司机、业务监测人员和系统管理人员四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、便捷的处理订单、智能派单、处理异常订单、处理异常行程等功能。

2）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。

3）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。

4）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。

5）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。

## 2.2 体系结构选择

本系统选用“三层 C/S 结构”作为体系结构。具体分析见 5.1 小节。

# 3. 体系结构目标和约束

## 3.1 体系结构目标

《轩轩打车》软件系统是为了改变传统打车方式，建立培养出大移动互联网时代下引领的用户现代化出行方式而开发的软件系统。该系统利用移动互联网特点，将线上线下融合，从打车初始阶段到下车使用线上支付车费，画出一个乘客与司机紧密相连的 o2o 闭环，优化打车者打车体验，改变传统司机等客方式，节约双方沟通成本，节省双方资源与时间。

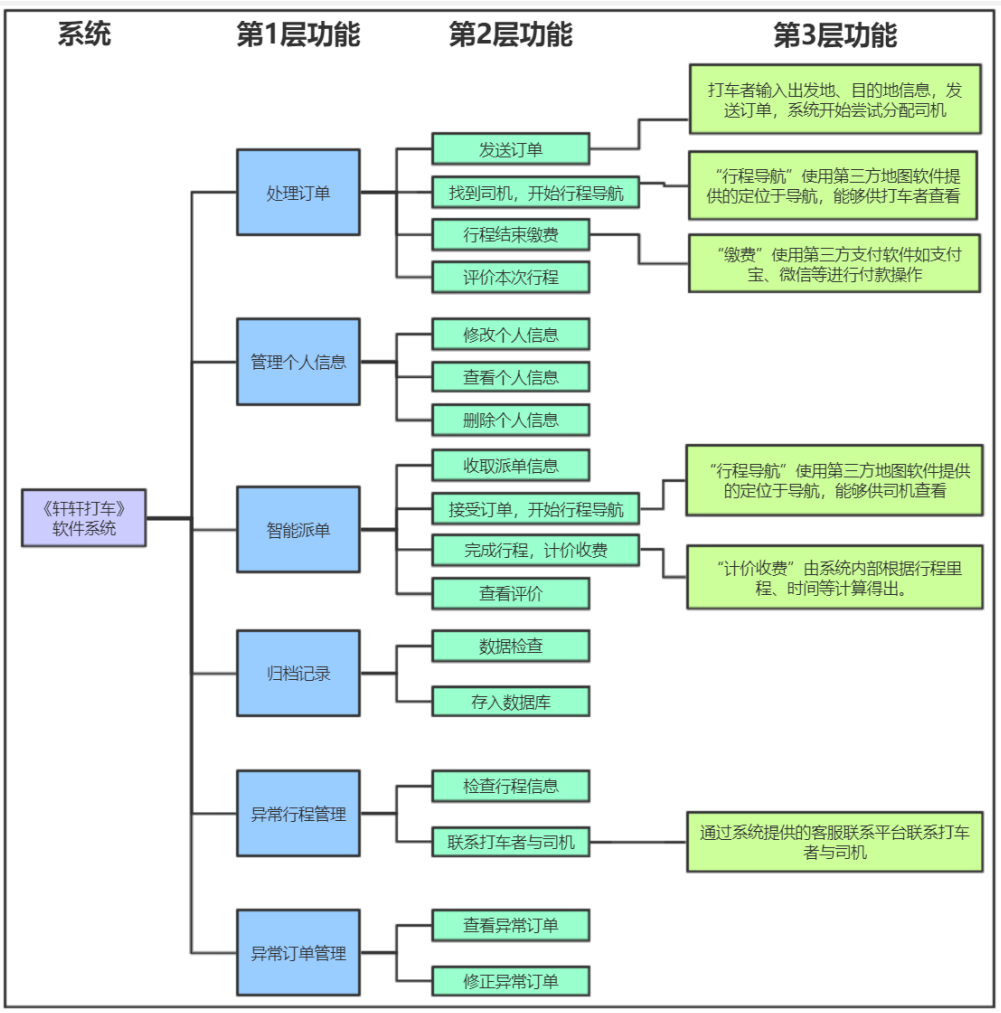
## 3.2 体系结构功能

本系统涉及的功能如下：

- **打车者：**
  - 能够查看当前地理位置信息、周边地图信息；
  - 能够选择出发地与目的地并发送订单；
  - 能够在行程中查看当前位置及导航；
  - 能够在行程结束后查看费用信息并缴费；
  - 能够在行程结束后对司机作出评价；
  - 能够通过该系统联系业务监测人员反馈问题。
- **司机：**
  - 能够查看当前地理位置信息、周边地图信息；
  - 能够接受派单；
  - 能够在行程中查看当前位置及导航；
  - 能够在行程结束后查看计费信息并收费；

- **业务监测人员：**
  - 能够查看某一行程详细信息；
  - 能够查看对应的评价信息及打车者、司机信息；
  - 能够于该系统中联系打车者、司机。
- **系统管理人员：**
  - 能够检测并处理订单错误；
  - 能够对用户信息进行管理。
- **数据库：**
  - 能够对将要存入的数据进行检查；
  - 能够存入数据；
  - 能提供对数据进行增删改查的接口。

系统功能结构图如下：




图表 3 《轩轩打车》软件系统功能结构图

## 3.3 用户特征

最终用户主要包括：打车者、司机、业务监测人员、系统管理人员。前两者主要学会软件如何操作即可。各用户特征如下表所示。

| 用户角色 | 特征 |
| --- | --- |
| 打车者 | 打车者年龄段多样，学习能力也高低不一。如老年人群对新事物的学习能力较弱，但大学生群体则学习较快。因此应设计教学过程、帮助文档等内容帮助打车者学习。<br>具有发布订单、取消订单、查看个人信息、查看订单对应司机部分信息、查看导航、修改个人部分信息的权限。 |
| 司机 | 分正规执照司机与黑车司机两类，应该对应不同司机有选择的派单。<br>具有接受或拒绝订单、查看个人信息、查看订单对应乘客信息、查看导航、修改个人部分信息的权限。 |
| 业务监测人员 | 招聘而来，具有较为专业的纠纷处理能力，能够快速学会系统操作方法，并具有足够耐心。<br>具有查看行程详细信息、查看打车者/司机详细信息、联系打车者/司机、接受问题反馈的权限。 |
| 系统管理人员 | 招聘而来，有专业知识及能力，能够快速学会系统操作方法，有对异常的敏感性。<br>具有检验并修复订单错误的权限，具有必要时手动派单的权限，具有修改用户信息的权限。 |

**表格 2 用户特征表**

## 3.4 体系结构约束

### 3.4.1 基本约束

| 基本要素 | 主要约束 |
| --- | --- |
| 项目基本运行范围 | 完成项目的开发与测试 |
| 项目开发时间 | 6 个月 |
| 项目开发成本 | 20~25 万元 |

| | |
|---|---|
| 支持手机类型 | 当前市面主流机型，如 iphone、华为、小米等 |
| 系统数据 | 具有足够的可靠性与安全性 |

**表格 3 基本约束表**

### 3.4.2 关键指标

1. 系统各功能界面跳转响应时间不超过 1s。
2. 对数据库的简单操作时间不大于 0.5s。
3. 系统应能及时更新打车者和司机的地图导航视图、信息应保持一致性，更新时间不超过 0.1s。
4. 导航地图加载时间不超过 5s，允许先局部生成地图信息再加载边界外的地图。
5. 行程计价算法速度快，计算时间小于 3s。
6. 系统应能允许一万名以上用户同时在线操作，具有较强的并发性。
7. 本打车软件系统应能长时间稳定运行，持续稳定运行时间不小于六小时。

### 3.4.3 设计约束

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：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。

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。

具体见下表：

| 质量或可信赖性属性 | 体系结构的需求 |
| --- | --- |
| 性能 | 对 90%的请求，系统必须在 2 秒内做出相应。 |
| 资源管理 | 服务器部件的内存必须具有 40%以上的预留量。 |
| 易用性 | 客户端必须容易使用。 |
| 可伸缩性 | 软件在高峰时期能够处理 50000 个以上并发用户。 |
| 可修改性 | 体系结构必须支持从当前的第 4 代语言向.NET |
| 密安性 | 所有操作必须被授权，并使用认证过程进行加密。 |
| 可使用性 | 系统必须 24x365 运行，可使用性达到 99%以上。 |
| 可靠性 | 不允许丢失信息，所有信息必须确定在 10 秒内完成。 |

**表格 4 质量或可信赖性属性**

### 3.4.4 假设和依赖

假设：

1. 用户手机具有网络功能并已接通网络；
2. 用户手机支持 GPS 功能；
3. 系统中已有当前城市的地图信息（或第三方导航功能信息）；
4. 司机、打车者已经过认证，数据库中已存储其对应信息。

依赖：

1. 依赖第三方支付平台（如支付宝、微信）完成扣费功能；
2. 依赖第三方导航软件（如高德地图、腾讯地图）等实现导航功能。

# 4. 用例视图

该部分为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/或用例集的重要输入。它描述了一组场景和/或用例，它们表示一些重要的中心功能。它还描述了一组场景和/或用例，这些场景和/或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定的、微妙的点。

## 4.1 体系结构用例

《轩轩打车》软件系统共有四种用户角色：打车者、司机、业务监测人员、系统管理人员。

首先，给出通用的个人信息管理功能介绍：

1) 介绍：

该功能主要面向打车者、司机与系统管理人员，用于不同角色对用户的个人信息进行管理。

2) 输入：

打车者/司机：查看信息请求、修改信息请求及修改数据

系统管理人员：查看信息请求及查看对象、删除信息请求及删除对象

3) 过程：

不同的用户发出不同请求

数据库相应不同请求并返回数据

用户进行查询/修改等操作

更新至数据库

4) 输出：

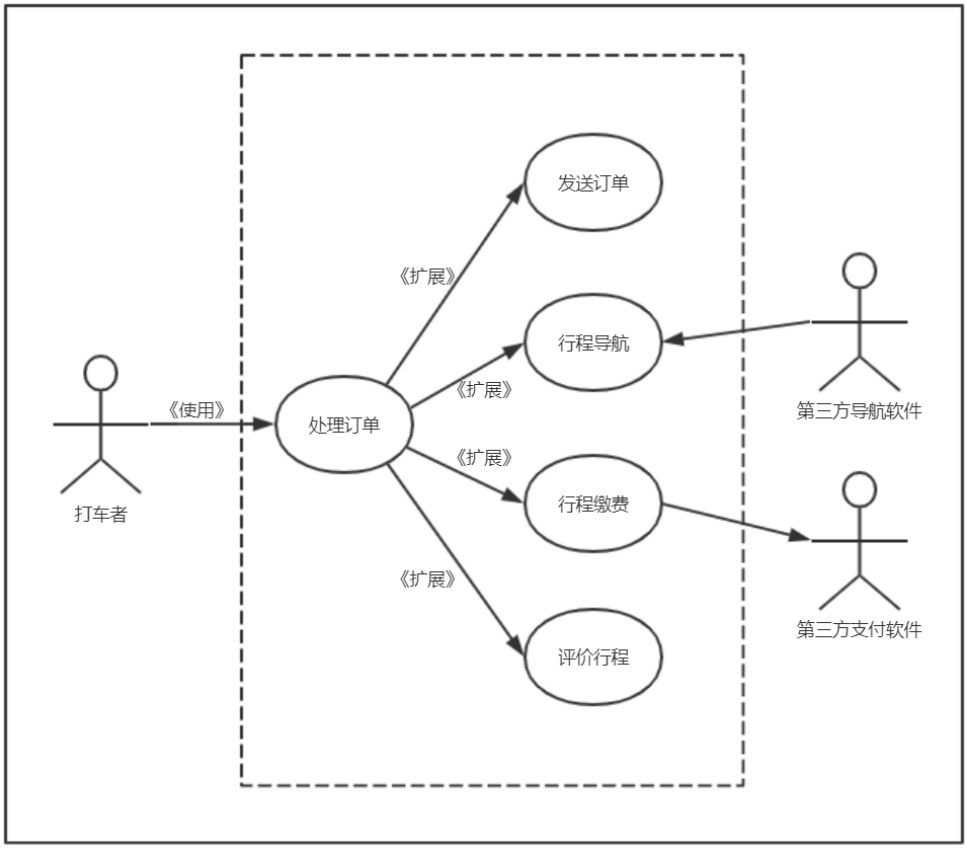
“操作成功”提示

数据库更新

5) 错误处理：

用户对数据库的操作违法，系统提示“操作错误，请检查后再来”

### 4.1.1 打车者




**图表 4 打车者用例图**

设计论述：

1)打车者用户使用的是处理订单功能。

2)处理订单扩展为四个子功能：

1. 发送订单
2. 行程导航
3. 行程缴费
4. 评价行程

3)注：实际打车者用户还使用登录及个人信息修改功能，由于是各用户通用，故不体现在单独的用例图内。

处理订单功能介绍：

1) 介绍：

该功能主要面向打车者，用于处理完整的一个打车流程（打车者视角）。

2) 输入：

打车者输入出发地及目的地信息；

3) 过程：

系统开始自动派单；

分配到司机，创建本次订单的行程信息，开始导航；

当行程结束后，根据行程信息得出缴费金额，通过第三方支付软件进行缴费；

缴费成功后对本次行程做出评价。

4) 输出：

完整的订单信息

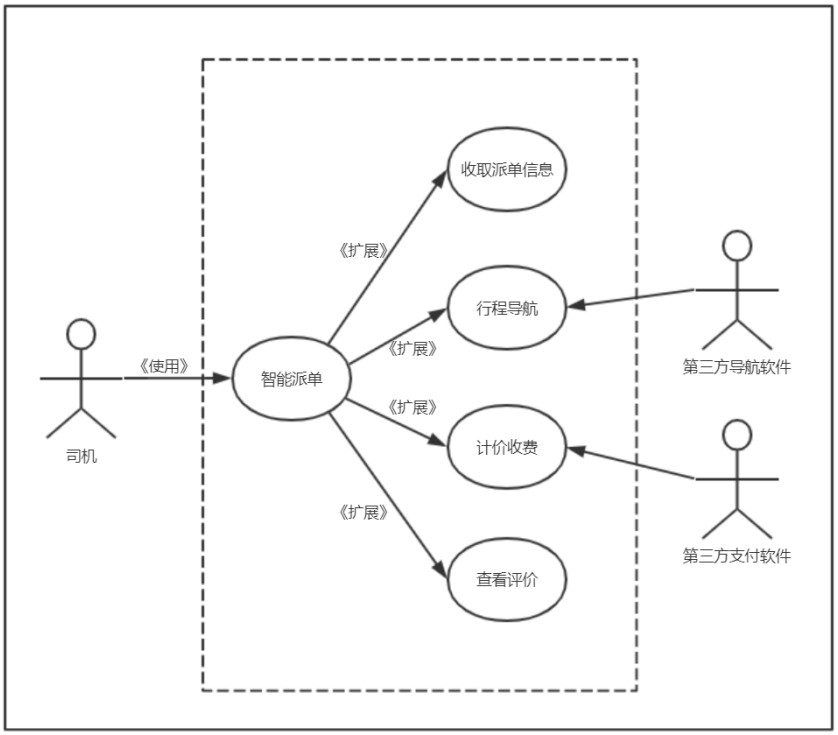
5) 错误处理：

无较近司机接单，系统提示“附近暂无司机，请耐心等待”

排队人数较多，系统提示“当前排队人数较多，请耐心等待”

支付失败，系统提示“支付失败，请重新支付”

### 4.1.2 司机




图表 5 司机用例图

设计论述：

1)司机用户使用的是智能派单功能。

2)智能派单扩展为四个子功能：

1. 收取派单信息
2. 行程导航
3. 计价收费
4. 查看评价

3)注：实际司机用户还使用登录及个人信息修改功能，由于是各用户通用，故不体现在单独的用例图内。

智能派单功能介绍：

1) 介绍：

该功能主要面向司机，用于处理完整的一个打车流程（司机视角）。

2) 输入：

司机准备接单请求

打车者发布的订单

3) 过程：

司机准备接单，系统会根据设计好的算法为司机派单，司机在使用系统时可以接收到派单信息；

司机接受派单后，创建本次订单的行程信息，开始导航；

行程结束后，根据行程信息的时间、里程等因素进行计价收费；

查看本次订单评价。

4) 输出：

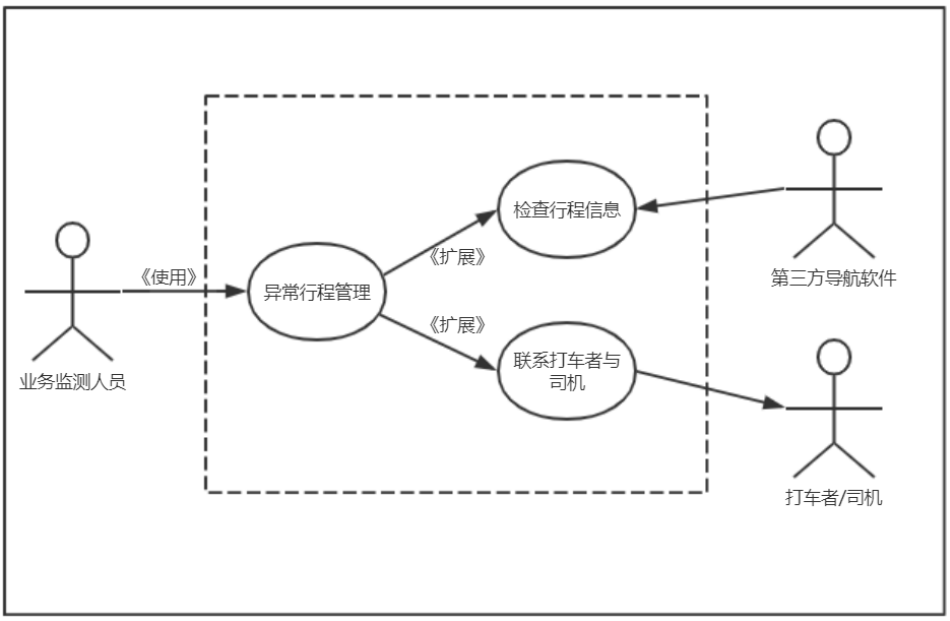
完整的订单信息

5) 错误处理：

无较近乘客发布订单，系统提示“附近暂无乘客”

查看评价时打车者尚未评价，系统提示“打车者尚未对本次服务进行评价”

### 4.1.3 业务监测人员




图表 6 业务监测人员用例图

设计论述：

1)业务监测人员用户使用的是异常行程管理功能。

2)异常行程管理扩展为两个子功能：

1. 检查行程信息
2. 联系打车者与司机

3)注：实际业务监测人员用户还使用登录功能，由于是各用户通用，故不体现在单独的用例图内。

异常行程管理功能介绍：

1) 介绍：

该功能主要面向业务监测人员。用于检查异常的行程并尝试解决。

2) 输入：

行程信息

3) 过程：

业务监测人员查看行程信息；

业务监测人员感到异常，通过系统联系司机与打车者。

4) 输出：

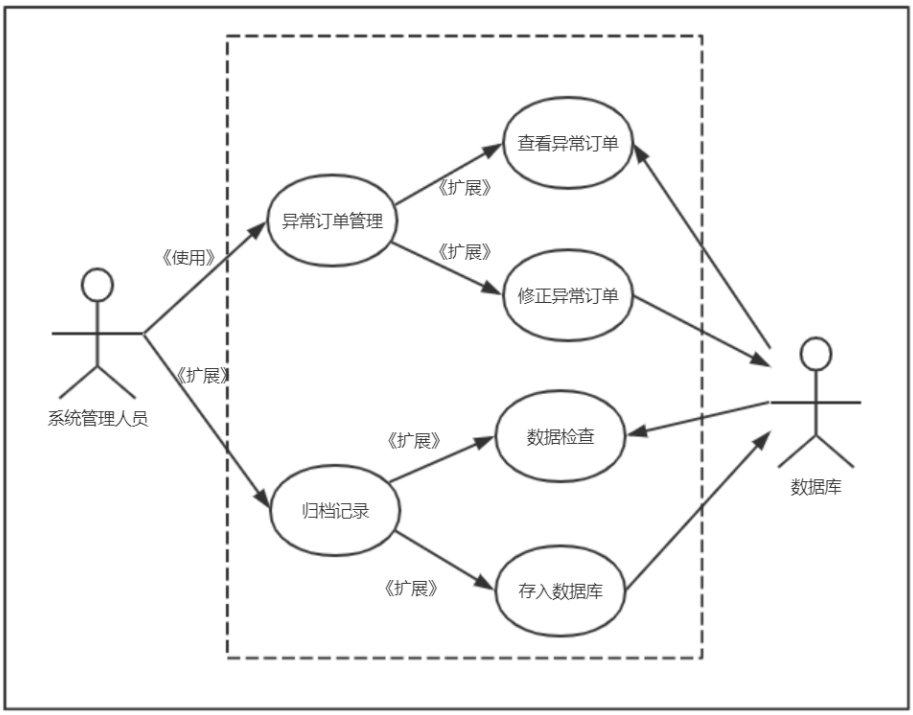
业务监测人员与打车者/司机的联系建立成功

行程恢复正常

5) 错误处理：

业务监测人员看不见行程信息，及时派遣维护人员维修。

### 4.1.4 系统管理人员




图表 7 系统管理人员用例图

设计论述：

1)系统管理人员用户使用的是异常订单管理功能与归档记录功能。

2)异常订单管理扩展为两个子功能：

1. 查看异常订单
2. 修正异常订单

3)归档记录扩展为两个子功能：

1. 数据检查
2. 存入数据库

4)注：实际系统管理人员用户还使用登录及个人信息修改功能，由于是各用户通用，故不体现在单独的用例图内。

异常订单管理功能介绍：

1) 介绍：

该功能主要面向系统管理人员。用于查看异常的订单并修改，再次尝试存储。

2) 输入：

异常订单信息

3) 过程：

数据库在数据检查时发现异常订单，通知系统管理人员；

系统管理人员查看异常订单；

系统管理人员对异常订单进行修改，使其正确；

再次申请存入数据库，转到归档记录功能。

4) 输出：

修正过后的完整订单信息

5) 错误处理：

数据库故障，派遣维修人员维修。

归档记录功能介绍：

1) 介绍：

该功能主要面向系统管理人员，并涉及数据库。用于对完成的订单进行检查并存储。

2) 输入：

完整的订单信息

3) 过程：

某一订单结束后，完整的订单信息自动申请存入数据库；

数据库对要存入的数据进行检查；

检查无误后，存入数据库。

4) 输出：

数据库更新

系统提示操作成功

5）错误处理：

若数据检查不通过，则跳转至异常订单管理功能。

数据库故障，及时派遣维修人员进行维修。

# 5. 逻辑视图

该部分是对体系结构逻辑视图的描述。

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。可以包括类图以说明在结构上重要的类，子系统，包和层之间的关系。

本项目大体逻辑视图如下：


图表 8 《轩轩打车》逻辑视图

## 5.1 体系结构选择

本软件系统体系结构选用**三层 C/S 体系结构**。理由如下：

C/S 结构（Client/Server，客户端/服务器架构），是大家熟知的软件系统体系结构，通过将任务合理分配到 Client 端和 Server 端，降低了系统的通讯开销，需要安装客户端才可进行管理操作。客户端和服务器端的程序不同，用户的程序主要在客户端，服务器端主要提供数据管理、数据共享、数据及系统维护和并发控制等，客户端程序主要完成用户的具体的业务。而三层 C/S 体系结构是在

C/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。

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：

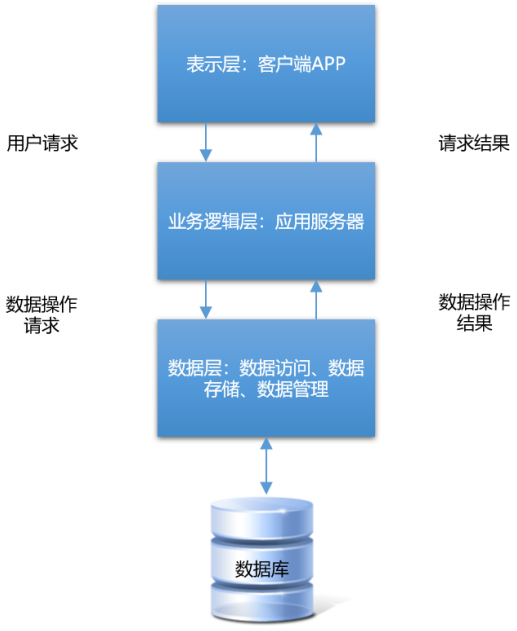
客户端运行用户界面，向用户提供数据输入输出，响应用户动作并控制用户界面，把业务逻辑与用户界面分开，简化客户端的工作。

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。




图表 9 三层 C/S 结构示意图

## 5.2 主要类的设计


**图表 10 类图设计**

通过对本打车软件系统的分析，得出上图六个主要类。下面对每个类的属性与方法进行介绍。

### 5.2.1 打车者类

属性

id：int 类型，系统自动分配的 id。

姓名：String 类型，打车者的姓名。

性别：String 类型，打车者的性别。

电话号：String 类型，打车者的电话号（也用作账号使用）。

密码：String 类型，用于登录的密码。

订单：订单类类型，指向当前打车者发布的订单；若没有当前订单，可为空。

方法

| 方法名 | 方法描述 |
| --- | --- |
| 修改个人信息 | 参数为新信息数值，对应于除订单外的属性；返回值为 bool，若修改成功返回 true。 |
| 查看个人信息 | 查看除订单外的个人信息。 |
| 发布订单 | 参数为出发地、目的地，发布一个订单；返 |

| | |
|---|---|
| | 回值为 bool，若发布成功返回 true。 |
| 查看行程信息 | 查看订单对应的行程信息，主要是查看地图及导航信息。 |
| 支付车费 | 行程结束后，支付对应的车费（与订单类内的方法相关）。 |
| 评价服务 | 支付车费结束后，评价本次服务。 |
| 查看历史订单 | 查看自己历史的订单。 |

**表格 5 打车者类方法表**

### 5.2.2 司机类

属性

id：int 类型，系统自动分配的 id。
姓名：String 类型，司机的姓名。
性别：String 类型，司机者的性别。
电话号：String 类型，司机者的电话号（也用作账号使用）。
密码：String 类型，用于登录的密码。
车型：String 类型，司机的车型信息。
星级：String 类型，司机的评级信息。
订单：订单类类型，指向当前司机接受的订单；若没有当前订单，可为空。

方法

| 方法名 | 方法描述 |
|---|---|
| 修改个人信息 | 参数为新信息数值，对应于除订单外的属性；返回值为 bool，若修改成功返回 true。 |
| 查看个人信息 | 查看除订单外的个人信息。 |
| 接受订单 | 系统开始自动选择订单分派给当前司机，得到订单。 |
| 查看行程信息 | 查看订单对应的行程信息，主要是查看地图及导航信息。 |
| 确认到达目的地 | 系统提示到达目的地（行程类方法）后，该功能可以结束行程，开始计费与评价环节。 |
| 查看历史订单 | 查看自己历史的订单。 |

**表格 6 司机类方法表**

### 5.2.3 业务监测人员类

属性

id：int 类型，系统自动分配的 id。
账号：String 类型，用于登录的账号。
密码：String 类型，用于登录的密码。

方法

| 方法名 | 方法描述 |
| --- | --- |
| 查看行程 | 参数为选中的行程，可以查看选中行程的信息。 |
| 联系行程司机 | 参数为选中的行程，可以根据行程找到其对应的订单，随后找到司机，通过系统与司机建立联系。 |
| 联系行程打车者 | 参数为选中的行程，可以根据行程找到其对应的订单，随后找到打车者，通过系统与打车者建立联系。 |

**表格 7 业务监测人员类方法表**

### 5.2.4 系统管理人员类

属性

id：int 类型，系统自动分配的 id。

账号：String 类型，用于登录的账号。

密码：String 类型，用于登录的密码。

方法

| 方法名 | 方法描述 |
| --- | --- |
| 查看个人信息 | 查看除订单外的个人信息。 |
| 查看用户信息 | 参数为用户（打车者/司机）id，查看对应 id 的用户信息。 |
| 删除用户信息 | 参数为用户（打车者/司机）id，删除对应 id 的用户。 |
| 查看异常订单 | 参数为异常订单，订单类的异常订单提示调用此方法，向系统管理人员弹出提示窗口查看异常订单。 |
| 修改异常订单 | 参数为异常订单，当查看异常订单后调用此方法，修改后再次尝试存入数据库（订单中的存入数据库方法）。 |

**表格 8 系统管理人员类方法表**

### 5.2.5 订单类

属性

id：int 类型，系统自动分配的 id。

打车者：打车者类类型，当前订单对应的打车者，不可为空。

司机：司机类类型，当前订单对应的司机，若还未找到司机，可为空。

出发地：String 类型，出发地信息，不可为空。

目的地：String 类型，目的地信息，不可为空。

订单日期：Date 类型，订单发布的日期，不可为空。

对应行程：行程类类型，对应行程用于地图显示和导航，若还未开始行程，可为空。

计费金额：double 类型，本次订单的计费金额，若还未结束行程，可为空。
支付状态：bool 类型，本次订单的支付状态，已支付为 true。
评价：String 类型，本次订单打车者给司机的评价，若未结束付费，可为空。

方法

| 方法名 | 方法描述 |
| --- | --- |
| 创建行程 | 当匹配到司机时行程开始，使用该方法创建一个行程。 |
| 查看地图及导航 | 查看行程中的地图及导航信息，可被打车者和司机使用。 |
| 完成行程 | 行程到达目的地，司机确认完成行程。 |
| 处理缴费 | 于完成行程后调用，调用打车者缴费方法。 |
| 处理评价 | 于处理缴费成功后调用，调用打车者评价方法。 |
| 存入数据库 | 于评价成功后调用，将当前完整订单信息存入数据库（先查后存）。 |
| 异常订单提示 | 若存入数据库过程中数据检查出现问题，则调用该方法，调用系统管理人员类中的查看异常订单方法。 |

### 5.2.6 行程类

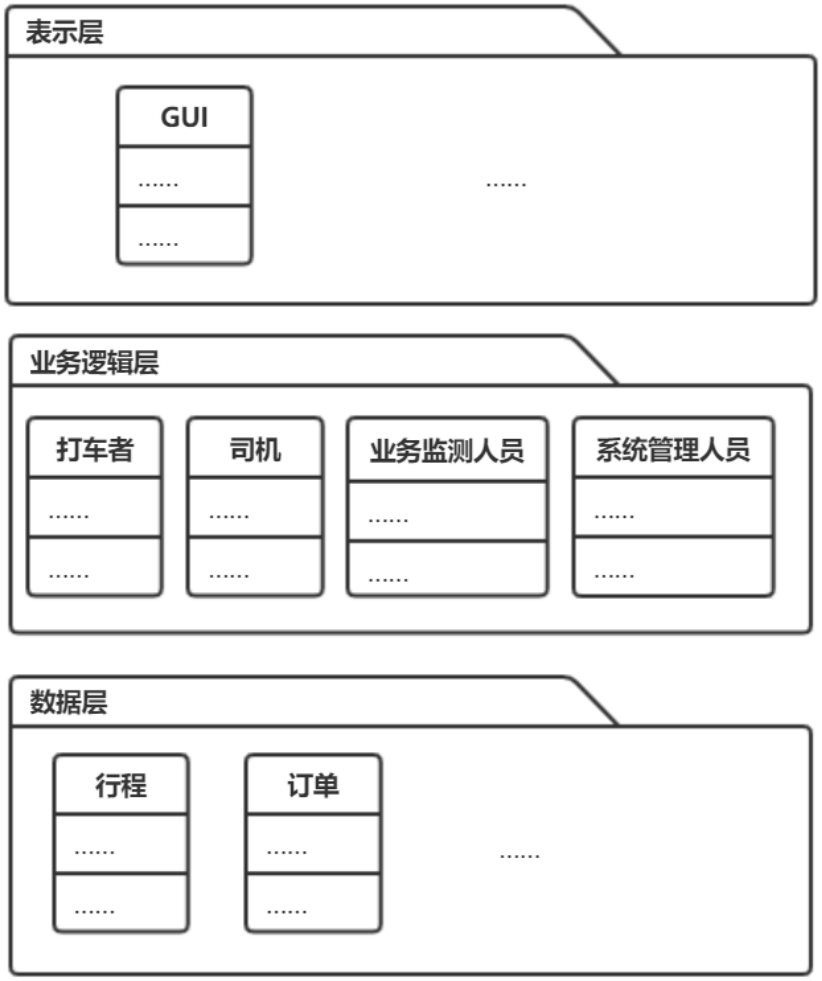
属性
对应订单：订单类类型，为当前行程附属于的订单。
规划路线：string 类型，当前规划的从出发地到目的地的路线提示信息。
里程：double 类型，当前已走过的里程数。
时间：time 类型，为当前行程耗费的时间。

方法

| 方法名 | 方法描述 |
| --- | --- |
| 规划路线并导航 | 根据订单中的出发地及目的地，调用第三方导航软件接口规划路线并导航。 |
| 提示到达目的地 | 若当前定位于目的地重合，则调用司机类中的完成行程方法。 |
| 计算费用 | 司机确认到达目的地后，调用此方法返回本次订单应付金额，返回值为 double 类型。 |

## 5.3 体系概述 - 包和子系统分层

根据 5.1 中对六个主要类的设计，同时考虑三层 C/S 结构，得出分层情况：




图表 11 包分层

表示层：

GUI 界面的相关类。

业务逻辑层：

打车者、司机、业务监测人员与系统管理人员四个类应位于“业务逻辑层”，处理来自用户的各种操作并对数据进行操作。

数据层：

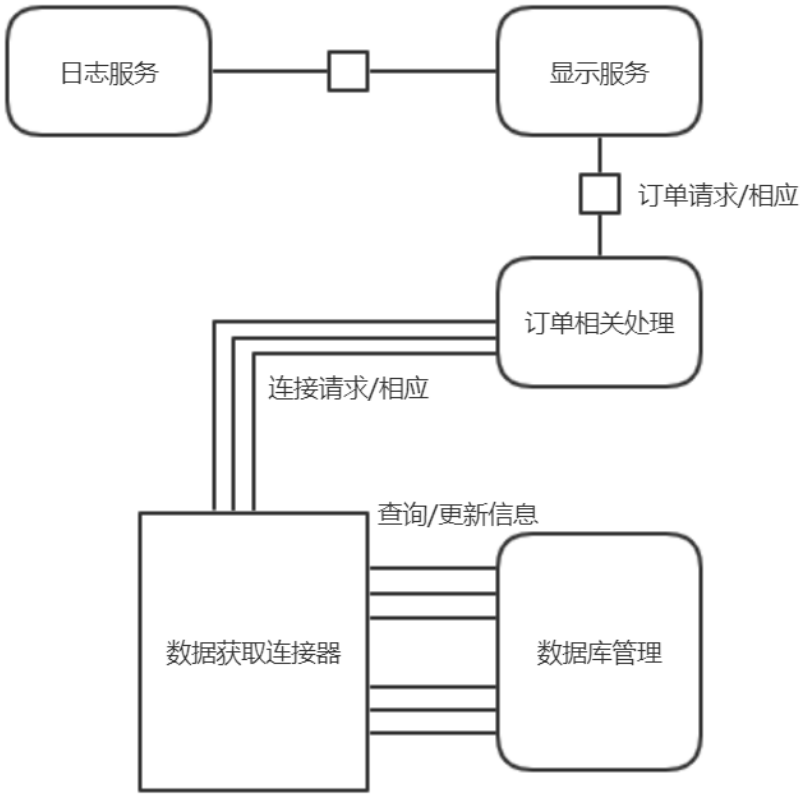
订单与行程这两个类属于系统中的“数据部分”，因此将其放在“数据层”暂存，由业务逻辑层中的类来确定这些数据的流向（存入数据库还是删除等）。

下面给出 DilipSoni 指出的体系结构四类划分如下：

## 5.4 概念级体系结构

概念级的体系结构是从需求工程阶段给出的系统模型直接进化过来的，它说明设计的元素以及元素领域的特定关系。

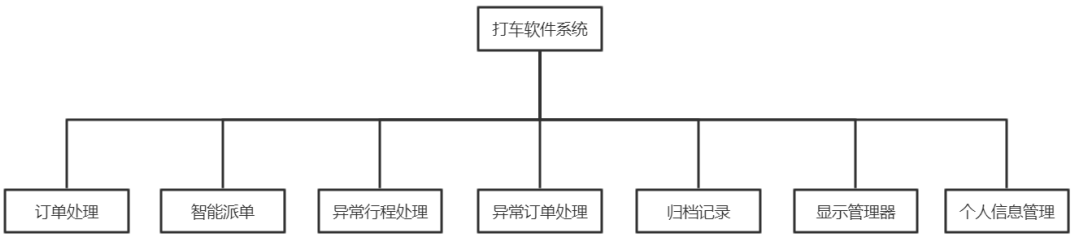
如下图所示，用圆角框代表功能部件，矩形框表示连接器。 除日志服务外，TSS 的概念级体系结构可分为显示服务、订单相关处理和数据库管理三个大功能部件及各自的连接器。



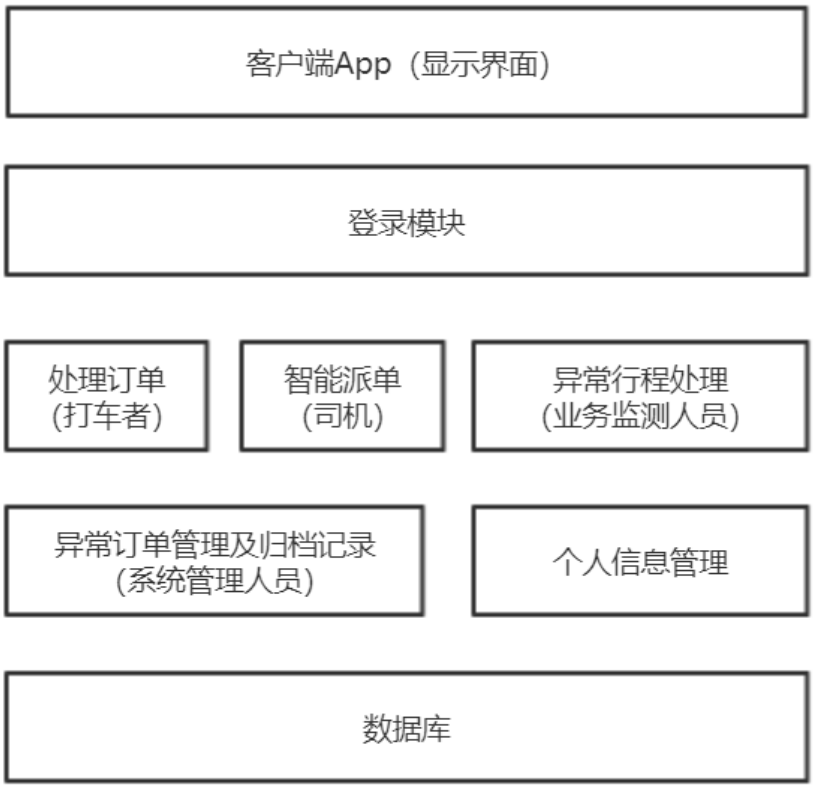

**图表 12 概念级体系结构**

## 5.5 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是现实的期望，特别是对于程序规模较大的系统，下面用两种方式表达模块化：图 12 为将系统按功能从逻辑上分解为系统、模块以及程序单元；图 13 为按系统的层次进行划分。如下：



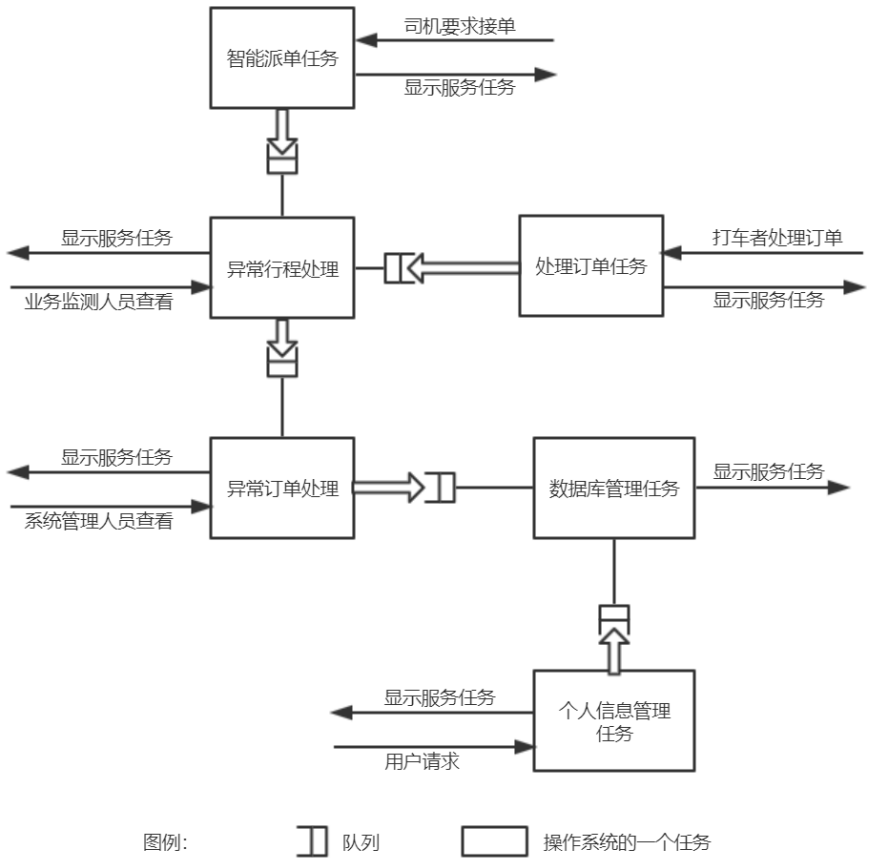

图表 13 打车软件系统的功能分解




图表 14 打车软件系统的分层结构

## 5.6 运行级体系结构

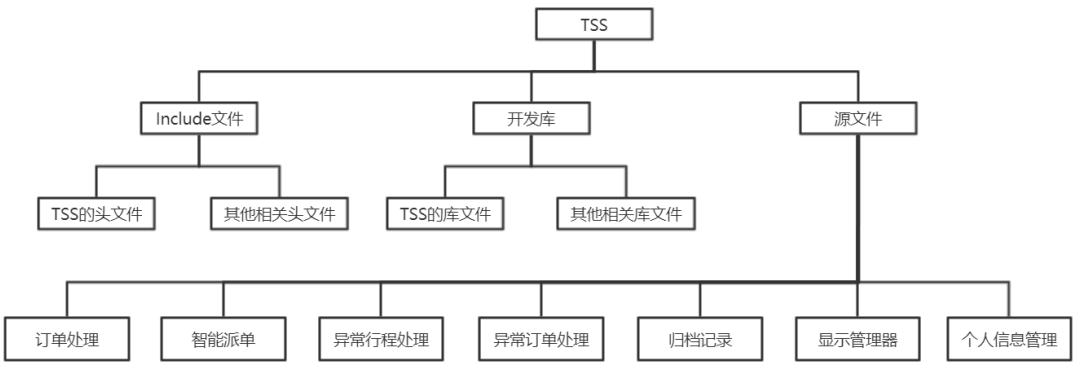
运行级的体系结构用来描述系统的动态结构。如下图




图表 15 运行级体系结构

## 5.7 代码级体系结构

代码体系结构为编程实现提供方便，在编程语言层面按版块、目录、文件和库的形式组织源代码。见下图：

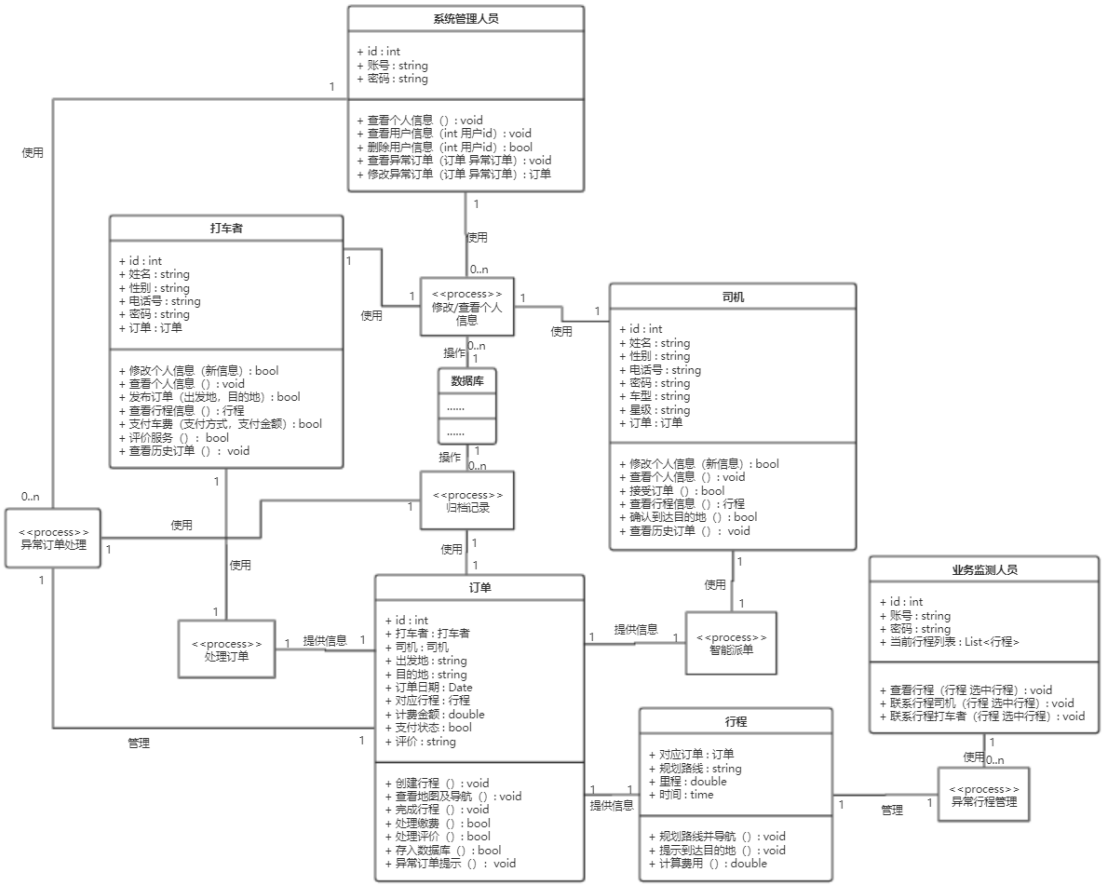



图表 16 代码级体系结构

# 6. 进程视图

该部位为对体系结构的进程视图的描述。描述系统执行中涉及的任务（进程和线程），它们的交互作用和配置。还描述了对象和类对任务的分配。

## 6.1 进程




图表 17 系统进程图

此图中，打车者、司机、业务监测人员与系统管理人员为四个进程，对应于终端。其余进程为不同终端执行，下面依次介绍。

### 6.1.1 修改/查看个人信息

该进程是对存于数据库中的个人信息进行修改与查看，因此多个进程都可能调用同一个数据库，一对多的关系。

打车者、司机进程可能执行此进程，用于查看/修改自己的个人信息，一对

一。

系统管理人员进程可能执行多个此进程，查看或修改多个用户的个人信息。

### 6.1.2 处理订单

此进程由打车者进程执行，负责一次打车流程中“打车者部分”的相关操作。

打车者在开启该进程后，向该进程提供了打车者信息，随后等待派单到司机。

当派单完成后，订单正式开始并生成行程信息，到达终点后打车者进行缴费、评价，随后完成订单，并创建归档记录进程向数据库中存入本次订单完整信息。

### 6.1.3 智能派单

此进程由司机进程执行，负责一次打车流程中“司机部分”的相关操作。

司机在开启该进程后，向该进程提供了司机信息，随后等待系统根据派单策略只能派单给本进程。

当派单完成后，订单正式开始并生成行程信息，到达终点后司机进行收费并可以查看评价，随后完成订单，并创建归档记录进程向数据库中存入本次订单完整信息。

### 6.1.4 异常行程处理

此进程由业务监测人员进程执行。

该进程与某一行程信息是一对一关系，一个进程负责查看并管理一个行程信息。业务监测人员可以使用多个该进程对多个行程情况进行监控并处理。

### 6.1.5 异常订单处理

此进程由系统管理人员进程执行。

该进程与某一完整订单信息是一对一关系，一个进程负责查看并管理一个订单信息。系统管理人员可以使用多个该进程对多个订单异常进行处理，随后使用“归档记录”进程存入数据库。

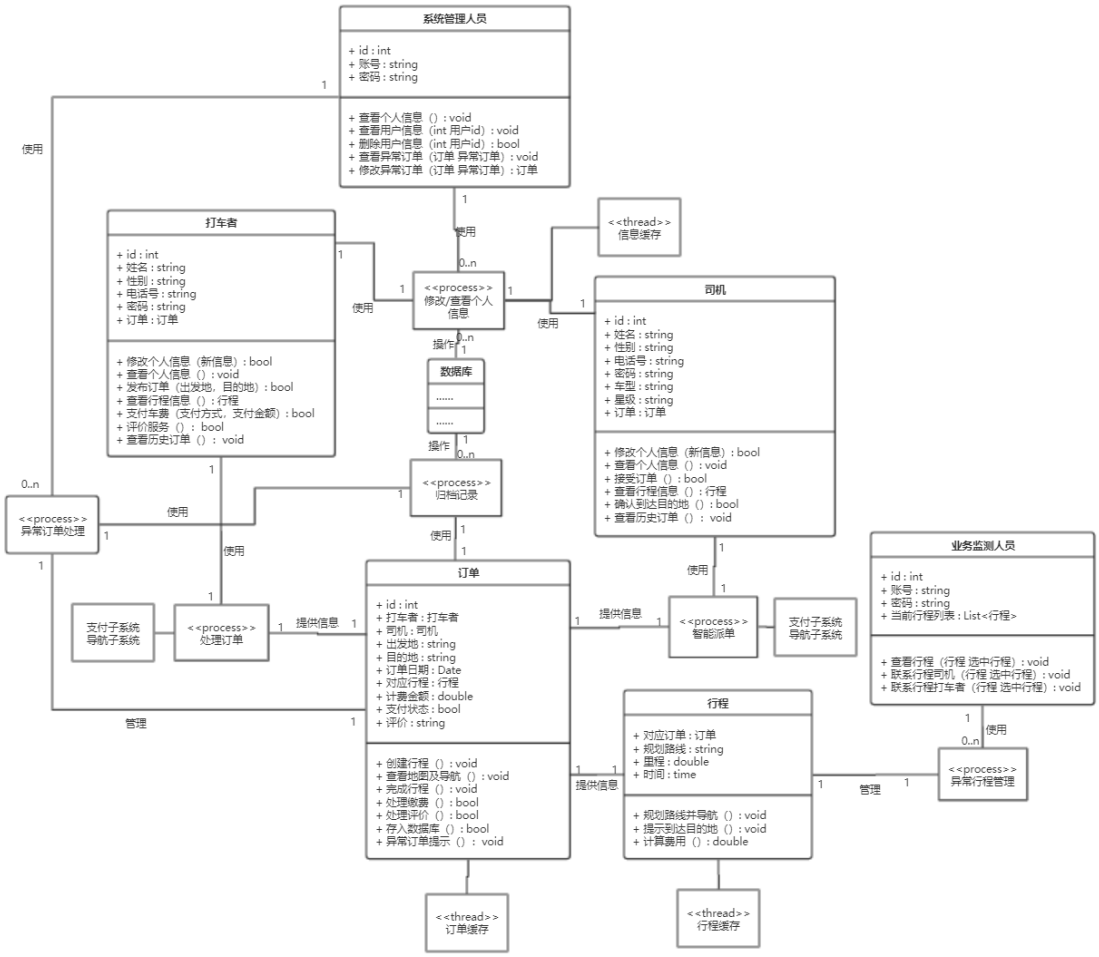
### 6.1.6 归档记录

此进程可能由系统管理人员、打车者或司机进程执行。用于将某一完整订单存入数据库中。

当订单完成且无异常时，由司机或打车者进程使用该进程将订单存入数据库。

当订单完成但出现异常时，首先为异常订单处理进程提供信息，系统管理人员使用异常订单处理进程处理该异常订单，处理完毕后使用归档记录进程存入数据库。

## 6.2 进程的设计要素




**图表 18 进程设计要素**

在本部分，对于处理订单和智能派单进程，其使用到了支付子系统与导航子系统；具有订单信息的进程（处理订单进程、智能派单进程）会开启线程暂存订单、开启线程暂存线程；修改/查看个人信息进程会开启线程缓存个人信息记录。

### 6.2.1 支付子系统与导航子系统

本系统会依赖第三方支付软件与第三方导航软件，具体是在打车行车过程中及结束后使用，故处理订单进程与智能派单进程都用到支付子系统与导航子系统。

### 6.2.2 信息缓存

在查看/修改个人信息时，该进程会开启线程用于暂存从数据库中得来的信息记录，方便查看或修改后再次存回。

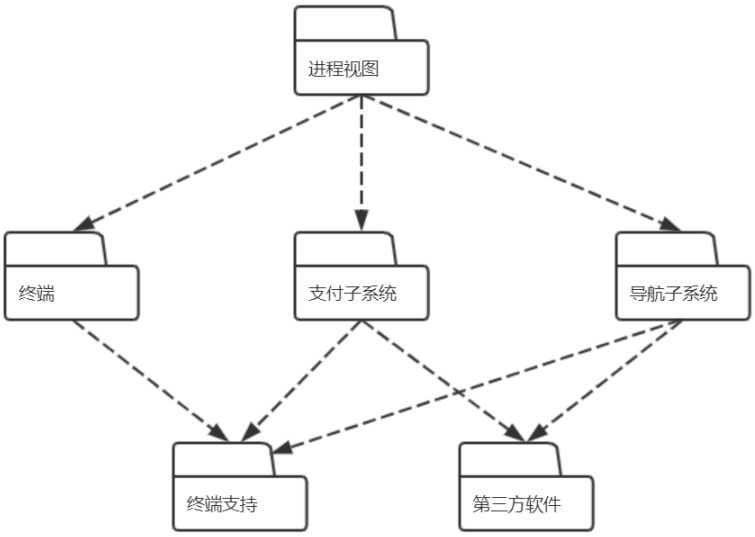
### 6.2.3 订单缓存

对于包含订单信息的进程，该进程会开启线程用于暂存订单信息，方便查看与存入数据库。

### 6.2.4 行程缓存

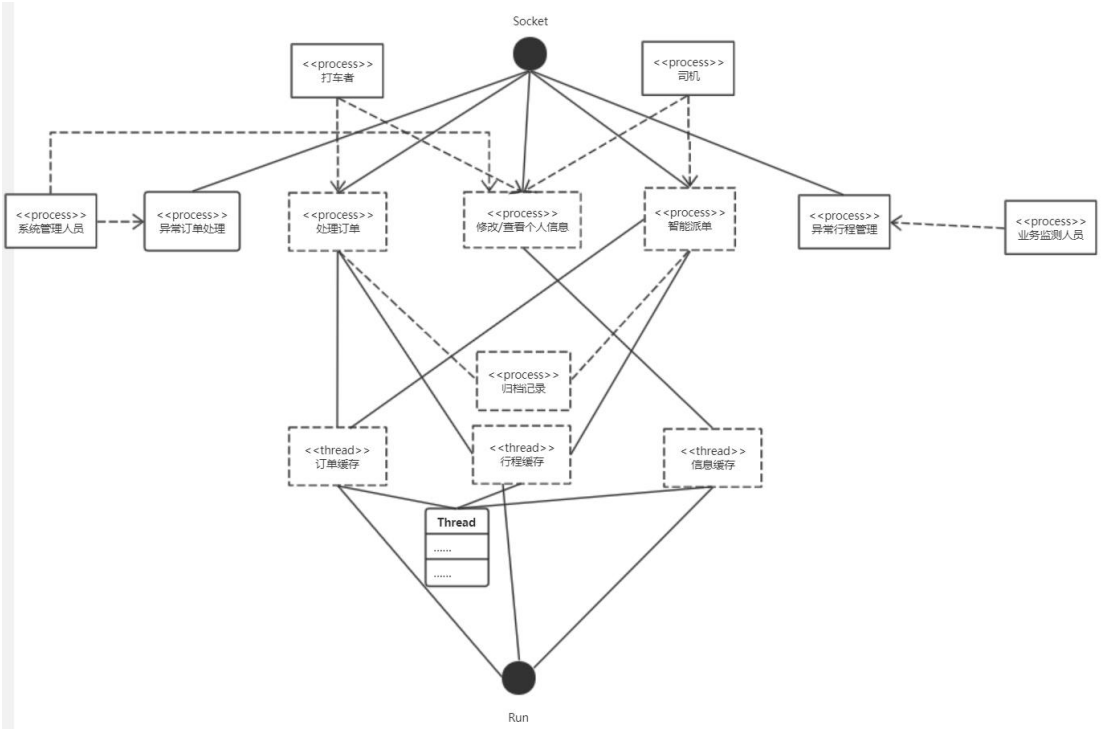
对于包含订单信息的进程，该进程会开启线程用于暂存行程信息，方便查看与存入数据库。

## 6.3 进程模型的设计模型依赖关系



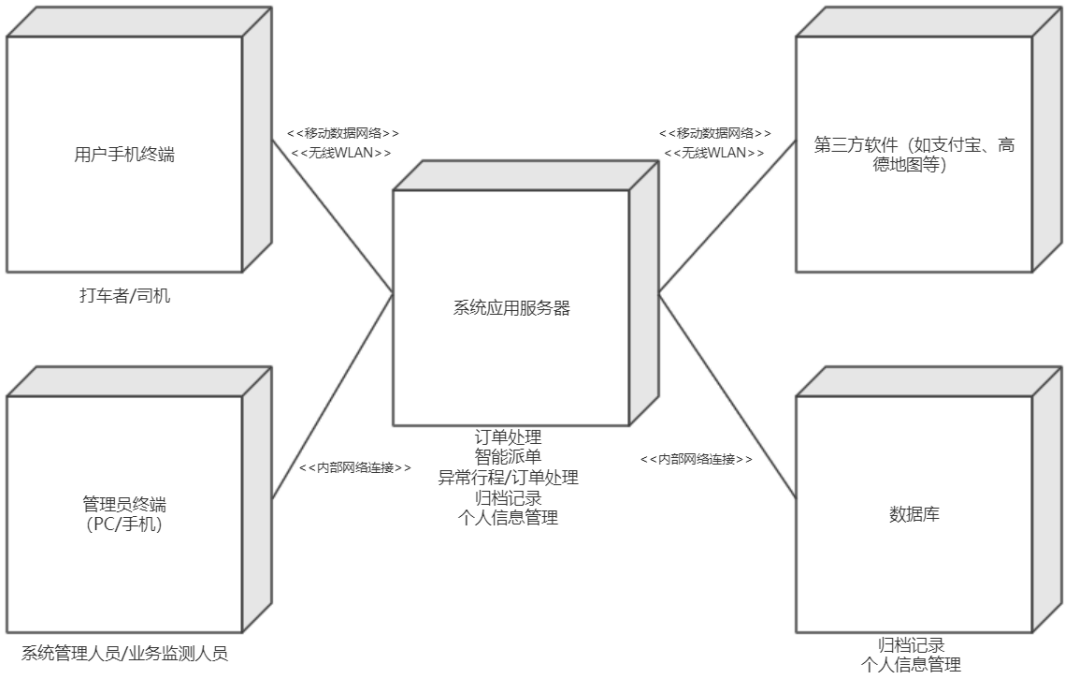

图表 19 过程模型到设计模型的依赖图

## 6.4 进程实现




图表 20 进程实现图

# 7. 部署视图




图表 21 系统部署视图

## 7.1 用户手机终端

用户（打车者/司机）通过自己的手机下载《轩轩打车》app，注册并登陆，随后通过移动数据网络或无限 WLAN 连接到系统应用服务器上，使用订单处理、智能派单、个人信息管理等功能，通过显示屏查看导航、订单详情等信息。

## 7.2 管理员终端

管理员（系统管理人员/业务监测人员）使用其终端（可以为 PC 电脑或者手机）登录进入《轩轩打车》App 的管理员端，随后通过系统内部网络连接到系统应用服务器上，处理异常行程、处理异常订单、归档记录、管理个人信息等，即管理员也有通过系统应用服务器对数据库进行操作的权限。并且，能通过显示屏查看操作结果。

## 7.3 系统应用服务器

是主要的系统服务器，打车者、司机、业务监测人员、系统管理人员都能连接到此服务器，对业务进行操作，并连接数据库进行必要操作，同时也能依赖第三方软件完成导航、缴费等工作。

系统为各个用户角色提供的功能主要都部署在服务器上，等待用户角色使用。根据功能不同，直接在服务器进行工作或连接第三方软件/数据库进行操作。

## 7.4 数据库

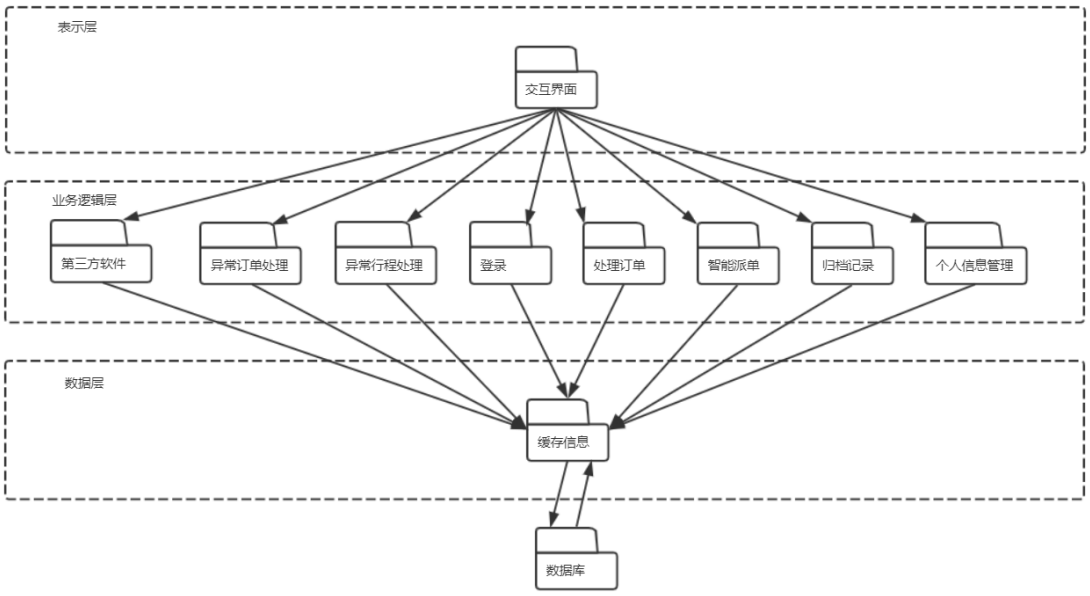
用于存储完整的订单、各类用户的个人信息，本系统选用 MySql 数据库。对于本系统，在：修改、查看个人信息，完整订单形成并保存，异常订单管理时，才会对数据库中一些持久性记录进行操作。

## 7.5 第三方软件

打车软件系统需要付费及地图导航功能，为此可以考虑依赖外部第三方软件，在打车者与司机对接产生订单及行程信息后，服务器依赖导航软件（如高德导航）为二者提供地图及导航功能；在行程完成后，打车者与司机需要完成交易，此时

服务器依赖交易软件（如支付宝）完成交易。诸如此类，也提高了系统某些接口的可重用性。

# 8. 实现视图




图表 22 实现视图

本部分给出 4+1 视图中的实现视图。

# 9. 规模与性能

本软件面向用户范围广、潜在用户数量多，且存在高峰时段，致使本打车软件系统需要较高的性能来在不同情况下完成其应当完成的功能。

由于信息大多存储在远程服务器与数据库中，故用户手机终端上的软件整体大小应小于 300MB，方便用户下载使用。

## 9.1 时间性能要求

1. 登录速度

在 5000 名用户同时尝试登录的情况下，响应时间不得超过 1s。

在 50000 名用户同时尝试登陆的情况下，响应时间不得超过 3s。

2. 个人信息管理速度

在 5000 名用户同时尝试查看/修改个人信息的情况下，响应时间不得超过

1s。

在 50000 名用户同时尝试查看/修改个人信息的情况下，响应时间不得超过 3s。

3. 处理订单速度

在 5000 名打车者同时尝试处理订单的情况下：

发出订单的响应时间不得超过 2s。

匹配司机成功后，刷新显示信息的响应时间不得超过 1s。

连接第三方软件获取导航/支付信息的响应时间不得超过 0.5s。

给出评价的响应时间不得超过 0.5s。

在 50000 名打车者同时尝试处理订单的情况下：

发出订单的响应时间不得超过 4s。

匹配司机成功后，刷新显示信息的响应时间不得超过 2s。

连接第三方软件获取导航/支付信息的响应时间不得超过 1s。

给出评价的响应时间不得超过 1s。

4. 智能派单速度

在 5000 名司机同时尝试接受智能派单的情况下：

匹配订单的响应时间不得超过 2s。

匹配成功后，刷新显示信息的响应时间不得超过 1s。

连接第三方软件获取导航/支付信息的响应时间不得超过 0.5s。

查看评价的响应时间不得超过 0.5s。

在 50000 名司机同时尝试接受智能派单的情况下：

匹配订单的响应时间不得超过 4s。

匹配成功后，刷新显示信息的响应时间不得超过 2s。

连接第三方软件获取导航/支付信息的响应时间不得超过 1s。

查看评价的响应时间不得超过 1s。

5. 异常行程处理速度

业务监测人员希望查看某一行程时，响应时间不应超过 1s。

6. 异常订单处理速度

系统管理人员查看异常订单时，响应时间不应超过 1s。

系统管理人员修改并归档记录异常订单时，响应时间不应超过 1.5s。

7. 归档记录速度

单次连接数据库进行单表操作的响应时间及处理时间不得超过 1s。

## 9.2 空间性能要求

1. 打车者/司机使用手机终端使用客户端，客户端大小应不大于 300MB。
2. 软件使用时，实时导航信息及支付信息与第三方软件交互得来，其使用期间占用内存不应超过 100MB。
3. 软件使用时会产生订单、行程缓存，占用内存不应超过 50MB。
4. 软件操作数据库查看信息时，会得到信息缓存，占用内存不应超过 50MB。

## 9.3 规模要求

1. 系统应至少同时支持 50000 名打车者同时使用。
2. 系统应至少同时支持 50000 名司机同时使用。
3. 因此，系统服务器应至少满足 100000 名用户同时连接。

## 9.4 满足要求的方法

由于本系统使用三层 C/S 架构进行部署，则大部分功能于服务器中调用。用户使用客户端时其响应速度很快，具体信息显示速度由网络连接情况决定。也由于大部分功能位于服务器中调用，故网络传输数据量较小，网络连接情况较好。

同时，由于服务器只提供功能接口，具体用户逻辑由客户端完成，所以服务器压力较小，可满足多名用户同时连接。

由于大部分信息存在数据库中，使用时查询并得到缓存即可，故在用户终端上占用的内存为一些缓存信息，不会很大。

# 10. 质量分析与评价

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。本部分首先对非功能性需求给出评价标准，随后给出场景分析来尝试评价。

## 10.1 非功能需求 —— 质量需求

### 10.1.1 易用性

1. 考虑到面向人群多样化，要求软件界面应简洁大方、有人性化，符合 GUI 相关设计标准，方便老人使用；同时针对特殊人群，应加入语音提示功能。

### 10.1.2 密安性

1. 该系统各用户有各自的权限（表现为可使用的功能不同），使用功能需要先登录通过认证
2. 该系统应保证自身完整性，避免非法手段操作系统关键信息。
3. 该系统应具备良好保密性，避免用户信息泄露。

### 10.1.3 可维护性

1. 该系统应容易诊断缺陷和失效原因及识别出待修改部分。
2. 系统应容易修改，方便开发人员后续修改和扩展。
3. 系统应有良好稳定性，防止由于软件修改造成意外错误的能力强。
4. 系统应容易测试，即在修改后应容易被测试和确认正常。

### 10.1.4 可移植性

1. 易安装性：该系统在多平台应容易安装，如安卓和 IOS。
2. 共存性：系统应当能够和其他软件共存于一个平台上。
3. 易替换性：系统容易卸载，也容易被高版本替换。

### 10.1.5 可扩展性

1. 该系统应具有良好扩展性，经过修改可以扩展到相关领域如共享单车等。
2. 一些接口如数据库操作接口、网络数据传输接口、登录接口、个人信息管理接口等可以复用，便于扩展。

## 10.2 非功能需求 —— 工程需求

### 10.2.1 逻辑数据库需求

该系统使用到了数据库，故本部分给出逻辑数据库设计，先进行分析，随后以 E-R 图

（实体-联系图）来展示逻辑数据库设计。

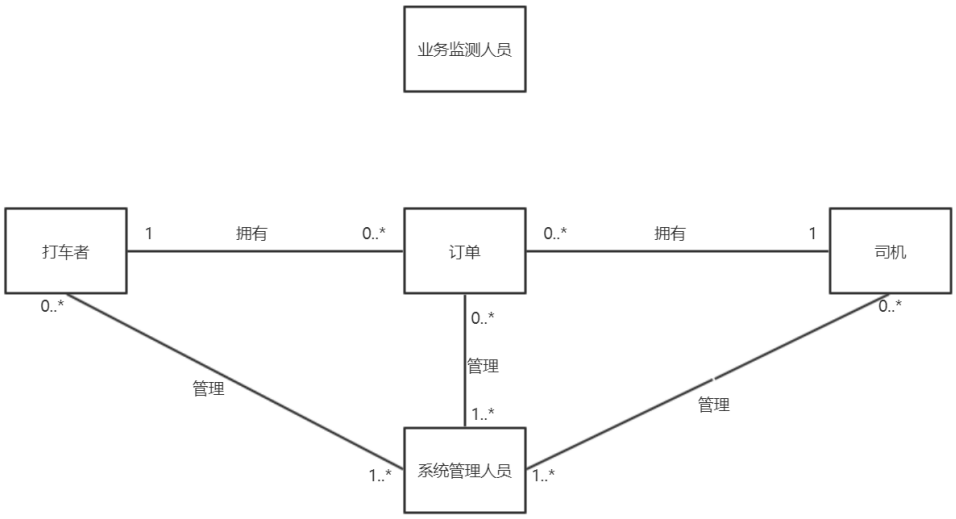
考虑该系统的四类用户：打车者、司机、业务监测人员、系统管理人员，需要在系统中分别用表进行存储（各自有 id，不放在一个表里）。

根据完整的打车流程，可以看出最后记录在数据库中的信息只有“完成的完整订单”，而行程信息在结束后直接消除即可，无需存储备案。

故确定了五个实体：四类用户与订单。

再来考虑实体间的关系：业务监测人员负责异常行程的管理，而行程信息不存储在数据库中，故业务监测人员与其他实体并无联系；系统管理人员负责管理人员信息与管理异常订单，故与打车者、司机与订单都有联系；打车者与订单、司机与订单皆有联系。

因此可以得出如下 E-R 图：




**图表 23 打车软件数据库 E-R 图**

设计思路：

1. 打车者/司机与订单是一对多关系。一个订单只有一个打车者/司机，而一个打车者/司机可以有多个订单。
2. 系统管理人员与除业务监测人员外的所有实体都是多对多关系。系统管理人员可以管理多个打车者/订单/司机，而一个打车者/订单/司机也可以被多个系统管理人员管理。

根据 E-R 图可以建立数据库的逻辑结构如下（下划线代表代表主键，斜体代表外键）：

打车者（*id*，姓名，性别，手机号，密码）

司机（*id*，姓名，性别，手机号，密码，车型，评级）

业务监测人员（id，账号，密码）

系统管理人员（*id*，账号，密码）

订单（id，打车者 id，司机 id，出发地，目的地，订单日期，计费金额，支付状态，评价）

## 10.3 场景分析

### 10.3.1 用例场景

用例场景是从使用者角度出发，描述用户所期望的与整个运行系统的交互。

| 场景编号 | 场景描述 |
| --- | --- |
| 场景 - 1 | • 打车者期望可以成功发出订单打车，在行程中能够查看导航信息，行程结束后能够正常缴费、发出评价。能够查看历史订单。<br>• 司机期望可以成功接收智能派单，在行程中能够查看导航信息，行程结束后能够正常收费、查看评价。能够查看历史订单。<br>• 业务监测人员期望能够正常查看想要查看的行程，并在发现异常时能够联系订单双方。<br>• 系统管理人员期望能够得到异常订单通知，正常查看并修改异常订单，归档记录存入数据库。<br>• 该场景代表了用户期望系统易于使用，即易用性。 |
| 场景 – 2 | • 用户进行各种操作时，系统响应成功情况应超过 90%，且响应时间应基本小于 5s 内。<br>• 对第三方软件的连接与信息获取时间应小于 5s，并能够在 1s 内刷新至用户终端显示器上。<br>• 该场景代表了系统的性能要求。 |
| 场景 – 3 | • 发生数据异常时，系统要通知系统管理人员，在其终端显示器上用红色字体显示出来。<br>• 本软件系统在发生停机故障之后，应该保证五分钟之内可以修复并且正常运行。<br>• 系统具有一定的容错能力和故障恢复能力，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，但不会导致系统崩溃无法使用。<br>• 该场景代表了用户期望的可靠性。 |
| 场景 - 4 | • 用户期望在登录后才能获取相应权限进行操作，同时数据不被泄露。<br>• 该场景代表了用户期望的密安性。 |
| 场景 - 5 | • 用户期望能够及时获得第三方软件的信息，显示器刷新延时小于 1s。<br>• 该场景代表了用户期望的系统性能 |

表格 9 用例场景

### 10.3.2 增长性场景

增长性场景是指预期未来系统修改时可能发生的场景。每个场景可能是相关的质量属性的衍生物。

| 场景编号 | 场景描述 |
| --- | --- |
| 场景 - 1 | • 可以增加新的数据服务器，将远程用户的访问时间进一步降低，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 | • 通过对数据库表结构的调整，把单表操作的检索时间再次降低至 0.5s 以内，更优更快。 |
| 场景 – 3 | • 可以增加新的第三方软件支持，将导航与支付的可选择性进一步提高，能够提高系统的可靠性。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 - 4 | • 可以增加新的业务如共享单车，藉由软件的可扩展性将软件进一步扩展，增加更多的业务，复用如数据库操作、网络传输数据、个人信息管理等接口。 |

**表格** 10 **增长性场景**

### 10.3.3 探索性场景

探索性场景是推动系统封装和降低工作压力的场景。场景的目标是揭示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。系统设计者不可能预见到未来所有的修改，但是，在某种程度上，可以为未来系统的修改给出更实际的需求。

| 场景编号 | 场景描述 |
| --- | --- |
| 场景 - 1 | • 正常情况下，当一半服务器宕机时，不影响整个系统的可使用性。 |
| 场景 – 2 | • 改进系统的可使用性，使其从 98%提升到 99.999% |
| 场景 – 3 | • 系统能够从 IOS 平台更换到 Android 平台。 |
| 场景 – 4 | • 当公司业务扩展时（如共享单车）可以增加如共享单车位置接口等，不超过 5 个人月的工作量。 |

**表格** 11 **探索性场景**

# A. 附录

## A.1 附录 1



## A.2 附录 2