**智慧交通系统设计**

**概要设计**

1. **数据结构**

数据结构设计主要针对数据库。本系统有三部分数据需要在数据库中存储，分别是车辆数据、地图数据和日志数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表名 | Vehicle | | | | |
| 功能描述 | 存储车辆属性和状态 | | | | |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 说明 |
| ID | serial | N |  | PK,ID,UQ | 车辆编号 |
| License | varchar | N |  | UQ | 车牌号 |
| isEmpty | boolean | N | TRUE |  | 车辆是否空闲 |
| Location | point | N |  |  | 车辆当前位置 |
| Destination | point | Y |  |  | 车辆目标位置 |
| Path | path | Y |  |  | 车辆短时行驶轨迹 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表名 | Vehicle | | | | |
| 功能描述 | 存储车辆属性和状态 | | | | |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 说明 |
| ID | serial | N |  | PK,ID,UQ | 车辆编号 |
| License | varchar | N |  | UQ | 车牌号 |
| isEmpty | boolean | N | TRUE |  | 车辆是否空闲 |
| Location | point | N |  |  | 车辆当前位置 |
| Destination | point | Y |  |  | 车辆目标位置 |
| Path | path | Y |  |  | 车辆短时行驶轨迹 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表名 | Map\_Loc | | | | |
| 功能描述 | 存储地图上的地点 | | | | |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 说明 |
| LID | serial | N |  | PK,ID,UQ | 地点编号 |
| Location | point | N |  | UQ | 地点坐标 |
| State | int | N |  |  | 地点状态 |
|  |  |  |  |  |  |
| 表名 | Map\_Road | | | | |
| 功能描述 | 存储地图上的道路 | | | | |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 说明 |
| RID | serial | N |  | PK,ID,UQ | 道路编号 |
| Begin | int | N |  |  | 道路起始地点编号 |
| End |  | N |  |  | 道路终止地点编号 |
| Length | int | N |  |  | 道路长度 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表名 | Log | | | | |
| 功能描述 | 日志，记录车辆历史行为 | | | | |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 说明 |
| LogID | serial | N |  | PK,ID,UQ | 日志记录编号 |
| VehID | int | N |  |  | 记录对应车辆编号 |
| StartTime | timestamp | N |  |  | 发车时间 |
| StopTime | timestamp | N |  |  | 停车时间 |
| StartLoc | point | N |  |  | 发车地点 |
| StopLoc | point | N |  |  | 停车地点 |
| Mileage | int | Y |  |  | 行驶里程 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表名 | User | | | | |
| 功能描述 | 记录用户（乘客）数据 | | | | |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 说明 |
| UserID | serial | N |  | PK,ID,UQ | 用户编号 |
| UserName | varchar | N |  | UQ | 用户名 |
| Password | varchar | N |  |  | 登录密码 |

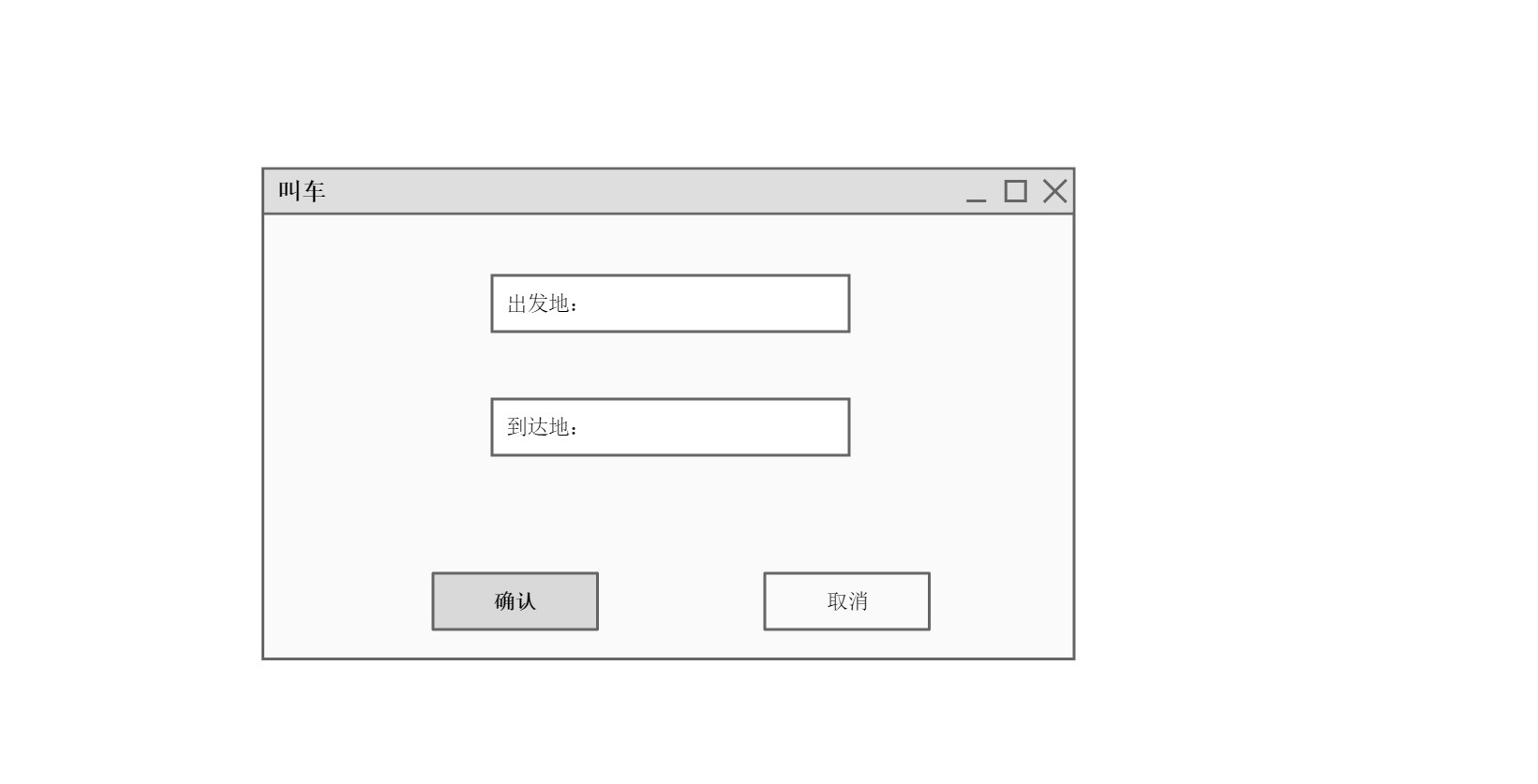
1. **UI设计**

本系统有两部分人机交互界面，一部分提供给乘客用于叫车，一部分提供给开发人员用于维护数据库

这是乘客界面主页，地图与车辆数据由系统给出



当乘客希望用车时，点击“叫车按钮”，弹出输入对话框。乘客可以输入出发地和目的地，可以点击“取消”按钮退回主页



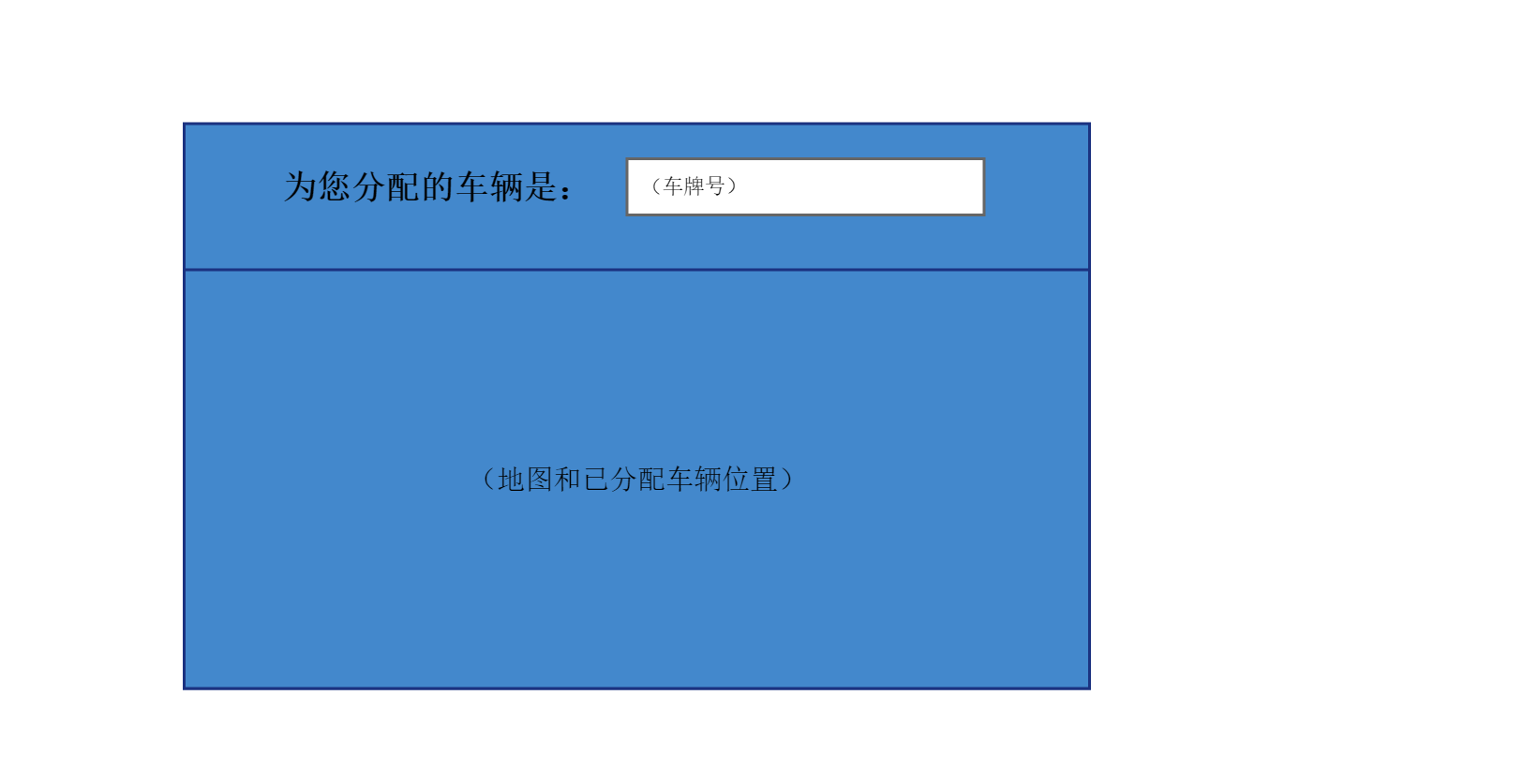
乘客输入完毕，点击“确认”按钮，决策引擎开始工作，弹出等待处理对话框



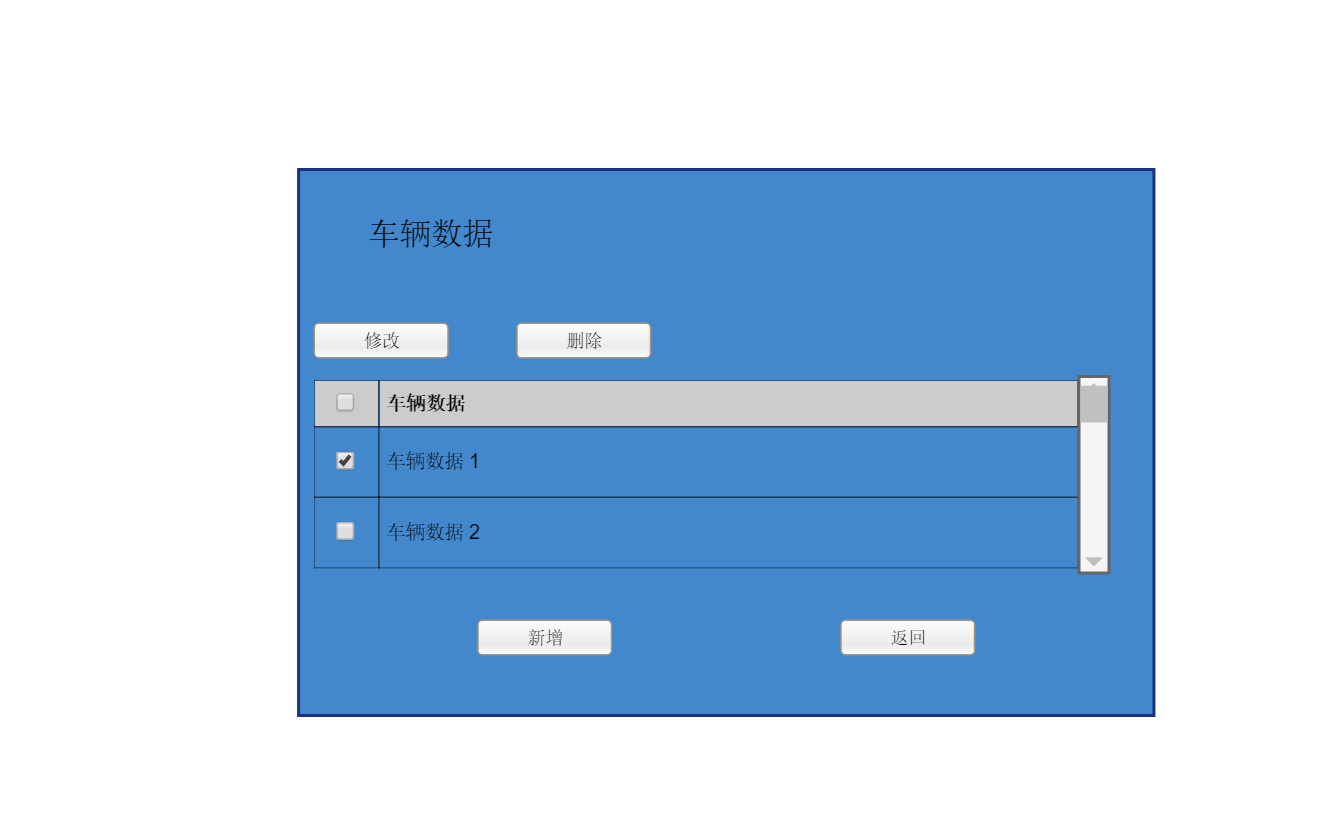
车辆调度成功，弹出成功对话框



乘客点击“OK”按钮或成功对话框显示一秒后，自动跳转至等车页，车牌号等数据由系统给出



以下是开发人员界面





开发人员可以审阅地图和车辆数据；用“新增”按钮新增数据；可以选中一条或多条数据后，用“修改”和“删除”按钮修改和删除数据

1. **模块分解及其接口**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BasicData | 基础数据管理，数据库实现 | mapPointAdd() | 地图上地点的新增 | 这里的接口是按着数据库写的，网络资料显示JAVA服务端与数据库通信用JDBC，还有一套接口，不知道该怎么写到设计里 |
| mapPointDelete() | 删除 |
| mapPointSet() | 修改 |
| mapPointGet() | 查询（返回单个数据） |
| mapPointFind() | 查询（返回符合条件的一组数据 |
| mapRoadAdd() | 地图上路径的新增 |
| mapRoadDelete() | |
| mapRoadSet() |  |
| mapRoadGet() |  |
| mapRoadFind() |  |
| vehicleAdd() | 车辆数据新增 |
| vehicleDelete() |  |
| vehicleSet() |  |
| vehicleGet() |  |
| vehicleFind() |  |
| userAdd() | 用户数据新增 |
| userDelete() |  |
| userSet() |  |
| userGet() |  |
| userFind() |  |
| Log | 日志，数据库实现 | logAdd() | 日志记录新增 |  |
| logDelete() |  |  |
| logSet() |  |  |
| logGet() |  |  |
| logFind |  |  |
| DataCache | 数据缓存模块，内存中实现 | requestAdd() | 新增一条乘客请求 | 利用JAVA的Queue接口相关方法 |
| requestPeek() | 读取当前待处理乘客请求 |
| requestRemove() | 移除已处理的请求 |
| requestReturn() | 返回请求处理状态 |
| logInCheck() | 登录请求验证 | 利用Map接口 |
| vehicleSet() | 更新某一车辆数据 | 利用List接口 |
| vehicleAdd() | 新增一条车辆数据 |
| vehicleGet() | 获取某一车辆数据 |
| vehicleRemove() | 删除某一车辆数据 |
| mapGet() | 获取地图数据 | graph类型要自己写 |
| stateGet() | 获取当前路网状态（地图和车） |  |
| Decision | 智慧决策模块 | resultSingle() | 针对乘客单次请求，给出一辆车的调度结果 |  |
| resultGlobal() | 给出所有车辆的全局调度结果 | 考虑堵车等情况 |