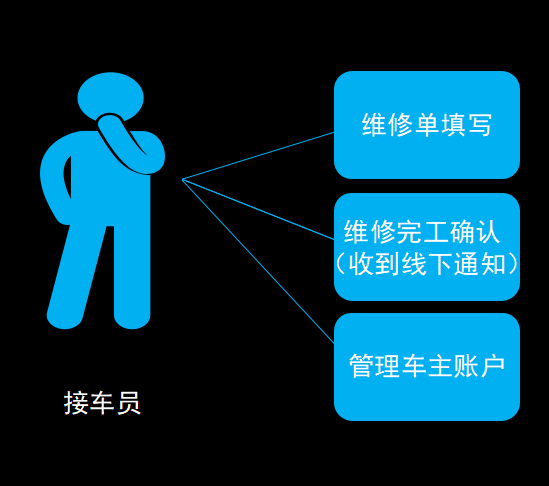
# 汽车维修系统

## （CarRepairSystem）

1. 系统分析
2. 用例



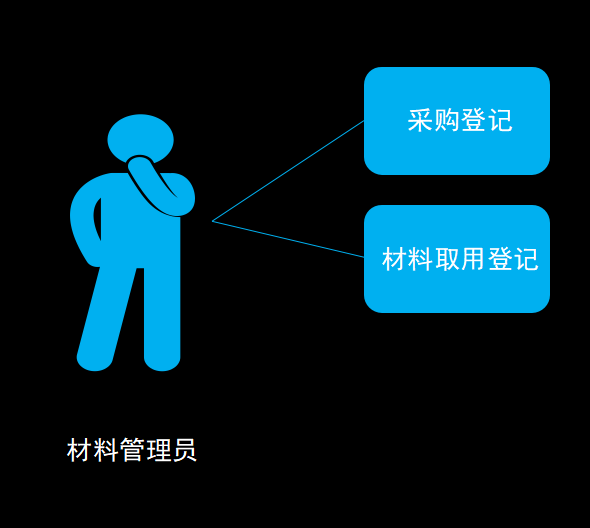
用户：接车员

用例说明：

接车员接待前来维修的车主，车主未注册需要取得车主的联系方式和姓名，车牌号，车牌进行注册，若车主已注册车辆未注册，则需要取得车主的联系方式，车牌号和车牌。

若都已经注册直接利用车牌号进行维修单的填写，填写信息接车员姓名，调度员姓名，还有维修金额，维修项目以及所需材料。

收到线下完工通知进行确认操作。

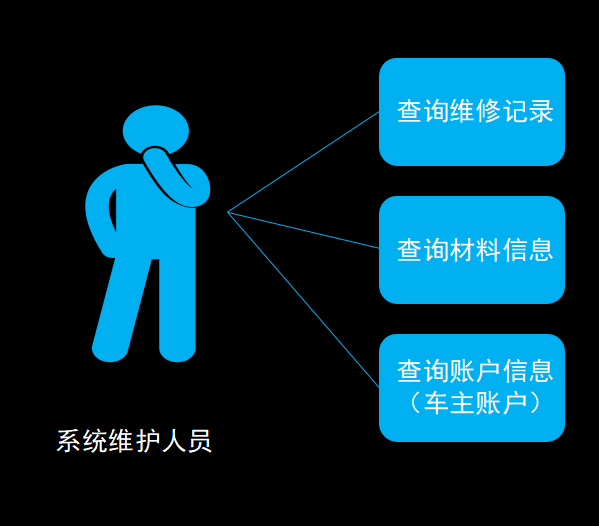


用户：材料管理员

用例说明：

采购人员携带材料和发票进行采购登记并将材料入库。

维修人员到此处利用维修单id获取材料，并将维修项目材料取用标识设置为完成。



用户：系统维护人员

用例说明：

对维修记录进行查询。

对材料信息进行查询。

对车主账户信息进行查询。

总结：主要是对与维修记录（维修单）、材料、车主账户信息的管理。

1. 系统设计
2. 总体设计

轻量级web框架Flask，轻量级数据库sqlite3，类模型利用flask-sqlalchemy，后台语言python（易于部署）前端页面html。

DAO层——》service层——》controller层（借鉴java spring框架，微服务框架，并不是一个用例一个controller）

数据管理——》业务流程，数据一致性——》控制转发

1. 数据库设计

car - car owner 多对一

car - repair order 一对多

repair order - repair material 多对多

*# RepairOrder class***class** RepairOrderDao(db.Model):  
  
 \_\_tablename\_\_ = **'repair\_order'** id = db.Column(db.Integer, primary\_key=True)  
 car\_collector\_name = db.Column(db.String(80))  
 dispatcher\_name = db.Column(db.String(80))  
 repairman\_name = db.Column(db.String(80))  
 inspector\_name = db.Column(db.String(80))  
 car\_id = db.Column(db.Integer)  
 repair\_money\_total = db.Column(db.Float)  
 repair\_order\_status = db.Column(db.Boolean)  
 create\_time = db.Column(db.DateTime)  
 update\_time = db.Column(db.DateTime)  
 is\_delete = db.Column(db.Boolean)  
  
*# CarOwner class***class** CarOwnerDao(db.Model):  
  
 \_\_tablename\_\_ = **'car\_owner'** id = db.Column(db.Integer, primary\_key=True)  
 car\_owner\_name = db.Column(db.String(80))  
 car\_owner\_number = db.Column(db.String(80), unique=True)  
 create\_time = db.Column(db.DateTime)  
 update\_time = db.Column(db.DateTime)  
 is\_delete = db.Column(db.Boolean)  
  
 **def** \_\_init\_\_(self, car\_owner\_name, car\_owner\_number, create\_time, update\_time):  
 self.car\_owner\_name = car\_owner\_name  
 self.car\_owner\_number = car\_owner\_number  
 self.create\_time = create\_time  
 self.update\_time = update\_time  
 self.is\_delete = False

*# Car class***class** CarDao(db.Model):  
  
 \_\_tablename\_\_ = **'car'** id = db.Column(db.Integer, primary\_key=True)  
 car\_owner\_id = db.Column(db.Integer)  
 car\_brand = db.Column(db.String(80))  
 plate\_number = db.Column(db.String(80), unique=True)  
 create\_time = db.Column(db.DateTime)  
 update\_time = db.Column(db.DateTime)  
 is\_delete = db.Column(db.Boolean)  
  
*# RepairProject class***class** RepairProjectDao(db.Model):  
  
 \_\_tablename\_\_ = **'repair\_project'** id = db.Column(db.Integer, primary\_key=True)  
 repair\_project\_name = db.Column(db.String(80))  
 repair\_material\_id = db.Column(db.Integer)  
 repair\_order\_id = db.Column(db.Integer)  
 repair\_material\_cost\_amount = db.Column(db.Integer)  
 repair\_material\_status = db.Column(db.Boolean)  
 create\_time = db.Column(db.DateTime)  
 update\_time = db.Column(db.DateTime)  
 is\_delete = db.Column(db.Boolean)  
  
*# RepairMaterial class***class** RepairMaterialDao(db.Model):  
  
 \_\_tablename\_\_ = **'repair\_material'** id = db.Column(db.Integer, primary\_key=True)  
 repair\_material\_name = db.Column(db.String(80), unique=True)  
 repair\_material\_has\_amount = db.Column(db.Integer, CheckConstraint(**'repair\_material\_has\_amount>0'**))  
 create\_time = db.Column(db.DateTime)  
 update\_time = db.Column(db.DateTime)  
 is\_delete = db.Column(db.Boolean)

1. 控制类设计

对于三类管理的数据，维修数据，车主数据，材料数据的增删查改和通过主键查询。

特殊车主通过手机号查询，车辆通过车牌号查询。

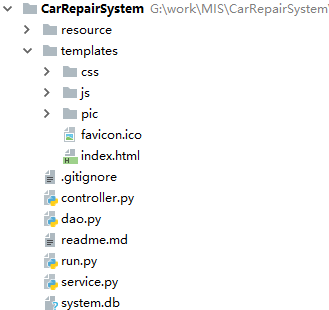
插入和更新的返回值规定为插入和更新后的对象。

**class** RepairOrderService:  
  
 **def** \_\_init\_\_(self):  
 **pass** *# 输入：订单信息 Dict  
 # 输出：订单对象 RepairOrderDao  
 # 功能：订单插入数据库，包括repair\_order表和repair\_project表* @staticmethod  
 **def** insert(insert\_dict):  
   
 *# 输入：维修项目信息 Dict  
 # 输出：维修项目对象 RepairProjectDao  
 # 功能：维修项目信息新增导入数据库 包括repair\_project表* @staticmethod  
 **def** insert\_repair\_project(insert\_dict):  
  
 *# 输入：订单信息 Dict  
 # 输出：订单对象 RepairOrderDao  
 # 功能：订单更新数据库，包括repair\_order表和repair\_project表* @staticmethod  
 **def** update(update\_dict):  
   
 *# 输入：start and page\_size  
 # 输出：订单对象列表 List<RepairOrderDao>  
 # 功能：查询订单信息* @staticmethod  
 **def** select(start, page\_size):  
   
 *# 输入：repair\_order\_id and start and page\_size  
 # 输出：订单对象列表 List<RepairProjectDao>  
 # 功能：查询材料项目信息* @staticmethod  
 **def** select\_repair\_project(repair\_order\_id, start, page\_size):  
   
 *# 输入：订单id Int  
 # 输出：  
 # 功能：删除订单 包括repair\_order表和repair\_project表* @staticmethod  
 **def** delete(repair\_order\_id):  
   
 *# 输入：维修项目id Int  
 # 输出：  
 # 功能：删除维修项目 包括repair\_project表* @staticmethod  
 **def** delete\_repair\_project(repair\_project\_id):  
   
 *# 输入：订单id  
 # 输出：订单对象 RepairOrderDao  
 # 功能：通过id查询订单* @staticmethod  
 **def** select\_by\_id(repair\_order\_id):  
   
  
**class** MaterialService:  
  
 **def** \_\_init\_\_(self):  
 **pass** *# 输入：材料信息 Dict  
 # 输出：材料对象 RepairMaterialDao  
 # 功能：材料插入数据库 包括repair\_material表* @staticmethod  
 **def** insert(insert\_dict):  
   
 *# 输入：材料信息 Dict  
 # 输出：材料对象 RepairMaterialDao  
 # 功能：材料信息更新数据库 包括repair\_material表* @staticmethod  
 **def** update(update\_dict):  
   
 *# 输入：订单id Int  
 # 输出：订单对象 RepairOrderDao  
 # 功能：订单材料出库更新数据库 包括repair\_project表和repair\_material表* @staticmethod  
 **def** confirm(repair\_order\_id):  
   
 *# 输入：start and page\_size  
 # 输出：材料对象列表 List<RepairMaterialDao>  
 # 功能：查询材料信息* @staticmethod  
 **def** select(start, page\_size):  
   
 *# 输入：材料id  
 # 输出：  
 # 功能：删除材料信息 包括repair\_material表* @staticmethod  
 **def** delete(repair\_material\_id):  
   
 *# 输入：材料id  
 # 输出：材料对象 RepairMaterialDao  
 # 功能：根据id查询材料* @staticmethod  
 **def** select\_by\_id(repair\_material\_id):  
   
  
**class** CarOwnerService:  
  
 **def** \_\_init\_\_(self):  
 **pass** *# 输入：车主信息 Dict  
 # 输出：车主对象 CarOwnerDao  
 # 功能：车主信息插入数据库 包括car\_owner表和car表* @staticmethod  
 **def** insert(insert\_dict):  
   
 *# 输入：车辆信息 Dict  
 # 输出：车辆对象 CarDao  
 # 功能：车辆信息插入数据库 包括car表* @staticmethod  
 **def** insert\_car(insert\_dict):  
   
 *# 输入：车辆id Int  
 # 输出：  
 # 功能：删除车辆信息 包括car表* @staticmethod  
 **def** delete\_car(car\_id):  
   
 *# 输入：车主信息 Dict  
 # 输出：车主对象 CarOwnerDao  
 # 功能：车主信息更新数据库 包括car\_owner表* @staticmethod  
 **def** update(update\_dict):  
   
 *# 输入：start and page\_size  
 # 输出：车主信息列表 List<CarOwnerDao>  
 # 功能：查询车主信息* @staticmethod  
 **def** select(start, page\_size):  
   
 *# 输入：car\_owner\_id and start and page\_size  
 # 输出：车辆信息列表 List<CarDao>  
 # 功能：查询车辆信息* @staticmethod  
 **def** select\_car(car\_owner\_id, start, page\_size):  
   
 *# 输入：车主id Int  
 # 输出：  
 # 功能：删除车主信息* @staticmethod  
 **def** delete(car\_owner\_id):  
   
 *# 输入：车主id Int  
 # 输出：车主对象 CarOwnerDao  
 # 功能：通过id查询车主* @staticmethod  
 **def** select\_by\_id(car\_owner\_id):  
   
 *# 输入：车主car\_owner\_number String  
 # 输出：车主对象 CarOwnerDao  
 # 功能：通过car\_owner\_number查询车主* @staticmethod  
 **def** select\_by\_car\_owner\_number(car\_owner\_number):  
   
 *# 输入：车辆plate\_number String  
 # 输出：车辆对象 CarDao  
 # 功能：通过plate\_number查询车辆* @staticmethod  
 **def** select\_car\_by\_plate\_number(plate\_number):

1. 界面设计

增删查改式界面，较简单。不涉及登入登出，系统目的就是管理数据流动和历史记录，以后在此基础上可以发展统计或者数据挖掘的服务。

1. 系统实施
2. 工具：git（代码管理），pycharm（python代码编写），notepad++（html代码编写），postman（测试）
3. 代码框架



1. 效果见VIDEO。