系统设计：

系统框架：

轻量级web框架Flask，轻量级数据库sqlite3，类模型利用sqlalchemy，后台语言python（易于部署）

DAO层——》service层——》controller层（借鉴java spring框架）

具体设计：

car - car owner 多对一

car - repair order 一对多

Repair order - repair project 一对多

Repair project - repair material 多对一

Dao：

# encoding=utf8

# 接车员 car\_collector

# 调度员 dispatcher

# 维修员 repairman

# 质检员 inspector

# 材料管理员 material\_manager

# 系统维护人员 system\_maintenance\_personnel

# 车主 car\_owner

# 维修项目 repair\_project

# 维修材料 repair\_material

from flask import Flask

from flask\_sqlalchemy import SQLAlchemy

app = Flask(\_\_name\_\_, static\_url\_path='', static\_folder='')

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///system.db'

db = SQLAlchemy(app)

# RepairOrder class

class RepairOrderDao(db.Model):

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\_start\_time = db.Column(db.DateTime)

repair\_end\_time = db.Column(db.DateTime)

is\_delete = db.Column(db.Boolean)

def \_\_init\_\_(self):

pass

# 接车员接到车主时，未注册先注册，注册完进行初始化订单操作

# 涉及内容，接车员名字，调度员名字， 车辆id， 维修费用， 维修开始时间

def collector(self, car\_collector\_name, dispatcher\_name, car\_id, repair\_money\_total, repair\_start\_time):

self.car\_collector\_name = car\_collector\_name

self.dispatcher\_name = dispatcher\_name

self.car\_id = car\_id

self.repair\_money\_total = repair\_money\_total

self.repair\_start\_time = repair\_start\_time

self.is\_delete = False

# 调度员接到任务后进行维修派工和质检派工

# 涉及内容， 维修员名字， 质检员名字

def dispatcher(self, repairman\_name, inspector\_name):

self.repairman\_name = repairman\_name

self.inspector\_name = inspector\_name

# 质检完成后打上维修结束时间

# 涉及内容， 维修结束时间

def finish(self, repair\_end\_time):

self.repair\_end\_time = repair\_end\_time

db.session.add(self)

db.session.commit()

# 删除记录

def delete(self):

RepairOrderDao.query.filter\_by(id=self.id).update({'is\_delete': True})

db.session.commit()

def \_\_repr\_\_(self):

return '<RepairOrder %r>' % self.id

# CarOwner class

class CarOwnerDao(db.Model):

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)

is\_delete = db.Column(db.Boolean)

def \_\_init\_\_(self, car\_owner\_name, car\_owner\_number):

self.car\_owner\_name = car\_owner\_name

self.car\_owner\_number = car\_owner\_number

self.is\_delete = False

def \_\_repr\_\_(self):

return '<CarOwner %r>' % self.id

# Car class

class CarDao(db.Model):

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)

is\_delete = db.Column(db.Boolean)

def \_\_repr\_\_(self):

return '<Car %r>' % self.id

# RepairProject class

class RepairProjectDao(db.Model):

id = db.Column(db.Integer, primary\_key=True)

repair\_material\_id = db.Column(db.Integer)

repair\_order\_id = db.Column(db.Integer)

repair\_material\_cost\_amount = db.Column(db.Integer)

is\_delete = db.Column(db.Boolean)

def \_\_repr\_\_(self):

return '<RepairProject %r>' % self.id

# RepairMaterial class

class RepairMaterialDao(db.Model):

id = db.Column(db.Integer, primary\_key=True)

repair\_material\_name = db.Column(db.String(80))

repair\_material\_has\_amount = db.Column(db.Integer)

is\_delete = db.Column(db.Boolean)

def \_\_repr\_\_(self):

return '<RepairMaterial %r>' % self.id

db.create\_all()

Service：

Controller：

# encoding=utf8

from service import app

from flask import request

class CollectorController:

def \_\_init\_\_(self):

pass

# 传入：接车员姓名 string， 调度员姓名 string， 维修金额 float， 车辆牌照 string

# 输出：维修单 id

# insert

@staticmethod

@app.route('/collector/writeRepairOrder')

def write\_repair\_order():

print request.args

return ''

# 传入：维修单 id

# 输出：维修单 id

# update

@staticmethod

@app.route('/collector/confirmFinish')

def confirm\_finish():

print request.args

return ''

# 传入：车主手机号 string

# 输出：车主 id

# select

@staticmethod

@app.route('/collector/checkCarOwner')

def check\_car\_owner():

print request.args

return ''

# 传入：车牌号 string

# 输出：车 id

# select

@staticmethod

@app.route('/collector/checkCar')

def check\_car():

print request.args

return ''

# 传入：车主手机号 string， 车主姓名 string，

# 输出：车主 id

# insert

@staticmethod

@app.route('/collector/insertCarOwner')

def insert\_car\_owner():

print request.args

return ''

# 传入：车主 id， 车牌号 string， 车的牌子 string

# 输出：车 id

# insert

@staticmethod

@app.route('/collector/insertCar')

def insert\_car():

print request.args

return ''

class DispatcherController:

def \_\_init\_\_(self):

pass

# 传入：维修员姓名 string， 质检员姓名 string

# 输出：维修单 id

# update

@staticmethod

@app.route('/dispatcher/chooseMan')

def choose\_man():

print request.args

return ''

class MaterialManagerController:

def \_\_init\_\_(self):

pass

# 传入：材料 id list， 材料数量 int list

# 输出：status

# insert and update

@staticmethod

@app.route('/materialManager/manage')

def manage():

print request.args

return ''

# 传入：

# 输出：

@staticmethod

@app.route('/materialManager/select')

def select():

print request.args

return ''

class SystemMaintenancePersonnelController:

def \_\_init\_\_(self):

pass

# 传入：

# 输出：

# select

@staticmethod

@app.route('/systemMaintenancePersonnel/selectRepairRecord')

def select\_repair\_record():

print request.args

return ''

# 传入：

# 输出：

# select

@staticmethod

@app.route('/systemMaintenancePersonnel/selectMaterialRecord')

def select\_material\_record():

print request.args

return ''

# 传入：

# 输出：

# select

@staticmethod

@app.route('/systemMaintenancePersonnel/selectCarOwnerRecord')

def select\_car\_owner\_record():

print request.args

return ''