车辆安全开发文档

一、项目介绍

为了提高司机发生事故时能够及时有效的得到诊治，也为医院诊断提供帮助，特此开发TransportSecurity系统。

本系统的用户主要分为以下两类：

1. 医院管理人员：接收事故信息，处理和委派外派医护人员
2. 外派医护人员：查看系统，根据指示到达事故地点展开诊治

项目展示地址：http://106.52.60.176/ts/accident/warning

gitee地址：https://gitee.com/vvwhyyy/transportsecurity

github地址：https://github.com/weihongyu1/transportsecurity

二、环境搭建

运行环境

1. 操作系统：Centos 7+
2. 数据库：MySQL8+
3. SDK：JDK1.8
4. 数据库连接池：Druid1.2.4

开发环境

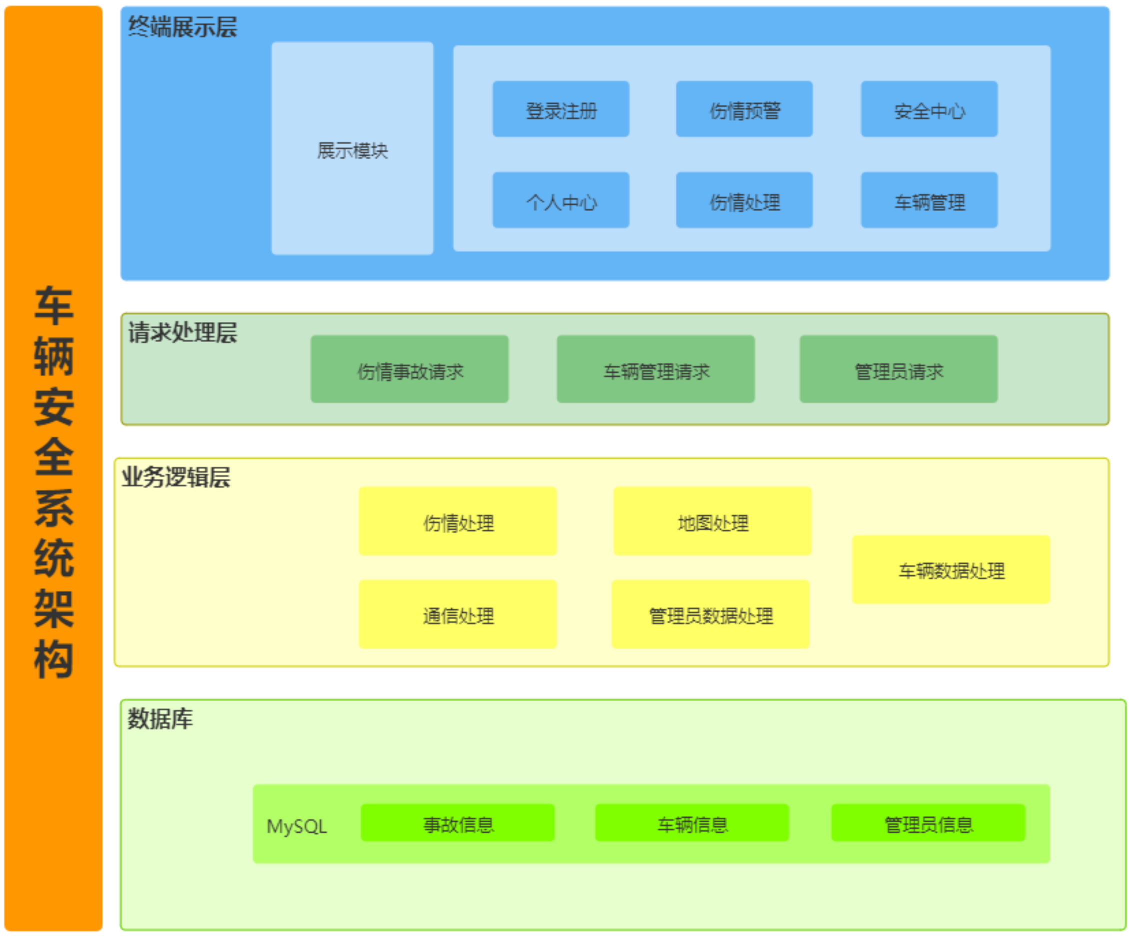
1. 操作系统：Windows10
2. 开发语言：前端。HTML，CSS，JavaScript，后台，Java
3. 开发环境：IntelliJ IDEA2020
4. 数据库：MySQl8.0+
5. SDK：maven3，JDK1.8

Maven依赖

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.2</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.why</groupId>
    <artifactId>transportsecurity_finally</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>transportsecurity_finally</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
            <version>2.2.0</version>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.33</version>
        </dependency>

        <!-- easyexcel文件依赖 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>easyexcel</artifactId>
            <version>2.2.10</version>
        </dependency>

        <!-- SM3,4加密算法 -->
        <dependency>
            <groupId>org.bouncycastle</groupId>
            <artifactId>bcprov-jdk15on</artifactId>
            <version>1.56</version>
        </dependency>

        <!-- 音频播放 -->
        <dependency>
            <groupId>javazoom</groupId>
            <artifactId>jlayer</artifactId>
            <version>1.0.1</version>
        </dependency>

        <!-- 邮件依赖 -->
        <dependency>
            <groupId>javax.mail</groupId>
            <artifactId>mail</artifactId>
            <version>1.4.7</version>
        </dependency>

        <!-- 短信依赖 -->
        <dependency>
            <groupId>com.github.qcloudsms</groupId>
            <artifactId>qcloudsms</artifactId>
            <version>1.0.6</version>
        </dependency>

        <!-- 分页插件 -->
        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper</artifactId>
            <version>5.2.0</version>
        </dependency>
        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper-spring-boot-autoconfigure</artifactId>
            <version>1.3.1</version>
        </dependency>
        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper-spring-boot-starter</artifactId>
            <version>1.3.1</version>
        </dependency>

        <!-- 整合swagger接口文档 -->
        <dependency>
            <groupId>com.spring4all</groupId>
            <artifactId>swagger-spring-boot-starter</artifactId>
            <version>1.9.1.RELEASE</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/com.alibaba/druid  Druid依赖-->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>1.2.4</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>

 <mainClass>com.why.transportsecurity_finally.TransportsecurityFinallyApplicatio
n</mainClass>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
                <executions>
                    <execution>
                        <goals>
                            <goal>repackage</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>

            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-resources-plugin</artifactId>
                <version>3.0.2</version>
            </plugin>

            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.22.2</version>
                <configuration>
                    <skipTests>true</skipTests>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

三、系统架构设计

项目借鉴传统的三层架构（即数据访问层、业务逻辑层和界面层）。



四、模块设计

4.1 管理员信息处理模块

4.1.1 登录、注册中的密码处理

在网站登陆时，密码一般采用加盐加密的形式来验证。其采用如下的一般过程：

注册阶段

1. 用户输入用户名和密码等相关信息，提交给服务端
2. 服务端产生一个随机字符串，称为盐值
3. 加盐：将盐值加入到用户密码末尾
4. 加密：使用对称密码算法，如SM4，AES，3DES等密码算法对其加密
5. 哈希：使用hash算法（SM3，MD5等）对加密后的字符串进行hash，产生新密码
6. 保存：将新密码和盐值保存到数据库中
7. 完成注册阶段

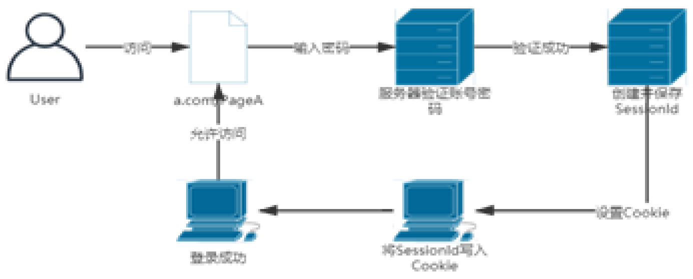
登录阶段

1. 用户输入用户名、密码提交给服务端
2. 服务端根据用户的用户名查询字段，查询不到返回
3. 将用户密码+查询到的盐值进行合并
4. 使用注册阶段对此字符串加盐加密
5. 登录时提交处理后的字符串和查询到的密码比较
6. 相等登录成功，否则返回

在此过程中使用对称加密保证了数据的机密性，哈希算法保证了数据的完整性，保证黑客即使拿到数据库中的数据也无法恢复出用户的密码信息。

4.1.2 Cookie + Session认证过程

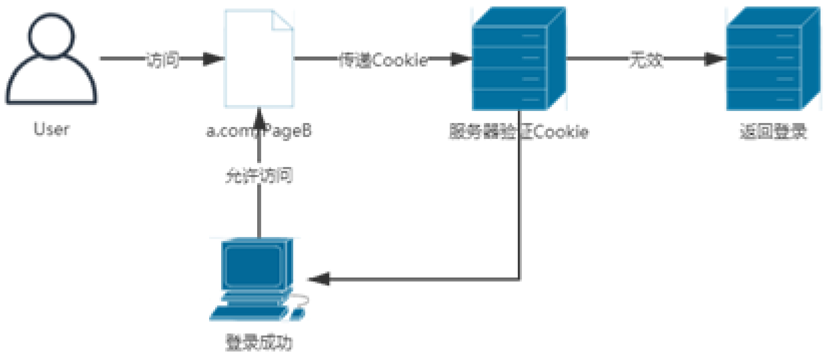
用户第一次登录



1. 用户访问a.com/PageA，并输入登录
2. 服务器使用上述密码验证成功后，会创建SessionId
3. 设置Cookie，将SessionId写入Cookie中

后续访问

第一次登录完成后，后续的访问就可以直接使用Cookie进行身份验证



1. 用户访问其它页面a.com/PageB，将设置好的Cookie传递个服务端
2. 服务端验证Cookie是否和Session中SessionId相同
3. 不相同，返回登录
4. 相同登录成功

Session和Cookie的方式解决了服务器判断请求是否来自同一个人的问题，也解决了判断用户的登录状态问题。

4.1.3 密码找回模块

管理员可根据绑定的邮箱或手机号码进行密码找回

1. 用户输入手机号/密码
2. 后台比对手机号和密码是否绑定
3. 若以绑定，发送验证码到手机或邮箱
4. 用户输入验证码登录
5. 进入系统后用户进行密码修改

4.2 事故处理模块

4.2.1 事故统计展示模块

1. 查询数据库中所有accident
2. 查询所有事故的车辆信息和加速度信息
3. 封装展示warning界面

4.2.2 碰撞数据处理

1. 接收前台传入的事故id参数
2. 查询事故信息、事故车辆信息、加速度信息
3. 根据经纬度进行逆地址解析
4. 封装碰撞数据返回到前台
5. view界面创建地图追踪，展示所有信息

4.3 车辆信息管理模块

4.3.1 所有车辆信息展示

查询所有车辆信息封装展示

4.3.2 车辆信息编辑模块

1. 管理员可对车辆信息进行编辑
2. 后台接受编辑后的数据进行保存
3. 管理员还可对不存在的车辆信息进行删除
4. 管理员可添加车辆信息

4.4 TCP通信模块

- 接收车载终端连接请求
- 创建服务端tcp通信线程
    - 创建线程
    - 处理数据
    - 添加至数据库
- 提交给线程池进行线程执行

五、数据库设计

5.1 管理员模块

5.1.1 管理员登录信息表

表名：tbl_admin

字段名：id（用户id，主键），u_name(用户名)，u_pwd(用户密码)

```sql
-- 管理员表设计
CREATE TABLE tbl_admin(
    id INT(10) PRIMARY KEY auto_increment,
    u_id VARCHAR(255) UNIQUE,
    u_name VARCHAR(255),
    u_pwd VARCHAR(255),
    u_salt VARCHAR(255)
);
```

5.1.2 管理员身份信息表

表名： tbl_admin_info

字段名： id（主键），u_phone，u_email，u_address，u_birth，u_date，u_id(外键，关联tbl_admin)

```
CREATE TABLE tbl_admin_info(
    id INT(10) PRIMARY KEY auto_increment,
    u_phone VARCHAR(50),
    u_email VARCHAR(255),
    u_address VARCHAR(255),
    u_birth VARCHAR(255),
    u_date VARCHAR(255),
    u_id INT(10),
    CONSTRAINT fk_ad_id FOREIGN KEY(u_id) REFERENCES tbl_admin(id)
);
```

5.2 事故信息模块

5.2.1 事故信息表

表名： tbl_accident_info

字段名： id（事故信息id，主键），a_date（事故日期，时间），lng（经度），lat（纬度），state（处理状态），v_id(车辆信息外键)

```
CREATE TABLE tbl_accident_info(
    id INT PRIMARY KEY auto_increment,
    a_date LONG NOT NULL,
    -- 经度
    lng DECIMAL(11,8) NOT NULL,
    lat DECIMAL(11,8) NOT NULL,
    -- 处理状态，0表示未处理，1表示已处理
    state INT DEFAULT(0) CHECK(state = 0 || state = 1),
    v_id INT,
    CONSTRAINT fk_vi_id FOREIGN KEY(v_id) REFERENCES tbl_vehicle_info(id)
);
```

5.2.2 ax加速度表

表名：tbl_ax

字段名：id（ax主键），acceleration_x（加速度ax），a_id(事故信息表id，外键)

```
-- ax加速度表
CREATE TABLE tbl_ax(
    id INT PRIMARY KEY auto_increment,
    acceleration_x DECIMAL(10,7),
    a_id INT,
    CONSTRAINT fk_ai_idx FOREIGN KEY(a_id) REFERENCES tbl_accident_info(id)
);
```

5.2.3 ay加速度表

表名：tbl_ay

字段名：id（主键），acceleration_y（加速度ay），a_id(事故信息表id，外键)

```sql
-- ay加速度表
CREATE TABLE tbl_ay(
    id INT PRIMARY KEY auto_increment,
    acceleration_y DECIMAL(10,7),
    a_id INT,
    CONSTRAINT fk_ai_idy FOREIGN KEY(a_id) REFERENCES tbl_accident_info(id)
);
```

5.3 车辆信息模块

5.3.1 车辆基本信息表

表名：tbl_vehicle_info

字段名：id(车辆id，主键)，v_name(车主姓名)，v_num(车牌号)，v_type（车辆类型）

```sql
-- 车辆基本信息表
CREATE TABLE tbl_vehicle_info(
    id INT PRIMARY KEY auto_increment,
    v_name VARCHAR(50),
    v_num VARCHAR(50),
    v_type VARCHAR(50)
);
```

六、实体类设计

6.1 管理员信息相关

6.1.1 Admin管理员登录注册信息

```java
package com.why.transportsecurity_finally.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * @Description TODO 管理员实体类
 * @Author why
 * @Date 2021/7/24 16:09
 * Version 1.0
 **/
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Admin {
    /**
     * id
     */
    private Integer id;
    /**
     * 用户id
     */
    private String uId;
    /**
     * 姓名
     */
    private String uName;
    /**
     * 密码
     */
    private String uPwd;
    /**
     * 盐值
     */
    private String uSalt;
}
```

6.1.2 管理员身份信息实体类

```java
package com.why.transportsecurity_finally.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.sql.Date;

/**
 * @Description TODO 管理员信息
 * @Author why
 * @Date 2021/7/27 10:42
 * Version 1.0
 **/
@Data
@NoArgsConstructor
@AllArgsConstructor
public class AdminInfo {
    /**
     * id
     */
    private Integer id;
    /**
     * 电话号
     */
    private String uPhone;
    /**
     * 邮箱号
     */
    private String uEmail;
    /**
     * 地址
     */
    private String uAddress;
    /**
     * 出生日期
     */
    private String uBirth;
    /**
     * 入职日期
     */
    private String uDate;
    /**
     * 外键uId
     */
    private Integer uId;
}
```

6.1.3 管理员信息封装整合实体类

```
package com.why.transportsecurity_finally.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.sql.Date;

/**
 * @Description TODO 封装个人中心信息
 * @Author why
 * @Date 2021/7/27 11:11
 * Version 1.0
 **/
@Data
@NoArgsConstructor
@AllArgsConstructor
public class AdminAll {
    /**
     * 用户id
     */
    private String uIds;
    /**
     * 姓名
     */
    private String uName;
    /**
     * id
     */
    private Integer id;
    /**
     * 电话号
     */
    private String uPhone;
    /**
     * 邮箱号
     */
    private String eMail;
    /**
     * 地址
     */
    private String uAddress;
    /**
     * 出生日期
     */
    private String uBirth;
    /**
     * 入职日期
     */
    private String uDate;
    /**
     * 外键uId
     */
    private Integer uId;
}
```

6.2 事故信息相关

6.2.1 事故信息

```
package com.why.transportsecurity_finally.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * @Description TODO 车辆事故信息实体类
 * @Author why
 * @Date 2021/7/21 10:27
 * Version 1.0
 **/
@Data
@AllArgsConstructor
@NoArgsConstructor
public class AccidentInfo {
    /**
     * id
     */
    private Integer id;
    /**
     * 时间
     */
    private long aDate;
    /**
     * 经度
     */
    private double lng;
    /**
     * 纬度
     */
    private double lat;
    /**
     * 处理状态，0表示未处理，1表示已处理
     */
    private int state;
    /**
     * 车辆基本信息id
     */
    private Integer vId;
}
```

6.2.2 加速度ax

```
package com.why.transportsecurity_finally.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * @Description TODO 横向加速度实体类
 * @Author why
 * @Date 2021/7/21 10:31
 * Version 1.0
 **/
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Ax {
    /**
     * id
     */
    private Integer id;
    /**
     * 横向加速度ax
     */
    private double accelerationX;
    /**
     * 车辆事故信息id
     */
    private Integer aId;
}
```

6.2.3 加速度ay

```
package com.why.transportsecurity_finally.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * @Description TODO 纵向加速的ay
 * @Author why
 * @Date 2021/7/21 10:33
 * Version 1.0
 **/
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Ay {
    /**
     * id
     */
    private Integer id;
    /**
     * ay
     */
    private double accelerationY;
    /**
     * 事故信息id
     */
    private Integer aId;
}
```

6.2.4 加速度封装

```
package com.why.transportsecurity_finally.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * @Description TODO 封装加速度信息
 * @Author why
 * @Date 2021/7/22 10:50
 * Version 1.0
 **/
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Acceleration {
    private double ax;
    private double ay;
}
```

6.2.5 封装统计展示信息

```
package com.why.transportsecurity_finally.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * @Description TODO 封装统计事故信息
 * @Author why
 * @Date 2021/7/21 11:43
 * Version 1.0
 **/
@Data
@AllArgsConstructor
@NoArgsConstructor
public class TotalInfo {
    /**
     * 车辆id
     */
    private Integer vId;
    /**
     * 事故id
     */
    private Integer aId;
    /**
     * 车牌号
     */
    private String vNum;
    /**
     * 车主姓名
     */
    private String vName;
    /**
     * 车辆类型
     */
    private String vType;
    /**
     * 日期
     */
    private String date;
    /**
     * 处理状态,true: 已处理, false: 未处理
     */
    private boolean status;
}
```

6.2.6 封装事故碰撞数据

```
package com.why.transportsecurity_finally.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * @Description TODO 封装展示细节信息
 * @Author why
 * @Date 2021/7/21 11:46
 * Version 1.0
 **/
@Data
@AllArgsConstructor
@NoArgsConstructor
public class DetailInfo {
    /**
     * 驾驶员损伤程度
     */
    private String driverDegreeOfDamage;
    /**
     * 后排乘客损伤程度
     */
    private String passengerDegreeOfDamage;
    /**
     * 日期
     */
    private String date;
    /**
     * 时间
     */
```

```java
    private String time;
    /**
     * 经度
     */
    private double lng;
    /**
     * 纬度
     */
    private double lat;
    /**
     * 地址
     */
    private String address;
    /**
     * 碰撞方向
     */
    private String direction;
    /**
     * 安全气囊是否弹开
     */
    private String isBounce;
    /**
     * 处理状态，true，已处理，false未处理
     */
    private boolean state;
}
```

6.2.7 封装excel数据下载model

```java
package com.why.transportsecurity_finally.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

/**
 * @Description TODO 数据下载实体类
 * @Author why
 * @Date 2021/7/22 11:07
 * Version 1.0
 **/
@Data
@NoArgsConstructor
@AllArgsConstructor
public class InfoExcelModel {
    private String vName;

    private String vNumber;

    private String vType;

    private String driverDegreeOfDamage;

    private String passengerDegreeOfDamage;

    private String date;

    private String time;

    private double lng;

    private double lat;

    private String address;

    private String collisionDirection;

    private String isBounce;

    private List<Double> adRAcceleration;

    private List<Double> adCAcceleration;
}
```

6.3 车辆信息相关

6.3.1 车辆信息

```
package com.why.transportsecurity_finally.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * @Description TODO 车辆基本信息实体类
 * @Author why
 * @Date 2021/7/21 10:23
 * Version 1.0
 **/
@Data
@AllArgsConstructor
@NoArgsConstructor
public class VehicleInfo {
    /**
     * id
     */
    private Integer id;
    /**
     * 车主姓名
     */
    private String vName;
    /**
     * 车牌号
     */
    private String vNum;
    /**
     * 车辆类型
     */
    private String vType;

}
```

七、mapper层接口设计

7.1 管理员相关

7.1.1 管理员AdminMapper接口

```java
package com.why.transportsecurity_finally.mapper;

import com.why.transportsecurity_finally.entity.Admin;
import com.why.transportsecurity_finally.entity.AdminInfo;
import org.apache.ibatis.annotations.Mapper;

/**
 * @Description TODO 管理员mapper
 * @Author why
 * @Date 2021/7/24 16:11
 * Version 1.0
 **/
@Mapper
public interface AdminMapper {

    /*****************************************登录注册
***********************************/


    /**
     * 根据uId查询管理员信息
     * @param uId
     * @return
     */
    public Admin getAdminById(String uId);

    /**
     * 根据id查询管理员
     * @param id
     * @return
     */
    public Admin getAdminByIdPre(Integer id);

    /**
     * 插入管理员信息
     * @param admin
     * @return
     */
    public void insertAdmin(Admin admin);

    /**
     * 更新数据
     * @param admin
     */
    public void updateAdmin(Admin admin);

    /**
     * 更新密码
     * @param admin
     */
    public void  updateAdminPwd(Admin admin);

    /*****************************************个人信息
***********************************/

    /**
     * 获取adminInfo
     * @param uId
     * @return
     */
    public AdminInfo getAdminInfoByUId(Integer uId);

    /**
     * 添加管理员信息
     * @param adminInfo
     * @return
     */
    public void insertAdminInfo(AdminInfo adminInfo);

    /**
     * 更新管理员数据
     * @param adminInfo
     */
    public void updateAdminInfo(AdminInfo adminInfo);
}
```

7.2 事故信息相关

7.2.1 事故信息mapper接口

```
package com.why.transportsecurity_finally.mapper;

import com.why.transportsecurity_finally.entity.AccidentInfo;
import com.why.transportsecurity_finally.entity.VehicleInfo;
import org.apache.ibatis.annotations.Mapper;

import java.util.List;

/**
 * @Description TODO 事故信息mapper
 * @Author why
 * @Date 2021/7/21 10:52
 * Version 1.0
 **/
@Mapper
public interface AccidentInfoMapper {
    /**
     * 根据车辆事故信息id查询事故信息
     * @param id 事故信息id
     * @return
     */
    public AccidentInfo getAccidentInfoById(Integer id);

    /**
     * 根据车辆基本信息id，查询车辆事故信息
     * @param vId 车辆基本信息id
     * @return
     */
    public List<AccidentInfo> getAccidentInfoByVId(Integer vId);

    /**
     * 获得所有已处理数据
     * @return
     */
    public List<AccidentInfo> getAllAccidentInfoSolve();

    /**
     * 获得所有未处理数据
     * @return
     */
    public List<AccidentInfo> getAllAccidentInfoUnSolve();

    /**
     * 更新处理状态
     * @param aID
     */
    public void updateState(Integer aID);

    public void deleteAccident(Integer vId);
}
```

7.2.2 加速度AxMapper接口

```
package com.why.transportsecurity_finally.mapper;


import com.why.transportsecurity_finally.entity.Ax;
import org.apache.ibatis.annotations.Mapper;

import java.util.List;

/**
 * @Description TODO ax加速度mapper
 * @Author why
 * @Date 2021/7/21 11:11
 * Version 1.0
 **/
@Mapper
public interface AxMapper {
    /**
     * 根据a_id查询加速度ax
     * @param aId
     * @return
     */
    public List<Double> getAx(Integer aId);

    /**
     * 插入ax加速度
     * @param ax
     */
    public void insertAx(Ax ax);

    /**
```

```
     * 删除ax
     * @param aId
     */
    public void deleteAx(Integer aId);
}
```

7.2.3 加速度AyMapper接口

```
package com.why.transportsecurity_finally.mapper;

import com.why.transportsecurity_finally.entity.Ay;
import org.apache.ibatis.annotations.Mapper;

import java.util.List;

/**
 * @Description TODO ay加速度mapper
 * @Author why
 * @Date 2021/7/21 11:18
 * Version 1.0
 **/
@Mapper
public interface AyMapper {

    /**
     * 根据aID查询ay
     * @param aID
     * @return
     */
    public List<Double> getAy(Integer aID);

    /**
     * 插入ay
     * @param ay
     */
    public void insertAy(Ay ay);

    /**
     * 删除ax
     * @param aId
     */
    public void deleteAy(Integer aId);
}
```

7.3 车辆信息相关

7.3.1 车辆信息VehicleinfoMapper

```
package com.why.transportsecurity_finally.mapper;


import com.why.transportsecurity_finally.entity.VehicleInfo;
import org.apache.ibatis.annotations.Mapper;

import java.util.List;
```

```java
/**
 * @Description TODO 车辆基本信息mapper
 * @Author why
 * @Date 2021/7/21 11:22
 * Version 1.0
 **/
@Mapper
public interface VehicleInfoMapper {

    /**
     * 根据车辆id查询车辆信息
     * @param id
     * @return
     */
    public VehicleInfo getVehicle(Integer id);

    /**
     * 根据车牌号查询车辆信息
     * @param vNum
     * @return
     */
    public VehicleInfo getVehicleByVNum(String vNum);

    /**
     * 获取所有车辆信息
     * @return
     */
    public List<VehicleInfo> getAll();

    /**
     * 添加司机信息
     * @param vehicleInfo
     */
    public void insertVehicle(VehicleInfo vehicleInfo);

    /**
     * 根据id删除车辆信息
     * @param id
     */
    public void deleteVehicle(Integer id);

    /**
     * 修改
     * @param vehicleInfo
     */
    public void updateVehicle(VehicleInfo vehicleInfo);


    /**
     * 根据车主姓名检索
     * @param vName
     * @return
     */
    public VehicleInfo getVehicleByVName(String vName);

    /**
     * 返回总记录数
     * @return
     */
    public Integer getCount();
}
```

八、Service层接口设计

8.1 管理员相关

8.1.1 AdminService接口设计

```
package com.why.transportsecurity_finally.service;

import com.why.transportsecurity_finally.entity.Admin;
import com.why.transportsecurity_finally.entity.AdminAll;
import com.why.transportsecurity_finally.entity.AdminInfo;

/**
 * @Description TODO 管理员登录注册
 * @Author why
 * @Date 2021/7/24 16:17
 * Version 1.0
 **/
public interface AdminService {

    /**
     * 登录
     * @param uId
     * @param pwd
     * @return
     */
    public boolean login(String uId,String pwd);

    /**
     * 返回用户姓名
     * @param uId
     * @return
     */
    public String loginName(String uId);

    /**
     * 注册
     * @param admin
     */
    public int register(Admin admin);

    /**
     * 根据uId获取admin
     * @param uId
     * @return
     */
    public Admin getAdmin(String uId);

    /**
     * 找回密码
     * @param uId
     * @param toAddress
     */
    public void findPwd(String uId,String toAddress);

    /*****************************************个人信息
***********************************/

    /**
     * 根据管理员id获取管理员信息
     * @param uId
     * @return
     */
    public AdminAll getAdminInfo(Integer uId);

    /**
     * 添加管理员信息
     * @param adminInfo
     * @return
     */
    public boolean insertAdminInfo(AdminInfo adminInfo);

    /**
     * 更新数据
     * @param adminAll
     * @return
     */
    public boolean updateAdminInfo(AdminAll adminAll);

    /**
     * 修改密码
     * @param uId
     * @param pwdNew
     * @param pwdOld
     * @return
     */
    public boolean updatePwd(Integer uId,String pwdNew,String pwdOld);

    /**
     * 修改密码，找回
     * @param uId
     * @param pwdNew
     * @return
     */
    public boolean updatePwdFind(String uId,String pwdNew);
}
```

8.2 事故业务相关

8.2.1 AccidentService接口设计

```
package com.why.transportsecurity_finally.service;

import com.why.transportsecurity_finally.entity.DetailInfo;
import com.why.transportsecurity_finally.entity.TotalInfo;
import com.why.transportsecurity_finally.entity.VehicleInfo;
import org.springframework.stereotype.Service;

import java.util.List;
```

```java
/**
 * @Description TODO 事故信息Service
 * @Author why
 * @Date 2021/7/21 11:42
 * Version 1.0
 **/
@Service
public interface AccidentService {
    /**
     * 返回统计信息
     *  1. 查询tbl_accident_info表
     *  2. 根据v_id查询tbl_vehicle_info表
     * @return
     */
    public List<TotalInfo> totalInfo();

    /**
     * 返回事故细节信息
     * @param VId 车辆基础信息id
     * @param aId 事故信息id
     * @return
     */
    public DetailInfo detailInfo(Integer VId, Integer aId);

    /**
     * 查询车辆基本信息
     * @param vId
     * @return
     */
    public VehicleInfo vehicleInfo(Integer vId);

    /**
     * 查询加速度ax
     * @param aId
     * @return
     */
    public List<Double> ax(Integer aId);

    /**
     * 查询加速度ay
     * @param aId
     * @return
     */
    public List<Double> ay(Integer aId);

    /**
     * 处理事故
     * @param aID
     */
    public void solveAccident(Integer aID);

    /**
     * 搜索功能
     * @param search
     * @return
     */
    public List<TotalInfo> search(String search);
}
```

8.3 车辆信息相关

8.3.1 VehicleInfoService接口设计

```java
package com.why.transportsecurity_finally.service;

import com.github.pagehelper.PageInfo;
import com.why.transportsecurity_finally.entity.VehicleInfo;

import java.util.List;
import java.util.Map;

/**
 * @Description TODO 车辆信息管理
 * @Author why
 * @Date 2021/7/26 14:39
 * Version 1.0
 **/
public interface VehicleService {

    /**
     * 添加车辆信息
     * @param vehicleInfo
     */
    public boolean insertVehicle(VehicleInfo vehicleInfo);

    /**
     * 删除车辆信息
     * @param id
     * @return
     */
    public boolean deleteVehicle(Integer id);

    /**
     * 获取所有车辆信息
     * @return
     */
    public Map<Integer,List<VehicleInfo>> getAll(Integer pageNum);

    /**
     * 修改车辆信息
     * @param vehicleInfo
     * @return
     */
    public boolean updateVehicle(VehicleInfo vehicleInfo);

    /**
     * 搜索
     * @param search
     * @return
     */
    public VehicleInfo search(String search);
}
```

九、Controller层控制转发设计

```java
package com.why.transportsecurity_finally.controller;

import com.why.transportsecurity_finally.entity.*;
import com.why.transportsecurity_finally.service.AccidentServiceImpl;
import com.why.transportsecurity_finally.service.AdminServiceImpl;
import com.why.transportsecurity_finally.service.VehicleServiceImpl;
import com.why.transportsecurity_finally.utils.*;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import lombok.extern.java.Log;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.websocket.server.PathParam;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

/**
 * @Description TODO 事故控制
 * @Author why
 * @Date 2021/7/21 15:54
 * Version 1.0
 **/
@Controller
@Log
@RequestMapping("/ts")
@Api(tags = "事故管理后台")
public class AccidentController {

    /*******************************************事故信息信息
****************************/

    @Autowired
    AccidentServiceImpl accidentService;

    @ApiOperation("事故列表")
    @RequestMapping("/accident/warning")
    public String warning(Model model){
        List<TotalInfo> totalInfos = accidentService.totalInfo();
        if (totalInfos.size() != 0){
            if (!totalInfos.get(0).isStatus()){
                model.addAttribute("flag",true);
            }else {
                model.addAttribute("flag",false);
            }
        }
        model.addAttribute("infos",totalInfos);
        return "accidentPage/warning";
    }

    @ApiOperation("事故数据展示，地址追踪")
    @RequestMapping("/accident/info")
    public String detailsInfo(HttpServletRequest request,Model model){
        String aIds = request.getParameter("aId");
        String vIds = request.getParameter("vId");
        int aId = Integer.valueOf(aIds);
        int vId = Integer.valueOf(vIds);
        model.addAttribute("aId",aId);
        model.addAttribute("vId",vId);

        DetailInfo detailInfo = accidentService.detailInfo(vId,aId);
        VehicleInfo vehicleInfo = accidentService.vehicleInfo(vId);
        model.addAttribute("vehicle",vehicleInfo);
        model.addAttribute("detailInfo",detailInfo);
        model.addAttribute("lng",detailInfo.getLng());
        model.addAttribute("lat",detailInfo.getLat());
        model.addAttribute("address",detailInfo.getAddress());
        return "accidentPage/detailsPage";
    }

    /**
     * 右侧信息展示
     * @param model
     * @param vId
     * @param aId
     * @return
     */
    @RequestMapping(value = "/accident/infoRight",method = RequestMethod.GET)
    public String showTask(Model model, @RequestParam("vId") Integer vId,
@RequestParam("aId") Integer aId,HttpServletRequest request){
        DetailInfo detailInfo = accidentService.detailInfo(vId, aId);
        model.addAttribute("detailInfo",detailInfo);
        VehicleInfo vehicleInfo = accidentService.vehicleInfo(vId);
        model.addAttribute("vehicle",vehicleInfo);
        List<Double> ax = accidentService.ax(aId);
        List<Double> ay = accidentService.ay(aId);
        List<Acceleration> axs = new ArrayList<>();
        for (int i = 0; i < 3; i++) {
            axs.add(new Acceleration(ax.get(i),ay.get(i)));
        }
        model.addAttribute("aId",aId);
        model.addAttribute("vId",vId);
        model.addAttribute("acceleration",axs);
        return "accidentPage/infoRight";
    }

    /**
     * 左侧信息展示
     * @param model
     * @param vId
     * @param aId
     * @return
     */
    @RequestMapping(value = "/accident/infoLeft",method = RequestMethod.GET)
    public String showTaskLeft(Model model, @RequestParam("vId") Integer vId,
@RequestParam("aId") Integer aId,HttpServletRequest request){
        DetailInfo detailInfo = accidentService.detailInfo(vId, aId);
        model.addAttribute("detailInfo",detailInfo);
        VehicleInfo vehicleInfo = accidentService.vehicleInfo(vId);
        model.addAttribute("vehicle",vehicleInfo);
        List<Double> ax = accidentService.ax(aId);
        List<Double> ay = accidentService.ay(aId);
        List<Acceleration> axs = new ArrayList<>();
        for (int i = 0; i < 3; i++) {
            axs.add(new Acceleration(ax.get(i),ay.get(i)));
        }
        model.addAttribute("aId",aId);
        model.addAttribute("vId",vId);
        model.addAttribute("acceleration",axs);
        return "accidentPage/infoLeft";
    }

    /**
     * 数据下载
     * @param response
     * @param vId
     * @param aId
     */
    @ApiOperation("碰撞数据下载")
    @RequestMapping("/accident/download")
    public void show(HttpServletResponse response, @RequestParam Integer vId,
@RequestParam Integer aId){
        //String vName, String vNumber, String vType, String degreeOfDamage,
String date,
        // String time, double lng, double lat, String address, String
collisionDirection,
        // String isBounce, double adRAcceleration, double adCAcceleration
        List<InfoExcelModel> list = new ArrayList<>();
        InfoExcelModel infoExcelModel = new InfoExcelModel();

        /**
         * 查出车辆基本信息
         */
        VehicleInfo vehicleInfo = accidentService.vehicleInfo(vId);
        infoExcelModel.setVName(vehicleInfo.getVName());
        infoExcelModel.setVNumber(vehicleInfo.getVNum());
        infoExcelModel.setVType(vehicleInfo.getVType());

        /**
         * 根据aId查询加速度信息
         */
        List<Double> ay = accidentService.ay(aId);
        List<Double> ax = accidentService.ax(aId);
        infoExcelModel.setAdRAcceleration(ax);
        infoExcelModel.setAdCAcceleration(ay);

        /**
         * 封装车辆事故基本信息
         */
        DetailInfo detailInfo = accidentService.detailInfo(vId, aId);
        //String vName, String vNumber, String vType, String degreeOfDamage,
String date,
        // String time, double lng, double lat, String address, String
collisionDirection,
        // String isBounce, double adRAcceleration, double adCAcceleration

 infoExcelModel.setDriverDegreeOfDamage(detailInfo.getDriverDegreeOfDamage());

 infoExcelModel.setPassengerDegreeOfDamage(detailInfo.getPassengerDegreeOfDamage
());

        infoExcelModel.setDate(detailInfo.getDate());
        infoExcelModel.setTime(detailInfo.getTime());
        infoExcelModel.setLng(detailInfo.getLng());
        infoExcelModel.setLat(detailInfo.getLat());
        infoExcelModel.setAddress(detailInfo.getAddress());
        infoExcelModel.setCollisionDirection(detailInfo.getDirection());
        infoExcelModel.setIsBounce(detailInfo.getIsBounce());

        list.add(infoExcelModel);

        OutputFileUtils.outputFile(list,response);
    }

    /**
     * 分装加速度信息 备注：需要修改
     * 将所有数据封装
     * @param model
     * @param aId
     * @return
     */
    @ApiOperation("数据分析，图表展示")
    @RequestMapping(value = "/accident/tu",method = RequestMethod.GET)
    public String axAndAy(Model model,@RequestParam("aId") Integer aId){
        List<Double> ax = accidentService.ax(aId);
        List<Double> ay = accidentService.ay(aId);
        model.addAttribute("ax",ax);
        model.addAttribute("ay",ay);
        return "accidentPage/tu";
    }

    @ApiOperation("处理事故")
    @RequestMapping(value = "/accident/solve")
    public String solveAccident(HttpServletRequest request,Model model){
        String aIds = request.getParameter("aId");
        String vIds = request.getParameter("vId");
        int aId = Integer.valueOf(aIds);
        int vId = Integer.valueOf(vIds);
        model.addAttribute("aId",aId);
        model.addAttribute("vId",vId);
        accidentService.solveAccident(Integer.valueOf(aId));
        DetailInfo detailInfo = accidentService.detailInfo(vId,aId);
        VehicleInfo vehicleInfo = accidentService.vehicleInfo(vId);
        model.addAttribute("vehicle",vehicleInfo);
        model.addAttribute("detailInfo",detailInfo);
        model.addAttribute("lng",detailInfo.getLng());
        model.addAttribute("lat",detailInfo.getLat());
        model.addAttribute("address",detailInfo.getAddress());
        System.out.println(detailInfo.isState());
        return "accidentPage/detailsPage";
    }

    /**
     * 搜寻事故信息
     * @param request
     * @param model
     * @return
     */
    @ApiOperation("搜索事故信息")
    @RequestMapping("/accident/totalSearch")
    public String search(HttpServletRequest request,Model model){
        String search = request.getParameter("search");
        if (search == null || search == ""){
            log.warning("搜索输入为空");
            return "redirect:/ts/accident/warning";
        }
        List<TotalInfo> totalInfos = accidentService.search(search);
        if (totalInfos.size() != 0){
            if (!totalInfos.get(0).isStatus()){
                model.addAttribute("flag",true);
            }else {
                model.addAttribute("flag",false);
            }
        }
        model.addAttribute("infos",totalInfos);
        return "accidentPage/warning";
    }


    /*******************************************管理员信息
****************************/

    @Autowired
    AdminServiceImpl adminService;

    @RequestMapping("/accident/showLogin")
    public String showLogin(){
        return "adminPage/signIn";
    }

    @RequestMapping("/accident/showRegister")
    public String showRegister(){
        return "adminPage/register";
    }

    @ApiOperation("登录")
    @RequestMapping("/accident/login")
    public String login(Model model, HttpServletRequest request,
HttpServletResponse response){
        String uId = request.getParameter("userId");
        String pwd = request.getParameter("pwd");
        boolean login = adminService.login(uId, pwd);
        if (!login){
            model.addAttribute("loginError","用户名或密码输入错误");
            return "adminPage/signIn";
        }
        CookieSessionUtils.cookieAndSession(request,response,uId);
        HttpSession session = request.getSession();
        session.setAttribute("login",adminService.loginName(uId));
        session.setAttribute("loginAdmin",uId);
        return "redirect:/ts/accident/warning";
    }

    @ApiOperation("注册")
    @RequestMapping("/accident/register")
    public String register(Model model,HttpServletRequest request){
        Admin admin = new Admin();
        String uId = request.getParameter("uId");
        String uName = request.getParameter("uName");
        String uPhone = request.getParameter("uPhone");
        String pwd = request.getParameter("pwd");

        admin.setUId(uId);
        admin.setUName(uName);
        admin.setUPwd(pwd);
        int register = adminService.register(admin);
        if (register == 1){
            model.addAttribute("registerError","输入不能为空");
            return "adminPage/register";
        }
        if (register == 2){
            model.addAttribute("registerError","用户名已存在");
            return "adminPage/register";
        }
        Admin admin1 = adminService.getAdmin(uId);
        AdminInfo adminInfo = new AdminInfo();
        adminInfo.setUId(admin1.getId());
        adminInfo.setUPhone(uPhone);
        adminService.insertAdminInfo(adminInfo);
        return "redirect:/ts/accident/showLogin";
    }

    @ApiOperation("注销账号")
    @RequestMapping("/accident/logOut")
    public String logOut(HttpServletRequest request,HttpServletResponse
response){
        log.info("正在登出...");
        CookieSessionUtils.deleteCookiesAndSession(request,response);
        return "redirect:/ts/accident/showLogin";
    }

    /**
     *个人中心
     * @return
     */
    @ApiOperation("个人中心")
    @RequestMapping("/accident/personal")
    public String showPerson(HttpServletRequest request,Model model){
        String uIds = (String) request.getSession().getAttribute("loginAdmin");
        Admin admin = adminService.getAdmin(uIds);
        if (admin == null){
            log.warning("未登录，请重新登录！");
            return "redirect:/ts/accident/showLogin";
        }
        AdminAll adminInfo = adminService.getAdminInfo(admin.getId());
        model.addAttribute("adminAll",adminInfo);
        return "adminPage/personalCenter";
    }

    /**
     * 基本信息修改
     * @param request
     * @return
     */
    @ApiOperation("管理员基本信息修改")
    @RequestMapping("/accident/personalUpdate")
    public String updateInfo(HttpServletRequest request){
        String uName = request.getParameter("uName");
        String uIds = request.getParameter("uIds");
        String uPhone = request.getParameter("uPhone");
        String uEmail = request.getParameter("uEmail");
        String uAddress = request.getParameter("uAddress");
        String uBirth = request.getParameter("uBirth");
        String uDate = request.getParameter("uDate");
        String id = request.getParameter("id");
        String uId = request.getParameter("uId");
        AdminAll adminAll = new AdminAll(uIds, uName, Integer.valueOf(id),
uPhone, uEmail, uAddress, uBirth, uDate, Integer.valueOf(uId));
        boolean b = adminService.updateAdminInfo(adminAll);
        if (!b){
            return "redirect:/ts/accident/personal";
        }
        return "redirect:/ts/accident/personal";
    }

    @ApiOperation("管理员设置新密码")
    @RequestMapping("/accident/newPwd")
    public String updatePwd(HttpServletRequest request){
        String uId = request.getParameter("uId");
        String oldPwd = request.getParameter("uOldPwd");
        String pwdNew = request.getParameter("uNewPwdPre");
        boolean b = adminService.updatePwd(Integer.valueOf(uId), pwdNew,
oldPwd);
        if (!b){
            return "redirect:/ts/accident/personal";
        }
        return "redirect:/ts/accident/personal";
    }

    @RequestMapping("/accident/showFindPre")
    public String findPwdPre(){
        return "adminPage/findPwd";
    }

    /**
     * 发送验证码
     * @param phone
     * @param uId
     */
    @ApiOperation("密码找回发送验证码")
    @RequestMapping(value = "/accident/sendPhone",method = RequestMethod.GET)
    @ResponseBody
    public void add(@PathParam("phone") String phone, @PathParam("uId") String
uId){
        if (phone == null){
            log.warning("电话号码输入为空！");
            return;
        }
        if (uId == null) {
            log.warning("uId输入为空");
            return;
        }
        //查询是否存在用户
        Admin admin = adminService.getAdmin(uId);
        if (admin == null){
            log.warning("密码找回，用户不存在");
            return;
        }
        AdminAll adminInfo = adminService.getAdminInfo(admin.getId());
        if (!adminInfo.getUPhone().equals(phone)){
            log.warning("输入电话未绑定");
            return;
        }
        //存在用户，修改密码，发送短信
        String newPwd = RandomUtils.getRandom();
        boolean b = adminService.updatePwdFind(uId, newPwd);
        if (!b){
            log.warning("找回密码，修改密码失败");
            return;
        }
        String[] phoneNum = {phone};
        String[] params = {newPwd};
        PhoneUtils.sendPhoneMail(phoneNum,params);
    }

    /**
     * 根据验证码登录
     * @param request
     * @param model
     * @return
     */
    @ApiOperation("验证码登录")
    @RequestMapping("/accident/findPwd")
    public String findPwd(HttpServletRequest request,Model
model,HttpServletResponse response){
        String uId = request.getParameter("uId");
        String toAddress = request.getParameter("toPwd");

        boolean login = adminService.login(uId, toAddress);
        if (!login){
            log.warning("验证码错误");
            model.addAttribute("msg","验证码错误");
            return "adminPage/findPwd";
        }
        CookieSessionUtils.cookieAndSession(request,response,uId);
        HttpSession session = request.getSession();
        session.setAttribute("login",adminService.loginName(uId));
        session.setAttribute("loginAdmin",uId);
        return "redirect:/ts/accident/warning";
    }
    /******************************************车辆信息
****************************/

    @Autowired
    VehicleServiceImpl vehicleService;

    /**
     * 展示所有车辆信息
     * @param model
     * @return
     */
    @ApiOperation("车辆信息统计")
    @RequestMapping("/accident/vehicles")
    public String vehicles(Model model,@PathParam("pageNum") Integer pageNum){
        Map<Integer, List<VehicleInfo>> serviceAll =
vehicleService.getAll(pageNum);
        Integer count = null;
        List<VehicleInfo> vehicles = null;
        for (Map.Entry entry : serviceAll.entrySet()) {
            count = (Integer) entry.getKey();
            vehicles = (List<VehicleInfo>) entry.getValue();
        }
        model.addAttribute("vehicles",vehicles);
        model.addAttribute("pageNum",pageNum);
        if ((count % 10) == 0){
            count = count / 10;
        }else {
            count = count / 10 +1;
        }
        if (count >= pageNum + 1){
            model.addAttribute("next1",true);
        }
        if (count >= pageNum + 2){
            model.addAttribute("next2",true);
        }
        return "vehiclePage/vehicleInfo";
    }

    /**
     * 修改车辆信息
     * @param request
     * @return
     */
    @ApiOperation("修改车辆信息")
    @RequestMapping("/accident/updateVehicle")
    public String updateVehicle(HttpServletRequest request){
        String id = request.getParameter("id");
        String vNum = request.getParameter("vNum");
        String vName = request.getParameter("vName");
        String vType = request.getParameter("vType");
        VehicleInfo vehicleInfo = new VehicleInfo(Integer.valueOf(id), vName,
vNum, vType);
        boolean b = vehicleService.updateVehicle(vehicleInfo);
        if (!b){
            log.warning(vNum + "信息更新失败");
            return "redirect:/ts/accident/vehicles";
        }
        log.info(vNum + "信息更新成功");
```

```java
        return "redirect:/ts/accident/vehicles?pageNum=1";
    }

    /**
     * 删除车辆信息
     * @param request
     * @return
     */
    @ApiOperation("删除车辆信息")
    @RequestMapping("/accident/deleteVehicle")
    public String deleteVehicle(HttpServletRequest request){
        String id = request.getParameter("id");
        String vNum = request.getParameter("vNum");
        boolean b = vehicleService.deleteVehicle(Integer.valueOf(id));
        if (!b){
            log.warning(vNum + "车辆信息删除失败");
            return "redirect:/ts/accident/vehicles";
        }
        log.warning(vNum + "车辆信息删除成功");
        return "redirect:/ts/accident/vehicles?pageNum=1";
    }

    @ApiOperation("添加车辆信息")
    @RequestMapping("/accident/insertVehicle")
    public String insertVehicle(HttpServletRequest request){
        String vName = request.getParameter("vName");
        String vNum = request.getParameter("vNum");
        String vType = request.getParameter("vType");
        boolean b = vehicleService.insertVehicle(new VehicleInfo(1, vName, vNum,
vType));
        if (!b){
            log.warning(vNum + "车辆信息添加失败");
            return "redirect:/ts/accident/vehicles";
        }
        log.warning(vNum + "车辆信息添加成功");
        return "redirect:/ts/accident/vehicles?pageNum=1";
    }

    @ApiOperation("搜索车辆信息")
    @RequestMapping("/accident/searchVehicle")
    public String searchvehicle(HttpServletRequest request,Model model){
        String search = request.getParameter("search");
        if (search == null || search == ""){
            return "redirect:/ts/accident/vehicles";
        }
        VehicleInfo vehicleInfo = vehicleService.search(search);
        List<VehicleInfo> vehicles = new ArrayList<>();
        vehicles.add(vehicleInfo);
        model.addAttribute("vehicles",vehicles);
        return "vehiclePage/vehicleInfo";
    }
}
```

十、Filter拦截器设计

10.1 登录认证Filter

```
package com.why.transportsecurity_finally.filter;

import com.why.transportsecurity_finally.utils.CookieSessionUtils;
import lombok.extern.java.Log;
import org.springframework.web.servlet.HandlerInterceptor;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.util.Arrays;
import java.util.Enumeration;

/**
 * @Description TODO 拦截器，登录检查
 * @Author why
 * @Date 2021/7/24 17:16
 * Version 1.0
 **/
@Log
public class LoginFilter implements HandlerInterceptor {

    /**
     * 目标方法执行前，登录检查
     *  1. 配置好拦截器拦截的请求
     *  2. 把拦截器配置放在容器中
     * @param request
     * @param response
     * @param handler
     * @return
     * @throws Exception
     */
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse
response, Object handler) throws Exception {
        log.info("LoginFilter拦截的请求: "+request.getRequestURI());
        /**
         * 登录检查逻辑
         */
        HttpSession session = request.getSession();
        Cookie[] cookies = request.getCookies();
        if (cookies==null){
            log.info("首次登录");
            response.sendRedirect("/ts/accident/showLogin");
            return false;
        }

        String ssid = null;
        for (Cookie cookie : cookies) {
            if (cookie.getName().equals("name") &&
session.getAttribute(cookie.getValue()) != null){
                ssid = (String) session.getAttribute(cookie.getValue());
            }
            if (cookie.getName().equals("val") &&
cookie.getValue().equals(ssid)){
                log.info(request.getRequestURL() + "登录验证成功");
                //验证成功放行
                return true;
            }
        }
        log.warning("未登录或session已失效");
        response.sendRedirect("/ts/accident/showLogin");
        return false;
    }

    /**
     * 目标方法执行后
     * @param request
     * @param response
     * @param handler
     * @param modelAndView
     * @throws Exception
     */
    @Override
    public void postHandle(HttpServletRequest request, HttpServletResponse
response, Object handler, ModelAndView modelAndView) throws Exception {

    }

    /**
     * 页面渲染后
     * @param request
     * @param response
     * @param handler
     * @param ex
     * @throws Exception
     */
    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse
response, Object handler, Exception ex) throws Exception {

    }
}
```

10.2 日志打印Filter

```
package com.why.transportsecurity_finally.filter;

import com.why.transportsecurity_finally.utils.IpUtils;
import lombok.extern.java.Log;
import org.springframework.web.servlet.HandlerInterceptor;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * @Description TODO 日志监控
 * @Author why
 * @Date 2021/7/25 10:38
 * Version 1.0
 **/
@Log
public class LogFilter implements HandlerInterceptor {
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse
response, Object handler) throws Exception {
        log.info(IpUtils.getIP(request) + "正在访问: "+request.getRequestURL());
        return true;
    }

    @Override
    public void postHandle(HttpServletRequest request, HttpServletResponse
response, Object handler, ModelAndView modelAndView) throws Exception {

    }

    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse
response, Object handler, Exception ex) throws Exception {

    }
}
```

十一、相关配置类

11.1 向Spring容器中添加filter组件

```
package com.why.transportsecurity_finally.config;

import com.why.transportsecurity_finally.filter.LogFilter;
import com.why.transportsecurity_finally.filter.LoginFilter;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

/**
 * @Description TODO 登陆检查filter配置
 * @Author why
 * @Date 2021/7/24 17:21
 * Version 1.0
 **/
@Configuration
public class AdminWebConfig implements WebMvcConfigurer {

    /**
     * 登陆检查filter配置
     * @param registry
     */
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new LoginFilter())
                .addPathPatterns("/**") //拦截的请求
.excludePathPatterns("/ts/accident/showLogin","/ts/accident/login","/ts/accident
/showRegister"

,"/ts/accident/register","/ts/accident/showFindPre","/ts/accident/findPwd","/ts/
accident/sendPhone"

,"/css/**","/img/**","/mapper/**","/js/**","/music/**","/favicon.ico","/error/**
","/druid/login.html");//放行的请求

        registry.addInterceptor(new LogFilter()).addPathPatterns("/**");
    }
}
```

11.2 Druid数据库连接池配置

```
package com.why.transportsecurity_finally.config;

import com.alibaba.druid.pool.DruidDataSource;
import com.alibaba.druid.support.http.StatViewServlet;
import com.alibaba.druid.support.http.WebStatFilter;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.boot.web.servlet.ServletRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import javax.sql.DataSource;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

/**
 * @Description TODO Druid配置使其他druid属性生效
 * @Author why
 * @Date 2021/7/30 14:58
 * Version 1.0
 **/
@Configuration
public class DruidConfig {
    @ConfigurationProperties(prefix = "spring.datasource")//将以spring.datasource
为前缀的属性绑定至容器
    @Bean
    public DataSource druid(){
        return new DruidDataSource();
    }

    /**
     * 配置druid监控
     */
    //1.配置管理后台的Servlet
    @Bean
    public ServletRegistrationBean statViewServlet(){
        ServletRegistrationBean bean = new ServletRegistrationBean(new
StatViewServlet(), "/druid/*");
        //配置初始化参数
        //配置的参数可在StatViewServlet()和StatViewServlet()的ResourceServlet父类中查看
        Map<String,String> initParams = new HashMap<>();
        //登录后台系统用户名
        initParams.put("loginUsername","admin");
        //登录后台系统密码
        initParams.put("loginPassword","123456");
        //允许谁访问
        initParams.put("allow","");//第二个参数不写或者为null时默认允许所有访问
        //配置拒绝谁访问
        initParams.put("deny","");
        bean.setInitParameters(initParams);
        return bean;
    }

    //2.配置一个监控的filter
    @Bean
    public FilterRegistrationBean webStatFilter(){
        FilterRegistrationBean bean = new FilterRegistrationBean();
        bean.setFilter(new WebStatFilter());
        //设置初始化参数
        //配置的参数可在WebStatFilter()中查看
        Map<String,String> initParams = new HashMap<>();
        //排除拦截的请求
        initParams.put("exclusions","*.js,*.css,/druid/*");
        bean.setInitParameters(initParams);
        //设置拦截的请求
        bean.setUrlPatterns(Arrays.asList("/*"));
        return bean;
    }
}
```

十二、配置文件

12.1 application.properties

```
#数据库
spring.datasource.username=root
#spring.datasource.password=why190810
#spring.datasource.url=jdbc:mysql://10.0.8.2:3306/transportsecurity_latest
spring.datasource.password=root
spring.datasource.url=jdbc:mysql://localhost:3306/transportsecurity_latest?zeroDateTimeBehavior=convertToNull&useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC&autoReconnect=true
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

#mybatis
mybatis.mapper-locations=classpath:mapper/*.xml
mybatis.type-aliases-package=com.why.transportsecurity_finally.entity

#开启驼峰命名
mybatis.configuration.map-underscore-to-camel-case=true

#日志的相关配置
logging.file.name=transportSecurity.log
logging.pattern.console=%d{yyyy-MM-dd HH:mm:SSS} [%thread] %-5level %logger{50}
- %msg%n
#指定文件中日志的输出格式
logging.pattern.file=%d{yyyy-MM-dd HH:mm:SSS} === [%thread] === %-5level ===
%logger{50} === %msg%n

#禁用模板引擎缓存
spring.thymeleaf.cache=false

#项目设置
server.port=80

#Swagger接口文档配置
swagger.base-path=/**
swagger.base-package=com.why
swagger.title=transport_security_new
swagger.description=基于Swagger构建的SpringBoot RESTApi文档
swagger.version=1.0
swagger.contact.name=why
swagger.contact.url=http://www.tsn.cn
swagger.contact.email=why_enterprise@163.com
```

12.1 application.yaml

```yaml
spring:
  datasource:
    #   数据源基本配置
    username: root
    password: root
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/transportsecurity_latest
    type: com.alibaba.druid.pool.DruidDataSource
    #   数据源其他配置
    initialSize: 5
    minIdle: 5
    maxActive: 20
    maxWait: 60000
    timeBetweenEvictionRunsMillis: 60000
    minEvictableIdleTimeMillis: 300000
    validationQuery: SELECT 1 FROM DUAL
    testWhileIdle: true
    testOnBorrow: false
    testOnReturn: false
    poolPreparedStatements: true
    #   配置监控统计拦截的filters，去掉后监控界面sql无法统计，'wall'用于防火墙
    filters: stat,wall
    maxPoolPreparedStatementPerConnectionSize: 20
    useGlobalDataSourceStat: true
    connectionProperties: druid.stat.mergeSql=true;druid.stat.slowSqlMillis=500
```

十三、工具类设计

13.1 事故相关

13.11 AccidentUtils

计算碰撞方向，损伤程度等

```java
package com.why.transportsecurity_finally.utils;

import java.util.Iterator;
import java.util.List;

/**
 * @Description TODO 事故处理工具类
 * @Author why
 * @Date 2021/7/21 14:20
 * Version 1.0
 **/
public class AccidentUtils {

    /**
     * 计算vx
     * @param ax
     * @return
     */
    public static double vx(List<Double> ax){
        double vx = 0.00;
        Iterator<Double> iterator = ax.iterator();
        while (iterator.hasNext()) {
            Double next = iterator.next();
            vx = vx + next;
        }
        return vx * 0.011;
    }

    /**
     * 计算vy
     * @param ay
     * @return
     */
    public static double vy(List<Double> ay){
        double vy = 0.00;
        Iterator<Double> iterator = ay.iterator();
        while (iterator.hasNext()) {
            Double next = iterator.next();
            vy = vy + next;
        }
        return vy * 0.021;
    }

    /**
     * 计算pdof
     * @param vx
     * @param vy
     * @return
     */
    public static double pdof(double vx,double vy){
        if (vx == 0.00){
            throw new RuntimeException("分母不能为零");
        }
        double pdof = 0.00;
        double temp = Math.toDegrees(Math.atan(vy/vx));
        if (vx >= 0 && vy <=0){
            pdof = temp + 180;
        }else if (vx >= 0 && vy >= 0){
            pdof = temp - 180;
        }else if (vx < 0){
            pdof = temp;
        }
        return pdof;
    }

    /**
     * 计算碰撞方向
     * 1：正面碰撞，2：左侧碰撞，3：追尾碰撞，4：右侧碰撞
     * @param pdof
     * @return
     */
    public static int direction(double pdof){
        int flag = 0;
        if (pdof >= -45 && pdof < 45){
            flag = 1;
        }else if (pdof >= -135 && pdof <-45){
            flag = 2;
        }else if (pdof >= -135 && pdof < 135){
            flag = 3;
        }else {
            flag = 4;
        }
        return flag;
    }

    /**
     * 计算驾驶员损伤程度
     * @param direction 碰撞方向
     * @param isBounce 安全气囊是否弹开，true：弹开，false：未弹开
     * @return true：严重损伤，false：一般损伤
     */
    public static boolean driverDegreeOfDamage(int direction,boolean
isBounce,double vx){
        double p = 0.00;
        if (direction == 1){//正面碰撞
            if (isBounce){//安全气囊弹开
                p = 1.000/(1.000 + Math.exp(7.91+0.145*vx));
            }else {//安全气囊未弹开
                p = 1.000 / (1.000 + Math.exp(5.43+0.145*vx));
            }
        }else if (direction == 2){
            if (isBounce){//安全气囊弹开
                p = 1.000/(1.000 + Math.exp(5.134+0.145*vx));
            }else {//安全气囊未弹开
                p = 1.000 / (1.000 + Math.exp(2.654+0.145*vx));
            }
        }

        return p > 0.189;
    }

    /**
     * 后排乘客损伤程度
     * @param vx
     * @return true：严重损伤，false：一般损伤
     */
    public static boolean passengerDegreeOfDamage(double vx){
        if (vx == 0.000){
            throw new RuntimeException("分母不能为0");
        }
        double p = 1.000 / (1.000 + Math.exp(10.614-3.477 * (1.000/vx) +
0.2128*vx));
        return p > 0.29;
    }
}
```

13.1.2 DateFormatUtis

日期格式化工具类

```java
package com.why.transportsecurity_finally.utils;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

/**
 * @Description TODO 时间格式化处理
 * @Author why
 * @Date 2021/6/4 13:32
 * Version 1.0
 **/
public class DateFormatUtils {

    /**
     * 返回当前时间的long值
     * @return
     */
    public static long dateLong(){
        return new Date().getTime();
    }

    /**
     * 时间戳long值转换为yyyy-MM-dd HH:mm:ss时间
     * @param time
     * @return
     */
    public static String myFormatTime(long time){
        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        return df.format(time);
    }

    public static java.sql.Date stringToDate(String dateStr){
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        Date date = null;
        try {
            date = format.parse(dateStr);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        java.sql.Date date1 = new java.sql.Date(date.getTime());
        return date1;
    }
}
```

13.1.3 OutputFileUtis

输出数据到excel文件工具类

```
package com.why.transportsecurity_finally.utils;

import com.why.transportsecurity_finally.entity.InfoExcelModel;
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFCellStyle;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.List;

/**
 * @Description TODO 文件下载
 * @Author why
 * @Date 2021/7/22 11:06
 * Version 1.0
 **/
public class OutputFileUtils {
    public static void outputFile(List<InfoExcelModel> list, HttpServletResponse response){
        //新建工作簿
        XSSFWorkbook workbook = new XSSFWorkbook();
        //列名数组
        String[] columnNames = {"车主","车牌号","车辆类型","驾驶员损伤情况","后排乘员损伤情况","日期","时间","经度","纬度","地址","碰撞方向","安全气囊是否弹开","横向加速度","纵向加速度"};

        //新建sheet
        XSSFSheet sheet = workbook.createSheet();
        //新建字体
        Font font = workbook.createFont();
        font.setFontName("simsun");
        font.setBold(true);
        font.setColor(IndexedColors.BLACK.index);

        //新建样式
        XSSFCellStyle titleStyle = workbook.createCellStyle();
        titleStyle.setAlignment(HorizontalAlignment.CENTER);
        titleStyle.setVerticalAlignment(VerticalAlignment.CENTER);
        titleStyle.setFillPattern(FillPatternType.SOLID_FOREGROUND);
        titleStyle.setFillForegroundColor(IndexedColors.YELLOW.index);
        titleStyle.setFont(font);

        //创建一行
        Row titleRow = sheet.createRow(0);
        //设置一行的单元格的值
        for (int i = 0; i < columnNames.length; i++) {
            Cell cell = titleRow.createCell(i);
            cell.setCellValue(columnNames[i]);
            cell.setCellStyle(titleStyle);
        }

        String number = null;
        //添加数据
        int lastRowNum = 0;
        for (int i = 0; i < list.size(); i++) {
            InfoExcelModel infoExcelModel = list.get(i);
            lastRowNum = lastRowNum + 1;
            //创建新行
            Row dataRow = sheet.createRow(lastRowNum);
            //创建单元格添加数据
            //"车主","车牌号","车辆类型","损伤情况","日期","时间","经度","纬度","地
址","碰撞方向","安全气囊是否弹开","横向加速度","纵向加速度"
            dataRow.createCell(0).setCellValue(infoExcelModel.getVName());
            number = infoExcelModel.getVNumber();
            dataRow.createCell(1).setCellValue(infoExcelModel.getVNumber());
            dataRow.createCell(2).setCellValue(infoExcelModel.getVType());

 dataRow.createCell(3).setCellValue(infoExcelModel.getDriverDegreeOfDamage());

 dataRow.createCell(4).setCellValue(infoExcelModel.getPassengerDegreeOfDamage())
;
            dataRow.createCell(5).setCellValue(infoExcelModel.getDate());
            dataRow.createCell(6).setCellValue(infoExcelModel.getTime());
            dataRow.createCell(7).setCellValue(infoExcelModel.getLng());
            dataRow.createCell(8).setCellValue(infoExcelModel.getLat());
            dataRow.createCell(9).setCellValue(infoExcelModel.getAddress());

 dataRow.createCell(10).setCellValue(infoExcelModel.getCollisionDirection());
            dataRow.createCell(11).setCellValue(infoExcelModel.getIsBounce());

 dataRow.createCell(12).setCellValue(infoExcelModel.getAdRAcceleration().get(0))
;

 dataRow.createCell(13).setCellValue(infoExcelModel.getAdCAcceleration().get(0))
;
        }
        if (list != null && list.size() > 0 ){
            InfoExcelModel infoExcelModel = list.get(0);
            int max = 0;
            boolean flag = true;
            if (infoExcelModel.getAdRAcceleration().size() >=
infoExcelModel.getAdCAcceleration().size()){
                max = infoExcelModel.getAdRAcceleration().size();
                flag = true;
            }else {
                max = infoExcelModel.getAdCAcceleration().size();
                flag = false;
            }
            int lastRowNumR = 1;
            for (int i = 0; i < max-1; i++) {
                Row dataRow = sheet.createRow(lastRowNumR + 1);
                double ax = 0.000;
                double ay = 0.000;
                if (flag){
                    if (i > infoExcelModel.getAdCAcceleration().size() - 2){
                        ay = 0.000;
                    }else {
                        ay = infoExcelModel.getAdCAcceleration().get(i+1);
                    }
                    ax = ax = infoExcelModel.getAdRAcceleration().get(i+1);
                }else {
                    if (i > infoExcelModel.getAdRAcceleration().size() -2){
                        ax = 0.000;
                    }else {
                        ax = infoExcelModel.getAdRAcceleration().get(i+1);
                    }
                    ay = infoExcelModel.getAdCAcceleration().get(i+1);
                }
                dataRow.createCell(12).setCellValue(ax);
                dataRow.createCell(13).setCellValue(ay);
                lastRowNumR++;
            }

        }

        try {
            response.setContentType("application/vnd.ms-excel");
            if (number != null){
                response.setHeader("content-Disposition", "attachment;filename="
+ URLEncoder.encode(number+".xlsx", "utf-8"));
            }else {
                response.setHeader("content-Disposition", "attachment;filename="
+ URLEncoder.encode("data.xlsx", "utf-8"));
            }
            response.setHeader("Access-Control-Expose-Headers", "content-
Disposition");
            OutputStream outputStream = response.getOutputStream();
            workbook.write(outputStream);
            outputStream.flush();
            outputStream.close();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

13.1.4 BaiduMapUtils

百度地图正\逆地址解析

```
package com.why.transportsecurity_finally.utils;
```

```java
import com.alibaba.fastjson.JSONObject;
import lombok.extern.slf4j.Slf4j;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;


/**
 * @Description TODO 百度地图逆地址解析
 * @Author why
 * @Date 2021/6/5 16:32
 * Version 1.0
 **/
@Slf4j
public class BaiduMapUtils {

    //应用ak
    private static final String MAP_AK = "pEW1fFdsLBZdKFhUUtmtSgtL4cyAdcxe";
    //接口地址
    public static String MAP_URL =
"http://api.map.baidu.com/reverse_geocoding/v3/?ak=" +
MAP_AK+"&output=json&coordtype=bd09ll$addressComponent=street_number&level=门址";
    /**
     * 将经纬度获取解析成详细地址
     *
     * @param lng
     *            经度
     * @param lat
     *            纬度
     * @return
     */
    public static String getAddress(double lng, double lat) {
        String address = "";
        String location = lat + "," + lng;
        BufferedReader in = null;
        URL url = null;
        URLConnection connection = null;
        try {
            url = new URL(MAP_URL + "&location=" + location);
            connection = url.openConnection();
            connection.setDoOutput(true);
            in = new BufferedReader(new InputStreamReader(url.openStream(),
"UTF-8"));
            String line;
            StringBuilder text = new StringBuilder("");
            while ((line = in.readLine()) != null) {
                text.append(line.trim());
            }
            JSONObject result = JSONObject.parseObject(text.toString());
            if (result != null && result.getIntValue("status") == 0) {
                log.debug(String.valueOf(result.getJSONObject("result")));
                address =
result.getJSONObject("result").getString("formatted_address") + "-" +
result.getJSONObject("result").getString("business");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return address;
    }

    /**
     * 将地址解析成经纬度
     *
     * @param address
     *            地址，例：浙江省杭州市西湖区
     * @return 返回经纬度数据。例：
{"lng":120.08899292561351,"lat":30.207036169515438}
     */
    public static JSONObject getPosition(String address) {
        BufferedReader in = null;
        URL url = null;
        URLConnection connection = null;
        try {
            url = new URL("http://api.map.baidu.com/geocoding/v3/?
address="+address+"&output=json&ak="+MAP_AK+"&callback=showLocation&confidence=1
00&comprehension=100&level=道路");
            connection = url.openConnection();
            connection.setDoOutput(true);
            in = new BufferedReader(new InputStreamReader(url.openStream(),
"UTF-8"));
            String line;
            StringBuilder text = new StringBuilder("");
            while ((line = in.readLine()) != null) {
                text.append(line.trim());
            }
            JSONObject result = JSONObject.parseObject(text.toString());
            if (result != null && result.getIntValue("status") == 0) {
                return result.getJSONObject("result").getJSONObject("location");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

13.2 管理员相关

13.2.1 CookieSessionUtils

添加\移除Cookie和Session

```
package com.why.transportsecurity_finally.utils;

import lombok.extern.java.Log;

import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.util.Collections;
import java.util.Enumeration;
```

```java
import java.util.UUID;

/**
 * @Description TODO cookie + session登录认证
 * @Author why
 * @Date 2021/7/24 17:07
 * Version 1.0
 **/
@Log
public class CookieSessionUtils {

    /**
     * 1.创建session
     * 2.创建cookie
     * @param request
     */
    public static void cookieAndSession(HttpServletRequest request,
HttpServletResponse response, String uId){
        String uuid = UUID.randomUUID().toString();
        HttpSession session = request.getSession();
        byte[] key = Sm4Util.getBytes("whylovewyy", 16);
        byte[] bytes = null;
        try {
            bytes = Sm4Util.encryptEcbPkcs5Padding(uId.getBytes(), key);
        } catch (Exception e) {
            e.printStackTrace();
        }
        session.setAttribute(uuid,bytes.toString());
        Cookie cookieName = new Cookie("name",uuid);
        Cookie cookieVal = new Cookie("val",bytes.toString());
        response.addCookie(cookieName);
        response.addCookie(cookieVal);
    }

    public static void deleteCookiesAndSession(HttpServletRequest request,
HttpServletResponse response){
        Cookie[] cookies = request.getCookies();
        for (Cookie cookie : cookies) {
            if (cookie.getName().equals("name")){
                cookie.setValue(null);
                response.addCookie(cookie);
            }
            if (cookie.getName().equals("val")){
                cookie.setValue(null);
                response.addCookie(cookie);
            }
        }
        HttpSession session = request.getSession();
        session.invalidate();
    }
}
```

13.2.2 IpUtis

获取访问用户的IP

```java
package com.why.transportsecurity_finally.utils;

import javax.servlet.http.HttpServletRequest;
import java.net.InetAddress;
import java.net.UnknownHostException;

/**
 * @Description TODO 获取真实IP
 * @Author why
 * @Date 2021/7/25 10:43
 * Version 1.0
 **/
public class IpUtils {
    /**
     * 获取客户端IP地址，针对Nginx等反代作处理
     * @param request
     * @return
     */
    public static String getIP(HttpServletRequest request){
        String ip = request.getHeader("x-forwarded-for");
        if(ip == null || ip.length() == 0 ||     "unknown".equalsIgnoreCase(ip))
{
            ip = request.getHeader("Proxy-Client-IP");
        }
        if(ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
            ip = request.getHeader("WL-Proxy-Client-IP");
        }
        if(ip==null || ip.length()==0 || "unknown".equalsIgnoreCase(ip)){
            ip=request.getHeader("X-Real-IP");
        }
        if(ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {
            ip = request.getRemoteAddr();
            if(ip.equals("127.0.0.1") || ip.equals("0:0:0:0:0:0:0:1")){
                //根据网卡取本机配置的IP
                InetAddress inet=null;
                try {
                    inet = InetAddress.getLocalHost();
                } catch (UnknownHostException e) {
                    e.printStackTrace();
                }
                ip= inet.getHostAddress();
            }
        }
        //对于通过多个代理的情况，第一个IP为客户端真实IP,多个IP按照','分割
        if(ip!=null && ip.length()>15){ //"***.***.***.***".length() = 15
            if(ip.indexOf(",")>0){
                ip = ip.substring(0,ip.indexOf(","));
            }
        }
        return ip;
    }
}
```

13.2.3 MailUtils

发送email进行密码找回

```
package com.why.transportsecurity_finally.utils;

import com.sun.mail.util.MailSSLSocketFactory;

import javax.mail.Address;
import javax.mail.Message;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.util.Date;
import java.util.Properties;

/**
 * @Description TODO 发送邮件依赖
 * @Author why
 * @Date 2021/7/27 16:55
 * Version 1.0
 **/
public class MailUtils {
    //邮件服务器主机名
    // QQ邮箱的 SMTP 服务器地址为: smtp.qq.com
    private static String myEmailSMTPHost = "smtp.qq.com";

    //发件人邮箱
    private static String myEmailAccount = "488009667@qq.com";

    //发件人邮箱密码（授权码）
    //在开启SMTP服务时会获取到一个授权码，把授权码填在这里
    private static String myEmailPassword = "vffphffvdvhgcaii";

    /**
     * 邮件单发（自由编辑短信，并发送，适用于私信）
     *
     * @param toEmailAddress 收件箱地址
     * @param emailTitle 邮件主题
     * @param emailContent 邮件内容
     * @throws Exception
     */
    public static void sEmail(String toEmailAddress, String emailTitle, String
emailContent) throws Exception{

        Properties props = new Properties();

        // 开启debug调试
        props.setProperty("mail.debug", "true");

        // 发送服务器需要身份验证
        props.setProperty("mail.smtp.auth", "true");

        // 端口号
        props.put("mail.smtp.port", 465);

        // 设置邮件服务器主机名
        props.setProperty("mail.smtp.host", myEmailSMTPHost);

        // 发送邮件协议名称
        props.setProperty("mail.transport.protocol", "smtp");

        /**SSL认证，注意腾讯邮箱是基于SSL加密的，所以需要开启才可以使用**/
        MailSSLSocketFactory sf = new MailSSLSocketFactory();
        sf.setTrustAllHosts(true);

        //设置是否使用ssl安全连接（一般都使用）
        props.put("mail.smtp.ssl.enable", "true");
        props.put("mail.smtp.ssl.socketFactory", sf);

        //创建会话
        Session session = Session.getInstance(props);

        //获取邮件对象
        //发送的消息，基于观察者模式进行设计的
        Message msg = new MimeMessage(session);

        //设置邮件标题
        msg.setSubject(emailTitle);

        //设置邮件内容
        //使用StringBuilder，因为StringBuilder加载速度会比String快，而且线程安全性也不错
        StringBuilder builder = new StringBuilder();

        //写入内容
        builder.append("\n" + emailContent);

        //设置显示的发件时间
        msg.setSentDate(new Date());

        //设置邮件内容
        msg.setText(builder.toString());

        //设置发件人邮箱
        // InternetAddress 的三个参数分别为: 发件人邮箱, 显示的昵称(只用于显示, 没有特别的
要求), 昵称的字符集编码
        msg.setFrom(new InternetAddress(myEmailAccount,"我的工作站", "UTF-8"));

        //得到邮差对象
        Transport transport = session.getTransport();

        //连接自己的邮箱账户
        //密码不是自己QQ邮箱的密码，而是在开启SMTP服务时所获取到的授权码
        //connect(host, user, password)
        transport.connect( myEmailSMTPHost, myEmailAccount, myEmailPassword);

        //发送邮件
        transport.sendMessage(msg, new Address[] { new
InternetAddress(toEmailAddress) });

        //将该邮件保存到本地
        OutputStream out = new FileOutputStream("MyEmail.eml");
        msg.writeTo(out);
        out.flush();
        out.close();

        transport.close();
    }

    public static void sendMail(String toAddress){
        //邮件主题
        String emailTitle = "邮箱验证";
        //邮件内容
        String emailContent = "transport security\r\n" +
                "\r\n" +
                "date: " +
DateFormatUtils.myFormatTime(DateFormatUtils.dateLong()) + "\r\n" +
                "\r\n" +
                "您正在进行找回密码，我们将您的密码暂时更改为：" + 123456 + "，请于5分钟内
完成验证！\r\n" +
                "\r\n" +
                "不要将此密码告知于他人，并在登陆后及时更改密码！\r\n" +
                "\r\n" +
                "登录地址：http:106.52.60.176/ts/accident/showLogin";
        //发送邮件
        try {
            MailUtils.sEmail(toAddress,emailTitle,emailContent);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

13.2.4 PhoneUtils

发送短信验证码

```
package com.why.transportsecurity_finally.utils;


import com.github.qcloudsms.SmsSingleSender;
import com.github.qcloudsms.SmsSingleSenderResult;
import com.github.qcloudsms.httpclient.HTTPException;
import lombok.extern.java.Log;
import org.json.JSONException;
import java.io.IOException;

/**
 * @Description TODO 发送短信
 * @Author why
 * @Date 2021/7/28 9:43
 * Version 1.0
 **/
@Log
public class PhoneUtils {

    /**
     * 返送短信
     * @param phoneNumbers 电话号码
     * @param params 短信内容
     */
    public static void sendPhoneMail(String[] phoneNumbers,String[] params){
        // 短信应用 SDK AppID
        int appid = 1400553539; // SDK AppID 以1400开头
// 短信应用 SDK AppKey
        String appkey = "9ae23a3768079d6c637375eb9351a57d";
// 短信模板 ID，需要在短信应用中申请
        int templateId = 1053503; // NOTE: 这里的模板 ID`7839`只是示例，真实的模板 ID
需要在短信控制台中申请
// 签名
        String smsSign = "风雪踏梦行"; // NOTE: 签名参数使用的是`签名内容`，而不是`签名
ID`。这里的签名"腾讯云"只是示例，真实的签名需要在短信控制台申请

        try {
            SmsSingleSender ssender = new SmsSingleSender(appid, appkey);
            SmsSingleSenderResult result = ssender.sendWithParam("86",
phoneNumbers[0],
                    templateId, params, smsSign, "", "");
            System.out.println(result);
        }  catch (JSONException e) {
            // JSON 解析错误
            log.warning("json解析错误");
            e.printStackTrace();
        } catch (IOException e) {
            // 网络 IO 错误
            log.warning("网络IO错误");
            e.printStackTrace();
        } catch (HTTPException e) {
            log.warning("错误");
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        String[] phone = {"15337086013"};
//          String[] phone = {"18694336926"};
        String random = RandomUtils.getRandom();
        String[] params = {random};
        sendPhoneMail(phone,params);
    }
}
```

13.2.5 RandomUtils

产生随机数

```
package com.why.transportsecurity_finally.utils;

import java.util.Date;
import java.util.Random;
import java.util.UUID;

/**
 * @Description TODO 随机数工具类
 * @Author why
 * @Date 2021/7/28 11:53
 * Version 1.0
 **/
public class RandomUtils {

    public static String getRandom(){
        Date date = new Date();
        Random random = new Random(date.getTime());
        int i = random.nextInt(1000000);
        return i+"";
    }
}
```

13.2.6 SaltUtis

数据加盐加密Hash

```
package com.why.transportsecurity_finally.utils;

import java.security.MessageDigest;
import java.util.Random;

/**
 * @Description TODO 加盐
 * @Author why
 * @Date 2021/6/15 13:56
 * Version 1.0
 **/
public class SaltUtil {

    /**
     * 加盐加密
     * @param psd
     * @return
     */
    public static String merge(String psd,String salt) {
        //加盐
        String str = Sm3Util.addSalt(psd, salt);
        //Sm4加密
        byte[] key = Sm4Util.getBytes("whylovewyy", 16);
        try {
            byte[] bytes = Sm4Util.encryptEcbPkcs5Padding(str.getBytes(),key);
            //hash
            String pwdNew = Sm3Util.hashStr(new String(bytes));
            return pwdNew;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

13.2.7 SM3Utils

SM3哈希密码算法

```
package com.why.transportsecurity_finally.utils;


import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.util.encoders.Hex;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Random;

/**
 * @Description TODO SM3哈希算法
 * @Author why
 * @Date 2021/6/15 13:44
 * Version 1.0
 **/
public class Sm3Util {
    public static String addSalt(String str,String salt){
        String newStr = str + salt;

        BouncyCastleProvider provider = new BouncyCastleProvider();
        try {
            MessageDigest digest = MessageDigest.getInstance("SM3",provider);
            byte[] encode =
Hex.encode(digest.digest(newStr.getBytes(StandardCharsets.UTF_8)));
            String s = new String(encode);
            return s;
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return new String();
    }

    /**
     * 随机生成10位字符串返回
     * @return
     */
    public static String creatSalt() {
        String str =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
        Random random = new Random();
        StringBuffer stringBuffer = new StringBuffer();
        for (int i = 0; i < 10; i++) {
            int number = random.nextInt(62);
            stringBuffer.append(str.charAt(number));
        }
        return stringBuffer.toString();
    }

    /**
     * 对字符串进行hash
     * @param str
     * @return
     */
    public static String hashStr(String str){
        BouncyCastleProvider provider = new BouncyCastleProvider();
        try {
            MessageDigest digest = MessageDigest.getInstance("SM3",provider);
            byte[] encode =
Hex.encode(digest.digest(str.getBytes(StandardCharsets.UTF_8)));
            String s = new String(encode);
            return s;
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return new String();
    }
}
```

13.2.8 SM4Utils

SM4对称加密算法

```
package com.why.transportsecurity_finally.utils;

import org.bouncycastle.jce.provider.BouncyCastleProvider;

import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.security.Security;

/**
 * @Description TODO SM4对称加密
 * @Author why
 * @Date 2021/6/15 13:55
 * Version 1.0
 **/
public class Sm4Util {
    private static final String ALGORITHM_NAME = "SM4";
    private static final String ALGORITHM_ECB_PKCS5PADDING =
"SM4/ECB/PKCS5Padding";

    /**
     * SM4算法目前只支持128位（即密钥16字节）
     */
    private static final int DEFAULT_KEY_SIZE = 128;

    static {
        // 防止内存中出现多次BouncyCastleProvider的实例
        if (null == Security.getProvider(BouncyCastleProvider.PROVIDER_NAME)) {
            Security.addProvider(new BouncyCastleProvider());
        }
    }

    private Sm4Util() {
    }

    /**
     * 获取指定长的的字符串对应的16进制字节码，如果长度不够，末位自动补0
     * @param s
     * @return
     */
    public static byte[] getBytes(String s, int length) {
        int fixLength = length - s.getBytes().length;
        if (s.getBytes().length < length) {
            byte[] S_bytes = new byte[length];
            System.arraycopy(s.getBytes(), 0, S_bytes, 0, s.getBytes().length);
            for (int x = length-fixLength; x < length; x++) {
                S_bytes[x] = 0x00;
            }
            return S_bytes;
        }
        return s.getBytes();
    }

    /**
     * 加密，SM4-ECB-PKCS5Padding
     *
     * @param data 要加密的明文
     * @param key  密钥16字节，使用Sm4Util.generateKey()生成
     * @return 加密后的密文
     * @throws Exception 加密异常
     */
    public static byte[] encryptEcbPkcs5Padding(byte[] data, byte[] key) throws
Exception {
        return sm4(data, key, ALGORITHM_ECB_PKCS5PADDING, null,
Cipher.ENCRYPT_MODE);
    }

    /**
     * 解密，SM4-ECB-PKCS5Padding
     *
     * @param data 要解密的密文
     * @param key  密钥16字节，使用Sm4Util.generateKey()生成
     * @return 解密后的明文
     * @throws Exception 解密异常
     */
    public static byte[] decryptEcbPkcs5Padding(byte[] data, byte[] key) throws
Exception {
        return sm4(data, key, ALGORITHM_ECB_PKCS5PADDING, null,
Cipher.DECRYPT_MODE);
    }

    /**
     * SM4对称加解密
     *
     * @param input   明文（加密模式）或密文（解密模式）
     * @param key     密钥
     * @param sm4mode sm4加密模式
     * @param iv      初始向量(ECB模式下传NULL)
     * @param mode    Cipher.ENCRYPT_MODE - 加密；Cipher.DECRYPT_MODE - 解密
     * @return 密文（加密模式）或明文（解密模式）
     * @throws Exception 加解密异常
     */
    private static byte[] sm4(byte[] input, byte[] key, String sm4mode, byte[]
iv, int mode)
            throws Exception {
```

```
        IvParameterSpec ivParameterSpec = null;
        if (null != iv) {
            ivParameterSpec = new IvParameterSpec(iv);
        }
        SecretKeySpec sm4Key = new SecretKeySpec(key, ALGORITHM_NAME);
        Cipher cipher = Cipher.getInstance(sm4mode,
BouncyCastleProvider.PROVIDER_NAME);
        if (null == ivParameterSpec) {
            cipher.init(mode, sm4Key);
        } else {
            cipher.init(mode, sm4Key, ivParameterSpec);
        }
        return cipher.doFinal(input);
    }
}
```

十四、TCP通信模块

14.1 创建服务端线程

负责与车载终端的通信，创建线程任务

```
package com.why.transportsecurity_finally.tcpSocket;

import lombok.extern.slf4j.Slf4j;
import java.io.*;
import java.net.Socket;
import java.net.ServerSocket;
/**
 * @Description TODO socket服务端
 * @Author why
 * @Date 2021/8/12 13:50
 * Version 1.0
 **/
@Slf4j
public class tcpServerSocket {
    public static void server(Integer port){
        log.info("服务端启动. . .");
        ServerSocket server = null;
        try {
            server = new ServerSocket(port);
            while (true){
                //获取一个客户端连接
                Socket accept = server.accept();
                //创建信息线程处理客户端信息
                new Thread(() -> {
                    InputStream is = null;
                    ByteArrayOutputStream baos = null;
                    try {
                        //获取字节输入流
                        is = accept.getInputStream();
                        byte[] buffer = new byte[1024];
                        int len;
                        baos = new ByteArrayOutputStream();
                        while ((len = is.read(buffer)) != -1){
                            //写到baos中，中有个数组，会自动扩容
                            baos.write(buffer,0,len);
                        }
                        log.info(accept.getRemoteSocketAddress()+" -
"+baos.toString());
                    } catch (IOException e) {
                        e.printStackTrace();
                    }finally {
                        try {
                            if (is != null){
                                is.close();
                            }
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
                },"server").start();
                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

14.2 线程池执行任务

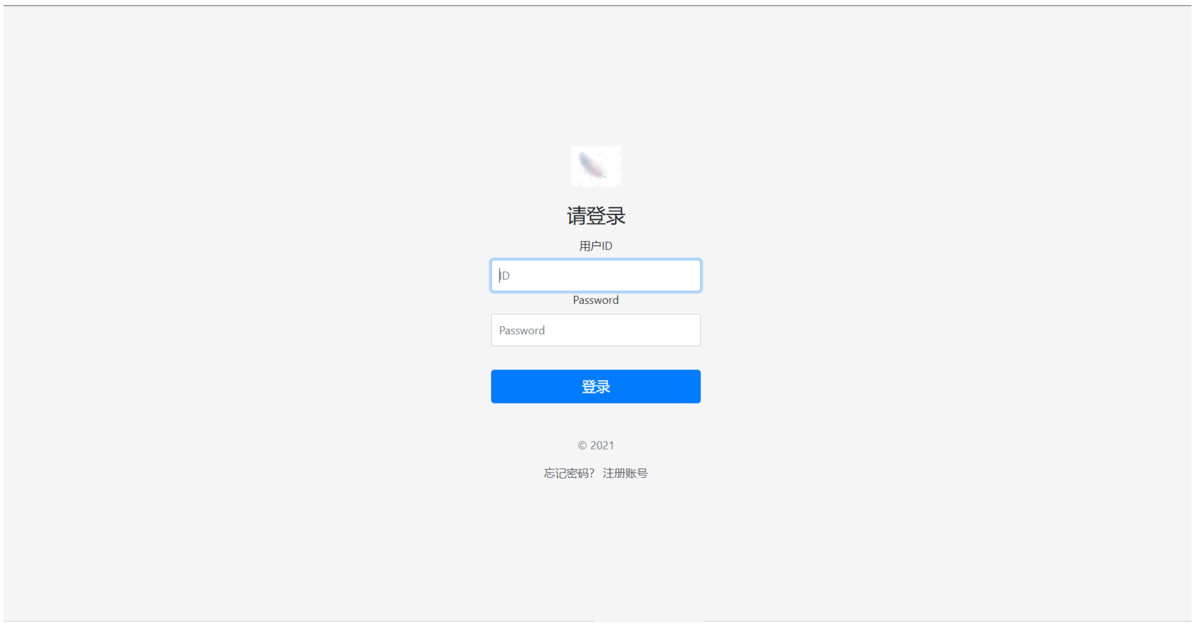
线程池执行已创建的线程任务，并随项目启动

```
package com.why.transportsecurity_finally.component;

import com.why.transportsecurity_finally.config.SocketConfig;
import com.why.transportsecurity_finally.tcpSocket.ServerTcp;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

import java.net.ServerSocket;
import java.net.Socket;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;

/**
 * @ClassName: SocketComponent
 * @Description: todo 让Socket程序跟随Springboot一起启动
 * @Author: why
 * @DateTime: 2021/8/26 17:55
 */
@Component
@Slf4j
public class SocketComponent implements CommandLineRunner {
    @Autowired
    private SocketConfig socketConfig;

    @Override
    public void run(String... args) throws Exception {
        ServerSocket server = null;
        Socket socket = null;
        server = new ServerSocket(socketConfig.getPort());
        log.info("设备服务器已启动，监听端口："+ socketConfig.getPort());
        //创建线程池
        ThreadPoolExecutor pool = new ThreadPoolExecutor(
                socketConfig.getPoolCore(),
                socketConfig.getPoolMax(),
                socketConfig.getPoolKeep(),
                TimeUnit.SECONDS,
                new ArrayBlockingQueue<Runnable>
(socketConfig.getPoolQueueInit()),
                new ThreadPoolExecutor.DiscardOldestPolicy()
        );
        //循环监听
        while (true){
            socket = server.accept();
            pool.execute(new ServerTcp(socket));
        }
    }
}
```

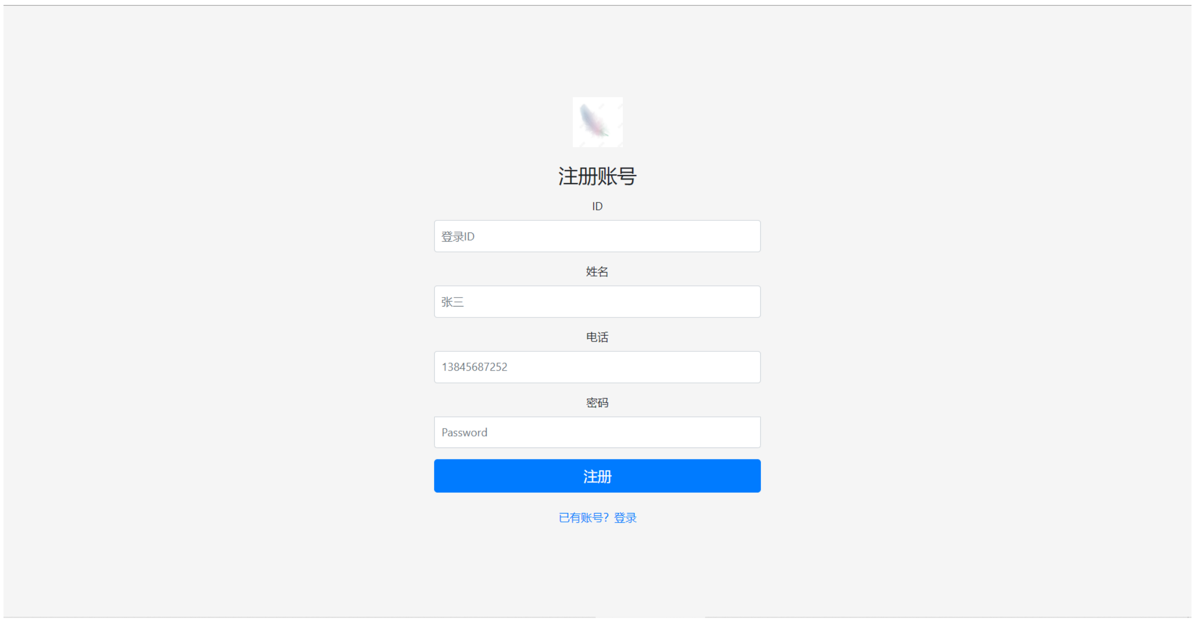
十五、 产品展示

15.1 管理员模块

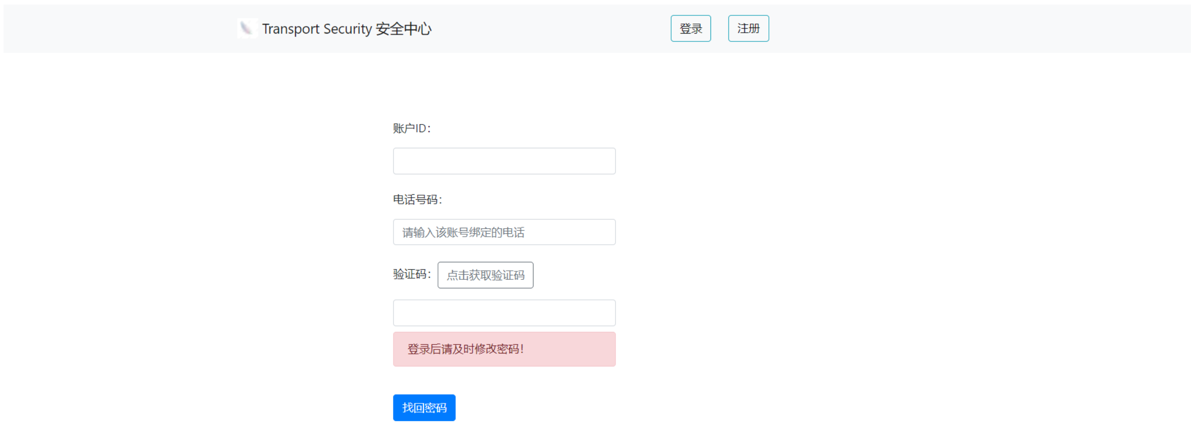
登录



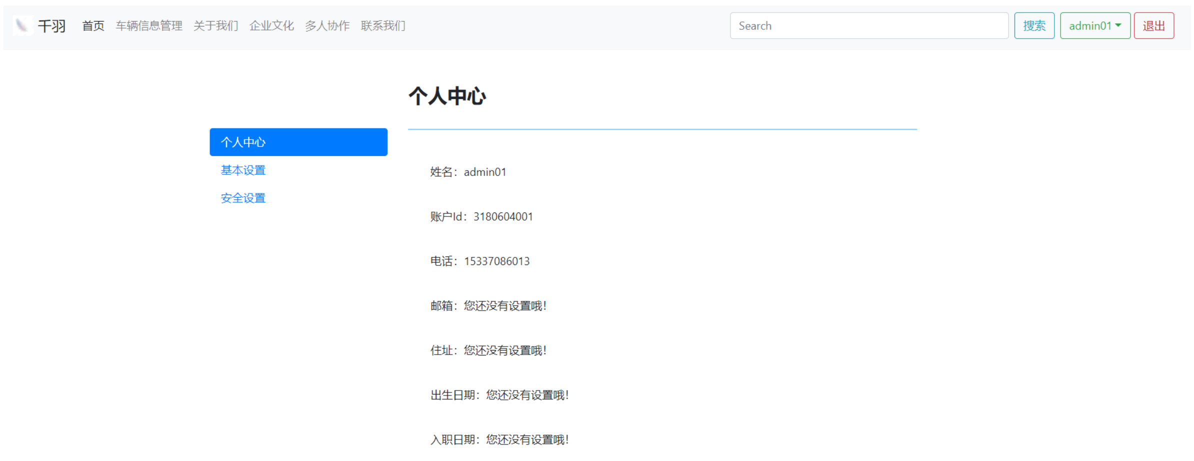
注册



找回密码



个人中心



姓名：admin01

账户Id：3180604001

电话：15337086013

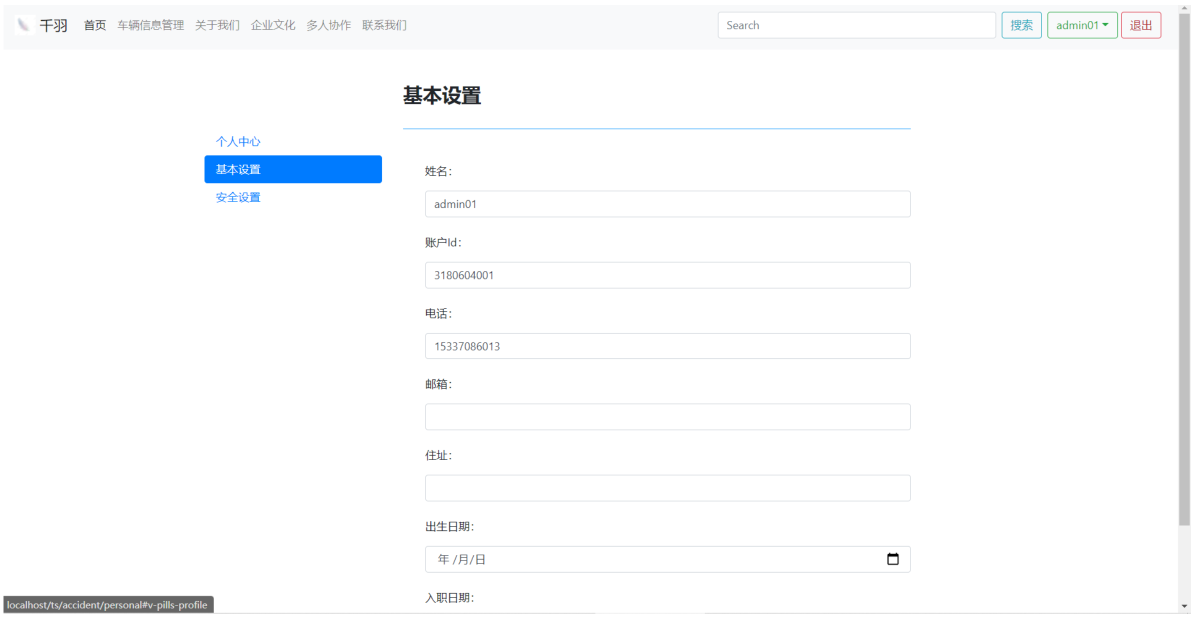
邮箱：您还没有设置哦！

住址：您还没有设置哦！

出生日期：您还没有设置哦！

入职日期：您还没有设置哦！

基本信息管理



基本设置

姓名：admin01

账户Id：3180604001

电话：15337086013

邮箱：

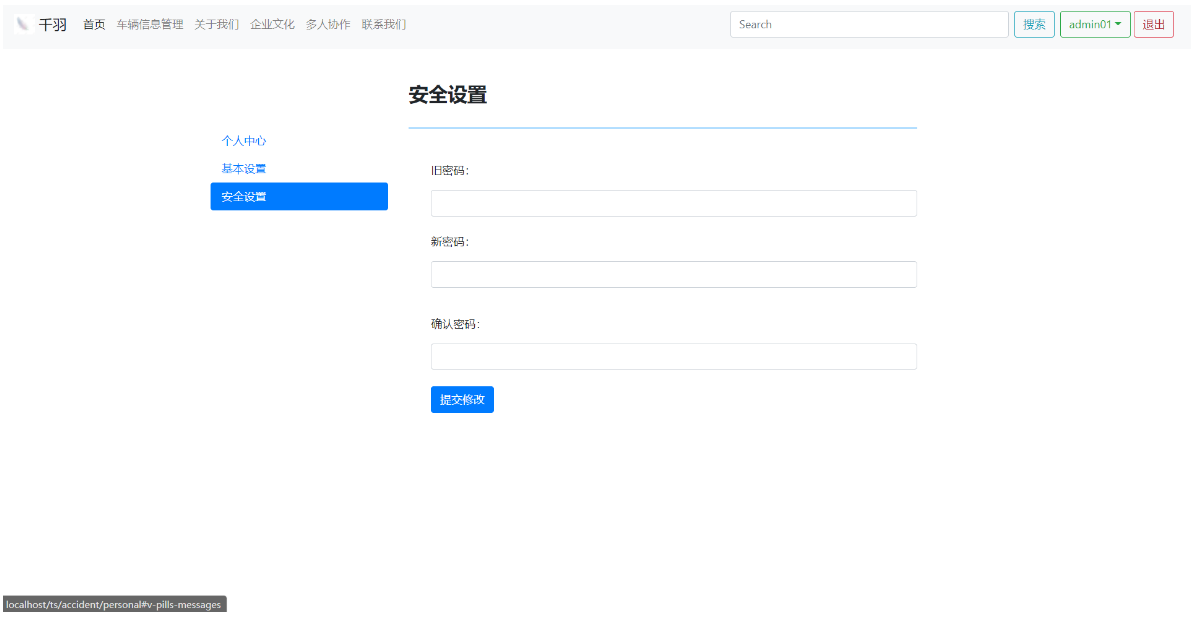
住址：

出生日期：年 /月/日

入职日期：

localhost/ts/accident/personal#v-pills-profile

修改密码



安全设置

旧密码：

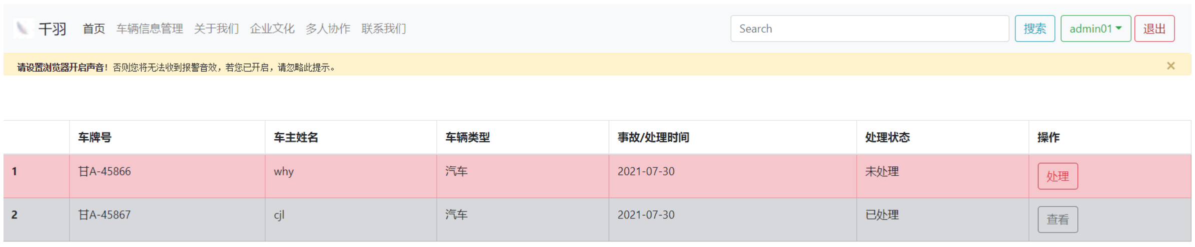
新密码：

确认密码：

localhost/ts/accident/personal#v-pills-messages

15.2 事故信息模块

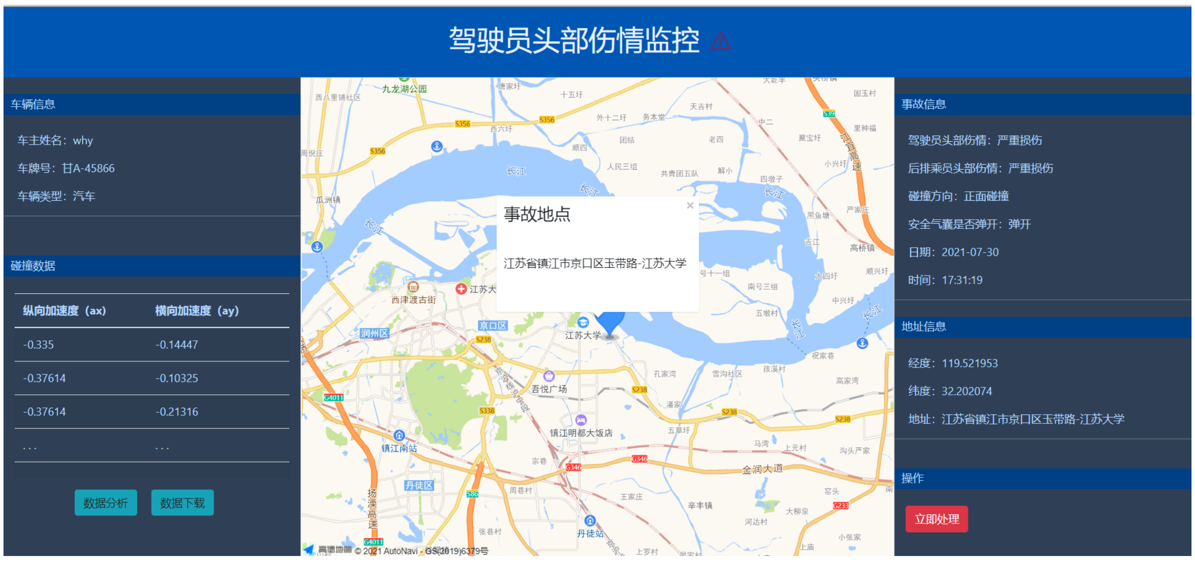
事故统计页面



请设置浏览器开启声音！否则您将无法收到报警音效，若您已开启，请忽略此提示。

	车牌号	车主姓名	车辆类型	事故/处理时间	处理状态	操作
1	甘A-45866	why	汽车	2021-07-30	未处理	处理
2	甘A-45867	cjl	汽车	2021-07-30	已处理	查看

碰撞数据页面



驾驶员头部伤情监控

车辆信息

车主姓名：why

车牌号：甘A-45866

车辆类型：汽车

碰撞数据

纵向加速度（ax）	横向加速度（ay）
-0.335	-0.14447
-0.37614	-0.10325
-0.37614	-0.21316
...	...

数据分析　数据下载

事故地点

江苏省镇江市京口区玉带路-江苏大学

事故信息

驾驶员头部伤情：严重损伤

后排乘员头部伤情：严重损伤

碰撞方向：正面碰撞

安全气囊是否弹开：弹开

日期：2021-07-30

时间：17:31:19

地址信息

经度：119.521953

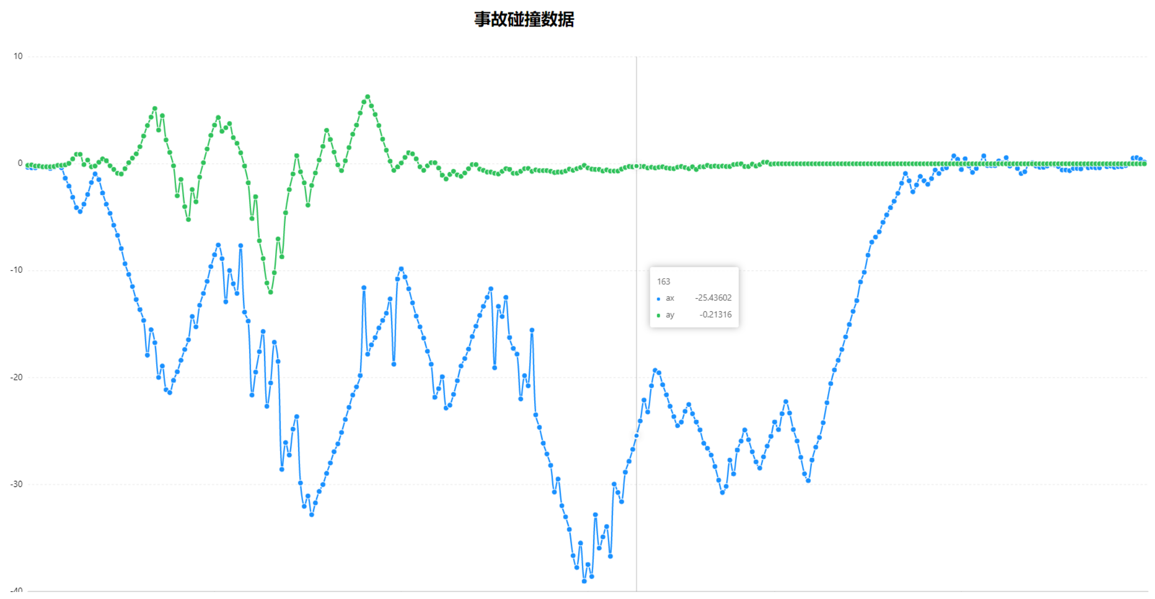
纬度：32.202074

地址：江苏省镇江市京口区玉带路-江苏大学

操作

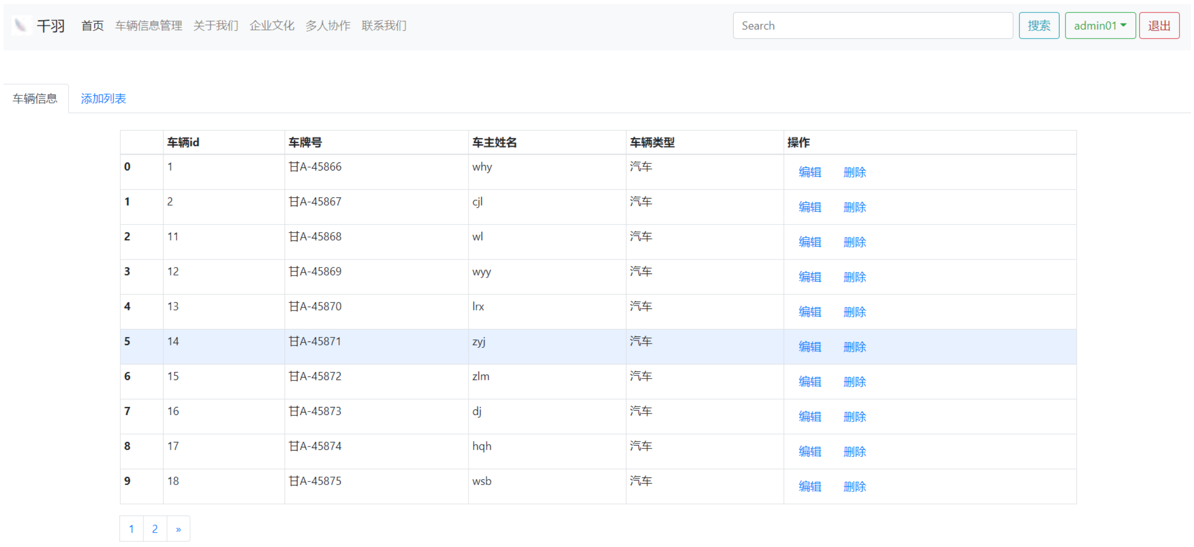
立即处理

数据分析页面



15.3 车辆信息模块

车辆信息展示页面



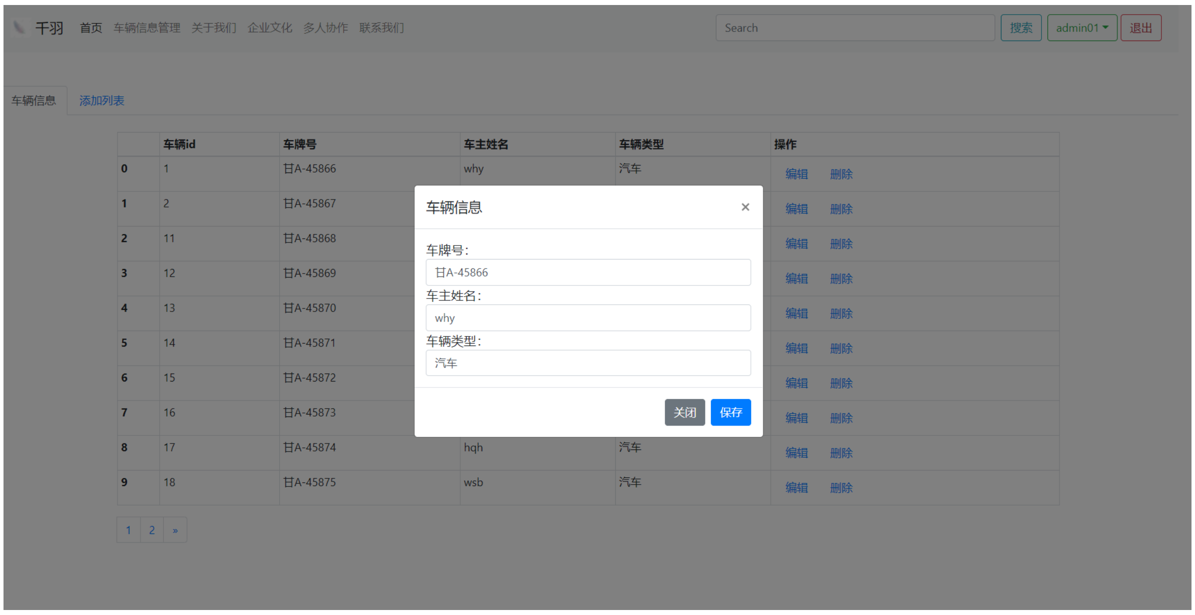
千羽　首页　车辆信息管理　关于我们　企业文化　多人协作　联系我们　Search　搜索　admin01　退出

车辆信息　添加列表

	车辆id	车牌号	车主姓名	车辆类型	操作
0	1	甘A-45866	why	汽车	编辑　删除
1	2	甘A-45867	cjl	汽车	编辑　删除
2	11	甘A-45868	wl	汽车	编辑　删除
3	12	甘A-45869	wyy	汽车	编辑　删除
4	13	甘A-45870	lrx	汽车	编辑　删除
5	14	甘A-45871	zyj	汽车	编辑　删除
6	15	甘A-45872	zlm	汽车	编辑　删除
7	16	甘A-45873	dj	汽车	编辑　删除
8	17	甘A-45874	hqh	汽车	编辑　删除
9	18	甘A-45875	wsb	汽车	编辑　删除

1　2　»

车辆信息编辑页面



千羽 首页 车辆信息管理 关于我们 企业文化 多人协作 联系我们 Search 搜索 admin01 退出

车辆信息 添加列表

	车辆id	车牌号	车主姓名	车辆类型	操作
0	1	甘A-45866	why	汽车	编辑 删除
1	2	甘A-45867			编辑 删除
2	11	甘A-45868			编辑 删除
3	12	甘A-45869			编辑 删除
4	13	甘A-45870			编辑 删除
5	14	甘A-45871			编辑 删除
6	15	甘A-45872			编辑 删除
7	16	甘A-45873			编辑 删除
8	17	甘A-45874	hqh	汽车	编辑 删除
9	18	甘A-45875	wsb	汽车	编辑 删除

1 2 »

车辆信息 ×

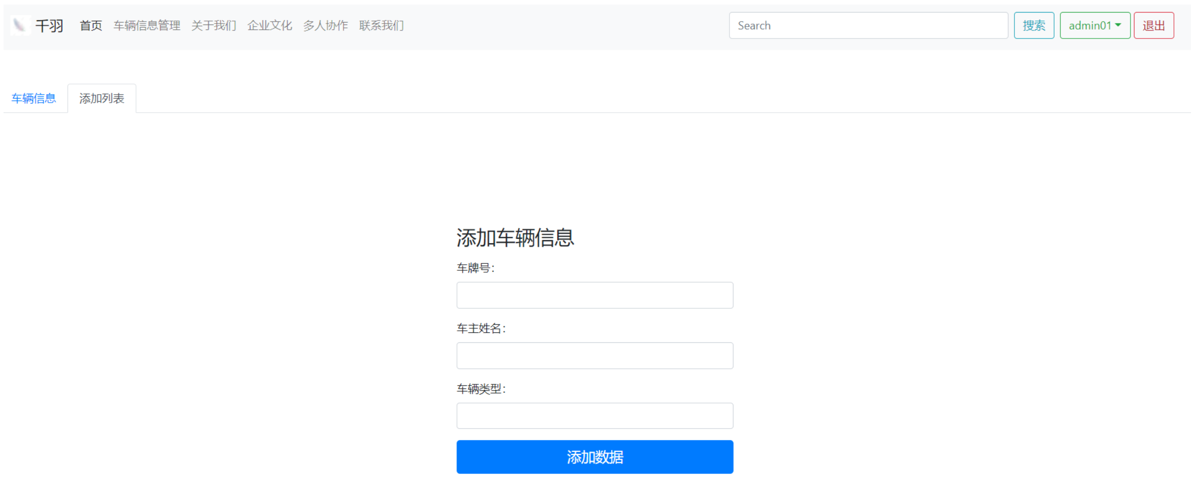
车牌号: 甘A-45866

车主姓名: why

车辆类型: 汽车

关闭 保存

车辆信息添加页面



千羽 首页 车辆信息管理 关于我们 企业文化 多人协作 联系我们 Search 搜索 admin01 退出

车辆信息 添加列表

添加车辆信息

车牌号:

车主姓名:

车辆类型:

添加数据