BaseContext.java

package com.zeb.water\_monitor.common;  
public class BaseContext {  
 private static final ThreadLocal<Integer> THREAD\_LOCAL = new ThreadLocal<>();  
 private BaseContext() {  
 }  
 public static void setCurrentId(Integer id) {  
 THREAD\_LOCAL.set(id);  
 }  
 public static Integer getCurrentId() {  
 return THREAD\_LOCAL.get();  
 }  
 public static void removeThreadLocal(){  
 THREAD\_LOCAL.remove();  
 }  
}

CustomException.java

package com.zeb.water\_monitor.common;  
public class CustomException extends RuntimeException {  
 public CustomException(String message){  
 super(message);  
 }  
}

GlobalExceptionHandler.java

package com.zeb.water\_monitor.common;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.context.support.DefaultMessageSourceResolvable;  
import org.springframework.http.converter.HttpMessageNotReadableException;  
import org.springframework.stereotype.Controller;  
import org.springframework.validation.BindException;  
import org.springframework.validation.FieldError;  
import org.springframework.web.bind.MethodArgumentNotValidException;  
import org.springframework.web.bind.annotation.ControllerAdvice;  
import org.springframework.web.bind.annotation.ExceptionHandler;  
import org.springframework.web.bind.annotation.ResponseBody;  
import org.springframework.web.bind.annotation.RestController;  
import javax.validation.ConstraintViolation;  
import javax.validation.ConstraintViolationException;  
import java.sql.SQLIntegrityConstraintViolationException;  
import java.util.List;  
import java.util.stream.Collectors;  
@ControllerAdvice(annotations = {RestController.class, Controller.class})  
@ResponseBody  
@Slf4j  
public class GlobalExceptionHandler {  
 @ExceptionHandler(SQLIntegrityConstraintViolationException.class)  
 public Result<String> exceptionHandler(SQLIntegrityConstraintViolationException ex) {  
 if (ex.getMessage().contains("Duplicate entry")) {  
 String[] split = ex.getMessage().split(" ");  
 String msg = split[2] + "已经存在";  
 return Result.error(msg);  
 } else if (ex.getMessage().contains("cannot be null")) {  
 String[] split = ex.getMessage().split(" ");  
 String msg = split[1] + "为空";  
 return Result.error(msg);  
 }  
 return Result.error("未知错误");  
 }  
 @ExceptionHandler(CustomException.class)  
 public Result<String> customException(CustomException customException) {  
 return Result.error(customException.getMessage());  
 }  
 @ExceptionHandler(MethodArgumentNotValidException.class)  
 public Result<String> MethodArgumentNotValidExceptionHandler(MethodArgumentNotValidException e) {  
 String message = e.getBindingResult().getAllErrors().stream().map(DefaultMessageSourceResolvable::getDefaultMessage).collect(Collectors.joining());  
 return Result.error(message);  
 }  
 @ExceptionHandler(BindException.class)  
 public Result<String> BindExceptionHandler(BindException e) {  
 List<FieldError> fieldErrors = e.getBindingResult().getFieldErrors();  
 List<String> msgList = fieldErrors.stream()  
 .map(o -> o.getDefaultMessage())  
 .collect(Collectors.toList());  
 log.error("Validation格式校验异常:-------------->{}", msgList);  
 return Result.error(msgList.toString());  
 }  
 @ExceptionHandler(ConstraintViolationException.class)  
 public Result<String> ConstraintViolationExceptionHandler(ConstraintViolationException e) {  
 String message = e.getConstraintViolations().stream().map(ConstraintViolation::getMessage).collect(Collectors.joining());  
 return Result.error(message);  
 }  
 @ExceptionHandler(HttpMessageNotReadableException.class)  
 public Result<String> HttpMessageNotReadableExceptionHandler(HttpMessageNotReadableException e) {  
 return Result.error("参数格式异常");  
 }  
}

JacksonObjectMapper.java

package com.zeb.water\_monitor.common;  
import com.fasterxml.jackson.databind.DeserializationFeature;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import com.fasterxml.jackson.databind.module.SimpleModule;  
import com.fasterxml.jackson.databind.ser.std.ToStringSerializer;  
import com.fasterxml.jackson.datatype.jsr310.deser.LocalDateDeserializer;  
import com.fasterxml.jackson.datatype.jsr310.deser.LocalDateTimeDeserializer;  
import com.fasterxml.jackson.datatype.jsr310.deser.LocalTimeDeserializer;  
import com.fasterxml.jackson.datatype.jsr310.ser.LocalDateSerializer;  
import com.fasterxml.jackson.datatype.jsr310.ser.LocalDateTimeSerializer;  
import com.fasterxml.jackson.datatype.jsr310.ser.LocalTimeSerializer;  
import java.math.BigInteger;  
import java.time.LocalDate;  
import java.time.LocalDateTime;  
import java.time.LocalTime;  
import java.time.format.DateTimeFormatter;  
import static com.fasterxml.jackson.databind.DeserializationFeature.FAIL\_ON\_UNKNOWN\_PROPERTIES;  
public class JacksonObjectMapper extends ObjectMapper {  
 public static final String DEFAULT\_DATE\_FORMAT = "yyyy-MM-dd";  
 public static final String DEFAULT\_DATE\_TIME\_FORMAT = "yyyy-MM-dd HH:mm:ss";  
 public static final String DEFAULT\_TIME\_FORMAT = "HH:mm:ss";  
 public JacksonObjectMapper() {  
 super();  
 this.configure(FAIL\_ON\_UNKNOWN\_PROPERTIES, false);  
 this.getDeserializationConfig().withoutFeatures(DeserializationFeature.FAIL\_ON\_UNKNOWN\_PROPERTIES);  
 SimpleModule simpleModule = new SimpleModule()  
 .addDeserializer(LocalDateTime.class, new LocalDateTimeDeserializer(DateTimeFormatter.ofPattern(DEFAULT\_DATE\_TIME\_FORMAT)))  
 .addDeserializer(LocalDate.class, new LocalDateDeserializer(DateTimeFormatter.ofPattern(DEFAULT\_DATE\_FORMAT)))  
 .addDeserializer(LocalTime.class, new LocalTimeDeserializer(DateTimeFormatter.ofPattern(DEFAULT\_TIME\_FORMAT)))  
 .addSerializer(BigInteger.class, ToStringSerializer.instance)  
 .addSerializer(LocalDateTime.class, new LocalDateTimeSerializer(DateTimeFormatter.ofPattern(DEFAULT\_DATE\_TIME\_FORMAT)))  
 .addSerializer(LocalDate.class, new LocalDateSerializer(DateTimeFormatter.ofPattern(DEFAULT\_DATE\_FORMAT)))  
 .addSerializer(LocalTime.class, new LocalTimeSerializer(DateTimeFormatter.ofPattern(DEFAULT\_TIME\_FORMAT)));  
 this.registerModule(simpleModule);  
 }  
}

Result.java

package com.zeb.water\_monitor.common;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
@Data  
public class Result<T> {  
 @ApiModelProperty("状态码")  
 private Integer code;  
 @ApiModelProperty("错误信息")  
 private String msg;  
 @ApiModelProperty("返回数据")  
 private T data;  
 public static <T> Result<T> success(T object) {  
 Result<T> r = new Result<T>();  
 r.data = object;  
 r.code = 200;  
 return r;  
 }  
 public static <T> Result<T> error(String msg) {  
 Result r = new Result();  
 r.msg = msg;  
 r.code = 500;  
 return r;  
 }  
 public static <T> Result<T> error(Integer code, String msg) {  
 Result r = new Result();  
 r.msg = msg;  
 r.code = code;  
 return r;  
 }  
}

CorsConfig.java

package com.zeb.water\_monitor.config;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.cors.CorsConfiguration;  
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;  
import org.springframework.web.filter.CorsFilter;  
@Configuration  
public class CorsConfig {  
 private static final long MAX\_AGE = 24 \* 60 \* 60;  
 @Bean  
 public CorsFilter corsFilter() {  
 UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();  
 CorsConfiguration corsConfiguration = new CorsConfiguration();  
 corsConfiguration.addAllowedOrigin("\*");  
 corsConfiguration.addAllowedHeader("\*");  
 corsConfiguration.addAllowedMethod("\*");  
 corsConfiguration.setMaxAge(MAX\_AGE);  
 source.registerCorsConfiguration("/\*\*", corsConfiguration);  
 return new CorsFilter(source);  
 }  
}

MybatisPlusConfig.java

package com.zeb.water\_monitor.config;  
import com.baomidou.mybatisplus.annotation.DbType;  
import com.baomidou.mybatisplus.extension.plugins.MybatisPlusInterceptor;  
import com.baomidou.mybatisplus.extension.plugins.inner.PaginationInnerInterceptor;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
@Configuration  
public class MybatisPlusConfig {  
 @Bean  
 public MybatisPlusInterceptor mybatisPlusInterceptor() {  
 MybatisPlusInterceptor interceptor = new MybatisPlusInterceptor();  
 interceptor.addInnerInterceptor(new PaginationInnerInterceptor(DbType.MYSQL));  
 return interceptor;  
 }  
}

MyWebConfig.java

package com.zeb.water\_monitor.config;  
import com.zeb.water\_monitor.common.JacksonObjectMapper;  
import com.zeb.water\_monitor.interceptor.LoginInterceptor;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.http.converter.HttpMessageConverter;  
import org.springframework.http.converter.json.MappingJackson2HttpMessageConverter;  
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
import springfox.documentation.spring.web.plugins.Docket;  
import java.util.List;  
@Configuration  
public abstract class MyWebConfig implements WebMvcConfigurer {  
 @Override  
 public void addInterceptors(InterceptorRegistry registry) {  
 LoginInterceptor loginInterceptor = new LoginInterceptor();  
 registry.addInterceptor(loginInterceptor)  
 .addPathPatterns("/\*\*")  
 .excludePathPatterns(  
 "/user/login",  
 "/user/register",  
 "/water/quality/query",  
 "/water/quality/all",  
 "/water/quality/station",  
 "/water/quality/recent",  
 "/csrf",  
 "/img/\*\*",  
 "/error",  
 "/common/\*\*",  
 "/doc.html",  
 "/webjars/\*\*",  
 "/swagger-resources/\*\*",  
 "/swagger-ui.html/\*\*",  
 "/favicon.ico",  
 "/v2/api-docs");  
 }  
 @Override  
 public void extendMessageConverters(List<HttpMessageConverter<?>> converters) {  
 MappingJackson2HttpMessageConverter messageConverter = new MappingJackson2HttpMessageConverter();  
 messageConverter.setObjectMapper(new JacksonObjectMapper());  
 converters.add(0, messageConverter);  
 }  
 @Bean  
 public abstract Docket createRestApi();  
}

SwaggerConfig.java

package com.zeb.water\_monitor.config;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;  
import springfox.documentation.builders.ApiInfoBuilder;  
import springfox.documentation.builders.PathSelectors;  
import springfox.documentation.builders.RequestHandlerSelectors;  
import springfox.documentation.service.ApiInfo;  
import springfox.documentation.spi.DocumentationType;  
import springfox.documentation.spring.web.plugins.Docket;  
import springfox.documentation.swagger2.annotations.EnableSwagger2;  
@Configuration  
@EnableSwagger2  
public class SwaggerConfig extends MyWebConfig {  
 @Value("${spring.swagger2.enabled}")  
 private Boolean enabled;  
 @Override  
 @Bean  
 public Docket createRestApi() {  
 return new Docket(DocumentationType.SWAGGER\_2)  
 .apiInfo(apiInfo())  
 .enable(enabled)  
 .select()  
 .apis(RequestHandlerSelectors.basePackage("com.zeb.water\_monitor.controller"))  
 .paths(PathSelectors.any())  
 .build();  
 }  
 private ApiInfo apiInfo() {  
 return new ApiInfoBuilder()  
 .title("水质质量监测系统")  
 .version("1.0")  
 .description("新增了本月不达标区域、获取所有监测区域接口；修改了分页查询水质数据接口（可以根据区域查询）、获取水质示意图接口（根据区域、时间、水质参数获取）、获取当日水质数据接口（可以选择区域）；新增area字段，注意：station（string）字段为监测船，经纬度字段还未添加")  
 .build();  
 }  
 @Override  
 public void addResourceHandlers(ResourceHandlerRegistry registry) {  
 registry.addResourceHandler("/\*\*").addResourceLocations("classpath:/static/");  
 registry.addResourceHandler("/swagger-ui.html").addResourceLocations("classpath:/META-INF/resources/");  
 registry.addResourceHandler("/webjars/\*\*").addResourceLocations("classpath:/META-INF/resources/webjars/");  
 }  
}

AreaController.java

package com.zeb.water\_monitor.controller;  
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;  
import com.zeb.water\_monitor.common.BaseContext;  
import com.zeb.water\_monitor.common.CustomException;  
import com.zeb.water\_monitor.common.Result;  
import com.zeb.water\_monitor.entity.Area;  
import com.zeb.water\_monitor.entity.User;  
import com.zeb.water\_monitor.entity.WaterQuality;  
import com.zeb.water\_monitor.enums.RoleEnum;  
import com.zeb.water\_monitor.service.AreaService;  
import com.zeb.water\_monitor.service.UserService;  
import com.zeb.water\_monitor.service.WaterQualityService;  
import io.swagger.annotations.ApiOperation;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.transaction.annotation.Transactional;  
import org.springframework.web.bind.annotation.\*;  
import java.util.List;  
import java.util.stream.Collectors;  
@Slf4j  
@RestController  
@RequestMapping("/area")  
public class AreaController {  
 @Autowired  
 private AreaService areaService;  
 @Autowired  
 private UserService userService;  
 @Autowired  
 private WaterQualityService waterQualityService;  
 @ApiOperation(value = "查询所有区域")  
 @GetMapping("/list")  
 public Result<List> getAllAreas() {  
 List<Area> list = areaService.list();  
 return Result.success(list);  
 }  
 @ApiOperation(value = "添加区域", notes = "只需传输区域名称参数name即可")  
 @PostMapping("/add")  
 public Result<String> addArea(@RequestBody Area area) {  
 areaService.save(area);  
 return Result.success("添加成功");  
 }  
 @ApiOperation(value = "删除区域", notes = "根据id进行删除")  
 @DeleteMapping("/delete/{id}")  
 public Result<String> deleteArea(@PathVariable("id") Integer id) {  
 Area area = areaService.getById(id);  
 if (area == null) {  
 throw new CustomException("无此id数据");  
 }  
 LambdaQueryWrapper<WaterQuality> wrapper = new LambdaQueryWrapper<>();  
 wrapper.eq(area.getName() != null, WaterQuality::getArea, area.getName());  
 List<WaterQuality> list = waterQualityService.list(wrapper);  
 log.info("list:" + list);  
 if (list != null && !list.isEmpty()) {  
 throw new CustomException("删除失败，请先删除或修改水质数据表中区域名为" + area.getName() + "的字段");  
 }  
 areaService.removeById(id);  
 return Result.success("删除成功");  
 }  
 @ApiOperation(value = "修改区域", notes = "")  
 @PutMapping("/edit")  
 @Transactional(rollbackFor = Exception.class)  
 public Result<String> editArea(@RequestBody Area area) {  
 Integer userId = BaseContext.getCurrentId();  
 User user = userService.getById(userId);  
 if (user.getRole() == RoleEnum.USER) {  
 return Result.error("用户权限不足");  
 }  
 Area byId = areaService.getById(area.getId());  
 if (byId == null) {  
 throw new CustomException("无此id数据");  
 }  
 areaService.updateById(area);  
 LambdaQueryWrapper<WaterQuality> wrapper = new LambdaQueryWrapper<>();  
 wrapper.eq(WaterQuality::getArea, byId.getName());  
 List<WaterQuality> list = waterQualityService.list(wrapper);  
 log.info("list:" + list);  
 list.stream().map(item -> {  
 item.setArea(area.getName());  
 waterQualityService.updateById(item);  
 return item;  
 }).collect(Collectors.toList());  
 return Result.success("修改成功");  
 }  
}

StationController.java

package com.zeb.water\_monitor.controller;  
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;  
import com.zeb.water\_monitor.common.BaseContext;  
import com.zeb.water\_monitor.common.CustomException;  
import com.zeb.water\_monitor.common.Result;  
import com.zeb.water\_monitor.entity.Station;  
import com.zeb.water\_monitor.entity.User;  
import com.zeb.water\_monitor.entity.WaterQuality;  
import com.zeb.water\_monitor.enums.RoleEnum;  
import com.zeb.water\_monitor.service.StationService;  
import com.zeb.water\_monitor.service.UserService;  
import com.zeb.water\_monitor.service.WaterQualityService;  
import io.swagger.annotations.ApiOperation;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.transaction.annotation.Transactional;  
import org.springframework.web.bind.annotation.\*;  
import java.util.List;  
import java.util.stream.Collectors;  
@RestController  
@RequestMapping("/station")  
public class StationController {  
 @Autowired  
 private StationService stationService;  
 @Autowired  
 private WaterQualityService waterQualityService;  
 @Autowired  
 private UserService userService;  
 @ApiOperation(value = "查询指定区域监测船", notes = "注意：获取监测船的接口之前写过，但此接口可以返回监测船的id，用于修改和删除操作；根据监测区域获取区域下所有监测船；area为监测区域，不传值代表查询所有区域")  
 @GetMapping("/list")  
 public Result<List> getAllStations(String area) {  
 LambdaQueryWrapper<Station> wrapper = new LambdaQueryWrapper<>();  
 wrapper.eq(area != null && !"".equals(area), Station::getArea, area);  
 List<Station> list = stationService.list(wrapper);  
 return Result.success(list);  
 }  
 @ApiOperation(value = "添加监测船", notes = "无需传输id")  
 @PostMapping("/add")  
 public Result<String> addStation(@RequestBody Station station) {  
 stationService.save(station);  
 return Result.success("添加成功");  
 }  
 @ApiOperation(value = "删除监测船", notes = "根据监测船id进行删除")  
 @DeleteMapping("/delete/{id}")  
 public Result<String> deleteArea(@PathVariable("id") Integer id) {  
 Station station = stationService.getById(id);  
 if (station == null) {  
 throw new CustomException("无此id数据");  
 }  
 LambdaQueryWrapper<WaterQuality> wrapper = new LambdaQueryWrapper<>();  
 wrapper.eq(station.getName() != null, WaterQuality::getStation, station.getName());  
 List<WaterQuality> list = waterQualityService.list(wrapper);  
 if (list != null && !list.isEmpty()) {  
 throw new CustomException("删除失败，请先删除或修改水质数据表中监测船为" + station.getName() + "的字段");  
 }  
 stationService.removeById(id);  
 return Result.success("删除成功");  
 }  
 @ApiOperation(value = "修改监测船", notes = "根据id修改监测船名称、所属的监测区域")  
 @PutMapping("/edit")  
 @Transactional(rollbackFor = Exception.class)  
 public Result<String> editArea(@RequestBody Station station) {  
 Integer userId = BaseContext.getCurrentId();  
 User user = userService.getById(userId);  
 if (user.getRole() == RoleEnum.USER) {  
 return Result.error("用户权限不足");  
 }  
 Station byId = stationService.getById(station.getId());  
 if (byId == null) {  
 throw new CustomException("无此id数据");  
 }  
 stationService.updateById(station);  
 LambdaQueryWrapper<WaterQuality> wrapper = new LambdaQueryWrapper<>();  
 wrapper.eq(station.getName() != null, WaterQuality::getArea, byId.getName());  
 List<WaterQuality> list = waterQualityService.list(wrapper);  
 list.stream().map(item -> {  
 item.setArea(station.getArea());  
 item.setStation(station.getName());  
 waterQualityService.updateById(item);  
 return item;  
 }).collect(Collectors.toList());  
 return Result.success("修改成功");  
 }  
}

UserController.java

package com.zeb.water\_monitor.controller;  
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;  
import com.zeb.water\_monitor.common.BaseContext;  
import com.zeb.water\_monitor.common.CustomException;  
import com.zeb.water\_monitor.common.Result;  
import com.zeb.water\_monitor.dto.UserLoginDTO;  
import com.zeb.water\_monitor.entity.User;  
import com.zeb.water\_monitor.enums.RoleEnum;  
import com.zeb.water\_monitor.service.UserService;  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiOperation;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.util.DigestUtils;  
import org.springframework.web.bind.annotation.\*;  
import javax.servlet.http.HttpServletRequest;  
import java.util.List;  
@RestController  
@RequestMapping("/user")  
@Slf4j  
@Api  
public class UserController {  
 @Autowired  
 private UserService userService;  
 @ApiOperation(value = "登录", notes = "登录功能，管理员账户及密码：")  
 @PostMapping("/login")  
 public Result<User> login(@RequestBody UserLoginDTO userLoginDTO, HttpServletRequest request) {  
 log.info("Login");  
 LambdaQueryWrapper<User> wrapper = new LambdaQueryWrapper<>();  
 wrapper.eq(User::getUsername, userLoginDTO.getUsername())  
 .last("limit 1");  
 User userInfo = userService.getOne(wrapper);  
 if (userInfo == null) {  
 throw new CustomException("用户不存在");  
 }  
 String password = DigestUtils.md5DigestAsHex(userLoginDTO.getPassword().getBytes());  
 if (!userInfo.getPassword().equals(password)) {  
 throw new CustomException("密码错误");  
 }  
 request.getSession().getServletContext().setAttribute("user", userInfo.getId());  
 log.info("登录成功:" + userInfo.getId());  
 return Result.success(userInfo);  
 }  
 @ApiOperation(value = "退出登录", notes = "退出登录")  
 @GetMapping("/logout")  
 public Result<String> loginOut(HttpServletRequest request) {  
 request.getSession().removeAttribute("user");  
 BaseContext.removeThreadLocal();  
 return Result.success("退出成功");  
 }  
 @ApiOperation(value = "注册用户", notes = "注册用户")  
 @PostMapping("/register")  
 public Result<String> register(String username, String password) {  
 log.info("user: " + username + password);  
 userService.register(new User(username, password));  
 return Result.success("注册用户成功");  
 }  
 @ApiOperation(value = "根据名称查找用户", notes = "根据名称查找用户")  
 @GetMapping("/query")  
 public Result<List<User>> getQueriedUsers(@RequestParam("name") String name) {  
 LambdaQueryWrapper<User> lambdaQueryWrapper = new LambdaQueryWrapper<>();  
 lambdaQueryWrapper.like(User::getUsername, name);  
 lambdaQueryWrapper.orderByAsc(User::getId);  
 List<User> users = userService.list(lambdaQueryWrapper);  
 if (users == null) {  
 throw new CustomException("获取用户失败，请联系管理员");  
 }  
 return Result.success(users);  
 }  
}

WaterQualityController.java

package com.zeb.water\_monitor.controller;  
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;  
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.zeb.water\_monitor.\*;  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiOperation;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.beans.BeanUtils;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.format.annotation.DateTimeFormat;  
import org.springframework.web.bind.annotation.\*;  
import java.math.BigDecimal;  
import java.math.RoundingMode;  
import java.time.LocalDate;  
import java.time.LocalDateTime;  
import java.util.\*;  
import java.util.stream.Collectors;  
@Slf4j  
@RestController  
@Api  
@RequestMapping("/water/quality")  
public class WaterQualityController {  
 @Autowired  
 private WaterQualityService waterQualityService;  
 @Autowired  
 private UserService userService;  
 @GetMapping("/query")  
 public Result<WaterQualityPageVO> getQueriedWaterQualities(WaterQualityQuery waterQualityQuery) {  
 log.info("水质查询条件:" + waterQualityQuery);  
 String station = waterQualityQuery.getStation();  
 LocalDate startDate = waterQualityQuery.getStartDate();  
 LocalDate endDate = waterQualityQuery.getEndDate();  
 String area = waterQualityQuery.getArea();  
 LambdaQueryWrapper<WaterQuality> wrapper = new LambdaQueryWrapper<>();  
 wrapper.eq(station != null, WaterQuality::getStation, station);  
 wrapper.ge(startDate != null, WaterQuality::getDate, startDate + " 00:00:00");  
 wrapper.le(endDate != null, WaterQuality::getDate, endDate + " 23:59:59");  
 wrapper.eq(area != null, WaterQuality::getArea, area);  
 wrapper.orderByDesc(WaterQuality::getDate);  
 Page<WaterQuality> pageInfo = new Page<>(waterQualityQuery.getPageNum(), waterQualityQuery.getPageSize());  
 Page<WaterQuality> page = waterQualityService.page(pageInfo, wrapper);  
 log.info("page:" + page);  
 WaterQualityPageVO waterQualityPageVO = new WaterQualityPageVO();  
 waterQualityPageVO.setRecords(page.getRecords());  
 waterQualityPageVO.setTotal(page.getTotal());  
 return Result.success(waterQualityPageVO);  
 }  
 @GetMapping("/areas")  
 public Result<List<String>> getAllAreas() {  
 QueryWrapper<WaterQuality> wrapper = new QueryWrapper<>();  
 wrapper.select("distinct area");  
 List<WaterQuality> list = waterQualityService.list(wrapper);  
 List<String> areas = list.stream().map(WaterQuality::getArea).collect(Collectors.toList());  
 return Result.success(areas);  
 }  
 @GetMapping("/all")  
 public Result<List<WaterQuality>> getAllWaterQualities() {  
 List<WaterQuality> waterQualities = waterQualityService.list();  
 return Result.success(waterQualities);  
 }  
  
 @GetMapping("/day")  
 public Result<List<WaterQuality>> getWaterQualitiesByDate(@DateTimeFormat(pattern = "yyyy-MM-dd") @RequestParam("day") LocalDate day, String area) {  
 log.info("day:" + day);  
 LambdaQueryWrapper<WaterQuality> wrapper = new LambdaQueryWrapper<>();  
 wrapper.ge(day != null, WaterQuality::getDate, day + " 00:00:00");  
 wrapper.le(day != null, WaterQuality::getDate, day + " 23:59:59");  
 wrapper.eq(area != null && !"".equals(area), WaterQuality::getArea, area);  
 List<WaterQuality> list = waterQualityService.list(wrapper);  
 log.info("list:" + list);  
 return Result.success(list);  
 }  
  
 @PutMapping("/edit")  
 public Result<String> editWaterQuality(@RequestBody WaterQuality waterQuality) {  
 Integer userId = BaseContext.getCurrentId();  
 log.info("userId:" + userId);  
 User user = userService.getById(userId);  
 if (user == null) {  
 throw new CustomException("用户未登录");  
 }  
 if (user.getRole() == RoleEnum.USER) {  
 return Result.error("用户权限不足");  
 }  
 if (waterQualityService.getById(waterQuality.getId()) == null) {  
 throw new CustomException("id错误");  
 }  
 if (waterQuality.getDate() != null) {  
 waterQuality.setDate(waterQuality.getDate());  
 }  
 if (!waterQualityService.updateById(waterQuality)) {  
 return Result.error("修改失败，请联系管理员");  
 }  
 return Result.success("修改成功");  
 }  
 @DeleteMapping("/delete/{id}")  
 public Result<String> deleteWaterQuality(@PathVariable("id") Integer id) {  
 Integer userId = BaseContext.getCurrentId();  
 User user = userService.getById(userId);  
 if (user == null) {  
 throw new CustomException("用户未登录");  
 }  
 if (user.getRole() == RoleEnum.USER) {  
 return Result.error("用户权限不足");  
 }  
 if (waterQualityService.getById(id) == null) {  
 throw new CustomException("无此水质数据");  
 }  
 if (!waterQualityService.removeById(id)) {  
 return Result.error("删除失败，请联系管理员");  
 }  
 return Result.success("删除成功");  
 }  
 @GetMapping("/station")  
 public Result<List<String>> getAllStations(String area) {  
 QueryWrapper<WaterQuality> wrapper = new QueryWrapper<>();  
 wrapper.select("DISTINCT station").lambda().eq(area != null, WaterQuality::getArea, area).orderByAsc(WaterQuality::getStation);  
 List<WaterQuality> list = waterQualityService.list(wrapper);  
 List<String> stations = list.stream().map(WaterQuality::getStation).collect(Collectors.toList());  
 return Result.success(stations);  
 }  
 @GetMapping("/plot")  
 @ApiOperation(value = "获取数据示意图", notes = "根据监测区域、时期、水质参数获取示意图，period的单位为月，period=2表示最近2个月；indicator水质参数，ph、temperature、tds、conductivity、turbidity")  
 public Result<PlotVO> getDataForPlot(@RequestParam(value = "area") String area,  
 @RequestParam(value = "period") Integer period,  
 @RequestParam(value = "indicator") String indicator) {  
 PlotVO plotVO = waterQualityService.getDataForPlot(area, period, indicator);  
 return Result.success(plotVO);  
 }  
 @GetMapping("/recent")  
 @ApiOperation(value = "查看最近数据", notes = "查看最近num条数据，num不可为空")  
 public Result<List<WaterQuality>> getRecentWaterQualities(@RequestParam(value = "num") Integer num) {  
 LambdaQueryWrapper<WaterQuality> wrapper = new LambdaQueryWrapper<>();  
 wrapper.orderByDesc(WaterQuality::getDate);  
 wrapper.last("limit " + num);  
 List<WaterQuality> list = waterQualityService.list(wrapper);  
 return Result.success(list);  
 }  
 @PostMapping("/add")  
 public Result<String> addWaterQuality(@RequestBody AddWaterQualityDTO addWaterQualityDTO) {  
 WaterQuality waterQuality = new WaterQuality();  
 BeanUtils.copyProperties(addWaterQualityDTO, waterQuality);  
 waterQuality.setDate(LocalDateTime.now());  
 if (!waterQualityService.save(waterQuality)) {  
 throw new CustomException("添加失败");  
 }  
 return Result.success("添加成功");  
 }  
 @GetMapping("/indicator")  
 public Result<List<String>> getAllIndicators() {  
 ArrayList<String> indicators = new ArrayList<>();  
 indicators.add("ph");  
 indicators.add("temperature");  
 indicators.add("conductivity");  
 indicators.add("turbidity");  
 indicators.add("tds");  
 return Result.success(indicators);  
 }  
 @GetMapping("/substandard")  
 public Result<SubstandardAreas> getSubstandardAreas() {  
 LocalDateTime now = LocalDateTime.now();  
 LocalDateTime start = now.minusDays(now.getDayOfMonth() + 1);  
 log.info("start:" + start);  
 LambdaQueryWrapper<WaterQuality> wrapper = new LambdaQueryWrapper<>();  
 wrapper.ge(WaterQuality::getDate, start).le(WaterQuality::getDate, now).orderByAsc(WaterQuality::getDate);  
 wrapper.lt(WaterQuality::getPh, 6)  
 .or().gt(WaterQuality::getPh, 9)  
 .or().lt(WaterQuality::getConductivity, 0.05)  
 .or().gt(WaterQuality::getConductivity, 3)  
 .or().lt(WaterQuality::getTurbidity, 1)  
 .or().gt(WaterQuality::getTurbidity, 50);  
 List<WaterQuality> list = waterQualityService.list(wrapper);  
 log.info("list:" + list);  
 Map<String, Integer> map = new HashMap<>(16);  
 for (WaterQuality waterQuality : list) {  
 String area = waterQuality.getArea();  
 map.put(area, map.getOrDefault(area, 0) + 1);  
 }  
 log.info("map:" + map);  
 Set<String> areas = map.keySet();  
 Collection<Integer> days = map.values();  
 log.info("areas:" + areas);  
 log.info("days:" + days);  
 SubstandardAreas substandardAreas = new SubstandardAreas();  
 substandardAreas.setAreas(areas);  
 substandardAreas.setDays(days);  
 return Result.success(substandardAreas);  
 }

AddWaterQualityDTO.java

package com.zeb.water\_monitor.dto;  
import com.baomidou.mybatisplus.annotation.TableField;  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
import java.io.Serializable;  
@Data  
public class AddWaterQualityDTO implements Serializable {  
 private static final long serialVersionUID = 1L;  
 @TableField("PH")  
 @ApiModelProperty("PH值")  
 private Float ph;  
 @TableField("temperature")  
 @ApiModelProperty("温度")  
 private Float temperature;  
 @TableField("conductivity")  
 @ApiModelProperty("电导率")  
 private Float conductivity;  
 @TableField("turbidity")  
 @ApiModelProperty("浊度")  
 private Float turbidity;  
 @TableField("tds")  
 @ApiModelProperty("溶氧")  
 private Float tds;  
 @TableField("station")  
 @ApiModelProperty("监测船")  
 private String station;  
 @TableField("area")  
 @ApiModelProperty("监测区域")  
 private String area;  
}

MqttDataDTO.java

package com.zeb.water\_monitor.dto;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
@Data  
public class MqttDataDTO {  
 @ApiModelProperty("ph值")  
 private Float PH;  
 @ApiModelProperty("温度")  
 private Float temper;  
 @ApiModelProperty("电导率")  
 private Float EC;  
 @ApiModelProperty("浊度")  
 private Float TU;  
}

PageInfo.java

package com.zeb.water\_monitor.dto;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
import javax.validation.constraints.NotNull;  
@Data  
public class PageInfo {  
 @NotNull  
 @ApiModelProperty("页码")  
 private Integer pageNum;  
 @NotNull  
 @ApiModelProperty("每页显示条数")  
 private Integer pageSize;  
}

UserLoginDTO.java

package com.zeb.water\_monitor.dto;  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
import javax.validation.constraints.NotBlank;  
@Data  
@ApiModel("登录用户信息")  
public class UserLoginDTO {  
 @NotBlank(message = "用户名不能为空")  
 @ApiModelProperty("用户名")  
 private String username;  
 @NotBlank(message = "密码不能为空")  
 @ApiModelProperty("密码")  
 private String password;  
}

WaterQualityQuery.java

package com.zeb.water\_monitor.dto;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
import org.springframework.format.annotation.DateTimeFormat;  
import java.time.LocalDate;  
@Data  
public class WaterQualityQuery extends PageInfo {  
 @ApiModelProperty("监测船")  
 private String station;  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 @ApiModelProperty("起始时间")  
 private LocalDate startDate;  
 @ApiModelProperty("结束时间")  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 private LocalDate endDate;  
 @ApiModelProperty("监测区域")  
 private String area;  
}

Area.java

package com.zeb.water\_monitor.entity;  
import com.baomidou.mybatisplus.annotation.IdType;  
import com.baomidou.mybatisplus.annotation.TableField;  
import com.baomidou.mybatisplus.annotation.TableId;  
import com.baomidou.mybatisplus.annotation.TableName;  
import java.io.Serializable;  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Getter;  
import lombok.Setter;  
@Getter  
@Setter  
@TableName("area")  
@ApiModel(value = "Area对象", description = "")  
public class Area implements Serializable {  
 private static final long serialVersionUID = 1L;  
 @ApiModelProperty("主键id")  
 @TableId(value = "id", type = IdType.AUTO)  
 private Integer id;  
 @ApiModelProperty("流域名")  
 @TableField("name")  
 private String name;  
}

Station.java

package com.zeb.water\_monitor.entity;  
import com.baomidou.mybatisplus.annotation.IdType;  
import com.baomidou.mybatisplus.annotation.TableField;  
import com.baomidou.mybatisplus.annotation.TableId;  
import com.baomidou.mybatisplus.annotation.TableName;  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Getter;  
import lombok.Setter;  
import java.io.Serializable;  
@Getter  
@Setter  
@TableName("station")  
@ApiModel(value = "Station对象", description = "")  
public class Station implements Serializable {  
 private static final long serialVersionUID = 1L;  
 @ApiModelProperty("主键id")  
 @TableId(value = "id", type = IdType.AUTO)  
 private Integer id;  
 @ApiModelProperty("监测船名称")  
 @TableField("name")  
 private String name;  
 @ApiModelProperty("监测区域")  
 @TableField("area")  
 private String area;  
}

User.java

package com.zeb.water\_monitor.entity;  
import com.baomidou.mybatisplus.annotation.\*;  
import com.fasterxml.jackson.annotation.JsonIgnore;  
import com.zeb.water\_monitor.enums.RoleEnum;  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
import java.io.Serializable;  
@Data  
@TableName("user")  
public class User implements Serializable {  
 private static final long serialVersionUID = 1L;  
 @TableId(value = "id", type = IdType.AUTO)  
 @ApiModelProperty("用户id")  
 private Integer id;  
 @TableField("username")  
 @ApiModelProperty("用户名")  
 private String username;  
 @TableField("password")  
 @ApiModelProperty("密码")  
 @JsonIgnore  
 private String password;  
 @ApiModelProperty("角色")  
 private RoleEnum role;  
 public User(String username, String password) {  
 this.username = username;  
 this.password = password;  
 }  
}

WaterQuality.java

package com.zeb.water\_monitor.entity;  
import com.baomidou.mybatisplus.annotation.\*;  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
import org.springframework.format.annotation.DateTimeFormat;  
import java.io.Serializable;  
import java.time.LocalDateTime;  
@Data  
@TableName("water\_quality")  
public class WaterQuality implements Serializable {  
 private static final long serialVersionUID = 1L;  
 @TableId(value = "id", type = IdType.AUTO)  
 @ApiModelProperty("id")  
 private Integer id;  
 @TableField("ph")  
 @ApiModelProperty("ph值")  
 private Float ph;  
 @TableField("temperature")  
 @ApiModelProperty("温度")  
 private Float temperature;  
 @TableField("conductivity")  
 @ApiModelProperty("电导率")  
 private Float conductivity;  
 @TableField("turbidity")  
 @ApiModelProperty("浊度")  
 private Float turbidity;  
 @TableField("tds")  
 @ApiModelProperty("溶氧")  
 private Float tds;  
 @TableField("date")  
 @ApiModelProperty("日期")  
 @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss")  
 private LocalDateTime date;  
 @TableField("station")  
 @ApiModelProperty("监测船")  
 private String station;  
 @TableField("area")  
 @ApiModelProperty("监测区域")  
 private String area;  
}

RoleEnum.java

package com.zeb.water\_monitor.enums;  
import com.baomidou.mybatisplus.annotation.EnumValue;  
import com.fasterxml.jackson.annotation.JsonValue;  
import lombok.Getter;  
@Getter  
public enum RoleEnum {  
 ADMIN(1, "admin"),  
 USER(2, "user");  
 @EnumValue  
 private Integer roleId;  
 @JsonValue  
 private String roleName;  
 RoleEnum(Integer id, String name) {  
 this.roleId = id;  
 this.roleName = name;  
 }  
}

LoginInterceptor.java

package com.zeb.water\_monitor.interceptor;  
import com.alibaba.fastjson.JSON;  
import com.zeb.water\_monitor.common.BaseContext;  
import com.zeb.water\_monitor.common.Result;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.stereotype.Component;  
import org.springframework.web.servlet.\*;  
import javax.servlet.http.\*;  
@Slf4j  
@Component  
public class LoginInterceptor implements HandlerInterceptor {  
 @Override  
 public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {  
 log.info("------------- LoginInterceptor 开始 -------------");  
 long startTime = System.currentTimeMillis();  
 request.setAttribute("requestStartTime", startTime);  
 if ("OPTIONS".equals(request.getMethod().toUpperCase())) {  
 return true;  
 }  
 String path = request.getRequestURL().toString();  
 log.info("接口登录拦截：，path：{}", path);  
 Integer user = (Integer) request.getSession().getServletContext().getAttribute("user");  
 if (user != null) {  
 log.info("用户线程id: {}", Thread.currentThread().getId());  
 BaseContext.setCurrentId(user);  
 return true;  
 }  
 log.info("用户未登录");  
 response.getWriter().write(JSON.toJSONString(Result.error("not login")));  
 return false;  
 }  
 @Override  
 public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) throws Exception {  
 long startTime = (Long) request.getAttribute("requestStartTime");  
 log.info("------------- LoginInterceptor 结束 耗时：{} ms -------------", System.currentTimeMillis() - startTime);  
 }  
}

WaterQualityMapper.java

package com.zeb.water\_monitor.mapper;  
import com.baomidou.mybatisplus.core.mapper.BaseMapper;  
import com.zeb.water\_monitor.entity.WaterQuality;  
import org.apache.ibatis.annotations.Mapper;  
@Mapper  
public interface WaterQualityMapper extends BaseMapper<WaterQuality> {  
}

SubScribeSample.java

package com.zeb.water\_monitor.mqtt;  
import com.alibaba.fastjson.JSON;  
import com.zeb.water\_monitor.dto.MqttDataDTO;  
import com.zeb.water\_monitor.entity.WaterQuality;  
import com.zeb.water\_monitor.service.WaterQualityService;  
import com.zeb.water\_monitor.utils.RandomNum;  
import lombok.extern.slf4j.Slf4j;  
import org.eclipse.paho.client.mqttv3.\*;  
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import java.time.LocalDateTime;  
@Slf4j  
public class SubScribeSample implements CommandLineRunner {  
 @Autowired  
 private WaterQualityService waterQualityService;  
 private static int count = 0;  
 private static final String CLIENT\_ID\_KEY = "waterhouduan";  
 private static final String BROKER = "tcp://39.106.34.203:1883";  
 private static final String TOPIC = "LeCar214923124s";  
 private MqttClient client;  
 public void insertData() {  
 String clientId = CLIENT\_ID\_KEY + String.format("%08d", RandomNum.generateRandomNum());  
 int qos = 0;  
 try {  
 log.info("尝试连接");  
 client = new MqttClient(BROKER, clientId, new MemoryPersistence());  
 MqttConnectOptions options = new MqttConnectOptions();  
 options.setConnectionTimeout(60);  
 options.setKeepAliveInterval(60);  
 client.setCallback(new MqttCallback() {  
 @Override  
 public void connectionLost(Throwable cause) {  
 log.info("connectionLost: " + cause.getMessage());  
 }  
 @Override  
 public void messageArrived(String topic, MqttMessage message) {  
 log.info("topic: " + topic);  
 log.info("Qos: " + message.getQos());  
 String content = new String(message.getPayload());  
 MqttDataDTO mqttDataDTO = JSON.parseObject(content.substring(content.indexOf("{")).replaceAll("TEMP:", ""), MqttDataDTO.class);  
 WaterQuality waterQuality = new WaterQuality();  
 waterQuality.setPh(mqttDataDTO.getPH());  
 waterQuality.setConductivity(mqttDataDTO.getEC());  
 waterQuality.setTemperature(mqttDataDTO.getTemper());  
 waterQuality.setTurbidity(mqttDataDTO.getTU());  
 waterQuality.setDate(LocalDateTime.now());  
 waterQualityService.save(waterQuality);  
 log.info("{}", waterQuality);  
 log.info("已经添加" + count++ + "条数据");  
 }  
 @Override  
 public void deliveryComplete(IMqttDeliveryToken token) {  
 log.info("deliveryComplete---------" + token.isComplete());  
 }  
 });  
 client.connect(options);  
 client.subscribe(TOPIC, qos);  
 log.info("接收数据中...");  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 if (client != null) {  
 client.close();  
 }  
 } catch (MqttException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 @Override  
 public void run(String... args) {  
 insertData();  
 }  
}

UserServiceImpl.java

package com.zeb.water\_monitor.service.Impl;  
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;  
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
import com.zeb.water\_monitor.common.CustomException;  
import com.zeb.water\_monitor.entity.User;  
import com.zeb.water\_monitor.enums.RoleEnum;  
import com.zeb.water\_monitor.mapper.UserMapper;  
import com.zeb.water\_monitor.service.UserService;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.stereotype.Service;  
import org.springframework.util.DigestUtils;  
@Service  
@Slf4j  
public class UserServiceImpl extends ServiceImpl<UserMapper, User> implements UserService {  
 @Override  
 public void editPassword(String originPassword, String newPassword, Integer id) {  
 User user = this.getById(id);  
 if (user == null) {  
 throw new CustomException("用户异常");  
 }  
 if (!user.getPassword().equals(DigestUtils.md5DigestAsHex(originPassword.getBytes()))) {  
 throw new CustomException("密码错误");  
 }  
 user.setPassword(DigestUtils.md5DigestAsHex(newPassword.getBytes()));  
 log.info("user：" + user);  
 this.saveOrUpdate(user);  
 }  
 @Override  
 public void register(User user) {  
 LambdaQueryWrapper<User> lambdaQueryWrapper = new LambdaQueryWrapper<>();  
 lambdaQueryWrapper.eq(User::getUsername, user.getUsername());  
 User one = this.getOne(lambdaQueryWrapper);  
 if (one != null) {  
 throw new CustomException("用户名" + one.getUsername() + "已经存在");  
 }  
 user.setPassword(DigestUtils.md5DigestAsHex(user.getPassword().getBytes()));  
 user.setRole(RoleEnum.USER);  
 this.save(user);  
 }  
}

WaterQualityServiceImpl.java

package com.zeb.water\_monitor.service.Impl;  
import com.baomidou.mybatisplus.core.conditions.query.\*;  
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
import com.zeb.water\_monitor.common.CustomException;  
import com.zeb.water\_monitor.entity.WaterQuality;  
import com.zeb.water\_monitor.mapper.WaterQualityMapper;  
import com.zeb.water\_monitor.service.WaterQualityService;  
import com.zeb.water\_monitor.vo.PlotVO;  
import com.zeb.water\_monitor.vo.Prediction;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.stereotype.Service;  
import java.math.BigDecimal;  
import java.time.LocalDateTime;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
import java.util.stream.Collectors;  
@Slf4j  
@Service  
public class WaterQualityServiceImpl extends ServiceImpl<WaterQualityMapper, WaterQuality> implements WaterQualityService {  
 @Override  
 public PlotVO getDataForPlot(String area, Integer period, String indicator) {  
 LocalDateTime now = LocalDateTime.now();  
 LocalDateTime start = now.minusMonths(period);  
 log.info("startTime:" + start);  
 log.info("endTime:" + now);  
 LambdaQueryWrapper<WaterQuality> wrapper = new LambdaQueryWrapper<>();  
 wrapper.ge(WaterQuality::getDate, start);  
 wrapper.le(WaterQuality::getDate, now);  
 wrapper.orderByAsc(WaterQuality::getDate);  
 wrapper.eq(area != null, WaterQuality::getArea, area);  
 List<WaterQuality> list = this.list(wrapper);  
 List<Float> specWaterQualities = new ArrayList<>();  
 List<String> dates = new ArrayList<>();  
 list.stream().map(item -> {  
 String dateStr = item.getDate().getYear() + "-" + item.getDate().getMonthValue() + "-" + item.getDate().getDayOfMonth();  
 dates.add(dateStr);  
 if ("ph".equalsIgnoreCase(indicator)) {  
 specWaterQualities.add(item.getPh());  
 } else if ("temperature".equalsIgnoreCase(indicator)) {  
 specWaterQualities.add(item.getTemperature());  
 } else if ("conductivity".equalsIgnoreCase(indicator)) {  
 specWaterQualities.add(item.getConductivity());  
 } else if ("turbidity".equalsIgnoreCase(indicator)) {  
 specWaterQualities.add(item.getTurbidity());  
 } else if ("tds".equalsIgnoreCase(indicator)) {  
 specWaterQualities.add(item.getTds());  
 } else {  
 throw new CustomException("参数传输错误，请联系管理员");  
 }  
 return item;  
 }).collect(Collectors.toList());  
 PlotVO plotVO = new PlotVO();  
 plotVO.setDates(dates);  
 plotVO.setSpecWaterQualities(specWaterQualities);  
 return plotVO;  
 }  
 @Override  
 public Prediction getDateForPrediction(String indicator) {  
 QueryWrapper<WaterQuality> wrapper = new QueryWrapper<>();  
 wrapper.select("distinct date").lambda().orderByDesc(WaterQuality::getDate).last("limit 5");  
 List<WaterQuality> list = this.list(wrapper);  
 List<String> dateStrs = new ArrayList<>();  
 List<Float> forPlot = new ArrayList<>();  
 List<Float> forPrediction = new ArrayList<>();  
 for (WaterQuality waterQuality : list) {  
 String dateStr = waterQuality.getDate().getYear() + "-" + waterQuality.getDate().getMonthValue() + "-" + waterQuality.getDate().getDayOfMonth();  
 dateStrs.add(dateStr);  
 List<WaterQuality> waterQualityList = this.getWaterQualitiesBySpecificDate(dateStr);  
 float average = calAverage(indicator, waterQualityList);  
 BigDecimal bigDecimal = BigDecimal.valueOf(average);  
 forPlot.add(bigDecimal.setScale(2, BigDecimal.ROUND\_HALF\_UP).floatValue());  
 }  
 for (int i = 0; i < 3; i++) {  
 forPrediction.add(forPlot.get(i));  
 }  
 Collections.reverse(dateStrs);  
 Collections.reverse(forPlot);  
 Collections.reverse(forPrediction);  
 Prediction predictionVO = new Prediction();  
 predictionVO.setDateStrs(dateStrs);  
 predictionVO.setForPlot(forPlot);  
 predictionVO.setForPrediction(forPrediction);  
 return predictionVO;  
 }  
 private List<WaterQuality> getWaterQualitiesBySpecificDate(String dateStr) {  
 String startDateStr = dateStr + " 00:00:00";  
 String endDateStr = dateStr + " 23:59:59";  
 LocalDateTime startDate = LocalDateTime.parse(startDateStr);  
 LocalDateTime endDate = LocalDateTime.parse(endDateStr);  
 LambdaQueryWrapper<WaterQuality> wrapper = new LambdaQueryWrapper<>();  
 wrapper.between(WaterQuality::getDate, startDate, endDate).orderByAsc(WaterQuality::getDate);  
 List<WaterQuality> list = this.list(wrapper);  
 log.info("当天水质数据量: " + list.size());  
 return list;  
 }  
 private float calAverage(String indicator, List<WaterQuality> waterQualities) {  
 float sum = 0;  
 for (WaterQuality waterQuality : waterQualities) {  
 if ("ph".equalsIgnoreCase(indicator)) {  
 sum += waterQuality.getPh();  
 } else if ("temperature".equalsIgnoreCase(indicator)) {  
 sum += waterQuality.getTemperature();  
 } else if ("conductivity".equalsIgnoreCase(indicator)) {  
 sum += waterQuality.getConductivity();  
 } else if ("turbidity".equalsIgnoreCase(indicator)) {  
 sum += waterQuality.getTurbidity();  
 } else if ("tds".equalsIgnoreCase(indicator)) {  
 sum += waterQuality.getTds();  
 } else {  
 throw new CustomException("参数错误，请联系管理员");  
 }  
 }  
 return sum / waterQualities.size();  
 }  
}

StationService.java

package com.zeb.water\_monitor.service;  
import com.zeb.water\_monitor.entity.Station;  
import com.baomidou.mybatisplus.extension.service.IService;  
public interface StationService extends IService<Station> {  
}

UserService.java

package com.zeb.water\_monitor.service;  
import com.baomidou.mybatisplus.extension.service.IService;  
import com.zeb.water\_monitor.entity.User;  
public interface UserService extends IService<User> {  
 void editPassword(String originPassword, String newPassword, Integer id);  
 void register(User user);  
}

WaterQualityService.java

package com.zeb.water\_monitor.service;  
import com.baomidou.mybatisplus.extension.service.IService;  
import com.zeb.water\_monitor.entity.WaterQuality;  
import com.zeb.water\_monitor.vo.PlotVO;  
import com.zeb.water\_monitor.vo.Prediction;  
public interface WaterQualityService extends IService<WaterQuality> {  
 PlotVO getDataForPlot(String area,Integer period,String indicator);  
 Prediction getDateForPrediction(String indicator);  
}

AutoGenerator.java

package com.zeb.water\_monitor.utils;  
import com.baomidou.mybatisplus.annotation.FieldFill;  
import com.baomidou.mybatisplus.core.mapper.BaseMapper;  
import com.baomidou.mybatisplus.generator.FastAutoGenerator;  
import com.baomidou.mybatisplus.generator.config.OutputFile;  
import com.baomidou.mybatisplus.generator.engine.FreemarkerTemplateEngine;  
import com.baomidou.mybatisplus.generator.fill.Column;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
public class AutoGenerator {  
 private static final String URL = "jdbc:mysql://47.120.7.229:3306/water\_monitor?serverTimezone=UTC";  
 private static final String USERNAME = "water\_monitor";  
 private static final String PASSWORD = "6ChtSe2Rrt";  
 public static void main(String[] args) {  
 List<String> tables = new ArrayList<>();  
 tables.add("station");  
 FastAutoGenerator.create(URL, USERNAME, PASSWORD)  
 .globalConfig(builder -> {  
 builder.author("zeb") //作者  
 .outputDir(System.getProperty("user.dir") + "\\src\\main\\java") //输出路径(写到java目录)  
 .enableSwagger() //开启swagger  
 .commentDate("yyyy-MM-dd")  
 .fileOverride(); //开启覆盖之前生成的文件  
 })  
 .packageConfig(builder -> {  
 builder.parent("com.zeb.water\_monitor")  
 .moduleName("")  
 .entity("entity")  
 .service("service")  
 .serviceImpl("service.Impl")  
 .controller("controller")  
 .mapper("mapper")  
 .xml("mapper")  
 .pathInfo(Collections.singletonMap(OutputFile.mapperXml, System.getProperty("user.dir") + "\\src\\main\\resources\\mapper"));  
 })  
 .strategyConfig(builder -> {  
 builder.addInclude(tables)  
 .serviceBuilder()  
 .formatServiceFileName("%sService")  
 .formatServiceImplFileName("%sServiceImpl")  
 .entityBuilder()  
 .addTableFills(new Column("create\_time", FieldFill.INSERT))  
 .addTableFills(new Column("update\_time", FieldFill.UPDATE))  
 .enableLombok()  
 .logicDeleteColumnName("deleted")  
 .enableTableFieldAnnotation()  
 .controllerBuilder()  
 .formatFileName("%sController")  
 .enableRestStyle()  
 .mapperBuilder()  
 .enableBaseResultMap() //生成通用的resultMap  
 .superClass(BaseMapper.class)  
 .formatMapperFileName("%sMapper")  
 .enableMapperAnnotation()  
 .formatXmlFileName("%sMapper");  
 })  
 .templateEngine(new FreemarkerTemplateEngine())  
 .execute();  
 }  
}

PlotVO.java

package com.zeb.water\_monitor.vo;  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
import java.util.List;  
@Data  
@ApiModel("水质数据示意图")  
public class PlotVO {  
 @ApiModelProperty("数值")  
 private List<Float> specWaterQualities;  
 @ApiModelProperty("时间轴")  
 private List<String> dates;  
}

Prediction.java

package com.zeb.water\_monitor.vo;  
import lombok.Data;  
import java.util.List;  
@Data  
public class Prediction {  
 private List<String> dateStrs;  
 private List<Float> forPlot;  
 private List<Float> forPrediction;  
 private Double predValue;  
}

PredictionRes.java

package com.zeb.water\_monitor.vo;  
import lombok.Data;  
@Data  
public class PredictionRes {  
}

SubstandardAreas.java

package com.zeb.water\_monitor.vo;  
import io.swagger.annotations.ApiModel;  
import io.swagger.annotations.ApiModelProperty;  
import lombok.Data;  
import java.util.Collection;  
import java.util.Set;  
@Data  
@ApiModel("不达标区域")  
public class SubstandardAreas {  
 @ApiModelProperty("不达标区域")  
 private Set<String> areas;  
 @ApiModelProperty("不达标天数")  
 private Collection<Integer> days;  
}

index.vue

<template>  
 <div>  
 <div class="titleFont">今日水质监测详细数据</div>  
 <el-row>  
 <el-card style="margin-bottom: 20px">  
 <el-form :inline="true" :model="Form">  
 <el-form-item label="监测区域">  
 <el-select  
 placeholder="请选择"  
 v-model="Form.area"  
 @change="getWaterData"  
 >  
 <el-option  
 v-for="item in areaslist"  
 :label="item.name"  
 :value="item.name"  
 ></el-option>  
 </el-select>  
 </el-form-item>  
 </el-form>  
 <el-button  
 type="primary"  
 style="margin-bottom: 20px"  
 icon="el-icon-plus"  
 @click="showDialog"  
 >添加数据</el-button  
 >  
 <el-table :data="reqData" border style="width: 100%">  
 <el-table-column prop="station" label="监测站" width="150">  
 </el-table-column>  
 <el-table-column prop="ph" label="酸碱度" width="100">  
 </el-table-column>  
 <el-table-column prop="tds" label="溶解氧" width="100">  
 </el-table-column>  
 <el-table-column prop="temperature" label="水温" width="100">  
 </el-table-column>  
 <el-table-column prop="turbidity" label="浊度" width="100">  
 </el-table-column>  
 <el-table-column prop="conductivity" label="电导性" width="100">  
 </el-table-column>  
 <el-table-column prop="date" label="监测时间" width="200">  
 </el-table-column>  
 </el-table-column> -->  
 <el-table-column prop="prop" label="操作">  
 <template slot-scope="{ row, $index }">  
 <el-button  
 type="info"  
 icon="el-icon-info"  
 size="mini"  
 @click="showInfo(row)"  
 >查看</el-button  
 >  
 <el-button  
 type="warning"  
 icon="el-icon-info"  
 size="mini"  
 @click="showInfo(row)"  
 >修改</el-button  
 >  
 <el-button  
 type="danger"  
 icon="el-icon-delete"  
 size="mini"  
 @click="deleteStick(row)"  
 >删除</el-button  
 >  
 </template>  
 </el-table-column>  
 </el-table>  
 </el-card>  
 <el-card>  
 <div  
 class="echart"  
 id="sortchart"  
 :style="{ float: 'left', width: '100%', height: '400px' }"  
 ></div>  
 </el-card>  
 </el-row>  
 <el-dialog title="水质信息" :visible.sync="dialogFormVisible">  
 <el-form style="width: 80%" :model="wForm">  
 <el-form-item label="监测区域" label-width="100px" prop="area">  
 <el-input v-model="wForm.area" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item  
 label="conductivity"  
 label-width="100px"  
 prop="conductivity"  
 >  
 <el-input v-model="wForm.conductivity" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="ph" label-width="100px" prop="ph">  
 <el-input v-model="wForm.ph" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="station" label-width="100px" prop="station">  
 <el-input v-model="wForm.station" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="tds" label-width="100px" prop="tds">  
 <el-input v-model="wForm.tds" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item  
 label="temperature"  
 label-width="100px"  
 prop="temperature"  
 >  
 <el-input v-model="wForm.temperature" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="turbidity" label-width="100px" prop="turbidity">  
 <el-input v-model="wForm.turbidity" auto-complete="off"></el-input>  
 </el-form-item>  
 </el-form>  
 <div slot="footer" class="dialog-footer">  
 <el-button @click="dialogFormVisible = false">取 消</el-button>  
 <el-button type="primary" @click="addOrUpdata">确 定</el-button>  
 </div>  
 </el-dialog>  
 <el-dialog title="水质信息" :visible.sync="dialogStateVisible">  
 <el-form style="width: 80%" :model="cform">  
 <el-form-item label="监测区域" label-width="100px" prop="area">  
 <el-select  
 placeholder="请选择"  
 v-model="Form.area"  
 @change="getWaterData"  
 >  
 <el-option  
 v-for="item in areaslist"  
 :label="item"  
 :value="item"  
 ></el-option>  
 </el-select>  
 </el-form-item>  
 <el-form-item  
 label="conductivity"  
 label-width="100px"  
 prop="conductivity"  
 >  
 <el-input v-model="cform.conductivity" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="ph" label-width="100px" prop="ph">  
 <el-input v-model="cform.ph" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="station" label-width="100px" prop="station">  
 <el-input v-model="cform.station" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="tds" label-width="100px" prop="tds">  
 <el-input v-model="cform.tds" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item  
 label="temperature"  
 label-width="100px"  
 prop="temperature"  
 >  
 <el-input v-model="cform.temperature" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="turbidity" label-width="100px" prop="turbidity">  
 <el-input v-model="cform.turbidity" auto-complete="off"></el-input>  
 </el-form-item>  
 </el-form>  
 <div slot="footer" class="dialog-footer">  
 <el-button @click="dialogStateVisible = false">取 消</el-button>  
 <el-button type="primary" @click="changeState">确 定</el-button>  
 </div>  
 </el-dialog>  
 style="width: 100%"  
 border  
 :data="sList"  
 @selection-change="handleSelectionChange"  
 >  
 <el-table-column  
 type="selection"  
 prop="prop"  
 label="序号"  
 width="80"  
 align="center"  
 ></el-table-column>  
 <el-table-column  
 prop="walkingStickId"  
 label="拐杖编号"  
 width="140"  
 ></el-table-column>  
 <el-table-column  
 prop="activationTime"  
 label="上次登录时间"  
 width="200"  
 ></el-table-column>  
 <el-table-column prop="sList" label="属性" width="width">  
 <template slot-scope="{ row, $index }">  
 <el-tag type="success" style="margin: 0px 5px"  
 >IP:{{ row.province }}</el-tag  
 >  
 <el-tag type="success" style="margin: 0px 5px">{{  
 row.operatingFrequency  
 }}</el-tag>  
 <el-tag type="success" style="margin: 0px 5px">{{  
 row.bindingState  
 }}</el-tag>  
 </template></el-table-column  
 >  
 <el-table-column prop="prop" label="操作" width="300">  
 <template slot-scope="{ row, $index }">  
 <el-button  
 type="info"  
 icon="el-icon-info"  
 size="mini"  
 @click="showInfo(row)"  
 >查看</el-button  
 >  
 <el-button  
 type="warning"  
 icon="el-icon-edit"  
 size="mini"  
 @click="showState(row)"  
 >解绑</el-button  
 >  
 <el-button  
 type="danger"  
 icon="el-icon-delete"  
 size="mini"  
 @click="deleteStick(row)"  
 >删除</el-button  
 >  
 </template>  
 </el-table-column>  
 </el-table> -->  
 <el-form style="width: 80%" :model="sForm">  
 <el-form-item label="当前用户状态" label-width="100px">  
 <span>{{ sForm.bindingState }}</span>  
 </el-form-item>  
 </el-form>  
 <div slot="footer" class="dialog-footer">  
 <el-button @click="dialogStateVisible = false">取消</el-button>  
 <el-button type="primary" @click="changeState">解禁/禁用</el-button>  
 </div>  
 </el-dialog> -->  
 <el-form style="width: 80%" :model="sForm">  
 <el-form-item label="当前用户状态" label-width="100px">  
 <span>{{ sForm.bindingState }}</span>  
 </el-form-item>  
 </el-form>  
 <div slot="footer" class="dialog-footer">  
 <el-button @click="dialogStateVisible = false">取消</el-button>  
 <el-button type="primary" @click="changeState">解禁/禁用</el-button>  
 </div>  
 </el-dialog> -->  
 style="text-align: center"  
 :total="total"  
 :current-page="page"  
 :page-size="limit"  
 :page-sizes="[3, 5, 10]"  
 layout="prev,pager,next,jumper,->,sizes,total"  
 @current-change="getStickList"  
 @size-change="handleSizeChange"  
 ></el-pagination> -->  
 </div>  
</template>  
 <script>  
import \* as echarts from "echarts";  
export default {  
 name: "monitor",  
 data() {  
 return {  
 wForm: {},  
 dialogFormVisible: false,  
 batchDeleteArr: [],  
 sList: [],  
 dialogStateVisible: false,  
 dialogInfoVisible: false,  
 cform: {},  
 page: 1,  
 limit: 5,  
 total: 0,  
 stickForm: {},  
 tableData: [  
 {  
 station: "01",  
 ph: "6.75",  
 Do: "0.2",  
 target: "合格",  
 },  
 {  
 station: "02",  
 ph: "7.55",  
 Do: " 0.35",  
 target: "合格",  
 },  
 {  
 station: "03",  
 ph: "6.83",  
 Do: "0.38",  
 target: "合格",  
 },  
 {  
 station: "04",  
 ph: "7.46",  
 Do: "0.45",  
 target: "合格",  
 },  
 {  
 station: "01",  
 ph: "6.75",  
 Do: "0.2",  
 target: "合格",  
 },  
 {  
 station: "02",  
 ph: "7.55",  
 Do: " 0.35",  
 target: "合格",  
 },  
 {  
 station: "03",  
 ph: "6.83",  
 Do: "0.38",  
 target: "合格",  
 },  
 {  
 station: "04",  
 ph: "7.46",  
 Do: "0.45",  
 target: "合格",  
 },  
 ],  
 reqData: [],  
 areaslist: [],  
 Form: {},  
 rules: {  
 userName: [  
 { required: true, message: "请输入用户名", trigger: "blur" },  
 {  
 min: 2,  
 max: 10,  
 message: "长度在 2 到 10 个字符",  
 trigger: "blur",  
 },  
 ],  
 logoUrl: [{ required: true, message: "请选择品牌图片" }],  
 },  
 sourceData: [],  
 option: {  
 dataset: [  
 {  
 dimensions: ["area", "day"],  
 source: [],  
 },  
 {  
 transform: {  
 type: "sort",  
 config: { dimension: "day", order: "desc" },  
 },  
 },  
 ],  
 xAxis: {  
 type: "category",  
 axisLabel: { interval: 0, rotate: 30 },  
 },  
 yAxis: {},  
 series: {  
 type: "bar",  
 encode: { x: "area", y: "day" },  
 datasetIndex: 1,  
 },  
 },  
 };  
 },  
 mounted() {  
 this.getsubstandardData();  
 this.getAreas();  
 },  
 watch: {  
 stickForm() {  
 this.getStickList();  
 },  
 deep: true,  
 },  
 methods: {  
 showDialog() {  
 this.dialogFormVisible = true;  
 this.wForm = {  
 area: "",  
 conductivity: "",  
 ph: "",  
 station: "",  
 tds: "",  
 temperature: "",  
 turbidity: "",  
 };  
 },  
 async getAreas() {  
 let res = await this.$API.water.reqAllAreas();  
 this.areaslist = res.data;  
 },  
 async getWaterData() {  
 let data = {  
 area: this.Form.area,  
 day: this.getNowFormatDate(),  
 };  
 let res = await this.$API.water.reqQualityDay(data);  
 this.reqData = res.data;  
 if (this.reqData.length == 0) {  
 this.$message({  
 message: "暂时没有数据！",  
 type: "warning",  
 });  
 }  
 },  
 getStickForm(form) {  
 this.stickForm = form;  
 this.getStickList();  
 },  
 showInfo(row) {  
 this.cform = { ...row };  
 this.dialogStateVisible = true;  
 },  
 showState(row) {  
 this.cform = { ...row };  
 this.dialogStateVisible = true;  
 },  
 async changeState() {  
 console.log(this.cform);  
 let res = await this.$API.water.changeWaterData(this.cform);  
 if (res.code == 200) {  
 if (res.code == 200) {  
 this.$message({  
 type: "success",  
 message: "修改成功!",  
 });  
 this.getWaterData();  
 }  
 }  
 },  
 async deleteStick(row) {  
 this.$confirm("确认删除该用户?", "提示", {  
 confirmButtonText: "确定",  
 cancelButtonText: "取消",  
 type: "warning",  
 })  
 .then(async () => {  
 let res = await this.$API.water.deleteWaterData(row.id);  
 if (res.code == 200) {  
 this.$message({  
 type: "success",  
 message: "删除成功!",  
 });  
 this.getWaterData();  
 }  
 })  
 .catch(() => {  
 this.$message({  
 type: "info",  
 message: "已取消删除",  
 });  
 });  
 },  
 handleSelectionChange(val) {  
 this.batchDeleteArr = val;  
 },  
 async batchDelete() {  
 let deleteData = this.batchDeleteArr.map((item) => {  
 const { walkingStickId } = item;  
 return walkingStickId;  
 });  
 this.$confirm("确认批量删除该用户?", "提示", {  
 confirmButtonText: "确定",  
 cancelButtonText: "取消",  
 type: "warning",  
 })  
 .then(async () => {  
 let res = await this.$API.stick.reqDeleteSticks(deleteData);  
 if (res.code == 200) {  
 this.$message({  
 type: "success",  
 message: "删除成功!",  
 });  
 this.getStickList(  
 this.sList.length > 1 ? this.page : this.page - 1  
 );  
 }  
 })  
 .catch(() => {  
 this.$message({  
 type: "info",  
 message: "已取消删除",  
 });  
 });  
 },  
 initEcharts() {  
 const myChart = echarts.init(document.getElementById("sortchart")); // 图标初始化  
 myChart.setOption(this.option); // 渲染页面  
 window.addEventListener("resize", () => {  
 myChart.resize();  
 });  
 },  
 async addOrUpdata() {  
 console.log("添加或者更新数据");  
 let res = await this.$API.water.addWaterData(this.wForm);  
 if (res.code == 200) {  
 this.dialogFormVisible = false;  
 this.$message({  
 message: "添加成功",  
 type: "success",  
 });  
 this.getWaterData();  
 }  
 },  
 async getsubstandardData() {  
 let res = await this.$API.water.reqSubstandardData();  
 console.log(res);  
 if (res.code == 200) {  
 let temp = [];  
 for (let i in res.data.areas) {  
 temp[i] = [res.data.areas[i], res.data.days[i]];  
 }  
 console.log(temp);  
 this.option.dataset[0].source = temp;  
 this.initEcharts();  
 }  
 },  
 },  
};  
</script>  
<style>  
.titleFont {  
 text-align: left;  
 font-weight: 600;  
 font-size: 24px;  
 margin-left: 20px;  
 margin-bottom: 15px;  
}  
</style>

index.vue

<template>  
 <div>  
 <el-card style="margin: 20px 0">  
 <CategorySelect  
 @getScreeningCondition="getScreeningCondition"  
 ></CategorySelect>  
 </el-card>  
 <el-card>  
 <el-button  
 type="danger"  
 icon="el-icon-delete"  
 style="margin-bottom: 5px; float: right"  
 :disabled="batchDeleteArr.length === 0"  
 @click="batchDelete"  
 >批量删除</el-button  
 >  
 <el-table  
 style="width: 100%"  
 border  
 stripe  
 :data="historyData"  
 :default-sort="{ prop: 'date', order: 'descending' }"  
 @selection-change="handleSelectionChange"  
 >  
 <el-table-column  
 type="selection"  
 prop="prop"  
 label="序号"  
 width="80"  
 align="center"  
 ></el-table-column>  
 <el-table-column  
 prop="date"  
 sortable  
 label="监测时间"  
 width="200"  
 ></el-table-column>  
 <el-table-column  
 prop="station"  
 label="监测船"  
 width="150"  
 ></el-table-column>  
 <el-table-column label="水质数据">  
 <el-table-column  
 prop="ph"  
 label="酸碱度(PH)"  
 width="100"  
 ></el-table-column>  
 <el-table-column  
 prop="tds"  
 label="溶解氧(DO mg/L)"  
 width="100"  
 ></el-table-column>  
 <el-table-column  
 prop="turbidity"  
 label="浊度(NTU)"  
 width="100"  
 ></el-table-column>  
 <el-table-column  
 prop="temperature"  
 label="水温(摄氏度)"  
 width="100"  
 ></el-table-column>  
 <el-table-column  
 prop="conductivity"  
 label="导电性"  
 width="100"  
 ></el-table-column>  
 </el-table-column>  
 <el-table-column prop="prop" label="操作">  
 <template slot-scope="{ row, $index }">  
 <el-button  
 type="info"  
 icon="el-icon-info"  
 size="mini"  
 @click="showInfo(row)"  
 >查看</el-button  
 >  
 <el-button  
 type="warning"  
 icon="el-icon-edit"  
 size="mini"  
 @click="showState(row)"  
 >编辑</el-button  
 >  
 <el-button  
 type="danger"  
 icon="el-icon-delete"  
 size="mini"  
 @click="deleteStick(row)"  
 >删除</el-button  
 >  
 </template>  
 </el-table-column>  
 </el-table>  
 <el-dialog title="用户状态" :visible.sync="dialogInfoVisible">  
 <el-form style="width: 80%" :model="sForm">  
 <el-form-item label="当前用户状态" label-width="100px">  
 <span>{{ sForm.bindingState }}</span>  
 </el-form-item>  
 </el-form>  
 <div slot="footer" class="dialog-footer">  
 <el-button @click="dialogStateVisible = false">取消</el-button>  
 <el-button type="primary" @click="changeState">解禁/禁用</el-button>  
 </div>  
 </el-dialog>  
 <el-dialog title="用户状态" :visible.sync="dialogStateVisible">  
 <el-form style="width: 80%" :model="sForm">  
 <el-form-item label="当前用户状态" label-width="100px">  
 <span>{{ sForm.bindingState }}</span>  
 </el-form-item>  
 </el-form>  
 <div slot="footer" class="dialog-footer">  
 <el-button @click="dialogStateVisible = false">取消</el-button>  
 <el-button type="primary" @click="changeState">解禁/禁用</el-button>  
 </div>  
 </el-dialog>  
 <el-pagination  
 style="text-align: center"  
 :total="total"  
 :current-page="page"  
 :page-size="limit"  
 :page-sizes="[3, 5, 10]"  
 layout="prev,pager,next,jumper,->,sizes,total"  
 @current-change="getWaterDataList"  
 @size-change="handleSizeChange"  
 ></el-pagination>  
 </el-card>  
 </div>  
</template>  
 <script>  
export default {  
 name: "monitor",  
 data() {  
 return {  
 batchDeleteArr: [],  
 sList: [],  
 dialogStateVisible: false,  
 dialogInfoVisible: false,  
 sForm: {},  
 page: 1,  
 limit: 5,  
 total: 0,  
 historyData: [],  
 ScreeningCondition: {},  
 };  
 },  
 watch: {  
 ScreeningCondition() {  
 this.getWaterDataList();  
 },  
 deep: true,  
 },  
 methods: {  
 getScreeningCondition(form) {  
 this.ScreeningCondition = form;  
 this.getWaterDataList();  
 },  
 async getWaterDataList(pager = 1) {  
 this.page = pager;  
 const { page, limit } = this;  
 let data = {  
 area: this.ScreeningCondition.area,  
 pageSize: limit,  
 pageNum: page,  
 endDate: this.ScreeningCondition.date[1],  
 stardDate: this.ScreeningCondition.date[0],  
 station: this.ScreeningCondition.station,  
 };  
 let res = await this.$API.water.reqHistoricalWaterQualityData(data);  
 this.historyData = res.data.records;  
 this.total = res.data.total;  
 },  
 handleSizeChange(limit) {  
 this.limit = limit;  
 this.getWaterDataList(this.page);  
 },  
 showInfo(row) {  
 console.log(row, "row");  
 this.sForm = { ...row };  
 this.dialogStateVisible = true;  
 },  
 showState(row) {  
 this.sForm = { ...row };  
 this.dialogStateVisible = true;  
 },  
 async changeState() {  
 let data = {  
 bindingState: this.sForm.bindingState == "已绑定" ? "未绑定" : "已绑定",  
 walkingStickId: this.sForm.walkingStickId,  
 };  
 let res = await this.$API.stick.reqChangeState(data);  
 console.log(res);  
 this.getWaterDataList();  
 },  
 async deleteStick(row) {  
 this.$confirm("确认删除该用户?", "提示", {  
 confirmButtonText: "确定",  
 cancelButtonText: "取消",  
 type: "warning",  
 })  
 .then(async () => {  
 let res = await this.$API.stick.reqDeleteStick(row.walkingStickId);  
 if (res.code == 200) {  
 this.$message({  
 type: "success",  
 message: "删除成功!",  
 });  
 this.getWaterDataList(  
 this.sList.length > 1 ? this.page : this.page - 1  
 );  
 }  
 })  
 .catch(() => {  
 this.$message({  
 type: "info",  
 message: "已取消删除",  
 });  
 });  
 },  
 handleSelectionChange(val) {  
 this.batchDeleteArr = val;  
 },  
 async batchDelete() {  
 let deleteData = this.batchDeleteArr.map((item) => {  
 const { walkingStickId } = item;  
 return walkingStickId;  
 });  
 this.$confirm("确认批量删除该用户?", "提示", {  
 confirmButtonText: "确定",  
 cancelButtonText: "取消",  
 type: "warning",  
 })  
 .then(async () => {  
 let res = await this.$API.stick.reqDeleteSticks(deleteData);  
 if (res.code == 200) {  
 this.$message({  
 type: "success",  
 message: "删除成功!",  
 });  
 this.getWaterDataList(  
 this.sList.length > 1 ? this.page : this.page - 1  
 );  
 }  
 })  
 .catch(() => {  
 this.$message({  
 type: "info",  
 message: "已取消删除",  
 });  
 });  
 },  
 },  
};  
</script>  
 <style>  
</style>

index.vue

<template>  
 <div>  
 <div class="title">  
 <el-form :inline="true" :model="condition" class="demo-form-inline">  
 <el-form-item label="水质指标">  
 <el-select v-model="condition.indicator">  
 <el-option  
 v-for="item in indicatorData"  
 :label="item"  
 :value="item"  
 ></el-option>  
 </el-select>  
 </el-form-item>  
 <el-form-item label="监测区域">  
 <el-select placeholder="请选择" v-model="condition.area">  
 <el-option  
 v-for="item in areaslist"  
 :label="item.name"  
 :value="item.name"  
 ></el-option>  
 </el-select>  
 </el-form-item>  
 <el-form-item label="时间">  
 <el-input v-model="condition.period" placeholder=""></el-input>  
 </el-form-item>  
 <el-form-item>  
 <el-button type="primary" icon="el-icon-search" @click="onSubmit"  
 >查询</el-button  
 >  
 </el-form-item>  
 </el-form>  
 </div>  
 <div class="titleFont line">  
 模型预计PH值:8.68  
 <div  
 class="echart"  
 id="mychart"  
 :style="{ float: 'left', width: '100%', height: '400px' }"  
 ></div>  
 </div>  
 </div>  
</template>  
<script>  
import \* as echarts from "echarts";  
export default {  
 name: "Trend",  
 data() {  
 return {  
 condition: {},  
 tableData: [  
 {  
 type: "LSTM",  
 error: "0.18",  
 people: "admin",  
 date: "2023-05-09",  
 },  
 {  
 type: "SVM",  
 error: "0.18",  
 people: "admin",  
 date: "2023-05-09",  
 },  
 {  
 type: "RVM",  
 error: "0.18",  
 people: "admin",  
 date: "2023-05-09",  
 },  
 ],  
 indicatorData: [],  
 areaslist: [],  
 echartData: {},  
 };  
 },  
 mounted() {  
 this.getIndicator();  
 this.getAreas();  
 },  
 methods: {  
 async getAreas() {  
 let res = await this.$API.water.reqAllAreas();  
 this.areaslist = res.data;  
 },  
 async onSubmit() {  
 console.log(this.condition);  
 let res = await this.$API.water.reqWaterPlot(this.condition);  
 console.log(res);  
 this.echartData.x = res.data.datas;  
 this.echartData.y = res.data.specWaterQualities;  
 this.initEcharts();  
 },  
 initEcharts() {  
 const option = {  
 xAxis: {  
 type: "category",  
 data: this.echartData.x,  
 },  
 yAxis: {  
 type: "value",  
 },  
 series: [  
 {  
 data: this.echartData.y,  
 type: "line",  
 },  
 ],  
 };  
 const myChart = echarts.init(document.getElementById("mychart")); // 图标初始化  
 myChart.setOption(option); // 渲染页面  
 window.addEventListener("resize", () => {  
 myChart.resize();  
 });  
 },  
 async getIndicator() {  
 let res = await this.$API.water.reqIndicator();  
 console.log(res);  
 if (res.code == 200) {  
 this.indicatorData = res.data;  
 }  
 },  
 },  
};  
</script>  
<style scoped>  
.titleFont {  
 text-align: center;  
 font-weight: 600;  
 font-size: 24px;  
 margin: 10px auto;  
}  
.line {  
 margin-top: 20px;  
 height: 200px; \*/  
}  
</style>

index.vue

<template>  
 <div>  
 <el-card style="margin: 20px 0">  
 <el-button  
 type="primary"  
 icon="el-icon-plus"  
 @click="dialogAddVisible = true"  
 >添加管理员</el-button  
 >  
 </el-card>  
 <el-card>  
 <el-table style="width: 100%" border :data="list">  
 <el-table-column  
 type="index"  
 prop="prop"  
 label="序号"  
 width="80"  
 align="center"  
 ></el-table-column>  
 <el-table-column  
 prop="adminName"  
 label="姓名"  
 width="140"  
 ></el-table-column>  
 <el-table-column  
 prop="adminId"  
 label="工号"  
 width="200"  
 ></el-table-column>  
 <el-table-column  
 prop="phoneNumber"  
 label="电话号码"  
 width="200"  
 ></el-table-column>  
 <el-table-column  
 prop="loginTime"  
 label="上次登录时间"  
 width="width"  
 ></el-table-column>  
 <el-table-column prop="prop" label="操作" width="300">  
 <template slot-scope="{ row, $index }">  
 <el-button  
 type="info"  
 icon="el-icon-info"  
 size="mini"  
 @click="checkAdministratorInfo(row)"  
 >查看</el-button  
 >  
 <el-button  
 type="warning"  
 icon="el-icon-edit"  
 size="mini"  
 @click="updateAdministratorInfo(row)"  
 >编辑</el-button  
 >  
 <el-button  
 type="danger"  
 icon="el-icon-delete"  
 size="mini"  
 @click="delectAdmin(row)"  
 >删除</el-button  
 >  
 </template>  
 </el-table-column>  
 </el-table>  
 <el-pagination  
 style="text-align: center"  
 :total="total"  
 :current-page="page"  
 :page-size="limit"  
 :page-sizes="[3, 5, 10]"  
 layout="prev,pager,next,jumper,->,sizes,total"  
 @current-change="getAdministratorList"  
 @size-change="handleSizeChange"  
 ></el-pagination>  
 <el-dialog :title="aList.adminName" :visible.sync="dialogTableVisible">  
 <el-form style="width: 80%" :model="aList" label-position="left">  
 <el-form-item label="工号" label-width="100px">  
 <span>{{ aList.adminId }}</span>  
 </el-form-item>  
 <el-form-item label="长号" label-width="100px">  
 <span>{{ aList.phoneNumber }}</span>  
 </el-form-item>  
 <el-form-item label="创建时间" label-width="100px">  
 <span>{{ aList.adminCreateData }}</span>  
 </el-form-item>  
 <el-form-item label="登录IP" label-width="100px">  
 <span>{{ aList.loginIp }}</span>  
 </el-form-item>  
 <el-form-item label="工作邮箱" label-width="100px">  
 <span>{{ aList.email }}</span>  
 </el-form-item>  
 <el-form-item label="最近登录IP" label-width="100px">  
 <span>{{ aList.loginIp }}</span>  
 </el-form-item>  
 </el-form>  
 </el-dialog>  
 <el-dialog title="管理员信息" :visible.sync="dialogStateVisible">  
 <el-form style="width: 80%" :model="aList" label-position="left">  
 <el-form-item label="用户名" label-width="100px" prop="userName">  
 <el-input v-model="aList.adminName" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="电话号码" label-width="100px" prop="userPhone">  
 <el-input  
 v-model="aList.phoneNumber"  
 auto-complete="off"  
 ></el-input>  
 </el-form-item>  
 <el-form-item label="工作邮箱" label-width="100px" prop="userEmail">  
 <el-input v-model="aList.email" auto-complete="off"></el-input>  
 </el-form-item>  
 </el-form>  
 <div slot="footer" class="dialog-footer">  
 <el-button  
 @click="  
 {  
 dialogStateVisible = false;  
 aList = {};  
 }  
 "  
 >取 消</el-button  
 >  
 <el-button type="primary" @click="changeAdminList">确 定</el-button>  
 </div>  
 </el-dialog>  
 <el-dialog title="管理员信息" :visible.sync="dialogAddVisible">  
 <el-form style="width: 80%" :model="aForm" label-position="left">  
 <el-form-item label="用户名" label-width="100px" prop="userName">  
 <el-input v-model="aForm.adminName" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="电话号码" label-width="100px" prop="userPhone">  
 <el-input  
 v-model="aForm.phoneNumber"  
 auto-complete="off"  
 ></el-input> </el-form-item  
 ><el-form-item label="密码" label-width="100px" prop="userName">  
 <el-input v-model="aForm.password" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="工作邮箱" label-width="100px" prop="userEmail">  
 <el-input v-model="aForm.email" auto-complete="off"></el-input>  
 </el-form-item>  
 </el-form>  
 <div slot="footer" class="dialog-footer">  
 <el-button @click="dialogStateVisible = false">取 消</el-button>  
 <el-button type="primary" @click="addAdministrator">确 定</el-button>  
 </div>  
 </el-dialog>  
 </el-card>  
 </div>  
</template>  
<script>  
export default {  
 name: "Root",  
 data() {  
 return {  
 page: 1,  
 limit: 5,  
 total: 0,  
 list: [],  
 aList: {},  
 aForm: {},  
 dialogTableVisible: false,  
 dialogStateVisible: false,  
 dialogAddVisible: false,  
 };  
 },  
 mounted() {  
 this.getAdministratorList();  
 },  
 methods: {  
 async getAdministratorList(pager = 1) {  
 this.page = pager;  
 const { page, limit } = this;  
 let res = await this.$API.userInfo.reqAdministratorList(  
 this.page,  
 this.limit  
 );  
 console.log(res.data.result);  
 if (res.code == 200) {  
 this.total = res.data.total;  
 this.list = res.data.result;  
 }  
 },  
 handleSizeChange(limit) {  
 this.limit = limit;  
 this.getAdministratorList(this.page);  
 },  
 checkAdministratorInfo(row) {  
 this.dialogTableVisible = true;  
 this.aList = { ...row };  
 },  
 async updateAdministratorInfo(row) {  
 console.log("row", row);  
 this.dialogStateVisible = true;  
 this.aList = { ...row };  
 },  
 async changeAdminList() {  
 let data = {  
 adminId: this.aList.adminId,  
 adminName: this.aList.adminName,  
 email: this.aList.email,  
 phoneNumber: this.aList.phoneNumber,  
 };  
 let res = await this.$API.userInfo.requpdateAdministratorInfo(data);  
 console.log(res);  
 if (res.code == 200) {  
 this.getAdministratorList();  
 this.aList = [];  
 this.$message({  
 type: "success",  
 message: "修改成功",  
 });  
 }  
 },  
 delectAdmin(row) {  
 this.$confirm("确认删除该用户?", "提示", {  
 confirmButtonText: "确定",  
 cancelButtonText: "取消",  
 type: "warning",  
 })  
 .then(async () => {  
 let result = await this.$API.userInfo.reqDeleteAdmin(row.adminId);  
 if (result.code == 200) {  
 this.$message({  
 type: "success",  
 message: "删除成功!",  
 });  
 this.getAdministratorList();  
 }  
 })  
 .catch(() => {  
 this.$message({  
 type: "info",  
 message: "已取消删除",  
 });  
 });  
 },  
 async addAdministrator() {  
 let res = await this.$API.userInfo.reqAddAdmin(this.aForm);  
 console.log(res);  
 if (res.code == 200) {  
 this.getAdministratorList();  
 this.$message({  
 type: "success",  
 message: "添加成功!",  
 });  
 this.dialogAddVisible = false;  
 }  
 },  
 },  
};  
</script>  
<style>  
</style>

index.vue

<template>  
 <div>  
 <el-button  
 type="primary"  
 icon="el-icon-plus"  
 style="margin: 10px 0px"  
 @click="showDialog"  
 >添加</el-button  
 >  
 <el-table style="width: 100%" border :data="list">  
 <el-table-column  
 type="index"  
 prop="prop"  
 label="序号"  
 width="80"  
 align="center"  
 ></el-table-column>  
 <el-table-column  
 prop="userName"  
 label="用户名"  
 width="width"  
 ></el-table-column>  
 <el-table-column  
 prop="phoneNumber"  
 label="电话号码"  
 width="width"  
 ></el-table-column>  
 <el-table-column  
 prop="email"  
 label="邮箱地址"  
 width="width"  
 ></el-table-column>  
 <el-table-column prop="userId" label="用户编号" width="width">  
 <img :src="row.logoUrl" style="width: 100px; height: 100px" />  
 </template> -->  
 </el-table-column>  
 <el-table-column prop="prop" label="操作" width="width">  
 <template slot-scope="{ row, $index }">  
 <el-button  
 type="info"  
 icon="el-icon-info"  
 size="mini"  
 @click="checkUserInfo(row)"  
 >查看</el-button  
 >  
 <el-button  
 type="warning"  
 icon="el-icon-edit"  
 size="mini"  
 @click="updateUserInfo(row)"  
 >修改状态</el-button  
 >  
 type="danger"  
 icon="el-icon-delete"  
 size="mini"  
 @click="deleteTradeMark(row)"  
 >删除</el-button  
 > -->  
 </template>  
 </el-table-column>  
 </el-table>  
 <el-pagination  
 style="text-align: center"  
 :total="total"  
 :current-page="page"  
 :page-size="limit"  
 :page-sizes="[3, 5, 10]"  
 layout="prev,pager,next,jumper,->,sizes,total"  
 @current-change="getUserList"  
 @size-change="handleSizeChange"  
 ></el-pagination>  
 <el-dialog title="用户信息" :visible.sync="dialogTableVisible">  
 <el-form  
 style="width: 80%"  
 :model="userForm"  
 :rules="rules"  
 ref="ruleForm"  
 >  
 <el-form-item label="用户名" label-width="100px">  
 <span>{{ userForm.userName }}</span>  
 </el-form-item>  
 <el-form-item label="状态" label-width="100px">  
 <span>{{ userForm.userStatus }}</span>  
 >{{ userForm.userStatus }}}</textarea  
 > -->  
 <span v-show="userForm.userStatus">禁用状态</span> -->  
 </el-form-item>  
 <el-form-item label="用户编号" label-width="100px">  
 <span>{{ userForm.userId }}</span>  
 </el-form-item>  
 <el-form-item label="电话号码" label-width="100px">  
 <span>{{ userForm.phoneNumber }}</span>  
 </el-form-item>  
 <el-form-item label="最近登录时间" label-width="100px">  
 <span>{{ userForm.loginTime }}</span>  
 </el-form-item>  
 <el-form-item label="最近登录IP" label-width="100px">  
 <span>{{ userForm.loginIp }}</span>  
 </el-form-item>  
 <el-form-item label="绑定拐杖编号" label-width="100px">  
 <span>{{ userForm.walkingStickId }}</span>  
 </el-form-item>  
 <span>{{ userForm.userPhone }}</span>  
 </el-form-item> -->  
 </el-form>  
 </el-dialog>  
 <el-dialog title="用户信息" :visible.sync="dialogFormVisible">  
 <el-form  
 style="width: 80%"  
 :model="userForm"  
 :rules="rules"  
 ref="ruleForm"  
 >  
 <el-form-item label="用户名" label-width="100px" prop="userName">  
 <el-input v-model="userForm.userName" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="电话号码" label-width="100px" prop="userPhone">  
 <el-input v-model="userForm.userPhone" auto-complete="off"></el-input>  
 </el-form-item>  
 <el-form-item label="邮箱地址" label-width="100px" prop="userEmail">  
 <el-input v-model="userForm.userEmail" auto-complete="off"></el-input>  
 </el-form-item>  
 </el-form>  
 <div slot="footer" class="dialog-footer">  
 <el-button @click="dialogFormVisible = false">取 消</el-button>  
 <el-button type="primary" @click="addOrUpdataTradeMark"  
 >确 定</el-button  
 >  
 </div>  
 </el-dialog>  
 <el-dialog title="用户状态" :visible.sync="dialogStateVisible">  
 <el-form  
 style="width: 80%"  
 :model="userForm"  
 :rules="rules"  
 ref="ruleForm"  
 >  
 <el-form-item label="当前用户状态" label-width="100px">  
 <span>{{ uState }}</span>  
 </el-form-item>  
 </el-form>  
 <div slot="footer" class="dialog-footer">  
 <el-button @click="dialogStateVisible = false">取 消</el-button>  
 <el-button type="primary" @click="changeUserState">解禁/禁用</el-button>  
 </div>  
 </el-dialog>  
 </div>  
</template>  
 <script>  
export default {  
 name: "Info",  
 data() {  
 return {  
 page: 1,  
 limit: 5,  
 total: 0,  
 list: [  
 ],  
 dialogFormVisible: false,  
 dialogTableVisible: false,  
 dialogStateVisible: false,  
 userForm: {  
 userName: "",  
 userPhone: "",  
 userId: "",  
 userEmail: "",  
 },  
 uState: "",  
 rules: {  
 userName: [  
 { required: true, message: "请输入用户名", trigger: "blur" },  
 {  
 min: 2,  
 max: 10,  
 message: "长度在 2 到 10 个字符",  
 trigger: "blur",  
 },  
 ],  
 logoUrl: [{ required: true, message: "请选择品牌图片" }],  
 },  
 };  
 },  
 watch: {  
 "userForm.userStatus"(noOld, old) {  
 return (this.uState = noOld == 0 ? "正常" : "禁用");  
 },  
 deep: true,  
 },  
 mounted() {  
 this.getUserList();  
 },  
 methods: {  
 async getUserList(pager = 1) {  
 this.page = pager;  
 const { page, limit } = this;  
 let res = await this.$API.userInfo.reqUserList(this.page, this.limit);  
 console.log(res.data.result);  
 if (res.code == 200) {  
 this.total = res.data.total;  
 this.list = res.data.result;  
 }  
 },  
 handleSizeChange(limit) {  
 this.limit = limit;  
 this.getUserList(this.page);  
 },  
 showDialog() {  
 this.dialogFormVisible = true;  
 this.userForm = {  
 userName: "",  
 userPhone: "",  
 userId: "",  
 userEmail: "",  
 };  
 },  
 checkUserInfo(row) {  
 this.dialogTableVisible = true;  
 console.log(row);  
 this.userForm = { ...row };  
 },  
 async updateUserInfo(row) {  
 console.log("row", row);  
 this.dialogStateVisible = true;  
 this.userForm = { ...row };  
 },  
 async changeUserState() {  
 let data = {  
 userId: this.userForm.userId,  
 userStatus: this.userForm.userStatus == 0 ? 1 : 0,  
 };  
 let res = await this.$API.userInfo.requpdateUserStatus(data);  
 console.log(res);  
 if (res.code == 200) {  
 this.getUserList();  
 this.$message({  
 type: "success",  
 message: "修改成功",  
 });  
 }  
 },  
 addOrUpdataTradeMark() {  
 this.$refs.ruleForm.validate(async (success) => {  
 if (success) {  
 this.dialogFormVisible = false;  
 let res = await this.$API.trademark.reqAddOrUpdataTradeMark(  
 this.userForm  
 );  
 if (res.code == 200) {  
 this.$message({  
 type: "success",  
 message: this.userForm.id ? "修改品牌成功" : "添加品牌成功",  
 });  
 this.getUserList(this.userForm.id ? this.page : 1);  
 }  
 } else {  
 console.log("error sumbit!");  
 return false;  
 }  
 });  
 },  
 deleteTradeMark(row) {  
 this.$confirm("确认删除该用户?", "提示", {  
 confirmButtonText: "确定",  
 cancelButtonText: "取消",  
 type: "warning",  
 })  
 .then(async () => {  
 let result = await this.$API.trademark.reqDeleteTradeMark(row.id);  
 if (result.code == 200) {  
 this.$message({  
 type: "success",  
 message: "删除成功!",  
 });  
 this.getUserList(this.list.length > 1 ? this.page : this.page - 1);  
 }  
 })  
 .catch(() => {  
 this.$message({  
 type: "info",  
 message: "已取消删除",  
 });  
 });  
 },  
 },  
};  
</script>  
<style>  
.avatar-uploader .el-upload {  
 border: 1px dashed #d9d9d9;  
 border-radius: 6px;  
 cursor: pointer;  
 position: relative;  
 overflow: hidden;  
}  
.avatar-uploader .el-upload:hover {  
 border-color: #409eff;  
}  
.avatar-uploader-icon {  
 font-size: 28px;  
 color: #8c939d;  
 width: 178px;  
 height: 178px;  
 line-height: 178px;  
 text-align: center;  
}  
.avatar {  
 width: 178px;  
 height: 178px;  
 display: block;  
}  
</style>