|  |
| --- |
| package cn.edu.usts.cs2022.controller;  import cn.edu.usts.cs2022.pojo.dto.LoginDTO; import cn.edu.usts.cs2022.pojo.po.\*; import cn.edu.usts.cs2022.service.AdminService; import cn.edu.usts.cs2022.utils.JwtUtil; import lombok.RequiredArgsConstructor; import org.springframework.web.bind.annotation.\*;  import java.util.HashMap; import java.util.List; import java.util.Map;  @RestController @RequestMapping("/admin") @RequiredArgsConstructor public class AdminController {    public final AdminService adminService;   @PostMapping("/login")  public Result login(@RequestBody LoginDTO loginDTO) {  String username = loginDTO.getUsername();  String password = loginDTO.getPassword();   Admin admin = adminService.login(username, password);   if (admin == null) {  return Result.*error*("用户名或密码错误");  } else {  Map<String, Object> claims = new HashMap<>();  String token = JwtUtil.*genToken*(claims);  System.*out*.println("登录成功"+token);  return Result.*success*(token);  }  }   */\*\*  \* 查询所有商家  \*/* @GetMapping("/merchant")  public Result<List<Merchant>> getAllMerchant() {  List<Merchant> list = adminService.getAllMerchant();  for (Merchant merchant : list) {  merchant.setPassword("\*\*\*\*\*\*\*\*\*");  }  return Result.*success*(list);  }   }  package cn.edu.usts.cs2022.controller;  import cn.edu.usts.cs2022.pojo.dto.CategoryDTO; import cn.edu.usts.cs2022.pojo.po.Category; import cn.edu.usts.cs2022.pojo.po.Product; import cn.edu.usts.cs2022.pojo.po.Result; import cn.edu.usts.cs2022.service.CategoryService; import lombok.RequiredArgsConstructor; import org.springframework.web.bind.annotation.\*;  import java.util.List;  @RestController @RequestMapping("/category") @RequiredArgsConstructor public class CategoryController {  private final CategoryService categoryService;  */\*\*  \* 新增分类  \*/* @PostMapping("/add")  public Result add(@RequestBody CategoryDTO categoryDTO) {  String categoryName = categoryDTO.getName();  categoryService.add(categoryName);  return Result.success();  }   */\*\*  \* 查询所有分类  \*/* @GetMapping("/all")  public Result<List<Category>> getAllCategory() {  List<Category> categoryList = categoryService.getAllCategory();  return Result.success(categoryList);  }    */\*\*  \* 删除分类  \*/* @DeleteMapping("/{id}")  public Result deleteCategory(@PathVariable Integer id) {  System.out.println(id);  categoryService.deleteCategory(id);  return Result.success();  }   */\*\*  \* 编辑分类  \*/* @PutMapping()  public Result updateCategory(@RequestBody Category category) {  categoryService.updateCategory(category);  return Result.success();  }   */\*\*  \* 根据id查询分类  \*/* @GetMapping("/{id}")  public Result getCategoryById(@PathVariable Integer id) {  String name = categoryService.getCategoryById(id);  return Result.success(name);  }    }  package cn.edu.usts.cs2022.controller;  import cn.edu.usts.cs2022.pojo.po.Result; import cn.edu.usts.cs2022.service.FavouriteService; import cn.edu.usts.cs2022.utils.ThreadLocalUtil; import lombok.RequiredArgsConstructor; import org.apache.ibatis.annotations.Delete; import org.springframework.web.bind.annotation.\*;  import java.util.Map;  @RestController @RequestMapping("/favourite") @RequiredArgsConstructor public class FavouriteController {   private final FavouriteService favouriteService;  @PostMapping("/{id}")  public Result favourite(@PathVariable("id") Integer productId) {  Map<String, Object> map = ThreadLocalUtil.get();  Integer userId = (Integer) map.get("userId");  favouriteService.favourite(userId, productId);  return Result.success();  }   @DeleteMapping("/{id}")  public Result cancelFavourite(@PathVariable("id") Integer productId) {  Map<String, Object> map = ThreadLocalUtil.get();  Integer userId = (Integer) map.get("userId");  favouriteService.cancelFavourite(userId, productId);  return Result.success();  } }  package cn.edu.usts.cs2022.controller;  import cn.edu.usts.cs2022.pojo.po.Result; import cn.edu.usts.cs2022.utils.AliOssUtil; import org.springframework.web.bind.annotation.PostMapping; import org.springframework.web.bind.annotation.RestController; import org.springframework.web.multipart.MultipartFile;  import java.util.UUID;  @RestController public class FileUploadController {   @PostMapping("/upload")  public Result<String> upload(MultipartFile file) throws Exception {  String originalFilename = file.getOriginalFilename();  //保证文件名字唯一 使用UUID  String filename = UUID.randomUUID().toString() + originalFilename.substring(originalFilename.lastIndexOf("."));   String url = AliOssUtil.uploadFile(filename, file.getInputStream());  return Result.success(url);  } }  package cn.edu.usts.cs2022.controller;  import cn.edu.usts.cs2022.pojo.dto.\*; import cn.edu.usts.cs2022.pojo.po.Merchant; import cn.edu.usts.cs2022.pojo.po.Result; import cn.edu.usts.cs2022.pojo.vo.MerchantVO; import cn.edu.usts.cs2022.service.MerchantService; import cn.edu.usts.cs2022.utils.JwtUtil; import lombok.RequiredArgsConstructor; import org.springframework.web.bind.annotation.\*;  import java.util.HashMap; import java.util.List; import java.util.Map;  @RestController @RequestMapping("/merchant") @RequiredArgsConstructor public class MerchantController {   private final MerchantService merchantService;   */\*\*  \* 商家登录  \* @param loginDTO  \* @return  \*/* @PostMapping("/login")  public Result login(@RequestBody LoginDTO loginDTO) {  String username = loginDTO.getUsername();  String password = loginDTO.getPassword();   Merchant merchant = merchantService.login(username, password);  if (merchant == null) {  return Result.error("用户名或密码错误");  } else {  Integer status = merchant.getStatus();  if (status == 0) {  return Result.error("商家已被禁用！请联系管理员");  }  Map<String, Object> claims = new HashMap<>();  claims.put("merchantId", merchant.getId());  String token = JwtUtil.genToken(claims);  MerchantDTO merchantDTO = new MerchantDTO();  merchantDTO.setMerchant(merchant);  merchantDTO.setMerchantId(merchant.getId());  merchantDTO.setToken(token);  return Result.success(merchantDTO);  }  }   */\*\*  \* 商家注册  \* @param registerDTO  \* @return  \*/* @PostMapping("/register")  public Result Register(@RequestBody RegisterDTO registerDTO) {  String username = registerDTO.getUsername();  String password = registerDTO.getPassword();  String rePassword = registerDTO.getRePassword();   if (!password.equals(rePassword)) {  return Result.error("两次密码输入不一致!");  }  Merchant merchant = merchantService.getMerchantByUsername(username);  if (merchant != null) {  return Result.error("用户名已存在");  }  merchantService.register(username, password);  return Result.success();  }   */\*\*  \* 查询所有商家  \*/* @GetMapping  public Result<List<Merchant>> getAllMerchants() {  List<Merchant> list = merchantService.getAllMerchant();  return Result.success(list);  }   */\*\*  \* 修改商家状态  \*/* @PutMapping("/status")  public Result ChangeStatus(@RequestBody Merchant merchant){  merchantService.changeStatus(merchant);  return Result.success();  }    @GetMapping("/getByProductId/{id}")  public Result<MerchantVO> getByProductId(@PathVariable("id") Integer id) {  MerchantVO merchantVO = merchantService.getByProductId(id);  return Result.success(merchantVO);  }   @GetMapping("{id}")  public Result<Merchant> getById(@PathVariable("id") Integer id) {  Merchant merchant = merchantService.getById(id);  return Result.success(merchant);  }   @PostMapping("/updatePassword")  public Result updatePassword(@RequestBody UpdatePasswordDTO updatePasswordDTO) {  String newPassword = updatePasswordDTO.getPassword();  String reNewPassword = updatePasswordDTO.getRePassword();  if (!newPassword.equals(reNewPassword)) {  return Result.error("两次密码不一致!");  }  merchantService.updatePassword(newPassword);  return Result.success();  }   @PutMapping("/update")  public Result update(@RequestBody UserUpdateDTO userUpdateDTO) {  merchantService.update(userUpdateDTO);  return Result.success();  }  }  package cn.edu.usts.cs2022.controller;  import cn.edu.usts.cs2022.pojo.dto.OrderDTO; import cn.edu.usts.cs2022.pojo.po.Order; import cn.edu.usts.cs2022.pojo.po.Result; import cn.edu.usts.cs2022.pojo.vo.OrderVO; import cn.edu.usts.cs2022.service.OrderService; import lombok.RequiredArgsConstructor; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RestController;  import org.springframework.web.bind.annotation.\*;  import java.util.List;  @RestController @RequestMapping("/order") @RequiredArgsConstructor public class OrderController {   private final OrderService orderService;   */\*\*  \* 查询所有订单  \*/* @GetMapping("/all")  public Result<List<Order>> selectAllOrder() {  List<Order> list = orderService.selectAllOrder();  return Result.success(list);  }  @PostMapping("/add")  public Result<Order> add(@RequestBody OrderDTO orderDTO) {  Order order = orderService.add(orderDTO);  return Result.success(order);  }   @PostMapping("/cancel/{number}")  public Result cancel(@PathVariable("number") String orderNumber) {  orderService.cancel(orderNumber);  return Result.success();  }   @PostMapping("/pay/{orderNumber}")  public Result pay(@PathVariable("orderNumber") String orderNumber) {  orderService.pay(orderNumber);  return Result.success();  }   */\*\*  \* 查询带有价格的订单信息  \*/* @GetMapping()  public Result<List<OrderVO>> selectOrderWithPrice(){  List<OrderVO> list = orderService.selectOrderWithPrice();  return Result.success(list);  }  }  package cn.edu.usts.cs2022.controller;   import cn.edu.usts.cs2022.pojo.dto.ProductDTO; import cn.edu.usts.cs2022.pojo.po.Product; import cn.edu.usts.cs2022.pojo.po.Result; import cn.edu.usts.cs2022.service.ProductService; import lombok.RequiredArgsConstructor;  import org.springframework.web.bind.annotation.\*;  import java.util.List;  @RestController @RequestMapping("/product") @RequiredArgsConstructor public class ProductController {   private final ProductService productService;   @PostMapping("/add")  public Result add(@RequestBody ProductDTO productDTO) {  productService.add(productDTO);  return Result.success();  }   @GetMapping("/all")  public Result<List<Product>> getAllProduct() {  List<Product> productList = productService.getAllProduct();  return Result.success(productList);  }   @GetMapping("/all/{merchantId}")  public Result<List<Product>> getAllByMerchantId(@PathVariable("merchantId") Integer merchantId) {  List<Product> productList = productService.getAllByMerchantId(merchantId);  return Result.success(productList);  }   @PutMapping  public Result update(@RequestBody Product product) {  productService.update(product);  return Result.success();  }   @GetMapping("/search")  public Result<List<Product>> search(String searchInfo) {  List<Product> productList = productService.search(searchInfo);  return Result.success(productList);  }   @GetMapping("/{id}")  public Result<Product> getById(@PathVariable("id") Integer id) {  Product product = productService.getById(id);  return Result.success(product);  }   @PostMapping("/getByIds")  public Result<List<Product>> getByIds(@RequestBody List<Integer> ids) {  if (ids.size() == 0) {  return Result.success(List.of());  }  List<Product> products = productService.getByIds(ids);  return Result.success(products);  }   */\*\*  \* 修改商品状态  \* @param product  \* @return  \*/* @PutMapping("/status")  public Result updateStatus(@RequestBody Product product) {  productService.updateStatus(product);  return Result.success();  }  }  package cn.edu.usts.cs2022.controller;  import cn.edu.usts.cs2022.pojo.dto.\*; import cn.edu.usts.cs2022.pojo.po.Favourite; import cn.edu.usts.cs2022.pojo.po.Result; import cn.edu.usts.cs2022.pojo.po.User; import cn.edu.usts.cs2022.service.UserService; import cn.edu.usts.cs2022.utils.JwtUtil; import lombok.RequiredArgsConstructor; import org.springframework.web.bind.annotation.\*;  import java.util.HashMap; import java.util.List; import java.util.Map;  @RestController @RequestMapping("/user") @RequiredArgsConstructor public class UserController {   private final UserService userService;   */\*\*  \* 用户登录  \* @param loginDTO  \* @return  \*/* @PostMapping("/login")  public Result Login(@RequestBody LoginDTO loginDTO) {  String username = loginDTO.getUsername();  String password = loginDTO.getPassword();   User user = userService.login(username, password);  if (user == null) {  return Result.error("用户名或密码错误");  } else {  Integer status = user.getStatus();  if (status == 0) {  return Result.error("用户已被禁用！请联系管理员");  }  Map<String, Object> claims = new HashMap<>();  claims.put("userId", user.getId());  String token = JwtUtil.genToken(claims);  UserDTO userDTO = new UserDTO();  userDTO.setUser(user);  userDTO.setUserId(user.getId());  userDTO.setToken(token);  return Result.success(userDTO);  }  }   @PostMapping("/register")  public Result Register(@RequestBody RegisterDTO registerDTO) {  String username = registerDTO.getUsername();  String password = registerDTO.getPassword();  String rePassword = registerDTO.getRePassword();   if (!password.equals(rePassword)) {  return Result.error("两次密码输入不一致!");  }  User user = userService.getUserByUsername(username);  if (user != null) {  return Result.error("用户名已存在");  }  userService.register(username, password);  return Result.success();  }   */\*\*  \* 查询所有用户  \*get  \*/* @GetMapping()  public Result<List<User>> getUserList() {  System.out.println("查询全部信息");  List<User> users = userService.getUserList();  for (User user : users) {  user.setPassword("\*\*\*\*\*\*\*\*\*");  }  return Result.success(users);  }   */\*\*  \* 修改用户状态  \*/* @PutMapping()  public Result changeUserStatus(@RequestBody User user) {  userService.changeUserStatus(user);  return Result.success("修改成功");  }   @GetMapping("/myFavourite/{id}")  public Result<List<Favourite>> getMyFavourite(@PathVariable("id") Integer userId) {  List<Favourite> favourites = userService.getMyFavourite(userId);  return Result.success(favourites);  }   @PostMapping("/countOrder")  public Result<Integer> countOrder(@RequestBody CountOrderDTO countOrderDTO) {  Integer count = userService.countOrder(countOrderDTO);  return Result.success(count);  }  @GetMapping("/countFavourite/{id}")  public Result<Integer> countFavourite(@PathVariable("id") Integer userId) {  Integer count = userService.countFavourite(userId);  return Result.success(count);  }   @PostMapping("/updatePassword")  public Result updatePassword(@RequestBody UpdatePasswordDTO updatePasswordDTO) {  String newPassword = updatePasswordDTO.getPassword();  String reNewPassword = updatePasswordDTO.getRePassword();  if (!newPassword.equals(reNewPassword)) {  return Result.error("两次密码不一致!");  }  userService.updatePassword(newPassword);  return Result.success();  }   @PutMapping("/update")  public Result update(@RequestBody UserUpdateDTO userUpdateDTO) {  userService.update(userUpdateDTO);  return Result.success();  }   @GetMapping("/{id}")  public Result<User> getById(@PathVariable("id") Integer id) {  User user = userService.getById(id);  return Result.success(user);  } }  package cn.edu.usts.cs2022.interceptor;   import cn.edu.usts.cs2022.utils.JwtUtil; import cn.edu.usts.cs2022.utils.ThreadLocalUtil; import org.springframework.stereotype.Component; import org.springframework.web.servlet.HandlerInterceptor;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.util.Map;  @Component public class LoginInterceptor implements HandlerInterceptor {  @Override  public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {  //令牌验证  String token = request.getHeader("Authorization");  try {  Map<String, Object> claims = JwtUtil.parseToken(token);  //把业务数据存储到ThreadLocal中  ThreadLocalUtil.set(claims);  //放行  return true;  } catch (Exception e) {  response.setStatus(401);  //不放行  return false;  }  } }  package cn.edu.usts.cs2022.mapper;   import cn.edu.usts.cs2022.pojo.po.Admin; import cn.edu.usts.cs2022.pojo.po.Category; import cn.edu.usts.cs2022.pojo.po.Merchant; import cn.edu.usts.cs2022.pojo.po.User; import org.apache.ibatis.annotations.\*;  import java.util.List;  @Mapper public interface AdminMapper {   @Select("select \* from admin where username = #{username} and password = #{password}")  Admin login(@Param("username") String username,  @Param("password") String password);     */\*\*  \* 查询所有商家  \* @return  \*/* @Select("select \* from merchant")  List<Merchant> selectMerchant();   }  package cn.edu.usts.cs2022.mapper;  import cn.edu.usts.cs2022.pojo.po.Category; import org.apache.ibatis.annotations.\*;  import java.util.List;  @Mapper public interface CategoryMapper {  @Insert("insert into category(name) values (#{name})")  void add(String name);   */\*\*  \* 查询所有分类  \* @return  \*/* @Select("select \* from category")  List<Category> getAllCategory();    */\*\*  \* 新增分类  \* @param category  \*/* @Insert("insert into category (name) values (#{name})")  void insertCategory(Category category);   */\*\*  \* 删除分类  \* @param id  \*/* @Delete("delete from category where id = #{id}")  void deleteCategory(Integer id);   */\*\*  \* 编辑分类  \* @param category  \*/* @Update("update category set name = #{name} where id = #{id}")  void updatecategory(Category category);   @Select("select name from category where id = #{id}")  String selectById(Integer id); }  package cn.edu.usts.cs2022.mapper;  import lombok.Data; import org.apache.ibatis.annotations.Delete; import org.apache.ibatis.annotations.Mapper; import org.apache.ibatis.annotations.Param;  @Mapper public interface FavouriteMapper {   void favourite(@Param("userId") Integer userId,  @Param("productId") Integer productId);   @Delete("delete from favourite where user\_id = #{userId} and product\_id = #{productId}")  void cancelFavourite(@Param("userId") Integer userId,  @Param("productId")Integer productId); }  package cn.edu.usts.cs2022.mapper;  import cn.edu.usts.cs2022.pojo.dto.UserUpdateDTO; import cn.edu.usts.cs2022.pojo.po.Merchant; import cn.edu.usts.cs2022.pojo.vo.MerchantVO; import org.apache.ibatis.annotations.Mapper; import org.apache.ibatis.annotations.Param; import org.apache.ibatis.annotations.Select; import org.apache.ibatis.annotations.Update;  import java.util.List;  @Mapper public interface MerchantMapper {  @Select("select \* from merchant where username = #{username} and password = #{password}")  Merchant login(@Param("username") String username,  @Param("password") String password);   @Select("select \* from merchant where username = #{username}")  Merchant getMerchantByUsername(@Param("username") String username);   void register(Merchant merchant);   */\*\*  \* 查询所有商家  \* @return  \*/* @Select("SELECT \* from merchant")  List<Merchant> selectAllMerchant();   @Update("update merchant set status = #{status} where id = #{id}")  void changeStatus(Merchant merchant);   MerchantVO getByProductId(@Param("id") Integer id);   @Select("select \* from merchant where id = #{id}")  Merchant getById(@Param("id") Integer id);   @Update("update merchant set password = #{newPassword} where id = #{userId}")  void updatePassword(@Param("userId") Integer userId,  @Param("newPassword") String newPassword);   void update(UserUpdateDTO userUpdateDTO); }  package cn.edu.usts.cs2022.mapper;  import cn.edu.usts.cs2022.pojo.po.Order; import cn.edu.usts.cs2022.pojo.vo.OrderVO; import org.apache.ibatis.annotations.Mapper; import org.apache.ibatis.annotations.Param; import org.apache.ibatis.annotations.Select; import org.apache.ibatis.annotations.Update;  import java.util.List;  @Mapper public interface OrderMapper {   */\*\*  \* 查询所有订单信息  \*  \* @return  \*/* @Select("SELECT \* from `order` ")  List<Order> selectAllOrder();   int add(Order order);    Order getById(Integer id);   @Update("update `order` set status = -1 where order\_number = #{orderNumber}")  void cancel(@Param("orderNumber") String orderNumber);   @Select("select \* from `order` where order\_number = #{orderNumber}")  Order getByNumber(@Param("orderNumber") String orderNumber);   @Update("update `order` set status = 1 where order\_number = #{orderNumber}")  void pay(String orderNumber);   List<OrderVO> selectOrderWithPrice(); }  package cn.edu.usts.cs2022.mapper;  import cn.edu.usts.cs2022.pojo.po.Product; import org.apache.ibatis.annotations.Mapper; import org.apache.ibatis.annotations.Param; import org.apache.ibatis.annotations.Select; import org.apache.ibatis.annotations.Update;  import java.util.List;  @Mapper public interface ProductMapper {  void add(Product product);   @Select("select \* from product")  List<Product> getAllProduct();   @Select("select \* from product where merchant\_id = #{merchantId}")  List<Product> getAllByMerchantId(Integer merchantId);   void update(Product product);   List<Product> search(String searchInfo);   @Select("select \* from product where id = #{id}")  Product getById(@Param("id") Integer id);   List<Product> getByIds(@Param("ids") List<Integer> ids);   */\*\*  \* 修改商品状态  \* @param product  \*/* @Update("update product set status = #{status} where id = #{id}")  void updateStatus(Product product); }  package cn.edu.usts.cs2022.mapper;  import cn.edu.usts.cs2022.pojo.dto.CountOrderDTO; import cn.edu.usts.cs2022.pojo.dto.UserUpdateDTO; import cn.edu.usts.cs2022.pojo.po.Favourite; import cn.edu.usts.cs2022.pojo.po.User; import org.apache.ibatis.annotations.Mapper; import org.apache.ibatis.annotations.Param; import org.apache.ibatis.annotations.Select; import org.apache.ibatis.annotations.Update;  import java.util.List;  @Mapper public interface UserMapper {   @Select("select \* from user where username = #{username} and password = #{password}")  User login(@Param("username") String username,  @Param("password") String password);   @Select("select \* from user where username = #{username}")  User getUserByUserName(String username);   void register(User user);   */\*\*  \* 获取全部用户信息  \* @return  \*/* @Select("select \* from user")  List<User> selectAllUsers();   */\*\*  \* 修改用户状态  \* @param user  \*/* @Update("UPDATE user set status = #{status} where id = #{id}")  void changeStatus(User user);   @Select("select \* from favourite where user\_id = #{userId}")  List<Favourite> getMyFavourite(Integer userId);   @Select("select count(\*) from favourite where user\_id = #{userId}")  Integer countFavourite(Integer userId);   @Update("update user set password = #{newPassword} where id = #{userId}")  void updatePassword(@Param("userId") Integer userId,  @Param("newPassword") String newPassword);   void update(UserUpdateDTO userUpdateDTO);   @Select("select \* from user where id = #{id}")  User getById(Integer id);   Integer countOrder(CountOrderDTO countOrderDTO); }  package cn.edu.usts.cs2022.pojo.dto;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class CategoryDTO {  private String name; }  package cn.edu.usts.cs2022.pojo.dto;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class CountOrderDTO {  private Integer userId;   private Integer status; }  package cn.edu.usts.cs2022.pojo.dto;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class LoginDTO {  private String username;  private String password; }  package cn.edu.usts.cs2022.pojo.dto;  import cn.edu.usts.cs2022.pojo.po.Merchant; import cn.edu.usts.cs2022.pojo.po.User; import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class MerchantDTO {  private Integer merchantId;   private Merchant merchant;   private String token; }  package cn.edu.usts.cs2022.pojo.dto;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class OrderDTO {  private Integer userId;   private Integer productId;   private Integer merchantId; }  package cn.edu.usts.cs2022.pojo.dto;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  import java.math.BigDecimal;  @Data @AllArgsConstructor @NoArgsConstructor public class ProductDTO {  private String name;   private BigDecimal price;   private Integer categoryId;   private String video;   private String image;   private String source;   private String description; }  package cn.edu.usts.cs2022.pojo.dto;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class RegisterDTO {  private String username;   private String password;   private String rePassword; }  package cn.edu.usts.cs2022.pojo.dto;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class UpdatePasswordDTO {  private String password;   private String rePassword; }  package cn.edu.usts.cs2022.pojo.dto;   import cn.edu.usts.cs2022.pojo.po.User; import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class UserDTO {  private Integer userId;   private User user;   private String token; }  package cn.edu.usts.cs2022.pojo.dto;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class UserUpdateDTO {  private Integer id;   private String avatar;   private String username;    private String phone; }  package cn.edu.usts.cs2022.pojo.po;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class Admin {  private Integer id;   private String username;   private String password; }  package cn.edu.usts.cs2022.pojo.po;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class Category {  private Integer id;   private String name; }  package cn.edu.usts.cs2022.pojo.po;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class Favourite {  private Integer id;   private Integer userId;   private Integer productId; }  package cn.edu.usts.cs2022.pojo.po;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  import java.time.LocalDateTime;  @Data @AllArgsConstructor @NoArgsConstructor public class Merchant {  private Integer id;   private String username;   private String password;   private String avatar;   private String phone;   private Integer status;   private LocalDateTime createTime;   private LocalDateTime updateTime; }  package cn.edu.usts.cs2022.pojo.po;   import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  import java.time.LocalDateTime;  @Data @AllArgsConstructor @NoArgsConstructor public class Order {  private Integer id;   private String orderNumber;   private Integer userId;   private Integer productId;   private Integer merchantId;   private Integer status; // (已取消: -1, 待支付: 0, 已完成: 1)   private LocalDateTime createTime; }  package cn.edu.usts.cs2022.pojo.po;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  import java.math.BigDecimal; import java.time.LocalDateTime;  @Data @AllArgsConstructor @NoArgsConstructor public class Product {  private Integer id;   private String name;   private BigDecimal price;   private Integer merchantId;   private Integer categoryId;   private String video;   private String image;   private String source;   private Integer status;   private String description;   private LocalDateTime createTime;   private LocalDateTime updateTime; }  package cn.edu.usts.cs2022.pojo.po;   import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  //统一响应结果 @NoArgsConstructor @AllArgsConstructor @Data public class Result<T> {  private Integer code;//业务状态码 0-成功 1-失败  private String message;//提示信息  private T data;//响应数据   //快速返回操作成功响应结果(带响应数据)  public static <E> Result<E> success(E data) {  return new Result<>(0, "操作成功", data);  }   //快速返回操作成功响应结果  public static Result success() {  return new Result(0, "操作成功", null);  }   public static Result error(String message) {  return new Result(1, message, null);  } }  package cn.edu.usts.cs2022.pojo.po;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  import java.time.LocalDateTime;  @Data @AllArgsConstructor @NoArgsConstructor public class User {  private Integer id;   private String username;   private String password;   private String avatar;   private String phone;   private Integer status;   private LocalDateTime createTime;   private LocalDateTime updateTime; }  package cn.edu.usts.cs2022.pojo.vo;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class MerchantVO {  private Integer id;   private String username;   private String avatar; }  package cn.edu.usts.cs2022.pojo.vo;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  import java.math.BigDecimal; import java.time.LocalDateTime;  @Data @AllArgsConstructor @NoArgsConstructor public class OrderVO {  //订单id  private Integer id;  //订单编号  private String orderNumber;  //购买用户id  private Integer userId;  //产品id  private Integer productId;  //商家id  private Integer merchantId;  //购买状态  private Integer status; // (已取消: -1, 待支付: 0, 已完成: 1)  //创建时间  private LocalDateTime createTime;  //商品价格  private BigDecimal price; }  package cn.edu.usts.cs2022.service.impl;  import cn.edu.usts.cs2022.mapper.AdminMapper; import cn.edu.usts.cs2022.pojo.po.Admin; import cn.edu.usts.cs2022.pojo.po.Category; import cn.edu.usts.cs2022.pojo.po.Merchant; import cn.edu.usts.cs2022.pojo.po.User; import cn.edu.usts.cs2022.service.AdminService; import lombok.RequiredArgsConstructor; import org.springframework.stereotype.Service;  import java.util.List;  @Service @RequiredArgsConstructor public class AdminServiceImpl implements AdminService {   public final AdminMapper adminMapper;  @Override  public Admin login(String username, String password) {  return adminMapper.login(username, password);  }       */\*\*  \* 获取所有商家  \* @return  \*/* @Override  public List<Merchant> getAllMerchant() {  return adminMapper.selectMerchant();  }   }  package cn.edu.usts.cs2022.service.impl;  import cn.edu.usts.cs2022.mapper.CategoryMapper; import cn.edu.usts.cs2022.pojo.po.Category; import cn.edu.usts.cs2022.service.CategoryService; import lombok.RequiredArgsConstructor; import org.springframework.stereotype.Service;  import java.util.List;  @Service @RequiredArgsConstructor public class CategoryServiceImpl implements CategoryService {   private final CategoryMapper categoryMapper;  @Override  public void add(String categoryName) {  categoryMapper.add(categoryName);  }   */\*\*  \* 获取全部分类  \* @return  \*/* @Override  public List<Category> getAllCategory() {  return categoryMapper.getAllCategory();  }   */\*\*  \* 新增分类  \* @param category  \*/* @Override  public void insertCategory(Category category) {  categoryMapper.insertCategory(category);  }   */\*\*  \* 删除分类  \* @param id  \*/* @Override  public void deleteCategory(Integer id) {  //TODO  //添加删除限制  /\* adminMapper.selectProductByCategoryId(id)  if()\*/  categoryMapper.deleteCategory(id);  }   */\*\*  \* 编辑分类  \* @param category  \*/* @Override  public void updateCategory(Category category) {  categoryMapper.updatecategory(category);  }   */\*\*  \* 根据id查询分类名称  \* @param id  \* @return  \*/* @Override  public String getCategoryById(Integer id) {  String name = categoryMapper.selectById(id);  return name;  } }  package cn.edu.usts.cs2022.service.impl;  import cn.edu.usts.cs2022.mapper.FavouriteMapper; import cn.edu.usts.cs2022.service.FavouriteService; import lombok.RequiredArgsConstructor; import org.springframework.stereotype.Service;  @Service @RequiredArgsConstructor public class FavouriteServiceImpl implements FavouriteService {   private final FavouriteMapper favouriteMapper;  @Override  public void favourite(Integer userId, Integer productId) {  favouriteMapper.favourite(userId, productId);  }   @Override  public void cancelFavourite(Integer userId, Integer productId) {  favouriteMapper.cancelFavourite(userId, productId);  } }  package cn.edu.usts.cs2022.service.impl;  import cn.edu.usts.cs2022.mapper.MerchantMapper; import cn.edu.usts.cs2022.mapper.ProductMapper; import cn.edu.usts.cs2022.pojo.dto.UserUpdateDTO; import cn.edu.usts.cs2022.pojo.po.Merchant; import cn.edu.usts.cs2022.pojo.vo.MerchantVO; import cn.edu.usts.cs2022.service.MerchantService; import cn.edu.usts.cs2022.utils.ThreadLocalUtil; import lombok.RequiredArgsConstructor; import org.springframework.stereotype.Service;  import java.time.LocalDateTime; import java.util.List; import java.util.Map;  @Service @RequiredArgsConstructor public class MerchantServiceImpl implements MerchantService {  private final MerchantMapper merchantMapper;  @Override  public Merchant login(String username, String password) {  return merchantMapper.login(username, password);  }   @Override  public Merchant getMerchantByUsername(String username) {  return merchantMapper.getMerchantByUsername(username);  }   @Override  public void register(String username, String password) {  Merchant merchant = new Merchant();  merchant.setUsername(username);  merchant.setPassword(password);  merchant.setStatus(1);  merchant.setCreateTime(LocalDateTime.*now*());  merchant.setUpdateTime(LocalDateTime.*now*());  merchantMapper.register(merchant);  }    */\*\*  \* 查询所有商家  \* @return  \*/* @Override  public List<Merchant> getAllMerchant() {  return merchantMapper.selectAllMerchant();  }   */\*\*  \* 修改商家状态  \* @param merchant  \*/* @Override  public void changeStatus(Merchant merchant) {  merchantMapper.changeStatus(merchant);  }   @Override  public MerchantVO getByProductId(Integer id) {  return merchantMapper.getByProductId(id);  }   @Override  public Merchant getById(Integer id) {  return merchantMapper.getById(id);  }   @Override  public void updatePassword(String newPassword) {  Map<String, Object> map = ThreadLocalUtil.*get*();  Integer userId = (Integer) map.get("merchantId");  merchantMapper.updatePassword(userId, newPassword);  }   @Override  public void update(UserUpdateDTO userUpdateDTO) {  merchantMapper.update(userUpdateDTO);  }  }  package cn.edu.usts.cs2022.service.impl;  import cn.edu.usts.cs2022.mapper.OrderMapper;  import cn.edu.usts.cs2022.pojo.dto.OrderDTO; import cn.edu.usts.cs2022.pojo.po.Order; import cn.edu.usts.cs2022.pojo.vo.OrderVO; import cn.edu.usts.cs2022.service.OrderService; import lombok.RequiredArgsConstructor; import org.springframework.beans.BeanUtils; import org.springframework.stereotype.Service;  import java.time.LocalDateTime; import java.util.List;   @Service @RequiredArgsConstructor public class OrderServiceImpl implements OrderService {   private final OrderMapper orderMapper;   */\*\*  \* 查询所有订单信息  \* @return  \*/* @Override  public List<Order> selectAllOrder() {  return orderMapper.selectAllOrder();  }   @Override  public Order add(OrderDTO orderDTO) {  Order order = new Order();  BeanUtils.copyProperties(orderDTO, order);  String orderNumber = String.valueOf(System.currentTimeMillis());  order.setOrderNumber(orderNumber);  order.setStatus(0);  order.setCreateTime(LocalDateTime.now());  orderMapper.add(order);   System.out.println(order.getId());  return order;  }   @Override  public void cancel(String orderNumber) {  orderMapper.cancel(orderNumber);  }   @Override  public void pay(String orderNumber) {  orderMapper.pay(orderNumber);  }   @Override  public List<OrderVO> selectOrderWithPrice() {  return orderMapper.selectOrderWithPrice();  } }  package cn.edu.usts.cs2022.service.impl;  import cn.edu.usts.cs2022.mapper.ProductMapper; import cn.edu.usts.cs2022.pojo.dto.ProductDTO; import cn.edu.usts.cs2022.pojo.po.Product; import cn.edu.usts.cs2022.service.ProductService; import cn.edu.usts.cs2022.utils.ThreadLocalUtil; import lombok.RequiredArgsConstructor; import org.springframework.beans.BeanUtils; import org.springframework.stereotype.Service;  import java.time.LocalDateTime; import java.util.List; import java.util.Map;  @Service @RequiredArgsConstructor public class ProductServiceImpl implements ProductService {   private final ProductMapper productMapper;   @Override  public void add(ProductDTO productDTO) {  Product product = new Product();   BeanUtils.*copyProperties*(productDTO, product);  Map<String, Object> map = ThreadLocalUtil.*get*();  Integer merchantId = (Integer) map.get("merchantId");  System.*out*.println(merchantId);  product.setMerchantId(merchantId);  product.setStatus(0); // 表示待审核  product.setCreateTime(LocalDateTime.*now*());  product.setUpdateTime(LocalDateTime.*now*());  productMapper.add(product);  }   @Override  public List<Product> getAllProduct() {  return productMapper.getAllProduct();  }   @Override  public List<Product> getAllByMerchantId(Integer merchantId) {  return productMapper.getAllByMerchantId(merchantId);  }   @Override  public void update(Product product) {  productMapper.update(product);  }   @Override  public List<Product> search(String searchInfo) {  List<Product> productList = productMapper.search(searchInfo);  return productList;  }   @Override  public Product getById(Integer id) {  return productMapper.getById(id);  }   @Override  public List<Product> getByIds(List<Integer> ids) {  return productMapper.getByIds(ids);  }   @Override  public void updateStatus(Product product) {  productMapper.updateStatus(product);  } }  package cn.edu.usts.cs2022.service.impl;  import cn.edu.usts.cs2022.mapper.UserMapper; import cn.edu.usts.cs2022.pojo.dto.CountOrderDTO; import cn.edu.usts.cs2022.pojo.dto.UserUpdateDTO; import cn.edu.usts.cs2022.pojo.po.Favourite; import cn.edu.usts.cs2022.pojo.po.User; import cn.edu.usts.cs2022.service.UserService; import cn.edu.usts.cs2022.utils.ThreadLocalUtil; import lombok.RequiredArgsConstructor; import org.springframework.stereotype.Service;  import java.time.LocalDateTime; import java.util.List; import java.util.Map;  @Service @RequiredArgsConstructor public class UserServiceImpl implements UserService {   private final UserMapper userMapper;   */\*\*  \* 用户登录  \*  \* @param username  \* @param password  \* @return  \*/* @Override  public User login(String username, String password) {  return userMapper.login(username, password);  }   @Override  public User getUserByUsername(String username) {  return userMapper.getUserByUserName(username);  }   @Override  public void register(String username, String password) {  User user = new User();  user.setUsername(username);  user.setPassword(password);  user.setStatus(1);  user.setCreateTime(LocalDateTime.now());  user.setUpdateTime(LocalDateTime.now());  userMapper.register(user);  }   */\*\*  \* 获取全部用户  \*  \* @return  \*/* @Override  public List<User> getUserList() {  return userMapper.selectAllUsers();  }   */\*\*  \* 修改用户状态  \* @param user  \*/* @Override  public void changeUserStatus(User user) {  userMapper.changeStatus(user);  }   @Override  public List<Favourite> getMyFavourite(Integer userId) {  return userMapper.getMyFavourite(userId);  }   @Override  public Integer countFavourite(Integer userId) {  return userMapper.countFavourite(userId);  }   @Override  public void updatePassword(String newPassword) {  Map<String, Object> map = ThreadLocalUtil.get();  Integer userId = (Integer) map.get("userId");  userMapper.updatePassword(userId, newPassword);  }   @Override  public void update(UserUpdateDTO userUpdateDTO) {  userMapper.update(userUpdateDTO);  }   @Override  public User getById(Integer id) {  return userMapper.getById(id);  }   @Override  public Integer countOrder(CountOrderDTO countOrderDTO) {  return userMapper.countOrder(countOrderDTO);  } }  package cn.edu.usts.cs2022.service;  import cn.edu.usts.cs2022.pojo.po.Admin; import cn.edu.usts.cs2022.pojo.po.Category; import cn.edu.usts.cs2022.pojo.po.Merchant; import cn.edu.usts.cs2022.pojo.po.User;  import java.util.List;  public interface AdminService {  Admin login(String username, String password);    */\*\*  \* 获取所有商家  \* @return  \*/* List<Merchant> getAllMerchant();    }  package cn.edu.usts.cs2022.service;  import cn.edu.usts.cs2022.pojo.po.Category;  import java.util.List;  public interface CategoryService {  void add(String categoryName);    */\*\*  \* 获取所有分类  \* @return  \*/* List<Category> getAllCategory();   */\*\*  \* 添加新的分类  \* @param category  \*/* void insertCategory(Category category);   */\*\*  \* 删除分类  \*/* void deleteCategory(Integer id);   */\*\*  \* 编辑分类  \* @param category  \*/* void updateCategory(Category category);    String getCategoryById(Integer id); }  package cn.edu.usts.cs2022.service;  public interface FavouriteService {  void favourite(Integer userId, Integer productId);   void cancelFavourite(Integer userId, Integer productId); }  package cn.edu.usts.cs2022.service;  import cn.edu.usts.cs2022.pojo.dto.UserUpdateDTO; import cn.edu.usts.cs2022.pojo.po.Merchant; import cn.edu.usts.cs2022.pojo.vo.MerchantVO;  import java.util.List;  public interface MerchantService {  Merchant login(String username, String password);   Merchant getMerchantByUsername(String username);   void register(String username, String password);   */\*\*  \* 查询所有商家  \* @return  \*/* List<Merchant> getAllMerchant();   */\*\*  \* 修改商家状态  \* @param merchant  \*/* void changeStatus(Merchant merchant);   MerchantVO getByProductId(Integer id);   Merchant getById(Integer id);   void updatePassword(String newPassword);   void update(UserUpdateDTO userUpdateDTO); }  package cn.edu.usts.cs2022.service;  import cn.edu.usts.cs2022.pojo.dto.OrderDTO; import cn.edu.usts.cs2022.pojo.po.Order; import cn.edu.usts.cs2022.pojo.vo.OrderVO;  import java.util.List;  public interface OrderService {  */\*\*  \* 查询所有订单信息  \* @return  \*/* List<Order> selectAllOrder();   Order add(OrderDTO orderDTO);   void cancel(String orderNumber);   void pay(String orderNumber);   */\*\*  \* 获取带有价格的订单  \* @return  \*/* List<OrderVO> selectOrderWithPrice(); }  package cn.edu.usts.cs2022.service;  import cn.edu.usts.cs2022.pojo.dto.ProductDTO; import cn.edu.usts.cs2022.pojo.po.Product; import org.apache.ibatis.annotations.Param;  import java.util.List;  public interface ProductService {  void add(ProductDTO productDTO);   List<Product> getAllProduct();   List<Product> getAllByMerchantId(Integer merchantId);   void update(Product product);   List<Product> search(String searchInfo);   Product getById(Integer id);   List<Product> getByIds(@Param("ids") List<Integer> ids);   */\*\*  \* 修改商品状态  \* @param product  \*/* void updateStatus(Product product); }  package cn.edu.usts.cs2022.service;  import cn.edu.usts.cs2022.pojo.dto.CountOrderDTO; import cn.edu.usts.cs2022.pojo.dto.UserUpdateDTO; import cn.edu.usts.cs2022.pojo.po.Favourite; import cn.edu.usts.cs2022.pojo.po.User;  import java.util.List;   public interface UserService {  User login(String username, String password);   User getUserByUsername(String username);   void register(String username, String password);   */\*\*  \* 获取全部用户信息  \* @return  \*/* List<User> getUserList();   */\*\*  \* 修改用户状态  \* @param user  \*/* void changeUserStatus(User user);   List<Favourite> getMyFavourite(Integer userId);   Integer countFavourite(Integer userId);   void updatePassword(String newPassword);   void update(UserUpdateDTO userUpdateDTO);   User getById(Integer id);   Integer countOrder(CountOrderDTO countOrderDTO); }  package cn.edu.usts.cs2022.utils;  import com.aliyun.oss.ClientException; import com.aliyun.oss.OSS; import com.aliyun.oss.OSSClientBuilder; import com.aliyun.oss.OSSException; import com.aliyun.oss.model.PutObjectRequest; import com.aliyun.oss.model.PutObjectResult;  import java.io.InputStream; public class AliOssUtil {   private static final String *ENDPOINT* = "https://oss-cn-hangzhou.aliyuncs.com";  private static final String *ACCESS\_KEY\_ID* = "";  private static final String *ACCESS\_KEY\_SECRET* = "";  private static final String *BUKET\_NAME* = "software-mall";  public static String uploadFile(String objectName, InputStream in) throws Exception {  // 创建OSSClient实例。  OSS ossClient = new OSSClientBuilder().build(*ENDPOINT*, *ACCESS\_KEY\_ID*, *ACCESS\_KEY\_SECRET*);  String url = "";  try {  // 填写字符串。  String content = "Hello OSS，你好世界";  // 创建PutObjectRequest对象。  PutObjectRequest putObjectRequest = new PutObjectRequest(*BUKET\_NAME*, objectName, in);   // 如果需要上传时设置存储类型和访问权限，请参考以下示例代码。  // ObjectMetadata metadata = new ObjectMetadata();  // metadata.setHeader(OSSHeaders.OSS\_STORAGE\_CLASS, StorageClass.Standard.toString());  // metadata.setObjectAcl(CannedAccessControlList.Private);  // putObjectRequest.setMetadata(metadata);   PutObjectResult result = ossClient.putObject(putObjectRequest);  url = "https://" + *BUKET\_NAME* + "." + *ENDPOINT*.substring(*ENDPOINT*.lastIndexOf("/") + 1) + "/" + objectName;   } catch (OSSException oe) {  System.*out*.println("Caught an OSSException, which means your request made it to OSS, "  + "but was rejected with an error response for some reason.");  System.*out*.println("Error Message:" + oe.getErrorMessage());  System.*out*.println("Error Code:" + oe.getErrorCode());  System.*out*.println("Request ID:" + oe.getRequestId());  System.*out*.println("Host ID:" + oe.getHostId());  } catch (ClientException ce) {  System.*out*.println("Caught an ClientException, which means the client encountered "  + "a serious internal problem while trying to communicate with OSS, "  + "such as not being able to access the network.");  System.*out*.println("Error Message:" + ce.getMessage());  } finally {  if (ossClient != null) {  ossClient.shutdown();  }  }  return url;  } }  package cn.edu.usts.cs2022.utils;  import com.auth0.jwt.JWT; import com.auth0.jwt.algorithms.Algorithm;  import java.util.Date; import java.util.Map;  public class JwtUtil {   private static final String KEY = "Kshqsz";    //接收业务数据,生成token并返回  public static String genToken(Map<String, Object> claims) {  return JWT.create()  .withClaim("claims", claims)  .withExpiresAt(new Date(System.currentTimeMillis() + 1000 \* 60 \* 60 \* 12 \* 12))  .sign(Algorithm.HMAC256(KEY));  }   //接收token,验证token,并返回业务数据  public static Map<String, Object> parseToken(String token) {  return JWT.require(Algorithm.HMAC256(KEY))  .build()  .verify(token)  .getClaim("claims")  .asMap();  }  }  package cn.edu.usts.cs2022.utils;  */\*\*  \* ThreadLocal 工具类  \*/* @SuppressWarnings("all") public class ThreadLocalUtil {  //提供ThreadLocal对象,  private static final ThreadLocal THREAD\_LOCAL = new ThreadLocal();   //根据键获取值  public static <T> T get(){  return (T) THREAD\_LOCAL.get();  }   //存储键值对  public static void set(Object value){  THREAD\_LOCAL.set(value);  }    //清除ThreadLocal 防止内存泄漏  public static void remove(){  THREAD\_LOCAL.remove();  } }  package cn.edu.usts.cs2022;  import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication public class SoftwareMallServerApplication {   public static void main(String[] args) {  SpringApplication.*run*(SoftwareMallServerApplication.class, args);  }  }  <?xml version="1.0" encoding="UTF-8" ?> <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace="cn.edu.usts.cs2022.mapper.FavouriteMapper">    <insert id="favourite">  insert into favourite(user\_id, product\_id)  values (#{userId}, #{productId})  </insert> </mapper>  <?xml version="1.0" encoding="UTF-8" ?> <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace="cn.edu.usts.cs2022.mapper.MerchantMapper">   <insert id="register">  insert into merchant(username, password, avatar, phone, status, create\_time, update\_time )  values (#{username}, #{password}, #{avatar}, #{phone}, #{status}, #{createTime}, #{updateTime})  </insert>  <update id="update">  update merchant  <set>  <if test="avatar != null"> avatar = #{avatar},</if>  <if test="phone != null"> phone = #{phone}, </if>  <if test="true"> update\_time = now()</if>  </set>  <where>  id = #{id}  </where>  </update>   <select id="getByProductId" resultType="cn.edu.usts.cs2022.pojo.vo.MerchantVO">  select merchant.id, merchant.username, merchant.avatar  from merchant  join product on merchant.id = product.merchant\_id  where product.id = #{id}  </select> </mapper>  <?xml version="1.0" encoding="UTF-8" ?> <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace="cn.edu.usts.cs2022.mapper.OrderMapper">   <!-- 插入订单 -->  <insert id="add" parameterType="cn.edu.usts.cs2022.pojo.po.Order" useGeneratedKeys="true" keyProperty="id">  INSERT INTO `order` (order\_number, user\_id, product\_id, merchant\_id, status, create\_time)  VALUES (#{orderNumber}, #{userId}, #{productId}, #{merchantId}, #{status}, #{createTime});  </insert>   <!-- 查询插入后的订单信息 -->  <select id="getById" parameterType="int" resultType="cn.edu.usts.cs2022.pojo.po.Order">  SELECT \* FROM `order` WHERE id = #{id};  </select>   <select id="selectOrderWithPrice" resultType="cn.edu.usts.cs2022.pojo.vo.OrderVO">  select `order`.\*,product.price from `order` left outer join product on order.product\_id = product.id  </select>  </mapper>  <?xml version="1.0" encoding="UTF-8" ?> <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace="cn.edu.usts.cs2022.mapper.ProductMapper">   <insert id="add">  insert into product(name, price, merchant\_id, category\_id, video, image, source, status, description, create\_time, update\_time )  values (#{name}, #{price}, #{merchantId}, #{categoryId}, #{video}, #{image}, #{source}, #{status}, #{description}, #{createTime}, #{updateTime})  </insert>  <update id="update">  update product  <set>  <if test="name != null"> name = #{name}, </if>  <if test="price != null"> price = #{price },</if>  <if test="categoryId != null"> category\_id = #{categoryId},</if>  <if test="video != null"> video = #{video},</if>  <if test="image != null"> image = #{image},</if>  <if test="source != null"> source = #{source },</if>  <if test="description != null"> description = #{description},</if>  <if test="true"> update\_time = now(),</if>  <if test="true"> status = 0 </if>  </set>  <where>  id = #{id}  </where>  </update>  <select id="search" resultType="cn.edu.usts.cs2022.pojo.po.Product">  select \* from product  left join category on product.category\_id = category.id  where product.name like concat('%', #{searchInfo}, '%')  or category.name like concat('%', #{searchInfo}, '%')  </select>  <select id="getByIds" resultType="cn.edu.usts.cs2022.pojo.po.Product">  select \* from product where id in  <foreach collection="ids" item="id" open="(" separator="," close=")">  #{id}  </foreach>  </select> </mapper>  <?xml version="1.0" encoding="UTF-8" ?> <!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace="cn.edu.usts.cs2022.mapper.UserMapper">   <insert id="register">  insert into user(username, password, avatar, phone, status, create\_time, update\_time )  values (#{username}, #{password}, #{avatar}, #{phone}, #{status}, #{createTime}, #{updateTime})  </insert>  <update id="update">  update user  <set>  <if test="avatar != null"> avatar = #{avatar},</if>  <if test="phone != null"> phone = #{phone}, </if>  <if test="true"> update\_time = now()</if>  </set>  <where>  id = #{id}  </where>  </update>  <select id="countOrder" resultType="java.lang.Integer">  select count(\*) from `order` where user\_id = #{userId} and status = #{status}  </select> </mapper>  <script setup>  import { watch } from 'vue';  import { ref } from 'vue'  import { useRouter, useRoute } from 'vue-router';  // 左侧导航项  const menuItems = ref([    {      category: '订单中心',      items: [        { label: '我的订单', route: '/orders' },      ]    },    {      category: '个人中心',      items: [        { label: '我的个人中心', route: '/profile' },        { label: '喜欢的商品', route: '/favorites' },      ]    }  ])  const route = useRoute()  const router = useRouter()  const currentRoute = ref(route.path)  watch(() => route.path, (newPath) => {    currentRoute.value = newPath;  });  // 跳转到指定路由  const navigateTo = (route) => {    router.push(route)    currentRoute.value = route  }  </script>  <template>    <div class="user-center">      <!-- 页面布局 -->      <div class="user-center-layout">        <!-- 左侧导航 -->        <el-card class="sidebar" shadow="never">          <div v-for="(menu, index) in menuItems" :key="index" class="menu-section">            <h3>{{ menu.category }}</h3>            <ul>              <li                v-for="(item, idx) in menu.items"                :key="idx"                :class="{ active: item.route === currentRoute }"                @click="navigateTo(item.route)"              >                {{ item.label }}              </li>            </ul>          </div>        </el-card>        <!-- 右侧内容 -->          <router-view />      </div>    </div>  </template>  <style scoped>  .user-center {    padding: 30px 136px;    background-color: #f5f5f5;    box-sizing: border-box;    min-height: 90vh;  }  .user-center-layout {    display: flex;    gap: 20px;  }  .sidebar {    width: 250px;    background: #fff;  }  .menu-section {    margin-bottom: 20px;  }  .menu-section h3 {    font-size: 16px;    font-weight: bold;    margin-bottom: 10px;  }  .menu-section ul {    list-style: none;    padding: 0;  }  .menu-section li {    padding: 5px 0;    cursor: pointer;    transition: all 0.3s;  }  .menu-section li:hover {    color: #409eff;  }  .content {    flex: 1;    padding: 20px;    }  .menu-section li {    padding: 5px 0;    cursor: pointer;    transition: all 0.3s;    color: #333;  }  .menu-section li.active {    color: #409eff;  }  .menu-section li:hover {    color: #409eff;  }  </style>  <script setup>  import { ref, onMounted } from 'vue'  import { useRouter } from 'vue-router';  import { useUserStore } from '@/stores';  import { userCountFavouriteService, userUpdatePasswordService, userUpdateService,           userGetByIdService} from '@/api/user'  import { Tickets, Edit, Checked} from '@element-plus/icons-vue'  import {userCountOrderService} from '@/api/user'  const userStore = useUserStore()  const editDialogVisible = ref(false) // 控制修改个人信息对话框  const changePasswordDialogVisible = ref(false) // 控制修改密码对话框  const userInfo = ref(userStore.user)  const changePasswordFormRef = ref(null)  const updateAvatar = (response) => {    editedUserInfo.value.avatar = response.data;    ElMessage.success("上传头像成功~")  }  const editedUserInfo = ref({    id: userInfo.value.id,    avatar: userInfo.value.avatar,    username: userInfo.value.username,    phone: userInfo.value.phone  })  // 打开对话框方法  const openEditDialog = () => {    editDialogVisible.value = true;  };  // 关闭对话框方法  const closeEditDialog = async () => {    editDialogVisible.value = false;  };  const getUser = async () => {    const res = await userGetByIdService(userInfo.value.id);    userStore.setUser(res.data.data);    userInfo.value = userStore.user;  }  const save = async () => {    await userUpdateService(editedUserInfo.value);    await getUser();    ElMessage.success("修改个人信息成功~");    editDialogVisible.value = false;  }  // 打开修改密码对话框  const openChangePasswordDialog = () => {    changePasswordDialogVisible.value = true  }  const checkPasswordMatch = (rule, value, callback) => {    if (value !== changePasswordForm.value.newPassword) {      callback(new Error('确认密码与新密码不一致'))    } else {      callback()    }  }  // 关闭修改密码对话框  const closeChangePasswordDialog = () => {    changePasswordForm.value = {      oldPassword: '',      newPassword: '',      rePassword: ''    }    changePasswordDialogVisible.value = false  }  const changePasswordForm = ref({    oldPassword: '',    newPassword: '',    rePassword: ''  })  const changePasswordRules = {    oldPassword: [      { required: true, message: '请输入旧密码', trigger: 'blur' }    ],    newPassword: [      { required: true, message: '请输入新密码', trigger: 'blur' },    ],    rePassword: [      { required: true, message: '请确认新密码', trigger: 'blur' },      { validator: checkPasswordMatch, trigger: 'blur' }    ]  }  const handleChangePassword = async () => {    // 这里可以调用修改密码的API接口    const passwordDTO = {      password: changePasswordForm.value.newPassword,      rePassword: changePasswordForm.value.rePassword    };    await userUpdatePasswordService(passwordDTO);    await getUser();    ElMessage.success('密码修改成功~')    changePasswordForm.value = {      oldPassword: '',      newPassword: '',      rePassword: ''    }    closeChangePasswordDialog()  }  const handleSubmitChangePassword = () => {      // 校验旧密码是否正确    if (changePasswordForm.value.oldPassword !== userInfo.value.password) {      ElMessage.error('旧密码不正确');      return; // 如果旧密码不正确，直接返回    }    // 表单校验    const form = changePasswordFormRef.value    form.validate((valid) => {      if (valid) {        handleChangePassword()      } else {        ElMessage.error('请填写正确的密码信息')      }    })  }  const stats = ref([    { label: '我的订单', count: 0, path: '/orders' },    { label: '待支付的订单', count: 0, path: '/orders' },    { label: '喜欢的商品', count: 1, path: '/favorites' },    { label: '已完成的订单', count: 0, path: '/orders' },  ])  onMounted( async () => {    await countFavourite();    await countOrder();  })  const countOrder = async () =>{    const userId = userInfo.value.id;    const res\_cancel = await userCountOrderService(      {        userId: userId,        status: -1      }    );    const res\_ready = await userCountOrderService(      {        userId: userId,        status: 0      }    );    const res\_finish= await userCountOrderService(      {        userId: userId,        status: 1      }    );    const cancel = res\_cancel.data.data;    const ready = res\_ready.data.data;    const finish = res\_finish.data.data;    const all = cancel + ready + finish;    stats.value[0].count = all;    stats.value[1].count = ready;    stats.value[3].count = finish;  }  const countFavourite = async () => {    const userId = userInfo.value.id;    const res = await userCountFavouriteService(userId);    stats.value[2].count = res.data.data;  }  const router = useRouter()  // 获取统计项颜色  const getStatColor = (index) => {    const colors = ['#FF7F00', '#409EFF', '#FF0000', '#67C23A']    return colors[index % colors.length]  }  const goToDetail = (stat) => {    router.push(stat.path)  }  </script>  <template>    <!-- 右侧内容 -->    <el-card class="content" shadow="never">      <!-- 用户信息和账号安全 -->      <div class="user-info-container">        <!-- 用户信息 -->        <div class="user-info">          <img class="avatar " :src="userInfo.avatar ? userInfo.avatar: 'https://via.placeholder.com/100' " alt="用户头像" />          <div>            <h2>{{ userInfo.username }}</h2>            <p> 您好~ </p>            <a href="#" class="edit-link" @click.prevent="openEditDialog">修改个人信息 &gt;</a>          </div>        </div>        <!-- 账号安全信息 -->        <div class="account-security">          <p>绑定手机：{{ userInfo.phone }}</p>        </div>      </div>      <hr style="margin-top: 40px; opacity: 0.4">        <!-- 数据统计 -->      <div class="stats">        <div          v-for="(stat, index) in stats"          :key="index"          class="stat-item"          @click="goToDetail(stat)"        >          <el-card class="stat-icon" shadow = "never":style="{ backgroundColor: stat.color || getStatColor(index) }" >            <el-icon>              <Tickets v-if="index==0"></Tickets>              <Edit v-if="index==1"></Edit>              <Checked v-if="index==3"></Checked>            </el-icon>            <i :class="'far fa-heart'" v-if="index===2" style="padding-bottom: 30px;"></i>          </el-card>          <div class="stat-info">            <h3>{{ stat.label }}</h3>            <p>{{ stat.count }}</p>            <a href="#" class="stat-link">查看{{ stat.label }}</a>          </div>        </div>      </div>          <!-- 修改个人信息对话框 -->        <el-dialog        v-model="editDialogVisible"        title="修改个人信息"        width="500px"        @close="closeEditDialog"      >        <el-form label-width="80px">          <!-- 修改头像 -->          <el-form-item label="头像">            <el-upload              action="/api/upload"              :on-success="updateAvatar"              :show-file-list="false"            >              <!-- 如果有头像，显示头像 -->              <img v-if="editedUserInfo.avatar" :src="editedUserInfo.avatar" class="uploaded-avatar" />                <!-- 如果没有头像，显示文字提示 -->              <div v-else class="avatar-placeholder">                点击上传头像              </div>            </el-upload>          </el-form-item>          <!-- 用户名（只读） -->          <el-form-item label="用户名">            <el-input v-model="editedUserInfo.username" placeholder="用户名" disabled></el-input>          </el-form-item>          <!-- 修改电话 -->          <el-form-item label="手机号">            <el-input v-model="editedUserInfo.phone" placeholder="请输入手机号"></el-input>          </el-form-item>        </el-form>        <template #footer>          <el-button type="text" @click="openChangePasswordDialog" style="margin-right: 200px; color: #409EFF;">修改密码</el-button>          <el-button @click="closeEditDialog">取消</el-button>          <el-button type="primary" @click="save">保存</el-button>        </template>      </el-dialog>     <!-- 修改密码对话框 -->      <el-dialog        v-model="changePasswordDialogVisible"        title="修改密码"        width="400px"        @close="closeChangePasswordDialog"      >        <el-form :model="changePasswordForm" :rules="changePasswordRules" ref="changePasswordFormRef" label-width="100px">          <el-form-item label="旧密码" prop="oldPassword">            <el-input type="password" v-model="changePasswordForm.oldPassword" placeholder="请输入旧密码"></el-input>          </el-form-item>          <el-form-item label="新密码" prop="newPassword">            <el-input type="password" v-model="changePasswordForm.newPassword" placeholder="请输入新密码"></el-input>          </el-form-item>          <el-form-item label="确认密码" prop="rePassword">            <el-input type="password" v-model="changePasswordForm.rePassword" placeholder="请再次输入新密码"></el-input>          </el-form-item>        </el-form>        <template #footer>          <el-button @click="closeChangePasswordDialog">取消</el-button>          <el-button type="primary" @click="handleSubmitChangePassword">确认</el-button>        </template>      </el-dialog>    </el-card>  </template>  <style scoped>  .content {    flex: 1;    padding: 20px;    }  /\* 用户信息与账号安全水平排列 \*/  .user-info-container {    display: flex;    align-items: center;    margin-bottom: 20px;    justify-content: space-between;    padding-left: 120px;  }  .user-info {    display: flex;    align-items: center;    }  .user-info .avatar {    width: 100px;    height: 100px;    border-radius: 50%;    margin-right: 20px;  }  .user-info h2 {    font-size: 24px;    margin: 0 0 10px;  }  .user-info p {    margin: 0 0 10px;    color: #888;  }  .edit-link {    color: #409eff;    text-decoration: none;    font-size: 14px;  }  .account-security p{    margin: 5px 0;    padding-right: 200px;  }  .security-level {    color: #67c23a;  }  .stats {    display: flex;    flex-wrap: wrap;    gap: 20px;    margin-top: 40px;    margin-left: 100px;  }  .stat-item {    cursor: pointer;    display: flex;    align-items: center;    width: calc(40% - 20px);    padding: 20px;    background: #fff;    border-radius: 8px;    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);    height: 210px;    padding-left: 20px;  }  .stat-item:hover {    transform: scale(1.05);  /\* 放大卡片 \*/    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);  /\* 添加阴影效果 \*/  }  .stat-icon {    width: 50px;    height: 50px;    border-radius: 50%;    display: flex;    justify-content: center;    align-items: center;    color: #fff;    font-size: 24px;    margin-right: 10px;  }  .stat-info h3 {    margin: 0;    font-size: 16px;    color: #333;  }  .stat-info p {    margin: 5px 0;    font-size: 24px;    color: #333;  }  .stat-link {    font-size: 14px;    color: #409eff;    text-decoration: none;  }  /\* 设置头像为圆形 \*/  .uploaded-avatar {    width: 100px;    /\* 设置宽度 \*/    height: 100px;   /\* 设置高度 \*/    border-radius: 50%;  /\* 圆形 \*/    object-fit: cover;   /\* 保持比例填充 \*/  }  /\* 头像空缺时的占位符样式 \*/  .avatar-placeholder {    width: 100px;    /\* 设置宽度 \*/    height: 100px;   /\* 设置高度 \*/    display: flex;    align-items: center;    justify-content: center;    border-radius: 50%;  /\* 圆形 \*/    background-color: #f5f5f5;  /\* 背景色 \*/    color: #606266;    /\* 文字颜色 \*/    font-size: 14px;    /\* 字体大小 \*/    cursor: pointer;   /\* 鼠标为指针 \*/    border: 1px dashed #dcdfe6;  /\* 边框 \*/    transition: background-color 0.3s; /\* 背景颜色变化 \*/  }  /\* 鼠标悬停时改变背景色 \*/  .avatar-placeholder:hover {    background-color: #e6f7ff;  }  .stat-item i.far.fa-heart {    border: none;    box-shadow: none;  }  </style>  <script setup>  import { ref, onMounted } from 'vue';  import { useUserStore } from '@/stores'; // 用户store（可根据需要修改）  import { productGetByIdService } from '@/api/product'; // 获取产品详情的API  import { orderAllService } from '@/api/order'; // 获取订单API  import { merchantGetByIdService } from '@/api/merchant'; // 获取商家信息API  import { useRouter } from 'vue-router';  const userStore = useUserStore();  const router = useRouter();  const user = userStore.user;  const orders = ref([]); // 存储订单数据  const filteredOrders = ref([]); // 用于存储筛选后的订单  const currentPage = ref(1); // 当前分页页码  const pageSize = ref(5); // 每页展示数量  // 筛选条件  const orderStatus = ref('all'); // 默认显示所有订单  // 获取订单列表并填充商品和商家信息  const getOrders = async () => {    const res = await orderAllService();    let rawOrders = res.data.data.filter(item => item.userId === user.id);    // 并行获取商品和商家信息    const enrichedOrders = await Promise.all(      rawOrders.map(async (order) => {        const [productRes, merchantRes] = await Promise.all([          productGetByIdService(order.productId),          merchantGetByIdService(order.merchantId),        ]);        return {          ...order,          productImage: productRes.data.data.image,          productName: productRes.data.data.name,          merchantName: merchantRes.data.data.username,        };      })    );    orders.value = enrichedOrders;    console.log('Orders:', orders.value); // 调试输出所有订单    applyFilter(); // 获取订单后应用筛选条件  };  // 筛选订单状态  const applyFilter = () => {    if (orderStatus.value === 'all') {      filteredOrders.value = orders.value;    } else {      filteredOrders.value = orders.value.filter(order => order.status === parseInt(orderStatus.value));    }    console.log('Filtered Orders:', filteredOrders.value); // 调试输出筛选后的订单  };  // 处理状态变化，更新筛选后的订单  const handleStatusChange = () => {    applyFilter();  };  // 分页改变时触发  const handlePageChange = (page) => {    currentPage.value = page;  };  // 支付操作  const handlePay = (order) => {    console.log('支付订单', order);    router.push({      path: '/payment',      query: {        order: JSON.stringify(order), // 序列化对象      },    });  };  // 下载操作  const handleDownload = async (order) => {    const res = await productGetByIdService(order.productId);    const product = res.data.data;    const url = product.source;      // 创建一个隐藏的 <a> 元素    const link = document.createElement('a');    link.href = url;    link.download = product.name; // 可选，设置下载文件的默认文件名    //link.target = '\_blank'; // 可选，避免干扰用户当前页面    document.body.appendChild(link); // 将 <a> 添加到文档中    link.click(); // 模拟点击下载    document.body.removeChild(link); // 下载完成后移除 <a>    console.log('下载软件', order);  };  // 重新查看商品操作  const handleViewProduct = (order) => {    router.push(`/productDetail/${order.productId}`)    console.log('查看商品', order);  };  // 获取订单状态的文本描述  const getOrderStatus = (status) => {    switch (status) {      case 0:        return '待支付';      case 1:        return '已完成';      case -1:        return '已取消';      default:        return '未知状态';    }  };  // 页面加载时获取数据  onMounted(() => {    getOrders();  });  </script>  <template>    <el-card class="content" shadow="never">      <h3>我的订单</h3>      <hr style="margin-top: 40px; opacity: 0.4" />        <!-- 筛选订单 -->      <el-select v-model="orderStatus" placeholder="筛选订单" @change="handleStatusChange" style="width: 200px; margin-bottom: 20px;">        <el-option label="全部" value="all"></el-option>        <el-option label="待支付" value="0"></el-option>        <el-option label="已完成" value="1"></el-option>        <el-option label="已取消" value="-1"></el-option>      </el-select>        <el-table :data="filteredOrders.slice((currentPage-1) \* pageSize, currentPage \* pageSize)" style="width: 100%">        <el-table-column label="商品图片" :width="  100">          <template #default="{ row }">            <img :src="row.productImage" alt="商品图片" style="width: 80px; height: 80px; object-fit: cover; display: block;" />          </template>        </el-table-column>        <el-table-column label="订单号" prop="orderNumber" />        <el-table-column label="商品名" prop="productName" />        <el-table-column label="商家名" prop="merchantName" />        <el-table-column label="订单状态" :formatter="(row) => getOrderStatus(row.status)" />        <el-table-column label="操作">          <template #default="{ row }">            <el-button v-if="row.status === 0" @click="handlePay(row)" size="mini" type="primary">去支付</el-button>            <el-button v-if="row.status === 1" @click="handleDownload(row)" size="mini" type="success">下载软件</el-button>            <el-button v-if="row.status === -1" @click="handleViewProduct(row)" size="mini" type="warning">查看商品</el-button>          </template>        </el-table-column>      </el-table>        <!-- 分页 -->      <div style="text-align: center; margin-top: 20px;">        <el-pagination          :current-page="currentPage"          :page-size="pageSize"          :total="filteredOrders.length"          layout="prev, pager, next"          @current-change="handlePageChange"        ></el-pagination>      </div>    </el-card>  </template>  <style scoped>  .no-data {    text-align: center;    color: #999;    font-size: 18px;    margin-top: 50px;  }  </style>  <script setup>  import { ref, onMounted } from 'vue';  import { userGetFavouriteService } from '@/api/user';  import { useUserStore } from '@/stores';  import { productGetByIdsService } from '@/api/product';  import { categoryGetAllService } from '@/api/category';  const categories = ref([]);  const userStore = useUserStore();  const user = userStore.user;  const favourites = ref([]);  // 当前页码  const currentPage = ref(1);  // 每页展示的数量  const itemsPerPage = ref(8);  // 分页后的结果  const pagedResults = ref([]);  const getFavourites = async () => {    const userId = user.id;    const res = await userGetFavouriteService(userId);    favourites.value = res.data.data;    favourites.value = favourites.value.map(item => item.productId);    const temp = favourites.value;    const res\_product = await productGetByIdsService(temp);    favourites.value = res\_product.data.data;    updatePagedResults();  // 更新分页数据  };  // 获取分类名称  const getCategoryName = (id) => {    const category = categories.value.find(item => item.id === id);    return category ? category.name : "未知分类";  };  // 更新分页数据  const updatePagedResults = () => {    const start = (currentPage.value - 1) \* itemsPerPage.value;    const end = start + itemsPerPage.value;    pagedResults.value = favourites.value.slice(start, end);  };  // 监听页码变化  const pageChange = (page) => {    currentPage.value = page;    updatePagedResults();  };  onMounted(async () => {    const category\_res = await categoryGetAllService();    categories.value = category\_res.data.data;    await getFavourites();  });  </script>  <template>    <el-card class="content" shadow="never">      <h3>我喜欢的商品</h3>      <hr style="margin-top: 40px; opacity: 0.4">      <div class="search-result-page">        <div v-if="favourites.length === 0" class="no-data">          <p>暂无喜欢的产品，快去收藏吧~</p>        </div>        <div v-else class="product-list">          <ProductCard            v-for="product in pagedResults"            :key="product.id"            :product="product"            :categoryName="getCategoryName(product.categoryId)"            :style="{ width: '250px' }"          />        </div>        <!-- 分页组件 -->        <el-pagination          v-if="favourites.length > 0"          :current-page="currentPage"          :page-size="itemsPerPage"          :total="favourites.length"          layout="prev, pager, next"          @current-change="pageChange"          class="pagination"        />      </div>    </el-card>  </template>  <style scoped>  .no-data {    text-align: center;    color: #999;    font-size: 18px;    margin-top: 50px;  }  .product-list {    display: flex;    flex-wrap: wrap;    gap: 10px; /\* 设置卡片之间的间距 \*/    justify-content: left; /\* 每一行的卡片居中 \*/  }  .content {    flex: 1;    padding: 20px;  }  .pagination {    padding-left: 45%;    text-align: center;    margin-top: 20px;  }  </style>  <script setup>  import { useRoute } from 'vue-router';  // 导入 useRoute  import { productGetByIdService } from '@/api/product';  import { onMounted, ref } from 'vue';  import { categoryGetAllService } from '@/api/category'  import { useUserStore } from '@/stores';  import { userGetFavouriteService } from '@/api/user'  import { favouriteAddService, favouriteDeleteService} from '@/api/favourite'  import { merchantGeyByProductIdService } from '@/api/merchant.js'  import { orderAddService } from '@/api/order'  import { useRouter } from 'vue-router';  // 使用 useRoute 获取当前路由信息  const route = useRoute();  const userStore = useUserStore();  const userId = userStore.user.id;  const router = useRouter();  const confirmOrder = async () => {    try {      // 调用订单服务      const res = await orderAddService({        userId,        productId,        merchantId: merchantInfo.value.id,      });      const order = res.data.data;      ElMessage.success('订单创建成功！即将跳转到支付页面...');      showDialog.value = false;      // 模拟跳转到支付页面      setTimeout(() => {        router.push({          path: '/payment',          query: {            order: JSON.stringify(order), // 序列化对象          },        });        //window.location.href = '/payment'; // 替换为实际的支付页面路径      }, 1000);    } catch (error) {      ElMessage.error('订单创建失败，请稍后再试！');    }  };  const categories = ref([]);  // 定义响应式变量  const product = ref({});  const productId = route.params.id;  const merchantInfo = ref({    id: '',    username: '商家名称',  // 假设商家的用户名    avatar: 'https://via.placeholder.com/100'  // 假设商家的头像  });  const showDialog = ref(false); // 控制对话框是否显示  const orderInfo = ref({    name: "",    image: "",    price: 0,    categoryName: "",    merchantName: "",  });  const showOrderDialog = () => {    orderInfo.value = {      name: product.value.name,      image: product.value.image,      price: product.value.price,      categoryName: getCategoryName(product.value.categoryId),      merchantName: merchantInfo.value.username,    };    showDialog.value = true;  };  const getMerchant = async () => {    const res = await merchantGeyByProductIdService(productId);    merchantInfo.value = res.data.data;  }  const likedProducts = ref([]);  // 获取商品分类名称  const getCategoryName = (id) => {    const category = categories.value.find(item => item.id === id);    return category ? category.name : "未知分类";  };  const getFavourites = async () => {    const favourite\_res = await userGetFavouriteService(userId);    likedProducts.value = favourite\_res.data.data.map(item => item.productId);  }  // 获取商品信息和分类  onMounted(async () => {    const res = await productGetByIdService(productId);    product.value = res.data.data;    const category\_res = await categoryGetAllService();    categories.value = category\_res.data.data;    getFavourites();    getMerchant();  });  const has = () => {    for (var i = 0; i < likedProducts.value.length; i++) {      if (likedProducts.value[i] == productId) {        return true;      }    }    return false;  }  // 切换商品的喜欢状态  const toggleLike = async (productId) => {    if (has()) {      // 如果已存在，则删除该商品ID      await favouriteDeleteService(productId);      await getFavourites();      ElMessage.success("取消收藏成功~")    } else {      // 如果不存在，则添加该商品ID      await favouriteAddService(productId);      await getFavourites();      ElMessage.success("收藏成功~")    }  };  </script>  <template>    <div class="product-detail-page">      <!-- 商品展示区域 -->      <div class="product-main">        <div class="product-image">          <img :src="product.image" alt="Product Image" class="product-main-image" />        </div>        <div class="product-info">          <h2 class="product-title">{{ product.name }}</h2>          <p class="category">{{ getCategoryName(product.categoryId) }}</p>          <p class="product-price">￥{{ product.price }}</p>          <p class="product-description-title">商品描述</p>          <p class="product-description">{{ product.description }}</p>          <div class="product-actions">            <button class="buy-button" @click="showOrderDialog">立即购买</button>            <!-- 喜欢按钮，修改样式使其与立即购买按钮一致 -->            <button              :class="{'like-button': true, 'liked': has(productId)}"              @click.stop="toggleLike(productId)"            >              <i                :class="has(productId) ? 'fas fa-heart' : 'far fa-heart'"                class="like-icon"              ></i>              喜欢            </button>          </div>        </div>      </div>      <hr style="margin-top: 50px;">      <div class="merchant-info">        <h3>商家:</h3>        <img :src="merchantInfo.avatar ? merchantInfo.avatar: 'https://via.placeholder.com/100'" alt="Merchant Avatar" class="merchant-avatar" />        <p class="merchant-username">{{ merchantInfo.username }}</p>      </div>      <!-- 视频播放器 -->      <div v-if="product.video" class="video-player-section">        <h3>演示视频</h3>        <video controls class="video-player">          <source :src="product.video" type="video/mp4" />          您的浏览器不支持视频播放。        </video>      </div>      <el-dialog        v-model="showDialog"        title="确认订单"        width="500px"        :close-on-click-modal="false"      >        <div class="dialog-content">          <img :src="orderInfo.image" alt="Product Image" class="dialog-image" />          <p><strong>商品名称：</strong>{{ orderInfo.name }}</p>          <p><strong>分类名称：</strong>{{ orderInfo.categoryName }}</p>          <p><strong>商家名称：</strong>{{ orderInfo.merchantName }}</p>          <p><strong>价格：</strong>￥{{ orderInfo.price }}</p>        </div>        <template #footer>          <el-button @click="showDialog = false">取消</el-button>          <el-button type="primary" @click="confirmOrder">确认下单</el-button>        </template>      </el-dialog>    </div>  </template>  <style scoped>  .dialog-content {    text-align: left;    font-size: 16px;  }  .dialog-image {    width: 100px;    height: 100px;    border-radius: 6px;    object-fit: cover;    margin-bottom: 20px;  }  .category {    font-size: 18px;    color: #777;    margin-top: 5px;    font-style: italic;  }  .product-detail-page {    padding: 20px;    max-width: 1200px;    margin: 0 auto;  }  .product-main {    display: flex;    gap: 100px;    margin-top: 20px;    margin-left: 100px;  }  .product-image {    flex: 1;    display: flex;    justify-content: center;    width: 100%;  }  .product-main-image {    width: 550px;    height: 550px;    object-fit: cover;  }  .product-info {    flex: 2;    display: flex;    flex-direction: column;    margin-top: 20px;  /\* 增加商品描述与图片的间距 \*/  }  .product-title {    font-size: 28px;    font-weight: bold;    margin-bottom: 10px;    align-self: flex-start;  /\* 对齐到左侧 \*/  }  .product-price {    font-size: 24px;    color: #f44336;    font-weight: 600;    align-self: flex-start;  /\* 对齐到左侧 \*/  }  .product-description-title {    font-size: 22px;    font-weight: 600;    align-self: flex-start;  /\* 对齐到左侧 \*/  }  .product-description {    font-size: 16px;    color: #555;    margin-bottom: 20px;    align-self: flex-start;  /\* 对齐到左侧 \*/  }  .product-actions {    display: flex;    gap: 20px;    margin-top: 155px;  }  .buy-button,  .like-button {    padding: 12px 30px;    font-size: 18px;    cursor: pointer;    border: none;    border-radius: 6px;    transition: background-color 0.3s;  }  /\* 立即购买按钮样式 \*/  .buy-button {    background-color: #ff9800;  /\* 橙色背景 \*/    color: white;  }  .buy-button:hover {    background-color: #f57c00;  /\* 橙色悬浮 \*/  }  /\* 喜欢按钮样式 \*/  .like-button {    background-color: #d3d3d3;  /\* 默认灰色 \*/    color: white;  }  .like-button:hover {    background-color: #9e9e9e;  /\* 灰色悬浮 \*/  }  /\* 喜欢按钮变红的样式 \*/  .like-button.liked {    background-color: #f44336;  /\* 红色背景 \*/  }  .like-button.liked:hover {    background-color: #d32f2f;  /\* 红色悬浮 \*/  }  /\* 爱心图标的颜色 \*/  .like-icon {    margin-right: 8px;    font-size: 18px;  }  .like-button.liked .like-icon {    color: red;  /\* 红色爱心 \*/  }  .merchant-info {    display: flex;    align-items: center;    gap: 15px;    margin-top: 20px;    margin-bottom: 40px;    margin-left: 100px;  }  .merchant-avatar {    width: 100px;    height: 100px;    border-radius: 50%;    object-fit: cover;  }  .merchant-username {    font-size: 20px;    font-weight: 600;    color: #333;  }  .video-player-section {    text-align: center;    margin-top: 40px;  }  .video-player {    width: 80%;    max-width: 800px;    height: 450px;    border-radius: 8px;    background-color: #000;  }  </style>  <script setup>  import { ref, onMounted } from 'vue'  import { useRoute, useRouter } from 'vue-router';  import { productGetByIdService } from '@/api/product';  import { useUserStore } from '@/stores';  import { merchantGetByIdService } from '@/api/merchant'  import { orderCancelService, orderPayService } from '@/api/order'  // 获取路由参数  const route = useRoute();  const router = useRouter();  const userStore = useUserStore();  const user = userStore.user;  const dialogVisible = ref(false);  const merchant = ref(null)  const product = ref(null)  const price = ref(0)  const order = JSON.parse(route.query.order);  const getMerchant = async () => {    const id = order.merchantId;    const response = await merchantGetByIdService(id);    merchant.value = response.data.data;  }  const pay = async () => {    const number = order.orderNumber;    await orderPayService(number);    ElMessage.success("支付成功~");    dialogVisible.value = true;  }  const getProduct = async () => {    const id = order.productId;    const response = await productGetByIdService(id);    product.value = response.data.data;    price.value = product.value.price;  }  const download = () => {    const url = product.value.source;      // 创建一个隐藏的 <a> 元素    const link = document.createElement('a');    link.href = url;    link.download = product.name; // 可选，设置下载文件的默认文件名    //link.target = '\_blank'; // 可选，避免干扰用户当前页面    document.body.appendChild(link); // 将 <a> 添加到文档中    link.click(); // 模拟点击下载    document.body.removeChild(link); // 下载完成后移除 <a>    router.push("/home")  }  // 解析订单对象  onMounted(async () => {    await getMerchant();    await getProduct();  })  // 取消订单函数（你可以根据需要实现）  const cancelOrder = () => {    ElMessageBox.confirm(      '你确定要取消订单吗？',      '确认',      {        confirmButtonText: 'OK',        cancelButtonText: 'Cancel',        type: 'warning',      }    )      .then(async () => {        await orderCancelService(order.orderNumber);        router.go(-1);        ElMessage({          type: 'success',          message: '订单取消成功',        })      })  }  </script>  <template>    <div class="payment-page">      <!-- 页面标题 -->      <h1 class="title">订单支付</h1>      <!-- 订单信息展示 -->      <el-card class="order-card" shadow="always">        <div class="order-info">          <el-row gutter={20}>            <!-- 左侧商品图片，只有在 product 数据加载完成后才渲染 -->            <el-col :span="8" class="product-image" v-if="product && product.image">              <img :src="product.image" alt="商品图片" class="product-img" />            </el-col>            <!-- 右侧订单信息 -->            <el-col :span="16">              <el-row>                <el-col :span="6"><strong>订单号：</strong></el-col>                <el-col :span="18">{{ order.orderNumber }}</el-col>              </el-row>              <el-row>                <el-col :span="6"><strong>用户名：</strong></el-col>                <el-col :span="18">{{ user.username }}</el-col>              </el-row>              <el-row>                <el-col :span="6"><strong>商品名：</strong></el-col>                <el-col :span="18">{{ product ? product.name : '加载中...' }}</el-col>              </el-row>              <el-row>                <el-col :span="6"><strong>商品价格：</strong></el-col>                <el-col :span="18">{{ price }}</el-col>              </el-row>              <el-row>                <el-col :span="6"><strong>商家名：</strong></el-col>                <el-col :span="18">{{ merchant ? merchant.username : '加载中...' }}</el-col>              </el-row>              <el-row>                <el-col :span="6"><strong>订单状态：</strong></el-col>                <el-col :span="18">{{ order.status === 0 ? '待支付' : '已完成' }}</el-col>              </el-row>            </el-col>          </el-row>        </div>        <!-- 支付和取消按钮区域 -->        <div class="payment-actions">          <el-button type="danger" size="large" class="cancel-button" @click="cancelOrder">            取消订单          </el-button>          <el-button type="primary" size="large" class="pay-button" @click="pay">            <i class="fab fa-weixin wechat-icon" style="padding-top: 10px;"></i> 使用微信支付          </el-button>        </div>      </el-card>                  <!-- 支付成功后弹窗 -->      <el-dialog v-model="dialogVisible" title="支付成功" width="400px">        <p>支付成功！现在可以下载软件。</p>        <div class="download-actions">          <el-button type="primary" size="large" @click="download">            下载软件          </el-button>        </div>      </el-dialog>    </div>  </template>  <style scoped>  /\* 弹窗的样式 \*/  .download-actions {    text-align: center;    margin-top: 20px;  }  .payment-page {    padding: 20px;    max-width: 800px;    margin: 0 auto;    text-align: center;  }  .title {    margin-bottom: 20px;    font-size: 32px;    font-weight: bold;    color: #333;  }  .order-card {    display: flex;    flex-direction: column;    margin-bottom: 40px;  }  .order-info {    font-size: 16px;    line-height: 1.8;  }  .product-image {    text-align: center;  }  .product-img {    width: 100%;    height: auto;    max-height: 200px;    object-fit: cover;    border-radius: 10px;  }  .payment-actions {    margin-left: 30%;    margin-top: 20px;    display: flex;    justify-content: center;    gap: 20px;  }  .pay-button {    background-color: #1aad19;    color: white;    font-size: 18px;    display: flex;    justify-content: center;    align-items: center;    padding: 15px 40px;    border-radius: 30px;  }  .pay-button:hover {    background-color: #128a13;  }  .wechat-icon {    width: 30px;    height: 30px;    margin-right: 10px;  }  .cancel-button {    background-color: #f56c6c;    color: white;    font-size: 18px;    padding: 15px 40px;    border-radius: 30px;  }  .cancel-button:hover {    background-color: #f44336;  }  </style>  <script setup>  import { userLoginService, userRegisterService } from '@/api/user'  import { ref, watch } from 'vue'  import { User, Lock } from '@element-plus/icons-vue'  import { useUserStore } from '@/stores';  import { useRouter } from 'vue-router';  const isRegister = ref(false)  const form = ref()  const register = async () => {    await form.value.validate()    await userRegisterService(formModel.value)    ElMessage.success("注册成功~")    isRegister.value = false  }  const userStore = useUserStore()  const router = useRouter()  const login = async () => {    await form.value.validate()    const res = await userLoginService(formModel.value)    userStore.setToken(res.data.data.token)    userStore.setUsername(res.data.data.user.username)    userStore.setUser(res.data.data.user)    ElMessage.success("登录成功~")    router.push('/home')  }  // register  const formModel = ref({    username: '',    password: '',    rePassword: ''  })  watch(isRegister, () => {    if (!isRegister.value) {      formModel.value.username = formModel.value.username;      formModel.value.password = '';    } else {      formModel.value = {      username: '',      password: '',      rePassword: ''    }    }  })  const rules = {    username: [      { required: true, message: '请输入用户名', trigger: 'blur' }, // check when lost focus      { min: 3, max: 10, message: '用户名必须为3~10位', trigger: 'blur' }    ],    password: [      { required: true, message: '请输入密码', trigger: 'blur' },      { pattern: /^\S{3,15}$/, message: '密码必须为3-15位非空字符', trigger: 'blur' }    ],    rePassword: [      { required: true, message: '请输入密码', trigger: 'blur' },      { pattern: /^\S{3,15}$/, message: '密码必须为3-15位非空字符', trigger: 'blur' },      {        validator: (rule, value, callback) => {          if (value !== formModel.value.password) {            callback(new Error('两次密码输入不一致'))          } else {            callback();          }        },        trigger: 'blur'      }    ]  }  </script>  <template>    <el-row class="login-page">      <el-col :span= 12 class="bg"></el-col>      <el-col :span="6" :offset="3" class="form">        <el-card class="box-card">          <el-form :model="formModel" :rules="rules" ref="form" size="large" autocomplete="off" v-if="isRegister">            <el-form-item>              <h1>软件商城注册</h1>            </el-form-item>            <el-form-item prop="username">              <el-input v-model="formModel.username" :prefix-icon="User" placeholder="请输入用户名"></el-input>            </el-form-item>            <el-form-item prop="password">              <el-input v-model="formModel.password" :prefix-icon="Lock" type="password" placeholder="请输入密码"></el-input>            </el-form-item>            <el-form-item prop="rePassword">              <el-input v-model="formModel.rePassword" :prefix-icon="Lock" type="password" placeholder="请输入再次密码" @keydown.enter="register"></el-input>            </el-form-item>            <el-form-item>              <el-button @click="register" class="button" type="primary" auto-insert-space> 注册 </el-button>            </el-form-item>            <el-form-item class="flex">              <el-link type="info" :underline="false" @click="isRegister = false"> ← 返回 </el-link>            </el-form-item>          </el-form>          <el-form :model="formModel" :rules="rules" ref="form" size="large" autocomplete="off" v-else>            <el-form-item>              <h1>软件商城登录</h1>            </el-form-item>            <el-form-item prop="username">              <el-input v-model="formModel.username" :prefix-icon="User" placeholder="请输入用户名"></el-input>            </el-form-item>            <el-form-item prop="password">              <el-input v-model="formModel.password" name="password" :prefix-icon="Lock" type="password"                placeholder="请输入密码" @keydown.enter="login">              </el-input>            </el-form-item>            <el-form-item class="flex">            </el-form-item>            <el-form-item>              <el-button @click="login" class="button" type="primary" auto-insert-space>登录</el-button>            </el-form-item>            <el-form-item class="flex">              <el-link type="info" :underline="false" @click="isRegister = true"> 注册 → </el-link>            </el-form-item>          </el-form>        </el-card>      </el-col>    </el-row>  </template>  <style lang="scss" scoped>  .login-page {    height: 98vh;    background-color: #fff;    .bg {      background:        //url('@/assets/logo2.png') no-repeat 60% center / 240px auto,        url('@/assets/bk.png') no-repeat center / cover;      border-radius: 0 20px 20px 0;    }    .form {      display: flex;      flex-direction: column;      justify-content: center;      user-select: none;      .title {        margin: 0 auto;      }      .button {        width: 100%;      }      .flex {        width: 100%;        display: flex;        justify-content: space-between;      }    }  }  </style>  <script setup>  import { ref } from 'vue'  import { useRouter } from 'vue-router';  import { ArrowDown, Search } from '@element-plus/icons-vue'  import { useUserStore } from '@/stores';  import { ElMessage } from 'element-plus';  const searchInfo = ref('')  // 顶部导航文字  const dialogTableVisible = ref(false)  const navItems = ref([    { name: "首页", path: "/home", isSearch: false },    { name: "管理系统", path: "", isSearch: true },    { name: "开发工具", path: "", isSearch: true },    { name: "操作系统", path: "", isSearch: true },    { name: "设计工具", path: "", isSearch: true },    { name: "办公与协作", path: "", isSearch: true },  ]);  // const userStore = useUserStore();  // if (userStore.token != '') {  //   navItems.value.push("个人中心")  // }  const router = useRouter()  const userStore = useUserStore()  const username = userStore.username  const user = ref(userStore.user)  const search = (keyword = '') => {    searchInfo.value = keyword || searchInfo.value;    router.push({ path: '/searchResult', query: { query: searchInfo.value, t: Date.now() } });  };  const handleNavClick = (item) => {    if (item.isSearch) {      // 如果是搜索项，调用搜索方法      search(item.name);      searchInfo.value = ''    } else if (item.path) {      // 如果是普通导航项，跳转到指定路径      router.push(item.path);    }  };  const welcome = ref("欢迎你~")  const logout = () => {    userStore.removeToken();    userStore.removeUsername();    router.push("/");    ElMessage.success("退出登录成功~")  }  const goToUserCenter = () => {    router.push("/userCenter")  }  const goToHome = () => {    searchInfo.value = ''    router.push("/home")  }  const gridData = [    { name: '韩守坤', from: '苏州科技大学', email: '2456480538@qq.com'},    { name: '郁竹一', from: '苏州科技大学', email: '2949621931@qq.com'},    { name: '吴纡怀', from: '苏州科技大学', email: '1085406285@qq.com'},    { name: '陈松',   from: '苏州科技大学', email: '2220834872@qq.com'},  ]  </script>  <template>    <el-dialog v-model="dialogTableVisible" title="About us" width="800">      <el-table :data="gridData">        <el-table-column property="name" label="Name" width="200" />        <el-table-column property="from" label="From" />        <el-table-column property="email" label="Contact us"></el-table-column>      </el-table>      <!-- 指导老师信息 -->      <div class="mentor-info">        <span>          指导老师:          <a            class="director"            href="https://eie.usts.edu.cn/info/1120/2941.htm"            target="\_blank"          >            奚雪峰教授          </a>          （电子与信息工程学院）        </span>      </div>    </el-dialog>    <div class="page-layout">      <!-- 顶部导航栏 -->      <el-header height="60px" class="top-nav">        <!-- Logo -->        <img src="../../assets/img/logo.png" alt="" width="60px" style="padding-left: 120px; padding-top: 20px; cursor: pointer;" @click="goToHome">        <h2 style="padding-top: 20px; padding-right: 20px;">软件商城</h2>        <!-- <h1 class="title" @click="goToHome">SoftWare</h1> -->        <!-- 导航文字居中 -->        <div class="nav-items">          <span v-for="(item, index) in navItems" :key="index" @click="handleNavClick(item)">{{ item.name }}</span>          <span @click="dialogTableVisible = true;">关于我们</span>        </div>        <div  class="dropdown-container">          <el-dropdown class="welcome" >            <span> {{ welcome }}</span>          </el-dropdown>          <el-dropdown class="welcome">            <span> {{ username }}</span>          </el-dropdown>          <el-dropdown class="user-dropdown">            <span>              <el-avatar :src="userStore.user.avatar" :size="55">              </el-avatar>             <el-icon>                <arrow-down />              </el-icon>            </span>            <template #dropdown>              <el-dropdown-menu>                <el-dropdown-item @click="goToUserCenter">个人中心</el-dropdown-item>                <el-dropdown-item @click="logout">退出登录</el-dropdown-item>              </el-dropdown-menu>            </template>          </el-dropdown>        </div>        <!-- 搜索框 -->        <el-input          v-model="searchInfo"          placeholder="搜索商品"          :prefix-icon="Search"          @keydown.enter="search()"          class="search-box"          size="small"        ></el-input>      </el-header>      <div>        <router-view></router-view>      </div>    </div>  </template>  <style scoped>  .top-nav {    align-items: center;    padding: 0 20px;    background-color: #ffffff;    margin-bottom: 10px;  }  .title {    padding-top: 15px;    margin-left: 120px;    cursor: pointer;  }  .nav-items {    padding-left: 100px;    padding-top: 15px;    display: flex;    gap: 15px;    font-size: 16px;    justify-content: center;    cursor: pointer;  }  .welcome span {    padding-top: 15px;    cursor: default;    display: inline-flex;    align-items: center;    outline: none !important;    border: none !important;    box-shadow: none !important;    margin-right: 10px;  }  .search-box {    margin-top: 15px;    margin-right: 120px;    width: 250px;    height: 45px;  }  /\* 去除个人中心按钮的边框和轮廓 \*/  .user-dropdown span {    margin-top: 10px;    cursor: pointer;    display: inline-flex;    align-items: center;    outline: none !important;    border: none !important;    box-shadow: none !important;    margin-right: 10px;  }  .el-header {    display: flex;    align-items: center;    justify-content: space-between;    padding: 0 20px;  }  .dropdown-container {    display: flex; /\* 将三个下拉框包裹在一个 flex 容器内 \*/    align-items: center;    gap: 5px; /\* 控制间距 \*/    margin-left: 0;  }  .avatar-icon {    margin-right: 5px; /\* 调整头像和文字间距 \*/  }  .mentor-info {    margin-right: 100px;    margin-top: 10px;    text-align: center;    font-size: 16px;    color: #333;  }  .director {    color: #0073e6;    cursor: pointer;    text-decoration: none; /\* 取消下划线 \*/  }  </style>  <script setup>  import { ref, watch } from 'vue';  import { useRoute } from 'vue-router';  import { productSearchService } from '@/api/product'  import { categoryGetAllService } from '@/api/category'  import { onMounted } from 'vue';  // 获取路由的查询参数  const route = useRoute();  const categories = ref([]);  // 当前页码  const currentPage = ref(1);  // 每页展示的数量  const itemsPerPage = ref(10);  // 搜索查询条件  const searchQuery = ref(route.query.query || '');  // 过滤后的搜索结果  const filteredResults = ref([]);  // 分页后的结果  const pagedResults = ref([]);  // 获取当前分类名称  const getCategoryName = (id) => {    const category = categories.value.find(item => item.id === id);    return category ? category.name : "未知分类";  }  // 过滤函数  const filterResults = async () => {    const res = await productSearchService(searchQuery.value);    filteredResults.value = res.data.data;    filteredResults.value = filteredResults.value.filter(item => item.status === 1);    updatePagedResults();  // 更新分页结果  };  // 更新分页结果  const updatePagedResults = () => {    const start = (currentPage.value - 1) \* itemsPerPage.value;    const end = start + itemsPerPage.value;    pagedResults.value = filteredResults.value.slice(start, end);  };  // 监听 searchQuery 和 route.query.query 变化，确保每次查询变化时都进行过滤  watch(    () => [route.query.query, route.query.t],    () => {      searchQuery.value = route.query.query || ''; // 更新 searchQuery      filterResults(); // 重新过滤    },    { immediate: true } // 页面加载时就调用一次过滤  );  onMounted(async () => {    const category\_res = await categoryGetAllService();    categories.value = category\_res.data.data;  });  </script>  <template>    <div class="search-result-page">      <div v-if="filteredResults.length === 0" class="no-data">        <p>暂无数据，请尝试其他搜索关键词。</p>      </div>      <div v-else class="product-list">        <ProductCard          v-for="product in pagedResults"          :key="product.id"          :product="product"          :categoryName="getCategoryName(product.categoryId)"        />      </div>      <!-- 分页组件 -->      <el-pagination        v-if="filteredResults.length > 0"        :current-page="currentPage"        :page-size="itemsPerPage"        :total="filteredResults.length"        layout="prev, pager, next"        @current-change="pageChange"        class="pagination"      />    </div>  </template>  <style scoped>  .no-data {    text-align: center;    color: #999;    font-size: 18px;    margin-top: 50px;  }  .category {    font-size: 14px;    color: #777;    margin-top: 5px;    font-style: italic;  }  .search-result-page {    padding: 20px;    margin-left: 120px;  }  .product-list {    display: flex;    flex-wrap: wrap;  /\* 允许换行 \*/    gap: 20px;  /\* 设置卡片之间的间距 \*/    justify-content: left;  /\* 每一行的卡片居中 \*/  }  .pagination {    padding-left: 40%;    text-align: center;    margin-top: 20px;  }  </style>  <script setup>  import { ref, onMounted } from 'vue'  import { productGetAllService } from '@/api/product'  import { categoryGetAllService} from '@/api/category'  import { useRouter } from 'vue-router';  import home1 from '@/assets/home1.png'  import home2 from '@/assets/home2.png'  const router = useRouter()  const detail = (id) => {    router.push(`/productDetail/${id}`);  };  const products = ref([])  const categories = ref([]);  const getCategoryName = (id) => {    const category = categories.value.find(item => item.id === id)    return category ? category.name : "未知分类"  }  onMounted( async() => {    const category\_res = await categoryGetAllService();    categories.value = category\_res.data.data;    const res = await productGetAllService();    products.value = res.data.data;    products.value = products.value.filter(item => item.status === 1).slice(0, 5);    getMenuItems();  })  // 图片轮播数据  const imageList = ref([    {id: 2, url: home1},    {id: 3, url: home2}  ])  const menuItems = ref([    { id: 0, title: '', subItems: [] },    { id: 1, title: '', subItems: [] },    { id: 2, title: '', subItems: [] },    { id: 3, title: '', subItems: [] },    { id: 4, title: '', subItems: [] },    { id: 5, title: '', subItems: [] },    { id: 6, title: '', subItems: [] },    { id: 7, title: '', subItems: [] }  ])  const getMenuItems = () => {    // 遍历分类数组，生成菜单项    categories.value.forEach((category, index) => {      // 防止超过菜单的长度      if (index >= 8) return;      // 从 products 中筛选出属于当前分类的商品      const relatedProducts = products.value.filter(        (product) => product.categoryId === category.id      );      // 构造菜单项      menuItems.value[index] = {        id: category.id,        title: category.name, // 分类名称        subItems: relatedProducts.slice(0, 3), // 存储完整的商品对象      };    });  };  // // 侧边栏菜单数据  // const menuItems = ref([  //   { id: 0, title: '开发工具', subItems: ['代码编辑器', 'IDE集成开发环境', '版本控制工具'] },  //   { id: 1, title: '设计工具', subItems: ['平面设计', 'UI/UX设计', '3D建模'] },  //   { id: 2, title: '办公与协作', subItems: ['文档编辑', '项目管理', '在线会议'] },  //   { id: 3, title: '教育与学习', subItems: ['语言学习', '编程课程', '在线教育平台'] },  //   { id: 4, title: '安全与优化', subItems: ['杀毒软件', '系统清理工具', '密码管理器'] },  //   { id: 5, title: '操作系统', subItems: ['Windows', 'Linux', 'macOS'] },  //   { id: 6, title: '企业软件', subItems: ['ERP系统', 'CRM系统', '数据分析工具'] },  //   { id: 7, title: '游戏与娱乐', subItems: ['PC游戏', '模拟器', '多媒体工具'] }  // ]);  </script>  <template>    <el-card class="box-card" shadow="never">      <el-container>        <!-- 左侧分类菜单 -->        <el-aside width="200px" class="menu-bar">          <el-menu default-active="0" class="el-menu-vertical-demo">            <el-sub-menu              v-for="item in menuItems"              :key="item.id"              :index="item.id.toString()"              class="sub\_menu"            >              <template #title>{{ item.title }}</template>              <!-- 展开子菜单项 -->              <el-menu-item                v-for="(subItem, subIndex) in item.subItems"                :key="subIndex"                :index="`${item.id}-${subIndex}`"                @click="detail(subItem.id)"              >              <div class="menu-item-content">                <img :src="subItem.image" alt="Item Image" class="menu-item-image" />                <span>{{ subItem.name }}</span>              </div>              </el-menu-item>            </el-sub-menu>          </el-menu>        </el-aside>        <!-- 主内容区域 -->        <el-main class="main-content">          <!-- 图片轮播 -->          <el-carousel height="510px" indicator-position="outside">            <el-carousel-item v-for="item in imageList" :key="item.id">              <div class="carousel-image-container">                <img :src="item.url" alt="" class="carousel-image" />              </div>            </el-carousel-item>          </el-carousel>        </el-main>      </el-container>      <h3>推荐商品</h3>      <div class="product-list">        <ProductCard          v-for="(item, index) in products"          :key="item.id"          :product="item"          :categoryName="getCategoryName(item.categoryId)"        />      </div>    </el-card>  </template>  <style scoped>  .product-list {    display: flex;    flex-wrap: wrap;  /\* 允许换行 \*/    gap: 14px;  /\* 设置卡片之间的间距 \*/    justify-content: left;  /\* 每一行的卡片居中 \*/  }  .sub\_menu {    background-color: #f5f5f5;  }  .box-card {    padding-left: 40px;    padding-right: 40px;    margin: 0 80px;    border: none; /\* 取消边框 \*/  }  .menu-bar {    background-color: #f5f5f5;    height: 510px;  }  .main-content {    padding: 0;    background-color: #ffffff;  }  /\* 图片轮播的图片容器，保证图片适应容器 \*/  .carousel-image-container {    width: 100%;    height: 100%;  }  .carousel-image {    width: 100%;    height: 100%;    object-fit: cover;  }  .product-item {    padding: 10px;    background-color: #eeeeee;    text-align: center;  }  .menu-item-content {    display: flex;    align-items: center;  /\* 垂直居中对齐图片和文本 \*/    gap: 10px;  /\* 图片和文本之间的间距 \*/  }  .menu-item-image {    width: 40px;   /\* 设置图片宽度 \*/    height: 40px;  /\* 设置图片高度 \*//\* 保持图片的比例，避免拉伸 \*/  }  </style>  import axios from 'axios'  import { useUserStore } from '@/stores'  import { ElMessage } from 'element-plus'  import router from '@/router'  const baseURL = '/api'  const instance = axios.create({      // TODO 1. 基础地址，超时时间      baseURL,      timeout: 10000  })  // 请求拦截器  instance.interceptors.request.use(      (config) => {          // TODO 2. 携带token          const userStore = useUserStore()          if (userStore.token) {              config.headers.Authorization = userStore.token          }          return config      },      (err) => Promise.reject(err)  )  // 响应拦截器  instance.interceptors.response.use(      (res) => {          // TODO 3. 处理业务失败          // TODO 4. 摘取核心响应数据          if (res.data.code === 0) {              return res          }          // 处理业务失败          ElMessage.error(res.data.message || '服务错误')          return Promise.reject(res.data)      },      (err) => {          // TODO 5. 处理401错误          if (err.response?.status === 401) {              router.push("/login")          }          // 错误默认提示          ElMessage.error(err.response.data.message || '服务错误')          return Promise.reject(err)      }  )  export default instance  export { baseURL }  import { defineStore } from 'pinia'  import { ref } from 'vue'  export const useUserStore = defineStore('user', () => {      const token = ref('')      const username = ref('')      const user = ref(null)      const setToken = (newToken) => {          token.value = newToken      }      const setUsername = (newUsername) => {          username.value = newUsername      }      const setUser = (newUser) => {          user.value = newUser      }      const removeToken = () => {          token.value = ''      }      const removeUsername = () => {          username.value = ''      }      const removeUser = () => {          user.value = ''      }      return {          token,          username,          user,          setToken,          setUsername,          setUser,          removeUsername,          removeToken,          setUser        }  }, {      persist: true  })  import { createPinia } from 'pinia'  import persist from 'pinia-plugin-persistedstate'  const pinia = createPinia()  pinia.use(persist)  export default pinia  export \* from './user'  <script setup>  import { useRouter } from 'vue-router';  const props = defineProps({    product: {      type: Object,      required: true,    },    categoryName: {      type: String,      required: true,    },  });  const router = useRouter();  const detail = (id) => {    router.push(`/productDetail/${id}`);  };  </script>  <template>    <el-card      class="product-card"      :body-style="{ padding: '20px' }"      @click="detail(product.id)"    >      <div class="card-header">        <img :src="product.image" alt="product image" class="product-image" />        <div class="product-info">          <h3>{{ product.name }}</h3>          <div class="category">{{ categoryName }}</div>          <div class="price">￥{{ product.price }}</div>        </div>      </div>    </el-card>  </template>  <style scoped>  .product-card {    width: 265px;    transition: transform 0.2s ease, box-shadow 0.2s ease;    cursor: pointer;    margin-bottom: 20px;  }  .product-card:hover {    transform: scale(1.05);    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);  }  .product-image {    width: 200px;    height: 200px;    object-fit: cover;    border-radius: 8px;  }  .product-info {    margin-top: 10px;  }  .category {    font-size: 14px;    color: #777;    margin-top: 5px;    font-style: italic;  }  .price {    font-size: 18px;    font-weight: bold;    margin-top: 10px;  }  </style>  import request from '@/utils/request.js'  // search api  export const productSearchService = (searchInfo) => {      return request.get("/product/search", {params: {searchInfo: searchInfo }});  }  export const productGetByIdService = (id) => {      return request.get(`/product/${id}`)  }  export const productGetAllService = () => {      return request.get('/product/all')  }  export const productGetByIdsService = (ids) => {      return request.post('/product/getByIds', ids);  }  import request from '@/utils/request.js'  export const orderAddService = ({ userId, productId, merchantId}) => {      return request.post("/order/add", {userId, productId, merchantId})  }  export const orderCancelService = (orderNumber) => {      return request.post(`/order/cancel/${orderNumber}`)  }  export const orderAllService = () => {      return request.get("/order/all")  }  export const orderPayService = (orderNumber) => {      return request.post(`/order/pay/${orderNumber}`)  }  import request from '@/utils/request.js'  export const merchantGeyByProductIdService = (id) => {      return request.get(`/merchant/getByProductId/${id}`);  }  export const merchantGetByIdService = (id) => {      return request.get(`/merchant/${id}`);  }  import request from '@/utils/request.js'  export const favouriteAddService = (productId) => {      return request.post(`/favourite/${productId}`);  }  export const favouriteDeleteService = (productId) => {      return request.delete(`/favourite/${productId}`);  }  import request from '@/utils/request.js'  export const categoryGetAllService = () => {      return request.get("/category/all");  }  import { createRouter, createWebHistory } from 'vue-router'  const router = createRouter({    history: createWebHistory(import.meta.env.BASE\_URL),    routes:    [      {        path: '/login',        component: () => import ('@/views/login/LoginPage.vue')      },      {        path: '/home',        component: () => import('@/views/layout/LayoutContainer.vue'),        redirect: '/homePage',        children: [          { path: '/homePage', component: () => import('@/views/home/HomePage.vue')},          { path: '/searchResult', component: () => import('@/views/home/SearchResult.vue')},          { path: '/productDetail/:id', component: () => import('@/views/product/ProductDetail.vue')},          { path: '/payment', component: () => import('@/views/order/Payment.vue')},          {            path: '/userCenter',            component: () => import('@/views/user/UserCenter.vue'),            redirect: '/profile',            children: [              { path: '/profile', component: () => import('@/views/user/Profile.vue')},              { path: '/orders', component: () => import('@/views/user/Orders.vue')},              { path: '/favorites', component: () => import('@/views/user/Favorites.vue')}            ]          }        ]      },      {        path: '/',        redirect: '/login'      },    ],  })  export default router  import request from '@/utils/request.js'  export const categoryAddService = ({name}) => {      return request.post("/category/add", {name});  }  export const categoryGetAllService = () => {      return request.get("/category/all");  }  import request from '@/utils/request.js'  // login api  export const merchantLoginService = ({ username, password }) => {      return request.post("/merchant/login", { username, password });  }  //注册账号 /merchant/regeits  post  export const merchantRegisterService = ({ username, password, rePassword }) => {      return request.post("/merchant/register", { username, password, rePassword });  }  export const  merchantUpdatePasswordService = ({password, rePassword}) => {      return request.post("/merchant/updatePassword", {password, rePassword})  }  export const merchantUpdateService = ({id, avatar, username, phone}) => {      return request.put("/merchant/update", {id, avatar, username, phone})  }  export const merchantGetByIdService = (id) => {      return request.get(`/merchant/${id}`)  }  //查询拥有的所有软件 /mrechant get  //编辑商品信息  /merchant put  //下架商品   /merchant post 修改状态即可  //添加新商品  //条件查询商品  //根据id查询商品  //查询所有订单信息  //根据订单查询  import request from '@/utils/request.js'  //查询所有订单信息  export const getAllOrder = () =>{    return request.get("/order/all")  }  export const getOrderWithPrice = () => {    return request.get("/order")  }  import request from '@/utils/request.js'  export const productAddService = (product) => {      return request.post("/product/add", product)  }  export const productGetAllByMerchantIdService = (merchantId) => {      return request.get(`/product/all/${merchantId}`)  }  export const productUpdateService = (product) => {      return request.put("/product", product)  }  // 查询所有商品  export const getAllProduct = () =>{      return request.get("/product/all")    }  import request from '@/utils/request.js'  //查询所有用户  export const admingetalluser = () =>{    return request.get("/user", {})  }  //修改用户状态  export const adminChangeStatus = ({id,status}) =>{    return request.put("/user", {id,status})  }  import { createRouter, createWebHistory } from 'vue-router'  import login from '../views/login/LoginIndex.vue'  import Product from '../views/product/ProductManage.vue'  import Home from '../views/home/HomeIndex.vue'  import Order from '../views/order/OrderIndex.vue'  import Merchant from '../views/merchant/MerchantIndex.vue'  import Count from '../views/count/CountIndex.vue'  const routes = [    {path:'/',component:login},    {      path: '/home',      component: Home,      redirect: '/home/product',      children: [        { path: 'product', component: Product },        { path: 'order',component: Order },        {path: 'merchant',component:Merchant},        {path:'count',component:Count}      ],    },  ]  const router = createRouter({    history: createWebHistory(import.meta.env.BASE\_URL),    routes  })  export default router  import { createPinia } from 'pinia'  import persist from 'pinia-plugin-persistedstate'  const pinia = createPinia()  pinia.use(persist)  export default pinia  export \* from './merchant'  import { defineStore } from 'pinia'  import { ref } from 'vue'  export const useMerchantStore = defineStore('merchant', () => {      const token = ref('')      const merchant = ref(null)      const setToken = (newToken) => {          token.value = newToken      }      const setMerchant = (newMerchant) => {          merchant.value = newMerchant      }      const removeToken = () => {          token.value = ''      }      const removeMerchant = () => {          merchant.value = null      }        return {          token,          merchant,          setToken,          setMerchant,          removeToken,          removeMerchant      }  }, {      persist: true  })  import axios from 'axios'  import { useMerchantStore } from '@/stores'  import { ElMessage } from 'element-plus'  import router from '@/router'  const baseURL = '/api'  const instance = axios.create({      // TODO 1. 基础地址，超时时间      baseURL,      timeout: 10000  })  // 请求拦截器  instance.interceptors.request.use(      (config) => {          // TODO 2. 携带token          const merchantStore = useMerchantStore()          if (merchantStore.token) {              config.headers.Authorization = merchantStore.token          }          return config      },      (err) => Promise.reject(err)  )  // 响应拦截器  instance.interceptors.response.use(      (res) => {          // TODO 3. 处理业务失败          // TODO 4. 摘取核心响应数据          if (res.data.code === 0) {              return res          }          // 处理业务失败          ElMessage.error(res.data.message || '服务错误')          return Promise.reject(res.data)      },      (err) => {          // TODO 5. 处理401错误          if (err.response?.status === 401) {              router.push("/login")          }          // 错误默认提示          ElMessage.error(err.response.data.message || '服务错误')          return Promise.reject(err)      }  )  export default instance  export { baseURL }  <template>    <div class="stats-container">      <h1>数据统计</h1>      <div class="stats-grid">        <div class="stats-card">          <h2>商品数量统计</h2>          <p>{{ productStats.count }}</p>        </div>        <div class="stats-card">          <h2>粉丝数量统计</h2>          <p>{{ merchantStats.count }}</p>        </div>        <div class="stats-card">          <h2>订单数据统计</h2>          <p>{{ orderStats.count }}</p>        </div>        <div class="stats-card">          <h2>销售额统计</h2>          <p>{{ salesStats.amount }}</p>        </div>      </div>    </div>  </template>  <script setup>  import { ref ,onMounted} from 'vue';  import { jwtDecode } from 'jwt-decode'  import {useMerchantStore} from '@/stores/merchant'  import {getAllProduct} from '@/api/product'  /\* import { admingetalluser} from '@/api/user';  import { getAllMerchant } from '../api/merchant'\*/  import { getOrderWithPrice } from '@/api/order';  const merchantID = ref()  //获取当前用户的id  const  decodeToken = () =>{    try {      console.log(useMerchantStore().token)      const token = jwtDecode(useMerchantStore().token)      merchantID.value = token.claims.merchantId      console.log(token)      console.log(merchantID.value)    } catch (error) {      console.error('Error decoding token:', error)      }  }  // 商品统计数据  const productStats = ref({    count: 0  });  // 商家统计数据  const merchantStats = ref({    count: 0  });  // 用户统计数据  const orderStats = ref({    count: 0  });  // 销售额统计数据  const salesStats = ref({    amount: 0  });  //获取上架商品数  const getAllProductList = async () => {    try {      const res = await getAllProduct();      console.log('获取商品列表成功:', res.data.data);        // 确保响应式      const filteredProduct = res.data.data.filter(product => product.status == 1 && product.merchantId ==  merchantID.value);      orderStats.value.count = filteredProduct.length       productStats.value.count = filteredProduct.length    } catch (error) {      console.error('获取商品列表失败:', error);    }  };  //获取所有订单  const getOrderList = async () =>{    try {      const res = await getOrderWithPrice();      console.log('获取订单成功:', res.data.data);      const filterOrder = res.data.data.filter(product => product.status == 1 && product.merchantId ==  merchantID.value);      const prices = filterOrder.map(product => product.price); // 提取每个商品的价格      const total = prices.reduce((sum, price) => sum + price, 0); // 计算价格的总和      salesStats.value.amount = total      } catch (error) {      console.error('获取用户失败:', error);    }  }  onMounted(()=>{      decodeToken()    getAllProductList()    getOrderList()  });  </script>  <style>  .stats-container {    max-width: 1200px;    margin: 0 auto;    padding: 20px;    text-align: center;  }  .stats-grid {    display: grid;    grid-template-columns: repeat(4, 1fr);    gap: 20px;    margin-top: 20px;  }  .stats-card {    background-color: #f7f7f7;    padding: 20px;    border-radius: 8px;    box-shadow: 0 2px 5px rgba(0,0,0,0.2);  }  .stats-card h2 {    margin: 0 0 10px 0;    color: #333;  }  .stats-card p {    margin: 0;    font-size: 1.5em;    color: #666;  }  </style>  <script setup>  import { ref } from 'vue';  import { useRouter } from 'vue-router';  import { Document, User, DataLine, Box, } from '@element-plus/icons-vue';  const router = useRouter();  const activeIndex = ref('product');  const showLogoutMenu = ref(false);  const navigateTo = (path) => {    router.push(path);  };  const handleSelect = (index) => {    activeIndex.value = index;    const routes = {      product: '/home/product',      order: '/home/order',      merchant: '/home/merchant',      count:'/home/count'    };    router.push(routes[index]);  };  const handleLogout = () => {    // 实现退出登录的逻辑    router.push("/")  };  </script>  <template>    <div class="home-container">      <header class="header">        <div class="logo">          <img src="../../assets/img/logo.png" alt="Logo" />        </div>        <div class="title">软件管理平台</div>        <div class="profile">          <div class="admin-avatar" @click="showLogoutMenu = !showLogoutMenu">            商家          </div>          <div v-if="showLogoutMenu" class="logout-menu">            <button @click="handleLogout">退出登录</button>          </div>        </div>      </header>      <el-row class="row">        <el-col :span="5" class="menu-column">          <el-menu            :default-active="activeIndex"            class="side-menu"            @select="handleSelect"            background-color="#f9f9f9"            text-color="#545c64"            active-text-color="#409EFF"          >          <el-menu-item index="count" @click="navigateTo('/home/count')">              <el-icon><DataLine/></el-icon>              数据统计            </el-menu-item>            <el-menu-item index="product" @click="navigateTo('/home/product')">              <el-icon><Box/></el-icon>              软件产品管理            </el-menu-item>            <el-menu-item index="order" @click="navigateTo('/home/order')">              <el-icon><Document/></el-icon>              查看订单            </el-menu-item>            <el-menu-item index="merchant" @click="navigateTo('/home/merchant')">              <el-icon><User/></el-icon>              个人信息            </el-menu-item>          </el-menu>        </el-col>        <el-col :span="19" class="content-column">          <div class="content">            <router-view></router-view>          </div>        </el-col>      </el-row>    </div>  </template>  <style scoped>  .home-container {    height: 98vh;  }  .header {    display: flex;    align-items: center;    justify-content: space-between;    background-color: #409eff;    color: #fff;    padding: 0 20px;    height: 60px;  }  .logo {    height: 40px;  }  .logo img {    height: 100%;  }  .title {    font-size: 20px;    font-weight: bold;  }  .profile {    position: relative;  }  .admin-avatar {    background-color: #fff;    color: #409eff;    padding: 8px 16px;    border-radius: 50%;    cursor: pointer;  }  .logout-menu {    position: absolute;    top: 50px;    right: 0;    background-color: #fff;    border: 1px solid #eaeaea;    padding: 10px 20px;    z-index: 1;  }  .logout-menu button {    border: none;    background-color: transparent;    cursor: pointer;  }  .row {    height: calc(100% - 60px);  }  .menu-column {    border-right: 1px solid #eaeaea;    height: 100%;  }  .side-menu {    height: 100%;  }  .content-column {    padding: 20px;    height: 100%;    overflow-y: auto;    background-color: #ffffff;  }  .content {    min-height: 100%;  }  </style>  <script setup>  import { ref } from 'vue';  import { useRouter } from 'vue-router';  import { merchantLoginService, merchantRegisterService } from '@/api/merchant';  import { useMerchantStore} from '@/stores';  const form = ref({    username: '',    password: '',  });  // 注册逻辑  const showRegisterModal = ref(false);  const registerForm = ref({    username: '',    password: '',    rePassword: '',  });  const registerFormRef = ref(null);  const merchantStore = useMerchantStore();   const handleRegister = async () => {    registerFormRef.value.validate(async (valid) => {      if (valid) {        try {          // 在这里添加注册的逻辑,例如调用后端 API 进行注册          await merchantRegisterService(registerForm.value)          ElMessage.success("注册成功~")          showRegisterModal.value = false;        } catch (error) {          console.error('注册失败:', error);        }      }    });  };  const rules = {    username: [{ required: true, message: '请输入用户名', trigger: 'blur' }],    password: [{ required: true, message: '请输入密码', trigger: 'blur' }],  };  const formRef = ref(null);  const router = useRouter();  const handleLogin = async () => {    const res = await merchantLoginService(form.value)    merchantStore.setToken(res.data.data.token)    merchantStore.setMerchant(res.data.data.merchant)    console.log(merchantStore.merchant)    ElMessage.success("登录成功");    router.push('/home');  };  </script>  <template>    <div class="login-container">      <el-card class="login-card">        <h2>软件商城商家登录</h2>        <el-form :model="form" :rules="rules" ref="formRef" label-position="top" status-icon>          <el-form-item label="用户名" prop="username">            <el-input v-model="form.username" placeholder="请输入用户名"></el-input>          </el-form-item>          <el-form-item label="密码" prop="password">            <el-input type="password" v-model="form.password" placeholder="请输入密码" @keydown.enter="handleLogin"></el-input>          </el-form-item>          <el-form-item>            <el-button type="primary" @click="handleLogin">登录</el-button>            <el-button type="text" @click="showRegisterModal = true">注册</el-button>          </el-form-item>        </el-form>      </el-card>      <el-dialog v-model="showRegisterModal" title="注册账号" width="400px">    <el-form :model="registerForm" :rules="registerRules" ref="registerFormRef" label-position="top" status-icon>      <el-form-item label="用户名" prop="username">        <el-input v-model="registerForm.username" placeholder="请输入用户名"></el-input>      </el-form-item>      <el-form-item label="密码" prop="password">        <el-input type="password" v-model="registerForm.password" placeholder="请输入密码"></el-input>      </el-form-item>      <el-form-item label="确认密码" prop="rePassword">        <el-input type="password" v-model="registerForm.rePassword" placeholder="请确认密码"></el-input>      </el-form-item>    </el-form>    <template #footer>      <span class="dialog-footer">        <el-button @click="showRegisterModal = false">取消</el-button>        <el-button type="primary" @click="handleRegister">注册</el-button>      </span>    </template>  </el-dialog>    </div>  </template>    <style>  .login-container {    display: flex;    align-items: center;    justify-content: center;    height: 98vh;    background-size: cover;    background-position: center;    background-repeat: no-repeat;    background-image: url('@/assets/img/bk1.png');  }  .login-card {    width: 400px;  }  </style>  <script setup>  import { ref } from 'vue';  import { useMerchantStore } from '@/stores';  // import { useRouter } from 'vue-router';  import { merchantUpdateService, merchantGetByIdService, merchantUpdatePasswordService } from '@/api/merchant'  import { ElMessage } from 'element-plus';  // const router = useRouter();  const merchantStore = useMerchantStore()  const userInfo = ref(merchantStore.merchant);  const showEditDialog = ref(false);  const showChangePasswordDialog = ref(false);  const passwordForm = ref({    oldPassword: '',    newPassword: '',    confirmPassword: '',  });  const editedUserInfo = ref({    id: userInfo.value.id,    avatar: userInfo.value.avatar,    username: userInfo.value.username,    phone: userInfo.value.phone,    password: userInfo.value.password  // 添加password  });  const updateAvatar = (response) => {    editedUserInfo.value.avatar = response.data;    ElMessage.success("上传头像成功~");  }  const getMerchant = async () => {    const res = await merchantGetByIdService(userInfo.value.id);    merchantStore.setMerchant(res.data.data);    userInfo.value = merchantStore.merchant;  }  const saveUserInfo = async () => {    await merchantUpdateService(editedUserInfo.value);    await getMerchant();    ElMessage.success("修改个人信息成功~")    showEditDialog.value = false;  };  const changePassword = async () => {    // 校验原密码和新密码    if (passwordForm.value.oldPassword !== userInfo.value.password) {      ElMessage.error("原密码不正确!");      return;    }    if (passwordForm.value.newPassword !== passwordForm.value.confirmPassword) {      ElMessage.error("新密码和确认密码不一致!");      return;    }    const passwordDTO = {      password: passwordForm.value.newPassword,      rePassword: passwordForm.value.confirmPassword    };    await merchantUpdatePasswordService(passwordDTO);    // 执行修改密码的逻辑    ElMessage.success("修改密码成功~")    showChangePasswordDialog.value = false;  };  </script>  <template>    <div class="merchant-info-container">      <el-card class="box-card">        <template #header>          <div class="card-header">            <span>个人信息</span>          </div>        </template>        <div class="user-profile">          <div class="avatar-wrapper">            <el-avatar :size="80" :src="userInfo.avatar" />          </div>          <div class="user-info">            <div class="username">用户名：{{ userInfo.username }}</div>            <div class="phone"> 手机号：{{ userInfo.phone }}</div>          </div>        </div>        <div class="actions">          <el-button type="primary" @click="showEditDialog = true">            编辑信息          </el-button>          <el-button type="primary" @click="showChangePasswordDialog = true">            修改密码          </el-button>        </div>      </el-card>      <el-dialog v-model="showEditDialog" title="编辑个人信息">        <el-form :model="editedUserInfo" label-width="80px">          <el-form-item label="用户名">            <el-input v-model="editedUserInfo.username" />          </el-form-item>          <el-form-item label="电话">            <el-input v-model="editedUserInfo.phone" />          </el-form-item>          <el-form-item label="头像">            <el-upload              action="/api/upload"              :on-success="updateAvatar"              :show-file-list="false"            >              <img v-if="editedUserInfo.avatar" :src="editedUserInfo.avatar" class="uploaded-avatar" />              <div v-else class="avatar-placeholder">点击上传头像</div>            </el-upload>          </el-form-item>        </el-form>        <template #footer>          <el-button @click="showEditDialog = false">取消</el-button>          <el-button type="primary" @click="saveUserInfo">保存</el-button>        </template>      </el-dialog>      <el-dialog v-model="showChangePasswordDialog" title="修改密码">        <el-form :model="passwordForm" label-width="80px">          <el-form-item label="原密码">            <el-input v-model="passwordForm.oldPassword" type="password" />          </el-form-item>          <el-form-item label="新密码">            <el-input v-model="passwordForm.newPassword" type="password" />          </el-form-item>          <el-form-item label="确认密码">            <el-input v-model="passwordForm.confirmPassword" type="password" />          </el-form-item>        </el-form>        <template #footer>          <el-button @click="showChangePasswordDialog = false">取消</el-button>          <el-button type="primary" @click="changePassword">保存</el-button>        </template>      </el-dialog>    </div>  </template>  <style scoped>  .merchant-info-container {    display: flex;    justify-content: center;    align-items: center;    height: 95vh;    background-color: #f5f5f5;  }  .box-card {    width: 500px;    height: 300px;  }  .card-header {    font-size: 18px;    font-weight: bold;  }  .user-profile {    display: flex;    align-items: center;    margin-bottom: 20px;  }  .avatar-wrapper {    margin-right: 20px;  }  .username {    font-size: 18px;    font-weight: bold;  }  .phone {    color: #666;  }  .actions {    display: flex;    justify-content: flex-end;  }  .actions button {    margin-left: 10px;  }  .uploaded-avatar {    width: 100px;    height: 100px;    border-radius: 50%;    object-fit: cover;  }  .avatar-placeholder {    width: 100px;    height: 100px;    display: flex;    align-items: center;    justify-content: center;    border-radius: 50%;    background-color: #f5f5f5;    color: #606266;    font-size: 14px;    cursor: pointer;    border: 1px dashed #dcdfe6;    transition: background-color 0.3s;  }  .avatar-placeholder:hover {    background-color: #e6f7ff;  }  </style>  <template>    <div class="order-management-container">      <h2>订单管理</h2>      <!-- 订单过滤器 -->      <div class="filters">        <div class="filter-item">          <span>订单编号:</span>          <el-input v-model="orderNo" placeholder="输入订单编号" />          <el-button type="primary" @click="searchOrders">查询</el-button>        </div>      </div>      <!-- 订单列表表格 -->      <el-table :data="paginatedOrders" style="width: 100%">        <el-table-column prop="id" label="订单ID" width="120"></el-table-column>        <el-table-column prop="orderNumber" label="订单编号" width="180"></el-table-column>        <el-table-column prop="productName" label="交易商品" width="180"></el-table-column>        <!-- <el-table-column prop="productLink" label="商品链接" width="180">          <template #default="scope">            <a :href="scope.row.productLink" target="\_blank">{{ scope.row.productLink }}</a>          </template>        </el-table-column>   -->        <el-table-column prop="createTime" label="交易时间" width="180"></el-table-column>        <el-table-column prop="userName" label="买家" width="120"></el-table-column>        <el-table-column prop="status" label="订单状态" width="120">          <template #default="scope">            <el-tag :type="getStatusTagType(scope.row.status)">{{ getStatusText(scope.row.status) }}</el-tag>          </template>        </el-table-column>      </el-table>      <!-- 分页控件 -->      <div class="pagination">        <el-pagination          v-model:current-page="currentPage"          v-model:page-size="pageSize"          :total="filteredOrders.length"          layout="prev, pager, next"          @current-change="handlePageChange"        />      </div>    </div>  </template>  <script setup>  import { ref, computed ,onMounted} from 'vue';  import { getAllOrder } from '@/api/order';  import { getAllProduct } from '@/api/product';  import { admingetalluser } from '@/api/user';  const orderList = ref([]);  const merchantMap = ref([])  const productMap = ref([])  const productMap2 = ref([])  const userMap = ref([])  const getOrderList = async () =>{    try {      const res = await getAllOrder();      console.log('获取订单成功:', res.data.data);      orderList.value.splice(0, orderList.value.length, ...res.data.data);      const ordersWithNames = orderList.value.map(order => {        return {          ...order,          merchantName: merchantMap[order.merchantId] || '未知商家' ,          productName: productMap[order.productId] || '未知商品',          productLink: productMap2[order.productId],          userName:userMap[order.userId]        };      });      orderList.value.splice(0, orderList.value.length, ...ordersWithNames);    } catch (error) {      console.error('获取用户失败:', error);    }  }  //获取产品列表  const getAllProductList = async () => {    try {      const res = await getAllProduct();      console.log('获取商品列表成功:', res.data.data);      productMap.value.splice(0, productMap.value.length, ...res.data.data);      res.data.data.forEach(product => {        productMap[product.id] = product.name;        });      productMap2.value.splice(0, productMap2.value.length, ...res.data.data);      res.data.data.forEach(product => {        productMap2[product.id] = product.source;      });    } catch (error) {      console.error('获取商品列表失败:', error);    }  };  //获取用户列表  const getAllUser = async () => {    try {      const res = await admingetalluser();      console.log('获取用户成功:', res.data.data);      userMap.value.splice(0, userMap.value.length, ...res.data.data);      res.data.data.forEach(user => {        userMap[user.id] = user.username;      });    } catch (error) {      console.error('获取用户失败:', error);    }  };      // 当前页码和每页数量  const currentPage = ref(1);  const pageSize = ref(10);  // 订单编号过滤器  const orderNo = ref('');  // 根据订单编号过滤订单列表  const filteredOrders = computed(() => {    let orders = [...orderList.value];    if (orderNo.value) {      orders = orders.filter((order) =>        order.orderNumber.toLowerCase().includes(orderNo.value.toLowerCase())      );    }    return orders;  });  // 分页后的订单列表  const paginatedOrders = computed(() => {    const start = (currentPage.value - 1) \* pageSize.value;    const end = start + pageSize.value;    return filteredOrders.value.slice(start, end);  });  //处理状态显示  const getStatusTagType = (status) => {    switch (status) {      case "1":        return 'success';      case 0:        return 'warning';      case -1:        return 'danger';      default:        return '';    }  };  const getStatusText = (status) => {      switch (status) {        case 1:          return '已完成';        case 0:          return '未支付';        case -1:          return '已取消';        default:          return '未知状态'; // 可选的默认状态      }    }  // 点击查询按钮时的处理函数  const searchOrders = () => {    currentPage.value = 1;  };  // 分页切换时的处理函数  const handlePageChange = () => {    // 处理分页切换  };  onMounted(() => {    getAllProductList()      .then(getAllUser)      .then(getOrderList)  });  </script>  <style scoped>  .order-management-container {    padding: 20px;  }  .filters {    display: flex;    justify-content: space-between;    align-items: center;    margin-bottom: 20px;  }  .filter-item {    display: flex;    align-items: center;  }  .filter-item span {    margin-right: 10px;    width: 120px;  }  .filter-item .el-input,  .filter-item .el-select {    margin-right: 10px;  }  .pagination {    display: flex;    justify-content: center;    margin-top: 20px;  }  </style>  <script setup>  import { ref, computed, onMounted } from 'vue';  import { categoryAddService, categoryGetAllService} from '@/api/category'  import { productAddService, productGetAllByMerchantIdService, productUpdateService } from '@/api/product'  import { ElMessage } from 'element-plus';  import { Plus } from '@element-plus/icons-vue'  import { useMerchantStore} from '@/stores';  const merchantStore = useMerchantStore();  const merchantId = merchantStore.merchant.id  const getCategoryName = (categoryId) => {    const category = categories.value.find(item => item.id === categoryId)    return category ? category.name : "未知分类"  }  const addCategory = async (name) => {    await categoryAddService({name});    addCategoryDialogVisible.value = false;    ElMessage.success("添加分类成功");    const res = await categoryGetAllService();    categories.value = res.data.data;  }  const addProduct = async () => {    await productAddService(newProduct.value)    ElMessage.success("添加商品成功")    const res = await productGetAllByMerchantIdService(merchantId);    products.value = res.data.data;    newProduct.value = {      name: '',      price: 0,      categoryId: '',      video: '',      image: '',      source: '',      description: '',    };    addProductDialogVisible.value = false  }  const updateImage = async (response) => {    editedProduct.value.image = response.data;    ElMessage.success("上传图片成功~");  }  const updateSource = async (response) => {    editedProduct.value.source = response.data;    ElMessage.success("上传文件成功~")  }  const updateVideo = async (response) => {    editedProduct.value.video = response.data;    ElMessage.success("上传演示成功~");  }  const addImage = async (response) => {    newProduct.value.image = response.data;    ElMessage.success("上传图片成功~");  }  const addSource = async (response) => {    newProduct.value.source = response.data;    ElMessage.success("上传文件成功~")  }  const addViedo= async (response) => {    newProduct.value.video = response.data;    ElMessage.success("上传演示视频成功~")  }  const products = ref([]);  const sortType = ref('priceAsc');  const searchText = ref('');  const selectedCategory = ref('');  const currentPage = ref(1);  const pageSize = ref(10);  const addProductDialogVisible = ref(false);  const editProductDialogVisible = ref(false);  const addCategoryDialogVisible = ref(false);  const newProduct = ref({    name: '',    price: 0,    categoryId: '',    video: '',    image: '',    source: '',    description: '',  });  const editedProduct = ref({});  const newCategory = ref('');  const filteredProducts = computed(() => {    let filtered = [...products.value];    if (searchText.value) {      filtered = filtered.filter(product =>        product.name.toLowerCase().includes(searchText.value.toLowerCase())      );    }    if (selectedCategory.value) {      filtered = filtered.filter(product => product.categoryId === selectedCategory.value);    }    return filtered;  });  const paginatedProducts = computed(() => {    const startIndex = (currentPage.value - 1) \* pageSize.value;    const endIndex = startIndex + pageSize.value;    return filteredProducts.value.slice(startIndex, endIndex);  });  // const categories = computed(() => {  //   return [...new Set(products.value.map(product => product.category))];  // });  const categories = ref([])  onMounted(async () => {    const res = await categoryGetAllService();    categories.value = res.data.data; // 将数据赋值给 ref    const productList = await productGetAllByMerchantIdService(merchantId);    products.value = productList.data.data;  });  function sortProducts() {    if (sortType.value === 'priceAsc') {      products.value.sort((a, b) => a.price - b.price);    } else {      products.value.sort((a, b) => b.price - a.price);    }  }  function searchProducts() {    currentPage.value = 1;  }  function filterByCategory() {    currentPage.value = 1;  }  function handlePageChange(page) {    currentPage.value = page;  }  function showAddProductDialog() {    addProductDialogVisible.value = true;  }  // function addProduct() {  //   products.value.push({ ...newProduct.value });  //   addProductDialogVisible.value = false;  //   newProduct.value = {  //     name: '',  //     description: '',  //     price: 0,  //     link: '',  //     image: '',  //     category: '',  //     status: 'pending'  //   };  // }  function editProduct(product) {    editedProduct.value = { ...product };    editProductDialogVisible.value = true;  }  const updateProduct = async () => {    await productUpdateService(editedProduct.value)    const productList = await productGetAllByMerchantIdService(merchantId);    products.value = productList.data.data;    editProductDialogVisible.value = false;    ElMessage.success("商品更新成功")  }  // function updateProduct() {  //   const index = products.value.findIndex(product => product.name === editedProduct.value.name);  //   products.value[index] = { ...editedProduct.value };  //   editProductDialogVisible.value = false;  // }  function deleteProduct(product) {    const index = products.value.indexOf(product);    products.value.splice(index, 1);  }  function showAddCategoryDialog() {    addCategoryDialogVisible.value = true;  }  // function addCategory() {  //   if (newCategory.value && !categories.value.includes(newCategory.value)) {  //     products.value.push({  //       name: newCategory.value,  //       description: '',  //       price: 0,  //       link: '',  //       image: '',  //       category: newCategory.value,  //       status: 'pending'  //     });  //     newCategory.value = '';  //   }  //   addCategoryDialogVisible.value = false;  // }  //  </script>  <template>    <div>      <h1>软件产品管理</h1>      <div class="filters">        <div class="filter-item">          <span>排序方式:</span>          <el-select v-model="sortType" @change="sortProducts">            <el-option label="价格升序" value="priceAsc"></el-option>            <el-option label="价格降序" value="priceDesc"></el-option>          </el-select>        </div>        <div class="filter-item">          <span>软件查询:</span>          <el-input v-model="searchText" placeholder="" />          <el-button type="primary" @click="searchProducts">搜索</el-button>        </div>        <div class="filter-item">          <span>软件分类:</span>          <el-select v-model="selectedCategory" @change="filterByCategory">            <el-option label="全部" value=""></el-option>            <el-option v-for="category in categories" :key="category.id" :label="category.name" :value="category.id"></el-option>          </el-select>          </div>        <div class="filter-item">          <el-button type="primary" @click="showAddProductDialog">上架新商品</el-button>        </div>      </div>      <el-table :data="paginatedProducts" style="width: 100%">        <el-table-column prop="image" label="软件图片" width="100">          <template #default="scope">            <img :src="scope.row.image" alt="商品图片" class="product-image" />          </template>        </el-table-column>        <el-table-column prop="name" label="软件名称" />        <el-table-column prop="description" label="软件描述" />        <el-table-column prop="price" label="软件价格" />        <el-table-column label="软件分类" >          <template #default="scope">            <span>{{ getCategoryName(scope.row.categoryId) }}</span>          </template>        </el-table-column>        <el-table-column label="状态">          <template #default="scope">            <el-tag v-if="scope.row.status === 0">审核中</el-tag>            <el-tag v-else-if="scope.row.status === 1" type="success">已上架</el-tag>            <el-tag v-else type="danger">已下架</el-tag>          </template>        </el-table-column>        <el-table-column label="操作">          <template #default="scope">            <el-button type="primary" @click="editProduct(scope.row)">编辑</el-button>            <el-button v-if="scope.row.status === 'published'" type="danger" @click="deleteProduct(scope.row)">下架</el-button>          </template>        </el-table-column>      </el-table>      <div class="pagination">        <el-pagination v-model:current-page="currentPage" v-model:page-size="pageSize" :total="filteredProducts.length" layout="prev, pager, next" @current-change="handlePageChange" />      </div>      <!-- 添加商品对话框 -->  <el-dialog v-model="addProductDialogVisible" title="上架新商品">    <el-form :model="newProduct" label-width="120px">      <el-form-item label="商品名称">        <el-input v-model="newProduct.name" />      </el-form-item>      <el-form-item label="商品描述">        <el-input v-model="newProduct.description" type="textarea" />      </el-form-item>      <el-form-item label="商品价格">        <el-input-number v-model="newProduct.price" :min="0" />      </el-form-item>      <el-form-item label="上传商品图片">        <el-upload          style="background-color: gainsboro"          class="avatar-uploader"          action="/api/upload"          :show-file-list="false"          :on-success="addImage"        >          <img v-if="newProduct.image" :src="newProduct.image" class="avatar" />          <el-icon v-else class="avatar-uploader-icon"><Plus /></el-icon>        </el-upload>      </el-form-item>      <el-form-item label="上传商品文件">        <el-upload            action="/api/upload"            :auto-upload="true"            :on-success="addSource"          >            <el-button size="middle" type="success"              >点击上传产品文件(压缩包)</el-button            >            </el-upload>      </el-form-item>      <el-form-item label="上传商品视频">        <el-upload            action="/api/upload"            :auto-upload="true"            :on-success="addViedo"          >            <el-button size="middle" type="primary"              >点击上传产品演示视频</el-button            >            </el-upload>      </el-form-item>      <el-form-item label="商品分类">        <el-select v-model="newProduct.categoryId" placeholder="选择分类">          <el-option v-for="category in categories" :key="category.id" :label="category.name" :value="category.id"></el-option>        </el-select>        <el-button type="primary" @click="showAddCategoryDialog">添加分类</el-button>      </el-form-item>    </el-form>    <span class="dialog-footer">      <el-button @click="addProductDialogVisible = false">取 消</el-button>      <el-button type="primary" @click="addProduct">确定</el-button>    </span>  </el-dialog>      <!-- 编辑商品对话框 -->    <el-dialog v-model="editProductDialogVisible" title="编辑商品">    <el-form :model="editedProduct" label-width="120px">      <el-form-item label="商品名称">        <el-input v-model="editedProduct.name" />      </el-form-item>      <el-form-item label="商品描述">        <el-input v-model="editedProduct.description" type="textarea" />      </el-form-item>      <el-form-item label="商品价格">        <el-input-number v-model="editedProduct.price" :min="0" />      </el-form-item>      <el-form-item label="上传商品图片">        <el-upload          style="background-color: gainsboro"          class="avatar-uploader"          action="/api/upload"          :show-file-list="false"          :on-success="updateImage"        >          <img v-if="editedProduct.image" :src="editedProduct.image" class="avatar" />          <el-icon v-else class="avatar-uploader-icon"><Plus /></el-icon>        </el-upload>      </el-form-item>      <el-form-item label="上传商品文件">        <el-upload            action="/api/upload"            :auto-upload="true"            :on-success="updateSource"          >            <el-button size="middle" type="success"              >点击上传产品文件(压缩包)</el-button            >            </el-upload>      </el-form-item>      <el-form-item label="上传商品视频">        <el-upload            action="/api/upload"            :auto-upload="true"            :on-success="updateVideo"          >            <el-button size="middle" type="primary"              >点击上传产品演示视频</el-button            >            </el-upload>      </el-form-item>      <el-form-item label="商品分类">        <el-select v-model="editedProduct.categoryId" placeholder="选择分类">          <el-option            v-for="category in categories"            :key="category.id"            :label="category.name"            :value="category.id"> <!-- 确保 value 类型与 editedProduct.categoryId 类型一致 -->          </el-option>        </el-select>        <el-button type="primary" @click="showAddCategoryDialog">添加分类</el-button>      </el-form-item>    </el-form>    <span  class="dialog-footer">          <el-button @click="editProductDialogVisible = false">取 消</el-button>          <el-button type="primary" @click="updateProduct">确定</el-button>      </span>  </el-dialog>      <!-- 添加分类对话框 -->      <el-dialog v-model="addCategoryDialogVisible" title="添加分类">        <el-form>          <el-form-item label="分类名称">            <el-input v-model="newCategory" />          </el-form-item>        </el-form>        <span  class="dialog-footer">          <el-button @click="addCategoryDialogVisible = false">取 消</el-button>          <el-button type="primary" @click="addCategory(newCategory)">确定</el-button>        </span>      </el-dialog>    </div>  </template>  <style>  .product-image {    width: 100px;    height: auto;  }  .filters {    margin-bottom: 20px;  }  .filter-item {    display: inline-block;    margin-right: 20px;  }  .pagination {    margin-top: 20px;  }  </style>   <style scoped>  .product-image {    width: 50px;    height: 50px;    object-fit: cover;  }  .filters {    display: flex;    justify-content: space-between;    align-items: center;    margin-bottom: 20px;  }  .filter-item {    display: flex;    align-items: center;  }  .filter-item span {    margin-right: 10px;    width: 130px;  }  .filter-item .el-input,  .filter-item .el-select {    margin-right: 10px;  }  .pagination {    display: flex;    justify-content: center;    margin-top: 20px;  }  .avatar-uploader .el-upload {    border: 1px dashed var(--el-border-color);    border-radius: 6px;    cursor: pointer;    position: relative;    overflow: hidden;    transition: var(--el-transition-duration-fast);  }  .avatar-uploader .el-upload:hover {    border-color: var(--el-color-primary);  }  .avatar-uploader .avatar {    width: 178px;    height: 178px;    display: block;  }  .el-icon.avatar-uploader-icon {    font-size: 28px;    color: #8c939d;    width: 178px;    height: 178px;    text-align: center;  }  .dialog-footer {    padding-left: 40%;  }  </style>  import request from '@/utils/request.js'  // login api  export const adminLoginService = ({ username, password }) => {      return request.post("/admin/login", { username, password });  }    //查询商家拥有商品  //查询所有订单  //根据订单编号查询订单  import request from '@/utils/request.js'  //查询分类  export const admingetAllCategory = () =>{    return request.get("/category/all")  }  //添加分类  export const adminInsertCategory = (name) =>{    return request.post("/category/add",{name})  }  //编辑分类  export const adminUpdateCategory = ({id,name}) =>{    return request.put("/category",{id,name})  }  //删除分类  export const admindeleteCategory = (id) => {    return request.delete(`/category/${id}`); // 注意这里的模板字符串  }  //根据id查询分类  export const getCategoryById = (id) =>{    return request.get(`/category/${id}`)  }  import request from '@/utils/request.js'  //查询所有商家  export const getAllMerchant = () =>{    return request.get("/merchant")  }  //修改商家状态  export const ChangeMerchantStatus = ({id,status}) =>{    return request.put("/merchant/status",{id,status})  }  import request from '@/utils/request.js'  //查询所有订单信息  export const getAllOrder = () =>{    return request.get("/order/all")  }  export const getOrderWithPrice = () => {    return request.get("/order")  }  import request from '@/utils/request.js'  // 查询所有商品  export const getAllProduct = () =>{    return request.get("/product/all")  }  //处理审核，同意或拒绝  下架商品  export const updateStatus = ({id,status}) => {    return request.put("/product/status",{id,status})  }  import request from '@/utils/request.js'  //查询所有用户  export const admingetalluser = () =>{    return request.get("/user", {})  }  //修改用户状态  export const adminChangeStatus = ({id,status}) =>{    return request.put("/user", {id,status})  }  import { createRouter, createWebHistory } from 'vue-router'  import login from '../views/LoginIndex.vue'  import Audit from '../views/ProductAudit.vue'  import Product from '../views/ProductManage.vue'  import Merchant from '../views/MerchantManger.vue'  import User from '../views/UserManager.vue'  import Home from '../views/HomeIndex.vue'  import Order from '../views/OrderIndex.vue'  import Category from '../views/CategoryIndex.vue'  import Count from '../views/CountIndex.vue'  const routes = [    { path:'/', component:login },    {      path: '/home',      component: Home,      redirect: '/home/count',      children: [        { path: 'audit', component: Audit },        { path: 'product', component: Product },        { path: 'merchant', component: Merchant },        { path: 'user', component: User },        { path: 'order',component: Order },        {path:'category',component:Category},        {path:'count',component:Count},      ],    },  ]  const router = createRouter({    history: createWebHistory(import.meta.env.BASE\_URL),    routes  })  export default router  import { defineStore } from 'pinia'  import { ref } from 'vue'  export const useAdminStore = defineStore('admin', () => {      const token = ref('')      const setToken = (newToken) => {          token.value = newToken      }      const removeToken = () => {          token.value = ''      }      return {          token,          setToken,          removeToken      }  }, {      persist: true  })  import { createPinia } from 'pinia'  import persist from 'pinia-plugin-persistedstate'  const pinia = createPinia()  pinia.use(persist)  export default pinia  export \* from './admin'  import axios from 'axios'  import { useAdminStore } from '@/stores'  import { ElMessage } from 'element-plus'  import router from '@/router'  const baseURL = '/api'  const instance = axios.create({      // TODO 1. 基础地址，超时时间      baseURL,      timeout: 10000  })  // 请求拦截器  instance.interceptors.request.use(      (config) => {          // TODO 2. 携带token          const adminStore = useAdminStore()          if (adminStore.token) {              config.headers.Authorization = adminStore.token            }          return config      },      (err) => Promise.reject(err)  )  // 响应拦截器  instance.interceptors.response.use(      (res) => {          // TODO 3. 处理业务失败          // TODO 4. 摘取核心响应数据          if (res.data.code === 0) {              return res          }          // 处理业务失败          ElMessage.error(res.data.message || '服务错误')          return Promise.reject(res.data)      },      (err) => {          // TODO 5. 处理401错误          if (err.response?.status === 401) {              router.push("/")          }          // 错误默认提示          ElMessage.error(err.response.data.message || '服务错误')          return Promise.reject(err)      }  )  export default instance  export { baseURL }  <template>    <div class="category-management">      <h1>分类管理</h1>      <el-card class="new-category-card">        <h2>新增分类</h2>        <el-input v-model="newCategoryName" placeholder="输入分类名称" />        <el-button type="primary" @click="addCategory">添加</el-button>      </el-card>      <el-card class="category-list-card">        <h2>分类列表</h2>        <el-table :data="categories" style="width: 100%">          <el-table-column prop="id" label="ID" width="100" />          <el-table-column prop="name" label="名称" />          <el-table-column label="操作" width="200">            <template #default="scope">              <el-button type="primary" @click="editCategory(scope.row)">编辑</el-button>              <el-button type="danger" @click="deleteCategory(scope.row.id)">删除</el-button>            </template>          </el-table-column>        </el-table>      </el-card>      <!-- 编辑分类的对话框 -->      <el-dialog title="编辑分类" v-model="isEditDialogVisible">        <el-form :model="currentCategory">          <el-form-item label="分类名称">            <el-input v-model="currentCategory.name"></el-input>          </el-form-item>        </el-form>        <span class="dialog-footer">          <el-button @click="isEditDialogVisible = false">取消</el-button>          <el-button type="primary" @click="updateCategory">更新</el-button>        </span>      </el-dialog>      </div>  </template>  <script setup>  import { ref, reactive, onMounted } from 'vue';  import {admingetAllCategory , adminInsertCategory,admindeleteCategory ,adminUpdateCategory} from '../api/category'  // 查询所有分类信息  const categories = reactive([  ]);  const getAllCategory = async () => {    try {      const res = await admingetAllCategory();      console.log('获取用户成功:', res.data.data);      // 确保响应式      categories.splice(0, categories.length, ...res.data.data);    } catch (error) {      console.error('获取用户失败:', error);    }  };  // 添加分类的方法  const newCategoryName = ref('');  const addCategory = async () => {    try{      const res = await adminInsertCategory(newCategoryName.value);      getAllCategory()      console.log('添加用户成功:', res.data.data);    }catch (error) {      console.error('添加用户失败:', error);    }  };  // 编辑分类的方法  const currentCategory = reactive({ id: '', name: '' });  const isEditDialogVisible = ref(false);  const editCategory = (category) => {    currentCategory.id = category.id;    currentCategory.name = category.name;    isEditDialogVisible.value = true;  };  const updateCategory = async () => {    try {      const res = await adminUpdateCategory(currentCategory);      getAllCategory();      isEditDialogVisible.value = false;      console.log('更新分类成功:', res.data);    } catch (error) {      console.error('更新分类失败:', error);    }  };  // 删除分类的方法  const deleteCategory = async (id) => {    const isConfirmed = window.confirm('确认删除该分类吗？');      if (isConfirmed) {      try{      console.log(id)      const res = await admindeleteCategory(id);      getAllCategory()      console.log('删除用户成功:', res.data);    }catch (error) {      console.error('删除用户失败:', error);    }    } else {      // 用户点击了“取消”，不执行操作      console.log('用户取消操作');    }    };  onMounted(() => {    console.log("页面渲染")    getAllCategory()    console.log("获取数据成功")  });  </script>  <style>  .category-management {    padding: 50px;  }  .new-category-card,  .category-list-card {    margin-bottom: 20px;  }  .new-category-card .el-input,  .new-category-card .el-button {    margin-top: 10px;  }  </style>  <template>    <div class="stats-container">      <h1>数据统计</h1>      <div class="stats-grid">        <div class="stats-card">          <h2>上架商品统计</h2>          <p>{{ productStats.count }}</p>        </div>        <div class="stats-card">          <h2>注册商家统计</h2>          <p>{{ merchantStats.count }}</p>        </div>        <div class="stats-card">          <h2>注册用户统计</h2>          <p>{{ userStats.count }}</p>        </div>        <div class="stats-card">          <h2>销售额统计</h2>          <p>{{ salesStats.amount }}￥</p>        </div>      </div>    </div>  </template>  <script setup>  import { ref ,onMounted} from 'vue';  import {getAllProduct} from '../api/product'  import { admingetalluser} from '@/api/user';  import { getAllMerchant } from '../api/merchant'  import { getOrderWithPrice } from '@/api/order';  // 上架商品数据  const productStats = ref({    count:0  });  // 注册商家数据  const merchantStats = ref({    count: 0  });  // 注册用户数据  const userStats = ref({    count: 0  });  // 销售额数据  const salesStats = ref({    amount: 0  });  //获取上架商品数  const getAllProductList = async () => {    try {      const res = await getAllProduct();      console.log('获取商品列表成功:', res.data.data);      // 确保响应式      const filteredProduct = res.data.data.filter(product => product.status == 1);       productStats.value.count = filteredProduct.length    } catch (error) {      console.error('获取商品列表失败:', error);    }  };  //获取所有用户  const getAllUser = async () => {    try {      const res = await admingetalluser();      console.log('获取用户成功:', res.data.data);      // 使用 userList.splice 来确保响应式      userStats.value.count=res.data.data.length    } catch (error) {      console.error('获取用户失败:', error);    }  };  //获取所有商家列表实现    const getMerchant = async () => {    try {      const res = await getAllMerchant();      console.log('获取用户成功:', res.data.data);      merchantStats.value.count = res.data.data.length    } catch (error) {      console.error('获取用户失败:', error);    }  };  const getOrderList = async () =>{    try {      const res = await getOrderWithPrice();      console.log('获取订单成功:', res.data.data);      const filterOrder = res.data.data.filter(product => product.status == 1);      const prices = filterOrder.map(product => product.price); // 提取每个商品的价格      const total = prices.reduce((sum, price) => sum + price, 0); // 计算价格的总和      salesStats.value.amount = total      } catch (error) {      console.error('获取用户失败:', error);    }  }  onMounted(() => {   getAllProductList()   getAllUser()   getMerchant()   getOrderList()    });  </script>  <style>  .stats-container {    max-width: 1200px;    margin: 0 auto;    padding: 20px;    text-align: center;  }  .stats-grid {    display: grid;    grid-template-columns: repeat(4, 1fr);    gap: 20px;    margin-top: 20px;  }  .stats-card {    background-color: #f7f7f7;    padding: 20px;    border-radius: 8px;    box-shadow: 0 2px 5px rgba(0,0,0,0.2);  }  .stats-card h2 {    margin: 0 0 10px 0;    color: #333;  }  .stats-card p {    margin: 0;    font-size: 1.5em;    color: #666;  }  </style>  <script setup>  import { ref } from 'vue';  import { useRouter } from 'vue-router';  import { useAdminStore } from '@/stores';  // 导入 Element Plus 中的图标  import { ShoppingCart, Document, User, DataLine, Box, Edit, Folder } from '@element-plus/icons-vue';  const router = useRouter();  const activeIndex = ref('audit');  const showLogoutMenu = ref(false);  const navigateTo = (path) => {    router.push(path);  };  const handleSelect = (index) => {    activeIndex.value = index;    const routes = {      audit: '/home/audit',      product: '/home/product',      merchant: '/home/merchant',      user: '/home/user',      order: '/home/order',      category: '/home/category',      count:'/home/count'    };    router.push(routes[index]);  };  const handleLogout = () => {    // 实现退出登录的逻辑    useAdminStore().removeToken(); // 删除token    router.push('/');    console.log('退出登录');  };  </script>  <template>    <div class="home-container">      <header class="header">        <div class="logo">          <img src="../assets/img/logo.png" alt="Logo"/>        </div>        <div class="title">软件管理平台</div>        <div class="profile">          <div class="admin-avatar" @click="showLogoutMenu = !showLogoutMenu">            管理员          </div>          <div v-if="showLogoutMenu" class="logout-menu">            <button @click="handleLogout">退出登录</button>          </div>        </div>      </header>      <el-row class="row">        <el-col :span="5" class="menu-column">          <el-menu            :default-active="activeIndex"            class="side-menu"            @select="handleSelect"            background-color="#f9f9f9"            text-color="#545c64"            active-text-color="#409EFF"          >          <el-menu-item index="count" @click="navigateTo('/home/count')">              <el-icon><DataLine /></el-icon>数据统计            </el-menu-item>            <el-menu-item index="audit" @click="navigateTo('/home/audit')">              <el-icon><Edit /></el-icon>软件上架审核            </el-menu-item>            <el-menu-item index="product" @click="navigateTo('/home/product')">              <el-icon><Box /></el-icon>软件产品管理            </el-menu-item>            <el-menu-item index="category" @click="navigateTo('/home/category')">              <el-icon><Folder /></el-icon>软件分类管理            </el-menu-item>            <el-menu-item index="user" @click="navigateTo('/home/user')">              <el-icon><User /></el-icon>用户管理            </el-menu-item>            <el-menu-item index="merchant" @click="navigateTo('/home/merchant')">              <el-icon><ShoppingCart /></el-icon>商家管理            </el-menu-item>            <el-menu-item index="order" @click="navigateTo('/home/order')">              <el-icon><Document /></el-icon>查看订单            </el-menu-item>          </el-menu>        </el-col>        <el-col :span="19" class="content-column">          <div class="content">            <router-view></router-view>          </div>        </el-col>      </el-row>    </div>  </template>  <style scoped>  .home-container {    height: 100vh;  }  .header {    display: flex;    align-items: center;    justify-content: space-between;    background-color: #409eff;    color: #fff;    padding: 0 20px;    height: 60px;  }  .logo {    height: 40px;  }  .logo img {    height: 100%;  }  .title {    font-size: 20px;    font-weight: bold;  }  .profile {    position: relative;  }  .admin-avatar {    background-color: #fff;    color: #409eff;    padding: 8px 16px;    border-radius: 20px;    cursor: pointer;  }  .logout-menu {    position: absolute;    top: 50px;    right: 0;    background-color: #fff;    border: 1px solid #eaeaea;    padding: 10px 20px;    z-index: 1;  }  .logout-menu button {    border: none;    background-color: transparent;    cursor: pointer;  }  .row {    height: calc(100% - 60px);  }  .menu-column {    border-right: 1px solid #eaeaea;    height: 100%;  }  .side-menu {    height: 100%;  }  .content-column {    padding: 20px;    height: 100%;    overflow-y: auto;    background-color: #ffffff;  }  .content {    min-height: 100%;  }  </style>  <script setup>  import { ref } from 'vue';  import { useRouter } from 'vue-router';  import { adminLoginService } from '@/api/admin';  import { useAdminStore} from '@/stores';  const form = ref({    username: '',    password: '',  });  const rules = {    username: [{ required: true, message: '请输入用户名', trigger: 'blur' }],    password: [{ required: true, message: '请输入密码', trigger: 'blur' }],  };  const formRef = ref(null);  const router = useRouter();  const adminStore = useAdminStore()  const handleLogin = async () => {    const res = await adminLoginService(form.value)    adminStore.setToken(res.data.data)      console.log('1111')    console.log(adminStore.token)      router.push('/home');    }  </script>  <template>    <div class="login-container">      <el-card class="login-card">        <h2>软件商城管理员登录登录</h2>        <el-form          :model="form"          :rules="rules"          ref="formRef"          label-position="top"          status-icon          @keydown.enter="handleLogin"        >          <el-form-item label="用户名" prop="username">            <el-input v-model="form.username" placeholder="请输入用户名"></el-input>          </el-form-item>          <el-form-item label="密码" prop="password">            <el-input type="password" v-model="form.password" placeholder="请输入密码"></el-input>          </el-form-item>          <el-form-item >            <el-button type="primary" @click="handleLogin">登录</el-button>          </el-form-item>        </el-form>      </el-card>    </div>  </template>  <style>  .login-container {    display: flex;    align-items: center;    justify-content: center;    height: 98vh;    background-size: cover;    background-position: center;    background-repeat: no-repeat;    background-image: url('../assets/img/bk.png');  }  .login-card {    width: 400px;  }  </style>  <template>    <div>      <h1>商家管理</h1>      <el-table :data="merchantList" style="width: 100%">        <el-table-column prop="avatar" label="头像" width="100">          <template #default="scope">            <img :src="scope.row.avatar" alt="Avatar" class="merchant-avatar" />          </template>        </el-table-column>        <el-table-column prop="username" label="商家名称" />        <el-table-column prop="phone" label="手机号" />        <el-table-column prop="createTime" label="创建时间" />        <el-table-column prop="updateTime" label="最后修改时间" />        <el-table-column prop="status" label="账号状态">          <template #default="scope">            <el-button              :type="scope.row.status == '1' ? 'success' : 'danger'"              @click="toggleStatus(scope.row)"            >              {{ scope.row.status == '1' ? '正常' : '禁用' }}            </el-button>          </template>        </el-table-column>        <el-table-column label="上架商品">          <template #default="scope">            <el-button type="primary" @click="showProductModal(scope.row)">              查看商品            </el-button>          </template>        </el-table-column>      </el-table>      <!-- 商品展示模态框 -->      <el-dialog v-model="showModal" title="上架商品" width="80%">        <div v-if="selectedMerchant">          <h3>{{ selectedMerchant }}的上架商品</h3>          <div>            <el-form :inline="true" class="demo-form-inline">              <!-- <el-form-item label="排序方式">                <el-select v-model="sortBy" placeholder="请选择">                  <el-option label="价格升序" value="priceAsc" />                  <el-option label="价格降序" value="priceDesc" />                </el-select>              </el-form-item>   -->              <!-- <el-form-item label="商品分类">                <el-select v-model="selectedCategory" placeholder="请选择">                  <el-option label="全部" value="" />                  <el-option                    v-for="category in categories"                    :key="category.name"                    :label="category.name"                    :value="category.name"                  />                </el-select>              </el-form-item>  -->            </el-form>          </div>          <el-table :data="product\_M.value" style="width: 100%">            <el-table-column prop="image" label="图片" width="100">              <template #default="scope">                <img :src="scope.row.image" alt="Product Image" width="50" />              </template>            </el-table-column>            <el-table-column prop="name" label="商品名称" />            <el-table-column prop="description" label="商品描述" />            <el-table-column prop="price" label="价格" />            <el-table-column prop="source" label="商品链接">              <template #default="scope">                <a :href="scope.row.link" target="\_blank">{{ scope.row.source }}</a>              </template>            </el-table-column>            <el-table-column prop="categoryName" label="商品分类" />          </el-table>        </div>        <template #footer>          <span class="dialog-footer">            <el-button @click="showModal = false">关闭</el-button>          </span>        </template>      </el-dialog>    </div>  </template>  <script setup>  import { reactive, ref, onMounted ,computed} from 'vue';  import { getAllMerchant ,ChangeMerchantStatus} from '../api/merchant'  import {admingetAllCategory} from '../api/category'  import { getAllProduct } from '../api/product'    //修改商家账号状态  const status = ref({    id: '',    status: '',  });  const toggleStatus = (row) => {    const isConfirmed = window.confirm('确定要修改该商家状态吗？');      if (isConfirmed) {      status.value.id = row.id    row.status = row.status == '1' ? '0' : '1';    status.value.status = row.status    console.log(status.value.id)    console.log(status.value.status)    ChangeMerchantStatus(status.value)    } else {      // 用户点击了“取消”，不执行操作      console.log('用户取消操作');    }    };    const showModal = ref(false);  const selectedMerchant = ref(null);    //获取商品列表  const product\_M = ([])  //点击打开商品列表  const showProductModal = (merchant) => {    console.log(merchant.username)    const merchantId = merchant.id    console.log(merchantId)    selectedMerchant.value = merchant.username;    console.log(paginatedProducts.value)    product\_M.value = paginatedProducts.value.filter(product => product.merchantId == merchantId);    console.log(product\_M.value)    showModal.value = true;  };  const productList = reactive([]);  const categories = reactive([]);  const categorieMap = reactive([]);  const merchants = reactive([]);  const merchantMap = reactive([]);  // 查询所有商品数据  const getAllProductList = async () => {    try {      const res = await getAllProduct();      console.log('获取商品列表成功:', res.data.data);      // 确保响应式      const filteredProduct = res.data.data.filter(product => product.status == 1);       // 替换ID为名称      const productsWithNames = filteredProduct.map(product => {        return {          ...product,          categoryName: categorieMap[product.categoryId] || '未知分类',          merchantName: merchantMap[product.merchantId] || '未知商家'        };      });      productList.splice(0, productList.length, ...productsWithNames);      console.log(productList)    } catch (error) {      console.error('获取商品列表失败:', error);    }  };  // 获取分类列表  const getAllCategory = async () => {    try {      const res = await admingetAllCategory();      console.log('获取用户成功:', res.data.data);      // 确保响应式      categories.splice(0, categories.length, ...res.data.data);      categorieMap.splice(0, categorieMap.length, ...res.data.data);      res.data.data.forEach(category => {        categorieMap[category.id] = category.name;      });    } catch (error) {      console.error('获取用户失败:', error);    }  };  //获取所有商家列表实现  const merchantList = reactive([]);  const getMerchant = async () => {    try {      const res = await getAllMerchant();      console.log('获取用户成功:', res.data.data);      // 使用 merchantList.splice 来确保响应式      merchantList.splice(0, merchantList.length, ...res.data.data);      merchantMap.splice(0, merchants.length, ...res.data.data);      res.data.data.forEach(merchant => {        merchantMap[merchant.id] = merchant.username;      });  // 这个步骤确保 userList 被更新并且是响应式的    } catch (error) {      console.error('获取用户失败:', error);    }  };  // 价格排序  const sortType = ref('');  // 搜索的关键词  const searchText = ref('');  // 搜索的分类  const selectedCategory = ref('');  // 分页当前页码  const currentPage = ref(1);  // 一页展示数据数量  const pageSize = ref(20);  // 计算属性，用于根据搜索文本和选定类别过滤产品  const filteredProducts = computed(() => {    // 创建产品列表的副本，以避免修改原始数组    let products = [...productList];      // 如果有搜索文本，按名称过滤产品    if (searchText.value) {      products = products.filter((product) =>        product.name.toLowerCase().includes(searchText.value.toLowerCase())      );    }      // 如果选择了类别，则按类别过滤产品    if (selectedCategory.value) {      products = products.filter((product) => product.categoryName == selectedCategory.value);    }    // 根据用户选择的排序类型对产品进行排序    if (sortType.value) { // 只有在用户选择了排序方式时才进行排序      products.sort((a, b) => {        return sortType.value === 'priceAsc' ? a.price - b.price : b.price - a.price;      });    }      // 返回过滤后的产品列表    return products;  });  // 计算属性，用于分页显示过滤后的产品  const paginatedProducts = computed(() => {    // 根据当前页面和页面大小计算起始索引    const start = (currentPage.value - 1) \* pageSize.value;    // 计算用于切片数组的结束索引    const end = start + pageSize.value;    // 返回当前页面的切片产品数组    return filteredProducts.value.slice(start, end);  });  onMounted(() => {    getMerchant()    getAllCategory()    console.log("查询商家列表成功")    getAllProductList()  });  </script>  <style scoped>  .merchant-avatar {    width: 50px;    height: 50px;    border-radius: 50%;  }  </style>  <template>    <div class="order-management-container">      <h2>订单管理</h2>      <div class="filters">        <div class="filter-item">          <span>订单编号:</span>          <el-input v-model="orderNo" placeholder="输入订单编号" />          <el-button type="primary" @click="searchOrders">查询</el-button>        </div>      </div>      <el-table :data="paginatedOrders" style="width: 100%">        <el-table-column prop="id" label="订单ID" width="120"></el-table-column>        <el-table-column prop="orderNumber" label="订单编号" width="180"></el-table-column>        <el-table-column prop="productName" label="交易商品" width="180"></el-table-column>        <!-- <el-table-column prop="productLink" label="商品链接" width="180">          <template #default="scope">            <a :href="scope.row.productLink" target="\_blank">{{ scope.row.productLink }}</a>          </template>        </el-table-column>   -->        <el-table-column prop="createTime" label="交易时间" width="180"></el-table-column>        <el-table-column prop="userName" label="买家" width="120"></el-table-column>        <el-table-column prop="merchantName" label="卖家" width="120"></el-table-column>        <el-table-column prop="status" label="订单状态" width="120">          <template #default="scope">            <el-tag :type="getStatusTagType(scope.row.status)">{{ getStatusText(scope.row.status) }}</el-tag>          </template>        </el-table-column>      </el-table>      <div class="pagination">        <el-pagination          v-model:current-page="currentPage"          v-model:page-size="pageSize"          :total="filteredOrders.length"          layout="prev, pager, next"          @current-change="handlePageChange"        />      </div>    </div>  </template>  <script setup>  import {  computed ,onMounted,ref} from 'vue';  import { getAllOrder } from '@/api/order';  import { getAllMerchant } from '@/api/merchant';  import { getAllProduct } from '@/api/product';  import { admingetalluser } from '@/api/user';  //获取订单列表  const orderList = ref([]);  const merchantMap = ref([])  const productMap = ref([])  const productMap2 = ref([])  const userMap = ref([])  const getOrderList = async () =>{    try {      const res = await getAllOrder();      console.log('获取订单成功:', res.data.data);      orderList.value.splice(0, orderList.value.length, ...res.data.data);      const ordersWithNames = orderList.value.map(order => {        return {          ...order,          merchantName: merchantMap[order.merchantId] || '未知商家' ,          productName: productMap[order.productId] || '未知商品',          productLink: productMap2[order.productId],          userName:userMap[order.userId]        };      });      orderList.value.splice(0, orderList.value.length, ...ordersWithNames);    } catch (error) {      console.error('获取用户失败:', error);    }  }  //获取商家列表  const getAllMerchants = async () => {    try {      const res = await getAllMerchant();      console.log('获取商家成功:', res.data.data);      // 确保响应式        merchantMap.value.splice(0, merchantMap.value.length, ...res.data.data);      res.data.data.forEach(merchant => {        merchantMap[merchant.id] = merchant.username;      });        } catch (error) {      console.error('获取用户失败:', error);    }  }  //获取产品列表  const getAllProductList = async () => {    try {      const res = await getAllProduct();      console.log('获取商品列表成功:', res.data.data);      productMap.value.splice(0, productMap.value.length, ...res.data.data);      res.data.data.forEach(product => {        productMap[product.id] = product.name;        });      productMap2.value.splice(0, productMap2.value.length, ...res.data.data);      res.data.data.forEach(product => {        productMap2[product.id] = product.source;      });    } catch (error) {      console.error('获取商品列表失败:', error);    }  };  //获取用户列表  const getAllUser = async () => {    try {      const res = await admingetalluser();      console.log('获取用户成功:', res.data.data);      userMap.value.splice(0, userMap.value.length, ...res.data.data);      res.data.data.forEach(user => {        userMap[user.id] = user.username;      });    } catch (error) {      console.error('获取用户失败:', error);    }  };  const currentPage = ref(1);  const pageSize = ref(10);  const orderNo = ref('');  const filteredOrders = computed(() => {    let orders = [...orderList.value];    if (orderNo.value) {      orders = orders.filter((order) =>        order.orderNumber.toLowerCase().includes(orderNo.value.toLowerCase())      );    }    return orders;  });  const paginatedOrders = computed(() => {    const start = (currentPage.value - 1) \* pageSize.value;    const end = start + pageSize.value;    return filteredOrders.value.slice(start, end);  });  const searchOrders = () => {    currentPage.value = 1;  };  const handlePageChange = () => {    // 处理分页切换  };  //处理状态显示  const getStatusTagType = (status) => {    switch (status) {      case "1":        return 'success';      case 0:        return 'warning';      case -1:        return 'danger';      default:        return '';    }  };  const getStatusText = (status) => {      switch (status) {        case 1:          return '已完成';        case 0:          return '未支付';        case -1:          return '已取消';        default:          return '未知状态'; // 可选的默认状态      }    }    onMounted(() => {    getAllProductList()    .then(getAllMerchants)    .then(getAllUser)    .then(getOrderList)        });  </script>  <style scoped>  .order-management-container {    padding: 20px;  }  .filters {    display: flex;    justify-content: space-between;    align-items: center;    margin-bottom: 20px;  }  .filter-item {    display: flex;    align-items: center;  }  .filter-item span {    margin-right: 10px;    width: 120px;  }  .filter-item .el-input,  .filter-item .el-select {    margin-right: 10px;  }  .pagination {    display: flex;    justify-content: center;    margin-top: 20px;  }  </style>  <template>    <div>      <h1>软件上架审核</h1>      <el-table :data="paginatedProducts" style="width: 100%">        <el-table-column prop="name" label="软件名称" />        <el-table-column prop="price" label="软件价格" />        <el-table-column prop="categoryName" label="软件分类" />        <el-table-column prop="merchantName" label="商家" />        <el-table-column label="操作">          <template #default="scope">            <div style="display: flex; justify-content: center;">              <el-button type="success" @click="approveProduct(scope.row)">同意</el-button>              <el-button type="danger" @click="rejectProduct(scope.row)">拒绝</el-button>              <el-button type="primary" @click="showProductDetails(scope.row)">查看详情</el-button>            </div>          </template>        </el-table-column>      </el-table>      <el-dialog v-model="showDetailsDialog" title="商品详情" width="50%" center>      <div v-if="selectedProduct" class="product-details">        <div class="product-image-video-container">          <!-- 上方展示动态图片 -->        <img          :src="selectedProduct.image"            alt="产品图片"            class="product-image"          />           <!-- 视频区域 -->          <div class="video-container">            <video              class="product-video"              v-if="selectedProduct.video"              :src="selectedProduct.video"              controls            >              您的浏览器不支持 video 标签。            </video>            <div v-else>没有可播放的视频！</div>          </div>        </div>        <div class="product-info">          <h2>{{ selectedProduct.name }}</h2>          <p>{{ selectedProduct.description }}</p>          <p>价格: {{ selectedProduct.price }}</p>          <p>分类: {{ selectedProduct.categoryName }}</p>          <p>商家: {{ selectedProduct.merchantName }}</p>          <div class="download-button-container">            <el-button type="primary" @click="downloadProduct(selectedProduct)">              下载软件            </el-button>          </div>        </div>      </div>    </el-dialog>    </div>  </template>  <script setup>    import { ref, reactive, computed ,onMounted} from 'vue';  import {admingetAllCategory} from '../api/category'  import { getAllProduct } from '../api/product'  import { getAllMerchant } from '../api/merchant'  import { updateStatus } from '../api/product';  import { ElMessage } from 'element-plus';      const productList = reactive([]);  const categories = reactive([]);  const categorieMap = reactive([]);  const merchants = reactive([]);  const merchantMap = reactive([]);  // 查询所有商品数据  const getAllProductList = async () => {    try {      const res = await getAllProduct();      console.log('获取商品列表成功:', res.data.data);      // 确保响应式      const filteredProduct = res.data.data.filter(product => product.status == 0);       // 替换ID为名称      const productsWithNames = filteredProduct.map(product => {        return {          ...product,          categoryName: categorieMap[product.categoryId] || '未知分类',          merchantName: merchantMap[product.merchantId] || '未知商家'        };      });      productList.splice(0, productList.length, ...productsWithNames);      console.log(productList)    } catch (error) {      console.error('获取商品列表失败:', error);    }  };  // 获取分类列表  const getAllCategory = async () => {    try {      const res = await admingetAllCategory();      console.log('获取用户成功:', res.data.data);      // 确保响应式      categories.splice(0, categories.length, ...res.data.data);      categorieMap.splice(0, categorieMap.length, ...res.data.data);      res.data.data.forEach(category => {        categorieMap[category.id] = category.name;      });    } catch (error) {      console.error('获取用户失败:', error);    }  };  //获取商家信息  const getAllMerchants = async () => {    try {      const res = await getAllMerchant();      console.log('获取商家成功:', res.data.data);      // 确保响应式      merchants.splice(0, merchants.length, ...res.data.data);      merchantMap.splice(0, merchants.length, ...res.data.data);      res.data.data.forEach(merchant => {        merchantMap[merchant.id] = merchant.username;      });    } catch (error) {      console.error('获取用户失败:', error);    }  }  // 价格排序  const sortType = ref('');  // 搜索的关键词  const searchText = ref('');  // 搜索的分类  const selectedCategory = ref('');  // 分页当前页码  const currentPage = ref(1);  // 一页展示数据数量  const pageSize = ref(20);  // 计算属性，用于根据搜索文本和选定类别过滤产品  const filteredProducts = computed(() => {    // 创建产品列表的副本，以避免修改原始数组    let products = [...productList];      // 如果有搜索文本，按名称过滤产品    if (searchText.value) {      products = products.filter((product) =>        product.name.toLowerCase().includes(searchText.value.toLowerCase())      );    }      // 如果选择了类别，则按类别过滤产品    if (selectedCategory.value) {      products = products.filter((product) => product.categoryName == selectedCategory.value);    }    // 根据用户选择的排序类型对产品进行排序    if (sortType.value) { // 只有在用户选择了排序方式时才进行排序      products.sort((a, b) => {        return sortType.value === 'priceAsc' ? a.price - b.price : b.price - a.price;      });    }      // 返回过滤后的产品列表    return products;  });  // 计算属性，用于分页显示过滤后的产品  const paginatedProducts = computed(() => {    // 根据当前页面和页面大小计算起始索引    const start = (currentPage.value - 1) \* pageSize.value;    // 计算用于切片数组的结束索引    const end = start + pageSize.value;    // 返回当前页面的切片产品数组    return filteredProducts.value.slice(start, end);  });  const showDetailsDialog = ref(false);  const selectedProduct = ref(null);  const approveProduct = async (product) => {    // 审核通过逻辑    const isConfirmed = window.confirm('确定同意通过审核吗？');      if (isConfirmed) {      try {      const res = await updateStatus({ id: product.id, status: 1 });      console.log(res)      getAllProductList()      ElMessage.success("操作成功~")    } catch (error) {      console.error('下架失败', error);    }    } else {      // 用户点击了“取消”，不执行操作      console.log('用户取消操作');    }  };  const rejectProduct = async (product) => {    // 审核失败逻辑    const isConfirmed = window.confirm('确定拒绝该审核吗？');      if (isConfirmed) {      try {      const res = await updateStatus({ id: product.id, status: -1 });      console.log(res)      getAllProductList()      ElMessage.success("操作成功~")    } catch (error) {      console.error('下架失败', error);    }    } else {      // 用户点击了“取消”，不执行操作      console.log('用户取消操作');    }    };  //展示商品详情  const showProductDetails = (product) => {    selectedProduct.value = product;    showDetailsDialog.value = true;  };  const downloadProduct = (product) => {      // 创建一个隐藏的 <a> 元素    const url = product.source;    const link = document.createElement('a');    link.href = url;    link.download = product.name; // 可选，设置下载文件的默认文件名    //link.target = '\_blank'; // 可选，避免干扰用户当前页面    document.body.appendChild(link); // 将 <a> 添加到文档中    link.click(); // 模拟点击下载    document.body.removeChild(link); // 下载完成后移除 <a>  };  onMounted(() => {    getAllCategory()    getAllMerchants().then(() =>{getAllProductList()})    // 在组件挂载时可以执行一些初始化操作  });  </script>  <style scoped>  .product-details {    display: flex;    justify-content: space-between;    align-items: center;  }  .product-image-container {    margin-bottom: 20px;  }  .product-image {    width: 100%;    height: 200px;    cursor: pointer;  }  .download-button-container {    margin-top: 20px;  }  .product-info {    flex-basis: 50%;    text-align: left;    padding-left: 20px;  }  .download-button-container {    display: flex;    justify-content: center;    margin-top: 20px;  }  </style>  <style scoped>  .product-details {    display: flex;  }  .product-image-video-container {    display: flex;    flex-direction: column; /\* 垂直排列 \*/    width: 50%; /\* 左侧宽度 \*/  }  .product-image {    width: 300px; /\* 使图片自适应 \*/    height: 200px; /\* 高度自适应 \*/  }  .video-container {    margin-top: 10px; /\* 图片与视频之间的间距 \*/  }  .product-video {    width: 300px ;/\* 视频宽度自适应 \*/    height: 200px; /\* 高度自适应 \*/  }  .product-info {    padding-left: 20px; /\* 右侧信息的内边距 \*/  }  </style>  <template>    <div>      <h1>软件产品管理</h1>      <div class="filters">        <div class="filter-item">          <span>排序方式:</span>          <el-select v-model="sortType">            <el-option label="默认排序" value=""></el-option>            <el-option label="价格升序" value="priceAsc"></el-option>            <el-option label="价格降序" value="priceDesc"></el-option>          </el-select>        </div>        <div class="filter-item">          <span>软件查询:</span>          <el-input v-model="searchText" placeholder="" />          <el-button type="primary" @click="searchProducts">搜索</el-button>        </div>        <div class="filter-item">          <span>软件分类:</span>          <el-select v-model="selectedCategory" @change="filterByCategory">            <el-option label="全部" value=""></el-option>            <el-option v-for="category in categories" :key="category.name" :label="category.name" :value="category.name"></el-option>          </el-select>        </div>      </div>      <el-table :data="paginatedProducts" style="width: 100%">        <el-table-column prop="image" label="软件图片" width="100">          <template #default="scope">            <img :src="scope.row.image" alt="商品图片" class="product-image" />          </template>        </el-table-column>        <el-table-column prop="name" label="软件名称" />        <el-table-column prop="price" label="软件价格" />        <el-table-column prop="categoryName" label="软件分类" />        <el-table-column prop="merchantName" label="商家" />        <el-table-column label="操作">          <template #default="scope">            <el-button type="danger" @click="deleteProduct(scope.row)">强制下架</el-button>          </template>        </el-table-column>      </el-table>      <div class="pagination">        <el-pagination          v-model:current-page="currentPage"          v-model:page-size="pageSize"          :total="filteredProducts.length"          layout="prev, pager, next"          @current-change="handlePageChange"        />      </div>    </div>  </template>  <script setup>  import { ref, reactive, computed ,onMounted} from 'vue';  import {admingetAllCategory} from '../api/category'  import { getAllProduct ,updateStatus} from '../api/product'  import { getAllMerchant } from '../api/merchant'  import { ElMessage } from 'element-plus';    const productList = reactive([]);  const categories = reactive([]);  const categorieMap = reactive([]);  const merchants = reactive([]);  const merchantMap = reactive([]);  // 查询所有商品数据  const getAllProductList = async () => {    try {      const res = await getAllProduct();      console.log('获取商品列表成功:', res.data.data);      // 确保响应式      const filteredProduct = res.data.data.filter(product => product.status == 1);       // 替换ID为名称      const productsWithNames = filteredProduct.map(product => {        return {          ...product,          categoryName: categorieMap[product.categoryId] || '未知分类',          merchantName: merchantMap[product.merchantId] || '未知商家'        };      });      productList.splice(0, productList.length, ...productsWithNames);      console.log(productList)    } catch (error) {      console.error('获取商品列表失败:', error);    }  };  //强制下架产品  const deleteProduct = async (row) =>{    try {      const res = await updateStatus({ id: row.id, status: -1 });      console.log(res)      getAllProductList()      ElMessage.success("操作成功~")    } catch (error) {      console.error('下架失败', error);    }  }  // 获取分类列表  const getAllCategory = async () => {    try {      const res = await admingetAllCategory();      console.log('获取用户成功:', res.data.data);      // 确保响应式      categories.splice(0, categories.length, ...res.data.data);      categorieMap.splice(0, categorieMap.length, ...res.data.data);      res.data.data.forEach(category => {        categorieMap[category.id] = category.name;      });    } catch (error) {      console.error('获取用户失败:', error);    }  };  //获取商家信息  const getAllMerchants = async () => {    try {      const res = await getAllMerchant();      console.log('获取商家成功:', res.data.data);      // 确保响应式      merchants.splice(0, merchants.length, ...res.data.data);      merchantMap.splice(0, merchants.length, ...res.data.data);      res.data.data.forEach(merchant => {        merchantMap[merchant.id] = merchant.username;      });    } catch (error) {      console.error('获取用户失败:', error);    }  }  // 价格排序  const sortType = ref('');  // 搜索的关键词  const searchText = ref('');  // 搜索的分类  const selectedCategory = ref('');  // 分页当前页码  const currentPage = ref(1);  // 一页展示数据数量  const pageSize = ref(20);  // 计算属性，用于根据搜索文本和选定类别过滤产品  const filteredProducts = computed(() => {    // 创建产品列表的副本，以避免修改原始数组    let products = [...productList];      // 如果有搜索文本，按名称过滤产品    if (searchText.value) {      products = products.filter((product) =>        product.name.toLowerCase().includes(searchText.value.toLowerCase())      );    }      // 如果选择了类别，则按类别过滤产品    if (selectedCategory.value) {      products = products.filter((product) => product.categoryName == selectedCategory.value);    }    // 根据用户选择的排序类型对产品进行排序    if (sortType.value) { // 只有在用户选择了排序方式时才进行排序      products.sort((a, b) => {        return sortType.value === 'priceAsc' ? a.price - b.price : b.price - a.price;      });    }      // 返回过滤后的产品列表    return products;  });  // 计算属性，用于分页显示过滤后的产品  const paginatedProducts = computed(() => {    // 根据当前页面和页面大小计算起始索引    const start = (currentPage.value - 1) \* pageSize.value;    // 计算用于切片数组的结束索引    const end = start + pageSize.value;    // 返回当前页面的切片产品数组    return filteredProducts.value.slice(start, end);  });  // 函数，当搜索产品时将当前页面重置为1  const searchProducts = () => {    currentPage.value = 1;  // 将当前页面重置为1  };    onMounted(() => {    getAllCategory()    getAllMerchants().then(() =>{getAllProductList()})    });  </script>  <style scoped>  .product-image {    width: 50px;    height: 50px;    object-fit: cover;  }  .filters {    display: flex;    justify-content: space-between;    align-items: center;    margin-bottom: 20px;  }  .filter-item {    display: flex;    align-items: center;  }  .filter-item span {    margin-right: 10px;    width: 130px;  }  .filter-item .el-input,  .filter-item .el-select {    margin-right: 10px;  }  .pagination {    display: flex;    justify-content: center;    margin-top: 20px;  }  </style>  <template>    <div>      <h1>用户管理</h1>      <el-table :data="userList" style="width: 100%">        <el-table-column prop="avatar" label="头像" width="100">          <template #default="scope">            <img :src="scope.row.avatar" alt="Avatar" class="user-avatar" />          </template>        </el-table-column>        <el-table-column prop="username" label="用户名" />        <el-table-column prop="phone" label="手机号" />        <el-table-column prop="createTime" label="创建时间" />        <el-table-column prop="updateTime" label="最后修改时间" />        <el-table-column prop="status" label="账号状态">          <template #default="scope">            <el-button              :type="scope.row.status == '1' ? 'success' : 'danger'"              @click="toggleStatus(scope.row)"            >              {{ scope.row.status == '1' ? '正常' : '禁用' }}            </el-button>          </template>        </el-table-column>      </el-table>    </div>  </template>  <script setup>  import { ref } from 'vue';  import { admingetalluser,adminChangeStatus} from '@/api/user';  import { reactive, onMounted } from 'vue';    const status = ref({    id: '',    status: '',  });  // 切换用户状态  const toggleStatus = (row) => {    // 根据当前状态切换为另一状态    const isConfirmed = window.confirm('确定要修改用户状态吗？');    if (isConfirmed) {      const userId = row.id;    status.value.id = userId    row.status = row.status == '1' ? '0' : '1';    status.value.status = row.status    console.log(status.value.id)    console.log(status.value.status)    handleChangeStatus()    ElMessage.success("操作成功")      console.log('用户确认操作');      // 执行相关操作的代码    } else {      // 用户点击了“取消”，不执行操作      console.log('用户取消操作');    }    };   const handleChangeStatus = async () => {    try {      const res = await adminChangeStatus(status.value);      console.log(res);    } catch (error) {      console.error('更新状态失败:', error);      // 如果更新失败，可能需要将状态重置为原来的状态      // 这里需要根据实际情况来决定是否需要重置状态    }  }  // 创建响应式数组，用于存储用户信息  const userList = reactive([]);  // 获取所有用户方法实现  const getAllUser = async () => {    try {      const res = await admingetalluser();      console.log('获取用户成功:', res.data.data);      // 使用 userList.splice 来确保响应式      userList.splice(0, userList.length, ...res.data.data); // 这个步骤确保 userList 被更新并且是响应式的    } catch (error) {      console.error('获取用户失败:', error);    }  };  // 页面初始化后调用方法获取用户数据  onMounted(() => {    getAllUser();  });  </script>  <style scoped>  .user-avatar {    width: 50px;    height: 50px;    border-radius: 50%;  }  </style>  <script setup>  </script>  <template>      <router-view />  </template>    import { createApp } from 'vue'  import App from './App.vue'  import router from './router'  import pinia from './stores'  const app = createApp(App)  app.use(pinia)  app.use(router)  app.mount('#app')  import { fileURLToPath, URL } from 'node:url'  import { defineConfig } from 'vite'  import vue from '@vitejs/plugin-vue'  import AutoImport from 'unplugin-auto-import/vite'  import Components from 'unplugin-vue-components/vite'  import { ElementPlusResolver } from 'unplugin-vue-components/resolvers'  // https://vite.dev/config/  export default defineConfig({    server: {      proxy: {        '/api': {          target: 'http://localhost:8080',          changeOrigin: true,          rewrite: (path) => path.replace(/^\/api/, '')        }      }    },    plugins: [      vue(),      AutoImport({        resolvers: [ElementPlusResolver()]      }),      Components({        resolvers: [ElementPlusResolver()]      })    ],    resolve: {      alias: {        '@': fileURLToPath(new URL('./src', import.meta.url))      },    },  }) |