

“宠物之家” 宠物服务平台

系统说明文档

姓名(组长): 梁笛笛 学号: 2312190403 班级: 计科 2303

姓名: 张昕玥玥 学号: 2312190404 班级: 计科 2303

一、 项目介绍, 背景及意义

宠物之家是一个基于现代 Web 技术的综合性宠物生活服务平台, 集成了宠物档案管理、专业洗护预约、宠物用品商城、AI 智能咨询四大核心功能。系统采用 Vue 3 + Spring Boot + MySQL 技术栈, 面向宠物主人提供一站式数字化宠物服务解决方案。通过模块化设计和响应式界面, 平台实现了从宠物信息记录、服务预约到商品购买的完整闭环, 为用户打造便捷、专业、智能的宠物养护体验。

随着我国宠物经济的快速增长和养宠人群的年轻化、专业化趋势, 传统宠物服务模式已无法满足现代养宠需求。宠物之家应运而生, 旨在通过技术创新解决这些痛点, 填补市场空白, 满足新一代宠物主人对高品质、便捷化宠物服务的迫切需求。

二、 技术栈及架构

1. 前端: Vue 3 + Vite+ Iconify + Element Plus
2. 后端: Java Spring Boot 框架
3. 数据库: MySQL
4. 图片存储: 阿里云对象存储 OSS
5. 系统集成: 通过 Nginx 反向代理实现前后端分离和跨域访问
6. AI 服务: DeepSeek API 集成

2.1 数据库设计

一、用户模块

表名	说明	字段数	备注
users	用户表	12	存储用户基本信息、登录凭证、角色等
user_addresses	用户收货地址表	11	支持多地址，含省市区详细地址
user_login_logs	用户登录日志表	6	记录登录 IP、时间、状态等信息
user_favorites	用户收藏表	5	记录用户收藏的商品

二、宠物管理模块

表名	说明	字段数	备注
pets	宠物信息表	15	记录宠物类型、品种、性别、生日、健康信息等

三、商品模块

表名	说明	字段数	备注
products	商品主表	23	商品基本信息、价格、库存、标签、评分等
product_categories	商品分类表	9	分类管理，含图标与排序
product_specifications	商品规格表	12	多规格支持，价格、库存、标签颜色等
product_details	商品详情表	4	存储 HTML 格式的详情描述
product_params	商品参数表	6	商品参数（如材质、尺寸等）
product_faqs	商品常见问题表	8	FAQ 问答，支持排序与启用状态
product_rating_distribution	商品评价分布表	6	按星级统计评价数量与百分比

四、购物车与订单模块

表名	说明	字段数	备注
cart_items	购物车表	11	用户购物车商品记录
orders	订单主表	21	订单状态、金额、收货信息、支付时间等
order_items	订单商品表	12	订单中每个商品的详细信息

五、预约服务模块

表名	说明	字段数	备注
service_items	预约服务项目表	14	服务项目基本信息、价格、时长、是否推荐等
service_details	预约服务详情表	11	服务详细介绍、适用宠物、推荐护理项等
service_care_items	护理项表	9	护理项目分类、单价、是否单次选购等
service_appointments	预约订单表	15	预约订单主表，含日期、时间段、总金额、状态等
service_order_items	预约订单项目表	12	预约订单中的服务项目明细
service_records	预约记录表	16	支付状态、评价、服务记录、完成时间等

六、表结构详细说明（部分关键字段）

表名	主要字段示例
users	id, username, password, nickname, phone, role, status, last_login_at
pets	user_id, pet_type, name, breed, gender, birth_date, weight, avatar_url
products	category_id, name, brand, price, original_price, stock, sales, rating, status, image_url
orders	order_no, user_id, status, total_amount, final_amount, payment_method, receiver_name, receiver_address
service_appointments	order_no, user_id, pet_id, appointment_date, time_slot, order_total_amount, status
service_records	appointment_id, payment_status, payment_method, rating, review_content, completed_at

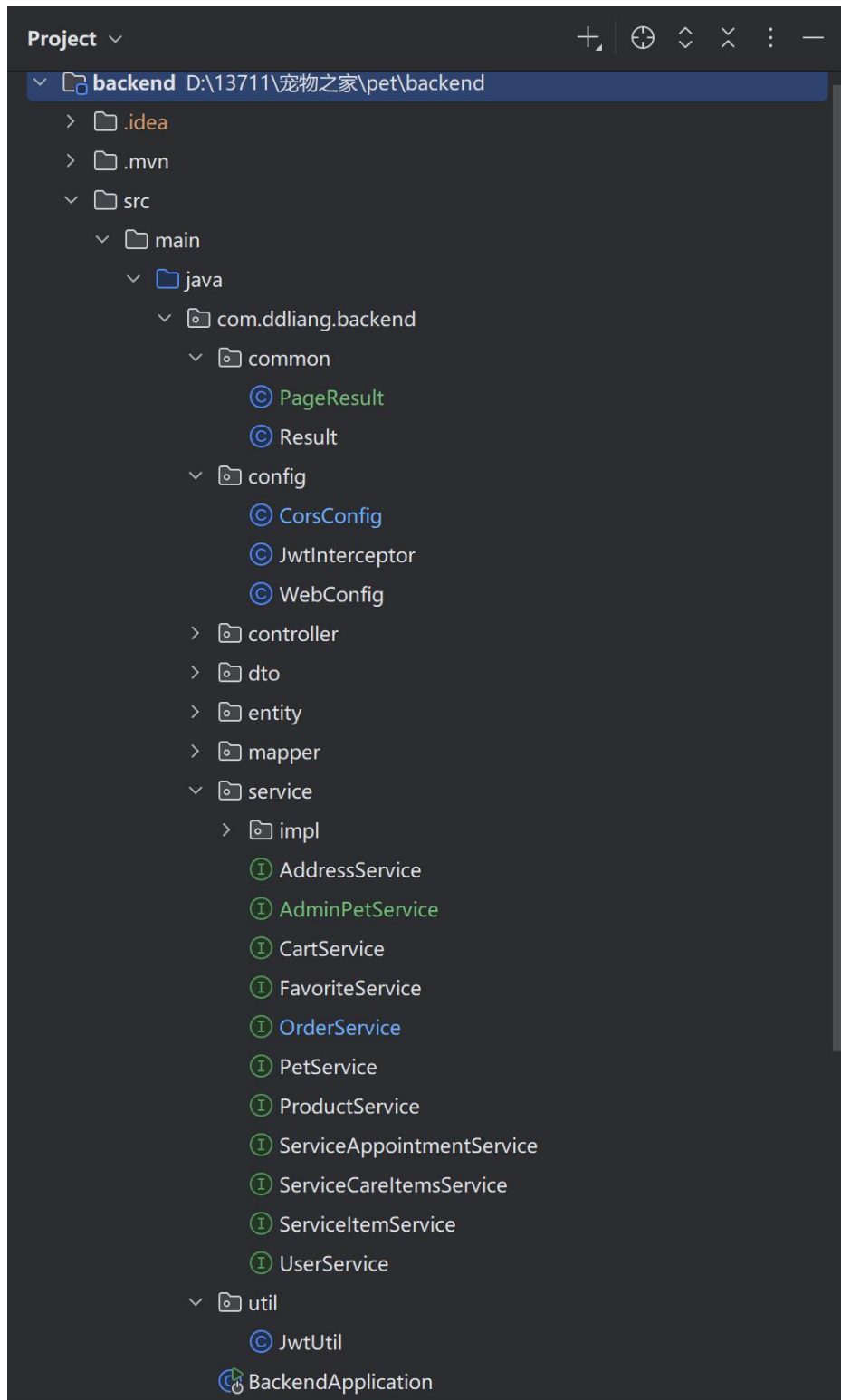
总结

- **总表数:** 21 张表
- **核心模块:** 用户、宠物、商品、订单、预约服务
- **扩展性:** 支持商品多规格、多分类、FAQ、评价分布、护理项扩展等
- **关联关系:** 通过外键（如 user_id、product_id、pet_id）实现数据关联与级联操作

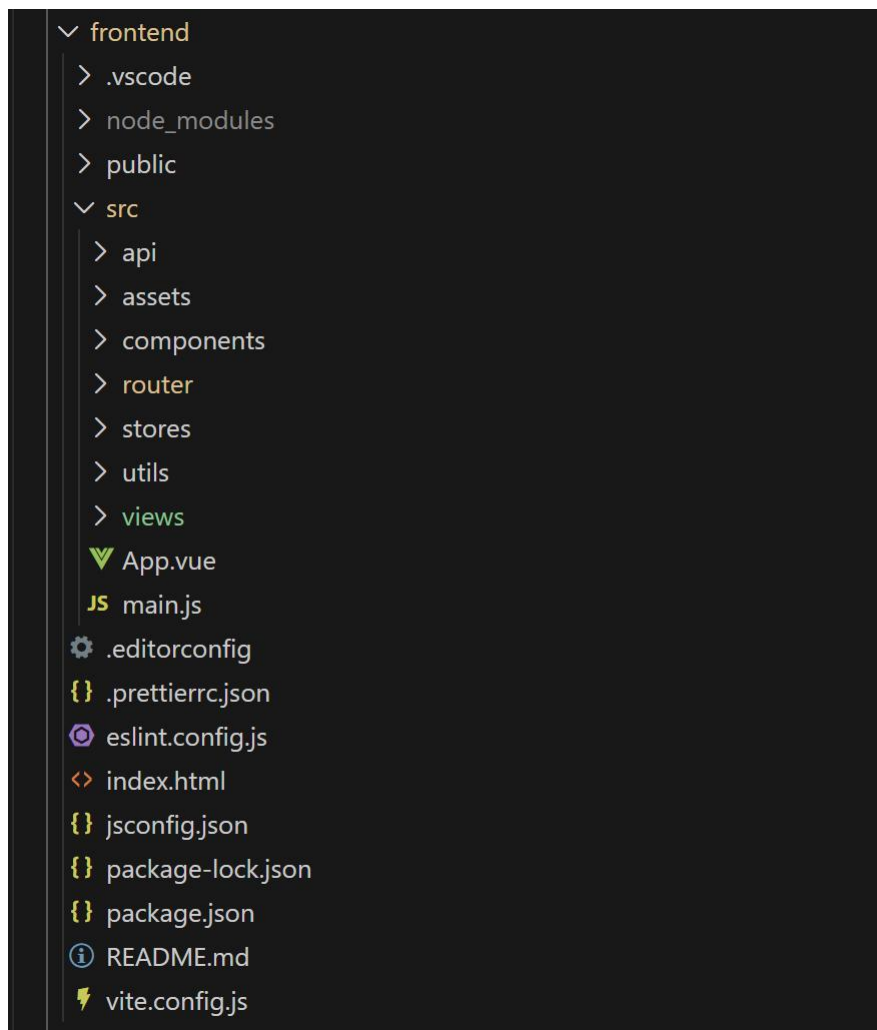
2.2 后端架构

Java 后端采用经典的 controller/service/entity 层次结构, 数据库采用 Mysql,

下图是 java 后端的代码结构图:



2.3 前端架构



2.3 核心功能

- **用户认证系统**：注册、登录、个人中心管理
- **商品管理**：宠物用品商城、购物车、订单管理、支付
- **宠物管理**：用户宠物信息管理、宠物档案
- **服务预约**：宠物护理服务预约、预约管理
- **管理后台**：用户管理、商品管理、订单管理、预约管理
- **AI 助手**：宠物健康咨询、饲养建议

前端部署在 Nginx 服务器的 -- 目录下；Nginx 反向代理模式，将来自前端的请

求转发到 Java 后端，同时 Nginx 负责前后端的跨域访问。

三. 关键技术及解决方案

3.1 JWT Token 身份认证（梁笛笛负责）

1. 实现 JWT-token
2. 登录时生成 JWT Token 的关键代码

```
22     public class AuthController {
49         /**
50          * 用户登录
51          */
52         @PostMapping("/login") 吕 liang725
53         public Result<?> login(@Valid @RequestBody LoginRequest request, BindingResult bindingResult) {
54             // 参数校验
55             if (bindingResult.hasErrors()) {
56                 return Result.error( code: 400, bindingResult.getFieldError().getDefaultMessage());
57             }
58
59             try {
60                 String token = userService.login(request);
61                 User user = userService.findByUsername(request.getUsername());
62
63                 Map<String, Object> data = new HashMap<>();
64                 data.put("token", token);
65                 data.put("userId", user.getId());
66                 data.put("username", user.getUsername());
67                 data.put("nickname", user.getNickname());
68                 data.put("avatar", user.getAvatar());
69                 data.put("role", user.getRole()); // 返回用户角色
70
71                 // 同时返回 userInfo 对象，方便前端使用
72                 Map<String, Object> userInfo = new HashMap<>();
73                 userInfo.put("id", user.getId());
74                 userInfo.put("username", user.getUsername());
75                 userInfo.put("nickname", user.getNickname());
76                 userInfo.put("avatar", user.getAvatar());
77                 userInfo.put("role", user.getRole());
78                 data.put("userInfo", userInfo);
79
80                 return Result.success( message: "登录成功", data);
81             } catch (RuntimeException e) {
82                 return Result.error( code: 400, e.getMessage());
83             }
84         }
85     }
```

3.正常访问 api, 验证是否可以访问的代码如下:

```
15 public class JwtInterceptor implements HandlerInterceptor {
36     @Override no usages 2 liang725
37     public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {
38         String path = request.getRequestURI();
39
40         // 放行OPTIONS请求
41         if ("OPTIONS".equals(request.getMethod())) {
42             return true;
43         }
44
45         // 检查是否是公开接口
46         if (isPublicPath(path)) {
47             return true;
48         }
49
50         // 验证Token
51         String authHeader = request.getHeader("Authorization");
52         if (authHeader == null || !authHeader.startsWith("Bearer ")) {
53             response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
54             response.setContentType("application/json;charset=UTF-8");
55             response.getWriter().write("{\"code\":\"401\",\"message\":\"未提供有效的Token\"}");
56             return false;
57         }
58
59         String token = authHeader.substring(beginIndex: 7);
60         try {
61             if (jwtUtil.isTokenExpired(token)) {
62                 response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
63                 response.setContentType("application/json;charset=UTF-8");
64                 response.getWriter().write("{\"code\":\"401\",\"message\":\"Token已过期\"}");
65                 return false;
66             }
67
68             // 将用户信息存入request属性
69             String username = jwtUtil.getUsernameFromToken(token);
70             Integer userId = jwtUtil.getUserIdFromToken(token);
71             request.setAttribute("username", username);
72
73             request.setAttribute("userId", userId);
74
75             return true;
76         } catch (Exception e) {
77             response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
78             response.setContentType("application/json;charset=UTF-8");
79             response.getWriter().write("{\"code\":\"401\",\"message\":\"Token验证失败\"}");
80             return false;
81         }
82     }
```


4、退出后，前端需要清除 token:

```
37   actions: {
38     // 检查认证状态
39     checkAuthStatus() {
40       const token = localStorage.getItem('token')
41       const role = localStorage.getItem('userRole')
42       this.token = token || ''
43       this.userRole = role || ''
44       this.isLoggedIn = !!token
45       return this.isLoggedIn
46     },
47
48     // 设置token和用户信息
49     setToken(token, userInfo = null) {
50       this.token = token
51       localStorage.setItem('token', token)
52       this.isLoggedIn = true
53
54       if (userInfo) {
55         this.userInfo = userInfo
56         this.userRole = userInfo.role || 'user'
57         localStorage.setItem('userRole', this.userRole)
58       }
59     },
60
61     // 清除token
62     clearToken() {
63       this.token = ''
64       this.userInfo = null
65       this.userRole = ''
66       this.isLoggedIn = false
67       localStorage.removeItem('token')
68       localStorage.removeItem('userRole')
69     },

```

3.2 服务预约系统（张昕玥负责）

前端 ServiceAppointmentView.vue 实现:

- 选择服务项目
- 选择宠物
- 选择预约日期和时间段
- 添加特殊需求备注
- 确认预约

管理端预约管理，后端 AdminAppointmentController 实现:

```

@GetMapping(new *)
public Result<?> getAppointmentList(
    HttpServletRequest request,
    @RequestParam(defaultValue = "1") Integer page,
    @RequestParam(defaultValue = "10") Integer pageSize,
    @RequestParam(required = false) String keyword,
    @RequestParam(required = false) String status,
    @RequestParam(required = false) String today,           // 当日筛选
    @RequestParam(required = false) String petType) {       // 动物种类筛选

    // 验证管理员权限
    if (!isAdmin(request)) {
        return Result.error( code: 403, message: "无权访问管理员功能");
    }

    try {
        int offset = (page - 1) * pageSize;
        List<Map<String, Object>> appointments;
        int total;

        // 构建查询参数
        Map<String, Object> params = new HashMap<>();
        params.put("offset", offset);
        params.put("limit", pageSize);

        if (keyword != null && !keyword.trim().isEmpty()) {
            params.put("keyword", keyword.trim());
        }
    }

```

```

        if (keyword != null && !keyword.trim().isEmpty()) {
            params.put("keyword", keyword.trim());
        }

        if (status != null && !status.trim().isEmpty() && !"all".equals(status)) {
            params.put("status", status.trim());
        }

        // 当日筛选
        if (today != null && !today.trim().isEmpty()) {
            params.put("today", today.trim());
        }

        // 动物种类筛选
        if (petType != null && !petType.trim().isEmpty()) {
            params.put("petType", petType.trim());
        }

        // 查询数据
        appointments = appointmentMapper.findAppointmentsForAdmin(params);
        total = appointmentMapper.countAppointmentsForAdmin(params);

        // 构建返回结果
        Map<String, Object> result = new HashMap<>();
        result.put("list", appointments);
        result.put("total", total);
        result.put("page", page);
        result.put("pageSize", pageSize);
    }
}

```

```

// 构建返回结果
Map<String, Object> result = new HashMap<>();
result.put("list", appointments);
result.put("total", total);
result.put("page", page);
result.put("pageSize", pageSize);

return Result.success(result);
} catch (Exception e) {
    e.printStackTrace();
    return Result.error( code: 500, message: "获取预约列表失败: " + e.getMessage());
}
}

```

功能:

- 预约列表查询（分页、搜索、筛选）
- 预约详情查看
- 预约状态更新（pending → confirmed → completed/cancelled）
- 预约统计

3.3 商品模块与图片存储（张昕玥玥）

商品查询功能

后端 ProductController 实现:

```

@GetMapping
public Result<List<ProductResponse>> getProductList(
    @RequestParam(required = false) String category,
    @RequestParam(defaultValue = "5") Integer limit,
    @RequestParam(required = false) String keyword,
    @RequestParam(defaultValue = "1") Integer page,
    @RequestParam(defaultValue = "10") Integer pageSize) {
    try {
        List<ProductResponse> products;

        if (keyword != null && !keyword.trim().isEmpty()) {
            // 如果有搜索关键词, 进行搜索
            products = productService.searchProducts(keyword, page, pageSize);
        } else if (category != null) {
            // 按分类获取商品
            products = productService.getProductsByCategory(category, limit);
        } else {
            // 获取所有商品 (分页)
            products = productService.getAllProducts(page, pageSize);
        }

        return Result.success(products);
    } catch (IllegalArgumentException e) {
        return Result.error( code: 400, e.getMessage());
    } catch (Exception e) {
        return Result.error("获取商品列表失败: " + e.getMessage());
    }
}

```

支持功能：

- 按分类筛选（hot/food/supply/toy）
- 按关键词搜索
- 分页查询

3.6 AI 问答系统（梁笛笛负责）

AI 服务集成

后端 AiController 使用 DeepSeek API:

```
@PostMapping("/chat")
public ResponseEntity<?> chat(@RequestBody Map<String, Object> request) {
    try {
        String userMessage = (String) request.get("message");
        if (userMessage == null || userMessage.trim().isEmpty()) {
            return ResponseEntity.badRequest().body(Map.of(
                k1: "code", v1: 400,
                k2: "message", v2: "消息不能为空"
            ));
        }

        // 构建DeepSeek API请求
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);
        headers.setBearerAuth(apiKey);

        // 系统提示词 - 宠物咨询专家
        String systemPrompt = "你是一个专业的宠物咨询专家，名叫'宠物小助手'。你精通各种宠物的饲养、健康、训练等知识。" +
            "请用友好、专业的语气回答用户关于宠物的问题。如果用户问的不是宠物相关问题，请礼貌地引导他们询问宠物相关话题。" +
            "回答要简洁明了，必要时可以分点列出。";

        Map<String, Object> body = new HashMap<>();
        body.put("model", "deepseek-chat"); // 最便宜的模型
        body.put("messages", Arrays.asList(
            Map.of(k1: "role", v1: "system", k2: "content", systemPrompt),
            Map.of(k1: "role", v1: "user", k2: "content", userMessage)
        ));
        body.put("temperature", 0.7);
        body.put("max_tokens", 500); // 限制token数量节省费用
    } catch (Exception e) {
        // 处理异常
    }
}
```

功能：

- 接收用户消息
- 调用 DeepSeek API 获取回复
- 返回 AI 的回复内容
- 推荐常见问题列表

3.7 实现 Nginx 的跨域（梁笛笛负责）

将项目部署在 VM 虚拟机上

- 1. 下载服务器版本的 ubuntu
- 2. 下载 windterm 工具

WindTerm_2.6.1

在 WindTerm_2.6.1 中搜索

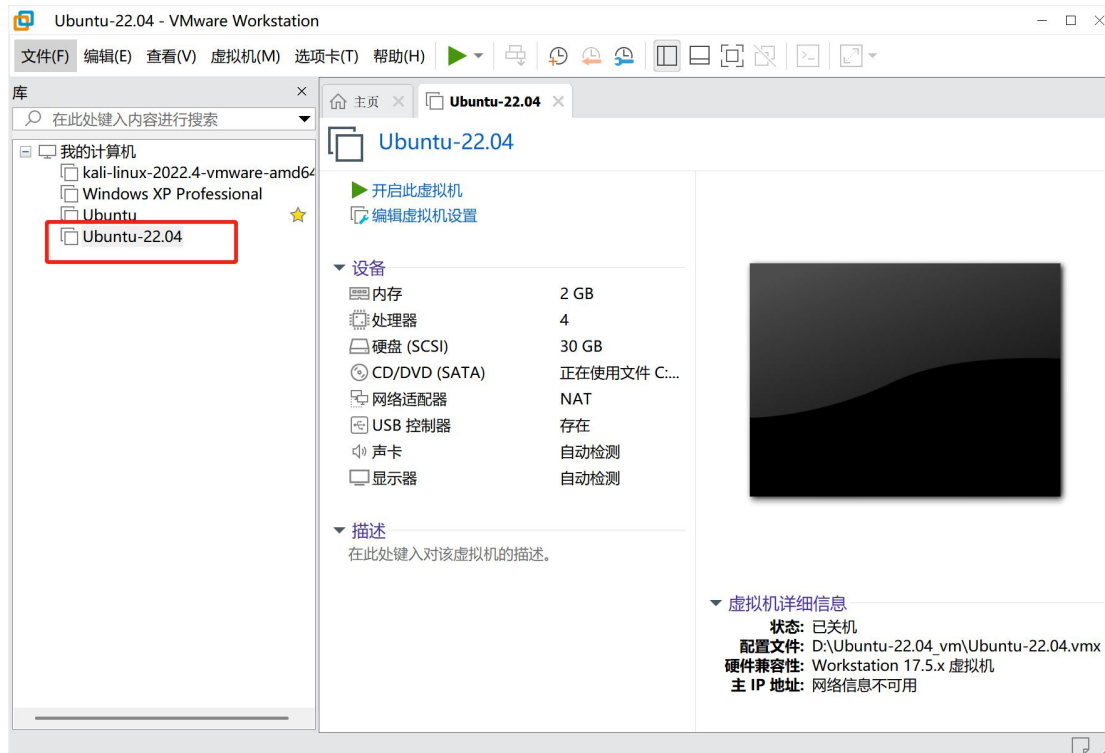
排序

查看

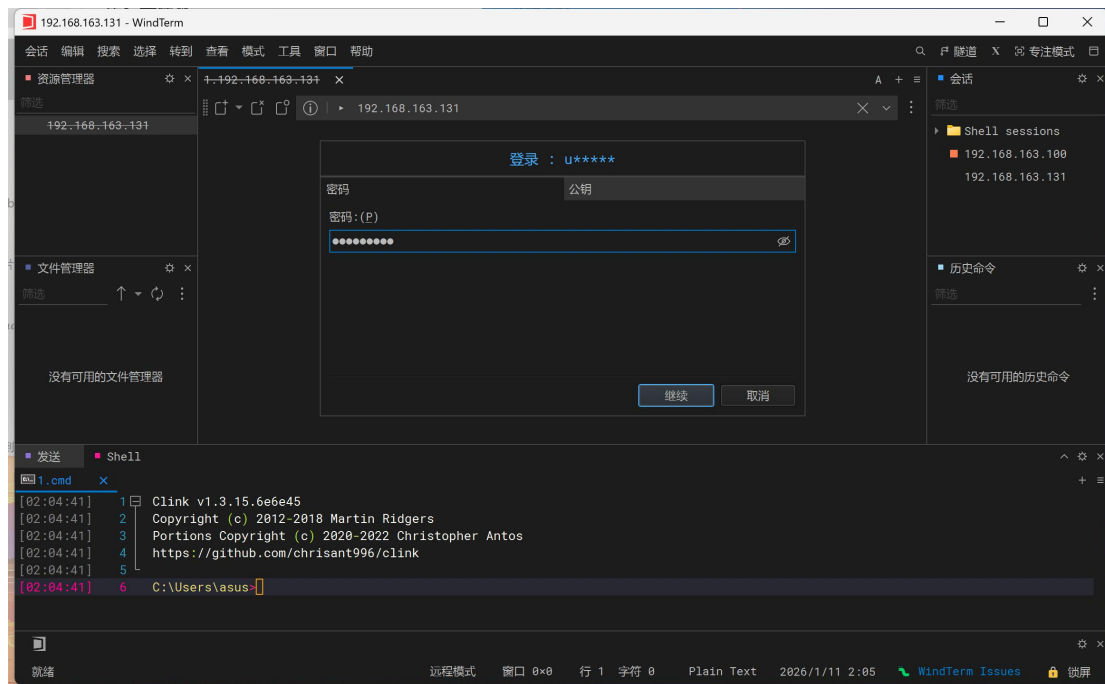
预览

名称	修改日期	类型	大小
.wind	2026/1/4 1:53	文件夹	
global	2024/1/18 4:14	文件夹	
plugins	2024/1/15 13:53	文件夹	
terminal	2024/1/15 13:54	文件夹	
vendors	2024/1/15 13:54	文件夹	
libcrypto.dll	2021/12/21 6:31	应用程序扩展	3,361 KB
license.txt	2023/11/15 10:32	文本文档	43 KB
msvcp140.dll	2021/12/15 2:09	应用程序扩展	553 KB
msvcp140_1.dll	2021/12/15 2:09	应用程序扩展	24 KB
msvcp140_2.dll	2021/12/15 2:09	应用程序扩展	182 KB
profiles.config	2026/1/4 1:53	CONFIG 文件	1 KB
qt.conf	2022/1/15 0:29	CONF 文件	1 KB
Qt5Core.dll	2020/11/6 13:29	应用程序扩展	5,883 KB
Qt5Gui.dll	2020/11/6 13:29	应用程序扩展	6,844 KB
Qt5Multimedia.dll	2020/11/6 17:08	应用程序扩展	729 KB
Qt5Network.dll	2020/11/6 13:29	应用程序扩展	1,309 KB
Qt5Qml.dll	2020/11/6 16:41	应用程序扩展	3,508 KB
Qt5SerialPort.dll	2020/11/6 16:26	应用程序扩展	74 KB
Qt5Svg.dll	2020/11/6 16:27	应用程序扩展	323 KB
Qt5Widgets.dll	2020/11/6 13:30	应用程序扩展	5,370 KB

- 3. 打开 VM 虚拟机，创建虚拟机并启动，启动之后进行相关配置



4.在 windterm 上登录 ubuntu 账号



4. 输入 “sudo bt 14” 打开浏览器链接

```
ubuntu@a:~$ sudo bt 14
=====
正在执行 (14)...
=====
sh: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
=====
BT-Panel default info!
=====
外网ipv4面板地址: https://183.137.32.194:41048/8536b6d7
内网面板地址: https://192.168.163.131:41048/8536b6d7
username: 5vtzgdd9
password: *****
Warning:
If you cannot access the panel,
release the following port (8888|888|80|443|20|21) in the security group
注意: 初始密码仅在首次登录面板前能正确获取, 其它时间请通过 bt 5 命令修改密码
=====
ubuntu@a:~$
```

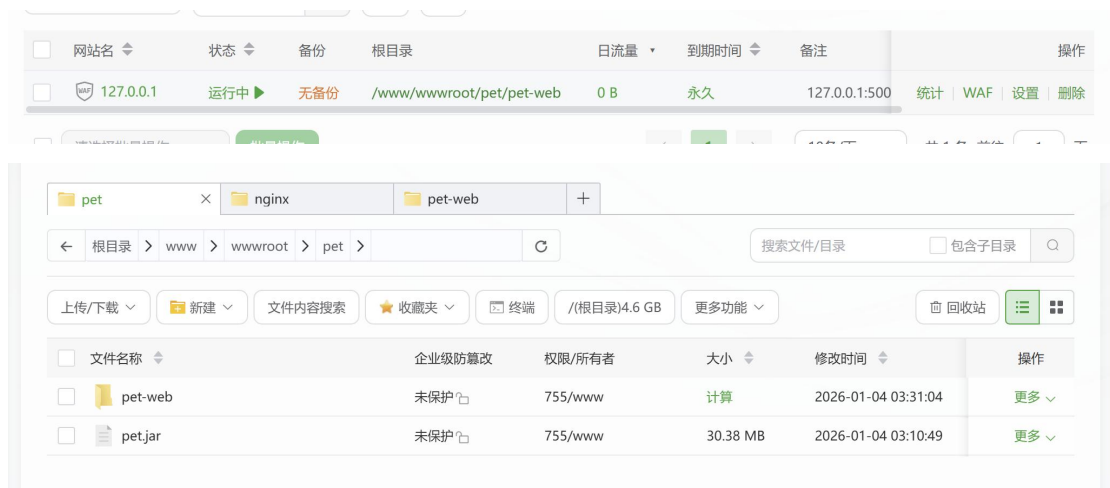
5.进入宝塔，根据上一步给出的用户名进行登录



5. 导入我们的数据库

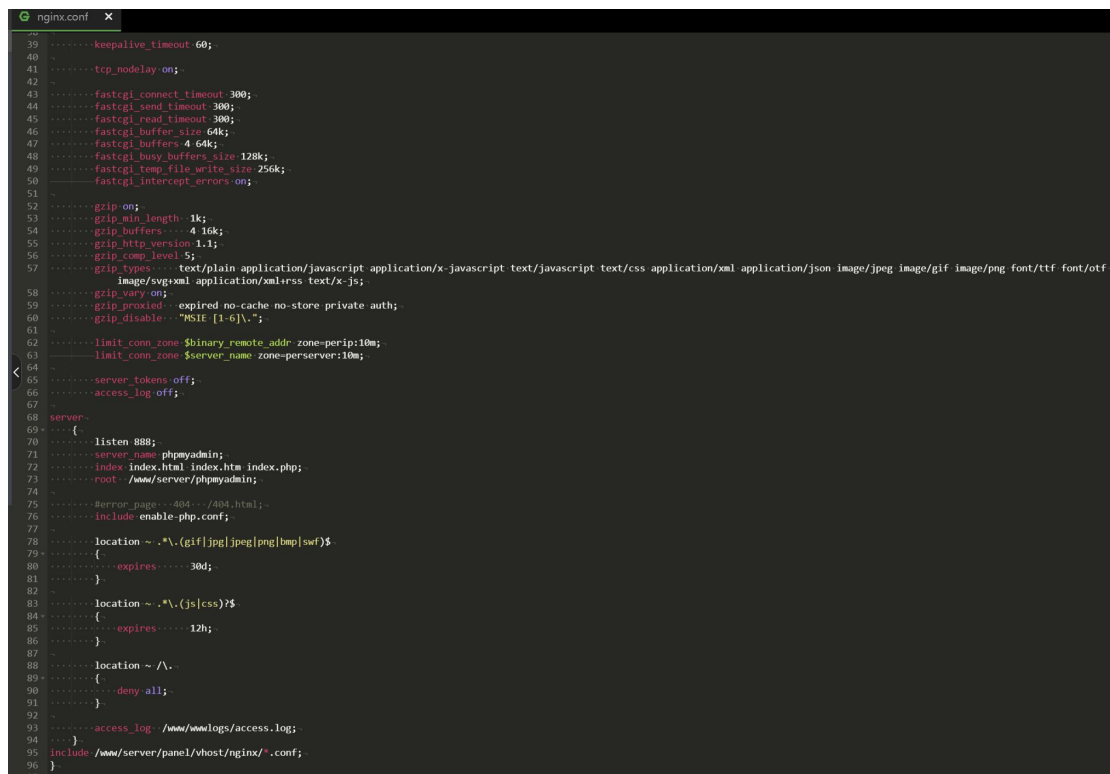


6 部署我们的项目：



6. 配置反向代理文件：

全局配置文件：



网站配置文件：


```
127.0.0.1.conf x
1 server {
2 {
3     listen 5000;
4     server_name 127.0.0.1;
5     index index.php index.html index.htm default.php default.htm default.html;
6     root /www/wwwroot/pet/pet-web;
7     #CERT-APPLY-CHECK--START-
8     #.用于SSL证书申请时的文件验证相关配置---请勿删除-
9     include /www/server/panel/vhost/nginx/well-known/127.0.0.1.conf;
10    #CERT-APPLY-CHECK--END-
11    include /www/server/panel/vhost/nginx/extension/127.0.0.1/*.conf;
12    ~
13    #SSL-START-SSL相关配置，请勿删除或修改下一行带注释的404规则-
14    #error_page 404 /404.html;
15    #SSL-END-
16    ~
17    #ERROR-PAGE-START-错误页配置，可以注释、删除或修改-
18    #error_page 404 /404.html;
19    #error_page 502 /502.html;
20    #ERROR-PAGE-END-
21    ~
22    #PHP-INFO-START--PHP引用配置，可以注释或修改-
23    #清理缓存规则-
24    ~
25    location ~ /purge/(.*) {
26        proxy_cache_purge cache_one $host$1$is_args$args;
27        #access_log /www/wwwlogs/127.0.0.1_purge_cache.log;
28    }
29    #引用反向代理规则，注释后配置的反向代理将无效-
30    include /www/server/panel/vhost/nginx/proxy/127.0.0.1/*.conf;
31    ~
32    ~
33    #解决刷新404问题-
34    location / {
35        try_files $uri $uri/ /index.html;
36    }
37    ~
38    include enable-php-00.conf;
39    #PHP-INFO-END-
40    ~
41    #REWRITE-START-URL重写规则引用，修改后将导致面板设置的伪静态规则失效-
42    include /www/server/panel/vhost/rewrite/127.0.0.1.conf;
43    #REWRITE-END-
44    ~
45    #禁止访问的文件或目录-
46    location ~ ^/(\.user.ini|\.htaccess|\.git|\.env|\.svn|\.project|LICENSE|README.md)
47    {
48        return 404;
49    }
50    ~
51    #一键申请ssl证书验证目录相关设置-
52    location ~ \.well-known{
53        allow all;
54    }
55    ~
56    #禁止在证书验证目录放入敏感文件-
57    if ( $uri ~ "^/\.well-known/.*\.(php|jsp|py|js|css|lua|ts|go|zip|tar\.gz|rar|7z|sql|bak)$" ) {
58        return 403;
59    }
60    ~
61    ~
62    ~
63    ~
64    access_log /www/wwwlogs/127.0.0.1.log;
65    error_log /www/wwwlogs/127.0.0.1.error.log;
66 }
```

五. 遇到的问题

图片存储解决方案：

1. 使用对象存储服务（阿里云 OSS）：
2. 上传到对象存储
3. 返回访问 URL



四、组内每个人负责的技术点及开发工作

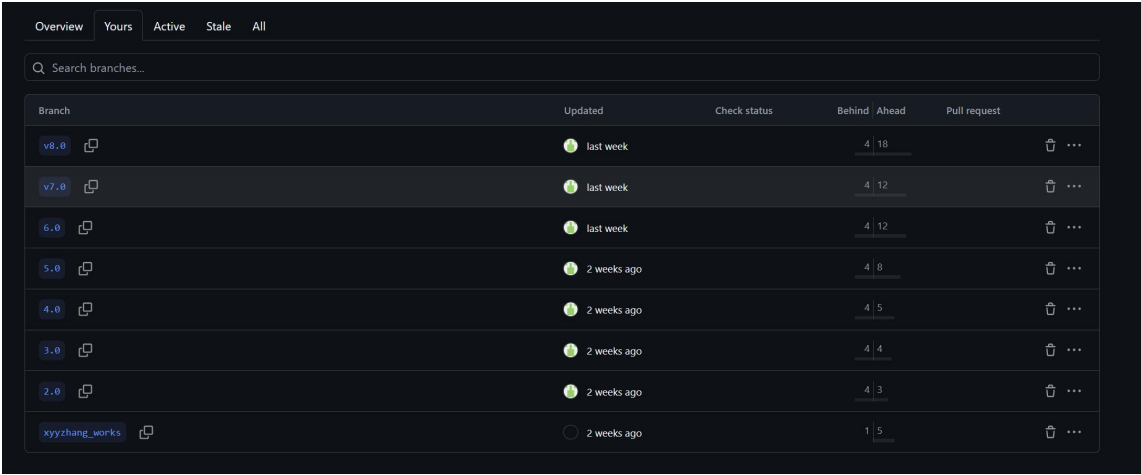
1.梁笛笛（用户认证、个人中心、商品、支付、管理端）

负责模块	关键技术	工作内容
用户认证、JWT 实现	JWT Token 认证	后端认证系统、Token 生成验证、前端登录流程
Nginx 配置、跨域处理	Nginx 反向代理、CORS 配置	部署配置、跨域解决方案、请求转发
商品模块、图片存储	对象存储(OSS)	商品查询
订单、支付模块	订单管理、支付接口、交易记录	订单流程、支付处理、回调处理
管理端用户管理	后台管理、权限控制、数据统计	管理界面、用户管理、数据展示
管理端商品管理	后台管理、权限控制、数据统计	管理界面、商品管理、数据展示
个人中心	购物信息统计	订单管理、地址管理、购物车管理，收管理、预约服务管理
AI 功能、对话系统	AI 问答系统	AI 助手实现

2.张昕玥玥（宠物管理、服务预约、护理项目、管理端）

负责模块	关键技术	工作内容
宠物管理	宠物信息管理、档案答应	宠物模块前后端、宠物档案
服务预约、预约管理	宠物预约系统	预约流程、时间管理、状态管理、预约查询
护理项目、项目套餐	项目管理、套餐组合、价格管理	项目维护、套餐管理、价格设置
管理端预约管理	后台管理、权限控制、数据统计	管理界面、宠物档案管理、数据展示
管理端宠物档案管理	后台管理、权限控制、数据统计	管理界面、宠物管理、预约管理、数据展示

Gitub 地址：<https://github.com/liang725/Pet Service Platform>



Branch	Updated	Check status	Behind	Ahead	Pull request
v8.0	last week		4	18	
v7.0	last week		4	12	
6.0	last week		4	12	
5.0	2 weeks ago		4	8	
4.0	2 weeks ago		4	5	
3.0	2 weeks ago		4	4	
2.0	2 weeks ago		4	3	
xyyzhang_works	2 weeks ago		1	5	

(具体 commit 操作见分支)