**《宠物托运智能服务系统》**

**详细设计说明书**

**课 程： 软件工程实训**

**队 伍： 深藏blue队**

**项目名称： GoGoPet**

**指导老师： 石秀金**

**成 员： 皮 超 221310417**

**赵鑫奕 221310109**

**张钰龙 221310131**

**李佳慧 221310203**

**丛梓璇 221310201**

**版本号V1.0**

**2025年4月2日**

目录

[1 引言 3](#_Toc194692457)

[1.1 编写目的 3](#_Toc194692458)

[1.2 项目概况 3](#_Toc194692459)

[1.3 术语定义 3](#_Toc194692460)

[1.4 参考资料 3](#_Toc194692461)

[2 系统概述 4](#_Toc194692462)

[3 程序设计详细描述 4](#_Toc194692463)

[3.1 具体功能 4](#_Toc194692464)

[3.1.1 注册功能 4](#_Toc194692465)

[3.1.2 登录功能 8](#_Toc194692466)

[3.1.3 个人信息管理功能 11](#_Toc194692467)

[3.1.4 宠物信息管理功能 14](#_Toc194692468)

[3.2 表结构说明 16](#_Toc194692469)

[3.2.1 注册功能所用到的表 16](#_Toc194692470)

[3.2.2 登录功能所用到的表 17](#_Toc194692471)

[3.2.3 个人信息管理功能所用到的表 17](#_Toc194692472)

[3.2.4 宠物信息管理功能所用到的表 18](#_Toc194692473)

[4 公用接口 19](#_Toc194692474)

[4.1 注册功能 19](#_Toc194692475)

[4.1.1全局变量 19](#_Toc194692476)

[4.1.2 公用界面 19](#_Toc194692477)

[4.1.3 公用函数和过程 19](#_Toc194692478)

[4.1.4 公用表辞典 19](#_Toc194692479)

[4.2 登录功能 20](#_Toc194692480)

[4.2.1全局变量 20](#_Toc194692481)

[4.2.2 公用界面 20](#_Toc194692482)

[4.2.3 公用函数和过程 20](#_Toc194692483)

[4.2.4 公用表辞典 21](#_Toc194692484)

[4.3 个人信息管理功能 21](#_Toc194692485)

[4.2.1全局变量 21](#_Toc194692486)

[4.2.2 公用界面 21](#_Toc194692487)

[4.2.3 公用函数和过程 22](#_Toc194692488)

[4.2.4 公用表辞典 22](#_Toc194692489)

[4.4 宠物信息管理功能 23](#_Toc194692490)

[4.1.1全局变量 23](#_Toc194692491)

[4.1.2 公用界面 23](#_Toc194692492)

[4.1.3 公用函数和过程 23](#_Toc194692493)

[4.1.4 公用表辞典 24](#_Toc194692494)

1 引言

* 1. 编写目的

本详细设计说明书用于指导用户管理模块的开发与实现，明确注册、登录、个人信息管理、宠物信息管理功能的设计细节，涵盖功能逻辑、接口定义、数据库设计及安全性要求，为开发、测试及验收提供明确依据，确保开发人员能够按照规范进行编码，并为测试人员提供参考依据。

* 1. 项目概况

用户管理模块是宠物托运智能服务系统的核心基础模块，旨在为系统用户提供完整的身份管理功能，包括注册、登录、个人信息维护及宠物信息管理。该模块通过安全验证和高效的数据交互，保障用户数据的隐私性与系统访问的合法性。

* 1. 术语定义

1. 表1 术语表

|  |  |
| --- | --- |
| 术语/缩写 | 定义 |
| JWT | JSON Web Token，用于用户身份验证的令牌机制 |
| AES-256 | 高级加密标准，用于用户密码存储加密 |
| RBAC | 基于角色的访问控制，限制用户操作权限 |
| t\_user | 用户表，存储用户ID、密码、身份等信息 |
| t\_pet | 宠物表，存储宠物ID、品种、健康状态等信息 |

1.4 参考资料

《需求规格说明书-V1.0》

《概要设计说明书》

2 系统概述

用户管理模块采用前后端分离架构实现：

**前端**：基于Vue3.js 实现用户交互界面。

**后端**：基于 Spring Boot + MyBatis 提供 RESTful API，通过 JWT 实现身份验证。

**数据库**：使用 MySQL 8.0 存储用户及宠物信息。

3 程序设计详细描述

3.1 具体功能

1. **注册功能**：新用户填写基本信息并完成验证，系统生成用户唯一标识。
2. **登录功能**：用户通过账号密码或验证码登录，系统签发 JWT 令牌。
3. **个人信息管理**：用户可修改密码、地址、头像等个人信息。
4. **宠物信息管理**：用户可添加、删除、修改宠物信息，并关联订单。

3.1.1 注册功能

| **模块编号** | M\_YHGLMK\_1 | **源程序文件** | Register.vue UserController.java |
| --- | --- | --- | --- |
| **功能** | 用户通过填写注册表单提交个人信息，系统验证信息有效性后存储至数据库。 | | |
| **输入参数** | 用户信息表（T\_User）页面上所有对应栏位值（参表结构） | | |
| **要访问的表** | 用户信息表（T\_User） | | |
| **模块时序图** | 图：时序图 | | |
| **输入界面** | 首页为浏览输入页，如下图：    图：浏览输入页面 | | |
| **输出界面** | 下图为货品浏览的输出页    图：浏览页面  **主要控件说明：**   * 在“Name”中输入用户名，名称必须至少包含 2 个字符； * 在“Password”中输入密码，密码必须至少为 8 个字符； * 在“Confirm Password”中再次输入密码，以确定密码； * 在“I am registering as a”中选择对应角色； * 勾选“I agree to the terms of service and privacy policy”； * 点击“Create account”，触发注册请求，后端校验并返回结果； * 点击“Sign in”，进入登录界面 | | |
| **出错处理** | 界面产生相应的提示  数据格式错误：实时提示格式错误（如密码不合规）；  数据重复：返回 400 状态码及具体错误信息（如“用户名已存在”）；  系统异常：返回 500 状态码，跳转至 error.html 页面。 | | |

3.1.2 登录功能

| **模块编号** | M\_YHGLMK\_2 | **源程序文件** | Login.vue UserController.java |
| --- | --- | --- | --- |
| **功能** | 用户通过输入用户名和密码进行身份验证，系统校验通过后返回访问令牌（JWT），并跳转至主页面；校验失败则提示错误信息。 | | |
| **输入参数** | |  |  |  |  | | --- | --- | --- | --- | | 参数名 | 描述 | 类型 | 验证规则 | | Username | 用户名 | String | 非空，4-20位字母数字组合 | | Password | 密码 | String | 非空，8-20位含大小写字母、数字、特殊符号 | | | |
| **要访问的表** | 用户信息表（T\_User） | | |
| **模块时序图** | 图：时序图 | | |
| **输入界面** | 首页为浏览输入页，如下图：    图：浏览输入页面 | | |
| **输出界面** | 成功登录系统  **主要控件说明：**   * 在“UserName”中输入用户名，名称必须至少包含 2 个字符； * 在“Password”中输入密码，密码必须至少为 8 个字符； * 在“Verification Code”中输入验证码； * 若忘记了密码，点击“Forgot password” 跳转至密码重置页面； * 点击“Login”，触发登录请求，后端校验并返回结果； * 点击“Register”，跳转至注册页面。 | | |
| **出错处理** | 界面产生相应的提示  数据格式错误：实时提示格式错误（如用户名为空）  用户不存在：返回 400 状态码及“用户名不存在”；  密码错误：返回 400 状态码及“密码错误”；  系统异常：返回 500 状态码，跳转至 error.html 页面。 | | |

3.1.3 个人信息管理功能

| **模块编号** | M\_YHGLMK\_3 | **源程序文件** | Profile.vue UserController.java |
| --- | --- | --- | --- |
| **功能** | 用户可查看及修改个人基本信息（如手机号、邮箱等），支持头像上传。 | | |
| **输入参数** | |  |  |  |  | | --- | --- | --- | --- | | 参数名 | 描述 | 类型 | 验证规则 | | Username | 用户名 | String | 非空，4-20位字母数字组合 | | Phone | 手机号 | String | 11位有效手机号 | | Email | 邮箱 | String | 符合RFC 5322标准 | | Avatar | 头像文件 | File | 格式为PNG/JPG | | | |
| **要访问的表** | 用户信息表（T\_User）、宠物信息表（T\_Pet） | | |
| **模块时序图** | 图：时序图 | | |
| **输入界面** | 首页为浏览输入页，如下图：    图：浏览输入页面 | | |
| **输出界面** | 信息修改成功  **主要控件说明：**   * 在“UserName”中输入要修改用户名，名称必须至少包含 2 个字符； * 在“Phone Number”中输入手机号码，要求是11为有效的手机号； * 在“Email Address”中输入邮箱号，符合RFC 5322标准； * 点击“Manager Pets”，管理宠物信息； * 点击“Edit”，跳转至宠物信息管理页面； * 点击“Change photo”，支持PNG/JPG格式，实时预览裁剪后效果； * 点击“Save Changes”，提交修改请求，后端校验并更新数据库； * 点击“Cancel”，重置表单至原始数据。 | | |
| **出错处理** | 界面产生相应的提示  文件过大：实时校验失败时阻止提交；  数据冲突：返回 400 状态码及具体错误（如“邮箱已被注册”）；  系统异常：返回 500 状态码，跳转至 error.html。 | | |

3.1.4 宠物信息管理功能

| **模块编号** | M\_YHGLMK\_4 | **源程序文件** | PetManagement.vue PetController.java |
| --- | --- | --- | --- |
| **功能** | 用户添加/删除/修改宠物信息（名称、品种、健康状态），系统校验宠物与订单的绑定关系。 | | |
| **输入参数** | |  |  |  |  | | --- | --- | --- | --- | | 参数名 | 描述 | 类型 | 验证规则 | | Petname | 宠物名称 | String | 非空 | | Breed | 品种 | String | 非空 | | Weight | 体重 | Double | 0-200 | | HealthStatus | 健康状态 | String | 非空，枚举值（健康/患病/康复中） | | | |
| **要访问的表** | 宠物信息表（T\_Pet） | | |
| **模块时序图** | 图：时序图 | | |
| **输入界面** | 如下图：    图：浏览输入页面 | | |
| **输出界面** | **主要控件说明：**   * 搜索框，按名称或品种模糊查询，实时过滤列表数据； * 分页组件，每页展示5条，支持跳转至指定页码； * 批量删除按钮，选中多条记录后批量删除（需管理员权限）。； * 导出Excel，将当前列表数据导出为Excel文件； | | |
| **出错处理** | 界面产生相应的提示  非法数据：输入非法数据时实时拦截并提示（如体重为负数）；  权限不足：返回 403 状态码，跳转至无权限提示页。  数据冲突：返回 400 状态码及具体错误（如“宠物名称重复”）。  系统异常：返回 500 状态码，跳转至 error.html。 | | |

3.2 表结构说明

3.2.1 注册功能所用到的表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 字段名 | 数据类型 | 允许空值 | 说明 |
| 用户ID | user\_id | VARCHAR(18) | N | 主键，唯一标识符 |
| 用户名 | user\_name | VARCHAR(63) | N | 注册信息 |
| 密码 | password | VARCHAR(63) | N | 注册信息 |
| 用户身份 | user\_type | CHAR(1) | N | 类型（用户/托运司机/管理员/……） |
| 用户地址 | user\_address | VARCHAR(63) | Y | 用户地址，限于用户、司机填写 |
| 用户头像 | picture\_id | VARCHAR(18) | Y | 外键，指明用户头像 |
| 公司ID | company\_id | VARCHAR(18) | Y | 外键，如果身份为司机则指明公司 |

3.2.2 登录功能所用到的表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 字段名 | 数据类型 | 允许空值 | 说明 |
| 用户ID | user\_id | VARCHAR(18) | N | 主键，唯一标识符 |
| 用户名 | user\_name | VARCHAR(63) | N | 注册信息 |
| 密码 | password | VARCHAR(63) | N | 注册信息 |
| 用户身份 | user\_type | CHAR(1) | N | 类型（用户/托运司机/管理员/……） |
| 用户地址 | user\_address | VARCHAR(63) | Y | 用户地址，限于用户、司机填写 |
| 用户头像 | picture\_id | VARCHAR(18) | Y | 外键，指明用户头像 |
| 公司ID | company\_id | VARCHAR(18) | Y | 外键，如果身份为司机则指明公司 |

3.2.3 个人信息管理功能所用到的表

用户表（t\_user）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 字段名 | 数据类型 | 允许空值 | 说明 |
| 用户ID | user\_id | VARCHAR(18) | N | 主键，唯一标识符 |
| 用户名 | user\_name | VARCHAR(63) | N | 注册信息 |
| 密码 | password | VARCHAR(63) | N | 注册信息 |
| 用户身份 | user\_type | CHAR(1) | N | 类型（用户/托运司机/管理员/……） |
| 用户地址 | user\_address | VARCHAR(63) | Y | 用户地址，限于用户、司机填写 |
| 用户头像 | picture\_id | VARCHAR(18) | Y | 外键，指明用户头像 |
| 公司ID | company\_id | VARCHAR(18) | Y | 外键，如果身份为司机则指明公司 |

宠物表（t\_pet）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 字段名 | 数据类型 | 允许空值 | 说明 |
| 宠物ID | pet\_id | VARCHAR(18) | N | 主键，唯一标识符 |
| 用户ID | user\_id | VARCHAR(18) | N | 外键，指明其主人 |
| 宠物名称 | pet\_name | VARCHAR(63) | N | 宠物信息 |
| 宠物品种 | pet\_breed | VARCHAR(63) | Y | 宠物信息 |
| 宠物体重 | pet\_weight | VARCHAR(18) | Y | 宠物信息 |
| 宠物健康状态 | pet\_health\_status | CHAR(1) | Y | 宠物信息 |

3.2.4 宠物信息管理功能所用到的表

宠物表（t\_pet）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 字段名 | 数据类型 | 允许空值 | 说明 |
| 宠物ID | pet\_id | VARCHAR(18) | N | 主键，唯一标识符 |
| 用户ID | user\_id | VARCHAR(18) | N | 外键，指明其主人 |
| 宠物名称 | pet\_name | VARCHAR(63) | N | 宠物信息 |
| 宠物品种 | pet\_breed | VARCHAR(63) | Y | 宠物信息 |
| 宠物体重 | pet\_weight | VARCHAR(18) | Y | 宠物信息 |
| 宠物健康状态 | pet\_health\_status | CHAR(1) | Y | 宠物信息 |

4 公用接口

4.1 注册功能

4.1.1全局变量

无

4.1.2 公用界面

无

4.1.3 公用函数和过程

**密码加密函数**：

public class SecurityUtils {  
 public static String encryptPassword(String password) {  
 // AES-256加密实现   
 return AES.encrypt(password, SECRET\_KEY);  
 }  
}

4.1.4 公用表辞典

**用户角色枚举（T\_User.role）**：

|  |  |
| --- | --- |
| **值** | **说明** |
| USER | 普通用户 |
| COMPANY | 托运公司账号 |
| ADMIN | 系统管理员 |

4.2 登录功能

4.2.1全局变量

**JWT密钥**：

@Value("${jwt.secret-key}")  
private String jwtSecretKey;

4.2.2 公用界面

**登录表单组件（Vue.js）**：

<template>  
 <el-form :model="loginForm" :rules="rules" @submit.prevent="submitForm">  
 <el-form-item label="用户名" prop="username">  
 <el-input v-model="loginForm.username" />  
 </el-form-item>  
 <el-form-item label="密码" prop="password">  
 <el-input type="password" v-model="loginForm.password" />  
 </el-form-item>  
 <el-checkbox v-model="rememberMe">记住我</el-checkbox>  
 <el-button type="primary" native-type="submit">登录</el-button>  
 </el-form>  
</template>

4.2.3 公用函数和过程

**密码验证函数**：

public class SecurityUtils {  
 public static boolean verifyPassword(String inputPassword, String storedHash) {  
 // 对比输入密码与数据库存储的AES加密哈希   
 return AES.decrypt(storedHash).equals(inputPassword);  
 }  
}

**JWT生成函数**：

public class SecurityUtils {  
@Component  
public class JwtTokenProvider {  
 public String generateToken(String username, UserRole role) {  
 return Jwts.builder()  
 .setSubject(username)  
 .claim("role", role)  
 .setExpiration(new Date(System.*currentTimeMillis*() + 3600000)) // 1小时过期   
 .signWith(SignatureAlgorithm.HS512, jwtSecretKey)  
 .compact();  
 }  
}

4.2.4 公用表辞典

**用户角色枚举（T\_User.role）**：

|  |  |
| --- | --- |
| **值** | **说明** |
| USER | 普通用户 |
| COMPANY | 托运公司账号 |
| ADMIN | 系统管理员 |

4.3 个人信息管理功能

4.2.1全局变量

无

4.2.2 公用界面

**头像上传组件（Vue.js）**：

<template>  
 <div class="avatar-uploader">  
 <el-upload  
 action="/api/upload/avatar"  
 :show-file-list="false"  
 :on-success="handleSuccess"  
 :before-upload="beforeUpload"  
 >  
 <img v-if="imageUrl" :src="imageUrl" class="avatar" />  
 <el-icon v-else class="avatar-uploader-icon"><Plus /></el-icon>  
 </el-upload>  
 </div>  
</template>

4.2.3 公用函数和过程

**文件上传服务**：

@Service  
public class OssService {  
 public String uploadAvatar(MultipartFile file) {  
 String fileName = UUID.randomUUID() + "\_" + file.getOriginalFilename();  
 OSSClient ossClient = new OSSClient(ossEndpoint, ossAccessKey, ossSecretKey);  
 ossClient.putObject("bucket-name", "avatars/" + fileName, file.getInputStream());  
 return "https://bucket-name.oss-cn-hangzhou.aliyuncs.com/avatars/" + fileName;  
 }  
}

4.2.4 公用表辞典

**用户角色权限表（T\_Role\_Permission）**：

|  |  |
| --- | --- |
| **值** | **说明** |
| USER | 修改个人信息、管理宠物 |
| ADMIN | 管理所有用户数据 |

4.4 宠物信息管理功能

4.1.1全局变量

**分页配置**：

@Value("${page.size.default:10}")  
private int defaultPageSize; // 默认每页显示条数

4.1.2 公用界面

**分页组件（Vue.js）**

<template>  
 <el-pagination  
 :current-page="currentPage"  
 :page-size="pageSize"  
 :total="total"  
 @current-change="handlePageChange"  
 layout="prev, pager, next"  
 />  
</template>

4.1.3 公用函数和过程

**数据导出服务**：

@Service  
public class ExcelExportService {  
 public void exportPets(List<Pet> pets, HttpServletResponse response) {  
 HSSFWorkbook workbook = new HSSFWorkbook();  
 HSSFSheet sheet = workbook.createSheet("宠物列表");  
 // 填充表头和数据行   
 response.setHeader("Content-Disposition", "attachment; filename=pets.xls");  
 workbook.write(response.getOutputStream());  
 }  
}

**权限校验拦截器**：

public class AuthInterceptor implements HandlerInterceptor {  
 @Override  
 public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) {  
 if (!UserUtils.hasPermission("PET\_MANAGE")) {  
 response.sendRedirect("/error/403");  
 return false;  
 }  
 return true;  
 }  
}

4.1.4 公用表辞典

**用户角色枚举（T\_User.role）**：

|  |  |
| --- | --- |
| **值** | **说明** |
| HEALTHY | 健康 |
| SICK | 患病 |
| RECOVERING | 康复中 |