**后端文档**

**后端技术栈详细选型**



**点击图片可查看完整电子表格**

核心语言用typescript

ide是vscode 用 prettier进行代码格式化 支持 formatonsave。格式化以eslint标准，eslint没有的标准可按默认的来（这些初始化也需要设置vscode的 workspace的settings.json， 不想使用 Prettier 的自定义配置所以不要有prettier.configPath这个配置）

**数据库设计 mysql 5.7+**

用户↔领养记录 1:N，宠物↔领养记录 1:N

**核心功能模块设计**

1. 用户认证系统

|  |
| --- |
| JavaScript *// 路由示例* router.post('/login',   passport.authenticate('local', { session: false }),  (req, res) => {  *// 生成JWT令牌*const token = jwt.sign({ id: req.user.id }, process.env.JWT\_SECRET, { expiresIn: '7d' });  res.json({ token });  } );  *// 中间件验证*const requireAuth = passport.authenticate('jwt', { session: false }); |

1. 宠物管理模块

* 端点设计：
* GET /pets 分页获取可领养宠物（支持species/size过滤）
* POST /pets 管理员添加新宠物（需JWT认证）
* PATCH /pets/:id/status 更新宠物状态

1. 领养申请系统

|  |
| --- |
| JavaScript *// 领养申请流程* router.post('/adoptions', requireAuth, [  check('pet\_id').isInt(),  check('application\_note').notEmpty() ], async (req, res) => {  *// 验证宠物是否可领养*const pet = await Pet.findByPk(req.body.pet\_id);  if (pet.status !== 'available') {  return res.status(400).json({ error: 'Pet not available' });  }  *// 创建申请记录...* }); |

**项目目录结构**

|  |
| --- |
| Nginx backend/ ├── config/ │ └── config.js *# 数据库环境配置* ├── migrations/ *# Sequelize迁移文件* ├── models/ │ ├── user.js *# 用户模型* │ ├── pet.js *# 宠物模型* │ └── adoption.js *# 领养记录模型* ├── routes/ │ ├── auth.js *# 认证路由* │ ├── pets.js *# 宠物路由* │ └── adoptions.js *# 领养路由* ├── middlewares/ │ ├── auth.js *# JWT验证中间件* │ └── validation.js *# 请求校验* ├── utils/ │ └── jwtStrategy.js *# Passport策略配置* └── app.js *# 主入口文件* |

**环境配置建议 .env**

|  |
| --- |
| TOML DB\_HOST=localhost DB\_USER=root DB\_PASS=yourpassword DB\_NAME=pet\_adoption JWT\_SECRET=your\_secure\_secret PORT=3000 |

**关键安全措施**

1. 密码存储：使用bcryptjs进行盐值加密（cost=12）
2. JWT设置：HTTP-only Cookie存储 + 7天有效期
3. SQL注入防护：Sequelize参数化查询
4. 速率限制：建议后期添加express-rate-limit

**开发路线建议**

**总周期**：12天（2023-08-01 至 2023-08-12）

**第一阶段：基础搭建（3天）**

* **Day 1 2023−08−01**  
  ✅ 项目初始化
* 创建Express项目骨架
* 安装核心依赖（Express, Sequelize, Dotenv）
* 配置基础路由测试端点
* **Day 2-3 2023−08−02 2023−08−03**  
  🔧 数据库建模（进行中）
* 使用Sequelize定义三张核心表模型（User/Pet/Adoption）
* 编写迁移脚本建立表关系（用户↔领养记录 1:N，宠物↔领养记录 1:N）
* 创建种子数据脚本（mock 10只宠物+5个用户）

**第二阶段：核心功能（6天）**

* **Day 4-6 2023−08−04 2023−08−06**  
  🛡️ 用户认证系统
* 实现注册/登录接口（密码bcrypt加密）
* 集成Passport JWT策略
* 开发鉴权中间件（区分user/admin角色）
* **Day 7-9 2023−08−07 2023−08−09**  
  🐕 宠物管理模块
* 宠物CRUD接口（管理员权限校验）
* 实现分页查询+多条件过滤（品种/体型）
* 宠物状态机设计（available→adopted）

**第三阶段：领养流程（4天）**

* **Day 10-11 2023−08−10 2023−08−11**  
  📝 申请逻辑实现
* 领养申请创建接口（校验宠物可领用状态）
* 用户历史申请记录查询
* **Day 12 2023−08−12**  
  🔄 状态管理
* 申请状态变更流程（pending→approved/rejected）
* 状态变更时联动更新宠物表状态

|  |
| --- |
| **下一步行动建议**：   1. 使用sequelize-cli init初始化项目结构 2. 执行npx sequelize model:generate创建基础模型 3. 优先实现用户注册/登录流程（包含JWT签发） 4. 部署MySQL后运行迁移文件：npx sequelize db:migrate |

此方案已预留扩展接口，后续可轻松添加：

* 宠物收藏功能
* 领养申请进度通知
* 多图片上传支持
* 用户个人信息管理

需要任何具体模块的代码实现细节或调整设计，请随时告诉我！