**<轻量博客引擎>**

**需求规格说明书**

**团队：菜鸟驿站 小组**

日期：2025年6月18号

目录

[**1.引言** 4](#_Toc201102034)

[**1.1. 编写**目的 4](#_Toc201102035)

[**1.2. 项目范围** 4](#_Toc201102036)

[**1.3. 术语定义** 5](#_Toc201102037)

[**1.4. 引用文件** 5](#_Toc201102038)

[**2. 总体描述** 5](#_Toc201102039)

[**2.1. 产品背景** 5](#_Toc201102040)

[**2.2. 产品描述** 6](#_Toc201102041)

[**2.2.1 系统接口** 6](#_Toc201102042)

[**2.2.2 用户界面** 7](#_Toc201102043)

[**2.2.3 硬件接口** 8](#_Toc201102044)

[**2.2.4 软件接口** 8](#_Toc201102045)

[**2.2.5 通信接口** 8](#_Toc201102046)

[**2.2.6 内存约束** 8](#_Toc201102047)

[**2.3. 产品功能** 8](#_Toc201102048)

[**2.5. 应用目标与作用范围** 9](#_Toc201102049)

[**2.6. 约束** 9](#_Toc201102050)

[**2.7. 假设与依赖关系** 9](#_Toc201102051)

[**3. 具体需求** 10](#_Toc201102052)

[**3.1外部接口** 10](#_Toc201102053)

[**3.2. 功能需求** 10](#_Toc201102054)

[**3.2.1 用例描述（Use Case）** 10](#_Toc201102055)

[**3.2.2 顺序图** 11](#_Toc201102056)

[**3.2.3 分析类图（逻辑建模）** 11](#_Toc201102057)

[**3.2.4 功能模块划分** 12](#_Toc201102058)

[**3.3. 性能需求** 12](#_Toc201102059)

[**3.3.1. 精度需求** 12](#_Toc201102060)

[**3.4 设计约束** 13](#_Toc201102061)

[**3.4.1 标准依从性** 13](#_Toc201102062)

[**3.5软件系统属性** 13](#_Toc201102063)

[**3.5.1 观感需求** 13](#_Toc201102064)

[**3.5.2 易用性需求** 13](#_Toc201102065)

[**3.5.3可执行性需求** 14](#_Toc201102066)

[**3.5.4安全性** 14](#_Toc201102067)

[**3.5.5 系统完整性** 14](#_Toc201102068)

[**3.5.6 可扩展性与可维护性** 14](#_Toc201102069)

**变更履历**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **版本** | **修订章节与内容** | **变更原因** | **修订人** | **审核人** | **修订日期** |
| **1** | **1.0** | **全文** | **建立初稿** | **杜嘉康** | **杜嘉康** | **2025-05-12** |
| **2** | **1.1** | **性能需求添加** | **内容修订** | **覃昶** | **杜嘉康** | **2025-06-18** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**说明：“变更原因”主要是分为：**

1. **建立初稿**
2. **内容修订**
3. **正式发布**

**1.引言**

**1.1. 编写**目的

本说明文档旨在系统性地阐述博客系统的项目背景与开发意义，明确系统的功能目标与技术实现路径，为后续的系统设计、数据库建模、功能开发及测试部署提供统一的规范与指导依据。文档内容包括：

* 系统需求分析与功能模块划分
* 技术选型与系统架构说明
* 数据库逻辑模型与物理模型设计
* 统实现目标与接口说明
* 系统测试范围与上线验收标准

通过本需求分析文档的编写，可为前后端开发人员提供准确的接口契约与设计指南，为数据库设计人员提供清晰的数据结构定义，同时为测试与部署工作提供基础参考。

**1.2. 项目范围**

博客（Blog，全称 Weblog）最初被设计用于用户在线发布日志与文章，经过多年发展已逐步成为创作者记录知识、表达观点、互动交流的重要媒介。

本博客系统为一个轻量级的内容管理平台，支持：

* 用户注册与登录
* 博文发布、分类、标签管理
* 评论与用户中心功能

系统采用当前流行的 Rust 语言进行开发，后端使用 Axum 框架 构建 Web 服务，SeaORM 作为 ORM 工具操作数据库，默认使用 SQLite，也可灵活切换为 MySQL。系统接口设计遵循 RESTful API 规范，前端可独立开发，也可通过 Postman等工具测试接口功能。

系统设计风格注重模块化、轻量级、高性能，适合个人博主或小型团队部署使用。

**1.3. 术语定义**

|  |  |
| --- | --- |
| **术语** | **定义** |
| 博主 | 拥有账号并具有管理内容权限的用户 |
| 文章 | 博主撰写的博客内容 |
| 标签 | 用于对文章进行分类和筛选的关键词 |
| JWT | JSON Web Token，用于用户身份认证 |

**1.4. 引用文件**

[1] Steve Klabnik, Carol Nichols. *The Rust Programming Language*（Rust官方权威书籍）  
[2] Mitchell, David. *Programming Web Services with Rust using Axum*（Axum 实战参考）  
[3] SeaQL. *SeaORM 官方文档*：https://www.sea-ql.org/SeaORM/  
[4] SQLite. *SQLite Documentation*：https://www.sqlite.org/docs.html

**2. 总体描述**

**2.1. 产品背景**

随着 Web 技术的发展与内容创作生态的成熟，传统的博客平台（如 CSDN、博客园、简书等）逐渐趋于中心化，限制了用户对内容的自主掌控权。越来越多的内容创作者、程序员及技术博主，开始寻求一种 “去平台化、自托管、高自由度” 的内容发布解决方案。

本项目即在此背景下提出，旨在构建一个以技术用户为核心、支持自定义部署、结构简洁的轻量级博客系统，帮助用户在私有服务器或本地环境中轻松搭建自己的博客。

该博客系统采用 Rust 语言构建，具备出色的性能与安全特性，同时结合 Axum 框架实现现代化 Web 架构，简化接口开发，提升系统扩展能力。

**2.2. 产品描述**

博客系统作为一种内容管理系统（CMS），旨在帮助用户记录、展示与传播信息，是开发者学习记录、内容创作者发布观点的重要工具。为更好满足现代用户对性能、安全、自主可控的需求，本系统在传统博客功能的基础上，采用 Rust 编程语言进行开发，构建出一个轻量级、高性能、可扩展的现代博客平台。

本系统基于 Rust + Axum 框架搭建后端，SeaORM 提供 ORM 支持，前端未强依赖特定框架，便于与 Vue、React 或移动端集成。采用 JWT 实现用户身份认证，接口遵循 RESTful 规范，支持 Postman、Swagger 等自动化测试工具。

系统包括前端展示、后台管理、评论互动三大核心部分，适配桌面和移动设备，支持浏览器直接访问和第三方 API 消费。

**2.2.1 系统接口**

**首页相关接口**

* GET /blogs: 分页获取所有博客（支持按时间排序）
* GET /categories?limit=n: 获取前 n 个分类
* GET /tags?limit=n: 获取前 n 个标签
* GET /blogs/latest?limit=n: 获取最新发布的 n 篇博客

**用户认证接口**

* POST /user/register: 用户注册
* POST /user/login: 用户登录，返回 JWT
* GET /user/profile: 获取当前用户信息
* POST /user/logout: 用户注销（Token 失效）

**博客管理接口**

* GET /admin/blogs: 后台分页查询博客
* GET /admin/blogs/search?title=...: 按标题查询
* GET /admin/blogs/category/:id: 根据分类查询
* POST /admin/blogs: 新增博客
* PUT /admin/blogs/:id: 修改博客
* DELETE /admin/blogs/:id: 删除博客

**标签接口**

* POST /admin/tags: 新增标签
* GET /admin/tags: 查询所有标签
* PUT /admin/tags/:id: 更新标签
* DELETE /admin/tags/:id: 删除标签
* GET /admin/tags/search?name=...: 查询标签

**分类接口**

* POST /admin/categories: 新增分类
* GET /admin/categories: 查询所有分类
* PUT /admin/categories/:id: 修改分类
* DELETE /admin/categories/:id: 删除分类
* GET /admin/categories/search?name=...: 查询分类

**评论接口**

* GET /comments?blog\_id=...: 查询博客下所有评论
* POST /comments: 发表评论
* GET /comments/search?title=...: 根据标题查评论

**归档页接口**

* GET /archive: 根据时间归档返回博客列表

**2.2.2 用户界面**

**管理员用户：**

* 登录后台系统
* 发布/修改/删除博客
* 管理博客的分类与标签
* 评论审核与管理
* 基于标题、分类、标签快速查询博客

**普通访客：**

* 浏览首页博客列表（分页）
* 查看某一分类或标签下的所有博客
* 查看年度归档、热门博客
* 全文搜索支持关键词模糊匹配
* 查看博客详情与评论列表
* 对博客发表评论

**2.2.3 硬件接口**

 适配 x86\_64、ARM 架构

 支持 PC 浏览器与移动端访问

 屏幕分辨率兼容范围：最小 360×640，最大 4096×2160

**2.2.4 软件接口**

 后端语言：Rust 1.70+

 Web 框架：Axum + Tokio + Tower

 数据库：SQLite（开发）、MySQL（部署）

 ORM 工具：SeaORM

 环境依赖：支持 Linux、Windows、macOS

**2.2.5 通信接口**

 支持 HTTP/1.1 和 HTTPS（推荐）

 接口基于 JSON 格式传输

 后端对外提供 RESTful API 接口

 认证方式采用 Bearer Token（JWT）

**2.2.6 内存约束**

 最小运行内存建议 ≥ 512MB

 磁盘空间需求 ≥ 100MB（SQLite）

**2.3. 产品功能**

现阶段功能：

 支持用户注册、登录、Token 认证机制

 博客文章管理：CRUD、Markdown 编辑、标签分类

 评论系统：访客可评论，管理员可审阅

 分类/标签系统：支持嵌套结构与统计

 首页、归档页、标签页、分类页的展示

 关键词全文搜索

 RESTful API 接口规范，方便前后端分离

**2.5. 应用目标与作用范围**

本系统面向对内容创作有持续需求的技术用户，包括：

* 青年开发者：记录成长路径与学习笔记
* 高校学生：展示项目成果、研究内容
* 自媒体创作者：搭建独立博客品牌
* 小型组织：发布技术公告与知识沉淀

他们重视系统性能、安全性以及对数据的自主可控，倾向使用命令行或 Git 工作流，乐于自建服务，适合 Rust 提供的精致控制与部署方式。

**2.6. 约束**

法律约束：系统部署与运营应遵守《中华人民共和国网络安全法》

安全性要求：接口需校验用户身份；数据库不得暴露公网

硬件要求：服务器推荐配置为 1核2G 内存、1G以上硬盘空间

编码语言：仅使用 Rust 实现后端逻辑，避免混合语言引入复杂性

可维护性：系统结构清晰、接口模块独立，易于后期扩展

**2.7. 假设与依赖关系**

 假设部署环境已具备 Rust 编译工具链及数据库运行条件

 假设前端开发人员熟悉与 RESTful API 对接方式

 系统开发依赖 tokio 异步运行时、axum 路由框架、SeaORM ORM

 假设使用者对基础命令行、JWT 身份认证机制有所了解

 项目开发团队熟悉 Git、VS Code 或 JetBrains RustRover 等工具链

**3. 具体需求**

**3.1外部接口**

本博客系统采用典型前后端分离架构，所有外部交互均通过 RESTful API 实现。接口请求与响应均使用 JSON 格式，通过 HTTP/HTTPS 传输。系统支持如下外部对接方式：

 Web 前端通过 HTTP 请求调用后端 API

 可集成 Swagger/Postman 接口文档测试平台

 支持接入第三方系统进行内容同步（预留 Webhook 接口）

无图形化 GUI 输入输出接口要求，CLI 或 GUI 客户端均可基于 API 实现交互。

**3.2. 功能需求**

**3.2.1 用例描述（Use Case）**

功能总览：

|  |  |  |
| --- | --- | --- |
| 用例名称 | 参与者 | 概要说明 |
| 用户注册/登录 | 访客 | 提供凭证注册账号，登录后返回 JWT |
| 创建/编辑博客 | 管理员 | 提交博客内容、分类、标签，存入数据库 |
| 浏览博客列表 | 访客/用户 | 分页获取博客概览信息 |
| 博客详情查看 | 访客/用户 | 根据 ID 查看博客全文及评论 |
| 博客评论 | 注册用户 | 登录后评论目标博客 |
| 管理标签/分类 | 管理员 | 维护分类/标签字典 |
| 搜索与归档浏览 | 所有用户 | 支持关键字搜索、时间归档过滤 |

**3.2.2 顺序图**

可包含以下关键业务顺序图（UML 可手动绘制）：

 管理员登录顺序图

 博客内容发布顺序图

 用户访问与评论顺序图

 博客分类、标签维护流程图

 归档与搜索流程顺序图

**3.2.3 分析类图（逻辑建模）**

主要实体类（即数据模型）：

 User 用户类

 Blog 博客类

 Comment 评论类

 Tag 标签类

 Category 分类类

 BlogTag 博客与标签多对多关联类

这些类可用于生成 SeaORM 实体模型，统一数据库设计与逻辑表示。

**3.2.4 功能模块划分**

|  |  |
| --- | --- |
| 模块 | **功能描述** |
| 首页 | 显示博客分页列表、分类/标签排行、推荐文章、最新文章 |
| 博客详情页 | 展示单篇博客详细内容，支持评论展示与发布 |
| 分类页 | 浏览全部分类及其下的博客文章 |
| 标签页 | 浏览全部标签及其下的博客文章 |
| 归档页 | 按时间归档展示博客，按年份统计 |
| 关于我页 | 展示博主简介与联系信息，可使用标签形式列出技能爱好 |
| 管理员登录页 | 登录后台系统（基于 Token 身份认证） |
| 博客管理模块 | 发布/修改/删除博客，支持搜索与过滤 |
| 标签管理模块 | 标签新增、修改、删除与查询 |
| 分类管理模块 | 分类新增、修改、删除与查询 |
| 评论管理模块 | 按时间或文章标题查看与管理评论（后端预留） |

**3.3. 性能需求**

**3.3.1. 精度需求**

|  |  |  |
| --- | --- | --- |
| 类别 | 字段 | 精度 |
| 用户 | 手机号 | 11位数字 |
| 用户名 | 6-18 位数字或字母 |
| 密码 | 6-18 位数字或字母 |
| 文章 | 书名 | 不超过 64 个字符 |
| 作者 | 不超过 64 个字符 |
| 简介 | 不超过 10000 个字符 |
| 连接 | 不超过 128 个字符 |
| 记录 | 借出时间 | 16 位数字或字母 |
| 归还时间 | 16 位数字或字母 |
| 借阅者 | 不超过 64 个字符 |
| 借阅状态 | 布尔型 |

**3.4 设计约束**

**3.4.1 标准依从性**

 参照《计算机软件开发规范 CB8566-88》《软件文件编制指南 CBB567-88》

 REST API 符合 OpenAPI 3.0 标准

 数据模型生成遵循 SeaORM 实体建模规范

 JWT 实现符合 RFC 7519 标准

 编码规范参照 Rust 官方及《Rust API Design Guidelines》

**3.5软件系统属性**

**3.5.1 观感需求**

整体页面简洁清新，模块布局合理，博客排版规范。推荐使用 TailwindCSS + Markdown 渲染器实现前端样式，支持代码高亮、图片嵌入等。

**3.5.2 易用性需求**

 操作简洁：后台功能一目了然

 Markdown 编辑器支持实时预览

 游客可直接浏览、评论需登录后操作

**3.5.3可执行性需求**

 首页加载时间不超过 200ms

 博客详情页加载不超过 500ms

 数据传输 JSON 格式，采用 GZIP 压缩优化带宽

**3.5.4安全性**

 所有数据接口均要求身份认证（JWT）

 管理端需配置 IP 白名单

 密码加密存储，禁止明文传输

**3.5.5 系统完整性**

 支持文章发布、归档、分类、标签、评论、权限控制等基本功能

 所有核心操作具备接口与后端落地实现

 数据结构支持后续导出、迁移、备份

**3.5.6 可扩展性与可维护性**

 模块划分清晰，接口注释完整，符合微服务拆分原则

 日志记录使用统一中间件（如 tracing）

 可通过 feature flag 或配置文件启用额外功能（如“点赞”）