科学技术与教育 2026年第6期
ISSN: 3079-4455

基于 Django 的轻量化 Bug 追踪管理系统的设计与实现

李宇轩，陈超滨，李俊华，肖伟平，叶锦菲
东莞城市学院，广东东莞，中国

【摘要】针对小型开发团队 Bug 管理混乱、流程不规范等问题，本文设计并实现一套轻量化 Bug 追踪管理系统。系统基于 Django 的 MTV 架构，前端采用 Bootstrap 与 jQuery 构建响应式界面，后端以 MySQL 为数据库，并集成腾讯云 COS 实现文件云存储。系统实现用户认证、项目管理、Bug 全生命周期管理、文件上传下载及数据可视化等功能。测试表明，系统运行稳定、流程完整、权限清晰，可有效提升小型团队的缺陷管理效率。

【关键词】Bug 追踪；Django 框架；轻量化；项目管理；状态流转

1.绪论

随着软件开发行业发展，小型开发团队与创新项目日益增多，缺陷（Bug）管理直接影响开发效率与产品质量。目前主流缺陷管理工具功能复杂、部署繁琐，难以适配小型团队轻量化、低成本的使用需求，多数团队存在 Bug 记录混乱、状态流转不规范、附件管理不便等问题。为此，本文设计并实现一套轻量化 Bug 追踪管理系统，以解决小型团队缺陷管理痛点，提升协作效率。

1.1 研究背景与意义

1.1.1 研究背景

软件开发过程中，缺陷管理不可或缺。现有工具（如 Jira、禅道）过于复杂，轻量工具又功能不足，无法满足小型团队与学生项目的实际需求。因此，亟需一款操作简单、轻量化、易部署的缺陷追踪系统。

1.1.2 研究意义

实践意义：系统贴合小型团队使用场景，实现 Bug 追踪、项目管理、文件云存储一体化，操作简便、部署成本低，可有效规范缺陷流程、提升协作效率。

理论意义：通过项目实践掌握 Django、MySQL、云存储等 Web 技术的综合应用，为同类轻量化管理系统的设计与开发提供参考。

2.相关技术介绍

本章简要介绍系统开发所使用的技术栈与工具，为系统实现提供技术支撑。

2.1 后端框架

系统采用 Django 4.2 作为后端开发框架，基于 MTV 架构模式，借助 ORM 实现数据库操作，内置用户认证、表单校验、路由管理等功能，开发高效、结构清晰。

2.2 前端技术

前端使用 HTML、CSS、JavaScript 结合 Bootstrap 5 构建响应式界面，通过 jQuery 实现表单校验与异步请求，保证页面简洁易用、交互流畅。

2.3 数据库

采用 MySQL 8.0 关系型数据库，用于存储用户、项目、Bug 工单、操作日志等核心业务数据，稳定高效、易于维护。

2.4 云存储服务

集成腾讯云 COS 对象存储，用于存放附件、截图等文件，减轻服务器压力，提升文件管理的安全性与可靠性。

3 系统设计

本章完成系统整体设计，包括总体架构、数据库结构与核心业务模块设计，是系统实现的重要依据。

3.1 总体架构设计

系统采用 Django 的 MTV 三层架构：

Model 层：负责数据模型与数据库交互；

Template 层：负责页面展示；

View 层：负责请求处理与业务逻辑。

整体分为表现层、业务逻辑层、数据访问层与基础设施层，结构清晰、易于维护。如图 1 所示。

3.2 数据库设计

数据库设计是系统设计的核心环节，直接影响系统的数据一致性、查询效率与扩展性。本系统主要实体包括：用户、项目、Bug 工单、文件。各实体联系如下：

(1)用户与项目：一个用户可参与多个项目，一个项目包含多个用户，为**多对多**关系。

(2)用户与 Bug 工单：一个用户可提交/处理多个 Bug，一个 Bug 对应唯一提交人与处理人，为**一对多**关系。

(3)项目与 Bug 工单：一个项目包含多个 Bug 工单，一个 Bug 归属于唯一项目，为**一对多**关系。

(4)Bug 工单与文件：一个 Bug 可包含多个附件，一个文件归属于唯一 Bug，为**一对多**关系。

系统 E-R 图如图 2 所示。

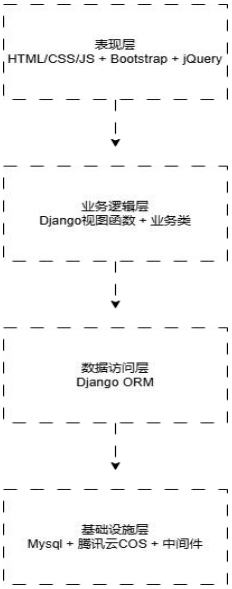

图 1.系统层次结构图

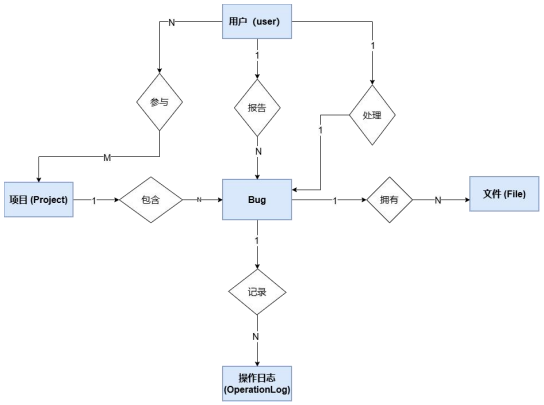

图 2.系统 ER 图

3.2.1 数据库逻辑结构设计

逻辑设计将 E-R 图转化为数据表，定义字段、类型与关联关系。系统基于 MySQL 8.0，共设计 6 张核心数据表：

（1）用户表基于 Django 内置 auth_user 扩展，字段包括：id（主键）、username（登录用户名，唯一非空）、password（加密密码，非空）、email（电子邮箱）、role（角色，默认为 developer）、avatar_url（头像地址）、is_active（是否启用，默认为 True）、last_login（最后登录时间）、date_joined（注册时间，非空）。

（2）项目表字段包括：id（主键）、name（项目名称，非空）、description（项目描述）、created_by_id（创建者 ID，外键关联用户表）、create_at（创建时间，非空）、update_at（更新时间）、is_active（项目是否启用，默认为 True）。

（3）项目成员表记录用户与项目的多对多关联，字段包括：id（主键）、project_id（项目 ID，外键）、user_id（用户 ID，外键）、role_in_project(项目内角色，默认为 member)、joined_at（加入时间，非空）。通过(project_id, user_id)联合唯一约束防止重复添加成员。

（4）Bug 工单表是系统核心表，字段包括：id（主键）、title（Bug 标题，非空）、description（详细描述，非空）、status（状态：pending/processing/resolved/closed，默认为 pending）、priority（优先级：low/medium/high/critical，默认为 medium）、project_id（所属项目 ID，外键）、reporter_id（报告人 ID，外键）、assignee_id（责任人 ID，外键）、created_at(创建时间，非空)、updated_at（更新时间）、resolved_at(解决时间)、version（发现版本）、steps_to_reproduce(重现步骤)。

（5）文件表记录上传至腾讯云 COS 的附件信息，字段包括：id（主键）、bug_id（所属 Bug 工单 ID，外键）、file_name（原始文件名，非空）、file_key（COS 存储键值，唯一非空）、file_size（文件大小，单位字节，非空）、file_type(MIME 类型)、uploaded_by_id（上传者 ID，外键）、uploaded_at(上传时间，非空）、download_url（访问 URL）。

（6）操作日志表记录用户关键操作行为，字段包括：id（主键）、bug_id（关联 Bug 工单 ID，外键）、operator_id（操作人 ID，外键）、operation_type（操作类型：create/assign/status_change/edit/delete，非空）、old_value（变更前值）、new_value（变更后值）、created_at(操作时间，非空)、ip_address（操作 IP）。

3.2.2 数据库物理结构设计

物理设计用于确定数据表存储规则，主要包括索引、存储引擎与字符集三部分。

（1） 索引设计

对常用查询字段建立索引：bug 表的 status、priority、assignee_id、project_id、reporter_id 建立普通索引；auth_user 的 username、file 的 file_key、project_member 的(project_id, user_id)建立唯一索引；operation_log 的 user_id 和 created_at 建立普通索引，以提升查询效率。

（2）存储引擎

所有表采用 InnoDB 引擎，支持事务、外键与行级锁，保证数据一致性与完整性。

（3）字符集

采用 utf8mb4 字符集，支持完整中文与特殊字符存储，显示稳定可靠。

3.3 核心模块详细设计

本节对 Bug 管理与文件上传两大核心模块进行设计，明确业务流程、状态规则与功能实现。

3.3.1Bug 管理模块设计

（1）功能概述

Bug 管理为系统核心模块，实现 Bug 从创建到关闭的全生命周期管理，支持新增、编辑、指派、状态流转等操作。

（2）Bug 状态机设计

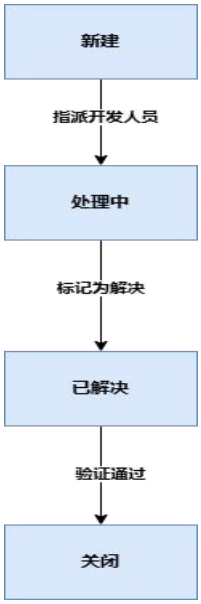

图 3.bug 处理流程图

状态流转规则（如图 3 所示）的约束条件如下：

仅新建状态的 Bug 可被指派给开发人员，变更为处理中状态

仅处理中状态的 Bug 可被标记为已解决

仅已解决状态的 Bug 可被测试人员验证后变更为关闭

已关闭的 Bug 不可再修改状态（若问题复发，应创建新 Bug）

状态变更时，系统自动记录操作日志并更新 updated_at 时间戳

（3）业务流程设计

①Bug 创建流程

测试人员进入创建页→填写信息→选择项目→指派开发→提交校验→系统自动填充信息→写入数据库→跳转详情页。

②状态流转流程（处理中→已解决）

开发人员查看指派 Bug→标记已解决→系统校验权限与状态→更新状态→记录日志→页面刷新。

（4）类设计

模块核心类与功能负责数据模型、表单校验、页面展示与状态更新。

3.3.2 文件上传模块设计

（1）功能概述

支持 Bug 附件上传，文件直传腾讯云 COS，降低服务器压力并提升上传速度。

（2）上传流程设计

采用前端直传 COS 模式，流程如下：

用户选择文件→前端请求凭证→后端生成临时密钥→前端直传云存储→上传成功返回 URL→元数据写入数据库。

（3）类设计

模块核心类负责文件元数据管理、COS 接口封装与上传处理。

（4）安全设计

系统通过格式限制、大小限制、临时凭证、文件重命名与防盗链保证上传安全。

4.系统实现与测试

本章展示系统功能实现效果，并通过测试验证系统稳定性。系统基于 Django MTV 架构开发，支持测试人员、管理员、开发人员三类角色协同完成缺陷管理。

4.1 系统实现概况

系统采用前后端结合方式实现，系统整体架构如图 4 所示。

图 4.系统整体架构图

4.2 核心功能实现

系统实现了项目管理、Bug 提交、状态流转、文件上传等核心功能。项目

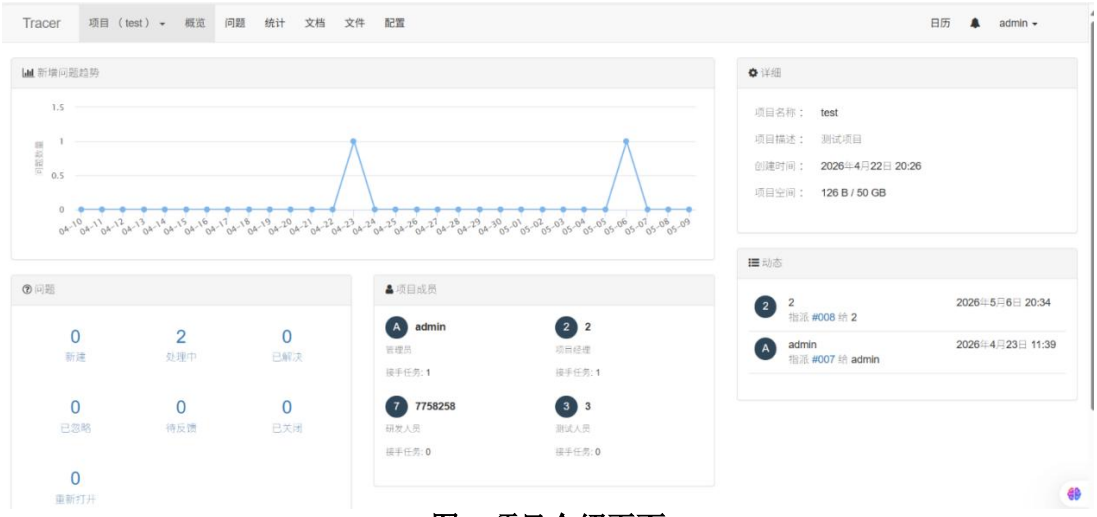

图 5.项目介绍页面

测试人员提交 Bug 后，管理员可分配给开发人员进行修复，完成后由测试人员验证关闭，流程闭环规范，如图 6 表示。

图 6.bug 提交后界面

文件统一存储至腾讯云 COS，同时支持数据统计、日历查看等辅助功能，界面简洁、操作便捷。

5.总结

本文设计并实现了一套基于 Django 的轻量化 Bug 追踪管理系统，完成需求分析、架构设计、功能开发与系统测试。系统实现了用户认证、项目管理、缺陷提交、状态流转、文件云存储等核心功能，流程规范、运行稳定，可满足小型开发团队轻量化、高效率的缺陷管理需求，达到了预期设计目标。

参考文献

[1] 黄锐军.Django Web 项目开发实战[M]. 北京：电子工业出版社，2021.
[2] 王珊，萨师煊.数据库系统概论（第 6 版）[M]. 北京：高等教育出版社，2022.
[3] 吴涛.MySQL 8.0 数据库从入门到精通[M]. 北京：清华大学出版社，2020.
[4] 赵伟.基于 Django 的缺陷管理系统设计与实现[J]. 信息技术，2023 (08):71-76.
[5] 陈丽.基于 Web 的 Bug 跟踪系统设计与关键技术[J]. 计算机系统应用，2022,31 (05):112-118.
[6] 刘畅.腾讯云对象存储在 Web 系统中的应用研究[J]. 数字技术与应用，2023,41 (02):68-70.