Gitea需求规格说明书

[v1.2]

|  |  |  |
| --- | --- | --- |
|  | 组员 | 日期 |
| 撰写 | 麦梓健 | 2020.3.21 |
| 校对 |  |  |

北京航空航天大学计算机学院

二〇二〇年三月二十一日

文档修改记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本号 | 日期 | 修改章节 | 修改说明 | 修改人 | 审核人 |
| 1.00 | 2020.3.21 |  | 初稿完成 | 麦梓健 | 麦梓健 |
| 1.10 | 2020.3.24 | 第三章、第五章 | 增加内容 | 孙维华 | 麦梓健 |
| 1.2 | 2020.4.1 | 全章节 | 格式修改、内容增加 | 麦梓健 | 麦梓健 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目录

[1 引言 5](#_Toc35976264)

[1.1 编写目的 5](#_Toc35976265)

[1.2 背景 5](#_Toc35976266)

[1.3 定义 5](#_Toc35976267)

[1.4 参考技术标准 7](#_Toc35976268)

[2 项目内容 8](#_Toc35976269)

[2.1 目标 8](#_Toc35976270)

[2.2 用户特点 8](#_Toc35976271)

[3 需求分析 9](#_Toc35976272)

[3.1 需求 9](#_Toc35976273)

[3.1.1 业务需求 9](#_Toc35976274)

[3.1.2 功能需求 9](#_Toc35976275)

[3.1.3 非功能需求 10](#_Toc35976276)

[3.2 框架及组件描述 11](#_Toc35976277)

[3.2.1 Macaron框架 12](#_Toc35976278)

[3.2.2 XORM库 12](#_Toc35976279)

[3.2.3 UI组件库 13](#_Toc35976280)

[3.3 用例建模 13](#_Toc35976281)

[3.3.1 用户角色 13](#_Toc35976282)

[3.3.2 用例图 14](#_Toc35976283)

[3.4 RUCM建模 15](#_Toc35976284)

[3.5 其他事务模块描述 22](#_Toc35976285)

[3.5.1 日志模块 22](#_Toc35976286)

[3.5.2 Toolbox工具箱 22](#_Toc35976287)

[3.5.3 缓存 22](#_Toc35976288)

[3.5.4 模块切换器 23](#_Toc35976289)

[3.5.5 bindata 23](#_Toc35976290)

[3.6 异常处理 23](#_Toc35976291)

[4 运行环境说明 24](#_Toc35976292)

[4.1 部署环境 24](#_Toc35976293)

[4.2 接口 24](#_Toc35976294)

[5 功能拓展与改进 25](#_Toc35976295)

[5.1 需求分析 25](#_Toc35976296)

[5.1.1 业务需求 25](#_Toc35976297)

[5.1.2 功能需求 26](#_Toc35976298)

[5.1.3 非功能需求 26](#_Toc35976299)

[5.2 用例建模 27](#_Toc35976300)

[5.2.1 用户角色 27](#_Toc35976301)

[5.2.2 用例图 27](#_Toc35976302)

[5.3 RUCM建模 28](#_Toc35976303)

# 引言

Gitea是轻量级的代码托管平台。本文主要分析Gitea的核心功能模块、非功能模块和业务逻辑，同时介绍改进需求和拓展模块。

## 编写目的

本软件需求规格说明书，是为软件设计、软件测试人员和用户编写的。

本软件需求规格说明书的适用者，包括开发测试人员、Gitea技术人员、项目其他相关人员以及Gitea的使用群体。

## 背景

软件名称：Gitea

项目组织机构：Gitea项目开发组和社区组织

项目实施机构：Gitea社区上的开发组

Git的出现：高速、简单的设计、对非线性开发模式的强力支持（允许成千上万个并行开发的分支）、完全分布式、有能力高效管理类似 Linux 内核一样的超大规模项目（速度和数据量）。

GitHub、GitLab等基于Git的代码托管平台，尽管功能强大，但是过重、过于复杂了，对于许多项目而言并不需要如此庞大的体系，因而轻量级的框架gogs等相继出现。

Gogs简单、轻便、运行快，只需要廉价的树莓派即可满足需求。但是gogs由于“独裁”性质的维护，使得由社区维护的gitea出现了。

## 定义

表格 1 术语定义

|  |  |
| --- | --- |
| Git钩子 | Git钩子是在Git仓库中特定事件发生时自动运行的脚本。可以定制一些钩子，这些钩子可以在特定的情况下被执行，分为Client端的钩子和Server端的钩子。 |
| Wiki | Wiki是一种在网络上开放且可供多人协同创作的超文本系统，由沃德·坎宁安于1995年首先开发，这种超文本系统支持面向社群的协作式写作，同时也包括一组支持这种写作 |
| 镜像仓库 | 用来保存制作好的容器的一个仓库，只不过这里的容器被做成了镜像，所以我们称其为镜像仓库。 |
| SSH | SSH 为建立在应用层基础上的安全协议。SSH 是较可靠，专为远程登录会话和其他网络服务提供安全性的协议 |
| Session | 会话是一种持久网络协议，在用户端和服务器端之间建立关联，从而起到交换数据包的作用机制。 |
| pull | 将数据从远程分支拉取到本地，取回远程主机某个分支的更新，再与本地的指定分支合并。其完整命令：git pull <远程主机名> <远程分支名>:<本地分支名>。 |
| push | 将数据从本地仓库上传到远程仓库。git push的一般形式为 git push <远程主机名> <本地分支名> <远程分支名> ，例如 git push origin master：refs/for/master ，即是将本地的master分支推送到远程主机origin上的对应master分支， origin 是远程主机名。第一个master是本地分支名，第二个master是远程分支名。 |
| add | git add [参数] <路径>　作用就是将我们需要提交的代码从工作区添加到暂存区，就是告诉git系统，我们要提交哪些文件，之后就可以使用git commit命令进行提交了。 |
| merge | 将数据拉取到本地仓库，并与相应分支进行融合，不清除旧版本。开发分支（dev）上的代码达到上线的标准后，要合并到 master 分支或者当master代码改动了，需要更新开发分支（dev）上的代码 |
| cache | Git的数据缓存，用于撤销提交等 |
| migration | 数据迁移。 |
| mock | mock测试就是在测试过程中，对于某些不容易构造或者不容易获取的对象，用一个虚拟的对象来创建以便测试的测试方法。这个虚拟的对象就是mock对象。mock对象就是真实对象在调试期间的代替品。 |

## 参考技术标准

参照如下标准：

* GB/T 19003-2008 软件工程
* GB/T 5538-1995 软件工程标准分类法
* GB/T 9385-2008 计算机软件需求规格说明
* GB/T 5532-2008 计算机软件测试规范
* GB/T 8567-2006 计算机软件文档编制规范

# 项目内容

## 目标

Gitea的出现是为了方便个人或者小型团队开发时进行方便快捷的代码管理和托管。一方面，Gitea同GitHub和GitLab相比减去了大量极少使用的代码模块，使得部署和运行要求大大下降，速度大为上升。另一方面，Gitea作为社区维护的开源项目，不仅可以依据需求选择版本，还可以根据个人和团队的要求进行方便的扩展开发。

## 用户特点

1. 有代码管理要求的个人和小型团队，且同时对服务器部署有速度等要求。
2. 对代码管理有独特要求、需要对已有框架进行扩展的开发团队。

# 需求分析

## 需求

### 业务需求

Git 是由 Linux Torvalds 开发的一个版本控制系统，现如今正在被全世界大量开发者使用。由于Git创新性的采用了分布式版本控制的设计，让Git成为了现今最热门的版本控制系统之一。Git 自身支持远程仓库特性，并提供了服务端解决方案，但是Git的远程仓库的实现对用户并不友好，对于多仓库、权限管理的支持十分简陋，也不支持像仓库工单（Issue）、项目 Wiki 等对于代码管理十分重要的功能。因此需要一个提供易用且完善功能的 Git 服务程序。Gitea 就是这样一个自托管的Git服务程序。他和GitHub, Bitbucket 还有 Gitlab等服务比较类似。Gitea的首要目标是创建一个极易安装、运行快速、安装和使用体验良好的自托管 Git 服务。

### 功能需求

为了实现一个功能完善，用户使用体验良好的自托管的Git服务，我们认为Gitea应有如下的功能需求。

1. **基本Git服务**

这是作为 Git服务最基础的功能。除了像 Git 内置的服务端一样支持本地仓库的推送和拉取外。Gitea还具有的特性有：对仓库主题描述、仓库内代码搜索、全局代码搜索、细粒度用户角色 (例如 Code, Issues, Wiki)、仓库活跃度页面、分支管理、在线代码编辑、项目的统计图表

1. **仓库工单（Issue）功能**

工单功能对于一个多人协作的项目来说，特别是开源项目，是尤其重要的，工单功能可以记录项目中存在的问题，并对问题的进度进行追踪。Gitea对于工单的具有如下支持：

1. 工单模板：为了方便复现和修复Bug，Gitea提供工单模板来规范工单发起人的工单内容与格式。
2. 评论反馈：对于一个工单，相关人员可以就问题原因和解决方案参与讨论。
3. 工单标签：使用标签可以对工单进行灵活分类，方便管理。比如工单可以按照进度贴上“待处理”、“进行中”、“已完成”和“已验收”标签，或者根据工单内容划分为“软件Bug”、“功能需求”等。
4. 其他工单功能：工单搜索、 指派工单负责人。
5. **合并请求（Pull/Merge requests）功能**

通过合并其他分支上的代码，可以很方便地整合其他人的工作并进行代码评审。Gitea对于合并请求具有如下支持：

1. 评论合并请求中的某行代码
2. 指定合并请求的审核人
3. 合并请求模板
4. **第三方集成**

作为代码托管平台，Gitea有很多需要与外部系统进行集成的需求，Gitea对与第三方集成提供了如下支持：

1. 自定义 Git 钩子：Gitea 能在特定的重要动作发生时触发自定义脚本，利用这一特性可以方便地实现自动部署等功能。
2. 外部登陆：Gitea支持 OpenId 和 OAuth 登陆协议，可以很容易地集成到现有的用户体系中。
3. 集成 Mattermost/Slack。Gitea对流行的Mattermost和Slack在线聊天服务进行了集成，用户可以在Mattermost/Slack中接收Git消息。

### 非功能需求

除了满足用户业务需求外，Gitea应满足以下非功能需求。

1. **高可靠性**

Gitea是一个面向组织或公众的多用户系统，需要保证系统服务的稳定性，需要能够承受住一定的并发请求。Gitea使用Macaron框架提供Web服务，Macaron是出色的高性能Web框架，能够满足高并发请求的场景。

1. **低资源开销**

Gitea最低的系统硬件要求为一个廉价的树莓派，对于团队使用，最低配置也仅需要单核1GB内存。

1. **多种数据库支持**

Gitea支持 MySQL、PostgreSQL、SQLite3, MSSQL 和 TiDB（实验性支持） 数据库。

1. **跨平台支持**

Gitea支持 Linux, macOS 和 Windows 以及各种架构，除了x86，amd64，还包括 ARM 和 PowerPC。

1. **对客户端提供广泛支持**

Gitea大部分功能通过浏览器来向用户呈现，Gitea支持主流浏览器以及显示器分辨率。

1. **部署简易性**

Gitea采用Go作为后端语言，这使Gitea只要生成一个可执行程序即可。

## 框架及组件描述

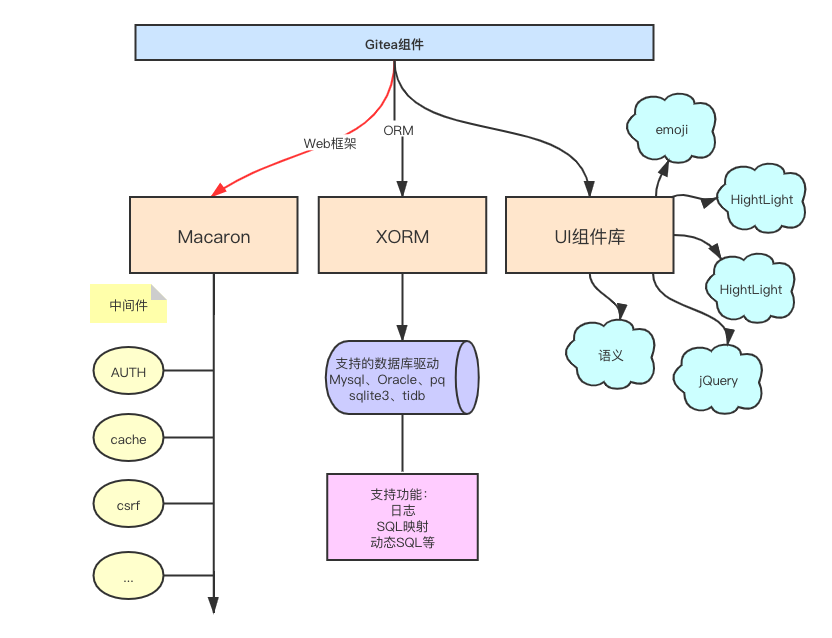


图 1 Gitea框架和组件图

### Macaron框架

Macaron框架是Go专用的一个快速开发工具，其具有强大功能：

1. 支持子路由的强大路由设计
2. 支持灵活多变的路由组合
3. 支持无限路由组的无限嵌套
4. 支持直接集成现有的服务
5. 支持运行时动态设置需要渲染的模板集
6. 支持对模块的轻松接入与解除
7. 采用 inject 提供的便利的依赖注入
8. 采用更好的路由层和更少的反射来提升执行速度

同时，Macaron本身已经集成了大量的中间件，包括

1. auth - HTTP 基本身份验证
2. bindata –静态文件和模版文件
3. binding – 数据绑定和验证
4. cache – 高速缓存服务
5. captcha – 验证码服务
6. csrf – 生成和验证令牌
7. i18n – 国际化和本地化
8. inject – 依赖注入
9. permissions2 - Cookies, users 和permissions
10. pongo2 - Pongo2 引擎
11. renders – 动态渲染
12. session – 会话管理
13. sockets – 套接字管理

除此之外，还有压缩、Oauth等等的中间件。

### XORM库

xorm是一个简单而强大的Go语言ORM库. 通过它可以使数据库操作非常简便。它同样具有强大的功能：

1. 支持Struct和数据库表之间的灵活映射，并支持自动同步
2. 事务支持，支持嵌套事务（支持类JAVA Spring的事务传播机制）
3. 同时支持原始SQL语句和ORM操作的混合执行
4. 支持级联加载Struct
5. 支持类ibatis方式配置SQL语句
6. 支持动态SQL功能
7. 支持数据库查询结果直接返回Json字符串和xml字符串
8. 支持SqlMap配置文件和SqlTemplate模板密文存储和解析
9. 支持缓存
10. 支持主从数据库(Master/Slave)数据库读写分离
11. 支持记录版本（即乐观锁）
12. 支持查询结果集导出csv、tsv、xml、json、xlsx、yaml、html功能

使用XORM库可以轻松的描述数据库数据存储结构和对象之间的关系，同时简单的实现各种所需的数据操作。

### UI组件库

UI组件库是由社区贡献者们提供的各种各样的特色功能插件，包括emoji文字、Github的图标、渲染文字、jQuery的时间器等等，能够满足特定的使用者群体的功能。

## 用例建模

### 用户角色

Gitea的整个生命周期中经历了多个阶段，可大致分为部署、管理、使用这三个阶段，因此用户角色可分为三类：

1. 部署者

部署者在受支持的操作系统上安装并配置Gitea的运行环境。他决定了Gitea能够提供的所有服务类型以及提供服务的方式。

1. 管理者

管理者作为特权用户，可以对Gitea运行时产生的数据进行操作。

1. 使用者

使用者是Gitea主要服务的用户群体。他们能够使用Gitea提供的大部分功能，并和其他用户进行交互。

### 用例图

图2展现了部署者的用例图模型。



图 2 部署者用例图

图3展现了管理者和使用者的用例图模型。



图 3 用户用例图

## RUCM建模

RUCM模型对用例进行规范描述。下面就以RUCM的标准对用例图中涉及的用例进行描述。



图 4 RUCM-设置数据库连接

如图4，在设置数据库连接时，需要测试连接成功后才能设置服务属性，否则返回连接失败的提示。

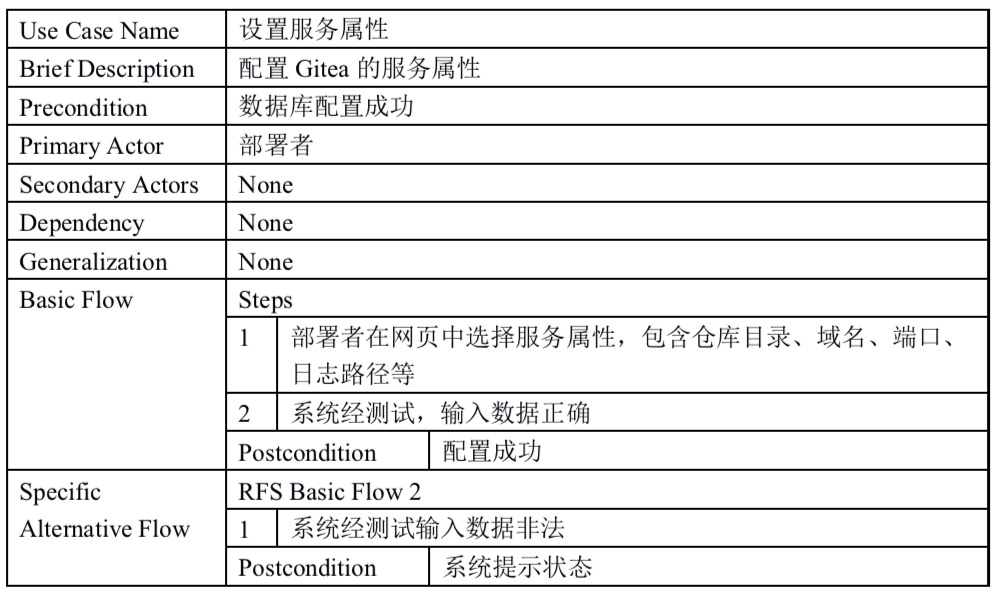


图 5 RUCM-设置服务属性

如图 5，在部署者设置服务属性之前，需要完成数据库的配置，并且确保配置有效，然后就可以在网页中设置服务属性。

在设置服务属性的时候，需要检查各个属性的输入是否合法，比如目录名称是否含有非法字符、目录的结构是否有效、域名是否有效、端口的位数与范围是否有效、日志路径是否存在且有效等等。输入与检验测试完成后，服务器返回配置成功的消息，否则提示输入数据非法。

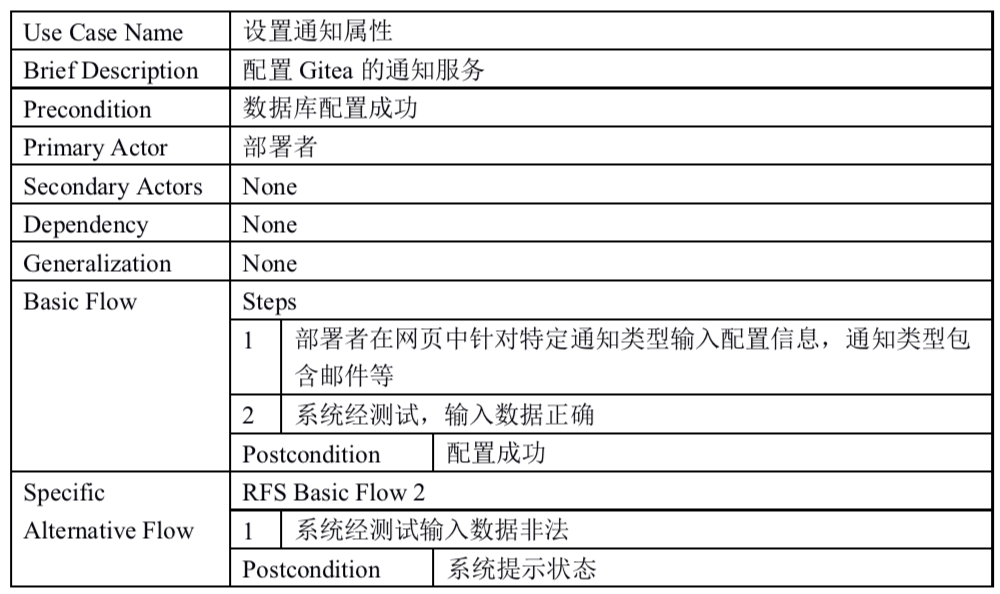


图 6 RUCM设置通知属性

如图6，部署者在设置通知属性的时候，同样要检验测试配置是否有效，比如对于邮件通知类型，邮件名是否有效、邮件内容采用的编码格式是否支持等等。

使用者和管理者作为Gitea的主要用户，功能比较繁杂，现就比较主要的功能进行RUCM建模。用户功能主要分为管理员和普通用户两个部分。

管理员功能主要包括：设置管理员账户，创建并管理git合并请求。普通用户功能包括：创建和删除账号，创建仓库和迁移仓库，以及创建文件。

如图7，设置管理员账号的时候，在保证数据库配置成功后，部署者输入管理员账号密码，然后系统测试名称是否含有非法字符、密码是否有效和符合强度要求。成功则创建管理员账户，失败则返回原因。

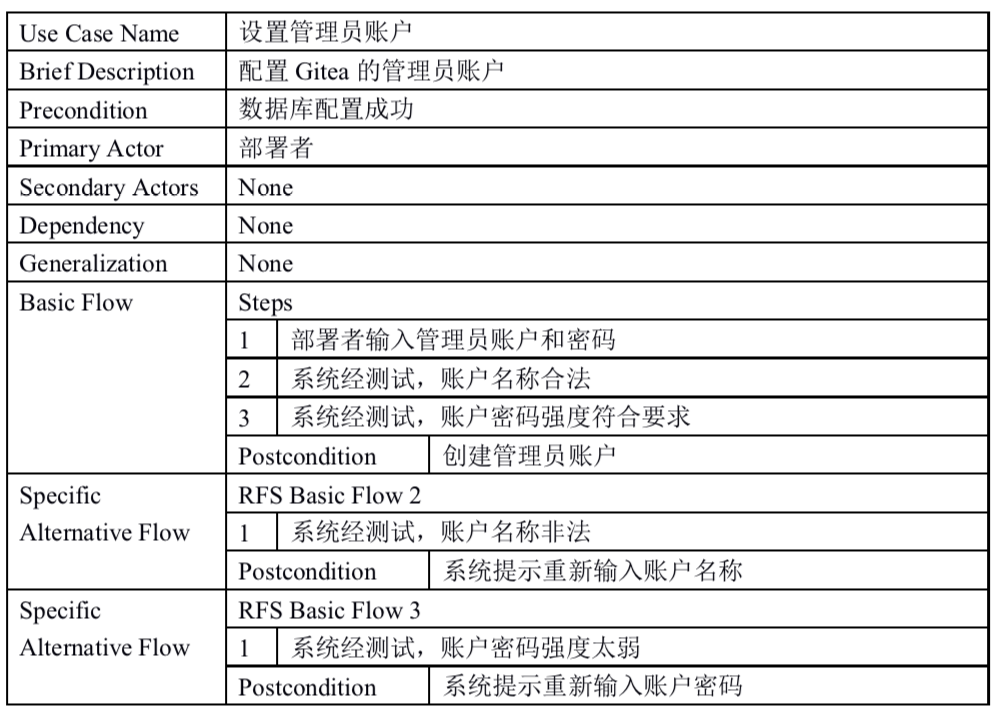


图 7 设置管理员账号



图 8 RUCM-创建合并请求

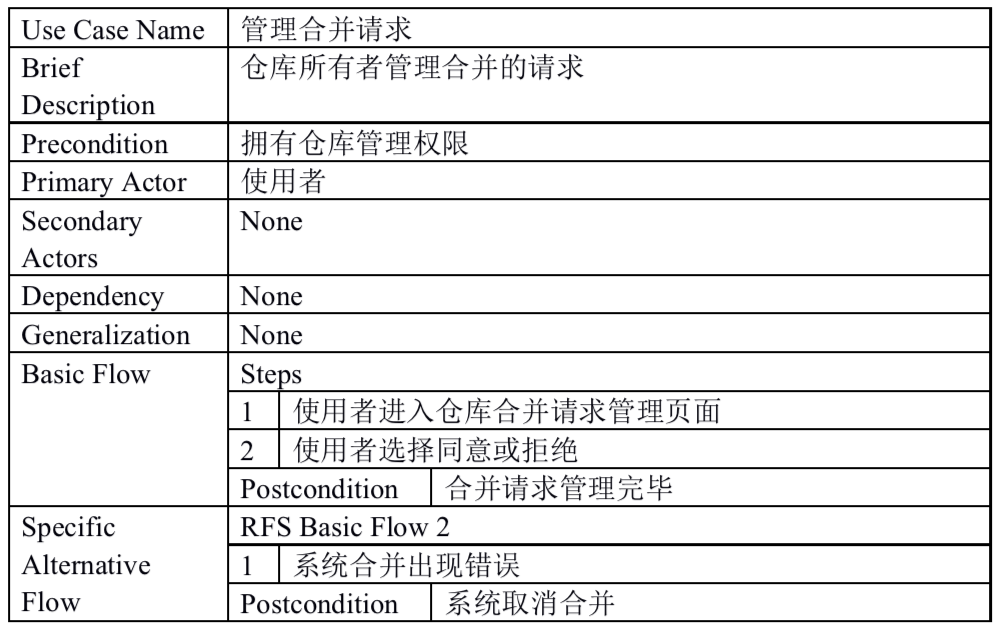


图 9 RUCM-管理合并请求

如图8，仓库的使用者在登陆自己的账户并迁移仓库成功后，可以提出合并的请求，系统在基于Hash算法分析两个分支的内容存储差异后，如果发现存在差异，则允许合并并创建合并请求，否则取消合并。创建合并请求后，向仓库的持有者发送通知，提醒有新的合并请求，并附带两个分支的分析报告。

如图9，在仓库管理员收到新的合并请求后，进入管理页面进行合并选择，如果同意且仓库合并成功后，管理流程结束，否则如果拒绝或者合并出错，则系统取消该次合并。

如图10，使用者可以向gitea系统申请创建用户，根据提示填写必要的账户、密码、邮箱等信息，然后系统校验完成，邮箱有效且帐号密码没有非法字符后，用户账户创建成功，最后会向用户的绑定邮箱发送用户账号成功的欢迎信息。如果信息填写不完整或者出现账号密码无效、邮箱无效等情况，则系统会给出信息缺失等提示信息。



图 10 RUCM-创建账户

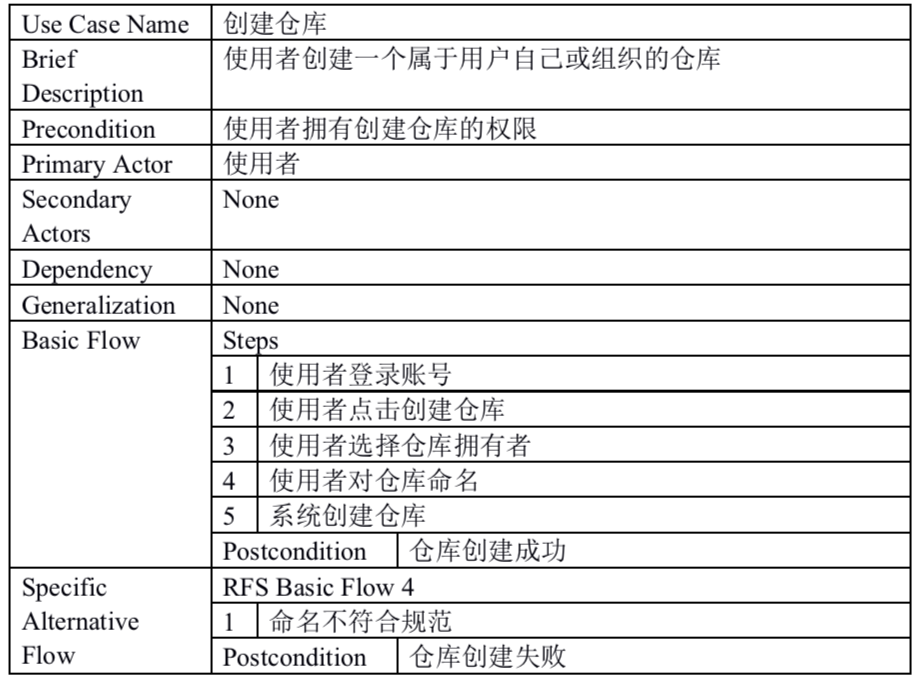


图 11 RUCM-创建仓库

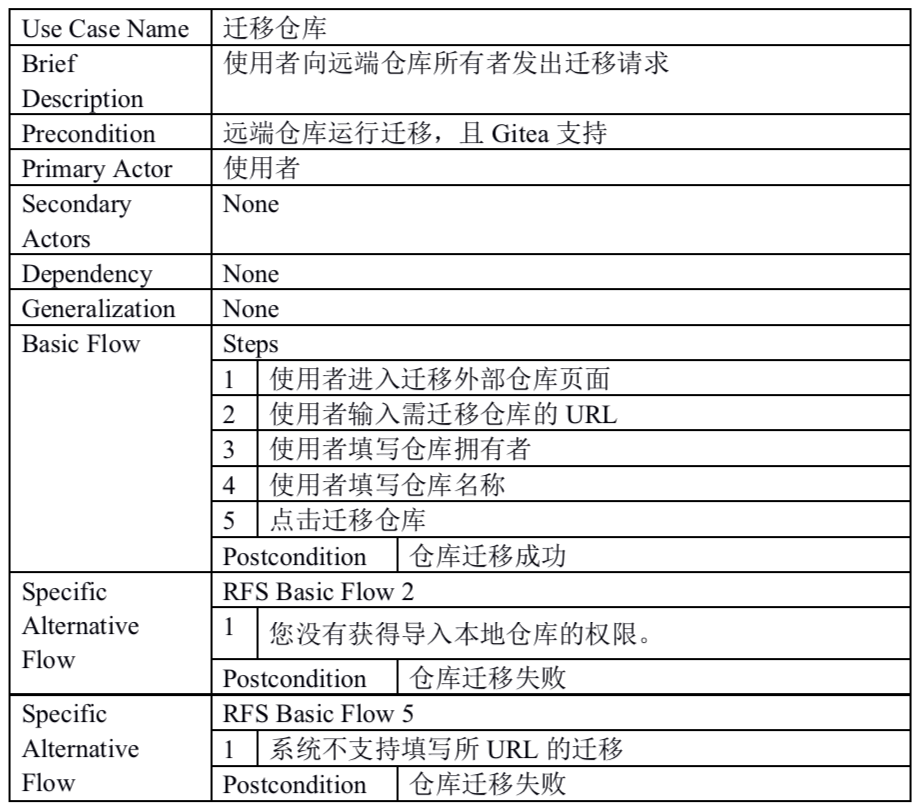


图 12 RUCM-迁移仓库

如图11和图12，使用者在创建完账户后，可以进行仓库的创建和迁移，仓库的命名会由系统进行有效性校验，无效命名会导致创建仓库失败。迁移仓库需要gitea支持的仓库类型，在配置完该远端仓库的URL、名字等必要信息后，Gitea会自动进行仓库迁移，成功后得到迁移后的仓库。迁移失败主要有三方面的原因，没有本地仓库更改的权限，需要管理员设置。或者Gitea不支持该类型仓库导致迁移失败，以及迁移信息有误，包括URL非法，仓库名称错误等等。

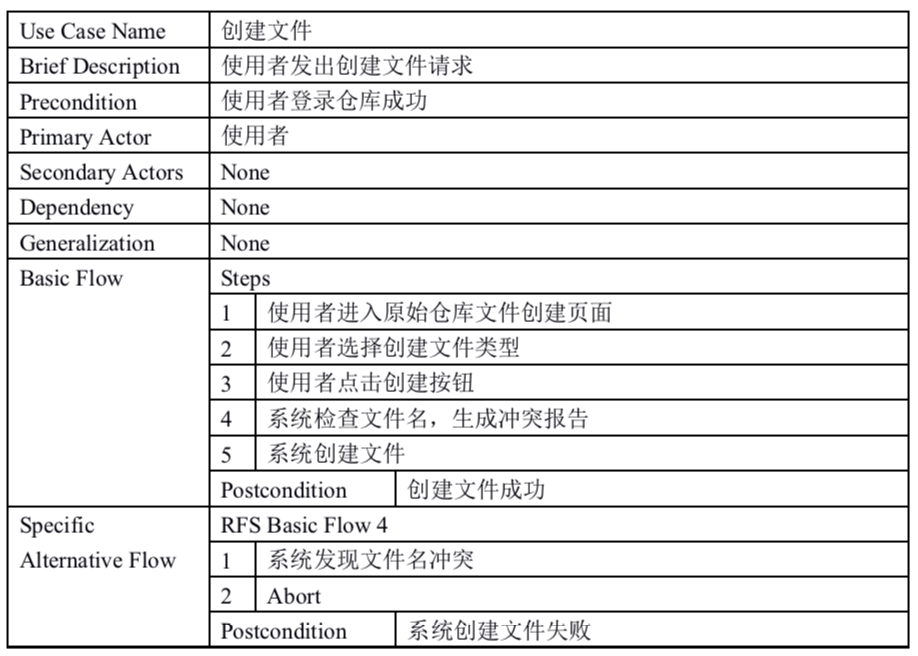


图 13 RUCM-创建文件



图 14 RUCM-删除用户

如图13，使用者在完成账户创建后，可以进行文件的创建，文件的创建主要检查文件名是否有效和是否与已有文件名冲突。

如图14，使用者可以对自己的账户进行删除，删除成功会发送邮件提示。

## 其他事务模块描述

### 日志模块

日志模块由XORM模块承担。

日志是一个接口，通过设置日志，可以显示SQL，警告以及错误等，默认的显示级别为INFO。可以设置日志的显示方式，这里总结为三种：

1. 控制台打印SQL语句
2. 控制台打印调试信息
3. 输出调试信息到文件中

### Toolbox工具箱

Toolbox进一步集成了大量的小型功能，比如进行服务器运行状况的健康检查，配置文件服务以及统计服务。该中间件可以单独使用命令行导入，也可以在开发环境进行导入。

### 缓存

xorm内置了一致性缓存支持，不过默认并没有开启。要开启缓存，需要在engine创建完后进行配置。

### 模块切换器

模块切换器为Macaron提供了主机切换功能。即在同一个程序中运行多个Macaron实例。如果这两个实例并不是平行关系而是父子关系，模块切换器一样可以达到效果。

### bindata

使用bindata可以将模版和公共目录转换为单独的包。

## 异常处理

Git模块的推拉等操作中出现的错误，会上抛到gitea.log.error中进行错误记录和处理。同样的，还有lexer模块中的一些异常也是。实质上gitea.log.error最终调用的还是fmt.error或者error.news。

Webhook模块的交付任务中，如果最终无法交付，则会抛到fmt.error中进行处理。同样的，password模块、记录测试模块以及bindata模块中出现的诸如文档类型不匹配、文档指针空的情况都会上抛到fmt.error进行处理。

# 运行环境说明

## 部署环境

表格 2 Gitea环境要求

|  |  |
| --- | --- |
| **类型** | **属性** |
| 操作系统 | 任何go语言支持的目标平台 |
| 编译软件要求 | >= go-1.11, git, npm |
| 运行硬件要求 | 内存 >= 1 GiB, 剩余空间 >= 500MiB |
| 运行软件要求 | 任何一种Gitea支持的数据库, git |

## 接口

1. 硬件接口：Gitea不直接使用或控制硬件。
2. 软件接口：Gitea提供Webhook方式扩充功能。
3. 通信接口：HTTP和HTTPS协议，SSH协议，Git协议，SMTP协议。
4. 用户接口：git工具，gitea命令行程序，WEB页面。

# 功能拓展与改进

结合上述对Gitea的自有功能的分析，我们提出了如下的功能拓展需求。

## 需求分析

通过对Gitea原有的功能体系进行梳理，可以发现Gitea已经向用户提供了比较完备的git托管平台功能，但是用户通知模块仍然比较薄弱，尚未形成一个完整的体系架构。另外，Gitea和其他托管平台的融合仍然有待提高。

### 业务需求

1. **通知**

对于普通用户来说，我们可以在以下的两个方面进行通知的改进：通知形式和通知内容。

通知形式上，Gitea已经为用户提供了邮件和网页消息两种通知方式。结合国内现状，越来越多的应用开始着力于移动端的功能开发，依托于现有的通用平台进行功能集成更是一种趋势。因此，Gitea的有必要结合移动端的优势，拓展通知的发布渠道。这为Gitea的架构设计提出了一些新的要求：

1. 设计一个通用的通知架构，利用用户注册的信息，将有关通知分发到各个目标。
2. 为用户预定义一些通知发布渠道，比如微信公众号等，并通过安装界面指导用户完成通知发布渠道的有关设置。
3. 部署者可以可选加载额外的模块，丰富Gitea实例的通知方式。

通知内容上，Gitea已经实现了一部分通知事件，但是缺乏对通知事件的统一管理。因此，有必要针对用户开发一个统一的通知内容管理界面，在其中完成通知事件的注册、取消以及通知形式的选择。另外，更加丰富的通知事件也有利于用户在平台上的参与度。如何快速定义和注册通知类型，则成为了丰富通知内容的一个要点。为了实现这一目标，在Gitea的代码中有必要引入更加抽象的通知类型，并简化通知的注册方式。

1. **多平台融合**

Gitea作为一个git托管平台，和其他类似的托管平台功能类似。为了促进其他平台的用户转移到Gitea平台上，有必要引入更加方便的数据转移方式。Gitea已经向用户提供了Github的迁移方式，但仍然没有Gitlab的相关选项。因此，我们将在Gitea中添加Gitlab的迁移方式，使用户能更加顺利地从Gitlab平台转移到Gitea平台上。

### 功能需求

针对上述的业务需求分析，我们总结出了以下的拓展功能：

1. 通知方式管理接口：负责管理Gitea提供的所有通知方式，提供一个统一的通知信息分发接口，向用户提供灵活的通知方式拓展接口，依据实现难度可以在编译期或者运行期进行通知方式功能模块加载。
2. 微信通知模块：为用户提供微信公众号的通知形式。
3. 通知事件管理接口：负责管理所有的通知事件，利用通知方式管理接口转发通知消息。
4. 通知事件定义接口：负责为开发者提供合理的通知事件编写接口，并自动向通知事件管理接口注册新的通知类型。
5. Gitlab迁移接口：负责迁移Gitlab平台上的仓库数据。

### 非功能需求

我们的功能拓展在满足正确性的基础上必须满足如下的非功能需求：

1. 功能兼容性。我们的功能需求需要在Gitea原有的功能上进一步开发，因此必须保证我们的代码不破坏原有代码的功能特性。
2. 运行兼容性。Gitea选择使用Go语言进行开发，获得了多个平台的兼容性。我们的功能拓展也必须尽量保持多平台兼容。
3. 可测试性。Gitea提供了完整的测试框架进行代码测试，我们的功能拓展必须利用原有的测试框架完成拓展代码的测试运行，并在文件结构中提供充分的测试用例完成功能测试。
4. 代码风格一致性。Gitea已经定义了一套代码规范，因此在Gitea基础上的开发必须保证风格保持一致。

## 用例建模

### 用户角色

功能扩展主要面对的有三类用户：

1. 开发自定义通知模块的开发者：开发者主要利用功能扩展提供的开发接口开发新的通知模块，以向Gitea提供可供加载的新型通知方式。
2. 安装Gitea的部署者：部署者在安装Gitea的时候可以选择通知模块加载至Gitea中，并对通知模块要求的额外配置进行相应设置。
3. 使用Gitea的用户：用户可以选择自己的通知内容和通知形式，创建新的仓库内容。

### 用例图



图15使用者和部署者用例图



图 16 开发者用例图

## RUCM建模



图 17 加载通知模块

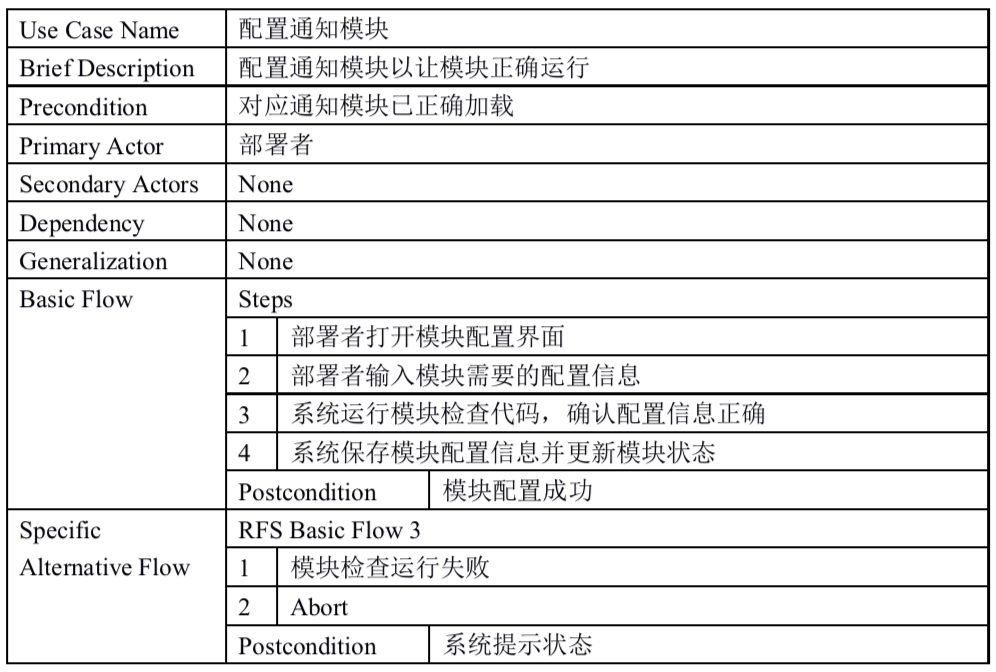


图 18 配置通知模块

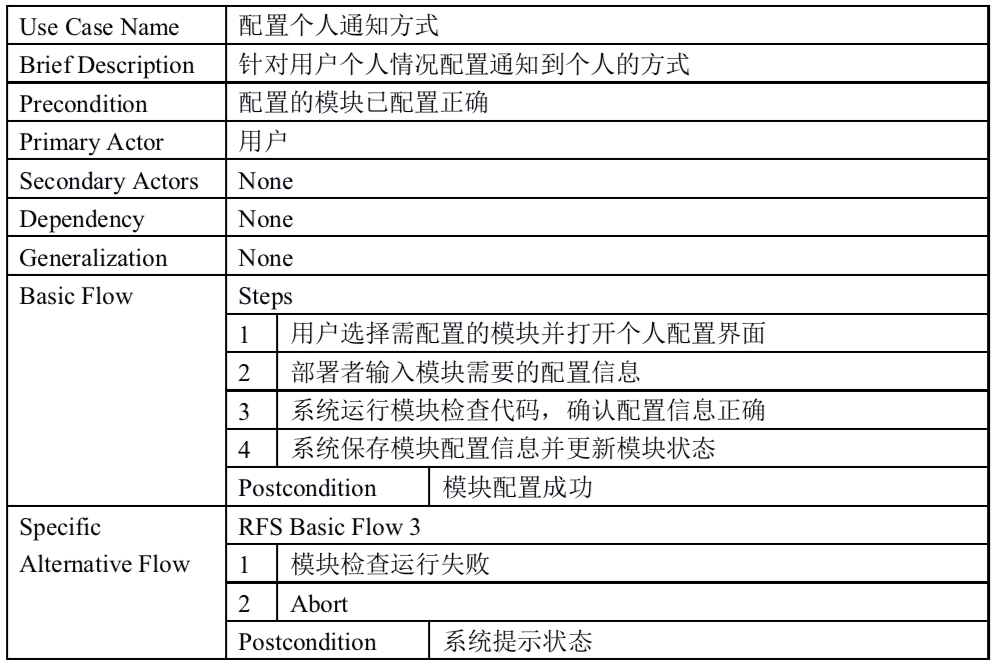


图 19 配置个人通知方式



图 20 订阅通知

如图17，在Gitea安装后，可以选择加载该通知模块，在校验模块正确性和完整性完成后，加载模块并配置初始信息。加载失败系统会提示状态。如图18、19，加载通知模块完成后，可以进行配置的更新，系统会自动检查配置的正确性，确保模块的正确运行。

如图20和图23，因为通知事件的级别不一、种类繁多，需要确定使用者自身关注的事件，通过订阅通知的子模块实现触发事件的通知或者解除通知。如图21，使用者还可以单独为该订阅事件设置通知方式。

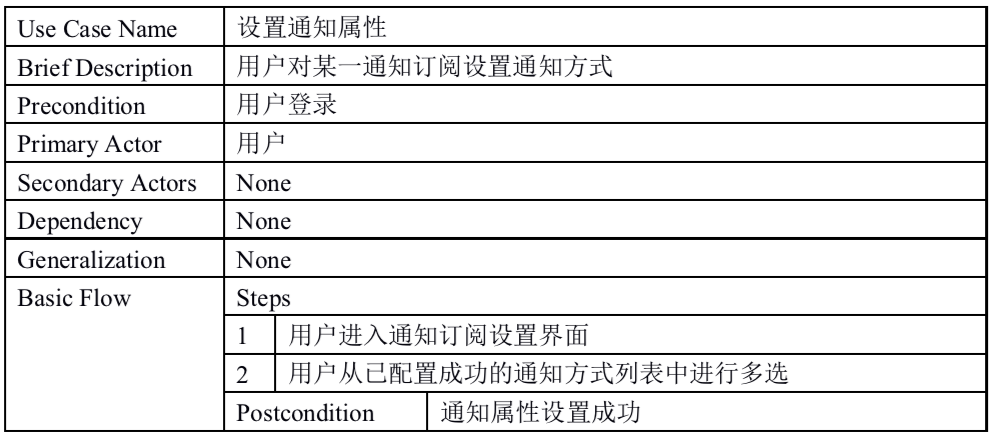


图 21 设置通知属性



图 22 迁移GitLab项目

如图22，在正确输入GitLab仓库的URL、用户名等信息后，系统会检查仓库的有效性，同时检查访问的权限，如果信息有效，Gitea会自定迁移GitLab项目仓库到使用者的Gitea仓库当中。否则，返回迁移失败的状态信息。

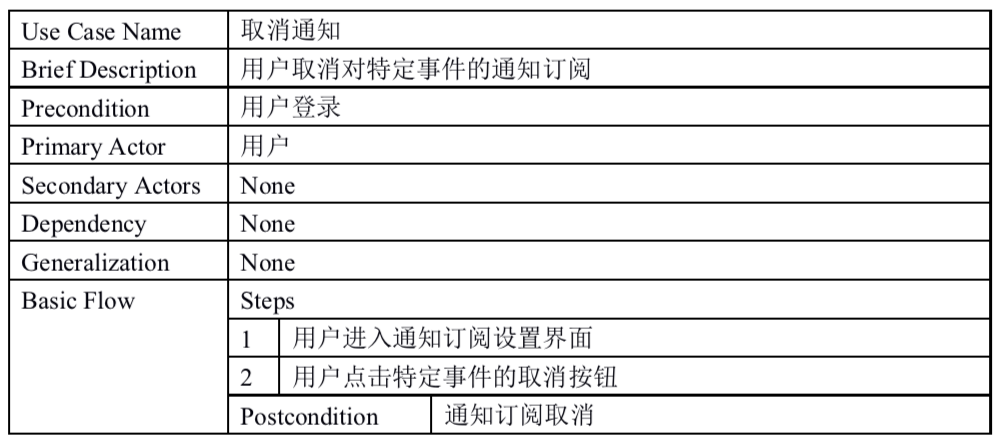


图 23 取消通知