# 绪论

## 课题研究背景

随着互联网技术的飞速发展，软件项目种类繁多，软件质量的保障显的尤为重要。然而，就目前而言，国的软件测试仍然处于起步阶段，例如一些小型的软件开发公司，软件测试人员专业程度不够，甚至没有专业的软件测试人员，更没有一套适合自己的 Bug 管理系统，有的软件公司还在使用表格来进行bug的描述，人工进行Bug的统计和追踪、统计，毫无时效性，严重影响软件开发的速度，并且难以确保软件的质量。

如何在多变、复杂的软件开发中进行Bug 管理，并针对团队的现状对bug问题单的流转进行相应的调整，让 Bug 管理系统可以更好的融入到团队开发中，从而更好的加快软件开发的速度，保证开发软件的质量，是每个软件公司都在考虑的问题。如何让 Bug 的流转更加灵活高效呢？这就需要工作流技术的支持。如今，hu工作流（Work Flow）引擎被广泛应用于各种信息化系统中将原本散乱甚至混乱的业务梳理后制定成业务规范流程，进而约束业务的规范化处理和运转。需求人员、开发人员共同协作制定了符合BPMN 2.0 规范的流程定义，之后将其部署到工作流引擎中，由它自动驱动业务流程的进行。目前，工作流引擎中的Activiti的兴起，可以将复杂的Bug 流转大大的简化，并且它集成的流程在线设计器也可以对bug 的流转进行相应的调整。

## 课题研究的现状

随着互联网技术的兴起与发展，人们对软件的需求越来越多，要求也越来越苛刻。为了提高软件开发的进度和质量，除了软件测试人员越发受到各个公司的亲赖外，简单、实用的bug管理系统也越发显的重要。。为了对软件开发周期中发生的 bug 有一个及时有效的管理方式，Bug 管理系统的设计与实现便显的尤为重要。

本系统的设计与实现，使用Java 框架 SSH并集成工作流引擎 Activiti 框架，提供在线的 Bug 问题单的流转设计，根据自己团队的需要，设计符合自己团队开发的的问题单流转流程。

## 课题研究的意义

决定做这个 bug 管理系统初衷，与我的一小段的实习经历有关，我曾在一家小公司从事 Java实习开发职位，因为没有专业的测试人员，也没有一个有效的 bug 管理工具，在团队软件开发的过程中，成员之间发现 bug 通过 QQ 的方式在群里交谈，对于发现的bug 没有相应的系统来对它们进行管理和追踪，从而严重的拖长了软件的交付周期，也影响了软件开发的质量。所以想要动手开发一套简单、实用的 bug 管理系统。

我们都知道，在软件开发的过程中，软件测试对软件开发的重要意义不言而喻，当下软件开发周期越来越短，用户队软件质量的要求也越来越高。传统的手工处理Bug 和复杂的Bug流转让参与用户要花费大量的精力去关心Bug 的流转问题上，这使得工作的重心产生了偏移，而且在实际的项目开发过程中，项目组的成员并不是一成不变的，软件bug的流转可能根据项目组人员的变动，职责的变动或者其他一些原因，产生变化，这个时候为了实时追踪 Bug ，就需要调整原有的Bug 流转流程，确保流转流程的完整性和有效性。

Bug 管理系统可以高效的流转测试人员提出的 bug，在 Activiti 的调动下，将bug 流转到下一个参与用户的活动任务（Task） 节点上，每个人并不需要关心整个流程的下一个节点，只需要认真自己处理分配给自己的任务，这样加快了软件测试的周期的同时也保证了软件开发的质量，可以有效的缩短开发周期，减少开发成本避免不必要的经济损失。

# 主要技术

## 开发环境

* 服务器中间件：在 Java EE 7 规范下（Servlet 3.1、JSP 2.3）下开发，应用服务器中间件 Tomcat。
* 数据库：MySQL 5.7。
* 开发环境：Java EE 、Maven、Git

## 主要技术简介

### SSH

SSH，即Spring MVC 、Spring 、Hibernate。是当下较为流行的一种Web应用程序开源框架。

* Spring MVC

Spring MVC框架是有一个MVC框架，通过实现Model-View-Controller模式来很好地将数据、业务与展现进行分离。Spring MVC 的设计是围绕 DispatcherServlet 展开，它是一个普通的轻量级的 Servlet。和Struts2 基于类的字段的方式获取属性值，Spring MVC 通过方法参数获取属性值更加的轻量级，也能避免不必要的线程安全问题。和 Spring 的集成也要比 Struts2 更高效和方便，Struts2 和 Spring 集成时，Spring 管理bean 的方式默认是单例，这样及其容易造成线程完全问题，通过设置将 Struts2 的 Action 修正为原型，这样会造成性能的缺失。而且由于设计问题，Struts2 利用过滤器(Filter)的方式虽然可以说很好的处理请求和响应，但对于不必要的请求与响应依然进行繁琐的递归处理，另外Struts2还存在者一些安全漏洞。相比来说，Spring MVC 作为 MVC 框架更加的轻量级、安全、高效。



图 2-1 Spring MVC 请求处理流程图

* Spring

Spring框架是一个基于 Java 开发的平台，它为你的 Java 引用提供了非常全面的技术架构的支持，让您可以专注于您的应用程序。

Spring 允许创建 POJO (plain old java object) 和 非侵入性的企业应用服务，是轻量级的容器。它的两大核心是DI（Dependency Injection ） 和 AOP(Aspect Oriented Programming)。DI，即依赖注入，通过Spring 容器来管理Java 应用程序中实例的生命周期，这样的好处就是可以再运行时，通过Spring 容器动态的创建实例，动态的为注入的对象添加新的行为。AOP，面向切面编程，可以进行功能性的扩展，例如日志记录，事物控制，权限控制等。Spring 功能十分强大。

对于Java 开发者来说，Spring 提供的便利性和优势不言而喻。首先，它是轻量级容器框架，没有侵入性；DI 和 IOC 更加容易组合对象之间的关系，通过面向接口编程，可以降低系统组件之间的耦合度；AOP 可以更加容易的进行功能扩展，遵循 OCP(Open Closed Principle)。提供对持久层（Persistence），事物（Transaction）的支持。除此之外，对于现存的各种框架，Spring 也提供了与它们整合的方案。Spring 更像是一场战争中的统帅，贯穿着表现层、业务层、持久层，管理着所有 bean 的生命周期和组合方式。



图 2-2 Spring 框架模块结构

* Hibernate

Hibernate 是一个基于 Java 开源的持久化中间件。它对 JDBC 做了轻量级的封装。采用 ORM(Object-Relational Mapping) 映射机制，负责实现 Java 对象关系转换为关系数据库之间的映射。Hibernate把 SQL 语句传给数据库，并且把数据库返回的结果分装成对象。内部封装了 JDBC 访问数据库的操作，向上层应用提供了面向对象的数据库访问 API。如下图 2-3所示，Hibernate 介于传统 Java 对象和数据库服务器之间，通过Hibernate API对数据进行持久化操作。



图 2-3 Hibernate 位置关系

### 工作流框架

一个完整的工作流生命周期会经过5个步骤，定义、发布、执行、监控、优化。对于工作流来说，最重要的便是它的描述文件BPMN（Business Process Modeling Notation）,即业务流程建模标注，是图形化对象的网图，包括活动和定义操作顺序的流控制，通俗来讲这个文件定义了 工作流的流程，是整个工作流业务流转的关键文件。BPMN 2.0由 BPMN 标准组织发布的，它不仅在流程设计器上可以通用，还可以在任何符合BPMN2.0 规范的流程引擎（如 Activiti ）上执行。Activiti 支持并扩展了BPMN 2.0。

Activiti 是一个针对企业用户、开发人员、系统管理员的轻量级工作流业务管理平台，其核心是使用 Java 开发的快速稳定的 BPMN 2.0 流程引擎。 Activiti 是在 ApacheV2 许可下发布的，可以运行在任何类型的 Java 程序中，例如服务器、集群、云服务等。 Activiti可以完美地与 Spring 集成。同时，基于简约思想的设计使 Activiti 非常轻量级，是目前非常流行的工作流框架。它再国外已经被很多厂商所使用，在国内的发展正在呈直线趋势上升。已经成立了由很多热心的技术爱好者参与的技术社区。很多项目，新产品也都开始采用 Activiti 作为新一代工作流引擎。

Activiti 流程设计器，Activiti 团队专门设计了用来设计BPMN 2.0 规范的流程设计器—Eclipse Designer。此外还有Signavio 公司为Activiti 定制的基于Web 的Activiti Modeler 流程设计器。它原生支持 Spring ，分离运行时和历史数据。

Activiti 数据持久化是 使用 MyBatis 作为和数据库交换数据，比 Hibernate 更加轻量级，通过最优的 SQL 语句执行 Command ，可以让引擎再速度上保持最高的性能。Activiti 也是采用分层设计，接口编程，并且提供了七大接口，基本满足了工作流相关的功能。

表格 2-1 Activiti 的 七大接口

|  |  |
| --- | --- |
| Service 接口 | 作用 |
| RepositoryService | 用于管理流程仓库，例如，部署、删除、读取流程资源 |
| IdentifyService | 可以管理和查询用户、组之间的关系 |
| RuntimeService | 可以处理所有正在运行状态的流程实例、任务等 |
| TaskService | 用于管理、查询任务，例如，签收、办理、指派等 |
| FormService | 用于读取和流程、任务相关的表单数据 |
| HistoryService | 可以查询所有历史数据，流程实例、任务、活动、变量、附件等 |
| ManagementService | 和具体业务无关，主要是可以查询引擎配置、数据库、作业等 |

### 权限框架

Apache Shiro 是一个 Java 安全框架。就目前而言， 使用Apache Shiro 的开发者逐渐多了起来。因为它使用起来十分的简单，和 Spring Security 相比它更加的轻量级。

Shiro 可以帮助我们完成，认证，授权，加密，会话管理，与Web 集成，缓存，但它不会去维护用户和维护权限，这些需要我们自己去设计和提供，然后通过相应的接口注入 Shiro 即可。



图 1-4 Shiro 框架模块结构

### 日志框架

SLF4J(Simple Logging Facade For Java)提供了操作日志的相应接口规范，而 Log4J(Log For Java ) 框架则是针对该规范接口的对应实现。使用 SLF4J相当于一层接口，使用设计模式中的外观模式，可以很好的分离实现，而不依赖于具体的日志实现矿建。通过默认的 log4.properties 文件，通过对日志输出级别的控制，和相应的设置，可以很容易的实现对程序运行过程中日志信息的过滤，并且可以对重要的日志信息进行文件存储。

### 布局框架

SiteMesh 是一个轻量级且灵活的Java web 应用框架，基于 Servlet 中的 Filter采用了设计模式中的装饰者模式(decorator pattern) 可以很容易的对页面进行布局。它的处理流程图如下图所示。



图 2-5 SiteMesh 布局框架请求处理流程

### 前端框架列表

* JS 框架：JQuery
* CSS 框架：Bootstrap inspinia
* 客户端校验：JQuery Validation
* 富文本：CKEcitor
* 文件管理：CKFinder
* 数据表格：jqGrid
* 对话框：jQuery jBox
* 下拉框：jQuery Select2
* 树形结构控件：JQuery zTree
* 图表控件：Chart
* webim 聊天插件：layim

# 需求分析

## 系统功能需求分析

通过分析，对Bug 管理系统进行如下设计。

为了体现软件的高内聚、低耦合，按功能模块进行划分，大致划分为如下模块：

* 系统模块：用户，角色，权限，部门 管理
* bug 模块：bug，项目管理
* 工作流模块：流程，模型管理
* 通信模块：站内通知，聊天管理

其中 Bug 管理模块为本系统的核心，在实际的开发过程中我们可以了解到，在软件测试的过程中，参与的角色可以划分为：

* 测试人员：负责提交 bug ；重新验证修复的bug
* 测试主管：负责审核新建的bug；确认bug 是否有效
* 项目经理： 负责处理有效bug
* 开发主管：负责处理有效 bug
* 开发人员：负责修复分配的 bug

根据软件测试过程中角色的划分，本系统在启动时，会提供上述分析的角色和系统管理员角色。也会提供一个平台默认的 bug 流转流程，即，测试人员发现提交bug，测试主管将有效 bug 流转到 项目经理，项目经理决定 bug 是否要被处理，若需要处理，则将bug 分配个开发人员。开发人员进行修复，修复完毕后，bug 将等待测试人员验证，验证已修复，则关闭bug。当然，通过在线的流程设计器，也可以对平台默认的bug 流转流程进行相应的调整。

在 bug 的流程处理过程中，可能会出现 bug 的分歧，需要及时沟通，需要简单的聊天功能。

综上分析，要求bug管理系统应努力实现如下功能：

* 细粒度的权限管理功能，保证系统的安全。
* 用户登录系统可以看到自己参与项目的bug 情况，以及待自己处理的任务等详细信息。
* 以图表的形式形象、生动的显示项目 bug 状态的分布图 和 bug 状态的趋势图。
* 在线流程设计器，新建模型流程或对已有的流程进行在线的修改优化。

## 关键业务流程

## bug 生命周期

从下图我们很直观的看出 bug 的生命周期。

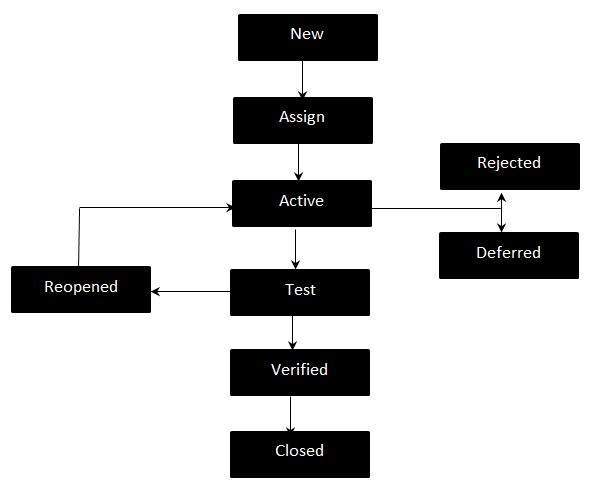


图 3-1 bug 生命周周期

通过上图的分析，我们可看出 bug 的生命周期一定程度反应一次Bug 提交的过程中 bug 的状态。根据图3-1 的上图分析，并结合相应的实际情况，对其进行调整，可以得到如下 Bug 状态：

* 新建(New)：缺陷首次被提交，默认状态为新建；
* 有效(Genuine)：审核通过的缺陷；
* 分配(Assign)：缺陷被分配给处理缺陷的成员；
* 打开(Open)：缺陷被分配的成员签收；
* 检测(Test)：缺陷被修复，待验证；
* 验证(Verified): 缺陷被成员验证；
* 重新打开(Reopened)：验证后，缺陷依然存在；
* 拒绝(Rejected)： 缺陷被拒绝；
* 延期(Deferred)：此次不修复该缺陷

### 3.2.2. bug 类型

在本系统中，我们将bug 类型划分为

* bug
* 改进
* 任务
* 需求

### 3.2.3. bug 优先级

在本系统中，依据 bug 的严重程度将bug 划分为如下优先级

* 紧急：非常严重 bug，需要立即解决，若不解决，基本功能无法使用，测试没

有办法进行。

* 高：严重bug
* 普通：常见的 bug
* 低：建议类 bug

### 3.2.4. BPMN 流程定义图

Bug 管理系统中 bug 的流程定义如下图所示

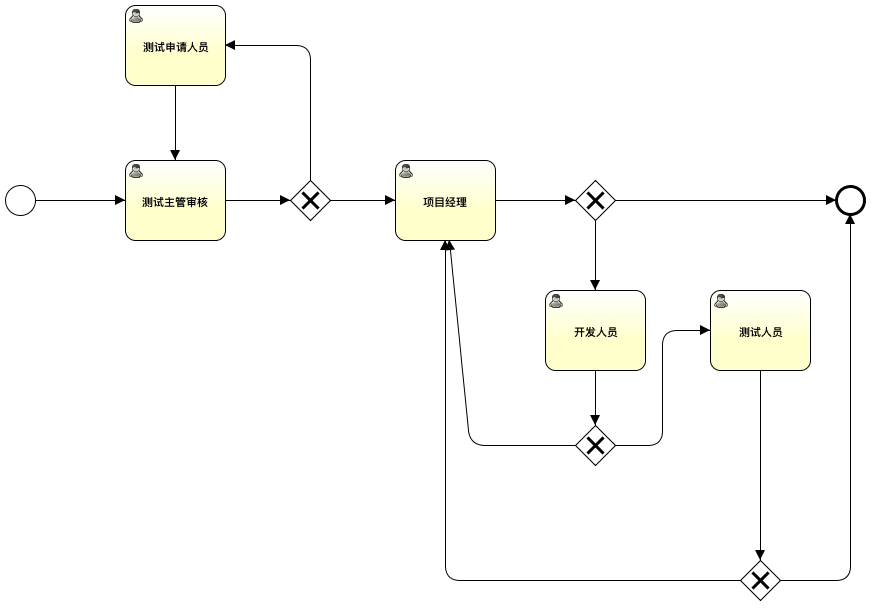


图 3-2 平台默认的 bug 流转流程图

## 系统用例图

### 系统模块

本模块由主要由用户管理，角色管理，权限管理，部门管理，字典管理，个人文件管理功能组成。

本平台初始化系统管理员拥有本系统最高权限，即可以对用户，角色，权限，部门，字典，文件进行管理。本平台普通角色成员，可以进行个人信息的管理，例如查看个人信息、修改个人信息、修改密码，个人文件管理，而项目经理在此基础之上，可以对用户进行一定程度上的管理。

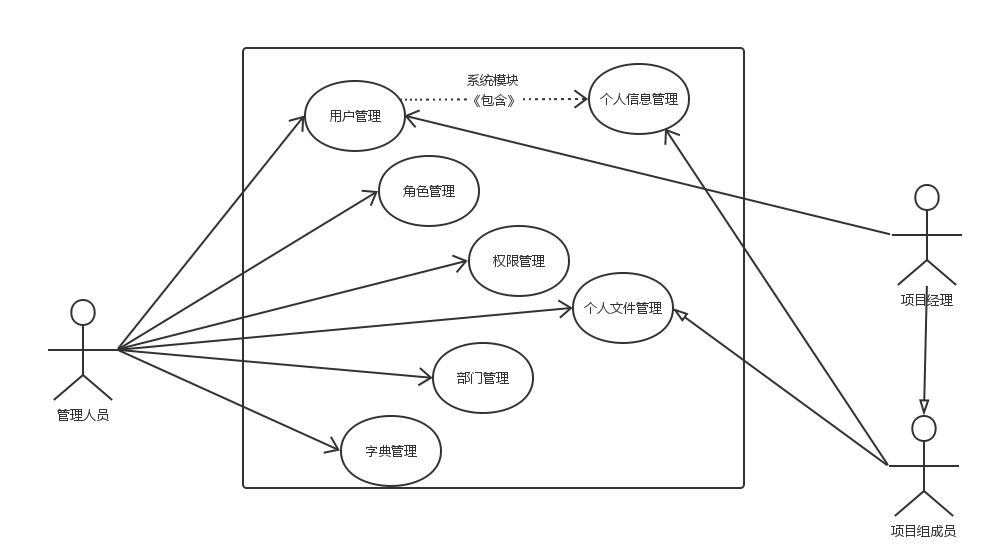


图3-3系统模块用例图

### bug 模块

本模块由项目管理 和 bug 管理组成。

项目管理包含项目创建、查看项目信息、编辑项目、给项目组分配人员、项目统计。 bug 管理包含 bug 提交、bug 统计、bug 任务分配、bug追踪、bug 查询。

项目经理负责创建项目及其版本号，测试人员选择项目及其对应的版本号提交软件测试过程中发现的 bug ，工作流模块的调度之下，bug 的任务流转到项目组中的各个成员。项目组成员选择对应的项目可以看到项目的基本信息，项目的bug 统计分布、分配给我的任务、我提交的bug、我参与的bug 流程。

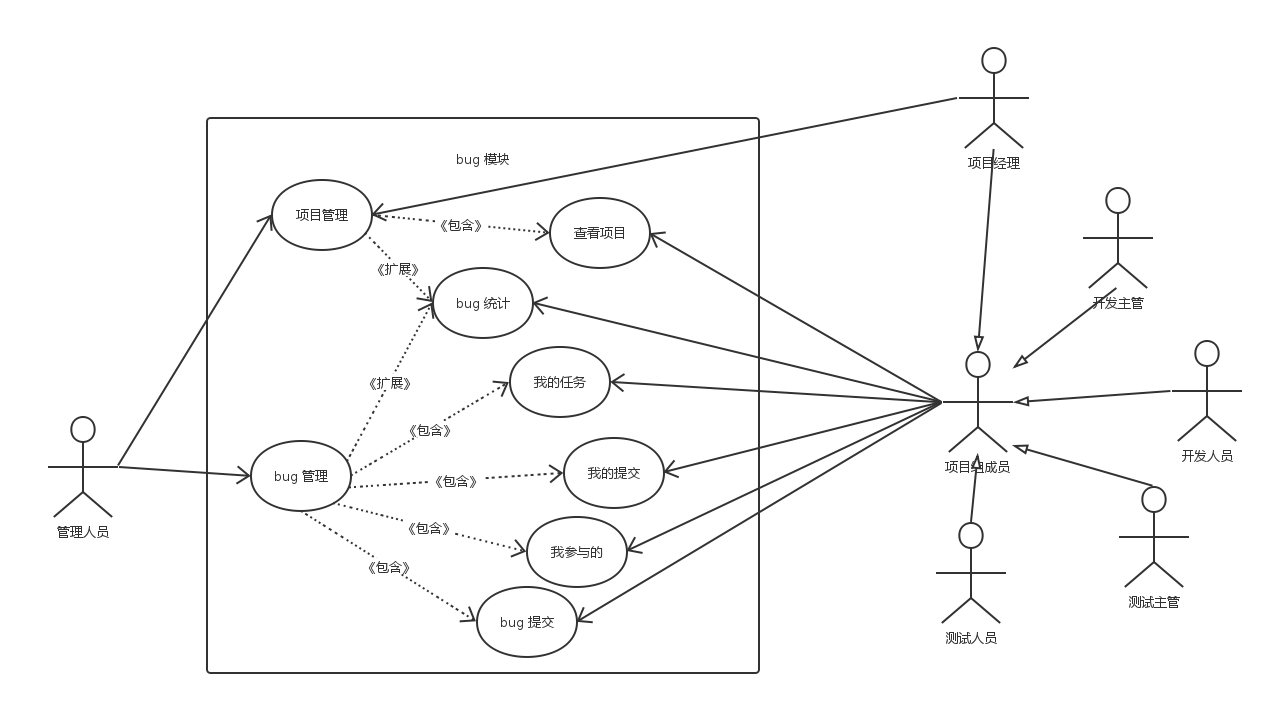


图3-4 bug 模块用例图

### 工作流模块

本模块由流程管理和模型管理两大部分组成。

模型管理包含新建模型、编辑模型、部署模型、导出模型、删除模型。流程模型包含挂起、删除、转化为模型、部署流程、删除运行中流程。

管理人员和项目经理通过新建模型，或是修改已有的模型，然后使用 Activiti 的在线流程编辑器，在本系统就可以新建 bug 流转流程或是对已有的 bug 流转流程进行修改。通过导入模型功能，导入模型进行在线编辑，然后进行部署，将模型转换为流程。通过对运行中的流程进行挂起，然后将其转化为模型，对其进行模型编辑，对模型进行调整和优化后，将其部署并转化为流程，对 bug 进行流程流转。

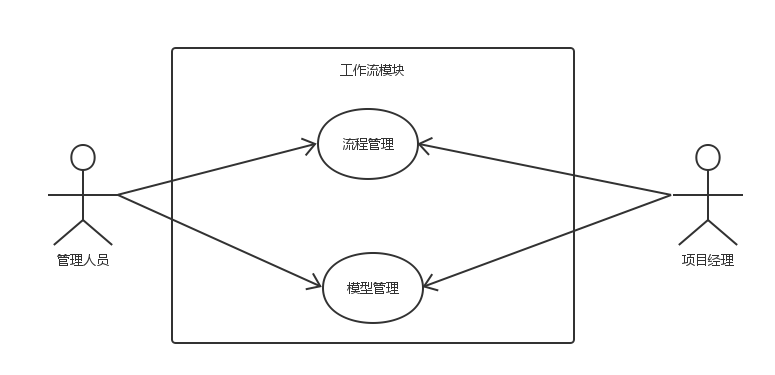


图 3-5 工作流模块用例图

### 通信模块

本模块分为站内通知和简单的实时聊天功能。

项目经理可以通过通知功能下达一些 bug 通知和要求，以及一些文档。平台用户也可以通过实时聊天功能，在对 bug 的处理过程中进行 bug 的沟通和交流。

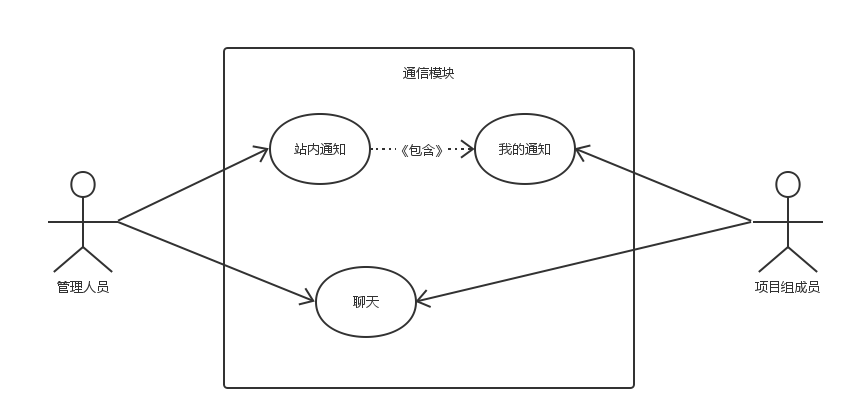


图 3-6通信模块用例图

# 总体设计

## 系统功能框架

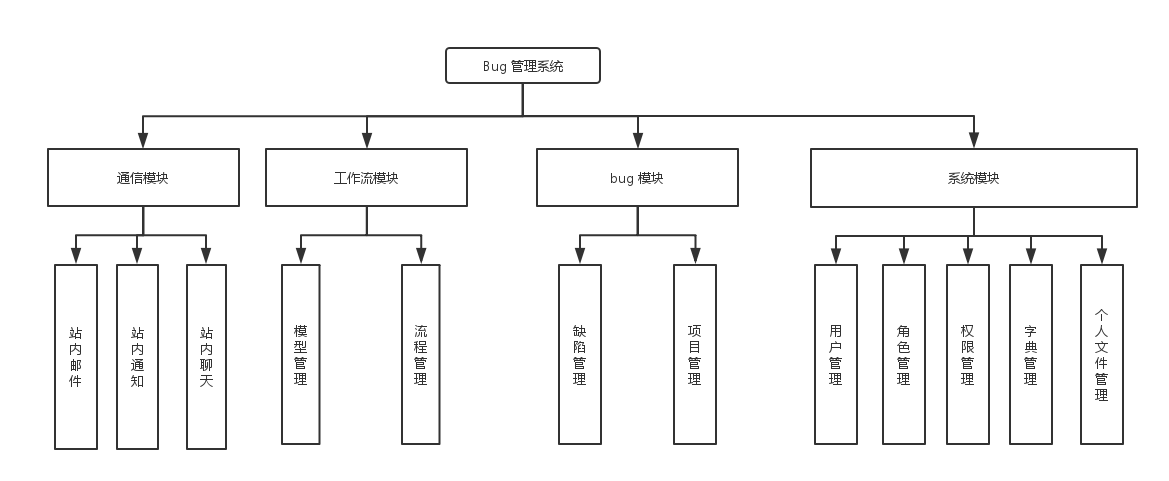


图 4-1 系统功能框架图

## 数据库模型设计

本 bug 管理系统共有 张表，除去工作流相关 23 张表外，和本系统业务相关的表有14张表，分析设计实体及其实体关系如下。

用户-角色：一个用户有多个角色，一个角色有多个用户。

用户-部门： 一个部门有多个用户，一个用户隶属于一个部门。

部门-部门：一个用户有多个下级部门，一个下级部门只有一个直接上级部门。

角色-权限： 一个角色拥有多个个权限，一个权限可以分配给多个角色。

用户-项目：一个用户参与多个权限项目，一个项目分配给多个用户。

bug-项目：一个缺陷属于一个项目，一个项目有多个缺陷。

用户-用户：一个用户有多个好友， 一个用户被多个人加为好友。

用户-bug ：一个用户报告多个 Bug，一个 Bug 只能被一个用户报告。

项目-版本：一个项目有多个版本，一个版本只对应一个项目。

用户-通知：一个用户发送多个通知，通知被下达给多个用户。

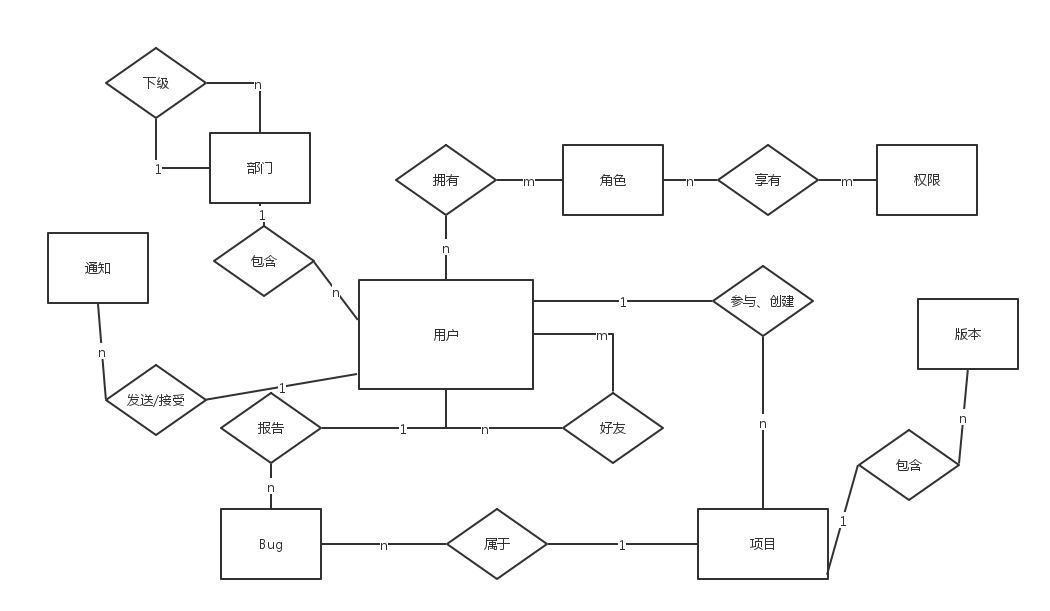


图 4-2 E-R 图

## 数据库表设计

* sys\_user 表是用来保存用户的基本信息

表格 4-1 用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| id | varchar(64) | NO | YES |  |
| company\_id | varchar(64) | NO |  | 公司编号 |
| office\_id | varchar(64) | NO |  | 部门编号 |
| login\_name | varchar(100) | NO |  | 登录名 |
| password | varchar(100) | NO |  | 密码 |
| no | varchar(100) | YES |  | 工号 |
| name | varchar(100) | NO |  | 姓名 |
| email | varchar(200) | YES |  | 邮件地址 |
| phone | varchar(200) | YES |  | 联系方式 |
| mobile | varchar(200) | YES |  | 手机号 |
| user\_type | char(1) | YES |  | 用户类型 |
| photo | varchar(1000) | YES |  | 照片 |
| login\_ip | varchar(100) | YES |  | 登录IP |
| login\_date | datetime | YES |  | 登录日期 |
| login\_flag | varchar(64) | YES |  | 登录标记 |

* sys\_role 表是用来保存角色的基本信息

表格 4-2 角色表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| id | varchar(64) | NO | YES | 编号 |
| office\_id | varchar(64) | YES |  | 归属机构 |
| name | varchar(100) | NO |  | 角色名称 |
| enname | varchar(255) | YES |  | 英文名称 |
| role\_type | varchar(255) | YES |  | 角色类型 |
| data\_scope | char(1) | YES |  | 数据范围 |
| is\_sys | varchar(64) | YES |  | 是否系统数据 |
| useable | varchar(64) | YES |  | 是否可用 |

* sys\_office 表是用来保存部门的基本信息

表格 4-3 部门表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| parent\_id | varchar(64) | NO |  | 父级编号 |
| parent\_ids | varchar(2000) | NO |  | 所有父级编号 |
| name | varchar(100) | NO |  | 名称 |
| sort | decimal(10,0) | NO |  | 排序 |
| code | varchar(100) | YES |  | 区域编码 |
| type | char(1) | NO |  | 机构类型 |
| grade | char(1) | NO |  | 机构等级 |
| address | varchar(255) | YES |  | 联系地址 |
| zip\_code | varchar(100) | YES |  | 邮政编码 |
| master | varchar(100) | YES |  | 负责人 |
| phone | varchar(200) | YES |  | 电话 |
| fax | varchar(200) | YES |  | 传真 |
| email | varchar(200) | YES |  | 邮箱 |
| useable | varchar(64) | YES |  | 是否启用 |

* sys\_menu 表是用来保存权限的基本信息

表 格 4-4 权限表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| id | varchar(64) | NO | YES | 编号 |
| parent\_id | varchar(64) | NO |  | 父级编号 |
| parent\_ids | varchar(2000) | NO |  | 所有父级编号 |
| name | varchar(100) | NO |  | 名称 |
| sort | decimal(10,0) | NO |  | 排序 |
| href | varchar(2000) | YES |  | 链接 |
| target | varchar(20) | YES |  | 目标 |
| icon | varchar(100) | YES |  | 图标 |
| is\_show | char(1) | NO |  | 是否在菜单中显示 |
| permission | varchar(200) | YES |  | 权限标识 |

* sys\_user\_role 表是用来保存用户角色关系

表 格 4-5 用户角色表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| user\_id | varchar(64) | NO | YES | 用户编号 |
| role\_id | varchar(64) | NO | YES | 角色编号 |

* sys\_role\_menu表是用来保存角色和权限关系

表 格 4-6 角色权限表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| role\_id | varchar(64) | NO | YES | 角色编号 |
| menu\_id | varchar(64) | NO | YES | 菜单编号 |

* sys\_role\_office 表是用来保存角色和部门关系

表 格 4-7 角色部门表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| role\_id | varchar(64) | NO | YES | 角色编号 |
| office\_id | varchar(64) | NO | YES | 机构编号 |

* sys\_user\_friend 表是用来保存用户好友关系

表 格 4-8 用户好友表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| id | varchar(64) | NO | YES | 编号 |
| userId | varchar(64) |  |  | 用户编号 |
| friendId | varchar(64) |  |  | 好友编号 |

* sys\_dict 表是用来保存系统数据字典信息

表 格 4-9 数据字典表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| id | varchar(64) | NO | YES | 编号 |
| value | varchar(100) | NO |  | 数据值 |
| label | varchar(100) | NO |  | 标签名 |
| type | varchar(100) | NO |  | 类型 |
| description | varchar(100) | NO |  | 描述 |
| sort | decimal(10,0) | NO |  | 排序 |
| parent\_id | varchar(64) | YES |  | 父级编号 |

* oa\_notify 表是用来保存通知信息

表 格 4-10 通知表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| id | varchar(64) | NO | YES | 编号 |
| type | char(1) | YES |  | 类型 |
| title | varchar(200) | YES |  | 标题 |
| content | varchar(2000) | YES |  | 内容 |
| files | varchar(2000) | YES |  | 附件 |
| status | char(1) | YES |  | 状态 |

* oa\_notify\_record 使用来保存通知关系信息

表 格 4-11 通知联系表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| id | varchar(64) | NO | YES | 编号 |
| oa\_notify\_id | varchar(64) | YES |  | 通知通告ID |
| user\_id | varchar(64) | YES |  | 接受人 |
| read\_flag | char(1) | YES |  | 阅读标记 |
| read\_date | date | YES |  | 阅读时间 |

* bug 表是用来保存bug 信息

表 格 4-12bug 内容表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| id | varchar(64) | NO | YES | 主键 |
| bug\_version\_id | varchar(64) | NO |  | 版本主键 |
| bug\_project\_id | varchar(64) | NO |  | 项目主键 |
| bug\_type | varchar(64) | YES |  | 类型 |
| bug\_status | varchar(64) | YES |  | 状态 |
| bug\_level | varchar(64) | YES |  | 优先级 |
| name | varchar(64) | YES |  | 名称 |
| summary | varchar(255) | YES |  | 简介 |
| description | longtext | YES |  | 内容详情 |
| file | varchar(255) | YES |  | 文件 |
| image | varchar(255) | YES |  | 图片 |
| proc\_ins\_id | varchar(64) | NO |  | 流程实例  编号 |

* bug\_project 表是用来保存项目信息

表 格 4-13 项目表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| id | varchar(64) | NO | YES | 编号 |
| name | varchar(64) |  |  | 名称 |
| summary | varchar(255) |  |  | 简介 |

* bug\_version 表是用来保存项目版本信息

表 格 4-14项目版本表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| id | varchar(64) | NO | YES | 编号 |
| version | varchar(64) | YES |  | 版本 |
| build | varchar(64) | YES |  | build |
| bug\_project\_id | varchar(64) | YES |  | 项目编号 |

* ACT\_GE\_BYTEARRAY保存流程定义 BPMN 流程定义描述文件，即用来保存二进制数据信息。

表 格 4-15 资源内容表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 主键 |
| REV\_ | int(11) | YES |  | 版本 |
| NAME\_ | varchar(255) | YES |  | 部署文件名 |
| DEPLOYMENT  \_ID\_ | varchar(64) | YES |  | 部署表编号 |
| BYTES\_ | longblob | YES |  | 部署文件 |
| GENERATED\_ | tinyint(4) | YES |  | 是否生成 |

* ACT\_GE\_PROPERTY 属性数据表，记录整个流程引擎级别的数据

表 格 4-16 资源属性表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| NAME\_ | varchar(64) | NO | YES | 名称 |
| VALUE\_ | varchar(300) | YES |  | 值 |
| REV\_ | int(11) | YES |  | 乐观锁 |

* ACT\_HI\_ACTINST 表用来保存历史节点数据信息，和HI\_TASKINST的区别在于 taskinst 只记录 usertask 内容

表 格 4-17 历史活动实例表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编程 |
| PROC\_DEF\_ID\_ | varchar(64) | NO |  | 流程定义编号 |
| PROC\_INST\_ID\_ | varchar(64) | NO |  | 流程实例编号 |
| EXECUTION\_ID\_ | varchar(64) | NO |  | 执行实例编号 |
| ACT\_ID\_ | varchar(255) | NO |  | 节点编号 |
| TASK\_ID\_ | varchar(64) | YES |  | 任务实例编号 |
| CALL\_PROC  \_INST\_ID\_ | varchar(64) | YES |  | 调用外部流程的流程实例ID |
| ACT\_NAME\_ | varchar(255) | YES |  | 节点名称 |
| ACT\_TYPE\_ | varchar(255) | NO |  | 节点类型 |
| ASSIGNEE\_ | varchar(255) | YES |  | 签收人 |
| START\_TIME\_ | datetime | NO |  | 开始时间 |
| END\_TIME\_ | datetime | YES |  | 结束时间 |
| DURATION\_ | bigint(20) | YES |  | 毫秒值 |
| TENANT\_ID\_ | varchar(255) | YES |  |  |

* ACT\_HI\_ATTACHMENT 表用来保存历史附件信息

表 格 4-18 历史活动附件表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| REV\_ | varchar(64) | YES |  | 乐观锁 |
| USER\_ID\_ | varchar(255) | YES |  | 用户编号 |
| NAME\_ | varchar(255) | YES |  | 名称 |
| DESCRIPTION\_ | varchar(4000) | YES |  | 描 述 |
| TYPE\_ | varchar(255) | YES |  | 类 型 |
| TASK\_ID\_ | varchar(64) | YES |  | 任 务实例  编 号 |
| PROC\_INST\_ID\_ | varchar(64) | YES |  | 流 程实例  编号 |
| URL\_ | varchar(4000) | YES |  | 附 件地址 |
| CONTENT\_ID\_ | varchar(64) | YES |  | 字节表编号 |

* ACT\_HI\_COMMENT表用来保存历史意见信息

表 格 4-19历史意见表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| TYPE\_ | varchar(255) | YES |  | 类型 |
| TIME\_ | datetime | NO |  | 时间 |
| USER\_ID\_ | varchar(255) | YES |  | 用户编号 |
| TASK\_ID\_ | varchar(64) | YES |  | 任务编号 |
| PROC\_INST\_ID\_ | varchar(64) | YES |  | 流程实例  编号 |
| ACTION\_ | varchar(255) | YES |  | 行为类型 |
| MESSAGE\_ | varchar(4000) | YES |  | 基本内容 |
| FULL\_MSG\_ | longblob | YES |  | 全部内容 |

* ACT\_HI\_DETAIL表用来保存历史详情信息，保存历史流程变量

表 格 4-20 历史详情表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| TYPE\_ | varchar(255) | NO |  | 类型 |
| PROC\_INST\_ID\_ | varchar(64) | YES |  | 流程实例  编号 |
| EXECUTION\_ID\_ | varchar(64) | YES |  | 执行实例  编号 |
| TASK\_ID\_ | varchar(64) | YES |  | 任务实例  编号 |
| ACT\_INST\_ID\_ | varchar(64) | YES |  | 节点实例  编号 |
| NAME\_ | varchar(255) | NO |  | 名称 |
| VAR\_TYPE\_ | varchar(255) | YES |  | 参数类型 |
| REV\_ | int(11) | YES |  | 乐观锁 |
| TIME\_ | datetime | NO |  | 时间戳 |
| BYTEARRAY\_ID\_ | varchar(64) | YES |  | 字节表  编号 |
| DOUBLE\_ | double | YES |  | 变量类型  Double |
| LONG\_ | bigint(20) | YES |  | 变量类型  Long |
| TEXT\_ | varchar(4000) | YES |  | 变量类型  String |
| TEXT2\_ | varchar(4000) | YES |  | 变量类型  JPA 持久化 |

* ACT\_HI\_IDENTITYLINK 表用来保存历史流程中参与者人员信息

表 格 4-21 历史参与人员表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| GROUP\_ID\_ | varchar(255) | YES |  | 组编号 |
| TYPE\_ | varchar(255) | YES |  | 类型 |
| USER\_ID\_ | varchar(255) | YES |  | 用户编号 |
| TASK\_ID\_ | varchar(64) | YES |  | 节点实例  编号 |
| PROC\_INST\_ID\_ | varchar(64) | YES |  | 流程实例  编号 |

* ACT\_HI\_PROCINST表用来保存历史流程实例信息

表 格 4-22 历史流程实例表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| PROC\_INST\_ID\_ | varchar(64) | NO |  | 流程实例编号 |
| BUSINESS\_KEY\_ | varchar(255) | YES |  | 业务主键 |
| PROC\_DEF\_ID\_ | varchar(64) | NO |  | 流程定义编号 |
| START\_TIME\_ | datetime | NO |  | 开始时间 |
| END\_TIME\_ | datetime | YES |  | 结束时间 |
| DURATION\_ | bigint(20) | YES |  | 耗时 |
| START\_USER\_ID\_ | varchar(255) | YES |  | 发起人 |
| START\_ACT\_ID\_ | varchar(255) | YES |  | 开始节点编号 |
| END\_ACT\_ID\_ | varchar(255) | YES |  | 结束节点编号 |
| SUPER\_PROCESS\_INSTANCE\_ID\_ | varchar(64) | YES |  | 父流程实例  编号 |
| DELETE\_REASON\_ | varchar(4000) | YES |  | 删除原因 |
| TENANT\_ID\_ | varchar(255) | YES |  |  |

* ACT\_HI\_TASKINST 表用来保存历史任务实例信息

表 格 4-23 历史任务实例表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 主键编号 |
| PROC\_DEF\_ID\_ | varchar(64) | YES |  | 流程定义编号 |
| TASK\_DEF\_KEY\_ | varchar(255) | YES |  | 节点定义  编号 |
| PROC\_INST\_ID\_ | varchar(64) | YES |  | 流程实例编号 |
| EXECUTION\_ID\_ | varchar(64) | YES |  | 执行实例编号 |
| NAME\_ | varchar(255) | YES |  | 名称 |
| PARENT\_TASK\_ID\_ | varchar(64) | YES |  | 父节点实例编号 |
| DESCRIPTION\_ | varchar(4000) | YES |  | 描述 |
| OWNER\_ | varchar(255) | YES |  | 签收人（默认为空，只有委托才有值） |
| ASSIGNEE\_ | varchar(255) | YES |  | 签收或被委托 |
| START\_TIME\_ | datetime | NO |  | 开始时间 |
| CLAIM\_TIME\_ | datetime | YES |  | 提醒时间 |
| END\_TIME\_ | datetime | YES |  | 结束时间 |
| DURATION\_ | bigint(20) | YES |  | 耗时 |
| DELETE\_REASON\_ | varchar(4000) | YES |  | 删除原因 |
| PRIORITY\_ | int(11) | YES |  | 优先级别 |
| DUE\_DATE\_ | datetime | YES |  | 过期时间 |
| FORM\_KEY\_ | varchar(255) | YES |  | 节点定义  formkey |
| CATEGORY\_ | varchar(255) | YES |  | 类型 |
| TENANT\_ID\_ | varchar(255) | YES |  |  |

* ACT\_HI\_VARINST表用来保存历史变量信息

表 格 2-24 历史变量表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| PROC\_INST\_ID\_ | varchar(64) | YES |  | 流程实例编号 |
| EXECUTION\_ID\_ | varchar(64) | YES |  | 执行实例编号 |
| TASK\_ID\_ | varchar(64) | YES |  | 任务实例编号 |
| NAME\_ | varchar(255) | NO |  | 参数名称 |
| VAR\_TYPE\_ | varchar(100) | YES |  | 参数类型 |
| REV\_ | int(11) | YES |  | 乐观锁 |
| BYTEARRAY\_ID\_ | varchar(64) | YES |  | 资源表  编号 |
| DOUBLE\_ | double | YES |  | 数据类型  Double |
| LONG\_ | bigint(20) | YES |  | 数据类型  Long |
| TEXT\_ | varchar(4000) | YES |  | 数据类型  String |
| TEXT2\_ | varchar(4000) | YES |  | 数据类型  JPA |
| CREATE\_TIME\_ | datetime | YES |  | 创建时间 |
| LAST\_UPDATED\_TIME\_ | datetime | YES |  | 最近更新时间 |

* ACT\_ID\_GROUP 表用来保存用户组信息记录

表 格 4-25 用户组信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 主键 |
| REV\_ | int(11) | YES |  | 乐观锁 |
| NAME\_ | varchar(255) | YES |  | 组名称 |
| TYPE\_ | varchar(255) | YES |  | 类型 |

* ACT\_ID\_INFO 表用来把保存用户扩展信息记录

表 格 4-26 用户信息扩展表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 主键 |
| REV\_ | int(11) | YES |  | 乐观锁 |
| USER\_ID\_ | varchar(64) | YES |  | 用户编号 |
| TYPE\_ | varchar(64) | YES |  | 类型 |
| KEY\_ | varchar(255) | YES |  | Key |
| VALUE\_ | varchar(255) | YES |  | 值 |
| PASSWORD\_ | longblob | YES |  | 密码 |
| PARENT\_ID\_ | varchar(255) | YES |  | 父主键 |

* ACT\_ID\_MEMBERSHIP 表用来保存用户与用户组对应信息记录

表 格 4-27 用户与用户组关系表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| USER\_ID\_ | varchar(64) | NO | YES | 用户编号 |
| GROUP\_ID\_ | varchar(64) | NO | YES | 组编号 |

* ACT\_ID\_USER 表用来保存用户信息表

表 格 4-28 用户信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| REV\_ | int(11) | YES |  | 乐观锁 |
| FIRST\_ | varchar(255) | YES |  | 姓 |
| LAST\_ | varchar(255) | YES |  | 名 |
| EMAIL\_ | varchar(255) | YES |  | 邮件地址 |
| PWD\_ | varchar(255) | YES |  | 密码 |
| PICTURE\_ID\_ | varchar(64) | YES |  | 图片编号 |

* ACT\_RE\_DEPLOYMENT 表用来保存部署流程定义时部署信息记录

表 格 4-29 部署信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| NAME\_ | varchar(255) | YES |  | 部署文件 |
| CATEGORY\_ | varchar(255) | YES |  | 分类 |
| TENANT\_ID\_ | varchar(255) | YES |  |  |
| DEPLOY\_TIME\_ | timestamp | NO |  | 部署世间 |

* ACT\_RE\_MODEL 表用通过流程设计器设计流程定义后，保存流程设计模型记录。

表 格 4-30 流程设计模型部署表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| REV\_ | int(11) | YES |  | 乐观锁 |
| NAME\_ | varchar(255) | YES |  | 名称 |
| KEY\_ | varchar(255) | YES |  | KEY |
| CATEGORY\_ | varchar(255) | YES |  | 分类 |
| CREATE\_TIME\_ | timestamp | YES |  | 创建时间 |
| LAST\_UPDATE\_TIME\_ | timestamp | YES |  | 最新修改  世间 |
| VERSION\_ | int(11) | YES |  | 版本 |
| META\_INFO\_ | varchar(4000) | YES |  | json 格式保存流程定义的信息 |
| DEPLOYMENT\_ID\_ | varchar(64) | YES |  | 部署编号 |
| EDITOR\_SOURCE\_  VALUE\_ID\_ | varchar(64) | YES |  |  |
| EDITOR\_SOURCE\_  EXTRA\_VALUE\_ID\_ | varchar(64) | YES |  |  |
| TENANT\_ID\_ | varchar(255) | YES |  |  |

* ACT\_RE\_PROCDEF 表用来保存流程定义数据记录。业务流程数据表，此表和 ACT\_RE\_DEPLOYMENT 是多对一关系。即，一次部署文件中会包含多个流程定义文件，每个流程定义对应 ACT\_RE\_PROCDEF 表内一行记录，每个流程定义的数据，都会对应于 ACT\_GE\_BYTEARRAY 表内的一个资源文件和 PNG 图片文件。

表 格 4-31 流程定义数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| REV\_ | int(11) | YES |  | 乐观锁 |
| CATEGORY\_ | varchar(255) | YES |  | 分类 |
| NAME\_ | varchar(255) | YES |  | 名称 |
| KEY\_ | varchar(255) | NO |  | 定义 key |
| VERSION\_ | int(11) | NO |  | 版本 |
| DEPLOYMENT\_ID\_ | varchar(64) | YES |  | 部署表编号 |
| RESOURCE\_NAME\_ | varchar(4000) | YES |  | BPMN  文件名称 |
| DGRM\_RESOURCE  \_NAME\_ | varchar(4000) | YES |  | png 图片名称 |
| DESCRIPTION\_ | varchar(4000) | YES |  | 描述 |
| HAS\_START\_  FORM\_KEY\_ | varchar(4000) | YES |  | 是否存在开始节点 formkey |
| SUSPENSION\_  STATE\_ | int(11) | YES |  | 是否挂起 |
| TENANT\_ID\_ | varchar(255) | YES |  |  |

* ACT\_RU\_EVENT\_SUBSCR 表用来记录运行时事件订阅

表 格 4-32 运行事件订阅表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| REV\_ | int(11) | YES |  | 乐观锁 |
| EVENT\_TYPE\_ | varchar(255) | NO |  | 事件类型 |
| EVENT\_NAME\_ | varchar(255) | YES |  | 事件名称 |
| EXECUTION\_ID\_ | varchar(64) | YES |  | 执行实例编号 |
| PROC\_INST\_ID\_ | varchar(64) | YES |  | 流程实例  编号 |
| ACTIVITY\_ID\_ | varchar(64) | YES |  | 活动实例  编号 |
| CONFIGURATION\_ | varchar(255) | YES |  | 配置 |
| CREATED\_ | timestamp | NO |  | 是否创建 |
| PROC\_DEF\_ID\_ | varchar(64) | YES |  | 流程定义 |
| TENANT\_ID\_ | varchar(255) | YES |  |  |

* ACT\_RU\_EXECUTION表用来保存运行时流程执行实例表

表 格 4-33运行时流程执行实例表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| REV\_ | int(11) | YES |  | 乐观锁 |
| PROC\_INST\_ID\_ | varchar(64) | YES |  | 流程实例编号 |
| BUSINESS\_KEY\_ | varchar(255) | YES |  | 业务主键编号 |
| PARENT\_ID\_ | varchar(64) | YES |  | 父节点实例编号 |
| PROC\_DEF\_ID\_ | varchar(64) | YES |  | 流程定义编号 |
| SUPER\_EXEC\_ | varchar(64) | YES |  |  |
| ACT\_ID\_ | varchar(255) | YES |  | 节点实例编号 |
| IS\_ACTIVE\_ | tinyint(4) | YES |  | 是否存活 |
| IS\_CONCURRENT\_ | tinyint(4) | YES |  | 是否为并行 |
| IS\_SCOPE\_ | tinyint(4) | YES |  |  |
| IS\_EVENT\_SCOPE\_ | tinyint(4) | YES |  |  |
| SUSPENSION  \_STATE\_ | int(11) | YES |  | 挂起状态  1激活2挂起 |
| CACHED\_ENT\_STATE\_ | int(11) | YES |  |  |
| TENANT\_ID\_ | varchar(255) | YES |  |  |

* ACT\_RU\_IDENTITYLINK 表用来保存运行时流程人员记录，主要存储任务节点与参与者的相关记录信息

表 格 4-34 运行时流程人员表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| REV\_ | int(11) | YES |  | 乐观锁 |
| GROUP\_ID\_ | varchar(255) | YES |  | 组编号 |
| TYPE\_ | varchar(255) | YES |  | 类型 |
| USER\_ID\_ | varchar(255) | YES |  | 用户编号 |
| TASK\_ID\_ | varchar(64) | YES |  | 节点实例编号 |
| PROC\_INST\_ID\_ | varchar(64) | YES |  | 流程实例编号 |
| PROC\_DEF\_ID\_ | varchar(64) | YES |  | 流程定义编号 |

* ACT\_RU\_JOB 表用来保存运行时定时任务记录

表 格 4-35 运行时定时任务数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| REV\_ | int(11) | YES |  | 版本 |
| TYPE\_ | varchar(255) | NO |  | 类型 |
| LOCK\_EXP\_TIME\_ | timestamp | YES |  | 锁释放时间 |
| LOCK\_OWNER\_ | varchar(255) | YES |  | 挂起着 |
| EXCLUSIVE\_ | tinyint(1) | YES |  |  |
| EXECUTION\_ID\_ | varchar(64) | YES |  | 执行实例编号 |
| PROCESS\_  INSTANCE\_ID\_ | varchar(64) | YES |  | 流程实例编号 |
| PROC\_DEF\_ID\_ | varchar(64) | YES |  | 流程定义  编号 |
| RETRIES\_ | int(11) | YES |  |  |
| EXCEPTION\_STACK\_ID\_ | varchar(64) | YES |  | 异常信息  编号 |
| EXCEPTION\_MSG\_ | varchar(4000) | YES |  | 异常信息 |
| DUEDATE\_ | timestamp | YES |  | 到期时间 |
| REPEAT\_ | varchar(255) | YES |  | 重复 |
| HANDLER\_TYPE\_ | varchar(255) | YES |  | 处理类型 |
| HANDLER\_CFG\_ | varchar(4000) | YES |  | 标识 |
| TENANT\_ID\_ | varchar(255) | YES |  |  |

* ACT\_RU\_TASK 表用来保存运行时任务节点记录

表 格 4-36 运行时任务节点表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| REV\_ | int(11) | YES |  | 乐观锁 |
| EXECUTION\_ID\_ | varchar(64) | YES |  | 执行实例编号 |
| PROC\_INST\_ID\_ | varchar(64) | YES |  | 流程实例编号 |
| PROC\_DEF\_ID\_ | varchar(64) | YES |  | 流程定义  编号 |
| NAME\_ | varchar(255) | YES |  | 节点定义名称 |
| PARENT\_TASK\_ID\_ | varchar(64) | YES |  | 父节点实例编号 |
| DESCRIPTION\_ | varchar(4000) | YES |  | 描述 |
| TASK\_DEF\_KEY\_ | varchar(255) | YES |  | 任务定义编号 |
| OWNER\_ | varchar(255) | YES |  | 拥有者（委托有值） |
| ASSIGNEE\_ | varchar(255) | YES |  | 签收人或委托人 |
| DELEGATION\_ | varchar(64) | YES |  | 委托类型 |
| PRIORITY\_ | int(11) | YES |  | 优先级别 |
| CREATE\_TIME\_ | timestamp | NO |  | 创建时间 |
| DUE\_DATE\_ | datetime | YES |  | 过期时间 |
| CATEGORY\_ | varchar(255) | YES |  | 创建时间 |
| SUSPENSION  \_STATE\_ | int(11) | YES |  | 是否挂起 |
| TENANT\_ID\_ | varchar(255) | YES |  |  |

* ACT\_RU\_VARIABLE 表用来保存运行时流程变量数据表

表 格 4-37 运行时流程变量数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 类型 | 是否可空 | 主键 | 描述 |
| ID\_ | varchar(64) | NO | YES | 编号 |
| REV\_ | int(11) | YES |  | 乐观锁 |
| TYPE\_ | varchar(255) | NO |  | 类型 |
| NAME\_ | varchar(255) | NO |  | 名称 |
| EXECUTION\_ID\_ | varchar(64) | YES |  | 执行实例编号 |
| PROC\_INST\_ID\_ | varchar(64) | YES |  | 流程实例编号 |
| TASK\_ID\_ | varchar(64) | YES |  | 节点实例编号 |
| BYTEARRAY\_ID\_ | varchar(64) | YES |  | 字节表编号 |
| DOUBLE\_ | double | YES |  | 变量类型 Double |
| LONG\_ | bigint(20) | YES |  | 变量类型Long |
| TEXT\_ | varchar(4000) | YES |  | 变量类型 String  若JPA ，值为jpa 对象的class |
| TEXT2\_ | varchar(4000) | YES |  | 若 JPA ，值为对象ID |

# 详细设计

## 系统模块

### 用户管理

* 修改用户名密码

注册用户进入系统，进入「个人资料」界面，在注册信息面板上，点击「编辑」，选择「更换密码」，显示该界面，然后根据界面信息提示，输入信息，确认信息无误后，点击「确定」，系统提示数据库修改密码信息。

图 5-1 修改密码事件流

|  |  |
| --- | --- |
| 内容 | 说明 |
| 用例编号 | Sys01 |
| 用例名称 | 修改用户名密码 |
| 用例说明 | 注册用户可以修改自己的密码 |
| 参与者 | 注册用户 |
| 前置条件 | 注册用户被识别，密码通过校验 |
| 后置条件 | 后台数据库保存修改后的新密码 |
| 基本路径 | 1. 进入个人资料界面，在注册信息面板上点击编辑，选择修改密码 2. 输入原密码，新密码，确认密码，单击保存 3. 提示修改密码成功。 |

### 角色权限管理

* 权限设置

管理员登录系统，进入「角色列表」页面，通过点击「权限设置」，该界面显示所有权限列表，权限列表进行勾选，点击「确认」保存权限设置。

图 5-2 权限设置事件流

|  |  |
| --- | --- |
| 内容 | 说明 |
| 用例编号 | Sys02 |
| 用例名称 | 权限设置 |
| 参与者 | 管理员 |
| 前置条件 | 管理员被识别和别授权 |
| 后置条件 | 后台数据库保存权限设置的信息 |
| 基本路径 | 1. 进入「角色列表」页面，点击「权限设置」。 2. 勾选权限进行权限分配，点击「确认」。 3. 提示权限设置成功，保存信息。 |
| 扩展路径 | 1. 设置成功后，拥有该角色的角色，可以操作相关信息 |

* 移除用户

管理员登录系统，在「角色列表」页面，通过点击「分配用户」，该界面显示已经分配的角色用户列表信息，通过点击该页面「移除」可移除指定用户。

图 5-3 移除用户事件流

|  |  |
| --- | --- |
| 内容 | 说明 |
| 用例编号 | Sys03 |
| 用例名称 | 移除用户 |
| 参与者 | 管理员 |
| 前置条件 | 管理员被标识和被授权 |
| 后置条件 | 后台数据库从角色中移除该用户 |
| 基本路径 | 1. 进入「角色列表」页面，通过点击「分配用户」。 2. 显示角色已分配用户列表信息，对指定用户点击「移除」，点击「确认」。 3. 显示移除成功，保存信息。 |
| 扩展路径 | 1. 移除成功后，该用户失去了该角色，没有了角色对应的权限。 |

* 分配用户

管理员登录系统，在「角色列表」页面，通过点击「分配用户」，在弹出的页面上显示已经分配的角色用户列表信息，通过点击「添加人员」按钮可以再添加新的用户

图 5-4分配用户 事件流

|  |  |
| --- | --- |
| 内容 | 说明 |
| 用例编号 | Sys04 |
| 用例名称 | 分配用户 |
| 参与者 | 管理员 |
| 前置条件 | 管理员被标识和被授权 |
| 后置条件 | 后台数据库在角色中关联该用户 |
| 基本路径 | 1. 进入「角色列表」页面，通过点击「分配用户」 2. 进入 「分配用户」页面，点击「添加人员」 3. 进入「选择用户」页面，在待选人员列表中选择要添加的人员到已选人员列表，点击「确定」。 4. 在「分配用户」页面上，可看到添加用户的信息。 |
| 扩展路径 | 添加成功后，该用户获得该角色，拥有了角色对应的权限。 |

### 字典管理

本系统使用字典管理。可以方便的管理数据库中表中某些字段的类别，比如 bug 表中 bug 状态字段、bug 类别字段、bug 优先级别字段可以在字典表中找到对应的类别标识，便于统一管理，这是对于数据库层面来说，对于 Java 编程的方式来说，所有的数据字典表在设计上，可以通过统一的方法，来获取每一个类别相关数据。

* 根据类型和值获取字典标签（列表取值）
  + DictUtils.getDictLabel(String value, String type, String defaultValue)
  + ${ fns:getDictLabel (value, type, defaultValue)}
* 根据类型和标签获取字典值（根据标签取值）
  + DictUtils.getDictValue(String label, String type, String defaultLabel)
  + ${fns:getDictValue(label, type, defaultValue)}
* 根据类型获取字典列表（下拉框，复选框，单选框）
  + DictUtils.getDictList(String type)
  + ${fns: getDictList(type)}

## Bug 模块

### bug 管理

* 新建问题

测试人员登录系统，再左侧导航栏中，选择「新建问题」，进入该界面，在新建问题界面输入 bug 详情，当测试人员确认发表时，系统提示数据库保存bug 信息。

图 5-5 新建问题事件流

|  |  |
| --- | --- |
| 内容 | 说明 |
| 用例编号 | Bug01 |
| 用例名称 | 新建问题 |
| 参与者 | 测试人员 |
| 前置条件 | 测试人员被标识和被授权 |
| 后置条件 | 后台数据库保存bug 所有相关信息 |
| 基本路径 | 1. 点击「新建问题」。 2. 输入 bug 详情，单击提交。 3. 显示新建成功，保存信息。 |
| 扩展路径 | 1. 提交成功后，bug 会被流转。 |

* 我的任务

得到分配的任务（bug）用户登录系统，再左侧导航栏中，选择「我的任务」，进入该界面，选择一条任务记录，点击「处理」，浏览任务，发表自己的意见，当用户确认提交时，系统提示数据库保存bug 处理信息。

图 5-5我的任务 事件流

|  |  |
| --- | --- |
| 内容 | 说明 |
| 用例编号 | Bug02 |
| 用例名称 | 我的任务 |
| 参与者 | 被分配的用户 |
| 前置条件 | 用户被识别和被授权 |
| 后置条件 | 后台数据库保存处理bug 相关信息 |
| 基本路径 | 1. 点击「我的任务」。 2. 浏览bug，输入 bug 意见，单击提交。 3. 显示处理成功，保存信息。 |
| 扩展路径 | 1. 提交成功后，bug 会被流转。 |

### 项目管理

* 项目添加成员

项目经理登录系统，在「项目列表」页面，通过点击「项目组成员」，在弹出的页面上显示已经项目下成员列表信息，通过点击「添加成员」按钮可以再添加新的用户。

图 5-6 项目添加成员ß 事件流

|  |  |
| --- | --- |
| 内容 | 说明 |
| 用例编号 | Bug03 |
| 用例名称 | 添加成员 |
| 参与者 | 项目经理 |
| 前置条件 | 项目经理被标识和被授权 |
| 后置条件 | 后台数据库在项目中关联该用户 |
| 基本路径 | 1. 进入「项目列表」页面，通过点击「项目组成员」 2. 进入 「项目组成员」页面，点击「添加人员」 3. 进入「选择人员」页面，在待选人员列表中选择要添加的人员到已选人员列表，点击「确定」。 4. 在「添加成员」页面上，可看到添加用户的信息。 |
| 扩展路径 | 1. 添加成功后，该用户参与了该项目，可以查看项目信息。 2. 可以对该项目进行 bug 管理。 |

## 工作流模块

### 模型管理

* 新建模型

管理员登录系统，进入「模型列表」页面，通过点击「新建模型」，显示该界面，根据界面提示信息，输入模型信息，点击「提交」保存模型。接着页面跳转到在线流程设计器界面，进行自定义流程设计。

图 5-7 新建模型 事件流

|  |  |
| --- | --- |
| 内容 | 说明 |
| 用例编号 | Act01 |
| 用例名称 | 新建模型 |
| 参与者 | 管理员 |
| 前置条件 | 管理员被识别和被授权 |
| 后置条件 | 后台数据库保存模型相关信息 |
| 基本路径 | 1. 进入「模型列表」页面，点击「新建模型」 2. 输入 bug 信息后，点击提交 3. 保存成功，数据库保存信息。 |
| 扩展路径 | 1. 提交成功后，跳转到流程设计器页面，可设计自定义流程。 |

### 流程管理

* 文件部署流程

管理员登录系统，进入「流程部署」页面，根据界面提示信息，选择部署文件，点击，系统提示转换成功。

图 5-8文件部署流程事件流

|  |  |
| --- | --- |
| 内容 | 说明 |
| 用例编号 | Act02 |
| 用例名称 | 部署流程 |
| 参与者 | 管理员 |
| 前置条件 | 管理员被标识或被授权 |
| 后置条件 | 后台数据库记录转换相关信息 |
| 基本路径 | 1. 进入「流程部署」页面 2. 上传文件，输入信息。点击提交。 3. 保存成功，数据库保存信息。 |

## 通信模块

### 站内通知

* 发送通知

用户登录系统，进入「通知列表」页面，通过点击「添加」，根据界面提示信息，填写信息并选择被通知成员，点击「确认」，系统提示数据库保存通知信息。

图 5-9 发送通知 事件流

|  |  |
| --- | --- |
| 内容 | 说明 |
| 用例编号 | Msg01 |
| 用例名称 | 发送通知 |
| 参与者 | 用户 |
| 前置条件 | 用户标识和被授权 |
| 后置条件 | 后台数据库保存通知信息。 |
| 基本路径 | 1. 进入「通知列表」页面，点击「添加」 2. 输入通知信息，选择被通知人后，点击提交 3. 保存成功，数据库保存信息。 |
| 扩展路径 | 1. 被通知用户查看通知相关信息。 |

### 简单实时聊天

* 聊天

平台用户登录系统，便可看到 聊天功能界面。点击用户头像，在对话框中输入聊天信息。

图 5-10移除用户 事件流

|  |  |
| --- | --- |
| 内容 | 说明 |
| 用例编号 | Msg02 |
| 用例名称 | 聊天 |
| 参与者 | 注册用户 |
| 前置条件 | 用户被标识 |
| 后置条件 |  |
| 基本路径 | 1. 登录系统 2. 点击聊天界面，点击要聊天用户头像 3. 输入聊天信息，进行简单通讯。 |

# 系统实现

## 系统模块

### 用户管理

* 用户登录

用户登录通过 Shiro 进行管理，未登录的用户若要进行有效资源的访问，会跳转到登录的页面；登录的用户只能访问自己有权限的资源，没有权限的资源，用户无法访问，会报异常。并且密码经过 SHA-1 进行加密。



图 6-1登录页面

### 角色权限管理

* 系统设置->角色管理->权限设置

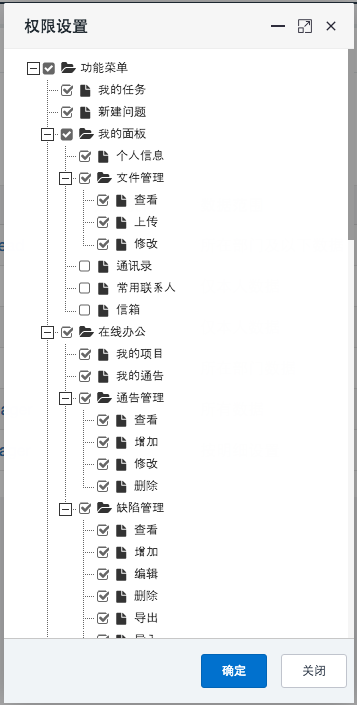


图 6-2 权限设置

* 系统设置->角色管理->分配用户，之后可以移除，也可以添加成员。

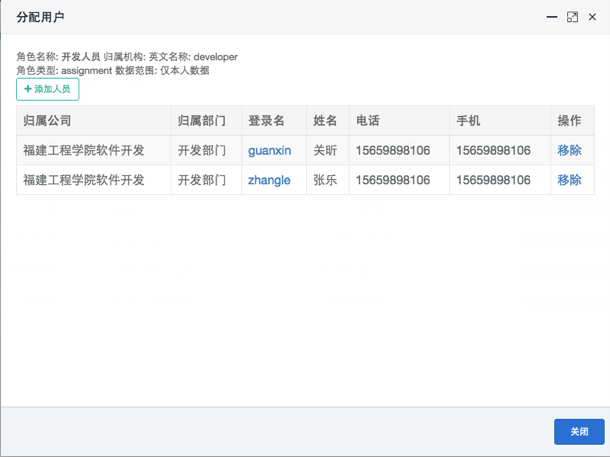


图 6-3 角色分配用户

### 字典管理

* 系统设置->字典管理->添加



图 6-4 数据字典添加功能

### 文件管理

* 系统管理->文件管理



图 6-5 文件夹管理

## Bug模块

### bug 管理

* 新建问题

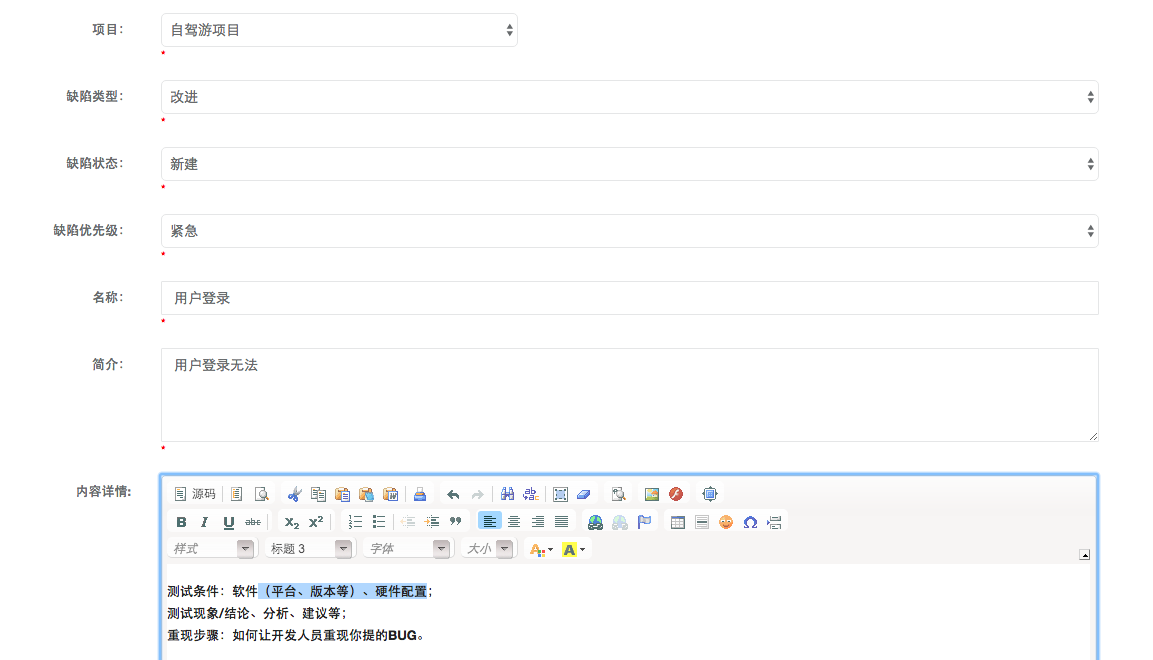


图 6-6 新建问题

### 项目管理

* 参与项目详情页

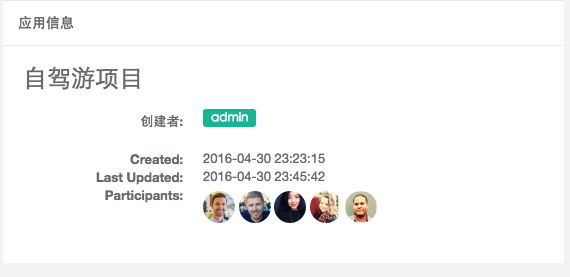


图 6-7项目信息

* 项目统计

本系统所有项目统计都是通过 Chart.js 来实现。前端页面通过发送 Ajax 异步请求，根据不同的需求编写不同的SQL查询语句，并结合 Chart.js 中图表类型的数据结构，将返回的结果封装成的 json 数据，返回给 Chart.js 的渲染图表的初始化构造函数中。

统计项目所有 bug 状态分布和bug 趋势，项目成员在导航栏上选择自己参与的项目，在统计部分，通过 Chart.js 图表显示 bug 状态分布。

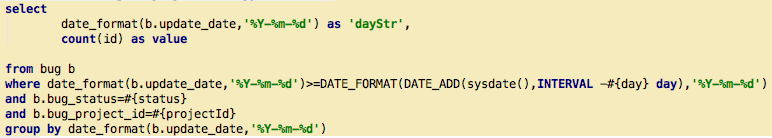


图 6-8统计 指定项目的bug 指定状态一段时间内每天数量趋势 SQL

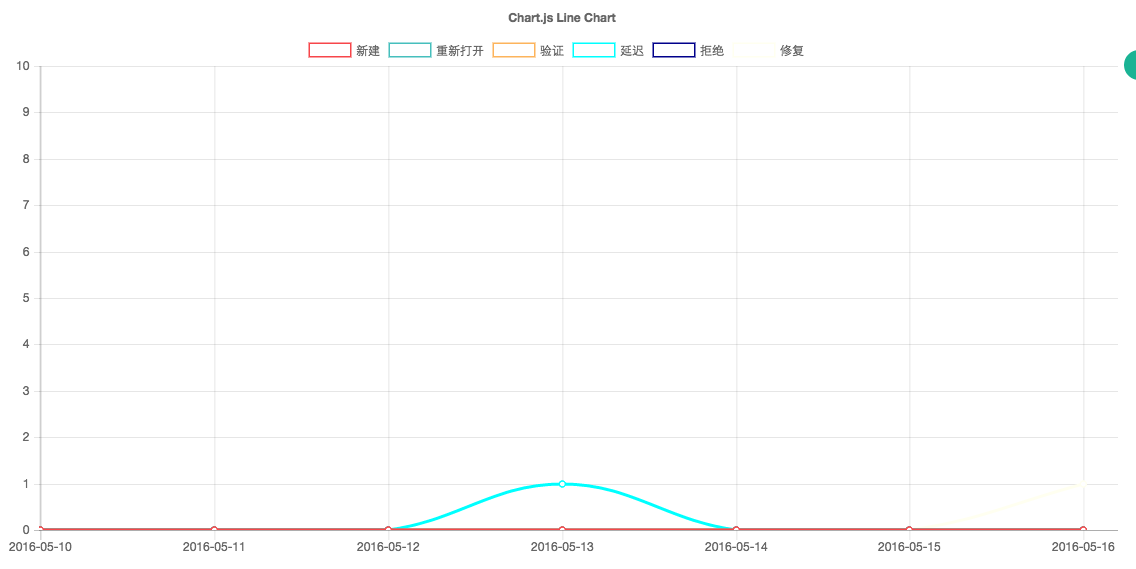


图6-9 bug 状态趋势图

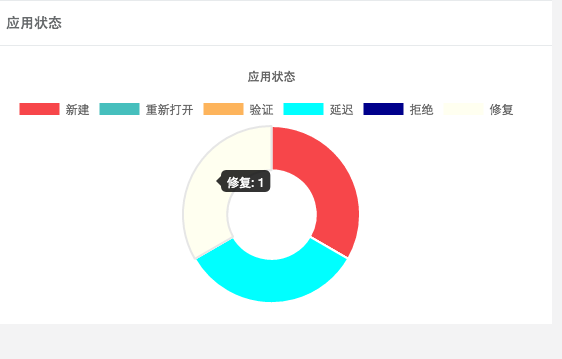


图 6-10 bug 状态分布图

## 工作流模块

### 模型管理

* 新建模型

新建模型，点击「提交」后，页面会跳转到在线设计流程定义页面，设计完毕后，点击「保存」图标进行保存。到模型管理列表，点击「部署」，对其进行部署。



图 6-11 添加模型

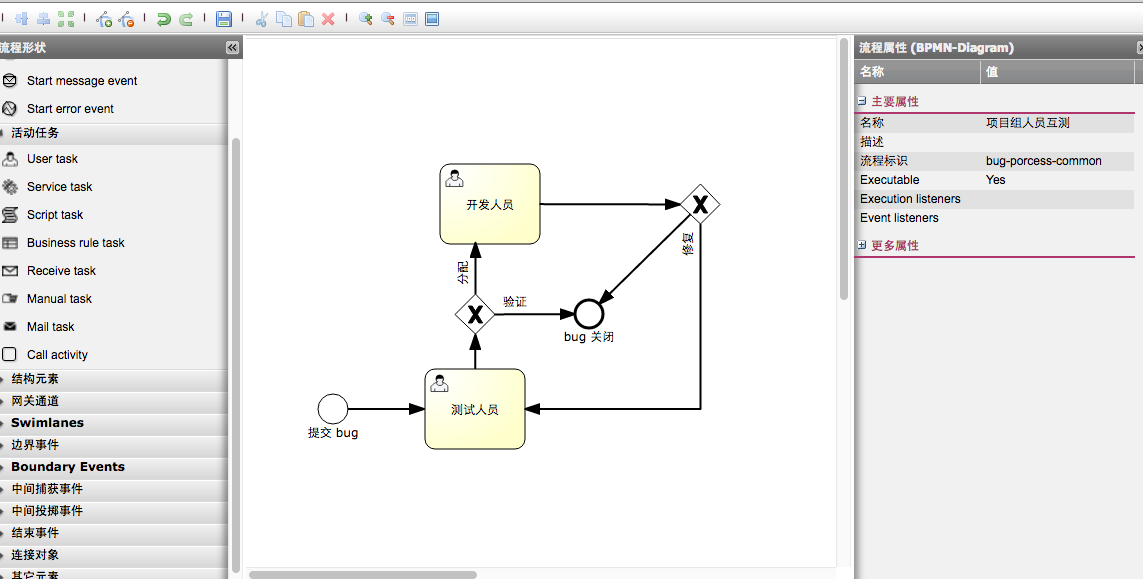


图 6-12 在线工作流设计器

### 流程管理

* 部署流程



图6-13上传文件部署流程

## 通讯模块

### 站内通知

向指定人员发送通知



图 6 -14发送通知

接受人登录后可以看到，任务提醒，点击相关通知记录，可查阅通知内容，并下载相关附件。



图6-15 接收到通知

### 简单聊天

在 Bug 的解决过程中，可能会出现 一些 无法解释清楚的bug，为了能够沟通清楚，这样就可以通过这个简单的即时聊天功能进行沟通。



图6-16 简单聊天

# 总结与展望

## 系统介绍

本 Bug 管理系统基于了多个优秀的开源项目和开源框架，吸取它们的经验，对他们进行整合封装而成的简单实用的 Bug 管理系统。

本系统是以 Spring Framework 为核心容器，Spring MVC 为展示层模型视图控制器，Hibernate 及 Mybatis为数据校验和数据访问层，Apache Shiro 为权限授权层，Activiti 为工作流引擎。系统主要包括基本系统模块、bug 模块、工作流模块和简单通信模块。前端界面采用简洁、大气、美观的响应式框架 Bootstrap 中主题 INSPINIA(Responsive Admin Theme) 。采用分层设计，双重校验、密码加密、访问验证，使用 Maven 做项目管理，Git 作为代码管理工具。

本Bug 管理系统主要用于在互联网公司中确保项目开发中软件的质量，并及时的反馈和统计项目中遇到的 bug。采用现在流行的工作流引擎 Activiti框架技术，并集成在线流程设计器。实现 bug的流转流程，和在线部署和优化 bug 流转流程。

该系统相比较于其他系统的优势

* 设计上采用模块设计，分层设计（展示层、业务逻辑层、数据访问层），高内聚低耦合，结构清晰明了。
* 采用 Activiti 工作流技术，实现复杂的 bug 流转流程，并提供在线修改 bug 流转流程的功能。
* 利用 Chart.js 生成形象丰富的图表记录，直观的展现项目的bug 状态分布 和走势。
* 采用简单、大气、美观的响应式前端框架 Bootstrap ，PC、手机、平板都可进行适配。
* 利用了 layim 插件，一款 webim 聊天插件 ，实现简单的实时通讯，可以实时和成员进行沟通。

## 系统存在不足

由于自己的开发经验，开发水平有限，又有加之设计与实现该系统的时间有限，Activiti 这个强大工作流框架的功能并没有完全而又灵活的应用到系统中来。在用户提交缺陷的这个页面上，bug 图片的上传，标记需要用户自行标记，带来诸多不便。

## 改进方案

针对不足，有两种方案，可以减少用户不必要的操作。

* 方案1

可以开发一个浏览器插件集成于本系统中，登录后，可以进行截图，标注，并直接上传至个人用户文件夹中。

* 方案2

现在移动设备非常流行，几乎人手一台，可以在 bug 提交页面上附上 该bug 二维码。

手机扫描二维码，进入系统的接口，登录后，可以进行拍照上传，从本地相册中上传图片到该bug 页面。

经过资料调查，这两中方案技术技术上都可行。

# 致谢

经过3 个多月的毕业设计终于就要结束了，经过这段时间学习，分析，设计，实现，最终基本实现了预期的功能。在这里，要感谢陈义老师对我提供的耐心指导、及时的帮助，让我能够顺利完成毕业设计。

大学四年的生活就要结束了，在这个自由的地方，不仅学到了很多专业知识，锻炼了独立思考，解决问题的能力，也明白了团队合作的重要性，更重要的是，我在大学里面找到了自己喜欢做的事情。感谢大学的这段时光，感谢学校为我们提供的学习环境和学习资源，感谢每一位老师的耐心教导，让我成为一名合格的学生。