

gITmining软件体系结构描述文档

V1.0



2015-12-23

|  |  |  |  |
| --- | --- | --- | --- |
| 变更历史 | | | |
| 版本号 | **更新时间** | **更新人** | **更新摘要** |
| V1.0 | 2016/2/28 | 董本超 | 完成体系结构描述文档概要 |
|  |  |  |  |

目录

[1引言 3](#_Toc438668479)

[1.1编制目的 3](#_Toc438668480)

[1.2词汇表 3](#_Toc438668481)

[1.3参考资料 3](#_Toc438668482)

[2产品概述 3](#_Toc438668483)

[3逻辑视角 3](#_Toc438668484)

[4组合视角 6](#_Toc438668485)

[4.1开发包图 6](#_Toc438668486)

[4.2运行时进程 9](#_Toc438668487)

[4.3物理部署 9](#_Toc438668488)

[5接口视角 10](#_Toc438668489)

[5.1模块的职责 10](#_Toc438668490)

[5.2用户界面层的分解 15](#_Toc438668491)

[5.2.1用户界面层模块的职责 16](#_Toc438668492)

[5.2.2用户界面层模块的接口规范 17](#_Toc438668493)

[5.2.3用户界面模块设计原理 17](#_Toc438668494)

[5.3业务逻辑层的分解 17](#_Toc438668495)

[5.3.1业务逻辑层模块的职责 18](#_Toc438668496)

[5.3.2业务逻辑层模块的接口规范 18](#_Toc438668497)

[5.4数据层的分解 31](#_Toc438668498)

[5.4.1数据层模块的职责 35](#_Toc438668499)

[5.4.2数据层模块的接口规范 35](#_Toc438668500)

[6信息视角 40](#_Toc438668501)

[6.1数据持久化对象 40](#_Toc438668502)

[6.2数据库表 43](#_Toc438668503)

# 1引言

## 1.1编制目的

本报告详细完成对GitMining系统的概要设计，达到指导详细设计和开发的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

## 1.2词汇表

|  |  |  |
| --- | --- | --- |
| **词汇名称** | **词汇含义** | **备注** |
| GitMining | 代码挖掘 | 挖掘并分析github网站上的用户和仓库数据 |

## 1.3参考资料

1.软件工程与计算(卷三) 团队与软件开发实践, 机械工业出版社, 2012;

2.GitMining迭代一用例文档;

3.GitMining迭代一需求规格说明文档;

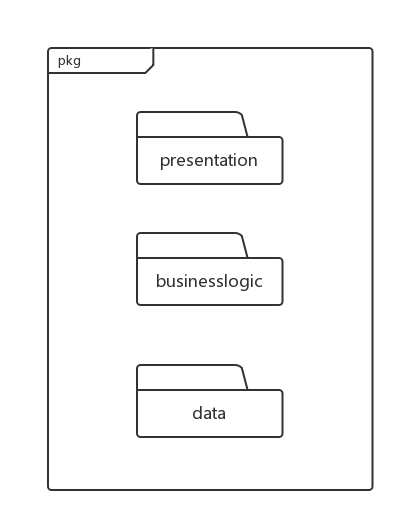
# 2产品概述

GitMining系统是获取并分析github网站上公开的用户信息和仓库信息并分析其相互之间关系、从而进行数据统计和归纳的系统。

该系统在后期迭代二或三中也将根据用户的兴趣进行分类数据检索和推荐，从而提供系统的实用性。

# 3逻辑视角

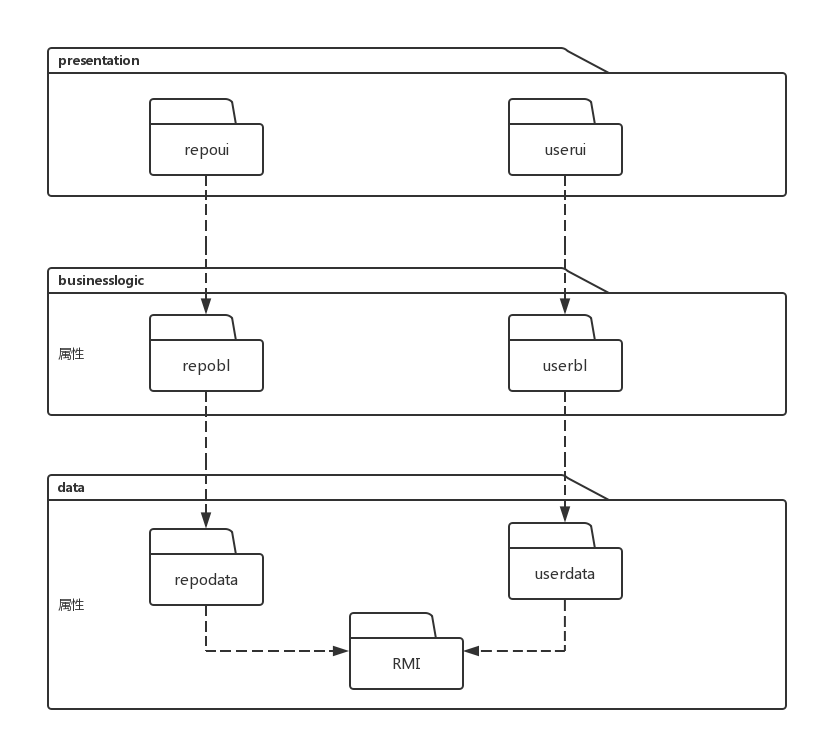
在迭代一阶段，系统暂且不需要服务器端。客户端将选择分层体系结构，将系统分为3层（展示层、业务逻辑层、数据层）。展示层包含GUI的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据访问(注意: 迭代一阶段数据层不存储数据，只通过已有的api进行数据获取)。分层体系结构的逻辑视角如图1所示。

图1 参考体系结构风格的包图表达逻辑视角

GitMining系统在迭代一阶段的主要功能包括GitHub仓库信息查询和用户信息查询，它们对应的逻辑包如下：

|  |  |
| --- | --- |
| **功能** | **对应逻辑包** |
| 仓库信息查询 | repoui, repobl, repodata |
| 用户信息查询 | userui, userbl, userdata |

由上表信息可以生成软件体系结构逻辑设计方案，如图2所示。

图2 软件体系结构逻辑设计方案

# 4组合视角

## 4.1开发包图

GitMining迭代一阶段的开发包设计如表1所示。

**表1 GitMining开发包设计**

|  |  |
| --- | --- |
| **开发（物理）包** | **依赖的其他开发包** |
| repoui | repoblservice |
| userui | userblservice |
| repoblservice | vo |
| userblservice | vo |
| repobl | repoblservice，repodataservice |
| userbl | userblservice， userdataservice |
| repodataservice | repodata |
| userdataservice | userdata |
| repodata |  |
| userdata |  |
| vo |  |

Gitmining系统的开发包图如图3所示

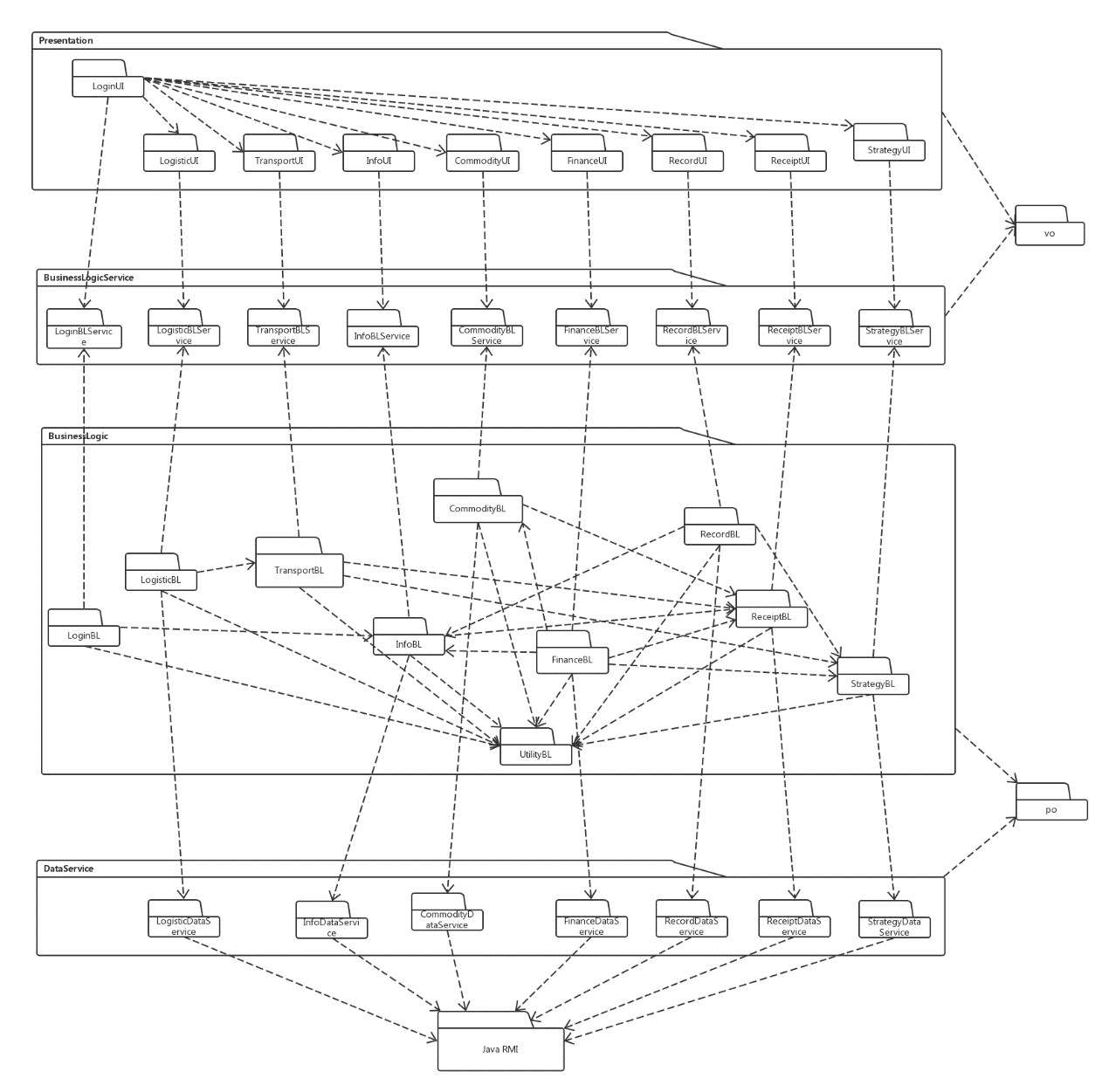
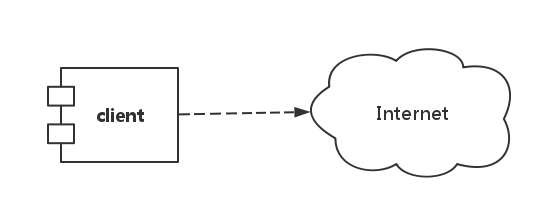


图3 GitMining系统开发包图

## 4.2运行时进程

在GitMining系统运行过程中，系统需要通过api从互联网获取信息，因此系统运行时必须联网，进程图如图4.

图4 进程图

# 5接口视角

## 5.1模块的职责

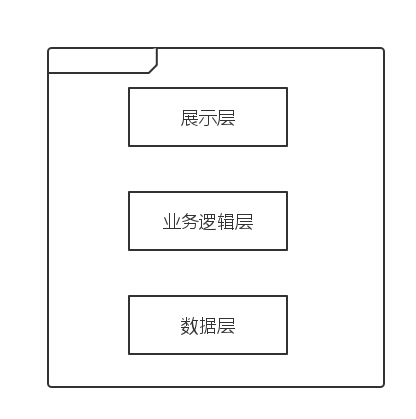
 GitMining系统的模块视图如图5所示。各层的职责如表2所示。

图5 客户端模块视图

**表2 GitMining系统各层的职责**

|  |  |
| --- | --- |
| **层** | **职责** |
| 展示层 | 将仓库或用户信息展示给系统的用户 |
| 业务逻辑层 | 对于用户界面的输入进行响应并进行业务逻辑处理，访问数据层 |
| 数据层 | 利用已有的api联网获取数据并传给逻辑层 |

每一层只是使用下方直接接触的层。 层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如表4所示。

**表3 层之间调用的接口**

|  |  |  |
| --- | --- | --- |
| **接口** | **服务调用方** | **服务提供方** |
| repoblservice  userblservice | 展示层 | 业务逻辑层 |
| repodataservice  userdataservice | 业务逻辑层 | 数据层 |

## 5.2用户界面层的分解

根据需求，系统存在？？？个用户界面：？？？。界面跳转如图6所示。

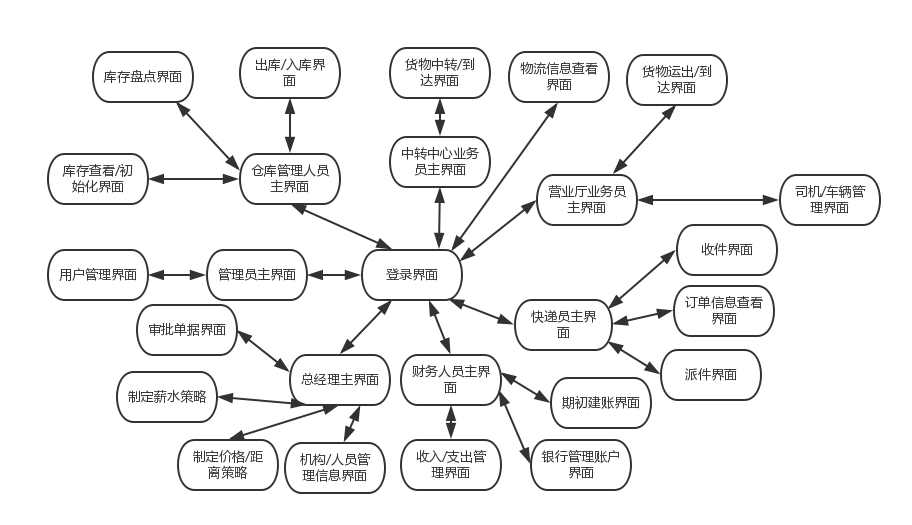


图6 用户界面跳转

## 5.3业务逻辑层的分解

业务逻辑层包括多个针对界面的业务逻辑处理对象。例如，repobl负责处理用户请求查询仓库信息的业务逻辑。具体职责说明如下。

### 5.3.1业务逻辑层模块的职责

业务逻辑层模块的职责如表4所示。

**表4 业务逻辑层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| repobl | 负责处理用户查询仓库信息的业务逻辑 |
| userbl | 负责处理用户查询github用户信息的业务逻辑 |

### 5.3.2业务逻辑层模块的接口规范

**表5 repobl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| repobl.getSearchResult(String keyword) | 语法 | public List<RepoBriefVO> getSearchResult(String keyword) |
| 前置条件 | 系统已正常打开 |
| 后置条件 | 根据搜索关键字返回搜索结果 |
| repobl.getRepoBasicInfo(String [] owner\_repoName) | 语法 | public RepoVO getRepoBasicInfo(String owner\_repoName) |
| 前置条件 | 已返回仓库列表 |
| 后置条件 | 返回单个仓库的基本信息 |
| repobl.getRepoOwner (String [] owner\_repoName\_ownerName) | 语法 | public OwnerVO getRepoOwner (String [] owner\_repoName\_ownerName) |
| 前置条件 | 用户正在查看仓库具体信息 |
| 后置条件 | 返回仓库所有者信息 |
| repobl.getRepoContributor(String [] owner\_repoName\_contriName) | 语法 | public List<ContributorVO> getRepoContributor(String [] owner\_repoName\_contriName) |
| 前置条件 | 用户正在查看仓库具体信息 |
| 后置条件 | 返回仓库贡献者信息 |
| repobl.getRepoCollaborator(String [] owner\_repoName\_collaName) | 语法 | public List<CollaboratorVO> getRepoCollaborator(String [] owner\_repoName\_contriName) |
| 前置条件 | 用户正在查看仓库具体信息 |
| 后置条件 | 返回仓库合作者信息 |
| repobl.getRepoBranch(String [] owner\_repoName) | 语法 | public List<BranchVO> getRepoBranch(String [] owner\_repoName) |
| 前置条件 | 用户正在查看仓库具体信息 |
| 后置条件 | 返回仓库版本信息 |
| repobl.getRepoFork(String [] owner\_repoName) | 语法 | public List<RepoBriefVO> getRepoFork(String [] owner\_repoName) |
| 前置条件 | 用户正在查看仓库具体信息 |
| 后置条件 | 返回仓库fork的其他项目信息 |
| repobl.getRepoCommit(String [] owner\_repoName) | 语法 | public List<CommitVO> getRepoCommit(String [] owner\_repoName) |
| 前置条件 | 用户正在查看仓库具体信息 |
| 后置条件 | 返回仓库commit信息 |
| repobl.getRepoIssue(String [] owner\_repoName) | 语法 | public List<IssueVO> getRepoIssue(String [] owner\_repoName) |
| 前置条件 | 用户正在查看仓库具体信息 |
| 后置条件 | 返回仓库issue信息列表 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| repodataservice.getSearchResult(String keyword) | 获取搜索结果列表 | |
| repodataservice.getRepoBasicInfo(String [] owner\_repoName) | 获取仓库基本信息 | |
| repodataservice.getRepoOwner (String [] owner\_repoName\_ownerName) | 获取仓库所有者信息 | |
| repodataservice.getRepoContributor(String [] owner\_repoName\_contriName) | 获取仓库贡献者信息 | |
| repodataservice.getRepoCollaborator(String [] owner\_repoName\_collaName) | 获取仓库合作者信息 | |
| repodataservice.getRepoBranch(String [] owner\_repoName) | 获取仓库版本信息 | |
| repodataservice.getRepoFork(String [] owner\_repoName) | 获取仓库被引用信息 | |
| repodataservice.getRepoCommit(String [] owner\_repoName) | 获取仓库提交信息 | |
| repodataservice.getRepoIssue(String [] owner\_repoName) | 获取仓库事务信息 | |

**表6 userbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| userbl.getSearchResult(String keyword) | 语法 | public List<UserBriefVO> getSearchResult(String keyword) |
| 前置条件 | 用户已正常进入该系统 |
| 后置条件 | 返回信息检索结果列表 |
| userbl.getUserInfo(String userName) | 语法 | public List<UserVO> getUserInfo (String userName) |
| 前置条件 | 已获取信息检索结果列表 |
| 后置条件 | 返回某用户的具体信息 |
| userbl.getUserRepos(String userName) | 语法 | public UserRelatedReposVO getUserRepos (String userName) |
| 前置条件 | 用户正在查看某github的user的详细信息 |
| 后置条件 | 返回某用户的相关项目信息，包括该用户拥有的、订阅的、合作的、贡献的项目 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| userdataservice.getSearchResult(String keyword) | 获取用户信息检索家俄国列表 | |
| userdataservice.getUserInfo(String userName) | 获取单个用户的相信信息 | |
| userdataservice.getUserRepos(String userName) | 获取单个用户相关的仓库列表 | |

## 5.4数据层的分解

由于该系统的迭代一阶段不涉及数据的本地存储，所有数据均直接从互联网上获得并及时解析，所以数据层暂且功能很少，只有从互联网获取数据并进行初步解析的功能。未来在迭代二或三阶段数据层将增加更多功能点。

### 5.4.1数据层模块的职责

由于各个功能模块对应数据层模块的职责基本相同，故这里只用一张表总结描述，如表18所示。

**表7 数据层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| repodataservice | 为repobl模块提供数据获取服务 |
| userdataservice | 为userbl模块提供数据获取服务 |

### 5.4.2数据层模块的接口规范

**表8 repodata数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| repodataservice.getSearchResult(String keyword) | 语法 | public List<RepoBriefVO> getSearchResult(String keyword) |
| 前置条件 | 无 |
| 后置条件 | 根据搜索关键字返回搜索结果 |
| repodataservice.getRepoBasicInfo(String [] owner\_repoName) | 语法 | public RepoVO getRepoBasicInfo(String owner\_repoName) |
| 前置条件 | 无 |
| 后置条件 | 返回单个仓库的基本信息 |
| repodataservice.getRepoOwner (String [] owner\_repoName\_ownerName) | 语法 | public OwnerVO getRepoOwner (String [] owner\_repoName\_ownerName) |
| 前置条件 | 无 |
| 后置条件 | 返回仓库所有者信息 |
| repodataservice.getRepoContributor(String [] owner\_repoName\_contriName) | 语法 | public List<ContributorVO> getRepoContributor(String [] owner\_repoName\_contriName) |
| 前置条件 | 无 |
| 后置条件 | 返回仓库贡献者信息 |
| repodataservice.getRepoCollaborator(String [] owner\_repoName\_collaName) | 语法 | public List<CollaboratorVO> getRepoCollaborator(String [] owner\_repoName\_contriName) |
| 前置条件 | 无 |
| 后置条件 | 返回仓库合作者信息 |
| repodataservice.getRepoBranch(String [] owner\_repoName) | 语法 | public List<BranchVO> getRepoBranch(String [] owner\_repoName) |
| 前置条件 | 无 |
| 后置条件 | 返回仓库版本信息 |
| repodataservice.getRepoFork(String [] owner\_repoName) | 语法 | public List<RepoBriefVO> getRepoFork(String [] owner\_repoName) |
| 前置条件 | 无 |
| 后置条件 | 返回仓库fork的其他项目信息 |
| repodataservice.getRepoCommit(String [] owner\_repoName) | 语法 | public List<CommitVO> getRepoCommit(String [] owner\_repoName) |
| 前置条件 | 无 |
| 后置条件 | 返回仓库commit信息 |
| repodataservice.getRepoIssue(String [] owner\_repoName) | 语法 | public List<IssueVO> getRepoIssue(String [] owner\_repoName) |
| 前置条件 | 无 |
| 后置条件 | 返回仓库issue信息列表 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| 公共接口[www.gitmining.net/api](http://www.gitmining.net/api)提供的数据服务 | 获取github相关repository和user信息 | |

**表9 userdata数据层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| userdataservice.getSearchResult(String keyword) | 语法 | public List<UserBriefVO> getSearchResult(String keyword) |
| 前置条件 | 无 |
| 后置条件 | 返回信息检索结果列表 |
| userdataservice.getUserInfo(String userName) | 语法 | public List<UserVO> getUserInfo (String userName) |
| 前置条件 | 无 |
| 后置条件 | 返回某用户的具体信息 |
| userdataservice.getUserRepos(String userName) | 语法 | public UserRelatedReposVO getUserRepos (String userName) |
| 前置条件 | 无 |
| 后置条件 | 返回某用户的相关项目信息，包括该用户拥有的、订阅的、合作的、贡献的项目 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| 公共接口[www.gitmining.net/api](http://www.gitmining.net/api)提供的数据服务 | 获取github相关repository和user信息 | |

# 6信息视角

迭代一尚不涉及数据持久化，所有数据均实时获取、解析和输出。