

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 |
| V1.1 | 2016/3/11 | 董本超 | 对设计文档进行修改和补充 |
| V2.0 | 2016/3/16 | 戴新颜 | 完成体系结构描述文档概要（Ⅱ） |

目录

[1引言 4](#_Toc445475635)

[1.1编制目的 4](#_Toc445475636)

[1.2词汇表 4](#_Toc445475637)

[1.3参考资料 4](#_Toc445475638)

[2产品概述 4](#_Toc445475639)

[3逻辑视角 5](#_Toc445475640)

[4组合视角 7](#_Toc445475641)

[4.1开发包图 7](#_Toc445475642)

[4.2运行时进程 8](#_Toc445475643)

[5接口视角 9](#_Toc445475644)

[5.1模块的职责 9](#_Toc445475645)

[5.2用户界面层的分解 10](#_Toc445475646)

[5.3业务逻辑层的分解 11](#_Toc445475647)

[5.3.1业务逻辑层模块的职责 11](#_Toc445475648)

[5.3.2业务逻辑层模块的接口规范 12](#_Toc445475649)

[5.4数据层的分解 19](#_Toc445475650)

[5.4.1数据层模块的职责 19](#_Toc445475651)

[5.4.2数据层模块的接口规范 19](#_Toc445475652)

[6信息视角 24](#_Toc445475653)

# 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 |
| repo | repository | 仓库 |

## 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迭代二用例文档;

3.Gitmining迭代二需求规格说明文档;

# 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。

目前迭代二提供基本数据的统计，方便用户直观分析。

该系统在后期迭代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

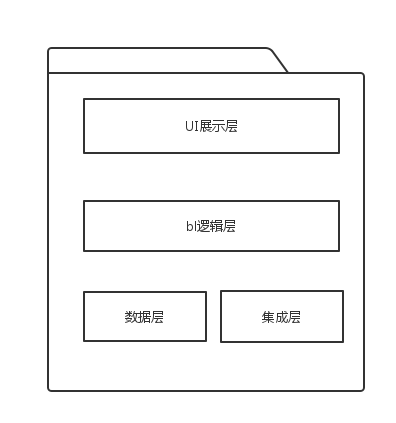
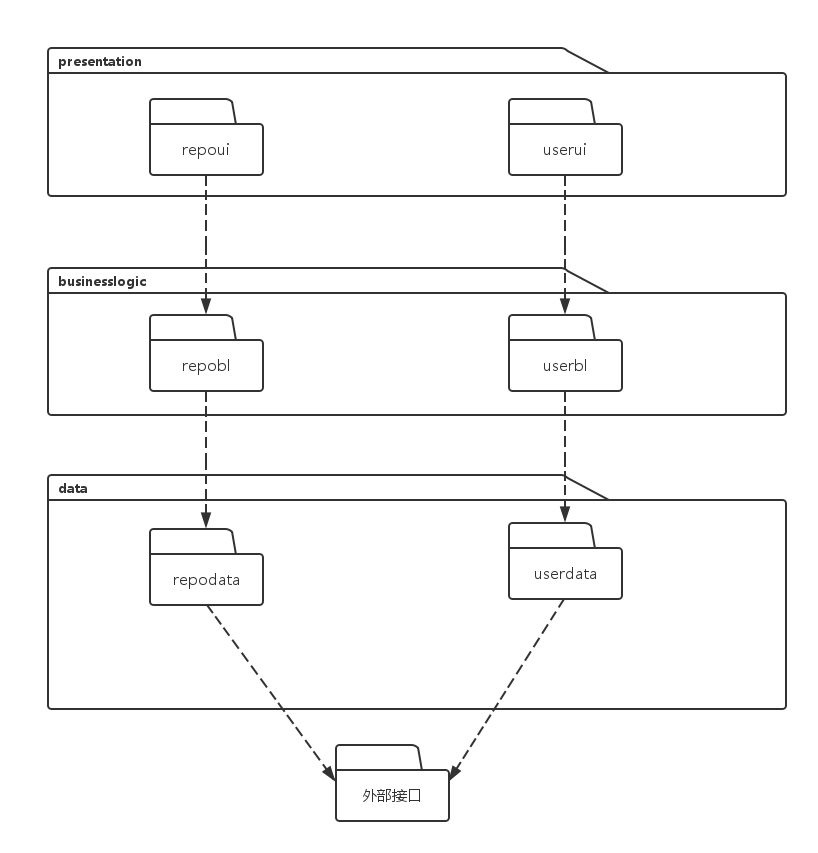
 在迭代二阶段，系统开始提供服务器端。客户端将选择分层体系结构，将系统分为3层（展示层、业务逻辑层、数据层）。展示层包含GUI的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据访问(注意: 迭代二阶段数据层存储部分数据，部分数据通过已有的api进行数据获取)。分层体系结构的逻辑视角如图1所示。

图1 参考体系结构风格的包图表达逻辑视角

Gitmining系统在迭代二阶段的主要功能包括迭代一拥有的GitHub仓库信息查询和用户信息查询，以及迭代二新增的仓库的统计信息与用户相关的统计信息，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 |
| 仓库信息统计 | repostatui,repostatbl,repostatdata |
| 用户信息统计 | userstatui,userstatbl,userstatdata |

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, vo |
| userbl | userblservice， userdataservice, vo |
| repodataservice | repodata, vo |
| userdataservice | userdata, vo |
| repodata | po |
| userdata | po |
| vo |  |
| po |  |

Gitmining系统的开发包图如图3所示

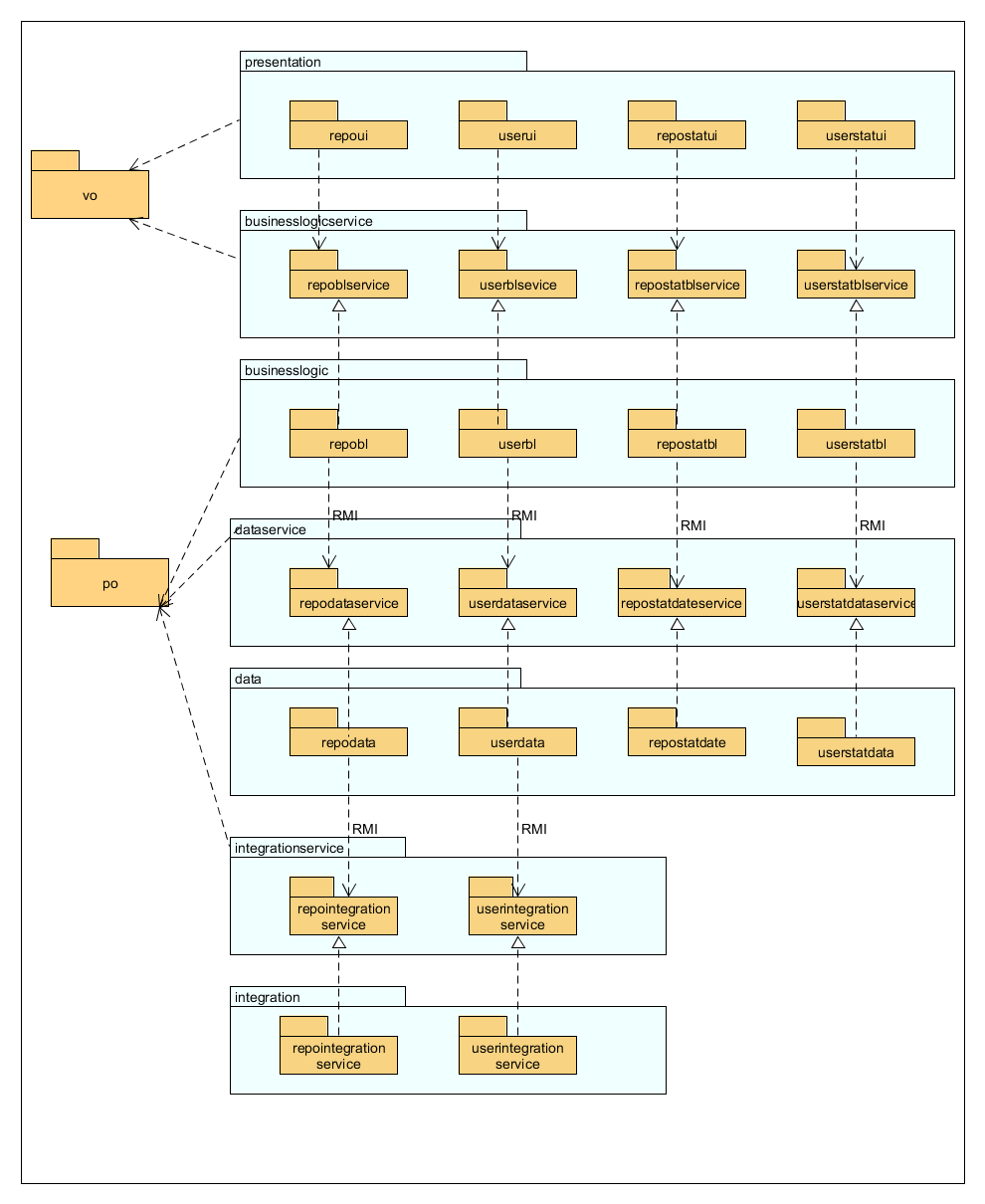
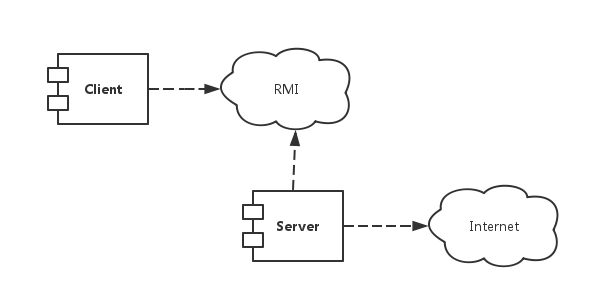


图3 Gitmining系统开发包图

## 4.2运行时进程

在Gitmining系统运行过程中，客户端从服务器获取数据，服务器根据需要定期从github或gitmining 的api更新服务器端存储的数据，进程图如图4.

图4 进程图

# 5接口视角

## 5.1模块的职责

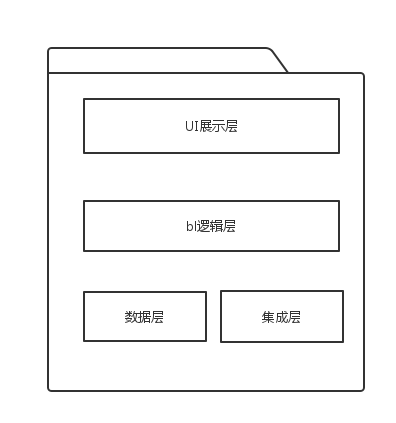
 Gitmining系统的模块视图如图5所示。各层的职责如表2所示。

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 |
| 集成层 | 利用已有的api联网获取数据并更新服务器端的数据 |
| 数据层 | 在远程服务端，向上向客户机提供数据服务 |

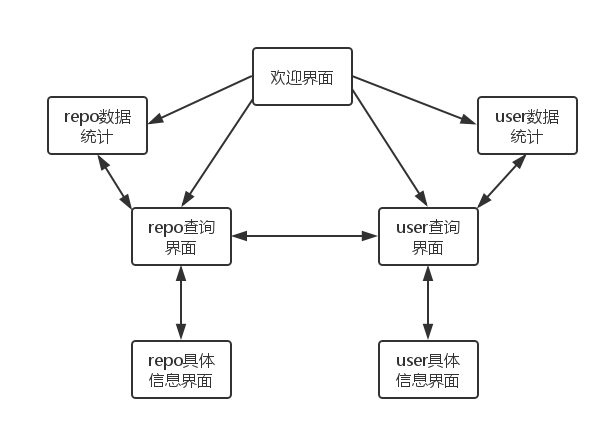
每一层只是使用下方直接接触的层。 层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如表4所示。

**表3 层之间调用的接口**

|  |  |  |
| --- | --- | --- |
| **接口** | **服务调用方** | **服务提供方** |
| repoblservice  userblservice  repostatblservice  userstatblservice | 展示层 | 业务逻辑层 |
| repodataservice  userdataservice  repostatdataservice  userstatdataservice | 业务逻辑层 | 数据层 |
| repointegrationservice  userintegrationservice | 业务逻辑层 | 集成层 |

## 5.2用户界面层的分解

根据需求，系统存在7个用户界面：欢迎界面，repo查询界面，repo具体信息界面，user查询界面，user具体信息界面，repo数据统计界面，user数据统计界面。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 |
| userstatbl | 负责处理与用户统计有关的逻辑业务 |
| repostatbl | 负责处理与仓库统计有关的逻辑业务 |

### 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 | 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.jumpToPage | 语法 | public List<RepoBriefVO> jumpToPage(int pageNum) |
| 前置条件 | 系统已正常打开 |
| 后置条件 | 跳转到制定页码 |
| repobl.nextPage | 语法 | public List<RepoBriefVO> nextPage() |
| 前置条件 | 系统已正常打开 |
| 后置条件 | 跳转到下一页 |
| repobl.previousPage | 语法 | public List<RepoBriefVO> previousPage() |
| 前置条件 | 系统已正常打开 |
| 后置条件 | 跳转到上一页 |
| repobl.sort | 语法 | public List<RepoBriefVO> sort(SortType sortType, boolean reverse); |
| 前置条件 | 系统已正常打开 |
| 后置条件 | 按照制定类型排序 |
| repobl.getRepoBasicInfo | 语法 | public RepoVO getRepoBasicInfo(String owner, String 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.getRepoContributor | 语法 | public List<UserBriefVO> getRepoContributor(String owner, String repoName) |
| 前置条件 | 用户正在查看仓库具体信息 |
| 后置条件 | 返回仓库贡献者信息 |
| repobl.getRepoCollaborator | 语法 | public List< UserBriefVO > getRepoCollaborator(String owner, String repoName) |
| 前置条件 | 用户正在查看仓库具体信息 |
| 后置条件 | 返回仓库合作者信息 |
| repobl.getRepoBranch | 语法 | public List<BranchVO> getRepoBranch(String owner, String 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 | 语法 | public List<RepoBriefVO> getRepoFork(String owner, String 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 | 语法 | public List<CommitVO> getRepoCommit(String owner, String 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 | 语法 | public List<IssueVO> getRepoIssue(String owner, String 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 |
| repobl.getShownList | 语法 | public List<RepoBriefVO> getShownRepoList(); |
| 前置条件 | 用户正在查看仓库具体信息 |
| 后置条件 | 返回仓库列表 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| repodataservice.getSearchResult | 获取搜索结果列表 | |
| repodataservice.getRepoBasicInfo | 获取仓库基本信息 | |
| repodataservice.getRepoContributor | 获取仓库贡献者信息 | |
| repodataservice.getRepoCollaborator | 获取仓库合作者信息 | |
| repodataservice.getRepoBranch | 获取仓库版本信息 | |
| repodataservice.getRepoFork | 获取仓库被引用信息 | |
| repodataservice.getRepoCommit | 获取仓库提交信息 | |
| repodataservice.getRepoIssue | 获取仓库事务信息 | |
| repodataservice.acceptVisitor | 利用visitor访问data层数据 | |

**表6 userbl模块的接口规范**

|  |  |  |
| --- | --- | --- |
| **提供的服务（供接口）** | | |
| userbl.getSearchResult | 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.jumpToPage |  | public List<UserBriefVO> jumpToPage(int pageNum) |
| 前置条件 | 用户已正常进入该系统 |
| 后置条件 | 跳转到指定页码 |
| userbl.getUserInfo | 语法 | public 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.getUserOwnRepos | 语法 | public List<RepoBriefVO> getUserOwnRepos (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 | 返回某用户的拥有的项目列表 |
| userbl.getUserSubscribeRepos | 语法 | public List<RepoBriefVO> getUserSubscribeRepos (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 | 返回某用户的订阅的项目列表 |
| userbl.getUserCollaborateRepos | 语法 | public List<RepoBriefVO> getUserSubscribeRepos (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 | 返回某用户的参加合作的项目列表 |
| userbl.getUserContributeRepos | 语法 | public List<RepoBriefVO> getUserSubscribeRepos (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 | 返回某用户的贡献的项目列表 |
| userbl.getShownList | 语法 | public List<UserBriefVO> getShownUserList(); |
| 前置条件 | 用户正在查看某github的user的详细信息 |
| 后置条件 | 返回用户信息列表 |
| **需要的服务（需接口）** | | |
| 服务名 | 服务 | |
| userdataservice.getSearchResult | 获取用户信息检索结果列表 | |
| userdataservice.getUserInfo | 获取单个用户的相信信息 | |
| userdataservice.getUserOwnRepos | 获取单个用户拥有的仓库列表 | |
| userdataservice.getUserSubscribeRepos | 获取单个用户订阅的仓库列表 | |
| userdataservice.getUserCollaborateRepos | 获取单个用户参加合作的仓库列表 | |
| userdataservice.getUserContributeRepos | 获取单个用户贡献的仓库列表 | |
| userdataservice.acceptVisitor | 利用visitor访问data层数据 | |

**表7 userstatbl模块的接口规范**

|  |  |
| --- | --- |
| **提供的服务（供接口）** | |
| userstatblservice.statUserType | 前置：无  后置：查找用户类型对应的用户量 |
| userstatblservice.statCompanyUser | 前置：无  后置：查找所有注册用户所属公司，以及对应用户数量 |
| usestatblservice.statUserblog | 前置：无  后置：查找所有博客，以及对应博客数量 |
| userstatblservice.statUserLocation | 前置：无  后置：返回所有地域以及对应地域的用户数量 |
| userstatblservice.statUserEmail | 前置：无  查找所有email类型以及对应email的用户数量 |
| userstatblservice.statCreateTime | 前置：无  后置：分析所有用户创建时间，并返回所有：{对应时间段内创建的用户数量} |
| userstatblservice.statUserOwnRepo | 前置：无  后置：分析所有用户拥有仓库数，并返回所有：{对应 仓库数的 用户数量} |
| userstatblservice.statUserGist | 前置：无  后置：分析所有用户拥有Gist数，并返回所有：{对应Gist数范围内的 用户数量} |
| userstatblservice.statUserFollowers | 前置：无  后置：分析所有用户拥有的follower数，并返回所有：  {对应follow范围内的 用户数量} |
| userstatblservice.statUserFollowing | 前置：无  后置：分析所有用户拥有的following数，并返回所有{对应following数范围内的 用户数量} |
| userstatblservice.statOrganizationUser | 前置：无  后置：通过用户分析所有用户所属组织，返回所有组织对应所有拥有的用户数目 |
| userstatblservice.statOrganizationRepo | 前置：无  后置：通过用户分析所有用户所属组织，返回所有组织对应所有拥有的仓库数目 |
| **需要的服务（需接口）** | |
| userstatdataservice.statUserType |  |
| userstatdataservice.statCompanyUser |  |
| usestatdataservice.statUserblog |  |
| userstatdataservice.statUserLocation |  |
| userstatdataservice.statUserEmail |  |
| userstatdataservice.statCreateTime |  |
| userstatdataservice.statUserOwnRepo |  |
| userstatdataservice.statUserGist |  |
| userstatdataservice.statUserFollowers |  |
| userstatdataservice.statUserFollowing |  |
| userstatdataservice.statOrganizationUser |  |
| userstatdataservice.statOrganizationRepo |  |

**表7 repostatbl模块的接口规范**

|  |  |
| --- | --- |
| **提供的服务（供接口）** | |
| repostatblservice.statLanguage | 前置：无  后置：统计所有项目语言，返回对应语言的使用情况。（代码行数与项目数） |
| repostatblservice.statCreateTime | 前置：无  后置：分析所有仓库创建时间，并返回所有：  {时间段，对应时间段内创建的用户数量} |
| repostatblservice.statContributors | 前置：无  后置：统计所有仓库的贡献者数目，返回所有：  {贡献者数，对应的 仓库数} |
| repostatblservice.statCollabrators | 前置：无  后置：统计所有仓库合作者数目，返回所有：  {合作者数目，对应的仓库数} |
| repostatblservice.statSize | 前置：无  后置：统计所有仓库大小。返回所有：  {仓库size范围， 对应范围内地仓库数} |
| repostatblservice.statSubscriber | 前置： 无  后组合i： 统计所有仓库订阅者数目，返回所有”  {仓库订阅者数目范围， 对应数目范围内的仓库数} |
| **需要的服务（需接口）** | |
| repostatdataservice.statLanguage |  |
| repostatdataservice.statCreateTime |  |
| repostatdataservice.statContributors |  |
| repostatdataservice.statCollabrators |  |
| repostatdataservice.statSize |  |
| repostatdataservice.statubscriber |  |

## 5.4数据层的分解

迭代二阶段相对于迭代一，在数据层上进行了较大的改进。

从迭代二开始， 我们将数据层独立出来形成一个单独的应用程序，形成服务器，而客户端将通过RMI和服务器相连，从而客户端将不再从api.gitmining.net网站和api.github.com网站上直接获取数据，而是从服务器获取数据。服务器将利用github或gitmining的接口定期更新自己存储的数据。

### 5.4.1数据层模块的职责

**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 |
| repostatdataservice | 为repostatbl模块提供数据获取与统计服务 |
| userstatdataservice | 为userdatabl模块提供数据获取与统计服务 |

### 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 | 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.acceptVisitor | 语法 | public List<RepoBriefVO> acceptVisitor(RepoVisitor visitor); |
| 前置条件 | 无 |
| 后置条件 | 调用visitor中的方法进行数据访问和分析 |
| repodataservice.getRepoBasicInfo | 语法 | public RepoVO getRepoBasicInfo(String owner, String repoName) |
| 前置条件 | 无 |
| 后置条件 | 返回单个仓库的基本信息 |
| repodataservice.getRepoContributor | 语法 | public List< UserBriefVO > getRepoContributor(String owner, String repoName) |
| 前置条件 | 无 |
| 后置条件 | 返回仓库贡献者信息 |
| repodataservice.getRepoCollaborator | 语法 | public List< UserBriefVO > getRepoCollaborator(String owner, String repoName) |
| 前置条件 | 无 |
| 后置条件 | 返回仓库合作者信息 |
| repodataservice.getRepoBranch | 语法 | public List<BranchVO> getRepoBranch(String owner, String repoName) |
| 前置条件 | 无 |
| 后置条件 | 返回仓库版本信息 |
| repodataservice.getRepoFork | 语法 | public List<RepoBriefVO> getRepoFork(String owner, String repoName) |
| 前置条件 | 无 |
| 后置条件 | 返回仓库fork的其他项目信息 |
| repodataservice.getRepoCommit | 语法 | public List<CommitVO> getRepoCommit(String owner, String repoName) |
| 前置条件 | 无 |
| 后置条件 | 返回仓库commit信息 |
| repodataservice.getRepoIssue | 语法 | public List<IssueVO> getRepoIssue(String owner, String 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 | |
| 公共接口  https://api.github.com | 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 | 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.acceptVisitor |  | public List<UserBriefVO> acceptVisitor(UserVisitor visitor); |
| 前置条件 | 无 |
| 后置条件 | 调用visitor访问数据 |
| userdataservice.getUserInfo | 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.getUserOwnRepos | 语法 | public List<RepoBriefVO> getUserOwnRepos (String userName) |
| 前置条件 | 无 |
| 后置条件 | 返回用户的拥有的项目 |
| userdataservice.getUserSubscribeRepos | 语法 | public List<RepoBriefVO> getUserSubscribeRepos (String userName) |
| 前置条件 | 无 |
| 后置条件 | 返回用户的订阅的项目 |
| userdataservice.getUserCollaborateRepos | 语法 | public List<RepoBriefVO> getUserCollaborateRepos (String userName) |
| 前置条件 | 无 |
| 后置条件 | 返回用户的参见合作的项目 |
| userdataservice.getUserContributeRepos | 语法 | public List<RepoBriefVO> getUserContributeRepos (String userName) |
| 前置条件 | 无 |
| 后置条件 | 返回用户的贡献的项目 |
| **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 | |
| 公共接口  https://api.github.com | 获取github相关repository和user信息 | |

**表9 repostatdata数据层模块的接口规范**

|  |  |
| --- | --- |
| **提供的接口（供接口）** | |
| userstatdataservice.statUserType |  |
| userstatdataservice.statCompanyUser |  |
| usestatdataservice.statUserblog |  |
| userstatdataservice.statUserLocation |  |
| userstatdataservice.statUserEmail |  |
| userstatdataservice.statCreateTime |  |
| userstatdataservice.statUserOwnRepo |  |
| userstatdataservice.statUserGist |  |
| userstatdataservice.statUserFollowers |  |
| userstatdataservice.statUserFollowing |  |
| userstatdataservice.statOrganizationUser |  |
| userstatdataservice.statOrganizationRepo |  |

**表9 repostatdata数据层模块的接口规范**

|  |  |
| --- | --- |
| **提供的接口（供接口）** | |
| repostatdataservice.statLanguage |  |
| repostatdataservice.statCreateTime |  |
| repostatdataservice.statContributors |  |
| repostatdataservice.statCollabrators |  |
| repostatdataservice.statSize |  |
| repostatdataservice.statubscriber |  |

# 6信息视角

迭代二将把所有需要的数据都存在服务器端，当客户端需要数据时会想服务器索取。服务器定期从互联网更新自己的数据。