

Design and coding

软件需求工程 Fissure组 milestone 4

组员： 庄毅非、李予谦、刘奕骁、应凌凯、何迪

1. 引言

1.1 编写目的

系统设计计划主要为使软件产品和软件项目满足规定的软件规格要求而确定软件系统的体系结构、组成部分、数据组织、模块、内外部接口。主要任务有：

- (1)建立软件产品和软件项目目标系统的总体结构；
- (2)总体设计；
- (3)接口设计；
- (4)运行设计；
- (5)系统数据结构设计；
- (6)系统出错处理设计。

1.2 项目简介

我们所要开发的开源项目分析平台面向的用户主要是开源项目管理者。通过使用我们开发的平台，开源项目管理者能够方便的管控项目的开发进程。

开源项目管理者可以使用我们的平台对软件开发的贡献者进行分析，从中将其归类为核心贡献者和其他贡献者，通过有效的措施增加贡献者的留存率，发现新的核心贡献者；可以可视化展示参与开发的人员的所属公司，方便管理者和对应的公司进行合作，更好推进项目发展；可以对项目的代码质量进行高效评估，确保项目代码质量。可以对项目的issue、pullrequest进行分析，从中挖掘出用户（主要是开发者）对软件所提出的需求，把我软件开发方向和里程碑。通过上述措施，更好地推进项目开发进程，确保开源项目长久运行下去。

2. 系统环境

2.1 运行环境

2.1.1 软件层面

开源项目分析平台 DoubleC 以网页的形式呈现数据，建议用户使用chrome进行浏览。

项目服务端运行在云服务器上，数据库使用mongodb，这是一种新兴的nosql数据库，能够支持数据的快速存取和修改操作。

2.1.2 硬件层面

硬件层面对于所使用的服务器有以下要求：

- CPU:主频大于 2.0GHz
- 内存:大于等于 2GB
- 硬盘:硬盘容量大于 200GB、硬盘转速大于等于 5400 转/分钟

- 网卡:百兆网卡
- 网线:具有良好的数据传输能力即可
- 键盘:可以满足正常使用即可
- 鼠标:可以满足正常使用即可
- 显示器:可以满足正常使用即可

2.2 测试环境

我们计划对win10, win11, macos和ubuntu平台进行测试

2.3 系统模块设计

1. 用户管理模块

1. 主要包括用户注册, 登录, 修改信息操作

2. 数据导入模块

1. 爬取开源项目 (来自github) 的commit信息
2. 爬取开源项目 (来自github) 的issue信息
3. 爬取开源项目 (来自github) 的pull信息
4. 爬取开源项目 (来自github) 的contributor信息
5. 爬取开源项目 (来自github) 的comment信息

3. 数据显示模块

1. 显示仓库总体信息
2. 以图表的形式可视化显示commit频率按照每月, 每年的变化
3. 以图表的形式可视化显示issue create/update/close频率按照每月, 每年的变化
4. 以图表的形式可视化显示pull create/update/close频率按照每月, 每年的变化
5. 以图表的形式可视化显示puller、issuer、committer数量随时间变化的情况, **反映社区热度**
6. 用多种图表反映issue从提出到第一次响应所需要的时间的变化 (包括最短时间, 前1/4时间, 前1/2时间, 前3/4时间和最长时间), **反映社区热度**
7. 用多种图表反映issue从提出到close所需要的时间的变化 (包括最短时间, 前1/4时间, 前1/2时间, 前3/4时间和最长时间), **反映社区热度**
8. 分析并显示不同时期issue和pull中主要讨论的话题
9. 实现两个仓库的对比

4. 数据分析模块

1. 实现对上述数据显示模块中可视化所需数据的快速计算和返回

2.3.2 应用架构

我们小组计划使用普遍使用的B/S架构进行开发

2.4 数据处理

2.4.1 数据存储

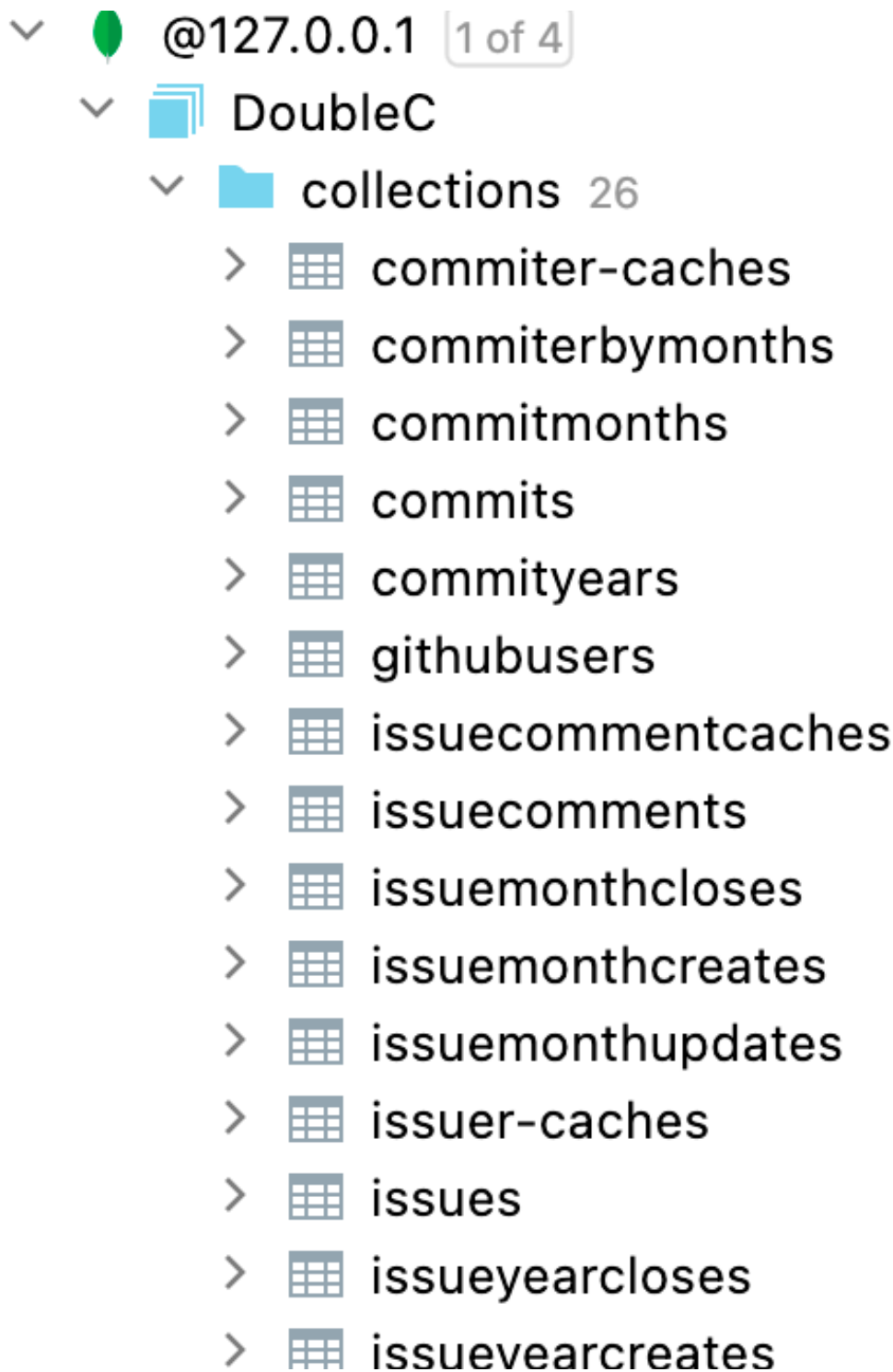
项目产品使用标准 Mongodb 数据库系统作为引擎，按照数据产生、转换和存储的策略， 通过将数据导入数据库的方式进行数据的存储操作。












2.4.2 数据安全

我们对服务器mongo的访问ip白名单，数据库访问用户名、密码（使用pwgen生成，保障安全性）和持有的权限进行了严格的限制，确保小组之外的成员无法访问数据库。

2.4.3 数据库设计

数据表总览。



- >  issueyearupdates
- >  puller-caches
- >  pullmonthcloses
- >  pullmonthcreates
- >  pullmonthupdates
- >  pulls
- >  pullyearcloses
- >  pullyearcreates
- >  pullyearupdates
- >  reposchemas
- >  users

各个数据表的定义，由于内容较多，我们放到了附录中进行展示。

3. 系统实现

//TODO

4. 附录

4.1 数据表定义

4.1.1 commit表定义

存储每个commit的基本信息。

```
1 | const CommitSchema = new mongoose.Schema({
2 |   sha: {
3 |     type: String,
4 |     required: [true, 'must provide sha'], //must have the property
```

```

5   },
6   url: {
7     type: String,
8     required: [true, 'must provide url'],
9   },
10  author_id: {
11    type: Number,
12  },
13  author_name: {
14    type: String,
15  },
16  author_email: {
17    type: String,
18  },
19  updated_at: {
20    type: String,
21  },
22  message: {
23    type: String,
24  },
25  repos_id: {
26    type: String,
27    required: [true, 'must provide repos_id'],
28  },
29  // 以下为冗余字段, 仅供查询使用
30  repo_owner: {
31    type: String,
32    required: [true, 'must provide repo_owner'],
33  },
34  repo_name: {
35    type: String,
36    required: [true, 'must provide repo_name'],
37  },
38 });

```

4.1.2 commityearcache表定义

缓存表, 存储每年的commit总数的变化情况。

```

1  const CommitYearCacheSchema = new mongoose.Schema({
2    commit_year_frequency: {
3      type: Object,
4      required: [true, 'must provide commit_year_frequency'],
5    },

```

```

6 // 以下为冗余字段，仅供查询使用
7 repo_owner: {
8   type: String,
9   required: [true, 'must provide repo_owner'],
10 },
11 repo_name: {
12   type: String,
13   required: [true, 'must provide repo_name'],
14 },
15 });
16

```

4.1.3 commitMonthCache表定义

缓存表，存储每月的commit频率的变化情况。

```

1 const CommitMonthCacheSchema = new mongoose.Schema({
2   commit_month_frequency: {
3     type: Object,
4     required: [true, 'must provide commit_month_frequency'],
5   },
6   // 以下为冗余字段，仅供查询使用
7   repo_owner: {
8     type: String,
9     required: [true, 'must provide repo_owner'],
10  },
11  repo_name: {
12    type: String,
13    required: [true, 'must provide repo_name'],
14  },
15 });
16

```

4.1.4 commiter表定义

缓存表，存储每月的commiter数量的变化情况。

```

1 const CommiterCacheSchema = new mongoose.Schema({
2   commiter_count: {
3     type: Object,
4     required: [true, 'must provide commiter_count'],
5   },
6   // 以下为冗余字段，仅供查询使用
7   repo_owner: {

```

```

8     type: String,
9     required: [true, 'must provide repo_owner'],
10  },
11  repo_name: {
12    type: String,
13    required: [true, 'must provide repo_name'],
14  },
15  });

```

4.1.5 githubuser表定义

存储github用户信息，用来分析公司贡献。

```

1  const GithubUserSchema = new mongoose.Schema({
2    login: {
3      type: String,
4      required: [true, 'must provide loginname'],
5    },
6    company: {
7      type: String,
8    },
9  });

```

4.1.6 issueComment表定义

存储了对issue进行评论（如果有）的第一条非机器人评论，用来分析issue的第一次得到响应时间。

```

1  const IssueCommentSchema = new mongoose.Schema({
2    url: {
3      type: String,
4      required: [true, 'must provide url'],
5    },
6    created_at: {
7      type: String,
8      required: [true, 'must provide created_at'],
9    },
10   issue_id: {
11     type: Number,
12   },
13   body: {
14     type: String,
15   },
16   issue_number: {
17     type: Number,

```

```

18   },
19   valid: {
20     type: Boolean,
21     default: true,
22   }, // 以下为冗余字段, 仅供查询使用
23   repo_owner: {
24     type: String,
25     required: [true, 'must provide repo_owner'],
26   },
27   repo_name: {
28     type: String,
29     required: [true, 'must provide repo_name'],
30   },
31 });

```

4.1.7 issueCommentCache表定义

缓存表, 存储已经计算出来的过去月份的平均响应时间, 避免重复计算。

```

1  const IssueCommentCacheSchema = new mongoose.Schema({
2    month: {
3      type: String,
4      required: [true, 'must provide month'],
5    },
6    value: {
7      type: Object,
8      require: [true, 'must provide value'],
9    },
10   // 以下为冗余字段, 仅供查询使用
11   repo_owner: {
12     type: String,
13     required: [true, 'must provide repo_owner'],
14   },
15   repo_name: {
16     type: String,
17     required: [true, 'must provide repo_name'],
18   },
19 });

```


4.1.8 issueMonthCloseCache表

缓存表，存储已经计算出来的过去月份的close数量，避免重复计算。

```
1  const IssueMonthCloseCacheSchema = new mongoose.Schema({
2    issue_month_close_frequency: {
3      type: Object,
4      required: [true, 'must provide issue_month_close_frequency'],
5    },
6    // 以下为冗余字段，仅供查询使用
7    repo_owner: {
8      type: String,
9      required: [true, 'must provide repo_owner'],
10   },
11   repo_name: {
12     type: String,
13     required: [true, 'must provide repo_name'],
14   },
15 });
16
17
```

4.1.9 issueMonthCreateCache

缓存表，存储已经计算出来的过去月份的create数量，避免重复计算。

```
1  const IssueMonthCreateCacheSchema = new mongoose.Schema({
2    issue_month_create_frequency: {
3      type: Object,
4      required: [true, 'must provide issue_month_create_frequency'],
5    },
6    // 以下为冗余字段，仅供查询使用
7    repo_owner: {
8      type: String,
9      required: [true, 'must provide repo_owner'],
10   },
11   repo_name: {
12     type: String,
13     required: [true, 'must provide repo_name'],
14   },
15 });
16
```

4.1.10 issueMonthUpdateCache

缓存表，存储已经计算出来的过去月份的update数量，避免重复计算。

```
1  const IssueMonthUpdateCacheSchema = new mongoose.Schema({
2    issue_month_update_frequency: {
3      type: Object,
4      required: [true, 'must provide issue_month_update_frequency'],
5    },
6    // 以下为冗余字段，仅供查询使用
7    repo_owner: {
8      type: String,
9      required: [true, 'must provide repo_owner'],
10   },
11   repo_name: {
12     type: String,
13     required: [true, 'must provide repo_name'],
14   },
15 });
```

4.1.11 issueYearCloseCache表

缓存表，存储已经计算出来的过去年份的close数量，避免重复计算。

```
1  const IssueYearCloseCacheSchema = new mongoose.Schema({
2    issue_year_close_frequency: {
3      type: Object,
4      required: [true, 'must provide issue_year_close_frequency'],
5    },
6    // 以下为冗余字段，仅供查询使用
7    repo_owner: {
8      type: String,
9      required: [true, 'must provide repo_owner'],
10   },
11   repo_name: {
12     type: String,
13     required: [true, 'must provide repo_name'],
14   },
15 });
```

4.1.12 issueYearCreateCache表

缓存表，存储已经计算出来的过去月份的create数量，避免重复计算。

```
1  const IssueYearCreateCacheSchema = new mongoose.Schema({
2    issue_year_create_frequency: {
3      type: Object,
4      required: [true, 'must provide issue_year_create_frequency'],
5    },
6    // 以下为冗余字段，仅供查询使用
7    repo_owner: {
8      type: String,
9      required: [true, 'must provide repo_owner'],
10   },
11   repo_name: {
12     type: String,
13     required: [true, 'must provide repo_name'],
14   },
15 });
16
```

4.1.13 issueMonthUpdateCache表

缓存表，存储已经计算出来的过去年份的update数量，避免重复计算。

```
1  const IssueYearUpdateCacheSchema = new mongoose.Schema({
2    issue_year_update_frequency: {
3      type: Object,
4      required: [true, 'must provide issue_year_update_frequency'],
5    },
6    // 以下为冗余字段，仅供查询使用
7    repo_owner: {
8      type: String,
9      required: [true, 'must provide repo_owner'],
10   },
11   repo_name: {
12     type: String,
13     required: [true, 'must provide repo_name'],
14   },
15 });
```

4.1.14 issue表定义

```
1  const IssueSchema = new mongoose.Schema({
2    id: {
3      type: Number,
4      required: [true, 'must provide id'], //must have the property
5    },
6    number: {
7      type: Number,
8    },
9    url: {
10     type: String,
11   },
12   title: {
13     type: String,
14   },
15   state: {
16     type: String,
17   },
18   is_locked: {
19     type: Boolean,
20   },
21   body: {
22     type: String,
23   },
24   created_at: {
25     type: String,
26     required: [true, 'must provide created_at'],
27   },
28   updated_at: {
29     type: String,
30     required: [true, 'must provide updated_at'],
31   },
32   closed_at: {
33     type: String,
34   },
35   repos_id: {
36     type: Number,
37   },
38   user_id: {
39     type: Number,
40   },
41   user_name: {
42     type: String,
43   },
```

```

44   comment_count: {
45     type: Number,
46     require: [true, 'must provide comment_count'],
47   },
48   labels: {
49     type: [String],
50   },
51   // 以下为冗余字段, 仅供查询使用
52   repo_owner: {
53     type: String,
54     required: [true, 'must provide repo_owner'],
55   },
56   repo_name: {
57     type: String,
58     required: [true, 'must provide repo_name'],
59   },
60 });
61

```

4.1.15 issueCache表定义

缓存表, 存储了过去每月的issuer数量变化

```

1  const IssuerCacheSchema = new mongoose.Schema({
2    issuer_count: {
3      type: Object,
4      required: [true, 'must provide issuer_count'],
5    },
6    // 以下为冗余字段, 仅供查询使用
7    repo_owner: {
8      type: String,
9      required: [true, 'must provide repo_owner'],
10   },
11   repo_name: {
12     type: String,
13     required: [true, 'must provide repo_name'],
14   },
15 });
16

```

4.1.16 pullMonthCloseCache表

缓存表，存储已经计算出来的过去月份的close数量，避免重复计算。

```
1  const PullMonthCloseCacheSchema = new mongoose.Schema({
2    pull_month_close_frequency: {
3      type: Object,
4      required: [true, 'must provide pull_month_close_frequency'],
5    },
6    // 以下为冗余字段，仅供查询使用
7    repo_owner: {
8      type: String,
9      required: [true, 'must provide repo_owner'],
10   },
11   repo_name: {
12     type: String,
13     required: [true, 'must provide repo_name'],
14   },
15 });
16
17
```

4.1.13 PullMonthCreateCache

缓存表，存储已经计算出来的过去月份的create数量，避免重复计算。

```
1  const PullMonthCreateCacheSchema = new mongoose.Schema({
2    pull_month_create_frequency: {
3      type: Object,
4      required: [true, 'must provide pull_month_create_frequency'],
5    },
6    // 以下为冗余字段，仅供查询使用
7    repo_owner: {
8      type: String,
9      required: [true, 'must provide repo_owner'],
10   },
11   repo_name: {
12     type: String,
13     required: [true, 'must provide repo_name'],
14   },
15 });
16
```

4.1.14 pullMonthUpdateCache

缓存表，存储已经计算出来的过去月份的update数量，避免重复计算。

```
1  const PullMonthUpdateCacheSchema = new mongoose.Schema({
2    pull_month_update_frequency: {
3      type: Object,
4      required: [true, 'must provide pull_month_update_frequency'],
5    },
6    // 以下为冗余字段，仅供查询使用
7    repo_owner: {
8      type: String,
9      required: [true, 'must provide repo_owner'],
10   },
11   repo_name: {
12     type: String,
13     required: [true, 'must provide repo_name'],
14   },
15 });
```

4.1.15 pullYearCloseCache表

缓存表，存储已经计算出来的过去年份的close数量，避免重复计算。

```
1  const PullYearCloseCacheSchema = new mongoose.Schema({
2    pull_year_close_frequency: {
3      type: Object,
4      required: [true, 'must provide pull_year_close_frequency'],
5    },
6    // 以下为冗余字段，仅供查询使用
7    repo_owner: {
8      type: String,
9      required: [true, 'must provide repo_owner'],
10   },
11   repo_name: {
12     type: String,
13     required: [true, 'must provide repo_name'],
14   },
15 });
```

4.1.16 pullYearCreateCache表

缓存表，存储已经计算出来的过去月份的create数量，避免重复计算。

```
1 const PullYearCreateCacheSchema = new mongoose.Schema({
2   pull_year_create_frequency: {
3     type: Object,
4     required: [true, 'must provide pull_year_create_frequency'],
5   },
6   // 以下为冗余字段，仅供查询使用
7   repo_owner: {
8     type: String,
9     required: [true, 'must provide repo_owner'],
10  },
11  repo_name: {
12    type: String,
13    required: [true, 'must provide repo_name'],
14  },
15 });
16
```

4.1.17 pullMonthUpdateCache表

缓存表，存储已经计算出来的过去年份的update数量，避免重复计算。

```
1 const PullYearUpdateCacheSchema = new mongoose.Schema({
2   pull_year_update_frequency: {
3     type: Object,
4     required: [true, 'must provide pull_year_update_frequency'],
5   },
6   // 以下为冗余字段，仅供查询使用
7   repo_owner: {
8     type: String,
9     required: [true, 'must provide repo_owner'],
10  },
11  repo_name: {
12    type: String,
13    required: [true, 'must provide repo_name'],
14  },
15 });
```


4.1.18 pull表定义

```
1 const PullSchema = new mongoose.Schema({
2   id: {
3     type: Number,
4     required: [true, 'must provide id'], //must have the property
5   },
6   url: {
7     type: String,
8     required: [true, 'must provide url'],
9   },
10  number: {
11    type: Number,
12    required: [true, 'must provide number'],
13  },
14  state: {
15    type: String,
16    required: [true, 'must provide state'],
17  },
18  title: {
19    type: String,
20    required: [true, 'must provide title'],
21  },
22  isLocked: {
23    type: Boolean,
24    required: [true, 'must provide isLocked'],
25  },
26  body: {
27    type: String,
28    required: [true, 'must provide body'],
29  },
30  created_at: {
31    type: String,
32    required: [true, 'must provide created_at'],
33  },
34  updated_at: {
35    type: String,
36    required: [true, 'must provide updated_at'],
37  },
38  closed_at: {
39    type: String,
40  },
41  is_merged: {
42    type: Boolean,
43    required: [true, 'must provide is_merged'],
```

```

44   },
45   repos_id: {
46     type: Number,
47     required: [`true`, 'must provide repo's id'],
48   },
49   user_id: {
50     type: String,
51   },
52   labels: {
53     type: [String],
54   },
55   // 以下为冗余字段，仅供查询使用
56   repo_owner: {
57     type: String,
58     required: [true, 'must provide repo_owner'],
59   },
60   repo_name: {
61     type: String,
62     required: [true, 'must provide repo_name'],
63   },
64 });
65

```

4.1.15 issueCache表定义

缓存表，存储了过去每月的issuer数量变化

```

1  const PullerCacheSchema = new mongoose.Schema({
2    puller_count: {
3      type: Object,
4      required: [true, 'must provide puller_count'],
5    },
6    // 以下为冗余字段，仅供查询使用
7    repo_owner: {
8      type: String,
9      required: [true, 'must provide repo_owner'],
10   },
11   repo_name: {
12     type: String,
13     required: [true, 'must provide repo_name'],
14   },
15 });
16

```

4.1.16 repo

描述仓库的基本信息。

```
1  const Repo = new mongoose.Schema({
2    name: {
3      type: String,
4      required: [true, 'must provide name'], //must have the property
5    },
6    owner: {
7      type: String,
8      required: [true, 'must provide owner'],
9    },
10   uploader: {
11     type: String,
12     required: [true, 'must provide uploader'],
13   },
14   forks: {
15     type: Number,
16     required: [true, 'must provide forks'],
17   },
18   stars: {
19     type: Number,
20     required: [true, 'must provide stars'],
21   },
22   open_issues: {
23     type: Number,
24     required: [true, 'must provide open_issues'],
25   },
26   commit_frequency: {
27     type: Object,
28     required: [true, 'must provide commit_frequency'],
29   },
30   issue_frequency: {
31     type: Object,
32     required: [true, 'must provide issue_frequency'],
33   },
34   contributors: {
35     type: [Object],
36     required: [true, 'must provide contributors'],
37   },
38   timeline: {
39     type: Object,
40   },
41   language: {
```

```

42     type: Object,
43   },
44   });
45

```

4.1.17 user表

描述系统注册用户的信息

```

1  const User = new mongoose.Schema({
2    username: {
3      type: String,
4      required: [true, 'Please provide name'],
5      maxLength: [20, 'Password should be less than 20 characters'],
6      minLength: [3, 'Password should be more than 3 characters'],
7      trim: true,
8    },
9    email: {
10     type: String,
11     required: [true, 'Please provide email'],
12     match: [
13       /^[^<>()[\]\.\,\;\s@"]+(\.[^<>()[\]\.\,\;\s@"]+)*|(".*")@(\[[0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\)|\([a-zA-Z\-\0-9]+\.[a-zA-Z]{2,})\)$/
14     ],
15     'Please provide a valid email',
16     unique: [true, 'The email is already used!'],
17   },
18   password: {
19     type: String,
20     required: [true, 'Please provide password'],
21     minLength: [6, 'Password should be more than 6 characters'],
22   }
23 }
24 })

```

