CS209A Project Report

Frontend: 陈菀婷 12011440

Backend: 段宜凯 12011221

Overview

As we know, GitHub is a website for developers to store and manage their code. Developers could also use GitHub to track the releases, versions, issues, commits (code changes) and discussions of their projects. For a given GitHub repository, we are interested in the following questions.

In this project, we mainly discuss three major problem: developers, issues and releases&commit

1. Developers
   How many developers have committed to this repo?
   Which developers are the most active (i.e., who committed the most)?
2. Issues
   How many issues are open and how many are closed?
   What is the typical issue resolution time (i.e., the duration between issue open time and issue close
   time) for this repo?
3. Releases and Commits
   How many releases are there in this repo?
   How many commits are made between each release?
   At which time (e.g., weekday, weekend, morning, evening, etc.) do developers made commits?
4. Advanced Requirements
   - Multiple repositories
     Your web application could handle multiple GitHub repos. Users could navigate through different repos to see the corresponding visualizations.
   - Issue topics
     A GitHub repo has an "issues" tab, in which there are many issue threads. In a single issue thread, developers often have many rounds of discussions on why the issue raises and how to solve it. For instance, in the spring-boot repo, you could see a list of issues here or click a single issue thread to see all of the discussions.

The architecture of the project is **Vue + SpringBoot**. The development of frontend and backend are splited, and as a result any of them can work separately. The interaction between the frontend and the backend are achieved through Rest API, and we use **Json** as the data exchange format.

In this report, we will introduce the features related to the evaluation and the structure of this project.

Features

Project Structure

Frontend

File tree

```
|  App.vue
|  main.js
|  router.js
|
├─assets
|  |  bg.jpg
|  |  first.jpg
|  |  lockLogin.png
|  |  logo.png
|  |  showcase.png
|  |
|  └─js
|          Blob.js
|          Export2Excel.js
|
├─common
|      bus.js
|      Tags.vue
|      theme.vue
|
├─components
|      Dialong.vue
|      Header.vue
|      Header1.vue
|      Header2.vue
|      HistoryDialong.vue
|      LeftMenu.vue
|      Menu.vue
|      UserDialong.vue
|
├─store
|      actions.js
|      getters.js
|      index.js
|      mutation-type.js
|      mutations.js
|      state.js
|
└─views
    |  Home.vue
    |  Index.vue
    |  Index1.vue
    |  Index2.vue
    |  isuess.vue
    |  other.vue
    |  releases.vue
    |
    └─Home
```

```
            dd.vue
            developer.vue
```

Structure

The whole frontend is based on `NodeJs` and `Vue.js` framework. The intention is to create a dynamic web application that allows user interaction from **web brosers** for an immersive user experience. *In this project we used Element UI and Echarts as a component gallery* (See资源 | Element;Apache ECharts).

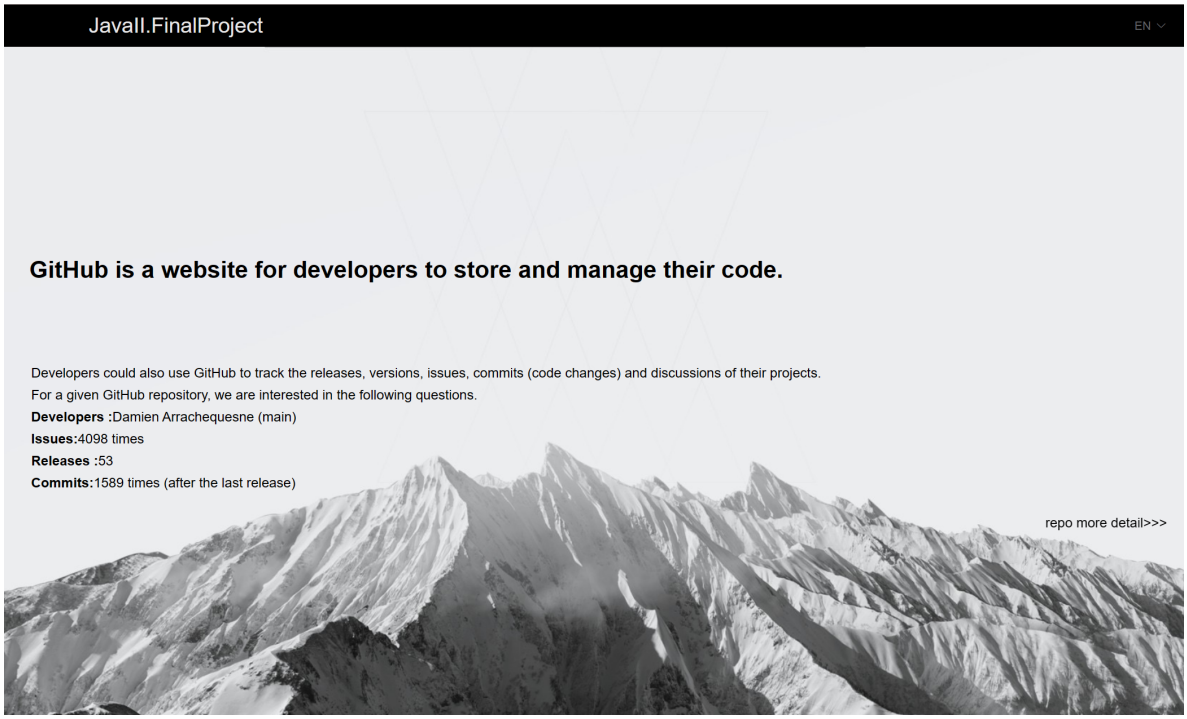
The frontend and the backend uses **Rest API** to communicate: The frontend uses `axios` method to get the data from the backend server and to post configurations and operations to the backend server for further actions.

DashBoard

In this views, we implement twoe main scenes.

Summary scene

Summary scene will displays simple repo information, such as main developer, latest issues, and original github address. And provide a jump path extending to the Detail plate.

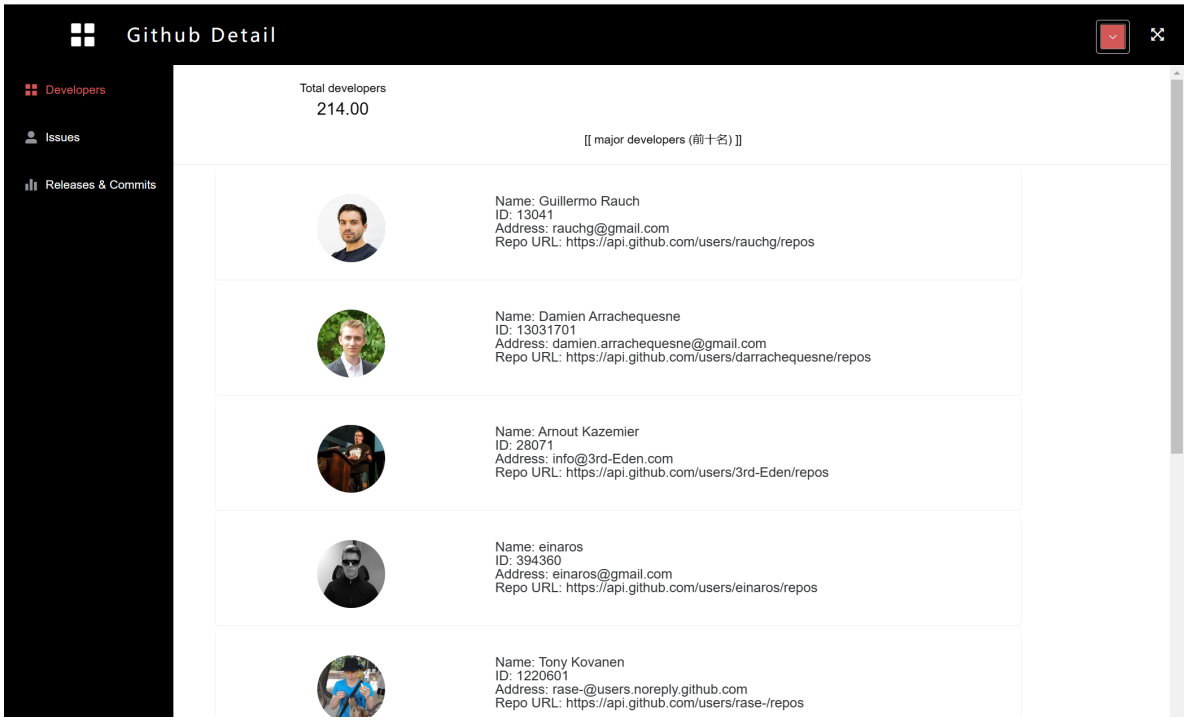


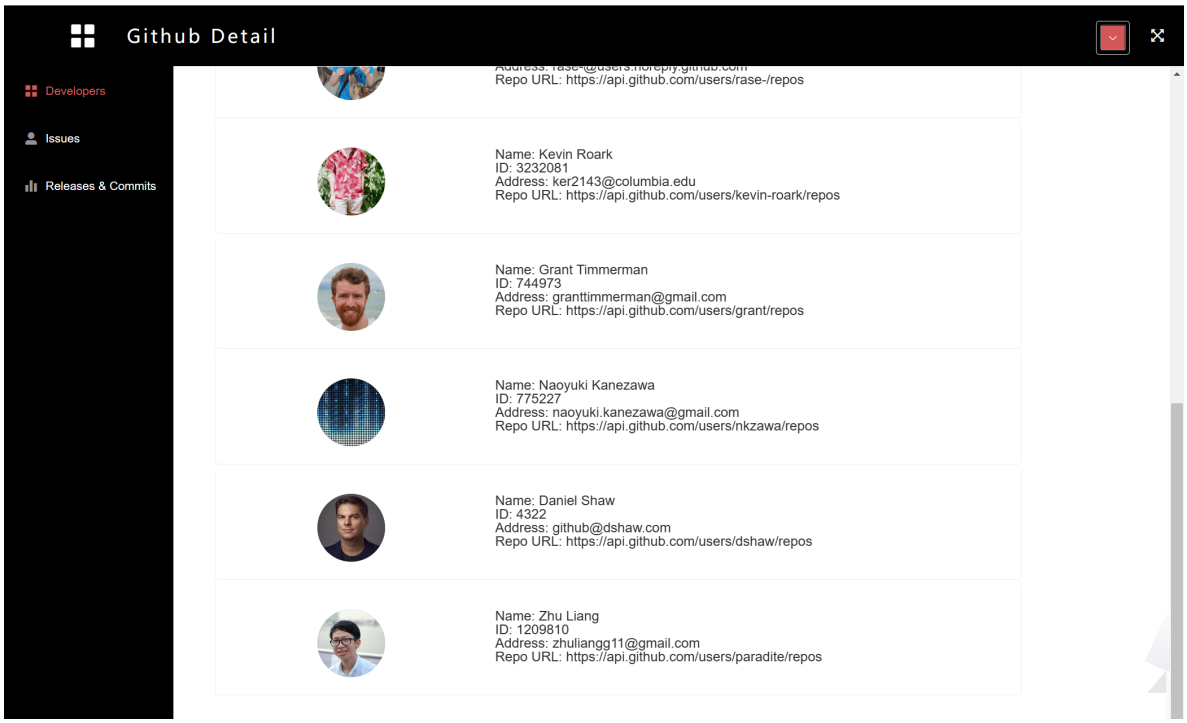
Detail panel

- developers

  As a visual display of repo's developers, it shows the total number of developers, as well as information about the major developers

  1. Total number
  2. Information of the major

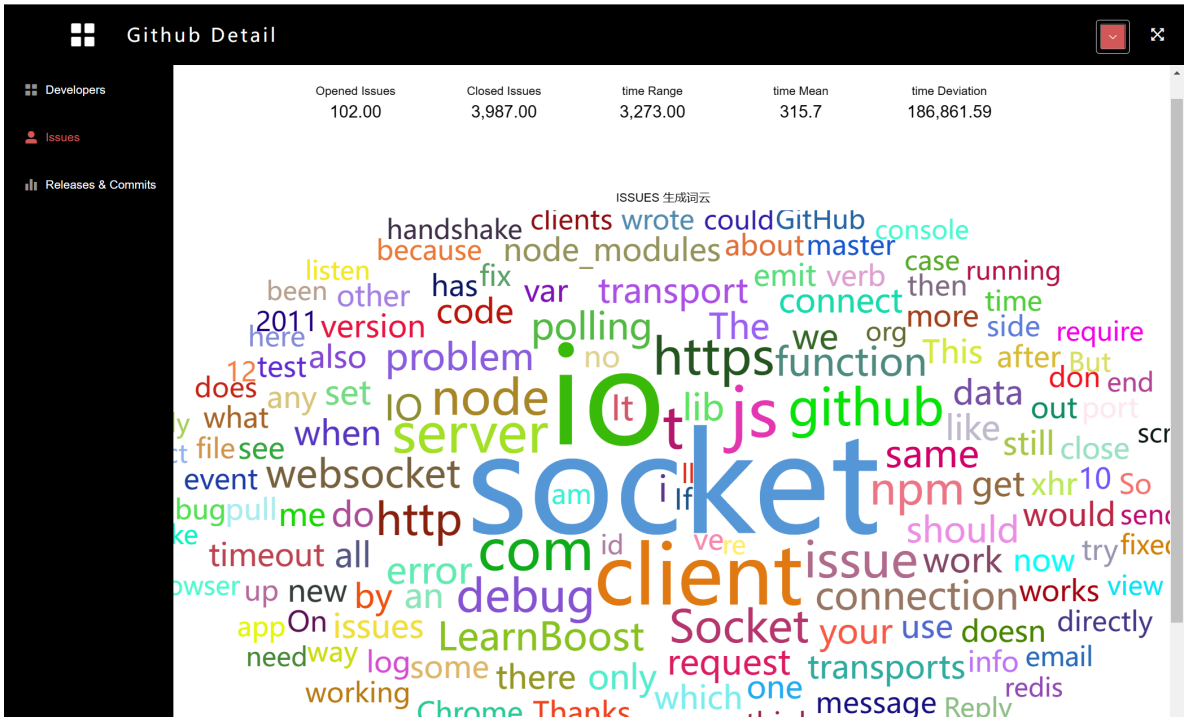




- issues

  As an issues presentation page, it presents the following information in visual digital form.

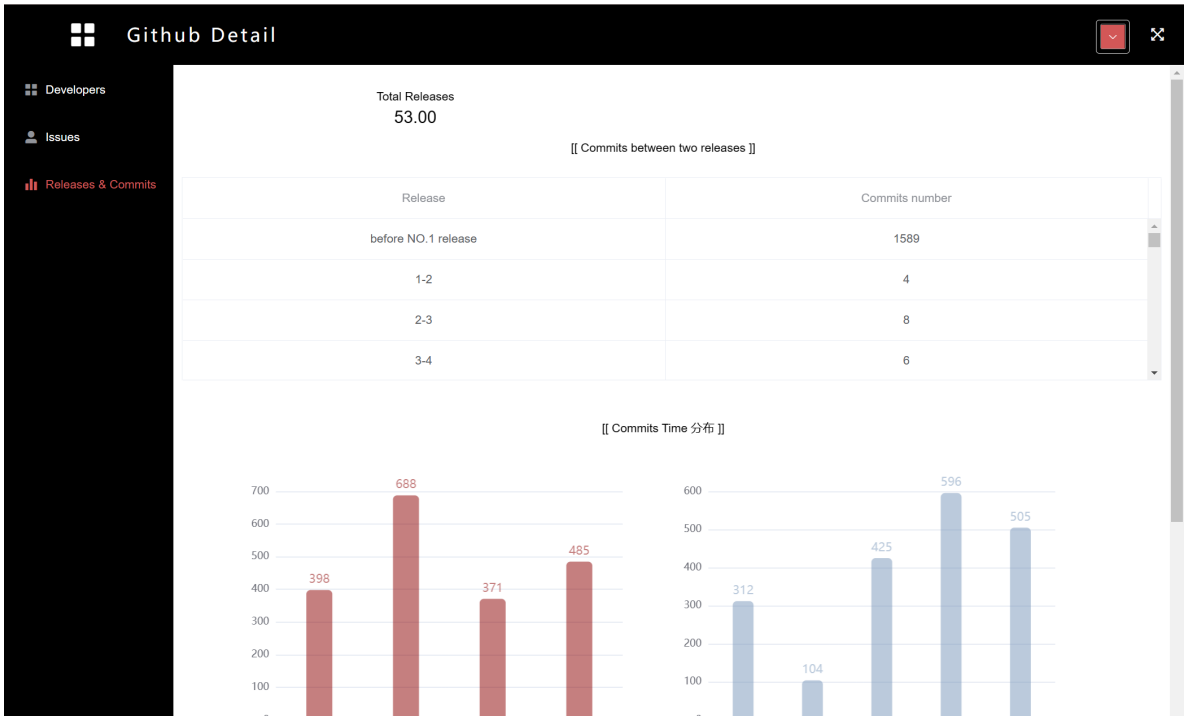
  1. the number of open issues
  2. the number of closed issues
  3. a typical treatment of issue resolution time, such as average value, extreme difference, variance
  4. the total number of releases

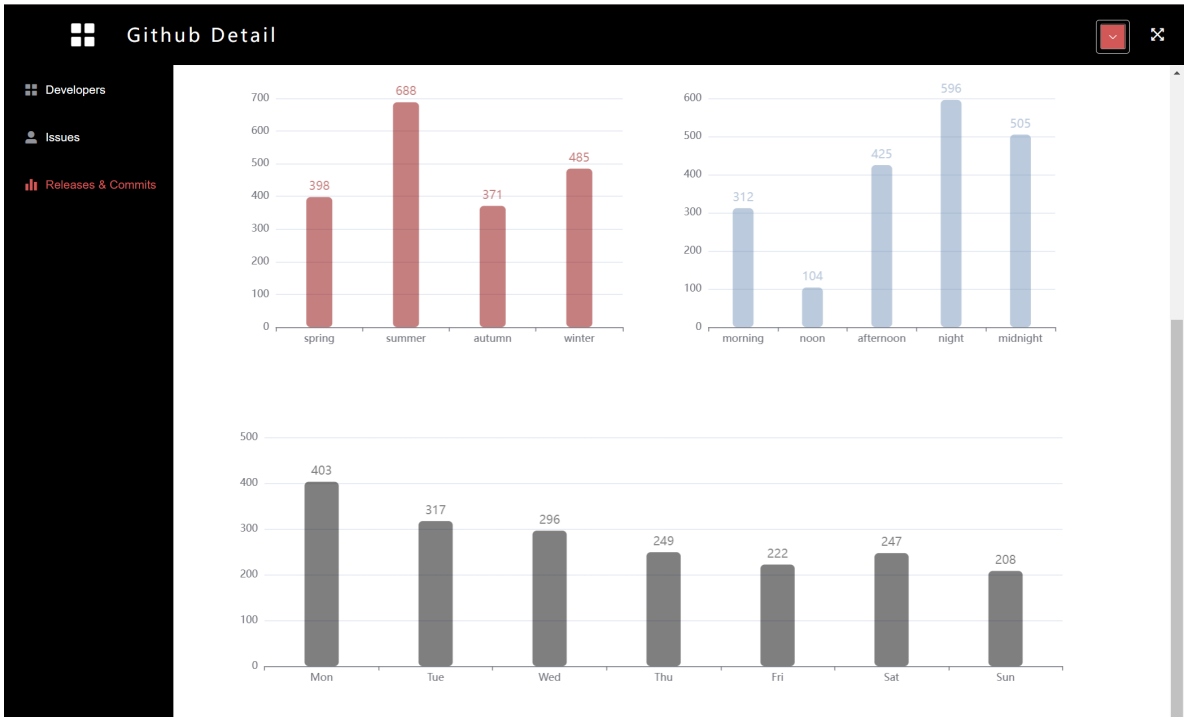


- releases and commits

  As a display page for releases and commits, it displays the total number of releases in visual numbers. Second, the page displays in tabular form the commits since the release (before the next releas), as well as commits since the final release. Moreover, the distribution of the commits time are displayed in bar graphs: quarter, midweek, day, and times of day on the X-axis, and commits are Y-axis.

  1. the total number of releases
  2. the commits after this release (before the next releas) & the commits times since the final release
  3. the distribution of the commits time:quarter, in a week, time frame in a day





Backend

File tree

```
|   .gitignore
|   HELP.md
|   mvnw
|   mvnw.cmd
|   pom.xml
|   SpringBootDemo.iml
|
├─.idea
|   |   .gitignore
|   |   checkstyle-idea.xml
|   |   compiler.xml
|   |   encodings.xml
|   |   jarRepositories.xml
|   |   misc.xml
|   |   modules.xml
|   |   workspace.xml
|   |
|   ├─inspectionProfiles
|   |       Project_Default.xml
|   |
|   └─libraries
|           Maven__ch_qos_logback_logback_classic_1_4_5.xml
|           Maven__ch_qos_logback_logback_core_1_4_5.xml
|           Maven__cn_hutool_hutool_all_5_8_10.xml
|           Maven__com_alibaba_fastjson2_fastjson2_2_0_19.xml
|           Maven__com_alibaba_fastjson2_fastjson2_extension_2_0_19.xml
|           Maven__com_alibaba_fastjson_2_0_19.xml
|           Maven__com_fasterxml_jackson_core_jackson_annotations_2_14_1.xml
|           Maven__com_fasterxml_jackson_core_jackson_core_2_14_1.xml
|           Maven__com_fasterxml_jackson_core_jackson_databind_2_14_1.xml
|
Maven__com_fasterxml_jackson_datatype_jackson_datatype_jdk8_2_14_1.xml
```

```
|
Maven__com_fasterxml_jackson_datatype_jackson_datatype_jsr310_2_14_1.xml
|
Maven__com_fasterxml_jackson_module_jackson_module_parameter_names_2_14_1.xml
|          Maven__com_jayway_jsonpath_json_path_2_7_0.xml
|
Maven__com_vaadin_external_google_android_json_0_0_20131108_vaadin1.xml
|          Maven__io_micrometer_micrometer_commons_1_10_2.xml
|          Maven__io_micrometer_micrometer_observation_1_10_2.xml
|          Maven__jakarta_activation_jakarta_activation_api_2_1_0.xml
|          Maven__jakarta_annotation_jakarta_annotation_api_2_1_1.xml
|          Maven__jakarta_xml_bind_jakarta_xml_bind_api_4_0_0.xml
|          Maven__net_bytebuddy_byte_buddy_1_12_19.xml
|          Maven__net_bytebuddy_byte_buddy_agent_1_12_19.xml
|          Maven__net_minidev_accessors_smart_2_4_8.xml
|          Maven__net_minidev_json_smart_2_4_8.xml
|          Maven__org_apache_logging_log4j_log4j_api_2_19_0.xml
|          Maven__org_apache_logging_log4j_log4j_to_slf4j_2_19_0.xml
|          Maven__org_apache_tomcat_embed_tomcat_embed_core_10_1_1.xml
|          Maven__org_apache_tomcat_embed_tomcat_embed_el_10_1_1.xml
|          Maven__org_apache_tomcat_embed_tomcat_embed_websocket_10_1_1.xml
|          Maven__org_apiguardian_apiguardian_api_1_1_2.xml
|          Maven__org_assertj_assertj_core_3_23_1.xml
|          Maven__org_hamcrest_hamcrest_2_2.xml
|          Maven__org_junit_jupiter_junit_jupiter_5_9_1.xml
|          Maven__org_junit_jupiter_junit_jupiter_api_5_9_1.xml
|          Maven__org_junit_jupiter_junit_jupiter_engine_5_9_1.xml
|          Maven__org_junit_jupiter_junit_jupiter_params_5_9_1.xml
|          Maven__org_junit_platform_junit_platform_commons_1_9_1.xml
|          Maven__org_junit_platform_junit_platform_engine_1_9_1.xml
|          Maven__org_mockito_mockito_core_4_8_1.xml
|          Maven__org_mockito_mockito_junit_jupiter_4_8_1.xml
|          Maven__org_objenesis_objenesis_3_2.xml
|          Maven__org_opentest4j_opentest4j_1_2_0.xml
|          Maven__org_ow2_asm_asm_9_1.xml
|          Maven__org_skyscreamer_jsonassert_1_5_1.xml
|          Maven__org_slf4j_jul_to_slf4j_2_0_4.xml
|          Maven__org_slf4j_slf4j_api_2_0_4.xml
|          Maven__org_springframework_boot_spring_boot_3_0_0.xml
|          Maven__org_springframework_boot_spring_boot_autoconfigure_3_0_0.xml
|          Maven__org_springframework_boot_spring_boot_starter_3_0_0.xml
|          Maven__org_springframework_boot_spring_boot_starter_json_3_0_0.xml
|          Maven__org_springframework_boot_spring_boot_starter_logging_3_0_0.xml
|          Maven__org_springframework_boot_spring_boot_starter_test_3_0_0.xml
|          Maven__org_springframework_boot_spring_boot_starter_tomcat_3_0_0.xml
|          Maven__org_springframework_boot_spring_boot_starter_web_3_0_0.xml
|          Maven__org_springframework_boot_spring_boot_test_3_0_0.xml
|
Maven__org_springframework_boot_spring_boot_test_autoconfigure_3_0_0.xml
|          Maven__org_springframework_spring_aop_6_0_2.xml
|          Maven__org_springframework_spring_beans_6_0_2.xml
|          Maven__org_springframework_spring_context_6_0_2.xml
|          Maven__org_springframework_spring_core_6_0_2.xml
|          Maven__org_springframework_spring_expression_6_0_2.xml
|          Maven__org_springframework_spring_jcl_6_0_2.xml
|          Maven__org_springframework_spring_test_6_0_2.xml
|          Maven__org_springframework_spring_webmvc_6_0_2.xml
|          Maven__org_springframework_spring_web_6_0_2.xml
```

```
|          Maven__org_xmlunit_xmlunit_core_2_9_0.xml
|          Maven__org_yaml_snakeyaml_1_33.xml
|
├─.mvn
|  └─wrapper
|          maven-wrapper.jar
|          maven-wrapper.properties
|
├─src
|  ├─main
|  |  ├─java
|  |  |  └─com
|  |  |      └─example
|  |  |          └─springbootdemo
|  |  |              |  SpringBootDemoApplication.java
|  |  |              |
|  |  |              ├─config
|  |  |              |      CrosConfig.java
|  |  |              |
|  |  |              ├─controller
|  |  |              |      WebController.java
|  |  |              |
|  |  |              ├─model
|  |  |              └─repository
|  |  |                      closed_issues.txt
|  |  |                      commits_data2.txt
|  |  |                      open_issues.txt
|  |  |                      releases.txt
|  |  |
|  |  └─resources
|  |      |  application.properties
|  |      |
|  |      ├─static
|  |      └─templates
|  └─test
|      └─java
|          └─com
|              └─example
|                  └─springbootdemo
|                          SpringBootDemoApplicationTests.java
|
└─target
    ├─classes
    |  |  application.properties
    |  |
    |  └─com
    |      └─example
    |          └─springbootdemo
    |              |  SpringBootDemoApplication.class
    |              |
    |              ├─config
    |              |      CrosConfig$1.class
    |              |      CrosConfig.class
    |              |
    |              └─controller
    |                      WebController.class
    |
    └─generated-sources
```

```
        └─annotations
```

Controller

getCommitters

```
 @GetMapping("/committers")
    public ArrayList<Object> getCommitters() {
        String s =
readFile("D:\\java\\projectf\\SpringBootDemo\\SpringBootDemo\\src\\main\\java\\c
om\\example\\springbootdemo\\repository\\commits_data2.txt");
        List<JSONObject> objects = JSONArray.parseArray(s, JSONObject.class);
        System.out.println(objects.size());
        ArrayList<JSONObject> commits = new ArrayList<>(objects);

        HashMap<String, Integer> committer_Map = new HashMap<>();
        for (JSONObject commit : commits) {
            String name =
commit.getJSONObject("commit").getJSONObject("author").get("name").toString();
            if (committer_Map.containsKey(name)) {
                committer_Map.put(name, committer_Map.get(name) + 1);
            } else {
                committer_Map.put(name, 1);
            }
        }
        List<Map.Entry<String, Integer>> infoIds = new ArrayList<>
(committer_Map.entrySet());
        infoIds.sort((o1, o2) -> {
            return (o2.getValue() - o1.getValue());
            //return (o1.getKey()).toString().compareTo(o2.getKey());
        });
        ArrayList<Developer> developers = new ArrayList<>();
        for (int i = 0; i < 10; i++) {
            for (JSONObject commit: commits) {
                if
(infoIds.get(i).getKey().equals(commit.getJSONObject("commit").getJSONObject("au
thor").get("name").toString())) {
                    String name =
commit.getJSONObject("commit").getJSONObject("author").get("name").toString();
                    String email =
commit.getJSONObject("commit").getJSONObject("author").get("email").toString();
                    if (commit.getJSONObject("author") != null) {
                        String repos_url =
commit.getJSONObject("author").get("repos_url").toString();
                        String id =
commit.getJSONObject("author").get("id").toString();
                        String login =
commit.getJSONObject("author").get("login").toString();
                        Developer developer = new Developer(name, email,
repos_url, id, "https://github.com/" + login +".png");
                        developers.add(developer);
                        break;
                    }

                }
```

```
            }
        }
        ArrayList<Object> arrayList = new ArrayList<>();
        arrayList.add(infoIds.size());
        arrayList.add(developers);
        return arrayList;

    }
```

This method returns the major developers and the total developers in this repo.

getIssues

```
@GetMapping("/issues")
    public ArrayList<Object> getIssues() {
        String s =
readFile("D:\\java\\projectf\\SpringBootDemo\\SpringBootDemo\\src\\main\\java\\c
om\\example\\springbootdemo\\repository\\open_issues.txt");
        List<JSONObject> open_issues = JSONArray.parseArray(s,
JSONObject.class);
        int open_number = open_issues.size();

        String s1 =
readFile("D:\\java\\projectf\\SpringBootDemo\\SpringBootDemo\\src\\main\\java\\c
om\\example\\springbootdemo\\repository\\closed_issues.txt");
        List<JSONObject> closed_issues = JSONArray.parseArray(s1,
JSONObject.class);
        int closed_number = closed_issues.size();

        HashMap<Integer, Integer> resolutionTime_Map = new HashMap<>();
        ArrayList<Long> minuteList = new ArrayList<>();
        for (JSONObject closed_issue : closed_issues) {
            String created_time =
closed_issue.get("created_at").toString().replaceAll("[a-zA-Z]", " ");
            String closed_time =
closed_issue.get("closed_at").toString().replaceAll("[a-zA-Z]", " ");

            SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
            try {
                Date created_date = simpleDateFormat.parse(created_time);
                Date closed_date = simpleDateFormat.parse(closed_time);
                int differentDays = differentDays(created_date, closed_date);
                long difference = closed_date.getTime() -
created_date.getTime();
                long minute_difference = difference / (1000 * 60 * 60 * 24);
                minuteList.add(minute_difference);
                if (resolutionTime_Map.containsKey(differentDays)) {
                    resolutionTime_Map.put(differentDays,
resolutionTime_Map.get(differentDays) + 1);
                } else {
                    resolutionTime_Map.put(differentDays, 1);
                }
            } catch (ParseException e) {
                throw new RuntimeException(e);
            }
```

```
        }
        List<Map.Entry<Integer, Integer>> infoIds = new ArrayList<>
(resolutionTime_Map.entrySet());
        infoIds.sort((o1, o2) -> {
            return (o2.getValue() - o1.getValue());
            //return (o1.getKey()).toString().compareTo(o2.getKey());
        });

        int count = 0;//总个数
        double sum = 0;//总和
        double average;//平均数
        double Dev;//总体方差
        count = minuteList.size();
        for (Long aLong : minuteList) {
            sum += aLong;
        }
        //求平均数
        average = sum/minuteList.size();
        DecimalFormat df = new DecimalFormat(".000");
        double dsum=0;
        for (Long aLong : minuteList) {
            double ss = aLong - average;
            dsum += Math.pow(ss, 2);
        }
        Dev = dsum / (minuteList.size()-1);

        Long max = Collections.max(minuteList);
        Long min = Collections.min(minuteList);
        Long range = max - min;

        HashMap<String, Object> typical = new HashMap<>();
        typical.put("极差", range);
        typical.put("平均值", average);
        typical.put("方差", Dev);

        ArrayList<Object> arrayList = new ArrayList<>();
        arrayList.add(open_number);
        arrayList.add(closed_number);
//        arrayList.add(infoIds);
//        arrayList.add(typical);
        arrayList.add(range);
        arrayList.add(average);
        arrayList.add(Dev);
        return arrayList;
    }
```

This method returns the open number, closed number, range, average, deviation of the whole issues.

getRelease

```
@GetMapping("/releases")
    public ArrayList<Object> getRelease() {
        //获取release次数
```

```
        String releaseString =
readFile("D:\\java\\projectf\\SpringBootDemo\\SpringBootDemo\\src\\main\\java\\c
om\\example\\springbootdemo\\repository\\releases.txt");
        List<JSONObject> releases = JSONArray.parseArray(releaseString,
JSONObject.class);

        //获取每次release之间的commit个数
        String commitString =
readFile("D:\\java\\projectf\\SpringBootDemo\\SpringBootDemo\\src\\main\\java\\c
om\\example\\springbootdemo\\repository\\commits_data2.txt");
        List<JSONObject> commits = JSONArray.parseArray(commitString,
JSONObject.class);
        ArrayList<Integer> commits_between = new ArrayList<>();

        try {
            for (int i = 0; i < releases.size(); i++) {
                SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-
MM-dd HH:mm:ss");
                String release_time =
releases.get(i).get("published_at").toString().replaceAll("[a-zA-Z]", " ");
                Date release_date = simpleDateFormat.parse(release_time);

                int commit_number = 0;
                for (JSONObject commit: commits) {
                    String commit_time =
commit.getJSONObject("commit").getJSONObject("author").get("date").toString().re
placeAll("[a-zA-Z]", " ");
                    Date commit_date = simpleDateFormat.parse(commit_time);
                    if (i == 0) {
                        if (commit_date.after(release_date)) {
                            commit_number++;
                        }
                    } else if (i == releases.size() - 1) {
                        if (commit_date.before(release_date)) {
                            commit_number++;
                        }
                    } else{
                        String last_release_time =
releases.get(i+1).get("published_at").toString().replaceAll("[a-zA-Z]", " ");
                        Date last_release_date =
simpleDateFormat.parse(last_release_time);
                        if (commit_date.before(release_date) &&
commit_date.after(last_release_date)) {
                            commit_number++;
                        }
                    }

                }
                commits_between.add(commit_number);

                if (i == 0) {
                    commit_number = 0;
                    for (JSONObject commit: commits) {
                        String commit_time =
commit.getJSONObject("commit").getJSONObject("author").get("date").toString().re
placeAll("[a-zA-Z]", " ");
                        Date commit_date = simpleDateFormat.parse(commit_time);
```

```
                        String last_release_time =
releases.get(i+1).get("published_at").toString().replaceAll("[a-zA-Z]", " ");
                        Date last_release_date =
simpleDateFormat.parse(last_release_time);
                        if (commit_date.before(release_date) &&
commit_date.after(last_release_date)) {
                            commit_number++;
                        }
                    }
                    commits_between.add(commit_number);
                }

            }
        } catch (ParseException e) {
            throw new RuntimeException(e);
        }

        //获取commit季节时段map
        //获取commit周中时段map
        //获取commit一天中时段map
        HashMap<String, Integer> quarter_commits = new HashMap<>();
        HashMap<String, Integer> week_commits = new HashMap<>();
        HashMap<String, Integer> day_commits = new HashMap<>();
        for (JSONObject commit: commits) {
            SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
            String commit_time =
commit.getJSONObject("commit").getJSONObject("author").get("date").toString().re
placeAll("[a-zA-Z]", " ");
            Date commit_date;
            try {
                commit_date = simpleDateFormat.parse(commit_time);
            } catch (ParseException e) {
                throw new RuntimeException(e);
            }

            //季节
            int commit_Quarter =
Integer.parseInt(DateUtil.yearAndQuarter(commit_date).substring(4));
            switch (commit_Quarter) {
                case 1:
                    if (quarter_commits.containsKey("第一季度")) {
                        quarter_commits.put("第一季度", quarter_commits.get("第一
季度") + 1);
                    } else {
                        quarter_commits.put("第一季度", 1);
                    }
                    break;
                case 2:
                    if (quarter_commits.containsKey("第二季度")) {
                        quarter_commits.put("第二季度", quarter_commits.get("第二
季度") + 1);
                    } else {
                        quarter_commits.put("第二季度", 1);
                    }
                    break;
                case 3:
                    if (quarter_commits.containsKey("第三季度")) {
```

```
                    quarter_commits.put("第三季度", quarter_commits.get("第三季度") + 1);
                } else {
                    quarter_commits.put("第三季度", 1);
                }
                break;
            case 4:
                if (quarter_commits.containsKey("第四季度")) {
                    quarter_commits.put("第四季度", quarter_commits.get("第四季度") + 1);
                } else {
                    quarter_commits.put("第四季度", 1);
                }
                break;
            default:
                System.out.println("出大问题");
        }

        //一周中的时段
        Calendar cal = Calendar.getInstance();
        cal.setTime(commit_date);
        String weekday;
        switch (cal.get(Calendar.DAY_OF_WEEK)) {
            case Calendar.MONDAY:
                weekday = "周一";
                if (week_commits.containsKey(weekday)) {
                    week_commits.put(weekday, week_commits.get(weekday) + 1);
                } else {
                    week_commits.put(weekday, 1);
                }
                break;
            case Calendar.TUESDAY:
                weekday = "周二";
                if (week_commits.containsKey(weekday)) {
                    week_commits.put(weekday, week_commits.get(weekday) + 1);
                } else {
                    week_commits.put(weekday, 1);
                }
                break;
            case Calendar.WEDNESDAY:
                weekday = "周三";
                if (week_commits.containsKey(weekday)) {
                    week_commits.put(weekday, week_commits.get(weekday) + 1);
                } else {
                    week_commits.put(weekday, 1);
                }
                break;
            case Calendar.THURSDAY:
                weekday = "周四";
                if (week_commits.containsKey(weekday)) {
                    week_commits.put(weekday, week_commits.get(weekday) + 1);
                } else {
                    week_commits.put(weekday, 1);
                }
```

```
                    break;
                case Calendar.FRIDAY:
                    weekday = "周五";
                    if (week_commits.containsKey(weekday)) {
                        week_commits.put(weekday, week_commits.get(weekday) +
1);
                    } else {
                        week_commits.put(weekday, 1);
                    }
                    break;
                case Calendar.SATURDAY:
                    weekday = "周六";
                    if (week_commits.containsKey(weekday)) {
                        week_commits.put(weekday, week_commits.get(weekday) +
1);
                    } else {
                        week_commits.put(weekday, 1);
                    }
                    break;
                case Calendar.SUNDAY:
                    weekday = "周日";
                    if (week_commits.containsKey(weekday)) {
                        week_commits.put(weekday, week_commits.get(weekday) +
1);
                    } else {
                        week_commits.put(weekday, 1);
                    }
                    break;
                default:
                    System.out.println("日期出问题");

            }

            //一天中时间段
            SimpleDateFormat df = new SimpleDateFormat("HH");
            String str = df.format(commit_date);
            int a = Integer.parseInt(str);
            if (a >= 0 && a <= 6) {
                String day = "凌晨";
                if (day_commits.containsKey(day)) {
                    day_commits.put(day, day_commits.get(day) + 1);
                } else {
                    day_commits.put(day, 1);
                }
            }
            if (a > 6 && a < 12) {
                String day = "上午";
                if (day_commits.containsKey(day)) {
                    day_commits.put(day, day_commits.get(day) + 1);
                } else {
                    day_commits.put(day, 1);
                }
            }
            if (a >= 12 && a <= 13) {
                String day = "中午";
                if (day_commits.containsKey(day)) {
                    day_commits.put(day, day_commits.get(day) + 1);
                } else {
```

```
                day_commits.put(day, 1);
            }
        }
        if (a > 13 && a <= 18) {
            String day = "下午";
            if (day_commits.containsKey(day)) {
                day_commits.put(day, day_commits.get(day) + 1);
            } else {
                day_commits.put(day, 1);
            }
        }
        if (a > 18 && a <= 24) {
            String day = "晚上";
            if (day_commits.containsKey(day)) {
                day_commits.put(day, day_commits.get(day) + 1);
            } else {
                day_commits.put(day, 1);
            }
        }

    }


    ArrayList<Integer> yData=new ArrayList<>();
    yData.add(quarter_commits.get("第一季度"));
    yData.add(quarter_commits.get("第二季度"));
    yData.add(quarter_commits.get("第三季度"));
    yData.add(quarter_commits.get("第四季度"));

    ArrayList<Integer> week=new ArrayList<>();
    week.add(week_commits.get("周一"));
    week.add(week_commits.get("周二"));
    week.add(week_commits.get("周三"));
    week.add(week_commits.get("周四"));
    week.add(week_commits.get("周五"));
    week.add(week_commits.get("周六"));
    week.add(week_commits.get("周日"));


    ArrayList<Integer> day=new ArrayList<>();
    day.add(day_commits.get("上午"));
    day.add(day_commits.get("中午"));
    day.add(day_commits.get("下午"));
    day.add(day_commits.get("晚上"));
    day.add(day_commits.get("凌晨"));


    ArrayList<Object> list = new ArrayList<>();
    list.add(releases.size());
    list.add(commits_between);
    list.add(yData);
    list.add(week);
    list.add(day);
    return list;
}
```

This method returns the releases times, commits between two issues, the quarter commits, the week day commits and the day commits.

GitHub REST API

To create integrations, retrieve data, and automate your workflows, build with the GitHub REST API.

When you make a request to the REST API, you will specify an HTTP method and a path. Additionally, you might also specify request headers and path, query, or body parameters. The API will return the response status code, response headers, and potentially a response body.

File Tree

```
D:.
|  Data.iml
|  pom.xml
|  WebCrawler.iml
|
├─.idea
|  |  .gitignore
|  |  checkstyle-idea.xml
|  |  compiler.xml
|  |  encodings.xml
|  |  jarRepositories.xml
|  |  misc.xml
|  |  uiDesigner.xml
|  |  workspace.xml
|  |
|  └artifacts
|          Data_jar.xml
|
├─Data
|  |  pom.xml
|  |
|  ├─src
|  |  ├─main
|  |  |  ├─java
|  |  |  |      Data.java
|  |  |  |      demo.java
|  |  |  |      Developer.java
|  |  |  |
|  |  |  └─resources
|  |  |      └─lib
|  |  |              fastjson-2.0.20.graal.jar
|  |  |              json-lib-1.1-jdk13.jar
|  |  |              json-simple-1.1.1.jar
|  |  |
|  |  └─test
|  |      └─java
|  └─target
|      ├─classes
|      |  |  Data.class
|      |  |  demo.class
|      |  |  Developer.class
```

```
|       |   |
|       |   └─lib
|       |           fastjson-2.0.20.graal.jar
|       |           json-lib-1.1-jdk13.jar
|       |           json-simple-1.1.1.jar
|       |
|       └─generated-sources
|           └─annotations
├─DataFile
|       closed_issues.txt
|       commits_data.txt
|       commits_data2.txt
|       issues_comments.txt
|       open_issues.txt
|       releases.txt
|
├─DataFile2
|       closed_issues.txt
|       commits_data2.txt
|       issues_comments.txt
|       open_issues.txt
|       releases.txt
|
└─src
    ├─main
    |   ├─java
    |   └─resources
    └─test
        └─java
```

Example Usage

framework and ideas

We get the returned json statement through the url request of GitHub's REST API. The parameters are per_page and page. The json statement is formatted using the formatjson method and stored in some files.

Example Usage

Search in GitHub

```
         public static void getJson() {
        int page = 1;
        int number;
        do {
            String url =
"https://api.github.com/repos/socket.io/socketio/releases?
per_page=100&page="+page;
            URL oracle;
            try {
                StringBuilder sb = new StringBuilder();
                oracle = new URL(url);
```

```
                HttpURLConnection conn = (HttpURLConnection)
oracle.openConnection();
                conn.setRequestProperty("Authorization", "token
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");
                BufferedReader br = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
                String inputLine;

                while((inputLine = br.readLine()) != null){
                    sb.append(inputLine);
                }

                String s = sb.toString();
                String s1 = formatJson(s);
                System.out.println(s1);

                FileWriter fileWriter = new
FileWriter("DataFile/releases.txt",true);
                fileWriter.write(s1);
                fileWriter.close();

                number = JSONArray.parseArray(s1, JSONObject.class).size();
                page++;
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        } while (number != 0);
    }

public static String formatJson(String str) {
        Object parse;
        try {
            parse = JSON.parse(str);
        } catch (Exception e) {
            str = replaceSpecial(str);
            parse = JSON.parse(str);
        }

        if (parse instanceof JSONArray) {
            return JSON.toJSONString(JSONArray.parseArray(str),
SerializerFeature.PrettyFormat);

        } else if (parse instanceof JSONObject) {
            return JSON.toJSONString(JSONObject.parseObject(str),
SerializerFeature.PrettyFormat);
        } else {

            return str;
        }
    }

public static String replaceSpecial(String str) {
        List<String> specialList = new ArrayList<>();

 Collections.addAll(specialList,"a","b","f","n","t","v","r","s","w","0","?","'",
"\"");
        char[] charArray = str.toCharArray();
        StringBuilder sb = new StringBuilder();
```

```
        for (int i = 0; i < charArray.length; i++) {
            if ("\\".equals(String.valueOf(charArray[i]))) {
                if (specialList.contains(String.valueOf(charArray[i+1]))) {
                    sb.append(charArray[i]);
                    continue;
                }
                if (getSlashNumberBefore(sb.toString(), i)%2!=0) {
                    sb.append(charArray[i]).append(charArray[i]);
                } else {
                    sb.append(charArray[i]);
                }
            } else {
                sb.append(charArray[i]);
            }
        }
        return sb.toString();
    }

private static int getSlashNumberBefore(String beforeStr, int i) {
        if (beforeStr.length() < i) {
            return i;
        }
        if ("\\".equals(beforeStr.substring(beforeStr.length() -i,
beforeStr.length() -i+1))) {
            i++;
            i = getSlashNumberBefore(beforeStr, i);
        }
        return i;
    }
```