Architecture

Documented Architecture:

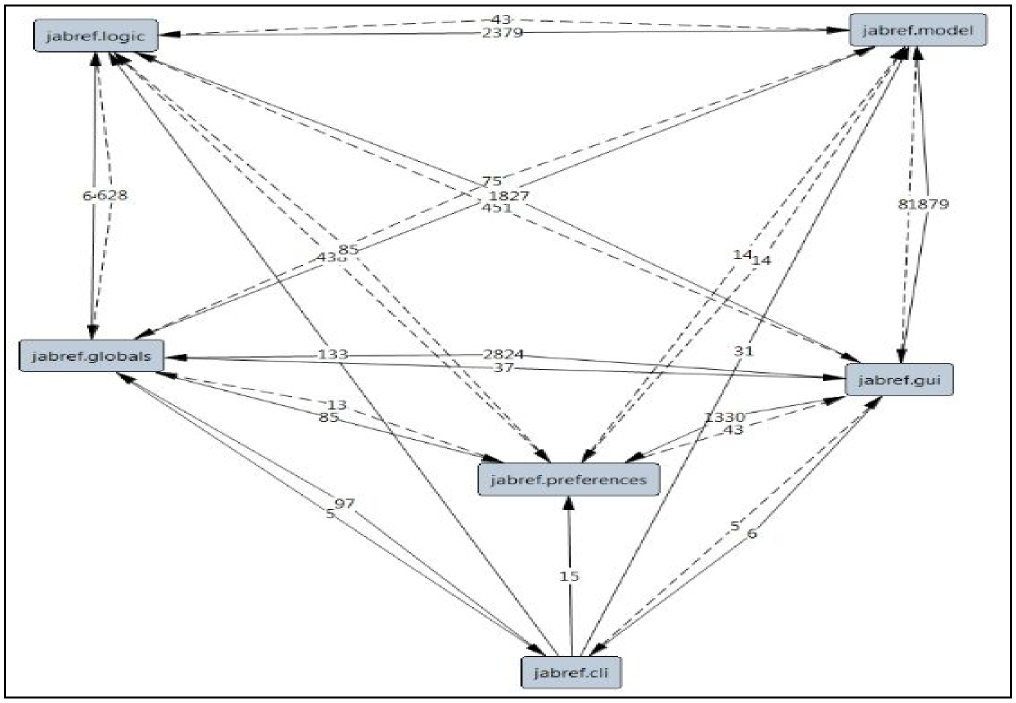


Figure (1) : Desired Architecture of JabRef

The desired architecture discussed by JabRef developers goes back to 2016[1] before which reportedly JabRef lacked architecture (Started development in 2003). The initial desired architecture is as indicated in Figure (1). Dashed arrows indicate forbidden dependencies, and complete arrows indicate allowed dependencies. The core content consists of three components: model, logic, and GUI accounting for MVC architecture. GUI is the outer layer that knows what the user is doing and makes interactions with users. Logic is the middle layer for dealing with input and output of data and manipulating models. The model is in the inner center and stores data like BibDatases, BibEntries, and Events. The dependencies are only directed from the outer layer towards the center. The model does not depend on other classes, logic only depends on the model. Globals hold front-end frameworks and global information. There are some utility packages including preferences and cli. For example, preferences represent the user's customized needs and usages. For this stage, the only allowed dependencies are the following:

```
jabref.logic -> jabref.model
jabref.globals -> everywhere
jabref.gui -> everywhere
jabref.cli -> jabref.preferences
jabref.cli -> jabref.globals
jabref.cli -> jabref.logic
jabref.cli -> jabref.model
```

As-implemented Architecture:

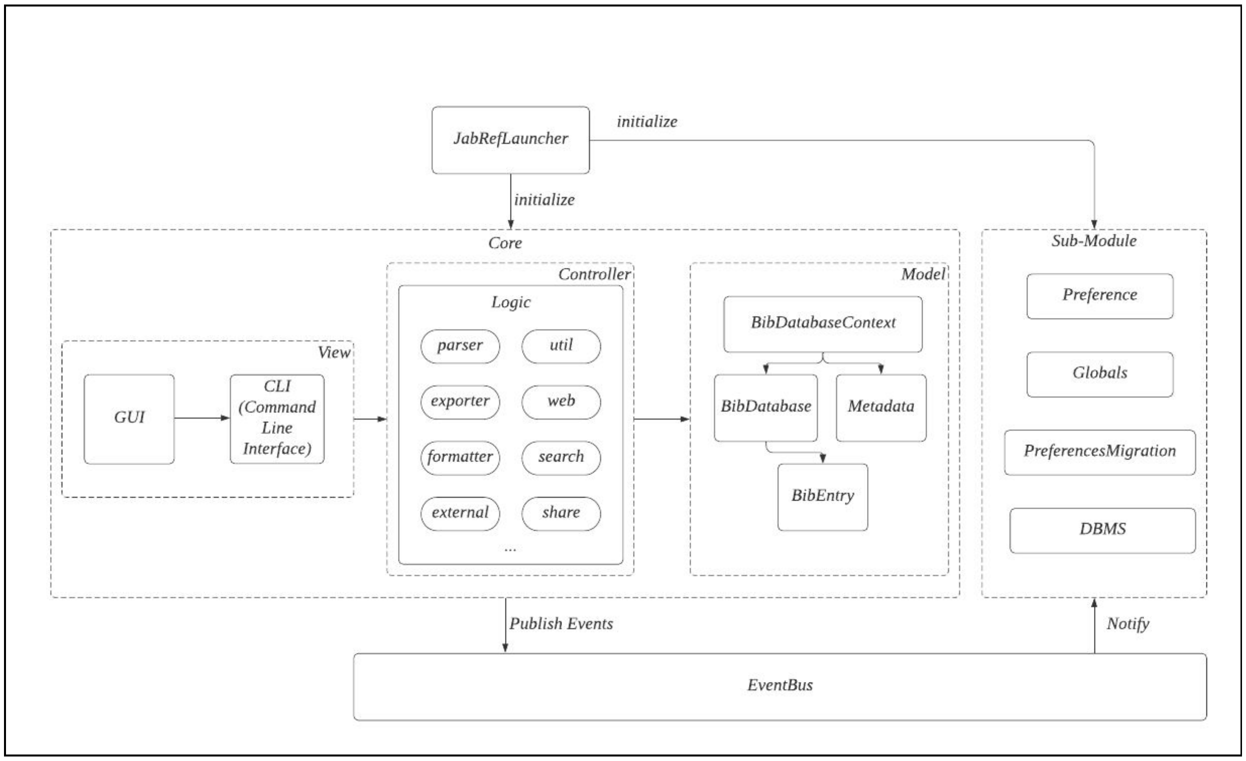
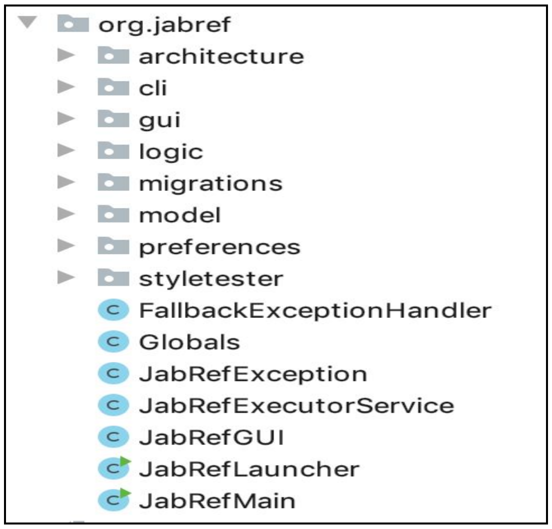


Figure (2) : As-Implemented Architecture of JabRef

To make our architectural diagram in figure (2), we checked the file directory [refer structure to the right], to begin with. Then, for each folder, we open the related files and try to find the purpose of the classes. In conjunction, we referred to the official document to see whether the As-implemented architecture follows the original design intended.



```
org.jabref
    architecture
    cli
    gui
    logic
    migrations
    model
    preferences
    styletester
    FallbackExceptionHandler
    Globals
    JabRefException
    JabRefExecutorService
    JabRefGUI
    JabRefLauncher
    JabRefMain
```

One possible way to ensure the control flow is to check the call graph. For example, in the package org.jabref.model.entry, there is BibEntry class, which is a basic class for the database model. We generated the call graph for a method in this class. We also set the node colors differently by class name. From figure (3), the blue highlighted box represents the functions that are in the GUI package, the red highlighted box represents the Logic part where all of the classes are present in the logic package. Finally, the yellow highlighted part contains the data operations. In this way, we realize that the control flow is GUI → Logic → Data Model, which is similar to the traditional MVC architecture.

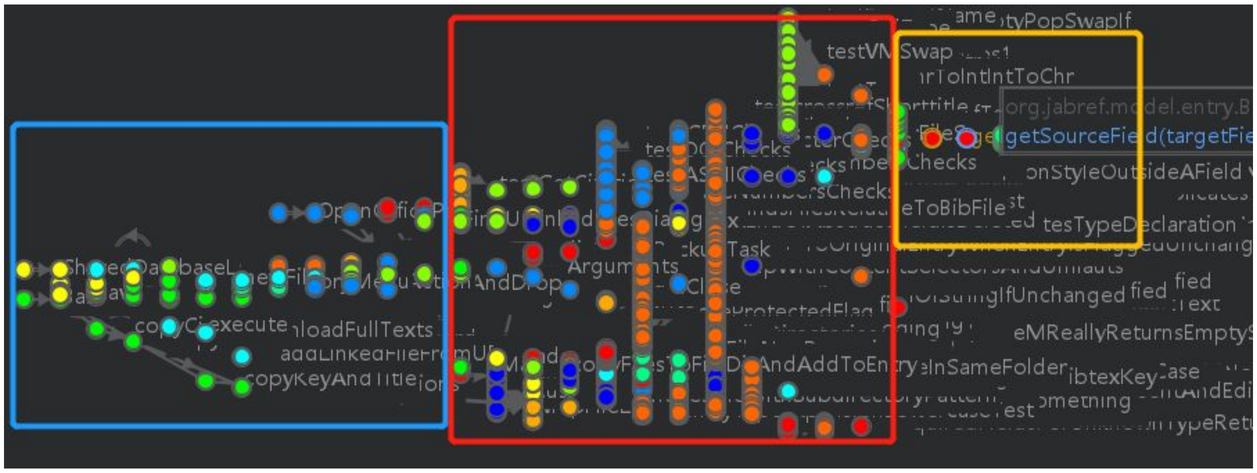


Figure (3) : Call graph for a method in BibEntry Class

We analyze other folders, in the same way, where we found annotations in the code to be very useful. For example, when we check the JabRefMain.java, there is an annotation that says “JabRef's main class to process command-line options and to start the UI”. JabRefMain is called in JabRefLauncher which is the start of the program and helped us in identifying the initialiser for the diagram.

Command Line Interface(CLI) plays as a bridge between GUI and Logic functionality. When we check the CLI, we find several operation functions' definitions, such as importFile, saveDatabase, exportFile, etc. We trace down and found that there is usually an object, which is defined in the Logic package, included in those functions. For example, importFile function has objects named ParserResult and ImportFormatReader, which is defined in the Logic package and contains specific methods according to different operations. Hence, it helped us in designing the Controller of the architectural diagram.

There are many reasons that we set eventbus as one of the components in the architecture. Firstly, from the previous homework, we know that eventbus is widely used in the system. Secondly, when we go through the code, we find eventbus is the main synchronization tool that database management system uses to synchronize local data changes. This information served crucially in defining connections.

Hence, in conclusion, we agree with JabRef following the MVC pattern as intended by the developers but there are more dependency relations[2] than a traditional MVC among the different components in JabRef. For instance, GUI can communicate with Logic which is the controller and Model and hence modify the Model by passing over Logic, this violates the traditional MVC. Next, Model contains the related data structures and is controlled by the controller (Logic) and does not interact with any other components. Some parts related to the logic are found in minor portions of the Model. Therefore, it is not purely present in the logic component which again violates traditional MVC. Finally, the Logic component which is the controller is responsible for fetching, handling imports, formatting and logging error details.

<u>Social Context</u>

State of the project:

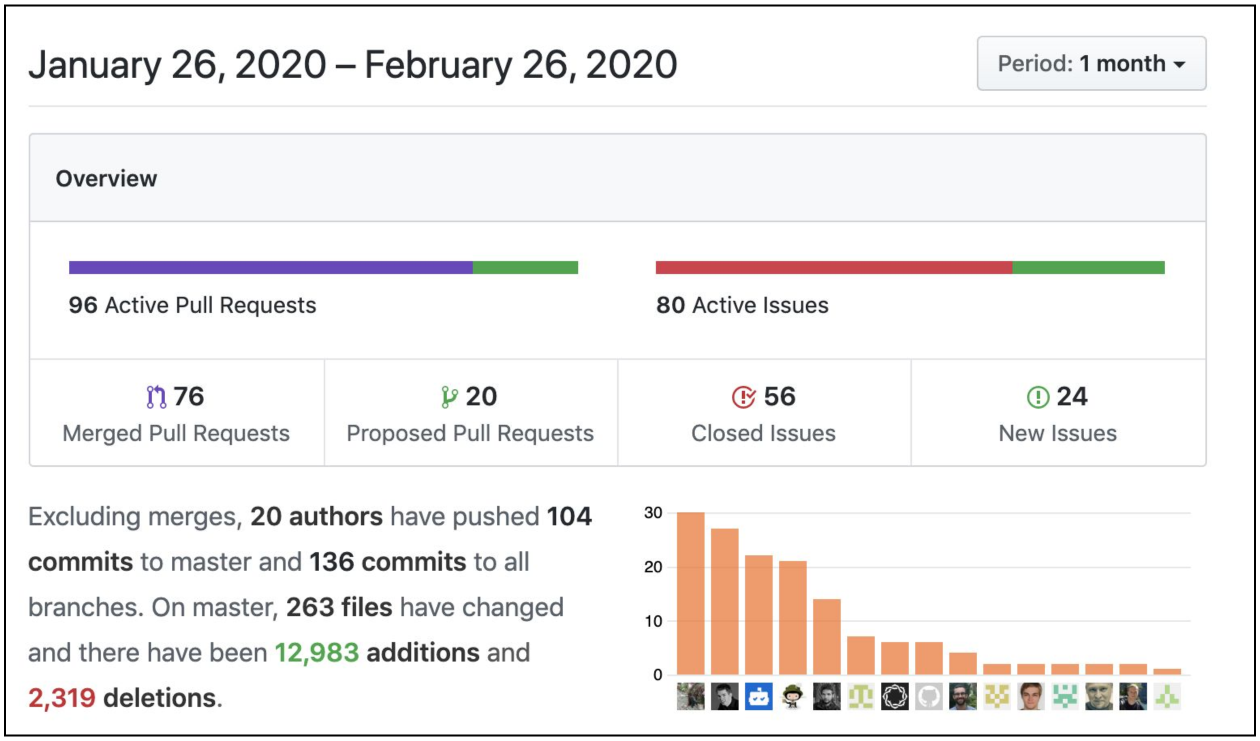
As stated in our previous report, the main users of JabRef are in Universities as it is a Reference Management Software. According to a survey[3] that JabRef's core development team carried out in 2015, we understand that German and English speakers are the main users of this tool. Upon further inquiry about the usage, it was found that more than 50% of the responders mainly used JabRef for professional purposes followed by PhD study purposes. The last category accounts for personal usage. Regarding the dominating field, we noticed that the academics in natural sciences used JabRef more, mainly in research areas concerning Physics and Biology. This was followed by formal sciences such as Computer Science and Mathematics. Least usage was found among the Humanities side in academia. Jabref is used as the primary software for Bibliography purposes for about 80% of the participants. The rest used it in conjunction with other software or did not use JabRef. JabRef is used mainly for Tex-related purposes for 80% of the participants and 10% use JabRef with word processors.

Jabref has been a project under development since November 2003. The latest available version of JabRef is Jabref 5.0 Beta which was released on December 15, 2019. The roadmap[4] accounting for the latest releases to understand general pattern is as follows:

Table (1) : Roadmap of JabRef

Version	**Date**
JabRef 5.0 Beta	12/15/2019
JabRef 5.0 Alpha	08/27/2019
JabRef 4.2	04/26/2018
JabRef 4.1	12/23/2017
JabRef 4.0	10/04/2017
JabRef 3.5	07/14/2016
JabRef 3.3	04/19/2016

As noted from Table (1), JabRef is an active project constantly being updated. This is also evident from active developers merging Pull requests, typically within two days.



January 26, 2020 – February 26, 2020

Period: **1 month**

Overview

96 Active Pull Requests

80 Active Issues

76 Merged Pull Requests

20 Proposed Pull Requests

56 Closed Issues

24 New Issues

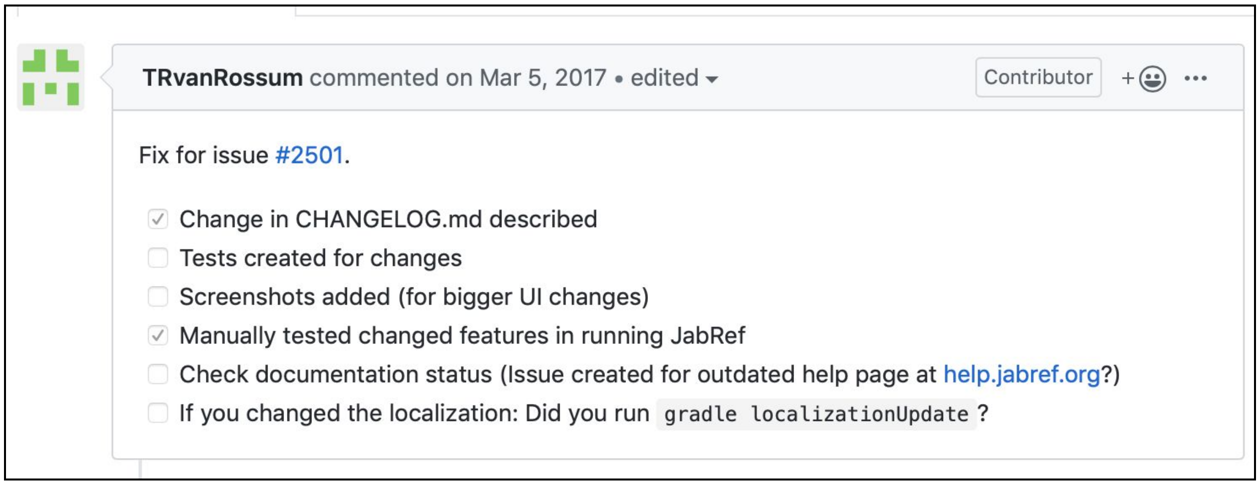
Excluding merges, **20 authors** have pushed **104 commits** to master and **136 commits** to all branches. On master, **263 files** have changed and there have been **12,983 additions** and **2,319 deletions**.

Figure (4) : JabRef Pulse[5] (01/26/2020 - 02/26/2020)

Among 8 of the core developers, we observed 6 of them making active commits over the past one month. The active developers in the order of commits are Oliver Kopp, Christoph Schwentker, Carl Christian Snethlage, Stefan Kolb, Tobias Diez and Linus Dietz. As observed from Figure (4), there have been about 136 commits to all branches and 24 new issues in the past one month.

Coding Standardization:
We found relevant details presented in this section with the help of JabRef's guide[6]. Hence, the developers interested in making such contributions should stick to the specifications mentioned in this manual. We summarize some of the interesting points:

- JabRef recommends sticking to general coding standards as mentioned in this GitHub page.
- JabRef prohibits throw and catch of "Exception" or "Throwable". Instead, forces the developer to extend JabRefException as this implements getLocalizedMessage() which aids in showing the user all the error messages.
- Users should not see the technical details behind the error in the UI and only a concise message. The errors should be logged instead.
- When using libraries, as Jabref is a huge project and there are tons of dependencies, it is expected to be added to the gradle file to make smooth transition, that is preferably use a version from ***jCenter***.
- ***Automated checklist in GitHub*** is used to evaluate Pull Request which makes understanding easier from both ends. (see figure (5))



Fix for issue #2501.

- [x] Change in CHANGELOG.md described
- [ ] Tests created for changes
- [ ] Screenshots added (for bigger UI changes)
- [x] Manually tested changed features in running JabRef
- [ ] Check documentation status (Issue created for outdated help page at help.jabref.org?)
- [ ] If you changed the localization: Did you run `gradle localizationUpdate`?

Figure (5) : Example of JabRef Pull Request's checklist

Testing Standardization:

- Testing is mainly conducted through ***jUnit 4*** which is a Java framework for Unit tests. The developers advise contributors to stick on not using the Test prefix as this is addressed in the class name. The test case name should have the entire description of the test. Each test case should strictly address only a part of the method in the class and test single functionality.
- ***Codacy*** is used to test for code coverage as well as feedback on the practice of coding styles.
- To facilitate Continuous Integration, ***travis CI*** is being used to perform tests in the system. ***CircleCI*** is responsible for generating binaries and to account for builds.

Submission Standards / Contribution:
Upon reading the guide[7], we found the following general sequence:

- Add change to CHANGELOG.md
- In order to facilitate author credits, the contributor is expected to run the script along with their pull request submission scripts/generate-authors.sh
- Recommends to write a good commit message as the first line acts as the name of the pull request and should ideally follow “fixes xyz”.
- It is essential to write test cases for the change we made.
- There are several pieces of information found regarding adding a library, making an architectural design and adding a new localization.lang entry in the document[7].
- Finally, make the pull request.

As indicated in the first point, CHANGELOG.md holds all the changes documented, this adheres to the **documentation standards** followed in the project. The documents holds the information under three categories, “Changed”, “Fixed” and “Removed” for functionalities. These are suffixed with GitHub issue numbers at the end of each log. (*Note that the interesting tools used were bolded in italics along the course of description in the section)*

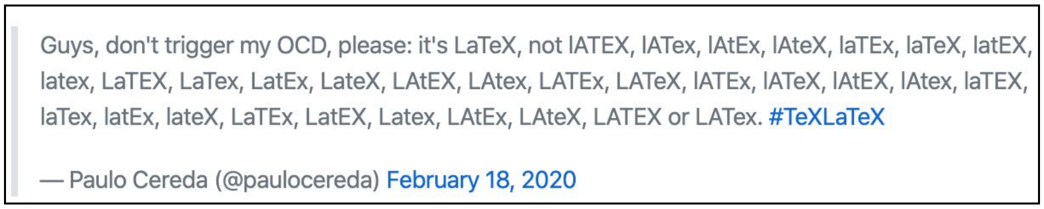
“Interesting” pull requests

1. **Restructure and improve docs for setting up IntelliJ** (PR #5960)
   The submitter modified one README file for setting up a local workspace, specifically for Intellij. The conversation between the contributor and the reviewer was simple. The reviewer said, "it was great stuff and would merge it to keep

things going". What we found interesting is that there were some required status checks for this commit. The branch must pass all the checks to be merged. Not every branch comes with checks, only protected branches selected by maintainers are enabled with required status checks (according to GitHub Help page). It was interesting to learn about these conditions.

2. **LaTeX is LaTeX - and not LaTex** [(PR #5978)]

   One of the core developers made this pull request to fix the casing of "LaTeX" to either "Latex" for code or "LaTeX" for code comments and user-facing. It made us laugh that the developer also quoted a very funny tweet (see below). We appreciate the sense of humor in this request.



> Guys, don't trigger my OCD, please: it's LaTeX, not lATEX, lATex, lAtEx, lAteX, laTEx, laTeX, latEX, latex, LaTEX, LaTex, LatEx, LateX, LAtEX, LAtex, LATEx, LATeX, lATEx, lATeX, lAtEX, lAtex, laTEX, laTex, latEx, lateX, LaTEx, LatEX, Latex, LAtEx, LAteX, LATEX or LATex. #TeXLaTeX
>
> — Paulo Cereda (@paulocereda) February 18, 2020

3. **Update build.gradle** [(PR #4967)]

   The submitter added a plugin(SonarQube) in build.gradle file, and listed several tasks to be checked off for making a formal pull request. However, there was zero context about why the submitter wanted to add this plugin. The reviewer enquired about the intention behind this pull request and mentioned in his comments that JabRef was not going to use SonarQue. The reviewer also labeled this pull request with "waiting-for-customer-feedback". Later on, the submitter replied, "Thanks" and the pull request was closed after that. This request is an interesting one because it seemed to be random and didn't follow the contribution policy of JabRef. Moreover, without context information provided, the pull request itself was hard to interpret.

4. **New Crowdin translations** [(PR #5990)]

   One of the core developers committed multiple times for adding multiple language options that utilize Crowdin. Crowdin is a closed source cloud-based company providing localization solutions for multilingual content. JabRef supports more than 20 languages for users' convenience. It is interesting to know that JabRef uses Crowdin services in which texts can be translated and proofread by linguists.

5. **Welcome @calixtus** [(PR #5822)]

This pull request serves as a welcome board for welcoming the new developer to join the JabRef team. It is marked with ❤ and 🚀Emoji, which is lovely. Also, it is interesting to notice the variety of pull request usages like this one.

“Interesting” issues

1. **High-level tutorial for getting started with JabRef development** (issue#5895)

What we find interesting is that this issue is not about the system itself, instead, this user is asking for a tutorial of JabRef. Developers do provide some instructions, but they are kind of abstract. Hence, although this issue seems strange, it seems reasonable.

2. **ScienceDirect Fulltext download does not work** (issue#5860)

ScienceDirect is one of the dependencies on which JabRef relies. It is interesting that one of the core developer's commented on this issue, stating that the bug cannot be fixed due to ScienceDirect’s security policies. The user who pointed out this issue replied with “That's too bad”.

3. **Font size is too small according to ISO 9241-303** (issue#4582)

What attracted us to the issue is ISO 9241-303. On looking up "ISO 9241-303" in Google, we understood that it stands for the Ergonomics of human-system interaction, that’s a professional expression! Another interesting factor in this issue is that one user says that it is difficult to read the Main Table entries on a 4K bright screen. Now, everyone gets to know he has a nice screen!

4. **MySQL connection exception** (issue#5886)

This user has some problems with MySQL connection. The issue itself is not that interesting, however, Koppor, who is the No.1 core developer of JabRef, sends a comment and asks this user, “Why are you using MySQL?”. Seems like Koppor dislikes MySQL!

5. **Find entries linked to a specific file** (issue#5052)

This user wants to attach one file to several different files. However, one of the core developers does not seem to understand the essence of this behavior, as noted from his comment “Why would you want to know this?”. Further explanation can be found in the issue. This is interesting because even though developers create a system from the bottom, sometimes they have not looked at all the possible usages of the system and they cannot understand why people are using the system in a specific way.

References

[1] JabRef Architecture, Link, Retrieved 2020-02-23.
[2] Delft Students on Software Architecture, *Delft University of Technology,* ***2017***, Link, Retrieved 2020-02-23.
[3] JabRef Survey Analysis, Link, Retrieved 2020-02-25.
[4] JabRef Blog, Link, Retrieved 2020-02-25.
[5] JabRef GitHub, Link, Retrieved 2020-02-25.
[6] JabRef CodeHowTos, Link, Retrieved 2020-02-26.
[7] JabRef contributions, Link, Retrieved 2020-02-26.
[8] JabRef Pull requests Github, Link, Retrieved 2020-02-23.
[9] JabRef Issues Github, Link, Retrieved 2020-02-23.