NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Deliverable Phase 2

Software Engineering
Scrum Team #28

Clara Sousa 58403 Paula Inês Lopes 58655 Pedro Reis 58751
Ricardo Pereira 57912 Rita Silva 57960

December 30, 2021

Contents

1 Document Description and Overall Review

It is the aim of this document to merge all the documentation produced by the Scrum Team throughout the 4 sprints during which the second phase of the project took place.

The development of this second phase started in Sprint 5, right after the delivery of the first one. Plans regarding schedules and work days and hours were made, and the ideas regarding the new functionalities to be added to JabRef were debated. In general, the S Team succeed to fulfill the plans of each sprint and the corresponding days, with the exception of one of the tasks in sprint 7, which was continued in the following sprint.

As to the development *per se*, the team added two brand new functionalities to JabRef: now, it is possible for users to import libraries from csv files and add new bibliographic references from a given author to their currently opened library - a functionality created using Google Scholar. These new functionalities were fully implemented and tests were created for them. A demonstration video can be found here.

A relevant aspect to mention is that during this second phase, the development of new functionalities was nearly always performed with all members of the S Team, ignoring the firstly created plan of dividing the team into 2 different sub-teams whose members whould exchange every sprint. The decision to work on the development stage as an "all together" situation was made with the intention to improve the team's productivity, since it became faster to spot mistakes and correct them, as well as prevent code smells.

In the team's GitHub repository, all Agile-related documentation can be found in the Agile Documents Compilation. The presented Work Breakdown Structure and Scrum Board documents correspond to their status at the end of each sprint, even though they were often changed several times within each sprint, which can be seen by those who analyse the GitHub history track.

2 Functional Requirements Analysis

The goal of this section is to mirror the features in a Requirements Analysis. Here, we present the new functionalities to be added using user stories, and the identification of authors and use cases is added, with a further description of the latter, so as to complete the requirements in the project guide.

2.1 Importing Bibliographic References from a CSV File

2.1.1 User Stories and Epics

Epic: "Import library from a CSV file."

User Stories:

- "As an author I want to import a library from a CSV file (sorted by author) so I can have access to the bibliographic references inside that file in my JabRef application."
- "As an author I want to click on the button "File" so that I can see the available options related to the management of files."
- "As an author I want to click on the button "Import" so that I can see the available options related to the management of imports."
- "As an author I want to click on the button "Import into current library" so that I can import the bibliographic references into the currently open library."
- "As an author I click on the csv file that I want to open so that I can use its bibliographic references in my JabRef application."

2.1.2 Involved Use Cases

The aim of this new functionality is to allow users to import bibliograhic references in existing csv files into their JabRef application.

Therefore, by looking at the use cases that compose JabRef, we can see that the new functionality allows the secondary actor "Out of system library" in the "Adding existing library" sub-use case to be a comma-separated values type of file.

The use cases involved in the implementations are "Add a Library to the Program" and "Add Existing Library" (sub-use case).

Add a Library to the Program

Identification of Actors

- Author: a user of JabRef. Characteristics: human, active.
- Out of system library: a library that already exists outside of the author's JabRef application. Characteristics: non-human, passive.

Use Case Description

- **Use Case Name**: Add a Library to the Program
- **Description**: An author adds a library to their JabRef application
- **Main Actor**: Author
- **Secondary Actor**: Out of system library

Add Existing Library

Identification of Actors

- Author: a user of JabRef. Characteristics: human, active.
- Out of system library: a library that already exists outside of the author's JabRef application. Characteristics: non-human, passive.

Use Case Description

- **Use Case Name**: Add Existing Library
- **Description**: An author adds a library that exists outside of the system (eg.: in the author's desktop) to their JabRef application
- **Main Actor**: Author
- **Secondary Actor**: Out of system library

2.2 Get More Bibliographic References from a Specific Author

2.2.1 User Stories and Epics

Epic: "Get more bibliographic references from a certain author."

User Stories:

- "As an author I want to be able to click on given a bibliographic reference in the currently open library so that I can find bibliographic references from the author correspondent to that reference."
- "As an author I want to right click on a bibliographic reference so that I can see the functionalities regarding that reference."
- "As an author I want to click on the button "Search Google Scholar" so that I can import ten other bibliographic references existent in Google Scholar from this author."

2.2.2 Involved Use Cases

With this new functionality, a user can click on a given bibliographic reference and choose to find more references with the same author of the clicked entry.

At first, the team thought a Google Scholar API could be used to find these references, but we ended up not needing it and using a scrapping library called **Jsoup**.

From the use case perspective, it is as if though a new behaviour is being added to the automatic addition of entries in the currently open library.

As a result, the use cases involved in the implementation of this new functionality are "Add entries" and "From Google Scholar" - a new sub-use case that establishes a generalization type of relationship with "Add entries".

Add entries

Identification of Actors:

- Author: a user of JabRef. Characteristics: human, active.
- Web: online scientific catalogues like CrossRef, Google Scholar or IEEEXplore. It can also represent online databases. Characteristics: non-human, passive.

Use Case Description

- **Use Case Name**: Add an Entry to the Program
- **Description**: An author collects entries automatically or manually to their JabRef application.
- **Main Actor**: Author
- **Secondary Actor**: Web

From Google Scholar

Identification of Actors:

- Author: a user of JabRef. Characteristics: human, active.
- Google Scholar: An online database with papers, articles, books, etc. Characteristics: non-human, passive.

Use Case Description

- **Use Case Name**: From Google Scholar
- **Description**: An author adds entries from a given author of books, papers, articles, etc., to their JabRef application.
- **Main Actor**: Author
- **Secondary Actor**: Google Scholar

2.3 Task management

In this section the team presents the management of tasks regarding the following sprints, based on the previously mentioned user stories.

TASK ID	TASK DESCRIPTION	RELATED FUNCTIONALITY
I1	Creating the new feature in JabRef	Import library from a CSV file
I2	Creating the button for the feature	
I3	Creating J-Unit tests	
I4	Creating demonstration video	
J1	Creating the new feature in JabRef	Get more bibliographic references from a certain author
J2	Creating the button for the feature	
J3	Creating J-Unit tests	
J4	Creating demonstration video	

3 Additional Notes

The purpose of this section is to provide complementary information regarding the development of the new functionalities.

3.1 Importing Bibliographic References from a CSV File

The goal of this implementation is to allow users to import bibliographic references from files with a csv extension.

In order to so, all a user must now do is click on the button "File", followed by "Import", and then select "Import into current library". Finally, the user must select which file and format they wish to import into the currently opened library (and this format can now be a csv extension!). The user can now find the references in their library, alphabetically sorted according to the first name of the author.

In addition to fulfilling the requirements established, the team also ended up doing a bit more.

As a matter of fact, the user can also select "any file" as the format, when choosing a file, and the program works just as well. Moreover, it is also possible to select to import the references in the file into a new library instead of importing them into the current one - the user must only select the option "Import into new library" in the options of "Import".

It is also important to mention that, in order to be able to import bibliographic references from a file, the team had to create an "expected format". This format corresponds to a specific order of fields; in this case, it would be *Entry Type, Author, Title, Year, Journal*. If a file has no references in this expected format, than no references are imported and a warning is presented to the user. If, however, a file has some references with the expected format and others without it, then the program will import the ones with the correct format and ignore the others.

3.1.1 Added Classes

Only one class was fully created "from scratch" for this functionality, the **CSVImporter** class, located in *jabref/src/main/java/org/jabref/logic/importer/fileformat/CSVImporter.java* .

The test classes for this implementation are **CSVImporterTest** and **CSVImporterTypesTest**

Their locations are, respectively, *jabref/src/test/java/org/jabref/logic/importer/fileformat/CSVImporterTest.java* and *jabref/src/test/java/org/jabref/logic/importer/fileformat/CSVImporterTypesTest.java*

3.1.2 Modified Classes

In order to allow this feature to fully function, only one class was modified, **ImportFormatReader**, located in *jabref/src/main/java/org/jabref/logic/importer/ImportFormatReader.java.*

The added code line allowed the new functionality to work by adding a new available file extension in the "Import into current library" button. The code snippet can be seen below; the new line is 55.

```
50          formats.add(new OvidImporter());
51          formats.add(new PdfMergeMetadataImporter(importerPreferences, importFormatPreferences));
52          formats.add(new PdfVerbatimBibTextImporter(importFormatPreferences));
53          formats.add(new PdfContentImporter(importFormatPreferences));
54          formats.add(new PdfEmbeddedBibFileImporter(importFormatPreferences));
55          formats.add(new CSVImporter());
56
57          if (importerPreferences.isGrobidEnabled()) {
58              formats.add(new PdfGrobidImporter(importerPreferences, importFormatPreferences));
59          }
```

3.2 Get More Bibliographic References from a Specific Author

The goal of this functionality was to allow a user to automatically import more bibliographic references from a selected author, being 10 the maximum amount of references that can be imported.

A user can do so, by right-clicking on a bibliographic reference in their currently opened library and select "Search Google Scholar". What our implementation does is search for more articles written or co-written by the author of this bibliographic reference, and import them into the current library.

In case the selected bibliographic reference refers to more than one author, JabRef will search Google Scholar for all the authors, and import a maximum of 10 entries in total. For example, if a certain reference has two authors and the first one only has 7 published articles in their Google Scholar profile, than the program will import those 7 references and look for another references from the next author. In case the total number of imported references does not reach 10, a warning message is presented to the user.

If the bibliographic reference refers to an author whose name is not associated with any of the profiles present in Google Scholar, another alert message is presented.

Lastly, it is essential to mention that during the development of this project the team noticed that some people do not have their "real" name in their Google Scholar profile; some of them add their academic level, like "PhD" (e.g.: "Miguel Dias, PhD"). In order to solve this issue, the S Team decided to change the way JabRef separates authors. Now, authors are separated by a semicolon (';') instead of comma (','), and when looking for a profile in Google Scholar, the name the program uses for search is made up of what comes before the comma (i.e., "Miguel Dias") followed by what comes after the comma (so, the name used in the search would really be "Miguel Dias, PhD", just like we needed it to be).

3.2.1 Added Classes

The class **GoogleScholarSeacher** was created to implement this new feature, and it is located in *jabref/src/main/java/org/jabref/logic/search/GoogleScholarSearcher.java*.

The test class for this implementation is called **GoogleScholarSearcherTest** and it is located in *jabref/src/test/java/org/jabref/logic/search/GoogleScholarSearcherTest.java*.

3.2.2 Modified Classes

For this new functionality, the modified classes were:

- **StandardActions**, in *jabref/src/main/java/org/jabref/gui/actions/StandardActions.java* . The added line, 38, allowed us to create the button "Search Google Scholar", and it is now part of an enumerate of actions.
- **RightClickMenu**, in *jabref/src/main/java/org/jabref/gui/maintable/RightClickMenu.java* . This new line, 81, allows us to associate the action itself to the command, by creating a Menu Item.

The code snippets can be seen below.

In **StandardAction**:

```
36      SEARCH_SHORTSCIENCE(Localization.lang( key: "Search ShortScience")),
37      SEARCH_GOOGLESCHOLAR(Localization.lang( key: "Search Google Scholar")),
38      MERGE_WITH_FETCHED_ENTRY(Localization.lang( key: "Get bibliographic data from %0",  ...params: "DOI/ISBN/...")),
```

In **RightClickMenu**:

```
81          contextMenu.getItems().add(factory.createMenuItem(StandardActions.SEARCH_GOOGLESCHOLAR, new GoogleScholarSearcher(entry.getEntry(),
82                  dialogService, libraryTab.frame(), stateManager)));
```