



## Rapporti Tecnici INAF INAF Technical Reports

<b>Number</b>	67
<b>Publication Year</b>	2021
<b>Acceptance in OA@INAF</b>	2021-01-13T15:00:24Z
<b>Title</b>	A web interface for evaluating astronomical proposals
<b>Authors</b>	ZORBA, SONIA; FONTANA, Adriano; PARIS, Diego; CAITO, LETIZIA
<b>Affiliation of first author</b>	O.A. Trieste
<b>Handle</b>	<a href="http://hdl.handle.net/20.500.12386/29742">http://hdl.handle.net/20.500.12386/29742</a> ; <a href="http://dx.doi.org/10.20371/INAF/TechRep/67">http://dx.doi.org/10.20371/INAF/TechRep/67</a>



A web interface for evaluating  
astronomical proposals

Issue/Rev. No.	1.0
Date	Jan 4, 2021
Page	1 of 10

# A web interface for evaluating astronomical proposals

**Issue/Rev. No.:** 1.0

**Date:** January 4, 2021

**Authors:** Sonia Zorba, Adriano Fontana, Diego Paris, Letizia  
Caito

**Approved by:** Riccardo Smareglia

## 1 Introduction

Astronomers who desire to obtain observing time on telescopes need to submit proposals describing requirements and goals of the researches they would like to perform. These proposals are then evaluated by other astronomers in order to assign the observing time. This process is repeated before each observing cycle.

This report describes a web application that has been built in order to facilitate proposal evaluation and that is currently used by Time Allocation Committees of TNG (Telescopio Nazionale Galileo), REM (Rapid Eye Mount) and LBT (Large Binocular Telescope).

## 2 Use cases

The application is used by two kind of users: administrators and referees (proposal evaluators). Both users access to the application using the same login form but the web interface offers different features to the two categories of users. Following subsections describe the features offered by the tool.

### 2.1 Administrators: defining survey structure

When an evaluation cycle is not in progress, administrators can click on the “Edit survey” button to build the structure of the survey. A survey is defined by a list of items and there are three kind of items:

- title: a label that identifies a group of questions
- rating: a numeric evaluation (a decimal value between 1 and 5)
- judgment: a free text field

### Survey editor

#	Type	Text	
1 ↓	TITLE	SCIENTIFIC VALUE ✎	✖
2 ↑ ↓	RATING	Overall scientific impact and timeliness ✎	✖
3 ↑ ↓	RATING	Suitability of the proposed telescope/instrument to the scientific goal ✎	✖
4 ↑ ↓	TITLE	PROPOSING TEAM ✎	✖
5 ↑ ↓	RATING	Strength of the proposing team, also based on previous use of INAF instrumentation ✎	✖
6 ↑ ↓	RATING	Data reduction plan ✎	✖
7 ↑ ↓	TITLE	RATIONALE ✎	✖
8 ↑ ↓	JUDGMENT	Strenghts ✎	✖
9 ↑	JUDGMENT	Weakness ✎	✖

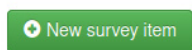
 New survey item

Figure 1: Screenshot of survey editor page. The sequence of items defined by administrators is used to display the survey questions to referees. Item order can be rearranged and items can be added, edited and deleted before the start of the cycle.

## 2.2 Administrators: uploading evaluation cycle files

Administrators prepare an evaluation cycle by uploading two CSV files and a set of PDF files:

- The first CSV file contains the list of the proposals to evaluate. Each row contains: the project identifier (usually an incremental number), information about the Principal Investigator (name, surname, e-mail address and institution), the title of the proposal, the name of the proposal PDF file and some additional metadata.
- Once the proposals metadata has been uploaded, the administrator can upload the second CSV file, that contains the list of the referees and the proposals they have to evaluate. Each row of the second CSV file contains: name and surname of the referee, e-mail address of the referee and the list of the identifiers of the projects he or she has to evaluate. When the file is uploaded, the application checks if each email address listed in it matches an existing user account. If no match is found a new user is created.
- After the two CSV files have been parsed, the administrator can upload the PDF files associated with the proposals. Names of these files have to match the names listed in the first CSV file.

## 2.3 Administrators: starting an evaluation cycle

When CSV and PDF files have been uploaded and a deadline has been set, the administrator starts the evaluation cycle by clicking on the “Start evaluation cycle” button and from that moment until the deadline, referees can evaluate the proposals. When the evaluation cycle is in progress it is not possible to edit the survey structure or upload new CSV or PDF files. However it is possible to add new referees to a proposal.

## 2.4 Referees: evaluating proposals

When a referee logs into the application, he or she sees the list of proposals to evaluate. It is possible to evaluate a single proposal by clicking on the “Evaluate proposal” button and answering the questions defined by the administrators. The interface displays the metadata associated with a proposal and provides a link for downloading its PDF document. The referee can change the evaluation multiple times and also decline the evaluation. When a referee is sure about the provided answers he or she can click on the “Conclude evaluations” button to confirm them and freeze the inserted data.

### Proposals to evaluate

Proposal data					Evaluated	Answers			
Title	Instrument	Time	Category	File		Overall sc...	Suitability...	Strength ...	Data redu...
Exploring the Universe wit...	TNG	17.5	A	<a href="#">2.pdf</a>	✓ <a href="#">Evaluate proposal</a>	3.4	4.2	4.2	4.4
REM contribution to the wo...	REM	10	D	<a href="#">3.pdf</a>	✗ <a href="#">Undo decline</a>				
Searching for optical pulse...	TNG	7.5	D	<a href="#">18.pdf</a>	✓ <a href="#">Evaluate proposal</a>	4.3	4.0	4.1	4.0
A multiwavelength view of t...	REM	37.8	D	<a href="#">23.pdf</a>	✓ <a href="#">Evaluate proposal</a>	4.3	4.3	4.3	3.9

[➡ Conclude evaluations](#)

Figure 2: Screenshot displaying the list of proposals a referee has to evaluate.

## Evaluate proposal

### Summary

Project ID	PI name and surname	PI e-mail	PI institution	Title	Instrument	Time	Category	File
2			ASI - Space Science Data Centre	Exploring the Universe with Gamma-Ray Burst Afterglows	TNG	17.5	A	<a href="#">2.pdf</a>

### Evaluation

#### SCIENTIFIC VALUE

**Overall scientific impact and timeliness**

3.4

**Suitability of the proposed telescope/instrument to the scientific goal**

4.2

Figure 3: Screenshot of an evaluation page.

## 2.5 Administrators: monitoring survey status

From the administrator dashboard it is possible to open a “Cycle summary” page by clicking on the “View results” button. This page displays the list of the referees involved in an evaluation cycle and shows the percentage of completion of the evaluations and the declined proposals. A dropdown menu allows to select previous evaluation cycles (already concluded). It is also possible to download a summary in CSV format by clicking on the “Download results” button.

## 2.6 Administrators: reassigning a declined proposal

Declined proposals are listed at the end of the “Cycle summary” page. Administrators can click on the gear button in order to assign the evaluation to a different referee. This opens the “Proposal Editor” page, that contains a button for adding new referees. The names of the referees who declined the evaluation are listed but stricken out. An e-mail message is sent to the newly added referees to warn them that there are new proposals to evaluate.

## 2.7 Administrators: adding referees when evaluation cycle is in progress

The proposals of the current evaluation cycle are listed in the “Evaluation cycle management” page. Even if it is not possible to upload new files when a cycle is ongoing, it is possible to add referees to a proposal by clicking on the pencil-shaped icon displayed at the beginning of each row.

## 2.8 Password update and recovery

All users can change their password by clicking on their name, that appears on the top menu bar. Under the login form there is also a “Forgot password?” link that opens a password recovery page. The page asks to insert the e-mail address and then sends a link containing a temporary token that can be used to reset the password.

### Proposal editor

Project ID	<input type="text" value="3"/>
PI name and surname	<input type="text" value=""/>
PI e-mail	<input type="text" value=""/>
PI institution	<input type="text" value="INAF - Osservatorio astronomico di Brera"/>
Title	<input type="text" value="REM contribution to the world-wide search for an electromagnetic counterpart of a gra"/>
Instrument	<input type="text" value="REM"/>
Time	<input type="text" value="10"/>

---

Referees	<ul style="list-style-type: none"> <li>• <input type="text" value=""/></li> <li>• <input type="text" value=""/></li> <li>• <input type="text" value=""/></li> </ul>	<input type="button" value="+ Add referee"/>
----------	---	--

---

Figure 4: Screenshot of the proposal editor page. Referees can be added at any time.

## 3 Implementation

### 3.1 Technologies

The tool is a Java EE application, based on JPA (Java Persistence API), JSF (Java Server Faces) and CDI (Context and Dependency Injection) and has been written using Java 7. Bootstrap framework is used for the front-end. Maven is used for packaging and dependency management. At the moment the data is stored into a MariaDB database, but since JPA is used, it should be easy to migrate to other RDBMS. Currently the application runs on a GlassFish 4.1 server, but it can be run in other containers (like Tomcat) including the proper libraries.

### 3.2 Database structure

Database structure is generated by JPA. This section lists the role of each table. Figure 5 shows the entity-relation diagram.

- **evaluation\_cycle**: represents a set of proposals to evaluate according to a survey. Each evaluation cycle has a deadline and evaluations can't be submitted before the cycle is started and after it is concluded. Administrators can't edit the cycle when evaluations are in progress;
- **survey\_item**: a survey is composed by a set of typed items (titles, ratings or judgments). The `_index` column is used to define the order of the items;
- **evaluation\_cycle\_survey\_item**: associates each evaluation cycle with a list of survey items. Items are copied from the previews cycle when a new cycle is created. In this way administrators can edit the survey without affecting the data associated with the already concluded cycles.
- **proposal**: represents proposals that have to be evaluated.



- `proposal_file`: metadata of the PDF files containing the proposal content.
- `evaluation_cycle_proposal`: associates each proposal with an evaluation cycle.
- `_user`: users of the tool (both administrators and referees).
- `evaluation`: each row represents the answer provided by a user to a survey item (rating or judgment) for a given proposal. The table contains both values for rating (in the `rate` column) and for judgments (in the `answer` column). These two columns can't be both not null at the same time.
- `proposal_referee`: associates each referee to the proposals he or she has to evaluate.
- `declined_proposal`: associates each referee to the proposals he or she has declined to evaluate.
- `cycle_completion`: records if a referee completed his/her evaluations.
- `password_recovery_token`: stores temporary tokens used to reset user credentials.

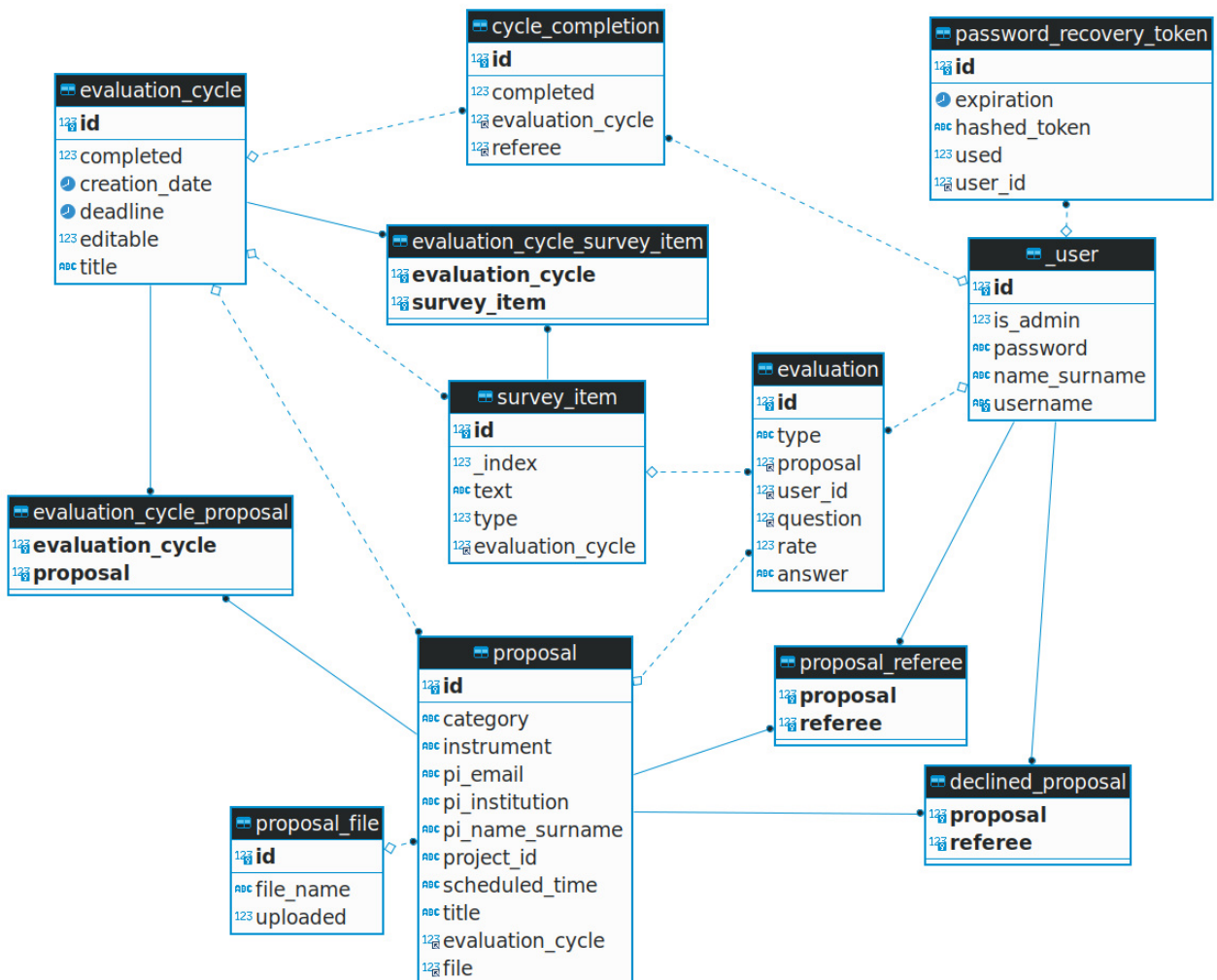


Figure 5: ER diagram



### 3.3 Classes

The business logic is handled by some scoped Java Beans. Scoped means that their life cycle is bound to the user session, a HTTP request or the application (in that case they are basically singletons). Some of these beans are mapped to a specific JSF (.xhtml) page. The following list describes the most important classes:

- **Controller:** it's a kind of DAO (Data Access Object) with a bit of extra logic in it. It is the only class that interacts with the JPA `EntityManager`.
- **AppProperties:** loads the configuration data from the `app.properties` file.
- **SurveyEditor:** session bean used to display the survey editor page to administrators.
- **EvaluationCycleManager:** session bean used to display the evaluation cycle manager page to administrators. It handles upload of CSV files.
- **CSVParserUtil:** utility class for parsing CSV files.
- **PDFManager:** handles upload and download of proposal PDF files.
- **ProposalEditor:** session bean used to display the proposal editor page to administrators.
- **CycleSummary:** session bean used to display the evaluation cycle status to administrators.
- **SummaryCSVBuilder:** builds the CSV file that can be downloaded from the cycle summary page.
- **ProposalsLister:** session bean used to display the proposals to evaluate to referees.
- **ProposalEvaluator:** session bean used to display the evaluation page to referees.
- **LoginBean:** handles login form submit.
- **FirstLogin:** it is used to set the password at the first login.
- **ForgotPassword:** sends password recovery e-mail message.
- **ResetPassword:** handles password reset using token received by e-mail.
- **UserSettings:** handles password update from user settings page.
- package `it.inaf.ia2.pwt.dm` contains the JPA entities used for mapping database content to Java objects. `IdEntity` class is used as base class for all JPA entities needing an auto-incremental id.

### 3.4 Business logic and implementation details


#### 3.4.1 First login

Password hash of newly created users is null. In this phase it is possible to login a first time without providing the password and the web interface will display a page for setting the password. Then, the hashed value of it will be stored into the database and used for the next logins. The hash is generated using the MySQL built-in function `PASSWORD()`.

#### 3.4.2 Editing survey structure

The session bean `SurveyEditor` displays the list of `SurveyItems` that compose the survey of the current cycle. The bean contains also an `inEditingItem` and an `itemToDelete` fields that are used to display the modal dialog for editing one item and the modal dialog that asks the confirmation before deleting an item. It also provides the `itemUp()` and `itemDown()` methods that are used to rearrange the order of the items. Data into the database is updated immediately when these methods are called.



	A web interface for evaluating astronomical proposals	<table border="1" style="width: 100%;"> <tr> <td style="width: 50%;">Issue/Rev. No.</td> <td>1.0</td> </tr> <tr> <td>Date</td> <td>Jan 4, 2021</td> </tr> <tr> <td>Page</td> <td>8 of 10</td> </tr> </table>	Issue/Rev. No.	1.0	Date	Jan 4, 2021	Page	8 of 10
Issue/Rev. No.	1.0							
Date	Jan 4, 2021							
Page	8 of 10							

### 3.4.3 Uploading CSV files

CSV files uploaded by administrators are parsed by the `CSVParserUtil` class. This utility class checks if the CSV files use UTF-8 encoding and have the correct number of columns. When uploading proposals, an error is thrown if a row contains a proposal id that has already been used in the current cycle. In the same way referees CSV parsing fails if a row refers to a proposal id that doesn't exist.

If the parsing is successful, data is immediately inserted into the database. If a referee doesn't match any existing user in the database, a new user is created.

### 3.4.4 Uploading PDF files

PDF files are stored in a directory that is configured in the `app.properties` file (`pdf_storage_path` key). For each cycle a new child directory is created and it is named as the cycle id. The `PDFManager` class handles the creation of these directories. If a PDF file has a name that doesn't match any of the names listed in the previously uploaded CSV file defining proposals, the file is not saved. If a file is uploaded twice the previous version is overwritten. When a PDF is uploaded the corresponding `uploaded` cell in the `file_info` table is set to true. This boolean flag is used to display the download link or not.

### 3.4.5 Listing proposals to evaluate

When a referee performs a login the `ProposalsLister` bean loads the proposals to evaluate and also an object that is an instance of `RefereeEvaluationStatus` class. That object is built merging data from the `proposal` table and the `evaluation` table and it is used to display a preview of the actions already performed by the referee. Indeed, near the list of the proposals, the referee can see a list of all the provided answers and a green check icon appears if an evaluation of a given proposal can be considered completed (that means there are no empty answers). If all the proposals have the green check or the evaluations have been declined the referee can click on the "Conclude evaluations" button. This action stores data in the `cycle_completion` table and freezes the interface, so that it can't be edited anymore.

### 3.4.6 Evaluating a proposal or declining the evaluation


When a referee opens the evaluation page, a list of `RefereeEvaluation` objects is loaded by the `ProposalE←valuator` bean. If the referee opened the evaluation for the first time, these objects are created based on the survey items list that defines the survey structure. Otherwise they are loaded from the last edit. Values are saved into the database only when the referee clicks on the "Save" button. The referee can also decline the evaluation from the same page. In that case a new row will be added into the `declined_proposal` table.

### 3.4.7 Handling concurrent edit

Referees can confirm their answers by clicking on the "Conclude evaluations" button. This action sets the completion of their evaluations. However, an administrator could assign another proposal to them and that would unset the completion. If a referee attempts to conclude the evaluations and in the same moment an administrator assigns a new proposal, inconsistencies can happen. For this reason an application scoped bean named `InProgressEvaluationsManager` contains some lock objects to ensure the atomicity of these operations. If the system detects that a new proposal has been assigned while a referee attempts to conclude the evaluations, the referee action is ignored and the page is reloaded, to display the new proposal to evaluate.

### 3.4.8 Keeping the session alive

The HTTP servlet session expires after some time (usually one hour). To avoid that a referee that was keeping the proposal evaluator tool opened while reading the PDF documents discovers that session is expired and it is necessary to repeat the login, a REST endpoint at `/service/keepalive` is used to prevent the session expiration until a page of the application is opened in the browser. The endpoint is called every minutes by a JavaScript script. It also returns a boolean that indicates if evaluation deadline has been reached. In that case the page is reloaded, so that the not editable version of the interface is displayed.

	<h2>A web interface for evaluating astronomical proposals</h2>	<table border="1"> <tr> <td>Issue/Rev. No.</td> <td>1.0</td> </tr> <tr> <td>Date</td> <td>Jan 4, 2021</td> </tr> <tr> <td>Page</td> <td>9 of 10</td> </tr> </table>	Issue/Rev. No.	1.0	Date	Jan 4, 2021	Page	9 of 10
Issue/Rev. No.	1.0							
Date	Jan 4, 2021							
Page	9 of 10							

### 3.4.9 Reset password token

When a user asks to reset the password a random token is generated. The hash of its value is stored into the database and the plaintext token is appended to a recovery password URL and is sent to the provided e-mail address. When the recovery page is opened from the received e-mail message, the token passed as parameter is hashed again and its value is compared to the one stored into the database. If the two values match, a form for resetting the password is shown. Tokens have a lifespan that can be configured in the `app.properties` file (`token_life` key, value expressed in days).

### 3.4.10 Loading animation

While AJAX calls are in progress a loading animation (spinner) is displayed. The animation is realized using the CSS3 `animation` property. The animated element is always present in the page but it is hidden when there are no active AJAX calls and displayed otherwise. To detect the AJAX calls the JSF JavaScript function `jsf.ajax.addOnEvent` is used.

### 3.4.11 Integration test

Source code contains also a simple integration test based on Arquillian framework. The goal is to test the critical features of the application in an environment that is as similar as possible to the production environment. For this reason the test is run in a real GlassFish test server and using a real MySQL database. Properties for connecting to the test server are configured in the `arquillian.xml` file. Test uses a separate `persistence.xml` file, having the `schema-generation` strategy set to `drop-and-create`. In this way the structure of the database is regenerated before each test run.

## 4 Build and deploy

### 4.1 Generate the war file

Source code can be retrieved from the following repository:

<https://www.ict.inaf.it/gitlab/ia2/ProposalWebTool>

The application can be built using the `mvn clean package -DskipTests` command. This generates a war file in the target folder that can be deployed on GlassFish server. Configuration properties listed in the `pom.xml` file are used to fill the placeholders in `src/main/resources/app.properties` file. In this way it is possible to specify the configuration properties build-time. Following example shows the Maven command that can be used to configure all the properties while generating the war file:

```
mvn clean package -DskipTests -Dpdf.storage.path=/home/glassfish/pwt/pdf -Dsmtplib.server=
  ↪ localhost -Dfrom.mail=noreply@ict.ia2.inaf.it -Dapp.base.url=https://ict.ia2.inaf.it/
  ↪ proposals -Dtoken.life=3 -Dtesting=false
```


The testing flag is used for running the application in testing mode, that means that e-mail messages are not sent.

### 4.2 Database setup

Create a MySQL/MariaDB database and a user having all privileges on it:

```
CREATE DATABASE pwt;
CREATE USER pwt@localhost IDENTIFIED BY 'mypassword';
GRANT ALL PRIVILEGES ON pwt.* TO pwt@localhost;
```

Create a new Connection Pool from GlassFish server admin panel (Resources → JDBC → JDBC Connection Pool). Select “`javax.sql.DataSource`” as “Resource Type” and “MySQL” as “Database Driver Vendor”, then configure the additional properties (`ServerName`, `DatabaseName`, `Username`, `Password`). It is possible to test

	<h2>A web interface for evaluating astronomical proposals</h2>	<table border="1"> <tr> <td>Issue/Rev. No.</td> <td>1.0</td> </tr> <tr> <td>Date</td> <td>Jan 4, 2021</td> </tr> <tr> <td>Page</td> <td>10 of 10</td> </tr> </table>	Issue/Rev. No.	1.0	Date	Jan 4, 2021	Page	10 of 10
Issue/Rev. No.	1.0							
Date	Jan 4, 2021							
Page	10 of 10							

the pool using the “Ping” button.

Then, create a new JDBC Resource named `jdbc/pwt` from GlassFish server admin panel (Resources → JDBC → JDBC Resources). Associate it with the previously created connection pool.

### 4.3 Creating administrator user

When the application war file is deployed for the first time, JPA generates the structure of the database associated with the `jdbc/pwt` resource.

The administrator user has to be created manually using the following SQL statement:

```
INSERT INTO _user(username, password, name_surname, is_admin) VALUES ('admin', PASSWORD('
↳ mypassword'), 'Admin', 1);
```

### 4.4 Manually update values in the database

In some cases, like postponing the deadline while an evaluation cycle is in progress, it is necessary to manually update the values in the database. The values are not immediately updated inside the application, due to JPA caching layer. To force the reload of the values from the database it is necessary to reload the application. This can be done without redeploying it by clicking on the “Reload” link, from the “Applications” section of the GlassFish server admin panel.

## 5 Conclusions

The tool has been used multiple times and we can consider it a mature product. In the past three years of activity it has been used by 130 referees to evaluate over 370 proposals.

Even if the initial requirements stated that the evaluation cycle mustn't be editable when it is in progress, it happened that administrators needed to make some changes, for example postponing the deadline, removing a referee, adding a proposal or fixing typos. In those cases it was necessary to update the data by acting directly into the database, because the web interface didn't allow those modifications on an ongoing cycle. So, future releases would relax these constraints.

The application relies on GlassFish and JSF, that are disused technologies. Future releases would port it to a different container and replace JSF with a different framework.