Product Backlog

2.1 Plugin Support

Priority: High
Cost: 14 Story Points

As Hanno (an administrator), I want plugin API support for the application, where I can create, add, modify plugins written in a programming language that the application can use to extract data from other catalogs.

Tasks:
1. Build/find a class library to make it easier to deal with web requests.
2. Create plugin interface.
3. Create a XML parser that will take a dictionary and output a XML format text, with fields in the same order as dictionary.
4. Create serializer class and add plugin loading functionality
5. Create two default plugins for exoplanet.eu and NASA Exoplanet Archive

---

2.2 Standardizing Units

Priority: High
Cost: 3 Story Points

As Hanno (an administrator), I want each plugin to be able to standardize units of measurements found in its catalogue to one that I specify in the settings of the application.
Tasks:

1. Find/build a unit conversion library
- Create a way of making general conversion (i.e., specify original unit and convert to whatever is defined in the settings)

2.3 Scheduling a Check for Updates

Priority: High
Cost: 5 Story Points

As Hanno (an administrator), I want the application to check the other exoplanet databases it is tracking, for new updates, on a specified interval (default is daily at 11:59pm).
Tasks:
1. Create a server framework that will run the plugins with threads
2. Have server constantly check system time

2.4 User Permissions

Priority: High
Cost: 5 Story Points

As Hanno (an administrator), I want to use GitHub repo collaborators as a way of distinguishing which user has access to use the application (manage pull-requests, run the server).

Tasks:
1. Create GitHub API management class, and add oauth2 login functionality
2. Add ability to check if user is a collaborator of the repo

---

2.5 Application Making Pull Requests

Priority: High
Cost: 18 Story Points

As Hanno (an administrator), I want the program to make pull requests to the OEC database when it finds changes in the databases that the application is monitoring, using an automated GitHub user account.

Tasks:
1. Add repo management functionality to GitHub API class
2. Add the ability to make pull requests to GitHub API class
3. Integrate GitHub API into a server so that pull-request are made with each new change
4. Add XML file reading to serializer
5. Add the ability to update a XML file with changes to serializer

---

2.6 Accepting/Rejecting Pending Pull Requests

Priority: High
Cost: 5 Story Points

As Irving (a user), I want to be able to accept, reject, or amend pull requests made by the application through a user interface by pressing a button on a pop-up window.

Tasks:
1. Add the ability to accept, reject, and change pull requests
2. Add pull-request to GUI
- Add viewing
- Add accepting/denying
- Add sorting to display

- Add editing

2.7 Merging Data from Plugins

Priority: High
Cost: 12 Story Points

As Hanno (an administrator), I want the data retrieved by a plugin to be merged with existing data in the Open Exoplanet Catalogue.

Tasks:
1. Write a class that merges two data types together
2. Turn the merged data back into an XML file

2.8 Source of the Update

Priority: Medium
Cost: 2 Story Points

As Hanno (an administrator), I want the website source of the new changes to be a comment in the pull request in the database.

Tasks:
1. Create a field that shows all of the publications related to a specific system. Shown when clicking on the system from the list of altered systems.
2. Make it mandatory for plugins to extract publication data

2.9 See History

Priority: Medium
Cost: 3 Story Points

As Hanno (an administrator), I want to see the history of pull requests, on a user interface.

Tasks:
1. Create a button on the Home screen that allows the user to go into the History screen.
2. Create a History screen that displays a list of the 20 most recent pull requests made by the GitHub bot.

2.10 Edit Pending Pull Requests

Priority: Medium
Cost: 4 Story Points

As Irving (a user), I want to be able to edit a pending pull request through a user interface by pressing an edit button when shown a list of new changes.

Tasks:
1. Create an edit button that opens an interface that allows the user to edit a specified XML file.
2. Update the user's current commit to the one that was edited using the interface.

2.11 History of Updates

Priority: Medium
Cost: 4 Story Points

As Irving (a user), I want to see the planets that have been most recently added/edited by the application in the OEC, using the user interface.

Tasks:
1. Use GitHub API and get the most recent changes
2. Format and parse data from the API and display it in the GUI

2.12 Force a Check for Updates

Priority: Medium
Cost: 3 Story Points

As Hanno (an administrator), I want to be able to force the application to run even if it is not scheduled to check the databases.

Tasks:
1. Create a button in the Home screen that allows the application to check the tracked databases for updates.

2.13 Creating a User Interface

Priority: Medium
Cost: 49 Story Points

As Irving (a user), I want to be able to interact with the application using a user interface.

Tasks:
1. Research UI implementation
2. Create basic UI application
3. Add GitHub login UI
4. Add force check button
5. Add a status checker
6. Add pull request viewer to UI
7. Add plugin management to UI
8. Add settings editor to UI

2.14 Difference Between Updates

Priority: Low
Cost: 25 Story Points

As Irving (a user), I want to be able to click on a newly updated system from a list in the UI and have lines highlighted, showcasing which parts in the XML file have changed.

Tasks:
1. Add repo checking to GUI
- Add the ability to see recent changes compared to the master branch
- Create an interactable list of all recent changes
- Create a GUI that will display changes of the XML files

2.15 Settings Form

Priority: Low
Cost: 15 Story Points

As Hanno (an administrator), I want to be able to change time controls, bot account settings, and conversion units, in a settings form in the application.

Tasks:
1. Use C# settings designer to create a settings form and maintain user settings
2. Add timed update controls to user settings
3. Add GitHub bot user account specification to user settings
4. Add a list of possible conversion units to user settings
5. Integrate user settings into the server with on demand changing
6. Add user settings to GUI

Changes from Deliverable 3:

We added a new user story to address the fact that Hanno ultimately wants the data we've pulled to be merged into his catalogue; user story 2.7 is the one that covers this issue. We've also adjusted some of the costs associated with the user stories as we worked more with the actual task. We found some stories were over estimated (like the implementation of the conversion library that went from 8 -> 3 points) and some were under estimated (like the implementation of the plug in library that went from 12 -> 14). We felt the priorities were accurately estimated and did not need changinging.