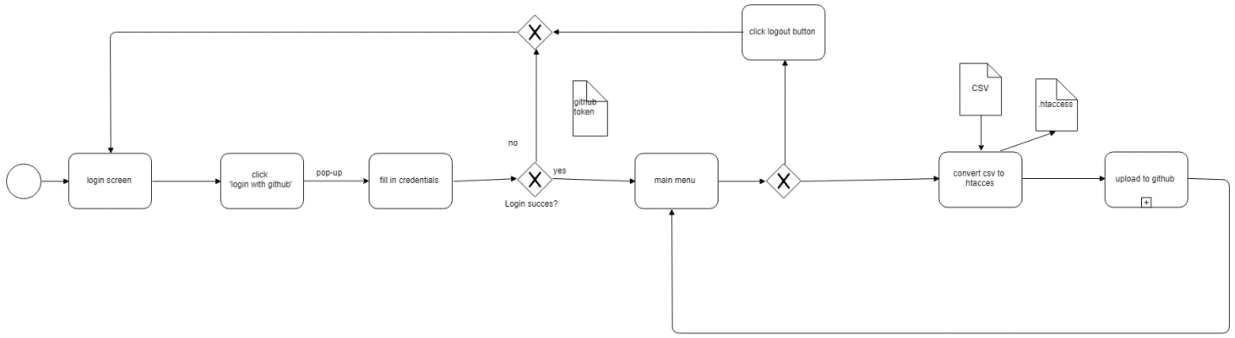
Culturize Workflow Description

Table of Contents

1. General

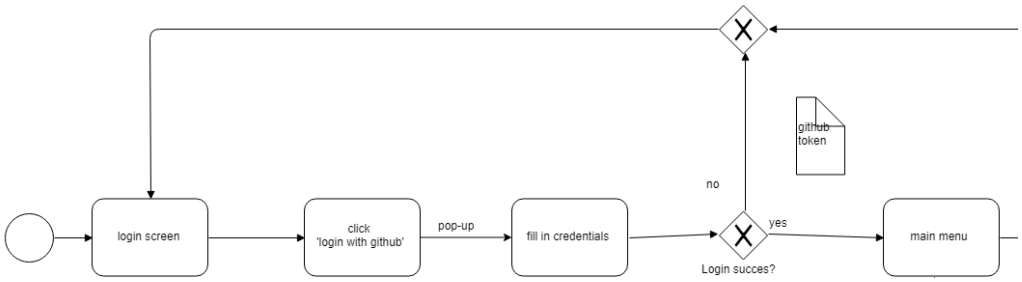
This document explains the technical flow of the Culturize application. First there will be a high-level, general explanation of the flow of the app. Afterwards, a more detailed explanation will be given about each part of the process.

2. High-level flow



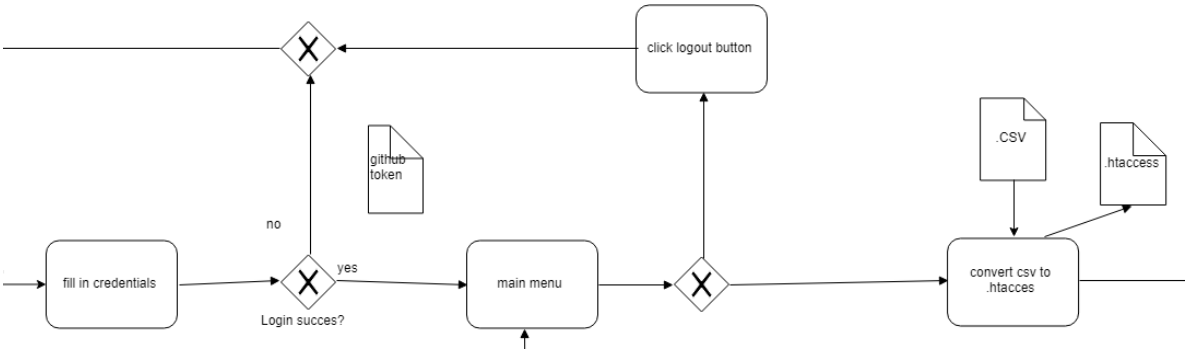
First the user will login using GitHub. This will generate a GitHub token, used for authorization with Git and the GitHub API. Once connected, the user will have access to the main menu where they can add the .CSV-file of their choice that will be converted to .htaccess, then, the .htaccess file will be pushed to a GitHub repo of their choice.

3. Logging in



The first screen the user sees is the “Login with GitHub” page. Clicking the “Sign-In” button will cause a pop-up from GitHub to appear where they can fill in their credentials (note that if the user has already used the app before, the popup won’t appear, and the rest of the process will happen in the background). If the login is not successful for any reason the app will stay on the login screen, and a successful login will redirect the user to the main menu. Successfully signing in with GitHub will get us a token that can be further used as a way of accessing the GitHub API. The GitHub token can also serve as a substitute for the password when interacting with Git.

4. Main menu



In this menu the user can use the main functionality of the app by using the form in the center, or log-out by clicking the link in the top right corner. This will cause the app to return to the log-in screen. The main purpose of the main menu is uploading the CSV-files that will be converted to htaccess-files. In the main menu there are also other configurations that should be filled in, like the subdirectory and the repository to which the htaccess-files should be pushed. Furthermore, advanced options are also available, these options are only meant for users with experience in GitHub and provide more control over what's happening.

4.1. CSV file path

This is the path to the CSV file that should be converted.

4.2. Subdirectory

This is the subdirectory on the GitHub repository to which the htaccess-files will be saved. A default value for this subdirectory can be configured in the configuration file of the program. We highly recommend configuring a default path, at least to the level of each museum. Also, writing a new htaccess-file to a subdirectory will override an older htaccess-file if present. This means that selecting appropriate subdirectories is very important. This should be explained thoroughly to clients.

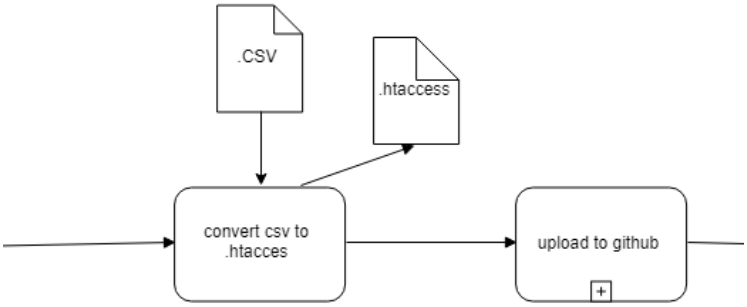
4.3. Push the changes to

This is the URL of the GitHub repository to which the htaccess-files should be pushed.

4.4. More advanced options

The more advanced options are meant for users with experience in git and the GitHub workflow. They allow more customization of the pull requests. This can be used for testing.

5. Convert csv to htaccess



After the user clicks on the submit button as all the required information has been filled in, the CSV-file will be converted to a htaccess-file. For a successful conversion at least two columns are required in the CSV-file, a PID or persistent ID of the object and a URL to which the persistent URI containing this PID should redirect. There is also the possibility of using a third column in the CSV-file. This column indicates the document type that will be linked to the PID (this column is optional by default but can be made mandatory). This "document type" column makes it possible to connect multiple URLs to 1 PID, for example a URL for data, a URL for a representation and so on. This column makes the linking of data possible. A fourth column which is also optional is the "enabled" column. This column should contain only ones and zeroes, indicating if the link that is set up in that row is active. This makes it possible to disable a link, for example when copyright issues appear.

In short, the columns will look like this:

PID	Document type (optional)	URL	Enabled (optional)	*

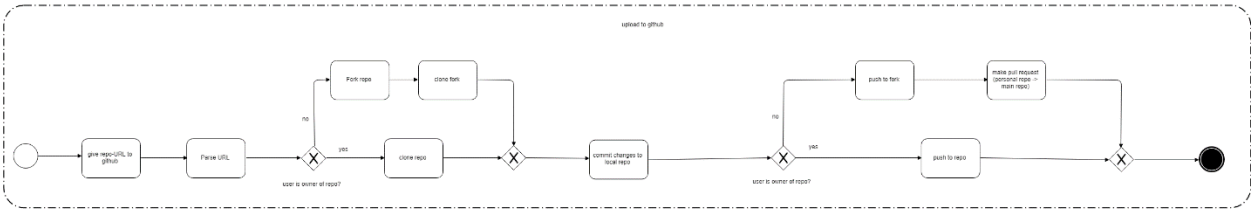
*More columns can be added but will be ignored by the culturize application.

The persistent URI's that are created by this conversion look as follows:

http://domainnameofserver/*subdirectory/subsubdirectory/.../documenttype*/**PID**

and redirect to the URL given in the CSV-file.

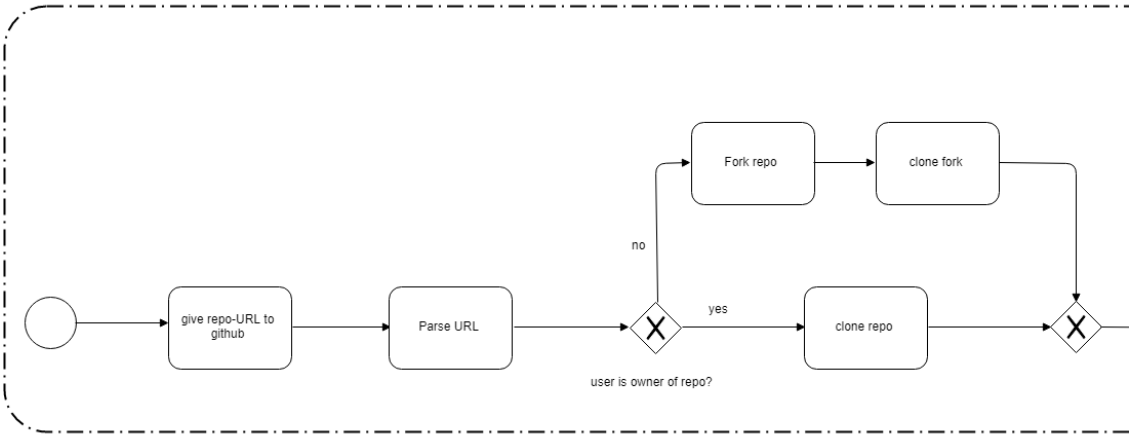
6. Upload to GitHub



After the conversion, the htaccess-file will be uploaded to GitHub, the general flow of this can be seen above. We will look at it in more detail in the following sections. In this process multiple API-calls are made to the GitHub-API. To facilitate the generation of fitting HTTP requests to the GitHub-API we make use of a library called octokit, the documentation of this library can be found here:

https://octokit.GitHub.io/rest.js

6.1. Parsing URL and forking repo

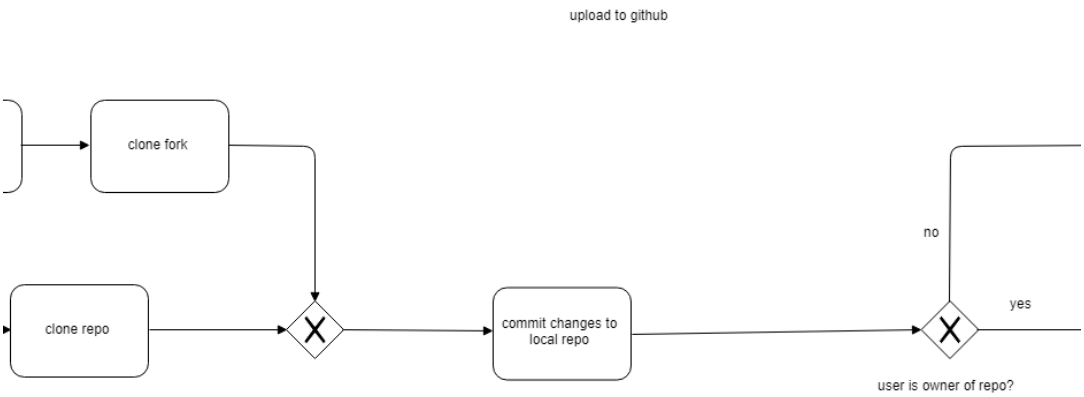


First the URL of the repository that was provided in the main menu is parsed to retrieve the name of the owner of the repo, and the name of the repo.

If the user that was logged in to GitHub earlier is not the owner of the repository to which the htaccess-files should be pushed, this repository will be forked on the user's account and the fork will be cloned, else, the app will simply clone the given repo.

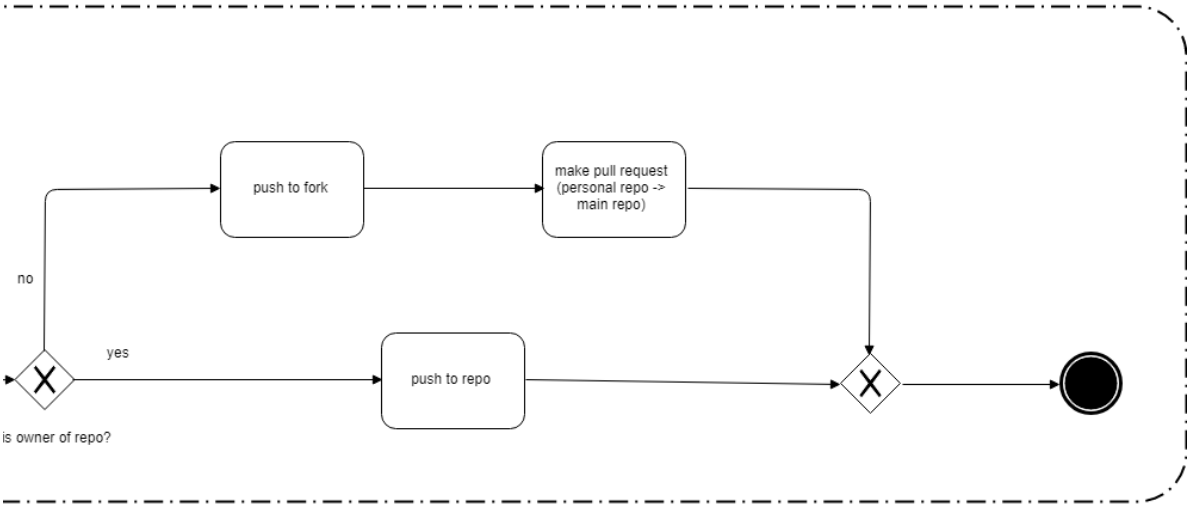
In the case where the user is not the owner of the repo, the API calls to fork will be made even if the user has already forked the repo (we do not attempt to detect such situations). We also make other calls to update the fork to the latest commit on the original repo, so the fork stays up to date.

6.2. Committing changes to repo



Now that we have a local copy of the GitHub repo, the .htaccess file is saved to the desired location. A commit is then created with the desired message.

6.3. Push changes to GitHub



After the changes have been committed to the local repository the changes will be pushed to GitHub. If the user is the owner of the repository, changes will be pushed directly to the target repo. Else, the changes will be pushed to the fork created earlier, and a pull request will be made on the user's behalf. This marks the end of the process for the Culturize App.