1 Methodik

TODO: Ausformulieren

Für die Datenerhebung wurden 101 Open Source Projekte auf GitHub ausgewählt. Laut Midha et al. lassen sich unterschiedliche Programmiersprache schlecht miteinander vergleichen [Mid12]. Um eine vergleichbare Basis zu schaffen wurde sich bei der Datenerhebung auf die Programmiersprachen JavaScript und TypeScript beschränkt.

TODO: zusammenhang zwischen JS und TS und warum auch TS Projekte mitaufgenommen werden

TODO: evtl. weiter Beschränkung erwähnen: Nur Bibliotheken Framekworks etc.

TODO: evtl. erwähnen warum gerade JavaScript

1.1 Händische Datenerfassung

TODO: Ausformulieren

TODO: ggf. mehr Projekte mit nicht MIT Lizenzen in Datenerhebung aufnehmen

Mithilfe einer selbstgeschrieben GUI wurden zunächst Eckdaten händisch erfasst, dazu gehören Besitzer und Name der ausgewählten GitHub Projektes sowie der Name des Projektes auf NPM. Des Weiteren wurden Daten erfasst, die man nicht automatisiert hätte erfassen können. Darunter gehören Daten wie: Qualität der Dokumentation, ob ein Projekt Sponsoren hat, wer hinter den Projekten steckt und um was für ein Typ an Projekt es sich handelt.

Mit der GitHub Suchfunktion[1] wurde nach JS/TS Projekten gesucht. Die Projekte werden absteigend nach Sternen wiedergegeben und auch so auch erfasst. Allerdings gibt es hier viele Projekte die keine Bibliotheken, Frameworks oder ähnliches sind, wie beispielsweise *30-seconds/30-seconds-of-code*[2]. Dieses und ähnliche Projekte wurden entsprechen für die Datenerfassung nicht berücksichtigt.

Während der Datenerfassung wurde festgestellt, dass die meisten Projekte eine MIT Lizenz haben, sodass später explizit nach nicht MIT lizenzierten Projekten gesucht wurde.

Alle Einträge wurden in einer CSV gespeichert und später weiter automatisiert verarbeitet, mehr dazu in Kapitel 1.2.

In den nächsten Unterkapiteln soll näher erklärt wie und nach welchen Kriterien die von Hand erfassten Daten zu Stande kommen und welche Bedeutung die Werte in der CSV haben.

1.1.1 Dokumentation

Bei der Qualität der Dokumentation wurde nach folgenden Kriterien bewertet

1 `https://github.com/search/advanced`
2 `https://github.com/30-seconds/30-seconds-of-code`

0 = keine Dokumentation
1 = basis Dokumentation, rein textuelle Dokumentation
2 = Dokumentationen mit Demos oder Live Beispielen, Einführungsvideos oder ähnliches

Hierfür wurde die Dokumentation auf GitHub selbst bzw. auf den externen Websiten überflogen und kategorisiert.

1.1.2 Sponsoren

Hier wurde für jedes Projekt geprüft, ob es Sponsoren hat. Hierfür gibt es oft auf der GitHub Page eine Link zur Sponsor Seite oder in den meisten Fällen findet sich in der *README.md* bzw. auf der Website des Projektes eine kurze Danksagung an die Top Sponsoren.

Die Anzahl an Sponsoren bzw. die Einnahmen durch Sponsoren wurden nicht beachtet, nur ob Sponsoren vorhanden sind.

0 = hat keine Sponsoren
1 = hat Sponsoren

1.1.3 Backed By

Im nächsten Schritt wurde vermerkt wer hinter einem Projekt steckt, hierbei gab es die Möglichkeiten:

0 = Eines reinen Community Projektes wie zum Beispiel VueJS,
1 = ein Projekt wie NodeJS welches von einer Stiftung wie der OpenJS[3] entwickelt oder unterstützt wird oder
2 = Projekte die von Unternehmen entwickelt werden, wie React von Facebook beispielsweise.

Diese Information fand sich entweder auf der GitHub bzw. Homepage des Projektes oder das Unternehmen ist der Besitzer des Repositories.

1.1.4 Kategorie

Im letzten Schritt wurde das Projekt nach Typ Kategorisiert, folgende Kategorien standen zur Auswahl.

Kategorie	Anzahl
Utility	40
UI	28
Application	12
Library	7
Framework	6
Test-Framework	5
Open-Core	2
API	1

Utility

3 https://openjsf.org/

- **Utility**, wird nicht direkt in Projekte eingebaut sondern als Tool verwendet. Beispiel: `shelljs/shelljs`[4]
- UI
- Application
- Library
- Framework
- Test-Framework
- Open-Core
- API

TODO: Unterscheidung zwischen Utility/Library etc.

TODO: Zahlen Ausfüllen, wv Projekte in welcher Kategorie

1.2 Automatisierte Datenerfassung

Im zweiten Teil der Datenerfassung wurden weitere Daten automatisiert erfasst. Hierfür wurde die GitHub API[5], NPM API[6] und für einige Daten Web-Scraping verwendet.

1.2.1 Daten aus der GitHub API

Mittels der GitHub API wurden folgende Daten gesammelt:

- Anzahl der GitHub-Sternen
- Erstellungsdatum des Repositories
- Vorhandensein einer *CODE_OF_CONDUCT.md*
- Vorhandensein einer *CONTRIBUTING.md*
- Lizenz
- Anzahl der Commits in den letzten 12 Monaten
- Anzahl der Issues in den letzten 12 Monaten

1.2.2 Daten aus der NPM API

Mit der NPM API wurden die Downloads der letzten 7 Tage abgefragt.

1.2.3 Daten aus dem Web-Scraping

Mithilfe von Web-Scraping wurden folgende Daten erfasst:

- Die Anzahl von *UsedBy* auf der GitHub Page des Projektes.
- Anzahl der Gesamt-Commits auf dem default Branch
- Anzahl der Contributor

Ein Contributor zählt nur dann als solche, wenn dessen Commit entweder auf dem *default* oder *gh-pages* Branch liegt. Commits auf anderen Branches werden nur dann gezählt, wenn ein merge auf einen der vorherig erwähnten Branches stattfindet. Gleiches gilt für Commits [GHa].

4 https://github.com/shelljs/shelljs
5 https://docs.github.com/en/rest
6 https://github.com/npm/registry

Literaturverzeichnis

[GHa] Why Are My Contributions Not Showing up on My Profile? https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-github-profile/managing-contribution-graphs-on-your-profile/why-are-my-contributions-not-showing-commit-was-not-made-in-the-default-or-gh-pages-branch. Zuletzt aufgerufen am 18.05.2022.

[Mid12] V. Midha und P. Palvia. Factors Affecting the Success of Open Source Software. *Journal of Systems and Software*, 85(4):895–905, Apr. 2012.