HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

qutebrowser made extensible

- Studienarbeit im Herbstsemester 2018/19 Informatik
- Autor: Florian Bruhin
- Betreuer: Prof. Stefan Keller, Institut für Software, HSR
- Industriepartner: -

Aufgabenstellung

Projektvorstellung: Von der Projektseite: "qutebrowser is a keyboard-focused browser with a minimal GUI. It's based on Python and PyQt5 and free software, licensed under the GPL. Das Projekt existiert seit Dezember 2013 und hat inzwischen einige tausend User. Eine Extension-API, um qutebrowser mit eigenem Code zu erweitern, war bereits lange ein oft geäusserter Wunsch - da es aber schwierig ist, eine solche nachträglich zu ändern (ohne dass Extensions angepasst werden müssen), sollte sie gut geplant und mit Liebe zum Detail umgesetzt werden. Als wissenschaftlicher Aspekt soll ein eher theoretisch-orientiertes Kapitel zu Principles of API Design sowie Python Type Annotations erarbeitet werden.

Motivation: Viele Benutzer von qutebrowser sind power-user und haben dadurch spezifische Wünsche und Workflows. Es soll diesen Benutzern möglich gemacht werden, sich einfach ein Extension für die gewünschte Funktionalität zu schreiben, um den Kern von qutebrowser schlank zu halten. Ausserdem soll auch ein Teil des momentanen Kerns in Extensions ausgelagert werden (welche mit qutebrowser mitgeliefert werden).

Stand: Da qutebrowser nun rund 5 Jahre gewachsen ist und anfänglich ohne grosse Software-Engineering-Kenntnisse entwickelt wurde, hat sich eine gewisse "Technical Debt" entwickelt. Viel wurde im Laufe der Jahre schon aufgeräumt, aber einige Baustellen sind weiterhin offen. Diese sollen als Teil der Studientarbeit erst aufgeräumt werden, bevor eine Extension API entwickelt wird, um einen Einfluss auf die API zu vermeiden. Insbesondere sind dies folgende Punkte:

- Die Dokumentation von qutebrowser nutzt ein Tool (asciidoc) welches einige Probleme mit sich bringt und inzwischen nicht mehr weiter entwickelt wird. Um die Extension API auch adäquat dokumentieren zu können, soll auf Sphinx umgestiegen werden. Ein externer qutebrowser-Contributor (Fritz Reichwald) begann bereits vor der Studienarbeit mit ersten Arbeiten daran, und führt diese weiterhin fort. Er soll deshalb dabei unterstützt werden und es soll an seine Arbeit angeknüpft werden.
- Einige Stellen im qutebrowser-Code besitzen bereits Type Hints, aber mypy wird noch nicht systematisch (z.B. via Travis CI) eingesetzt. Dies soll geändert werden, und mindestens der Code, welcher über die Extension API exponiert wird, soll mit Type Hints versehen werden.

Vorgehen: Eine Extension API zu konzipieren birgt eine zentrale Frage: Wie mächtig soll die API sein und wie simpel - und dadurch stabil (im Sinne von "nicht ändernd") - wird sie? Dies wurde letztlich Firefox zum Verhängnis, denn das Extension-System von Firefox war mächtig, aber viel zu komplex und hat damit wichtige Änderungen an Firefox selbst erschwert. Als Resultat musste Firefox die XUL-API aufgeben und zwang damit alle bestehenden Add-ons zu einem grossen Rewrite.

In qutebrowser soll daher initial nur sehr wenig Funktionalität über die entsprechend sauber geplante Extension API exponiert werden. Es soll in agilen Zyklen gearbeitet werden, die die neue Extension-Funktionalität mit einem Refactoring von bestehendem Code verbinden:

- Via Use-Cases bzw. konkreten Extension-Ideen wird eine API für eine bestimmte Extension-Funktionalität ausgearbeitet.
- Diese API wird in qutebrowser implementiert.

- Bestehender Code im Core-Teil von qutebrowser, der ähnliche Funktionalität benutzt, wird umgeschrieben, um die Extension API zu benutzen. Als Beispiel: Nachdem eine API hinzugefügt wird, um Netzwerk-Requests zu blockieren, kann der in qutebrowser eingebaute Adblocker umgeschrieben werden, um diese API zu benutzen.

Lieferobjekte

1. qutebrowser refactored und ergänzt mit Extension API und internen Extensions
2. Zusätzliches Repository für Extensions.
3. Dokumentation (englisch) inkl. theoretischem Kapitel und dokumentiertem Extension API (API-Docs separat), Textabstract (englisch), Management Summary (englisch)
4. Software (englisch).
5. Die vom Studiengang geforderten bzw. empfohlenen Lieferobjekte: Poster (nur digital), Broschüren-Abstract, kein Kurzvideo.

Vorgaben/Rahmenbedingungen

- Grösstenteils gegeben durch das bestehende Projekt:
  - Python 3 (3.5+)
  - PyQt5 basierend auf Qt5 (5.7+)
  - pytest als Test Framework
  - Diverse Code Quality Tools (pylint/flake8/etc.)
  - Neu dazu kommen soll MyPy mit Type Annotations
- Allgemein:
  - Moderne SW-Entwicklung mit Unit Testing und kontinuierlicher Integration.
  - Nichtfunktionale Anforderungen: keine besonderen

Vorgehen und Arbeitsweise: Die Studierenden wählen nach Rücksprache ein Vorgehensmodell zur Softwareentwicklung. Es gibt wöchentliche Meetings mit vorbereiteten Unterlagen; wobei Ausnahmen vereinbart werden können.

Dokumentation

Zur Dokumentation (vgl. auch Lieferobjekte oben):

- Die Abgabe ist so zu gliedern, dass die obigen Inhalte klar erkenntlich und auffindbar sind (einheitliche Nummerierung).
- Die Zitate sind zu kennzeichnen, die Quelle ist anzugeben.
- Verwendete Dokumente und Literatur sind in einem Literaturverzeichnis aufzuführen (nicht ausschliesslich Wikipedia-Links auflisten).
- Dokumentation des Projektverlaufes, Planung etc.
- Weitere Dokumente (z.B. Kurzbeschreibung, Eigenständigkeitserklärung, Nutzungsrechte) gemäss Vorgaben des Studiengangs und Absprache mit dem Betreuer.

Form der Dokumentation zuhanden Betreuer (Studiengang siehe separate Instruktionen):

- Bericht gebunden (1 Exemplar)
- Alle Dokumente und Quellen der erstellten Software auf USB-Stick.

Bewertung

Es gelten die üblichen Regelungen zum Ablauf und zur Bewertung der Studienarbeit des Studiengangs Informatik mit besonderem Gewicht auf moderne Softwareentwicklung wie folgt:

- Projektorganisation (Gewichtung ca. 1/5)
- Bericht, Gliederung, Sprache (Gewichtung ca. 1/5)
- Inhalt inkl. Code (Gewichtung ca. 2/5)
- Gesamteindruck inkl. Kommunikation mit Industriepartner oder weiteren Beteiligten (Gewichtung ca. 1/5).

Ein wichtiger Bestandteil der Arbeit ist, dass eine lauffähige, getestete Software abgeliefert wird (inkl. Softwaredokumentation).

Weitere Beteiligte

Freiwillige aus der qutebrowser-Community.