Emergent Architecture Design
Octopeer for BitBucket

Out of Context

Draft Version 0.4 (Sprint 4)

Date:
20-05-2016

Group members:
Arthur Guijt
4377338
a.guijt@student.tudelft.nl

Lars Stegman
4365801
l.s.stegman@student.tudelft.nl

Laurens Kroesen
4350286
l.kroesen@student.tudelft.nl

Cas Buijs
4345185
s.j.m.buijs@student.tudelft.nl

Thomas Overklift
4080890
t.a.r.overkliftvaupelklein@student.tudelft.nl

Responsible Teacher:
Alberto Bacchelli
a.bacchelli@tudelft.nl

Teaching Assistants:
Bastiaan Reijm
A.B.Reijm@student.tudelft.nl
Aaron Ang
A.W.Z.Ang@student.tudelft.nl

Introduction

This document provides an architectural overview of the tool ‘Octopeer for Bitbucket’. The document is split up in several sections. First some design goals are discussed. Design goals are guidelines or practices that should be followed or incorporated when building the tool. Satisfying all design goals builds confidence in the fact that the final product can pass quality checks.
In the rest of the document we discuss components that form the system and clarify some design choices.

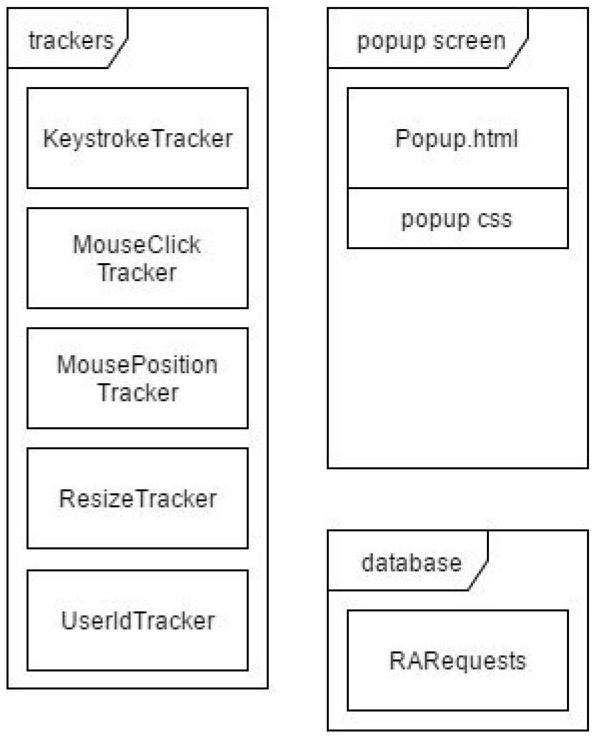
1.1 Design goals

The main design goal of the project is satisfying all product quality properties included in the ISO/IEC 25010:2011 standard. This means the system should be:

- Functionally suitable
    - This means the tool should be complete and correct in the sense that the user can rely on the fact that the data that the tool provides is correct and useful (gives insight in pull request statistics).
- Perform efficiently
    - This means that a user shouldn’t experience any performance loss in the browser (e.g. lagging) when the tool is running, while the tool should at the same time be able to collect enough data to be functionally suitable.
- Compatible
    - This means that the tool should be able to work with other software. The tool only needs to work in Chrome (per the non-functional requirements), but it should be able to run at the same time as other Chrome extensions.
- Usable
    - This means that the tool should have an understandable and easy to use user interface, so that any user can start using the tool effectively without problems.
- Reliable
    - This means that the tool should be robust, and crash as little as possible. The tool should also be able to restart, without crashing the operating system, if it does crash.
- Secure
    - This means that user data should be stored safely, and anonymously if the user so desires.
- Maintainable
    - This means that the tool should be well written, and properly formatted. Because of this, it can be easily maintained or extended.
- Portable
    - This means that the tool should be able to work on other systems (with little effort). The tool can be ported directly to any machine with the Google Chrome browser. Apart from that it needs to be adaptable in such a way that it can be used in a different browser, without having to rewrite the entire tool.

Apart from meeting the design goals that can be extracted from this ISO standard, we also want to focus on code quality and code review in particular. Because the tool itself is meant to analyze pull requests, we aim to use the tool ourselves in order to improve our own code review practices, as well as to improve the tool in turn with the experience gained from using it.

Software architecture views



Subsystem decomposition (sub-systems and dependencies between them)

We expect the application to consist of a few components:

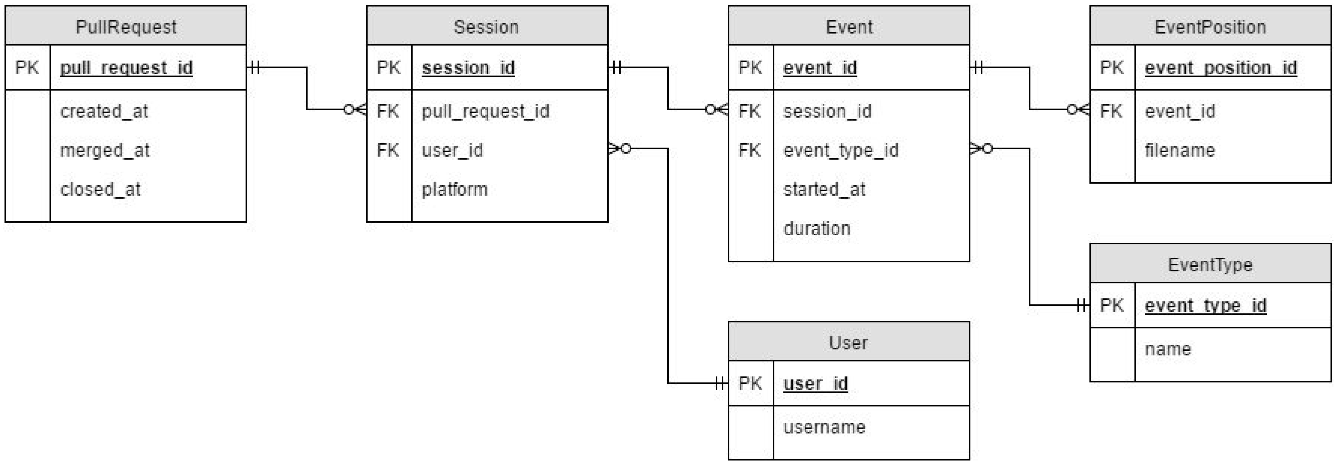
- **Trackers** keep track of the events we want data on, for example mouse movement, window resize, keystrokes but also current user. Should rely on having a connector.
- **Connectors** provide a layer between the tracker and the AJAX calls towards the (database) server.
- The **user interface** is the component that the user interacts with, depends on having trackers to enable & disable.
- A possible extra subsystem is the **analytics component** which would provide a service to the user interface (and possibly other components) by providing information about - for example - a certain pull request and simple statistics. Requires a connector to fetch information from a central server or a connector to a local data source.

Hardware/software mapping (mapping of sub-systems to processes and computers, communication between computers)

This product does not need specific hardware. If the user has Google Chrome as a browser and a host to provide the database hosting, it can start using the extension. An Internet connection to visit BitBucket is also quite helpful, since the extension is used to look at pull request behaviour on BitBucket.

The computers only need a connection to the Internet to communicate with the database and BitBucket. If the database is hosted inside an organization, the computers do not need to connect to an external database. An Internet connection is not optional however, since a connection to BitBucket is still needed.

Persistent data management, database design



Concurrency

Although there are not a lot of concurrency problems, we still need to make sure we do not send duplicate information to the database. We also need to make sure that previous information is removed from the session when the browser is closed.

We will look into problems concerning multiple open tabs or PRs at the same time later.

Glossary

- Bitbucket
    - Hosting provider for Git based code repositories.
- Git
    - A version control system for source code that models version data structure as directed acyclic graphs.
- GitHub
    - Hosting provider for Git based code repositories.
- Pull Request (PR)
    - A request from a developer to incorporate his code changes and /or additions into the existing code base. The new code is typically reviewed by other developers before being merged.