Hashtux Project Summary

Aman Dirar Jerker Ersare Jonas Kahler
Dennis Karlberg Niklas le Comte Marco Trifance
Ivo Vryashkov

BSc. Software Engineering and Management,
Department of Applied IT,
Gothenburg University

January 2016

Contents

Chapter 1

Retrospectives Summary

1.1 Sprint 1

During sprint one we had a lot of discussions concerning our product and what features it should have. Initially we were informed we would start with a two week sprint, but in the middle of the sprint due to unexpected changes we were asked to move to the other slot so our first sprint ended up being only one week long.

During the sprint meetings we learned that we need to ensure that anyone who is absent is updated in what weve been discussing concerning priorities, business case and any other important decisions. We decided to solve this by writing logs of any meetings. We also learned that we need to be better at dividing user stories into tasks and assigning the responsibility of each tasks completion to a team member.

We did not get done that much during this sprint due to the time change mentioned, and the fact that we had two other ongoing courses at the same time. The main accomplishment during sprint one was deciding all the details of the product vision.

1.2 Sprint 2

During sprint two we started different learning tasks we created. As Imed suggested during the previous sprint meeting, we included knowledge acquisition tasks into our backlog, all related to problems and decisions such as which DBMS to use.We held a meeting during the second week of the sprint where we all introduced each other to what we had learned. This also meant that after this sprint we had a lot of technical subjects to put down on the lessons learned report.

We didnt start to produce any serious code during this sprint, most of our time was dedicated to knowledge acquisition, experimentation and research on external packages and APIs. We got briefly acquainted with things like CouchDB, the Cowboy HTTP library, the Bootstrap web framework, and rebar3.

During this sprint we were also more consistent with taking notes during meetings, also including taking photos of notes on design decisions on our whiteboard.

1.3 Sprint 3

During the planning for this sprint we had a meeting with our supervisor, who was concerned about our progress as we were still just transitioning into the coding phase.
At this meeting we split up the group into different subteams, and the development into different areas:

- Dennis and Aman: frontend design and functionality
- Jerker: mid layer and communication between frontend and backend
- Ivo and Marco: mid layer and connecting to social media service APIs
- Jonas and Niklas: database connection and logic

We also created tasks for the first user story we started working on, Twitter search. Since our application has relatively large user stories and centers around a few features that involve the whole application, we worked on this user story in the following sprint as well.

Within the subteams we started out doing some pair programming to make sure that everybody was familiar with the area they were going to work on. This was a good way to share knowledge between those who would keep working closely together during the project.

We met and worked every day, either on the project or an assignment from another course. A small problem we had during this sprint was that a lot of time went into discussions about the different aspects of the product, which could have been used to do some more productive things.

One thing that we took from the sprint meeting with Imed is that we as a group needed to define more clearly what "done means. Since then, we wrote acceptance criteria for all tasks.

This was a really productive sprint for the group, maybe because we were well-prepared from our earlier sprints, every subteam got a lot done.

1.4 Sprint 4

In this sprint we mainly focused on implementation. We managed to integrate distinct parts together and have a first working version, which was important for us because we had a bit of a slow start. We had surprisingly few problems when integrating the different parts (client UI, PHP application + HTTP handler in Erlang, "miners, and database logic).

One thing we learned during this sprint was that we need to address group related problems and talk about them. We had a discussion session and talked about the work distribution and contributions. In particular we had one member of the team that we felt was hard to involve to the same extent as everybody else. We decided to bring this issue up during the sprint meeting, but also reached some conclusions on our own, such as that all of us need to make sure everyone is involved and active in choosing tasks (that are within reach and that is motivating for each person) for each sprint.

Despite that three members in our team were often busy with supervision session for first year students, we still managed to meet regularly, adjusting our schedule to meet everyone's needs. This is one of our groups strengths: that we can communicate in a good way and find a solution that fits all of us.

We finished everything except a few minor details for the Twitter search user story. Simultaneously, we worked on the Instagram search and Youtube search user stories. Since the main infrastructure was now somewhat in place, these user stories were completed with less effort than the Twitter search user story, as was expected.

1.5 Sprint 5

This sprint we put more effort into deciding on tasks suitable for each member of the team and it worked out well. During this sprint we finally had a lot of time to commit to our project. This resulted in good progress in terms of tangible features. During the second week of this sprint we had members of the group that could not be present at the university for our daily meetings due to illness or other personal reasons. This did not disrupt our progress significantly.

At this point we had implemented most of the promised features, which felt nice, but there were still a few features left to create, and we knew we were going to have to spend time solving some bugs, making general improvements and finishing the SAD document.

At the end of this sprint we got the news that we would be assigned a new supervisor and product owner. This change required the team to meet with the new product owner to present the current state of the project and receive any hints or new requirement that could be requested by the new owner. Since the new product owner did not require any major addition or modification to the existing system, we proceeded in our implementation as planned.

1.6 Sprint 6

During this sprint we worked remotely for three days and used Skype, Facebook Messenger and a TODO list in our repository to communicate. This worked really well for us because everyone knew what to do. We had a remote sprint meeting each day where each member presented what they were working on and how it was going. After meeting almost every day in the university it was nice to have a more relaxed period when we could work from home.

We also reviewed the terms of use for the different social media APIs closely. This research required some features to be redesigned. For example, we modified the outlook of the tweets (Twitter posts) rendered in our frontend to be compliant with the display requirements.

During this sprint almost all the implementation work was done and only minor fixes were left, except for the frontend where naturally some features were still not implemented. We felt that we were close to the finish line and started to talk about what we can do to improve the product in the future.

1.7 Sprint 7

This sprint was about finishing up the project - finding and fixing bugs, finishing the statistics page and the SAD document. We were meeting almost every day during this sprint to wrap it up. Learning from our experience from the last project where we started to produce the documentation too late, in this project we planned it from the beginning and made sure that the main architectural decisions were reflected in our documentation. This way of handling documentation allowed all the team members to always be updated on major changes and to have a clear understanding of the main flow and structure of the system.

We split up the work so that a few of us worked on tweaking, resolving bugs, creating the last small features and finishing the product. The rest worked on documentation. This worked well and was efficient because we minimized the risk for merge conflicts or integration problems when we had fewer people working on the product, and the half of the team that worked on documentation could prepare drafts that the other half could later easily review, so that in the end everybody had left their points of view in the documentation.

Chapter 2

Work Distribution and Personal Reflections

2.1 Jerker Ersare

I'm happy to have learned enough Erlang to pass the certification exam, and to get to know some nice libraries and learn about JSON. I generally like having an overview and understanding of large parts of projects, and in this particular project I handled the communication between the front end and the back end, which allowed and required me to understand a lot of what everybody else was working on.

Worked on:

- Rebar configuration (initializes basic application source code and folder structure), dependencies, created the top-level supervisor structure.
- The PHP to Erlang communication, including a HTTP handler in Erlang using Cowboy (http_handler.erl), and PHP code (ajax*.php, request.php, server_manager.php) that connects to the HTTP interface of an available Erlang backend server, which provides an interface for the client code, for example through AJAX requests.
- "Main flow server/worker structure responsible for the overarching business logic in the backend server such as routing a request to miners when appropriate (copied the structure skeleton from Ivos miner server/workers!) (main_flow_*.erl).
- Assisted Dennis with URL rewrites, jQuery requests and gave some hints about parsing JSON.
- A number of fixes and small features on the front end, such as: Made forward and backward browser buttons work, proper browser history handling. Made checks on frontend for illegal characters in search terms. Made a popup window for paying with an integrated Paypal button with different price alternatives.
- The feature of showing trending searches on HashTux + Twitter trends on the front page (programming-wise) (twitter_popular.php, popular.js).

2.2 Aman Ghezai

Working in this project was a challenge but as the same time educative and interesting. It was challenging because I needed to learn new technologies and adapt them as fast as possible in order to implement them in the project. The fact that we were seven people working in a single project was also a new experience and had its challenges for example having to distribute equal amount of task that is motivating and engaging to everyone. During this project I was mainly working in the frontend design and functionality with Dennis and also in handling the statistics and delivering the statistics. This give me the opportunity to learn working with JSON objects and external libraries, handling HTTP requests and web development in general.

Worked on:

- Handling the jQuery requests and parsing into javascript objects for the User Habits Statistics.
- The design of the stats page (stats.css).
- Processing the user habit statistics and presenting it in a table format with the option of time frame in (stats.php).
- Page freezing functionality in (freeze.js) where all the freezing functionalities for the grid and tiles in the (search.php) are handled.
- Worked with sending HTTP request from PHP to Erlang during the first sprints with Jerker, which was then further developed by Jerker in later sprints.

2.3 Jonas Kahler

Looking back, I am very happy about the project and its outcome. As in previous projects, I had the chance to work on my favorite part - the database/data storage. Especially the custom connector Niklas and me wrote is making me very proud.
Working with Erlang and CouchDB was a great choice and I think the experience will be helpful for later employment.
Another thing I really enjoyed was getting into the fault-tolerant mindset of Erlang which allowed me to write the database address server in the way it is written.

Worked on:

- Database setup on the Linux nodes including basic installation of CouchDB as well as configuring the installation.
- Database connector modules handling the REST connection to the database as well as the operations for the database (couch_connector.erl, couch_operations.erl).
- Database redundancy for having multiple copies of our user habit data (db_replicator.erl).
- Database cleanup for assuring compact database files (db_cleaner.erl, db_eraser.erl).
- Supervisor structure of the database module with different supervisors as well as worker skeletons (db_sup.erl, db_worker_sup.erl).
- Dispenser for handling the database requests (db_serv.erl).
- Map functions for fetching results from the database (db_designdocs.erl).
- Filtering results from the database (db_options_handler.erl, db_filter.erl).
- MapReduce module which is capable of rereducing with spawned processes for parsing the statistics (db_reduce.erl).
- Module for converting Twitter dates into an epoch timestamp (dateconv.erl).
- Module for checking database connections (db_addr_serv). This gen_server also orders the databases (the first one available first, second one second, ...).

2.4 Dennis Karlberg

For me this project was very interesting and enjoyable to work on. Even though Erlang is a very interesting language, I decided to mainly focus on the front-end design during this project, but always making sure that I understood what was going on in the back-end. This was due to the fact that within this field, web-development is what interests me the most and this was the only project where I would have the chance to work with it in this program. In terms of responsibilities; I was in charge of the front-end design and functionalities i.e. fetching of data through the PHP connector, grid generation and functionalities, menus etc. One of the more challenging and interesting features that I developed was the grid representation of our data and all of its functionalities. We did not find any external library that suited our vision so I decided to develop one from scratch using JavaScript.

Worked on:

- The visual grid representation of the all data (grid.js, refresh.js, freeze.js, frontpagegrid.js).
- The design of the front page and grid page i.e. HTML/CSS (search.php, index.php, hashtux.css).
- Fetching data through the PHP connector (search.php, index.php).
- All icons, buttons and logos used across all pages (images/*).
- Front-end filtering and option functionalities (options.js).
- Front-end history search functionality (search.php).
- Making sure that we met all display requirements set by the API's we use.

2.5 Niklas le Comte

This project was really interesting to work on. For this project I was mainly working on the database part of the project with Jonas where we started of with pair-programming with the connector to CouchDB and the first operations. Since I have not worked on this part in previous projects it was fun to learn another area of software development. The most fun and frustrating things was erlang. Sometimes there was more struggling moments and sometimes it was going great, but it was really nice to see how efficient erlang is as a language and how much you can improve your own code. I really think that the project turned out great and the teamwork was good within the team.

Worked on:

- Database connector modules handling the REST connection to the database as well as the operations for the database (couch_connector.erl, couch_operations.erl).
- Caching user habit data every hour so that the request for the statistics is retrieved faster with already sorted data (db_cacher.erl).
- All the database workers doing the different operations on the database (db_hash_reader.erl, db_hash_writer.erl, db_userstats_reader.erl, db_userstats_writer.erl).
- Initial state of the db_options_handler that Jonas modified later while I worked on something else (db_options_handler.erl).
- Helped with the statistics page making sure that the query worked for the user habit data and made the JSON into an array of JavaScript objects (stats.php, userstats_fetcher.js).

2.6 Marco Trifance

I consider this project as a good opportunity to learn about distributed systems and Software Architecture. I was mainly responsible for querying the Twitter API and Youtube Data API and for the conversion of fetched data into our internal representation format. This gave me the chance to expand my knowledge in Erlang and the OTP framework, reuse of external dependencies, handle queries to external APIs and finally work with JSON objects. One of the most interesting activities was working with the APIs. Adjusting the API queries to the parameters provided by our application while ensuring that our quality requirements were not affected was sometimes challenging but entertaining. Finally, the investigation of the Terms of Use gave me a broader understanding and awareness of the implications and limitations deriving from working with external APIs.

Worked on:

- Set up Twitter and Youtube applications to allow API queries.
- Handling Twitter API search requests (twitter_search.erl, apis_aux.erl).
- Handling Youtube Data API search requests (youtube_search.erl, apis_aux.erl).
- Processing and filtering the JSON objects returned by the APIs into our internal format - (parser.erl).
- Identify requirements for compliance to APIs Terms of Use.

2.7 Ivo Vryashkov

I found this project very interesting and educational. It was a challenge because of the new way we had to think about a software system - its architecture and making it distributed. I got to work on the middle layer and connecting to the different social media services APIs. This helped me very much in learning Erlang as well as working with external APIs. Furthermore, I learned JSON and how CouchDB works which I consider valuable skills.

Worked on:

- Miner module structure - all miner_*.erl files making the overall structure tree (top supervisor for the miner module and consequent workers and other supervisors) - responsible for handling requests for searching the different social media services.
- Introduced a queue limit for the number of workers processing search requests that can be running on a single server - used to distribute workload (miner_server.erl).
- Handling Instagram search requests - ig_search.erl - and processing and filtering the returned results from the API into an acceptable internal (HashTux) format.
- Handling the returned search results (filtered/unfiltered) from all social media services and sending them back to the main_flow_worker, respective, writing them to the database (miner_dbwriter.erl).
- Database and application installation and management of one of the physical servers (ivo.hashtux.com).