masp

make a simple project

Projektbericht

Dokumentation unserer projektorganisatorischen Vorgehensweise

Sven Baer

Phillipp Engelke

Alexa Grube

Tim Härtel

Moritz Marquardt

Otto von Guericke Universität Magdeburg - 4. Juli 2019

Inhaltsverzeichnis

1 Vorwort

Dies ist der Projektbericht zu unserem Softwareprojekt an der Otto von Guericke Universität Magdeburg, welches wir im Rahmen unseres Bachelorstudiums absolviert haben.

Der Aufbau des Projektberichts ist so gewählt, dass wir erst eine kurze Übersicht über die Projektidee geben und uns dann dem Projektmanagement zuwenden. Dabei gehen wir zuerst auf unser Vorgehen und dann auf die Rollen der einzelnen Personen im Team ein. Auch die zeitliche Abarbeitung der User Stories aus dem Pflichtenheft wird dargestellt.

Am Ende reflektieren wir noch unsere Projektarbeit.

2 Projektidee & Umsetzung

masp ist eine Software, die es vor allem kleinen Teams ermöglichen soll, Projekte mit kleinstmöglichem Mehraufwand für die Organisation umzusetzen. Die Software soll genutzt werden, um Projekte zu planen und während des laufenden Projektes einen Überblick zu behalten, wo der aktuelle Projektfortschritt liegt. Eine Softwareinstanz soll exakt ein Projekt umfassen.

Dabei haben wir einen Schwerpunkt auf die Modularität der Software gelegt, damit es später auch möglich ist, weitere Module zu entwickeln, um die Software auch im beruflichen oder Vereinsumfeld für die ganzheitliche Planung von Projekten sinnvoll einsetzen zu können.

Dieses Ziel konnten wir aus unserer Sicht erfolgreich umsetzen, wir hoffen auf eine erfolgreiche Weiterentwicklung des Projekts außerhalb des Studiums, Pläne für die Zukunft beinhalten beispielsweise eine Zeiterfassung oder die Verwaltung von Budgets & Finanzen.

3 Projektmanagement-Vorgehen

Begonnen haben wir das Projekt mit der Analyse unserer Anforderungen an eine solche Software, und daraus folgend der Erstellung des Pflichtenhefts. Die Anforderungen sind in Form von User Stories direkt in dieses eingegangen.

Da der Großteil des Teams noch nicht mit den von uns gewählten Sprachen gearbeitet hatte, und insgesamt eine hohe Lernmotivation bestand, war für uns klar, dass das ein gemeinsames Lernprojekt wird, in dem wir flexibel auf Probleme und neue Erkenntnisse eingehen müssen. Unter anderem deshalb haben wir uns für ein agiles Projektmanagement entschieden, und das Pflichtenheft nur als Referenz für die von uns anfangs gewünschten Anforderungen verwendet.

Dadurch, dass unsere Anforderungen nicht von einem Kunden vorgegeben wurden, sondern von uns selbst spezifiziert wurden und wir ein Eigeninteresse an der Erfüllung dieser Anforderungen haben, haben wir uns gegen den direkten Einsatz von Scrum entschieden. Wir haben jedoch nur bestimmte Aspekte von Scrum verwendet oder leicht abgeändert, um sie an unsere Bedürfnisse anzupassen.

So haben wir unsere User Stories in einem Product Backlog dargestellt, um unseren Projektfortschritt zu verfolgen und einfach Prioritäten setzen zu können. Bei wöchentlichen Treffen während des Semesters konnten wir über den Fortschritt und die Probleme der einzelnen Teammitglieder sprechen und gegebenenfalls Änderungen am Product Backlog oder unseren Prioritäten vornehmen. Die Arbeit in (flexiblen) Sprints erleichterte es uns zudem, unsere Zeit während des Semesters einteilen zu können.

4 Rollen im Team

Unser Team besteht aus 5 Bachelor-Studenten aus drei verschiedenen Studienrichtungen der Informatik. Moritz Marquardt und Tim Härtel studieren Ingenieurinformatik, Sven Baer studiert Informatik und Alexandra Grube und Phillipp Engelke studieren Wirtschaftsinformatik. Wir alle sind im 6. Fachsemester unseres Bachelorstudiums.

Während unseres Studiums haben wir alle Einführung in die Informatik und Algorithmen und Datenstrukturen belegt und hatten ein wenig Erfahrung in der Programmierung in Java. Da wir uns aber aus Interesse für die moderne Programmiersprache Go von Google entschieden hatten, konnte man zwar Kernkonzepte übertragen, mussten aber einige neue Konzepte erlernen. Da Moritz und Phillipp schon mit der Sprache gearbeitet hatten, konnten Sie dem Rest des Teams in zwei Workshops die grundlegenden Konzepte der Sprache sowie eigene Erfahrungen aus der Entwicklung vermitteln.

Durch den Aufbau des Projekts als Lernprojekt haben darum Alexa, Sven und Tim den größten Teil der eigentlichen Entwicklung neuer Funktionen übernommen, meist jeweils gemeinsam mit entweder Moritz oder Phillipp. Moritz und Phillipp waren darüber hinaus hauptsächlich für Code Reviews in Form der Bearbeitung & Integration von Pull Requests sowie das Beheben von Fehlern verantwortlich.

Die Struktur des Basismoduls entstand in gemeinsamer Arbeit auf Grundlage eines Konzepts von Moritz.
Für das Design waren Alexa und Moritz zuständig, für dessen Implementierung im Frontend Sven, Tim und Moritz.

Am Nutzermodul haben Alexa und Phillipp große Teile der Implementation durchgeführt, während Tim und Moritz sich auf das Backend des Aufgabenmoduls sowie Sven, Moritz und Alexa auf das Frontend des Aufgabenmoduls konzentriert haben.

Durch diese Aufteilung konnten wir eine gleichmäßige Verteilung von Arbeit unter den Teammitgliedern gewährleisten.

5 Arbeit mit dem Code

Für die Zusammenarbeit am Code haben wir uns für eine Lösung mit mehreren Git-Repositories entschieden, da wir das Projekt in mehrere Module geteilt haben.

Für jede Aufgabe wurde ein neuer Branch erstellt, auf dem diese Aufgabe dann implementiert wurde. Wenn jemand der Meinung war, dass seine Aufgabe abgeschlossen war, wurde Moritz oder Phillipp Bescheid gesagt - in Form eines Pull Requests, bei dem der jeweils nicht an der Aufgabe beteiligte zugewiesen wurde. Dieser hat sich dann den Code auf Logikfehler angeschaut und händisch getestet.
Wenn die Tests erfolgreich waren, wurde der Code in den `master`-Branch gemerged.

Da das Projekt während der Entwicklung noch nicht für den Produktiveinsatz bestimmt war, haben wir uns gegen die Verwendung eines `development`-Branches entschieden.

6 Durchführung von Tests

Tests haben wir nach einem Zwei-Augen-Prinzip durchgeführt. So war jeder Entwickler dazu angehalten seinen neuen Code zu testen, bevor er ihn zur Code Review freigibt. Die Code Review wurde immer von einem anderen Nutzer durchgeführt, der möglichst nicht an der Entwicklung der Funktion beteiligt war. Dadurch konnten von vorneherein viele Quellen von Bugs vermieden werden.

Die Software als ganzes wurde daraufhin von uns selbst für die Verwaltung der offenen Aufgaben verwendet, um Fehler zu finden, sowie durch Freunde im privaten Umfeld getestet und wo möglich für sonstige Projekte eingesetzt (z. B. die Administration der FaRaFIN-Server).

Automatische Tests haben wir um Zeit zu sparen nicht angefertigt, wobei man hier hinzufügen muss, dass durch eine von vornherein gute Planung die meisten Fehler vermieden werden konnten, die leicht von automatischen Tests abgefangen werden könnten. Für eine effektive Weiterentwicklung nach Abschluss des Projekts wäre dies jedoch eine sinnvolle Methode für zusätzliche Tests.

7 Zeitlicher Ablauf

Wir haben unser Softwareprojekt in 8 Sprints unterteilt. Jedem Sprint sind verschiedene Aufgaben zugeordnet. Wenn die Aufgabe aus dem Pflichtenheft stammt, steht die Nummer der User-Story in Klammern hinter dem Titel.

1. Sprint

Zeitraum: 07.11.2018 - 05.12.2018

- Modul-Lader [4.1.9.]
- Datenbankverbindung [4.1.7.]
- Mailversand [4.1.8.]
- Logging [4.1.10.]

2. Sprint

Zeitraum: 05.12.2018 - 02.01.2019

- Datenbankverbindung [4.1.7]
  wurde aus dem ersten Sprint weitergeführt
- Speicherung [4.1.13.]
- Webserver [4.1.6.]
- Benutzeroberfläche [4.1.5.]
- Projekt anlegen [4.1.1.]

3. Sprint

Zeitraum 02.01.2019 - 30.01.2019

- Projekteinstellungen [4.1.2.]
- Nutzerlist [4.2.6.]
- UI-Fluss [4.1.4.]

Übertragungen von Aufgaben aus dem dritten in den vierten Sprint erklärt sich durch die Klausurenphase.

4. Sprint

Zeitraum 15.02.2019 - 17.03.2019

- Docker-Image
  Da das Projekt als Docker-Image bereitgestellt wird, muss dieses Image erstellt und konfiguriert werden.
- Globale IDs [4.1.12.]
- Erstellen von Aufgaben [4.3.1.] (Backend)
- Bearbeiten von Aufgaben [4.3.2.] (Backend)
- Löschen von Aufgaben [4.3.3.] (Backend)
- Nutzer erstellen [4.2.4.] (Backend)
- Nutzeranmeldung [4.2.2.] (Backend)
- Rechteabfrage [4.2.10.]

5. Sprint

Zeitraum 17.03.2019 - 22.04.2019

- Metadaten hinzufügen [4.3.6.] (Backend)
- Anlegen des ersten Nutzers [4.2.7.]
- Rechtevergabe [4.2.9.] (Backend)
- Nutzer löschen [4.2.3.] (Backend)
- Nutzer bearbeiten [4.2.1.] (Backend)

6. Sprint

Zeitraum 22.04.2019 - 22.05.2019

- Nutzeranmeldung [4.2.2.] (Frontend)
- Nutzer durchsuchen
  Mit dem Suchfeld kann man jetzt auch die Nutzernamen durchsuchen, wenn man sich in der Nutzerliste befindet.
- Anlegen des ersten Nutzers [4.2.7.] *Passwort für den ersten Nutzer kann bei der Installation gesetzt oder generiert werden*
- Nutzer bearbeiten [4.2.1.] (Frontend)
- Suche/Filtern nach Aufgaben [4.3.4.]
- Exportieren von Aufgaben [4.3.7.]

- Erstellen von Aufgaben [4.3.1.] (Frontend)
- Bearbeiten von Aufgaben [4.3.2.] (Frontend)
- Löschen von Aufgaben [4.3.3.] (Frontend)

7. Sprint

Zeitraum 22.05.2019 - 22.06.2019

- Anonyme Nutzer [4.2.8.]
  Rechte für Bearbeitung wurden aufgeschoben
- Gewichtung von Aufgaben [4.3.8.]
- Metadaten hinzufügen [4.3.6.] (Frontend)
- Sortieren von Aufgaben [4.3.5.]

8. Sprint

Zeitraum 22.06.2019 - 04.07.2019

- Entwicklerhandbuch
- Benutzerhandbuch
- Projektbericht
- Bugfixes
- Logoänderung [4.1.3.]

Noch offen aufgrund niedriger Priorität

- Übersetzung [4.1.11.]
- E-Mail-Verifikation [4.2.5.]
- Suche [4.3.4.]
  Filtern ist implementiert, Suche macht erst bei mehr als einem Inhaltsmodul Sinn
- Rechte für Bearbeitung durch anonyme Nutzer

8 Erfahrungen aus dem Softwareprojekt

Nachdem das Projekt nun abgeschlossen ist, möchten wir einige Dinge reflektieren, die wir daraus lernen konnten:

1. **Dokumentation**: Die User Stories aus dem Pflichtenheft spiegeln zwar gut wieder, was die Software tun soll, jedoch sind sie für die Entwicklung manchmal zu groß, sodass man die Funktion in einem Sprint gar nicht schafft, wenn man nicht vollzeit am Projekt arbeitet. Außerdem sind die User Stories keine gute Spezifikation für die Umsetzung einer API oder einer graphischen Oberfläche.
   → **APIs entwerfen und Mockups für die gesamte Applikation erstellen, bevor man anfängt, Code zu schreiben.**
2. **User Stories**: Durch die Größe der User Stories wurden einige Funktionen zwar im Backend schon für eine andere Aufgabe benötigt, aber im Frontend noch nicht relevant sind. Jedoch war ein Arbeitspaket theoretisch immer eine User Story.
   → **User Stories in kleinere Arbeitspakete aufteilen und zwischen Front- und Backend unterscheiden.**
3. **Nutzer- und Basismodul zusammenfassen**: Wir wollten unsere Software komplett modular aufziehen, dabei haben wir es aber etwas übertrieben. Wir haben das Nutzermodul aus dem Basismodul ausgekoppelt, sodass das Problem auftrat, dass die Funktionen des Nutzermoduls nicht auf der API des Basismoduls definiert werden konnten. Außerdem kann das Basismodul nicht auf das Nutzermodul zugreifen, da es sonst einen Import-Loop gibt
   → **Nutzerverwaltung in das Basismodul integrieren**
4. **Digitale Teamkommunikation**: Wir haben im Team über eine Telegram-Gruppe kommuniziert. Durch die unstrukturierte Natur dieser Kommunikationsart gehen die für einen relevanten Informationen schnell zwischen den irrelevanten unter.
   → **Verwendung eines anderen Kommunikationsmediums mit festen Regeln, in dem die relevanten Informationen leicht zugänglich sind.**
5. **Meetings**: Wir haben uns wöchentlich im Team getroffen. Dabei hatten unsere Treffen keinen Moderator, wodurch die Treffen nicht immer strukturiert abliefen. So wurde zwar über die geplanten und umgesetzten Funktionen und Probleme gesprochen, aber es wurden nicht immer alle neuen Funktionen vorgestellt, auf die andere zugreifen können. Mit einem zusätzlichen Moderator und einer festgelegten Struktur wären die Meetings sehr viel produktiver und effizienter.
   → **Moderator festlegen, der eine Tagesordnung durchsetzt und dafür sorgt, dass die Struktur eingehalten wird.**