Entwicklung

Eingesetzte Technologien

Server

Für die serverseitige Software haben wir uns bewusst gegen XAMPP entschieden, und uns für den Server ExpressJS[1] entschieden. Gründe hierfür u.a. waren die weitaus einfachere, einheitliche Konfiguration in einer Datei, welche von den Teammitgliedern nicht geändert werden muss und die erheblich höhere Antwortgeschwindigkeit.

Git und GitHub

Um effizient und koordiniert in einer Gruppe ein Softwareprodukt entwickeln zu können, haben wir Git bzw. den Dienstleister GitHub[2] als Softwaretool in den Entwicklungsprozess aufgenommen.

Git ermöglicht es, Zwischenstände von Quellcode einer Software zu speichern und dezentral auf einem oder mehreren Servern anderen Teilnehmern den Zugang zu ermöglichen, während auf dem lokalen Gerät weitergearbeitet werden kann.

Die Software nutzt dafür sog. *Commits* (Zwischenstände) welche manuell vom Nutzer auf einem sog. *Branch* erstellt werden.

Ein *Branch* kann dabei als Arbeitsumgebung bzw. Zeitstrahl eines Nutzers verstanden werden, welcher mit einem eindeutigen Namen versehen ist. Dieser enthält jeweils eine Kopie des Gesamtprojektes.

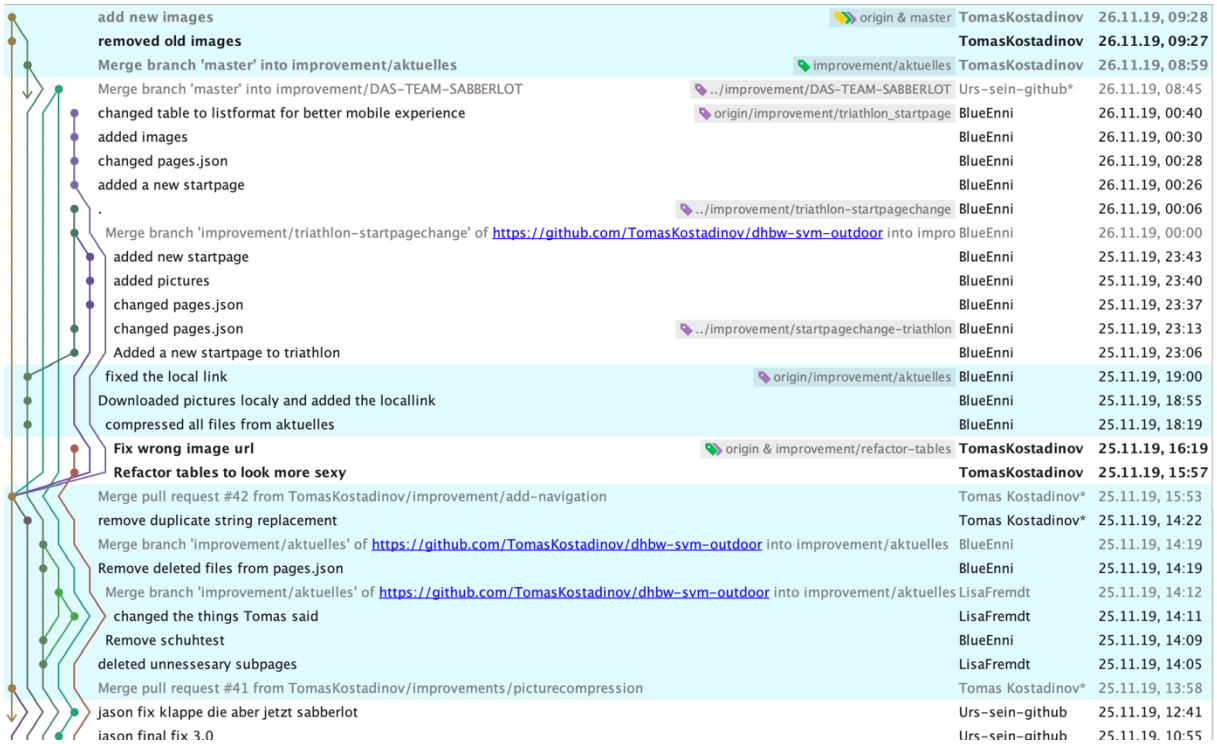

Abbildung 1 Bsp.: Der Entwicklungsverlauf des Projektes, wobei Branches als Linie und Commits als Punkt/Zeile dargestellt wurden.

[1] Siehe http://expressjs.com/de
[2] Siehe http://github.com

Git ermöglicht es, jederzeit zu einem vorherigen Stand zurückzukehren, falls die Entwicklung in eine falsche Richtung gegangen ist oder ein Teil unabsichtlich gelöscht oder geändert wurde.

GitHub erweitert Git um weitere Funktionen und ermöglicht eine Partizipation aller Gruppenmitglieder durch die Möglichkeit der Erstellung von Anmerkungen, Kommentaren und Verbesserungsvorschlägen.

Deployment

Um die Entwicklung zu vereinheitlichen und schnell sichtbare Ergebnisse zu liefern, haben wir uns dafür entschieden, automatisches Deployment mit Heroku[3] zu integrieren.

Dabei wird der jeweils aktuellste Stand des Projektes isoliert und automatisiert auf einem eigenen virtuellen Server bereitgestellt. Für jeden Branch, für den ein Pullrequest (siehe weiter unten) angelegt wurde, wird ebenfalls automatisch ein neuer virtueller Server aufgesetzt. So können klar abgegrenzt jeweils die Änderungen kontrolliert werden, ohne jeweils den Quellcode lokal herunterladen und ausführen zu müssen.

Entwicklungsprozess

Der Entwicklungsprozess lief, bestimmt von Git, dabei folgendermaßen ab:

1. Planung

Das Feature, welches umgesetzt werden soll, wurde durch die Gruppe geplant und ein Verantwortlicher mit der Umsetzung beauftragt.

Beispiel: Teammitglied A wird durch die Gruppe beauftragt, die Triathlonunterseite zu implementieren.

2. Umsetzung

Der für die Umsetzung Verantwortliche erstellt das neue Feature auf einer eigenen Kopie des **master**-Branchs (Anmerkung: Der sog. master ist standartmäßig der erste Branch. Er enthält den aktuellsten, fertigen Quellcode).

Zur Überprüfung wird der Server lokal ausgeführt und visuell das Ergebnis des Codes angeschaut.

Nach Erstellung und Fertigstellung wird der Stand durch den Verantwortlichen *committed* und auf GitHub hochgeladen (sog. *push*).

*Beispiel: Teammitglied A setzt die Triathlonunterseite in HTML und CSS um, committed sie auf dem Branch **improvement/add-triathlon** und pusht sie auf den Server.*

3. Überprüfung

Um das neue Feature in den master-Branch einzufügen, muss ein Pullrequest auf GitHub erstellt werden. Bei diesem gibt der Ersteller desselben an, welche Teile des Projekts warum geändert wurden (Bsp. Vorher-Nachher Screenshot).

*Beispiel: Teammitglied A erstellt einen Pullrequest von **improvement/add-triathlon** nach **master**. Dieser Erhält automatisch eine Nummerierung (#34) und ein Deployment (svm-dh-pr-34.herokuapp.com).*

Um die Qualität des Quellcodes des Projekts sichern zu können, haben wir uns entschieden, dass zu jeder Weiterentwicklung durch ein Teammitglied ein *Code Review* durchgeführt werden muss. Jedes Teammitglied ist dabei berechtigt, ein Code Review abzugeben, es ist jedoch mindestens ein

[3] https://heroku.com

zustimmendes Review notwendig, um die Änderungen zu bestätigen. Der Ersteller kann dabei auch ein Teammitglied bestimmen, welches das Code Review durchführen soll.

Beispiel: Teammitglied A fordert ein Code Review von Teammitglied B an.

Das Code Review besteht aus zwei Teilen:

1. Dem buchstäblichen Code Review

   Dabei wird der Quellcode selbst bzw. dessen Delta (Neue/Gelöschte Zeilen) Zeile für Zeile durchgegangen und, falls notwendig, kommentiert und mit Anmerkungen versehen. Falls Änderungen notwendig sind, wird der Pullrequest temporär abgelehnt, bis diese Umgesetzt wurden.

   Beispiel: In der Datei der Startseite der Triathlon-Mannschaft tritt ein Syntaxfehler auf. Teammitglied B bemerkt dies und fordert Änderungen an.

2. Dem visuellen Review

   Beim visuellem Review klickt sich der Reviewer durch die neuen (und alten) Teile der Webseite auf dem Deployment des Pullrequests. Falls es auf diesem visuelle Probleme gibt, merkt der Reviewer dies in dem zugehörigen Pullrequest an und lehnt diesen ebenfalls temporär ab.

   Beispiel: Eine Bilddatei wird trotz korrekter Syntax nicht geladen. Teammitglied B bemerkt dies und fordert von A Änderungen an.

Wenn alle angemerkten und geforderten Änderungen gemacht – oder etwa keine Notwendig waren – kann der Reviewer den Status des Pullrequest auf *approved* setzen und so in den **master** einfügen.

Dieser wird dann autmatisch deployed.

Optimierungen

Die komplette Webseite wurde darauf optimiert, auf möglichst vielen Endgeräten nutzbar zu sein. Dabei wurde auf Nutzbarkeit, Design und Geschwindigkeit geachtet (so wurden z.B. alle Bilder verlustfrei komprimiert, was zu einer Minderung der Downloadgröße um 70% führte). Änderungen wurden direkt auf Mobil- und weiteren Endgeräten getestet, unter anderem auch auf unterschiedlichen Smartphones mit unterschiedlichen Betriebssystemen oder gar einer SmartWatch.

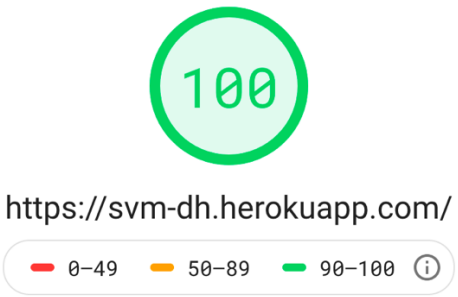

Abbildung 2

Des weiteren wurde die Webseite für Suchmaschinen wie Google optimiert. Auf dem von Google bereitgestellten Testtool „PageSpeed Insights“[4] erhielt unsere Website 100 von 100 möglichen Punkten (d.h. die Bestnote, siehe Abb. 2) wobei Ladezeiten, Design und weitere Optimierungen automatisiert bewertet wurden. Zum Vergleich: die Webseite der Dualen Hochschule Heidenheim erhielt lediglich 60 von 100 Punkten.

Diese Bewertung ist ein großer Faktor in der Bewertung der Relevanz der Seite für Suchmaschinen und somit maßgeblich ausschlaggebend für den Listenplatz auf der Ergebnisseite.

Weitere Optimierungen der Website sind ein sich mit dem Hell-/Dunkelmodus des Betriebssystems anpassender Dunkelmodus der Website und Optimierungen für eine bessere Darstellung der Website in den sozialen Netzwerken mithilfe des Open Graph Protokolls[5].

[4] https://developers.google.com/speed/pagespeed/insights
[5] https://developers.facebook.com/docs/sharing/webmasters?locale=de_DE