Dokumentation

Inhaltsverzeichnis

Einleitung

Wir (die Gruppe Mittwoch-15:15) haben uns dieses Semester für die Entwicklung eines Kalenders entschieden. Zuerst kam uns die Idee, den Kalender anhand seiner Individualisierungsmöglichkeiten hervorstehen zu lassen. Über die Zeit haben wir uns kollektiv entschieden, den Kalender anhand seiner Geschwindigkeit auszuzeichnen. Dies haben wir anhand von Dart (was zu Javascript kompiliert wird) umgesetzt, da die Seite dann nicht bei jeder Umleitung auf neuen Inhalt komplett neu geladen werden muss. Die Inhalte, wie etwa Termine oder Kontakte, werden anhand einer Rest-API geregelt.
Als unsere Leinwand der Kreativität nutzten wir HTML, CSS, Dart (also Javascript), Aqueduct (für die Datenbanken) und PSQL. Aqueduct ist ein Objekt orientiertes, Multi Threaded, HTTP Server Framework, welches wir für die Rest-API genutzt haben.
Den ersten Sprint haben wir zu einem Großteil damit verbracht, uns zu überlegen, welche Software wir nutzen wollen, unsere Workspaces einzurichten, den Papierkram zu erledigen (etwa User Stories in Issues umwandeln) und uns in Dart einzuarbeiten. Dies beinhaltete auch das Erstellen eines Grundgerüsts für unser Projekt in HTML, sodass wir nach dem ersten Sprint schon das grobe Aussehen unserer Software umgesetzt hatten. Der erste Sprint ist auch der Sprint, in dem wir uns am unsichersten waren, wie wir unsere Issues einschätzen sollten, da wir uns dem Ausmaß unserer Aufgaben noch nicht ganz bewusst waren. Dies änderte sich schon in dem zweiten Sprint, da wir nach der ersten Abgabe hilfreiches Feedback bekommen haben.
Im zweiten Sprint stand für uns das Innenleben der Software im Vordergrund. Wir haben uns mit Routing (also der Weiterleitung zwischen den verschiedenen Inhalten/Seiten) auseinandergesetzt und dann separat an den verschiedenen Inhalten gearbeitet. Nun war das Innenleben so weit fertig, dass jeder komfortabel in seinem Abteil arbeiten konnte.
Der dritte Sprint war für uns für das Aussehen und die Datenbankanbindung. Dies nahm viel Zeit in Anspruch und musste mit Hilfe von Engelhardt umgesetzt werden. Unseren Code hat er dabei natürlich nicht verändert, wir setzten uns mit ihm zusammen, um zu sehen, wie wir unsere Idee auf dem Server umsetzen konnte, da Gitlab bis dahin einige Sachen nicht unterstützte, die für unser Projekt vital wären. Wir haben außerdem die Oberfläche überarbeitet und unsere Software auf die Usability-Tests vorbereitet. Hier war etwa die Platzierung der Buttons oder die Größe wichtig.
Zwischen dem dritten und vierten Sprint konnten wir dann unsere Usability-Tests mit Hilfe unserer Schwestergruppe (Mittwoch-14:30) durchführen. Anhand dieser haben wir uns Besserungen abgeleitet, welche wir dann im folgenden und letzten Sprint umgesetzt haben. Zusätzlich haben wir in unserem letzten Sprint kleinere Lücken gefüllt. Abgesehen davon haben wir uns schon auf die Präsentation vorbereitet und die Dokumentation zusammengefügt.

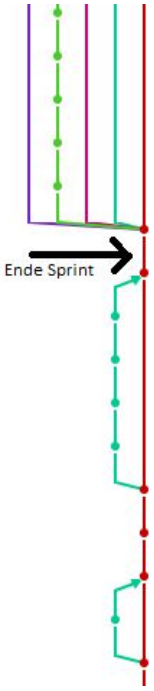
Wir haben am Anfang des Semesters alle nur Dienstags gleichzeitig Zeit gehabt, was dazu führte, dass Dienstagvormittag zu unserem Standardtermin zum Programmieren wurde. Meist wurden zwischen Mittwoch und Samstag (also die folgenden ein bis zwei Tage) die Issues vergeben, welche dann bis zu der darauffolgenden Woche fertig sein sollten. Sollte es Probleme gegeben haben oder etwas nicht fertig geworden sein, so konnte man sich

Dienstags nochmal zusammensetzen und alles besprechen oder zusammen programmieren, wie es in der ersten Woche der Fall war.

Definition of Done

Hier haben wir uns für “manuelle Tests und Kommentiert” entschieden. Dadurch, dass sich unsere Idee nur mit einer Menge Oberfläche realisieren ließe, haben wir entschieden, dass es uns unnötige Arbeit sparen würde, wenn wir die Tests manuell durchführen. Diese werden in den meisten Fällen auch sofort richtig angezeigt, wodurch man erkennen kann, wo die Fehler etwa liegen könnten. Dies bedeutet natürlich nicht, dass nicht getestet wurde. Unit Tests wurden geschrieben und decken den Großteil des Unit Codes ab.

Branching-Model



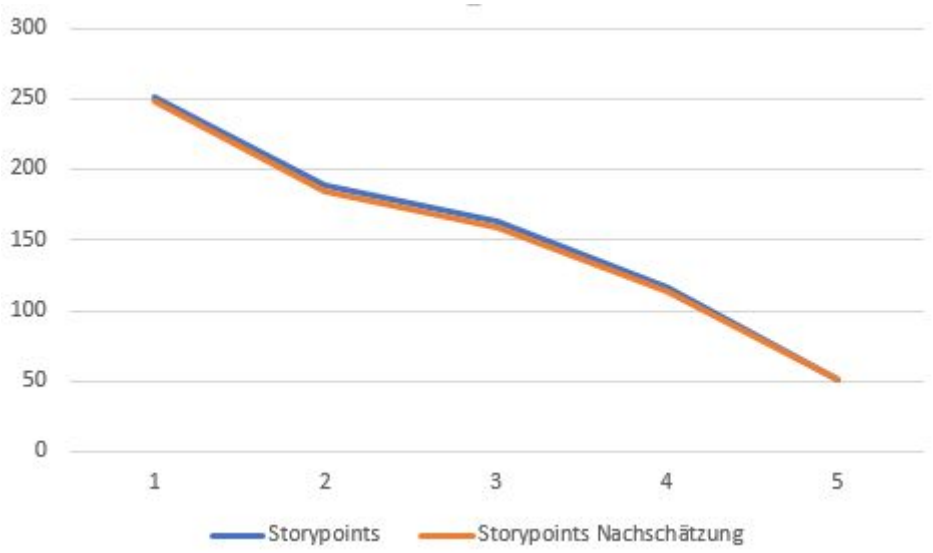
Wir habe uns für das Feature-Branch Model entschieden. In diesem wird jede Addition in einer eigenen Branch entwickelt und bei Fertigstellung mit dem “master”-Branch gemergt. Dies hat den Vorteil, dass man stets neue Features in der laufenden/Master Version besitzt und trotzdem in der eigenen Branch entwickeln kann, was uns dem Konzept des “Rapid Prototyping” am nähesten erschien. Gegen Ende eines Sprints (meist einen Tag vorher) haben wir diese Branches dann gemergt und geschaut, wo es Komplikationen gab. Das Mergen wurde natürlich nicht nur auf einen Tag beschränkt, da wir etwa die erste Woche insgesamt nur eine Branch hatten, um die HTML und das Gerüst unserer Software zu bauen. Hier stand etwa das Erstellen der ReadMe im Vordergrund, was wir zusammen an einem Computer bearbeitet haben. Am Ende des Sprints wurde dann auf die Master Branch gemergt.

Test / Build Automatisierung

Für die Testautomatisierung nutzen wir das Big Bang Verfahren, welches alles auf einmal testet. Unser Programm lässt sich nicht deployen, sollten wir Fehler oder ähnliches im Code haben. Dieses Prinzip funktionierte bei uns gut mit unserem Branching Modell zusammen, da wir dadurch die Features in den Feature Branches testen konnten.
Wir nutzen für die Build Automatisierung Docker Images, welche auf Dart von Google basieren. Docker baut dann anhand von Befehlen die Software, welche in ein Image gespeichert und deployed wird. Auf Maven haben wir verzichtet, Yaml nutzen wir jedoch. Anhand von Yaml können wir festlegen, welche Befehle und in welcher Reihenfolge diese im Build Prozess verwendet werden.

Schätzungen und Velocity

Wir haben unsere Software beim ersten Mal schätzen auf 252 Storypoints geschätzt, welches dann am Ende auf 249 nachgeschätzt wurde. Unsere durchschnittliche Velocity ist 50,5 Storypoints pro Sprint. Anfangs waren wir uns nicht wirklich sicher, wie man mit den Storypoints arbeitet, was wir nach dem zweiten Sprint dann jedoch langsam in unsere Arbeitsweise integriert hatten.



Anhand dieses Burndown Graphen kann man ablesen, dass wir im ersten und im letzten Sprint die meisten Storypoints verarbeitet haben (63 und 66 Storypoints). Am Ende haben wir immer noch einige Storypoints über. Dies sind Ideen, die es am Ende nicht in das Projekt geschafft haben und daher gestrichen wurden. Die Issues bleiben bestehen für den Fall, dass wir den Kalender weiterentwickeln wollten.

Anhand dieses Balkendiagramms können wir die Durchschnittliche Abweichung jedes Sprints einsehen. Am ersten Sprint hatten wir zwar keine große Abweichung, das lässt sich jedoch dadurch erklären, dass es sich hier um einen absoluten Graphen handelt und sich Schätzungen für zu viele und zu wenig Punkte gegenseitig ausgleichen. Die nachfolgenden Wochen handelt es sich dreimal um den selben Wert, dies ist jedoch Zufall, da wir einerseits

Punkte abgearbeitet und nachgeschätzt haben und sich diese meist am Ende eines Sprints einfach auf die selbe Summe beliefen.

Am Ende blieben leider einige unverbrauchte Storypoints über. Dies lässt sich dadurch erklären, dass wir anfangs nicht wussten, wie groß unser Projekt werden würde und am Ende haben wir trotzdem funktionierende Software. Es fiel uns daher nicht schwer diese letzten Punkte als optional anzusehen. Unsere Updatefunktionalität wurde im Middleman noch nicht implementiert, was dazu führt, dass man seine Daten nicht ändern kann.

Usability Test

Wir haben für die Usability Tests einen Testbogen erstellt, dieser sah wie folgt aus:

Usability Test

Aufgaben zum Umgang mit der Software

Alle Aufgaben sind auf der Website https://projects.mylab.th-luebeck.de/calendar/#/login zu lösen.

Aufgabe 1

Erstellen Sie sich einen Account mit Ihren gewünschten Nutzerdaten.

Aufgabe 2

Loggen Sie sich mit Ihrem Nicknamen/Ihrer Email und Ihrem Passwort ein.

Aufgabe 3

Erstellen Sie zwei neue Termine.

Aufgabe 4

Finden Sie in der Kalenderansicht heraus, wie Sie Ihre Termindetails anschauen und bearbeiten können und den Termin bei Bedarf löschen können.

Aufgabe 5

Finden Sie heraus, wie Sie sich all Ihre Termine auf einen Blick anschauen können.

Aufgabe 7

Finden Sie Informationen über Ihren hinzugefügten Kontakt heraus.

Aufgabe 8

Löschen Sie den Kontakt wieder.

Bewertung der Software

Bitte geben Sie Ihre Beurteilung ab.

Um das Produkt zu bewerten, füllen Sie bitte den nachfolgenden Fragebogen aus. Er besteht aus Gegensatzpaaren von Eigenschaften, die das Produkt haben kann. Abstufungen zwischen den Gegensätzen sind durch Kreise dargestellt. Durch Ankreuzen eines dieser Kreise können Sie Ihre Zustimmung zu einem Begriff äußern.

Beispiel:

attraktiv	○	⊗	○	○	○	○	○	unattraktiv

Mit dieser Beurteilung sagen Sie aus, dass Sie das Produkt eher attraktiv als unattraktiv einschätzen.

Entscheiden Sie möglichst spontan. Es ist wichtig, dass Sie nicht lange über die Begriffe nachdenken, damit Ihre unmittelbare Einschätzung zum Tragen kommt.

Bitte kreuzen Sie immer eine Antwort an, auch wenn Sie bei der Einschätzung zu einem Begriffspaar unsicher sind oder finden, dass es nicht so gut zum Produkt passt.

Es gibt keine „richtige“ oder „falsche“ Antwort. Ihre persönliche Meinung zählt!

Bitte geben Sie nun Ihre Einschätzung des Produkts ab. Kreuzen Sie bitte nur einen Kreis pro Zeile an.

behindernd	o o o o o o o	unterstützend
kompliziert	o o o o o o o	einfach
ineffizient	o o o o o o o	effizient
verwirrend	o o o o o o o	übersichtlich
langweilig	o o o o o o o	spannend
uninteressant	o o o o o o o	interessant
konventionell	o o o o o o o	originell
herkömmlich	o o o o o o o	neuartig

Was würden Sie anders machen?

Beschreiben Sie die Website in 3 Worten:

Wie kamen Sie mit den Aufgaben zurecht?

Vielen Dank fürs Mitmachen!

Usability-Test Auswertung und Bericht

Ort des Problems:

- Dashboard: Einen automatischen oder manuellen Refresh einbauen.
- Logout: Den Logout bestätigen, damit man nicht beim versehentlichen Klick auf den Button direkt ausgeloggt wird.
- Registrieren: Ein Feld hinzufügen, in dem das Passwort bestätigt werden muss, falls der Nutzer sich vertippt hat.
- Agenda: Namen ändern, da dieser nicht intuitiv dem Nutzer die Funktion schildert.
- Kontakte:
    - Der Weg zu den Kontaktdetails ist nicht intuitiv/schwer zu finden.
    - Beschreibungen zu den Feldern bei den Kontaktdetails.
    - Checkboxen, bei denen man die Kontakte auswählen kann, um sie danach gemeinsam zu löschen.
    - Suchfunktion für Kontakte.
- Kalender- und Tagesansicht:
    - Die Zahl, die die vorhandenen Termine an einem Tag angibt, besser sichtbar machen/deutlicher machen.
    - Den aktuellen Tag anzeigen lassen/highlighten.

    - Zurück-Button zur Kalenderansicht, wenn man sich über diese einen Termin hat anzeigen lassen.
    - In der Tagesansicht einen Button zum Erstellen eines neuen Termins.
- Termine:
    - Pflichtfelder bei Termin erstellen überdenken (Beispiel: Ort ist Pflicht, Zeit nicht)
    - Constraints für die Felder einrichten (Aktuell kann man bei der Uhrzeit einfach einen String reinschreiben)
    - Ein Notizfeld für die Termine

User-Stories:

Bestimmte User-Stories bei denen Probleme auftraten oder es Verbesserungsvorschläge gab:

- US 2: Abmelden
- US 3: Account erstellen
- US 9: Termin erstellen
- US 13: Termin bearbeiten
- US 18: Kontakt bearbeiten
- US 23: Kalenderansicht

Weiterhin fanden unsere Tester die Software einfach zu bedienen, effizient, intuitiv, übersichtlich und interessant, wobei sie trotzdem herkömmlich und eher konventionell geworden ist. Unter anderem durch unser Dashboard wurde die Software allerdings auch als spannend anstatt langweilig bewertet.

Mit unseren Aufgaben kamen die Tester gut klar.

Umsetzung der Ergebnisse

Von den Vorschlägen, die wir durch die Usability-Tests bekommen haben, haben wir alle Vorschläge umgesetzt bis auf den ersten Punkt bei der Kalender- und Tagesansicht, da es dort Probleme mit CSS gab.
Wir haben uns dafür entschieden, alle anderen Punkte umzusetzen, da diese unserer Meinung nach überschaubar und sinnvoll für die Benutzung und Navigation durch die Software waren und die Umsetzung der Vorschläge vom Zeitaufwand ebenfalls nicht den Rahmen gesprengt haben.