Softwaretechnik II

Dokumentation

AppDish

Eine Webanwendung für kochbegeisterte Menschen, in der sie Rezepte suchen, erstellen, bewerten und speichern können.

Erstellt von:

Qendrim Berisha

Ares Djabbar

Rushwan Fattah

Metin Haji-Hossein

Saykhan Khasukhanov

Alexandra Sowah

Inhaltsverzeichnis

1 *Softwarearchitektur und schwierige Implementierungsprobleme*

Komponenten der Spring Boot Webanwendung

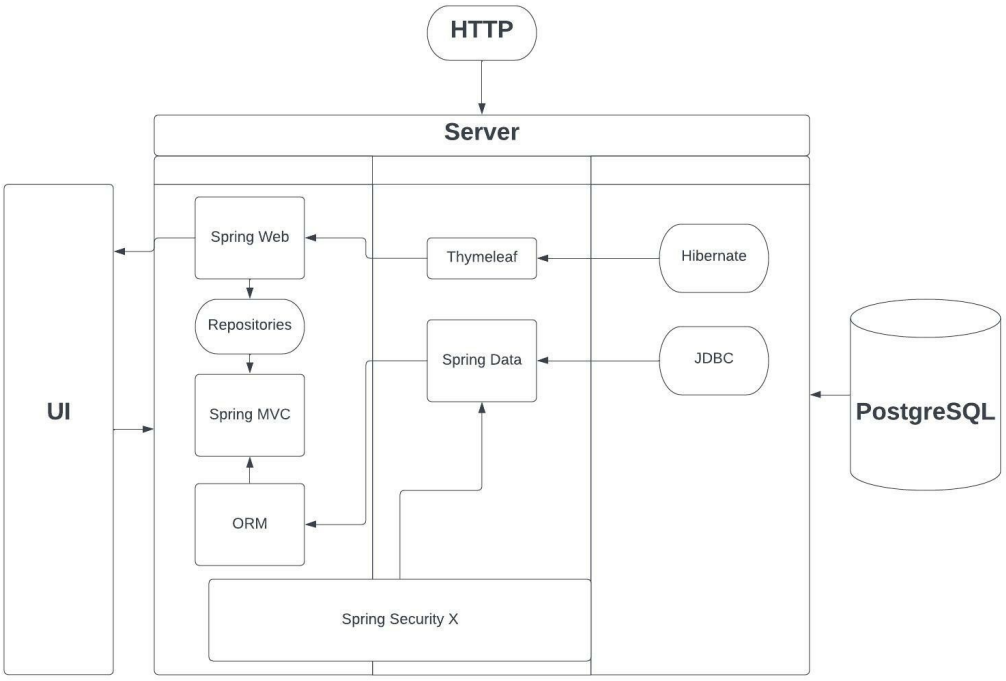
Für unsere Spring Boot Webanwendung wurden folgende Komponenten eingesetzt:

- Spring Boot für die Autokonfiguration und das Self-Containment
- Spring Data JPA mit Hibernate ORM und der Datenbank PostgreSQL
- Spring MVC für die Steuerung der Oberfläche
- Thymeleaf als HTML5 Template Engine mit dem CSS Framework Bootstrap
- Java SE 11 mit Apache Maven und IntelliJ Idea Ultimate

Zudem wurden folgende Verknüpfungen eingesetzt:

- Verbindung über Docker
- Datenbankanschauung über DBeaver Enterprise Edition
- Postman zum erstellen und verwenden von API's

Softwarearchitektur



Schwierige Implementierungsprobleme

Spring Security

Nach anraten von Herrn Hamann, haben wir versucht die Funktion „Login“ über Spring Security umzusetzen. Dies gelang uns anfangs, jedoch brachte es einige Probleme mit sich, wodurch andere Issues nicht korrekt bearbeitet werden konnten.

Aus diesem Grund entschieden wir uns im Nachhinein gegen Spring Security. Hiernach konnten wir auch alle Probleme beheben und somit die weiteren Issues bearbeiten.

2 Testkonzept

Für die Webanwendung wurden zum testen atomarer Einheiten im Quellcode Unit Tests erstellt. Diese lassen sich schnell und automatisiert ausführen, lassen eine gute Fehlerbestimmung zu und sind sehr detailliert.

Unit Tests

Bei den Unit Tests wird die Klasse isoliert getestet, sodass alle Methodenaufrufe mit anderen

Klassen durch Mocks ersetzt werden. Dazu wurde für jede zu testende Klasse eine Klasse erstellt, welche mit dem Namen der zu testenden Klasse benannt wurde, gefolgt von „Test“. Jede Methode in dieser Klasse entspricht einem Test. Dabei sollen ca. 20% des Codes abgedeckt werden.

JUnit 5 und parametrisierte Tests

Mittels JUnit 5, einem Unit-Testing-Framework für Java, wird das Schreiben von Entwicklertests mit Funktionen wir parametrisierte Tests erleichtert. Die Funktion ermöglicht es, eine einzelne Testmethode mehrmals mit unterschiedlichen Parametern auszuführen. Für die Verwendung von parametrisierten JUnit 5-Tests muss dass Artefakt junit-jupiter-params von der Junit-Plattform importiert werden.

3 Definition of Done

Definition of Done

Die Definition of Done enthält folgende Punkte, die mit jedem Sprint eingehalten wurden:

- ✔ Die Akzeptanzkriterien wurden eingehalten
- ✔ Das Projekt wurde auf Gitlab gepusht
- ✔ Die gesamten Unit Tests laufen erfolgreich durch
- ✔ **Coding Guidelines** wurden eingehalten
- ✔ Der Gesamte Code wurde gereviewt (abgenommen durch Product Owner)
- ✔ Die Benutzerdokumentation wurde erstellt und ist verfügbar

Coding Guidelines

- ✔ Code auf englisch
- ✔ Sinnvolle Kommentare
- ✔ Sinnvolle und kompakte Variablennamen
- ✔ Sinnvolles Design-Pattern
- ✔ Sinnvolle Formatierung
- ✔ Kein redundanter Code

4 Branching Modell

Für jedes zu programmierende Issue wurde eine neue Branch erstellt, welche am Ende eines jeden Sprint in den main Branch zusammengeführt wurden.

GitHub Flow Branching Modell

GitHub Flow hat nur Feature Branches und einen main Branch.

Die main Branch enthält den produktionsreifen Code. Die anderen Branches, Feature Branches, enthalten Arbeiten an neuen Funktionen, sowie Fehlerbehebungen und werden wieder in die main Branch zusammengeführt, sobald die Arbeit abgeschlossen und ordnungsgemäß überprüft wurde.

5 Test-, Buildautomatisierung, CI

Testautomatisierung

Die Testautomatisierung erfolgt über JUnit 5 mit parametrisierten Tests. Hierfür wurden Testklassen mit Testmethoden erstellt. Bei den Tests wird die Klasse isoliert getestet, sodass alle Methodenaufrufe mit anderen Klassen durch Mocks ersetzt werden.

Buildautomatisierung

Die Anwendung wurde über das Buildautomatisierungstool Apache Maven gebaut. Dadurch lassen sich Tests ohne weitere Konfiguration über Apache Maven ausführen. Dies mindert das Risiko von Fehlern im Prozess am Ende der Entwicklung und spart somit Zeit.

CI (Continous Integration)

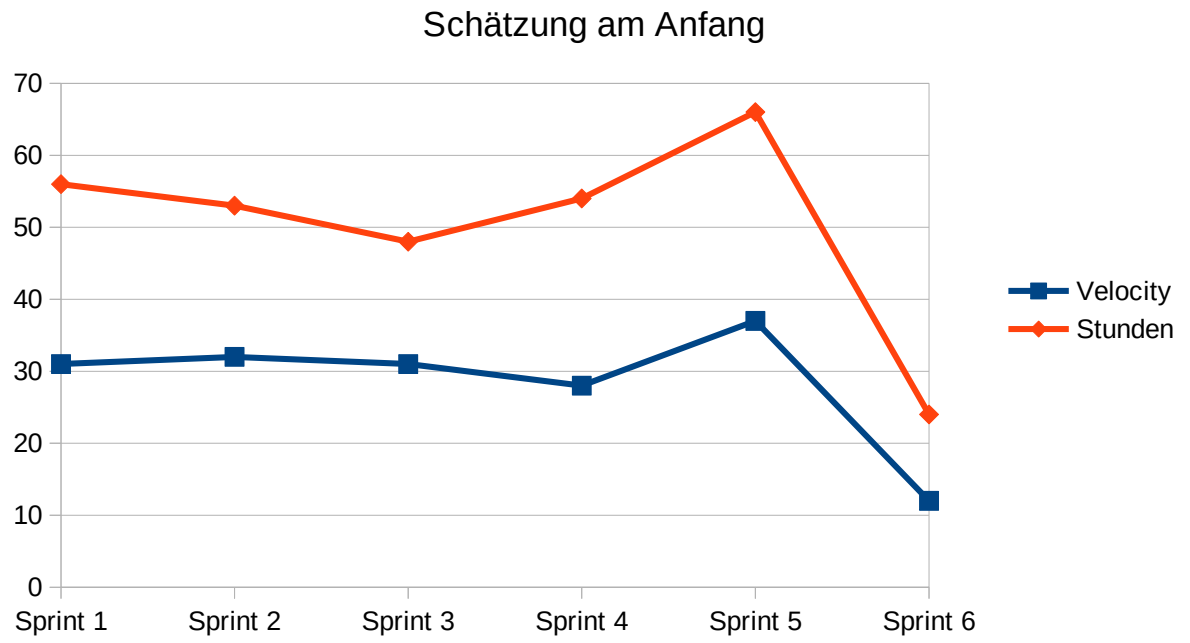
Unsere Versionskontrolle Gitlab bringt ein eigenes Tool für die CI mit. Nach jedem Commit in Gitlab wird die Software kompiliert und die Unit Tests werden ausgeführt. Die CI baut und testet über unser Buildautomatisierungstool Apache Maven.

Mit Hilfe der Build Pipeline, die eine Sequenz von Schritten ist, die nach jeder Änderung ausgeführt wird, erhalten wir kontinuierlich Feedback, da das Durchlaufen der Pipeline abgebrochen wird, sobald ein Schritt einen Fehler verursacht.

6 *Schätzung am Anfang*

Issue	**Story Points**	**Stunden**
Sprint 1		
Schätzen des Backlogs	3	7
Issues mit User Stories	8	13
Schätzung erster Issues	8	13
Etablierung des Prozesses	8	13
Definition of Done	3	7
Anlegen von Lanes	1	4
Summe	**31**	**56**
Sprint 2		
Rezept erstellen	8	13
Rezept ansehen	8	13
Rezept kommentieren	3	7
Server erstellen	13	20
Summe	**32**	**53**
Sprint 3		
Startseite anzeigen	8	13
Seitenmenü anzeigen	5	10
Registrieren	5	10
Kommentareinsicht anzeigen	13	15
Summe	**31**	**48**
Sprint 4		
Rezept bearbeiten	8	13
Anmelden	8	13
Abmelden	3	7
Kommentar löschen	3	7
Registrierung bestätigen	3	7
Rezept löschen	3	7
Summe	**28**	**54**
Sprint 5		
Konto löschen	3	7
Benutzerkonto bearbeiten	8	13
Nutzerverwaltung öffnen	5	10

Eigene Rezepte anzeigen	8	13
Rezept suchen	8	13
Rezept bewerten	5	10
Summe	**37**	**66**
Sprint 6		
Usability Tests	3	7
Kommentare bearbeiten	1	4
Ordner für Rezeptsammlung erstellen	5	10
Codefeinschliff	3	7
Dokumentation	8	13
Summe	**20**	**41**
Insgesamt	**171**	**301**

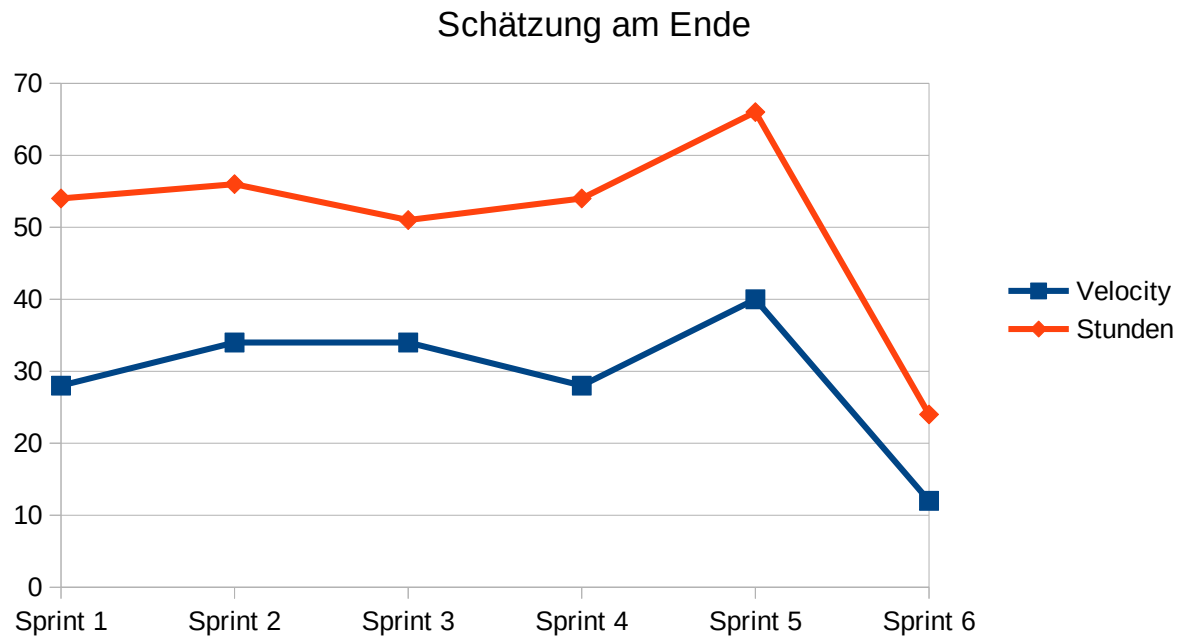


7 *Schätzung am Ende*

Issue	**Post Story Points**	**Stunden**
Sprint 1		

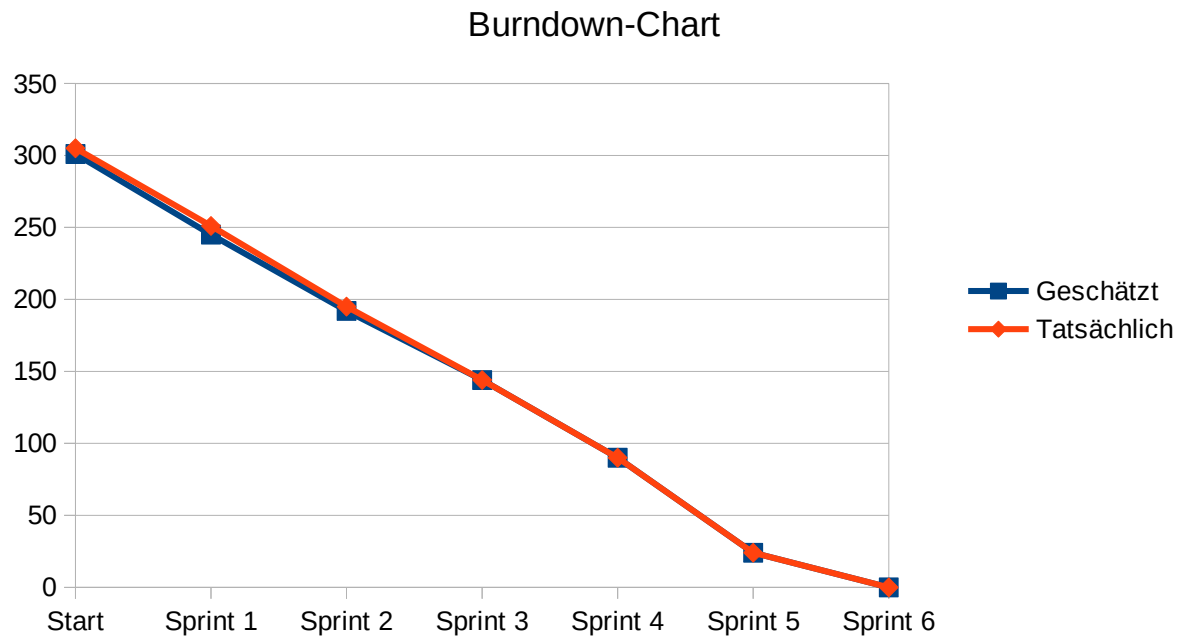
Schätzen des Backlogs	3	7
Issues mit User Stories	8	13
Schätzung erster Issues	5	10
Etablierung des Prozesses	8	13
Definition of Done	3	7
Anlegen von Lanes	1	4
Summe	**28**	**54**
Sprint 2		
Rezept erstellen	8	13
Rezept ansehen	5	10
Rezept kommentieren	8	13
Server erstellen	13	20
Summe	**34**	**56**
Sprint 3		
Startseite anzeigen	8	13
Seitenmenü anzeigen	8	13
Registrieren	5	10
Kommentareinsicht anzeigen	13	15
Summe	**34**	**51**
Sprint 4		
Rezept bearbeiten	8	13
Anmelden	8	13
Abmelden	3	7
Kommentar löschen	3	7
Registrierung bestätigen	3	7
Rezept löschen	3	7
Summe	**28**	**54**
Sprint 5		
Konto löschen	3	7
Benutzerkonto bearbeiten	8	13
Nutzerverwaltung öffnen	5	10
Eigene Rezepte anzeigen	8	13
Rezept suchen	8	13
Rezept bewerten	5	10
Summe	**37**	**66**
Sprint 6		

Usability Tests	3	7
Kommentare bearbeiten	1	4
Ordner für Rezeptsammlung erstellen	5	10
Dokumentation	8	13
Summe	**17**	**34**
Insgesamt	**178**	**315**



8 *Burndown-Chart*

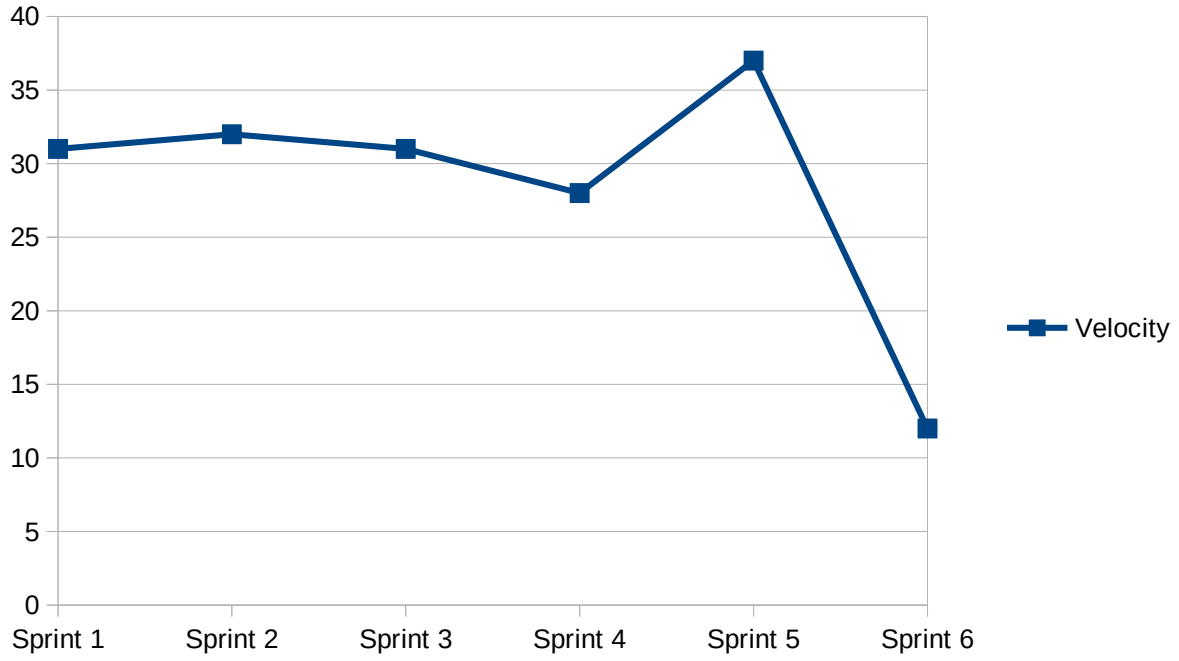
Sprint	**Geschätzt**	**Tatsächlich**
Start	301	305
Sprint 1	245	251
Sprint 2	192	195
Sprint 3	144	144
Sprint 4	90	90
Sprint 5	24	24
Sprint 6	0	0



9 *Velocity des Teams*

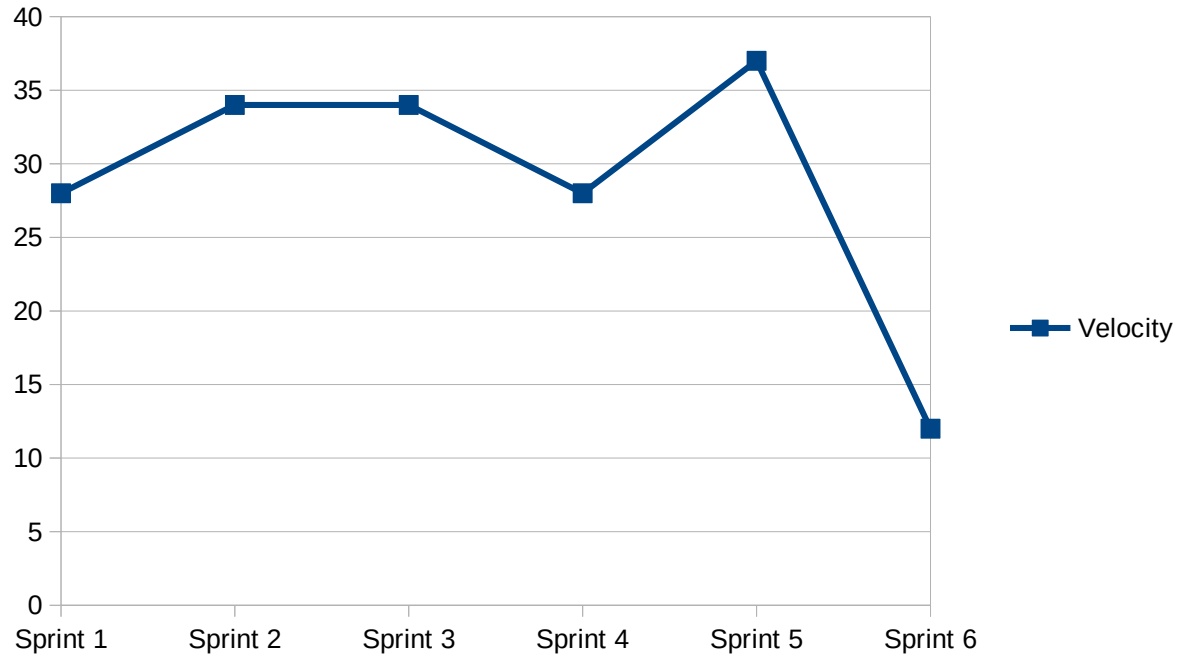
Velocity – Story Points

Sprint	**Velocity**
Sprint 1	31
Sprint 2	32
Sprint 3	31
Sprint 4	28
Sprint 5	37
Sprint 6	12



Velocity – Post Story Points

Sprint	Velocity
Sprint 1	28
Sprint 2	34
Sprint 3	34
Sprint 4	28
Sprint 5	37
Sprint 6	12



10 Kurze schriftliche Reflektion über Probleme bei der Anwendung der gelehrten Konzepte im Sprint

Pipeline

Das größte Problem bei der Umsetzung der Pipeline, welche automatisierte Tests durchführen sollte, war das Gitlab keinen Zugriff auf unsere Datenbank hatte, da wir mit einer lokalen Datenbank gearbeitet haben. Das Problem haben wir erkannt, indem wir die Pipeline langsam runtergebrochen und einzelne Teile des Codes ausgegliedert haben. Durch einsetzen einer richtigen PostgreSQL-Datenbank lief die Pipeline dann ohne Probleme.

11 Protokoll der Tätigkeiten der einzelnen Gruppenmitglieder mit Einschätzung des Anteils an der Gesamtleistung

Qendrim Berisha		
Aufgabe	**Post Story Points**	**Stunden**
Schätzen des Backlogs	3	7
Issues mit User Stories	8	13
Schätzung erster Issues	5	10
Etablierung des Prozesses	8	13
Definition of Done	3	7
Anlegen von Lanes	1	4
Rezept erstellen	8	13
Rezept ansehen	5	10
Product Owner – Sprint 2	1	4
Startseite anzeigen	8	13
Seitenmenü anzeigen	8	13
Registrieren	5	10
Rezept bearbeiten	8	13
Konto löschen	3	7
Benutzerkonto bearbeiten	8	13
Nutzerverwaltung öffnen	5	10
Eigene Rezepte anzeigen	8	13
Usability Tests	3	7
Ordner für Rezeptsammlung erstellen	5	10
Dokumentation	8	13
Product Owner – Sprint 6	1	4
Insgesamt	**112**	**207**

Ares Djabbar		
Aufgabe	**Post Story Points**	**Stunden**
Schätzen des Backlogs	3	7
Issues mit User Stories	8	13
Schätzung erster Issues	5	10

Etablierung des Prozesses	8	13
Definition of Done	3	7
Anlegen von Lanes	1	4
Rezept kommentieren	8	13
Server anlegen	13	20
Scrum Master – Sprint 2	1	4
Kommentareinsicht anzeigen	13	20
Scrum Master – Sprint 3	1	4
Anmelden	8	13
Abmelden	3	7
Kommentar löschen	3	7
Rezept suchen	8	13
Product Owner – Sprint 5	1	4
Insgesamt	**87**	**159**

Rushwan Fattah		
Aufgabe	**Post Story Points**	**Stunden**
Schätzen des Backlogs	3	7
Issues mit User Stories	8	13
Schätzung erster Issues	5	10
Etablierung des Prozesses	8	13
Definition of Done	3	7
Anlegen von Lanes	1	4
Rezept kommentieren	8	13
Server anlegen	13	20
Kommentareinsicht anzeigen	13	20
Anmelden	8	13
Abmelden	3	7
Kommentar löschen	3	7
Product Owner – Sprint 4	1	4
Rezept suchen	8	13
Usability Tests	3	7
Kommentare bearbeiten	1	4
Dokumentation	8	13

Insgesamt	**97**	**175**

Saykhan Khasukhanov		
Aufgabe	**Post Story Points**	**Stunden**
Schätzen des Backlogs	3	7
Issues mit User Stories	8	13
Schätzung erster Issues	5	10
Etablierung des Prozesses	8	13
Definition of Done	3	7
Anlegen von Lanes	1	4
Rezept erstellen	8	13
Rezept ansehen	5	10
Startseite anzeigen	8	13
Seitenmenü anzeigen	8	13
Registrieren	5	10
Product Owner – Sprint 3	1	4
Registrierung bestätigen	3	7
Scrum Master – Sprint 4	1	4
Konto löschen	3	7
Benutzerkonto bearbeiten	8	13
Nutzerverwaltung öffnen	5	10
Eigene Rezepte anzeigen	8	13
Usability Tests	3	7
Ordner für Rezeptsammlung erstellen	5	10
Dokumentation	8	13
Insgesamt	**107**	**201**

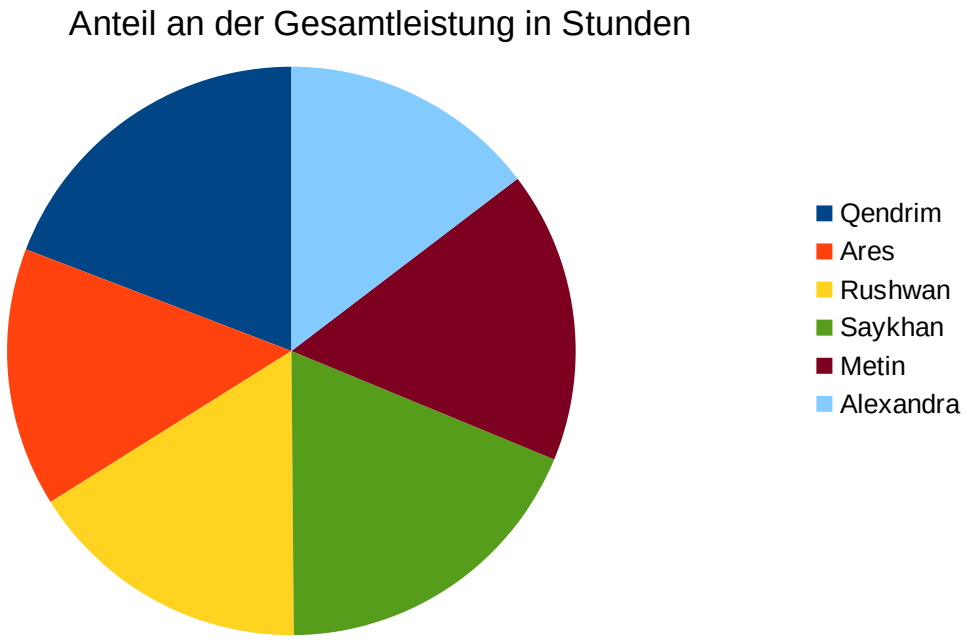
Metin Haji-Hossein		
Aufgabe	**Post Story Points**	**Stunden**
Schätzen des Backlogs	3	7
Issues mit User Stories	8	13
Schätzung erster Issues	5	10

Etablierung des Prozesses	8	13
Definition of Done	3	7
Anlegen von Lanes	1	4
Scrum Master – Sprint 1	1	4
Rezept kommentieren	8	13
Server anlegen	13	20
Kommentareinsicht anzeigen	13	20
Anmelden	8	13
Abmelden	3	7
Kommentar löschen	3	7
Rezept suchen	8	13
Scrum Master – Sprint 5	1	4
Usability Tests	3	7
Kommentare bearbeiten	1	4
Dokumentation	8	13
Insgesamt	**98**	**179**

Alexandra Sowah		
Aufgabe	**Post Story Points**	**Stunden**
Schätzen des Backlogs	3	7
Issues mit User Stories	8	13
Schätzung erster Issues	5	10
Etablierung des Prozesses	8	13
Definition of Done	3	7
Anlegen von Lanes	1	4
Product Owner – Sprint 1	1	4
Rezept erstellen	8	13
Rezept ansehen	5	10
Startseite anzeigen	8	13
Seitenmenü anzeigen	8	13
Registrieren	5	10
Rezept löschen	3	7
Rezept bewerten	5	10
Usability Tests	3	7

Dokumentation	8	13
Scrum Master – Sprint 6	1	4
Insgesamt	**83**	**158**

Name	**Stunden**	**Prozent**
Qendrim Berisha	207	19,18443
Ares Djabbar	159	14,73587
Rushwan Fattah	175	16,21872
Saykhan Khasukhanov	201	18,62836
Metin Haji-Hossein	179	16,58943
Alexandra Sowah	158	14,64319



12 Usability: Planungsdokumente (Leitfaden, zeitliche Planung, Methoden), Bericht

Planung

Guten Tag TesterIn!

Vielen Dank, dass Sie sich bereit erklärt haben an unseren Usability Tests teilzunehmen.

AppDish ist eine Webseite, erstellt von 6 Studenten der Technischen Hochschule Lübeck. AppDish ist eine Seite in der sowohl Besucher als auch User ihre eigenen Rezepte teilen können.

Wir würden Sie gerne bitten Die Webseite anhand der folgenden Aufgaben zu testen und uns Ihr Feedback da zu lassen.

- Registrieren Sie sich auf der Seite Appdish.
- Loggen Sie sich hiernach mit Ihren Anmeldedaten ein.
- Erstellen Sie ein Rezept.
- Öffnen Sie Ihr Rezept.
- Bearbeiten Sie Ihr Rezept.
- Kommentieren Sie Ihr Rezept.
- Bearbeiten Sie Ihren Kommentar.
- Löschen Sie Ihr Kommentar.
- Schauen Sie sich Ihre eigenen Rezepte an.
- Löschen Sie Ihr Rezept.
- Melden Sie sich ab.
- Bedienen Sie die Seite als unangemeldeter Besucher.

Nun bitten wir Sie, uns folgendes Fragen, die uns als Feedback dienen, zu beantworten:

- Beschreiben Sie AppDish mit 3 Worten.
- Wie schwer war AppDish zu bedienen?
- Welche Anforderungen haben Sie an AppDish?
- Wie fanden Sie das Design?
- Was würden Sie anders machen?

Sollten Sie weitere Vorschläge/Anregungen/Kritik zu der Webseite haben, würden wir Sie bitten, dieses zu äußern.

Vielen Dank für die Teilnahme!

Ergebnisse

TesterIn 1:

Beschreiben Sie Appdish mit 3 Worten:
- praktisch, übersichtlich, einfach

Wie schwer war Appdish zu bedienen?
- Die Webseite ist einfach zu bedienen

Welche Anforderungen haben Sie an Appdish?
- Rezepte ansehen, bewerten, in einem Ordner speichern

Wie fanden Sie das Design?
- Ansprechend, passend zur Idee der Webseite

Was fanden Sie gut? Was würden Sie anders machen?
- Ich finde es gut, dass man Rezepte ansehen kann, ohne angemeldet zu sein und dass Gerichte eine Durchschnittsbewertung
- Ich würde ändern, dass die höchstbewerteten Gerichte ganz oben angezeigt werden

TesterIn 2:

Beschreiben Sie Appdish mit 3 Worten:
- einfach, ansprechend, kreativ

Wie schwer war Appdish zu bedienen?
- gar nicht schwer, einfach zu folgen

Welche Anforderungen haben Sie an Appdish?
- Rezepte erstellen/ansehen, löschen und kommentieren

Wie fanden Sie das Design?
- eintönig, dennoch passend

Was fanden Sie gut? Was würden Sie anders machen?
- Ich finde es gut, dass man Rezepte bewerten kann
- Mir fehlen entsprechende Bilder zu den Gerichten

TesterIn 3:

Beschreiben Sie Appdish mit 3 Worten:
- logisch, passend, einfach

Wie schwer war Appdish zu bedienen?
- nicht kompliziert, einfach zu bedienen

Welche Anforderungen haben Sie an Appdish?
- Zutaten und Bilder zu den Gerichten

Wie fanden Sie das Design?
- schön, passende Bilder

Was fanden Sie gut? Was würden Sie anders machen?
- Ich finde es gut, dass man jedem erstellten Rezept einen Titel geben kann
- Ich würde nichts ändern

TesterIn 4:

Beschreiben Sie Appdish mit 3 Worten:
- vielseitig, öffentlich, einfallsreich

Wie schwer war Appdish zu bedienen?
- komplex, jedoch nicht schwer

Welche Anforderungen haben Sie an Appdish?
- Rezepte suchen/ansehen, erstellen und bewerten

Wie fanden Sie das Design?
- gut, erinnert an eine andere Rezeptwebseite

Was fanden Sie gut? Was würden Sie anders machen?
- Ich finde es gut, dass man die Zubereitung zu einem Gericht sieht
- Ich würde anstatt Sterne Likes/Dislikes bevorzugen

TesterIn 5:

Beschreiben Sie Appdish mit 3 Worten:
- Eine gute Webseite

Wie schwer war Appdish zu bedienen?
- geht so

Welche Anforderungen haben Sie an Appdish?
- Meine Anforderungen an Appdish sind, dass ich Rezepte ansehen kann, die Zubereitung verständlich erklärt wird und ich somit die Rezepte nachkochen kann

Wie fanden Sie das Design?
- gut, leider kein darkmode möglich

Was fanden Sie gut? Was würden Sie anders machen?
- Ich finde es gut, dass man ein Gericht nach der Suche filtern kann
- darkmode-Option, Übersetzer

Auswertung

Positiv:
- einfache Bedienung
- Konzept wurde verstanden
- Anforderungen wurden zu 80% erfüllt
- Design war passend
- Beschreibung zu AppDish war plausibel

Negativ:
- keine Bilder zu den Gerichten
- Design ist nicht auffällig
- Bewertungssystem ist alt

Handlungsempfehlungen:
- Das Bewertungssystem besteht aus Sternen, da jeder seinen eigenen Geschmack hat und „dislikes“ führt zu einem negativen Einfluss zu einem Gericht/User.
- Bei der Webseite wurde vorwiegend auf die Funktionalität geachtet und ein einfaches Design gewählt.

Identifizierte Probleme:
- Das Team hat sich dazu entschieden, Bilder mit hinein zu programmiert, da die Visualisierung der Gerichte fehlt.
- Die Anzeige der gesuchten Rezepte sollte eine Reihenfolgenfunktion haben, sodass die bestbewertesten Gerichte oben stehen.

13 Link

Link zu unserem Gitlab Repository

https://git.mylab.th-luebeck.de/saykhan.khasukhanov/swtii