



HTBL Imst
AUFBAULEHRGANG FÜR INFORMATIK



Übungszettel – Prozessmanagement

Name: Michael Bogensberger

Datum: 20.04.2022

1	Inhalt	
2	Was ist Scrum?	3
3	Ereignisse beim Scrum	3
3.1	Spring Planning	3
3.2	Daily Scrum	3
3.3	Sprint Review	3
3.4	Sprint Retrospective	3
3.5	Backlog Refinement	3
4	Rollen im Scrum	3
4.1	Product Owner	4
4.2	Scrum Master	4
4.3	Entwicklungsteam	4
5	Artefakte beim Scrum	4
5.1	User Stories	4
5.2	Product Backlog	4
5.3	Sprint Backlog	4
5.4	Product Increment	4
5.5	Burndown-Chart	5
6	Scrum in GitHub	6
6.1	ZenHub	6
6.2	Abwicklung in GitHub	7
7	User-Stories	7
8	Agiles PM, Versionskontrolle und CI	8
8.1	Trunkbased Development	8
8.2	CI/CD & Pipelines	8
8.3	Gitflow vs. Trunk-basierte Entwicklung	9
8.4	Vorteile der Trunk-basierten Entwicklung	9
8.5	GitHub Actions	9
9	Agiles PM in GitHub	9
10	Abbildungsverzeichnis	11

2 Was ist Scrum?

Scrum ist eine Methode für die agile Produktentwicklung. Scrum erfreut sich zunehmend an großer Beliebtheit. Bei Scrum muss das Team nicht nur effizient zusammenarbeiten, um ein Ziel zu erreichen, sondern auch flexibel und schnell auf Veränderungen im Spiel reagieren.

Scrum setzt auf hochqualifizierte und interdisziplinäre Teams. Das Ziel ist vorgegeben, allerdings kann das Team entscheiden, wie dieses umgesetzt wird. Man arbeitet empirisch (erfahrungsbasiert), inkrementell (in kleinen Schritten) und iterativ (wiederholend). Die Projektlaufzeit ist eingeteilt in zwei- bis vierwöchige Sprints.

3 Ereignisse beim Scrum

Ein Spring besteht grundsätzlich aus den Folgenden Ereignisse (Meetings):

- Spring Planning
- Daily Scrums
- Spring Review
- Sprint Retrospective
- Backlog Refinement

3.1 Spring Planning

Das Sprint Planning ist ein Treffen zur Definition eines Sprint-Ziels zur Erreichung des Produkt-Ziels und zur Planung der Backlog Items, die im aktuellen Sprint umgesetzt werden sollen.

3.2 Daily Scrum

Das Daily Scrum ist ein tägliches Meeting, zu dem sich die Entwickler treffen, um sich gegenseitig über den Fortschritt in Richtung des vereinbarten Sprint-Ziels, die anstehenden Tätigkeiten und mögliche Probleme bzw. Hindernisse auszutauschen.

3.3 Sprint Review

Das Sprint Review ist ein Treffen am Ende eines Sprints zur Begutachtung der erledigten Arbeit in Bezug auf das gesteckte Sprint-Ziel.

3.4 Sprint Retrospective

Die Sprint Retrospektive dient dem Rückblick auf den vergangenen Sprint und der Frage, wie Verbesserungen in den Bereichen: Produkt, Werkzeuge, Prozesse, Beziehungen und beteiligte Personen herbeigeführt werden können.

3.5 Backlog Refinement

Das Backlog Refinement ist ein kontinuierlicher Prozess zur Pflege und Weiterentwicklung des Product Backlogs mit dem Ziel, die Inhalte des Backlogs so aufzubereiten, dass sie sich gut für das Sprint Planning nutzen lassen.

4 Rollen im Scrum

Die drei zentralen Rollen in Scrum sind Product Owner, Scrum Master und Entwicklungsteam. Jeder davon kommt dabei eine eigene Management-Funktion mit spezifischen Verantwortlichkeiten und Aufgaben zugewiesen. Nur wenn diese perfekt miteinander harmonieren, können Projekte erfolgreich verlaufen.

4.1 Product Owner

Die Aufgabe des Product Owner besteht darin, die Interessen der Anwender und Stakeholder (User Stories) genau zu kennen und konsequent zu vertreten. Dazu betrachtet er das Vorhaben strikt aus deren Perspektive.

4.2 Scrum Master

Der Scrum Master fungiert im Scrum-Prozess als Moderator und Dienstleister und organisiert die Kommunikation des Entwicklungsteams mit der „Außenwelt“. Ihm obliegt es, über die Einhaltung der Werte und Regeln eines Projekts zu wachen und geeignete Rahmenbedingungen für einen erfolgreichen Projektverlauf zu schaffen.

4.3 Entwicklungsteam

Das Team in Scrum setzt sich in aller Regel aus fünf bis zehn Projekt-Mitarbeitern zusammen und ist interdisziplinär aufgestellt. Spezifische Hierarchien zwischen den einzelnen Kompetenzbereichen sind dabei bis auf Weiteres nicht vorgesehen. Während der Sprints organisiert sich das Team selbst und realisiert eigenverantwortlich die jeweils geforderten neuen Produkt-Inkremente.

5 Artefakte beim Scrum

Beim Scrum gibt es mehrere wichtige Artefakte. Darunter zählen unter anderem User Stories, Product Backlog, Sprint Backlog, Product Increment und Burndown-Charts.

5.1 User Stories

Eine User Story ist eine in Alltagssprache formulierte Software-Anforderung. Sie ist bewusst kurz gehalten und umfasst in der Regel nicht mehr als zwei Sätze.

5.2 Product Backlog

Das Product Backlog besteht aus einer Liste mit Punkten (User Stories), die bei der Entwicklung des Produkts erledigt werden müssen. Im Product Backlog werden Änderungen und Ergänzungen aufgeführt, die auf das Produkt anzuwenden sind.

5.3 Sprint Backlog

In jedem Sprint soll ein funktionsfähiges Zwischenprodukt entwickelt werden. Deshalb wird bereits vorab im Sprint Planning Meeting entschieden, welche Anforderungen aus dem Product Backlog im nächsten Sprint bearbeitet werden sollen. Nachdem die Anforderungen ausgewählt wurden, werden sie im jeweiligen Sprint Backlog festgehalten.

5.4 Product Increment

Ein Inkrement ist das aktuelle fertig gestellte Produkt. Es besteht also aus allen umgesetzten Arbeiten der vorangegangenen Sprints. Es ist ein Produkt in seiner Entstehung. Am Ende eines jeden Sprints muss ein neues Increment entstehen.

5.5 Burndown-Chart

Ein Burn-Down-Chart ist eine grafische Darstellung für den verbleibenden Aufwand in einem Projekt, in Relation zur verbleibenden Zeit.

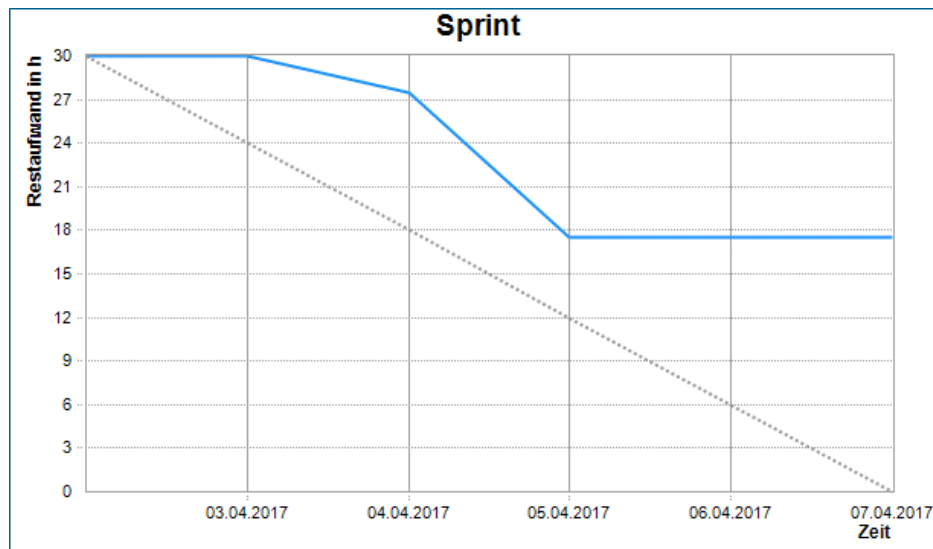


Abbildung 1 Burndown-Chart

In diesem Beispiel sehen Sie, dass

- der Sprint eine Länge von 5 Tagen hat (siehe x-Achse mit der Zeitangabe)
- und der Aufwand der Tasks zu Anfang des Sprints 30 Stunden entspricht.
- Der Idealaufwand ist die graue, linear verlaufende Kurve. Die blaue Linie ist der Restaufwand. Was können Sie von diesem Burn Down Chart ablesen?
- An Tag 1 entspricht der Restaufwand noch 30 Stunden. Die Entwickler haben höchstwahrscheinlich an ihren Tasks gearbeitet, aber schätzen am Ende des Arbeitstages, dass sie noch genau so viel Aufwand wie am Morgen haben werden.
- An Tag 2 wird der Restaufwand geringer geschätzt. Sie können sehen, wie der Restaufwand noch 27,5 Stunden beträgt.
- Am dritten Tag geben die Entwickler eine positivere Prognose: Der Restaufwand fällt um 10 Stunden auf 17,5 Stunden
- Was passiert aber an Tag 4 und 5? Plötzlich sieht es nach Stillstand aus, so, als hätte niemand mehr gearbeitet.

6 Scrum in GitHub

Seit Oktober 2016 bietet GitHub eine Möglichkeit, GitHub Issues, Pull Requests und Notes mit Projects zu verfolgen. Mit GitHub-Projekten können Sie Boards im Kanban-Stil für die Verwaltung der Arbeit haben und separate Code-Repositories aufteilen.

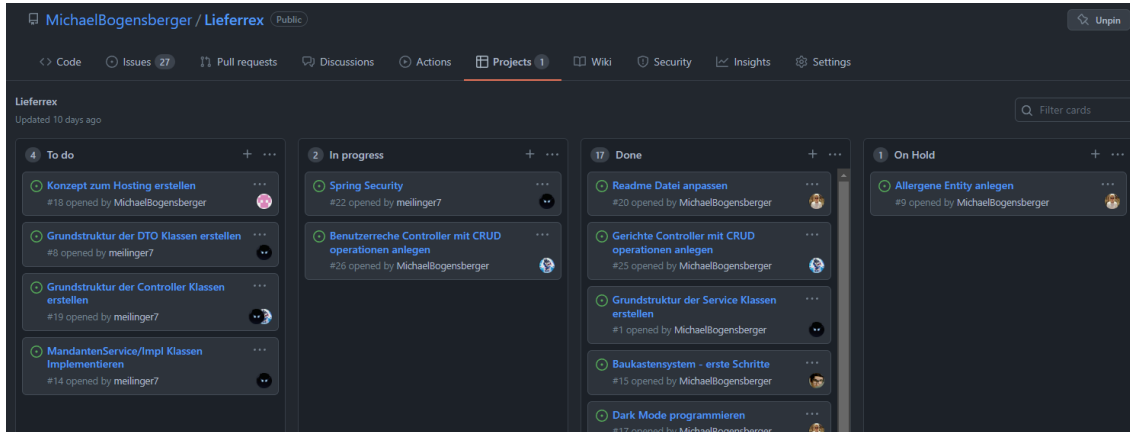


Abbildung 2 GitHub Kanban

6.1 ZenHub

ZenHub ermöglicht eine agile Entwicklung in GitHub. Es ist ein besserer Weg, um deine GitHub-Issues, Multi-Repo-Boards, Epics und Berichte zu verwalten - ohne GitHub verlassen zu müssen.

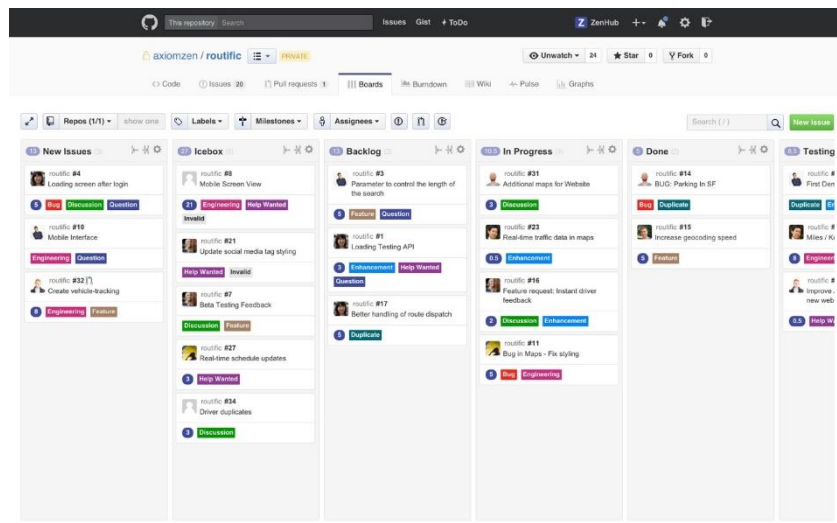


Abbildung 3 ZenHub Extension

6.2 Abwicklung in GitHub

- 1) Richten Sie GitLab oder GitHub so ein, dass sie die Plattformen möglichst gut bei der Abwicklung der SCRUM-Prozesse unterstützen können.
- 2) Pflegen Sie die User-Stories aus der vorhergehenden Aufgabe ein und spielen Sie den gesamten Scrum-Prozess mit GitLab durch.

Dokumentieren Sie den kompletten Vorgang anhand eines Textdokuments incl. Screenshots und Beschreibungen. **Link zum GitHub Repo bzw. zum Kanban:** [Link](#)

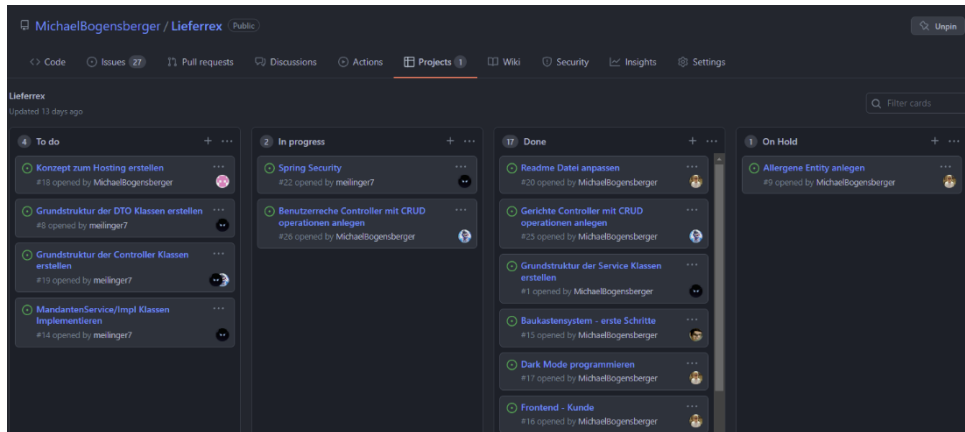


Abbildung 4 GitHub Kanban

7 User-Stories

Denken Sie sich ein fiktives Softwareentwicklungsprojekt aus und definieren Sie mindestens 5 UserStories für das Projekt. Verwenden sie dazu User-Story-Cards:

Priority <div style="border: 1px solid black; padding: 10px; font-size: 2em; margin: 10px auto; width: 40px;">2</div>	Kunde Dark-Mode Story <div style="border: 1px solid black; padding: 10px; margin: 10px auto; width: 80%;"> Als Kunde möchte ich über die Option eines Dark-Mode verfügen, um es meiner Präferenz anzupassen. </div>	Story Points <div style="border: 1px solid black; padding: 10px; font-size: 2em; margin: 10px auto; width: 40px;">1</div>
Risk <div style="border: 1px solid black; padding: 10px; font-size: 2em; margin: 10px auto; width: 40px;">1</div>		Post-SP <div style="border: 1px solid black; padding: 10px; font-size: 2em; margin: 10px auto; width: 40px;">1,5</div>

Kunde | Dark-Mode

Die Story kann durch End-zu-End Tests gut getestet werden. Dazu wird der Dark-Mode eingeschaltet und danach werden die CSS Properties kontrolliert.

Die restlichen User Stories sind in der in GitHub gespeicherten Excel Dateien zu finden! Link: [GitHub Repo](#). Des Weiteren sind zusätzliche User Stories in folgendem [Repo](#) unter „documentation“ in der PDF zu finden.

8 Agiles PM, Versionskontrolle und CI

Recherchieren und überlegen Sie, wie leichtgewichtige Versionskontrollmechanismen wie Trunkbased Development oder GitHub Flow zusammen mit Continuous Integration Features der gängigen Codeversionierungsplattformen (GitHub Actions, GitLab CI/CD) und automatisiertem Testen möglichst produktiv und effizient in Verbindung mit SCRUM eingesetzt werden können.

8.1 Trunkbased Development

Trunk-basierte Entwicklung ist eine Praktik der Versionskontrolle, bei der Entwickler kleine, häufige Änderungen in einem Kern- oder Haupt-Branch ("Trunk") zusammenführen. Dadurch werden die Zusammenführungs- und Integrationsphasen optimiert. Diese Vorgehensweise trägt zum Erreichen von CI/CD bei und optimiert die Softwareauslieferung und die organisatorische Effizienz.

8.2 CI/CD & Pipelines

Grundsätzlich unterstützen CI/CD – also Continuous Integration, Continuous Delivery und Continuous Deployment – Unternehmen bei der Umsetzung ihrer DevOps-Bemühungen. Denn die Grundlage von DevOps, also einer zunehmenden Verzahnung der Entwicklung (Development = Dev) und dem IT-Betrieb (Operations = Ops), ist ein möglichst hoher Automatisierungsgrad.

Eine CI/CD-Pipeline umfasst mehrere Schritte, die zur Bereitstellung einer neuen Softwareversion ausgeführt werden müssen. **CI/CD ist also das zusammenführen von Branches zu einem Main Branch.**

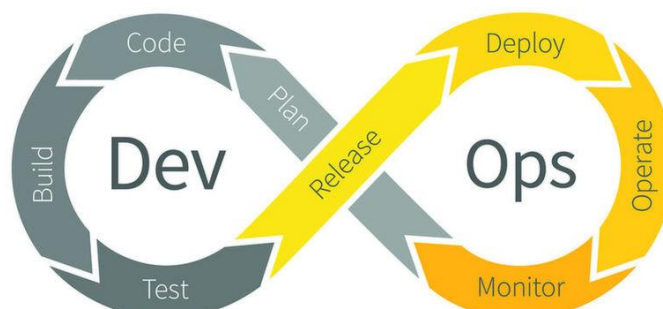


Abbildung 5 DevOps Grafik

8.3 Gitflow vs. Trunk-basierte Entwicklung

Gitflow ist ein alternatives Git-Branching-Modell mit langlebigen Feature-Branched und mehreren primären Branches. Bei Gitflow wird im Vergleich zur Trunk-basierten Entwicklung mit einer höheren Anzahl an Branches gearbeitet, die zudem langlebiger sind.

8.4 Vorteile der Trunk-basierten Entwicklung

Trunk-basierte Entwicklung ist ein für Continuous Integration erforderliches Verfahren. Wenn Build- und Testprozesse automatisiert sind, Entwickler jedoch an isolierten, langwierigen Feature-Branched arbeiten, die selten in einen gemeinsamen Branch integriert werden, wird Continuous Integration ihr Potenzial nicht entfalten können.

8.5 GitHub Actions

Mit GitHub Actions ist es möglich, beliebige Workflows in einem GitHub Repository zu definieren und dort auch automatisch ausführen zu lassen. Das können Builds und Tests zur kontinuierlichen Integration sein, aber auch völlig unabhängige Prozesse, zum Beispiel zur regelmäßigen Prüfung bestimmter Anforderungen.

9 Agiles PM in GitHub

Spielen Sie einen kompletten Prozess im Sinne von 5) praktisch durch.

- Setzen Sie die bekannten Versionierungsfunktionen ein.
- Setzen Sie die in den vorhergehenden Aufgaben ermittelten SCRUM-Funktionen ein.
- Wenden Sie eine leichtgewichtige Branching-Strategie an
- Wenden sie automatische Tests und CI/CD-Pipelines an.

Dokumentieren Sie den durchgespielten Prozess in einem Textdokument mit Screenshots.

Das Repo ist unter folgendem Link zu finden: [GitHub Repo](#)

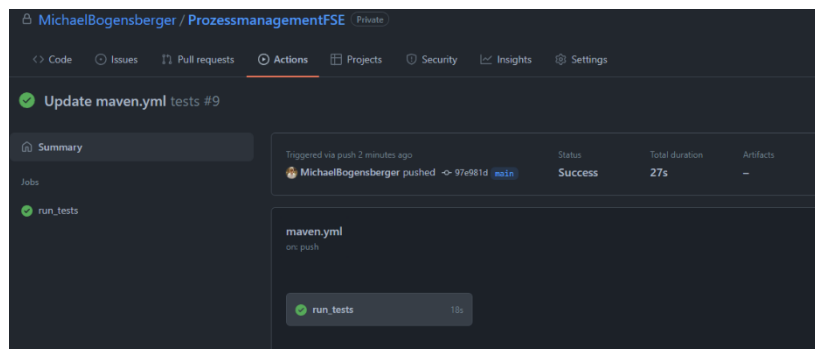


Abbildung 6 GitHub Actions

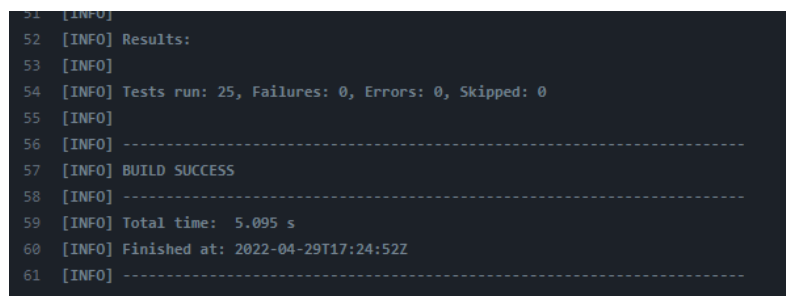
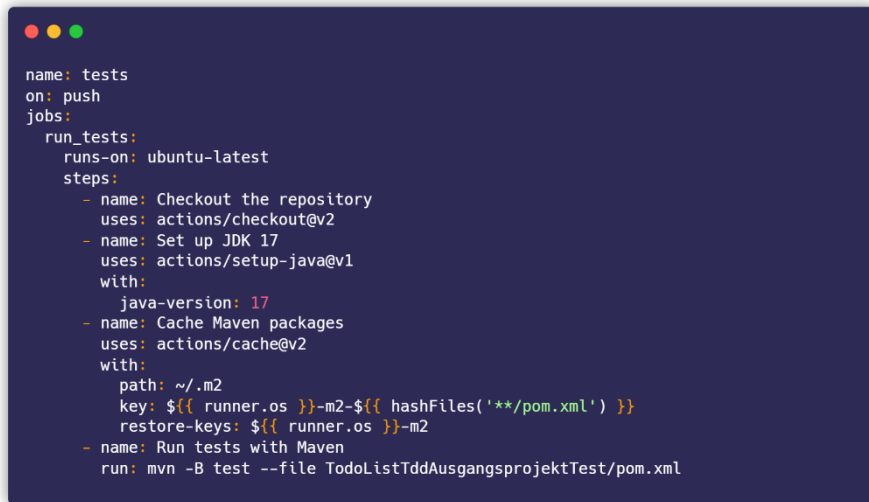


Abbildung 7 GitHub Actions Result

In folgendem Bild ist die GitHub Actions Konfigurationsdatei zu sehen. Hier wird ein Build erstellt, die Maven Packages gecached und schließlich die Tests gestartet.

A screenshot of a code editor showing a GitHub Actions workflow file. The file is named 'tests' and is triggered on a 'push' event. It defines a job 'run_tests' that runs on 'ubuntu-latest'. The job consists of three steps: 1. 'Checkout the repository' using 'actions/checkout@v2'. 2. 'Set up JDK 17' using 'actions/setup-java@v1' with 'java-version: 17'. 3. 'Cache Maven packages' using 'actions/cache@v2' with a path of '~/.m2' and a key that includes the runner OS and a hash of all pom.xml files. The final step is 'Run tests with Maven' using 'mvn -B test --file TodoListTddAusgangsprojektTest/pom.xml'.

```
name: tests
on: push
jobs:
  run_tests:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout the repository
        uses: actions/checkout@v2
      - name: Set up JDK 17
        uses: actions/setup-java@v1
        with:
          java-version: 17
      - name: Cache Maven packages
        uses: actions/cache@v2
        with:
          path: ~/.m2
          key: ${ runner.os }-m2-${ hashFiles('**/pom.xml') }
          restore-keys: ${ runner.os }-m2
      - name: Run tests with Maven
        run: mvn -B test --file TodoListTddAusgangsprojektTest/pom.xml
```

Abbildung 8 GitHub Actions YAML

10 Abbildungsverzeichnis

Abbildung 1 Burndown-Chart.....	5
Abbildung 2 GitHub Kanban	6
Abbildung 3 ZenHub Extention.....	6
Abbildung 4 GitHub Kanban	7
Abbildung 5 DevOps Grafik	8
Abbildung 6 GitHub Actions	9
Abbildung 7 GitHub Actions Result.....	9
Abbildung 8 GitHub Actions YAML	10