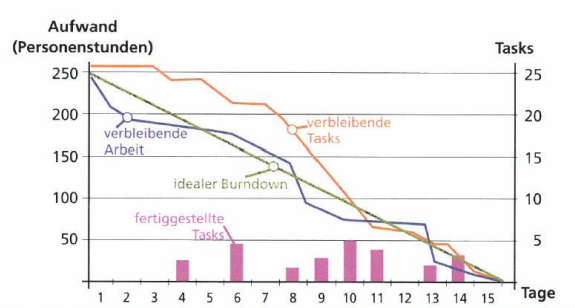
ÜbunGszettel Prozessmanagement

# aUFGABE 1: SCRUM-GRUNDLAGEN

Arbeiten Sie sich in die wesentlichen Aspekte von SCRUM ein:

1. Ablauf: Wie laufen SCRUM-Projekte ab?
2. Ereignisse: Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective, Backlog Refinement
   1. **Sprint Planning**: legt Arbeitspensum des Scrum-Teams für den kommenden Sprint fest
   2. **Daily Scrum**: dauert 15 Minuten. Jedes Teammitglied beantwortet kurz drei Fragen:
      1. Was habe ich seit dem letzten Daily-Scrum getan?
      2. Was hat mich dabei behindert?
      3. Was werde ich bis zum nächsten Daily-Scrum tun?
   3. **Sprint Review**: Am Ende des Sprints präsentiert das Team dem Product-Owner die neue Funktionalität
   4. **Sprint Retrospective**: Was haben wir gelernt? Was lässt sich verbessern?
   5. **Backlog Refinement**: Definition von Anforderungen. Definition-of-Done. Aufwandsschätzung. Priorisierung. Ergänzung und Streichung von Einträgen.
3. Rollen: Scrum-Master, Product-Owner, Developer
   1. **Scrum-Master**: Verantwortlich für Prozess und korrekte Implementierung. Vermittelt zwischen Team und Kunde.
   2. **Product Owner**:
      1. Vertretung der Interessen des Kunden
      2. Zusammenarbeit mit dem Entwicklungsteam und dem Scrum Master
      3. Verwaltung des Product Backlogs
      4. Präsentation der Fortschritte
      5. Entwicklung einer Vision des Endprodukts
   3. **Developer**:
      1. besteht aus Fachleuten: Software-Designer, Architekten, Business-Analysten, Dokumentationsexperten und Software-Tester
      2. arbeiten an einem Produkt-Inkrement
      3. jedes Mitglied liefert ein funktionales Element oder einen Teil des Gesamtproduktes
      4. Geschäftsanforderungen verstehen
      5. Entwicklung des Produkts
4. Artefakte: User Stories, Product Backlog, Sprint Backlog, Product Increment, Burndown-Chart
   1. **User Stories**: informelle, allgemeine Erklärung eines Software-Features, die aus der Sicht des Endbenutzers verfasst wurden
   2. **Product Backlog**: enthält alle bekannten Anforderungen an das System
   3. **Product Increment**: Summe aller Product-Backlog-Einträge des aktuellen Sprints. Am Ende des Sprints muss das neue Inkrement in einem nutzbaren Zustand sein und der Definition-of-Done entsprechen.
   4. **Burndown-Chart**: dient der Visualisierung bereits geleisteter und noch verbleibender Arbeit



Sie sollten nach der Bearbeitung einen guten Überblick über alle oben genannten Punkte haben. Insbesondere sollten Sie das Zusammenspiel zwischen SCRUM-Prozess, Ereignissen, beteiligten Rollen sowie resultierenden Artefakten gut beherrschen.

Dokumentieren Sie die ihrer Meinung nach wesentlichen Aspekte in einem Textdokument.

# AUFGABE 2: SCRUM MIT GITLAB ODER GITHUB

Ermitteln Sie, wie man in GitLab oder GitHub SCRUM-Projektmanagement durchführt. Setzen Sie dazu die Möglichkeiten in den beiden Plattformen kreativ ein, um die in 1) ermittelten Prozesse gut abzubilden.

Sie sollten nach der Bearbeitung eine für Sie konsistente Vorgehensweise zum Einsatz von GitLab oder GitHub zur Unterstützung von SCRUM-Projekten gefunden haben.

Dokumentiren Sie die gefundene Vorgehensweise. Stellen Sie dazu insb. gegenüber, welche Funktion in GitLab / GitHub wie und zu welchem Zweck eingesetzt werden können, um den SCRUM-Prozess zu unterstützen.

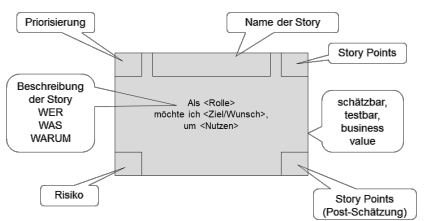
# AUFGABE 3: USER-STORIES

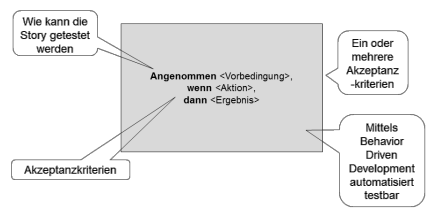
1. Bilden Sie eine Gruppe von 4 Studenten. Pro Student ergeben sich folgende Rollen:
   1. Product Owner
   2. Scrum Master
   3. Entwickler 1
   4. Entwickler 2 (falls nur 3 Gruppenmitglieder, genügt nur 1 Entwickler
2. Denken Sie sich ein fiktives Softwareentwicklungsprojekt aus und definieren Sie mindestens 5 UserStories für das Projekt. Verwenden sie dazu User-Story-Cards:
   1. Beurteilen Sie die Qualität ihrer User-Stories nach den INVEST-Kriterien.
      1. Independent: Unabhängig von anderen User-Stories
      2. Negotiable: Kurzbeschreibung ohne zu viele Details
      3. Valuable: Muss Wert für Kunden haben
      4. Estimable: Aufwand sollte abschätzbar sein
      5. Small: Jede Story sollte in 2-3 Tagen umsetzbar sein
      6. Testable: Muss testbar sein, um feststellen zu können, ob sie abgeschlossen ist.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* 1. Pro User-Story-Card definieren Sie abschließend Tasks (Aufgaben für Entwickler und Tester).





Als Dokumentation für dieses Aufgabe gelten die fertigen User-Story-Karten.

# AUFGABE 4: ABWICKLUNG MIT GITLAB ODER 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.

# AUFGABE 5: 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.

Beschreiben Sie ihre Ergebnisse in einem Textdokument.

## Trunkbased Development

Ist eine Praktik der Versionskontrolle, bei der Entwickler kleine, häufige Änderungen in einem Kern- oder Haupt-Branch („Trunk“) zusammenführen. Wird häufig von DevOps-Teams verwendet und ist Teil des DevOps-Lebenszyklus, da sie die Zusammenführungs- und Integrationsphasen optimiert. Entwickler können im Vergleich zu anderen langlebigen Branching-Strategien für Funktionen kurzlebige Branches mit wenigen kleinen Commits erstellen. Bei zunehmender Komplexität der Codebasis und wachsenden Teams trägt die Trunk-basierte Entwicklung dazu bei, einen kontinuierlichen Flow von Produktions-Releases sicherzustellen.

### Gitflow vs. Trunk-basierte Entwicklung

Gitflow ist ein alternatives Git-Branching-Modell mit langlebigen Feature-Branches und mehreren primären Branches. Höhere Anzahl an Branches als bei Trunk-basierter Entwicklung. Entwickler erstellen einen Feature-Branch und führen den Merge mit dem Haupt-Trunk-Branch erst durch, wenn das Feature vollständig ist.

### Vorteile der Trunk-basierten Entwicklung

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

verringert Reibungspunkte bei der Code-Integration. Wenn Aufgaben abgeschlossen werden, müssen sie den neuen Code in den Haupt-Branch mergen, aber erst wenn Builds erfolgreich waren.

### Fazit

Die Trunk-basierte Entwicklung ist derzeit der Standard für leistungsstarke Entwicklerteams, da sie mithilfe einer vereinfachten Git-Branching-Strategie einen Rhythmus für Software-Releases vorgibt und aufrechterhält. Darüber hinaus bietet die Trunk-basierte Entwicklung den Entwicklerteams mehr Flexibilität und Kontrolle darüber, wie sie Software an den Endbenutzer liefern.

## GitHub Flow

Der Fokus liegt auf Branches. Das ermöglicht den Teams frei zu experimentieren und regelmäßig Deployments zu machen.

Wie GitHub Flow funktioniert:

* Neuen Branch erstellen
* Änderungen durchführen und Commits adden
* Pull Request öffnen
* Andere schauen drüber
* Deployen
* Mergen

### Master Branch

Der Master-Branch ist stabil und er ist immer funktionsfähig beim Deployment. Alles das ist den Master-Branch gepusht wird, muss funktionsfähig und getestet sein.