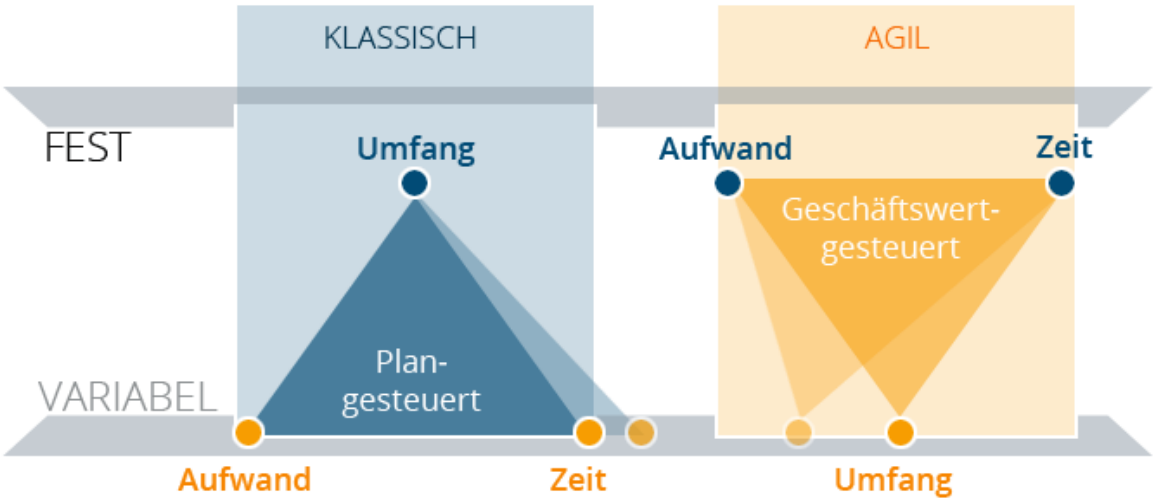
Agiles Projektmanagement

Agiles Projektmanagement beschreibt Verfahrensmethoden zur Steuerung und Planung von Projekten, wobei sich auf **Flexibilität**, schnelle **Anpassungsfähigkeit** und die Fähigkeit, schnell auf Veränderungen reagieren zu können fokussiert wird. Im agilen Projektmanagement wird die **Zusammenarbeit** zwischen den **Projektteams** und den **interessierten Parteien** (z.B. Kunden, Benutzer, Sponsoren) betont, um die **Projektziele schnell** und **effektiv** zu **erreichen**.



Stellt man das Klassische Projektmanagement dem agilen gegenüber, unterscheiden sich diese schon in ihrem Ziel. Im klassischen PJM liegt der Fokus in der Erreichung des vordefinierten Zieles, wobei hierbei Aufwand und Zeit angepasst wird, um diesen zu erreichen.

Beim agilen PJM entgegen sind Zeiträume und Aufwand (Arbeitsleistung) fix vordefiniert und der Produktumfang gilt als variabel. Ziel ist es den höchsten Geschäftswert / den größten Fortschritt mit den vorhanden Ressourcen zu erzielen.

Agiles PJM hinterfragt dir Rollen, Prozesse und Projektpläne des klassischen PJM und legt viel mehr Wert auf eine intensive Einbindung des Kunden/Auftraggebers während des gesamten Projektes. Das Liefern von regelmäßigen Ergebnissen ist hierbei ein essenzieller Bestandteil.

Ebenso heißt es Änderungen willkommen, weil sich nur so die besten Resultate liefern lassen. Vor allem Anforderungen, die diese ständige Entwicklung und Veränderung widerspiegeln, spielen hier eine wesentliche Rolle. Die Werte des agilen PJMs sind im **Agilen Manifest** niedergeschrieben:

- Menschen und Interaktionen sind wichtiger als Prozesse und Werkzeuge
  Ständige Kommunikation miteinander ist essentiell, Prozesse dürfen diese nicht einschränken

- Funktionierende Software ist wichtiger als eine umfassende Dokumentation

- Zusammenarbeit mit dem Kunden ist wichtiger als die Vertragsverhandlung
  Besser gemeinsam schon frühzeitig zusammenarbeiten, als lange mühsame Verträge auszuarbeiten

- Reagieren auf Veränderungen ist wichtiger als das Festhalten an einem Plan
  Flexibilität und Offenheit für neue Ideen und Anforderungen ist essenziell

Die agilen Prinzipien:

Die agilen Prinzipien beschreiben einen groben Leitfaden des agilen Projektmanagements, hier mit Hinblick auf die Software-Entwicklung:

- Höchste Priorität ist die Zufriedenstellung des Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software

- Veränderungen werden selbst spät in der Entwicklung zum Wettbewerbsvorteil des Kunden genutzt

- Funktionierende Software wird in regelmäßigen, bevorzugt kurzen Zeitspannen geliefert (wenige Wochen oder Monate)

- Fachexperten und Entwickler kommunizieren täglich während des Projektes
- Projektmitarbeiter erhalten ein angemessenes Arbeitsumfeld und genügend Unterstützung, um motiviert ihre Aufgaben zu erfüllen

- Informationen werden nach Möglichkeit im Gespräch von Angesicht zu Angesicht übermittelt

- Als wichtigstes Fortschrittsmaß gilt die Funktionsfähigkeit der Software

- Arbeitstempo wird gleichmäßig von Auftraggebern, Entwicklern und Anwendern für eine nachhaltige Entwicklung eingehalten

- Ständiger Fokus auf technische Exzellenz und gutes Design erhöhen die Agilität

- Einfachheit ist essenziell

- Teams organisieren sich selbst, weil dadurch die besten Systemarchitekturen, Anforderungen und Designs entstehen

- Teams reflektieren in regelmäßigen Abständen das eigene Verhalten, um effizienter zu werden

Klassisches und agiles Projektmanagement im Vergleich

Klassisches Projektmanagement

- ✓ UMFANG IST FEST, ZEIT UND AUFWAND SIND VARIABEL
- ✓ LINEARER PROZESS (WASSERFALL-MODELL): ENTWICKLUNG VON PHASE ZU PHASE
- ✓ PROZESS IST FEST
- ✓ EINFLUSS VON STAKEHOLDERN SINKT IM VERLAUF DES PROJEKTS
- ✓ ANFORDERUNGEN WERDEN NUR AM ANFANG ERFASST (Z. B. IN EINEM LASTENHEFT)
- ✓ ERGEBNISSE WERDEN NUR AM ENDE DES PROJEKTS GELIEFERT UND BEWERTET
- ✓ PROJEKTMANAGER MANAGT UND VERANTWORTET DAS GESAMTE PROJEKT
- ✓ KOMMUNIKATION IN LANGEN MEETINGS UND DURCH DOKUMENTE

Agiles Projektmanagement

- ✓ ZEIT UND AUFWAND SIND FEST, UMFANG IST VARIABEL
- ✓ ITERATIVER PROZESS: DURCHLAUF ALLER PHASEN IN EINER ITERATION
- ✓ PROZESS WIRD FORTLAUFEND VERBESSERT
- ✓ EINFLUSS DER STAKEHOLDER IST KONSTANT IM PROJEKT
- ✓ ANFORDERUNGEN WERDEN KONTINUIERLICH ERFASST (Z. B. DURCH BACKLOGS)
- ✓ ERGEBNISSE WERDEN IM PROJEKTVERLAUF REGELMÄSSIG GELIEFERT UND BEWERTET
- ✓ TEAM MANAGT SICH SELBST UND ÜBERNIMMT ZUSAMMEN DIE VERANTWORTUNG
- ✓ KOMMUNIKATION IM KURZEN, TÄGLICHEN MEETING UND WENIG DOKUMENTATION

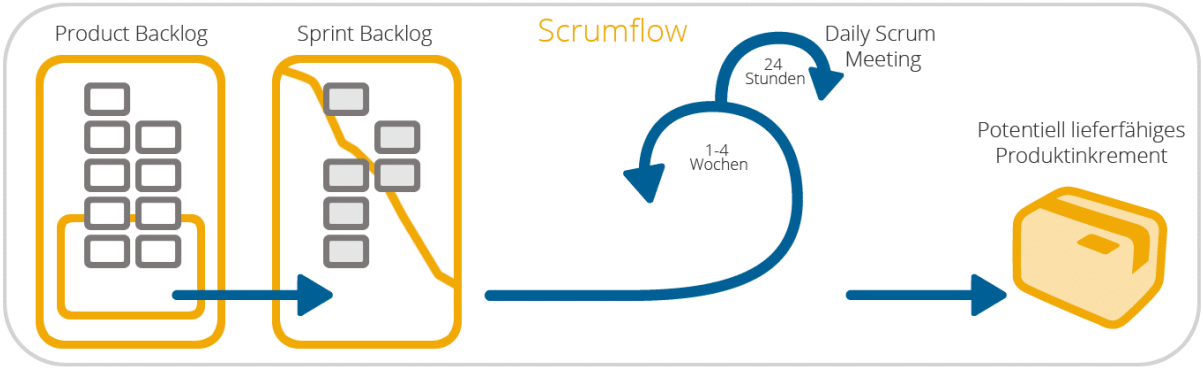
Quellen:

https://www.microtool.de/wissen-online/was-ist-agiles-projektmanagement/

Methoden des agilen Projektmanagements:

1. SCRUM

SCRUM ist wohl eine der bekanntesten Prozesse im agilen Projektmanagement. Der Aufbau basiert hierbei auf dem Agilen Manifest und den Agilen Prinzipien, aus welchen sich in SCRUM folgendes Modell ableiten lässt:



- Im **Product Backlog** werden alle Anforderungen gesammelt und gespeichert. Sie werden vom Product Owner erfasst, formuliert und aktuell gehalten. Sie werden hierin priorisiert, wobei es auch möglich ist Anforderungen wieder zu streichen. Das Entwicklungsteam schätz den Umsetzungsaufwand der Anforderungen.

- **Sprint Backlog** sind alle Anforderungen der aktuellen Entwicklungsphase (**Sprints**). Während des Sprints (fixe Dauer von üblicherweise 1-4 Wochen (gleiche Länge für alle Sprints!) werden die im Sprint Backlog befindlichen Anforderungen abgearbeitet.

- Bei SCRUM bezeichnen wir diese Anforderungen bzw. Arbeitspakete als sogenannte **User Stories**, welche eine in Alltagssprache formulierte Software-Anforderung darstellen. Sie ist bewusst kurzgehalten und umfasst in der Regel nicht mehr als zwei Sätze. Sie werden zusammen mit Akzeptanztests zur Spezifikation von Anforderungen eingesetzt. User Stories sind meistens nochmals in einzelne Tasks (auch Tickets (Kanban)) unterteilt.

- Das Entwicklungsteam trifft sich täglich für eine kurze Besprechung in einem Daily Srum zur Absprache der geleisteten Arbeit zur Setzung der Tagesziele und etwaige Hindernisse.

- **Rollen:** *Das Projektteam organisiert sich hierbei selbst folgende Rollen werden eingenommen:*

  - **Entwicklungsteam**
    - Zwischen 3 bis 9 Personen
    - Organisiert sich selbst
    - Übernimmt Umsetzung und Schätzung der Anforderungen

- **Product Owner**
  - Verwaltet Anforderungen des Projekts
  - Kommunikation mit Kunden/Auftraggebern

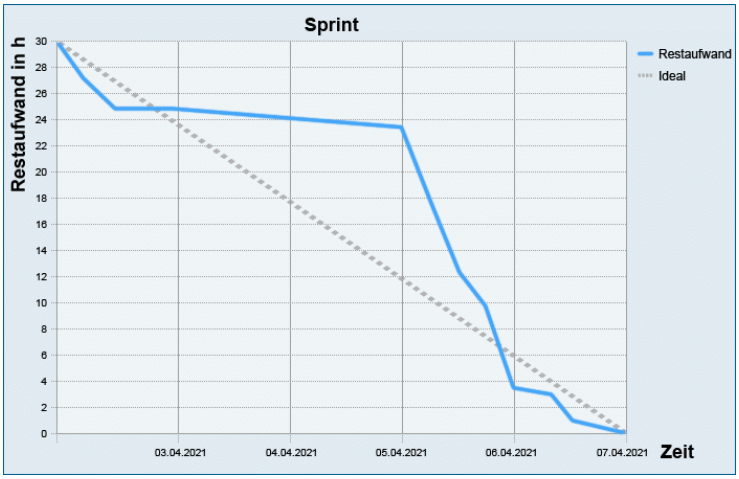
- **SCRUM Master**
  - Korrekte Anwendung der SCRUM-Prozesse
  - Überwacht Arbeiten und Prozesse laut SCRUM
  - Schützt vor unberechtigten Eingriffen in die Entwicklung

- **Stakeholder / Auftraggeber**
  - Keine definierte SCRUM-Rolle, trotz maßgebliche Wirken am Projekt
  - Vorgabe / Abnahme der Anforderungen

- Die Entwicklung der Software geschieht in der Struktur er Sprints und Releases. Ein Sprint entspricht einem Entwicklungszyklus von einer bis vier Wochen. Die genaue Länge legt das Team fest, ein Sprint sollte jedoch immer gleich lang sein. Am Ende eines Sprints liefern Sie ein funktionsfähiges Produktinkrement. In einem Release wird eine Produktversion freigegeben.

- Wenn das erste **Product Backlog** im Scrum-Projekt steht, setzen sich das Team und der Product Owner in einem **Sprint Planning** zusammen. Hier entscheiden sie, welches **Ziel** für den **nächsten Sprint** geplant ist und welche Anforderungen sie dafür umsetzen müssen. Das Entwicklungsteam **schätzt** dafür den **Umsetzungsaufwand** jeder Anforderung bzw. User Story. Auf Basis dieser Aufwandsschätzung und den Prioritäten, die der Product Owner vergeben hat, **wählt** das **Entwicklungsteam** eine **Anzahl** von **Anforderungen** für den nächsten Sprint – ein sogenanntes **Sprint Backlog** entsteht. Die **Entwickler** müssen **alle User Storys**, die dieses **Backlog** enthält, **selbstorganisiert in diesem Sprint umsetzen**. Den **Fortschritt** halten Sie fest, indem sie jeden Tag den **Restaufwand notieren**. Auf diese Weise sinkt der **Restaufwand** einer Anforderung am Ende des Sprints **auf 0**: Die User Story wurde umgesetzt. Solch eine Entwicklung lässt sich in einem sogenannten **Burndown Chart** schön darstellen.

- **BurnDown Chart**

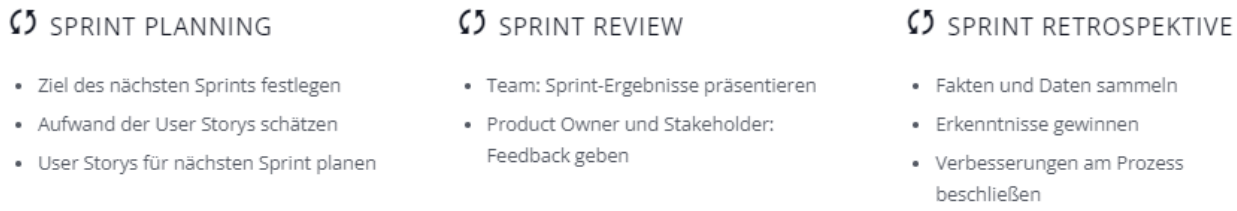


Das Burndown Chart stellt den Restaufwand des aktuellen Sprints und somit auch den Fortschritt da.

Es lässt sich hieraus die Auslastung des Teams, wie auch das Erreichen des Sprint-Ziels ablesen.

- Am **Ende** des **Sprints** trifft sich das Team in einem **Sprint Review**, das idealerweise nicht länger als eine Stunde dauern sollte. Die Entwickler **besprechen** ihre **Ergebnisse** und prüfen, ob sie das angestrebte Ziel erreicht haben. Auch die **Stakeholder beteiligen** sich an diesem Treffen und **Geben aktiv Feedback**. Der **Product Owner überarbeitet** das **Product Backlog** und zusammen mit dem Team **plant** er das **Ziel** und die Anforderungen für den **nächsten Sprint** im Scrum-Projekt.

- Als Abschluss des Sprints führt das Team unter Moderation des Scrum Masters oder eines anderen Moderators eine Sprint Retrospektive durch. Diese soll mindestens einen Verbesserungsvorschlag bezüglich des Prozesses hervorbringen.



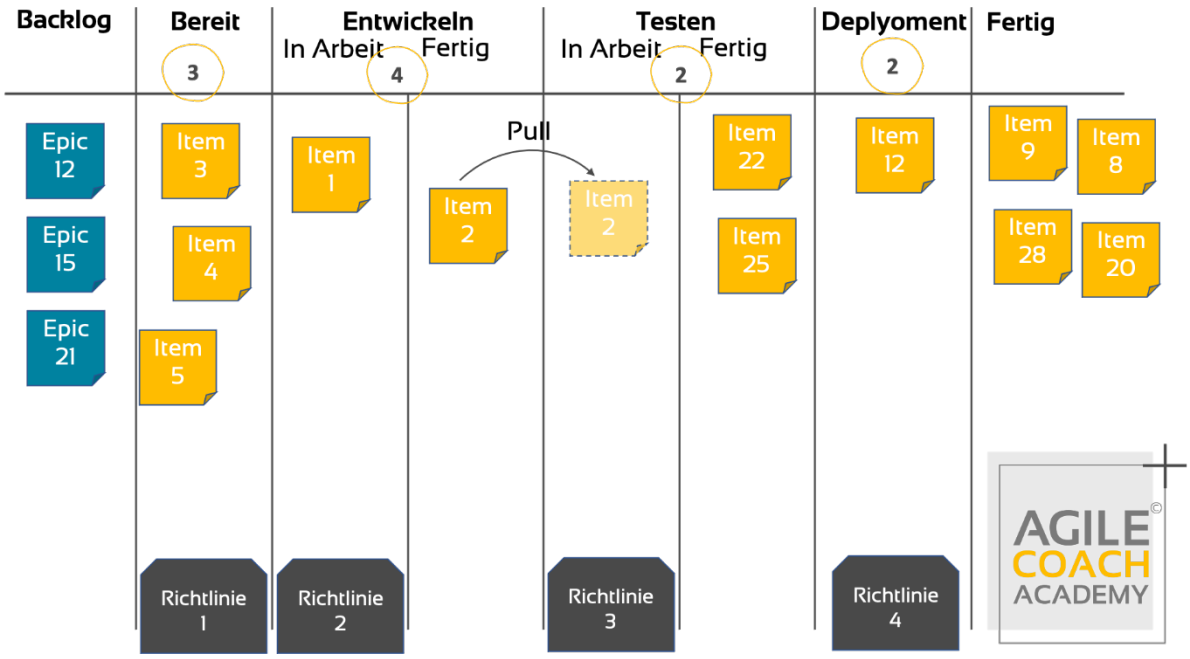
- Scrum lebt von einer lebendigen Kommunikation – nicht nur zwischen Stakeholdern und Product Owner, sondern auch innerhalb des Entwicklungsteams.
  Deswegen kommen die **Entwickler** zu Beginn jedes Arbeitstages zu einem **Daily Scrum** zusammen: In etwa **15 Minuten** berichten die Beteiligten, was sie am letzten Tag erreicht haben, was sie zum nächsten Daily Scrum erreichen möchten und welche Hindernisse ihnen dabei möglicherweise im Weg stehen.
  Wenn Probleme auftreten, helfen sich die Teammitglieder nach Bedarf gegenseitig. Auch der Product Owner und Scrum Master sind oftmals anwesend, wenn auch nur passiv daran beteiligt.

Quellen:

https://www.microtool.de/wissen-online/wie-funktioniert-scrum/

https://www.microtool.de/wissen-online/was-ist-ein-burn-down-chart/

2. Kanban



- Neben Scrum zählt Kanban zu der zweiten großen Vorgangsweise beim Agilen Projektmanagement. Die Methode stammt aus Japan, aus der Autoproduktion (Toyota), zur Optimierung des Materialverbrauches. Hieraus entwickelte sich die **Pull-Methode**, bei welcher erst Nachschub, bzw. neue Aufgaben, angefordert werden, wenn diese benötigt werden.

- Das zentrale Element von Kanban ist das Kanban-Board, auf welchem sich die einzelnen Aufgaben/Anforderungen in Tabellarischer Form, je nach ihrem Status wiederfinden.

- Die Struktur des Kanban Boards ist in ihrer Anzahl der Spalten frei wählbar, wobei mind. 3 Spalten vorhanden sein müssen (Backlog – WiP (Work in Progress) – Finished).
  Oft benötigt man in der Praxis noch zusätzliche Spalten, wie auch eine Priorisierung, welche mithilfe von **Swimmlanes** umgesetzt werden kann. Dabei handelt es sich um horizontale Linien, die den Work-in-Progress-Bereich unterteilen. Hierbei kann das Team in einem oberen Bereich (einer FastLane) alle Aufträge einfügen, die schneller als andere bearbeitet werden müssen, und weniger zeitintensive Aufträge weiter unten eintragen. So kann sich jedes Teammitglied schnell einen Überblick über die derzeitigen Prioritäten verschaffen.

- In der Software-Entwicklung wird hierbei zu Beginn der Backlog mit den Anforderungen des Kunden gefüllt, welche bei Gebrauch entnommen und bearbeitet werden. Ist ein Arbeitsschritt am Arbeitspaket/an der Anforderung/Anfrage (**Ticket**) geschehen wird dieses in den nächsten Arbeitsschritt verschoben. Dies wird so lange durchlaufen, bis ein Ticket den Zustand **Finished** erhält.

- Durch diese Methodik lässt sich der Workflow stark verbessern, da jede Anforderung strukturiert alle Passagen der Entwicklung durchläuft. Es resultiert eine **Prozessoptimierung**, dessen Leitspruch „**Stop starting – start finishing!**“ lautet.

- Man fokussiert sich mehr auf die Abarbeitung der einzelnen Pakete in kleinen Schritten hintereinander, anstatt eine Vielzahl von Paketen anzufangen und quasi parallel zu entwickeln.

- Neben der erhöhten Transparenz der Arbeit liefert Kanban gleichzeitig eine **Begrenzung der Aufträge**. Durch Richtlinien/Regeln pro Spalte wird geregelt, wie viele Pakete sich zur gleichen Zeit in einer Spalte befinden dürfen. Wird diese unterschritten werden neue Pakete aus dem Backlog gezogen. Sollte sie überschritten werden, müssen erstmals die anderen Pakete fertiggestellt werden. Weiters können Regeln aufgestellt werden, welche Eigenschaften ein Arbeitspaket besitzen, muss um in die nächste Stufe aufsteigen zu dürfen, um dies im Team zu vereinheitlichen.

- Praktiken von Kanban:

1. **Visualisierung** Das Kanban-Board ist eine Visualisierung der Arbeitsabläufe. Die Gestaltung selbst bleibt aber relativ offen. Wichtig ist nur, dass Stationen klar sind und für jede Spalte das entsprechende Limit angezeigt wird.
2. **Limitierung**: Jede Spalte darf nur eine maximale Anzahl an Aufträgen enthalten. Erst wenn eine Auftragskarte weiter nach rechts wandert, darf sich das Team eine neue Karte von links nehmen. Dies führt zwangsläufig zu einem effizienteren Workflow.
3. **Management**: Während des Arbeitsprozesses kann es zu Blockaden und Engpässen kommen. In solchen Situationen ist es notwendig, den Fokus des Teams darauf zu legen, diese Störungen aus dem Weg zu schaffen. Außerdem kann die Beobachtung des Workflows dafür sorgen, Kapazitäten langfristig korrekt zu verteilen.
4. **Regulierung**: Explizite Prozessregeln sind dafür gedacht, die Arbeitsabläufe transparenter und klarer zu gestalten. Zu solchen Regeln gehört z. B. die Festlegung der Limits, aber auch eine Definition, ab wann eine Aufgabe als erledigt gilt. Prozessregeln müssen ebenfalls ein sichtbarer und veränderbarer Teil des Kanban-Boards sein.
5. **Feedback**: Rückmeldungen sind ein notwendiger Teil von Arbeitsabläufen, denn nur so lassen sich diese verbessern. Dafür sind **regelmäßige Meetings** vorgesehen, sogenannte **Kadenzen**. Anders als Scrum gibt Kanban aber kein starres Gerüst für solche Treffen.
6. **Kaizen**: Prozesse im Team sollen mit Kanban kontinuierlich verbessert werden. Die Theorie geht somit davon aus, dass man kein Optimum erreichen kann, sondern dauerhaft an Verbesserungen arbeitet.

Vorteile	Nachteile
Offenes Prinzip	Bedarf übergreifender Kompetenzen
Mehr Transparenz	Fehlende Zeitplanung kann Probleme bei Deadlines erzeugen
Gleichmäßiger Workflow	Arbeit muss sich in einzelne Schritte aufteilen lassen
Stetige Verbesserung	
Lässt sich in vielen Situationen anwenden	
Einfache Integration	

Quellen:

https://www.ionos.at/digitalguide/websites/web-entwicklung/kanban/