**Projektdokumentation**

**Carscout**

**Von Can Dündar, Antonios Vamvakos, Milan Skripalle, Philipp Heitmann und Daniel Peters**

**Inhaltverzeichnis**

1. Aufgabenstellung
   1. Gegenstand des Projektauftrags
   2. Grund der Auswahl des Projektauftrags
2. Ist-Analyse
   1. Welches Problem ist zu lösen?
   2. Wie wird der zugrunde liegende Arbeitsprozess momentan gelöst?
   3. Was ist der genaue Zweck des Projekts?
3. Soll-Konzept
   1. Wie soll die Lösung für das Problem aussehen?
   2. Wo liegen die Vor-/Nachteile unserer Lösung?
   3. Beschreibung der gewählten Software-Architektur
   4. Skizzierung der Vorgehensweise
   5. Graphische Darstellung des Konzepts (UML)
   6. Projektstrukturplan
   7. Zeitplanung (GANTT)
4. Beschreibung der Durchführung
   1. Beschreibung der einzelnen Iterationen
   2. Skizzierung der aufgetretenen Problematiken
      1. Beschreibung wie diese gelöst wurden
5. Fazit
   1. Erkenntnisgewinn
   2. Probleme

**Aufgabenstellung**

**Gegenstand des Projektauftrags**

Der Projektauftrag umfasst das Erstellen einer Plattform zur Inserierung von gebrauchten und neuwertigen Fahrzeugen im Internet. Die Plattform soll einen Markt für Käufer und Verkäufer der entsprechenden Produkte bereitstellen und die beiden Vertragspartner dabei unterstützen die jeweiligen Ziele, sprich ein passendes Angebot finden oder ein Angebot erstellen, zu erreichen.

**Grund der Auswahl des Projektauftrags**

Der erwähnte Projektauftrag wurde gewählt, da durch diesen ein zunächst simpel wirkendes Konzept umgesetzt werden kann, welches grundlegende Elemente wie beispielweise eine Registrierung, Kommunikation mit Datenbanken etc. enthält. Allerdings mit tieferem Eintauchen in die Thematik Raum für ein breites Spektrum an verschiedenen, komplexen Problemstellungen liefert und somit Herausforderungen bietet. Des Weiteren ist das Thema der Automobile und die damit verbundene Inserierung von gebrauchten oder auch neuwertigen Fahrzeugen auf einem Onlineportal, ein interessantes sowie zukunftsfähiges Konzept.

**Ist-Analyse**

**Welches Problem ist zu lösen?**

Verkäufer von Automobilen stehen heutzutage vor der Herausforderung ein komplexes Produkt mit zahlreichen Eigenschaften und Besonderheiten, sowie technischen Merkmalen und einer hohen Variantenvielfalt auch im Internet anzubieten. Hier sind keine spontanen Rückfragen oder Diskussionen Vorort möglich. Ein Kunde entscheidet innerhalb von wenigen Sekunden, ob das Angebot für ihn interessant ist oder nicht. In der Wirtschaft außerhalb des Internets hat der Verkäufer mehr Möglichkeiten den Kunden für ein Angebot zu begeistern, welches im Internet nicht so einfach möglich ist. Hierzu wird ein Produkt benötigt mit dem die Automobile einfach, dabei trotzdem vollständig und mit ausreichender Komplexität als entsprechendes Angebot abgebildet werden können und der Händler die Möglichkeit hat durch ein simples, aber gut überlegtes Design alle wichtigen Informationen präsent und nicht überladen darzustellen. Der Markt benötigt zudem ein durch alle Benutzerschichten einfach bedienbares Produkt. Aus Sicht der Käufer bieten sich zusätzlich weitere Anforderungen: Der riesige Markt an gebrauchten Automobilen muss übersichtlich und beherrschbar dargestellt werden und die gebotenen Informationen sollten schnell überblickt und verarbeitet werden können. Funktionalitäten wie das Vergleichen von verschiedenen Angeboten, sowie das Filtern der Fahrzeuge nach selbst ausgewählten Kriterien etc. müssen zudem auch intuitiv gestaltet werden und den Benutzer nicht vor große Schwierigkeiten stellen.

**Wie wird der zugrunde liegende Arbeitsprozess momentan gelöst?**

Momentan wird der Arbeitsprozess zum einen ohne eine Onlineplattform gelöst indem ein Kunde einen Händler aufsucht und sich bei diesem die stark eingeschränkte Auswahl an Angeboten ansieht. Zum anderen gibt es bereits einige große Anbieter von Onlineplattformen, welche den Markt der Gebrauchtfahrzeuge abbilden. Diese haben allerdings den großen Nachteil, dass dort die Übersichtlichkeit durch eine enorme Vielzahl an Funktionalitäten stark eingeschränkt wird und somit potenzielle Neukunden eventuell abgeschreckt oder überfordert werden. Viele dieser Funktionalitäten werden teils überhaupt nicht von den Benutzern benötigt.

**Was ist der genaue Zweck des Projekts?**

Das eigentliche Problem besteht darin, dass es wahrscheinlich Käufer oder Verkäufer gibt, die nie einen Kauf bzw. Verkauf über solche Services abwickeln, da ihnen die Komplexität dieser Plattformen schlicht vor deren Benutzung abschreckt. Deren Bedürfnis ein passendes Fahrzeug zu finden oder ein Inserat einzustellen kann somit nicht erfüllt werden. Außerdem fehlt den bisherigen Plattformen ein intuitives System zum Vergleichen verschiedener Angebote.

**Soll-Konzept**

**Wie soll die Lösung für das Problem aussehen?**

Wir bieten eine Lösung indem wir eine Plattform bereitstellen, welche durch simples Design und intuitive Bedienung dem Benutzer große Hürden nimmt. Die Angebotserstellung wird durch unsere Plattform denkbar einfach gestaltet, sodass jeder Verkäufer, egal ob Händler oder Privatperson problemlos ein Auto inserieren kann. Für die Käufer der Fahrzeuge bieten wir eine gut strukturierte Benutzeroberfläche mit simpler Suche und groben Übersichtskarten der Angebote. Die Details eines Angebots werden dem Benutzer gut sortiert auf einer entsprechenden Unterseite dargestellt. Im späteren Verlauf des Projekts folgen weitere Funktionalitäten wie das bereits angesprochene Vergleichen und Filtern von Angeboten.

**Wo liegen die Vor- / Nachteile unserer Lösung?**

Die Vorteile liegen in darin, dass durch die einfache Benutzung ein erweiterter Benutzerkreis Interesse hat unsere Plattform zu nutzen. Die Benutzerfreundlichkeit erlaubt es Kunden sich jederzeit ohne vorherige Planung Angebote anzusehen, spontane Benutzung zwischendurch wird gerade von Kunden mit begrenzter Zeit sehr geschätzt. Es wird sich auf die grundlegenden Informationen und Daten zu beschränkt.

Ein Nachteil ist, dass wir von einem absoluten Basisstand ohne jegliche Benutzerstämme oder Angebotsdaten anfangen müssen. Wir verkaufen selbst keine Autos, sondern bieten nur eine Plattform. Das bedeutet für uns, dass wir erst eine Basis an Händlern aufbauen müssen, denn erst wenn genügend Autos angeboten werden, haben Kunden einen Grund, unsere Plattform zu nutzen. Dazu es wird sehr schwierig, so stark etablierten und weit verbreiteten Anbietern wie Autoscout oder mobile.de, welche im Grunde den Markt dominieren, Kunden abzuwerben. Andererseits hat ein Händler, welcher bereits diese bestehenden Seiten verwendet keinen Nachteil, wenn er dieselben Autos auch bei uns einstellt. Dieser erweitert damit tatsächlich seinen potentiellen Kundenkreis und erschafft sich so eine größere Reichweite für seine Angebote.

**Beschreibung der gewählten Software-Architektur**

Für die Umsetzung des Projekts wurde als Technologiebasis Angular in der Version 5.2.4 in Verbindung mit TypeScript und Node.js gewählt. Dadurch wurde uns bei der Architektur bereits das Prinzip der Komponenten, verschachtelt in einem Baum vorgeschrieben. Wobei in unserem Projekt keine starke Verschachtelung notwendig war und wir uns eher an einer flachen Hierarchie orientierten.

Generell besteht eine der erwähnten Komponenten aus einem Template mit HTML Struktur, einer Komponentenklasse und optional einem Stylesheet. Das Template strukturiert die Benutzeroberfläche und ist für die Darstellung der Informationen relevant. Die Komponentenklasse enthält die Logik und Daten für die View, jedoch keine komplexe Business-Logik. Diese sollte möglichst in eigene Services ausgelagert werden. Zusätzlich dazu gibt es in unserem Projekt pro eigenständiger Komponente, eine Datei mit automatisierten Tests, um die Fehler in der Software bestmöglich zu minimieren. Der Ansatz des Test-Driven-Development (TDD) fand auch Beachtung in diesem Projekt.

Unser Projekt „carscout“ besteht aus insgesamt zwei einzelnen Projekten. Eines davon „backend“ stellt die Backend-Funktionalität dar. Dort wird mit der Datenbank kommuniziert und eine REST-Schnittstelle aufgebaut. Für das Frontend ist das Projekt „project“ zuständig. Dort werden die Komponenten, welche das Frontend abbilden, hinterlegt. Das Frontend kann sich dementsprechend über die REST-Schnittstelle von dem Backend die notwendigen Daten holen und diese für den Anwender anzeigen und auf Aktionen des selbigen reagieren.

Das Backend enthält also DAOs (Data Access Object) für die Kommunikation mit der Datenbank. Models, um die einzelnen Objektklassen abzubilden und Router, um das Routing umzusetzen.

Das Frontend besteht aus einem Routing Module, den gleichen Models, welche im Backend existieren zuzüglichen einer Enumeration von Message-Types, verschiedenen Services und diversen Komponenten. Die Services können per Dependency Injection in den Komponenten verwendet werden. Als Services implementierten wir einen Service für die REST-Schnittstelle, einen zum Darstellen von Messages für den Benutzer und einen, um die Suche nach Angeboten zu ermöglichen. Die Komponenten für den Aufbau der Anwendung werden im Folgenden dargestellt:

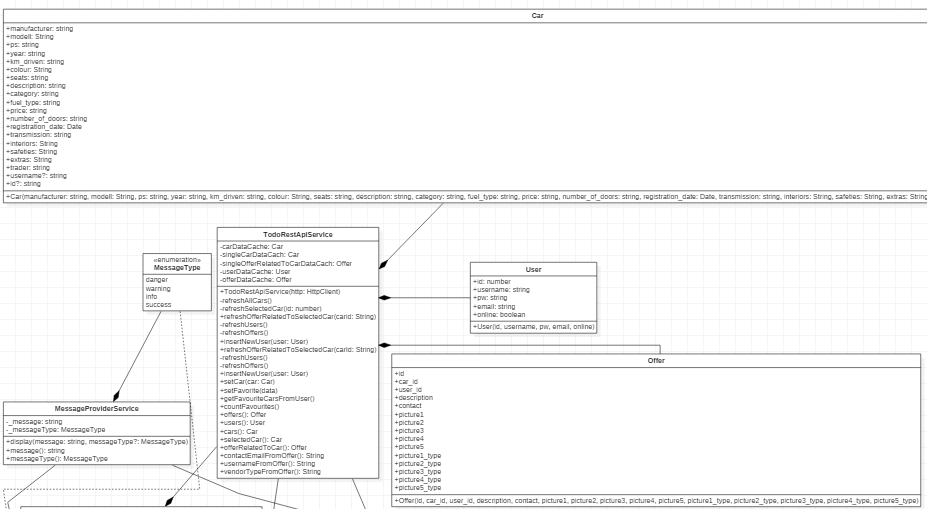
* Angebotsdetailseite (angebots-detail)
* Favoriten (favorites)
* Startseite (home)
* Rechtsinformationen (legal-information)
  + ABG (agb)
  + Impressum (impressum)
  + Datenschutz (privacy-policy)
* Login (login)
* Logout (logout)
* Benutzermitteilungen (messagebox)
* Mein Konto (my-account)
* Fehlerseite (not-found)
* Angebotserstellung (proposal)
* Registrierung (register)
* Ergebnisseite (result-page)
* Suche (search)

Unabhängig von der erwähnten Architektur achteten wir darauf, dass jede Komponente nur eine Zuständigkeit hat und diese voneinander klar getrennt sind. Durch das Prinzip der einzelnen Komponenten und Services ist eine einfache Austauschbarkeit und damit leichte Wartung und Erweiterung des Projekts gegeben.

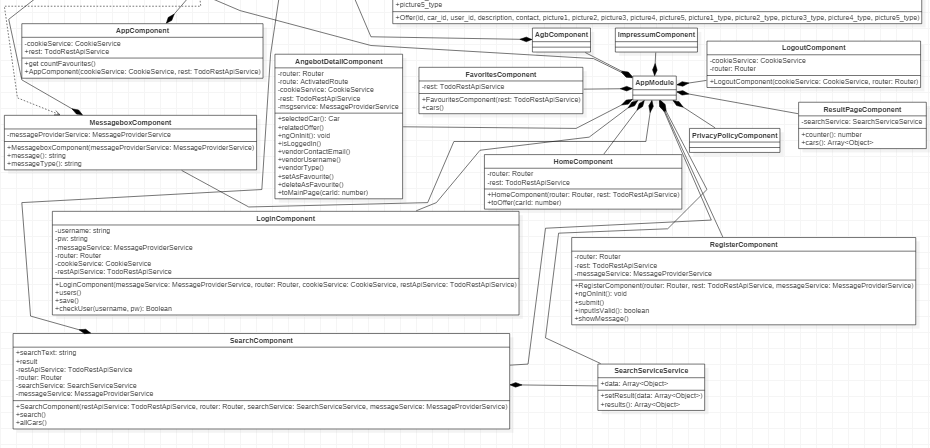
**Skizzierung der Vorgehensweise**

**Graphische Darstellung des Konzepts (UML)**

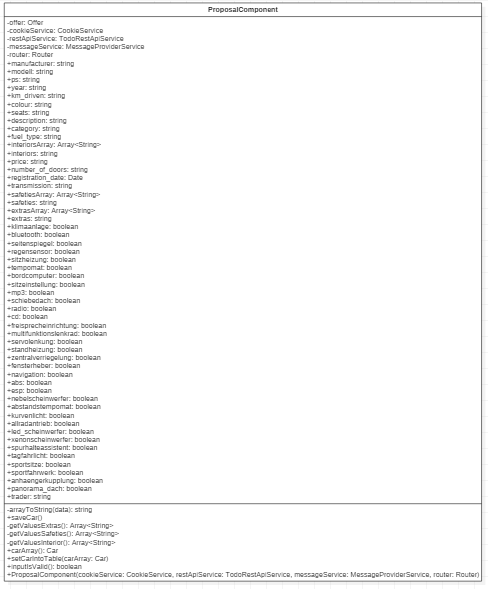
Teil 1

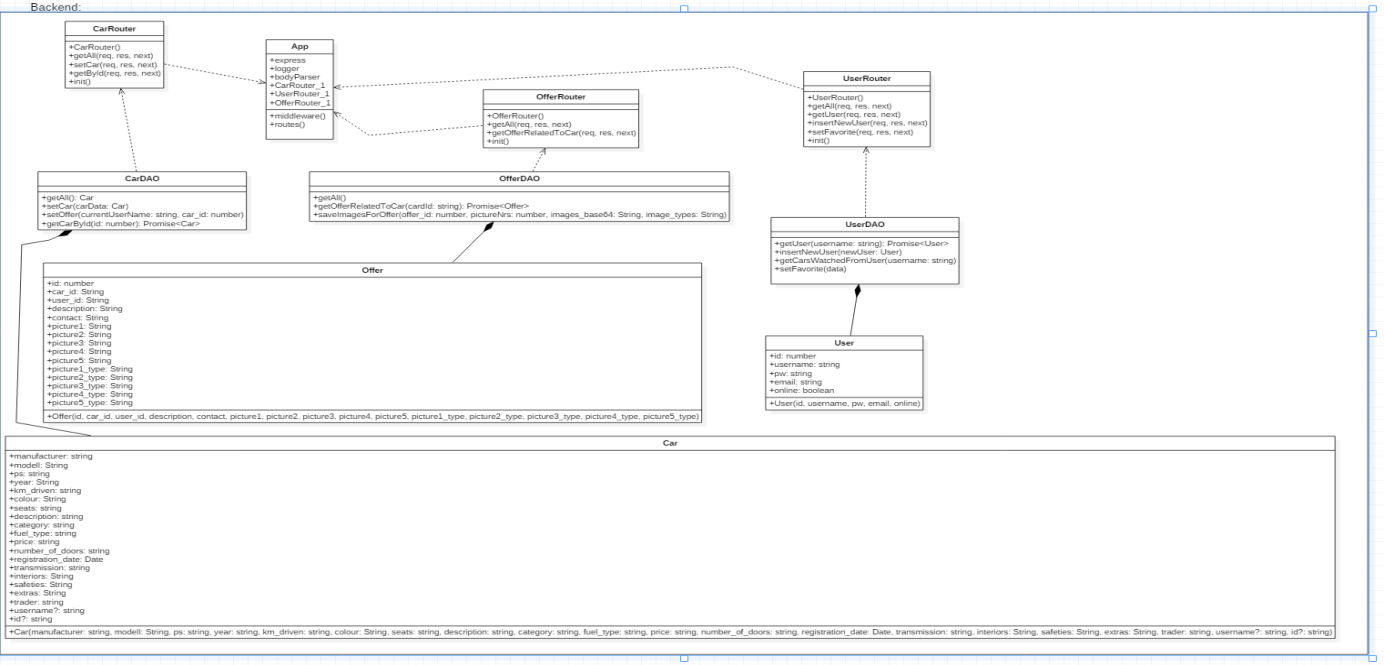


Teil 2



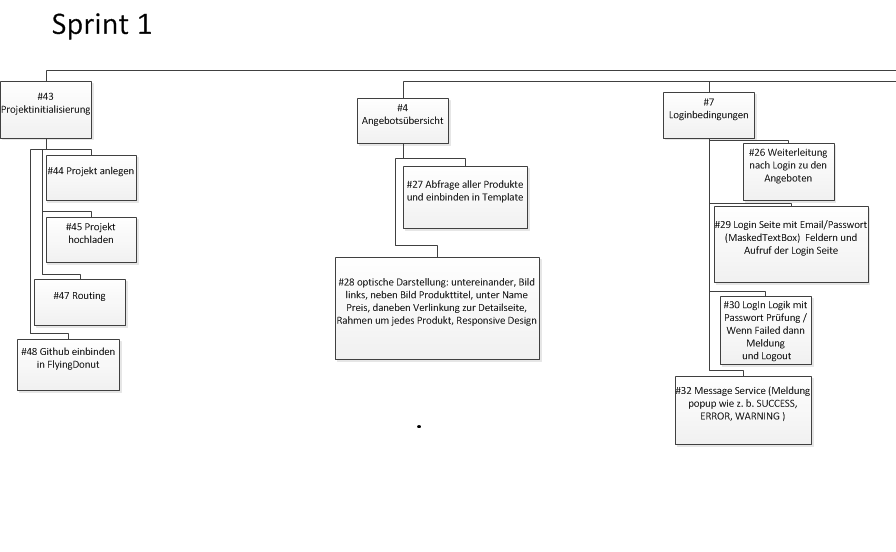
Teil 3



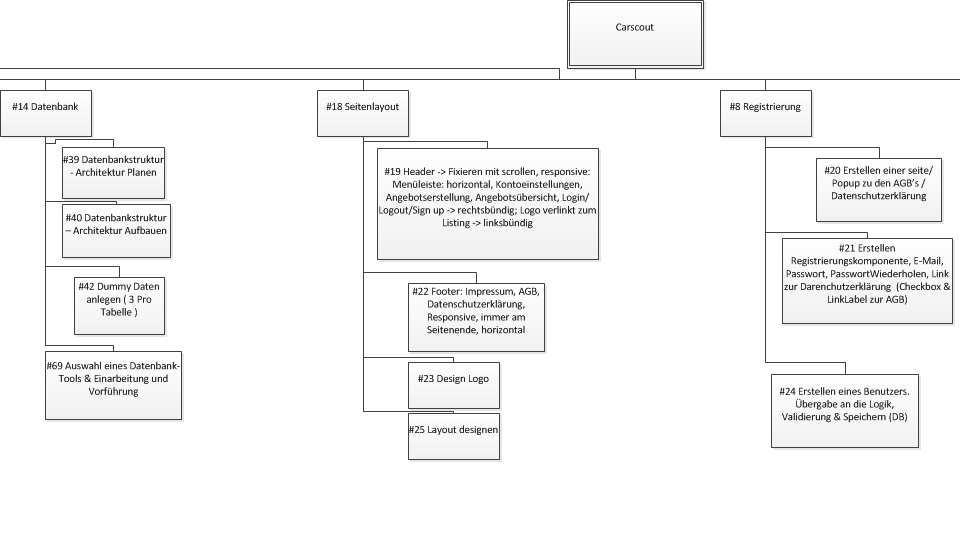
Teil 4 (Backend) 

**Projektstrukturplan**

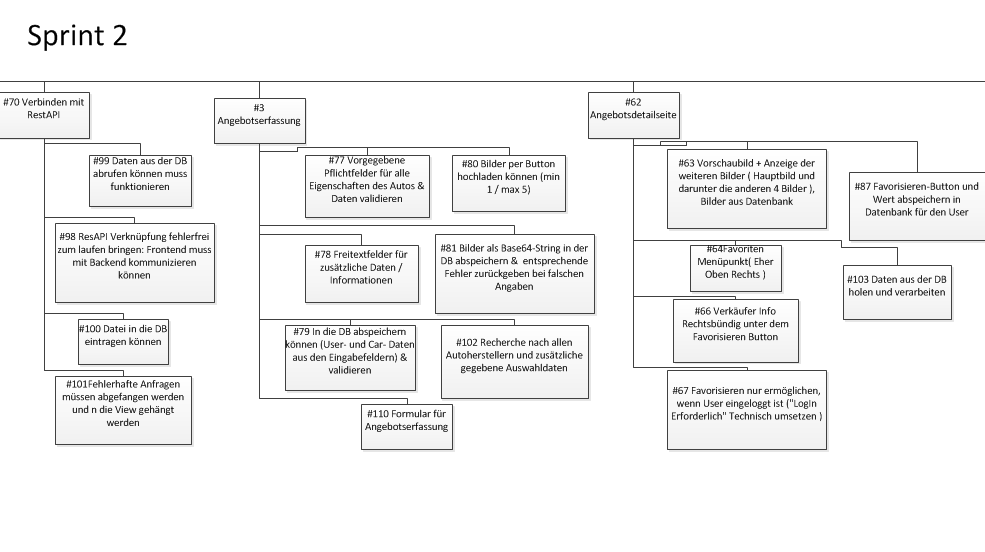
Teil 1



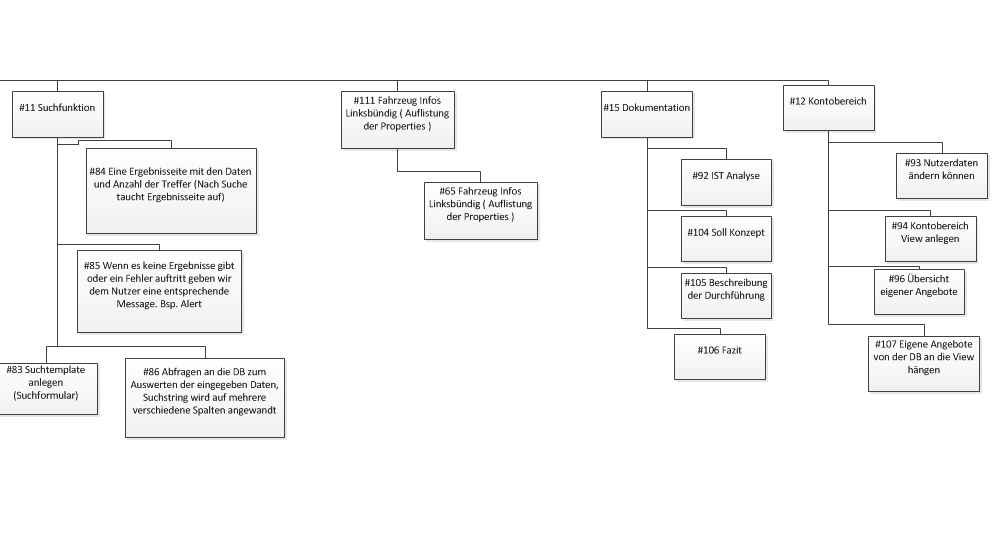
Teil 2



Teil 3



Teil 4



**Zeitplanung (GANTT)**



**Beschreibung der Durchführung**

Der für die Durchführung der Entwicklung gewählte Ansatz soll entsprechend der Prinzipien von agiler Softwareentwicklung diversen verschiedenen Anforderungen genügen. Bedient wird sich hierbei an dem Prinzip des Entwicklungsprozesses „Scrum“. Diese Anforderungen sind unter anderem notwendig, um die Softwareentwicklung in einem Team generell zu ermöglichen bzw. zu organisieren. Dadurch wird die Entwicklung für essentielle, aber nicht planbare Änderungen offengehalten, allerdings geschlossen für Überplanung der Ressourcen des Teams. Für die Planung des Projekts werden die einzelnen Funktionalitäten und andere große Aufgabenbereiche geclustert und anschließend in kleine Teilaufgaben mit detaillierter Beschreibung der Anforderungen eingeteilt. Diese Teilaufgaben werden mit deren zeitlichen Aufwänden geschätzt und den verschiedenen Sprints bzw. Iterationen zugeordnet. Diese Aufgaben können sich innerhalb der Entwicklungszyklen per „Pull“-Prinzip genommen und bearbeitet werden. Es werden keine Aufgaben von einer leitenden Person zugeteilt, sondern jeder entscheidet selber welche Aufgaben man bearbeitet. Die so entstehende Eigenorganisation des Teams ist ein wesentlicher Bestandteil der agilen Softwareentwicklung und soll dazu beitragen kurze Durchlaufzeiten der einzelnen Tickets zu erreichen. Um den Prozess abzubilden nutzt unser Team das Online-Tool „Flying Donut“ in der Version v1.93.3. Ergänzend dazu wird das Versionskontrollsystem Git eingesetzt. Dadurch wird es uns ermöglicht einzelne Zwischenstände der Software festzuhalten und organisiert an der Software Änderungen vorzunehmen. „Flying Donut“ besitzt eine Git Integration, somit ist es möglich die Softwareänderungen den Tickets zuzuordnen.

**Beschreibung der einzelnen Iterationen**

Eine Iteration des Entwicklungszyklus hat planmäßig die Dauer von einem Berufsschulblock in welchem uns durchschnittlich 12 Schulstunden oder neun Zeitstunden in der Berufsschule zur Verfügung stehen. Die gesamte Entwicklungszeit umfasst dabei zwei vollständige Sprints zur Entwicklung, sowie einen weiteren halben Sprint, welcher wahlweise als Verlängerung des zweiten Sprints betrachtet werden kann. Innerhalb der Sprints finden regelmäßig verschiedene Termine mit dem Team statt. Hierzu zählen Standups / Daily Scrum’s zur Besprechung aktueller Themen und Probleme. Sowie vor jedem Sprint ein Grooming für die kommende Iteration und nach den Sprints eine Retrospektive. Das Grooming beschäftigt sich mit dem Planen der Inhalte für den kommenden Sprint, die Retrospektive mit der kontinuierlichen Verbesserung des Entwicklungsprozesses, sowie mit allgemeinen Schwierigkeiten und Problemen, allerdings natürlich auch mit positiven Aspekten.

**Skizzierung der aufgetretenen Problematiken**

Bei der Durchführung traten durch diverse Faktoren verschiedene Probleme auf. Die langen Zeiten zwischen den jeweiligen Berufsschulblöcken sorgten dafür, dass es jedes Mal eine Einarbeitungsphase notwendig war, welche einige Zeit benötigt. Daneben hat ein neu zusammengestelltes Team anfangs auch Schwierigkeiten die eigene Teamleistung einzuschätzen, wodurch es zu Fehleinschätzungen kam. Außerdem empfanden wir das Tool „Flying Donut“, welches eigentlich den Entwicklungsprozess unterstützen sollte, als nicht intuitiv und umständlich zu bedienen. Fehler wie beispielsweise das Loggen von überschüssiger Zeit an einem Ticket, konnten aus unserer Sicht nicht rückgängig gemacht werden. Somit wurden natürlich auch Statistiken verfälscht.

**Beschreibung wie diese gelöst wurden**

Um die negativen Auswirkungen der auftretenden Probleme möglichst gering zu halten oder im Optimalfall sogar komplett zu eliminieren bedienten wir uns verschiedener Hilfsmittel. Um während den langen Pausen zwischen den Schulblöcken weiter organisiert an dem Projekt arbeiten zu können, wurde beispielsweise über verschiedene Kommunikationskanäle über das Projekt diskutiert und sich gegenseitig unterstützt. Auf Fehleinschätzungen und nicht planbare Probleme versuchten wir mit Anpassungen der Ziele und einer angepassten Priorisierung der Features zu reagieren.

**Fazit**

**Erkenntnisgewinn**

In Betrachtung des gesamten Projekts über den kompletten Zeitraum konnten diverse Erkenntnisgewinne verzeichnet werden. Bezüglich der Planung ist es definitiv notwendig zeitliche Puffer für nicht vorhersehbare Schwierigkeiten oder Veränderungen der Ressourcen etc. einzuplanen. Falls diese nicht benötigt werden, sind diese Zeiträume mit optionalen Features zu füllen. Bei der Planung ist auch von Vorteil, wenn bereits grobe technische Schwierigkeiten aufgenommen werden, damit diese bereits bei der Gesamtkonzeption Beachtung finden können. Fehler während der Konzeption von Features können zudem später weitreichende Folgen haben. Gerade bei absolut fundamentalen Bestandteilen des Projekts wie beispielweise der Datenbank ist es vorteilhaft mehr Zeit einzuplanen, um das Konzept bereits in Betrachtung der späteren Features zu überprüfen. Neben den negativen Aspekten, aus denen wir Erkenntnisse gewonnen haben, wurde natürlich auch während des Projekts viel über das allgemeine Zusammenarbeiten, das Organisieren eines Projekts und die verwendeten Technologien gelernt.

**Probleme**

Neben den bereits erwähnten Problemen stellten sich unserem Team weitere teilweise nicht vorhersehbare Probleme und Schwierigkeiten. Zunächst einmal zu den personellen Ressourcen: Die Umsetzung und der Umfang des Projekts wurde anfangs geplant mit der Verfügbarkeit von sechs Personen. Nach kurzer Zeit fiel jedoch eine davon aus und somit wurde die verfügbare Arbeitskraft um 20% gesenkt. Diese 20% konnten von uns im Verlauf des Projekts nicht wieder aufgeholt werden und fehlten schlichtweg.

Des Weiteren gab es unteranderem Probleme mit den Node-Modulen, die fehlerhaft installiert wurden, und ständig neue Schwierigkeiten verursachten. Selbst Neuinstallationen, löschen des entsprechenden Ordners und das Leeren des npm Caches konnten die Fehler nicht lösen. Außerdem störte es, dass einige Module weitere Modul-Installationen benötigten, weil die Dependencies nicht automatisch mit heruntergeladen wurden. Das tief verschachtelte Ordnersystem mit teilweise einer enormen Anzahl an Dateien führte beim Verschieben oder einfacher Einsicht in den Ordner zu Verlangsamungen. Es traten hin und wieder durch die Beschränkungen der PCs der Schule bedingte Zugriffsrecht-Konflikte auf, Git-Extensions konnten nicht starten oder stürzten bei Merges und Commits ab, weil es laut Fehlermeldung auf bestimmte Dateien keinen lesenden Zugriff bekam. Zudem dauerte das Einrichten der PCs, um diese in der Schule für die Entwicklung vorzubereiten, selbst das Hochfahren und Laden des Benutzerprofils, jedesmal einige Zeit. Diese Probleme traten auf einem normalen Heim-PC nie auf. Des Weiteren wurde sobald eine externe Festplatte oder USB-Stick an einem Schul-PC abgeschlossen wurde, auf dem Speichermedium Dateien beschädigt. Bei jedem Anschließen mussten zuerst eine Analyse und Dateireparatur durchgeführt werden, wodurch weitere Zeit verloren ging. Zuletzt fehlten bei Flying Donut für die zeitliche Übersicht helfende Informationen, vor allem Abschlusszeitpunkte, wann ein Ticket auf Done gesetzt wurde.