Abschlussprüfung Sommer 2024

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

Digitalisierung von Erstellung, Verwaltung & Validierung der Reisekostenabrechnungen mithilfe einer Single-Page-Application (SPA).

17.04.2024

Ricardo Hoffmann

Elpke 19a

33605 Bielefeld

Ausbildungsbetrieb

DTS Systeme GmbH

Schrewestraße 2

32051 Herford

# Inhaltsverzeichnis

# Einführung

Diese Projektdokumentation beschreibt den Ablauf des Abschlussprojekts, das der Autor im Rahmen seiner Abschlussprüfung zum Fachinformatiker für Anwendungsentwicklung bei der DTS Systeme GmbH durchgeführt hat. Die DTS Systeme GmbH ist der Ausbildungsbetrieb des Autors und wird im Abschnitt Projektumfeld näher beschrieben.

# Projektumfeld

# DTS Systeme GmbH ist ein internationaler Anbieter von IT-Lösungen und Services sowie Hersteller von Securitysoftware. Mit über 400 Mitarbeitern an 14 Standorten bieten wir unser Know-how in den Bereichen Datacenter, Technologies und Security an.

# Das Team für Software Development entwickelt und betreut Software wie DTS Monitoring, DTS Cockpit, DTS Identity Management oder DTS Identity as a Service und betreibt den hauseigenen Shop DTS Cloud Portal.

# Ausganssituation

Die DTS Systeme GmbH verwendet derzeit eine komplizierte, fehleranfällige und manipulationsanfällige Microsoft-Excel-Datei zur Erstellung von Reisekostenabrechnungen. Der Prozess ist außerdem sehr zeitaufwändig. Aus den genannten Gründen soll diese Microsoft-Excel-Datei abgelöst werden.

Die Microsoft-Excel-Datei wurde ursprünglich von einem externen Dienstleister im Auftrag der DTS Systeme GmbH erstellt. Seitdem wurde sie nicht mehr weiterentwickelt oder angepasst, weshalb einige ursprüngliche Funktionen und Felder nicht mehr benötigt werden.

# Projektbeschreibung

Das in der Ausgangssituation beschriebene Tool soll nun abgelöst werden.

Im Rahmen dieses Projekts soll eine neue Lösung implementiert werden, welches langfristig eingeführt werden soll. Anschließend soll das Lösung so erweitert werden, dass es in vollem Umfang genutzt werden kann.

Grob sind folgende Anforderungen an die Neuentwicklung zu stellen:

* Erstellung und Einreichung von Reisekostenabrechnungen

Nutzer sollen die Möglichkeit haben, Reisekostenabrechnungen in Form eines regelbasierten Chatbots zu erstellen.

* Berechnung von Kosten und Pauschalen

Die Softwarelösung sollte in der Lage sein, Berechnungen im Hintergrund durchzuführen und diese bei Bedarf anzuzeigen.

* Genehmigung und Ablehnung von Reisekostenabrechnungen

Vorgesetzte und Prüfer sollten die Möglichkeit haben, Reisekostenabrechnungen zu genehmigen oder abzulehnen.

* Integration des Hauseigenen IDP

Die Login-, Rechte- und Nutzerverwaltung der Softwarelösung soll durch den hauseigene IDP erfolgen, so dass keine zusätzlichen Zugangsdaten benötigt werden.

* Prüfung von Reisekostenabrechnungen

Die Prüfer sollten in der Lage sein, die Reisekostenabrechnung in einer tabellarischen Ansicht im Detail einzusehen und sich die Berechnungen im Detail anzeigen zu lassen.

Eine detaillierte Auflistung der Anforderungen an die Neuentwicklung ist im Anhang unter Lastenheft zu finden.

# Projektziel

Das Ziel soll sein, über ein neues, selbst entwickeltes Tool zu verfügen, das auf den Servern der DTS-Gruppe gehostet werden kann und für Mitarbeitende über einen Webbrowser erreichbar ist. Damit soll die allgemeine Unzufriedenheit mit der aktuellen Lösung und der zusätzliche Arbeitsaufwand für die Erstellenden, Prüfenden und Freigebenden Personengruppen reduzieren und vereinfachen.

# Projektschnittstellen

# Wie in der Projektbeschreibung erwähnt, wird die Softwarelösung eine Schnittstelle zum hauseigenen IDP verwenden, um Login, Rechte und Benutzerverwaltung zu übernehmen. Darüber hinaus verfügt die Softwarelösung über eine HTTP-API zur Kommunikation zwischen Front- und Backend sowie über eine Datenbankanbindung zur Speicherung der erforderlichen Daten.

# Projektabgrenzung

Da die zur Verfügung stehende Zeit begrenzt ist, muss der Projektumfang entsprechend eingegrenzt werden. So wurde unter anderem bewusst auf die Möglichkeit der Anpassung und Speicherung von Pauschalen für bestimmte Zeiträume verzichtet, da dies nicht in den engen Projektzeitplan passen würde. Eine detailliertere Auflistung der bewusst weggelassenen Features, um die Softwarelösung vollumfänglich nutzen zu können, ist unter Ausblick zu finden.

# Projektphasen und Zeitplanung

# Wie von der IHK Ostwestfalen zu Bielefeld vorgegeben, stehen für die Durchführung dieses Projektes 80 Stunden zur Verfügung. Eine Übersicht über die ursprüngliche Schätzung der Arbeitspakete (und die tatsächlich benötigte Zeit) kann dem Soll/Ist Vergleich entnommen werden.

# Ressourcenplanung

# Nach der Erstellung des Zeitplans wurde eine Übersicht über die Ressourcen erstellt, die voraussichtlich während der Projektlaufzeit benötigt werden. Diese wurde während der Projektlaufzeit laufend aktualisiert, so dass auch spontan hinzukommende Ressourcen abgebildet sind. Die Ressourcenplanung ist im Anhang unter Verwendete Ressourcen zu finden. Bei der Auswahl der Software wurde darauf geachtet, dass bereits Kenntnisse oder Erfahrungen mit der jeweiligen Software vorhanden waren, um einen möglichen Zeitverlust durch Einarbeitung möglichst gering zu halten. Darüber hinaus wurden die jeweiligen Lizenz- und Nutzungsbedingungen der verschiedenen Lösungen berücksichtigt.

# Ist Analyse

Wie bereits unter Ausgangssituation beschrieben, wird derzeit eine veraltete und unübersichtliche / fehleranfällige Microsoft Excel Datei zur Erstellung der Reisekostenabrechnung verwendet. Ein Bild dieser Microsoft Excel Datei befindet sich im Anhang unter Microsoft Excel Datei.

Grob Beschrieben muss der Mitarbeiter im Außendienst zunächst die Microsoft Excel Datei mit

Der Prozess zur Einreichung einer Reisekostenabrechnung ist im Anhang unter Prozess alt zu finden.

# Herausforderungen

# Wie in der Ausgangssituation beschrieben, ist diese Microsoft Excel Datei komplex, was zu Fehlern führt, wie z.B. dass Bedingungen falsch verstanden werden und dementsprechend eingetragen werden, obwohl sie nicht eingetragen werden sollten. Grundsätzlich kann es aufgrund der Art der Microsoft Excel Datei zu Manipulationen der Berechnungslogik oder der Pauschalen kommen. Zudem kommt es häufig vor, dass notwendige Belege nicht beigefügt werden oder die angegebenen Kosten nicht validiert werden können. Diese Herausforderungen erschweren den Einsatz in den Abteilungen, in denen es verwendet wird.

# Wirtschaftlichkeit

Von folgenden Stunden kosten der Mitarbeiter gehen wir aus:  
Accounting/ Payroll Accounting/ Betreuerin: 30€  
Mitarbeiter im Außendienst: 25€  
GF: 100€  
Azubi/Autor: 7€  
Junior Software Developer: 27€   
Vorgesetzter: 45€

Folgende Annahmen gelten universell:

* 20 Reisekostenabrechnungen im Monat
* Genehmigung des Vorgesetzten - 1 Vorgesetzter à 5 Minuten
* Genehmigung der GF - 1 GF einmal im Monat à 5 Minuten
* 10% der Ersteller wenden sich mit Fragen direkt an die Account Abteilung - 1 Mitarbeiter Accounting & 1 Mitarbeiter im Außendienst à 10 Minuten

Folgende Annahmen gelten für den Prozess rund um die Reisekostenabrechnung mit der bestehenden Microsoft Excel Datei:

* Reisekostenabrechnung erstellen - 1 Mitarbeiter im Außendienst à 15 Minuten
* Jeden 2. Monat Einführung eines Mitarbeiters - 2 Mitarbeiter im Außendienst à 1 Stunde
* 20% der Ersteller bitten Kollegen um Hilfe - 2 Mitarbeiter im Außendienst à 15 Minuten
* 90% der Reisekostenabrechnung wird über die Hauspost geliefert - 1 Azubi à 10 Minuten
* 10% der Reisekostenabrechnung wird selbst gebracht – 1 Mitarbeiter im Außendienst à 10 Minuten
* Prüfung auf Manipulation - 1 Mitarbeiter Payroll Accounting à 10 Minuten
* Prüfung auf Richtigkeit - 1 Mitarbeiter Accounting à 12 Minuten
* 25% der Reisekostenabrechnung müssen korrigiert werden – 1 Mitarbeiter im Außendienst à 10 Minuten
* 25% der Reisekostenabrechnung müssen nachgeprüft werden - 1 Mitarbeiter Accounting à 7 Minuten

Folgende Annahmen gelten für den Prozess rund um die Reisekostenabrechnung mit der Neuentwickelten Softwarelösung Travel-Assistant:

* Reisekostenabrechnung erstellen - 1 Mitarbeiter im Außendienst à 20 Minuten
* 10% der Ersteller bitten Kollegen um Hilfe - 2 Mitarbeiter im Außendienst à 5 Minuten
* Prüfung auf Richtigkeit - 1 Mitarbeiter Accounting à 7 Minuten
* 10% der Reisekostenabrechnung müssen korrigiert werden – 1 Mitarbeiter im Außendienst à 7 Minuten
* 10% der Reisekostenabrechnung müssen nachgeprüft werden - 1 Mitarbeiter Accounting à 5 Minuten
* Wartung der Softwarelösung – 1 Junior Software Developer à 2 Stunden

Entwicklungskosten des Travel-Assistant

* Implementierung - Autor à 80 Stunden
* Hilfestellung/Rat - Betreuerin à 5 Stunden
* Meetings/Abnahme und diverse Rückfragen - 1 Autor, 2 Accounting à 6 Stunden
* Feedback/Testing - 1 Mitarbeiter im Außendienst à 1 Stunde

Damit die Softwarelösung Vollumfänglich eingesetzt werden kann muss diese um einige Features weiterentwickelt werden, siehe Ausblick. Angenommen wird:

* Weiterentwicklung - 1 Junior Software Developer à 40 Stunden
* Zusätzliche Meetings - 1 Junior Software Developer, 2 Mitarbeiter Accounting à 4 Stunden

Diese Annahmen sind basierend auf Durchschnittliche bzw. erwartete werte, von denen man realistisch ausgehen kann.

# Anwendungsfälle und Benutzerklassifizierung

Um herauszufinden, wie das Berechtigungskonzept aussehen sollte, wurde analysiert, welche Personengruppen wie mit dem Tool arbeiten müssen.

Grundsätzlich lassen sich folgende drei Personengruppen feststellen:

• Reguläre Benutzer

Jede mitarbeitende Person, die berechtigt ist, Reisekostenabrechnungen einzureichen

• Vorgesetzte/GF

Personen, die berechtigt sind, Reisekostenabrechnungen anderer Personen zu bestätigen

• Prüfer

Personen, die zur Verwaltung von Benutzern und sonstigen Ressourcen berechtigt sind.

Außerdem ist im Anhang ein Anwendungsfalldiagramm zu finden welches die möglichen Aktionen der Jeweiligen Personengruppen zugeordnet wurden.

# Applikationsstart

Wie bereits unter Projektziel genannt, soll die Applikation als Webapplikation

entwickelt werden. Ein Grund dafür ist, dass man die Applikation so in der Zukunft

vergleichsweise einfach aktualisieren kann. Außerdem spricht die Tatsache dafür, dass man so von der Browserumgebung, mit zum Beispiel Passwortmanagern, profitiert.

Die Serverapplikation soll in JavaScript entwickelt werden. Es wurde sich für

JavaScript entschieden, da im Team Software Development grundsätzlich auf

JavaScript gesetzt wird. Die SPA soll mit React und JavaScript entwickelt werden. Die Wahl von React ergibt sich, da zahlreiche Applikationen der DTS-Gruppe bereits mit React umgesetzt worden sind oder aktuell umgesetzt werden. Dadurch kann gegebenenfalls vom Know-How anderer Personen im Team profitiert werden.

# Applikationsarchitektur

Die Applikation wird in zwei Teilapplikationen entwickelt werden: Frontend und

Backend. Diese Unterscheidung wird nötig, da sich bereits für die Implementierung einer SPA entschieden wurde. Eine SPA kann nicht sicher mit Daten interagieren, was eine Backendapplikation nötig macht.

# Frontend

Das Frontend wird dabei von einer SPA repräsentiert. Eine SPA ist eine Applikation,

die vollständig im Browser läuft. Im Gegensatz zu regulären Webanwendungen wird

nicht mit jeder Interaktion mit dem Server HTML zurückgegeben und vom Browser

dargestellt. Stattdessen wird die Interaktion durch browserseitig ausgeführtes

JavaScript durchgeführt.

# Backend

Das Backend ist die Teilapplikation, die auf dem Server läuft und die gesamte

Datenverwaltung sowie Businesslogik enthält. Das Backend speichert die Daten in einer

MongoDB und greift mit Mongoose darauf zu.

Es wird eine HTTP- API bereitgestellt, mit der das Frontend mit dem Backend

kommunizieren kann. Darüber können also Reisekostenabrechnungen eingereicht, bearbeitet,

genehmigt und geprüft werden.

Das Backend wird in der MSC-Architektur entwickelt. Dabei werden die Komponenten

nach Typ, geteilt. Bei der MSC-Architektur gibt es folgende Komponentenarten:

## 1.Model

Das Model beschreibt die zu speichernden Daten im Rahmen einer einzelnen

Entität. Außerdem werden im Model die klassischen CRUD-Operationen

durchgeführt.

## 2. Service

Im Service findet die gesamte Logik statt. Services sind die einzigen Komponenten

der Applikation, die mit Models interagieren. Sie bieten die Möglichkeit, die

CRUD-, und gegebenenfalls noch weitere, Operationen auszuführen.

## 3. Controller

Controller sind die Komponenten, die die tatsächliche Interaktion mit der

Außenwelt, also dem HTTP-Client bestreiten. Sie interagieren mit den Services,

um die Anfragen des Clients auszuführen.

Neben diesen Hauptkomponenten gibt es noch folgende Komponenten:

## 1. Middleware

Neben den mitgelieferten oder nachinstallierten Middlewares können auch eigene

Middlewares erstellt werden. Im konkreten Fall wird eine Middleware genutzt, die

den Zugang zu bestimmten Bereichen der Applikation beschränkt. Dadurch wird

eine Anfrage, bevor sie überhaupt im Controller ankommt, bereits beantwortet,

falls bestimmte Anforderungen nicht erfüllt werden.

## 2. Utilities

Utilities sind Hilfsfunktionen, die in keinen der Bereiche wirklich passen, die aber

auch unabhängig von diesen verwendet werden können.

# Models und Datenstruktur

Durch die Anforderungen ist klar, dass mehrere Models benötigt werden.

So wird ein Model für Länder und ihre Pauschalen benötigt. Die Reisekostenabrechnung an sich welche sich aus dem Chat verlauf ergibt sowie grundlegende Informationen und Dienstreisen welche Informationen aus dem Chat verwertet. Sowie ein Model, welches einige grundlegende Berechnung speichert und Benutzer um die jeweiligen Vorgesetzten zu speichern.

# Feststellung der Benutzerberechtigungen

Das in Benutzerklassifizierung beschriebene Benutzerkonzept sieht drei

Benutzerklassen vor. Dabei wird davon ausgegangen, dass jeder Nutzer, der sich an der Applikation bzw. den IDP einloggt, initial ein regulärer Benutzer ist. Im Anschluss wird dann geprüft, ob ein Nutzer über eine Rolle verfügt, die über den IDP dem Nutzer zugeordnet wird.

# Durchführungsphase

## Auswahl von Libraries

Bei der Auswahl von Libraries wurde einigen Aspekten eine besondere Beachtung

geschenkt:

• Lizenz

Selbstverständlich muss die Lizenz der Library die Nutzung im konkreten Umfeld

erlauben.

• Beliebtheit

Für die Beliebtheit wurde sich an den Metriken der GitHub-Stars und

wöchentlichen Downloads bedient. Je beliebter eine Library ist, desto höher ist

die Wahrscheinlichkeit, dass sie langfristig weiter gepflegt wird.

• Dependencies

Neben der Lizenz und der Beliebtheit ist auch die Anzahl und Art der

Dependencies der Library ein wichtiger Punkt. Eine Library, die weniger weitere

Dependencies hat, kann besser eingesetzt werden, da man so besser kontrollieren

kann, welcher Code in das Projekt gelangt.

## Umgang mit Git/Source Code Management und Aufgabenaufteilung/management

### git

Im Unternehmen wird Git zur Versionsverwaltung genutzt. Git ist ein Versionskontrollsystem, das ursprünglich zur Entwicklung des Linux-Kernels entwickelt worden ist. Git wurde 2005 veröffentlicht und ist heute de-facto Industriestandard

### Git Flow

Beim Workflow wird sich an Git-Flow orientiert. Git-Flow ist ein Branching-Modell, das 2010 von Vincent Driessen veröffentlicht worden ist. In Git-Flow gibt es zwei Haupt-Branches: master und develop. Der develop-Branch enthält zu jeder Zeit eine lauffähige Version der Applikation, der master-Branch enthalt die aktuell produktive Version. Neben diesen beiden Haupt-Branches gibt es sogenannte Feature-Branches. In ihnen werden neue Features hinzugefügt und weiterentwickelt. In der Regel werden die Branch-Bezeichnungen mit feature/-geprefixt. Neue Features werden aus dem develop-Branch erstellt und auch wieder zurück in diesen gemerget. Außerdem gibt es die Release-Branches. In ihnen werden letzte Vorbereitungen für ein bevorstehendes Release getroffen, wie zum Beispiel die Inkrementierung der Versionsnummer. In der Regel werden die Branch-Bezeichnungen mit release/ geprefixt. Neue Releases werden aus dem develop-Branch erstellt und sowohl in den master-Branch als auch in den develop-Branch zurück gemerget. Sollte es in der Produktivumgebung zu Fehlern kommen, werden diese in Hotfixes behoben. Dazu werden Hotfix-Branches genutzt, die mit hotfix/ geprefixt werden und aus dem master-Branch erstellt werden. Fehlerbehebungen werden dann, genau wie die Releasevorbereitungen, in sowohl master als auch in develop gemerget. Diese Mechanik wird angewandt, um sicherzustellen, dass niemals Features, die noch nicht released waren, im Rahmen eines Hotfixes released werden. Aktuellen Empfehlungen nach ist der master-Branch aus politischen Gründen nun als main zu bezeichnen.

### Aufgabenaufteilung/Aufgabenmanagement

Als das Vorgehensmodell hat der Autor sich für das Kanbanboard entschieden, da dieses Hauptsächlich in dem Software-Entwicklungsteam verwendet wird. Die Anforderungen wurden in Aufgaben unterteilt und grob priorisiert, hierzu wurde das Online Web Tool Miro verwendet und die Aufgaben und deren Status zu visualisieren.

Ein Auszug, der des Kanbanboards ist im Anhang zu finden.

## Erstellung des Applikationsgrundgerüsts

Zuerst wurde das Grundgerüst der Applikation vorbereitet. Dazu wurde eine Git-Repository erstellt. In der Repository wurden zwei Hauptverzeichnisse erstellt: Backend und Frontend. Im Backend-Verzeichnis befindet sich das JavaScript-Projekt der Backendapplikation, im frontend-Verzeichnis das mit vite erstellte React- JavaScript-Projekt. Außerdem befinden sich im Hauptverzeichnis die .gitignore-Datei zum Ausschließen von Dateien aus dem Git-Repository hierfür wurde eine Template für gängige Datei(Endungen) herangezogen, eine .env-Datei mit dazugehöriger .env.sample-Datei(im Anhang zu finden) zum Bereitstellen von Umgebungsvariablen, wie z.B. client Id und secret um sich bei dem IDP zu authentifizieren. Und die für Docker nötige Dateien docker-compose.yml und Dockerfile.

# Vorbereitung und Implementierung Businesslogik der Backendapplikation

Nachdem das Grundgerüst der Applikation aufgebaut war, wurde sich der Implementierung der Backendapplikation gewidmet. Dabei wurde zuerst die Berechnungslogik der Pauschalen angegangen da diese Berechnungsgrundlagen schon in der Microsoft Excel Datei vorhanden waren und so einfach analysiert und in Programm Code umgesetzt wurden konnten. Danach wurde mit einer simplen http API begonnen, die dann im Verlaufe der Implementierung der SPA immer der konkreten Anforderung entsprechend angepasst wurde. Mit den Entstehenden Endpoints für das Frontend wurde parallel auch die Backend Logik für die CRUD-Operationen entwickelt. Bei der Implementation einer so umfangreichen Applikation werden nahezu immer externe Libraries eingesetzt. Das ist auch in diesem Projekt passiert. Diese Übersicht enthält die wichtigsten genutzten Libraries:

express ist eine der Libraries, die eingesetzt werden, wenn im Node.js-Umfeld ein HTTP-Server bereitgestellt werden soll. Es wird eine auf Middleware basierende API bereitgestellt, mit der man nahezu alle Szenarien abdecken kann.

mongoose ist das ODM für MongoDB. Man kann damit die zu verwaltenden Datenstrukturen beschreiben und so eine gewisse Sicherheit über die tatsächlich vorhandenen Daten haben.

DTS entwickelte interne libraries kamen auch zum Nutzen wie der dts-node-oidc-client welche wir für interne applikationen zu unserem eigenen IDP verwenden und der dts-node-logger welcher ein erweiterten logger für node.js Anwendungen speziel für DTS-Produkte verwendet wird.

Im Anhang findet sich unter Libraries (Backend) eine detailliertere Übersicht über die verwendeten Libraries.

# Verbindung zum IDP

Bei jeder Anfrage an das Backend wir mithilfe einer Authentifizierungs Middleware geprüft, ob der mitgesendete Token (wenn vorhanden ist) valide ist, falls nicht wird der Nutzer auf die Login Seite des IDP geleitet, wo der Nutzer sich Authentifizieren muss, sobald das stattgefunden hat, wird der Nutzer zurück auf den Travel-Assistant mit einem Token geleitet. Dieser Token enthält essenziellen Informationen wie Vor-/Nachname, E-Mail und welche Rollen der Nutzer hat, um dementsprechend um das Frontend darauf auszurichten.

# Vorbereitung der Single-Page-Application

Nachdem der Großteil der API im Backend verfügbar war, wurde mit der Implementierung der SPA begonnen. Dabei wurde sich den unternehmensinternen Stylingrichtlinien und der entsprechenden Vorlage bedient, um das Projekt entsprechend den Vorgaben zu gestalten. Die Vorlage ist allerdings für eine herkömmliche, auf ejs basierende, Webapplikation ausgerichtet und musste daher zuerst zur Nutzung mit React angepasst werden. Außerdem sind während der Implementierung immer wieder kleinere Mängel und Unschönheiten aufgefallen, die behoben werden mussten. Im Anhang findet sich unter Libraries (Frontend) eine detaillierte Übersicht über die verwendeten Libraries.

# Soll/Ist-Vergleich

Das Projekt wurde wie geplant umgesetzt. Die Zeitplanung ist allerdings nicht vollständig aufgegangen:

In der Projektdefinitionsphase konnte man die eingeplante Zeit einhalten und sogar verkürzen. Durch diverse in der Vergangenheit liegende Berührungspunkte ist die Ist-Analysen relativ einfach ausgefallen.

Die Projekt Planungsphase wurde die grundlegende Zeit eingehalten jedoch mit kleineren Schwankungen, welche sich innerhalb dieser Phase durch andere ausgeglichen hatten. So musste für das Erstellen des Anwendungsfalldiagramms und Sketches weniger Zeit in Anspruch genommen werden da im Schulischen Kontext eine Auffrischung stattfand.

Da In der Entwicklung einige Probleme auftraten im Bezug auf Speicherung der benötigten Inhalte musste die Datenbank einige male angepasst werden bzw. teilweise um konzeptioniert werden.

Den Großteil der Stunden nimmt die Projektdurchführungsphase in Anspruch. Diese Phase hat mehr Zeit in Anspruch genommen als ursprünglich geplant.  
So wurden für Erstellung der Docker Umgebung und Grundstruktur der Webanwendung weniger Zeit in Anspruch genommen da durch die Regelmäßige Anwendung in anderen Projekten dieses schneller umgesetzt werden konnte als anfangs geplant. Massiv Zeit konnte bei der Berechnungslogik eingespart werden. Grund hierfür sind wegfallende Berechnungen, welche nicht mehr benötigt werden, somit ist auch der Auffand für Qualitätssicherung und Test dieser Kristischen Funktionen weniger geworden. Etwas unterschätzt war der Aufwand der Erstellung und Strukturierung der Fragen. So musste, um Überblick zu behalten, wie die Abfolge der Fragen stattfand ein Dessicion Tree erstellt werden, um nachvollziehen zu können an welcher stelle der code/chat sich befindet und zu welcher folge fragen diese führt. Bei der Implementierung der Mobil-First Oberflächen wurde mehr Zeit in Anspruch genommen da anfängliche Schwierigkeiten mit dem Framework React hatte und zusätzliche Recherche benötigt wurde. Die Intigration des IDP’s hat ebenso mehr zeit in Anspruch genommen da der Autor bisher nur Server Side Applikation mit dem IDP verbunden hat und nicht mit einer SPA welche durch einen Umweg

Die Projektabschlussphase wurde auch etwas zeit eingespart da die Abnahme nur durch eine Präsentation des Tools und den Ausblick enthielt da der Travel-Assisant in der ersten version welches sie ist nicht produktiv einsetzbar ist. Ebenso wurde Zeit eingespart bei der internen Dokumentation da sich immer parallel zum Erstellen von Funktionen eine entsprechende Dokumentation dieser erstellt wurde mithilfe von JSdoc. Die Zeit für die Erstellung von der Projekt Dokumentation ist etwas höher ausgefallen als geplant durch den schieren umfang dieser.

|  |  |  |
| --- | --- | --- |
| **Bezeichnung** | **Eingeplant** | **Tatsächlich** |
| **Projektdefinitionsphase**   * Durchführung Ist-Analyse * Durchführung Amortisationsrechnung * Erstellung eines Lastenhefts (Unterstützung des Fachbereichs) | **4**  1  1  2 | **3,5**  0,5  1  2 |
| **Projektplanungsphase**   * Erstellung eines Anwendungsfalldiagramm * Erstellung eines Aktivitätsdiagramm * Erstellung eines Pflichtenheftes * Datenbank Modellierung * Erstellung von Sketches * Einteilung von Arbeitspaketen (Kanban Board) | **10**  1  1  2  3  2  1 | **10**  0,5  1  2  4  1,5  1 |
| **Projektdurchführungsphase**   * Container Umgebung erstellen * Grundstruktur der Webanwendung Implementieren * IDP-Integration * Chatfragen für den Regelbasierenden Chatbot formulieren & strukturieren * Implementierung der Backend-Logik   + Berechnungslogik (basierend auf Pauschalen)   + Create, Read, Update, Delete (CRUD) Funktionen für die Reisekostenabrechnung   + Application Programming Interface (API) für das Frontend zur Datenbereitstellung * Implementierung des Frontends   + Mobile-First Oberfläche für CRUD-Operation der Reisekostenabrechnung   + Tabellenansicht zur Überprüfung der Reisekostenabrechnung * Qualitätssicherung und Tests (der Kritischen Berechnungsfunktionen) | **50**  1  5  1  4  7  6  4  8  8  6 | **51,5**  0,5  4  2  6  5  8  4  12  8  4 |
| **Projektabschlussphase**   * Abnahme * Interne Dokumentation * Projektdokumentation | **16**  2  3  11 | **15**  1  2  12 |

# Lesson Learned

Im Laufe des Projekts sind einige Herausforderungen auf den Autor zugekommen. Diese wurden im Allgemeinen jedoch gemeistert, was zu einem Lerneffekt geführt hat. Außerdem konnten die Fähigkeiten im Bereich der Frontendentwicklung ausgebaut werden - sonst liegt der Fokus eher im Bereich der Backendentwicklung. Eine tatsächlich zu benennende Lesson Learned ist, dass es manchmal sinnvoller ist, eine simplere Lösung zu verwenden als eine vermeintlich schönere aber wesentlich komplexere - das sorgt dafür, dass der Code einfach verständlicher und besser nachzuvollziehen ist.

# Ausblick

Wie bereits mehrfach erwähnt, ist das Produkt in seiner aktuellen Form noch nicht produktiv einzusetzen. Dazu werden noch einige Funktionalitäten benötigt:

* Englische Übersetzung nicht Deutschsprachige Kollegen
* Benachrichtigungen per E-Mail (z.B. Freigaben oder Status Änderungen)
* Konfiguration von Pauschalen
* Logging von Benutzer Aktionen