



Industrie- und Handelskammer
Ostwestfalen zu Bielefeld

Abschlussprüfung Sommer 2024
Fachinformatiker für Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

Travel-Assistant

Digitalisierung von Erstellung, Verwaltung & Validierung der
Reisekostenabrechnungen mithilfe einer Single-Page-Application

17. April 2024

Ricardo Hoffmann

Elpke 19a
33605 Bielefeld



Ausbildungsbetrieb:

DTS Systeme GmbH
Schrewestraße 2
32051 Herford

Inhaltsverzeichnis

1	Einführung und Definitionsphase	4
1.1	Projektumfeld	4
1.2	Ausgangssituation	4
1.3	Projektbeschreibung	5
1.4	Projektziel	5
1.5	Projektschnittstellen	6
1.6	Projektabgrenzung	6
2	Projektplanung	6
2.1	Projektphasen und Zeitplanung	6
2.2	Ressourcenplanung	7
3	Analysephase	7
3.1	Ist-Analyse	7
3.1.1	Reisekostenabrechnungsprozess	7
3.1.2	Herausforderungen	8
3.2	Soll-Zustand	8
3.3	Wirtschaftlichkeit	8
3.4	Anwendungsfälle und Benutzerklassifizierung	11
4	Planungsphase	12
4.1	Applikationsart	12
4.2	Applikationsarchitektur	12
4.2.1	Frontend	12
4.2.2	Backend	13
4.3	Modelle und Datenstruktur	14
4.3.1	Chatbot Datenstruktur und Logik	15
4.4	Feststellung der Benutzerberechtigungen	15
5	Durchführungsphase	16
5.1	Auswahl von Libraries	16
5.2	Versionierung	16
5.3	Aufgabenaufteilung/Aufgabenmanagement	17

5.4	Vorbereitung des Applikationsgrundgerüsts	17
5.5	Implementierung Backendapplikation	17
5.6	Implementierung Single-Page-Application	18
5.7	Implementierung Chatbot	18
5.8	Verbindung zum IdP	18
5.9	Testing	19
6	Abschlussphase	19
6.1	Soll/Ist-Vergleich	19
6.2	Lessons Learned	20
6.3	Ausblick	21
7	Glossar	21
8	Anhang	23
8.1	Microsoft Excel Datei	24
8.2	Lastenheft	25
8.3	Verwendete Ressourcen	29
8.4	Soll-/Ist-Zeitplanung	30
8.5	Prozess Alt	31
8.6	Amortisationsrechnung	32
8.7	Anwendungsfalldiagramm	36
8.8	IdP-Login	37
8.9	Kanban-Board	38
8.10	Libraries	39
8.10.1	Libraries (Backend)	39
8.10.2	Libraries (Frontend)	39
8.11	Code-Ausschnitte	40
8.11.1	Berechnungs-Ausschnitt	40
8.11.2	Unittest-Ausschnitt	40
8.11.3	Chatbot-Ausschnitt	41
8.12	Decision Tree	41
8.13	Sketches	48



Legende:

1. [Links innerhalb des Dokuments](#) 🔗
2. [Links zu externen Internetseiten](#) ↗
3. [Links zu Glossareinträgen](#)

1 Einführung und Definitionsphase

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.

1.1 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.

1.2 Ausgangssituation

Die DTS Systeme GmbH verwendet aktuell eine Komplexe, Fehler-/ Manipulationsanfällige Microsoft Excel Datei (MED) zur Erstellung von Reisekostenabrechnungen (RAs). Der Prozess ist außerdem sehr zeitaufwändig. Aus den genannten Gründen soll diese MED abgelöst werden. Die MED wurde ursprünglich von einem externen Dienstleister im Auftrag der DTS Systeme GmbH erstellt. Seitdem wurde sie nicht mehr weiterentwickelt oder angepasst. Ein Bild der derzeit verwendeten MED ist im Anhang unter [Microsoft Excel Datei](#) 🔗 zu finden.



1.3 Projektbeschreibung

Die in der [Ausgangssituation](#) beschriebene MED soll nun abgelöst werden. Im Rahmen dieses Projekts soll eine neue Lösung implementiert werden, welches langfristig eingeführt werden soll. Anschließend soll die Lösung so erweitert werden, dass es in vollem Umfang genutzt werden kann. Grob werden folgende Anforderungen an die Neuentwicklung gestellt:

1. Erstellung und Einreichung von Reisekostenabrechnungen

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

2. Berechnung von Kosten und Pauschalen

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

3. Genehmigung und Ablehnung von Reisekostenabrechnungen

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

4. Integration des Hauseigenen Identity Provider (IdP)

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

5. Prüfung von Reisekostenabrechnungen

Die Prüfer sollten in der Lage sein, die RA 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.

1.4 Projektziel

Das Ziel ist eine neue, selbst entwickelte Softwarelösung, die die aktuelle Methode der Erstellung von RA ablöst. Die Software soll auf dem Server der DTS-Gruppe gehostet werden und über einen Webbrowser für Mitarbeiter (MA) erreichbar sein. Dadurch soll



die allgemeine Unzufriedenheit mit der aktuellen Lösung reduziert und der Arbeitsaufwand für die erstellenden, prüfenden und freigebenden Personengruppen vereinfacht werden.

1.5 Projektschnittstellen

Wie in der [Projektbeschreibung](#) erwähnt, wird die Softwarelösung eine Schnittstelle zum hauseigenen IdP verwenden, um Login, Rechte und Benutzerverwaltung zu übernehmen. Darüber hinaus wird eine REST-API zur Kommunikation zwischen Front- und Backend sowie über eine Datenbankbindung zur Speicherung der erforderlichen Daten implementiert.

1.6 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.



2 Projektplanung

2.1 Projektphasen und Zeitplanung

Wie von der IHK Ostwestfalen zu Bielefeld vorgegeben, stehen für die Durchführung dieses Projektes 80 Stunden zur Verfügung. Dazu wurden grobe Arbeitspakete definiert und der Zeitaufwand für deren Bearbeitung geschätzt. Eine Übersicht über die Arbeitspakete und die ursprüngliche Schätzung des Zeitaufwandes (und der tatsächlich benötigten Zeit) ist im Anhang unter Zeitplan [Soll-/Ist-Zeitplanung](#) zu finden.




2.2 Ressourcenplanung

Eine Übersicht über die während der Projektlaufzeit voraussichtlich benötigten Ressourcen findet sich im Anhang unter [Verwendete Ressourcen](#) . Diese Übersicht wurde während der Projektlaufzeit laufend aktualisiert, so dass auch während der Projektlaufzeit neu hinzukommende Ressourcen aufgeführt sind. Bei der Auswahl der Software wurde darauf geachtet, dass bereits Kenntnisse oder Erfahrungen vorhanden waren, um mögliche Risiken zu minimieren und den Zeitaufwand für die Einarbeitung so gering wie möglich zu halten. Darüber hinaus wurden die jeweiligen Lizenz- und Nutzungsbedingungen berücksichtigt. Weitere Aspekte sind unter [Auswahl von Libraries](#)  zu finden.


3 Analysephase

3.1 Ist-Analyse

Wie bereits unter [Ausgangssituation](#)  beschrieben, wird derzeit eine veraltete und unübersichtliche / fehleranfällige MED zur Erstellung der RA verwendet.

3.1.1 Reisekostenabrechnungsprozess

Der MA im Außendienst füllt die RA aus und legt sie seinem Vorgesetzten (und ggf. der Geschäftsführung (GF)) vor, welche prüfen, ob die Dienstreise(n) genehmigt wurde(n). Ist dies der Fall, wird diese an die Payroll Accounting Abteilung weitergeleitet und dort auf Manipulationen überprüft. Weist die RA keine Manipulationen vor, kann sie an die Accounting Abteilung weitergeleitet werden, die die Berechnungen und die Übereinstimmung der Belege mit den eingetragenen Kosten überprüft. Ist die RA in Ordnung, kann eine entsprechende Auszahlung veranlasst werden. Sollte jedoch in einem der oben genannten Fälle etwas mit der RA nicht in Ordnung sein, muss diese vom Mitarbeiter bearbeitet werden oder wird schlichtweg abgelehnt.

Eine grobe Darstellung des aktuellen Reisekostenabrechnungsprozesses ist in visueller Form im Anhang unter [Prozess Alt](#)  zu finden.



3.1.2 Herausforderungen

Wie in der [Ausgangssituation](#) beschrieben, ist diese MED 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 MED 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 diese verwendet wird.

3.2 Soll-Zustand

Die MED soll durch eine moderne SPA ersetzt werden. Diese SPA soll die Möglichkeit bieten, eine RA digital zu erstellen, zu bearbeiten, freizugeben und zu prüfen. Ein regelbasierter Chatbot soll dem MA zu der Dienstreise Fragen stellen und die Antworten im Hintergrund verarbeiten. Im Anhang unter [Lastenheft](#) können nähere Details zum Soll-Zustand/Anforderungen entnommen werden.

3.3 Wirtschaftlichkeit

Da diese Softwarelösung die veraltete MED ablösen soll, spielt die Wirtschaftlichkeit eine untergeordnete Rolle. Dennoch können monatliche Einsparungen erzielt werden, die sich anhand folgender Annahmen/Durchschnittswerten ergeben.

Von folgenden Personalkosten wird ausgegangen:

- Azubi/Autor: 7€/h
- MA im Außendienst: 25€/h
- Junior Software Developer: 27€/h
- Accounting Abteilung/ Payroll Accounting/ Betreuerin: 30€/h
- Vorgesetzter: 45€/h
- GF: 100€/h



Folgende Annahmen gelten universell:

- 20 RAs werden im Monat eingereicht
- 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 MA Accounting, 1 MA im Außendienst à 10 Minuten

Folgende Annahmen gelten für den Prozess rund um die RA mit der bestehenden MED:

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

Summe der monatlichen Kosten durch die aktuelle Lösung: 589,33 €

Folgende Annahmen gelten für den Prozess rund um die RA mit der Neu entwickelten Softwarelösung Travel-Assistant:

- RA erstellen - 1 Mitarbeiter im Außendienst à 20 Minuten
- 10% der Ersteller bitten Kollegen um Hilfe - 2 MA im Außendienst à 5 Minuten
- Prüfung auf Richtigkeit - 1 MA Accounting à 7 Minuten



- 10% der RAs müssen korrigiert werden – 1 MA im Außendienst à 7 Minuten
- 10% der RAs müssen nachgeprüft werden - 1 MA Accounting à 5 Minuten
- Wartung der Softwarelösung – 1 Junior Software Developer à 2 Stunden

Summe der monatlichen Kosten durch die neuentwickelte Softwarelösung: 411,50€

Monatliche Kosten Ersparnis: 177,83€

Entwicklungskosten des Travel-Assistant:

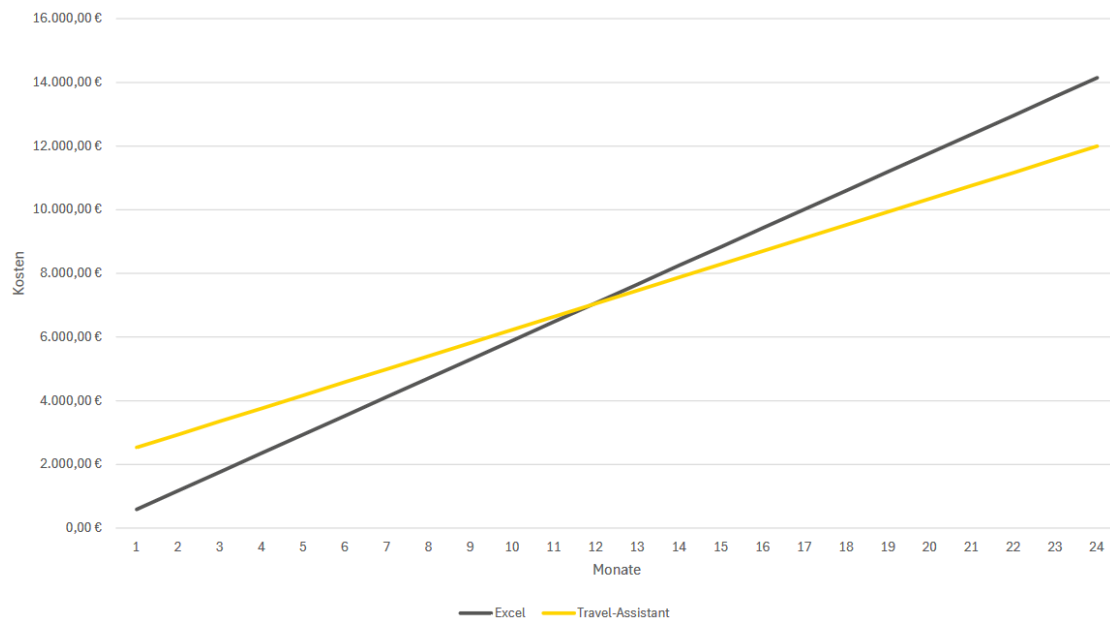
- Implementierung - Autor à 80 Stunden
- Hilfestellung/Rat - Betreuerin à 5 Stunden
- Meetings/Abnahme und diverse Rückfragen - 1 Autor, 2 MA Accounting à 6 Stunden
- Feedback/Testing - 1 MA im Außendienst à 1 Stunde

Summe Projekt Entwicklungskosten: 1.095,00€

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

- Weiterentwicklung - 1 Junior Software Developer à 25 Stunden
- Zusätzliche Meetings - 1 Junior Software Developer, 2 MA Accounting à 4 Stunden

Summe Entwicklungskosten bis zum Vollumfänglich einsatz: 1.023€ (2.118€)



Damit ergibt sich ein Break Even Point bei ungefähr 12 Monaten.

Diese Annahmen basieren auf Durchschnitts- oder Erwartungswerten, von denen realistischerweise ausgegangen werden kann. Detaillierte Berechnung ist im Anhang unter [Amortisationsrechnung](#) zu finden.

3.4 Anwendungsfälle und Benutzerklassifizierung

Um herauszufinden, wie das Berechtigungskonzept aussehen sollte, wurde analysiert, welche Personengruppen wie mit der Softwarelösung arbeiten müssen. Grundsätzlich lassen sich folgende drei Personengruppen feststellen:

1. Reguläre Benutzer

Jede mitarbeitende Person, die berechtigt ist, RAs einzureichen.

2. Vorgesetzte/GF

Personen, die berechtigt sind, RAs anderer Personen zu bestätigen.

3. Prüfer

Personen, die berechtigt sind, RAs auf ihre Richtigkeit zu prüfen.



Im Anhang ist ein [Anwendungsfalldiagramm](#) zu finden welches möglichen Aktionen der Jeweiligen Personengruppen zuordnet.

4 Planungsphase

Nachdem die aktuelle Situation analysiert wurde, wurde mit der Planung der Applikation begonnen. Die folgenden Abschnitte Orientieren sich anhand der vorgaben des [Pflichtenhefts](#) // **TODO!!** IST NOCH NICHT FERTIG .

4.1 Applikationsart

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. Der Technologie Stack **MERN** wird für die Erstellung der Softwarelösung gewählt, weil dieser generell etabliert und bereits in dem Software Development Team verwendet und in diverse Projekten bereits umgesetzt wird. Dadurch kann gegebenenfalls vom Know-How anderer Personen im Team profitiert werden. So wird die Serverapplikation mit **Express** auf der Laufzeitumgebung **Node.js** und **SPA** mit **React** entwickelt und **MongoDB** zur Speicherung von Daten verwendet werden.

4.2 Applikationsarchitektur

Die Softwarelösung wird in zwei Teilapplikationen entwickelt: [Frontend](#) und [Backend](#). Eine Backend Applikation ist Notwenig da sich bereits auf eine **SPA** festgelegt wurde. Da eine **SPA** nicht sicher mit Daten interagieren kann.

4.2.1 Frontend

Das Frontend wird durch eine **SPA** repräsentiert. Eine **SPA** ist eine Applikation, die vollständig im Browser ausgeführt wird. Im Gegensatz zu üblichen Server-Side Webanwendungen wird nicht bei jeder Interaktion mit dem Server **HTML** zurückgegeben und vom Browser dargestellt. Statt dessen wird die Interaktion durch browserseitiges **JavaScript**



ausgeführt. So wird beispielsweise bei der Erstellung einer RA mithilfe des Chatbots, eine Anfrage an das Backend gestellt, welches dann eine passende Frage zurückgibt. Diese Informationen werden anschließend vom Frontend verarbeitet und angezeigt. Das Frontend wird nach dem Layout der zuvor erstellten Sketches (im Anhang unter [Sketches](#) zu finden) entwickelt und gestaltet.

4.2.2 Backend

Das Backend läuft auf dem Server und übernimmt die gesamte Datenverwaltung und Businesslogik. Mit Mongoose kann das Backend dann auf die gespeicherten Daten der MongoDB zugreifen. Da die beiden Teilapplikationen miteinander kommunizieren müssen stellt das Backend eine REST-API zur Verfügung. Über diese kann dann das Frontend Informationen senden. Um so z.B. RAs einreichen zu können. Das Backend wird auf Basis der MSC-Architektur entwickelt. Dabei werden die Komponenten nach Typ unterteilt. In der MSC-Architektur gibt es folgende Komponententypen:

1. Model

Das Model beschreibt die zu speichernden Daten im Rahmen einer einzelnen Entität. So werden im Model die klassischen CRUD-Operationen durchgeführt. Wie das Kreieren einer RA.

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. Wie die Hintergrundberechnungen von Pauschalen.

3. Controller

Controller sind die Komponenten, die die tatsächliche Interaktion mit der Außenwelt, also dem HTTP-Client übernehmen. Sie interagieren mit den Services, um die Anfragen des Clients auszuführen. Der Punkt, an dem die Daten vom Frontend an die Services zur Verarbeitung gesendet werden.

Neben diesen Hauptkomponenten gibt es die folgenden Komponenten:



- **Middleware**

Neben den vorinstallierten oder heruntergeladenen Middlewares können auch eigene erstellt werden. Im konkreten Fall wird eine Middleware verwendet, die den Zugriff auf bestimmte Bereiche der Applikation einschränkt. Dadurch wird eine Anfrage bereits vor dem Eintreffen im Controller beantwortet, falls bestimmte Anforderungen nicht erfüllt sind. So können z.B. nur Benutzer mit der Prüfer Rolle die Prüfungsansicht einsehen.

- **Tests**

Die Tests dienen dazu, die einzelnen Komponenten auf verschiedene erwartete Ergebnisse hin zu überprüfen. Dabei werden die kritischen Berechnungsfunktionen auf ihr vorgesehenes Verhalten geprüft.

Eine Kernkomponente des Backends ist die Logik des Chatbots. Hierfür werden Endpoints bereitgestellt, die über das Frontend erreicht werden können. Bei jeder Benutzerantwort erhält das Backend eine Anfrage, die die ursprüngliche Frage und die dazugehörige Antwort enthält. Das Backend prüft nun, welche Folgefrage anhand der gegebenen Antwort zu dieser ursprünglichen Frage zurück an das Frontend gegeben wird. Parallel dazu wird die Antwort gespeichert.

4.3 Modelle und Datenstruktur

Die Anforderungen machen deutlich, dass mehrere Modelle und Datenstrukturen erforderlich sind. So wird ein Modell für die Länder und deren Pauschalen benötigt, die Reisekostenabrechnung, die sich aus dem Chatverlauf und Dienstreisen ergibt welche die Informationen aus dem Chat verarbeitet abspeichert, sowie ein Modell, das einige Berechnungsgrundlagen speichert. Außerdem wird ein Benutzermodell benötigt, um die jeweiligen Vorgesetzten zu speichern und ein Modell welches die Fragen für den Chatbot abspeichert. Die Chatverläufe müssen deswegen gespeichert werden, um eine spätere Anpassung der RA zu ermöglichen.



4.3.1 Chatbot Datenstruktur und Logik

Die Struktur wurde so gestaltet, dass das Backend schnell und einfach die nächste Folgefrage ermitteln kann. Hierbei verweist die Ausgangsfrage auf die Folgefragen. Diese Struktur bietet Vorteile in Bezug auf die Wartbarkeit, da bei Anpassungen an den Fragen keine Änderungen am Quellcode notwendig sind.

Die Chatbot-Fragen sind in der Datenbank wie folgt gespeichert:

```
{
  "questionId": "ask.mealsProvided.boolean",
  "content": "Wurde an einem der Tage Frühstück, Mittagessen oder Abendessen gestellt?",
  "followingAnswerType": "boolean",
  "followingAnswerAttribute": "mealsProvided",
  "editable": false,
  "condition": "truthy",
  "nextQuestions": {
    "true": "ask.breakfastProvided.boolean",
    "false": "ask.paidForTransport.boolean"
  }
}
```

Um Folgefragen zu ermitteln, wird das Objekt `nextQuestions` in Verbindung mit dem Attribut `condition` verwendet, um eine If-Abfrage zu bilden. Dadurch wird geprüft, ob die Antwort auf diese Frage `true` oder `false` ist, und basierend auf dem Ergebnis wird die entsprechende Frage aus `nextQuestions` an das Frontend zurückgegeben.

Bei Fragen die keine `condition` haben, also keine Abzweigungen, enthält das `nextQuestions` Objekt ein Attribut namens `default` diese wird dann dem Frontend zurückgegeben.

Im Anhang ist eine visualisierte Form der Fragenabfolge des Chatbots in Form eines [Decision Trees](#) zu finden.

4.4 Feststellung der Benutzerberechtigungen

Das unter [Benutzerklassifizierung](#) beschriebene Benutzerkonzept sieht drei Benutzerklassen vor. Es wird davon ausgegangen, dass jeder Benutzer, der sich an der Anwendung bzw. am IdP anmeldet, zunächst ein Regulärer Benutzer ist. Anschließend wird geprüft, ob ein Benutzer über eine Rolle verfügt, die ihm über den IdP zugewiesen wurde.



5 Durchführungsphase

5.1 Auswahl von Libraries

Bei der Auswahl der Libraries wurden einige Aspekte besonders berücksichtigt:

- Lizenz

Die Lizenz der Library muss die Nutzung im konkreten Kontext erlauben. Darunter gelten z.B. die GNU oder MIT Lizenzen

- Beliebtheit

Für die Beliebtheit wurde sich an den Metriken der wöchentlichen Downloads bedient. Je beliebter eine Library ist, desto höher ist die Wahrscheinlichkeit, dass sie langfristig weiter gepflegt wird.

- Dependencies¹

Die Anzahl und Art der Dependencies der Library ist ein nicht unwesentlicher Faktor. Eine Library welche weniger Dependencies beinhaltet, kann besser eingesetzt werden. So kann man besser kontrollieren welcher Code in das Projekt gelangt.

- Schwachstellen

Die Sicherheit der Library ist einer der wichtigsten Punkte. Daher ist es ratsam, eine Library mit weniger Dependencies zu verwenden, da sie weniger Risikopotential bietet. Und darauf zu achten das vorhandene Schwachstellen Risiko arm sind.


5.2 Versionierung

Im Unternehmen wird Git zur Versionsverwaltung eingesetzt. Git ist ein Versionskontrollsystem. Git wurde 2005 veröffentlicht und ist heute de facto Industriestandard. Auch für dieses Projekt wird Git verwendet.

¹engl. Abhängigkeiten, weitere einzubindende Pakete, ohne die die gegebene Library nicht funktioniert




5.3 Aufgabenaufteilung/Aufgabenmanagement

Als Vorgehensmodell wurde das Kanbanboard, da dieses hauptsächlich im Softwareentwicklungsteam verwendet wird. Die Anforderungen wurden in Tasks unterteilt und grob priorisiert, dazu wurde das Online-Webtool [miro](#) zur Visualisierung der Tasks und deren Status verwendet. Ein Auszug des [Kanbanboards](#)  ist im Anhang zu finden.

5.4 Vorbereitung des Applikationsgrundgerüsts

Zunächst wurde das Grundgerüst der Anwendung vorbereitet. Dazu wurde ein [Git-Repository](#) angelegt. Im Repository wurden zwei Hauptverzeichnisse angelegt: Backend und Frontend. Im Backend-Verzeichnis befindet sich das [JavaScript-Projekt](#) der Backend-Applikation, 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](#), wofür ein Template für gängige Datei(-endungen) verwendet wurde, eine [.env-Datei](#) mit zugehöriger [.env.sample](#) Datei zum Bereitstellen von Umgebungsvariablen, wie z.B. [client-id](#) und [secret](#) zur Authentifizierung beim [IdP](#) und die für Docker notwendigen Dateien [docker-compose.yml](#) und [Dockerfile](#).

5.5 Implementierung Backendapplikation

Nachdem das Grundgerüst der Anwendung erstellt war, wurde mit der Implementierung der Backend-Anwendung begonnen. Dabei wurde zunächst die Berechnungslogik der Pauschalen in den Fokus genommen (Ein Ausschnitt einer Funktion ist im Anhang unter [Berechnungs-Ausschnitt](#)  zu finden), da diese Berechnungsgrundlagen bereits in der [MED](#) vorlagen und somit einfach analysiert und in Programmcode umgesetzt werden konnten. Danach wurde mit einer einfachen [REST-API](#) begonnen, die dann im Laufe der Implementierung der [SPA](#) immer wieder an die konkreten Anforderungen angepasst wurde. Mit den entstehenden Endpoints für das Frontend wurde parallel auch die Backend-Logik für die [CRUD-Operationen](#) entwickelt. Bei der Implementierung einer solch umfangreichen Anwendung wird fast immer auf externe Bibliotheken zurückgegriffen.

Auch intern entwickelte DTS-Libraries werden genutzt, wie beispielsweise der [dts-node](#)-



oidc-client für interne Anwendungen, welche den hauseigenen IdP nutzen und der dts-node-logger, ein erweiterter Logger für Node.js-Anwendungen, der speziell für DTS-Produkte verwendet wird.

Im Anhang befindet sich unter [Libraries \(Backend\)](#) eine detaillierte Übersicht über die verwendeten Bibliotheken.

5.6 Implementierung Single-Page-Application

Nach Fertigstellung des Großteils der API im Backend wurde mit der Implementierung der SPA begonnen. Dabei wurden die unternehmensinternen Stylingrichtlinien und die entsprechende Vorlage verwendet, um das Projekt gemäß den Vorgaben zu gestalten. Im Anhang befindet sich unter [Libraries \(Frontend\)](#) eine detaillierte Übersicht über die verwendeten Libraries.

5.7 Implementierung Chatbot

Nachdem nun die CRUD-Services und das Frontend grob vorhanden waren, wurde mit der Implementation der Chatbot Logik begonnen. Das Backend erhält nun Informationen wie im Abschnitt [Chatbot Datenstruktur und Logik](#) beschrieben. Diese Informationen werden dann an die Services gesendet wie z.B. der Service welcher die nächste Frage ermitteln soll (Ein Quellcode Ausschnitt ist unter [Chatbot-Quellcode](#) zu finden). Im Gleichen Schritt speichert das backend die Änderung am Chat. Entscheidet sich der MA die RA freizugeben, wird erstmal der Chat ausgewertet. Mit diesen Ausgewerteten Daten werden dann Berechnungen durchgeführt welche dann in dem Konstrukt der kann nun die RA an mit den vorhanden Daten angereichert werden


5.8 Verbindung zum IdP

Bei jeder Anfrage an das Backend wird mithilfe einer Authentifizierungs Middleware geprüft, ob der mitgesendete JWT valide ist. Falls nicht, wird der Nutzer auf die Login-Seite des IdP geleitet (Ausschnitt davon kann im Anhang unter [IDP-Login](#) eingesehen werden), wo er sich authentifizieren muss. Sobald dies stattgefunden hat, wird der Nutzer zurück zum Travel-Assistant mit einem JWT geleitet. Dieser JWT enthält wichtige



Informationen wie Vor- und Nachname, E-Mail-Adresse und Nutzerrollen. Diese Informationen werden genutzt, um das Frontend entsprechend auszurichten.

5.9 Testing

Nachdem die Berechnungslogik implementiert wurde, muss diese aufgrund des ausdrücklichen Kundenwunsches ausgiebig getestet werden. Hierfür wurde `jest` benutzt. Ein Auszug eines Unittest ist im Anhang unter [Unittest-Ausschnitt](#)  zu finden.

Nachdem eine Lauffähige Version vorhanden war wurde diese mithilfe eines Außen-dienst `MA` auf Funktionalität getestet und um Feedback einzuholen.

6 Abschlussphase

6.1 Soll/Ist-Vergleich

Das Projekt wurde wie geplant umgesetzt. Allerdings konnte der Zeitplan nicht vollständig eingehalten werden:

In der **Projektdefinitionsphase** konnte die eingeplante Zeit eingehalten und sogar verkürzt werden. Die Ist-Analyse fiel aufgrund diverser in der Vergangenheit liegender Berührungspunkte relativ einfach aus.

Die **Planungsphase** des Projekts wurde grundsätzlich eingehalten, jedoch gab es kleinere Schwankungen, die innerhalb dieser Phase durch andere ausgeglichen wurden. Für das Erstellen des Anwendungsfalldiagramms und der Sketche musste weniger Zeit aufgewendet werden, da im schulischen Kontext eine Auffrischung stattfand. Aufgrund von Speicherproblemen während der Entwicklung musste die Datenbank mehrmals angepasst werden.

Die **Projektdurchführungsphase** nimmt den Großteil der Stunden in Anspruch. Diese Phase benötigte mehr Zeit als ursprünglich geplant. Für die Erstellung der Docker-Umgebung und der Grundstruktur der Webanwendung wurde weniger Zeit benötigt, da dies aufgrund der regelmäßigen Anwendung in anderen Projekten schneller umgesetzt werden konnte als ursprünglich geplant. Bei der Berechnungslogik konnte Zeit einge-



spart werden. Der Grund dafür liegt in wegfallenden Berechnungen/Feldern, die nicht mehr benötigt werden. Dadurch ist auch der Aufwand für die Tests dieser kritischen Funktionen geringer geworden. Der Aufwand für die Erstellung und Strukturierung der Fragen wurde jedoch unterschätzt. Um den Überblick über die Abfolge der Fragen zu behalten, musste ein Decision Tree (im Anhang unter [Decision Tree](#) zu finden) erstellt werden. Bei der Implementierung der Mobil-First-Oberflächen hat die Verwendung des Frameworks `React` anfangs Schwierigkeiten bereitet und zusätzliche Recherche erfordert. Auch die Integration des `IdP` hat mehr Zeit in Anspruch genommen, da der Autor bisher nur Server-Side Anwendungen mit dem `IdP` verbunden hat.

In der **Projektabschlussphase** konnte wiederum Zeit eingespart werden, da die Abnahme nur aus einer Präsentation des Tools und einem Ausblick bestand, da die Softwarelösung in der ersten Version noch nicht produktiv einsetzbar ist. Auch bei der internen Dokumentation wurde Zeit eingespart, da parallel zum Erstellen von Funktionen eine entsprechende Dokumentation mithilfe von `JSdoc` erstellt wurde. Der Zeitaufwand für die Erstellung der Projekt-Dokumentation ist höher als geplant gewesen, aufgrund ihres Umfangs.

Eine Übersicht über die geplanten und tatsächlich benötigten Stunden für die einzelnen Phasen und Arbeitspakete ist im Anhang unter [Soll-Ist-Zeitplan](#) zu finden.

6.2 Lessons Learned

Im Laufe des Projekts gab es, wie auch zuvor im Soll/Ist-Vergleich beschrieben, einige Herausforderungen, welche jedoch überwunden werden konnten. Die Erstellung dieser Softwarelösung und die einhergehenden Lerneffekte haben sich als sehr wertvoll erwiesen. So konnten Fähigkeiten im Bereich der Frontendentwicklung deutlich gefestigt und erweitert werden, da der Fokus im sonstigen Arbeitsalltag auf der Backend-Entwicklung liegt. Planung ist sehr wichtig, das hat sich bei dem Erstellen der Softwarelösung deutlich bemerkbar gemacht. Das Projekt verlief ohne große Hürden, da in vielen Phasen auf die Planung der vorangegangenen Phasen zurückgegriffen werden konnte.



6.3 Ausblick

Wie bereits erwähnt, ist die Softwarelösung in seiner jetzigen Form noch nicht produktiv einsetzbar. Es werden noch folgende Funktionalitäten benötigt:

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

7 Glossar

Glossar

API *Application Programming Interface* , 6, 13, 17, 18, 22

Cookie kleine Textdaten, die vom Browser gespeichert und bei HTTP-Requests mitgesendet werden , 39

CORS *Cross-Origin Resource Sharing* , 39

CRUD *Create, Read, Update, Delete* , 17, 18

CSS *Cascading Stylesheets* , 39

Express ein serverseitiges Webframework für die JavaScript-basierte Plattform Node.js, <https://expressjs.com> , 12, 22, 39

GF Geschäftsführung , 7, 8, 9, 11, 32, 33

Git Versionsverwaltungssystem, <https://git-scm.com> , 16, 17

HTML *Hypertext Markup Language*, XML-basierte Beschreibungsnotation für Webumgebungen , 12



HTTP *Hypertext Transfer Protocol* , 13, 21, 39

IdP Identity Provider , 5, 6, 15, 17, 18, 20

JavaScript Skriptsprache, die eine der Haupttechnologien des WWW ist , 12, 17, 22

jest ein JavaScript Testing Framework, <https://jestjs.io/> , 19

Jsdoc eine Auszeichnungssprache, die zum Annotieren von JavaScript-Quelldateien verwendet wird, <https://jsdoc.app/> , 20

JSON *JavaScript Object Notation*, Textbasiertes Datenformat zum Datenaustausch zwischen mehreren Anwendungen , 22

JWT *JSON Web Token*, <https://jwt.io/introduction> , 18

MA Mitarbeiter , 5, 7, 8, 9, 10, 18, 19, 32, 33

MED Microsoft Excel Datei , 4, 5, 7, 8, 9, 17

MERN Ein Technologie-Stack, welcher aus den Komponenten: MongoDB, Express, React und Node.js. besteht , 12

miro Online-Kollaborationsplattform für u.a. Diagramme, Mindmaps und Unterschiedliche Boards, <https://miro.com/de/> , 17

MongoDB Dokumentenbasiertes NoSQL-Datenbanksystem, <https://www.mongodb.com> , 12, 13, 22, 39

Mongoose ODM-Library zur Verbindung mit MongoDB, <https://mongoosejs.com> , 13

MSC *Model, Service, Controller*, Architektur zur Gestaltung von Applikationen , 13

Node.js JavaScript-Laufzeitumgebung, <https://nodejs.com> , 12

ODM *Object Data Modeling* , 22, 39

Promise JavaScript-API zum Abhandeln von asynchron ablaufenden Prozessen. engl. Promise: Versprechen, es wird *await*-ed, dass die Promise eingelöst wird und man so die Daten erhält , 39




RA Reisekostenabrechnung , 4, 5, 7, 8, 9, 10, 11, 13, 14, 18, 32, 33, 34, 48, 49

React Open Source Single Page Application Framework. Umfasst vergleichsweise geringen Funktionsumfang out-of-the-box, ist im Umkehrschluss aber wesentlich flexibler und zum Beispiel im Web und Nativ anwendbar , 12, 17, 20, 22, 39

REST Representational State Transfer , 6, 13, 17

SPA *Single Page Application* , 8, 12, 17, 18, 23

Visual Studio Code Entwicklungsumgebung/Texteditor, <https://code.visualstudio.com> 

vite Buildtool für diverse SPA-Frameworks, <https://vitejs.dev>  , 17, 39

XML *eXtensible Markup Language* , 21

8 Anhang



8.1 Microsoft Excel Datei

Reisezeit				Reiseziel		Verpflegung				Fahrtkosten				Privater PKW				Einzelnachweis						
Reisestag* Datum	An- oder Abreise*	Freitag	Reisezeit		Standortreise (eintrag)	Reiseziel und Anlass (Erläuterung bitte auf extra Blatt)	Land*	Verpflegung				Fahrtkosten				Privater PKW				Einzelnachweis				
			Antritt	Ende				ganztägig	Dauer Std.	Verpfle- gung- pau- schale	Früh- stück ein- schlie- ßen	Früh- stück ein- schlie- ßen	Mittag- essen ein- schlie- ßen	Abend- essen ein- schlie- ßen	Kor- rektur für Mahl- zeiten	Flug- ticket verkehrt	Bahn, Bus, Nah- verkehr	Taxi	km	Anz. Fahr- ten (ge- sam)	Kilo- meter- geld (Privat)	bei Privat-Über- nachtung bitte anreuen	Hotel- geld	Trink- geld
01.03.24							Deutschland																	
02.03.24			08:00	24:00		Reise Athen	- Athen			27,00														27,00
03.03.24			00:00	24:00		Reise Athen	- Athen			40,00														32,00
04.03.24			00:00	24:00		Reise Thessaloniki	Griechenland			36,00														14,40
05.03.24			00:00	24:00		Reise Thessaloniki	Griechenland			36,00														151,00
06.03.24			00:00	07:00		Reise Thessaloniki	Griechenland																	20,00
07.03.24							Deutschland																	
08.03.24			08:00	14:00		Besuch Niederlassung Köln	Deutschland																	16,30
09.03.24							Deutschland																	
10.03.24							Deutschland																	
11.03.24			08:00	17:00		Kunde Aschendorff	Deutschland			14,00														29,00
12.03.24							Österreich																	
13.03.24							Österreich																	
14.03.24							Deutschland																	
15.03.24							Deutschland																	
16.03.24							Deutschland																	
17.03.24							Deutschland																	
18.03.24							Deutschland																	
19.03.24			07:15	18:15		Besichtigung Immobilien Eurotitan HR	Deutschland			14,00														14,00
20.03.24							Deutschland																	
21.03.24							Deutschland																	
22.03.24							Deutschland																	
23.03.24							Deutschland																	
24.03.24							Deutschland																	
25.03.24							Deutschland																	
26.03.24							Deutschland																	
27.03.24							Deutschland																	
28.03.24							Deutschland																	
29.03.24							Deutschland																	
30.03.24							Deutschland																	
31.03.24							Deutschland																	

Ort, Datum:

Herford, 03.04.2024

Unterschrift Mitarbeiter:

Unterschrift Vorgesetzter:

Unterschrift Geschäftsführer:

Erhaltene Vorschüsse:

0,00

Gesamtbetrag:

303,70



8.2 Lastenheft

Zieldefinition

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

Ist Zustand

Mitarbeiter im Außendienst müssen für Dienstreisen eine Reisekostenabrechnung (im Folgenden: RA) erstellen. Diese RA muss bis zum 8. des Folgemonats eingereicht werden. Bisher erstellten die Außendienstmitarbeiter die Reisekostenabrechnung mit Hilfe einer komplexen und sehr unübersichtlichen Microsoft-Excel-Dateivorlage (im Folgenden Excel-Datei genannt). Aufgrund dieser Komplexität und der fehlenden Hilfestellung kommt es vor allem bei den ersten Anwendungen zu vielen Rückfragen in der Accounting Abteilung oder bei anderen Kollegen. In die Excel-Datei, die auch Berechnungen durchführt, müssen verschiedene Angaben zur Dienstreise eingetragen werden. Nachdem die Excel-Datei ausgefüllt wurde, muss diese ausgedruckt und vom Vorgesetzten genehmigt und unterschrieben werden. Wurde die RA genehmigt, muss diese mit Quittungen bzw. Belegen per „Hauspost“ (interner Abhol- und Postdienst am Standort Herford), was im günstigsten Fall einen halben Tag dauert, oder per Post, was 2-3 Tage dauert, an die Buchhaltung geschickt werden. Diese prüfen die RAs und geben sie bei Fehlern an den Außendienstmitarbeiter zur Anpassung zurück. Ist die RA korrekt ausgefüllt & sind die notwendigen Belege & Quittungen vorhanden, wird diese dann an die Payroll Accounting Abteilung weitergeleitet, damit die Reisekosten erstattet werden können.

Soll Zustand

Die Excel-Datei soll durch eine moderne SPA ersetzt werden. Diese SPA soll die Möglichkeit bieten, eine RA digital zu erstellen, freizugeben und zu prüfen. Die Außendienstmitarbeiter sollen mit Hilfe eines regelbasierten Chatbots durch das Ausfüllen einer RA geführt werden. Die Bearbeitung soll jederzeit zu einem anderen Zeitpunkt fortgesetzt werden können. Sobald alle notwendigen Angaben gemacht wurden, kann der Außendienstmitarbeiter die RA freigeben. Diese muss dann zunächst vom jeweiligen Teamleiter/Vorgesetzten über die SPA bestätigt werden, bevor sie der Accounting Abteilung zur Prüfung vorgelegt wird. Die SPA sollte dann die Berechnungen der RA aufschlüsseln, damit die Buchhaltung diese besser nachvollziehen/prüfen kann. Werden Unstimmig-



keiten/Unvollständigkeiten festgestellt, so wird die RA mit dem entsprechenden Vermerk zur Bearbeitung an den Außendienstmitarbeiter zurückgegeben und nach Anpassung wieder an die Accounting Abteilung freigegeben. Sobald die RA fehlerfrei ist, kann die Accounting Abteilung eine Auszahlung veranlassen lassen.

Funktionale Anforderungen

1. Datenbank + Anbindung zur Speicherung
 - (a) Vollständige/ Unvollständige RA Daten speichern können
 - (b) RA Daten speichern können
 - (c) Belege (Bilder) speichern können
2. Erstellung einer RA
 - (a) Erstellung durch ein Regelbasierenden Chatbot
 - (b) Möglichkeit bieten zu gewissen „fragen“ zusätzlich belege hochzuladen
3. Übersicht der RAs
 - (a) Nutzer können alle selbst erstellten RAs einsehen
 - (b) Anzeige des jeweiligen Status der RAs
4. Übersicht Freigabe der RAs
 - (a) Vorgesetzte haben eine Übersicht über eingereichte RAs für ihre zuständigen Mitarbeiter
5. Freigabe/Ablehnung von RAs durch Vorgesetzten
6. Übersicht Prüfung der RAs
 - (a) Prüfer können alle vom Nutzer & Vorgesetzten Freigegebenen RAs einsehen
7. Prüfung der RAs
 - (a) Prüfer haben eine detaillierte Ansicht der RA
 - (b) Prüfer haben eine Aufschlüsselung der Berechnungen einer RA
 - (c) Prüfer haben die Möglichkeit Vermerke zu schreiben



(d) Prüfer haben die Möglichkeit die RA freizugeben oder zurückzusenden

8. Statische Implementierung von Pauschalen

(a) Kilometer Pauschale

(b) Länder Pauschalen

i. Pauschalbeträge 8-24 Std. oder (An/Abreisetag)

ii. Pauschalbeträge Ganztags

iii. Pauschalbeträge Privatübernachtung

(c) Prozentuale Abzüge für erhaltene Mahlzeiten

9. Berechnung von Pauschalen

(a) Pauschalen/Werte sollen im Hintergrunde anhand der Benutzer Angaben berechnet werden

(b) Nachvollziehbarkeit soll für Prüfer gegeben sein

10. Login, Rechte und Benutzer Verwaltung

(a) Integration des Hauseigenen Identity Providers (IDP)

(b) Benutzer & Rechte werden über den hauseigenen IDP verwaltet

Nicht Funktionale Anforderungen

1. Benutzung des Corporate Designs

2. Sprache für die Oberflächen: Deutsch

3. Erstellung von RAs soll durch Fragestellungen vom Regelbasierenden Chatbot auch für neu Einsteiger selbsterklärend sein

4. Zwischen Speicherung

(a) Von RAs

(b) Von Prüfungen der RAs

5. Intuitive Benutzung

Rollen



- Regulärer Benutzer – Mitarbeiter im Außendienst
- Prüfer – Mitarbeiter in der Accounting Abteilung
- Vorgesetzter - Übergeordneter Mitarbeiter des Außendienst Mitarbeiters

Status

- Offen – noch nicht zur Prüfung/Freigabe gesendete RA
- In Bearbeitung – RA ist zur Bearbeitung/Freigabe bei der Accounting Abteilung/ Vorgesetzten
- Fehlerhaft – RA wurde mit einem Vermerk zurück gegeben
- Fertig – Wurde erfolgreich abgearbeitet

Abnahme- und Testkriterien

1. Reibungslose Erstellung und Verwaltung von RAs sowie die Prüfung soll gegeben sein
2. Erfüllt alle Funktionalen Anforderungen
3. Ist auf Server Lauffähig
4. Code abschnitte, welche Berechnungen anhand von Pauschelen durchführen müssen ausführlich getestet und auf Richtigkeit geprüft werden



8.3 Verwendete Ressourcen

1. Hardware

- Betrieblich bereitgestellter Bildschirmarbeitsplatz (2x Monitor, Tastatur, Maus, Dockingstation)
- Betrieblich bereitgestelltes Notebook

2. Software

- Windows 11
- Visual Studio Code
- Git
- Docker
- Node.js
- Windows Terminal
- Firefox Developer Edition
- Draw.io
- Texmaker

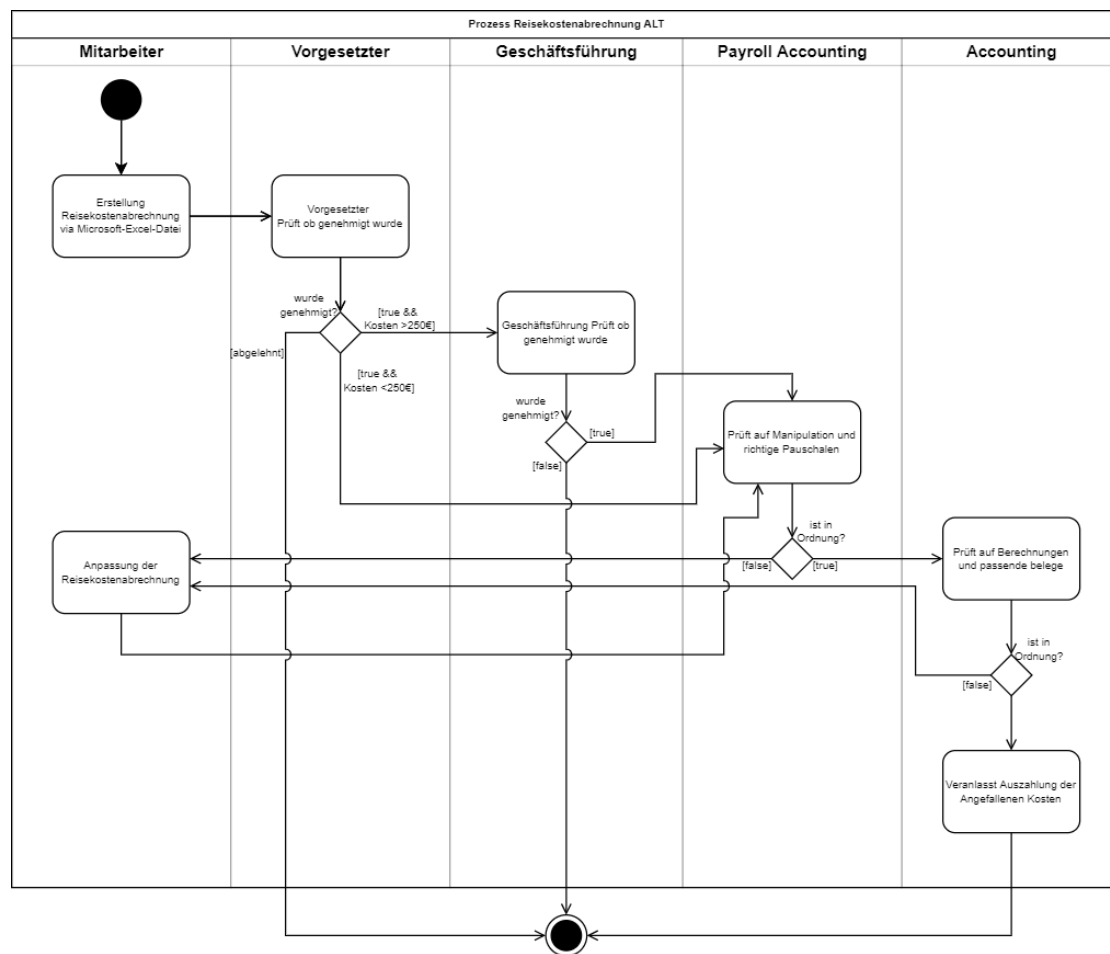


8.4 Soll-/Ist-Zeitplanung

Bezeichnung	geplant	tatsächlich
Projektdefinitionsphase	4	3,5
<ul style="list-style-type: none"> Durchführung Ist-Analyse Durchführung Amortisationsrechnung Erstellung eines Lastenhefts (Unterstützung des Fachbereichs) 	1 1 2	0,5 1 2
Projektplanungsphase	10	10
<ul style="list-style-type: none"> Erstellung eines Anwendungsfalldiagramm Erstellung eines Aktivitätsdiagramm Erstellung eines Pflichtenheftes Datenbank Modellierung Erstellung von Sketches Einteilung von Arbeitspaketen (Kanban Board) 	1 1 2 3 2 1	0,5 1 2 4 1,5 1
Projektdurchführungsphase	50	51,5
<ul style="list-style-type: none"> Container Umgebung erstellen Grundstruktur der Webanwendung Implementieren IDP-Integration Chatfragen für den Regelbasierenden Chatbot formulieren & strukturieren Implementierung der Backend-Logik <ul style="list-style-type: none"> 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 <ul style="list-style-type: none"> Mobile-First Oberfläche für CRUD-Operation der Reisekostenabrechnung Tabellenansicht zur Überprüfung der Reisekostenabrechnung Qualitätssicherung und Tests (der Kritischen Berechnungsfunktionen) 	1 5 1 4 7 6 4 8 8 6	0,5 4 2 6 5 8 4 12 8 4
Projektabschlussphase	16	15
<ul style="list-style-type: none"> Abnahme Interne Dokumentation Projektdokumentation 	2 3 11	1 2 12
Gesamtstundenzahl für die Projektarbeit:	80	80



8.5 Prozess Alt





8.6 Amortisationsrechnung

Berechnung der monatlichen Kosten durch die aktuelle Lösung:

RA§ erstellen:

$$20RA§ \cdot \frac{15\text{minuten}}{60\text{minuten}} \cdot 25€/h = \mathbf{125€}$$

Einarbeitung MA im Außendienst:

$$0,5 \cdot 2MA \cdot 25€/h = \mathbf{25€}$$

Aushelfen:

$$20\% \cdot 20RA§ \cdot 2MA \cdot \frac{15\text{minuten}}{60\text{minuten}} \cdot 25€/h = \mathbf{50€}$$

Genehmigung Vorgesetzter:

$$20RA§ \cdot \frac{5\text{minuten}}{60\text{minuten}} \cdot 45€/h = \mathbf{75€}$$

Genehmigung GF:

$$1RA§ \cdot \frac{5\text{minuten}}{60\text{minuten}} \cdot 100€/h = \mathbf{8,33€}$$

Hauspost:

$$90\% \cdot 20RA§ \cdot \frac{10\text{minuten}}{60\text{minuten}} \cdot 7€/h = \mathbf{21€}$$

Selbst gebracht:

$$10\% \cdot 20RA§ \cdot \frac{10\text{minuten}}{60\text{minuten}} \cdot 25€/h = \mathbf{8,33€}$$

Prüfung Payroll Accounting:

$$20RA§ \cdot \frac{10\text{minuten}}{60\text{minuten}} \cdot 30€/h = \mathbf{100€}$$

Prüfung Accounting:

$$20RA§ \cdot \frac{12\text{minuten}}{60\text{minuten}} \cdot 30€/h = \mathbf{120€}$$

Nachbearbeitung:

$$25\% \cdot 20RA§ \cdot \frac{10\text{minuten}}{60\text{minuten}} \cdot 25€/h = \mathbf{20,83€}$$



Nachprüfung:

$$25\% \cdot 20RA_s \cdot \frac{7\text{minuten}}{60\text{minuten}} \cdot 30\text{€}/h = \mathbf{17,50\text{€}}$$

Rückfragen Accounting:

$$10\% \cdot 20RA_s \cdot \frac{10\text{minuten}}{60\text{minuten}} \cdot (25\text{€}/h + 30\text{€}/h) = \mathbf{18,33\text{€}}$$

Summe der monatlichen Kosten durch die aktuelle Lösung:

$$125\text{€} + 25\text{€} + 50\text{€} + 75\text{€} + 8,33\text{€} + 21\text{€} + 8,33\text{€} + 100\text{€} + 120\text{€} + 20,83\text{€} + 17,50\text{€} + 18,33\text{€} = \mathbf{589,33\text{€}}$$

Berechnung der monatlichen Kosten durch die neuentwickelte Softwarelösung: RA_s erstellen:

$$20RA_s \cdot \frac{20\text{minuten}}{60\text{minuten}} \cdot 25\text{€}/h = \mathbf{166,67\text{€}}$$

Aushelfen:

$$10\% \cdot 20RA_s \cdot 2MA \cdot \frac{5\text{minuten}}{60\text{minuten}} \cdot 25\text{€}/h = \mathbf{50\text{€}}$$

Genehmigung Vorgesetzter:

$$20RA_s \cdot \frac{5\text{minuten}}{60\text{minuten}} \cdot 45\text{€}/h = \mathbf{75\text{€}}$$

Genehmigung GF:

$$1RA_s \cdot \frac{5\text{minuten}}{60\text{minuten}} \cdot 100\text{€}/h = \mathbf{8,33\text{€}}$$

Prüfung Accounting:

$$20RA_s \cdot \frac{7\text{minuten}}{60\text{minuten}} \cdot 30\text{€}/h = \mathbf{70\text{€}}$$

Nachbearbeitung:

$$10\% \cdot 20RA_s \cdot \frac{7\text{minuten}}{60\text{minuten}} \cdot 25\text{€}/h = \mathbf{5,83\text{€}}$$

Nachprüfung:

$$10\% \cdot 20RA_s \cdot \frac{5\text{minuten}}{60\text{minuten}} \cdot 30\text{€}/h = \mathbf{5\text{€}}$$



Rückfragen Accounting:

$$10\% \cdot 20 \text{RAs} \cdot \frac{10 \text{minuten}}{60 \text{minuten}} \cdot (25\text{€}/h + 30\text{€}/h) = \mathbf{18,33\text{€}}$$

Wartung:

$$2 \text{Std.} \cdot 27\text{€}/h = \mathbf{54\text{€}}$$

Summe der monatlichen Kosten durch die neuentwickelte Softwarelösung:

$$166,67\text{€} + 50\text{€} + 75\text{€} + 8,33\text{€} + 70\text{€} + 5,83\text{€} + 5\text{€} + 18,33\text{€} + 54\text{€} = \mathbf{411,50\text{€}}$$

Berechnung Entwicklungskosten des Travel-Assistant: Implementierung:

$$80 \text{Stunden} \cdot 7\text{€}/h = \mathbf{560\text{€}}$$

Hilfestellung/Rat:

$$5 \text{Stunden} \cdot 30\text{€}/h = \mathbf{150\text{€}}$$

Meetings:

$$6 \text{Stunden} \cdot 2 \cdot 30\text{€}/h = \mathbf{360\text{€}}$$

Feedback/Testing:

$$1 \text{Stunden} \cdot 25\text{€}/h = \mathbf{25\text{€}}$$

Summe Projekt Entwicklungskosten:

$$560\text{€} + 150\text{€} + 360\text{€} + 25\text{€} = \mathbf{1.095\text{€}}$$

Berechnung Weiterentwicklungskosten: Weiterentwicklung:

$$25 \text{Stunden} \cdot 27\text{€}/h = \mathbf{675\text{€}}$$

Zusätzliche Meetings:

$$4 \text{Stunden} \cdot (27\text{€}/h + (2 \cdot 30\text{€}/h)) = \mathbf{348\text{€}}$$



Summe Weiterentwicklungskosten:

$$675\text{€} + 348\text{€} = \mathbf{1.023\text{€}}$$

Summe Entwicklungskosten bis zum Vollumfänglichen einsatz:

$$1.095\text{€} + 1.023\text{€} = \mathbf{2.118\text{€}}$$

Berechnung des Break-even-Points:

$$\frac{2.118\text{€}}{589,33\text{€} - 411,50\text{€}} \approx \mathbf{11,91} \Rightarrow \mathbf{12}$$

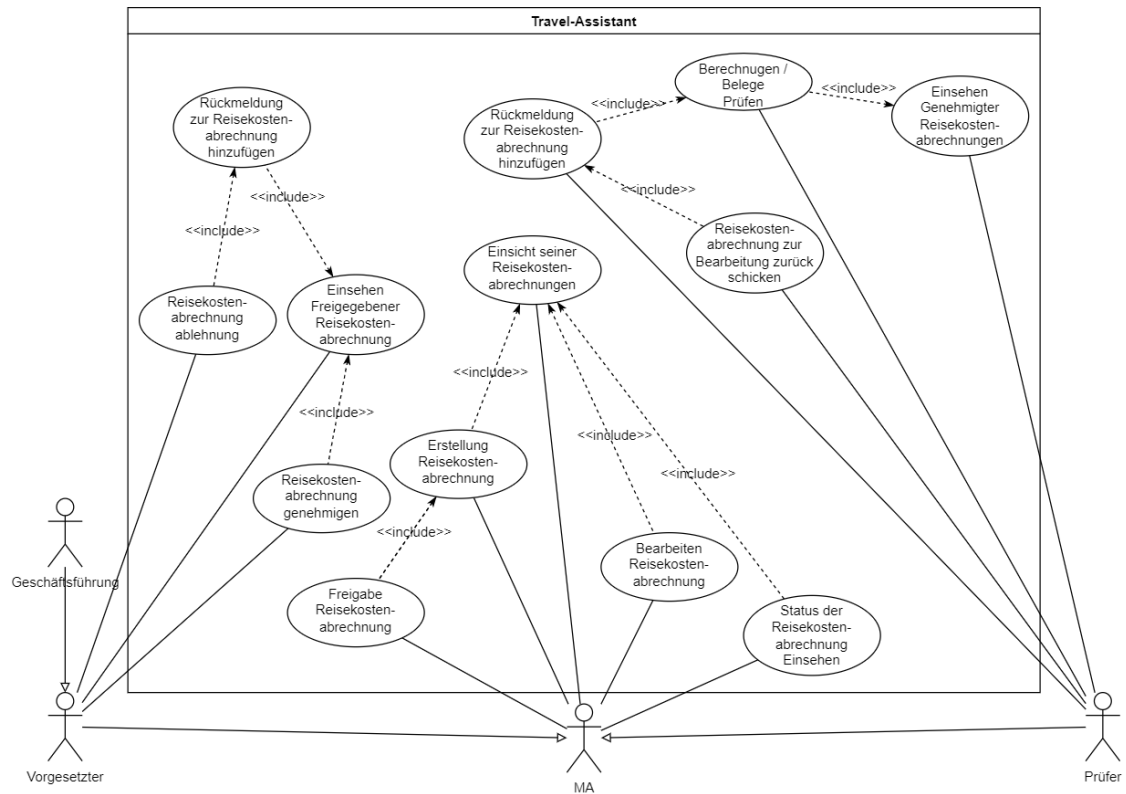
Monatliche Kosteneinsparungen:

$$589,33\text{€} - 411,50\text{€} = \mathbf{177,83\text{€}}$$

Nach etwa 12 Monaten haben sich die Entwicklungskosten durch die eingesparten Kosten amortisiert. Ab diesem Zeitpunkt spart die neu entwickelte Softwarelösung 177,83 € pro Monat.




8.7 Anwendungsfalldiagramm






8.8 IdP-Login



welcome

Melden Sie sich bei DTS an, um fortzufahren



Geben Sie Ihr Passwort ein

Geben Sie Ihr Passwort für DTS ein, um weiterzumachen

[Ändern](#)



8.9 Kanban-Board

Backlog | 5 ...

Funktion zum Upload von Belegen

View zur Prüfung der Abrechnung

View zur Freigabe von Vorgesetztem

IDP Integration

Funktion zur Freigabe von

In Bearbeitung | 1

View zur erstellung von
Abrechnungen (Chatbot)

+

Fertig | 13

View zur übersicht der erstellten
Abrechnungen

Chatbot Fragen erstellen &
strukturieren

Datenbank anbindung

Erstellung von Sketches



8.10 Libraries

8.10.1 Libraries (Backend)

dependencies	
Library	Kurzbeschreibung
axios	Promise-basierte HTTP-Client-Library
cors	Middleware um CORS einzustellen
body-parser	Middleware für Express, um Request-Bodies zu lesen
cookie-parser	Middleware für Express, um Cookies zu lesen
express	Middleware-basierte HTTP-Server-Library
mongoose	ODM für MongoDB
dts-node-logger	Eine DTS Interner Logger für Node Applikationen
dts-node-oidc-client	Eine DTS Interne Node OIDC Client Library
eslint & addons	Linter und Codeformatter
jest	Testing Framework
nodemon	library für automatischen Neustart bei Änderungen

Tabelle 1: Libraries (Backend)

8.10.2 Libraries (Frontend)

dependencies	
Library	Kurzbeschreibung
axios	Promise-basierte HTTP-Client-Library
bootstrap	CSS-Framework
bootstrap-icons	Iconset von dem gleichen Team wie bootstrap
react	siehe React
react-bootstrap	React-Components für bootstrap
react-bootstrap-icons	React-Components für bootstrap-icons
react-dom	React-Addon für den Browser
react-router-dom	Routing-Funktionalität für React-Applikationen im Browser
react-multi-date-picker	React datepicker component
eslint & addons	Linter und Codeformatter
vite	siehe vite

Tabelle 2: Libraries (Frontend)



8.11 Code-Ausschnitte

8.11.1 Berechnungs-Ausschnitt

8.11.2 Unittest-Ausschnitt

```
describe('Calculates meal deduction', function () {
  describe('Positive tests', function () {
    test('Checks that the correct meal deduction is returned', async () => {
      try {
        expect(await CalculateReportService.calculateMealDeduction(36, true, true, false)).toEqual(21.6);
        expect(await CalculateReportService.calculateMealDeduction(40, true, false, false)).toEqual(8);
        expect(await CalculateReportService.calculateMealDeduction(40, false, false, false)).toEqual(0);
        expect(await CalculateReportService.calculateMealDeduction(36, true, true, true)).toEqual(36);
        expect(await CalculateReportService.calculateMealDeduction(0, true, true, true)).toEqual(0);
      } catch (error) {
        expect(error).toBeUndefined();
      }
    });
  });
  describe('Negative tests', function () {
    test('Checks that the correct error is thrown', async () => {
      try {
        expect(await CalculateReportService.calculateMealDeduction()).toThrow('mealAllowance is undefined');
        expect(await CalculateReportService.calculateMealDeduction(undefined, false, true, false))
          .toThrow('mealAllowance is undefined');
      } catch (error) {
        expect(error).toBeDefined();
      }
    });
  });
});
```

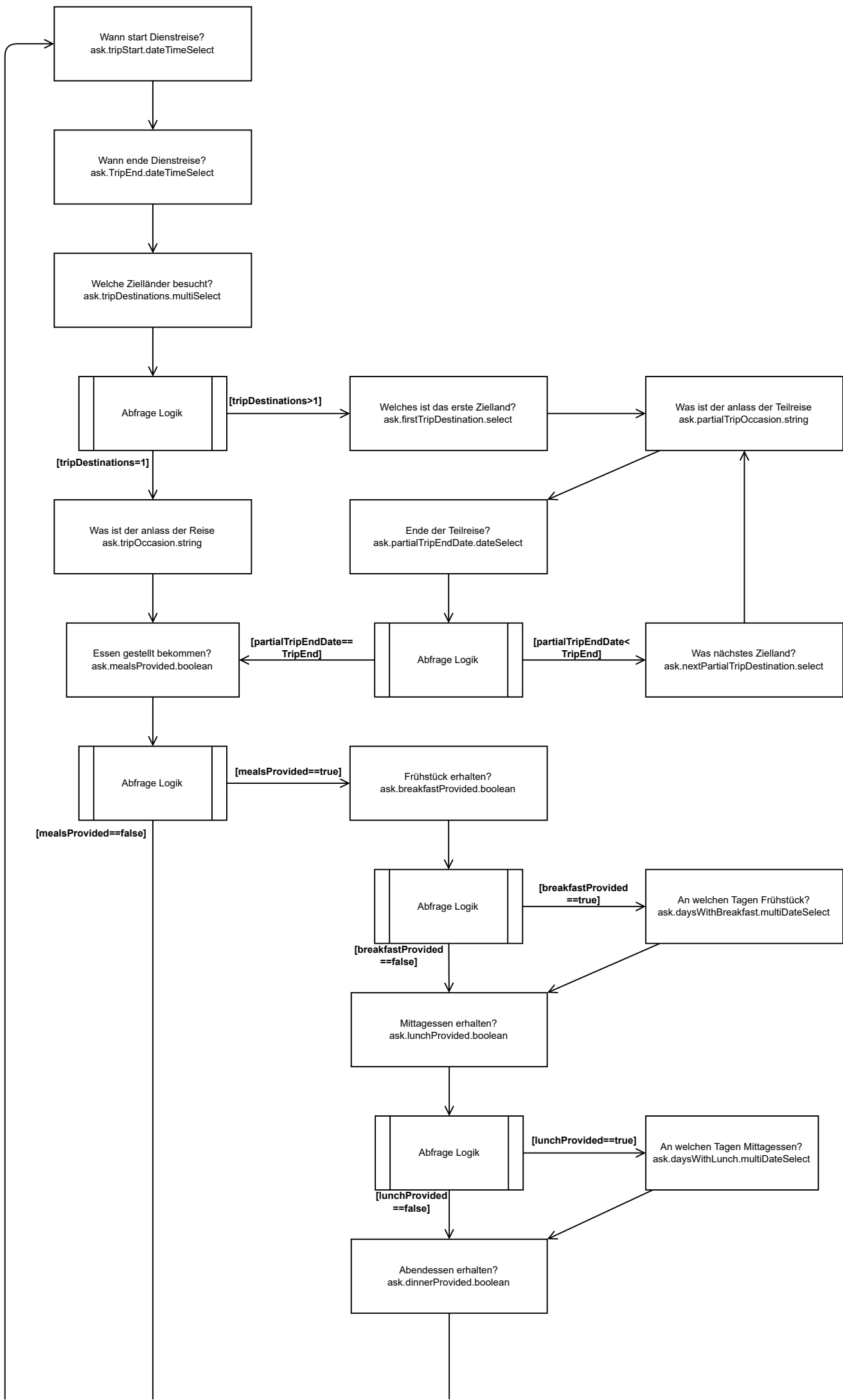


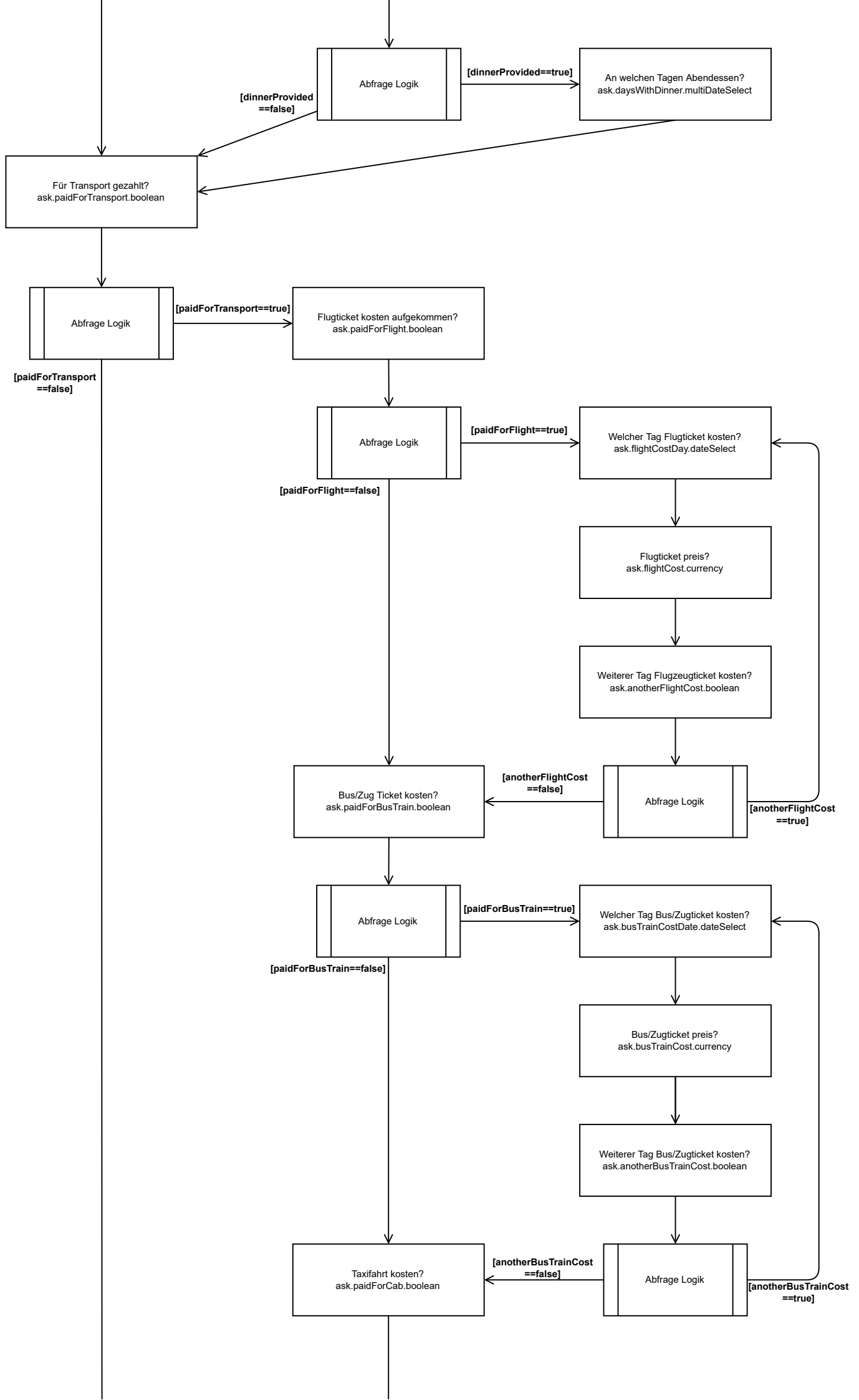

8.11.3 Chatbot-Ausschnitt

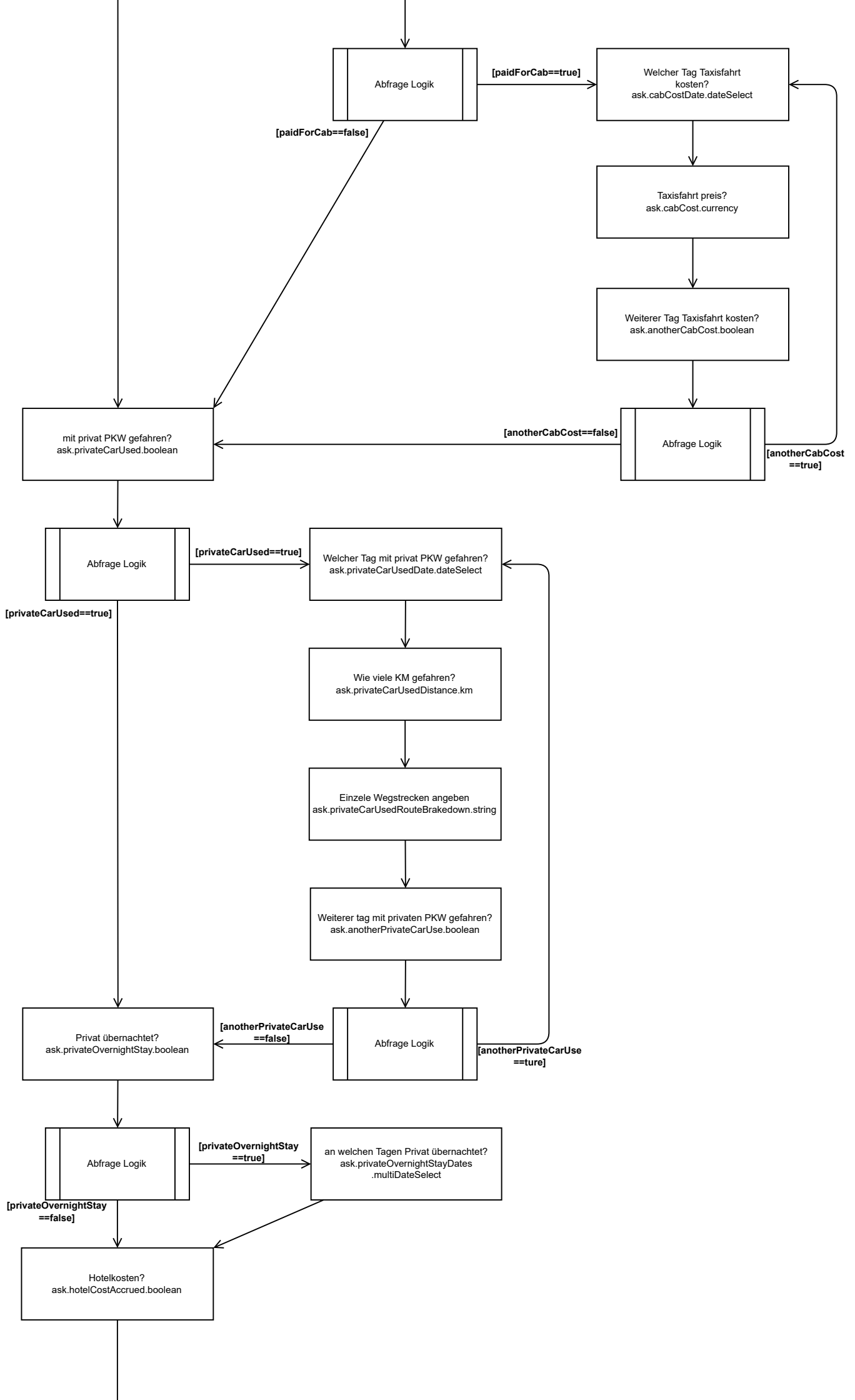
```
/**
 * @description this function gets the follow-up question
 * @param {Object} question the origin question object from the frontend
 * @param {String} answer the user answer to the origin question
 * @returns the follow-up question
 */
async function getFollowUpQuestion(question, answer, travelExpenseReport){
  if (typeof (question.condition) === 'undefined'){
    return await QuestionService.getOneQuestionById(question.nextQuestion.default);
  }
  const conditionResult = getResultOfCondition(question.condition, answer, travelExpenseReport);
  if(conditionResult){
    return await QuestionService.getOneQuestionById(question.nextQuestion.true)
  }
  return await QuestionService.getOneQuestionById(question.nextQuestion.false);
}

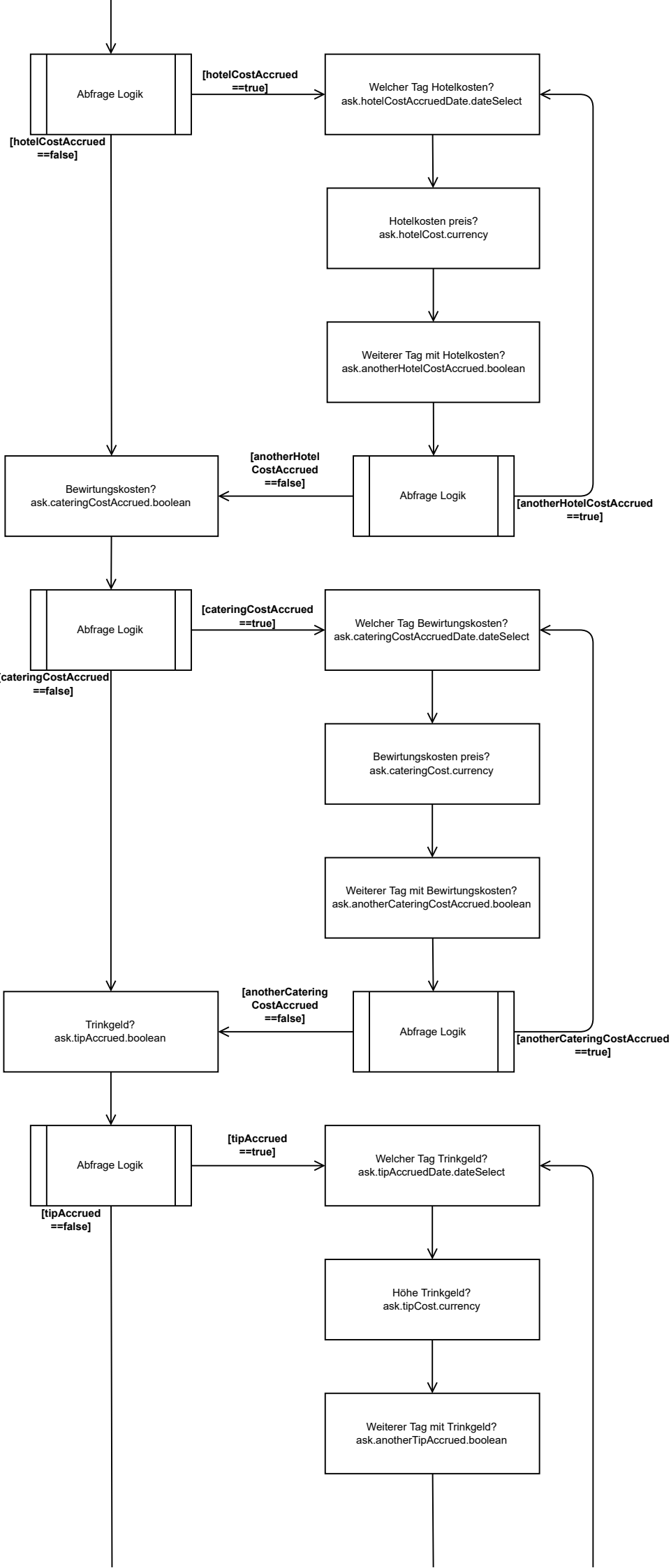
/**
 * @description this function is getting
 * @param {String} condition the condition for the if-statement
 * @param {String} answer the user answer to the origin question
 * @param {Object} travelExpenseReport the existing travel expense report
 * @returns the result of the condition
 */
async function getResultOfCondition(condition, answer, travelExpenseReport){
  switch (condition) {
    case 'truthy':
      return (answer === true) ? true : false;
    case 'gt1':
      return (answer.length > 1) ? true : false;
    case 'isTripEndDate':
      return (Date.parse(answer) === travelExpenseReport.tripEndDate) ? true : false;
    default:
      throw new Error('no matching condition found')
  }
}
```

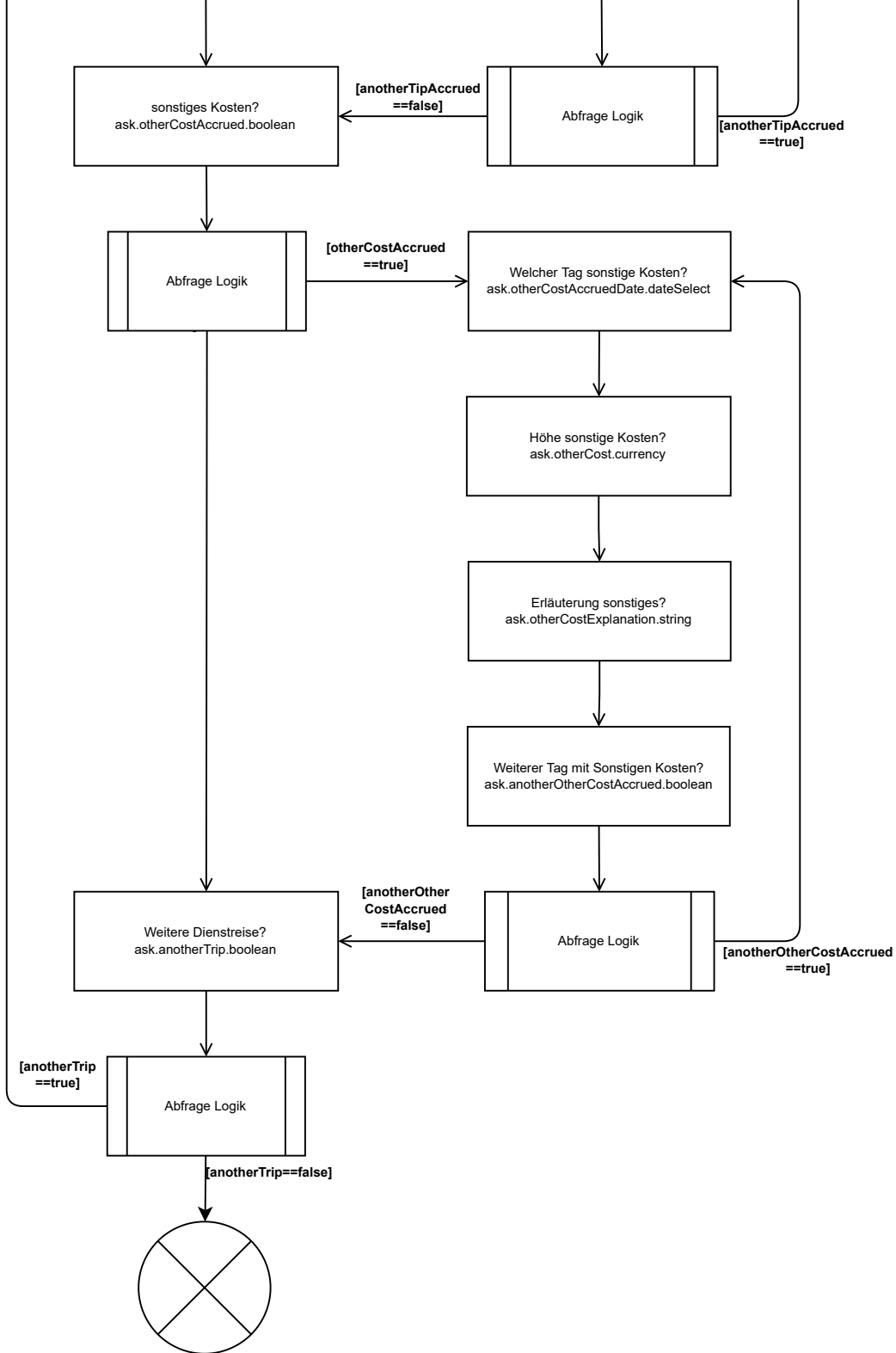
8.12 Decision Tree















8.13 Sketches

