TAU INF202 Software Engineering

Individuelles Projekt

Pflichtenheft

Projektdokumentation

Version: 2023.1.0

Status: freigegeben

Schwarze Texte sind INVARIANT (zu behalten)!
Die blauen und roten Texte sind zu ersetzen oder zu entfernen!

ReservAPP

Verantwortliche/r:

Teoman Turan, e180503040@stud.tau.edu.tr

Barış Emre Yaşar, e180503038@stud.tau.edu.tr

Stakeholder: DI. Ömer Karacan, omer.karacan@tau.edu.tr

Dokumentenverwaltung

Dokument-Historie

Version	Status *)	Datum	Verantwortlicher	Änderungsgrund
v0.1	entwurf	07.04.2023	Ö. Karacan	erste Entwurf fürs Projekt
v1.0	freigegeben	16.04.2023	Ö. Karacan	erste stabile Version fürs Projekt

**) Sofern im Projekt nicht anders vereinbart, sind folgende Statusbezeichnungen zu verwenden (in obiger Tabelle und am Deckblatt):*
Dokument-Status: Entwurf / in Review / freigegeben (abgegeben)

Dokument wurde mit folgenden Tools erstellt:
Microsoft Office Word
Lucid App

Inhaltsverzeichnis

1. Einleitung

Dieses Dokument dient dazu die für die Produktrealisierung verbindlich gestellten Anforderungen zu definieren, vollständig und konsistent zu beschreiben.

Alle Anforderungen (versehen mit eindeutigen IDs) an das System werden aus der Sicht vom Auftraggeber (Kunden, Anwender, Stakeholder) dokumentiert.

In diesem Kapitel wird die Bedeutung dieses Dokuments hervorgehoben und die einzelnen Kapitel werden kurz vorgestellt, damit der Leser schnell einen Überblick über den Inhalt gewinnen kann.

Dieses Dokument dient dazu, die fürs Prototyping von Reservapp verbindlich gestellten Anforderungen zu definieren und sie vollständig und konsistent zu beschreiben.

Die Use Cases und Anforderungen sind aus der Sicht des Stakeholders beschrieben.

Die graphische Oberfläche zur Überwachung des Reservapp ist aus der Sicht eines Autofahrers beschrieben.

Im Kapitel 2 „Ausgangssituation und Ziele“ werden die Ausgangssituation und der Grund zur Wahl von ReservAPP anschaulich dargestellt.

Im Kapitel 3 „Gesamtarchitektur“ sind die physikalische und die konzeptionelle Architektur des Systems, und die wichtigsten Subsysteme (Komponenten), die Anwender und die notwendigen Kommunikationsschnittstellen dargestellt. Hier sind auch zusätzliche Anforderungen an die Architektur oder Komponenten zu finden.

Im Kapitel 4 „Funktionale Anforderungen“ beinhaltet die Beschreibung der funktionalen Anforderungen durch die Ablaufbeschreibungen (User Stories), die Anwendungsfällen (Use Cases), und die technischen und fachlichen Anforderungen (Requirements). Alle betriebsrelevanten Daten werden durch die Datenmodellen definiert.

Im Kapitel 5 „Nichtfunktionale Anforderungen“ werden die funktionalen Anforderungen durch diejenigen Anforderungen erweitert, die keine funktionalen Anforderungen sind.

Im Kapitel 6 „Abnahmekriterien“ sind die Abgabeartefakten festgelegt, die ohne Abstimmung des Stakeholders nicht zu manipulieren sind.

Im Kapitel 7 „Projekt Meilensteine“ sind die wichtigsten Termine aufgelistet, die den Fortschritt der Teilergebnisse des Projektes definieren.

Im Kapitel 8 „Referenzen“ sind die wichtigsten Referenzen aufgelistet.

2. Ausgangssituation und Ziele

In diesem Kapitel werden die Ausgangssituation und der Grund zur Wahl des Projektthemas anschaulich dargestellt.

Problemstellung (Funktionalität)
Es wird beschrieben, welches Problem (Funktionalität) durch den Einsatz des neuen Systems gelöst wird.

Wir brauchen eine Lösung für unsere Kunden, die von uns eine Reservierung bekommen wollen. In dieser Phase sollte unser Hauptziel sein, sicherzustellen, dass sie keine Schwierigkeiten oder Zeitverschwendung erleben. Kunden können auch die Bewertungen von den Hotels sehen und dann besser entscheiden, welche sie auswählen.

Stakeholder (Anwender):
Es wird beschrieben, welche Zielgruppen durch den Einsatz des neuen Systems am meisten profitieren.

Sowohl Hotels als auch Kunden, die in den Urlaub fahren möchten, werden damit am meisten profitieren. Sie werden keine physische Interaktion brauchen.

Ziele (Lösung)
Es wird beschrieben, welche qualitative und quantitative Ziele durch die Lösung (durch das neue System) erreicht werden.

Hotel Management Webseiten fehlen viele Anwendungen. Beispielsweise kann man ihre Urlaube nicht bewerten. Man kann nicht seiner eigenen Hotelraum & Urlaub Option Punkte geben. Oder man kann nicht an das schöne Andenken, was man im gewählten Hotel gelebt hat.. Wir haben eine Website entwickelt, um dieses Problem zu lösen. Auf dieser Webseite kann man sowohl den Hotelraum reservieren und seine Erinnerungen schreiben im Blog.

3. Gesamtarchitektur

Um die Visualisierung der Anwenderanforderungen zu ermöglichen, soll Gesamtsystemarchitektur illustriert werden, die die Sichtweise des Anwenders repräsentiert und nicht die technische Sichtweise des Systemanalytikers beziehungsweise des Systemarchitekten.

Es soll eine konzeptionelle Architektur unter der Berücksichtigung der Kommunikation (Interaktionen) mit den externen Systemen erstellt werden (Anm.: Vergleich mit einer technischen Systemarchitektur).

Darüber hinaus, in der Gesamtsystemarchitektur sollten die zukünftigen Systembestandteile (Komponente) identifiziert und festgeschrieben werden.

Einleitung
Eine kurze Beschreibung, was dieses Kapitel beinhaltet.

In diesem Kapitel werden unsere Systemgrenzen definiert. Die notwendigen Datenstrukturen sind definiert und modelliert. ReservAPP ist eine webbasierte Anwendung, die die Verwaltung von Reservierungen für Hotels vereinfacht. Die Gesamtarchitektur von ReservAPP besteht aus verschiedenen Komponenten, die eng miteinander verbunden sind und zusammenarbeiten, um eine nahtlose Erfahrung für Benutzer und Unternehmen zu gewährleisten.

Gesamtarchitektur
Die Illustration der konzeptionellen Architektur durch die funktionellen Blöcken in free-style oder in UML-Diagrammen soll erstellt werden, wobei die wichtigste Komponente und Schnittstellen hervorgehoben sind.

Die Gesamtarchitektur von Reservapp basiert auf einer Client-Server-Architektur. Es gibt drei Hauptkomponenten in der Architektur: den Client, den Webserver und die Datenbank.

Komponente <x>
Die hervorgehobenen Komponenten und Komponentenschnittstellen sollen hier beschrieben werden.

Client-Komponente:
Die Client-Komponente ist die Benutzeroberfläche von Reservapp, die Benutzern ermöglicht, Reservierungen zu suchen, zu erstellen und zu verwalten. Der Client ist eine Single-Page-Anwendung.

Webserver-Komponente:
Der Webserver ist die Komponente, die die Geschäftslogik von Reservapp enthält. Der Webserver ist in C# geschrieben und verwendet ASP.NET. Es bietet eine REST-API-Schnittstelle für die Kommunikation mit dem Client und enthält die Kernfunktionen von Reservapp, wie z.B. die Verwaltung von Reservierungen, Benutzern.

Datenbank-Komponente:
Die Datenbank ist die Komponente, die die persistenten Daten von Reservapp speichert. Es ist eine relationale Datenbank, die in SQL geschrieben ist. Die Datenbank enthält Tabellen für Benutzer, Reservierungen und andere relevante Daten.

Externe Schnittstellen
Die hervorgehobenen Systemschnittstellen sollen hier beschrieben werden.

Wir brauchen keine externen Schnittstellen.

4. Funktionale Anforderungen

Funktionale Anforderungen beschreiben die Gesamtfunktionalität eines Systems, die die Zielgruppe erwartet, um mit Hilfe des Systems ein Problem zu lösen.

Die Anforderungen werden aus Ablaufbeschreibungen (User Stories) zur Nutzung des Systems abgeleitet. Aus diesem Grund, die Beschreibung der funktionalen Anforderungen erfolgt durch die Anwendungsfällen (Use Cases). Ein Anwendungsfall beschreibt die Interaktion zwischen dem Anwender (oder einem externen System) und dem zu realisierenden System abläuft.
Die Gesamtheit der Anwendungsfälle definiert das Systemverhalten und hilft enorm ein System aus der Sicht eines Anwenders zu definieren.

Die technischen und fachlichen Vorgaben werden als einzelnen Anforderungen definiert und den Use Cases zugeordnet. Solche Anforderungen sollen kurz, prägnant und direkt (active voice) formuliert werden

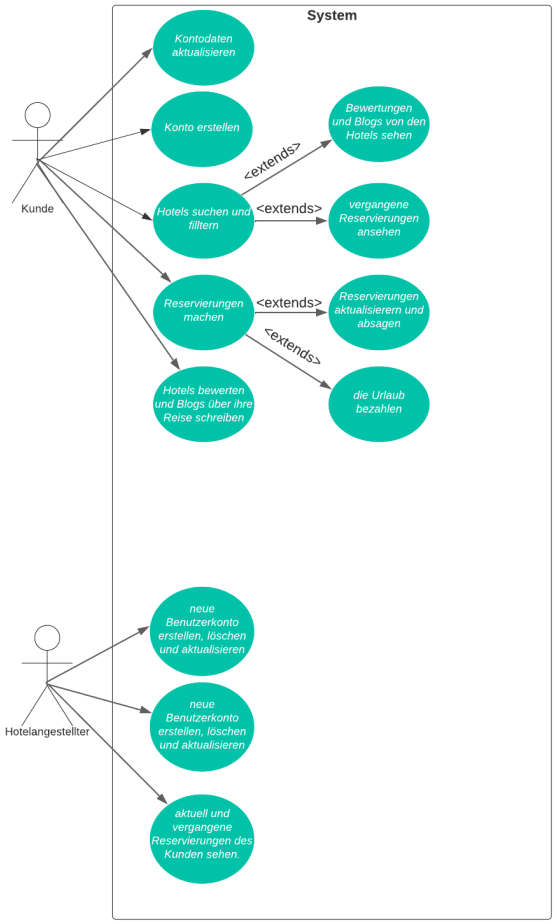
Darüber hinaus, im Rahmen der funktionalen Anforderungen wird ein erstes Datenmodell erstellt, das sowohl ein fachliches, aber auch schnittstellenspezifisches Datenmodell beinhaltet. Das letztere ist besonders wichtig, wenn die Systemfunktionalität auch als API zur Verfügung gestellt werden soll.

Einleitung

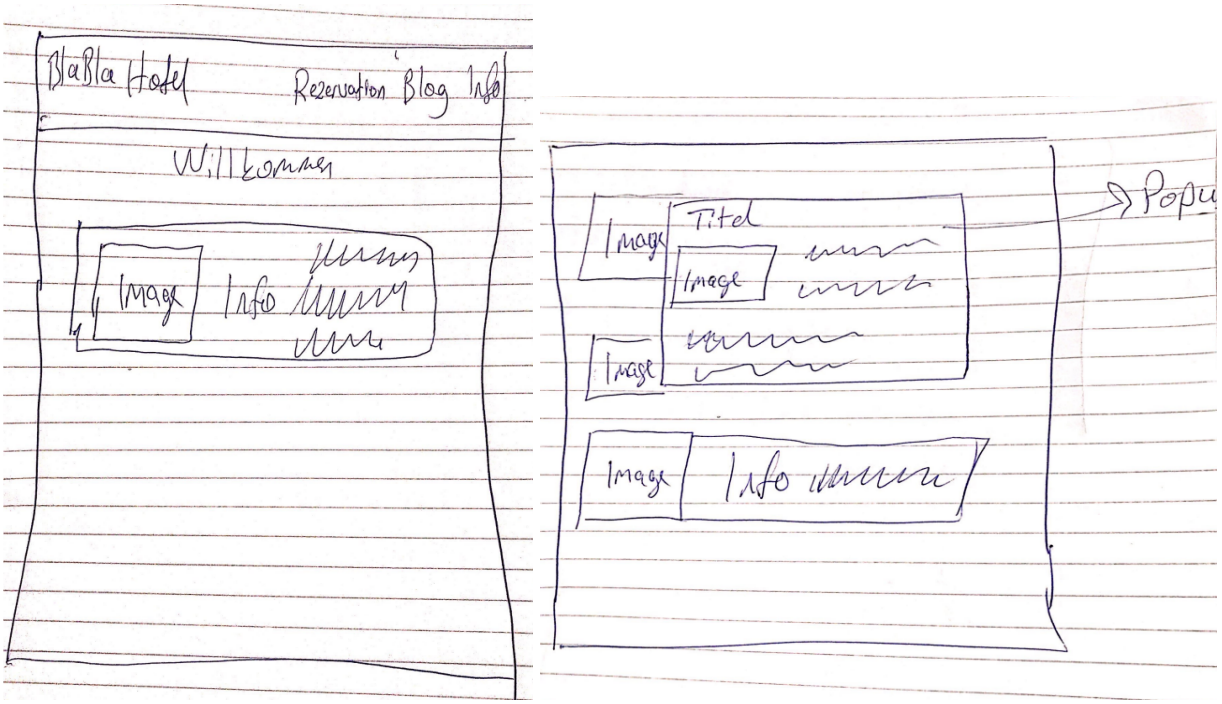
Eine kurze Beschreibung, was dieses Kapitel beinhaltet.

In diesem Kapitel sind Anforderungen (inklusive User Stories und Use Cases) an das Gesamtsystem, aber auch an die einzelnen Systemkomponenten definiert.

Use Cases



GUI Entwurf



Technischen und fachliche Anforderungen

Die einzelnen Anforderungen, die in Use Cases nicht explizit erfasst werden können, sollen kurz, prägnant und direkt (active voice) formuliert werden. Diese Anforderungen sind für den Anwender transparent wie softwaretechnische Vorgaben.

Anmerkung: Alternativ zur API-Schnittstelle kann eine CLI-Schnittstelle realisiert werden.

5. Nichtfunktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben Anforderungen an das System, die nicht-fachlicher Natur sind, z.B. Qualitätsanforderungen, Sicherheitsanforderungen oder Performanceanforderungen. Solche Anforderungen sollen im Architekturentwurf berücksichtigt werden.

Im Allgemeinen werden diejenigen Anforderungen, die keine funktionalen Anforderungen sind, als nicht-funktionalen Anforderungen deklarieret.

Nicht-funktionale Anforderungen sollen messbar beschrieben sein.

Einleitung
Eine kurze Beschreibung, was dieses Kapitel beinhaltet.

In diesem Kapitel sind die nicht-funktionalen Anforderungen ans Gesamtsystem aber auch an die einzelnen Systemkomponenten definiert. Es wird besonders auf die Software Qualität Wert gelegt (Testing).

Nicht-funktionale Anforderungen an die Systemarchitektur (Architekturmuster, Deployment)
Die nicht-funktionalen Anforderungen an die Systemarchitektur sollen hier aufgelistet sein.

Die Kommunikation der Systemkomponenten untereinander unterliegt der Beschränkungen der REST-Architektur und die Interaktionen den RESTful Webservices.

Nicht-funktionale Anforderungen an die Entwicklungsumgebung
Die nicht-funktionalen Anforderungen an die Entwicklungsumgebung sollen hier aufgelistet sein.

Die Entwicklungsumgebung ist frei wählbar

Nicht-funktionale Anforderungen an die Entwicklungswerkzeuge (Sprache, IDE, Frameworks)
Die nicht-funktionalen Anforderungen an die Entwicklungswerkzeuge sollen hier aufgelistet sein.

HTML5: HTML5 ist eine Auszeichnungssprache, die zum Strukturieren und Präsentieren von Inhalten im World Wide Web verwendet wird. Es ist die fünfte und letzte große HTML-Version, die eine Empfehlung des World Wide Web Consortium (W3C) ist. Die aktuelle Spezifikation ist als HTML Living Standard bekannt.

CSS: Cascading Style Sheets (CSS) ist eine Stylesheet-Sprache, die zum Beschreiben der Präsentation eines Dokuments verwendet wird, das in einer Auszeichnungssprache wie HTML geschrieben wurde.

Bootstrap: Bootstrap ist ein kostenloses Open-Source-CSS-Framework, das auf die responsive Frontend-Webentwicklung ausgerichtet ist. Es enthält CSS- und (optional) JavaScript-basierte Designvorlagen für Typografie, Formulare, Schaltflächen, Navigation und andere Schnittstellen Komponenten.

Javascript: JavaScript ist eine dynamische Programmiersprache, die in Webbrowsern weit verbreitet ist. Dank der in JavaScript geschriebenen clientseitigen Skripts werden Funktionen wie die Interaktion mit dem Benutzer, die Steuerung des Browsers, die asynchrone Kommunikation mit dem Server und das Ändern des Inhalts der Webseite bereitgestellt. Auch serverseitig ist JavaScript dank Plattformen wie Node.js weit verbreitet.

ASP.NET: ASP.NET ist ein serverseitiges Open-Source-Webanwendungs-Framework, das für die Webentwicklung entwickelt wurde, um dynamische Webseiten zu erstellen. Es wurde von Microsoft

entwickelt, um Programmierern die Erstellung dynamischer Websites, Anwendungen und Dienste zu ermöglichen.

Nicht-funktionale Anforderungen an die Teststrategie (Qualitätssicherung)
Die nicht-funktionalen Anforderungen an die Teststrategie sollen hier beschrieben sein.

Das Hauptziel dieses Testprozesses ist es, ReservAPP mit all ihren Aspekten zu evaluieren und vor dem ersten Deployment zu kontrollieren. System Tests werden sowohl RESTful Service und Frontend benutzt. Testverfahren wird manuell durchgeführt. Test-Faelle werden von Anwendungsfälle und Features abgeleitet und detailliert geschrieben.

Abnahmekriterien

Die Abnahmekriterien sind durch den Stakeholder definiert und sie dürfen nur mit Zustimmung des Stakeholders neu definiert, geändert oder erweitert werden.

Das Projekt wird mit den folgenden Artefakten abgegeben:

- Dokumentation:
  - Pflichtenheft: *ReservAPP_Pflichtenheft.docx*
- Software
  - Link zu GitHub Projekt: *git-hub-link*
- Evidenz:
  - System/Software-Demo via Videoclip: *videoclip-link*

Anm.: Die Abgabetermine der Projetartefakts werden durch den Stakeholder festgelegt!

6. Projekt Meilensteine

Diese Kapitel beinhaltet die wichtigsten Meilensteine, die der Stakeholder festgelegt hat.

Folgende Meilensteine sind verbindlich definiert:
Meilenstein #1: Das Lastenheft ist fertiggestellt und mit dem Stakeholder abgestimmt.
Meilenstein #2: Das Pflichtenheft ist fertiggestellt und mit dem Stakeholder abgestimmt.
Meilenstein #3: (An diesem Meilenstein ist die Architektur im Vordergrund.)

- Komponenten-basierte Architektur ist entworfen und komponentenweise implementiert.
- Die externen Schnittstellen (Webservices) wurden entworfen/implementiert.
- Die GUI Komponente ist ansatzweise fertig.

Meilenstein #4: (An diesem Meilenstein ist das Testen im Vordergrund.)

- Die Test-Cases wurden aus den Use Cases abgeleitet und ansatzweise beschrieben.
- Die Testumgebung ist vorbereitet und ein Smoke Test ist durchgeführt.

Meilenstein #5

- Das Projekt ist per Vereinbarung abgegeben.

7. Referenzen

Diese Kapitel beinhaltet die wichtigsten Literaturquellen aufgelistet.