TAU INF202 Software Engineering

Individuelles Projekt

Pflichtenheft

Projektdokumentation

Version: 2023.v1.0

Status: In Review

Parkplatzverwaltung

Verantwortliche/r:

Altay Yavuz Elmas, e200503046@stud.tau.edu.tr

Ahmet Efe Kutbay, e200503019@stud.tau.edu.tr

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

Template: Projektdokumentation.doc / Version vom 29.09.2020

Dokumentenverwaltung

Dokument-Historie

Version	Status *)	Datum	Verantwortlicher	Änderungsgrund
v0.1	Entwurf	30.03.2023	A. Yavuz Elmas A. Efe Kutbay	Erstellung des ersten Entwurfs
v0.5	Entwurf	05.04.2023	A. Yavuz Elmas A. Efe Kutbay	Weiterentwicklung des ersten Entwurfs
v1.0	in Review	14.04.2023	A. Yavuz Elmas A. Efe Kutbay	Fertigung der Pflichtenheft

**) 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
Google Docs
Lucidchart

Inhaltsverzeichnis

1. Einleitung

Dieses Dokument dient dazu, die fürs Prototyping eines Parkvorgangs 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 Parkvorgangs ist aus der Sicht eines Parkaufsehers beschrieben

Im Kapitel 2 „ Ausgangssituation und Ziele “ sind die Ausgangssituation und der Grund zur Wahl vom Parkplatz 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 technischen und fachlichen Anforderungen (Requirements). Alle betriebsrelevanten Daten werden durch die Datenmodellen definiert.

Im Kapitel 5 „ Nichtfunktionale Anforderungen “ sind 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.

Ausgangssituation und Ziele

Einleitung

Das ausgewählte Thema ist aus dem Bereich von Parkplatzmanagement, wo die Verwaltung des Parkplatzes durch die Integration der Informationstechnologie ein höheres „Intelligenzniveau“ erlangt.
Eine dieser intelligenten, funktionalen Einheiten ist die sogenannte Software “Parkplatzverwaltung”.

Problemstellung (Funktionalität)

Die Anzahl und Bedeutung von Parkplatzgeschäften nimmt mit der zunehmenden Nutzung von Fahrzeugen zu. An dieser Stelle ist die Erleichterung des Managements dieser Unternehmen sehr wichtig für den Komfort der Verwaltung und hochwertigen Service.

Zu diesem Zweck entwickelte Software "Parkplatzverwaltung" soll für die Parkplatzbetreibern das Verwalten erleichtern.

Stakeholder (Anwender):

Die Verwaltung des Parkplatzes bedient dieses System am meisten und die Qualität des Services wird sich erhöhen. Dadurch werden die Kunden dieses Betriebs davon profitieren.

Systemumfeld (Einsatzumgebung)

Das ganze System mit Software implementiert. Der Benutzer muss das System über einen Computer bedienen.

Rahmenbedingung (Einschränkungen)

Die wichtigsten Einschränkungen befinden sich in der Wahl der Software Tools.

- Als Software-Entwicklungstool soll entweder Eclipse oder IntelliJ verwendet werden,
- die Backend Applikationen sollen mit Java Framework und Spring Framework,
- die Frontend Applikationen mit JavaFX Rich Client Technologie realisiert werden, und
- die persistenten Daten sollen in einer SQL-Datenbank abgespeichert werden.

Ziele (Lösung)

Mit der in den vorherigen Abschnitten genannten Motivation sollte ein System vorbereitet werden, das Funktionen für den angestrebten Zweck enthält. Auf Funktionen muss von der zu präparierenden Schnittstelle aus zugegriffen werden.

Die Persistenz soll mit einem Datenbanksystem erfolgen.

2. Gesamtarchitektur

Einleitung

In diesem Kapitel sind die Systemgrenzen definiert, die mit adäquaten Abbildungen präzisiert. Unter Berücksichtigung der Systemgrenzen sind per Anwender die externen Schnittstellen definiert (graphische Schnittstellen und API). Die notwendigen Systemkomponenten und Datenstrukturen sind definiert und modelliert.

Dieses Kapitel führt zu dem besseren Verständnis des angeforderten Systems und dadurch genaue Definition der funktionalen und nicht-funktionalen Anforderungen.

Gesamtarchitektur

PAU (Parking Automation Unit) enthält die Grundfunktionen des Systems und fungiert als eine Brücke zwischen der graphischen Schnittstelle und der Datenbank. Über die graphische Schnittstelle wird auf die PAU zugegriffen und die von der PAU angebotenen Dienste können genutzt werden.

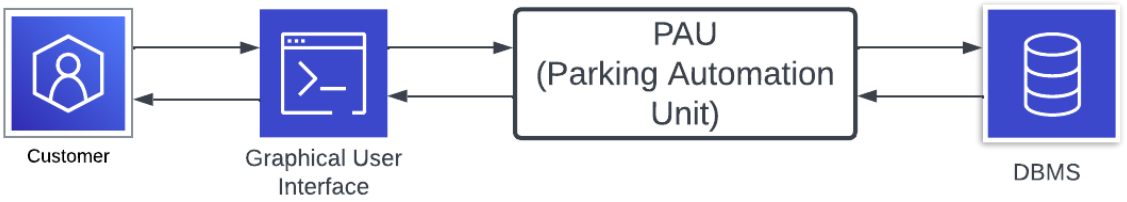

Abbildung 1: Gesamtsystemarchitektur

Die Komponentenarchitektur soll nach dem Architekturmuster entworfen werden.

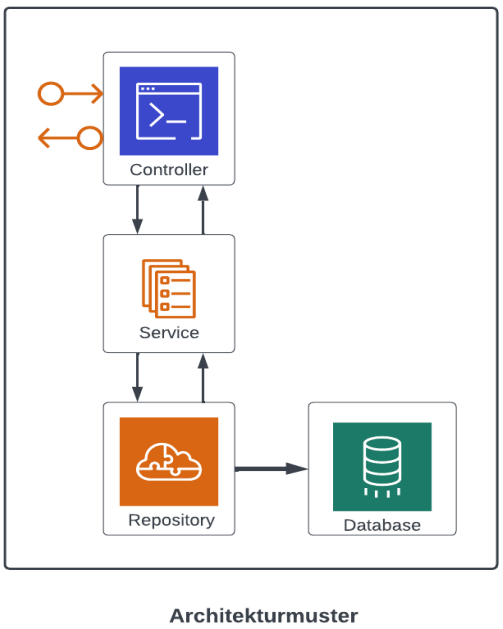

Abbildung 2: Architekturmuster

Komponente PAU (Parking Automation Unit)

Die PAU bietet dem Benutzer alle Informationen über den Parkplatz. Darüber hinaus stellt es die Dienste zur Verfügung, damit der Benutzer Parkplatzverwaltungsaufgaben einfach ausführen kann.

Von PAU angebotene Dienste:

- Parkplatz hinzufügen
- Parkplatz löschen
- Parkplatzdetails verändern
- Einparken
- Ausparken
- Tickets anzeigen
- Parkplätze anzeigen
- Fahrzeuge anzeigen

3. Funktionale Anforderungen

Einleitung

In diesem Kapitel sind Anforderungen (inklusive User Stories und Use Cases) an das Gesamtsystem definiert.

PAU (Parking Automation Unit) Use Cases

- **/PAU-1/** Durch die Eingabe von notwendigen Informationen wie Fahrzeugtyp, Nummernschild und Slot ID können die Fahrzeuge, die auf dem Parkplatz kommen, im System erfasst werden.

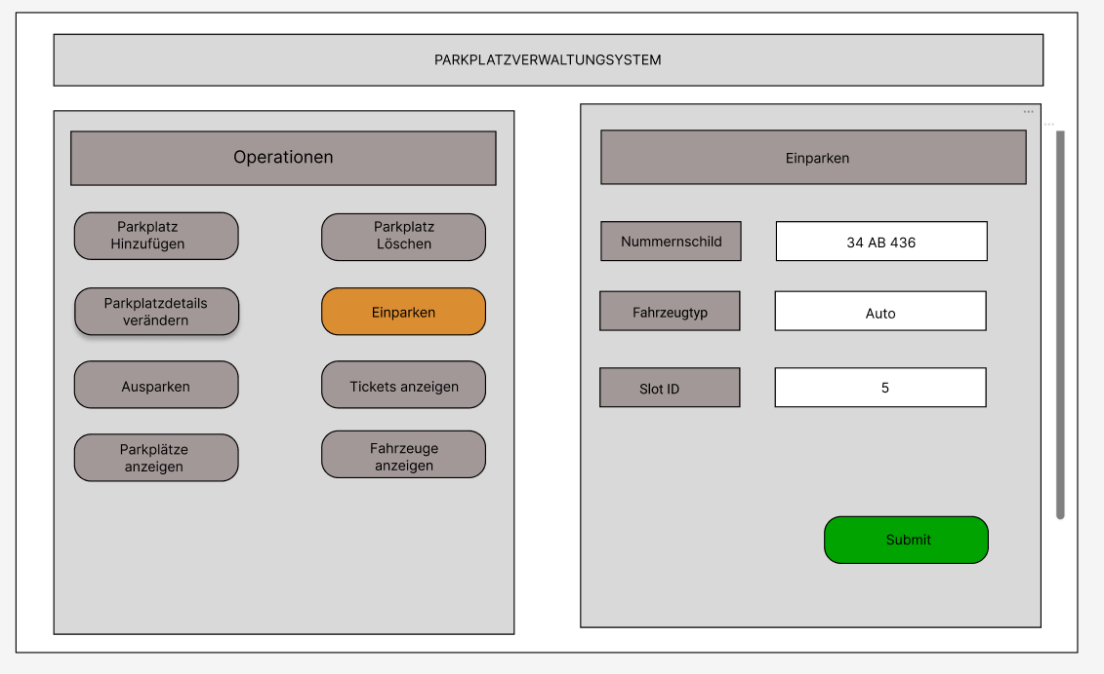

Abbildung 3: Einparken

- **/PAU-2/** Durch die Eingabe von notwendigen Informationen wie Parkplatz-ID, Name, Adresse und Anzahl der Parklücken können neue Parkplätze erstellt und im System erfasst werden. Für die Adresse müssen Straße, Stadt, Postleitzahl und Land eingegeben werden.

- **/PAU-3/** Durch PAU können die Fahrzeuge, die den Parkplatz verlassen, aus dem System durch den Button entfernt werden.

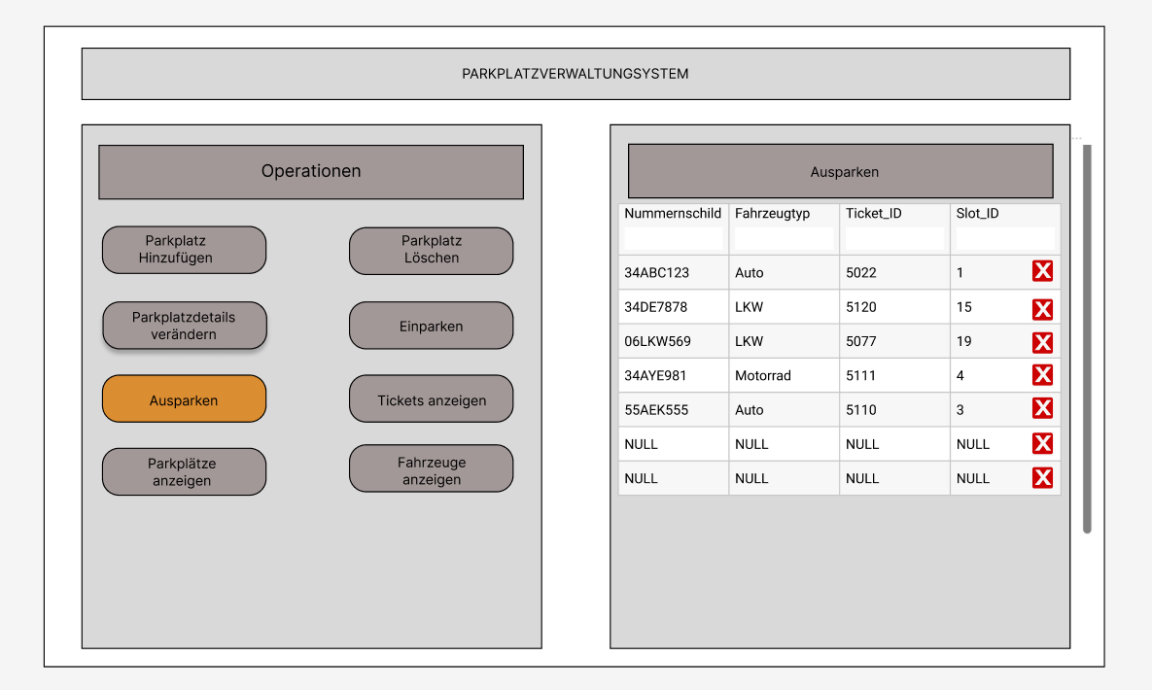

Abbildung 4: Ausparken

- **/PAU-4/** Durch PAU können die Parkplätze aus dem System durch den Button entfernt werden.

- **/PAU-5/** Durch PAU können die Details von Parkplätzen aus dem System mit notwendigen Informationen, wie Parkplatz-ID, Name, Slot-ID und Adresse verändert werden. Parkplatz-ID ist für die Suche von Parkplätze benötigt und kann nicht verändert werden.

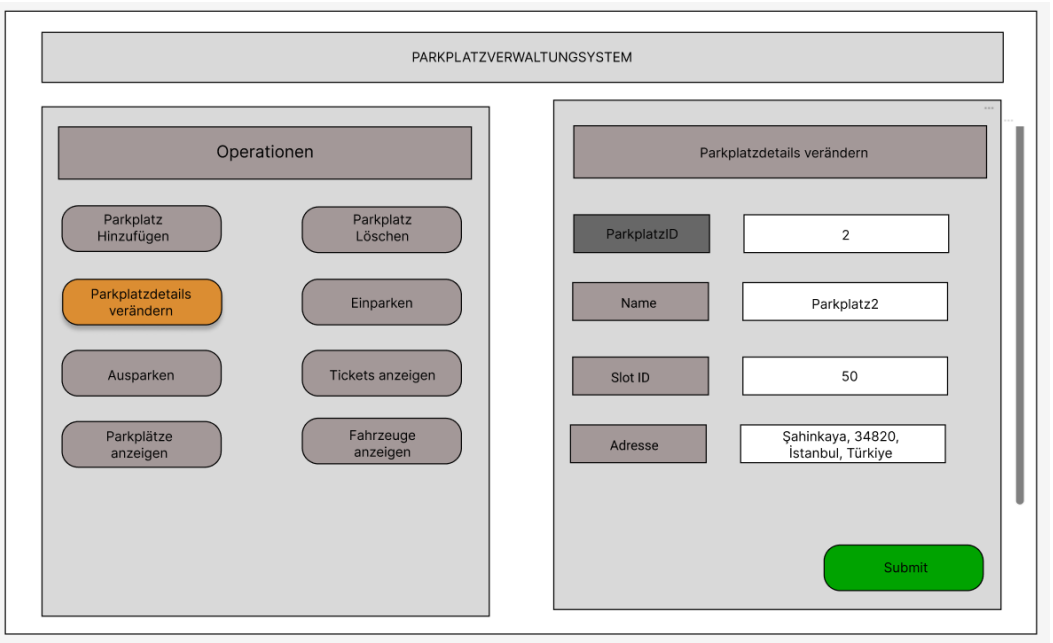

Abbildung 5: Parkplatzdetails verändern

- **/PAU-6/** Durch PAU können die Informationen von Parkplätzen angezeigt werden.

- **/PAU-7/** Durch PAU können die Informationen von Tickets angezeigt werden.

- **/PAU-8/** Durch PAU können die Informationen von Fahrzeugen angezeigt werden.

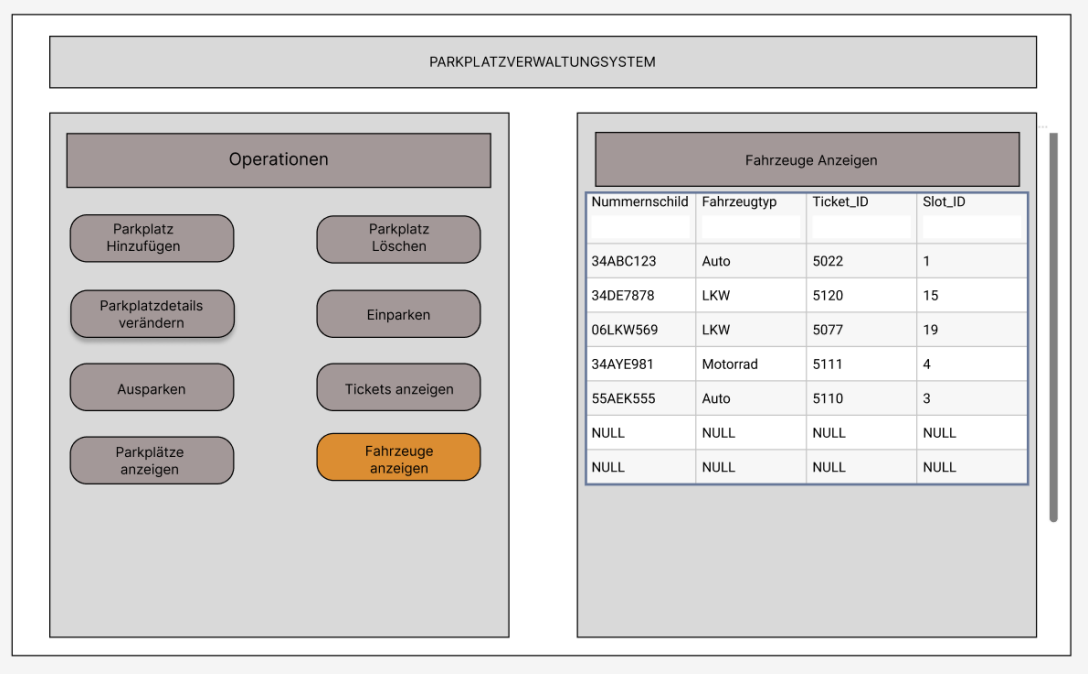

Abbildung 6: Fahrzeuge anzeigen

API Use Cases

Ein API Use Case fungiert als eindeutige RESTful-Schnittstelle, die eine Anforderung vom Client an den Server weiterleitet. Die API stellt die Systemfunktionalität bereit und dadurch den Zugriff auf die Geschäftsdaten durch ein RESTful Service Interface (API-Gateway).

- /API-1/ Die Schnittstelle ist als RESTful Web Services definiert.

RESTful Web Services	Use Cases
Post	Parkplatz hinzufügen
Delete	Parkplatz löschen
Put	Parkplatzdetails verändern
Post	Einparken
Delete, Put	Ausparken
Get	Tickets anzeigen
Get	Parkplätze anzeigen
Get	Fahrzeuge anzeigen
Post	Ticket erstellen
Post	Parklücke erstellen

Abbildung 7: API-Schnittstelle

Technischen und fachliche Anforderungen

- /SYS-1/ Das Parkplatzverwaltungssystem soll eine einheitliche Architektur aufweisen. Die Komponente PAU ist die zentrale und einzige Komponente.
- /SYS-2/ PAU soll mit DBMS-Komponente kommunizieren.
- /SYS 3/ Der Kunde steuert das System über PAU, die zentrale Komponente ist.

Datenmodell

- /DAT-1/ Die relevanten Systemparameter sind in einer externen Datenbank abzuspeichern. Die Liste der Parameter wird später durch den Stakeholder zur Verfügung gestellt.
- /DAT-2/ Ein externes DBMS (eine SQL-Datenbank) ist für die Persistenz der Daten
- notwendig. Es kann ein Entity/Relationship Diagramm erstellt werden.
- /DAT-3/ Das DBMS soll zum Speichern der Daten per Komponente gewählt und
- eingerichtet sein.
- /DAT-4/ Das Datenmodell soll die Parameter der API-Schnittstellen berücksichtigen.
- /DAT-5/ Das Datenmodell soll im UML Klassendiagramm modelliert werden.

Die Daten und Objekte, die von dem System verarbeitet werden, müssen in einem bestimmten Format vorliegen, und dieses Format wird im UML-Diagramm angezeigt.

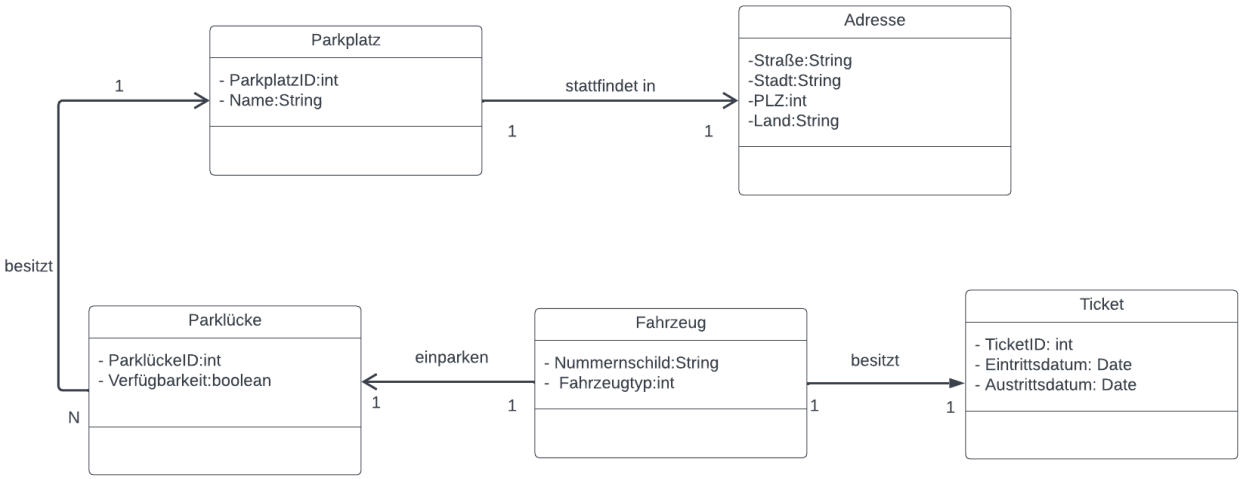

Abbildung 8: UML-Diagramm von den Datenstrukturen des Systems

4. Nichtfunktionale Anforderungen

Einleitung

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)

- /SYS-1/ Die Kommunikation der Systemkomponenten untereinander unterliegt der Beschränkungen der REST-Architektur und die Interaktionen den RESTful Webservices.

Nicht-funktionale Anforderungen an die Entwicklungsumgebung

- /DEV-1/ Die Entwicklungsumgebung ist IntelliJ IDEA.

Nicht-funktionale Anforderungen an die Entwicklungswerkzeuge (Sprache, IDE, Frameworks)

- /TOL-1/ Die Backend-Applikationen sollen mit Spring Boot Framework implementiert werden.
- /TOL-2/ Die Frontend Applikation soll mit JavaFX Rich Client Technologie realisiert werden.
- /TOL-3/ Die persistenten Daten sollen in einer SQL-Datenbank abgespeichert werden.

Nicht-funktionale Anforderungen an die Teststrategie (Qualitätssicherung)

- /TEST-1/ Die PAU Komponente kann getrennt vom System getestet werden.
- /TEST-2/ Alle Use Cases, User Stories und Anforderungen sollen getestet und berichtet werden
- /TEST-3/ Jeder System-Service soll Black-Box getestet werden.

5. 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: *INF202-Parkplatzverwaltung-Pflichtenheft-2023.v1.0.docx*
- Software
  - Link zu GitHub Projekt:
    https://github.com/orgs/Gruppe1-Fulya/teams/a-quadrat-time-complexity
- Evidenz:
  - System/Software-Demo via Videoclip:

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

6. Projekt Meilensteine

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