TAU INF202 Software Engineering

Individuelles Projekt

Pflichtenheft

Projektdokumentation

Version: 2023.v0.1

Status: Meilensteine 2

Autohaus Projekt

Verantwortliche/r: Ömer Karacan, omer.karacan@tau.edu.tr

Berater: Berkan Turkoglu, e190501030@stud.tau.edu.tr

Berater: Melih Eyuboglu, e190503020@stud.tau.edu.tr

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

Dokumentenverwaltung

Dokument-Historie

Version	Status *)	Datum	Verantwortlicher	Änderungsgrund
v1.0	freigegeben	19.09.2022	Ö. Karacan	Vorlage wurde für die Studentenprojekte freigegeben

**) 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

Inhaltverzeichnis

1. Einleitung

In diesem Dokument sind die Definitionen der Anforderungen an ein lokales Fahrzeugortungssystem in einem Autohaus dokumentiert. Diese Dokumentanforderungen werden vollständig und einheitlich beschrieben.

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

Wie die Anwendungsschnittstelle aussehen wird, wurde im Einklang mit den Erfahrungen und Wünschen der Benutzer erstellt.

Kapitel 2, „Ausgangslage und Ziele“, beschreibt den Grund für dieses Projekt und was damit entwickelt werden soll.

Im 3. Kapitel, genannt „Allgemeine Architektur“, finden sich Erläuterungen zu den notwendigen physikalischen und konzeptionellen Teilkomponenten des Systems. Wenn es eine notwendige Anforderung an die Architektur gibt, wird dies auch hier erklärt

Im Kapitel 4 „Funktionale Anforderungen“ werden alle funktionalen Anforderungen im Projekt erläutert. Dazu gehören User Story, Use Cases und fachliche Anforderungen.

Im 5. Kapitel „nichtfunktionale Anforderungen“ steht nicht das, was das System tut, sondern wie es tut im Vordergrund und wird so erklärt.

In Kapitel 6 werden detaillierte und robuste Akzeptanzkriterien mit der 5W-1H-Methode festgelegt. Diese Kriterien werden in diesem Abschnitt erläutert.

In Kapitel 7 werden Meilensteine aufgelistet und ihre Auswirkungen auf das Projekt erläutert.

Wichtige Referenzen sind in Kapitel 8 "Referenzen" aufgeführt.

2. Ausgangssituation und Ziele

2.1 Einleitung

Kleine Unternehmen und Handwerker nutzen heute einige lokale Softwaredienste. Es wurde beobachtet, dass diese Nutzungsprozesse für Unternehmen sehr vorteilhaft sind. In diesem Zusammenhang werden in diesem Abschnitt die Anwendung dieses Systems in einem Autohaus und seine Ziele erläutert.

2.2 Problemstellung

Eine AutoHaus hat einige Probleme mit dem Verkauf der Autos. Die Verkäufe müssen erhöht werden.

Die Lösung dafür:

Ein Programm mit der Interface soll die Verkäufe einfacher machen.

Gewünschte Autos müssen direkt mit dem Filter gefunden werden.

Rahmenbedingungen

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

- Als Software - Entwicklungstool soll IntelliJ verwendet werden.
- Die Backend Applikationen sollen mit Java Framework und Spring Framework,
- Die Frontend Applikationen mit JavaFX realisiert werden, und
- Die persistenten Daten sollen in einem SQL - Datenbank mit der Sprache MySQL abgespeichert werden.

3 Gesamtarchitektur

3.1 Einleitung

In diesem Abschnitt soll der Umfang des Projekts besser verstanden werden. Die Umrisse und Grenzen des Systems sind festgelegt. Die Benutzeroberfläche wurde nach Kundenwunsch erstellt. Systemkomponenten im Projekt werden erklärt und Informationen über mögliche Systemkomponenten gegeben.

Mit diesem Abschnitt werden die funktionalen und nicht-funktionalen Anforderungen des angeforderten Systems besser verstanden.

3.2 Gesamtarchitektur

Die allgemeine Architektur wird durch Visualisierung mit dem folgenden Diagramm erklärt.

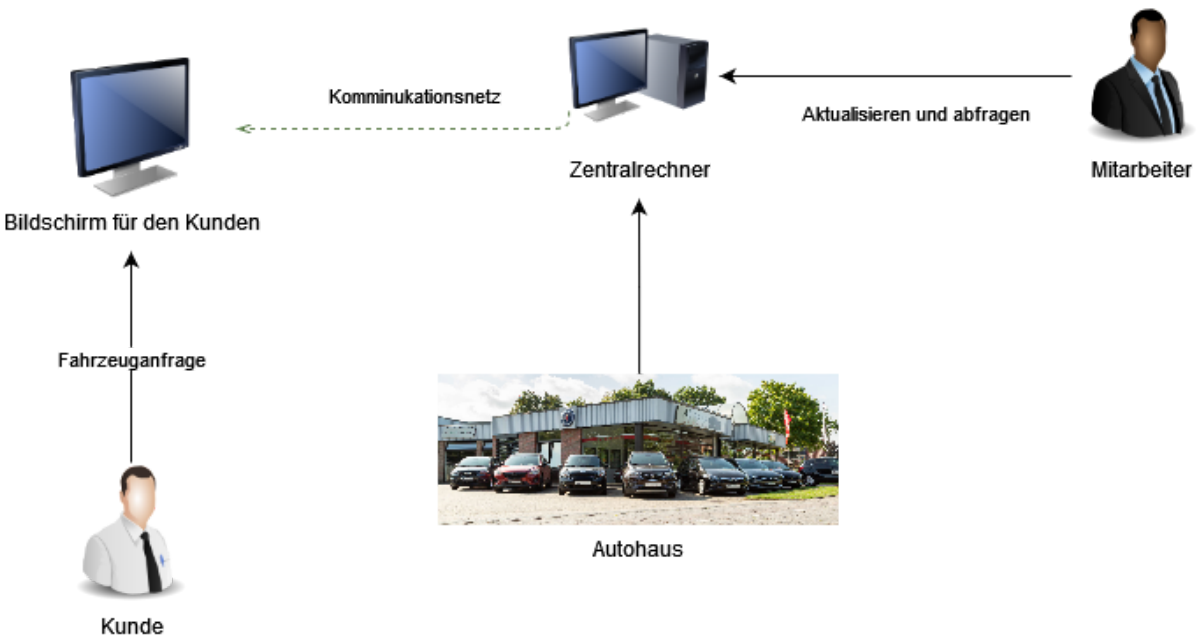
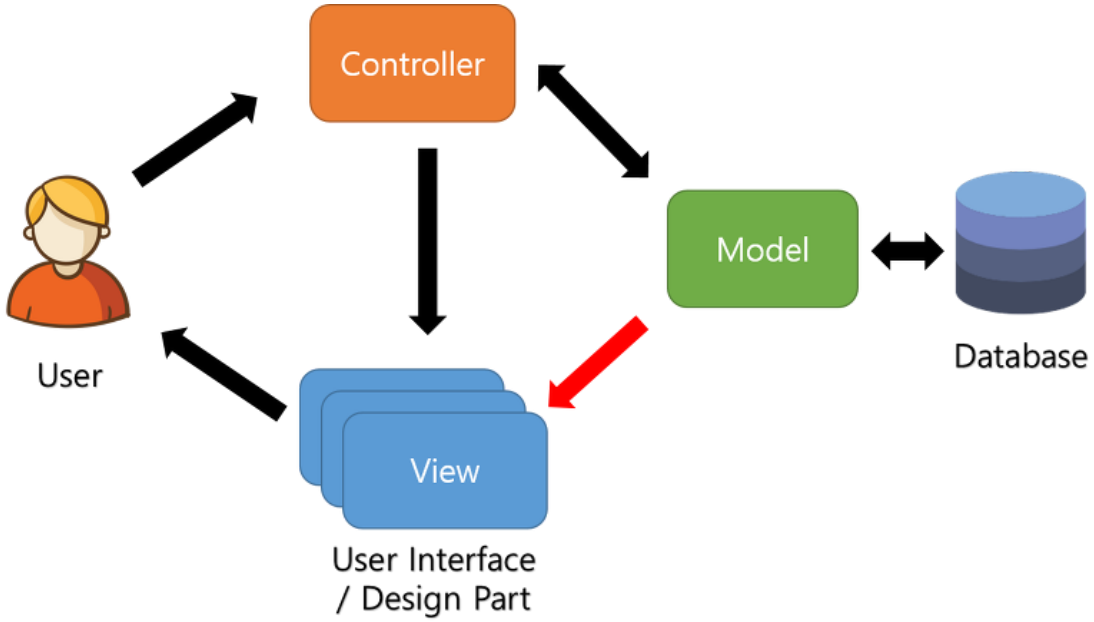


*Abbildung SEQ Abbildung * ARABIC 1 : Visuelle Darstellung der allgemeinen Architektur*



Um die Grenzen eines Systems besser zu verstehen, ist es sehr hilfreich, eine allgemeine Architektur in Form eines Diagramms vorzubereiten. In der in Bild zwei gezeigten allgemeinen Architektur werden die Systemgrenzen für die Beteiligten festgelegt und die notwendigen Komponenten erläutert.

Die Komponentenarchitektur soll nach dem Architekturmuster „Model-View-Controller“ (MVC) entworfen werden, siehe das Spring Framework für Detailinformation.

3.3 Komponente Zentralrechner

Der zentrale Computer ist für den Geschäftsinhaber. Von diesem Host aus kann der Geschäftsinhaber Daten über seine Autos sehen oder manipulieren und sie ändern.

Dieser zentrale Computer ist das Verwaltungsgerät, auf dem alle Operationen durchgeführt werden können. Daher können nur zuständige Personen im Autohaus auf diesen Computer zugreifen.

3.4 Komponente Bildschirm für den Kunden

Wenn Kunden ins Autohaus kommen, können sie auf diesem Bildschirm die Marke/das Modell ihres Wunschautos abfragen. Dieser Bildschirm wird so lange wie möglich taktil und für alle zugänglich sein.

*Abbildung SEQ Abbildung * ARABIC 2: Visualisierte Erklärung des Model-View-Controllers*

3.5 Externe Schnittstellen

Die externen Schnittstellen sind:

1) Zentrale Computerschnittstelle: Für Autohausmitarbeiter, um Informationen in der Datenbank zu aktualisieren und zu manipulieren
2) Benutzeroberfläche des Kundenbildschirms: Damit Kunden Fahrzeuganfragen ohne Unterstützung durch Mitarbeiter stellen können

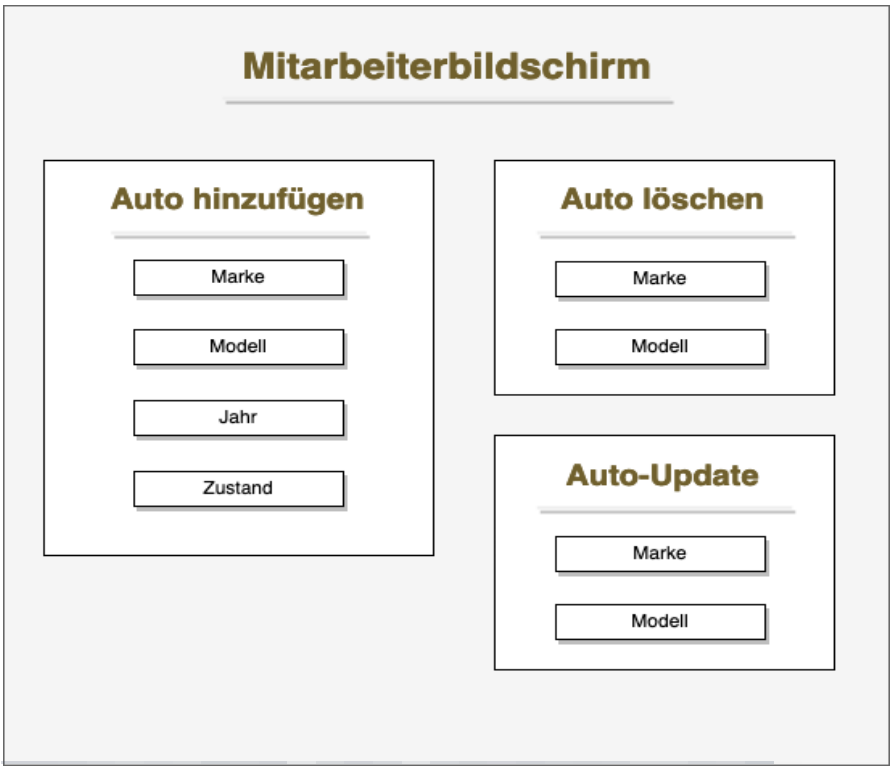
4 Funktionale Anforderungen

4.1 Einleitung

In diesem Abschnitt werden sowohl Anforderungen an das Gesamtsystem als auch an die einzelnen Komponenten des Systems beschrieben. Hierbei werden sowohl User Stories als auch Use Cases verwendet.

4.2 ZR Use Cases / User Stories

- **/ZR-1/** Die Funktionen, auf die nur Mitarbeiter auf dem zentralen Computer zugreifen können, sollten unter 3 Hauptüberschriften zusammengefasst werden.
- **/ZR-2/** Diese drei Hauptüberschriften sollten im Wesentlichen als Hinzufügen, Löschen und Aktualisieren definiert werden.
- **/ZR-3/** Das Konzeptdesign der Schnittstelle des Zentralcomputers ist unten angegeben, siehe Abbildung 3 ZR GUI Prototype.

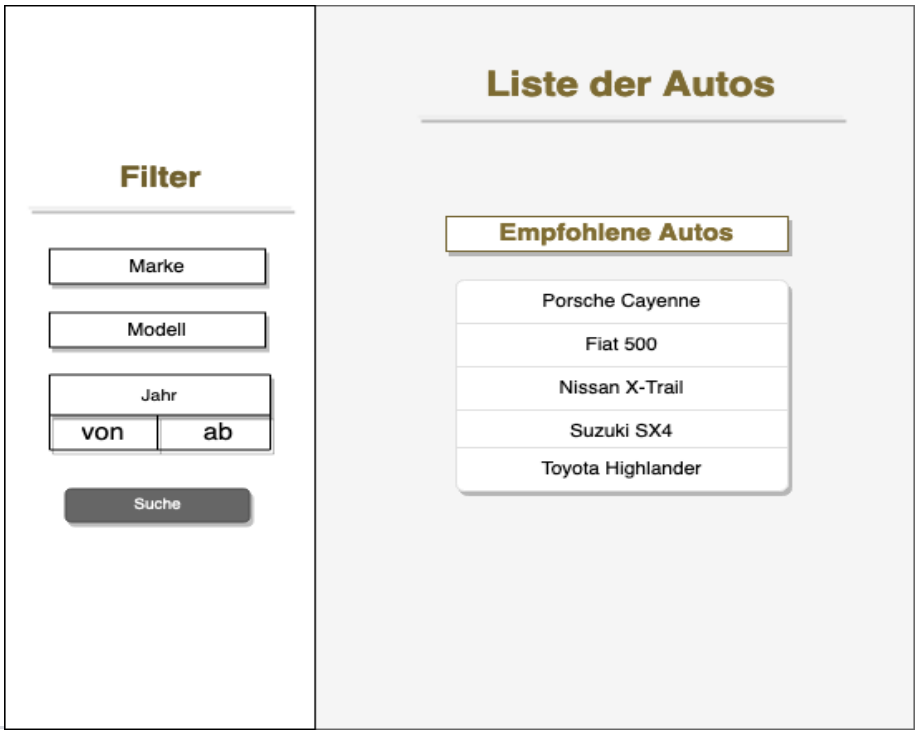


- **/ZR-4/** Um ein Auto hinzuzufügen, müssen Sie die Marke, das Modell und das Baujahr des Autos eingeben. Optional wird der Status des Autos eingetragen.
- **/ZR-5/** Wird beim Hinzufügen eines Autos die Statusinformation „verkauft“ eingetragen, ist die Angabe von wem und an wen der Verkauf getätigt wurde zwingend erforderlich.
- **/ZR-6/** Beim Löschvorgang wird das zu löschende Fahrzeug aus der Datenbank gesucht und das zugehörige Fahrzeug aus dem System gelöscht.
- **/ZR-7/** Beim Aktualisieren der Fahrzeuginformationen wird zuerst das zu aktualisierende Fahrzeug gefunden, und dann wird das zu aktualisierende Merkmal ausgewählt und aktualisiert.

4.3 BFK Use Cases / User Stories

- **/BFK-1/** Der für Kunden erstellte Bildschirm hilft Kunden, Autos selbst ohne Hilfe von Mitarbeitern zu finden.
- **/BFK-2/** Die Bildschirmoberfläche ist grundsätzlich in 3 Abschnitte unterteilt. Dies sind die Schnellsuche, die Fahrzeugliste und der Hilferuf, siehe Abbildung 4 GUI-Prototyp.

- **/BFK-3/** In der Schnellsuche sollte nur ein Schlüsselwort eingegeben werden. Unter diesem Stichwort werden die zugehörigen Autos aufgelistet.
- **/BFK-4/** Wenn auf die Schaltfläche „Autos“ geklickt wird, sollten sich zwei Abschnitte auf dem Bildschirm öffnen. Das erste ist das Filtern und das zweite sind empfohlene Tools. Empfohlene Fahrzeuge werden anhand der meistgesehenen Fahrzeuge ermittelt. Siehe Abbildung 5.



4.4 Datenmodell

- **/DAT-1/ Es** ist erforderlich, ein externes Datenbanksystem (DBMS), das auf SQL basiert, zu verwenden, um die Daten persistent zu speichern. Um das Datenbankschema zu planen, kann ein Entity-Relationship-Diagramm erstellt werden.
- **/DAT-2/ Das** Datenmodell soll im UML Klassendiagramm modelliert werden.
- **/DAT-3/ Die** Daten sollten leicht aus der Datenbank abgerufen und verarbeitet werden können.
- **/DAT-4/ Beim** Entwurf des Datenmodells sollte berücksichtigt werden, welche Parameter von den API-Schnittstellen benötigt werden, um sicherzustellen, dass das Datenmodell diese Daten effektiv unterstützt und bereitstellt. Es ist wichtig, dass das Datenmodell so gestaltet wird, dass es die Anforderungen der API-Schnittstellen erfüllt, um eine reibungslose Interaktion zwischen der Anwendung und der API zu gewährleisten.

5 Nichtfunktionale Anforderungen

5.1 Einleitung

In diesem Abschnitt sind die nicht-funktionalen Anforderungen an das Gesamtsystem sowie an die einzelnen Komponenten des Systems beschrieben. Der Fokus liegt hierbei auf der Softwarequalität und es wird insbesondere auf das Testen eingegangen.

5.2 Nicht-funktionale Anforderungen an die Systemarchitektur

- **/SYS-1/** Die Architektur für das Deployment sieht vor, dass jede Systemkomponente in einem eigenen Betriebssystem-Prozess ausgeführt werden sollte, um eine verteilte Architektur zu ermöglichen. Das bedeutet, dass jedes Teil des Systems in einem eigenen Prozess läuft und somit unabhängig voneinander arbeitet. Dies kann die Stabilität und Skalierbarkeit des Systems verbessern

- **/SYS-2/** Die Systemarchitektur sollte sicherstellen, dass die Komponenten unabhängig voneinander entwickelt und ausgetauscht werden können, um eine hohe Flexibilität und Erweiterbarkeit zu gewährleisten.

5.3 Nicht-funktionale Anforderungen an die Entwicklungsumgebung

- **/DEV-1/** Es besteht keine Einschränkung bei der Wahl der Entwicklungsumgebung. Es kann jede gewünschte Entwicklungsumgebung verwendet werden.

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

- **/TOL-1/** Es ist festgelegt, dass für die Umsetzung der Backend-Anwendungen ein Java-Framework genutzt werden muss.
- **/TOL-2/** Für die Implementierung der Frontend-Anwendung ist die Verwendung der JavaFX Rich Client-Technologie vorgesehen.
- **/TOL-3/** Die Daten, die persistent gespeichert werden sollen, müssen in einer SQL-Datenbank mit der Sprache MySQL abgelegt werden.

5.5 Nicht-funktionale Anforderungen an die Teststrategie (Qualitätssicherung)

- **/TEST-1/** Die Testfälle müssen eine hohe Testabdeckung aufweisen, um sicherzustellen, dass alle Funktionen und Anforderungen der Software getestet werden.
- **/TEST-2/** Es ist erforderlich, dass alle Anforderungen, User Stories und Use Cases getestet und dokumentiert werden.
- **/TEST-3/** Die Tests müssen sicherstellen, dass die Software benutzerfreundlich und intuitiv zu bedienen ist.
- **/TEST-4/** Die Tests müssen sicherstellen, dass die Software zuverlässig und stabil läuft und keine unerwarteten Fehler oder Abstürze verursacht.

6 Abnahmekriterien

Das Projekt wird mit den folgenden Artefakten abgegeben:

- Dokumentation:
    - Pflichtenheft: INF202 - AutoHaus -Auto Galerie-Pflichtenheft-2023.v1.0.doc
- Software:
    - Link zu GitHub Projekt: https://github.com/orgs/Gruppe1-Fulya/teams/java-benders

7 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.