

TAU INF202 Software Engineering
Individuelles Projekt
Pflichtenheft

Projektdokumentation

Version: **2023.1.0**

Status: **Entwurf**

Autowasch Termine

Verantwortliche/r:

Hüseyin Tuğşad Şen e200503047@stud.tau.edu.tr

Mert Ali Hanbay e200503009@stud.tau.edu.tr

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

Dokumentenverwaltung

Dokument-Historie

Version	Status *)	Datum	Verantwortlicher	Änderungsgrund
v1.0	Entwurf	03.04.2023	Hüseyin Tuğsad Şen, Mert Ali Hanbay	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

Lucid Chart

Autowäschern

Inhaltsverzeichnis

1. Einleitung	4
2. Ausgangssituation und Ziele	5
3. Gesamtarchitektur	6
4. Funktionale Anforderungen	7
5. Nichtfunktionale Anforderungen	8
6. Abnahmekriterien	9
7. Projekt Meilensteine	10
8. Referenzen	11

Einleitung

Das Ziel dieses Projekts ist die Entwicklung einer App, die es Autowäschern ermöglicht, ihre Kunden über ihre Freizeiten, Preise, Telefonnummer und Adresse zu informieren und Kunden die Möglichkeit gibt, Termine zur Autowäsche zu vereinbaren.

Die Anwendung wird es Kunden ermöglichen, alle verfügbaren Autowäscher in ihrer Nähe zu sehen und deren freien Termine einzusehen, um einen Termin zu buchen. Die Buchung erfolgt durch die Eingabe von persönlichen Informationen wie Name, Telefonnummer und Kennzeichen.

Die Anwendung wird ein Datenbanksystem verwenden, um die Informationen der Kunden und Autowäscher zu speichern und zu verwalten. Es wird auch in der Lage sein, aktuelle Terminzeiten in der Termintabelle abzurufen, um Kunden genaue Informationen über die verfügbaren Termine zu geben.

Unser Ziel ist es, eine benutzerfreundliche, zuverlässige und sichere Anwendung zu entwickeln, die es den Benutzern ermöglicht, schnell und einfach Termine zur Autowäsche zu buchen und den die Möglichkeit gibt, ihre Kunden effektiv zu verwalten. Wir werden uns bemühen, eine hohe Qualität der Anwendung und des Kundensupports sicherzustellen, um das Vertrauen und die Zufriedenheit unserer Benutzer zu gewinnen.

Ausgangssituation und Ziele

Die Autowäschebranche ist ein wichtiger Teil der Automobilindustrie und wird von Millionen von Menschen weltweit genutzt. In der heutigen digitalen Welt suchen immer mehr Kunden nach schnellen und einfachen Möglichkeiten, um Termine zu buchen und Informationen über verfügbare Autowäscher und deren Preise und Dienstleistungen zu erhalten. Die meisten Autowäscher verfügen jedoch nicht über ein effektives System, um ihre Kunden zu informieren und Termine zu verwalten, was zu Frustrationen bei Kunden und Autowäschern führen kann.

Einleitung

In diesem Kapitel werden die wichtigsten Aspekte des Projekts vorgestellt, darunter die Problemstellung, das Systemumfeld, die Rahmenbedingungen, die Ziele und die Nutzergruppen. Es wird erläutert, welches Problem das Projekt lösen soll und welche Ziele damit verfolgt werden.

Problemstellung (Funktionalität)

Das Hauptproblem besteht darin, dass Kunden Schwierigkeiten haben, Termine für die Autowäsche zu vereinbaren, da die Verfügbarkeit der Waschanlagen nicht immer transparent ist. Das neue System soll dieses Problem lösen, indem es den Kunden einen einfachen und effizienten Weg bietet, Termine für die Autowäsche zu buchen und Informationen zu den verfügbaren Waschanlagen zu erhalten.

Stakeholder (Anwender):

Die Stakeholder umfassen Autowäscher und deren Kunden. Autowäscher profitieren durch die erhöhte Effizienz und Kunden profitieren durch die einfache und bequeme Möglichkeit, Termine für die Autowäsche zu vereinbaren.

Systemumfeld (Einsatzumgebung)

Das System wird als Desktop-App entwickelt und soll von Autowäschern und deren Kunden auf ihren Computern genutzt werden können. Das System soll in der Lage sein, eine große Anzahl von Anfragen von Autowäschern und Kunden zu verarbeiten.

Rahmenbedingung (Einschränkungen)

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

- Als Software-Entwicklungstool soll entweder Visual Studio Code verwendet werden,
- die Backend Applikationen sollen mit Java Framework und Spring Framework,
- die Frontend Applikationen mit Flutter Technologie realisiert werden, und
- die persistenten Daten sollen in einer SQL-Datenbank abgespeichert werden.

Ziele (Lösung).

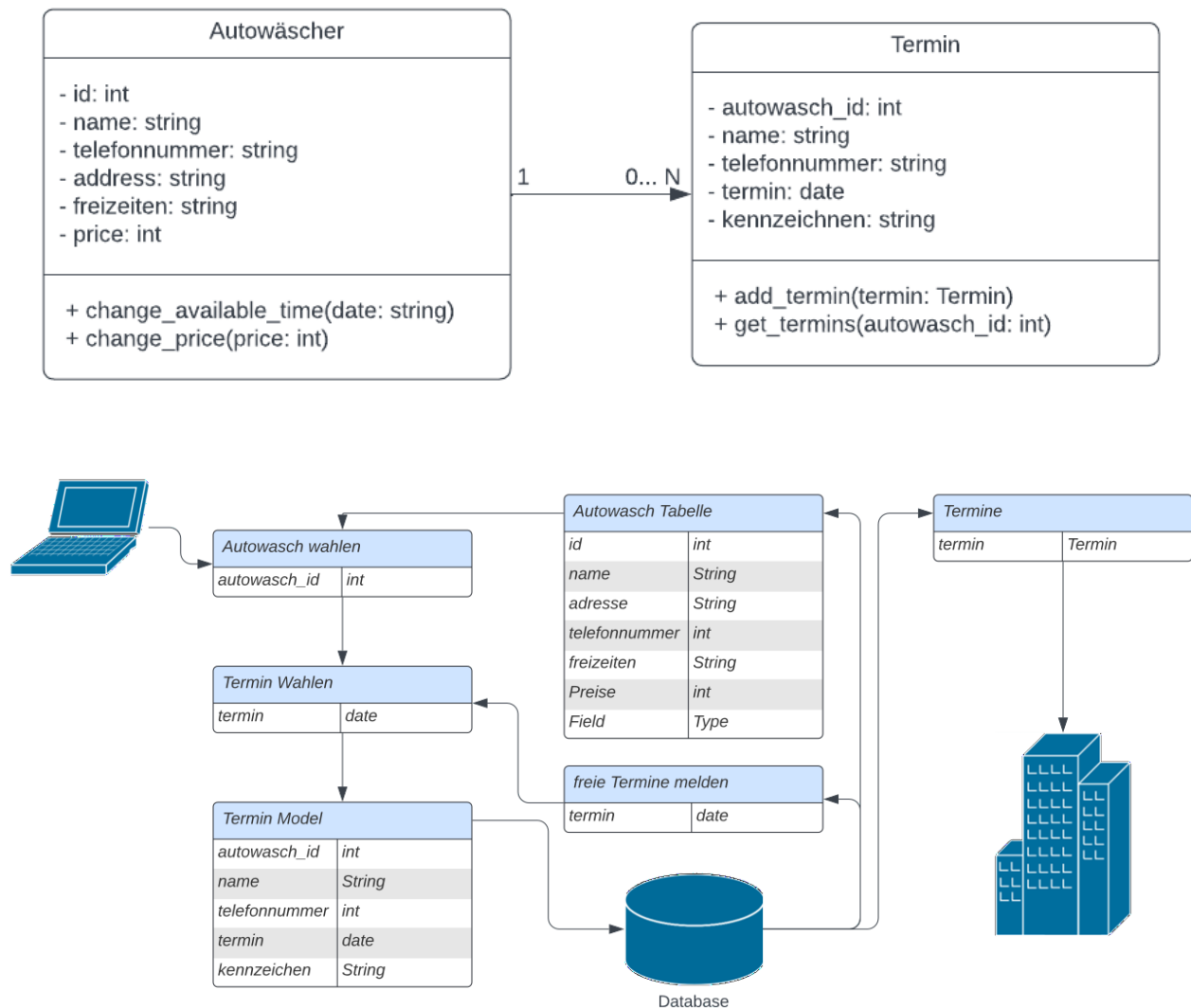
Das Ziel dieses Projekts ist es, eine benutzerfreundliche App zu entwickeln, die Autowäschern und Kunden eine einfache Möglichkeit bietet, miteinander zu interagieren und Termine zu vereinbaren. Die App soll es Autowäschern ermöglichen, ihre Dienstleistungen zu bewerben, Kunden über ihre Verfügbarkeit und Preise zu informieren und Buchungen zu verwalten. Kunden sollen die Möglichkeit haben, verschiedene Autowäscher zu suchen, Bewertungen zu lesen und Termine zu buchen, die ihren Bedürfnissen entsprechen. Die App soll es beiden Seiten erleichtern, miteinander zu kommunizieren und zu koordinieren, was zu einer effektiveren und effizienteren Autowasch-Industrie führen wird.

Gesamtarchitektur

Einleitung

Die Autowasch-Termin-App soll es Autowäschern ermöglichen, ihren Kunden Termine für Autowäsche anzubieten und Kunden die Möglichkeit geben, diese Termine bequem zu buchen. Die Gesamtarchitektur der App soll sicherstellen, dass die Anwendung benutzerfreundlich und zuverlässig ist, sowie die Daten der Kunden und Autowäschern sicher und effizient verwaltet werden.

Gesamtarchitektur



Externe Schnittstellen

Die externen Schnittstellen sind:

1. Die Anwendung leitet den Benutzer zu einer externen Navigations-App (Google Maps usw.), um die Entfernung zwischen dem Kunden und dem Fahrzeugreiniger zu bestimmen. Die externe Navigations-App verwendet die Standort-API des Computers.

Funktionale Anforderungen

Einleitung

Die Autowasch-Termin-App ist eine Anwendung, die es den Kunden ermöglicht, Autowäsche-Termine online zu buchen und Autowäscher können ihre Dienstleistungen und Verfügbarkeiten verwalten. Das Ziel dieser Anwendung ist es, den Prozess der Terminvereinbarung für Autowäsche für Kunden zu vereinfachen und Autowäschern dabei zu helfen, ihre Dienstleistungen effektiver zu verwalten.

UI Use Cases

1. Registrierung und Anmeldung:

- In app gibt es keine Registrierung Systeme, Benutzern kann alle Autowäschern, ihre Preisangebote und sehen frei Termine, danach müssen sie ihre Name, Telefonnummer, Terminsstunde und ihre Autos Kennzeichen um ein Termin zu buchen.

2. Suche nach einem Autowäscher:

- Ein Benutzer kann nach einem Autowäscher mit Autowäschern Name suchen.
- Die App zeigt dem Benutzer eine Liste von verfügbaren Autowäschern mit ihren Informationen.

3. Anzeige von Autowäscher-Informationen:

- Ein Benutzer kann sich die Details eines Autowäschers anzeigen lassen, einschließlich der Adresse, der Telefonnummer, der Öffnungszeiten und der Preise.
- Der Benutzer kann auch die verfügbaren Termine und Dienstleistungen des Autowäschern sehen.

4. Buchung eines Termins:

- Ein Benutzer kann einen Termin bei einem Autowäscher buchen, indem er das Datum und die Uhrzeit auswählt und seine Kontaktinformationen angibt.
- Die App bestätigt die Buchung und zeigt dem Benutzer eine Bestätigungsseite.


5. Stornierung eines Termins:

- Ein Benutzer kann einen gebuchten Termin stornieren, indem er das Datum, die Uhrzeit und seine Autos Kennzeichen auswählt und den Stornierungsgrund angibt.
- Die App bestätigt die Stornierung und aktualisiert die verfügbaren Termine des Autowäschern.


Autowasch Termine

Home

Autowäscher Panel




Autowäschern Liste




Kaya Autowasch
Kavacık Beykoz/İstanbul
09.00 - 18.00
+90 555 542 14 51

150 ₺




Han Autowasch
Levent Beşiktaş/İstanbul
12.00 - 20.00
+90 534 345 14 23

200 ₺



Shen Autowasch
Çağlayan Kağıthane/İstanbul
09.00 - 23.00
+90 544 442 18 11

100₺



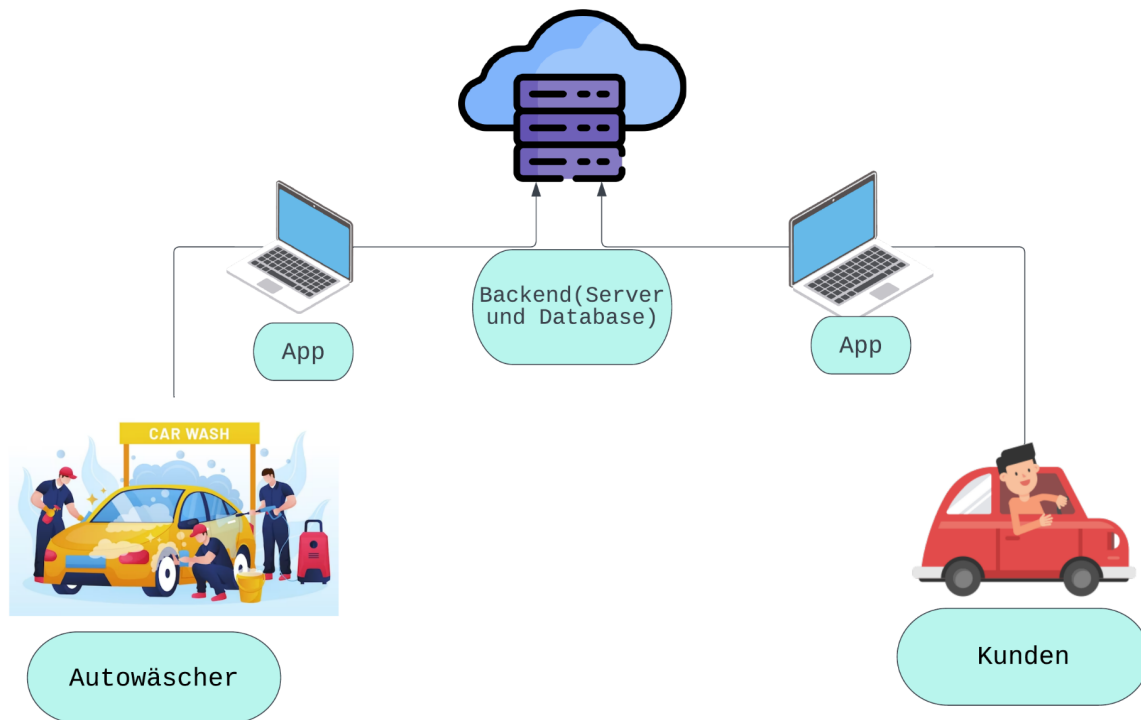
Imamson Autowasch
Atakent Küçükçekmece/İstanbul
08.00 - 20.00
+90 543 742 88 61

250₺

API Use Cases

1. **GET** `getAutoWascher(int: id)`: API, die Informationen über die Autowaschanlage mit der angegebenen ID zurückgibt, einschließlich der Öffnungszeiten, Preise, Telefonnummer und Adresse.
2. **GET** `getTermin(int: id)`: API, die registrierten Termine für die angegebene Autowaschanlage anzeigt.
3. **POST** `addTermin(String: Namen, int: Telefonnummer, String: Kennzeichen)`: API, einen neuen Auto-Waschtermin hinzufügt, indem Informationen wie Name, Telefonnummer und Kennzeichen des Fahrzeugs in die Datenbank eingegeben werden.
4. **POST** `addAutoWascher(String: Öffnungszeiten, String: Namen, int: Preise, int: Telefonnummer, String: Adresse, int: Password)`: API, eine neue Autowaschanlage hinzufügt, indem Informationen wie Öffnungszeiten, Name, Telefonnummer, Adresse und Preise eingegeben werden. Wenn das korrekte Passwort angegeben wird, fügt das API diese Informationen zur Datenbank hinzu.
5. **POST** `deleteTermin(String Kennzeichen,)`: API, den registrierten Termin für das angegebene Fahrzeugkennzeichen löscht.

Technischen und fachliche Anforderungen



Datenmodel

Autowasch Tabelle

id	name	adresse	telefonnummer	freizeiten	preise
1	Kaya Autowasch	Kavacık/Beykoz	0546 678 12 34	07.00-21.00	100
2	Han Autowasch	Harbiye/Şişli	0533 323 99 99	09.00-23.00	200
3	Shen Autowasch	Merkez/Kağıthane	0599 111 11 11	09.00-21.00	150

Termin Tabelle

autowash_id	name	telefonnummer	termin	kennzeichen
1	Ahmet İmamoğlu	0558 741 54 23	11.00-12.00	34 JK 198
3	Betül Çakıroğlu	0555 545 55 55	15.00-16.00	28 FB 666

Nichtfunktionale Anforderungen

Einleitung

In diesem Kapitel werden die nicht-funktionalen Anforderungen für die Autowasch-Termin App beschrieben, um sicherzustellen, dass das System den Erwartungen der Benutzer entspricht und effektiv und effizient betrieben werden kann.

Nicht-funktionale Anforderungen an die Systemarchitektur (Architekturmuster, Deployment)

1. **Skalierbarkeit:** Das System sollte skalierbar sein, um eine wachsende Anzahl von Benutzern und Anfragen zu unterstützen.
2. **Zuverlässigkeit:** Das System sollte eine hohe Verfügbarkeit und Ausfallsicherheit gewährleisten, um sicherzustellen, dass Benutzer jederzeit auf das System zugreifen können.
3. **Sicherheit:** Das System sollte sicher sein und Schutz vor Angriffen und unbefugtem Zugriff bieten. Es sollte auch sicherstellen, dass Daten während der Übertragung und Speicherung verschlüsselt werden.
4. **Modularität:** Das System sollte modular aufgebaut sein, um eine einfache Wartbarkeit und Skalierbarkeit zu ermöglichen.
5. **Flexibilität:** Das System sollte flexibel sein und es dem Benutzer ermöglichen, neue Funktionen oder Module hinzuzufügen, um den Anforderungen des Geschäfts gerecht zu werden.

Nicht-funktionale Anforderungen an die Entwicklungsumgebung

1. **Kompatibilität:** Die Entwicklungsumgebung sollte mit verschiedenen Betriebssystemen und Hardware Konfigurationen kompatibel sein, um sicherzustellen, dass das Team effizient zusammenarbeiten kann.
2. **Leistung:** Die Entwicklungsumgebung sollte schnell und reaktionsfähig sein, um eine schnelle Iteration und Entwicklung zu ermöglichen.
3. **Skalierbarkeit:** Die Entwicklungsumgebung sollte skalierbar sein, um das Wachstum des Projekts zu unterstützen und sicherzustellen, dass sie auch bei zunehmender Komplexität und Größe effektiv bleibt.
4. **Zuverlässigkeit:** Die Entwicklungsumgebung sollte zuverlässig sein und die Integrität des Quellcodes sicherstellen, um Fehler oder Verlust von Arbeit zu vermeiden.
5. **Sicherheit:** Die Entwicklungsumgebung sollte sicher sein und sicherstellen, dass vertrauliche Daten des Projekts geschützt sind.
6. **Versionierung:** Die Entwicklungsumgebung sollte eine effektive Versionierung Funktion bieten, um eine vollständige Nachverfolgung von Änderungen und Entwicklungsverlauf zu ermöglichen.
7. **Integration:** Die Entwicklungsumgebung sollte in der Lage sein, mit anderen Tools und Technologien effektiv zu integrieren, um die Zusammenarbeit und Integration mit anderen Teams und Systemen zu erleichtern.
8. **Dokumentation:** Die Entwicklungsumgebung sollte es ermöglichen, die Dokumentation effektiv zu verwalten und zu aktualisieren, um sicherzustellen, dass das Projekt stets gut dokumentiert ist.

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

1. **Programmiersprache:** Die App wird in Flutter mit Dart entwickelt. Flutter haben sich in der Desktop App-Entwicklung bewährt und bieten eine gute Performance und ein großes Ökosystem an Libraries und Tools.
2. **IDE:** Die Entwicklungsumgebung soll Visual Studio Code (VSCode) sein. VSCode ist eine kostenlose und leichtgewichtige IDE mit vielen Erweiterungen und Integrationen in die Flutter-Entwicklung.
3. **Framework:** Die Backend-API der App wird mit Java Spring Framework entwickelt. Java Spring bietet eine robuste und skalierbare Plattform für die Entwicklung von RESTful APIs.
4. **Datenbank:** Die App soll eine MySQL-Datenbank verwenden. MySQL ist eine bewährte und zuverlässige Datenbanklösung mit einer hohen Verfügbarkeit und Performance.
5. **Versionskontrolle:** Das Versionskontrollsystem soll Git sein, um eine effektive Zusammenarbeit im Team zu ermöglichen und die Code-Qualität zu gewährleisten.

Nicht-funktionale Anforderungen an die Teststrategie (Qualitätssicherung)

1. **Zuverlässigkeit:** Die App sollte zuverlässig und fehlerfrei funktionieren. Dies erfordert eine umfassende Teststrategie, um Fehler zu identifizieren und zu beheben, bevor die App veröffentlicht wird.
2. **Sicherheit:** Die App sollte sicher sein und keine personenbezogenen Daten oder sensiblen Informationen preisgeben. Es ist wichtig, eine umfassende Sicherheitsstrategie zu implementieren, um sicherzustellen, dass die Daten der Benutzer sicher sind.
3. **Leistung:** Die App sollte schnell und effizient laufen. Es ist wichtig, die Leistung der App zu testen, um sicherzustellen, dass sie reibungslos funktioniert, selbst wenn viele Benutzer gleichzeitig darauf zugreifen.
4. **Skalierbarkeit:** Die App sollte skalierbar sein, um den Anforderungen des Benutzers Anstiegs gerecht zu werden. Es ist wichtig, die App auf ihre Skalierbarkeit zu testen und sicherzustellen, dass sie mit zunehmender Nutzerzahl reibungslos funktioniert.
5. **Kompatibilität:** Die App sollte auf verschiedenen Geräten und Plattformen kompatibel sein. Es ist wichtig, die App auf verschiedenen Geräten und Betriebssystemen zu testen, um sicherzustellen, dass sie reibungslos funktioniert und eine einheitliche Benutzererfahrung bietet.

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-AutowaschTermine-Pflichtenheft-2023.v1.0.docx
- Software
 - Link zu GitHub Projekt: https://github.com/Gruppe1-Fulya/autowasch_termine
- Evidenz:
 - System/Software-Demo via Videoclip: [videoclip-link](#)

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

Projekt Meilensteine

Folgende Meilensteine sind verbindlich definiert:

Meilenstein #1: Das Lastenheft wurde erfolgreich abgeschlossen und in Absprache mit allen Stakeholdern finalisiert.

Meilenstein #2: Das Pflichtenheft wurde erfolgreich abgeschlossen und in Absprache mit allen Stakeholdern finalisiert

Referenzen

Diese Kapitel beinhaltet die wichtigsten Literaturquellen aufgelistet.