Inhalt

[Einführung und Zeile 2](#_Toc145206644)

[Aufgabenstellung 2](#_Toc145206645)

[UseCases 2](#_Toc145206646)

[UC01: CRUD Customers 2](#_Toc145206647)

[UC02: Search Customers 2](#_Toc145206648)

[UC03: CRUD Cars 2](#_Toc145206649)

[UC04: Search Cars 2](#_Toc145206650)

[UC05: Daily Fee 3](#_Toc145206651)

[UC06: Create Reservation 3](#_Toc145206652)

[UC07: Generate Contracts 3](#_Toc145206653)

[Qualitätsziele 3](#_Toc145206654)

[Stakeholder 3](#_Toc145206655)

[Randbedingungen 3](#_Toc145206656)

[Kontextabgrenzung 3](#_Toc145206657)

[Lösungsstrategie 4](#_Toc145206658)

[Bausteinsicht 4](#_Toc145206659)

[Context 4](#_Toc145206660)

[Containers 5](#_Toc145206661)

[Components 6](#_Toc145206662)

[Classes 7](#_Toc145206663)

[Domainmodell 8](#_Toc145206664)

[Verteilungssicht 8](#_Toc145206665)

[Deployment View 8](#_Toc145206666)

[Logical View 9](#_Toc145206667)

CarRent

# Einführung und Zeile

## Aufgabenstellung

Es soll ein neues Autovermietungssystem „CarRent“ erstellt werden. Das System soll aus einem Server-Teil und optional einen Web-Client bestehen.

**Folgende Detailinformationen liegen unstrukturiert über das zu entwickelnde System vor:**

* *Der Sachbearbeiter kann Kunden mit Namen und Adresse und Kundennummer im System verwalten, d.h. erfassen, bearbeiten, löschen und den Kunden mit dessen Namen oder Kundennummer suchen.*
* *Der Sachbearbeiter kann zudem die Autos von CarRent verwalten und nach denen suchen.*
* *Jedes Auto kann einer bestimmten Klasse zwischen Luxusklasse, Mittelklasse oder Einfachklasse zugeordnet werden und besitzt zudem eine Marke, einen Typ und eine eindeutige Identifikation.*
* *Jede Klasse besitzt eine Tagesgebühr.*
* *Bei einer neuen Reservation kann der Kunde ein Auto aus einer bestimmten Klasse wählen. Er muss zudem die Anzahl der Tage angeben, die er das Auto gerne mieten möchte. Dabei werden die Gesamtkosten berechnet. Wird die Reservation gespeichert, so wird sie mit einer Reservationsnummer ablegt.*
* *Bei Abholung des Autos wird die Reservation in einen Mietvertrag umgewandelt.*

## UseCases

### UC01: CRUD Customers

Der Sachbearbeiter pflegt und aktualisiert die Kundeninformationen im System (CRUD).

### UC02: Search Customers

Der Sachbearbeiter führt eine Suche nach Kunden im System durch, indem er entweder deren Kundennummer oder ihren Namen verwendet.

### UC03: CRUD Cars

Der Sachbearbeiter verwaltet die Fahrzeugdaten im System (CRUD). Dabei kann er Fahrzeugen verschiedene Kategorien (Luxusklasse, Mittelklasse, Einfachklasse), Marken, Typen und eindeutige Identifikationsnummern zuweisen.

### UC04: Search Cars

Der Sachbearbeiter sucht und identifiziert Fahrzeuge im System anhand ihrer eindeutigen Identifikationsnummern.

### UC05: Daily Fee

Der Sachbearbeiter hat die Befugnis, für jede Fahrzeugklasse eine individuelle Tagesgebühr festzulegen.

### UC06: Create Reservation

Kunden können Reservierungen für bestimmte Fahrzeugkategorien und für eine festgelegte Anzahl von Tagen erstellen. Das System berechnet automatisch die Gesamtkosten. Bei der Speicherung der Reservierung wird eine eindeutige Reservierungsnummer vom System generiert.

### UC07: Generate Contracts

Das System erstellt einen Mietvertrag auf Basis der getätigten Reservierung, sobald der Kunde das gewünschte Fahrzeug in Empfang nimmt.

## Qualitätsziele

|  |  |  |
| --- | --- | --- |
| **Priorität** | **Qualitätsziel** | **Szenario** |
| 1 | Performante User Experience | Interaktion mit Website |
| 2 | Data Consistency | Ausfall von Server Darf nicht zu Datenverlusten führen |
| 3 | System avialability | Auch bei Ausfall eines Servers soll System innerhalb von 2 Minuten wieder verfügbar sein |

## Stakeholder

|  |  |  |
| --- | --- | --- |
| **Rolle** | **Kontakt** | **Erwartungshaltung** |
| Lehrer | Cedric Menzi | Dokumentation nach arc42 |

# Randbedingungen

* Die Daten sollen durch das Repository-Pattern in eine Datenbank gespeichert werden können.
* Die Business Logik wird auf der Serverseite ausgeführt und bietet eine REST-API an.
* Zuerst wird ein Monolith erstellt, der später in eine Microservice-Architektur umgewandelt werden soll.

# Kontextabgrenzung

* Die Anmeldung und Authentifizierung von Kunden und Sachbearbeitern wird in diesem Projekt nicht behandelt.

# Lösungsstrategie

* Die Integration des Repository-Patterns
* Die Abstraktion der Presistence und Business Layers mittels Schnittstellen
* Die Bereitstellung für den Zugriff über REST-API
* Das Design erfolgt zunächst als Monolith, jedoch mit einer Vorbereitung für die zukünftige Umstellung auf eine Microservice-Architektur.
* Die Anwendung wird nach den Prinzipien des Domain-Driven Designs gestaltet.

# Bausteinsicht

## Context

Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte Beschreibung

## Containers

Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte Beschreibung

## Components

Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

Automatisch generierte Beschreibung

## Classes

Ein Bild, das Text, Screenshot, Diagramm, Rechteck enthält.

Automatisch generierte Beschreibung

## Domainmodell

Ein Bild, das Text, Screenshot, Diagramm, parallel enthält.

Automatisch generierte Beschreibung

# Verteilungssicht

## Deployment View

Ein Bild, das Text, Diagramm, Screenshot, Reihe enthält.

Automatisch generierte Beschreibung

## Logical View

Ein Bild, das Text, Diagramm, Screenshot, Rechteck enthält.

Automatisch generierte Beschreibung