I290 - Projektseminar - WiSe 22/23

Inhaltsverzeichnis

1. Aufgabenstellung und Ausgangssituation

1.1. Vision

Ziel dieses Projektseminars ist es, für die im Rahmen der Module Datenbanksysteme I und II verwendete Trommelhelden-Datenbank ein Web-Frontend zu entwickeln, welche grundlegenden Funktionen zur Datenverwaltung des fiktiven Waschmaschinen-Reparaturservices "Trommelhelden" beinhalten soll. Dies umfasst die Verwaltung von Auftragsdaten, Kundendaten, Mitarbeiterdaten und Daten der Ersatzteile, die während eines Auftrages zum Einsatz kommen. In Hinblick auf die Laufzeit sollen verschiedene Implementierungsalternativen aufgezeigt werden, um die Unverzichtbarkeit verschiedener Datenbankkenntnisse und -methoden praktisch zu vermitteln.

1.2. Aufbau der Trommelhelden-Datenbank

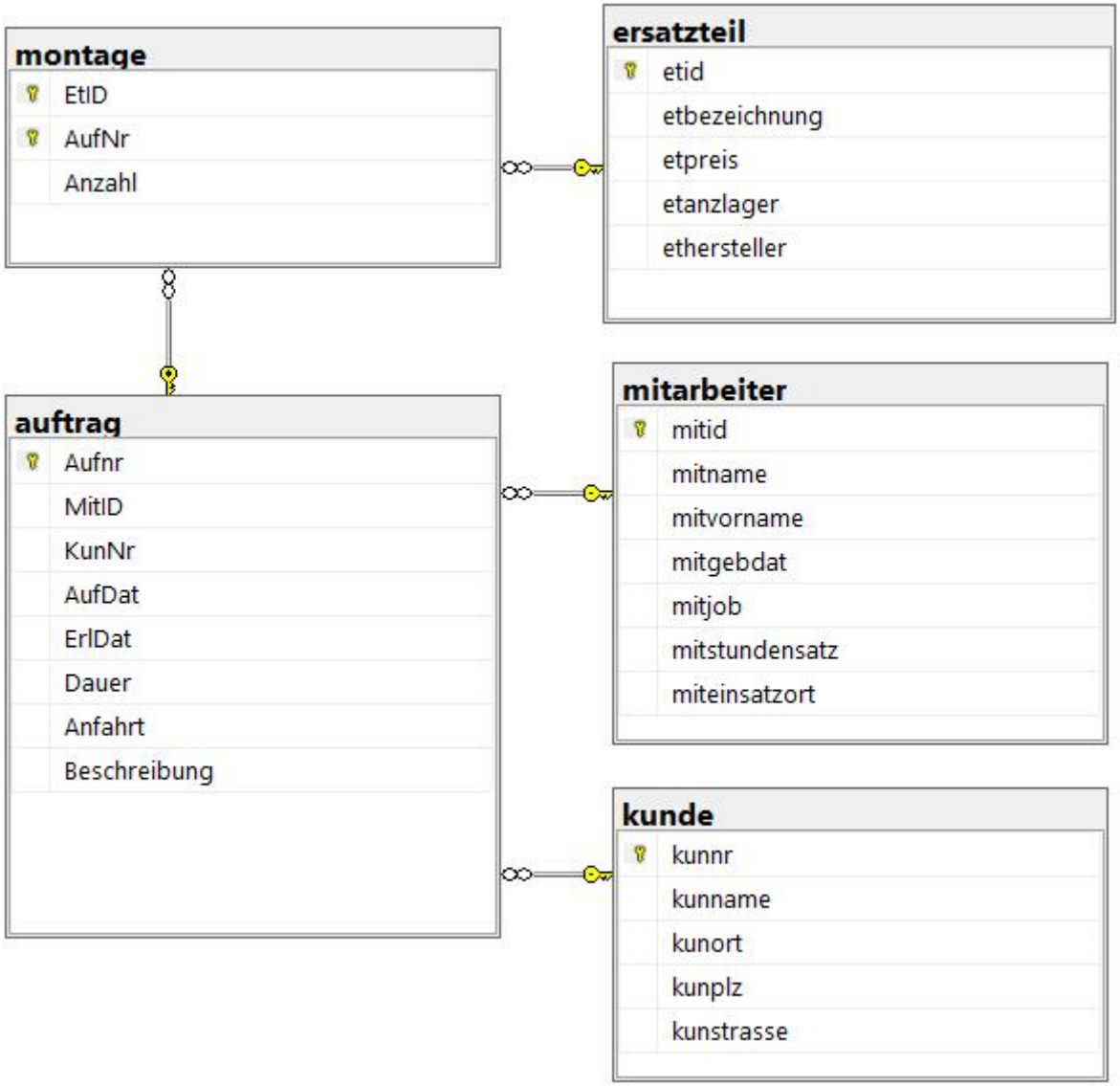
Generell arbeitet die Firma auf folgende Weise: Das Unternehmen nimmt von den Kunden Reparaturaufträge entgegen. Dabei werden die Daten des Kunden, das Auftragsdatum und ggf. eine Beschreibung des konkreten Problems erfasst. Anschließend wird der offene Auftrag einem Mitarbeiter zugeteilt, der nach der Bearbeitung das Erledigungsdatum, Anfahrtsdaten sowie Ersatzteile notiert, die bei der Reparatur zum Einsatz kamen.

Für einen neuen Datensatz in der Auftragstabelle, welcher eindeutig über eine Auftrags-ID identifiziert werden kann, wird also neben der Problembeschreibung und dem Auftragsdatum eine Kunden-ID benötigt. Diese verweist auf einen bereits existierenden Eintrag in der Kunden-Tabelle, welcher den Namen des Kunden sowie die Adresse, bestehend aus Straße, Postleitzahl und Wohnort, beinhaltet.

Wurde der Auftrag einem Mitarbeiter zugewiesen, wird die dazugehörige Mitarbeiter-ID dem Auftragsdatensatz hinzugefügt. Auch diese bezieht sich auf einen Datensatz in einer separaten Tabelle, welche alle eingestellten Mitarbeiter listet. Zusätzliche zum vollständigen Namen des Mitarbeitenden und des Geburtsdatums wird auch der konkrete Jobtitel (Meister, Monteur, Azubi..), der dazugehörige Stundenlohn und der Einsatzort gespeichert.

Wurde der Auftrag beim Kunden abgeschlossen, wird der Datensatz in der Auftragstabelle ergänzt um das Erledigungsdatum, die Dauer des Auftrages in Stunden, die zurückgelegte Strecke für die eigentliche Anfahrt in Kilometer und ggf. eine Beschreibung der erbrachten Serviceleistung. Sollten bei dem Auftrag Ersatzteile verbaut worden sein, entsteht ein Eintrag in der Montage-Tabelle. Diese listet zum jeweiligen Auftrag die Anzahl und die spezifische ID des genutzten Teils.

Zu der passenden ID enthält die Ersatzteil-Tabelle weitere Informationen. Dazu zählt die Bezeichnung des Bauteils, der Preis, die vorhandene Menge im Lager und der Hersteller.



Datenbankdiagramm der Trommelheldendatenbank

2. Anforderungen

2.1. Systemweite Anforderungen

2.1.1. Qualitätsanforderungen

- **Deployment**
  Von Anfang an war geplant, die fertiggestellte Applikation Studierenden zur Verfügung zu stellen, welche sich im Rahmen der Module Datenbanksysteme I und II ebenfalls mit der Trommelheldendatenbank beschäftigen. Hierfür bedarf es ein möglichst einfaches Verfahren, welches die komplexe Installation weiterer Hilfsmodule mit einschließt.
- **Datenbestand**
  Daten sollen zwar neu angelegt, verändert oder gelöscht werden können, doch soll bei der Neustart der Applikation immer wieder der Ausgangsdatenbestand vorhanden sein.
- **Implementierungsalternativen**
  Die zeitliche Auswirkung von Abfragen innerhalb bzw. außerhalb der Datenbankschicht soll untersucht werden. Für eine geeignete Gegenüberstellung braucht es daher mindestens zwei sich unterschiedene Implementierungen ein und derselben Funktion.
- **Laufzeitmessung**
  Für die zu untersuchenden Implementierungsalternativen wird eine Messmethodik benötigt, welche die konkrete Laufzeit der Alternativen festhält und für die weitere Bearbeitung bereitstellt.

2.1.2. Einschränkungen

- **Datenbank**
  Die Verwendung einer bereits bestehenden Trommelheldendatenbank wird vorausgesetzt, daher kommt auch das bereits in den Modulen Datenbanksysteme I und II verwendete relationale SQL Datenbankmanagementsystem von Microsoft zum Einsatz.
- **Webframework**
  Für die Umsetzung des Frontends soll ein Javaskript-basiertes Webframework genutzt werden.

2.2. Use Cases

2.2.1. Use-Case 01: Verwaltung von Kunden

Kurzbeschreibung
Das System ermöglicht es Kunden neu anzulegen, zu suchen, zu bearbeiten oder zu löschen.

Kunde neu anlegen

1. der Use-Case beginnt, wenn der Kundentab ausgewählt wurde
2. dem System wird mitgeteilt, dass ein neuer Kunde angelegt werden soll
3. die Kundendaten werden dem System übergeben
4. der Use-Case ist abgeschlossen, sobald die Eingaben gespeichert wurden

Kunden suchen

1. der Use-Case beginnt, wenn der Kundentab ausgewählt wurde
2. dem System können verschiedene Suchkriterien übergeben werden
3. anschließend wird die Suche gestartet
4. der Use-Case ist abgeschlossen, sobald die Suche beendet wurde

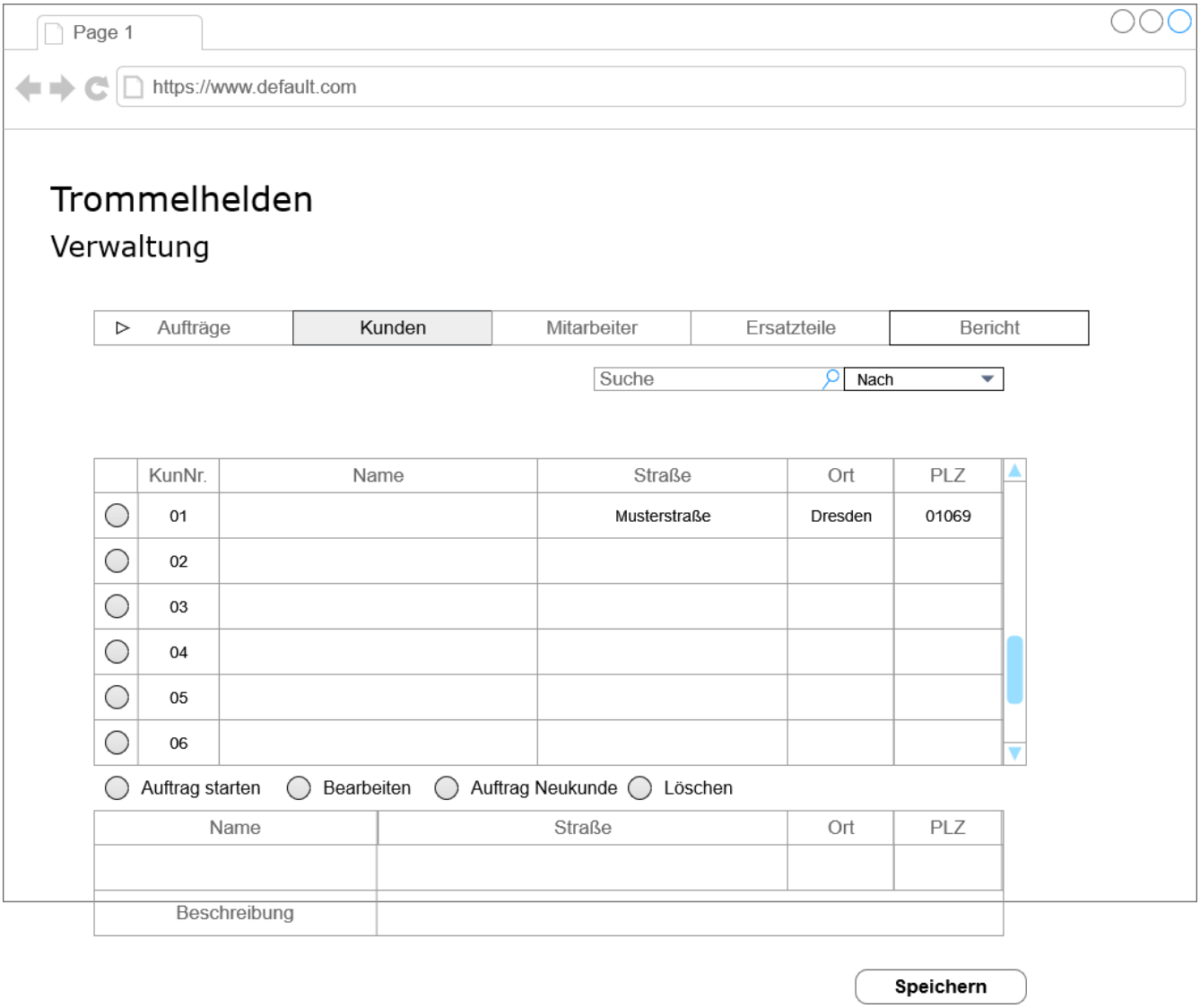
Kunde bearbeiten

1. der Use-Case beginnt, wenn der Kundentab und die Bearbeitungsfunktion eines Kunden ausgewählt wurde
2. die Kundendaten können nun beliebig bearbeitet werden
3. der Use-Case ist abgeschlossen, sobald die Änderungen gespeichert wurden

Kunde löschen

1. der Use-Case beginnt, wenn der Kundentab und die Lösch-Funktion eines Kunden ausgewählt wurde
2. der Use-Case ist abgeschlossen, sobald dem System das Vorhaben bestätigt wurde

Wireframe für den Kundentab



Kundentab-Wireframe

2.2.2. Use-Case 02: Verwaltung von Mitarbeitern

Kurzbeschreibung
Das System ermöglicht es Mitarbeiter neu anzulegen, zu suchen, zu bearbeiten oder zu löschen.

Mitarbeiter neu anlegen

1. der Use-Case beginnt, wenn der Mitarbeitertab ausgewählt wurde
2. dem System wird mitgeteilt, dass ein neuer Mitarbeiter angelegt werden soll
3. die Mitarbeiterdaten werden dem System übergeben
4. der Use-Case ist abgeschlossen, sobald die Eingaben gespeichert wurden

Mitarbeiter suchen

1. der Use-Case beginnt, wenn der Mitarbeitertab ausgewählt wurde
2. dem System können verschiedene Suchkriterien übergeben werden
3. anschließend wird die Suche gestartet
4. der Use-Case ist abgeschlossen, sobald die Suche beendet wurde

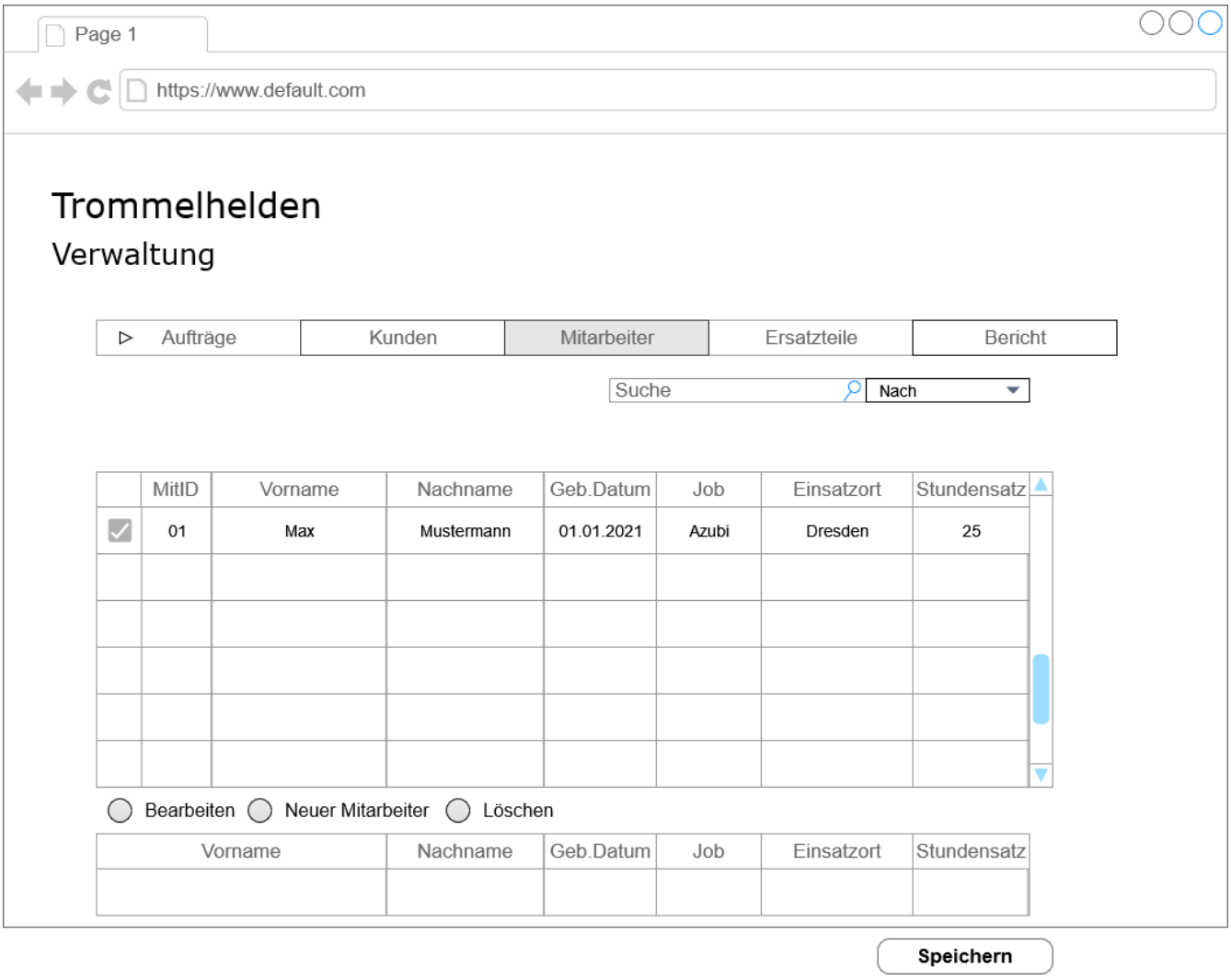
Mitarbeiter bearbeiten

1. der Use-Case beginnt, wenn die Bearbeitungsfunktion eines Mitarbeiters ausgewählt wurde
2. die Mitarbeiterdaten können nun beliebig bearbeitet werden
3. der Use-Case ist abgeschlossen, sobald die Änderungen gespeichert wurden

Mitarbeiter löschen

1. der Use-Case beginnt, wenn die Lösch-Funktion eines Mitarbeiters ausgewählt wurde
2. der Use-Case ist abgeschlossen, sobald dem System das Vorhaben bestätigt wurde

Wireframe für den Mitarbeitertab



Mitarbeitertab-Wireframe

2.2.3. Use-Case 03: Verwaltung von Aufträgen

Kurzbeschreibung
Das System ermöglicht es Aufträge anzulegen, offene Aufträge einzusehen, offene Aufträge einzusehen, offene Aufträge Mitarbeitern zuzuordnen und Aufträge abzuschließen

Auftrag annehmen
Vorbedingung: der Auftraggeber ist Teil der Kundendatei

1. der Use-Case beginnt, wenn der Kundentab und die Auftragsfunktion eines Kunden ausgewählt wurden
2. dem System wird eine Beschreibung des Auftrages übergeben
3. der Use-Case ist abgeschlossen, sobald der neue Auftrag gespeichert wurde

Auftrag zuweisen

1. der Use-Case beginnt, wenn der Auftragstab ausgewählt wurde
2. der Use-Case ist abgeschlossen, sobald dem offenen Auftrag ein Mitarbeiter zugewiesen wurde

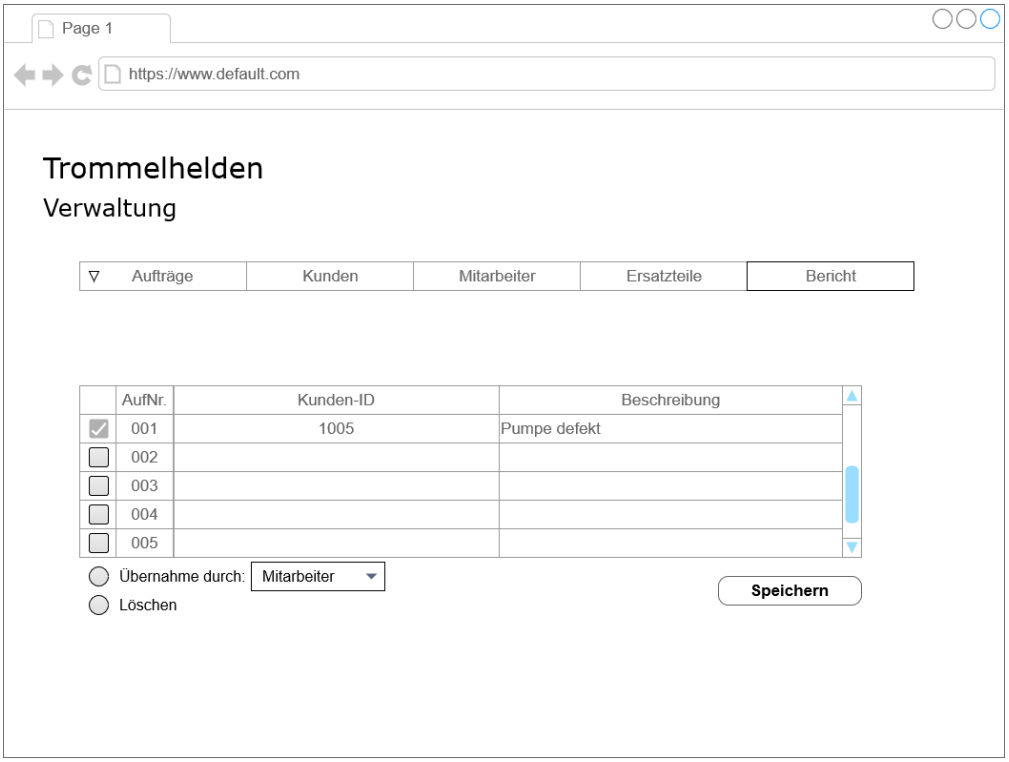
offene Aufträge einsehen

1. der Use-Case beginnt, wenn der Auftragstab ausgewählt wurde
2. über die Angabe eines Mitarbeiters können offene Aufträge eingesehen werden, die speziell von diesem Mitarbeiter bearbeitet werden sollen
3. damit ist der Use-Case angeschlossen

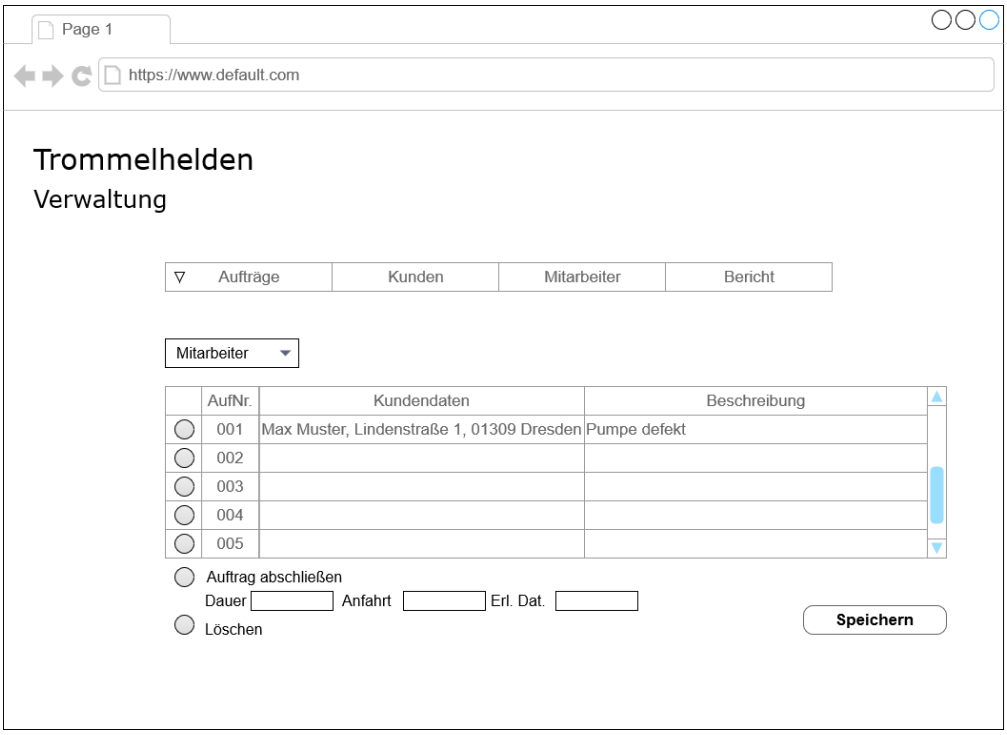
Auftrag abschließen

1. der Use-Case beginnt, wenn der Auftragstab sowie die Abschluss-Funktion ausgewählt wurden
2. dem System werden die restlichen Auftragsdaten übergeben
3. der Use-Case ist abgeschlossen, sobald die Eingaben bestätigt wurden

Wireframes für den Auftragstab



Auftrag annehmen / Neue Aufträge



Offene Aufträge

2.2.4. Weitere Bestandteile

Ersatzteile verwalten
Für die Verwaltung von Ersatzteilen wird ebenfalls ein eigener Tab benötigt. Die benötigten Funktionalitäten (Erstellen, Bearbeiten, Löschen) decken sich mit denen, die bereits in den Use-Cases 01 und 02 verschriftlicht worden sind, weshalb von einer ausführlichen Ausführung abgesehen wurde.

Archiv
Dem Auftragstab untergeordnet existiert ein Tab, welcher abgeschlossene Aufträge enthält. Mitarbeitende können hier Korrekturen an jenen Aufträgen vornehmen.

Bericht
Unter diesem Tab haben die Mitarbeitenden die Möglichkeit, alle Ersatzteile einzusehen, die häufiger verwendet wurden als die von ihnen eingegebene Anzahl. Dies ermöglicht es ihnen, schnell und einfach die am häufigsten gefragten Ersatzteile zu identifizieren und ihre Bestände entsprechend anzupassen.

3. Projektorganisation

3.1. Arbeit im Team

Alias	Teammitglied	Fachsemester/Studiengang
Di	Quang Duy Pham	5. Sem. Allgemeine Informatik
Beni	Thanh Binh Nguyen	7. Sem. Allgemeine Informatik
Sabine	Sabine Adam	7. Sem. Allgemeine Informatik
Vitali	Vitali Tichonow	5. Sem. Allgemeine Informatik

3.1.1. Rollen

Eine klassische Rollenverteilung fand innerhalb des Projektseminars nicht konkret statt. Allerdings haben sich nach den ersten Wochen Mitglieder der Gruppe durch ihre Taten augenscheinlich selbst in die jeweiligen Rollen versetzt, welche nachträglich wie folgt identifiziert werden können:

Für die erste Aufgabe, die grobe Analyse und Erstellung der Wireframes, habt sich das ganze Team zusammengefunden, sodass diese Rolle keiner einzelnen Person zugeschrieben werden kann.

Sabine fand sich nach kurzer Zeit in der Rolle des Managements wieder. Regelmäßig motivierte sie das Team, koordinierte zusätzliche Treffen für ein gemeinsames Arbeiten und übernahm hauptsächlich das Reden während der wöchentlichen Meetings, um den Stand des Projektes darzulegen.

Da Di schon Erfahrungen im Umgang mit dem gewählten Framework hatte, war er der inoffizielle Hauptentwickler der Gruppe. Natürlich war jedes einzelne Mitglied an der Programmierung selbst beteiligt, doch Di war der erste Ansprechpartner, wenn es zu Fragen zu beispielsweise hartnäckigen Fehlermeldungen kam.

Die Rolle des Testers wurde ebenfalls jedem zu Teil. Nachdem eine neue Funktion implementiert und auf GitHub geladen wurde, wurde das Team meist direkt nachrichtlich verständigt und so zum ausgiebigen Testen aufgefordert.

3.2. Werkzeuge

3.2.1. Kommunikation

Gleich zu Beginn des Projektes wurde ein Discord Server aufgesetzt, der wöchentlich für virtuelle Treffen genutzt werden sollte. Durch die Einrichtung verschiedener Kanäle,auf die gezielt nur bestimmte Personen Zugriff hatten, konnte zwischen offiziellen Meetings zusammen mit Herrn Professor Thiele und Team-Meetings unterschieden werden.

Des Weiteren wurde am Anfang der Gruppenarbeit eine WhatsApp-Gruppe eingerichtet, in der zumeist Termine für zusätzliche Meetings ausgemacht wurden. Kleinere Probleme oder Fragen konnten so textuell schnell geklärt werden.

3.2.2. Dokumentation

Tool	Verwendungszweck
git/ Github	Versionsverwaltung des gesamten Projektes
Asciidoc	Verfassung schriftlicher Dokumentationen Codedokumentation
Diagrams.net	Erstellung von Diagrammen und Wireframes

3.2.3. Framework

Obwohl das Team am Anfang des Projektes nach Beratung für das in Python-geschriebene Webframework Flask entschieden hatte, wurde nach kurzer Zeit beschlossen, auf React umzusteigen. Diese Entscheidung wurde getroffen, da ein Gruppenmitglied bereits über grundlegende Kenntnisse in React verfügte. Der Umstieg ermöglichte es dem Team, die Entwicklungszeit zu beschleunigen.

React ist kein Framework, sondern eine JavaScript-Bibliothek für die Erstellung von Benutzeroberflächen (UI). Es ermöglicht, einzelne UI-Komponenten zu erstellen, die unabhängig voneinander verwendet werden können und sich automatisch aktualisieren, wenn sich die Daten ändern.

Außerdem ist React eine der am weitesten verbreiteten JavaScript-Bibliotheken. Das Erlangen der Kenntnisse darin kann die Chancen im späteren Berufsleben erhöhen, wenn man sich für eine Karriere in der Webentwicklung entscheidet.

3.2.4. Deployment

Um nach Fertigstellung der Software diese auch anderen Studierenden zur Verfügung stellen zu können, haben wir uns dazu entschieden, die Applikation mit Hilfe von Docker auszuliefern.
Mittels Docker können Anwendungen schneller bereitgestellt werden. Die Applikation kann hierbei in einem oder mehreren isolierten "Containern" ausgeführt werden, wodurch eine (zeit-) aufwendige Installation benötigter Module vermieden werden kann.

Unser Projekt gliedert sich in drei verschiedene Container:
Einer beinhaltet den MS SQL Server mit samt dem Datenbestand der Trommelheldendatenbank, einer den Client der App und einer den Server.

Systemvoraussetzungen

Mindestanforderungen an Hardware

- Ein internetfähiges Gerät (Smartphone, Tablet, PC) für die Nutzung der Anwendung
- PC/Server mit Docker und Git
- Für Docker: Windows, Linux, Mac
- Microsoft SQL Server weist Kompatibilitätsprobleme unter Mac mit Apple-Chip auf

Softwareanforderungen

- Aktueller Webbrowser (Chrome Ver. 109.0.5414.119 oder höher, Safari Ver. 16.1 oder höher) mit HTML5- und Javascript Unterstützung
- Docker Compose Ver. 3.3 oder höher

Systemeinrichtung

Die Software ist eine webbasierte Anwendung, die sich in einem privaten Repository auf GitHub befindet (Link). Die Anwendung besteht aus mehreren Ordnern, Frontend- und Backend-Komponenten.

Bevor mit der Installation fortgefahren wird, ist es empfehlenswert, einen Fork vom Repository zu erzeugen und im eigenen Account abzuspeichern.

Im Terminal:

```
$ git fork https://github.com/EliSa-GH/ps-trommelhelden.git
```

Um die Anwendung von GitHub auf Ihren eigenen PC/Server zu speichern, muss diese zuerst von GitHub geladen werden.

```
$ git clone https://github.com/EliSa-GH/ps-trommelhelden.git
```

Die Anwendung besteht aus zwei Backend-Komponenten: Webserver(Node.js Ver. 19.1) und MSSQL (SQL Server 2019) und einer Frontend-Komponente (React Ver. 11.10.5), die in einem Docker-Container gespeichert sind.

Um die Software zu installieren, soll ein Docker Container ausgeführt werden. Dafür gibt es zwei Möglichkeiten:

- Im Ordner "sql-server" unter "deployment" die Datei "docker-compose.yml" per Drag and Drop in den Docker Desktop schieben und anschließend auf "Start" drücken. → Diese Vorgehensweise funktioniert nur unter Windows/Mac
- Im Terminal den Ordner "sql-server" unter "deployment" wählen und im Anschluss mit dem Befehl "docker-compose up" die Software ausführen (Docker muss gestartet sein):
  1. Zum Ordner navigieren:

     ```
     $ cd pfad/zum/Ordner/ps-trommelhelden
     ```

  2. In Unterordner wechseln:

     ```
     $ cd deployment/sql-server/
     ```

  3. Server starten:
     - Für neuere Docker Versionen:

       ```
       $ docker compose up
       ```

- Für ältere Docker Versionen:

```
$ docker-compose up
```

Bevor die Anwendung aufgerufen werden kann, muss sichergestellt werden, dass alle Docker-Container gestartet worden sind. Gegebenenfalls muss der "node" Container manuell gestartet werden, damit die Daten in der Applikation selbst zur Verfügung stehen. Erst anschließend kann im Webbrowser in der Adresszeile "localhost:3000" eingegeben werden. Daraufhin wird die Webseite geöffnet.

Fehlerbehebung

Die meisten Fehler, die auftreten können, werden mit Docker zusammenhängen.

Die typischen Fehler und deren Behebung finden Sie unter folgendem Link:

- Docker Troubleshoot

Weitere Dokumentationen

- Docker
- React
- Node.js
- Microsoft SQL

3.3. Entwicklungsprozess und zeitlicher Ablauf

Schon zu Beginn des Projektes war der grobe Umfang der Programmierarbeit bekannt, da Funktionalitäten wie das Neuanlegen, Bearbeiten und Löschen zu den Mindestanforderungen gehörten. Nach einer ersten Analyse und Erstellung von Use-Cases sowie den dazugehörigen Wireframes konnte mit dem eigentlichen Entwicklungsprozess begonnen werden.
Mittels einer iterativen Entwicklung konnte ein regelmäßiger Fortschritt erzielt werden, der in den wöchentlichen Meetings präsentiert und diskutiert werden konnte. Nach der Fertigstellung der Kunden-, Mitarbeiter- und Ersatzteilseite konnte der Auftrags-Tab weiter ausgebaut werden, welcher bei der Erstellung eines Auftrages die Möglichkeit bieten sollte, einen Trigger in die ID-Erzeugung mit einzubeziehen statt diese manuell zu vergeben. Das letzte Inkrement der Applikation bildete die Seite "Bericht", welche nach Eingabe einer natürlichen Zahl, Ersatzteile listet, welche mindestens so oft verwendet wurden, wie angegeben.

3.4. Probleme und Lösungen

Probleme mit JSON: Am Anfang des Projekts entschied das Team, Python Flask zur Verwaltung von API-Anfragen aus der Frontend-Anwendung und SQLAlchemy als Werkzeug zum Abrufen von Daten aus dem SQL Server zu verwenden. Während des Projekts stellten sie jedoch fest, dass die Verwendung von Python-Dictionaries durch SQLAlchemy zur Speicherung von Daten für das Projekt ungeeignet war, da eine Serialisierung der Dictionary-Daten in JSON-Format erforderlich

war. Darüber hinaus gab es Probleme bei der Serialisierung von Decimal- und Datetime-Typen. Daher entschied das Team, zu einer vertrauteren Sprache für die Backend-Entwicklung zu wechseln, JavaScript. Sie untersuchten mehrere Backend-Frameworks und wählten schließlich Express aus, um die Backend-Seite zu handhaben. Nach dem Wechsel stellten sie keine strukturellen Probleme fest.

4. Funktionsdokumentation

4.1. Funktionsbeschreibung

4.1.1. CREATE

Die Funktion "Create" oder "Anlegen" wird im Seiten Auftrag, Kunden, Mitarbeiter und Ersatzteile implementiert.

Auftrag Anlegen

- Die Erstellung einer neuen Auftragsinstanz erfolgt mit der Option, einen Trigger zu nutzen oder nicht. Wenn die Option "mit Trigger" gewählt wurde, wird ein Auftrag mit der nächstmöglichen Nummer erstellt. Dies basiert auf der letzten höchsten Auftragsnummer.
    - *Beispiel: Wenn die letzte Auftragsnummer 1000 ist, wird bei der Auswahl der Option "mit Trigger" automatisch eine neue Auftragsnummer erzeugt. Ohne spezifische Angabe wird die Nummer um eins erhöht und der neue Auftrag mit der Nummer 1001 erstellt.*
- Der Kundenname ist ein erforderliches Feld, um den Auftrag zuordnen zu können. Sollte der Kundenname oder die Nummer nicht in der Datenbank vorhanden sein, wird eine Option zur Erstellung des Kunden angezeigt. Nach dem Klicken wird die Kundenseite geöffnet, um die Erstellung durchzuführen.
- Es gibt optional Felder wie Dauer, MitID, Anfahrt und Beschreibung, die ausgefüllt werden können. Das Auftragsdatum wird automatisch mit dem aktuellen Datum gefüllt.
- **ENTWICKLUNG:** Entwicklungsdatei liegt in

```
~client/src/components/pages/Auftraege/Submenu/Anlegen.jsx
```

    - Die Funktion ist als Singlepage implementiert und deshalb wird es als eine eigene Seite angezeigt.

Kunden Anlegen

- Erzeugt eine neue Kundeninstanz in der Datenbank auf Grundlage des Namens, des Wohnortes, der Adresse und einer selbstgewählten Kundennummer.
- **ENTWICKLUNG:** Entwicklunsgdatei liegt in

```
~client/src/components/pages/Kunden.jsx
```

    - Funktion ist implementiert als DialogBox innerhalb einer Seite und wird als PopUp angezeigt.

Mitarbeiter Anlegen

- Eine neue Mitarbeiterinstanz wird in der Datenbank erstellt.

- Es gibt keine Pflichtfelder. Die angezeigten Felder sind MitID, Name, Vorname, Geburtsdatum, Job, Stundensatz und Einsatzort.
- **ENTWICKLUNG:** Entwicklungsdatei liegt in

```
~client/src/components/pages/Mitarbeiter.jsx
```

  - Funktion ist implementiert als DialogBox innerhalb einer Seite und wird als PopUp angezeigt.

Ersatzteil Anlegen

- Anlegen eines neuen Ersatzteildatensatzes in der Datenbank. Funktion ist ohne Trigger und muss selbst spezifiziert werden.
- **ENTWICKLUNG:** Entwicklungdatei liegt in

```
~client/src/components/pages/Ersatzteil.jsx
```

  - Funktion ist implementiert als DialogBox innerhalb einer Seite und wird als PopUp angezeigt.

4.1.2. DELETE

Auftrag, Kunden, Mitarbeiter, Ersatzteil

- Die Funktion "Delete" oder "Löschen" ist in allen vier Hauptseiten identisch und ermöglicht das Entfernen der ausgewählten Instanz aus der Datenbank.
- Es besteht die Option, mehrere Datensätze gleichzeitig zu löschen (multiples Löschen). Dazu können auf der linken Seite die jeweiligen Checkboxen mit einem Haken versehen werden, um die entsprechenden Zeilen auszuwählen.
- Durch das Klicken auf "Löschen" werden dem User die IDs der ausgewählten Datensätze angezeigt. Erst nach einer erfolgten Bestätigung der Auswahl werden die Daten durch ein erneutes Klicken auf "Löschen" sicher aus dem System entfernt, sofern kein Verweis auf diese existiert.

4.1.3. EDIT

Auftrag, Kunden, Mitarbeiter, Ersatzteil

- Die Edit-Funktion ist eine wichtige Funktion, die es ermöglicht, bereits erstellte Datensätze zu bearbeiten bzw. anzupassen. Dies kann sehr nützlich sein, wenn es darum geht, Fehler in den Daten zu korrigieren oder aktualisierte Informationen hinzuzufügen. Mit der Edit-Funktion können auch bestimmte Teile eines Datensatzes geändert werden, ohne den gesamten Datensatz neu erstellen zu müssen. Dies kann Zeit und Ressourcen sparen und die Datenqualität verbessern.
- Gleichzeitig kann nur ein Eintrag bearbeitet werden, demzufolge bei der Auswahl mehrerer

Einträge wird der Button ausgegraut. Dasselbe passiert mit dem Button "Anlegen", falls ein zu bearbeitender Eintrag ausgewählt wird.

- Außerdem gibt es datenbankspezifische Abhängigkeiten, wodurch nicht jedes Feld bearbeitet werden kann, wie zum Beispiel die laufenden Nummern (MitID, EtID, KunNr, AufNr).
- Darüber hinaus darf bei den Kundendaten keine beliebige Postleitzahl eingegeben werden - sie muss immer aus 5 Ziffern bestehen.
- Diese Funktion ist an vier Stellen (Auftrag, Kunden, Mitarbeiter, Ersatzteil) zu finden und wurde an allen vier Stellen analog implementiert. Durch die konsistente Implementierung wird die Pflege des Quellcodes minimiert.

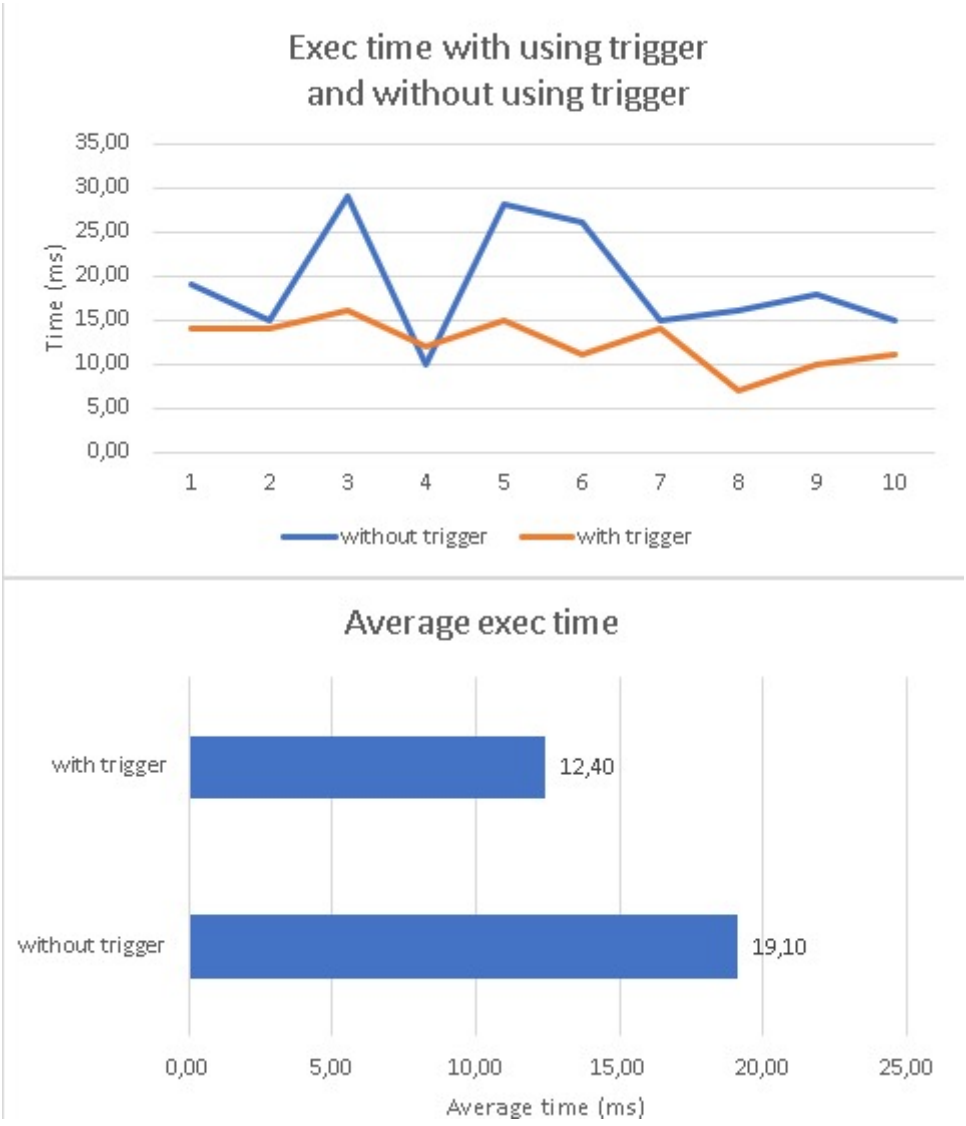
5. Ergebnisse und abschließendes Fazit

5.1. Ergebnisse

Es wurden verschiedene Möglichkeiten implementiert, um neue Datensätze anzulegen. Diese Methoden beziehen sich auf verschiedene Entitäten in der Anwendung, wie z.B. Mitarbeiter, Kunden oder Jobs.

Eine Möglichkeit besteht darin, die Nummer in das Eingabefeld einzugeben. Wenn der Benutzer die ID eingibt, werden die entsprechenden Datensätze in der Anwendung geladen und überprüft, ob die ID-Nummer bereits vergeben ist. Falls die ID bereits vergeben ist, wird dem Benutzer eine Fehlermeldung angezeigt und er muss eine andere ID eingeben. Wenn die ID verfügbar ist, wird der Datensatz angelegt.

Alternativ kann die ID auch automatisch generiert werden, ohne dass der Benutzer sie eingeben muss. In diesem Fall wird ein Trigger in der Datenbank aktiviert, der eine neue ID für den Datensatz generiert. Diese Methode ist hilfreich, wenn der Benutzer keine spezifische ID auswählen möchte oder wenn die IDs automatisch vergeben werden sollen.



Ausführungszeiten mit und ohne Trigger

Um die Leistungsfähigkeit unserer Methoden zu überprüfen und zu vergleichen, welche Methode effizienter ist, wurden die Ausführungszeiten gemessen. Anschließend wurden die Ergebnisse in zwei Diagrammen dargestellt. Dabei wurde festgestellt, dass die Ausführungszeiten mit Trigger bei jeder Ausführung relativ gleichmäßig und niedrig ist, während die Ausführungszeit ohne Trigger sprunghaft und höher ist.

Es lässt sich vermuten, dass der inkonsistente Verlauf bei den Ladezeiten ohne Trigger auf die Menge der Datensätze, die geladen werden müssen oder auf die Prozesse im Hintergrund, die parallel ausgeführt werden, zurückzuführen ist.

Im zweiten Diagramm wird ein Vergleich der durchschnittlichen Ausführungszeiten dargestellt. Die Ausführungszeiten mit Trigger sind im Durchschnitt um 6,7 ms geringer als ohne Trigger.

5.2. Fazit

Aufgrund der gemessenen Zeiten lässt sich sagen, dass eine datenbanknahe Implementation von verschiedenen Funktionen sinnvoller ist, und vorhandene Mechanismen wie beispielsweise Trigger bevorzugt verwendet werden sollten. Eine automatische Vergabe von fortlaufenden IDs ist ebenfalls sinnvoll und trägt zur Konsistenz der Datenbank bei.

Die Entscheidung, ob es ausreicht, Check-Klauseln auf der Datenbank zu haben oder ob die Eingabefelder der Benutzeroberfläche auch auf die Eingaben abgestimmt sein sollten, hängt von verschiedenen Faktoren ab. Es ist jedoch in der Regel eine gute Absicherung, Checks auch im UI einzubauen, um beispielsweise die Integrität von Datenbeständen zu wahren.

Insgesamt aber sollten Checks sowohl auf der Datenbank als auch im UI implementiert werden, um Sicherheitslücken wie zum Beispiel die SQL-Injection vorzubeugen.